programmez!
#199 - Septembre 2016
le magazine des développeurs

# Préparez-vous à Java 9

# Le futur de C# C# 7.0

# Les développeurs révolutionnent la santé

# Programmer sans coder


©marekkuliasz

# Créez votre 1er jeu avec Xamarin

# Raspberry Pi est mort, vive les concurrents

©Nongkran_ch

M 04319 - 199 - F: 6,50 € - RD

Printed in EU - Imprimé en UE - BELGIQUE 7 € - Canada 9,80 $ CAN - SUISSE 13,10 FS - DOM Surf 7,50 € - TOM 1020 XPF - MAROC 55 DH

ikoula
HÉBERGEUR CLOUD

TROPHÉES DUCLOUD by EuroCloud
MEILLEUR SERVICE CLOUD D'INFRASTRUCTURE 2015

# LE CLOUD GAULOIS, UNE RÉALITÉ ! VENEZ TESTER SA PUISSANCE

## EXPRESS HOSTING

Cloud Public
Serveur Virtuel
Serveur Dédié
Nom de domaine
Hébergement Web

sales@ikoula.com
**01 84 01 02 66**
express.ikoula.com

## ENTERPRISE SERVICES

Cloud Privé
Infogérance
PRA/PCA
Haute disponibilité
Datacenter

sales-ies@ikoula.com
**01 78 76 35 58**
ies.ikoula.com

## EX10

Cloud Hybride
Exchange
Lync
Sharepoint
Plateforme Collaborative

sales@ex10.biz
**01 84 01 02 53**
www.ex10.biz

# Une journée sans fin ou jouer à Pokémon Go ?

La vie d'un dév n'est pas un long fleuve de codes tranquille.
Parfois la journée débute par un simple :
5 % de batterie, car tu as oublié de brancher ton smartphone pour la nuit (pourquoi j'ai regardé Mr Robot jusqu'à 2h du mat ?)

Là tu te dis, ben ok, pas grave, j'ai un chargeur au taf (warning : quand tu penses ça, tu as souvent tout faux). Mais tu te dis, argh, je ne vais pas pouvoir chasser de Pokémon dans le métro... euh, relire du code (c'est mieux si le chef de projet te demande).

Il n'est même pas 9h quand le big boss te dit : tu vas aller avec le commercial pour voir un client... (je passe en mode débug)

Plusieurs exceptions possibles :
- ✖ Le client parle technique et tu ne comprends rien (c'est comme si tu fais du CSS sur un mega projet, mais que tu n'en as jamais fait)
- ✖ Le commercial annonce un délai de développements d'1 semaine et tu hallucines
- ✖ C'est quand qu'on parle code ?

Bref, tu as envie de faire un débug sur un code en production juste pour se marrer un peu (bien entendu, nous n'avons jamais fait ça, mais alors jamais !) ou lancer un refactoring, juste pour voir le résultat (ultime étape avant de coder en Cobol, donc à manipuler avec prudence).

Et tu te rends compte qu'il n'est pas encore midi et ton poste freeze pour la 10e fois à cause d'un patch système qui casse tout alors que tu as mis 6 mois à régler ta machine au pixel près ! (où est le sysadmin pour que je lui dise ce que j'en pense ?).

Mais ouf, la longue 1ere journée, après les vacances, se termine et tu dois prendre une des options suivantes :
- ✖ Le sysadmin te dit qu'il code l'infrastructure
- ✖ Fusion de branches à 18h
- ✖ Review de codes du stagiaire
- ✖ Monter un serveur web sur une instance cloud pour ouvrir un fichier HTML (juste pour le fun)

Bon retour dans la matrice !

ftonic@programmez.com

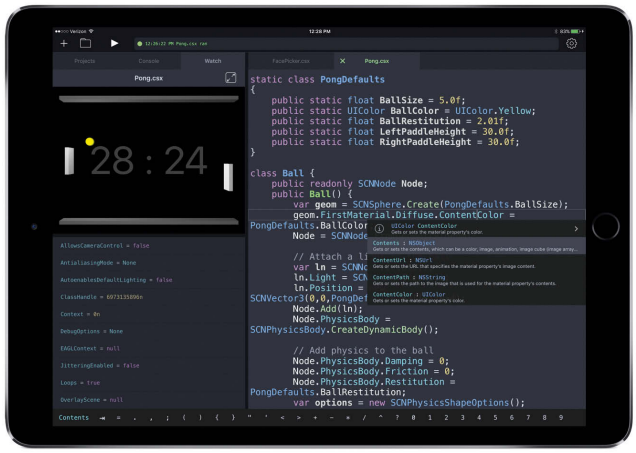


## Evénement / 30 septembre

# Le 200e numéro de programmez!

## UN IPAD COMME POSTE DE DÉVELOPPEMENT ?

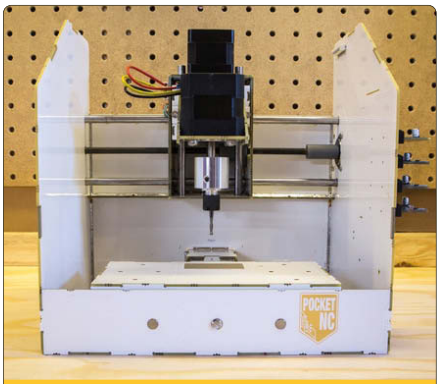
Et si l'iPad devenait un poste de développement comme un autre, comme l'étaient déjà les Surface ? Deux grosses apps sont désormais disponibles :



- Swift Playground (Apple) : présenté en juin dernier. Cette app gratuite tournera sur le futur iOS 10 et permet d'apprendre à coder avec le langage Swift. L'environnement est orienté éducation et apprentissage. L'interface est très propre et permettra de bien comprendre les fondamentaux de la programmation. Avec fonction partage. On ne disposera des mêmes outils qu'avec un IDE normal mais c'est prometteur.
- Continuous (Krueger Systems) est plus ambitieux et propose un véritable environnement intégré sur sa tablette. Il permet de coder en C# et F#. L'éditeur de code ressemble beaucoup à celui de VS Code et l'exécution se fait en live.
Tout est compilé directement sur le terminal. On utilisera les librairies natives iOS et Xamarin (.Net).

## Développeur Ruby / Rails : une opportunité ?

On ne peut pas parler de déferlente mais le profil Ruby et Rails reste toujours d'actualité même si dans la masse des recrutements, Ruby demeure très marginal mais le langage est un peu plus visible qu'il y a 12-18 ans. Depuis 2 ans, Ruby et Rails étaient largement en perte de vitesse et dans les index de langages, Ruby est très loin derrière. Malgré tout, on ne constate pas de réel impact sur les salaires, cependant, même pour un développeur Ruby débutant, vous pouvez espérer une petite plus value, à vous de bien négocier. Par contre, un dév expert peut dépasser les 60, voire, les 65 k€ / an. Ne misez pas uniquement sur Ruby pour votre profil mais n'hésitez pas à considérer cette piste. Mais attention tout de même aux offres full stack pour des salaires trop bas (-35 k€). La plateforme Rails est désormais en version 5 depuis le 30 juin.



Tu cherches un machine à commandes numériques ? Le projet kickstarter FR4 Machine Shield est fait pour toi. Ce projet propose une CNC 3 axes à monter. Elle possédera un système gravage laser ! Le projet est open source et fonctionne avec de l'Arduino ou de la Pi. A partir de 350 $ (sans le port). Page : https://www.kickstarter.com/projects/1090944145/fr4-machine-shield

Des chercheurs de l'université d'Harward créent un robot aquatique mais incluant des cellules cardiaques de rat. Ces cellules ont été modifiées pour pouvoir réagir à la lumière. Impressionnant : http://diseasebiophysics.seas.harvard.edu

## L'INDEX TIOBE DU MOIS

| 08/16 | 08/15 | Evolution | Langage | % | Evolution en % |
|---|---|---|---|---|---|
| 1 | 1 | | Java | 19.010% | -0.26% |
| 2 | 2 | | C | 11.303% | -3.43% |
| 3 | 3 | | C++ | 5.800% | -1.94% |
| 4 | 4 | | C# | 4.907% | +0.07% |
| 5 | 5 | | Python | 4.404% | +0.34% |
| 6 | 7 | ^ | PHP | 3.173% | +0.44% |
| 7 | 9 | ^ | JavaScript | 2.705% | +0.54% |
| 8 | 8 | | VB .NET | 2.518% | -0.19% |
| 9 | 10 | ^ | Perl | 2.511% | +0.39% |
| 10 | 12 | ^ | Assembleur | 2.364% | +0.60% |

C# repasse devant Python qui garde une bonne dynamique. Swift est désormais 14e au classement et Ruby arrive 12e.

Après **Windows 10 Redstone**, cet été, **Redstone 2** sortira courant 2017, on annonce déjà le printemps, sans plus de précision.

**Apple** installe un laboratoire de recherche au coeur de Grenoble

Nintendo a surfé sur le succès cet été : après l'incroyable succès de Pokémon Go, voici le retour de la **console NES**. Cette mini-version est attendue mi-novembre avec 30 jeux. Environ 60 €.

**Voiture autonome :** garder ou non les mains sur le volant ? Le débat est relancé aux USA.

Le projet **ITER** prend du retard et va coûter très cher : + 18 milliards et une utilisation optimale pas avant 2035

**Samsung** sort un SSD de 4 To au modeste prix de 1529,90 €


**Planet of the Apps**, une télé-réalité montrant des développeurs. Emission commandée par Apple.

# WEBDEV®

## NOUVELLE VERSION 21

# CRÉEZ FACILEMENT DES SITES «RESPONSIVE WEB DESIGN»

## ACCÉDANT À VOS BASES DE DONNÉES


### RENDEZ VOS SITES «MOBILE FRIENDLY»

**WEBDEV 21** vous permet de rendre facilement vos sites dynamiques «Mobile Friendly». Créez un seul site pour toutes les cibles.
Les sites que vous créez sont ainsi mieux référencés par Google.
**Responsive Web Design** et Dynamic Serving sont à votre service dans WEBDEV 21.

*Vous disposez d'applications WINDEV ? Elles sont compatibles WEBDEV 21 !*

**DÉVELOPPEZ 10 FOIS PLUS VITE**

**www.pcsoft.fr**

*+ de 100 témoignages sur le site*

## septembre

### Hardwear.io 2016 :

**22 & 23 septembre/La Hague/Hollande**

Pour la 2e année, la conférence sécurité matérielle revient. Elle se tiendra en Hollande fin septembre. Avec les multiplications des matériels et logiciels embarqués (voitures, IoT, avions, médecine, dans les maisons, etc.), on oublie parfois un peu vite la sécurité. Trois catégories de sessions seront disponibles : recherches actuelles, nouvelles recherches et les outils. L'objectif est de parler de sécurité offensive et défensive. Les domaines abordés seront très nombreux : processeurs, embarqué, IoT médical, firmware, test de pénétration hardware, etc.
Site officiel : http://hardwear.io


### FrenchKit :

**23 & 24 septembre/Paris**

Une grande conférence développeur dédiée à iOS et macOS se tiendra à Paris. Xebia et CocoaHeads Paris organisent cet événement. Une quinzaine de sessions sont prévues la première journée et une seconde journée orientée communauté avec du live-coding, des ateliers, du hacking et des démos. + 200 personnes sont attendues.
Site officiel : http://frenchkit.fr

### Agile Pays Basque :

**23 & 24 Septembre**

La 1ère édition de l'Agile Pays-Basque aura lieu les 23 et 24 Septembre 2016 à Bidart (à côté de Biarritz) : 2 jours de conférences, ateliers, retours d'expérience sur les sujets liés à l'agilité : Scrum, Kanban, DevOps. Nous aborderons aussi des thèmes tels que le Lean Startup, l'Entreprise libérée, la Facilitation graphique et tous les sujets qui inspirent l'écosystème agile. Nous donnons rendez-vous à tous les professionnels, les enseignants et les étudiants qui souhaiteraient savoir comment s'y prendre, découvrir, échanger, apprendre, expérimenter, tester, approfondir leurs connaissances, ou simplement discuter autour de l'agilité.
Plus d'info sur : http://agile-paysbasque.fr/

## octobre

### Microsoft experiences :

**4 & 5 octobre/Paris**

Les TechDays sont morts, vive Experiences. Microsoft a décidé de réinventer sa grande conférence annuelle, les MS TechDays qui étaient l'événement technique de l'année. Désormais, l'événement aura lieu en octobre et sur 2 jours uniquement, au lieu de 3 ! Cela signifie moins de sessions. Le salon se découpera en deux parties : journée business et journée technique. La journée technique s'articulera autour de 95 sessions... le choix va être difficile !
Et cette année, Scott Guthrie sera l'invité vedette.
Site officiel : https://experiences.microsoft.fr

### D Day 2e édition :

**7 octobre/Marseille**

Journée d'échanges consacrée aux acteurs de l'écosystème DevOps, Cloud et Docker, la 2ème édition de DevOps D DAY, organisée par Treeptik, réunira des speakers internationaux de renom et proposera en nouveauté des sessions de training Docker animées par des experts reconnus. Cette année, Amazon Web Services et Microsoft se positionnent en tant que partenaires. Sont attendus les représentants de tous les métiers de l'informatique : développeurs, testeurs, architectes, administrateurs systèmes, chef de projets, DSI... au total Treeptik attend plus de 500 participants.
Amazon Web Services, Microsoft, Treeptik, Docker, GitHub, Orange... seront présents.
Programme et inscriptions : www.devops-dday.com

### dotGo :

**10 octobre/Paris**

Le langage Go sera largement à l'honneur à Paris durant la conférence dotGo. Les conférences dot rassemblent les meilleurs speakers du domaine. Cette année, Robot Griesemer, co-créateur du langage, sera là. Conférence uniquement en Anglais.
Site officiel : http://www.dotgo.eu

### Forum PHP :

**27 & 28 octobre/Montrouge**

C'est LE rendez-vous des communautés PHP. On va beaucoup parler PHP 7 et lever le voile sur le futur du langage.
Site : http://afup.org

## novembre

### DevFest Toulouse 2016 :

**3 novembre**

Basé sur une contraction de "Developer's Festival", le DevFest rassemble les plus grands événements organisés par les communautés autour des technologies Google : les Google Developer Groups. Pendant une journée vous pourrez assister à des talks autour des technologies affiliées à Google (Web Apps, Mobile, Tools & Methods). Les communautés toulousaines vous attendent pour cette première édition dans la ville rose ! Plus d'informations et inscription : https://devfesttoulouse.fr/

### Paris Open World Summit 2016 :

**16 & 17 novembre/Plaine Saint-Denis**

La nouvelle édition de la grande conférence open source française se tiendra en novembre prochain. Le thème central sera l'innovation qui s'appuie de plus en plus sur l'open source, l'open data, l'open hardware, etc. A cette thématique principale, trois volets seront abordés : la technologie, l'entreprise et la société. Chacun de ces secteurs a des défis propres à relever.
Pour en savoir plus : http://www.opensourcesummit.paris

# Abonnez-vous à programmez!

le magazine des développeurs

**1 an** ..... 49€ + 1€ = **50€***
11 numéros + un livre au choix

**2 ans** ..... 79€ + 1€ = **80€***
22 numéros + un livre au choix

**Etudiant** ..... 39€ + 1€ = **40€***
1 an - 11 numéros + un livre au choix

## Pour 1€ de + un livre numérique des Editions ENI au choix

valeur : 29,26 €

ou

valeur : 40,50 €

**PDF** ..... **30€***
1 an - 11 numéros + une vidéo au choix

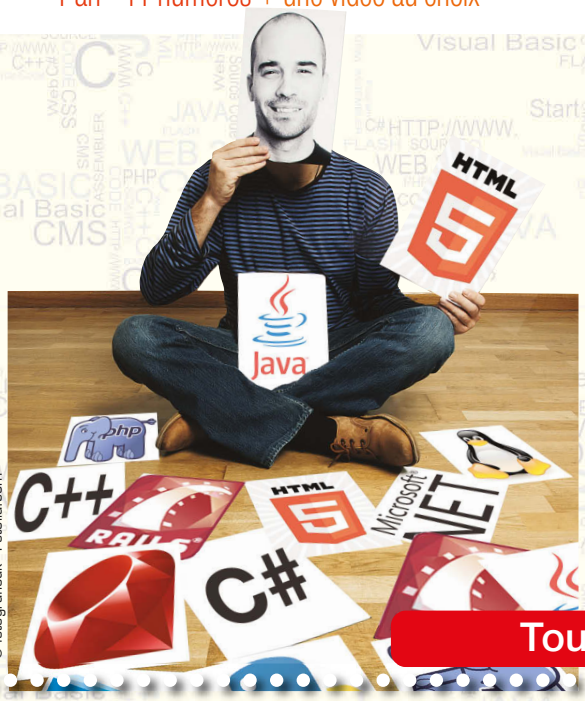
**JavaScript**
Développez un client Web en Full JavaScript

valeur : 29,99 €

ou

**Applications mobiles multiplateformes**
Technologie et contexte d'utilisation

valeur : 29,99 €



© fotografiedk - Fotolia.com

**Vous souhaitez abonner vos équipes ? Demandez nos tarifs dédiés* : redaction@programmez.com**
**(* à partir de 5 personnes)**

*(*) durée de l'offre du 31 mai au 30 septembre*
*Tarifs France métropolitaine*

**Toutes nos offres sur www.programmez.com**

# Oui, je m'abonne

*ABONNEMENT à retourner avec votre règlement à :*
*Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex.*

☐ ***Abonnement 1 an au magazine :*** 49 €
☐ ***Abonnement 2 ans au magazine :*** 79 €
☐ ***Abonnement étudiant 1 an au magazine :*** 39 €
Photocopie de la carte d'étudiant à joindre

☐ ***Abonnement 1 an + Livre numérique*** ☐ ***AngularJS ou*** ☐ ***Git :*** 50 €
☐ ***Abonnement 2 ans + Livre numérique*** ☐ ***AngularJS ou*** ☐ ***Git :*** 80 €
☐ ***Abonnement étudiant 1 an + Livre numérique*** ☐ ***AngularJS ou*** ☐ ***Git :*** 40 €
Photocopie de la carte d'étudiant à joindre

PROG 199
*Offre limitée, valable jusqu'au 30 septembre 2016*

☐ M. ☐ Mme Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

***email indispensable pour l'envoi d'informations relatives à votre abonnement***

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !
☐ Je souhaite régler à réception de facture

** Tarifs France métropolitaine*

# Coder *sans programmer, programmer sans* coder !


source : marekuliasz

*C'est véritablement au début des années 90 que j'ai compris la signification du "programmer sans coder", ou presque, avec des environnements tels que HyperCard (logiciel mythique pour moi), les premières versions de Visual Basic (et surtout à partir de VB 3, 1993), sans oublier des logiciels comme 4D (version 5.x puis les bêtas de la v6) et MS Access... Des débats souvent enragés existaient : les uns accusant les autres de faire de la programmation Légo. Bref de ne pas réellement programmer !*

*L'idée derrière cette expression est de simplifier le codage même si ces outils ne suppriment pas totalement le code. Ils mâchent le travail et génèrent une partie du code de base. On assemble des briques, des composants (notamment pour créer des interfaces rapidement ou intégrer des liens vers les bases de données) et on ajoute le logique (= la glue) pour tout faire fonctionner.*

*Au début des années 2000, nous parlions beaucoup des approches MDD, MDA, les projets et les applications pilotés par des modèles. L'ambition, à cette époque, était énorme : générer la quasi-totalité du code et des interfaces grâce à ces modèles. Des dizaines d'outils existaient sur le marché.*

*Dans Programmez !, dès les années 2002-2003, nous en parlions régulièrement. Et nous étions assez enthousiastes, mais de nombreux problèmes existaient notamment sur la cohérence entre le modèle et le code quand on changait un des deux, il fallait regénérer et modifier côté l'autre à la volée. La qualité du code n'était pas toujours bonne. La partie interface obtenait de meilleurs résultats.*

*De nombreuses applications n'ont pas besoin d'un développement exigeant avec une équipe, des outils dédiés, etc. On veut juste un outil pour gérer le stock d'un entrepôt ou pour aider les commerciaux, etc. Pour le développeur, ces outils, payants ou gratuits, sont intéressants pour développer rapidement des projets ne nécessitant pas la grosse artillerie technique et pour optimiser son travail et son temps. Et finalement, réinventer la roue à chaque développement prend du temps et ne sert à rien.*

François Tonic

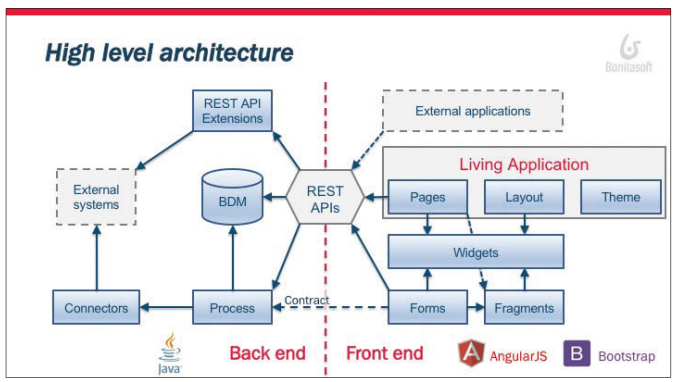
## Des outils et un marché important

*Si pour le développeur, la notion de programmer sans coder peut paraître saugrenue, voire irréaliste, il existe de nombreux outils pour le faire, ou tout du moins, simplifier au maximum le code à écrire.*

Début 2015, les analystes du cabinet Gartner prédisaient que 50 % des apps mobiles seraient créées à l'horizon 2018 par des personnes non techniques (pour les gens du métier, du business, etc.), le département informatique se concentrant sur les applications les plus stratégiques et critiques. Cette approche permet une grande flexibilité dans les applications et de répondre rapidement à des besoins précis, mais ne nécessitant pas l'usage d'une équipe de développeurs, ni d'un développement très long. Le développeur doit aussi s'y intéresser pour répondre rapidement à des demandes d'une PME, d'artisans, etc. Il existe plusieurs méthodes pour réaliser des applications sans programmer. Par exemple, il est possible d'utiliser la notion de paramètres et de description fonctionnelle comme dans l'environnement NOUT. Une autre approche consiste à s'appuyer sur le BPM (Business Process Management).

### Le BPM comme solution, et les autres

Le BPM permet de modéliser tous les processus métiers et les flux. Il permet de générer et d'automatiser les processus de l'entreprise, d'un département, d'un service. Il est avant tout utilisé par les équipes métiers, parfois les utilisateurs. En allant plus loin, le BPM peut servir à décrire une application tout en ayant un langage commun, entre les développeurs et




les services non techniques. Le développeur peut aussi utiliser directement le BPM pour décrire l'application. L'éditeur Bonitasoft utilise le BPM pour son environnement Bonita BPM. Pour générer l'application, l'outil s'appuie sur le modèle MVC permettant de séparer toutes les couches de l'application. Et si le modèle de génération change, la mise à jour se fera en douceur, par contre, pour une modification importante, il faut régénérer l'ensemble du projet. Une des ambitions est de minimiser les allers-retours entre les différentes équipes.

Outre Bonitasoft, les outils BPM ne manquent pas : jBPM, Activiti, Camunda, Appian, Tibco, Oracle, Pega... Les grandes entreprises et les institutions sont les premiers clients de ces solutions et la notion de BPM, de modélisation leur parlent. Mais le BPM n'est pas l'unique solution, loin de là. SoftwareZator (http://softwarezator.velersoftware.com) est un environnement intégré pour créer des formulaires, des interfaces, des interactions. La création est avant tout visuelle et ensuite on paramètre les objets. Et à chaque objet, par exemple, un bouton, un champ, on peut rattacher des actions. Bien que l'outil ne soit plus mis à jour, cela démontre la puissance de ce genre d'environnement pour créer rapidement des logiciels sur mesure pour des besoins précis. Dans le même genre, il existe le projet open source, PWCT (http://doublesvsoop.sourceforge.net). Il simplifie la création d'applications, mais demande une plus grande maîtrise. De plus, l'interface n'est pas aussi fluide que d'autres solutions de programmation sans coder. Le projet existe depuis 10 ans et de nouvelles versions sortent régulièrement.

Plusieurs approches techniques sont disponibles sur le marché :

- Environnement orienté programmation visuelle : on compose l'interface et les fonctions par glisser-déposer en supprimant le code à écrire. On va très vite et on paramètre, on configure les briques fonctionnelles. Idéal pour des apps événementielles, des apps vitrines, des catalogues.
- Environnement orienté modèle : l'usage d'une approche à la UML ou BPM permet de construire des apps d'entreprises et pour des métiers complexes.
- Environnement orienté langage / DSL : ces outils reposent sur un langage, mais plus simple que les langages de programmation.

## PROGRAMMATION INTERDITE

*Dans NOUT Builder, l'outil de développement d'applications Windows et Web, il est non seulement possible de tout réaliser sans programmer, mais la programmation est interdite.*

L'équipe de NOUT a poussé le concept à l'extrême. En décidant que NOUT Builder ne permettrait pas de programmer, ils se sont contraints à propulser le concept de paramétrage à un niveau jamais vu. En effet, beaucoup de solutions de développement permettent de réaliser des applications sans programmer, mais au bout d'un moment il faut toujours coder... Pas dans NOUT Builder.

### Mais si je veux vraiment programmer ?

NOUT Builder permet de décrire des formules évoluées qui donnent les règles de mise à jour des champs. Donc il est facile de mettre à jour un ou plusieurs champs avec la valeur d'un autre même s'il n'y a pas de code sur les évènements de sortie de champ. Les automatismes permettent de décrire des évènements déclenchants (ex : création de commande) et les traitements à réaliser (ex : réservation de stock), y compris des traitements répétitifs comme pour des algorithmes itératifs ou récursifs. Le canevas de NOUT Builder pose des régles de construction de l'application et il lui donne ainsi une logique ergonomique qui assure une unité très pratique pour l'apprentissage des futurs utilisateurs. Surtout par paramétrage la construction et la maintenance évolutive de l'application sont beaucoup plus rapides que par programmation. Le système est bien pensé, pas besoin de réinventer la roue. Messagerie et planning intégré, interface LDAP, système d'import/export, système d'annuler refaire, multi-langue, diagramme de Gantt, graphiques... Autant de fonctions déjà présentes que vous pouvez utiliser si besoin. Les requêtes à la base de données de NOUT Builder sont déjà optimisées. De plus le système de maintenance de NOUT Builder, qui analyse les usages des utilisateurs, crée des index pour accélérer encore les traitements. NOUT Builder a fait ses preuves sur plus de 200 applications métier complexes très différentes dont beaucoup d'ERP. La version 11 utilise le framework Symfony. Il est donc possible de créer des applications Web responsives sur plateforme Symfony sans programmer. Un outil à suivre de près donc...

### Pour une entreprise, il faut réfléchir à cette option

Une entreprise n'a pas toujours à sa disposition d'équipes techniques ni de développeurs sous la main et même dans une grande société, parfois, les équipes veulent disposer d'un outil immédiatement pour remplir telle action au quotidien. Et un "développement" est alors réalisé dans un coin, c'est que l'on appelle le "shadow it", une informatique qui échappe au service informatique.

### Quelques solutions du marché

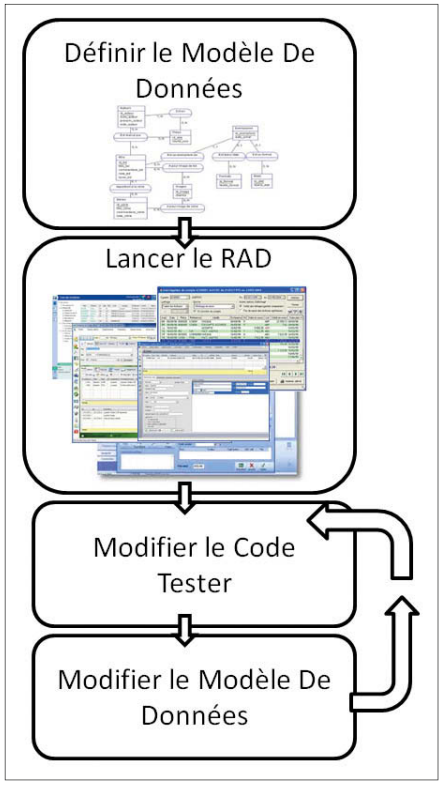
| Outils | Tarifs | Cibles | Commentaires |
|---|---|---|---|
| Canvas https://www.gocanvas.com | à partir de 30 $ / mois. | iOS, Android, Windows | Création rapide de formulaires. Intégration du code à barres, signature, GPS, etc. |
| Zoho Creator https://www.zoho.com/creator/ | à partir de 5 $ / mois. | iOS, Android, Windows | Outil Cloud pour des apps d'entreprises (CRM, donnée...). Nombreuses extensions disponibles |
| Intuit QuickBase http://quickbase.intuit.com | à partir de 15 $ / mois. | mode Cloud | Pour apps métiers et bases de données avec reporting |
| FileMaker Pro http://www.filemaker.com | 349 € H.T. (version personnelle) | macOS, Windows, iOS | Environnement orienté SGBD |
| Salesforce App Cloud Lighning http://www.salesforce.com/platform/lightning/ | A partir de 25 $ / mois | Service SaaS | Plateforme Cloud complète orientée entreprise et donnée |
| App Press https://www.app-press.com | à partir de 30 $ / mois | Android, iOS | Créer rapidement une app mobile visuelle et selon vos besoins. API disponibles. |
| AppArchitect http://www.apparchitect.com | Gratuit (composer) | Android, iOS | Construire une app mobile avec des blocs |
| Alpha Anywhere http://www.alphasoftware.com | 1 499 $ / an | Apps hostées | Environnement avec de nombreux connecteurs, mode offline disponible, portabilité mobile |
| ViziApps http://www.viziapps.com | à partir de 39 $ / mois | Android, iOS | Outil orienté développeur et apps mobiles. Les fonctions dépendent de la version |
| Form.com https://form.com | N.C. | Web, Android, iOS | Orienté formulaire pour Web et mobile |
| Sprighlty https://www.microsoft.com/en-us/garage/#app-sprightly | Gratuit | Android, iOS | Créer très rapidement des apps de catalogue, des e-cartes, etc. |
| iBuildApp http://ibuildapp.com | A partir de 39 € / mois | Android, iOS | Création d'apps mobiles. Tout usage. |
| Microsoft App Studio https://appstudio.windows.com/fr-fr | Gratuit | Windows Windows Phone | Environnement très complet en ligne pour créer en quelques clics une app Windows. Compatible Visual Studio |
| Andromo http://www.andromo.com | Gratuit & versions payantes | Android | Créer des apps Android |
| Appy Pie http://www.appypie.com | Gratuit & versions payantes | Web Android iOS Windows | Solution orientée jeu, édition. Nombreuses fonctions disponibles |
| WinDev / WebDev https://www.pcsoft.fr | 1 650 € H.T. | Android iOS Windows Web | Environnement complet pour construire et déployer très rapidement une application, une app mobile. |
| Bubble https://bubble.is | A partir de 19 $ / mois | Web | Environnement visuel de développement. La création se fait par glisser-déposer et paramétrage |

*Tableau non exhaustif.*

# Une application Web responsive sur Symfony sans programmer

*Avec l'outil de développement sans programmation NOUT Builder 11, l'interface web utilise le framework Symfony pour une ergonomie épurée et un grand confort d'utilisation.*

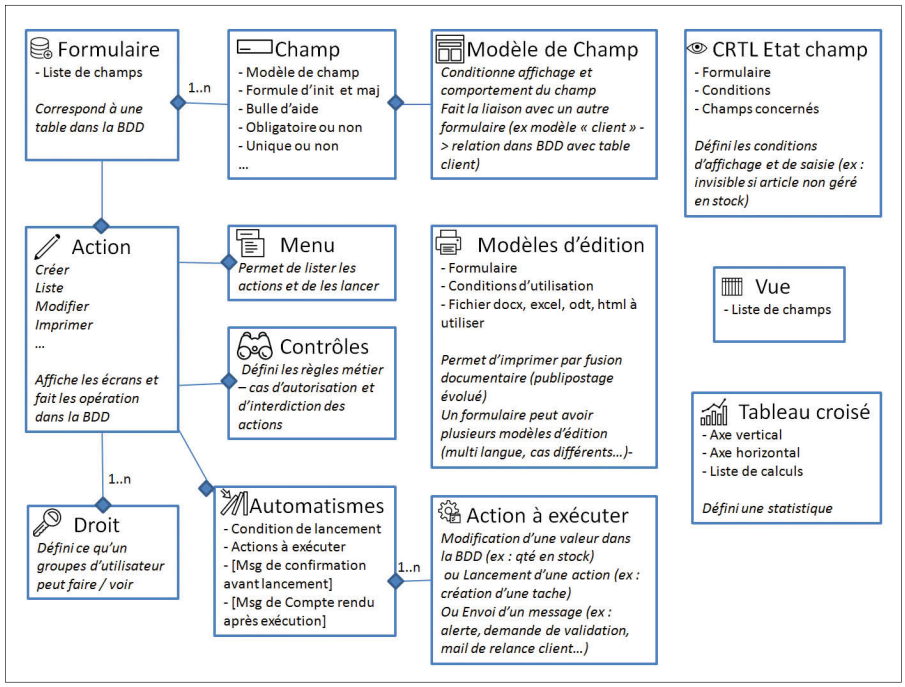
Octobre 2015, se rassemblent comme tous les ans depuis 17 ans les meilleurs développeurs du monde pour la RAD Race. La compétition se déroule pour la seconde année, à Utrecht, aux Pays Bas. Les équipes sont dans les starting bloks. Les cafés, les barres chocolatées et les routeurs sont posés sur les tables et tous sont prêts à s'affronter sans merci durant un jour et demi au cours duquel le jury prend un malin plaisir à changer le cahier des charges.
Parmi les favoris, une entreprise française : NOUT qui a déjà remporté le concours l'année précédente en surprenant tout le monde. En effet, l'équipe de NOUT n'a pas saisi une seule ligne de code pour remporter la compétition 2014 et ils ne le feront pas en 2015 non plus. Jérôme Olivares et Miren Lafourcade sont là pour démontrer qu'il est non seulement possible de décrire entièrement un logiciel avec du paramétrage simple, mais aussi que c'est bien plus rapide que de programmer. Le verdict tombe sans appel, NOUT remporte une nouvelle fois la compétition 2015.



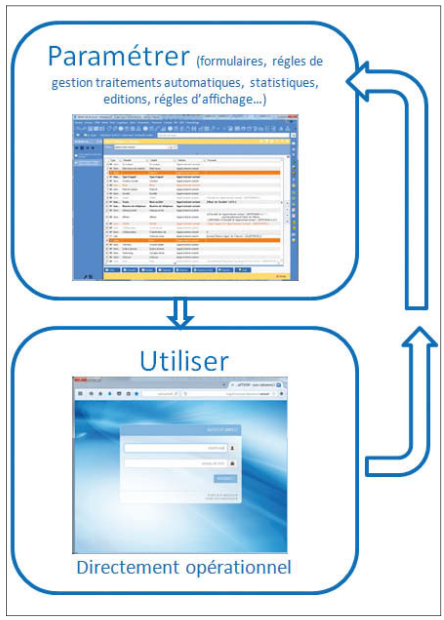

*Avec programmation*

## Fini le RAD, vive les fonctions dynamiques

Il a fallu 12 ans et 5 millions de lignes de code à NOUT pour produire NOUT Builder son système expert auquel on donne par paramétrage la description du logiciel et qui agit comme on le lui a demandé. Avec NOUT Builder, il est possible de créer des logiciels en mode stand alone, Client/Serveur ou des applications Web dynamiques sans programmer. Avec ce système, NOUT a réalisé plus de 200 applications dont l'ERP SIMAX. SIMAX est le premier ERP entièrement paramétrable. Dans SIMAX aucune fonction ou écran, aucune requête, aucun traitement de gestion n'est programmé, pas même des fonctions évoluées comme l'ordonnancement, la gestion de stocks ou les calculs de besoin net. Tout est décrit et modifiable facilement par des utilisateurs non développeurs.
Au lieu de se tourner vers une technique RAD, c'est-à-dire de génération automatique de code à partir d'une description, le système expert consulte la description fonctionnelle (champs présents, règles de gestion, traitement automatiques, règles d'affichage...) pour s'exécuter comme demandé. Avec NOUT Builder, c'est la fonction qui crée l'organe. Ainsi, lorsque vous ajoutez un champ dans un formulaire (le champ date de naissance dans la fiche client par exemple), NOUT Builder modifie dynamiquement la structure de la base de données pour que cela marche. Enfin la possibilité de créer une application facilement pour les non experts ! Et pour les spécialistes, quel gain de temps ! Pour plus d'information www.nout.fr ✖




*Fonctionnement du paramétrage (Simax)*




*Sans programmation*

Retrouvez toutes nos vidéos sur http://www.nout.fr/videos/

# Utopie ou réalité ?

*Est-il possible de programmer sans coder ? Qu'on soit partisan du oui ou fidèle au non, pour donner cohérence à sa position on est contraint d'interpréter les mots « programmer » et « coder » d'une manière particulière. Notre réponse reflètera nos croyances et nos envies, nos peurs et nos intérêts. La réponse concernera peut-être des domaines particuliers. Elle dépendra des moyens et des outils qu'on s'autorise ou qu'on s'interdit d'utiliser. Elle découlera des styles ou approches qu'on a choisis pour exercer notre métier. Je propose une exploration des idées, des interprétations, des outils, des styles, et des domaines d'applications pour que vous puissiez vous aussi adopter une posture par rapport à la question.*

Frédéric Fadel
*+ 35 ans de programmation, co-fondateur &* **CTO d'Aspectize**

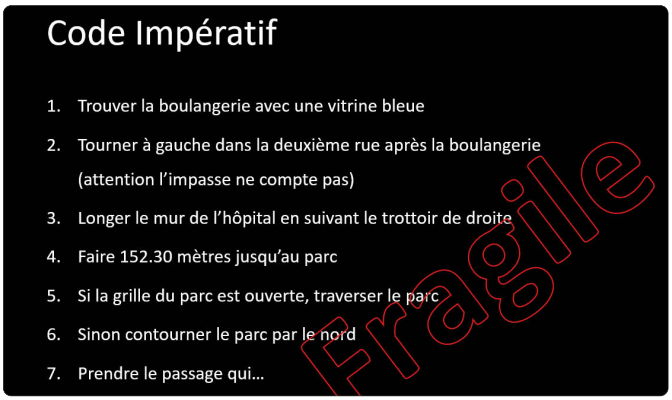
## Les définitions usuelles

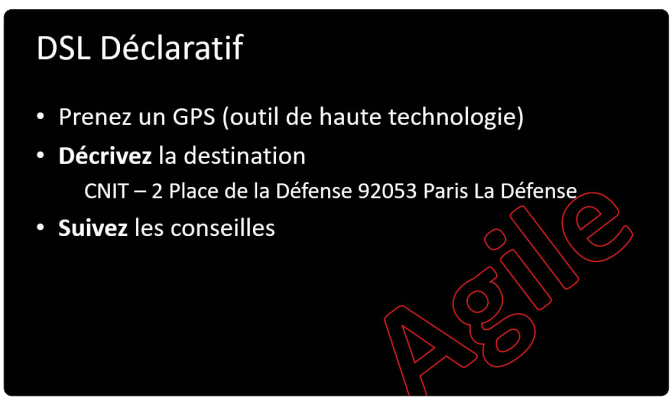
- Programmer : Planifier, organiser une suite d'opérations.
- Programmable : Que l'on peut programmer ; dont on peut régler à l'avance la mise en œuvre. *Calculateur, magnétoscope, ordinateur, prise de courant, téléviseur programmable.*
- Le mot *Ordinateur* (1955) inventé par IBM pour le marché français a été fortement inspiré par une description de la machine analytique de Babbage (1837) : « Pour aller prendre et reporter les nombres... et pour les soumettre à l'opération demandée, il faut qu'il y ait dans la machine un organe spécial et variable : c'est l'ordonnateur. Cet ordonnateur est constitué simplement par des feuilles de carton ajourées, analogues à celle des métiers Jacquard... ». Le mot anglais « computer » signifiant, lui, calculateur - alors que la plupart des programmes calculent fort peu.

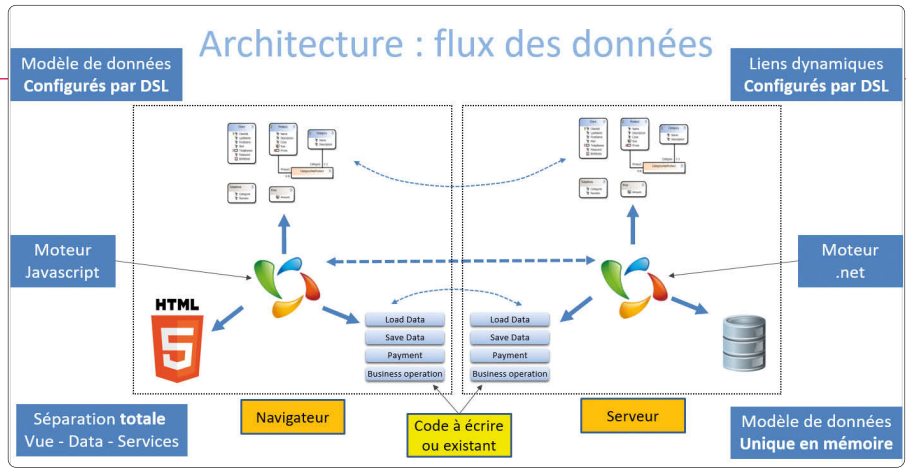
Ces définitions font apparaître une idée assez répandue concernant notre métier : la planification d'avance d'une suite d'opérations répétables, idée sous-jacente à la plupart des programmes conçus et développés aujourd'hui.
Or, adopter cette idée c'est déjà s'enfermer dans le schéma qui a tendance à confondre programmer et coder. Nous allons voir qu'on peut imaginer des situations - certaines sont même très courantes - où la description d'un programme ou d'une partie de programme ne se fait pas par du code (suite d'opérations à exécuter les unes après les autres).
Un programme accomplit une tâche précise, satisfait un besoin exprimé. Il pourrait afficher sur votre téléphone la liste des films que vous aimeriez voir, il pourrait même vous proposer de les acheter et de les voir. Un autre pourrait faire une prévision météo selon un modèle, c'est-à-dire des données et des équations différentielles. D'autres programmes pourraient servir à animer des dessins, jouer à Go, aider à diagnostiquer une maladie, aider à apprendre un sujet, conduire votre voiture ou vous aider à créer un autre programme. Un code est une représentation par des nombres ou des symboles particuliers : le code Morse utilisé pour coder l'alphabet avec des suites de tirets et de points ; le code ASCII ou UNICODE utilisé pour numéroter les caractères. On géolocalise avec un code, les 3 nombres correspondant à la longitude, la latitude et l'altitude. L'alphabet est une façon de coder les sons de notre parole. On code aussi quand on échange des secrets.

## De l'imagination au code

Programmer c'est **créer** un programme. C'est un travail intellectuel de compréhension, de conception et de création alors que coder c'est transcrire sous un format accessible à une machine le résultat de cette création. Ecrire un roman n'est pas la même chose qu'aligner des lettres lisibles par un lecteur. Créer un programme n'est pas aligner des instructions lisibles par une machine.
Au niveau de nos machines, un programme - l'œuvre du programmeur - est représenté par une suite de zéros et de uns disposés dans sa mémoire. La machine est conçue pour les exécuter, les faire circuler dans ses circuits, comme on fait circuler des pièces à assembler dans une chaîne de montage d'une usine.
Les *premiers* programmes dans les années 1940 ont dû être codés directement avec des zéros et des uns et ils étaient *débuggés* au voltmètre. Mais très vite des programmes produisant d'autres programmes ont été codés. Ces programmes de plus haut niveau permettent de transmettre les instructions à la machine dans un *format* plus *facile* à déchiffrer par un humain qu'une suite de zéros et de uns. On parle de langage de programmation. Plus la syntaxe de ce langage est proche de la machine et de son fonctionnement, plus on a tendance à dire que c'est du code. Plus elle est proche du domaine d'application du programme, moins on aura tendance à dire que c'est du code.
Il y a plusieurs types de langages : les langages généralistes comme le C ou F# et les DSL (Domain Specific Language). Il y a des DSL célèbres comme SQL et le CSS, et d'autres moins célèbres comme Gherkin utilisé pour décrire des règles dans l'approche BDD ou TDD. Utiliser un DSL réduit le code le plus souvent.






Un exemple : si on doit colorier en rouge une zone de l'écran, on peut opter pour le codage et écrire des instructions avec les primitives de notre carte graphique en utilisant un langage et un style impératif comme le C. On va alors adresser un à un les pixels de la zone en question pour fixer leur couleur à rouge. Ou on peut opter pour une voie déclarative et écrire en CSS l'expression : **.zone { background-color : red ; }** Malgré la présence un peu cryptique des accolades et du point-virgule, cette expression n'est pas du code, étant donné sa concision, son adéquation avec la tâche à accomplir et son aspect déclaratif.

Tous les DSL ne sont pas faciles à lire, certains comme les expressions régulières, bien que très concis, peuvent être aussi très cryptiques. Devineriez-vous le sens de l'expression suivante (en Javascript) : **/^1?$|^(11+?)\1+$/** (1)

Des langages de programmation, il y en a beaucoup. L'histoire, l'égo, le snobisme, le syndrome NIH (Not Invented Here), le manque d'imagination ou tout simplement la jeunesse de l'informatique ont fait qu'ils se ressemblent tous. Ils sont généralistes, élémentaires, proches de la machine et visent à tout faire. Ils invitent régulièrement à réinventer la roue. Ils ont donné tout ce qu'ils pouvaient. Pour aller au-delà, pour gagner des ordres de grandeurs sur la concision, pour écrire peu ou pas de code, il faut chercher la solution ailleurs.

## Quand programmer n'est pas coder

Un environnement et une interface de communication adéquats peuvent remplacer du code.

Ainsi quand on programme une cafetière ou un magnétoscope, quand on utilise un service comme *ifttt.com* (If This Then That) ou Flow (Microsoft) ou quand on saisit une formule de calcul dans Excel, on est en train de cliquer sur quelques boutons ou saisir quelques paramètres, et automatiquement l'électronique sous-jacente ou un ou plusieurs ordinateurs distribués aux quatre coins du monde se mettront en action pour exaucer notre intention. Peut-on dire qu'on a codé IFTTT, Excel ou la cafetière ?

Dans le cas de la cafetière la tâche était assez simple ; « chauffer le café pour 8h30 sauf le week-end ». Le programme consiste à choisir deux ou trois paramètres, de s'assurer que l'eau et le café soient disponibles et le reste est automatique. Le programme est court et ses paramètres correspondent au besoin. C'est le cas aussi pour les recettes d'IFTTT ou les formules d'Excel : la création du programme est simple et ses paramètres sont accessibles facilement et surtout ils sont en adéquation avec le besoin.

De plus, si on devait coder un programme pour comparer graphiquement le chiffre d'affaires trimestriel d'un produit sur les trois dernières années, sans Excel ça aurait pris des mois de travail pour un résultat moins abouti et qui de plus se périmerait très vite.

## Code périssable ou code réutilisable

Le code est périssable. Quand les premiers disques optiques sont arrivés sur le marché, leurs interfaces de programmation étaient assez pauvres, grosso-modo on disposait de 640 millions de cellules et on pouvait écrire dedans ! Si on voulait organiser l'espace et nommer des zones, il fallait écrire le code correspondant à ce besoin avant de pouvoir utiliser le nouveau support de stockage ! Un autre fabriquant, un autre matériel, une autre interface de programmation et il fallait tout recoder.

Aujourd'hui nos systèmes d'exploitation modernes contiennent déjà ce code. A travers des standards qu'ils utilisent, ils garantissent que les périphériques futurs fonctionneront de la même façon. Que le périphérique de stockage soit magnétique, optique ou électronique, pour celui qui veut stocker 300 octets sur le support, le code à écrire doit être court, standard et adéquat avec le besoin de stockage. Il est surtout indépendant des particularités du périphérique.

On peut voir le système d'exploitation comme un logiciel, comme un outil, un environnement adéquat qui permet de se réutiliser. Il permet la réutilisation des entrées/sorties, des accès aux périphériques. Il élimine la nécessité d'écrire du code spécifique. Ici la réutilisation n'est pas le fruit d'un langage ou d'un style de programmation particulier, mais de l'existence d'un environnement logiciel cohérent pour exécuter les programmes. Ce que nos systèmes d'exploitation font pour le matériel, les navigateurs le font pour les systèmes d'exploitation. Quand le besoin exige de tourner sur quasiment tous les processeurs, de supporter tous les systèmes d'exploitation, on a le choix entre coder N fois, une fois pour chaque environnement, ou coder une fois pour le navigateur grâce au standard du web (HTML, CSS, Javascript, HTTP, HTTPS).

Evidemment l'approche navigateur ne convient pas bien à tous les programmes. C'est utopique de chercher une approche, un environnement ou un langage universel.

## Programmer sans coder, utopie ou réalité ?

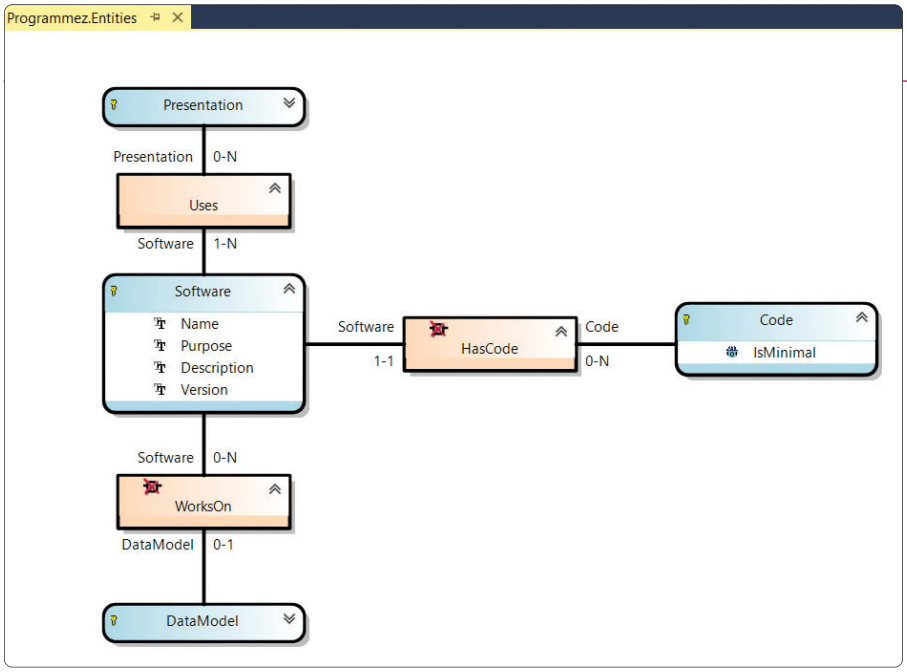
Pour certains besoins, Excel nous permet de programmer sans coder. Pour d'autres besoins, c'est le navigateur qui nous délivre de l'inutilité de recoder N fois le même programme. Les premiers blogs ont été codés à la main, de nos jours on trouve facilement des outils en ligne pour pouvoir monter son blog en quelques clics. Pareil pour des sites Web statiques, nul besoin de code mais d'un outillage et d'un environnement adéquats. Des outils comme Blockly (Google) ou Scratch (MIT) permettent l'initiation à la programmation de façon visuelle avec un environnement graphique sans code (textuel) mais l'esprit reste très orienté code impératif.

D'autres outils comme Bubble (bubble.is) permettent de créer et d'héberger des applications simples. Dans le domaine des jeux on peut parler de BuildBox, chez Microsoft on peut parler d'AppStudio ou de PowerApps, ces outils visent tous à supprimer complètement le code.

Alan Kay (Prix Turing 2003), dans son excellent discours « The computer Revolution Hasn't Happened Yet », rêvait du jour où au lieu d'écrire du code on cultiverait du code ! Le travail du programmeur ressemblerait plus à celui du jardinier qu'à celui de l'industriel. Ce jour n'est peut-être pas si loin.

Des systèmes comme Watson (IBM) savent répondre à des questions et analyser des données en *lisant* des documents codés pour les humains (en anglais) et pas en exécutant du code écrit pour la machine (en C# ou Javascript). Le programme AlphaGo (Google), champion du monde de Go, n'a pas été codé dans le sens où un code écrit par un humain lui a décrit la marche à suivre, il n'a pas gagné parce qu'il exécute très vite des instructions. A priori il n'existe aucun document, aucun livre qui

*(1) – c'est un détecteur de nombre premier, plus exactement de nombre pas premier. Si on code un nombre N par une chaîne contenant autant de 1 et on le fait tester par l'expression régulière, le test échouera pour les N qui seront premiers.*




explique comment battre le champion du monde (humain) de Go. Ce n'est pas le code d'AlphaGo qui est champion du monde, mais l'expérience d'AlphaGo en tant que robot qui sait *apprendre et s'améliorer* : il a analysé des millions de parties de Go et en a tiré *ses conclusions* sur comment bien jouer. C'est avec ce même type d'approche de **Machine Learning** que Skype (Microsoft) peut traduire en direct des conversations de vive voix entre sept langues différentes. D'autres programmes sont entrainés à identifier des tableaux de Van Gogh ou des photos de chats. Ces programmes ne fonctionnent pas au code, s'ils se trompent - peut-on parler de bug ? – il n'y a nulle part un code qui permettrait au programmeur d'aller le corriger. La Machine Learning ne permettra pas à la CAF de faire chaque mois le calcul des allocations familiales pour des millions de foyers, mais elle a son domaine, domaine qui est inaccessible au code impératif classique.

## Est-il possible de programmer sans code en 2016 ?

La réponse n'est pas simple. Avec les outils adéquats et l'environnement adéquat, dans des domaines précis ou pour des cas simples, il est possible de créer des programmes sans code.

Les développeurs de jeu l'ont bien compris : ils utilisent des moteurs 2D/3D comme BabylonJS ou Unity, ils utilisent des moteurs physiques, et minimisent ainsi leur code. Ils ne recodent pas les aspects indépendants du jeu, les ombres, les lumières, les textures, le positionnement dans l'espace, les reflets, les rebonds... Ils ne s'occupent que des aspects qui nécessitent intelligence et imagination humaine.

La bonne question, celle qui est plus réaliste, c'est donc comment faire pour programmer avec un minimum de code ? En effet le code, par son manque de concision, par sa fragilité, par le fait qu'il est destiné aux machines et pas aux hommes, constitue un goulot d'étranglement difficile à produire, coûteux à maintenir. Contrairement au français, en informatique c'est plus facile d'écrire du code que d'en lire. Pour améliorer un code, il faut le lire et le comprendre. On passe 10 fois plus de temps à lire et déchiffrer du vieux code que de le recoder. Que faire ? Passer son temps à coder et recoder ce qui a été codé mille fois déjà, ou essayer d'organiser son code de telle sorte que les parties réutilisables soient réutilisées ?

C'est dans cette perspective que Aspectize a mis au point des outils pour développeurs : deux DSL intégrés à Visual Studio, un DSL textuel pour décrire les liens entre l'interface utilisateur et les données, et un DSL graphique pour décrire le schéma des données ; deux moteurs techniques, un côté client pour s'occuper de la production automatique du HTML dans le navigateur et un autre côté serveur pour s'occuper de la lecture, écriture et transport des données. Et un portail DevOps pour déployer, configurer et gérer les versions des applications dans Azure. Dans le cadre des applications de gestion Web et Mobile, SaaS, en utilisant ces deux DSL et en écrivant le code métier en .net et Javascript, un développeur peut programmer à résultat égal avec 20 voire 30 fois moins de code. Programmer c'est orchestrer le flux de l'information entre les machines et les hommes. Les ordinateurs sont partout, sur nos bureaux, dans nos poches et bientôt dans n'importe quel appareil qui bénéficierait d'établir une communication avec un autre. Le logiciel transforme tous les métiers, nous ne devons pas être freinés par le coût et la complexité galopante du code.

Minimisons-le.

# La génération de code

Si un code s'exécute, s'il n'est pas sous forme binaire dans votre environnement et si vous ne l'avez pas écrit, c'est que quelqu'un d'autre l'a écrit ! Quelqu'un d'autre ou un robot générateur de code, commandé par vous.

La génération de code peut se faire à différents moments : elle peut se faire pendant la phase dite design time, c'est-à-dire lors de l'écriture du programme, avant la compilation en binaires, c'est ainsi que les wizards et designers de Visual Studio génèrent du code standard et répétitif ; la génération peut se faire dans la phase d'exécution de votre programme juste avant que le programme démarre. C'est ainsi que les développeurs Asp.net bénéficient de la souplesse de cet environnement. En effet Asp.net génère, compile et exécute du code quand votre programme démarre. Elle peut se faire même plus tard en pleine exécution de votre programme, c'est ainsi que le XmlSerializer de .net fonctionne. On pourrait même dire que tout .net fonctionne ainsi, puisque le vrai code, le binaire, les zéros et les uns pour la machine sont produits en .net au fur et à mesure de l'exécution du programme. Ainsi certaines optimisations peuvent être faites tardivement une fois qu'on a découvert l'environnement de l'exécution : le type du système d'exploitation, client ou serveur, le type et le nombre de processeurs, etc. Pour tout savoir sur pourquoi et comment utiliser un générateur de code, lire : "La génération de code : un bien ou un mal ?" sur le blog de SoftFluent (24/11/2015).

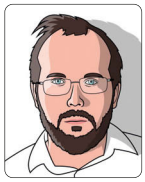
Dans la catégorie de générateurs de code, on peut citer CodeFluent Entities (SoftFluent) et la plateforme SaaS de Generative Objects.

CodeFluent Entities est un outil de modélisation et de génération d'applications .net, intégré à Visual Studio. Il permet de générer les couches métier et d'accès aux données des applications pour accéder aux bases de données relationnelles célèbres du marché à partir d'un modèle. Le code généré est lisible et extensible.

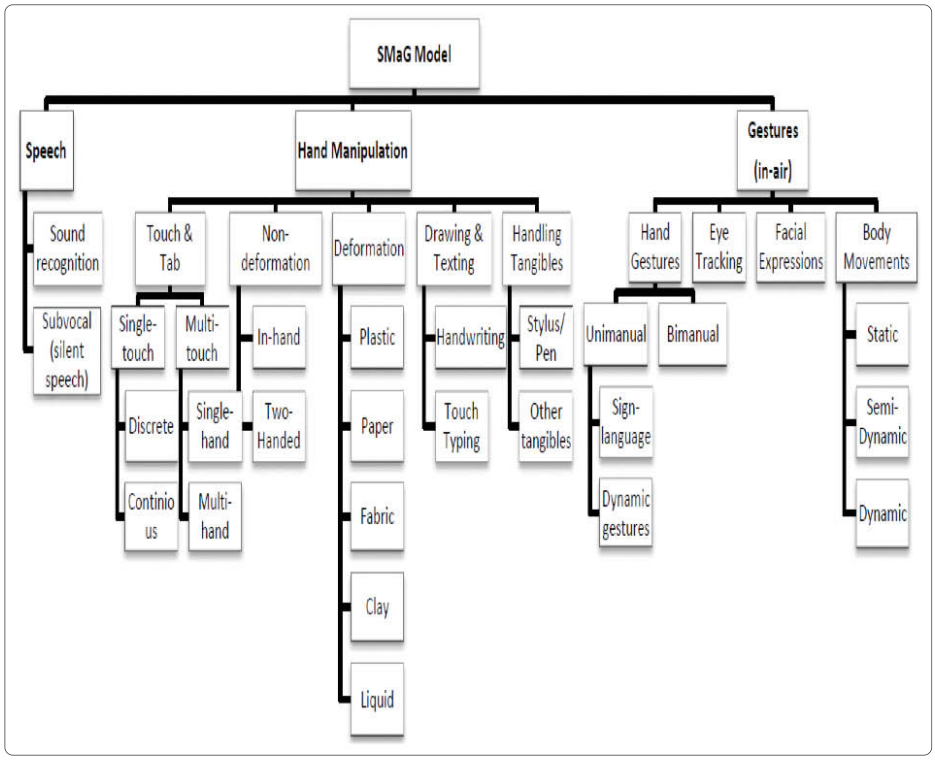
La plateforme de Generative Objects permet de générer des applications métiers Web complètes : base de données, couche métier, couche APIs Json/REST, front HTML/JS, tests unitaires code et UI (Selenium). La génération de code est transparente pour les applications simples, et le code généré, de qualité humaine, est extensible par des développeurs sur les projets plus complexes.

# CLI, GUI, NUI, OUI : tout est question d'interface

*Durant la conférence GitHub Satellite 2016 d'avril dernier à Amsterdam, nous avons pu suivre une conférence très intéressante sur comment créer et prototyper son objet connecté, donnée par Erika Stanley, le fameux IoT, avec une réflexion sur l'interface et l'interaction homme – machine que l'IoT modifie profondément.*



François Tonic
*Programmez!*




Quand on crée un objet, il ne faut pas oublier un petit élément tout bête : l'interface, l'interaction homme-machine. Cette interaction n'est pas identique selon les formes et la nature de l'objet.

Nous définissons 4 principales interactions (modèles d'interaction) :

- CLI : ligne de commande
- GUI : interface graphique
- NUI : interface dite naturelle
- OUI : interface dite organique

En IoT, il ne faut jamais oublier la dimension interface / interactions. Car, l'objet peut prendre différentes formes, différents moyens d'interactions, voire, aucune interaction avec l'utilisateur. Basiquement, nous avons le tactile, l'écran, les pointeurs. Et les objets sans écran physique se multiplient.

Plusieurs défis se dressent devant vous dans le design IoT : nous n'avons plus forcément de « premier plan » pour définir l'interface, la capacité à s'intégrer avec les autres plateformes et objets, découverte des fonctionnalités, basse consommation. Trois principes peuvent être définis et influencer son choix : la forme (form factor), les interfaces naturelles et les microinteractions. En NUI, les entrées sont diverses : le toucher, les gestes (gesture), la voix, les mouvements du corps / yeux / tête. En sortie, les réponses aux actions que nous avons en entrée peuvent se présenter sous une forme visuelle, sonore ou haptique. Par microinteraction, comprenons qu'à un moment donné, l'objet tournera autour d'un seul cas d'usage et n'aura alors qu'une seule tâche principale. Selon le modèle d'interaction, l'expérience sera totalement différente. En CLI, nous serons en environnement statique et direct. En GUI, l'environnement se voudra responsive et indirect dans les interactions (dans le sens où nous utilisons une souris ou tout autre pointeur pour interagir), en NUI, nous n'avons pas d'intermédiaire pour agir. Pour exemple, nous utiliserons la voix, les gestes.

## Le modèle OUI repose sur 3 piliers

Par Organic User Interface, OUI, nous entendons une combinaison d'interface naturelle et d'interface tangible (TUI) pour être plus proche des sens humains et avoir des interactions naturelles et fluides. Cependant, la définition du OUI varie d'un expert à un autre.

Aujourd'hui, le modèle OUI repose sur un modèle d'interaction basé sur SMaG. SMaG est la contraction de Speech Manipulation air-Gesture, bref de la voix et des gestes. L'OUI ne définit donc pas forcément une nouvelle génération d'interface, mais reprend d'autres types d'interfaces pour aboutir à autre chose.

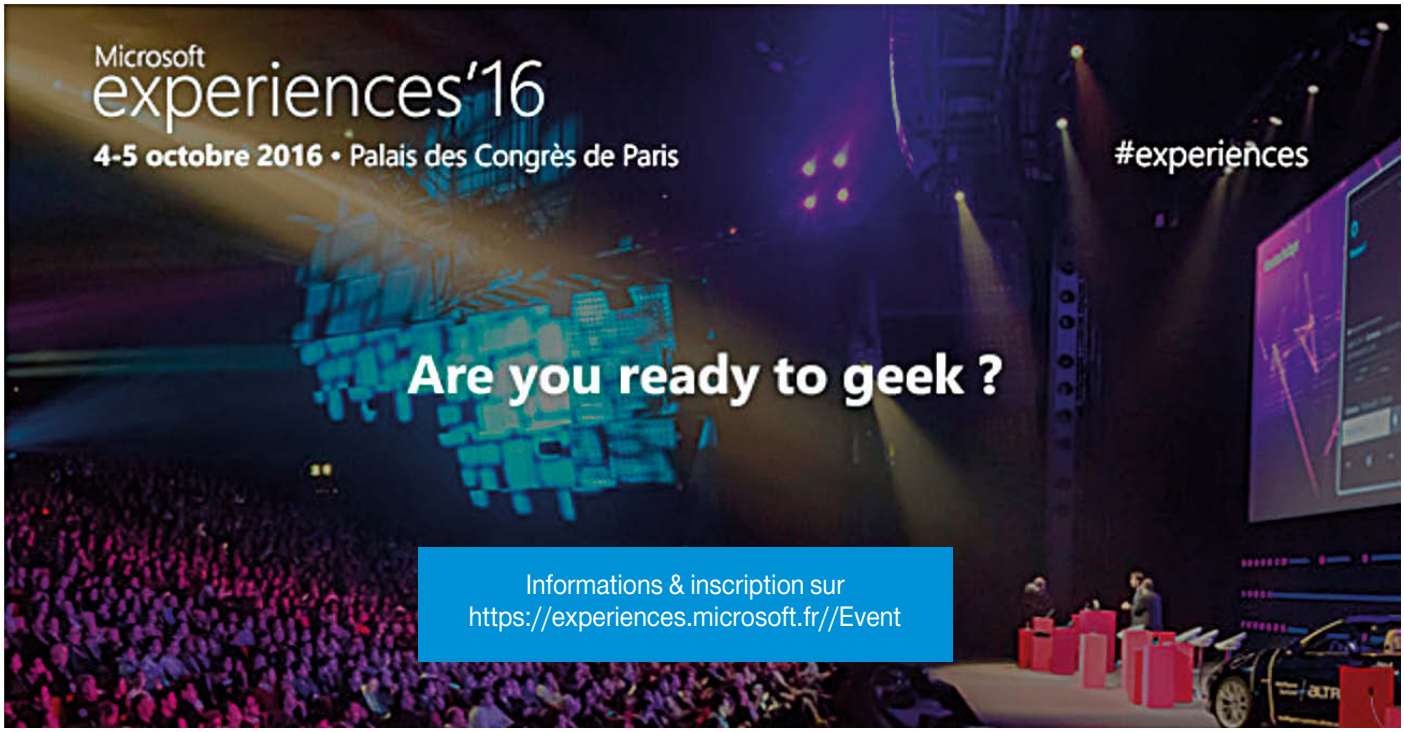
Dans l'approche SMaG, nous aurons, pour simplifier les choses :

- La voix ;
- Les manipulations : tactile dite discrète, tactile continu (par exemple pour dessiner, écrire), la nécessité d'avoir une ou deux mains pour manipuler l'objet. La notion de tangible s'exprime par la nécessité d'avoir un stylet ou tout autre pointeur ;
- Gestes : principalement les mains, mais cela peut être le mouvement du corps, le tracking des yeux, le visage, etc.

Je sens que votre cerveau chauffe et vous ne voyez pas beaucoup d'intérêt à ce que j'écris. Prenons quelques exemples très simples pour comprendre les différentes notions :

| capteur | Entrée ou Sortie | Nature de l'interface | Interaction |
|---|---|---|---|
| écran OLED 0,96'' | Sortie | GUI | statique |
| capteur ultrason | Entrée | NUI + OUI | directe sans intermédiaire |
| capteur de gestes | Entrée | NUI + OUI | intuitive<br>fluide<br>extensible |
| matrice LED | Sortie | NUI | directe sans réelle interaction |

Le capteur de gestes (ci-après) rentre dans le modèle SMaG et donc l'OUI.




**Pour sa dixième édition, l'événement Microsoft tech·days se transforme et devient Microsoft experiences'16. Les 4 et 5 octobre au Palais des Congrès de Paris, nous vous proposons de vivre de nouvelles expériences avec nos clients, nos partenaires et nos communautés.**

**Scott Guthrie,** Vice-Président de l'activité Cloud et Entreprise de Microsoft

**Plénière d'ouverture**

Scott Guthrie vous présentera la vision cloud de Microsoft et les dernières nouveautés. À ses côtés, nos clients livreront leur témoignage, démos exclusives à l'appui. Venez découvrir comment les technologies Microsoft vous aident à transformer votre organisation !

La journée du 5 octobre sera dédiée aux enjeux technologiques de la transformation numérique. Au programme : une plénière d'ouverture, 150 sessions et talks techniques, plus de 100 exposants et des meet-ups en fin de journée. Découvrez un aperçu de l'agenda ci-dessous.

Venez vivre de nouvelles #experiences...
Réservez dès maintenant votre place !

## LE DEVELOPPEMENT WEB : TOUR D'HORIZON

De nos jours, faire du développement web nécessite de connaître une quantité incroyable de technologies, frameworks et librairies. Nous vous proposons de faire un tour d'horizon du web d'aujourd'hui. Nous parlerons backend, frontend et nous évoquerons aussi le futur. Si vous voulez savoir où en est le web sans des heures de recherche, vous êtes au bon endroit !

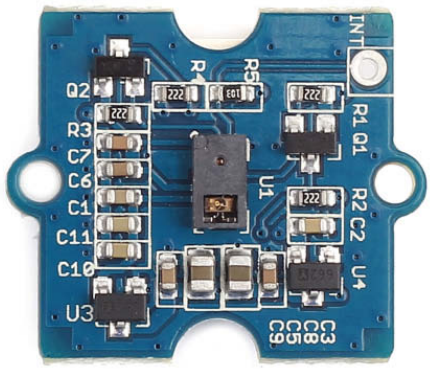
## QUELLES ARCHITECTURES POUR VOS APPLICATIONS CLOUD, DE LA VM AU CONTENEUR : ÇA PAAS OU ÇA CAAS ?

Dans cette session, nous passerons en revue les différents choix d'architectures rendus possibles par Microsoft Azure pour vos applications : machines virtuelles, services de plateformes comme Azure Service Fabric et Azure App Services, solutions d'hébergement de conteneurs Docker sont autant de possibilités qui s'offrent à vous dès aujourd'hui !

## UWP + XAMARIN - DEVELOPPER SON APPLICATION MOBILE EN C# POUR WINDOWS, ANDROID ET IOS

Tour d'horizon des solutions UWP (Universal Windows Plaform) et Xamarin pour développer des applications mobiles en C#. Découvrez comment partager cette base de code (et même vos interfaces en s'appuyant sur Xamarin Forms) entre vos différents projets Windows, Android et iOS.

## Hum, moi plus rien comprendre !




Par définition, la NUI fournit des interactions intuitives, et donc naturelles, dans le monde réel. A cela se rajoutent deux types de matériels / objets :

- Déformable ;
- Non-déformable.

Par non-déformable, il faut entendre des modules à formes fixes comme un écran non souple, un bouton physique, etc. Par définition, il s'agit de tout matériel, tout objet (physique ou non) que l'on ne peut pas déformer, modifier (dans sa forme). Prenons le projet ARA, le smartphone modulaire de Google, il est à la fois non-déformable et déformable (socle matériel avec module par défaut et modules interchangeables). Un écran souple est déformable. Il peut donc y avoir un impact réel et concret sur l'interaction et les entrées / sorties.
Un objet déformable : https://www.youtube.com/watch?v=piKBDjyYpW0
Une chose est sûre : il faut que les interactions soient propres, lisibles, répondent au contexte et au type d'objet. En effet, le capteur va varier selon tous ces points critiques. Vous n'allez pas mettre une gesture pour allumer une lampe, surtout en absence de lumière. Le contexte de l'objet est crucial à comprendre pour proposer les bonnes interactions, et les bons capteurs. L'usage de l'objet est un moyen de savoir quelles interactions / interfaces sont possibles et souhaitables. Si mes mains sont occupées, vous n'allez pas intégrer une gesture ni un moteur haptique sur un pot de fleurs (une fleur reconnaît les vibrations c'est bien connu). Ce sont des évidences, mais il faut toujours avoir en tête l'usage et le contexte. En milieu alimentaire ou médical, il faudra s'assurer de la conformité des capteurs (exigence de stérilité notamment).
Autre question à se poser : suis-je dans des usages de capteurs 2D ou 3D (= dimension spatiale de son environnement) ?
Pour tout contact répondant à un usage ou une fonction, l'objet doit, en retour, fournir une réponse.

## Des objets sans affichage physique

Nous sommes habitués à avoir des informations, des notifications, des interactions grâce aux surfaces d'affichage comme des écrans LCD, OLED, tactile ou non. Mais aujourd'hui de plus en plus d'objets ne possèdent pas d'affichage. Les exemplaires sont multiples : les IoT de type Amazon Echo, raquette connectée, wearable vestimentaire... Ces objets existent depuis longtemps, mais aujourd'hui, ils possèdent une certaine autonomie et des capteurs plus nombreux.
Pour pouvoir interagir avec eux, plusieurs solutions : connexion avec un autre objet ou utilisation de ce que l'on appelle des microinteractions. Ces actions doivent répondre à un usage très précis. Dans ce modèle : nous avons le déclencheur, la règle (fonction) à exécuter, un retour d'information, et on reboucle.
Nous pouvons avoir une interaction par une ambiance lumineuse ou une action lumière par une matrice led. Imaginons un vêtement connecté couplé à un système de navigation comme un GPS, Maps, etc. Comment faire de la navigation en temps réel quand vous êtes à vélo ou déficient visuel ? Vous pouvez envisager un guidage audio et/ou par des notifications haptiques : droite, gauche, arrêt. Mais se pose alors le problème de la compréhension immédiate de la notification haptique : avoir des vibrations claires et identifiables facilement. Un autre problème survient : ne pas multiplier inutilement les notifications au risque de perdre l'utilisateur. Quelles informations doivent déclencher une action auprès de la personne et comment ? Nous n'avons pas abordé ici les problématiques de communication et de traitements des données des capteurs et de l'environnement de son IoT. La connectivité est cruciale et chaque type de connectivité correspond à des usages très précis.

| | voix | données | audio | vidéo | état |
|---|---|---|---|---|---|
| bluetooth | x | x | x | | |
| BLE | | | | | x |
| WiFi | x | x | x | x | |
| direct WiFi | x | x | x | | |
| ZigBee | | | | | x |

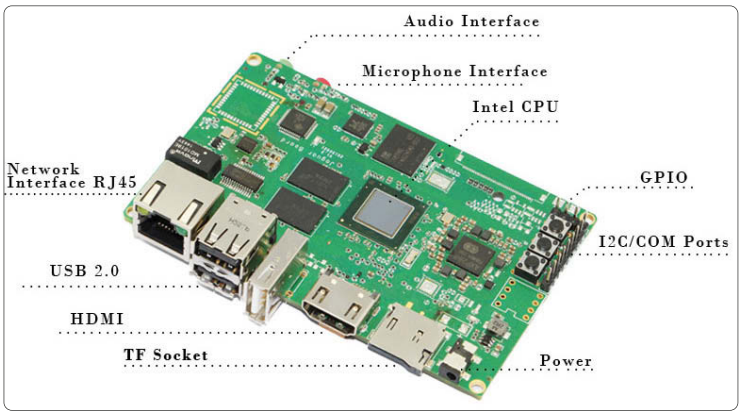
Ayez toujours en tête que votre IoT doit être intelligemment conçu avec les bons capteurs, la bonne forme et les bons principes d'interaction homme – machine. L'environnement d'usage doit peser sur vos choix. Par exemple, ne pas utiliser une gesture quand il n'y a pas de lumière, ne pas utiliser un affichage quand l'utilisateur ne doit pas l'utiliser ou ne pas le voir, etc.

# Et le wearable ?

Il y a 1 an, on parlait partout de wearable, de l'information/technologie à porter sur soi. Si la tempête médiatique est retombée, ce n'est pas pour autant que le wearable est un échec. Mais le marché est très lent à démarrer. Nous voyons les matériels wearables, de type montres et bracelets sportifs, se vendre, mais les usages ne sont pas encore assez marqués, ni les fonctionnalités. L'autonomie est une des questions essentielles avec la durée de vie de ces matériels, les mises à jour logicielles, etc. L'autre wearable est orienté prototypage, maker, développeur. Et là, on constate que la situation commence réellement à bouger. Ainsi, durant la dernière Google I/O, le projet Jacquard a été dévoilé par Google, en partenariat avec Lewi's. Le plus intéressant dans ce projet est le tissage d'un tissu "électronique compatible" : fils conducteur, zones tactiles/sensibles pour pouvoir interagir. Si l'idée est intéressante, le passage à la production industrielle est encore problématique : usages, fonctionnalités à proposer, durabilité des fils, usure, etc. Les premières démos d'interactivité avec un bouton dédié montrent clairement que nous sommes qu'au début de l'électronique vestimentaire. Le modèle d'interactivité se pose : faut-il une approche NUI/OUI ? Car nous sommes dans des contextes non tangibles, sans écrans et il faut des actions simples, car si vous êtes en vélo, l'utilisateur doit pouvoir agir très rapidement et recevoir des notifications compréhensibles sans perdre de vue la route. Côté plateformes de prototypages pour les makers et développeurs, il existe d'ores et déjà des cartes pertinentes : Metawear, LilyPad, Flora, Gemma... Seeed Studio avait créé un super projet : Seeed Film. Il s'agissait d'un film électronique souple avec capable de supporter différents capteurs (Seeed Motion Frame). Malheureusement ce projet est aujourd'hui arrêté, mais toujours trouvable. Le constructeur conseille maintenant la plateforme Xadow. La partie développement a été améliorée par rapport aux premières versions et au kit RePhone parfois difficile à monter et à coder.

# JaguarBoard : enfin une carte x86 digne de ce nom !

*À la rédaction, nous l'attendions avec impatience. Et enfin, début août, nous avons pu découvrir la JaguarBoard, en provenance directe de Hong Kong. Elle est plus imposante qu'une Raspberry Pi 3 mais moins que la grande Pine64. Premières manipulations.*




François Tonic
*Programmez!*

D'emblée, la carte s'impose par ses dimensions et son poids. Le gros radiateur ne passe pas inaperçu. Sa mission est de refroidir le processeur Intel Atom. Pour le reste, c'est du classique : 3 USB 2.0, 1 port Ethernet (10/100), 1 HDMI 1.4, connecteurs audio, port pour carte mémoire, 4 GPIO et des broches I2C/Com. Le plus intéressant est un stockage interne par défaut au format eMMC (16 Go). Côté mémoire vive, 1 Go : bien, mais aurait pu mieux faire. Par contre, au prix auquel la carte est vendue, on regrette l'absence d'un Ethernet 1 Go, du réseau sans-fil, d'un vrai GPIO, voire, même d'un stockage de 32 Go. Par défaut, la Jaguar embarque une distribution Linux, une Fedora Server. Vous pouvez utiliser des systèmes Linux (Ubuntu, CentOS, Fedora) et Windows. C'est l'avantage d'être sur une architecture x86. Car finalement, la Jaguar n'est pas une simple carte à la ODROID ou Raspberry, mais un mini-PC en puissance. Pour la partie graphique, la GPU est une Intel HD Graphic. On ressent très rapidement les limitations du port Ethernet.

Il ne faut cependant pas attendre des miracles côté performances de la Jaguar. Ainsi, en démarrage à froid, il faut environ 33 secondes pour pouvoir se loguer. Malgré tout, la Jaguar peut se révéler plus polyvalente que d'autres boards grâce à son architecture x86. Ainsi, si vous voulez monter votre barebone Docker avec la Jaguar, pas de souci ! Et l'utilisation de Docker ne pose pas de problèmes sur la board, mais attention aux performances de la carte et surtout du stockage limité à 16 Go. Passons sur l'utilité des GPIO, bien trop limités pour être intéressants. Par défaut, vous ne pourrez pas installer Windows, vous devez mettre à jour l'EFI via le shell EFI. Cependant, la mise à jour ne permet pas toujours de booter en mode Windows. La carte a du potentiel, mais les instabilités, la nécessité de mettre à jour l'EFI, le stockage interne limité, gâchent l'utilisation.

## LES +

- Architecture x86
- Stockage interne
- Dimensions

## LES –

- Prix
- Ressources matérielles
- Chauffe de la carte
- Introuvable en France
- Pas de WiFi

# Jetez votre Raspberry Pi ! Il y a (bien) mieux !

*Si Raspberry Pi est le nom qui revient le plus souvent, la carte n'est plus toute seule dans sa catégorie pas chère, fonctionnelle et pratique, que ce soit en ARM ou en x86. Depuis 2 ans, l'offre s'est considérablement diversifiée. La Pi 2 avait su remettre Raspberry Pi en ordre de bataille mais la Pi 3 a montré peu de franches évolutions. Cette version nous a déçus surtout quand on commence à regarder ailleurs.*

*Raspberry Pi 3*

François Tonic
*Programmez!*

Raspberry Pi serait la seule carte de sa catégorie (petite, pas chère, puissante) pour beaucoup de personnes. Mais il en existe de nombreuses autres, peu connues en France. La petite déception que nous avons eue sur la Pi 3 nous incite à regarder les alternatives. Nous avons sélectionné l'ODROID-C2, la Pine64 et la Banana Pro.

## Tableau comparatif

En rouge : les points (vraiment) négatifs
En vert : les points (très) positifs

## La Pi n'est plus la star des cartes

Les Pi et Pi 2 avaient su créer la surprise et répondre à un besoin réel pour créer de petits barebones, des mediacenters, des mini-PC.
Pour 35 $, l'offre était excitante, même si l'addition monte rapidement pour ajouter l'alimentation, la carte SD, les câbles, le clavier, etc. La Pi 3, sortie à peine 1 an après la Pi 2, est une simple itération :

- Un processeur plus véloce et 64 bits qui n'apporte finalement pas grand chose car la mémoire vive reste bloquée à 1 Go. Les distributions Linux 64 bit pour la Pi 3 ne sont pas encore stabilisées et peu disponibles ;
- Un port Ethernet toujours bloqué à 10/100 Mb ;
- USB 2 uniquement ;
- Une connectique de stockage limitée : SD et un USB trop lent pour un disque externe ;
- HDMI 1.4.

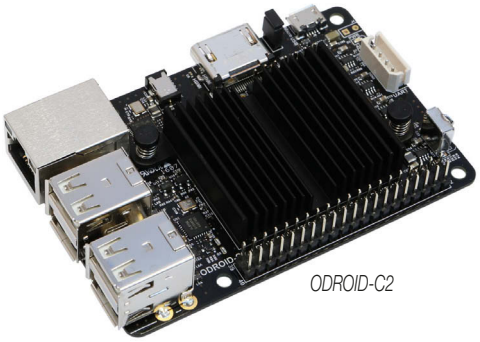
Attention, je ne dis pas de revendre votre Pi. Elle demeure une excellente carte. Mais les choix conservateurs brident beaucoup les performances et les possibilités. L'ajout du WiFi et du Bluetooth est une excellente nouvelle mais ne doit pas faire oublier le reste. Aucun port eMMC ou SATA n'est disponible ce qui oblige à passer par une carte SD (forcément aux performances limitées) ou par un stockage USB comme le PiDrive. Mais, l'utilisation du PiDrive s'est révélée décevante : performances médiocres, pas de boot direct sur le disque.
Le port HDMI reste bloqué en version 1.4 ce qui bride les performances d'affichage, notamment avec les nouveaux formats comme le 4K, mais il a le mérite d'être correct et de s'afficher partout ou presque. On attend toujours le support stable d'Android, en plus de Linux et de Windows 10 IoT.
Pi a l'avantage d'avoir une très large communauté, très active et du support de nombreux éditeurs et constructeurs (outils, capteurs, extensions). C'est un avantage énorme sur la concurrence dont les communautés sont faibles, voire, inexistantes en France. Et surtout, très tôt, Raspberry Pi a misé sur les makers, les développeurs. Les GPIO constituent une voie royale pour les hacks et montages.
Raspberry Pi n'est pas morte, absolument pas. Elle a eu le grand mérite d'ouvrir de nouvelles perspectives pour les utilisateurs, les makers, les développeurs. Il est normal que d'autres projets s'y soient engouffrés mais la Pi demeure la référence.
Malgré tout, Raspberry Pi doit maintenant démontrer sa capacité à se renouveler et à oser de nouveaux choix. La Raspberry Zero ne nous a pas convaincu surtout en comparaison avec la C.H.I.P.

| | Raspberry Pi 3 | ODROID-C2 | Banana Pro | Pine64 (1) |
|---|---|---|---|---|
| CPU | Cortex A53<br>1,2 Ghz | Cortex A53<br>2 Ghz | Cortex A7<br>1 Ghz | Cortex A53<br>1,2 Ghz |
| CPU 64 bits | oui | oui | non | oui |
| GPU | VideoCore IV | Penta Core | Mali400MP2 | Mali400MP2 |
| Mémoire vive | 1 Go | 2 Go | 1 Go | 1 Go (2) |
| Stockage | SD | SD<br>eMMC | SD | SD |
| Port SATA | non | non | oui (v2) | non |
| Ethernet | 10/100 | 1 Gb | 1 Gb | 1 Gb (3) |
| Sans-fil | Bluetooth<br>Wifi | Non | Wifi (4) | non (5) |
| USB 2 | 4 | 4<br>USB OTG | 2 | 2 |
| Vidéo | HDMI 1.4 | HDMI 2.0 (6) | HDMI 1.4 | HDMI 1.4 |
| OS supportés | Linux<br>Windows 10 IoT | Linux<br>Android | Linux<br>Android | Linux<br>Android |
| Interfaces | 40 GPIO<br>UART<br>I2C<br>CSI<br>DSI | 40 GPIO<br>UART<br>I2C<br>récepteur IR<br>I2S Audio | 40 GPIO<br>UART<br>I2C<br>I2S Audio<br>Camera | 2 x GPIO<br>UART<br>I2C<br>Camera |
| GPIO compatible Pi | - | pas totalement | oui | oui |
| Dimensions | 85x56 mm | 85x56 mm | 92 x 60 mm | 127 x 79 mm |
| écosystème / communauté en France | **** | * | ** | * |
| Tarif officiel (7) | 35 $ | 40-45 $ | env. 35 $ | 19 $ |
| Tarif en France (8) | 35-40 € | env. 59 € | 45-50 € | ? |

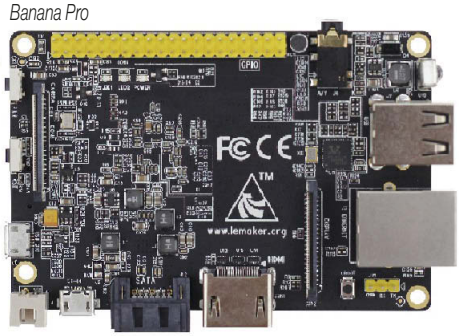
*(1) Version 1 Go testée à la rédaction*
*(2) Pine64 propose de 512 Mo à 2 Go, selon le modèle*
*(3) Pas sur toutes les versions de la board*
*(4) Bluetooth en option*
*(5) Module Wifi + Bluetooth en option, prix : 10,99 $*
*(6) Attention à la compatibilité des écrans*
*(7) Le tarif officiel est à relativiser car il faut rajouter la carte SD, l'alimentation. Vous pouvez quasiment doubler le prix*
*(8) Les écarts sont importants entre le prix officiel et ceux disponibles en France. La Pi 3 est facilement trouvable, l'ODROID était difficile à trouver mais la situation s'améliore et les tarifs sont élevés, idem pour la Banana Pro.*

## ODROID-C2 : une vraie bonne surprise !

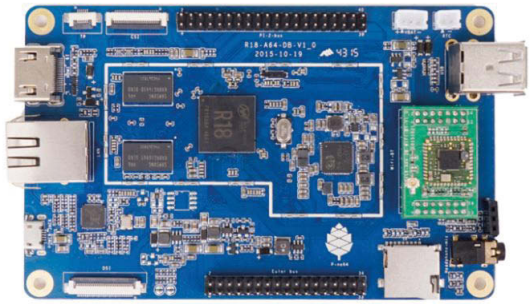
Quasiment inconnue en France, cette carte Coréenne était introuvable il y a encore quelques mois. Mais peu à peu, un intérêt commence à apparaître, notamment suite à la reconnaissance des cartes Banana Pi. Pour nous, l'ODROID est potentiellement la


ODROID-C2

Banana Pro



Pine64



« killer-pi » et l'alternative la plus cohérente. Le processeur est identique, avec une fréquence plus élevée. La véritable différence se fait avec des systèmes Linux 64 bits disponibles et stables. La mémoire vive de 2 Go est le minimum vital en environnement 64 bits.
Surtout, l'ODROID tape là où ça fait mal pour la Pi :

- Ethernet 1 Gb : et ça c'est un avantage non négligeable dans un monde où le 1 Gb est le standard ;
- Extension de stockage via un port eMMC ;
- HDMI 2.0 ;
- Support d'Android sur la board.

La possibilité d'ajouter très facilement un stockage est pour nous un critère important. Oui, on peut rajouter un stockage externe sur la Pi mais en USB et franchement, les performances actuelles sont médiocres. L'absence de tout stockage supplémentaire directement sur la Pi nous a toujours embêté. L'ODROID propose un port eMMC, plusieurs capacités sont disponibles, même si les prix de ces extensions sont un peu élevés (18 $ pour un stockage de 8 Go, tarif officiel). Il ne faut pas s'attendre à des performances explosives par rapport aux cartes SD mais elles sont globalement meilleures, notamment sur le temps du démarrage système (jusqu'à 15-20 % de gagner). Mais l'essentiel est la possibilité d'avoir un stockage interne. Mais il manque un port SATA pour gagner réellement en performances. Le port HDMI 2.0 est appréciable pour les débits et les formats les plus récents comme le 4K, mais attention à ne pas en abuser car ces vidéos exigent des ressources matérielles très importantes. Un démarrage à froid, avec Raspbian, montre la bonne performance de l'ODROID + eMMC : environ 16s contre 28s pour la Banana Pro (SD) et 32s pour la Pi 3 (PiDrive). L'ODROID tire profit du processeur et de la mémoire, plus généreux. En utilisation, on sent que la carte a de la puissance et qu'elle sait répondre aux exigences. Les 40 GPIO reprennent le schéma du modèle C1 et sont compatibles avec les GPIO de la Pi, exceptées pour les broches 37, 38 et 40, dédiées à l'analogique. Autre différence, les GPIO supportent uniquement le 3,3V. Cette limitation pourra être gênante pour certains montages car de nombreux capteurs nécessitent du 5V pour fonctionner correctement. On dispose du support de la librairie WiringPi. Cette librairie est importante car elle est utilisée sur la Raspberry Pi. Elle va faciliter l'utilisation d'extensions dédiées à la Pi et pour migrer des montages. Attention à bien repérer les correspondances des broches GPIO Pi sur l'ODROID. On n'aime pas du tout le format du connecteur d'alimentation, un format que l'on a rarement chez soi. Autre point négatif, les tarifs de la carte en France sont trop élevés (env. 59 € !) et les distributeurs peu nombreux. Il lui manque les modules sans-fil et un tarif plus attractif pour devenir le tueur de Pi !

## Pine64 : la véritable « killer pi » mais devra faire sa place !

Gros succès sur Kickstarter, le projet Pine64 est désormais disponible. Nous avons reçu la carte fin juillet en version 1 Go. Les délais peuvent être très longs (+ 2 mois parfois) et le support n'est pas très réactif. Cette carte se compare facilement à la Pi 3. Son prix est particulièrement intéressant : 19 $ (version 1 Go). Trois défauts : une GPU faible, 2 USB uniquement et aucun stockage interne. D'autre part, la connexion sans-fil est en option, dommage. Cependant, la Pine64 propose des ressources intéressantes : nombreux GPIO, support d'Android, Ethernet 1 Gb (sur certaines versions). Si vous n'avez pas besoin de connectivité sans fil, la Pine64 est réellement un excellent choix pour un prix agressif. Détail non négligeable : la carte est plus encombrante que les concurrentes. Là encore, petite déception. Actuellement, les systèmes disponibles se limitent à Ubuntu, Debian, Arc, Remix OS et Android. La communauté étant encore faible, les projets ne sont pas nombreux sur cette carte. Attention à bien choisir votre carte SD car les performances sont impactées par celle-ci et évitez les systèmes trop exigeants en interface. A l'usage, nous avons constaté une chauffe relativement importante de la carte. Le site pine64.pro rassemble une importante documentation et les images des systèmes. On dispose aussi des librairies GPIO dédiées à la carte, ainsi que plusieurs autres librairies. Un des avantages est la compatibilité avec les GPIO de la Pi. L'autre avantage est d'avoir un 2e groupe GPIO : le bus Euler. Euler apporte des ports supplémentaires. D'autre part, le site officiel propose de nombreux accessoires et modules supplémentaires : caméra, capteurs divers et variés, ports I2C, shield de développement, programmateur UART, Zwave, écran LCD. Et les tarifs proposés sont plutôt corrects. Vous pouvez utiliser des capteurs génériques.

## Banana Pro : pas mal du tout !

Banana Pi propose plusieurs cartes avec des spécifications différentes. La version Pro est un bon compromis et se compare facilement à la Pi 3. Elle possède le même processeur mais légèrement moins rapide et une puce graphique moins véloce. Cela se ressent immédiatement sur la fluidité des affichages graphiques (préférez un Linux sans interface). La possibilité d'utiliser un disque SATA est un plus, et les performances s'en ressentent, tout comme la présence d'un Ethernet 1 Gb. Par contre, les ports camera et display ne sont pas compatibles avec ceux de la Pi. Comme pour les autres boards, il faut utiliser des images système adaptées. Le choix est moins étendu que les concurrents. Pour assurer la compatibilité avec les broches de la Pi, on dispose de la librairie WiringPi et de RPi.GPIO. Des modules et capteurs dédiés à la Banana sont disponibles : température, caméra, shields d'extensions. Mais attention à la disponibilité en France et aux tarifs, souvent élevés.

## Bon ok, la Pi reste la référence, mais...

Oui, la Pi demeure la référence des cartes, surtout avec sa communauté très active. Elle est facile à trouver, les systèmes disponibles sont nombreux, sauf sur les versions 64 bits. Sur les autres cartes, l'offre système est plus réduite et pas toujours à jour. Mais l'Odroid-C2 fait parfaitement tourner un Ubuntu Mate 64, ce qu'une Banana Pro aura bien du mal à faire à cause de sa GPU, et un CPU un peu ancien. La C2 tire profit des meilleures ressources (mémoire, CPU, GPU). La Pine64 est en concurrence frontale avec un tarif attrayant ce que l'ODROID n'a pas, pour le moment. ⊞

# De l'importance de la formation dans la vie d'un développeur

*La carrière d'un développeur est rarement un long fleuve tranquille. La technologie évoluant rapidement ainsi que les outils, les méthodes et les langages, le développeur doit très régulièrement se (re)mettre à niveau, suivre les évolutions de son environnement de programmation mais aussi avoir une veille technologie constante. La formation est un élément important à ne surtout pas négliger. Que vous soyez en entreprise, en SSII ou indépendant, il ne faut jamais hésiter à se former pour apprendre de nouvelles choses ou tout simplement pour approfondir son environnement. Gregory Serfaty revient pour nous sur sa carrière et pourquoi la formation l'a aidé à évoluer dans son job et dans ses compétences.*

Actuellement Lead dev PHP dans une agence web, GoProd, j'ai un parcours de dev assez atypique. En effet ayant fait des études de math/info il y a environ 20 ans, et après une mauvaise expérience professionnelle, j'ai quitté le monde de l'informatique pour d'autres horizons. Après avoir pas mal baroudé, j'ai voulu stabiliser ma vie et revenir à mes premiers amours: le développement.

Je me suis orienté vers PHP car c'était le langage le plus proche de ma formation de développeur C. Je me suis donc retourné vers tous les moyens possibles pour me former (tuto, internet, et centre de formation) car il faut dire que ce n'est pas évident en France de se reconvertir après 30 ans. J'ai eu la chance de tomber sur une boite de formation Anaska, et d'être remis à jour par Julien Pauli (bien connu dans la communauté PHP). Après quelques expériences en freelance et grâce à cette formation, j'ai pu décrocher un emploi dans une startup en tant que dev PHP. Je me suis beaucoup formé ensuite sur internet et avec quelques livres car seul développeur dans ma société ce n'était pas évident. Lors de la liquidation de celle-ci j'ai eu l'opportunité de m'orienter vers du DevOps, pour cela j'ai fait appelle à Openska. J'ai choisi cette boite de formation car Cyril (Pierre de Geyer) m'avait déjà convaincu avec Anaska. Je savais que les formateurs étaient de bonne qualité, très pro et j'aimais cette approche mains dans le cambouis (beaucoup de TP). Je n'ai pas été déçu, des formateurs de grande écoute et qui connaissent leur sujet.

## Vers le framework

Mon expérience de DevOps ne m'ayant pas donné entière satisfaction, je suis resté dans le dev en voulant m'orienter vers du Symfony, pour cela toujours grâce à Openska et à Sensio j'ai pu me former sur ce framework.

Lors de mes différents entretiens, le fait de mentionner cette formation m'a permis de palier mon manque d'expérience sur le framework. Une fois encore, j'ai eu la chance d'être entre de bonnes mains (merci Hugo), cette formation a eu pour effet de combler mes lacunes et de compléter mes connaissances apprises sur différents sites tel que devandclick.fr

## Rester à jour

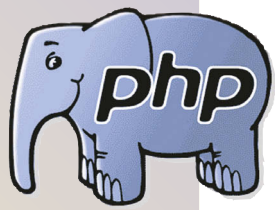
Après toutes ces années, je peux dire que le métier de développeur a bien évolué. Au début de mon expérience la connaissance simple de php suffisait pour être dev. Aujourd'hui plusieurs métiers sont sortis (integrateur, dev front, de backend, DevOps...). La connaissance de frameworks aussi et devenue une obligation. De même pour le versionning, la gestion serveur, docker, etc. Nous devons en permanence nous maintenir à jour, pour cela Internet reste un bon moyen, ainsi que les collègues et les formations. Dans mon cas, la formation m'a permis de comprendre des technos plus rapidement, la mise en situation est une chose qui marche beaucoup pour moi. J'ai besoin de pratiquer pour que cela rentre dans ma tête. Donc l'interaction avec un formateur est plus efficace que de lire un livre ou la documentation.

## Ne pas négliger la préparation

En revanche, je pense qu'il faut connaître un peu le sujet avant d'entreprendre une formation, sinon on peut être vite largué. Trop d'infos à emmagasiner en (très) peu de temps. Je conseillerais de préparer sa formation si on veut qu'elle soit efficace. J'ai vu des personnes faire des formations sur des frameworks sans comprendre l'objet. Je dirais donc que l'on ne devient pas expert en 5 jours, cela se saurait depuis longtemps. La formation nous fait gagner un peu de temps et surtout nous guide. Ensuite rien ne vaut la pratique. ⊞

## PHP : un marché toujours porteur

PHP demeure, en France, un marché actif et dynamique, aussi bien sur les nouveaux projets que les projets en maintenance. Mais attention, comme l'a précisé Gregory Serfaty, le langage seul ne suffit plus. Vous devez maîtriser un ou plusieurs frameworks et outils. Le marché est segmenté en 4 grands outils / CMS : Drupal, WordPress, Symfony et Zend Framework. Cette spécialisation est indispensable et faites-le dès le départ.




Les offres d'emplois proposent généralement 3 types de recrutements :

- Développeur / intégreur Web ou fullstack Web avec compétences PHP ;
- Développeur PHP sans spécialisation particulière ;
- Développeur spécialisé sur un framework / CMS.

Côté salaire, rien de neuf par rapport aux dernières études publiées et notamment celle de l'AFUP. Pour un débutant, le salaire moyen est de 30 000 €, pour un profil expert avec 10 ans d'expérience, + 50 000 €. Bien entendu, les écarts sont très grands pour un même profil et la région. Le profil expert, et particulièrement avec la maîtrise d'un framework spécifique, peut vite atteindre 55 – 60 000 €. Un profil débutant se retrouvera régulièrement à – 30 000 €.

Bref, formez-vous aux nouveaux outils et aux technologies du marché. Vous pourrez ainsi sortir du lot. Soyez actif (ve) dans les communautés. Votre activité communautaire pourra être un plus positif lors d'une embauche. Vous pouvez aussi tenter votre chance en indépendant.

# CheckMyHTTPS : un module de détection d'Interception de flux WEB chiffrés

1ère partie

*L'idée originale de « CheckMyHTTPS » est née dans les locaux de l'ESIEA (École d'ingénieurs en sciences et technologies du numérique) au sein du laboratoire de recherche CNS/(C+V)° (Cryptologie et Virologie Opérationnelles). Il y a maintenant un an, nous étions alors en troisième année de notre cursus de 5 ans, nous avons eu une discussion avec Rexy, un enseignant chercheur du laboratoire. Il nous a proposé d'étudier techniquement les différents risques qu'un utilisateur pouvait rencontrer quand il consulte des sites WEB via le protocole HTTPS. Il nous a aussi demandé d'étudier une technique qui permettrait aux usagers de se protéger contre ces risques. Nous étions déjà conscients que « SSL » (Secure Socket Layer), la couche de sécurité de HTTPS, était attaquée de toutes parts même dans sa version la plus récente rebaptisée « TLS » (Transport Layer Security). Après avoir inventorié les scénarios d'attaque sur ce protocole, nous nous sommes alors interrogés sur la manière de les détecter facilement.*

Raphaël PION & Hugo Meziani
**Project leader : Rexy**

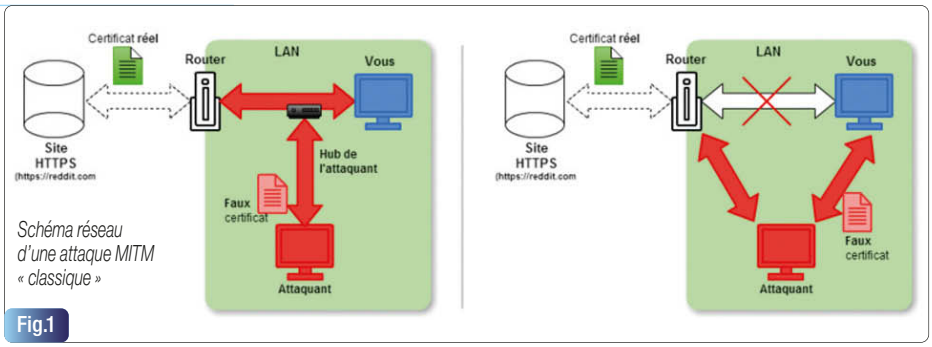



*Schéma réseau d'une attaque MITM « classique »*

Fig.1

Nous nous sommes donc penchés sur les techniques d'interception d'une connexion par un tiers puis au déchiffrement du flux capturé (attaque souvent appelée : « Man-In-The-Middle SSL » ou « MiTM »). Après avoir expérimenté ces techniques, nous avons décidé de développer un détecteur d'interception. Un premier démonstrateur nous a permis de valider le concept. Les évolutions apportées à ce premier démonstrateur ont permis la création d'un véritable outil de détection capable de mettre en évidence les trois scénarios d'interception de flux HTTPS que nous avons jugé les plus réalistes aujourd'hui.

## Premier scénario : le pirate informatique

Les techniques d'interception de flux WEB chiffrés sont aujourd'hui matures, disponibles et de plus en plus simples à exploiter. Ainsi, un pirate découvrant un réseau local WIFI ou filaire non sécurisé, qu'il soit domestique ou d'entreprise, peut assez facilement exploiter ces techniques pour intercepter, analyser ou modifier des flux qu'ils soient sécurisés ou non. Avec le temps, plusieurs techniques de sécurisation des réseaux ont été implémentées par les constructeurs : chiffrement « WPA2 », 802.1X, protection contre l'usurpation d'adresse MAC, « client isolation », « anti-rebond », etc.). Cela rend de plus en plus difficile l'écoute par des techniques dites d'homme du milieu (« ManInTheMiddle - MITM »). Fig.1.

Dans ces conditions, nous avons préféré un scénario plus réaliste aujourd'hui qui consiste à présenter un faux point d'accès WIFI intégrant un outil d'interception SSL/TLS (type « sslsplit »).

Dans la nature et pour être plus efficace, ces points d'accès « pirates » diffusent un nom de réseau susceptible d'attirer naturellement les utilisateurs en fonction de l'endroit ou des circonstances (« Free-SNCF », « WIFI-Hôtel-esperance », « Palais-des-congres-gratuit », etc.). Pour être encore plus efficace, le pirate exploitera préalablement une « fonctionnalité » des ordinateurs portables, tablettes ou smartphones.

En effet, ceux-ci tentent de se reconnecter automatiquement aux points d'accès WiFi sur lesquels ils ont déjà réussi et validé une connexion...

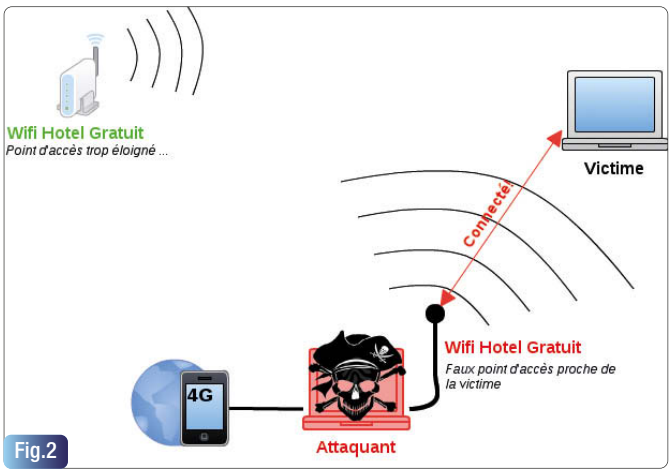



Fig.2

*Faux point d'accès WiFi créé par le pirate*

En utilisant cette particularité, le pirate peut créer un point d'accès WIFI possédant des noms de réseau connus de l'équipement de la victime. Fig.2.

Une fois l'utilisateur connecté au faux point d'accès, l'objectif pour le pirate est de se faire passer pour le serveur ou le site Internet consulté en présentant au navigateur de l'utilisateur un faux certificat de sécurité (certificat possédant le bon nom de site, mais signé par une autorité de certification forcément inconnue des navigateurs WEB).

L'attaquant impose sur cette connexion son propre certificat SSL qui lui

permet de déchiffrer le trafic de sa victime. Le navigateur de celle-ci lance alors une alerte de certificat invalide. Fig.3.

La victime peut donc refuser ce certificat et stopper la connexion. Le plus souvent, face à une urgence de situation (besoin absolu de consulter ses courriels ou un site particulier), ou par simple méconnaissance, elle va accepter ce certificat et rendre ses informations sensibles disponibles pour un tiers. L'extension CheckMyHTTPS permet de mettre en évidence ce type d'attaque en alertant l'utilisateur avec un cadenas de couleur rouge et une notification. Fig.4.

Face à cette menace, les navigateurs ont intégré une protection complémentaire appelée « HSTS » (HTTP Strict Transport Security). Lors de la première connexion sur un site en HTTPS, le navigateur va enregistrer le certificat que le serveur lui envoie. Lors de connexions ultérieures sur ce site, toutes modifications relatives à ce certificat engendreront une alerte et la connexion sera interrompue.


Fig.3

*Alerte de Firefox quand il ne peut valider le certificat présenté*

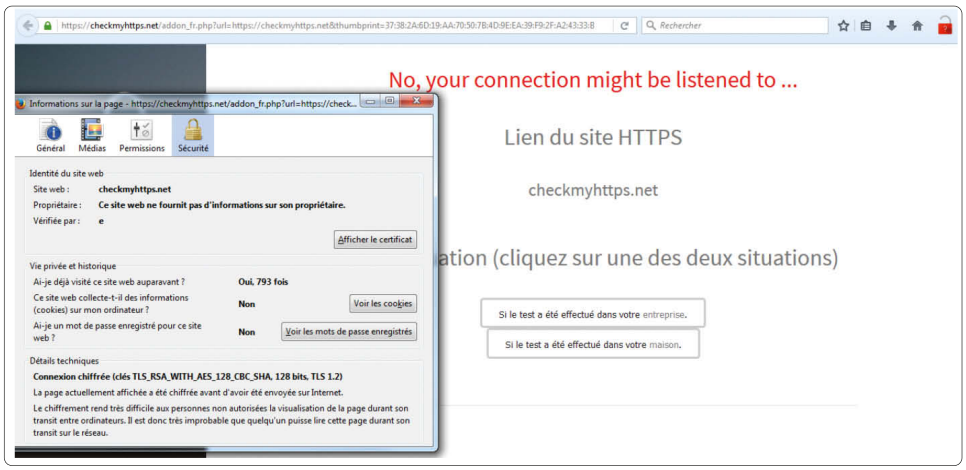



Fig.4 *MITM détecté par CheckMyHTTPS (cadenas rouge en haut à droite)*

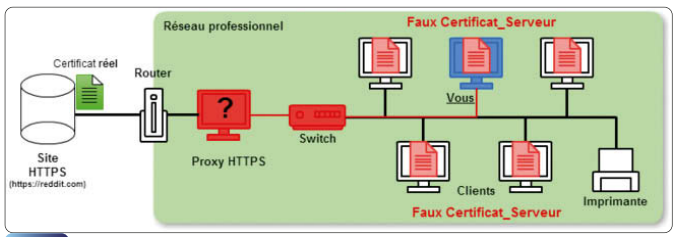
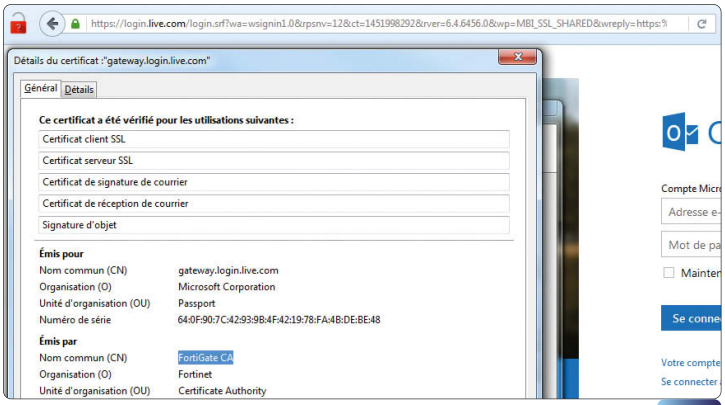



Fig.5 *SSL Inspection activée sur « proxy SSL » installé dans un réseau d'entreprise*




*CheckMyHTTPS détecte que le certificat serveur du site consulté (Microsoft) a été échangé par celui du parefeu de l'entreprise (Fortinet). (cadenas rouge en haut à gauche)*

Fig.6

## Deuxième scénario : les passerelles de sécurité d'entreprise (appliances de sécurité)

Depuis quelques années déjà, les équipements de sécurité déployés dans les entreprises en frontière d'Internet (pare-feu, système de prévention d'intrusion, etc.) intègrent une fonction permettant d'intercepter, de déchiffrer et d'analyser les flux chiffrés par SSL/TLS (protocoles HTTPS, POP3S, SMTPS, xxxxS). Cette fonction est souvent nommée « SSL inspection ». Ces équipements sont intercalés entre le Réseau Local d'Entreprise (RLE) et Internet afin de contrôler et filtrer les connexions entrantes et sortantes (fonction de « proxy). Cela permet à l'entreprise de se protéger à la fois d'éventuelles attaques et de lutter contre la fuite d'information.

Cette fonction est parfois activée par défaut ; surtout quand l'équipement est de conception étrangère où les règles liées au respect de la vie privée ne sont pas aussi abouties qu'en France. Dans ces conditions, tous les flux sécurisés sortants de l'entreprise sont déchiffrés par cet équipement sans que les utilisateurs (ni même parfois l'administrateur) en soient informés.

En France, cette fonctionnalité peut être activée sous condition d'informer les utilisateurs. Ils pourront ainsi modifier ou non leur habitude de surf en conscience. Cela se traduit généralement par une rubrique spéciale insérée au sein d'une charte informatique signée par l'ensemble des collaborateurs de l'entreprise.

Pour parfaire cette interception SSL, l'administrateur peut bien entendu diffuser aux navigateurs WEB de son réseau (via une GPO ou via son outil de gestion de parc) le certificat de l'autorité de certification exploité par le module d'inspection SSL. De cette manière, les navigateurs WEB n'émettront plus d'alertes de sécurité quand ils recevront le « faux » certificat. Fig.5.

À titre d'exemple, nous avons pu tester cette fonctionnalité sur un pare-feu Fortigate 240D. Dans cet équipement, la fonctionnalité « inspection SSL » est présente sous forme d'un module activable d'un simple clic et qui déclenche l'interception de toutes les connexions sécurisées transitant entre Internet et le RLE. L'extension « CheckMyHTTPS » a permis de mettre en évidence l'interception en alertant l'utilisateur avec un cadenas de couleur rouge et une notification. Fig.6.

## Troisième scénario : les logiciels antivirus

Par défaut, des antivirus tels que « Avast ! » ou « Kaspersky » interceptent et déchiffrent les connexions sécurisées de type HTTPS. Les éditeurs de ces antivirus avancent vouloir protéger l'utilisateur du contenu malveillant pouvant être trouvé sur Internet. Cela permet cependant à leurs logiciels d'être capables d'analyser les connexions chiffrées. Sans être exhaustifs, nous avons identifié au moins quatre antivirus pratiquant ces interceptions.

| Logiciel Antivirus | SSL Inspection Par défaut ? | |
|---|---|---|
| Avast ! | OUI | OUI |
| Kaspersky | OUI | OUI (sur certains sites) |
| BitDefender | OUI | NON |
| ESET | OUI | NON |

*Tableau 1 : Inspection SSL activée par défaut sur les antivirus du marché*

L'extension « CheckmyHTTPS » permet de mettre en évidence ce type d'attaque en alertant l'utilisateur avec un cadenas de couleur rouge et une notification.

***Suite dans le numéro 200.***

# Les développeurs révolutionnent la santé

*Nous avons beaucoup hésité sur l'intitulé de ce dossier, car est-ce le développeur ou la technologie au sens large qui révolutionne le monde de la santé. Un peu des deux, car les fournisseurs de technologies sont très actifs depuis quelques années sur ce marché de la santé 2.0 ou de la e-Santé.*
*Mais comme toujours, une technologie sans usages ni applications ne sert à rien, et là, le développeur prend sa place.*


source : HASLOO

De plus en plus, les éditeurs technologiques font contribuer les médecins et les hôpitaux pour éprouver, compléter, tester les outils, les SDK, les projets qui seront mis ensuite sur le marché. Pour ne citer qu'eux, Apple, Google, Microsoft, IBM investissent des centaines de millions dans ce marché.

À quoi ressemble ce marché ? Une des difficultés est de définir de quoi on parle. Les termes santé et médical sont très précis et les législations des pays encadrent les pratiques et les technologies (logicielles et matérielles) doivent rentrer dans ces critères. Ce n'est donc pas un hasard si les constructeurs de traqueurs d'activité ou de logiciels dédiés ne parlent pas de santé ou d'outil médical, mais plutôt de bien-être, de sport, pour éviter toute polémique et fausse appréciation des utilisateurs.

Au-delà des logiciels divers et variés (pour le cabinet médical, pour le courrier du médecin, etc.), marché très convenu avec peu de surprises, des révolutions majeures arrivent déjà. Nous le voyons parfaitement depuis plusieurs années dans les milieux hospitaliers avec la télémédecine, les robots qui prennent un peu plus de place chaque année. Durant le salon Viva Technology de cet été, nous avons pu voir de vraies solutions innovantes et ce n'est qu'un début. L'impression 3D, encore très discrète en France, explose dans d'autres pays : prothèses diverses (notamment dentaires), impression d'une partie du crâne, etc. Ce domaine va croître très rapidement, car l'impression 3D permettra de produire des prothèses plus rapidement, avec une précision très grande et pour un coût qui va décroître (espérons-le). Nous le constatons pour les prothèses de mains pour les pays pauvres.

Dans le marché de la e-Santé nous pouvons citer la médecine/santé personnalisée, le big data (par exemple les grandes études sur des maladies nécessitant des milliers de malades pour récolter et analyser les données), les nouveaux matériels (capteurs pour smartphones, IoT orientés préventions et santé, impression 3D), nouvelles expériences patients – médecins. Mais il est difficile de cerner ce marché, car finalement, la e-Santé, le bien-être en général, est un marché multiforme qui ne se limite pas à la seule notion de médecine ou de santé.

Nous voyons aussi comment les grands acteurs technologiques avancent prudemment, itération après itération. Ainsi, l'Apple Watch a déçu beaucoup d'attentes, car on s'attendait à une profusion de capteurs orientés santé, mais Apple, comme les autres, se heurte à la fiabilité, et à la précision des capteurs et des logiciels. Et les certifications et homologations ne s'obtiennent pas en quelques mois. Pour ces acteurs, il faut savoir être patient.

Selon StartUp Health, pour les premiers mois de 2016, 3,9 milliards $ ont été investis, en Amérique du Nord, dans le développement, les éditeurs et startups. Peut-on estimer le marché ? Pour la France, on parle de 3-4 milliards € d'ici 2020. Au niveau mondial, les rapports et analystes évoquent un marché de 400 milliards $ d'ici 2022. À voir si ces estimations se réaliseront.

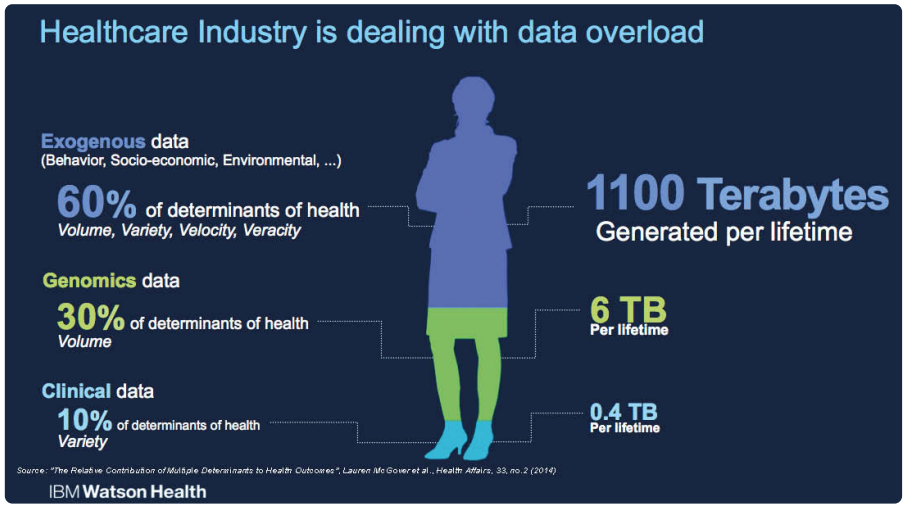
Si vous maîtrisez un peu le code, les technologies, l'électronique, vous pouvez relativement facilement concevoir votre IoT, votre objet de santé ou tout simplement une app mobile orientée santé, sport, etc. Le choix ne manque pas dans les cartes de prototypages et les capteurs. À vous de jouer !

La rédaction

# D'Hippocrate à Watson.

*Les développeurs auraient un grand rôle à jouer dans le développement de la e-santé. Analyse.*

Jean Michel Billaut
*Président fondateur de l'Atelier de BNP Paribas. À la retraite (et amputé d'une jambe).*
Jmbillaut@yahoo.fr




## Les technologies avancent vite, de plus en plus vite.

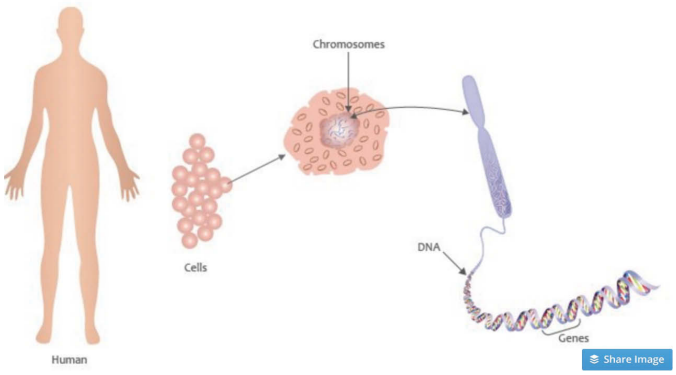
Dans le domaine de la santé, nous allons passer du monde du curatif (on attend d'être malade pour aller voir le médecin et/ou nous emmener à l'hôpital qui va essayer de nous réparer) à un monde du prédictif. À savoir : prévoir que vous pourrions avoir telle maladie, et y remédier avant que celle-ci ne se déclare. Obama parle aussi de « precision medecine ». Et certains de penser que la santé va être une affaire de « data », avec moins de chimie et de molécules à terme. Donc de numérique. Donc affaire de développeurs.

## Comment cela est-il possible ?

Plusieurs ingrédients sont nécessaires et sont en train de se mettre en place sous nos yeux :

1. Il faut des clouds puissants et gigantesques comme ceux d'Amazon, Google, Apple, Microsoft, IBM, Alibaba, Tencent, Baïdu, Qihoo 360, etc. Ils vont recueillir toutes les données des malades ET des bien portants ou supposés tels.
2. Quels types de données ? Des données génétiques (le séquençage du génome de l'Être humain – on commence à penser que cela va être la base de la médecine future), des données remontées par divers Internet des objets de santé (analyse de sang, qualité du sommeil, calcul du métabolisme, etc..). Et bientôt les données de spectromètres de poche qui vont analyser en temps réel les matières organiques qui sont dans nos assiettes et nos verres. La nourriture a en effet une grande importance pour notre métabolisme et notre santé. Ces clouds vont stocker toutes ces données pour des centaines milliers, des millions, d'Êtres humains. iCarbonX en Chine veut gérer la santé de 100 millions d'humains !
3. On va « cruncher » ces données dans tous les sens avec des algorithmes pointus de big data, de « machine learning ». Et cela de façon individuelle pour un individu donné, mais aussi en masse sur des échantillons plus ou moins importants de gens qui ont telle ou telle caractéristique. L'objectif est simple : essayer d'apporter les meilleurs remèdes personnalisés aux malades, et par la même occasion découvrir les mécanismes du déclenchement du cancer, du diabète, etc. Pour les éradiquer du genre humain. Rien que cela. Inutile de dire que celui qui trouve... D'où l'effervescence actuelle.

Voilà en gros le paysage tel qu'il se dessine actuellement.




## Qui fait quoi ?

Les pays les plus en pointe sont la Chine avec le Beijing Genomics Intitute (ou BGI), IcarbonX (valorisé 1 milliard de $, 9 mois après sa création), l'Angleterre avec l'opération 100 000 génomes, lancés il y a 3 ans par David Cameron. Opération qui vient d'être rejointe par l'Écosse, l'Irlande, l'Islande, et diverses startups (dont Illumina, un fabricant américains de séquenceurs professionnel). Et naturellement les USA, qui une fois le concept de « precision medecine » annoncé par Obama il y a un peu plus d'un an, se mettent maintenant au travail avec le « Moonshot Cancer » et le « Cohort Program » qui a l'objectif de séquencer un million de personnes volontaires malades ou non.

Comme on le voit avec ce bref descriptif, la « course à l'échalote » est lancée, à coup de millions et de milliards de dollars. Les VC américains ont financé environ 2000 startups depuis 2011, pour 10 milliards de dollars.

### Who's who

- Hippocrate : médecin grec du Vème siècle av. J.-C., considéré comme le père de la médecine dite moderne.
- Watson : le « machine learning » d'IBM qui est de plus en plus utilisé dans toutes les activités humaines, et notamment dans la santé et la médecine.

## Et la France ?

Comme vous le savez la France est toujours en retard dans ces affaires numériques à base de données. Nos élites sont peu intéressées par le digital. Cependant, le gouvernement a annoncé en juin dernier son « Programme Génomique 2025 », avec un retard de 3 à 4 ans sur ceux déjà partis. Un document de 170 pages, a été publié. Coût de notre programme : 600 millions d'€.

Attendons ce que cela va donner. Car si cela rate, nos milleniums feront suivre leur santé sur les grandes plateformes de e-santé qui commencent à se placer aux USA et en Chine. Selon un business model à la Netflix : 9 $/mois.

## Et vous, les développeurs ?

Vous avez certainement l'habitude de développer des applications pour le commerce et le marketing électroniques, les jeux vidéo, etc. Vous avez désormais de nouveaux débouchés dans la e-santé, qui à terme seront beaucoup plus importants que les secteurs investis jusqu'à pré-

| Calendrier | |
|---|---|
| 2016 | **Élaboration du cahier des charges des plateformes et sélection des sites**<br>**Mise en œuvre des trois premières plateformes**<br>Choix des technologies de séquence, définition de l'architecture de stockage et de calcul qui sera déployée (calculateur à mémoire partagée, cluster de calcul…), spécification des logiciels (achat des licences, conformité logicielle) |
| 2017 | **Déploiement des solutions matérielles et logicielles spécifiées sur les trois premières plateformes**<br>Installation des logiciels d'alignement et d'annotation automatique de variants (premier niveau d'analyse de données)<br>Installation des logiciels de collaboration avec le CAD<br>Envoi des premières données vers le CAD parallèlement mis en service<br>Certification de la plateforme en tant qu'hébergeur de données de santé<br>Démarrage de la production en lien avec le CAD |
| 2017-2020 | **Déploiement des neuf autres plateformes** |

sent dans le numérique. Soit dans des startups e-santé, soit dans les grands « machins » français 1.0 (mais je ne vous le conseille pas trop), soit chez les Gafa américains et/ou chinois.

## Quelques remarques

1. On va assister dans les quelques années à venir à la disruption des langues. Vous pourrez donc travailler pour des Chinois ou autres, sans apprendre leurs langues. En « temps réel » avec Skype translator, ili wearable translator, etc. C'est la magie du « machine learning ».
2. On va aussi assister à la mise en place d'une espèce de supranationalité. L'Estonie a démarré son programme de e-résidents. Le pays attend 10 millions de e-résidents d'ici 2025. D'autres pays vont suivre.
3. L'open source dans le domaine de la e-santé, n'est pas en reste, loin de là. Une startup française (Omictools.com) vient d'ouvrir un site recensant plus de 12 000 applications e-santé open source. Il y en aurait 50 000 dans le Monde. Et ce, dans tous les domaines : médecine personnelle, du family genome, du personal genome, etc.

Bref de nouvelles opportunités pour vous. Bonne chance dans le Futur.

Je vous conseille quelques lectures complémentaires :

Mes newsletters e-santé :
http://billaut.typepad.com/jm/e-sant%C3%A9/

L'élite française face au digital : http://billaut.typepad.com/jm/2013/07/la-grande-élite-française-nest-pas-adaptée-au-numérique-loin-sen-faut.html

Quelques sites intéressants :
La licorne chinoise e-santé iCarbonX :
http://www.icarbonx.com/en/
http://www.asianscientist.com/2016/07/print/meet-chinas-first-biotech-unicorn-wang-jun-icarbonx-bgi/3/

Le BGI chinois
https://fr.wikipedia.org/wiki/Beijing_Genomics_Institute

Qu'est-ce que le génome ?
https://fr.wikipedia.org/wiki/Génome

Le couteau suisse de la génétique
http://www.cite-sciences.fr/fr/ressources/science-actualites/detail/news/crispr-cas9-le-couteau-suisse-qui-revolutionne-la-genetique/?tx_news_pi1%5Bcontroller%5D=News&tx_news_pi1%5Baction%5D=detail&cHash=37ce7a688e0683ccfe5178db483d8b84

Google dans la e-santé :
https://www.theguardian.com/technology/2016/may/04/google-deepmind-access-healthcare-data-patients

IBM dans la e-santé :
http://ww2.kqed.org/futureofyou/2015/07/07/ibm-hopes-its-watson-supercomputer-can-improve-americas-health/?utm_content=buffer87e16&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer

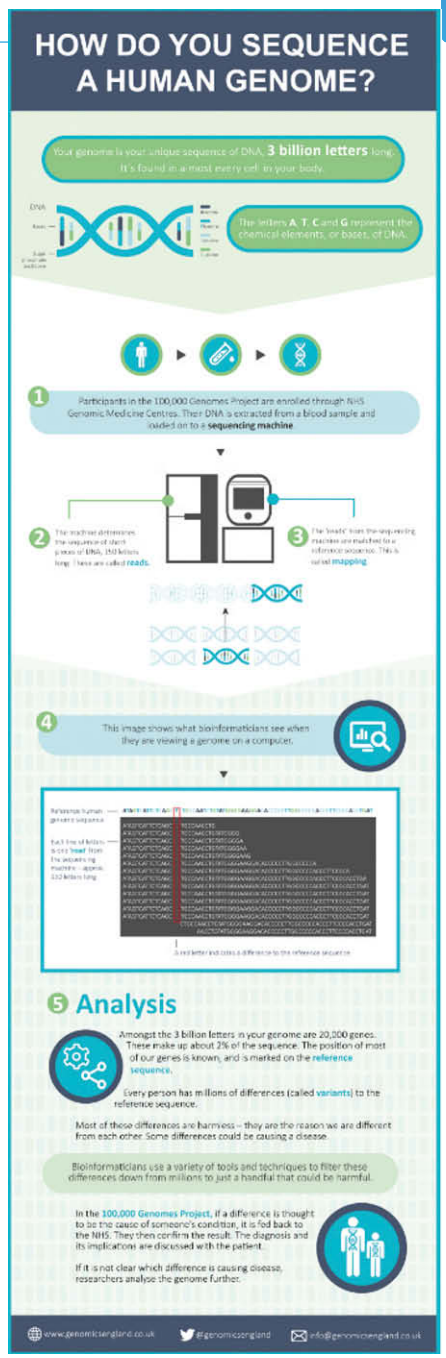
Apple dans la e-santé :
http://fieldguide.gizmodo.com/how-to-track-your-life-with-apple-health-1671669616

Cohort programm
http://www.lapresse.ca/sciences/medecine/201602/25/01-4954450-etats-unis-creation-dune-cohorte-de-recherche-medicale-dun-million-de-personnes.php

Moonshot cancer
https://www.statnews.com/2016/04/20/biden-moonshot-scientists/

100 000 génomes anglais
https://www.genomicsengland.co.uk/taking-part/the-process/

La blockchain en e-santé en Estonie
https://news.bitcoin.com/estonian-health-records-secured-by-blockchain/




Association Française e-santé
http://www.france-ehealthtech.org/

L'open source dans la e-santé
https://omictools.com/

Quel avenir pour les médecins ?
http://thehealthcareblog.com/blog/2012/08/31/vinod-khosla-technology-will-replace-80-percent-of-docs/

Devoxx : ma présentation de la e-santé
https://www.youtube.com/watch?v=8agk41AL1g4

La médecine va se faire uberiser à l'horizon 2025
http://www.frenchweb.fr/la-medecine-va-se-faire-uberiser-a-lhorizon-2025/222241

Pour ceux que cela intéresse. Vous pouvez suivre ma veille e-santé (mais pas que) sur mon flux twitter, https://twitter.com/billaut
et mon mur Facebook : facebook.com/jean.m.billaut

# La santé connectée et les apps

*Cet article a pour but de partager avec vous l'état de l'art de ce marché en vous décrivant les caractéristiques atypiques de celui-ci ainsi que ses opportunités. En effet, les objets et apps issues de la santé connectée prennent chaque jour de plus en plus de place dans nos vies. Vous pouvez donc vous dire qu'il est facile de mettre une nouvelle app sur store. Est-ce si simple de développer une nouvelle app pour la santé connectée ? Quelles sont les règles et normes à respecter avant de se lancer dans un nouveau projet ? Toutes ces questions se posent et nous tenterons d'y répondre dans ces quelques lignes.*

Kevin TRELOHAN
*Responsable des applications collaboratives au CHU de Nantes*
*Vice-Président des Communautés MS*
**Microsoft MVP SharePoint Server**

## Les normes et protocoles

### Normes

- EN1060-46
- directive 93/42/CEE
- 1999/5/CE dite "R&TTE
- EN60950-1:2006 + A11:2009 + A1:2010 + A12: 2011.

### Protocoles

- AAMI 1
- BHS 2
- DIN 3
- ESH 4 /HTA5

Comme vous le constatez, il existe beaucoup de normes, mais, est-ce que votre application doit respecter le cahier des exigences d'une d'entre elles ? L'ANSM a étudié tout récemment la question et voici sa conclusion : tout dépend du but de votre application. Si celle-ci n'a que pour principal objectif de traquer l'activité en comptant uniquement les pas et mouvements afin de faire mieux dormir vos utilisateurs en leur prodiguant des conseils alors, vous ne rentreriez pas dans le cadre complexe des normes et protocoles indiqués précédemment. Dans ce cas, votre app ne serait donc pas un « dispositif médical » et ne rentrerait donc pas dans le cadre des classes précitées également.

Pourquoi devriez-vous vous y intéresser ? Si vous utilisez le rythme cardiaque ou toute autre mesure physiologique de votre utilisateur alors, la question du respect des normes et classe se repose. Source : http://ansm.sante.fr/Activites/Mise-sur-le-marche-des-dispositifs-medicaux-et-dispositifs-medicaux-de-diagnostic-in-vitro-DM-DMIA-DMDIV/Logiciels-et-applications-mobiles-en-sante/(offset)/1#paragraph_78807

Comment vous renseigner ? Le mieux est de prendre contact avec l'ANSM. Si votre application ne rentre pas dans cette définition alors vous devrez vous conformer aux normes pour les dispositifs médicaux et donc l'ANSM vous conseille de suivre son guide de recommandations issues du dernier rapport d'étude publié le 13 juillet. Source : http://ansm.sante.fr/content/download/90293/1134423/version/2/file/Rapport+logiciels+-+juillet-2016-serma-ANSM.pdf

## Le vocabulaire afin de ne pas vous y perdre

### Santé

La santé est un état de complet bien-être physique, mental et social, et ne consiste pas seulement en une absence de maladie ou d'infirmité. Source : http://www.qualitiso.com/esante-quantified-self-msante-telemedecine-definition

### e-sante

e-santé et la m-santé = santé physique
Combinaison de produits et de services qui capturent, enregistrent et affichent des données et des informations, par voie électronique. Source : http://www.qualitiso.com/esante-quantified-self-msante-telemedecine-definition

### m-santé

Pratiques médicales et de santé publique supportées par des appareils mobiles, tels que les téléphones mobiles, les dispositifs de surveillance des patients, les PDA et autres appareils sans fil. Source : http://www.qualitiso.com/esante-quantified-self-msante-telemedecine-definition

### quantified self

Renvoie à un ensemble de pratiques variées qui ont toutes pour point commun, de mesurer et de comparer avec d'autres personnes des variables relatives à son mode de vie.

### Auto-mesure

Renvoie à un ensemble de pratiques variées qui ont toutes pour point commun, de mesurer des variables relatives à son mode de vie. Contrairement au « QUANTIFIED SELF » ces mesures ne sont pas partagées avec d'autres personnes.

### télé-médecine

Pratique médicale à distance utilisant les TIC entre professionnels et patients. On peut également pratiquer l'auto-mesure dans cette pratique et transmettre les données aux praticiens
Source : http://www.qualitiso.com/esante-quantified-self-msante-telemedecine-definition

### Dispositif médical

C'est un instrument, appareil, équipement ou encore un logiciel destiné, par son fabricant, à être utilisé chez l'homme à des fins, notamment, de diagnostic, de prévention, de contrôle, de traitement, d'atténuation d'une maladie ou d'une blessure. (directive 93/42/CEE relative aux dispositifs médicaux). Les dispositifs médicaux sont répertoriés en quatre classes.
Source : http://ansm.sante.fr/Produits-de-sante/Dispositifs-medicaux

Classe I : les fauteuils roulants, les bandes de contention, les scalpels...
Classe IIa : les lentilles de contact, les agrafes cutanées, les couronnes dentaires, les appareils d'aide auditive, des dispositifs de conservation de tissus ou de cellules à long terme, les échographes,...
Classe IIb : les hémodialyseurs, les pompes à perfusion, les préservatifs, les sutures internes, les systèmes de radiothérapie,...
Classe III : stent coronaire actif, prothèse de hanche,...

Source : http://www.aviesan.fr/fr/aviesan/accueil/menu-header/vademecum-reglementaire/dispositif-medical-definition

## Responsabilités et risques pour vous développeurs

La mise sur le marché des dispositifs médicaux est réalisée sous la responsabilité de leur fabricant qu'il soit Français, Européen ou étranger après qu'il y ait apposé le marquage CE.
Pour en savoir plus sur le marquage CE : http://www.economie.gouv.fr/dgccrf/Publications/Vie-pratique/Fiches-pratiques/Le-marquage-CE
Source : http://ansm.sante.fr/Produits-de-sante/Dispositifs-medicaux
Que cela soit pour un matériel ou un logiciel, vous êtes responsable des désagréments qui pourraient résulter de l'utilisation de votre app et

des conseils que vous prodiguez à vos utilisateurs.

**Attention : l'ANSM ne vérifie pas votre app à sa mise sur le marché.**

En effet, l'ANSM ne peut pas contrôler les apps mises sur les stores. Les tests s'il devait y en avoir seraient sans doute les suivants :

- Vérifier la liste des fonctionnalités ;
- Vérifier que les fonctionnalités sont classées dans la bonne classe et que l'app est conforme à la norme afférente ;
- Vérifier la qualité des données recueillies ;
- Vérifier la sécurité de l'échange de données entre le smartphone, l'objet connecté et le serveur de données ;
- Si l'app est associée à un objet de votre fabrication, vérifier que l'objet est hypoallergénique et non invasif et ne puisse pas perturber son utilisateur avec l'émission de fréquences ou ondes non recommandée pour l'être humain.

Donc c'est à vous de vous autocontrôler et prouver que vous respectez les exigences des normes et à minima celles du marquage CE en plus des normes et classes citées ci-dessus.
Voici quelques exemples :

- http://bewell-connect.net/files/products_conformity/DECLARATION-CONFORMITE-BW-BA1-MYTENSIO.pdf

- http://bewell-connect.net/files/products_conformity/DECLARATION_CONFORMITE_BW-F17_MYCOACH.pdf

- http://bewell-connect.net/files/products_conformity/DECLARATION_CONFORMITE_BW-OX1_MYOXY.pdf

## Le choix du matériel

Maintenant, en connaissance de cause, vous pouvez également vous demander si vous allez utiliser uniquement le gps ainsi que le capteur de mouvements intégré au smartphone ou vous associer à un objet connecté existant ou bien encore en créer un nouveau de toutes pièces.
Dans ce dernier cas, sachez que vous pouvez également faire fabriquer votre propre traqueur ou objet de santé connecté. Les fabricants sont souvent chinois on les trouve sur ce site :
https://french.alibaba.com/product-detail/unique-weight-management-bluetooth-body-fat-measuring-high-quality-digital-scale-60512697903.html?spm=a2700.7787047.0.0.2H0HmZ

Dans le premier cas, sachez qu'il y a des différences notables en termes de précision des mesures et collectes de données.
Il existe de très nombreux capteurs orientés santé, sport et bien-être. Tous ne se valent pas mais on un but commun. Mesurer et enregistrer une donnée pertinente et fiable.

## Le choix des API de développement

Certains constructeurs, vous offrent la possibilité de consommer des données directement auprès d'eux en utilisant leurs apis.
Voici une liste non exhaustive de ces apis :
https://dev.fitbit.com/docs/
http://developer.garmin.com/garmin-connect-api/overview/
http://developer.ihealthlabs.com/
https://developer.microsoft.com/en-us/windows/featured/lumia
http://developer.apple.com/healthkit/
https://developers.google.com/health/
Google a mis fin à son programme en 2011. Il a été remplacé par : https://developer.android.com/training/building-wearables.html

## La sécurisation des données passage obligatoire

En France il existe en plus des normes, des réglementations additionnelles à respecter. Ainsi, la CNIL vous oblige à faire une déclaration concernant les données que vous pourriez conserver sur vos utilisateurs :
https://www.cnil.fr/fr/comprendre-vos-obligations

- La déclaration du fichier (base de données) contenant les informations sur vos utilisateurs : https://www.cnil.fr/fr/declarer-un-fichier
- Si vous stockez ces données à l'étranger voici la déclaration additionnelle : https://www.cnil.fr/fr/transferer-des-donnees-hors-de-lue

Vous devez également en suivant ces recommandations, mettre en ligne votre politique concernant l'utilisation de ces données collectées. Vous pouvez prendre exemple sur ces fabricants :
http://www.terraillon.fr/fr/mentions-l-gales
https://www.fitbit.com/fr/privacy
http://www.garmin.com/fr-FR/legal/privacy-statement

Si votre équipement est un dispositif médical : Ne pas oublier les préconisations de la HAS http://www.has-sante.fr/portail/upload/docs/application/pdf/2009-12/guide_pratique_dm.pdf

## Lancez-vous !

Félicitations, vous avez choisi de vous lancer ! Pour bien commencer, sachez que dans le monde médical en France, les traqueurs et autres objets connectés de santé sont pour l'instant toujours considérés comme des « gadgets ». Donc ce marché de la santé est encore un marché de niche sur lequel vous pouvez bien entendu vous engouffrer mais, sans garantie de succès.
Comme évoqué plus haut avec les objets et apps connectées, vous devez définir leur classe d'appartenance avant de coder. Pour savoir si votre app ou objet connecté est de telle ou telle autre classe, utiliser le questionnaire en ligne suivant : http://www.eqms.io/#!/online-tools/medical-devices/93-42-EEC-classification. Par exemple si votre app permet d'afficher en temps réel le rythme cardiaque de son utilisateur, alors celle-ci peut être classée soit en IIa soit en IIb. C'est pourquoi il est important de bien maitriser l'ensemble de toutes ces exigences européennes et françaises afin que votre application ne soit pas dans l'illégalité réglementaire lors de sa publication sur un store.

Aussi, vous vous focaliserez sur le marché dit du grand public, qui, lui, comprend dès maintenant l'intérêt de s'auto-mesurer. Vous avez ensuite à faire votre choix concernant la plateforme pour le déploiement de votre App.

Sachez qu'à ce jour, les app store d'Apple et de Google sont les plus rentables. Cependant, n'oubliez pas non plus le store Microsoft qui avec son nouveau modèle d'UWP d'app vous permet d'accéder à plusieurs centaines de millions de terminaux dans le monde. Le conseil est donc de trouver un outil de développement multi-plateformes ainsi vous limiterez vos efforts.

Maintenant que vous avez choisi vos outils, il ne vous reste plus qu'à trouver l'idée qui va révolutionner la vie de vos utilisateurs. Un grand nombre d'applications existe déjà sur les différents stores, c'est à vous d'étudier le marché existant et de vous différencier.

Ainsi, par exemple beaucoup d'applications enregistrent les activités et le sommeil mais peu d'entre elles, fournissent des analyses intéressantes et pertinentes pour l'utilisateur. Enfin, encore moins fournissent de manière proactive des conseils à appliquer.
Enfin, à ce jour aucune ne vérifie l'application de ces conseils prodigués.

Mon conseil pour vous lancer en toute sérénité est de vous faire aider par des professionnels, pour cela, entourez-vous d'une équipe pluridisciplinaire et notamment appartenant au corps médical.
Enfin considérant que le champ des normes et classes est assez complexe, je vous conseille de prendre contact directement avec l'ANSM pour décrire votre projet et les questions que vous pourriez vous poser.

# API de Microsoft Health

*Microsoft Health Cloud API permet aux développeurs d'améliorer leurs applications et services avec des données provenant de Microsoft Health. Un an après la sortie de la* Technical Preview *en juillet 2015, la version 1 vient d'avoir une nouvelle mise à jour ouvrant encore les vannes.*

Christophe Peugnet
*MVP Windows Development chez* **SodeaSoft**
EBLM http://www.sodeasoft.com
Blog : http://www.peug.net - Twitter : @tossnet1

Nous allons voir comment on peut créer des applications accédant aux données de Microsoft Health stockées dans le Cloud via leur API.

## Mais avant tout qu'est-ce que Microsoft Health ?

MS Health est un service qui collecte et analyse les données de santé et de fitness provenant par exemple du bracelet Microsoft Band ou les smartphones compatibles avec les « Données de Mouvement ». Ça fournit un aperçu intéressant comme la qualité du sommeil pour la version 2 de la Band mais aussi un suivi des différents entraînements physiques, de l'estimation des calories brûlées et bien plus encore.
Microsoft Health est une multiplateforme qui peut être accessible de n'importe quel appareil Windows, Android, iOS ou d'un navigateur Internet.
Par exemple la capture ci-contre correspond à mes données depuis l'application Windows 10 : Fig.1. Ou par exemple depuis le tableau de bord de Microsoft Health sur le site web https://dashboard.microsofthealth.com qui peut, lui, brasser des données que le développeur ne pourra pas lui-même interroger comme par exemple comparer mes données avec la moyenne d'autres personnes : Fig.2.
Ce qui nous intéresse maintenant c'est qu'un développeur tiers peut aussi accéder à ses données et proposer une nouvelle application grâce aux API fournies. Les données retournées seront au format JSON ce qui ne posera aucun problème au développeur. La connexion sera sécurisée par une traditionnelle authentification OAuth 2.0 via le compte Microsoft. Une prise en charge CORS (Cross-Origin Resource Sharing) est aussi disponible pour ceux qui souhaitent par exemple intégrer facilement une application Web dans un site.
Il existe pour l'instant 4 points d'entrées (EndPoint) en lecture seule.

- **Profile :** qui donne le nom de l'utilisateur, son poids et sa taille ;
- **Devices :** liste tous les appareils connectés à Microsoft Health que l'utilisateur possède ;
- **Summaries :** retourne l'ensemble des données sur le nombre de pas effectués, les calories brûlées, la distance parcourue, le rythme cardiaque, les heures d'activités sur une période donnée ;
- **Activities** : renvoie des données sur les activités particulières comme les exercices, le vélo, la randonnée, les parcours de Golf, mais aussi les données sur le suivi de votre sommeil etc.

La dernière mise à jour de juin apporte la possibilité de récupérer les données des randonnées, ajoute plus d'informations liées aux activités telles que les noms de celles-ci, les pauses prises... On peut aussi récupérer le nombre de marches d'escaliers montées et ajouter plus de données pour les golfeurs sur leurs parcours.
On va retrouver ces API très bien documentées sur le site http://developer.microsoftband.com/cloudAPI/Explorer - Fig.3.
Pour créer votre application, rendez-vous sur le site

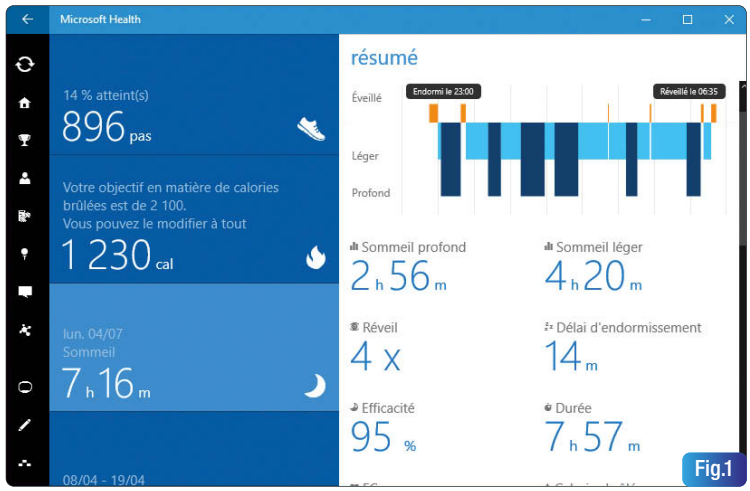


Fig.1

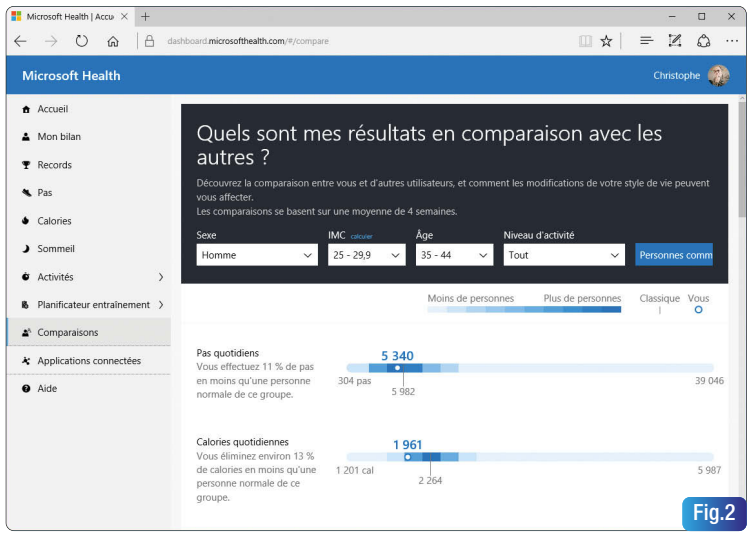


Fig.2

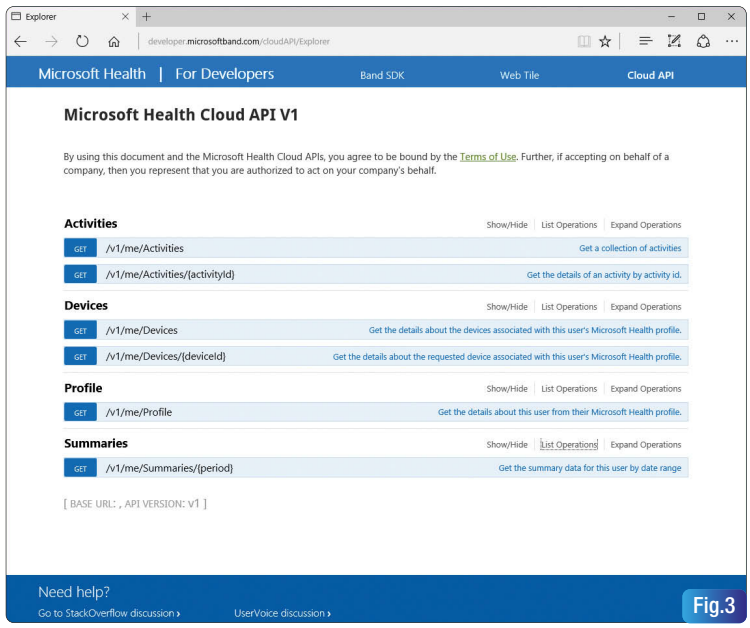


Fig.3

https://app.dev.microsoft.com sur lequel vous récupérez votre IDClient et votre clé secrète.
Comme dit plus haut, vous devrez obligatoirement vous connecter avec un compte Microsoft pour récupérer vos données et il faudra alors préciser quelles données vous allez prendre (scopes).
Voici la liste des possible : mshealth.ReadDevices, mshealth.ReadActivityHistory, mshealth.ReadActivityLocation, mshealth.ReadProfile. Ainsi Microsoft demandera à l'utilisateur s'il accepte : Fig.4.
A partir de là, le plus difficile est fait car ensuite pour par exemple récupérer les données « **Profile** », il suffira en C# (*par exemple*) d'interroger l'API comme ceci :

```
var uriBuilder = new UriBuilder("https://api.microsofthealth.net/");
uriBuilder.Path += "v1/me/Profile";

var request = HttpWebRequest.Create(uriBuilder.Uri);

request.Headers[HttpRequestHeader.Authorization] = string.Format("bearer {0}", this._LiveId
Credentials.AccessToken);

using (var response = await request.GetResponseAsync())
{
  using (var stream = response.GetResponseStream())
  {
    using (var reader = new StreamReader(stream))
    {
      this.Zone.Text = reader.ReadToEnd();
    }
  }
}
```

Fig.4

Astuces : depuis Visual Studio 2015 vous pouvez demander nativement à Visual Studio de créer les classes d'un résultat JSON depuis le menu EDIT : Fig.5. Certains *EndPoint* nécessiteront de passer des arguments complémentaires comme **Summaries**. Avec l'Explorateur d'API, nous avons un très bon descriptif : Fig.6.
Si on reprend le code précédent, on utilisera cette API comme ceci pour retourner l'ensemble de mes pas quotidien :

```
var uriBuilder = new UriBuilder("https://api.microsofthealth.net/");
uriBuilder.Path += "v1/summaries/Daily";
uriBuilder.Query = string.Format("startTime={0}", DateTime.Now.AddYears(-1).ToUniversal
Time().ToString("yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fff'Z'"));
```

Enjoy !

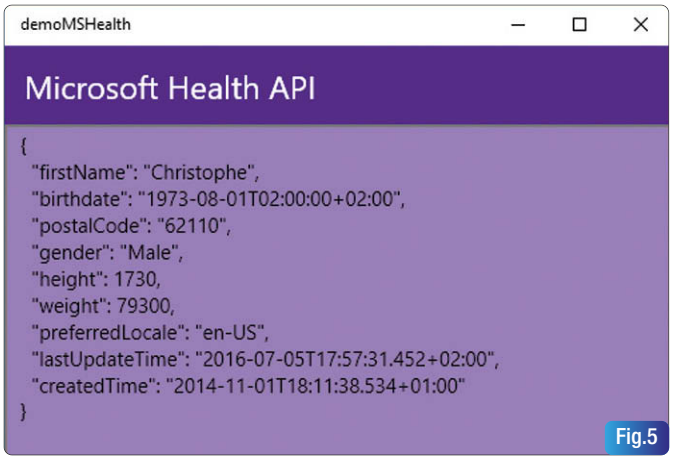


Fig.5

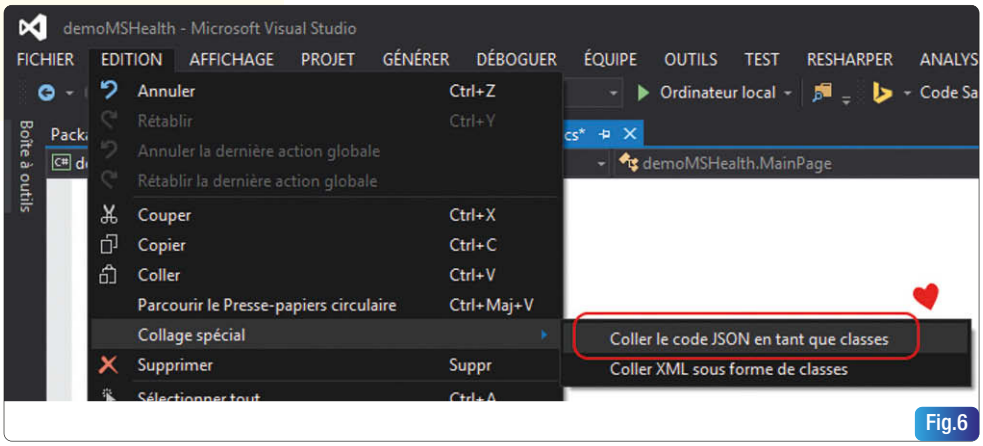
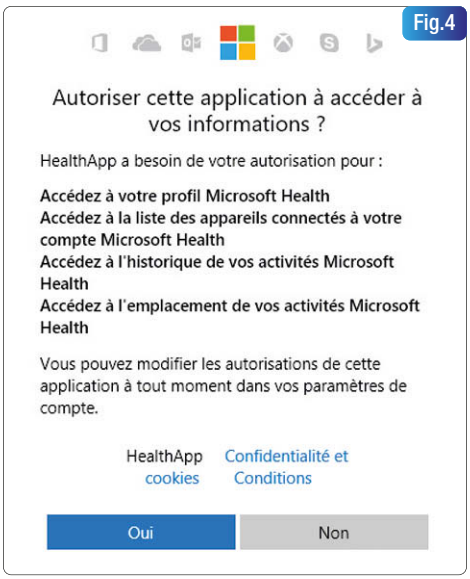


Fig.6



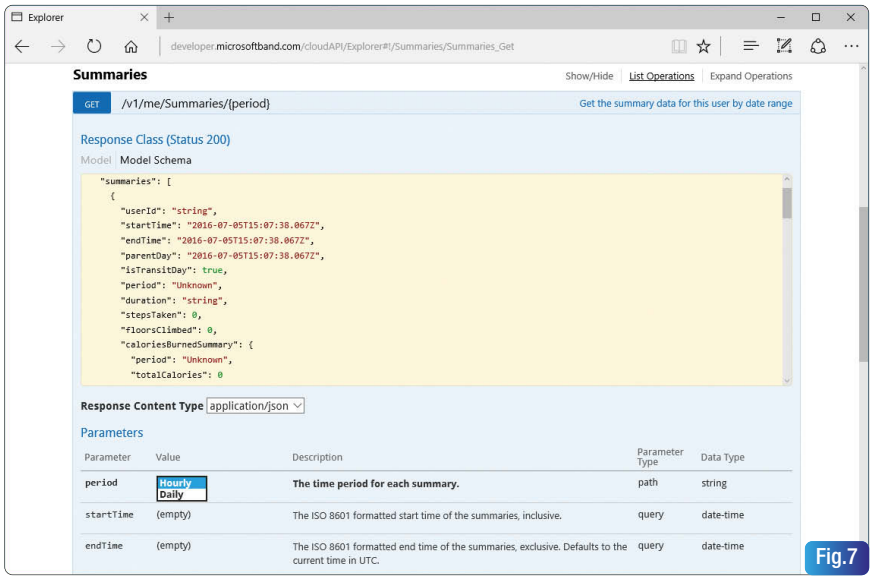



Fig.7

# Apple et la santé : au-delà du suivi personnel

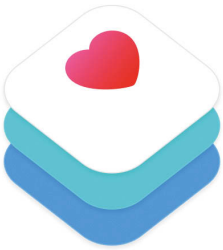
*Depuis plusieurs années, Apple a associé à ses appareils des apps et des fonctionnalités pour promouvoir la santé à différents niveaux. Que ce soit pour permettre à tout un chacun d'avoir un rythme de vie sain ou de suivre une activité sportive intense, mais aussi pour permettre à des chercheurs de conduire des études médicales à grande échelle ou bien aux professionnels de santé d'effectuer un suivi de leurs patients.*

Pascal Batty
*Expert iOS chez* **SOAT**

Plus que les pièces d'un puzzle, ces éléments s'emboîtent et dépendent les uns des autres. Healthkit permet de consolider les informations des capteurs et des apps, ResearchKit est un framework de création d'applications de suivi pour de la recherche et CareKit aide les professionnels à créer des apps afin de suivre leurs patients à distance.

Tous ces outils dépendent eux-mêmes de la plateforme matérielle d'Apple, notamment l'iPhone et l'Apple Watch, qui assurent non seulement la collecte de certaines données mais aussi leur stockage sécurisé.

## Healthkit et l'application Santé



Quelques mois avant de dévoiler sa montre connectée, Apple annonçait en juin 2014 HealthKit, une fonctionnalité d'iOS 8 qui met à disposition des développeurs un dépôt unique pour stocker et récupérer les données de santé du possesseur de l'iPhone. Si l'utilisateur l'autorise, il peut par exemple permettre à son application de suivi de footing un accès aux calories de ses repas, renseignées par une autre application.

Il évite ainsi de renseigner plusieurs fois la même information et les apps peuvent proposer des fonctions de suivi plus adaptées en allant chercher des infos qui sortent de leur propre silo. Si certaines de ces données sont alimentées par des apps, l'appareil peut en remplir certaines en autonomie. Depuis le modèle 5s, l'iPhone est équipé d'un coprocesseur "M" dont le rôle est de collecter certaines données comme les mouvements et l'altitude sans surcharger la batterie. Cela permet à l'appareil de compter le nombre de pas ou les étages montés sans qu'aucune app tierce soit installée. La sécurité est un aspect important pour Apple et les données de santé sont un point particulièrement sensible. Toutes les informations sont chiffrées et stockées sur l'appareil uniquement, et les apps qui s'appuient dessus ne peuvent les interroger en tâche de fond que si l'appareil est déverrouillé.

Les réglages de confidentialité permettent à l'utilisateur de choisir à quelles données accède chaque application compatible. Il est également possible de désactiver le comptage des pas que l'iPhone effectue automatiquement.

Apple exige des développeurs d'applications qu'ils n'expédient pas les données de santé sur Internet sans l'accord de l'utilisateur, et que ces données ne soient pas transmises à des tiers ou utilisées à des fins publicitaires. Cependant, il n'existe probablement aucun moyen de s'assurer que ces règles sont respectées. En mai 2016 le Norwegian Consumer Concil, qui lutte pour la protection des consommateurs, rapportait par exemple que l'application RunKeeper suivait la position de l'utilisateur plusieurs heures après une session de footing et la partageait avec une société de publicité. Le développeur s'est depuis excusé et a promis de rentrer dans le rang.

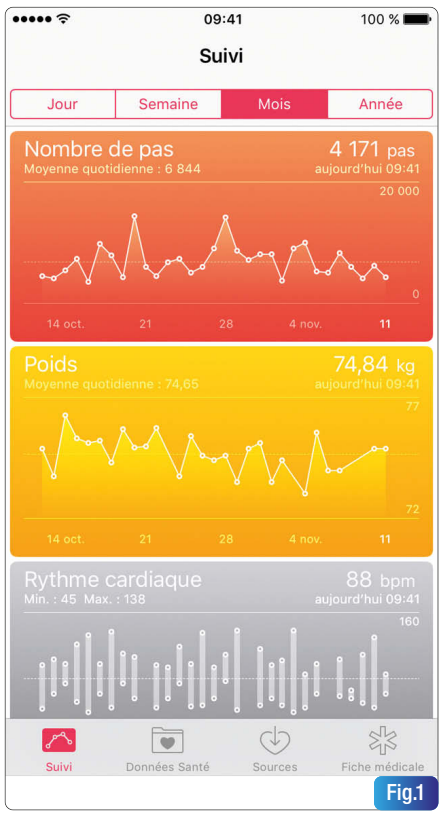


Fig.1

HealthKit est transparent pour l'utilisateur, qui peut à tout moment consulter et modifier les données stockées grâce à l'app Santé Fig.1, pré-installée sur l'iPhone depuis iOS 8.

Elle permet de visualiser toutes les données sous forme de graphes, que ce soit celles que l'appareil a récupérées par lui-même comme les pas, ou celles fournies par des applications tierces. On y retrouve les réglages de confidentialité et il est également possible de supprimer ou d'ajouter des données manuellement dans le dépôt.

On peut aussi y renseigner une "fiche médicale" avec plusieurs données comme le groupe sanguin, des allergies ou un personne à contacter ; visible depuis l'écran d'appel d'urgence sans avoir à déverrouiller l'appareil. Elle peut permettre à quelqu'un d'avoir des informations essentielles sur l'utilisateur en cas d'accident ou d'attaque par exemple, mais il est peu probable que ces informations soient recevables pour une prise en charge hospitalière.

Comme de nombreuses fonctionnalités de l'iPhone, HealthKit a été adopté par de nombreux développeurs très vite après sa sortie. Aujourd'hui, la quasi-totalité des apps de santé et de fitness s'appuient dessus pour échanger leurs informations, afin de proposer aux utilisateurs des fonctionnalités plus riches.

## Utiliser HealthKit

La lecture et l'écriture de données dans HealthKit est relativement simple. Votre application doit avant tout créer un objet *HKHealthStore*, qui servira d'intermédiaire avec la base de données HealthKit. L'étape suivante est de demander l'autorisation d'accéder en lecture et en écriture aux données qui vous intéressent. Sur votre *HKHealthStore*, la méthode *requestAuthorizationToShareTypes(readTypes:completion:)* affiche un formulaire permettant à l'utilisateur d'accepter ou refuser chaque demande, en lecture ou en écriture.

Vous pouvez vérifier les autorisations obtenues en écriture à l'aide de la méthode *authorizationStatusForType()* mais les autorisations en lecture ne vous sont pas communiquées ; à la place les requêtes retourneront une absence de données ou seulement celles que vous aurez inscrites.
Pour inscrire des données, vous devez créer des objets des différentes sous-classes de HKSample, en utilisant les différents constructeurs. Vous ne devez jamais sous-classer ces objets. Vous pourrez ensuite les inscrire dans votre *HKHealthStore* avec la méthode *saveObject(withCompletion:)* ou *saveObjects(withCompletion:)* pour en inscrire plusieurs.
Pour interroger les données de la base, créez un objet *HKQuery* dans lequel vous indiquerez le type de résultat, des options de filtrage et de tri ainsi qu'une closure à exécuter à la fin et passez-le à la méthode *executeQuery()* de votre *HKHealthStore*. Plusieurs types de requêtes sont disponibles comme les requêtes simples de points de données *HKSampleQuery*, les requêtes de statistiques *HKStatisticsQuery* et les requêtes continues *HKObserverQuery*.

## Le rôle de l'Apple Watch

L'Apple Watch est un atout majeur dans la stratégie d'Apple concernant la santé et elle complète le smartphone dans ce sens.
Elle est capable d'alimenter ce dépôt avec des données de mouvement plus précises, mais aussi certaines informations que l'iPhone ne sait pas obtenir, comme le rythme cardiaque et les activités sportives.


Si l'app Santé sur iPhone présente les données de façon très sobre, la Watch propose à l'utilisateur un suivi beaucoup plus stylisé et ludique de l'activité, avec ses trois anneaux à remplir chaque jour : "Me lever", "Bouger" et "M'entraîner". Elle tapote le poignet de son propriétaire pour lui signaler qu'il est resté assis pendant trop longtemps, compte ses pas de la journée et surveille son rythme cardiaque pour identifier les phases d'activité sportive. Au fil de la journée, les cercles se remplissent à mesure que l'utilisateur complète ses objectifs personnels et des médailles récompensent sa progression. Avec watchOS 3 prévu pour cet automne, Apple ajoutera un aspect social et compétitif au suivi d'activité avec la possibilité de partager ses résultats avec ses amis et leur envoyer des défis. La nouvelle version voit aussi apparaître une app pour pratiquer la respiration profonde et écarter le stress. À l'instar du rappel pour se lever, il est possible de demander à la montre de nous rappeler régulièrement de respirer un coup. Et comme la santé concerne tout le monde, les cas d'utilisation seront adaptés aux usagers en fauteuil roulant.

## ResearchKit, un framework pour la recherche médicale

Apple ne voit pas seulement dans sa plateforme un outil servant à chacun de suivre son activité physique, elle voit aussi dans ses ordinateurs de poche ou de poignet un moyen d'améliorer considérablement la recherche médicale.
La façon traditionnelle de conduire une recherche médicale demande énormément de temps pour traiter un petit nombre de participants, qui sont en plus très souvent rassemblés sur une même zone géographique. Cela pénalise la quantité et la qualité des données sur lesquelles les scientifiques peuvent travailler et donc à la fois le temps pris pour la recherche et les résultats possibles.
ResearchKit, présenté en mars 2015 par Apple, vise à changer l'échelle de l'étude de recherche médicale en faisant de tout possesseur d'iPhone un participant potentiel.
Il s'agit d'un framework open source contenant tous les outils nécessaires pour développer une app d'étude. ResearchKit contient les éléments de construction pour réaliser une app permettant à chaque participant d'être informé du sujet de la recherche, de donner son consentement et d'effectuer des tâches régulières dont les résultats sont transmis aux chercheurs automatiquement.Fig.2.
Certaines activités collectent les informations via HealthKit. Le framework permet de présenter ces données à l'utilisateur sous forme de graphe afin que l'étude lui permette également de constater ses résultats.
Le blog du projet Researchkit (http://researchkit.org/blog.html) donne la parole à des équipes pluridisciplinaires pour évoquer certains projets conçus grâce à, et autour de ResearchKit, les challenges et les succès rencontrés. Que ce soit afin de mieux comprendre la maladie de Parkinson, chercher à détecter les signes de l'autisme chez les enfants, voire prédire une attaque cardiaque imminente à l'aide d'une Watch, les projets sont variés et ambitieux. Leur efficacité est décuplée par le nombre de personnes que les chercheurs peuvent cibler à l'aide d'une app, plutôt qu'en passant par les canaux traditionnels.
Des initiatives innovantes s'appuient sur ResearchKit comme Sea Hero Quest Fig.3, un jeu vidéo conçu en partenariat avec l'University College de Londres. Le joueur doit mémoriser une carte maritime avant de naviguer de balise en balise, ses actions sont collectées afin de mieux comprendre les mécanismes de la mémoire et faire avancer notre compréhension de la maladie d'Alzheimer. La présentation du jeu explique qu'y jouer seulement 2 minutes permet de collecter l'équivalent de 5 heures de recherche traditionnelle, un avantage considérable. Les avis collectés auprès des chercheurs sont unanimes : la possibilité de toucher un très grand nombre de personnes à travers le monde et de collecter leurs résultats d'étude automatiquement démultiplie la qualité de la recherche.
Il est encore trop tôt pour créditer à ResearchKit des avancées notables mais il semble tout-à-fait clair que l'utilisation de nos smartphones dans le cadre de la recherche médicale présente un progrès considérable.

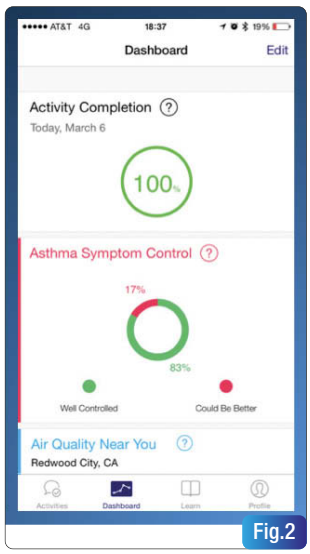


Fig.2

Fig.3

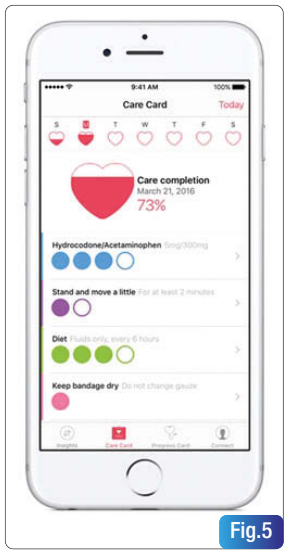


Fig.5

## Utiliser ResearchKit

Afin de présenter au sujet de l'étude une tâche à accomplir, qui peut être composée de plusieurs étapes, ResearchKit propose un *ViewController* dédié : *ORKTaskViewController*. Pour l'utiliser, vous devez l'alimenter avec un objet abstrait *ORKTask*, lui-même composé de plusieurs instances d'*ORKStep*, chacune étant une étape de l'activité. Vous pouvez enchaîner plusieurs types d'étapes : instructions, demandes de consentement, questions ou encore des étapes actives comme tapoter des doigts sur l'écran ou marcher quelques pas. Fig.4. *OKRTaskViewController* va à son tour contenir un *OKRStepViewController* pour chaque étape (*OKRStep*) de la tâche, qu'il affichera tour à tour au sujet de l'étude. Pendant le passage des étapes ou à l'issue de la tâche, des classes contenant les résultats sont alimentées et peuvent être exploitées en tâche de fond : *OKRTaskResult* et *OKRStepResult*.
Afin d'afficher des graphes, ResearchKit propose plusieurs classes de vues que vous pouvez manipuler : *OKRPieChartView*, *OKRLineGraphChartView* et *OKRDiscreteGraphChartView*.

## CareKit, pour le suivi des patients

Avec CareKit, l'iPhone aide le suivi médical d'un patient et peut lui permettre d'échanger avec son médecin. Ce framework open source, propose des modules basés sur ResearchKit pour construire une app de suivi médical.
Une app CareKit peut aisément présenter des interfaces pour rappeler au patient quelles actions il doit faire régulièrement comme prendre ses médicaments, faire une promenade ou renouveler un pansement. L'interface est moins ludique que le suivi d'activité de la Watch mais les principes de "ludification" s'y retrouvent : objectifs quotidiens et cases à cocher pour chaque tâche. Fig.5. Le patient peut également renseigner des informations chaque jour comme son niveau de douleur, l'évaluation de son humeur ou sa glycémie. Ces données peuvent par exemple être affichées sous forme de graphe pour lui permettre de suivre la progression de son état.
Le dernier module concerne les personnes entourant le patient, comme les membres de sa famille et les praticiens concernés par son état. Depuis cet écran le patient peut par exemple contacter directement ces personnes ou leur envoyer les informations collectées par l'app.
Des applications de suivi utilisant CareKit sont déjà disponibles dans l'App Store, comme One Drop pour les patients diabétiques, Glow pour le suivi de la fertilité ou Start pour surveiller la prise d'anti-dépresseurs.
Certains hôpitaux proposent également une application à leurs patients pour assurer leur suivi, par exemple en post-opératoire ou pour les personnes atteintes de maladies chroniques.

## Utiliser CareKit

4 modules sont proposés par CareKit pour afficher et collecter de l'information. Chacun est un ViewController et la façon traditionnelle de les arranger est une barre d'onglets (mais rien ne vous empêche de les utiliser autrement).
CareCard ou *OCKCareCardViewController* est l'écran quotidien du patient. On y affiche principalement des activités à effectuer régulièrement comme une prise de traitement ou une marche de plusieurs minutes. Chacune de ces activités est représentée par un objet *OCKCarePlanActivity* de type Intervention. Cet objet contient des informations sur l'activité comme son nom, son rythme et un paragraphe d'explication accompagné d'une image. Pour chaque occurrence de l'activité, CareKit créera un objet *OCKCarePlanEvent* pour suivre la progression. Le suivi des symptômes et des mesures est présenté par l'objet *OCKSymptomTrackerViewController*. Lui aussi s'appuie sur des instances de *OCKCarePlanActivity* mais de type Assessment, afin de demander au patient de renseigner des informations de suivi ou de répondre à une question comme une évaluation de la douleur par exemple. De la même manière il est possible de définir des activités de suivi régulières et d'en suivre la progression à l'aide des *OCKCarePlanEvent*.
Le troisième module permet au patient de visualiser ses progrès via le *OCKInsightsVewController*, qui s'alimente d'une collection d'objets *OCKInsightItem* qui peuvent être des messages ou des graphes à afficher.
Le quatrième module s'appelle Connect ou *OCKConnectViewController*, il peut afficher un ensemble de contacts ou *OCKContact*, des structures de données stockant les informations de contact des personnes pertinentes comme le praticien ou la famille proche.
Deux autres modules travaillent en arrière-plan. *OCKCarePlanStore* est la base de données où toutes les informations sont stockées, et *OCKDocument* permet de créer un document PDF ou HTML contenant des informations de CareKit pour les transmettre à un des contacts.

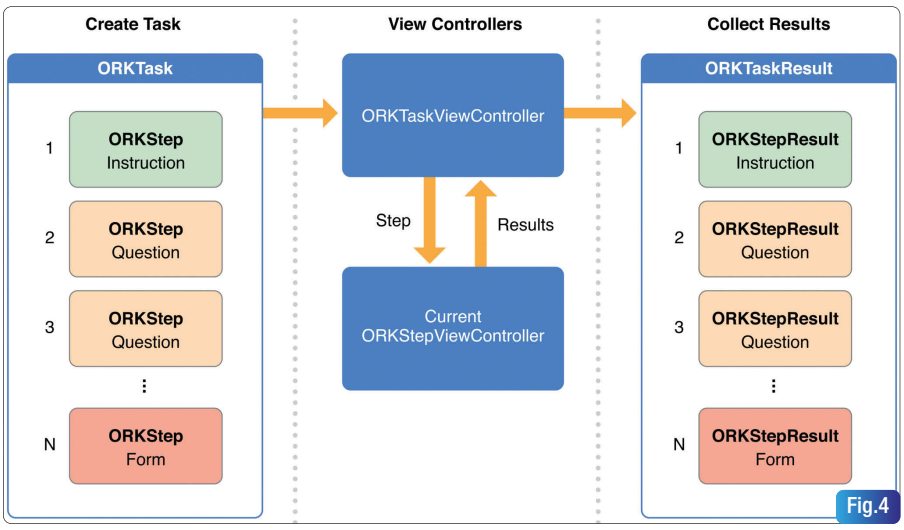


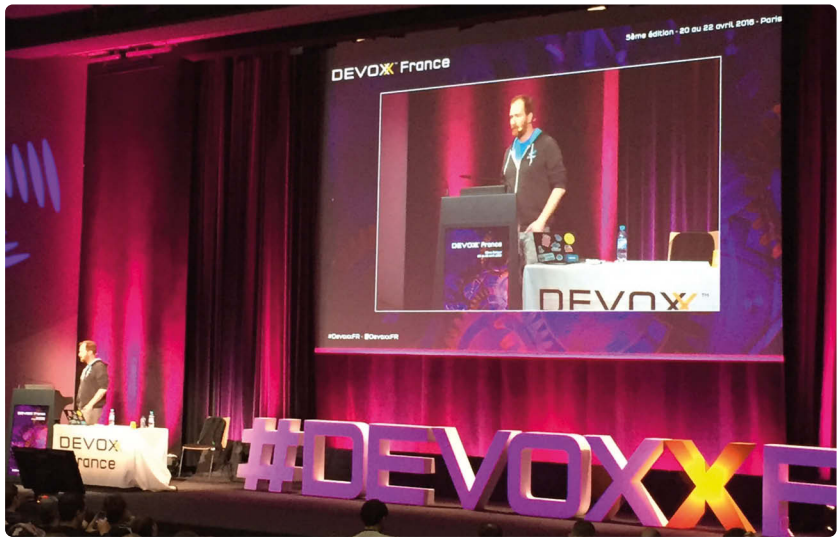
Fig.4

## À nous de jouer !

Avec ses initiatives plus sérieuses, Apple semble chercher à changer notre regard sur nos smartphones et nos montres connectés, et leur potentiel dans notre vie, que ce soit au quotidien ou à long terme.
L'avenir nous dira si nos "gadgets" ont la capacité d'améliorer notre vie en tant que patients ou de démultiplier notre capacité à conduire et suivre des recherches médicales, mais les outils sont là, gratuits et ouverts. HealthKit permet d'inventer de nouvelles façons de travailler avec les données de santé sans se soucier de leur collecte et si ResearchKit et CareKit semblent se limiter à des cas d'usage plus spécifiques, leurs utilisations dépassent déjà le stade de l'application officielle d'un hôpital ou d'une université, avec Sea Hero Quest, Glow, One Drop et Start par exemple. La plupart de ces applications sont limitées au territoire américain, des opportunités sont présentes en France et dans d'autres pays. C'est en fabriquant de telles apps et en les utilisant que nous serons capables de faire la différence. Il serait dommage que ces ordinateurs surpuissants que l'on promène partout avec nous ne nous servent qu'à suivre nos calories ingérées et compter nos pas de la journée.

# Devoxx France 2016

suite et fin




**Nous poursuivons notre résumé de la conférence Devoxx France 2016 d'avril dernier. Dans cette partie, nous allons parler prospective, hacker & devops et développement mobile. Les vidéos des sessions sont disponibles sur la chaîne officielle :** *https://goo.gl/VJQiYx* *Enjoy !*

## En mode H2G2

*Parmi les nombreuses sessions techniques de Devoxx France, on trouve toujours une session qui nous intrigue. Impossible de ne pas cocher la session Hitch's Hacker Guide to Docker's galaxy. Et le résultat a été top avec un speaker en pleine forme, Hicham Tolimat.*

François Tonic
*Programmez!*

L'idée est simple : DevOps, avec Docker, vu par un hacker. Et il est temps d'y mettre de la sécurité, élément que l'on oublie un peu trop souvent dans le mouvement DevOps et même parfois dans les infrastructures Docker. On revient naturellement aux fondamentaux : virtualisation vs conteneurs et au DevOps et comment tout cela s'harmonise ensemble avec l'intégration continue, les tests, le reportage, le commit des projets, le code, etc. Mais il y a tout de même un problème de fond qu'il ne faut pas négliger : la sécurité dans les conteneurs, que se soit Docker ou d'autres solutions. Il ne faut pas négliger la sécurité des conteneurs, souvenez-vous des polémiques sur les machines virtuelles et les rootkits. Au moins un exploit a été réalisé par l'équipe FlawCheck. Le conteneur est une véritable révolution pour les environnements ARM et x86 mais il y a une petite peur avec ARM car il faut « tout » refaire. Mais quand vos conteneurs sont bien sécurisés, pas la peine de réinventer la roue à chaque fois, appliquer le contenu. Par exemple, Flocker qui réinvente le stockage, il faut arrêter de le faire. Un volume de stockage peut être vu comme un fichier, donc on peut utiliser du torrent. Le P2P est une bonne approche. Utilisez ce qui existe déjà, tel que Consul (voir Programmez ! 198).

### Le développeur est un trou de sécurité (ou pas)

Trop souvent on a opposé développeur et sécurité et c'est vrai que la sécurité est, encore, trop souvent la dernière roue d'un projet et intervient trop tardivement dans celui-ci. Et quand on regarde les conteneurs et Docker, on s'aperçoit que le développeur fait aussi du code système, ce qui n'est pas sans problème. Quand un développeur pousse un conteneur, les tests de sécurité ont-ils été réalisés ? Le conteneur, fait par le développeur, respecte-t-il les règles de sécurité ? Il faut faire très attention.

En réalité, la sécurité est complexe mais de facto, on étend la surface d'attaque et les failles sont plus nombreuses. Parfois, le code utilisé n'est pas testé et avec un risque potentiel. On fragilise donc le code et le projet tout entier.

Finalement, écrire des rapports est plus long que de corriger le bug, la faille. On multiplie les allers-retours mais pour quel résultat ? Mieux vaut communiquer directement entre les équipes, et le DevOps aide à aller plus vide, à être plus agile. Mais il faut comprendre le contexte. Sans cela, la communication ne se fera pas. C'est pour cela que le DevOpsSecu est important à intégrer, notamment dans une approche d'intégration continue. Il faut absolument faire de la sécurité dans le cycle de développement, tout comme, on fait les tests très tôt dans ce cycle.

Comme Hicham le disait si bien : nous faisons le pompier mais il faut faire de la prévention. Car, agir en pompier, c'est agir dans l'urgence mais cela ne règle en rien le problème de fond : les équipes et les méthodes utilisées. Il existe différents canaux que le développeur et les équipes peuvent utiliser : Github, CVSS, etc. Le hacker a son rôle à coeur et peut conseiller et aider les développeurs.

Il ne faut pas agir après la mise en production, c'est trop tard !

# Comment concevoir la navigation mobile ?

*Développeur chez PALO IT, j'ai eu l'occasion d'assister au Devoxx France. Après lecture du programme, j'ai rapidement été attiré par la conférence sur l'UX Mobile proposée par Amélie Boucher, experte reconnue de l'expérience utilisateur Web & Mobile. En effet, là où il y a encore quelques années, on se posait encore la question de comment adapter une navigation Web classique vers une navigation mobile, on s'interroge aujourd'hui sur comment concevoir de A à Z la meilleure navigation mobile possible. En me basant sur sa conférence et des recherches complémentaires, je propose à travers cet article un état de l'art de la navigation mobile.*

Yassin CHABEB
*Développeur chez*
**PALO IT France**
*Yassin justifie de 5 ans d'expérience dans l'univers du Développement Web et de la Programmation Java dans l'Industrie et la R&D. Il est souvent intervenu en méthodologie Agile lors de ses différentes missions en tant que Développeur Web.*

Lors de la création d'une application ou d'un site Web mobile, il y a différentes façons d'aborder les approches de conception mobile. Pour toutes ces approches, la navigation est l'une des plus importantes considérations dans l'UX et la construction de l'interface utilisateur. Pour un Développeur ou un Designer, deux cas se présentent : le premier se résume par la question suivante : comment puis-je adapter une navigation Web classique vers une navigation mobile d'un site habituellement consulté depuis un ordinateur par l'utilisateur, sans le frustrer à cause d'une spécificité de l'environnement mobile ?

Le deuxième cas présente deux situations qui n'ont aucune exigence liée à un site Web classique existant comme dans le premier cas : comment construire de A à Z une navigation mobile fonctionnellement exhaustive ? Ou comment construire une application dont la navigation se distingue d'une dizaine (voire une centaine) d'applications équivalentes et séduire ainsi plus d'utilisateurs ?

Dans les deux cas, seules la navigation mobile et l'UX sont capables d'influencer le succès ou l'échec du produit mobile (site, application, widget, etc.). En résumé, cela dépend entièrement de l'UX que vous voulez donner et souvent de l'aisance ou parfois de l'originalité de la navigation. Avant de rechercher l'originalité, rappelons d'abord la liste des approches et des éléments graphiques les plus connus et utilisés en navigation mobile.

Au début de mes recherches, je pensais qu'il y avait au maximum une dizaine d'éléments mais en fait, il y en a moins d'une trentaine, principalement en variantes que nous utilisons chaque jour inconsciemment et que les Designers essaient de croiser afin d'en faire des entités hybrides pour satisfaire notre envie de navigation simple et efficace.

S'il y a autant d'éléments, c'est qu'aucun d'entre eux n'est adapté à toutes les conceptions et qu'il existe des avantages et des inconvénients pour chaque type de navigation. Ce qui nous amène à la question existentielle : « Quelle est la meilleure approche de navigation mobile ? ». Question à laquelle j'ai fini par donner la réponse éternelle : « Cela dépend ! ». Heureusement que les bonnes pratiques sont toujours là pour vous guider lorsque vous "Programmez".

Les bonnes pratiques sont notre seul garant de l'équilibre fragile de la navigation mobile sur un écran de quelques centimètres manipulé souvent avec un seul doigt.

Elles sont notre ultime arme contre une anarchie palpable, qui nous guette à chaque action inattendue (clic futile, écran inutile, bouton mal placé, menu trop long, etc.).

## L'UX MOBILE : ENTRE EVOLUTION ET REVOLUTION

La navigation mobile est une question de temps et d'espace : vous avez quelques secondes pour séduire l'utilisateur et lui faire oublier de réfléchir où cliquer (de l'hypnose en quelque sorte !). Faire adopter un mécanisme de navigation à une personne est un processus qui passe par le choix du type de navigation, des éléments, de l'anticipation des actions et parfois d'astuces d'apprentissage intelligent.

Le cerveau humain est parfaitement intra-connecté : il récupère l'information nécessaire pour réaliser subtilement l'action adéquate, mais il décroche rapidement dès que l'outil n'est pas à la hauteur. Le monde entier et en particulier notre patrimoine Internet et Web se projette dans le mobile. Imaginez toutes les barrières qui sautent entre Internet et les personnes ne manipulant pas d'ordinateur qui, du jour au lendemain, ont le monde au bout du doigt. Les services, les APIs et le Cloud ont besoin que ce nouvel environnement réussisse à donner de nouvelles capacités à chaque individu.

L'enjeu n'a jamais été aussi important économiquement, politiquement et socialement. L'UX est au cœur de la deuxième plus grande expérience humaine dans le secteur de l'info-communication. Il est attaché à la fois à une évolution (d'Internet et du Web) et à une révolution (dans le comportement humain). Et ce basculement du Web classique au mobile s'annonce déjà autant compliqué que passionnant.

*" Internet est la première chose que l'homme a créée sans la comprendre, c'est la plus grande expérience en matière d'anarchie jamais réalisée"— Eric Schmidt.*

## MENUS, PAGINATION ET DEFILEMENT : LES APPROCHES DE NAVIGATION

Principalement, les approches de navigation mobile tournent autour des trois principes suivants :

- Les Menus ;
- La Pagination ;
- Le Défilement (*Scrolling*).

Ces principes utilisent une liste d'éléments et de concepts d'interaction afin d'exposer leur mécanisme de navigation et d'étaler les différentes étapes d'actions, de recherche, de sélection, de transitions et/ou d'affichage de contenu. Voici une liste plus ou moins exhaustive (certains termes sont en français, d'autres en anglais) que nous détaillerons plus bas :

*« List, tap bar, tiroir caché, pop over, mur d'icônes, navigation bar, hamburger menu, menu à gauche, menu déroulant, menu en accordéon, off canvas, pas de sous-menu, fil d'ariane, top navigation, menu en footer, spinner wheel ».*

Les applications (et les Designers derrière) ne cessent de proposer aux utilisateurs de nouvelles astuces de navigation grâce aux combinaisons réalisées avec subtilité en se basant sur cette liste. Il faut toujours être inventif et garder à

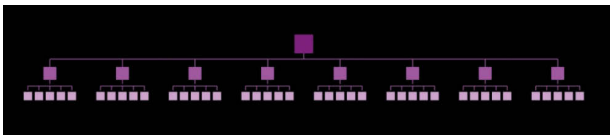


*Fig.1 Architecture plate*

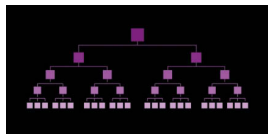


*Fig.2 Architecture profonde*

l'esprit que certains éléments peuvent être utiles pour un ou plusieurs types de navigation dès qu'ils répondent à leurs principes :

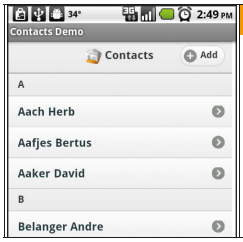
- La navigation par les menus peut être construite à base de tout élément ou suite d'éléments adjacents et / ou imbriqués et / ou hiérarchiques qui peuvent tous être sélectionnables ou présentant un élément sélectionnable (cliquable). La hiérarchisation des menus reste la technique la plus subtile afin de passer d'une architecture d'information plate (*cf. fig.1*) à une architecture profonde (*cf. fig.2*). Elle doit être intuitive afin que l'utilisateur puisse deviner la branche à déplier. Cette architecture en profondeur est plus fréquente en mobile.
- La pagination permet une navigation à base d'éléments visuellement plus discrets (ou avec moins d'interactions) que les éléments de la navigation en menus ; le but étant toujours de récupérer la suite (ou l'avant) de l'écran (la page) en cours. Les éléments de navigation doivent permettre d'afficher un contenu de même type (image, texte, etc.), contrairement à une navigation par menu offrant différentes actions pour afficher un contenu différent. C'est pour cette raison que souvent, les glissements de doigts, les boutons-flèches et autres éléments compatibles sont utilisés.
- La navigation par défilement utilise des éléments assurant un déroulement du contenu avec un chargement de contenu plus ou moins intelligent et automatique. Cette approche emprunte souvent aux deux autres quelques éléments afin de casser la monotonie et remédier aux effets parfois désagréables pour la vision humaine. Certaines personnes sont sensibles au défilement continu suite à la vérification furtive du contenu déroulé avant qu'il ne disparaisse. Pour éviter que leurs yeux ne se fatiguent, on peut utiliser les menus ou une pagination perpendiculaire au défilement pour équilibrer la navigation et faire de petites pauses Fig.1 et 2.

Avant d'étudier les avantages et les inconvénients de chaque composant de navigation, rappelons que la valeur ajoutée d'un élément ou d'un concept dans un processus de navigation ainsi que son rôle dépendent de plusieurs paramètres. Les choix et les valeurs assignées à ces paramètres font qu'un double-clic sera adapté dans tel contexte de tel type de navigation, ou qu'une transition sera utile sur tel écran dans un autre type de navigation.

Certains éléments sont beaucoup plus flexibles que d'autres sur les marges des valeurs et les choix qu'ils peuvent offrir. Ces paramètres évoluent selon les 3 axes de l'interaction mobile, à savoir l'espace tactile, la vision et le son. Voici la liste des paramètres les plus influents à prendre en compte tout au long de la conception de votre navigation mobile :

- La taille ;
- La position (sur un des quatre bords de l'écran / flottant / au coin) ;
- Le moment d'apparition / une action nécessaire ou pas pour l'affichage ;
- Le moment de disparition / une action nécessaire ou pas pour le masquage ;
- La couleur / la transparence ;
- Le sens de défilement ;
- La notification (modale ou flottante / furtive, dynamique ou statique) ;
- Avec animation ou pas, si oui sa durée, serait-elle accompagnée d'un son ou pas ;
- Le taux de réutilisation de l'écran, zones statiques inchangées ou avec un contenu dynamique.

## LES AVANTAGES ET LES INCONVÉNIENTS DES COMPOSANTS DE NAVIGATION




**List**

- avec/sans sous-listes
- ordonnée ou pas
- indexée ou pas
- avec/sans barre de défilement

**+**

Mise en place simple
Connaissance d'utilisation et recherche rapide
Pas de limite

**-**

Peu attrayant et motivant à la découverte
Navigation monotone si la taille est grande

**Tap bar**

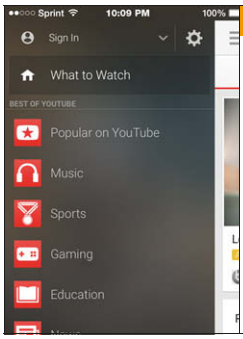
- souvent en bas
- icônes/textes
- assimilée à la pagination

**+**

Intuitive : 1 clic = 1 écran
Utilisation aisée et pas de recherche
Facilement accessible si affichée en bas

**-**

Limitée à 5 choix au maximum
Exigence sur la haute expressivité des icônes
Péniblement accessible si affichée en haut




**Tiroir caché**

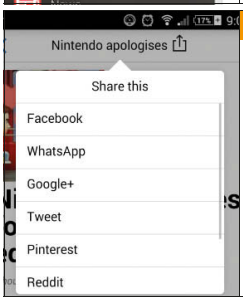
- avec/sans bouton afficheur
- affichable ou non par glissement
- par défaut masqué totalement ou partiellement

**+**

Economie d'espace
Utilisation de plus en plus familière
Possibilité de se contenter de glisser

**-**

2 clics au minimum (1glissement + 1clic) pour passer à un autre écran
Bouton afficheur ou éléments du haut péniblement accessibles avec une seule main
Ecran en cours décalé par le tiroir, parfois lent s'il y a animation, et subite s'il n'y en a pas




**Pop over**

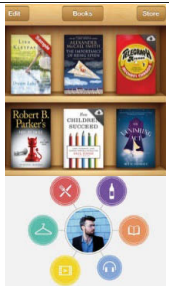
- affichable par un clic
- indiquée souvent par un triangle orienté en bas
- assimilée à un menu flottant

**+**

Simple dès qu'on repère le petit triangle
Visuellement plus légère qu'un menu
Permet de distribuer l'affichage de petits menus sur les éléments susceptibles d'offrir des actions

**-**

2 clics au minimum pour passer à un autre écran
Péniblement accessible si elle est affichée en haut de l'écran



**Mur d'icônes**

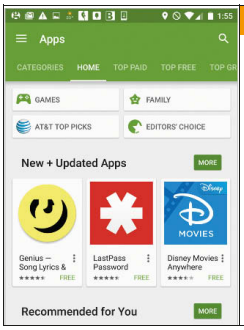
- avec/sans textes
- ordonnée ou pas
- indexée ou pas
- avec/sans barre de défilement
- imbriquer des icônes de différentes tailles

**+**

Utilisation aisée
Contenu accessible au bout d'un clic
Souvent motivant à la découverte

**-**

Si pas assez étudié, peut mener facilement au désordre et à la confusion
Exigence sur la haute expressivité et la qualité des icônes




### Navigation bar

- souvent en haut
- souvent en textes
- souvent glissante

**+**

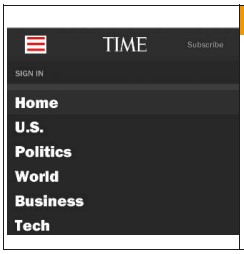
Courte recherche avec un glissement puis 1 clic pour avoir 1 écran

Plus de contenu qu'une tab-bar, prend moins de place qu'un menu vertical

**-**

Il faut la limiter de préférence à une dizaine de choix sinon la navigation devient très plate

Péniblement accessible pour un glissement et un clic si elle est affichée en haut (pour une utilisation avec une seule main)




### Hamburger menu

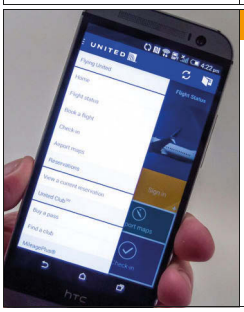
- avec/sans champ de recherche au moment de l'affichage du menu
- avec/sans élément de notification (avec/sans chiffre comme pour les messages non lus)

**+**

Economie d'espace
Simple dès qu'on repère le petit hamburger
Utilisation de plus en plus familière

**-**

2 clics au minimum pour passer à un autre écran
La partie d'en haut, péniblement accessible avec une seule main



### Menu à gauche

- menu statique en haut de la page
- affichable par glissement
- par défaut masqué totalement ou partiellement

**+**

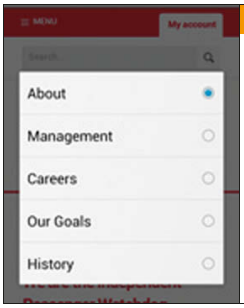
Sans indice, un utilisateur commence par chercher le menu à gauche intuitivement (le réflexe du glissement sur des applications phares comme Facebook)

**-**

La partie d'en haut est péniblement accessible en particulier pour les droitiers

L'écran en cours est décalé par le menu

Parfois, on doit glisser pour le masquer après le choix d'une rubrique (l'enfer !)




### Menu déroulant

- un clic lance une liste modale
- souvent que du texte
- la sélection d'un choix ferme la liste et charge la page

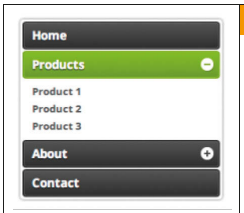
**+**

Simple d'utilisation dès qu'on repère le menu
Ne nécessite pas de recherche si la liste est courte
Flexible (l'ajout de nouvelles entrées ne pose pas de problème)

**-**

Fermeture obligatoire avec le bouton physique du smartphone si on ne veut plus choisir

Pas assez de place pour plus de design, souvent pas d'icônes donc moins attrayant




### Menu en accordéon

- un clic ouvre le sous niveau
- souvent que du texte
- le choix ne ferme pas la liste et charge la page

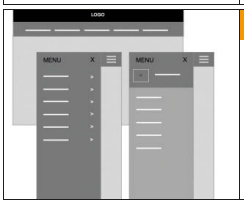
**+**

Accès facile aux sous menus
Idéal pour une hiérarchie à la fois peu plate et peu profonde

**-**

Fermeture obligatoire avec le bouton physique du smartphone si on ne veut plus choisir

A part les couleurs, pas assez de place pour les icônes et le design des sous niveaux




### Off canvas

- bouton retour en arrière pour revenir au niveau supérieur
- naviguer verticalement dans l'architecture

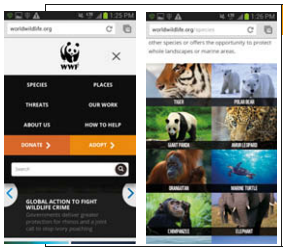
**+**

Simplifie les menus complexes à plusieurs niveaux

**-**

A éviter si l'utilisateur doit vite comparer des sous-catégories entre elles

Un peu lourd à mettre en place




### Pas de sous-menus

- depuis les catégories du menu principal on navigue en une page dédiée aux sous-catégories

**+**

Favorise la découverte grâce à l'espace :
la possibilité de design, la qualité des images

**-**

Oblige le retour en arrière et ralentit la navigation si l'architecture est profonde

A éviter si l'utilisateur doit vite comparer des sous-catégories au niveau 2 ou plus entre elles

## COMMENT CHOISIR SON APPROCHE ?

Au début, la navigation mobile sur les sites Web était plus ou moins une version allégée de leur navigation bureau. L'espace ne permettait pas d'offrir la même expérience utilisateur. Conserver la même navigation sur les plateformes mobiles et bureau est très difficile.

L'approche de calquer la version bureau n'était pas la meilleure option. Heureusement, avec le temps, les plates-formes mobiles ont vu leurs approches de navigation évoluer rapidement, soit parce qu'on ne retrouve plus de version bureau comme point de départ (par exemple, lors de la création de nouvelles applications et jeux mobiles), soit parce qu'on souhaite exploiter autrement le mobile pour explorer et naviguer de façon plus innovante que sur un ordinateur. Finalement, la contrainte de l'espace semble plutôt alimenter une créativité dans les approches de navigation plus qu'un manque d'options. Bousculer les habitudes des utilisateurs présente un risque qu'on ne peut éviter qu'avec des approches intuitives et facilement adoptables.

Malgré la dizaine d'année d'efforts et d'approches diverses et variées, la route est encore longue. En réalité, chaque approche est en quête d'une navigation offrant une découverte du contenu facile, accessible et qui prend peu de place à l'écran.

Etaler la navigation sur l'écran ou la cacher dans une icône/menu, chaque approche a des avantages et des inconvénients. Mais la première clé de la réussite du choix reste principalement liée au contenu.

Chaque type de contenu a une solution appropriée. Une navigation ne peut être construite indépendamment du contenu qu'elle expose. Disons alors que le contenu est notre premier paramètre de choix d'approche.

Si on réfléchit à l'instant à partir duquel l'utilisateur commence à naviguer et exploiter le contenu, un second paramètre entre en compte : le temps. On pense souvent au temps d'interaction avant que l'utilisateur obtienne ce qu'il souhaite, y compris le nombre de clics ou de transitions, la durée d'une animation ou d'un chargement, mais on pense moins à un autre facteur temps extrêmement important dans la mise en place d'une approche : le temps d'apprentissage de navigation. Plus une navigation est rapidement acquise, plus elle est vite oubliée et fera partie intégrante des réflexes d'utilisateurs, qui fera d'elle une approche innée au fil du temps, familière et facilement reconnaissable ce qui « miraculeusement » fera d'elle « LA » approche adéquate à tel ou tel type d'application.

Le temps d'apprentissage est tellement impor-

tant qu'on ne peut guère l'exclure. C'est pour cette raison que de plus en plus d'applications mobiles ajoutent une surcouche sur l'écran comme guide de navigation avec des instructions et des astuces pour rappeler ou enrichir l'expérience utilisateur tout au long de sa navigation. Bien évidemment derrière, le temps d'interaction et de réaction des écrans doit être à la hauteur, sinon la navigation serait un véritable calvaire et l'utilisateur n'adhèrerait plus à l'application.

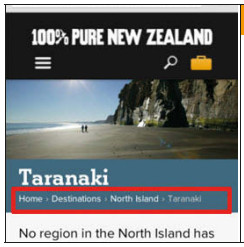
Ainsi, comme pour tout processus d'exploitation d'un outil technologique, le temps est un paramètre primordial, et la navigation mobile est loin d'être une exception vu qu'elle prétend nous faciliter notre quotidien si rythmé.

En résumé, il existe deux paramètres principaux : d'abord, le contenu et les métaphores visuelles que le Designer utilise afin que l'utilisateur comprenne ce qu'il voit (icônes, images, couleurs, symboles, etc.), et ce parfois en se basant sur sa mémoire profonde et l'inconscient (par exemple, une maison signifie Home ce qui signifie la page principale). Ensuite, le temps de navigation que le Designer essaie d'optimiser afin que l'utilisateur vive une belle expérience à chaque lancement de l'application.

## EN CONCLUSION

Il n'existe pas une approche universelle mais plutôt une démarche de construction de navigation en fonction des paramètres qui dépendent du contenu et du facteur temps tout en respectant certaines bonnes pratiques. La liste des bonnes pratiques ne pouvant être exhaustive, j'ai choisi de retenir les suivantes :

- Envisager des pages / écrans de taille raisonnable  pages (ne pas intégrer trop de contenus sur une seule page) ;
- Prévoir des éléments cliquables de taille assez large (surtout pour les applications dédiées aux hommes) ;
- Intégrer un espace suffisant entre les éléments cliquables ;
- Fournir à l'utilisateur un élément qui lui permet de se retrouver dans la navigation globale (numéro de page, fil d'ariane, etc.) ;
- Faciliter la navigation horizontale pour permettre à l'utilisateur d'accéder aux catégories principales en cas d'architecture d'information profonde.
- Permettre à l'utilisateur de naviguer en un seul clic / swipe (glisssement de doigt) entre deux écrans ;
- Prévoir l'ajustement de l'affichage en cas de zoom (et donc ajouter / masquer des options de navigation) ;
- Privilégier les déroulements de menus verticaux plutôt qu'horizontaux ;
- Privilégier les accordéons aux longs menus verticaux sans fin ;
- Eviter la redondance d'informations d'un écran à l'autre ;
- Lister les pages principales en premier ;
- Limiter les éléments de navigation à 8 par écran ;
- Rester intuitif : le signe + dans un menu engendre l'action « déplier » et vice versa ;
- Orienter la navigation vers l'action et la découverte (pas d'interaction inutile) ;
- Rendre tous les éléments de navigation accessibles (de préférence avec une seule main), compréhensibles (un utilisateur ne doit pas hésiter entre plusieurs boutons) et ergonomiques ;
- Prévoir une recherche ultra réactive et adaptée (textes, icônes) s'il y a beaucoup de contenus à découvrir.




**Le fil d'Ariane**

- positionner une page dans
- une arborescence
- niveaux supérieurs cliquables

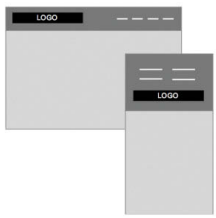
**+**

Indique à l'utilisateur où il se trouve

Permet de revenir aux niveaux de navigation supérieurs

**-**

Perte d'espace pour afficher le fil surtout s'il est long ou constitué de longs termes




**La top navigation**

- de 2 à 4 choix (le moins épais possible)
- fixes ou affichables par glissement
- si pas fixes alors de préférence masqués partiellement pour aider à les retrouver

**+**

Mise en place facile

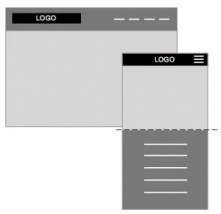
Si fixe alors directement visible ou avec un simple glissement

**-**

Utilisation difficile si l'arborescence est dense ou profonde

Peu flexible : difficile d'ajouter un choix non prévu d'avance

Pour avoir des choix aisément cliquables, il faut dédier assez de place en haut de l'écran




**Le menu en footer**

- un clic en haut bascule l'affichage en bas de page pour retrouver la navigation

**+**

Facile à mettre en place

En-tête de page aéré vu que le menu ne s'affiche en bas que sous demande

**-**

Utilisation difficile si l'arborescence est dense ou profonde

Le saut vers le bas de la page peut être perturbant voire gênant si l'utilisateur souhaite vérifier une information en haut de l'écran avant de naviguer


**Le spinner wheel**

- une roue tournante
- un clic pour changer de niveau ou de type d'action
- choix distingués par texte et couleurs (ou symboles)

**+**

Fun et attrayant

Entête de page aéré vu que le menu ne s'affiche en bas que sous demande

**-**

Utilisation difficile si l'arborescence est dense ou profonde

Le saut vers le bas de la page peut être perturbant voire gênant si l'utilisateur souhaite vérifier une information en haut de l'écran avant de naviguer

**Sources :**

Présentation Devoxx France 2016 d'Amélie Boucher :
http://fr.slideshare.net/amelieboucher/concevoir-la-navigation-sur-mobile
Modèles et patterns de navigation mobile
http://www.loriskumo.com/modeles-de-navigation-mobile/
Basic patterns for mobile navigation
https://www.nngroup.com/articles/mobile-navigation-patterns/
Choisir la bonne navigation sur son site mobile
http://www.adviso.ca/blog/2013/04/05/choisir-la-bonne-navigation-pour-son-site-mobile-2/
Les différents types de navigation sur mobile
http://www.ergognome.com/conception-mobile/les-differents-types-de-navigation-mobile/
5 creative mobile UI patterns
https://studio.uxpin.com/blog/5-creative-mobile-ui-patterns-navigation/

# Retour sur la keynote de clôture

*Dans le grand amphithéâtre principal du Palais des Congrès, la Keynote de clôture de ce Devoxx 2016 est sur le point de commencer... A la manière des conférences des géants de la Silicon Valley, l'agitation des uns se mêle aux conversations des autres autour des sujets en vogue de cette édition 2016 !*

Loïc LE GOFF
*Ingénieur Etude & Développement chez* **Netapsys Conseil**

Alors que certains manifestent leur intérêt pour React JS, d'autres se montrent sceptiques face à l'arrivée d'Angular 2... et puis il y a ceux qui pianotent sur le clavier de leur téléphone pour envoyer des tweets qui apparaîtront quelques instant plus tard sur le grand écran de l'amphithéâtre...

## Responsabilité et éthique des développeurs

Les lumières s'éteignent et cette dernière Keynote s'ouvre sur le discours de l'avocat américain Richard Fontana. Il soulève l'intéressante question des problématiques juridiques liées au développement, de son impact sur notre quotidien et notre santé, et comment la législation évolue progressivement pour adapter notre société et ses lois à ces nouvelles évolutions technologiques. Son discours se concentre sur la notion d'éthique ancrée depuis de nombreuses décennies dans certains secteurs comme la construction civile, la médecine ou le bâtiment ; les acteurs de ces domaines sont conscients de l'impact de leur travail sur ce qui les entoure, une pratique que l'on ne retrouve pas dans le secteur informatique mais qui pourrait évoluer avec l'Open Source. L'évolution rapide de l'informatique telle qu'elle est dictée par la loi de Moore propulse nos technologies toujours plus loin, à tel point que nos sociétés ne sont pas en mesure de réagir assez vite. Face à ce constat, il y a deux axes possibles pour les développeurs pour agir et accompagner notre société dans sa modernisation : la responsabilité, puisqu'un développeur influence et façonne notre société future, et l'éthique, car même si nous sommes libres de nos actes, nous devons acquérir une certaine éthique pour faire évoluer nos sociétés dans le bon sens.

## L'utopie et les mutations de travail

Puis c'est au tour de Sébastien Broca, sociologue, chercheur et maître de conférences à Paris 8 de nous présenter « Demain » avec un discours sur l'utopie et les mutations du travail dans l'économie numérique. En tant que développeur, il est vrai que nous participons à l'automatisation et l'amélioration des processus et donc à la destruction d'emplois. Cette automatisation massive soulève depuis de nombreuses années des craintes quant à l'avenir du travail. La rétrospective des mutations du travail au cours des deux derniers siècles conduite par Sébastien Broca, permet de se rendre compte que cette crainte n'est pas nouvelle mais bien au contraire qu'elle a toujours été présente lors des grandes avancées industrielles et technologiques de notre civilisation Nous n'assistons donc pas à la fin du travail mais plutôt à l'une "de ses nombreuses transformations historiques". Un constat dont nous devrions peut-être nous réjouir en prenant la littérature utopique et ses alternatives selon lequel la fin du travail pourrait devenir une finalité, un objectif, plutôt qu'une crainte. Même si l'idée peut paraître excentrique, l'exemple des sociétés de la Silicon Valley, comme Google, Red Hat véhiculent une image complètement nouvelle du travail qui a perdu cette connotation de pénibilité avec pour seule motivation la rémunération pour devenir un véritable lieu d'épanouissement, d'accomplissement professionnel et parfois même de détente... le numérique est donc en passe de modifier nos façons de travailler, de collaborer pour un but commun en poussant la société à se réinventer et redéfinir le concept du travail qui s'est au fil du temps éloigné d'une réalité numérique.

## Et si l'informatique prenait le pouvoir sur la politique ?

Enseignant à SciencesPo et journaliste critique, Fabrice Epelboin porte un œil sévère sur les failles de sécurité et nous parle de politique mais aussi d'informatique, et comment l'une influence l'autre et la modifie jour après jour sans que nous en ayant véritablement conscience.

Aujourd'hui l'information reste le processus cognitif fondamental de notre société. Si sur le plan politique cette information est aujourd'hui devenu confuse - on assiste ces dernières années à une profonde incompréhension de la politique par les différentes classes sociales - l'information au sens informatique (la "data") reste l'essence de notre société moderne. Les données produites chaque jour par les utilisateurs ont beaucoup plus de valeur que l'informatique elle-même ou que "les lois qui sont débattues chaque jour à l'assemblée nationale".

Les données modifient la société et la loi et il est intéressant de noter comment l'informatique a pris le pas sur la politique depuis déjà plus de 20 ans : "le code c'est la loi". Alors si on analyse les 3 valeurs fondatrices de notre constitution, il est intéressant de constater que le code a apporté bien plus à la République que la politique : ainsi l'arrivée des réseaux sociaux a repoussé les limites de la Liberté d'expression et Internet a offert une égalité devant l'information, l'accès à la culture et au savoir. Si on s'intéresse maintenant à la fraternité, là encore l'Open Source a renforcé cette valeur en proposant un concept collaboratif rapprochant les développeurs dans un but commun au-delà même des frontières.

La confiance d'un point de vue informatique, repose sur deux notions : la Transparence, portée par l'Open Source, et la Sécurité qui reste aujourd'hui un vrai problème et un axe d'amélioration pour le futur.

## Clôture de la keynote

Le dernier intervenant, Matti Schneider nous parle de société, de gouvernement et de modernité. Pour l'Etat, il incube des services numériques et démontre ainsi qu'un Etat qui "grandit par notre savoir est une société pleine d'espoir". Avec ces nouvelles perspectives sur la manière dont notre société tente d'évoluer, Il clôt la plénière en synthétisant les propos de ses prédécesseurs en matière d'Action Individuelle : "une personne seule peut faire beaucoup", qu'il soit développeur ou non mais également d'Action Collective : "le numérique démultiplie les possibilités de collaborations". Lorsque l'on compare l'Action Publique et le numérique, nous nous rendons compte que l'un fait partie d'un cadre très rigide tandis que l'essence même du numérique est d'être souple et d'avoir la possibilité de se réinventer au quotidien.

Pourtant le cas de l'Estonie démontre que l'on peut conjuguer l'un et l'autre en proposant le service public numérique le plus avancé au monde. En effet il est aujourd'hui possible en Estonie de s'identifier, payer ses impôts en ligne, créer une société. Pour en arriver là, le pays a dû adapter son cadre législatif, pour mieux coller à une réalité numérique. On pourrait croire que ce cas est isolé, mais nombre de gouvernements comme le Royaume Unis ou les USA se sont penchés sur la problématique de simplification des services numériques administratifs. En France, l'administration française tente de s'adapter à une réalité numérique, mais il est à souligner que nous pouvons tous participer à cette « Action Publique » pour construire la société française de demain.

## Conclusion

Il est difficile de synthétiser la richesse des contenus et thèmes de cette conférence de clôture qui aborde le passé, le présent comme le futur de l'informatique au travers de ses mutations et de sa perfectibilité auxquelles nous participons en tant que développeurs. Pour ma part c'est une première et une opportunité intéressante de découvrir des sujets aussi divers et variés gravitant toujours autour de l'informatique sans jamais rentrer dans la technique. Un contrepied intéressant avec les thèmes des conférences du salon plus spécifiques et plus ciblés, qui donne à réfléchir sur l'influence de notre métier chaque jour.

# Azure API Management

*Ecrire une bonne API Web qui répond à un besoin technique et fonctionnel n'est plus vraiment complexe. Mais faire en sorte que cette API soit (bien) utilisée par des tiers peut l'être ! Il faut la sécuriser, la documenter, la packager, voir la commercialiser. Et l'effort de développement pour ceci est conséquent, cela peut doubler le temps de développement initial. Et c'est justement le rôle d'Azure API Management de réduire considérablement cet effort.*

Florent Santin et Thibaut Ranise
**Infinite Square**
http://blogs.infinitesquare.com

## Ecrire et héberger une API

Azure API Management n'est pas un service d'hébergement d'API. Il se positionne en intermédiaire entre l'API et ses utilisateurs, à savoir les développeurs qui vont manipuler l'API. Pour pouvoir l'utiliser, il faut donc dans un premier temps écrire et héberger une API. Pour l'écriture, peu importe le langage utilisé, à partir du moment où l'API est requêtable via les standards HTML, qu'ils soient REST, SOAP ou autre. Pour l'hébergement, API Management peut se connecter sur une API hébergée dans Azure, chez un hébergeur de Cloud tiers ou même sur un serveur On Premise. Pour permettre ce dernier scénario, il est possible de configurer une ligne sécurisée vers le serveur cible, en utilisant par exemple un serveur VPN. Fig.1.

## Le rôle d'API Management

Dans le monde de l'API publique, il y a plusieurs acteurs :

- Le développeur créateur de l'API qui souhaite la publier ;
- Les développeurs externes qui veulent la consommer ;
- Les applications qui veulent pouvoir s'y authentifier pour l'utiliser.

Azure API Management répond au besoin de chaque acteur.

Pour les créateurs d'APIs, il met à disposition un portail d'administration, de configuration et de gestion de la diffusion des APIs.

Pour les développeurs consommateurs d'APIs, un portail pour explorer l'API, lire sa documentation et la tester directement depuis un navigateur Web. Et pour les applications, un proxy serveur / re-routing fournissant un mécanisme d'authentification via clé primaire / secondaire pouvant être invalidée et rafraîchie à la demande. Fig.2.

## Exposer une API

L'administrateur d'API Management dispose d'un tableau de bord détaillé sur l'utilisation de ses API par utilisateur, avec des données détaillées sur le nombre d'appels, le temps de réponse moyen, les erreurs et la bande passante consommée. Fig.3.

Pour exposer une API, il suffit de décrire sa signature d'exposition publique et de la mapper avec la signature de l'API source. A ce niveau, il est possible de documenter l'API, et d'ajouter des fonctionnalités de mise en cache des appels directement au niveau du proxy d'API Management, afin de réduire l'utilisation de l'API source.

Les APIs à exposer peuvent être déclarées une à une à la main, ou bien être importées massivement et automatiquement en utilisant une description WADL ou Swagger. Fig.4.

## Packager une API

Une fois exposée, l'API doit être packagée dans un produit pour être mise à disposition des développeurs.

Un produit contient un ensemble d'API, et définit des droits d'accès et des règles de transformations sur celles-ci. En termes de sécurité d'accès, l'administrateur peut choisir de rendre son produit complètement public, d'imposer une identification pour pouvoir l'utiliser (le développeur doit se créer un compte sur le portail API Management et souscrire à l'API) ou de choisir de gérer les accréditations manuellement via la console d'administration. Sur ce dernier point, le portail API Management fournit tous les workflows pour simplifier au quotidien cette gestion de droits, l'administrateur disposant de tous les outils pour gérer finement ses utilisateurs de produits.

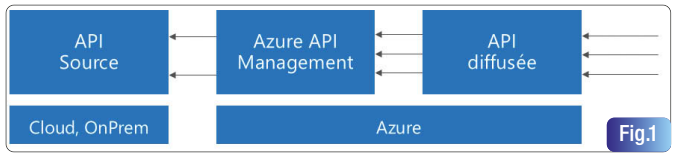


Fig.1

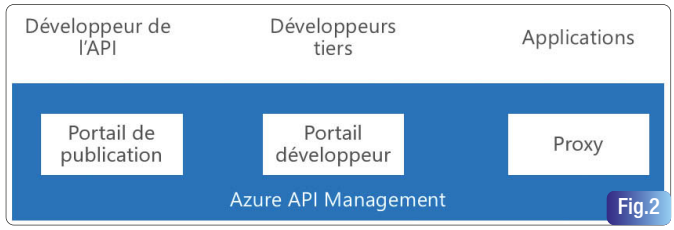


Fig.2

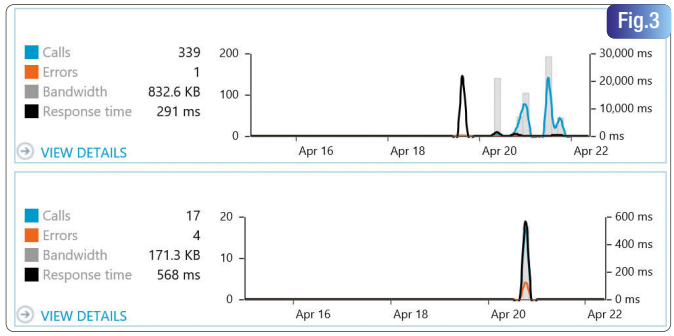


Fig.3

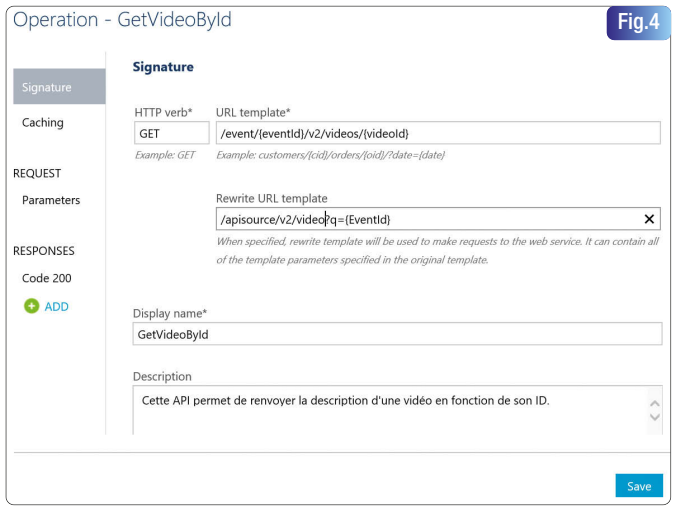


Fig.4

## Configurer une API

Pour aller plus loin, il est également possible d'enrichir un produit avec des « policies », règles de configuration ou de transformation avancées.

Par exemple, il est possible, par configuration, de limiter l'accès aux API d'un produit : brider le nombre d'appels par minute et par développeur, effectuer un filtre IP pour n'autoriser les appels que depuis une seule adresse, autoriser ou pas les appels Cross Domains...

En termes de transformation, là aussi, il est possible d'effectuer des modifications très fines entre l'API exposée et l'appel effectué à l'API source : modification des entêtes HTML, transfert d'un paramètre GET en paramètre POST, transformation d'un appel REST en appel SOAP en effectuant une transformation JSON vers XML... Ces transformations sont très pratiques pour « moderniser » une vieille API et l'exposer, sans modification de son code, selon les standards actuels. Dans un mode de configuration avancé, il est même possible d'écrire ses règles de transformation en script C# (en s'appuyant sur un sous-ensemble du Framework .NET), ce qui étend quasiment à l'infini le panel des possibles.

Ces règles de configuration ou de transformation peuvent être appliquées au niveau de l'intégralité du produit, ou bien finement par API. Elles s'effectuent en éditant un fichier de configurant XML. Fig.5.

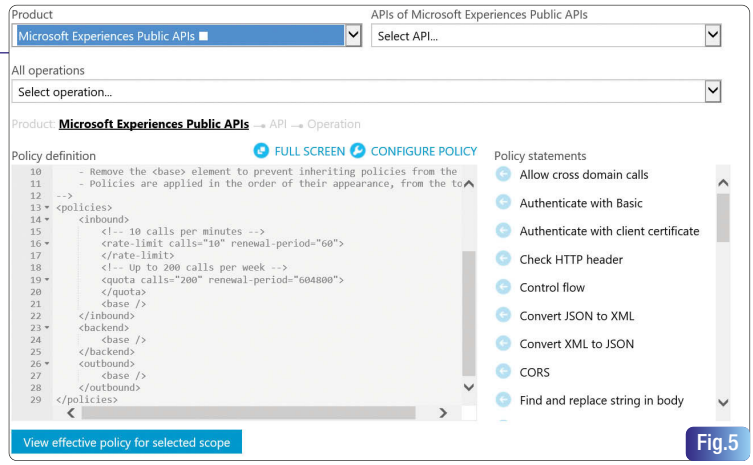


Fig.5

## Pour le consommateur de l'API

Une fois le couple API et produits prêts, il reste à préparer le portail d'accès des développeurs.

Azure API Management génère automatiquement un portail d'accès aux API et produits, avec gestion de l'identification. Etant basé sur le CMS open source Orchard, ce portail peut être 100% personnalisable par l'administrateur, par configuration, sans aucune ligne de code : modifier le style et les couleurs, changer la disposition du contenu, ajouter des widgets...

Le portail développeur permet :

- De se créer un compte de développeur ;
- De consulter la liste des produits et de s'abonner à un produit avant d'obtenir les clés d'accès aux APIs ;
- D'explorer la liste des APIs ;
- De consulter unitaire une API.

Sur ce dernier point, la page de description d'un appel API est très riche. On y retrouve la signature complète de celle-ci, une description des arguments requis pour l'appeler et du résultat renvoyé, ainsi que des exemples de code en Curl, C#, Java, JavaScript, ObjC, PHP, Python et Ruby pouvant être copiés / collés pour instantanément consommer l'API depuis une application. Il est même possible depuis l'interface Web, de tester un appel à l'API. API Management s'occupe de pré-remplir automatiquement la clé d'accès à l'API pour permettre au développeur d'effectuer son appel en modifiant les paramètres d'entrée dans l'interface, ou même directement le contenu de la requête HTTP effectuée. Fig.6.

On notera également qu'il est possible de spécifier le header HTTP nommé « Ocp-Apim-Trace ». Lorsque celui-ci est positionné à « True », API Management effectue les requêtes à votre API et insère, dans les headers, un lien vers un fichier de trace, que vous pouvez utiliser pour debugger/comprendre en cas de problème. Ce fichier de trace contient les appels faits à l'API, les valeurs de retours, la liste des paramètres, les différents headers envoyés/reçus, etc.

## Mode de commercialisation

Azure API Management est un service de la plateforme Microsoft Azure.

Les instances d'API management sont disponibles sous trois modes de licence : développeur, standard, premium, chaque mode augmentant les capacités. Comme tout service Cloud, les capacités de chaque mode sont liées à la consommation, selon plusieurs critères :

- Un coût à la journée ;
- Le nombre d'appels d'API effectués ;

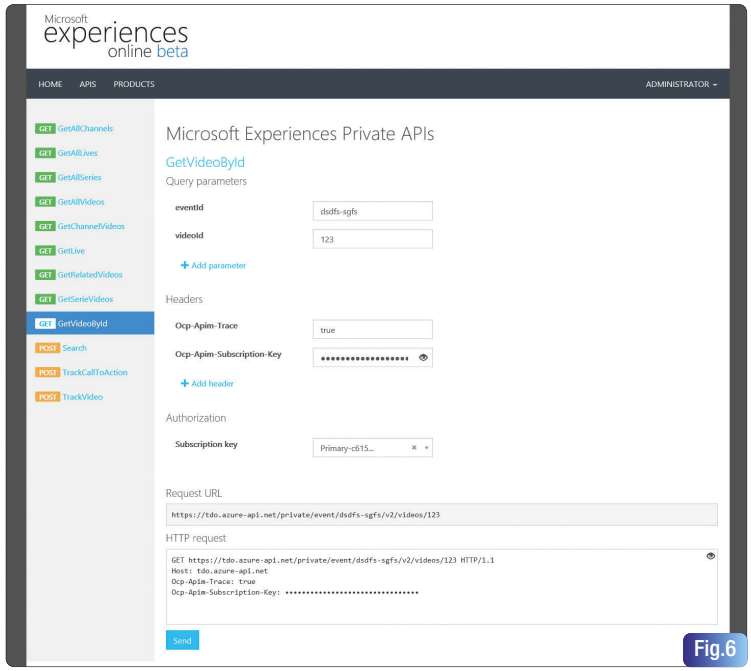


Fig.6

- Le volume de données transférées (bande passante sortante) ;
- Le volume de données mis en cache.

En termes de montée en charge, à partir de la version standard, une instance d'API Management peut absorber plus de 1000 requêtes / seconde. Pour augmenter la capacité, il est possible d'utiliser une version premium (5000 requêtes / seconde) ou tout simplement d'ajouter plusieurs instances en parallèle. La version premium permet également un mode de déploiement sur plusieurs Datacenter dans le monde, pour augmenter les temps de réponse sur des APIs pouvant être consommées sur toutes les plaques géographiques.

## Conclusion

Azure API Management est un vrai accélérateur de mise à disposition d'API, dans tout contexte applicatif.

Pour les applications API à usage grand public, il permet de maîtriser finement la diffusion en offrant de la traçabilité sur l'usage et des fonctionnalités pour brider l'utilisation, afin d'éviter par exemple qu'un développeur tiers n'écroule involontairement l'API source. Il est également fort utile pour les gens souhaitant commercialiser des APIs.

Dans un contexte d'entreprise, il permet de documenter et de centraliser l'ensemble des jeux d'APIs de l'entreprise. Très intéressant par exemple pour évaluer les dépendances et savoir très simplement quelles applications utilisent quelles APIs. Si vous souhaitez tester le portail développeur d'Azure API Management, Microsoft France a mis publiquement à disposition l'accès à l'API de sa plateforme vidéo « Microsoft Experiences » à l'adresse http://tdo.portal.azure-api.net/

# Un JDK enfin modulaire avec Java 9

*Initialement attendue pour 2016, la mouture numéro 9 de Java a finalement été décalée à Mars 2017. Raison invoquée pour justifier ce report, qui devient habituel à chaque nouvelle version de Java, le fameux projet Jigsaw sensé doter Java d'un système de modules performant. En attendant la mise à disposition de Java 9, nous vous proposons un tour d'horizon des principales nouveautés de cette nouvelle version.*

Sylvain SAUREL
*Ingénieur d'Etudes Java / Android*
sylvain.saurel@gmail.com – www.all4android.net

Huit ans après sa création, le projet Jigsaw, visant à rendre modulaire la plateforme Java via l'introduction d'un système de modules performant, est enfin prêt à prendre son envol et à devenir le cœur du JDK. La trajectoire de ce projet aura été plus que sinueuse, et la version cible pour son intégration n'aura cessé de changer passant de Java 7 à Java 8 pour être enfin intégrée dans Java 9. Non sans mal cependant, puisqu'il aura encore fallu décaler d'une année la sortie de Java 9 en la fixant à mars 2017 ! Alors à qui la faute ? A Un manque d'engagement de la part d'Oracle dans la plateforme Java ? Ou peut-être bien à un périmètre qui n'a cessé d'évoluer depuis la création de Jigsaw. Difficile de répondre à cette question avec précision mais l'important est bien ailleurs désormais. Après ces longues années d'attente, la plateforme Java va enfin disposer d'un système de modules lui permettant de franchir un cap et de rester une plateforme moderne.

## Qu'est-ce qu'un Module ?

Au cœur de toute plateforme modulaire se trouve la notion de module. Décrire ce qu'est un module au sein de Jigsaw est assez simple : il s'agit d'une partie d'un logiciel qui doit définir les réponses aux trois questions suivantes au sein d'un fichier nommé module-info.java :

- Quel est le nom de cette partie ?
- Que doit-elle exporter ?
- De quoi a-t-elle besoin ?


*Figure 1 : Un Module basique* Fig.1

La réponse à ces trois questions est donnée respectivement via le nom du module, ce qu'elle exporte et enfin ce dont elle a besoin (Fig.1).
Afin d'éviter tous risques de conflits, il sera préférable de suffixer le nom du module avec le package du programme auquel le module sera rattaché. Pour répondre à la seconde question, le module devra fournir une liste de tous ses packages qui sont considérés comme public API afin de les rendre utilisables par d'autres modules. Ainsi, si une classe ne fait pas partie d'un package exporté, elle ne pourra être utilisée à l'extérieur du module, et ce même si elle est marquée comme public. Enfin, la réponse à la troisième question va consister à déclarer la liste des modules nécessaires au module en cours de définition. Toutes les classes des packages exposés publiquement par ces modules seront alors accessibles par le module en cours de définition.
Il s'agit ici d'une évolution majeure en termes de visibilité des classes puisque depuis la création de Java et jusqu'à Java 8, chaque type public présent au sein du classpath était accessible par n'importe quel autre type. Avec Jigsaw, l'accessibilité des types Java va passer de :

- Public ;
- Private ;
- Default ;
- Protected.

à :

- Public pour tout le monde qui vient lire le module (exports) ;
- Public à des modules particuliers venant lire le module (exports to) ;
- Public à toutes les classes au sein du module lui-même ;
- Private ;
- Default ;
- Protected.

Dans tous les cas, il faudra un certain temps d'adaptation aux développeurs Java pour acquérir la nouvelle logique associée à la visibilité introduite par Jigsaw.

## Un JDK modularisé

Les dépendances entre modules doivent former un graphe acyclique permettant ainsi d'éviter les dépendances circulaires. Afin de mettre en pratique ce principe, les équipes d'Oracle en charge de Java ont eu fort à faire pour modulariser le JDK. En effet, ce dernier comportait jusqu'alors un grand nombre de dépendances circulaires rendant difficile sa compréhension. A la racine du graphe de dépendances du runtime Java se trouve java.base qui est le seul module possédant uniquement des dépendances entrantes. Chaque module créé va lire java.base de manière implicite de la même manière que chaque objet dérive de java.lang.Object par exemple. Le module java.base exportant un certain nombre de packages tels que java.lang, java.util ou encore java.math.
Avec Java 9, les développeurs vont ainsi disposer d'un JDK modulaire ce qui signifie qu'il sera possible de définir précisément les modules du runtime Java nécessaires à la bonne exécution d'une application. En effet, quel intérêt d'avoir un environnement supportant Swing ou Corba pour une simple application réalisant des traitements batchs locaux en arrière-plan ? Nous verrons comment créer l'environnement Java nécessaire à la bonne exécution d'une application par la suite.

## Création d'un Module

La théorie autour de Jigsaw présentée, il est temps de passer à la pratique. Nous allons définir un module ayant vocation de proposer une opération de vérification des codes postaux. Ce module est défini au sein d'un fichier module-info.java :

```
module com.ssaurel.zipvalidator {
  exports com.ssaurel.zipvalidator.api;
}
```

Nommé *com.ssaurel.zipvalidator*, notre module exporte le contenu du package *com.ssaurel.zipvalidator.api* et ne lit aucun autre module excepté java.base de manière implicite. Ce module est lu par le module de vérification d'adresse *com.ssaurel.addresschecker* défini comme suit :

```
module com.ssaurel.addresschecker {
  exports com.ssaurel.addresschecker.api;
  requires com.ssaurel.zipvalidator;
}
```

L'arborescence des fichiers de nos modules est présentée à la Fig.2.
Il faut noter que, par convention, les modules sont placés au sein de répertoires ayant le même nom que le module qu'ils contiennent. Le code source des classes de nos modules n'ayant pas grand intérêt pour le sujet qui nous concerne, à savoir la prise en main de Jigsaw, il n'est pas détaillé dans cet article.

## Utilisation d'un Module

Nos modules définis, il est temps de les compiler afin de pouvoir les utiliser. Dans un premier temps, nous allons compiler le module com.ssaurel.zipvalidator qui ne nécessite aucun autre module pour fonctionner. Cette compilation est réalisée de manière assez classique avec l'exécutable javac et la ligne de commande suivante :

```
javac -d com.ssaurel.zipvalidator \
 $(find com.ssaurel.zipvalidator -name "*.java")
```

La commande find étant utilisée ici simplement pour lister tous les fichiers sources Java au sein du répertoire du module. Maintenant, il faut compiler notre second module qui, lui, a besoin de lire le module com.ssaurel.zipvalidator . Il sera donc nécessaire de donner des précisions au compilateur javac quant à notre structure de modules. Pour ce faire, Jigsaw introduit l'option -modulepath (utilisable via la forme raccourcie -mp également). La compilation du module com.ssaurel.addressechecker est donc réalisée comme suit :

```
javac -modulepath . -d com.ssaurel.addresschecker \
 $(find com.ssaurel.addresschecker -name "*.java")
```

Ici, on précise à javac que les modules compilés sont à chercher au sein du répertoire courant. L'approche est similaire à celle utilisée avec le classpath. Compiler les modules de manière séparée n'est pas forcément très pratique et c'est pourquoi l'option -moduleclasspath a été introduite au sein de javac. La compilation de nos modules peut directement se faire comme suit :

```
javac -d . -modulesourcepath . $(find . -name "*.java")
```

La compilation réalisée avec succès, il est possible d'exécuter notre module et de vérifier si le code postal 13013 est correct :

```
java -mp . -m com.ssaurel.addresschecker/com.ssaurel.addresschecker.api.Run 13013
```

Ici, on définit où Java doit trouver les modules compilés ainsi que le module à appeler et enfin le point d'entrée du programme en y passant en entrée le paramètre attendu.

## Création de Jars Modulaires

Dans la "vraie vie", les programmes Java sont rarement utilisés en fournissant uniquement les classes compilées. Ils sont packagés au sein de fichiers de type Jar. Pour répondre à ce besoin essentiel, Jigsaw introduit la notion de Jar modulaire. Un Jar modulaire est relativement proche d'un Jar classique à cela près qu'il contient un fichier compilé nommé module-info.class en son sein. Pour construire un Jar modulaire nommé zipvalidator.jar, il suffit d'utiliser l'exécutable jar comme suit :

```
jar --create --file bin/zipvalidator.jar \
--module-version=1.0 -C com.ssaurel.zipvalidator .
```

Ici, on définit le nom et l'emplacement du Jar modulaire zipvalidator.jar

```
modules/
├── com.ssaurel.addresschecker
│   ├── com
│   │   └── ssaurel
│   │       └── addresschecker
│   │           ├── api
│   │           │   ├── AddressChecker.java
│   │           │   └── Run.java
│   │           └── internal
│   │               └── AddressCheckerImpl.java
│   └── module-info.java
├── com.ssaurel.zipvalidator
│   ├── com
│   │   └── ssaurel
│   │       └── zipvalidator
│   │           ├── api
│   │           │   ├── ZipCodeValidator.java
│   │           │   └── ZipCodeValidatorFactory.java
│   │           ├── internal
│   │           │   └── ZipCodeValidatorImpl.java
│   │           └── model
│   └── module-info.java
```

Fig.2

*Figure 2 : Arborescence associée à nos Modules*

ainsi que la version du module et le nom du module à packager. Enfin, il est nécessaire de packager le module com.ssaurel.addresschecker :

```
jar --create --file=bin/addresschecker.jar --module-version=1.0 \
--main-class=com.ssaurel.addresschecker.api.Run \
-C com.ssaurel.addresschecker .
```

Il faut noter que le point d'entrée d'un Jar modulaire, à savoir sa classe contenant la fameuse méthode main, n'est pas défini au sein du fichier module-info.java comme cela avait été initialement prévu par l'équipe en charge de Jigsaw, mais au sein du fichier Manifest présent au sein du Jar comme cela est fait habituellement.
Les Jars modulaires générés ayant été placés au sein du répertoire bin, l'exécution du module se fera comme suit :

```
java -mp bin -m com.ssaurel.addresschecker 13013
```

Le point d'entrée du module ayant été défini durant sa création, et écrit au sein du fichier manifest, il n'est pas nécessaire de le préciser au lancement à l'exécutable Java.

## Création de JVMs sur mesure

Parmi les différents nouveaux outils qui seront introduits par Java 9, il en est un essentiel qui permettra aux développeurs de créer leurs propres distributions de JVM sur mesure. Baptisé jlink, ce nouvel outil met à profit la nature modulaire du JDK sous Java 9. Il devient ainsi possible de définir quels modules sont nécessaires à l'exécution d'un programme et d'inclure seulement ces derniers au sein d'une JVM taillée sur mesure. Pour créer une JVM spécifique incluant nos modules, l'outil jlink peut être utilisé comme suit :

```
jlink --modulepath $JAVA9_BIN/../../images/jmods/:modules/bin
   --addmods com.ssaurel.addresschecker --output myjdk
```

Ici, jlink prend en entrée le chemin vers les modules du JDK et nos modules, les modules à inclure et enfin le dossier de sortie de la JVM créée. Le contenu de cette dernière est présenté à la Fig.3.
La création d'un JVM spécifique pour l'exécution de nos modules permet

```
myjdk/
├── bin
│   ├── com.ssaurel.addresschecker
│   ├── java
│   └── keytool
├── conf
│   ├── net.properties
│   └── security
│       ├── java.policy
│       └── java.security
└── lib
    ├── classlist
    ├── jli
    │   └── libjli.dylib
    ├── jspawnhelper
    ├── jvm.cfg
    ├── libjava.dylib
    ├── libjimage.dylib
    ├── libjsig.diz
    ├── libjsig.dylib
    ├── libnet.dylib
    ├── libnio.dylib
    ├── libosxsecurity.dylib
    ├── libverify.dylib
    ├── libzip.dylib
    ├── modules
    │   └── bootmodules.jimage
    ├── security
    │   ├── US_export_policy.jar
    │   ├── blacklisted.certs
    │   ├── cacerts
    │   └── local_policy.jar
    ├── server
    │   ├── Xusage.txt
    │   ├── libjsig.diz
    │   ├── libjsig.dylib
    │   ├── libjvm.diz
    │   └── libjvm.dylib
    └── tzdb.dat
```

Fig.3

*JVM créée sur mesure*

de réduire l'environnement d'exécution à 47 MB. En appliquant un certain nombre d'options de compression offertes par jlink, il est même possible de descendre en dessous des 20 MB !

L'exécutable Java permet également de lister les modules contenus au sein de notre distribution de la JVM comme suit :

```
java –listmods
```

Le résultat obtenu nous montre la présence de nos deux modules ainsi que du module java.base en version 9.0 :

```
com.ssaurel.addresschecker
com.ssaurel.zipvalidator
java.base@9.0
```

Pour exécuter notre module et tester la validité d'un code postal, nous pouvons appeler l'application directement comme suit :

```
./myjdk/bin/com.ssaurel.addresschecker 13013
```

Ce rapide tour d'horizon de Jigsaw nous aura permis de prendre conscience du changement que la modularisation du JDK va engendrer auprès des développeurs d'applications Java. La chaîne de build entière s'en trouve affectée et des outils de build populaires comme Maven ou Gradle ainsi que les IDEs devront également s'adapter. Le processus d'adaptation sera sûrement difficile dans un premier temps mais le jeu en vaut la chandelle puisque la modularité apportée au sein des applications et l'allègement du runtime Java seront des éléments essentiels pour la pérennité de la plateforme Java sur le long terme.

## HTTP/2

Avec la mise en place de Jigsaw, Java 9 est une mouture ayant fait de la modularité son cheval de bataille. Cependant, d'autres fonctionnalités ont également été ajoutées afin de faciliter le travail des développeurs d'applications Java. Le support du nouveau standard HTTP/2 fait partie de celles-ci. La version existante du protocole HTTP, la 1.1, date de 1999 et souffre d'un certain nombre de limitations. La plateforme Java propose un support du protocole HTTP depuis sa version 1.0 mais le support n'est pas orienté API et pas réellement spécifique au protocole HTTP avec notamment la classe URL. De fait, la plupart des développeurs ont recours à une bibliothèque tierce, telle que Apache HttpComponents, pour réaliser des appels HTTP.

Le support de HTTP/2 avec la mise en place d'une API dédiée au sein de Java 9 est donc un pas important pour la plateforme et son avenir à moyen terme. La nouvelle API tire un trait sur le passé, en abandonnant la volonté d'indépendance avec le protocole. A contrario, l'API cible uniquement HTTP en gardant à l'esprit que HTTP/2 ne change pas fondamentalement la sémantique du protocole. Ceci permet d'avoir une API indépendante de la version du protocole. Avec Java 9 et la nouvelle API, une requête HTTP pourra réalisée et gérée comme suit :

```
HttpResponse response = HttpRequest
    .create(new URI("http://www.ssaurel.com/blog"))
    .body(noBody())
    .GET().send();

int responseCode = response.responseCode();
String responseBody = response.body(asString());

System.out.println(responseBody);
```

S'appuyant sur le design pattern Builder, la nouvelle API sera beaucoup plus simple et agréable à utiliser au quotidien pour les développeurs qui pourront se passer de l'utilisation de bibliothèques tierces.

Autre nouveauté sympathique proposée par l'API HTTP/2, la possibilité de réaliser des appels asynchrones gérés dans un thread séparé en arrière-plan par la JVM :

```
HttpRequest req = HttpRequest
    .create(new URI("http://www.ssaurel.com/blog"))
    .body(noBody())
    .GET();

CompletableFuture<HttpResponse> aResp = req.sendAsync();
Thread.sleep(10);

if (!aResp.isDone()) {
  aResp.cancel(true);
  System.out.println("Echec pour obtenir une réponse rapidement ...");
  return;
}

HttpResponse response = aResp.get();
```

## Process API

Depuis sa création, la plateforme Java souffre d'un certain manque pour contrôler et gérer les processus système. Prenons un exemple simple : récupérer le PID du processus courant. Pour réaliser cette opération actuellement, il est nécessaire de recourir à du code natif ou d'utiliser des hacks dépendants de la plateforme cible. En Java 8, cela pourrait se faire comme suit :

```
public static void main(String[] args) throws Exception {
  Process proc = Runtime.getRuntime().exec(new String[]{ "/bin/sh", "-c", "echo $PPID" });

  if (proc.waitFor() == 0) {
    InputStream in = proc.getInputStream();
    int available = in.available();
    byte[] outputBytes = new byte[available];
    in.read(outputBytes);

    String pid = new String(outputBytes);
    System.out.println("PID = " + pid);
  }
}
```

En Java 9, il est désormais possible de réaliser cette opération très facilement avec la mise à jour de l'API Process comme suit :

```
System.out.println("PID = " + Process.getCurrentPid());
```

En sus, cette solution fonctionne sur toutes les plateformes, ce qui est un

avantage énorme. L'API Process va également permettre de récupérer le nom et l'état des processus tout en obtenant un certain nombre d'informations les concernant. On peut ainsi imaginer récupérer tous les processus en cours d'exécution sur le système avec le code suivant :

```
import java.time.Instant;
import java.time.Duration;
import java.time.temporal.ChronoUnit;

public class ProcessAPIDemoUtil{
 public static void printProcessDetails(ProcessHandle currentProcess){
  ProcessHandle.Info currentProcessInfo = currentProcess.info();

  if ( currentProcessInfo.command().orElse("").equals("")){
   return;
  }
  System.out.println("Processus PID: " + currentProcess.getPid());
  System.out.println("Commande Pathname : " + currentProcessInfo.command().orElse(""));

  String[] arguments = currentProcessInfo.arguments().orElse(new String[]{});

  if ( arguments.length != 0){
   System.out.print("Arguments: ");

   for(String arg : arguments){
    System.out.print(arg + " ");
   }

   System.out.println();
  }

  System.out.println("Démarré à : " + currentProcessInfo.startInstant().orElse(Instant.now())
.toString());
  System.out.println("Démarré depuis : " + currentProcessInfo.totalCpuDuration().orElse
(Duration.ofMillis(0)).toMillis() + "ms");
  System.out.println("Propriétaire : " + currentProcessInfo.user().orElse(""));
 }
}

public class Demo{
 public static void main(String[] args){
  // filtrage sur processus ayant une commande / on prend 5 résultats
  ProcessHandle.allProcesses()
   .filter(processHandle -> processHandle.info().command().isPresent())
   .limit(5)
   .forEach((process) ->{
    ProcessAPIDemoUtil.printProcessDetails(process);
   });
 }
}
```

## JShell REPL

Intégré tardivement à la liste des fonctionnalités potentielles, le projet Kulla fera bien partie de Java 9. Il vise à doter la plateforme d'un outil de type REPL (Read-Eval-Print-Loop) permettant d'exécuter des blocs de code Java sans avoir à les intégrer au sein de classes ou méthodes. Ce projet est désormais plus connu sous le nom de JShell. JShell doit faciliter la programmation exploratoire en devenant l'outil idéal pour tester de nouvelles APIs rapidement et permettre à des débutants de découvrir les bases du langage Java. JShell accepte les blocs de codes, les variables, les méthodes, les définitions de classes mais également les imports et les expressions simples. L'avantage étant que tous ces éléments sont évalués directement par JShell. Une fois l'outil JShell lancé au sein d'un terminal, la commande /help est disponible pour lister l'ensemble des options proposées. Il est bon de noter que l'outil propose une complétion du code ce qui est très pratique à l'usage.

Pour commencer, il est possible de définir une simple expression arithmétique puis d'imprimer son résultat :

```
-> 3 * (4 + 5)
| Expression value is: 27
|  assigned to temporary variable $1 of type int

-> System.out.println($1);
27
```

La commande /list permet ensuite de lister l'ensemble des lignes de codes saisies au cours de la session courante de JShell :

```
-> /list

 9 : 3 * (4 + 5)
 10 : System.out.println($1);
```

On peut ensuite définir une variable de type String puis lister l'ensemble des variables de la session courante via la commande /vars :

```
-> String s = "Sylvain Saurel"
| Added variable s of type String with initial value "Sylvain Saurel"

-> /vars
|  int $1 = 27
|  String s = "Sylvain Saurel"
```

La définition de classes ne pose pas plus de problèmes :

```
-> class Pet {}
| Added class Pet

-> class Cat extends Pet {}
| Added class Cat
```

Il est également possible d'utiliser des APIs Java telles que Swing pour tester la construction d'une interface graphique rapidement ou réaliser un appel réseau via la classe URL :

```
-> URL obj = new URL("http://www.ssaurel.com/blog")
| Added variable obj of type URL with initial value http://www.ssaurel.com/blog

-> URLConnection conn = obj.openConnection()
| Added variable conn of type URLConnection with initial value sun.net.www.protocol.http.
HttpUrlConnection:http://www.ssaurel.com/blog

-> conn.getHeaderFields()
| Expression value is : ...
```

Proposant toutes les fonctionnalités attendues d'un outil REPL, JShell se révèlera être un grand outil d'apprentissage ou d'exploration proche de

l'expérience proposée par Scala REPL. En outre, la possibilité de charger des Jars externes via la commande /classpath permettra de tester rapidement et directement des APIs au sein de JShell.

## Divers

Outre ces nouveautés majeures, cette nouvelle version de Java viendra avec son lot de petites évolutions plus ou moins mineures visant soit à améliorer l'usage au quotidien de la plateforme soit à préparer des évolutions futures. Parmi ces nouveautés, on peut citer :

- le positionnement de G1 comme garbage collector par défaut afin de maximiser les performances de la JVM.
- La segmentation du code mise en cache en trois zones distinctes (JVM Internal pour le code devant toujours rester en cache, Profiled code pour le code mis en cache pour une courte durée, Non-Profiled code pour le code mis en cache pour une longue durée) afin d'améliorer les performances de la JVM à l'exécution.
- L'évolution du compilateur javac pour que l'utilisation de tous les cœurs disponibles sur la machine hôte soit généralisée pour tous les projets Java. Cette évolution améliorera sensiblement les temps de compilation.
- La mise à disposition d'options pour mieux contrôler les compilateurs au sein de la JVM pour proposer plus de possibilités de gestion au runtime.
- La mise à jour de l'API Unicode existante pour supporter la version 7.0 du standard Unicode.
- La poursuite du Projet Coin, qui avait fait évoluer la syntaxe du langage avec Java 7, avec quelques légers ajustements tels que l'emploi de variables n'étant pas marquées final au sein des blocs try-with-resources par exemple.
- La définition d'une API de logging basique directement au sein de la plateforme.

Enfin, on pourra regretter le retrait de Java 9 de la JSR 354 visant à intégrer l'API Money and Currency pour doter la plateforme d'une solution standard de gestion monétaire. De même, il avait été proposé de réaliser et d'intégrer une API légère permettant de manipuler des données au format JSON au sein de la plateforme mais cela a été reporté à Java 10 au minimum car jugé non prioritaire par Oracle.

## Conclusion

Avec l'introduction des Lambdas et des Streams, Java 8 avait été une version majeure de la plateforme visant à proposer aux développeurs un support de la programmation fonctionnelle en Java. Le succès aura été au rendez-vous et les impacts nombreux sur la manière dont les développeurs réalisent des applications en Java. Avec la mise à disposition du tant attendu système de module Jigsaw, Java 9 promet également d'impacter considérablement la vie des développeurs d'applications qui devront acquérir au plus vite la logique de modules associée à ce nouveau système. Ces modifications en profondeur du runtime Java vont également permettre à la plateforme de rester compétitive à moyen terme en offrant des performances accrues au runtime. Il reste désormais simplement à espérer que la date de mise à disposition soit tenue et que Java 9 sorte bien en Mars 2017. ⊞

## ET JAVA EE ?

Sur Java 9, nous l'avons vu, la situation semble désormais claire : disponibilité le 23 mars, sauf gros problème de dernière minute. Oracle avait dévoilé en 2012 une longue roadmap sur la JDK : 9 (2015), 10 (2017), 11 (2019), 12 (2021). Désormais, tout est décalé d'au moins 2 ans.

Sur la partie JDK /Java SE, la situation semble claire et la communauté est très attentive (OpenJDK). Par contre, sur la partie Java Enterprise Edition (Java EE), depuis quelques mois, la situation se tend et le silence d'Oracle n'a pas aidé à clarifier la situation. Oracle soutient-il le développement de Java EE ? JEE est une plate-forme critique pour de nombreuses entreprises car elle est à cœur des infrastructures logicielles. Un long article d'Ars Technica rappelle les méandres de JEE chez Oracle et des travaux qui ne sont plus réalisés dans certaines spécifications (ex. : le prochain JSF) et les spécifications du prochain JEE qui patinent. Oracle a-t-il réorienté les ingénieurs et développeurs sur d'autres projets ?

La mauvaise tournure du procès contre Google autour de Java a-t-elle influencé Oracle dans son attitude sur JEE ? Le silence de l'éditeur alimente les craintes et une pétition, Java EE Guardians, a été lancée pour demander à Oracle de relancer le développement ou de libérer la plate-forme. On attend la conférence JavaOne de septembre pour en savoir plus.

La rédaction

# C# 7 : les futures nouveautés

*Le compilateur Roslyn étant disponible en open source sur la plateforme GitHub, il est possible de connaître les spécifications des langages supportés par ce dernier. D'ailleurs C# 6 est à peine sorti ; il y a moins d'un an que les spécifications de la prochaine version C# 7 apparaissent déjà. En effet, cela permet à la communauté des développeurs de suivre, de proposer ou même de participer à l'élaboration de cette nouvelle monture.*

Thibaut RANISE
tranise@infinitesquare.com
http://blog.infinitesquare.com/b/tranise

Daniel DJORDJEVIC
ddjordjevic@infinitesquare.com
http://blog.infinitesquare.com/b/ddjordjevic

Maxime CAROUL
mcaroul@infinitesquare.com
http://blogs.infinitesquare.com/b/mcaroul

La responsabilité de la conception du C# 7 est composée d'une petite équipe menée par Anders Hejlsberg en tant que chief language architect. Lorsque l'équipe décide de faire avancer une idée de fonctionnalité, un owner est désigné parmi les membres de l'équipe pour être responsable de rassembler les feedbacks, suivre les avancements et maintenir un document décrivant la fonctionnalité et son état actuel.
Il est important de noter que l'équipe de conception du langage C# est toujours en charge du langage. Les commentaires sur GitHub et les votes sur le UserVoice sont suivis avec une très grande attention mais la décision finale reste de leur côté. Ce n'est pas un processus démocratique dans lequel une fonctionnalité qui obtiendrait un large soutien de la part de la communauté serait forcément implémentée.
L'objectif n'est pas d'empêcher ou de brider l'évolution du langage mais au contraire de veiller à ce que le C# reste élégant, cohérent sur le long terme. A l'heure actuelle, il est possible de distinguer trois types de statuts pour chaque fonctionnalité suivie par l'équipe de conception :

- Celles déjà disponibles ;
- Celles en prototypages ;
- Celles sous spécifications.

## Fonctionnalités déjà disponibles

### Fonctions Locales

La version 7 de C# permettra de définir une fonction à l'intérieur du scope d'une autre fonction, on pourra parler de fonction « Parent » et de fonction(s) « Enfants ». Actuellement, il est d'usage de créer des méthodes « privées » qui sont utilisées par des méthodes « publiques ». Cependant ces méthodes privées ne sont bien souvent utilisées que par une seule méthode publique : ainsi une méthode privée B est utilisée uniquement par une méthode publique A, leur couplage est fort et le fait de créer une seule méthode publique A avec une fonction locale rend inutile l'implémentation d'une méthode supplémentaire. En C#6, il est possible de mettre en place ce genre de scenario en utilisant une expression lambda, cependant ces dernières sont moins lisibles et sont moins performantes car en arrière-plan elles instancient puis appellent un « delegate ». Dans l'exemple suivant une fonction locale a pour responsabilité de construire une url.

```
private static Uri CheckAndBuildUri(string serverAdress, int serverPort, string controllerName, string
 actionName)
{
  if (!serverAdress.StartsWith("http"))
  {
    throw new ArgumentException();
  }

  if (serverPort < 80 && serverPort > 90)
  {
    throw new Exception("Invalid port !");
  }

  // Local function
  Uri FormatUrl (string baseAdress, int port, string route)
  {
    return new Uri($"{baseAdress}/{port}/{route}");
  }

  return FormatUrl(serverAdress, serverPort, $"{controllerName}/{actionName}");
}
```

### Literal binary et digit separator

Fonctionnalité mineure mais utile, nous avons maintenant la possibilité d'écrire des littéraux numériques sous forme binaire. Cette fonctionnalité peut se révéler utile à des fins d'apprentissage et pour rédiger des masques de bits. De plus, pour améliorer la visibilité des grands nombres décimaux, hexadécimaux et binaires, des séparateurs permettent de grouper les chiffres :

```
var values = new[] {0b11001011, 0b01100001, 0b01010111, 0b1_000_1000};
bool isPair = values.Sum() % 2 == 0;
```

### Ref local et ref returns

Historiquement, pour travailler avec les pointeurs en C#, il est d'usage d'utiliser du code « unsafe » ou de passer des paramètres par référence en utilisant le mot clef « ref », cette fonctionnalité s'étend maintenant aux variables locales et valeurs de retours. Une méthode est donc capable de renvoyer un « pointeur » vers un emplacement mémoire, on parle alors de « ref return ». Pour stocker une « ref return » (qui est une adresse mémoire) il faut utiliser une « ref locale ».
Dans l'exemple ci-dessous, une méthode locale « GetNameByRef » renvoie l'adresse mémoire d'une valeur d'un tableau et non pas une copie de la valeur du tableau spécifiée.

```
static void Main(string[] args)
{
  ref string GetNameByRef(int index, string[] namesTable)
  {
    return ref namesTable[index];
  }

  var names = new[] {"Thibaut", "Daniel", "Maxime"};
```

```
    // use a "ref local" to store the "ref return"
    ref string name = ref GetNameByRef(1, names);

    name = "Teddy";

    Console.WriteLine(names[1]);
    Console.Read();
}
// Output : "Teddy"
```

### Pattern matching

Dans les langages fonctionnels comme F# ou Scala le « pattern matching » n'a rien de nouveau, cependant en C# c'est une fonctionnalité qui est attendue depuis longtemps ! Le pattern matching est une fonctionnalité qui s'utilise avec l'opérateur « is » : lors d'un test sur le type d'une variable avec l'opérateur « IS », il est maintenant possible de déclarer une variable à droite du type évalué pour récupérer la valeur de l'objet qui est testé si le résultat du test « true » :
Dans le cas suivant, on teste le type de la variable « age » et on effectue un traitement différent si la variable est de type « string » ou « int » en récupérant la valeur de l'objet testé dans une variable « i » :

```
object age = getAgeValue();

if(age is string i)
{
    Console.WriteLine($"I'm {i} years old !");
}
else if(age is int i)
{
    Console.WriteLine($"Next year I will be {i + 1} years old !");
}
```

Ce pattern matching est utilisable à l'intérieur de l'instruction « switch », et le mot clef « when » permet de spécifier une condition supplémentaire en plus du test effectué sur le type comme ceci :

```
object v = getAgeValue();
switch (v)
{
    case int i when i < 5:
        Console.WriteLine($"You are {v}, you are young");
        break;
    case int i when i > 5 && i < int.MaxValue:
        Console.WriteLine($"You are {v}, it's time to learn C# !");
        break;
    default:
        Console.WriteLine($"You are {v}");
        break;
}
```

L'utilisation de l'instruction « switch » est relativement verbeuse, un nouveau mot clef « match » permet d'alléger le code et faire du pattern matching sous forme d'expression. Dans l'exemple suivant, on teste directement le type d'une variable nommée « age » au sein d'une expression, d'ailleurs le mot clef « let » est utilisé, c'est ici un simple sucre syntaxique qui permet de ne pas explicitement réassigner la variable « age » avec un var, l'utilisation de let va créer en interne un « readonly var », il sera donc impossible d'assigner une nouvelle valeur à la variable « ageInformations ».

```
object age = getAgeValue();

let ageInformations = age match
  (
    case int i : $"I'm {i} years old"
    case DateTime d : $"Date of birth : {d.ToShortDateString()}"
    case * : throw new ArgumentException()
  );

Console.WriteLine(ageInformations);
```

Enfin, il est intéressant de mettre en avant l'expression « throw », cette dernière est maintenant utilisable dans de nouveaux contextes :

- Comme second ou troisième paramètre d'une opération ternaire (voir exemple ci-dessous) ;
- Comme second paramètre après l'opération « coalescing » null ;
- À la fin d'une expression « match » (dans une clause « else » par exemple) ;
- Comme body d'une méthode ou d'une expression lambda.

```
int day = 1;
int month = 2;
string result = (day == 11 && month == 2) ? "Birthday !" : throw new Exception("Wrong birthday date");
```

## Fonctionnalités prototypées

### Tuples

#### *Un nouveau concept*

Il arrive souvent en tant que développeur d'avoir besoin d'un set de données typées et temporaires, sans avoir une certaine logique permettant de considérer cet ensemble comme un modèle, et donc de créer un type rien que pour cela. En se basant sur des concepts de Tuples venant d'autres langages, le C# pourrait envisager une approche similaire. La situation la plus courante, une liste de paramètres pour une méthode, possède un support syntaxique en C#. Cependant, l'autre situation la plus courante, une liste de résultat, n'est pas supportée.

#### *Etat des lieux*

Pour un tel comportement, nous avons la possibilité suivante pour l'instant :

```
public void Compute(IEnumerable<int> values, out int sum, out int count){…}

int sum, count;
Compute(values, out sum, out count);
```

Cependant, cette approche n'est pas utilisable dans des méthodes asynchrones, et c'est assez déplaisant à utiliser. Il faut dans un premier temps déclarer les variables, ensuite les passer en paramètre « out », et enfin on peut les consommer. Le côté positif, c'est que les variables sont nommées, ce qui permet d'éviter toute confusion. Sinon, il est possible d'utiliser les System.Tuple actuels :

```
public Tuple<int,int> Compute(IEnumerable<int> values){…}

Tuple<int,int> result = Compute(values);
```

Ici, ça fonctionne avec des méthodes asynchrones, et il ne faut que deux déclarations pour consommer la donnée. Cependant, l'inconvénient majeur, c'est qu'on doit utiliser Item1, Item2, ... qui n'indiquent pas clairement ce que contient la donnée. De plus, instancier un objet de type Tuple a un coût. Une troisième méthode est possible, en créant un type pour nos données de retour :

```
public struct ComputeResult { public int Sum; public int Count; }
public ComputeResult Compute(IEnumerable<int> values){…}
ComputeResult computeResult = Compute(values);
```

Cette fois ci, nous avons quelque chose de plus intéressant. On peut l'utiliser avec de l'asynchrone, les noms des champs sont explicites, et nous évitons tout coût supplémentaire étant donné que l'on manipule une structure. Le seul désavantage à l'utilisation de cette méthode est la déclaration même du type. La déclaration elle-même est insignifiante en soi, mais la donnée n'étant pas forcément un concept très précis, il peut être compliqué de trouver un nom concis et représentatif.

### *Les types Tuple*

Si le cas d'utilisation le plus courant est d'avoir plusieurs résultats, il semble raisonnable d'aborder une certaine symétrie entre la liste des paramètres et celle des arguments. Les types Tuple seraient introduits avec une syntaxe très similaire à une liste de paramètres :

```
public (int sum, int count) Compute(IEnumerable<int> values) {…}

var t = Compute(myValues);
Console.WriteLine($"Sum: {t.sum}, count: {t.count}");
```

La syntaxe `(int sum, int count)` représente une structure anonyme possédant des champs publics avec le nom et le type spécifié. On oublie donc le concept d'ordre (Item1, Item2, etc...) et on a beaucoup plus de clarté dans le code. Bien entendu, on pourrait wrapper le tout dans de l'asynchrone en renvoyant une Task :

```
public async Task<(int sum, int count)> Compute(IEnumerable<int> values) {…}
```

Sans addition supplémentaire au langage C#, nous pourrions créer un Tuple de la sorte :

```
var t = new (int sum, int count) { sum = 0, count = 0 };
```

Dans la proposition originale, l'auteur met en avant le besoin d'une syntaxe particulière pour les Tuples, ce qui semble cohérent vu la déclaration supposée qui est verbeuse. La première proposition serait :

```
public (int sum, int count) Compute(IEnumerable<int> values)
{
    var s = 0; var c = 0;
    foreach (var value in values) { s += value; c++; }
    return (s, c);
}
```

Ensuite, en utilisant les arguments nommés comme une analogie de syntaxe, il pourrait être possible de donner des noms de champ de Tuple directement :

```
public (int sum, int count) Compute(IEnumerable<int> values)
{
    var res = (sum: 0, count: 0);
    foreach (var value in values) { res.sum += value; res.count++; }
    return res;
}
```

### *Déconstruction*

Enfin, le dernier point est la déconstruction du Tuple. Usuellement, le groupement des données est « forcé » et le développeur ne veut pas manipuler le Tuple en tant qu'objet. A la place, il faudrait récupérer immédiatement les composants de ce Tuple. De manière générale, les langages proposant des Tuples utilisent une syntaxe de déconstruction pour recevoir et séparer le Tuple en données unitaires :

```
(int sum, int count) = Compute(myValues);
Console.WriteLine($"Sum: {sum}, count: {count}");
```

## Fonctionnalités en prototypage

### ValueTask

En C# 6, les méthodes comportant le mot clé async dans leur signature ne peuvent retourner que void, Task ou Task<T>. Le type ValueTask<T> propose de retourner une Task<T> uniquement lorsque c'est nécessaire, la motivation ici est uniquement de l'ordre de la performance. Prenons l'exemple d'une interface qui définit une méthode de repository comme ci-dessous.

```
public interface IOrderRepository
{
    Task<Order> GetByIdAsync(int id);
}
```

Si son implémentation est asynchrone jusque-là rien d'anormal :

```
public class DbOrderRepository : IOrderRepository
{
    private readonly MyDbContext _db;

    public DbOrderRepository(MyDbContext db)
    {
        _db = db;
    }

    public async Task<Order> GetByIdAsync(int id)
    {
        // Récupération asynchrone des données
        return await _db.Orders.FirstOrDefaultAsync(o => o.Id == id);
    }
}
```

Par contre si celle-ci est synchrone alors à chaque appel de cette méthode, une tâche est créée inutilement :

```
public class InMemoryOrderRepository : IOrderRepository
{
    private readonly IList<Order> _orders;

    public InMemoryOrderRepository()
    {
        _orders = new List<Order>();
        //...
    }

    public Task<Order> GetByIdAsync(int id)
    {
        // Récupération synchrone des données
        return Task.FromResult(_orders.FirstOrDefault(o => o.Id == id));
    }
}
```

L'intérêt principal du type ValueTask<T> est de réduire l'overhead associé à l'instanciation d'une tâche sur les portions de code synchrone qui sont exécutées très fréquemment. Attention cependant, celui-ci n'a pas pour vocation de remplacer le type Task<T> qui reste d'ailleurs conseillé comme type de retour par défaut d'une méthode asynchrone. Néanmoins, sachez qu'ASP.Net utilise déjà en interne le type ValueTask<T> et que celui-ci est déjà disponible dans la librairie System.Threading.Tasks.Extensions en version prerelease.

### Private protected

En C#, il est déjà possible de spécifier deux modificateurs d'accès à la suite comme « protected internal » qui correspond à l'union du « protected » et de l'« internal » pour autoriser l'accès aux types déclarés à la même assembly ou à travers une classe dérivée dans une autre assembly. Désormais il est proposé de gérer également le modificateur d'accès « private protected » qui serait l'association des deux et permettrait d'accéder à un membre pour les types dérivés de la classe contenus au sein d'une même assembly.
A noter que les structures ne supportent pas l'héritage, par conséquent l'accessibilité d'un membre d'une structure ne pourra pas être protected, protected internal ou même **private protected**.

## Fonctionnalités en cours de spécification

### Async Main

De nombreux programmes, surtout les petits que l'on écrit rapidement, sont parfois entièrement asynchrones. Actuellement, les développeurs doivent créer eux-mêmes un pont entre le synchrone et l'asynchrone, car la méthode Main doit être synchrone. On a donc affaire à des solutions du genre :

```
static async Task WaitAsync()
{
   await Task.Delay(1000);
}

static void Main()
{
   WaitAsync().GetAwaiter().GetResult();
}
```

Devoir faire cela soi-même peut être une source d'erreur et est une perte de temps. Par exemple, si on utilisait WaitAsync().Wait(), les exceptions levées dans notre méthode seraient wrappées dans une AggregateException. Ecrire WaitAsync() sans Wait ni GetResult ferait que le programme pourrait se fermer avant même que la méthode ait fini son travail asynchrone.
Au lieu d'obliger le développeur à gérer cela de lui-même, l'idée serait de proposer une méthode Main asynchrone.
Les signatures proposées seraient :

```
async Task Main()
async Task<int> Main()
async Task Main(string[])
async Task<int> Main(string[])
```

Comme vous le savez sûrement, la CLR n'accepte pas les méthodes asynchrones, ce serait donc le compilateur C# qui s'occuperait de générer le code correspondant :

```
async Task<int> Main(string[] args)
{
   // Code asynchrone du développeur
}

int GeneratedMain(string[] args)
{
   return Main(args).GetAwaiter().GetResult();
}
```

Un problème de rétrocompatibilité serait soulevé ici, car nous aurions plusieurs points d'entrée disponibles. Il faudrait alors que le compilateur recherche un Main asynchrone uniquement s'il ne trouve pas de Main synchrone. Si un Main synchrone est trouvé, alors un avertissement apparaitrait pour le Main asynchrone qui serait trouvé à posteriori.

### Records

Le concept de record est une nouvelle de façon de déclarer une classe ou une structure simplement. Pour être compatible, il faut que le type contienne uniquement des propriétés comme ci-dessous :

```
public class Employee
{
   private int _id;
   private string _name;

   public Employee(int id, string name)
   {
      _id = id;
      _name = name;
   }

   public int Id => _id;
   public string Name => _name;
}
```

Pour sa déclaration en record il suffirait d'écrire une seule ligne pour avoir le même résultat qu'au-dessus :

```
class Employee(int Id, string Name);
```

Le but ici est de réduire la quantité de code que le développeur doit écrire. Au final, la classe générée implémentera automatiquement :

- Le constructeur ;
- Les propriétés en lecture seule ;
- L'interface IEquatable<Employee> et la méthode ToString() retournera les valeurs du record ;
- La méthode ToString() ;
- Le support « **is** » du pattern matching vu précédemment.

### With Exprs

En association avec les records, le support des expressions **with** permet l'instanciation d'un nouvel objet basé sur l'existant comme par exemple :

```
Employee emp1 = new Employee(1, "Ranise");
Employee emp2 = emp1 with { Name = "Djordjevic" };
```

Le code généré ressemblera à ceci :

```
Employee emp2 = new Employee(name: "Djordjevic", id: emp1.Id);
```

## Conclusion

En conclusion, C# 7 ne révolutionnera pas notre manière de programmer mais cette nouvelle version apporte son lot de nouveautés. D'ailleurs celles-ci ne sont pas uniquement d'ordre "sucre syntaxique" mais apportent également un gain de performance comme le type ValueTask<T>.
Pour suivre d'encore plus près l'élaboration de cette future version de C#, nous vous invitons à suivre la page https://github.com/dotnet/roslyn/blob/master/docs/Language%20Feature%20Status.md qui relate le statut et la planification de chaque fonctionnalité suivie du langage.

# Mon premier jeu Cross Platform avec CocosSharp

1ère partie

*CocosSharp est un framework de développement de jeux 2D multiplateformes (Android, iOS, et Windows). Il s'agit du portage XAMARIN du projet open source Cocos2D intégrant un moteur physique et une gestion poussée des animations, des sprites, des transitions, ... Et quoi de mieux qu'un petit jeu rétro pour découvrir un peu plus cet univers ?*

François Botte
*Microsoft Technical Lead,*
**SQLI Toulouse**

## Rétrogaming : retour vers le futur...

Les moins jeunes d'entre nous doivent se souvenir des premiers jeux électroniques de poche produits par Nintendo dans les années 80. CocosSharp pour XAMARIN semble être le bon choix pour moderniser l'un des plus célèbres des jeux de la firme japonaise : « FIRE » : Fig.1.

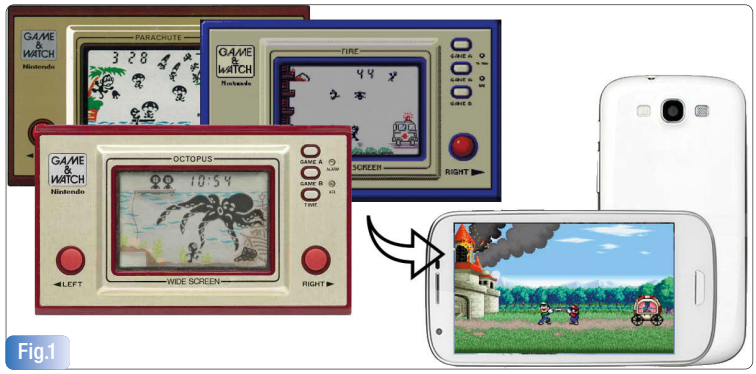


Fig.1

## Etape 1 : Initialisation du projet Android

Une fois les prérequis complétés, lancez Microsoft Visual Studio et créez un nouveau projet.

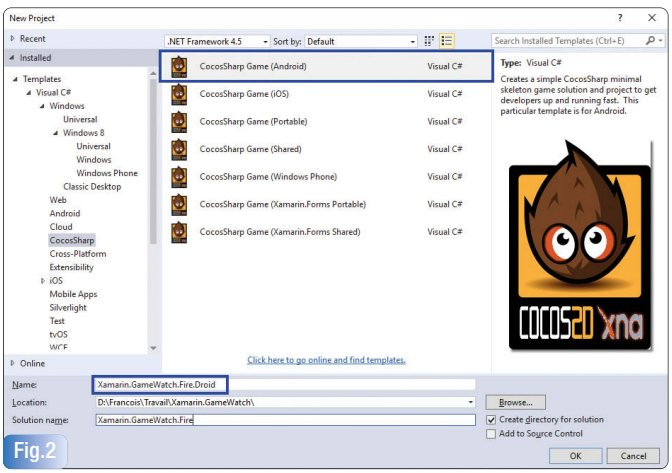


Fig.2

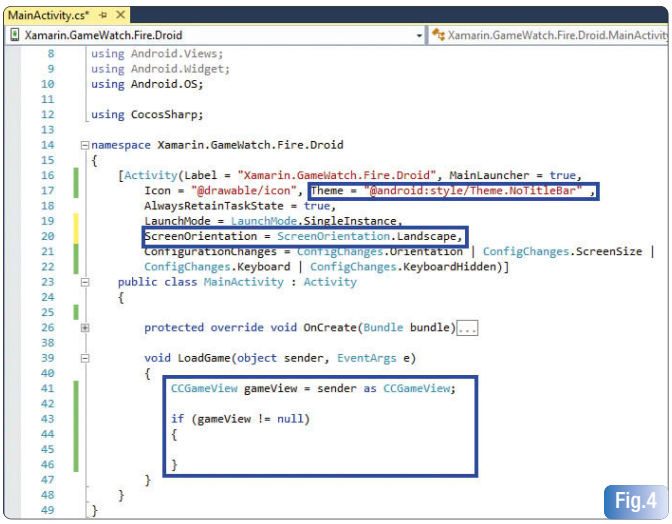


Fig.4

Dans la fenêtre de paramétrage du projet, sélectionnez le template pour CocosSharp (Android) et configurez les noms et le répertoire de votre solution (par convention, les projets ciblant Android se terminent par « .Droid ») : Fig.2.

Une fois la solution créée, votre explorateur de projets se compose ainsi (notez la présence des librairies CocosSharp, MonoGame.Framework et Mono.Android) : Fig.3.

Commençons par faire un peu de ménage en supprimant le fichier « *GameLayer.cs* », nous le rajouterons plus tard dans la solution.

Editez le fichier « *MainActivity.cs* » afin de lui attribuer le style qui permet de supprimer la barre de titre. Définissez l'orientation de l'écran de notre jeu afin qu'il soit toujours orienté en paysage et modifiez la procédure LoadGame afin qu'elle ressemble à celle-ci : Fig.4. La partie Android est maintenant prête à recevoir notre projet partagé qui contiendra toute la logique du jeu.

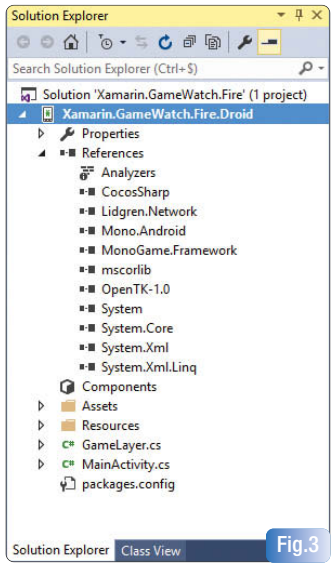


Fig.3

## Etape 2 : Initialisation du projet iOS

Rajoutez un nouveau projet à la solution existante (clic droit dans l'explorateur de solution) : Fig.5.

Dans la fenêtre de paramétrage, sélectionnez le template pour CocosSharp (iOS) et configurez le nom du projet (par convention, les projets ciblant iOS se terminent par « .iOS »).

Comme pour le projet Android,

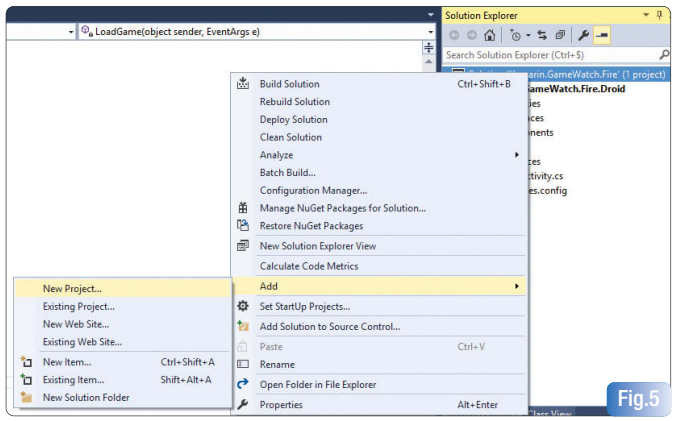


Fig.5

commençons par faire un peu de ménage en supprimant les fichiers ci-dessous : Fig.6.

Ouvrez la fenêtre des propriétés en cliquant sur le projet (clic droit) et sélectionnez l'onglet « iOS Application » afin de sélectionner l'orientation paysage uniquement et cibler la version de déploiement : Fig.7.

Editez dans le fichier « *ViewController.cs* » la procédure LoadGame afin qu'elle ressemble à celle ci-dessous : Fig.8. La partie iOS est maintenant prête elle aussi à recevoir notre projet partagé.

## Etape 3 : Initialisation du projet partagé

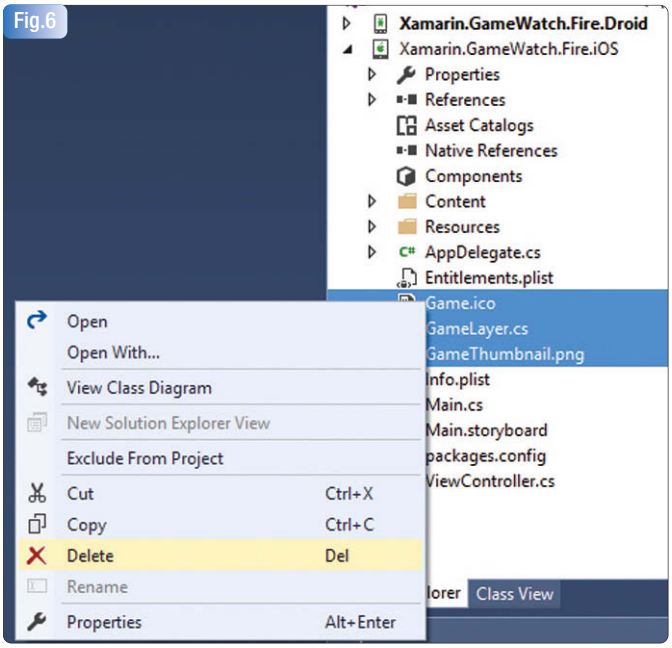
Rajoutez un nouveau projet à la solution existante (clic droit dans l'explorateur de solution) : Fig.9.

Le projet partagé va contenir toute la logique de notre jeu. Android et iOS utiliseront donc le même code mutualisé afin d'afficher le rendu graphique et gérer la physique. Rajoutez une nouvelle classe « *GameController.cs* » à ce projet. Cette classe statique va nous permettre de définir la résolution de notre écran et initialiser la première scène de notre jeu. Elle permet aussi de définir l'endroit où sont stockées nos ressources (animations, pixels, sprites, ...). Editez le fichier « *GameController.cs* » pour rendre la classe publique et statique. Rajoutez une propriété statique « GameView » de type « CCGameView » (toutes les classes, méthodes, propriétés, ... de type CocosSharp commencent par CC...). « CCGameView » est le point d'entrée de l'application CocosSharp. C'est elle qui gère les animations, les transitions, les effets sonores, ... et surtout le lancement du jeu : Fig.10.

**Note :** *nous n'avons pas encore fait référence à notre projet partagé, ne vous inquiétez donc pas pour l'instant des références à utiliser dans la partie des déclarations de notre classe.*

Enfin, il ne nous reste plus qu'à initialiser notre contrôleur en lui indiquant la résolution à utiliser, la localisation des ressources et la scène à démarrer : Fig.11.

Fig.6



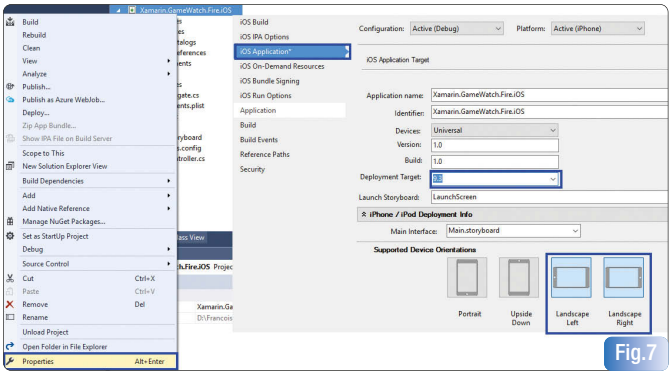



Fig.7

Fig.8

```
namespace Xamarin.GameWatch.Fire.iOS
{
    public partial class ViewController : UIViewController
    {
        public ViewController(IntPtr handle)...
        public override void ViewDidLoad()...
        public override void ViewWillDisappear(bool animated)...
        public override void ViewDidAppear(bool animated)...
        public override void DidReceiveMemoryWarning()...
        void LoadGame(object sender, EventArgs e)
        {
            CCGameView gameView = sender as CCGameView;
            if (gameView != null)
            {
            }
        }
    }
}
```

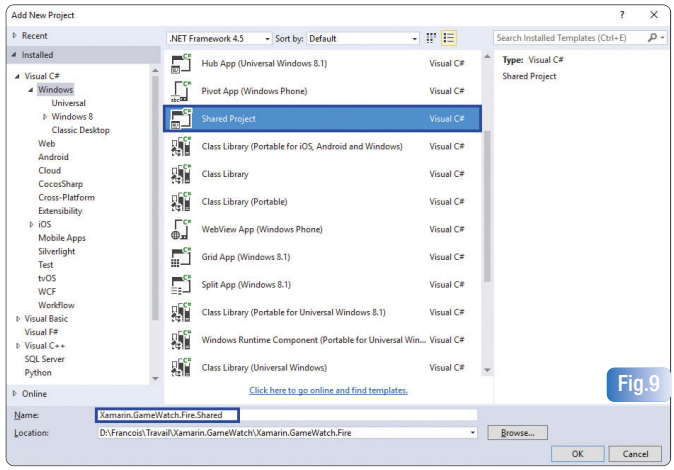



Fig.9

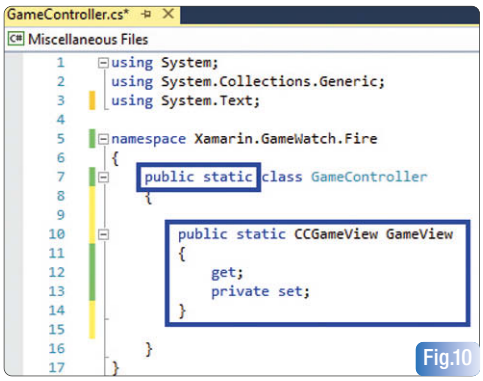


Fig.10

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Xamarin.GameWatch.Fire
{
    public static class GameController
    {
        public static CCGameView GameView
        {
            get;
            private set;
        }
    }
}
```

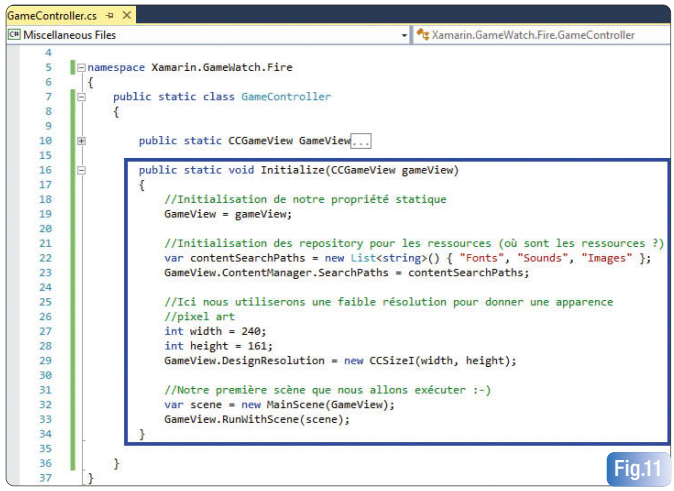


Fig.11

```
namespace Xamarin.GameWatch.Fire
{
    public static class GameController
    {
        public static CCGameView GameView...

        public static void Initialize(CCGameView gameView)
        {
            //Initialisation de notre propriété statique
            GameView = gameView;

            //Initialisation des repository pour les ressources (où sont les ressources ?)
            var contentSearchPaths = new List<string>() { "Fonts", "Sounds", "Images" };
            GameView.ContentManager.SearchPaths = contentSearchPaths;

            //Ici nous utiliserons une faible résolution pour donner une apparence
            //pixel art
            int width = 240;
            int height = 161;
            GameView.DesignResolution = new CCSizeI(width, height);

            //Notre première scène que nous allons exécuter :-)
            var scene = new MainScene(GameView);
            GameView.RunWithScene(scene);
        }
    }
}
```

Notre classe est prête et exécutera notre première scène « *MainScene* ». Rajoutez une nouvelle classe « *MainScene.cs* » au projet partagé : Fig.12.
Une scène dans CocosSharp est comme une scène au théâtre. Il y a des décors, des acteurs qui dialoguent et se déplacent sur l'estrade, de la musique, etc. Et pour organiser tout cela, il faut un metteur en scène qui dirigera les éléments qui feront de la pièce un succès ou non. Ici, notre metteur en scène serait la classe « *GameController.cs* » et la scène serait la classe « *MainScene.cs* » : Fig.13.
Enfin, pour rajouter des acteurs à notre scène, il faut une sorte de décor ou d'estrade pour pouvoir placer les éléments. C'est la variable privée « layer » de type « CCLayer » ajoutée lors de l'initialisation de notre scène qui s'en occupera. C'est peut-être le moment de tester tout cela sur nos appareils afin d'avoir le rendu de notre jeu. Rajoutez donc un acteur pour avoir une représentation visuelle de notre scène : Fig.14.

## Etape 4.1 : Testez notre jeu sous Android

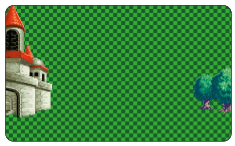
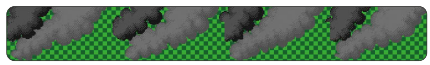
Rajoutez au projet « Xamarin.GameWatch.Fire.Droid » la référence au projet partagé : Fig.15 et 16. Une fois fait, rajoutez dans la procédure « *LoadGame* » de la classe « *MainActivity.cs* » du projet « Xamarin.Game Watch.Fire.Droid » l'appel à notre contrôleur du projet partagé comme ceci : Fig.17. Enfin, il ne reste plus que les références manquantes à rajouter à nos deux classes « *MainScene.cs* » et « *GameController.cs* » : Fig.18.
Exécutez l'application Android sur votre appareil ou simulateur.

## Etape 4.2 : Testez notre jeu sous iOS

Rajoutez au projet « Xamarin.GameWatch.Fire.iOS » la référence au projet partagé (exactement de la même manière que le projet pour la version Android ). Une fois fait, rajoutez dans la procédure « *LoadGame* » de la classe « *ViewController.cs* » du projet « Xamarin.GameWatch.Fire.iOS » l'appel à notre contrôleur du projet partagé comme ceci : Fig.19.
Exécutez l'application iOS sur votre appareil ou simulateur.

## Etape 5 : Création du décor animé

Dans cette partie, nous allons créer le décor de notre jeu. Il se constitue de plusieurs parties (les layers) que nous allons superposer et animer si nécessaire. Nous allons découvrir deux nouveaux types de classe CocosSharp qui sont les « CCSprite » et les « CCSpriteSheet ». Le premier représente un élément statique de notre jeu, c'est-à-dire une ressource fixe (par exemple un fichier imag) alors que le second se compose de plusieurs éléments statiques qui peuvent être enchaînés rapidement afin de créer une animation (exactement comme le principe des dessins animés). Si nous décomposons notre décor en plusieurs éléments, nous pouvons établir une sorte de cartographie des éléments de type « CCSprite » ou « CCSpriteSheet » à superposer dans le bon ordre :

| Nom | Type | Ressource |
|---|---|---|
| title.png | CCSprite (Fixe) |  |
| castle.png | CCSprite (Fixe) |  |
| smoke.png | CCSpriteSheet (Animé) |  |
| fire.png | CCSpriteSheet (Animé) |  |

Editez la classe « MainScene.cs » en créant une nouvelle méthode privée nommée « CreateBackGround » (profitez-en pour supprimer ou mettre en commentaire notre test permettant d'afficher un label « Bienvenue sur CocosSharp ») : Fig.20.
La méthode « CreateBackGround » instancie une classe de type

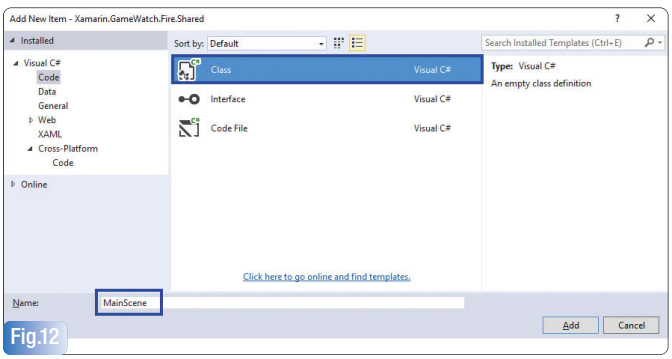


Fig.12

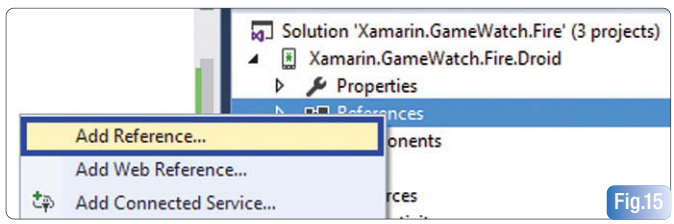


Fig.15

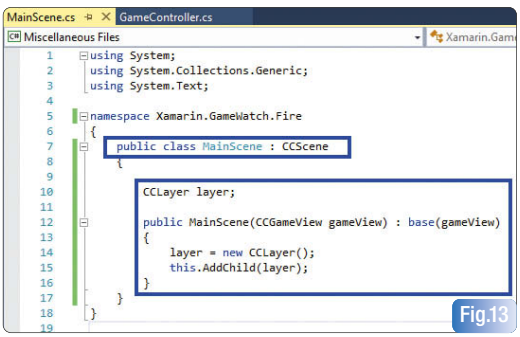


Fig.13

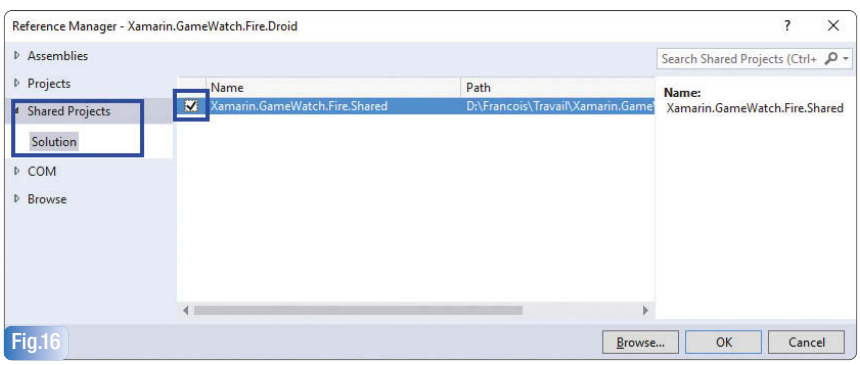


Fig.16

Fig.14

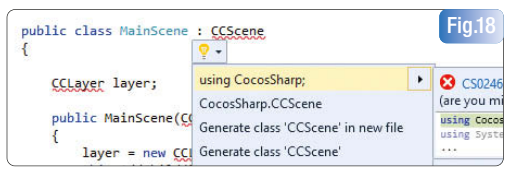


Fig.18

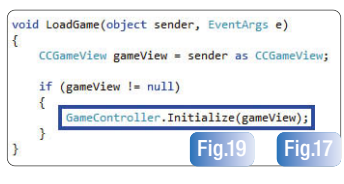


Fig.19
Fig.17

« CCSprite » avec une ressource « title.png ». Cette ressource doit être placée dans le répertoire « Assets/Content » de notre application (Android et iOS) : Fig.21.

**Note :** *Toutes les ressources sont fournies en téléchargement dans le lien « ressources.zip » disponible à la fin de l'article.* Fig.22.

Dans l'univers CocosSharp, les coordonnées (x,y) d'un « CCSprite » sont positionnées au milieu de la ressource. La propriété « AnchorPoint » permet de les redéfinir. Ainsi, nous indiquons que le centre de la ressource est maintenant en bas à gauche et non au milieu. Voici un exemple sans et avec : Fig.23.

Enfin, la propriété « IsAntiAliased » permet de ne pas lisser la ressource. C'est le cas ici car nous voulons avoir un aspect pixélisé pour avoir le rendu « rétro gaming ». N'oubliez pas de rajouter les ressources au projet iOS (vous pouvez les ajouter en tant que lien pour éviter de multiplier les fichiers identiques) : Fig.24 et 25.

Comme pour le fond, rajoutez maintenant le château en créant une nouvelle méthode « CreateCastle ». N'oubliez pas l'appel de la méthode dans le constructeur de notre scène : Fig.26.

Le château n'est pas placé correctement sur l'axe des y. Il devrait être « docké » vers le haut. Définissez la propriété « PositionY » ainsi : Fig.27.

Pour rajouter la fumée, créez une nouvelle méthode « CreateSmoke ». Cette méthode utilisera « CreateAnimationFromSpriteSheet » qui permet à partir d'une liste d'extraire tous les sprites afin de les enchaîner rapidement, créant ainsi notre animation : Fig.28.

**Note :** *N'oubliez pas de rajouter les ressources smoke.plist et smoke.png aux projet Android et iOS.*

Exécuter le projet (Android ou iOS) : Fig.29.

La fumée s'anime mais elle est au-dessus de notre château. C'est normal puisque les « layers » s'ajoutent dans l'ordre de création. Il faut donc positionner notre méthode avant l'ajout du château comme ceci : Fig.30.

De la même façon, rajoutez le feu au château et notre décor animé sera complet : Fig.31.

Exécuter le projet (Android ou iOS).

*Suite dans le numéro 200*

Fig.20

```
public MainScene(CCGameView gameView) : base(gameView)
{
    layer = new CCLayer();
    this.AddChild(layer);

    //Création de notre décor animé
    CreateBackGround();

}

private void CreateBackGround()
{
    var sprite = new CCSprite("title.png");
    sprite.AnchorPoint = new CCPoint(0, 0);
    sprite.IsAntialiased = false;
    layer.AddChild(sprite);
}
```

Fig.22

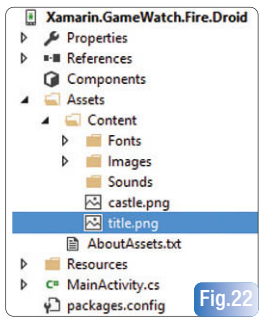



Fig.21

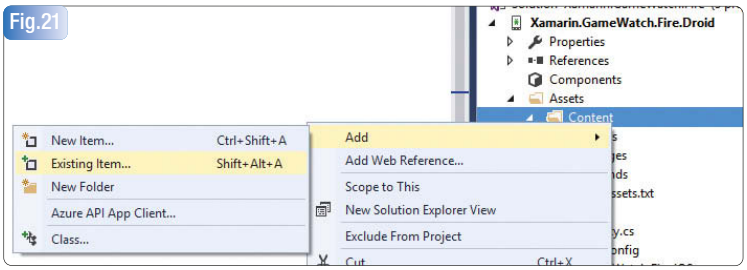



Fig.23

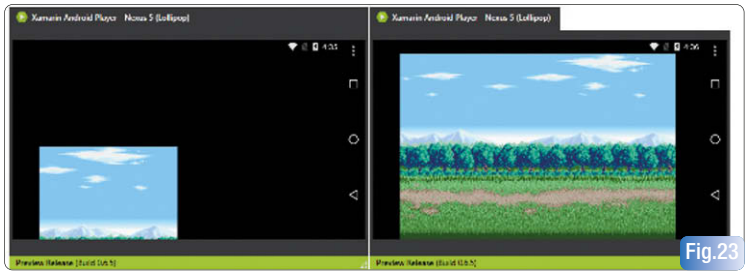


Fig.24

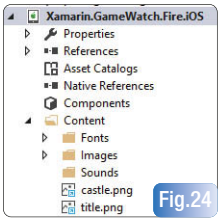



Fig.25

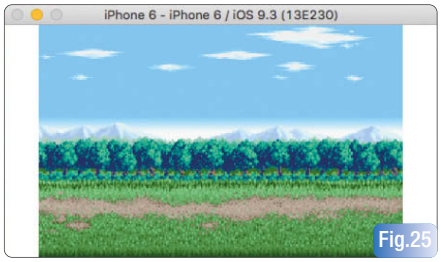



Fig.26

```
public MainScene(CCGameView gameView) : base(gameView)
{
    layer = new CCLayer();
    this.AddChild(layer);

    //Création de notre décor animé
    CreateBackGround();
    CreateCastle();

}

private void CreateBackGround()...

private void CreateCastle()
{
    var sprite = new CCSprite("castle.png");
    sprite.AnchorPoint = new CCPoint(0, 0);
    sprite.IsAntialiased = false;
    layer.AddChild(sprite);
}
```

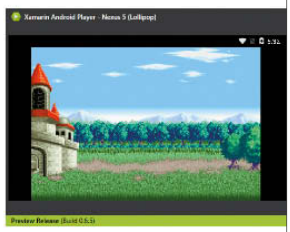


Fig.27

```
sprite.AnchorPoint = new CCPoint(0, 0);
sprite.PositionY = layer.ContentSize.Height - sprite.ContentSize.Height;
sprite.IsAntialiased = false;
```

Fig.28

```
    CreateBackGround();
    CreateCastle();
    CreateSmoke();
}

private void CreateBackGround()...

private void CreateCastle()...

void CreateSmoke()
{
    var sprite = CreateAnimationFromSpriteSheet("smoke.plist", "smoke", 0.5f);
    sprite.AnchorPoint = new CCPoint(0, 0);
    sprite.PositionX = 10;
    sprite.PositionY = layer.ContentSize.Height - sprite.ContentSize.Height;
    sprite.IsAntialiased = false;
    layer.AddChild(sprite);
}

private CCSprite CreateAnimationFromSpriteSheet(string plist, string frame, float delay, bool launchAnimation=true)
{
    //Déclaration de notre animation et du mode de répétition
    //Ici RepeatForever = animation en boucle infini
    CCAnimation animation;
    CCRepeatForever repeatForever;

    var spriteSheet = new CCSpriteSheet(plist);
    var animationFrames = spriteSheet.Frames.FindAll((x) => x.TextureFilename.StartsWith(frame));

    //Le delay est la vitesse d'animation
    animation = new CCAnimation(animationFrames, delay);
    repeatForever = new CCRepeatForever(new CCAnimate(animation));

    var sprite= new CCSprite(animationFrames.First());
    if (launchAnimation) sprite.RunAction(repeatForever);

    return sprite;
}
```

Fig.29

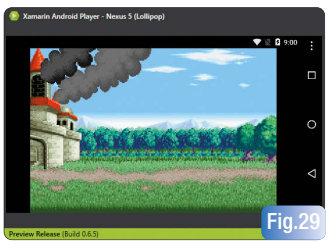


Fig.30

```
//Création de notre décor animé
CreateBackGround();
CreateSmoke();
CreateCastle();
```

Fig.31

```
    CreateSmoke();
    CreateCastle();
    CreateFire();
}

private void CreateBackGround()...

private void CreateCastle()...

void CreateSmoke()...

void CreateFire()
{
    var sprite = CreateAnimationFromSpriteSheet("fire.plist", "fire", 0.3f);
    sprite.AnchorPoint = new CCPoint(0, 0);
    sprite.PositionY = layer.ContentSize.Height - sprite.ContentSize.Height;
    sprite.IsAntialiased = false;
    layer.AddChild(sprite);
}
```

# Comment assurer une belle vie à son application mobile ?

Partie 2

*Dans notre dossier constituant les bonnes pratiques offrant la prospérité à une application mobile, nous avons vu dans le numéro précédent comment automatiser le build et le déploiement. Ces maillons de la chaîne DevOps permettent de répondre en partie aux attentes en termes de qualité, et se voient complétés avec les maillons concernant les tests automatiques.*

Mathilde Roussel
*R&D Software Engineer chez* **MEGA International**
https://twitter.com/Math_Roussel

Jason De Oliveira
*CTO | MVP chez* **MEGA International**
http://www.jasondeoliveira.com

En effet, si les tests manuels réalisés par des bêta-testeurs sont très utiles, ils ne sont pas très pratiques et ne suffisent malheureusement pas. Pour pallier ce problème, les tests unitaires permettent dans un premier temps une cohérence du projet au niveau des fonctionnalités et de la logique, et évitent souvent des régressions dans le code. Avec son fidèle compagnon, les tests UI, ils forment une première barrière efficace pour contrer les attaques de bugs.

## Xamarin Test Cloud, une véritable armée au service de notre application

### Tester une application sur un périphérique, c'est bien, tester sur tous les périphériques possibles, c'est mieux !

Xamarin Test Cloud (ou XTC), un service lancé en 2014 par Xamarin, fournit la possibilité de déployer et de tester une application sur plus de 2000 périphériques physiques, tout ceci depuis le cloud. Ce service facilite donc grandement l'étape des tests sur périphériques réels, indispensable pour s'assurer que l'application fonctionnera sur toutes les versions des OS, toutes les tailles d'écran, et ce quelle que soit la configuration des téléphones. L'analyse des résultats des tests, comme les éventuels crashs ou les performances, se fait grâce à des captures d'écrans, logs et métriques disponibles sur la plateforme. Comme toute application de qualité qui se respecte, le projet MySuperInventoryManager va lui aussi devoir se confronter à Xamarin Test Cloud et prouver sa valeur.

### Les tests UI en local

Avant de penser à passer les tests UI sur la plateforme Cloud, ceux-ci doivent être réalisés en local. Logique, il est inutile de lancer un bataillon de tests sur une armée de périphériques si ces tests ne se sont pas, au préalable, entrainés sur des pantins en paille.

Pour commencer, il faut créer de nouveaux projets dans Visual Studio. Deux choix sont possibles : créer un unique projet de tests UI cross-plateforme, qui mélangera les tests Android et iOS, ou créer un projet de tests UI pour Android et un autre pour iOS. Le premier choix permet de regrouper des tests UI qui seraient identiques sur les deux plateformes (remplissage et envoi d'un formulaire par exemple), mais perd un peu en lisibilité quand des tests sont spécifiques à une plateforme. Le deuxième choix offre une segmentation des plateformes bien nette mais, de façon antagoniste à la première solution, oblige à dupliquer certains tests qui seraient identiques.

Dans le cas de l'exemple MySuperInventoryManager, nous partirons sur la première solution, mais sans nous préoccuper des tests d'interface sur iOS. Le template de projet Visual Studio utilisé sera donc de type « Xamarin.UITest | Cross-Platform », présent dans la catégorie « Test ». Celui-ci installe les packages NuGet nécessaires, crée une classe d'initialisation (qui permettra, selon la plateforme, de lancer la bonne application), une première classe de test avec un Setup lancé avant chaque test, et un test basique. Pour lancer les tests UI, il est possible de passer soit par l'émulateur, soit par un périphérique physique.

Il est important de noter que le fonctionnement des tests UI n'est pas possible sur tous les émulateurs Android disponibles. En effet, seuls Xamarin Android Player et Genymotion sont capables de faire tourner ces tests d'interface, ce qui veut dire que l'émulateur par défaut, ou celui proposé par Visual Studio (qui fonctionnent avec Hyper-V), ne sont pas opérationnels pour cette partie. Si le test sur périphérique physique est la solution choisie, il faut réaliser les différentes étapes pour activer le périphérique en mode développeur.

Dans le cadre des tests pour une application iOS, il faut, comme souvent, passer par le Mac pour le build, ainsi que par l'émulateur ou le périphérique physique. Tout comme pour Android, passer par le périphérique physique requiert l'activation du téléphone en mode développeur, et il faudra également activer l'UI Automation, option disponible dans la catégorie « Développeurs », dans l'application « Réglages » de l'iPhone.

Il faut alors lancer une vérification pour s'assurer que la configuration avec l'émulateur fonctionne et que le test de base est valide. Ensuite, il est temps d'entamer les choses sérieuses et de définir des scénarios de test. Un premier test simple, mais important, serait de vérifier si, lorsqu'on ajoute un nouveau produit dans la liste, il se retrouve bien présent dans celle-ci de façon graphique. Il faut donc rajouter un nouveau test dans la classe Test.cs qui, selon les bonnes pratiques de nommage, possède un nom explicite sur ce qui est testé et attendu.

```
IApp _app;

[SetUp]
public void BeforeEachTest()
{
  _app = AppInitializer.StartApp(platform);
}

void AddItemToList(string name)
{
  _app.Tap(element => element.Marked("addProductName"));
  _app.EnterText(name);
  _app.Tap(element => element.Marked("AddProductButton"));
```

```
  _app.Tap(element => element.Marked("addProductName"));
  _app.ClearText();
  _app.DismissKeyboard();
}
[Test]
public void Add_Product_To_List_Product_Should_Be_On_Display()
{
  AddItemToList("New Item");
  var result = _app.Query(element => element.Marked("New Item"));
  Assert.IsTrue(result.Any(), "The 'New Item' wasn't added :(");
}
```

La propriété _app, initialisée avant chaque test, est l'instance courante de l'application. Elle est centrale car c'est le pont pour interagir avec l'application. Pour plus de lisibilité, une méthode « AddItemToList(...) » a été créée, et simule les actions de saisie et d'ajout d'un élément à la liste. C'est une action basique sur cet écran de l'application, et extraire ces quelques lignes dans une méthode à part permet de ne pas répéter le code et d'avoir des tests plus courts, qui se concentrent sur le test en lui-même. A l'intérieur de cette méthode, l'application va remonter un élément marqué avec « addProductName », qui est un champ texte. La méthode « Marked » va pouvoir chercher, dans l'ensemble de l'arborescence de la page, un élément dont l'id ou le contenu correspond à ce qui lui est passé en paramètre. Un focus est effectué sur cet élément, le texte y est saisi, le bouton d'ajout est déclenché, le champ texte de nouveau sélectionné est vidé, et le clavier disparaît enfin.

Le test « Add_Product_To_List_Should_Be_On_Display » créé en-dessous va d'abord appeler la méthode d'ajout d'un produit appelé « New Item ». Ici encore, l'application va ensuite requêter un élément qui porterait le texte « New Item » et qui devrait normalement être une nouvelle cellule de la ListView. Enfin, il faut vérifier, grâce à un Assert, si l'élément a été trouvé. Si c'est le cas, le test passe, sinon le message d'erreur spécifié sera indiqué. Pour peupler un peu plus l'application, un autre test est ajouté, pour vérifier qu'un même produit n'est pas ajouté deux fois dans la liste.

Les tests UI passent dans Visual Studio et sur l'émulateur (Fig.1), et même s'ils prennent un temps plus conséquent que des tests unitaires (ici 36 secondes de moyenne), ils sont fonctionnels et prêts à affronter l'armée de périphériques que Xamarin Test Cloud met à disposition.

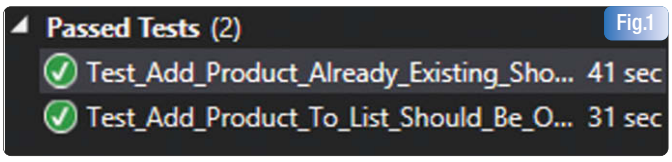


Fig.1

## Les tests déployés sur le Cloud

L'envoi manuel des tests UI sur Xamarin Test Cloud peut se faire de deux manières : depuis Visual Studio (en sélectionnant « Run in Test Cloud » lors du clic droit sur le projet de test) ou depuis la ligne de commande. En brut, cette ligne de commande a cette forme : test-cloud.exe submit <PATH-TO-APK> <TEAM API KEY> –devices=<DEVICES> –assembly-dir=<PATH-TO-TEST-ASSEMBLY-DIR> –user=<EMAIL>. Depuis la plateforme web de XTC, il est facile de générer une partie des variables manquantes à la ligne de commande. Pour cela, il faut lancer un « New Test Run », sélectionner le projet, le ou les périphériques voulus, ainsi que la série et la langue. Grâce à toutes ces informations, une ligne de commande plus complète est générée. Elle est disponible sous deux formes, pour OS X ou Windows, et renseigne entre autres l'API Key, un id correspondant au choix de périphérique et le nom de l'application. Le choix de passer par Visual Studio reste tout de même plus simple, puisqu'il va ouvrir dans le navigateur une page pour choisir les périphériques, la série et le langage (Fig.2). Le reste est récupéré automatiquement par Xamarin Test Cloud.

Une fois le choix des périphériques terminé (il est possible d'en choisir un nombre « illimité », même avec la version d'essai), il est temps de lancer la ligne de commande ou de valider depuis le navigateur pour démarrer les tests. Ceux-ci passent par une première étape de validation, de recherche des périphériques disponibles, et c'est parti !

Le dashboard regroupe les derniers tests effectués, avec quelques métriques (Fig.3). Evidemment, le temps d'exécution des est plus ou moins

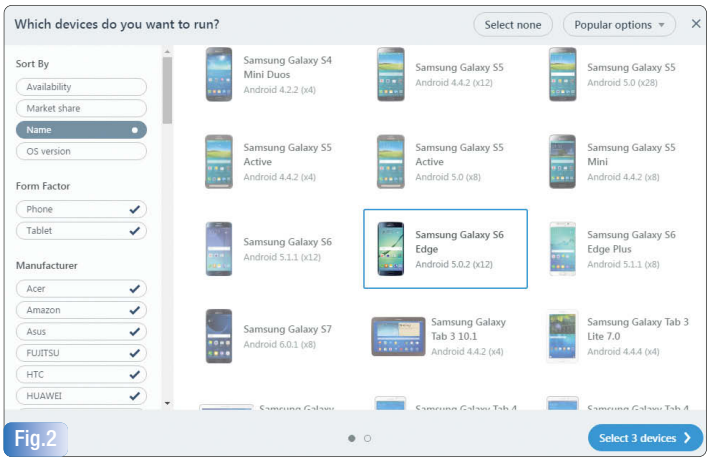


Fig.2

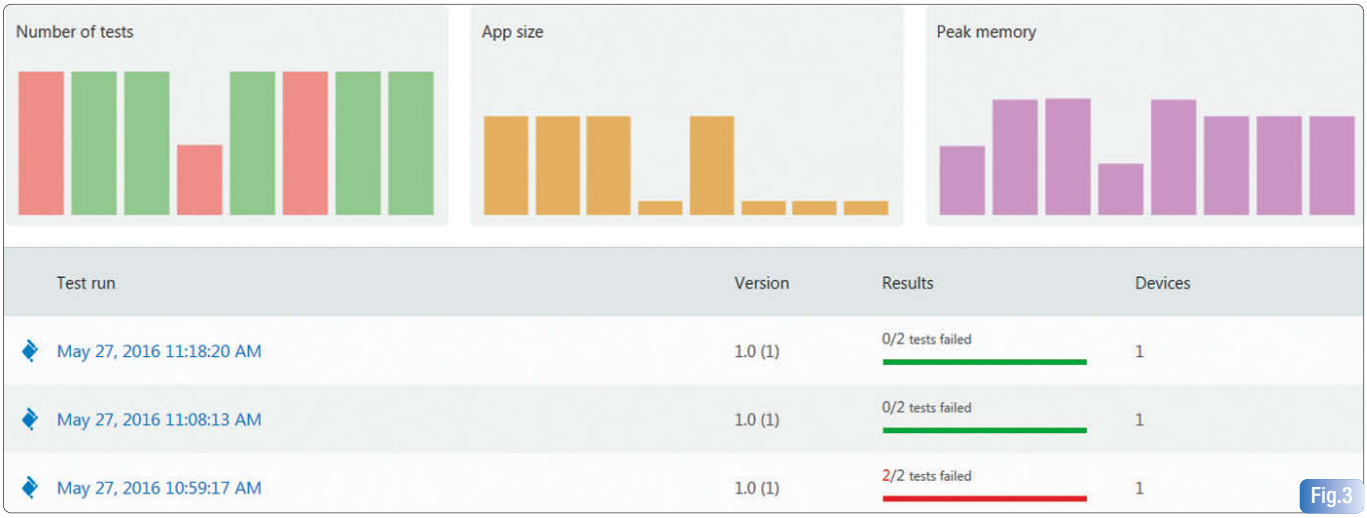


Fig.3

long, selon le nombre de périphériques choisis, la disponibilité de ceux-ci, et le nombre de tests à réaliser. Dans le cas de l'exemple, avec 2 tests UI et 3 périphériques choisis, le temps total atteint 8 minutes. Attention à ne pas se faire avoir, quelle que soit la façon dont sont lancés les tests, il faut bien avoir le fichier .apk de l'application Android à jour ! Sinon, il y a de grandes chances que les tests et le code ne correspondent pas, ce qui conduit à un assaut perdu.

Il est possible de voir un résumé du test lancé (Fig.4). Plusieurs informations importantes sont affichées, mais il existe une vision plus détaillée par test et périphérique, en cliquant sur un test dans la liste. Dans le détail des informations, l'usage de la mémoire, l'usage CPU, la Stack Trace et les logs sont disponibles, et permettent de mieux traquer les erreurs ou crashs éventuels (Fig.5).

## L'automatisation des tests

Après une vérification du passage des tests UI sur Xamarin Test Cloud en manuel, il faut maintenant rejoindre la partie créée dans le précédent article, l'intégration continue ! En effet, c'est une première victoire que d'avoir des tests UI fonctionnels, mais il manque tout de même l'automatisation pour parfaire le tout.

Une petite mise en garde afin d'éviter de perdre du temps : il faut penser à ajouter la permission INTERNET dans l'AndroidManifest d'une application Android avant de lancer les tests automatiques depuis Visual Studio Team Services.

Si ce n'est pas le cas, après la configuration et le lancement d'un pipeline de build, un message d'erreur explicite apparaîtra dans les logs. De ce fait, afin d'éviter un test à blanc, il est important de le faire au préalable.

Il est donc temps de laisser de côté Visual Studio et le code, pour retrouver Visual Studio Team Services et la page « Build ». Dans le cas de l'exemple de MySuperInventoryManager, le pipeline de build est déjà en place. Il est lancé à chaque nouveau commit, et contient des étapes de compilation et publication de l'application Android. Au départ, deux étapes avaient été désactivées : le MSBuild et le test sur Xamarin Test Cloud. L'étape MSBuild a été supprimée ; elle n'est plus utile car l'étape Visual Studio Build se charge de compiler l'ensemble de la solution.

En revanche l'étape Xamarin Test Cloud peut être réactivée. L'étape MSBuild est indispensable si la partie Visual Studio Build ne compile pas l'ensemble de la solution.

Il faut en effet que le projet de test UI soit compilé et que ses assemblies soient disponibles. Les deux solutions sont viables, le plus important étant d'avoir les tests assemblies pour configurer l'étape qui suit, qui n'est autre que le test avec Xamarin Test Cloud.

Ajoutée à la suite, elle requiert le fichier .apk de l'application, la clé d'API de Xamarin Test Cloud et l'email utilisateur sur la plateforme (Fig.6). Il faut également renseigner l'id correspondant aux périphériques choisis pour les tests. Pour récupérer cet id, il suffit, comme expliqué un peu avant de choisir « New Test Run » sur la plateforme web de XTC, de sélectionner les périphériques et, à la dernière étape, d'extraire l'id après le « –devices » de la ligne de commande générée.

Attention, plus le nombre de périphériques est important plus le temps total du pipeline de build sera important. Enfin, il faut renseigner le dossier des assemblies de test générées dans l'étape précédente, dont le chemin sera dans ce cas $(build.binariesdirectory)/$(BuildConfiguration)/test-assembly. Le reste des options peut rester par défaut pour une utilisation basique.

Le nouveau pipeline de build est prêt, il ne reste plus qu'à l'envoyer sur le champ de bataille et vérifier qu'il revient victorieux ! Pour MySuperInventoryManager, qui est un projet qui possède deux tests UI et qui va tourner sur un seul périphérique sur Xamarin Test Cloud, le pipeline complet de build prend environ 9 minutes. L'étape la plus longue est bien évidemment Xamarin Test Cloud, puisqu'il faut parfois attendre plusieurs minutes qu'un périphérique soit disponible. Néanmoins, tout s'est bien déroulé et, dans le résumé du build, les informations capitales concernant la réussite ou non des tests sont affichées (Fig.7). Si on souhaite plus de détails, il faut en revanche passer sur l'interface de Xamarin Test Cloud.

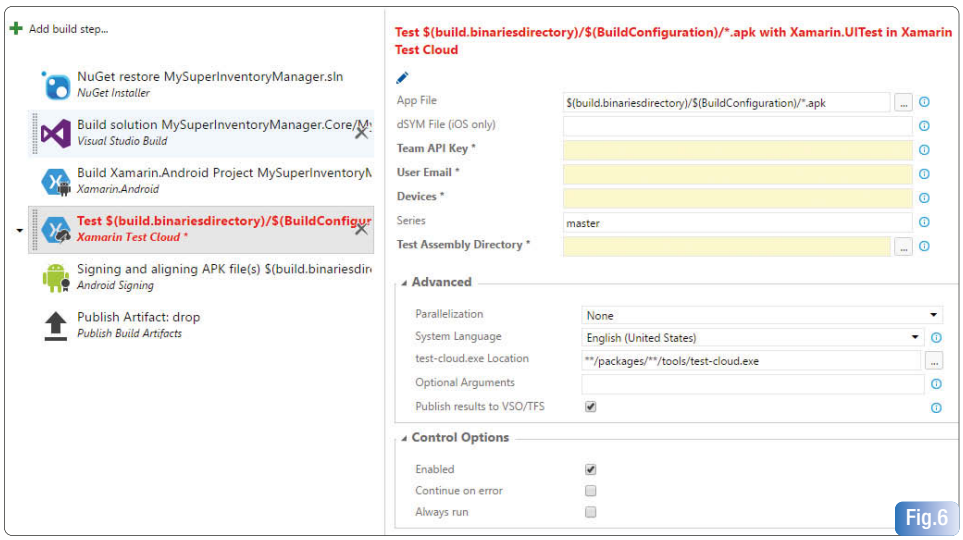


Fig.6

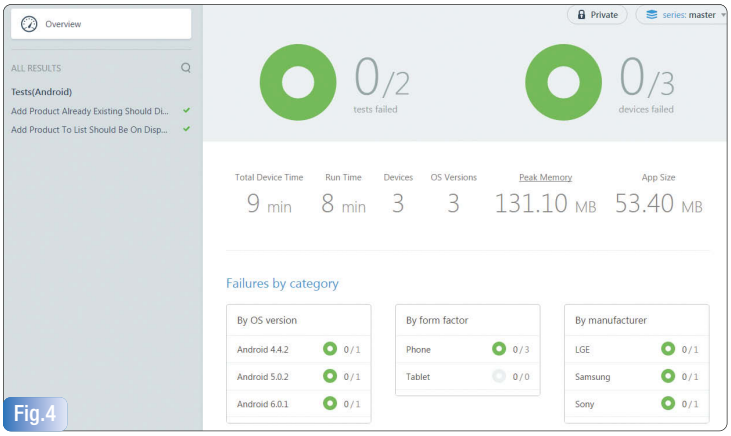


Fig.4

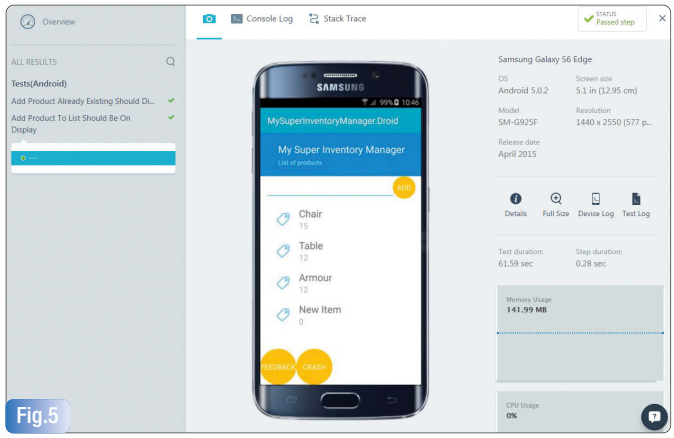


Fig.5

## Les tests unitaires !

### Parce que les tests, c'est la vie.

Comment oublier de parler de tests unitaires lorsqu'on parle de tests graphiques ? Aujourd'hui, ils sont familiers pour beaucoup de développeurs et acteurs du département qualité d'un projet. Le projet MySuperInventoryManager ne va pas non plus y échapper. Une petite précision cependant avant de commencer : lors de la création du projet de tests UI, deux package NuGet ont été installés : NUnit et NUnitTestAdapter. En effet, les tests UI se basent sur ce framework de test, mais sur une version bien précise : la version 2.6.4 pour NUnit et la version 2 pour NUnit TestAdapter. Il existe des versions plus récentes pour ces deux packages, et même si la tentation peut être grande de les mettre à jour, il vaut mieux éviter, pour que le lancement des tests dans Xamarin Test Cloud fonctionne. En effet, ce service ne supporte pas pour l'instant de version supérieure à NUnit 2.6.4. Ce point impacte le projet de tests unitaires, car il utilise lui aussi NUnit. Il est vivement recommandé d'avoir les mêmes versions de packages pour assurer le bon fonctionnement du pipeline de build.

Dans le cas de MySuperInventoryManager, un projet de tests unitaires va être dédié aux tests du projet Core. Les parties Core et Services prenant le parti du pattern IoC et de l'injection de dépendances, il est nécessaire dans le projet de tests de Mocker les services, grâce à Moq. La méthode Initialize, annotée avec l'attribut [SetUp] va se charger de cela, pour définir le comportement souhaité lors de l'appel à des méthodes des services.

```
private MainViewModel _mainViewModel;
private Mock<IProductService> _productServiceMock;

[SetUp]
public void Initialize()
{
_productServiceMock = new Mock<IProductService>();
_productServiceMock.Setup(x => x.GetAllProducts()).Returns(MockDatas.Products);
_mainViewModel = new MainViewModel(_productServiceMock.Object);
}
```

Ensuite, tout comme les tests UI, deux tests unitaires sont ajoutés pour ce projet, dont un qui va vérifier le comportement en cas de liste de produits null.

```
[Test]
public void Get_All_Products_With_List_Null_Should_Return_List_Not_Null()
{
_productServiceMock.Setup(x => x.GetAllProducts()).Returns((List<Product>)null);
_mainViewModel.GetProducts();
Assert.IsTrue(_mainViewModel.Products != null);
}
```

Si la structure du test nous intéresse peu ici, nous allons nous focaliser sur la façon dont lui et ses camarades vont s'intégrer au processus de build sur Visual Studio Team Services. La configuration est très simple : depuis la plateforme de VSTS, toujours dans le même build definition, il faudra modifier une étape et en rajouter une.

La première chose à faire est donc de modifier l'étape du « Visual Studio Build ». Jusqu'à présent, le chemin du .csproj du projet Core était renseigné, mais pour faire fonctionner les tests, le nouveau chemin choisi sera celui du .sln, pour builder toute la solution d'un coup, tests y compris.

La suite consiste à ajouter une étape de type « Visual Studio Test », qui possède dans sa configuration par défaut une regex pour tester les assemblies de test : ***test*.dll;-:**\obj**. L'idée de laisser ce chemin est tentante, mais il faut néanmoins faire attention. Dans le cas du projet MySuperInventoryManager, les tests UI qui ont été créés sont des tests UI pour Android mais aussi pour iOS. Or, cela devient une habitude, qui dit iOS dit besoin d'un Mac. Et cette regex telle qu'elle va non seulement lancer les tests unitaires, mais aussi les tests UI. Sauf que cette étape sur VSTS va exécuter une tâche Windows, ce qui va être fatal aux tests UI qui vont émettre une erreur due à iOS. Moralité de l'histoire, une pirouette possible est de prendre le parti d'une convention de nommage différente pour les tests unitaires et tests UI. Cela permettra de changer la regex ci-dessus pour qu'elle cible uniquement le ou les projets de tests unitaires (la tâche des tests UI étant déléguée à Xamarin Test Cloud). Ici, les projets de tests unitaires finiront tous par « tests », la regex sera donc comme suit : ***tests.dll;-:**\obj**. Un lancement du pipeline permet de voir si les tests unitaires ont été découverts, s'ils sont passés, et si le reste n'est pas impacté par ces changements. Si tout fonctionne correctement, un petit diagramme dans le résumé du build (quand celui-ci est terminé sans erreur) permet de voir le nombre de tests, ceux qui ont réussi, ceux qui ont échoué, etc. (Fig.8).

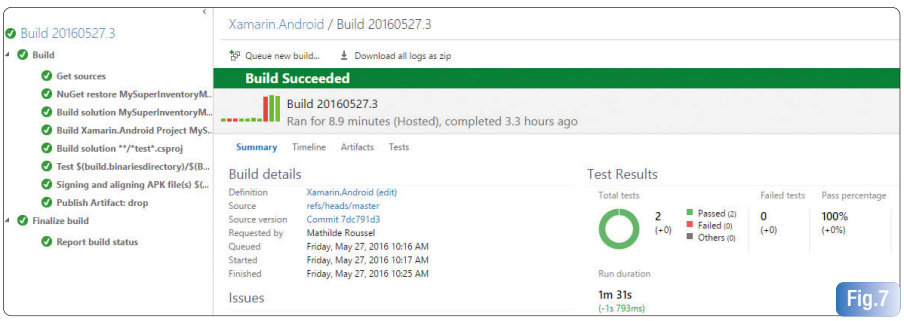


Fig.7

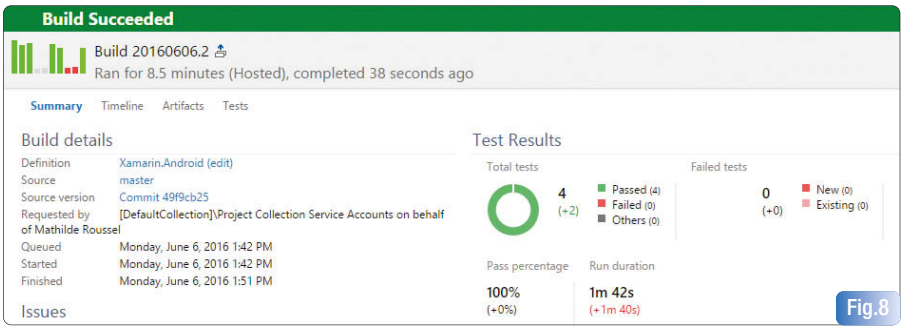


Fig.8

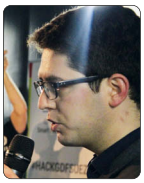
## Conclusion

L'article précédent (Programmez 198) a montré comment utiliser Visual Studio Team Services dans un contexte « DevOps Mobile ». Dans cet article, nous avons vu comment pousser encore plus loin les pratiques « DevOps » grâce à l'automatisation des tests UI et tests unitaires pour toujours assurer une bonne qualité ainsi qu'une vraie maîtrise de votre code. Comme un vrai chevalier sans peur, vous pouvez alors faire évoluer les fonctionnalités de vos applications mobiles sans crainte.

Restez connectés car les prochains articles aborderont les sujets des bêtas, ou comment tirer le meilleur parti de HockeyApp, et l'utilisation poussée de Visual Studio Team Services, ou comment bien découper les étapes de build et release pour un projet industrialisé. Ces différents sujets pourront faire de vous un vrai maître du développement mobile.

# Créer votre tablette tactile avec Node.js 2epartie

*Avec Javascript tout devient possible, ou presque ! Nous allons donc prendre le contrôle d'une Raspberry Pi et de son écran tactile, afin de lancer au démarrage une application programmée en technologie Web. Le tout en partant d'une version de base de linux (Raspbian lite).*



Matthieu Bouilloux
*Développeur web et applicatif, utilisant principalement des technologies issues du monde du web. Intéressé par tout type de nouvelle technologie tel que l'Iot ou encore l'analyse des marchés boursier.*
*Site web:* http://www.katlea-edition.com
*Email:* matthieu@katlea-edition.com

## Le fichier package.json (rappel du n°198)

Pour notre navigateur internet il faut informer les sites internet que nous sommes dans un format tablette. La navigation se faisant dans des iframes, la méthode la plus simple reste de modifier le « user-agent » par défaut et de le remplacer par exemple, par celui d'un iPad.

```
"user-agent":"Mozilla/5.0 (iPad; CPU OS 5_1_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Mobile/9B206",
```

Nous n'avons pas de dossier node_modules et donc de dépendances. En effet, elles ne sont pas requises par les fonctions que nous allons aborder. Mais rien ne vous empêche d'installer un package npm dans le dossier :

```
sudo npm install socket.io –save
```

En « sudo » de préférence, « socket.io » représente un package npm et l'option « –save » sert à modifier automatiquement les dépendances du fichier « package.json ». Après cette commande, vous pouvez appeler dans le DOM les packages node.js comme dans un script node classique. Ce qui nous donne au final pour le fichier package.json :

```
{
 "name": "nw-demo",
 "version": "0.0.1",
 "main": "index.html",
 "window": {
  "toolbar": false,
  "frame": false,
  "fullscreen": true
 },
 "chromium-args" : "--touch-devices=6 --touch-events=enabled --enable-pinch",
 "user-agent":"Mozilla/5.0 (iPad; CPU OS 5_1_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Mobile/9B206",
 "dependencies": {
 }
}
```

## MDL : Material Design Lite

J'ai fait le choix d'utiliser MDL : https://getmdl.io pour obtenir rapidement une interface en simili Material Design et ce sans trop alourdir l'application. Les fichiers requis sont :

/css/material.min.css
/css/material.min.css.map
/js/material.min.js
/js/material.min.js.map
/fonts/MaterialIcons-Regular.woff2

Je vous invite à consulter le site pour prendre connaissance des éléments : https://getmdl.io/components/

## Le fichier index.html

### Le head

On appelle le css du material design, le main.css servant pour nos propres styles, et le fichier overlay.css correspondant aux styles du clavier virtuel.

```
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/material.min.css">
  <link rel="stylesheet" type="text/css" href="css/main.css">
  <link rel="stylesheet" type="text/css" href="css/overlay.css">
</head>
```

### La base HTML

La base est composée de 4 parties. Le header permet de naviguer dans le système, le main affiche les applications, le dialog est utilisé comme popup et enfin le clavier en html. Cette méthode est un peu brute pour le clavier mais elle ne pose aucun problème de compatibilité.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/material.min.css">
  <link rel="stylesheet" type="text/css" href="css/main.css">
  <link rel="stylesheet" type="text/css" href="css/overlay.css">
 </head>
 <body>
  <div class="mdl-layout mdl-js-layout mdl-layout--fixed-header">
            /* HEADER */
            /* MAIN */
            /* DIALOG (popup) */
            /* KEYBOARD */
  </div>
 </body>
</html>
```

### Le Header

Le header reprend le titre de l'application en cours (#current-app), ainsi que les boutons système (accueil, WiFi, clavier) et la navigation entre les applications par onglet. L'appel d'une application système se fait en Javascript par un « system('nom_du_dossier_de_l'application') ».
Code complet sur le site www.programmez.com.

## Le burger menu

Les applications placées dans le dossier app sont automatiquement ajoutées à la liste du Burger Menu.

```
<!-- Le classique Burger Menu qui accueillera la liste des applications situées dans le dossier app -->
<div class="mdl-layout__drawer">
  <span class="mdl-layout-title">Applications</span>
  <nav id="menu" class="mdl-navigation">

  </nav>
</div>
```

## Le conteneur des applications

Les applications sont contenues dans les balises « section », elles-mêmes reliées à la navigation par onglet « #tab » du header.

```
<!-- Le contenu des applications, sous forme de section -->
<main class="mdl-layout__content noselect">
  <section class="mdl-layout__tab-panel is-active" id="system">
    <div class="page-content"></div>
  </section>
</main>
```

## la popup

La popup peut être utilisée de manière générique. Ici je l'utilise pour afficher les réseaux wifi (#wifi-list).

```
<!-- La popup pour les réseaux wifi -->
<dialog id="dialog-wifi" class="mdl-dialog">
  <h4 class="mdl-dialog__title">Wifi networks</h4>
  <div class="mdl-dialog__content">
    <ul id="wifi-list">

    </ul>
  </div>
  <div class="mdl-dialog__actions">
    <button type="button" class="mdl-button close">Close</button>
  </div>
</dialog>
```

## Le clavier Virtuel

Brutal et efficace, il suffit de faire une boucle sur les tags « li » du conteneur « #lettes » et ainsi d'ajouter un « eventListenner » sur chaque itération.
Code complet sur le site www.programmez.com.

## Les scripts

En premier on appelle le fichier Javascript de Material Design Lite.
Les fichiers uuids.js et scripts.js sont des « helpers » qui permettent de se passer de jQuery, dont l'intérêt est limité, vu qu'ici nous avons au maximum 6 «helpers ». Il est donc inutile d'alourdir l'application jQuery.
Le fichier navigation.js sert à charger et à supprimer dynamiquement les applications et donc leurs contenus HTML/CSS/JAVASCRIPT et leurs configurations.
Une fois le système initialisé, on appelle la page d'accueil par « system ('home') ;

On définit des variables globales :

- overlay pour le clavier virtuel :
  - show : état de visibilité du clavier ;
  - target : cible de l'élément où insérer le texte.
- Keyboard : pour le clavier en plein écran avec une balise « textarea » sous forme d'application system :
  - Title : Espace pour du texte au-dessus de la « textarea » ;
  - Fn : utilisé pour attribuer une fonction qui va traiter le texte du « textarea » dès que l'utilisateur valide (callback) ;
  - Placeholder : pour pré-remplir la « textarea ».
- Pour l'application Keyboard, il est nécessaire de réduire le header pour gagner en visibilité. C'est là qu'intervient la fonction fullscreen.
- Le fichier wifi.js s'occupe de la relation entre le DOM et le service /services/wifi.js que nous verrons par la suite.
- Pour terminer, le fichier overlay.js est utilisé pour le clavier en « overlay »

```
<script src="js/material.min.js"></script>
<script src="js/uuid.js"></script>
<script src="js/script.js"></script>
<script src="js/navigation.js"></script>
<script>

  system('home');

  var overlay = {
    show:false,
    target:null
  };

  var keyboard = {
    title:"",
    fn:null,
    placeholder: null
  };

  var screenSize = false;

  function fullScreen(){
    hideOverlay();
    var head = document.querySelector('header');
    if(screenSize){
      head.style.marginTop = 0;
      screenSize = false;
    }else{
      head.style.marginTop = '-' + (parseInt(getComputedStyle(head).height) - 5) + 'px';
      screenSize = true;
    }
  }

  var wifi = require('./services/wifi.js');
</script>
<script src="js/wifi.js"></script>
<script src="js/overlay.js"></script>
```

## Le service /services/wifi.js

Les commandes linux de base :

### Killall wpa_supplicant

Pour être sûr d'éviter les conflits ou une interface encore active, on détruit toutes les instances du programme wpa_supplicant

### ifconfig

Ce qui nous intéresse ici, c'est l'interface sans fil active, mentionnée par wlan0.

```
wlan0     Link encap:Ethernet  HWaddr b8:27:eb:6d:93:9d
```

Dans le fichier de données data.json, on note l'interface active wlan0.

```
{"interface":"wlan0","network":{}}
```

La variable network enregistrera les réseaux WiFi.
Pour appeler les commandes, on utilise la fonction « exec » de l'appel des processus de Node.js child_process.

```
require('child_process').exec
```

Et on récupère le retour de cette fonction dans l'argument stdout passé en paramètre.

### IFCONFIG : INTERFACE UP

Si l'interface n'est pas active, on l'active. On vérifie cela grâce aux expressions régulières dans le retour des stdout à l'aide de la fonction test de Javascript.

```
Ifconfig wlan0 up
```

### iwconfig

On lance ensuite un intervalle qui vérifie si l'interface WiFi est connectée à un réseau via le SSID. Une expression régulière et une fonction Javascript « match » vont permettre d'identifier soit un ESSID, soit un « off/any » signifiant que l'interface WiFi n'est connectée à aucun réseau.

### Iwlist : interface scan

Cette commande permet un scan des réseaux sur l'interface ; en utilisant les expressions régulières, on actualise la liste des réseaux.

### La connexion

Elle s'établit comme vu précédemment dans l'article, à l'exception que l'on insère des marqueurs dans la chaîne de caractères pour les remplacer dynamiquement.

```
wpa_passphrase ":SSID" :PWD > /home/pi/wpa-temp.conf && sudo wpa_supplicant -B -D
 nl80211,wext -i :INTERFACE -c /home/pi/wpa-temp.conf && rm /home/pi/wpa-temp.conf
```

Voici une fonction pour échapper les caractères spéciaux lors du replacement du password : PWD

```
function strToPwd(str){
  return str.replace(/[!@#$%^&*()+=\-[\]\\';,./{}|":<>?~_]/g, "\\$&");
}
```

## Le fichier /js/wifi.js

Il gère la relation entre le DOM et le service WiFi.

On ajoute les fonctions de lancement de la popup avec la liste des réseaux WiFi. A la sélection du réseau WiFi, on lance la fonction wifiConnect.

La fonction wifiConnect lance le clavier en plein écran et ajoute en callback. La connexion au réseau est une fonction, qui crée un intervalle de 10 secondes, qui vérifie si la connexion est établie, et si oui l'enregistre dans le fichier data.json.

Un intervalle met à jour la popup et le statut de l'icône de réseau WiFi.

## L'architecture d'une application

Le nom du dossier de l'application est utilisé comme identifiant. Donc on évite les espaces et les caractères spéciaux par précaution.

Le dossier d'une application contient :

Un fichier config.js contenant le nom de l'application chargée à chaque démarrage. C'est une base d'exemple de fichier de configuration au format json. La structure HTML et le CSS sont dans les fichiers respectifs index.html et index.css.

Ceux-ci une fois chargés dans le DOM déclenchent le lancement du fichier Javascript situé dans le fichier index.js.

Vous pouvez appeler des packages node.js dans les fichiers Javascript chargés dynamiquement sans aucun problème. Une fois l'application terminée, le fichier /js/navigation.js retire tous les fichiers du DOM. Au redémarrage, il relance l'application comme pour la première fois. En exemple dans le projet attaché, vous avez le classique hello world, un Paint et un navigateur Internet à base d'iframe. Pour l'application Paint, le Chromium-args « –touch-devices=X –touch-events=enabled renvoie les informations du pointeur sur les « touch events ».

## Le fichier /js/navigation.js

Pour l'expliquer simplement, il y a deux fonctions, loadjscssfile et removejscssfile, qui respectivement chargent et retirent dynamiquement le contenu de l'application (Structure, styles et scripts). Il y a aussi deux fonctions différentes pour charger soit les applications systèmes, soit les applications standards. Lancer Node Webkit au démarrage de X.

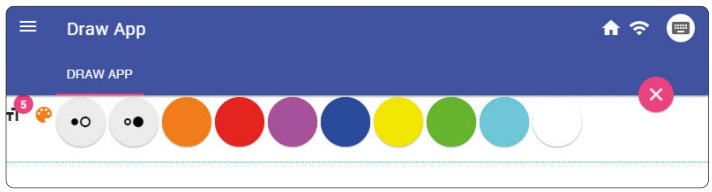
Il suffit d'éditer le fichier « .xssesionrc » situé dans le répertoire /home/pi

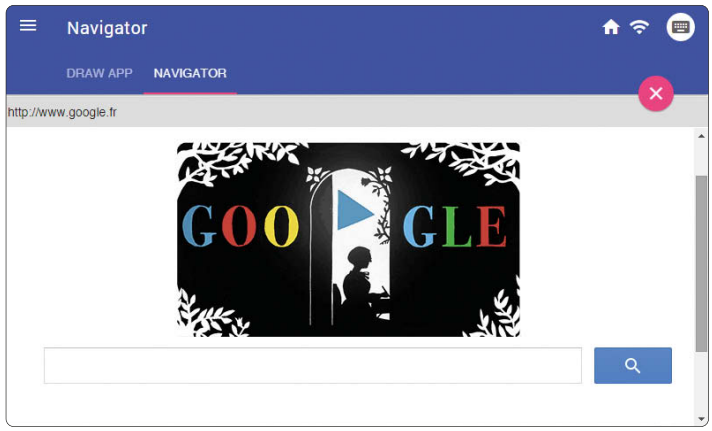
```
cd
sudo nano .xsessionrc
```

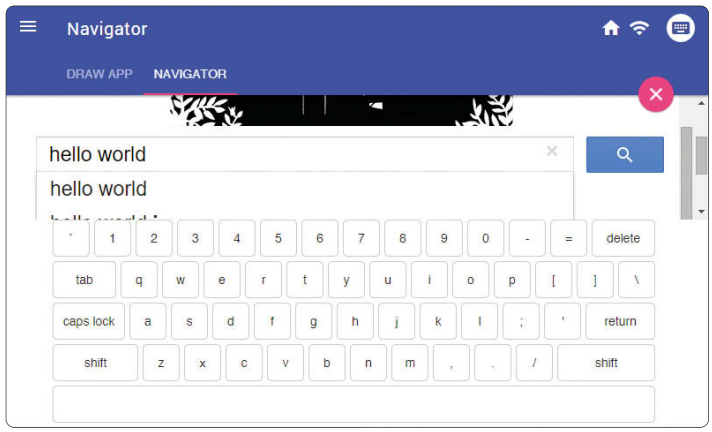
Et on ajoute la ligne suivante à la fin du fichier (ou « nwapp » est le dossier de votre application, situé dans le dossier utilisateur de pi).

```
sudo DISPLAY=:0 nw /home/pi/nwapp &
```

## Quelques Aperçus




## CONCLUSION

Voici donc comment après quelques heures de recherche et de développement, on se retrouve avec sa propre tablette tactile. Vous pouvez aussi envisager de faire des « closures » pour les applications et revoir un peu la structure pour un rendu plus propre.

Mais le Proof of Concept est validé ! Il ne vous reste plus qu'à modifier l'interface à votre convenance et à développer les applications de base de votre tablette. Et pourquoi pas développer un éditeur d'image avancé à l'aide des packages node.js ?

# IoT : connecter une usine avec un Arduino 2e partie

*Les objets connectés se répandent de plus en plus dans notre monde. On voit ainsi de nombreux produits grand public se vendre, comme les montres connectées, ou les capteurs d'activités. Les entreprises découvrent avec joie ces nouveaux appareils, qui leur permettent de répondre à des besoins spécifiques.*

Vincent Piard,
*Expert Technique,*
**SQLI Nantes**


Aurélien Fourmi,
*Expert Technique,*
**SQLI Nantes**

La première étape concernant cette boîte est la fixation des deux éléments clés que sont la breadboard et l'Arduino. Pour la breadboard, le fond est autocollant. A noter que nous avons choisi volontairement de garder une breadboard afin de ne pas multiplier les soudures (car ce n'est pas notre fort) et surtout nous n'avons pas de prérogative particulière concernant la taille de la boîte. Donc en version optimisée le détecteur de présence connecté pourrait faire simplement la taille d'un Arduino. Mais ce souci de miniaturisation a été écarté de notre POC. Fig.4.

A partir de là, il reste 2 étapes pour la finalisation du montage : la soudure des diodes et le découpage de l'emplacement du capteur. Passons rapidement sur la soudure qui après quelques essais se révèle satisfaisante. Il a fallu ensuite creuser un peu le plastique à l'aide d'un foret fin pour fixer les diodes. Le découpage de la boîte fut assez laborieux mais avec un bon cutter cela se fait plutôt aisément. Il ne resta plus qu'à scotcher le capteur RFID sur la boîte et voilà notre premier détecteur de présence connecté achevé. Fig. 6, 7 et 8.

## Coder avec l'Arduino

### Choix du langage

Arduino est basé sur les langages C/C++. Il est plus aisé de coder avec ces langages sur l'Arduino, mais il est possible d'utiliser d'autres alternatives. Ainsi certains logiciels comme ardublock permettent de décrire visuellement un programme Arduino, sans avoir connaissance du langage de programmation. Cette méthode est en particulier utile pour l'apprentissage de l'Arduino de manière ludique. Aussi, il est possible d'utiliser d'autres langages de programmation (Java, Python...) en utilisant des librairies capables de communiquer par l'intermédiaire des ports série de l'Arduino. Pour notre besoin, nous avons choisi de rester sur le langage C# pour plusieurs raisons :

- La facilité de mise en œuvre, car elle est nativement supportée par l'IDE Arduino ;
- La documentation plus importante et les exemples plus nombreux en langage C# ;
- Le langage bas niveau permettant de mieux voir les interactions entre le code et le matériel (les composants électroniques).

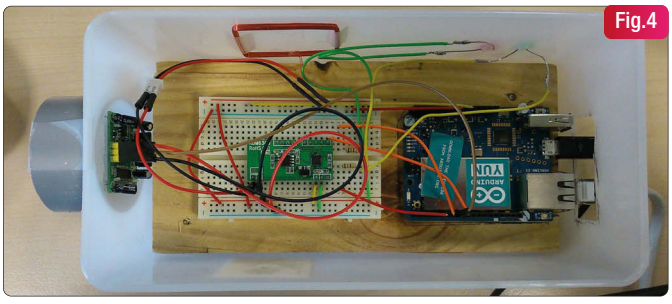


Fig.4


Fig.6


Fig.7

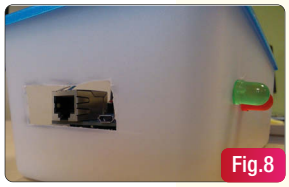


Fig.8

### Environnement de développement

Arduino est constituée du contrôleur matériel, mais aussi d'un environnement de développement, nommé Arduino IDE. Cet IDE propose de décrire nos programmes sous formes de "sketchs".

L'IDE dispose d'exemples typiques de sketchs Arduino et propose également l'import de librairies pour communiquer avec les composants électronique les plus courants. Enfin, l'IDE détecte la carte Arduino branchée en USB sur votre poste, afin d'installer le sketch compilé sur la carte, une fois le développement terminé. L'intégration est simple, en un clic et quelques secondes d'attente, le sketch peut être testé sur l'Arduino !

Un sketch est en fait un programme C# constitué de 2 méthodes principales : setup() et loop(). Vous l'auriez compris, setup() initialise le programme et les composants utilisés. Tandis que loop() est une boucle principale qui contient la logique du programme. Vous l'avez deviné, on n'utilise ici aucune programmation événementielle, on utilise simplement la boucle principale ! C'est old school et très basique.

### Les pièges dans le code

- Le langage

Le langage est sans doute la 1ère difficulté rencontrée. De nos jours, nous utilisons des langages de plus en plus haut niveau, avec plusieurs couches d'abstractions, qui nous habituent à un code concis et efficace. Là, en langage C, c'est plus laborieux. Il faut déjà mettre à jour nos connaissances sur le langage, et les algorithmes qui semblent simples sur le papier sont plus compliqués à mettre en œuvre. Je ne souhaite pas critiquer le langage C en faveur d'un autre langage, mais je sais que l'adaptation au langage C depuis un langage haut niveau (Groovy,JavaScript,etc) est difficile.

- La communication avec les composants électroniques

Les composants électroniques utilisent des standards de communication bien établis et constituent une difficulté majeure. Par exemple, le composant RFID émet un code en une série de bits, qu'il faut décoder selon un standard bien précis : l'interface wiegand. Finalement, nous avons repris du code existant pour décoder ce standard. Voici la méthode en question, ce qui vous fera découvrir ou redécouvrir les joies du langage C !

```
void recevoirCodeRFID()
{
  unsigned char rxByte;
  rxByte = RFID.read();
  if (tagComingFlag == 0 && rxByte == 0x02) {
    tagRX[tagCounter] = rxByte;
    tagComingFlag = 1;
    tagCounter++;
  } else if (tagComingFlag == 1) {
    tagRX[tagCounter] = rxByte;
    tagCounter++;
    if (tagCounter == 14)
    {
```

```
      tagCounter = 0;
      tagComingFlag = 0;
      tagRecdFlag = 1;

      while (RFID.available()>0){
       RFID.read();
      }
      stopMethod = true;
    }
  } else {
    tagComingFlag = 0;
    tagCounter = 0;
  }
}
```

En bref, la méthode détecte l'arrivée d'un octet signalant le début de la transmission du code RFID (ce code de début est 02). Puis on lit octet par octet le code entier. Une fois le code entier lu (14 octets), on vide le buffer du composant avec une série d'appels à read().

- La gestion du temps

Notre Arduino a pour but d'envoyer des données d'activité de l'utilisateur, et a besoin de dater chaque activité utilisateur avant de l'envoyer sur le Cloud. Mais comment avoir accès à la date et l'heure depuis un Arduino ?
Dans un sketch Arduino, on a bien accès à une méthode getMillis(), qui donne le temps écoulé depuis le début du programme ! C'est utile pour avoir un écart de temps, mais insuffisant pour avoir une date complète, avec le jour et le mois. La solution que l'on a trouvée, nous est fournie par le système linux embarqué sur le contrôleur Yun. A chaque démarrage du système, à condition que la carte ait accès à un Wifi, le système synchronise la date avec le protocole NTP, rendant disponible la date dans notre programme. Dans ce cas, un simple appel au système linux de l'Arduino suffit :

```
void recupererDateCourante() {
 p.begin("sh");
 p.addParameter("/sketchArduino/dateCourante.sh");
 p.run();
 dateCourante = "";
 int i =0;
 while( p.available() > 0) {
  char c =p.read();
  if(i<=18) {
   dateCourante += String(c);
  }
  i++;
 }
}
```

### *Le debugging*

Debugger avec l'Arduino n'est pas forcément aisé, dépendant de l'IDE que vous utiliserez. Ayant utilisé seulement l'Arduino IDE de la fondation Arduino, cet IDE ne permet pas le debug. Nous avons donc utilisé l'écriture dans la console avec les Serial.print() ! Mais d'autres IDE proposent le debug de l'Arduino, comme VisualStudio. Donc, avant de vous lancer dans un projet Arduino, choisissez avec attention l'IDE que vous utiliserez, cela peut s'avérer décisif !

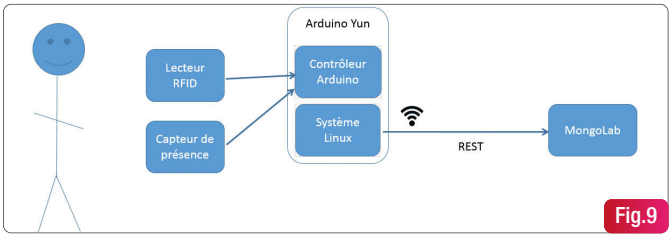


Fig.9

### *Architecture logicielle*

Dans notre cas, l'Arduino sert à récupérer 2 informations : le badge d'un utilisateur et sa présence sur le poste de travail. Ces informations sont remontées par services REST à une base de données noSQL. Voici le schéma représentant l'architecture de notre système. Fig.9. Cette architecture permet de rendre "connecté" cet objet, car il est relié en temps réel à cette base de données. Notre objet enregistre une session de travail, caractérisée par une identification utilisateur (le badge) et le temps de présence de cet utilisateur sur son poste de travail. A chaque fois que la session de travail est terminée, c'est-à-dire quand l'utilisateur a quitté son poste, l'information de session de travail est envoyée sur le réseau. Cette session de travail est un simple objet JSON, contenant les caractéristiques suivantes :

- Identification du badge ;
- Date et heure de début de la session de travail ;
- Date et heure de fin de la session de travail.

### *Restitution des données*

Maintenant que nos sessions de travail sont enregistrées dans la base de données, on peut en faire ce que l'on veut et les restituer sur des interfaces diverses et variées. Dans notre cas, le but était de les afficher sur une application Android et aussi sur un dashboard kibana.
L'application Android cible les utilisateurs des postes de travail, afin qu'ils puissent visualiser leur temps de travail sur les postes pénibles. Ils peuvent ainsi parcourir le temps passé selon une période de temps. Le dashboard Kibana est plutôt destiné à des équipes RH, afin d'avoir une visualisation plus globale du temps passé par tous les employés sur les postes pénibles. Voici les captures d'écran de l'application Android telle qu'elle a été implémentée (les critères type de pénibilité et type de machine ne sont pas implémentés) Fig.10.

## Conclusion

Cette étude nous a permis de monter en compétences sur un certain nombre de technologies liées à l'IOT. Nous avons surtout réalisé qu'un projet IOT englobe un nombre important de technologies et qu'il est intellectuellement enrichissant de travailler dessus. En effet, nous avons aussi bien fait de la conception 3D, du montage électronique ou bien encore des bases NoSQL, ce qui forme une stack complète.
Malgré quelques erreurs, nous sommes parvenus à réaliser un boitier DIY qui permet de comptabiliser le temps de présence sur un poste pénible.
L'étape suivante sera de prendre en compte plus largement l'environnement du poste de travail dit "pénible" (objets lourds, température, bruit, toxicité...). A ces fins, d'autres pistes de mesures peuvent être explorées comme la mesure avec des objets connectés portés par l'employé (wearable) ou bien par les outils manipulés (un objet lourd, un outil dangereux, etc.). L'iOT est un vaste domaine en pleine expansion mais accessible à tout à chacun pour une mise de départ faible. Alors n'hésitez pas et lancez-vous !

Fig.10

# Approche pour faire du développement réseau cross-plateforme avec NeoMAD

2e partie

*Cet article plus technique va présenter comment faire une application réseau très performante avec la librairie NeoMAD. Nous expliquerons par la pratique comment réaliser cette application mobile multiplateforme.*

Alice Barralon
*Scrum Master et consultant indépendant*
a.barralon@oriions.com

## Pourquoi Volley ?

La librairie HTTP proposée par NeoMAD repose sur le même fonctionnement que la méthode *'Volley'* proposée par Android (se référer à la documentation de l'API Android :
http://developer.android.com/training/volley/index.html)
Outre le fait d'être facile, puissant et rapide, *'Volley'* propose de nombreux avantages :

- Mécanisme de thread pool configurable ;
- Planification automatique des requêtes réseau et priorisation (par exemple, on peut vouloir définir une priorité haute pour le téléchargement de données importantes comme les informations du client, et une priorité plus faible pour le téléchargement d'images) ;
- Multiples connexions réseau simultanées ;
- Mise en cache sur le disque et la mémoire se fait automatiquement; de plus elle est standard à la norme HTTP ;
- Gestion des annulations par cas (demande unique ou par bloc) pour éviter que les requêtes ne soient exécutées inutilement ;
- Les résultats des requêtes sont automatiquement mis en cache sur le disque et en mémoire ;
- Les données sont triées, ce qui simplifie l'affichage dans l'interface utilisateur (les données sont récupérées de manière asynchrone à partir du réseau) ;
- Outils de debugging et de tracing du chemin des requêtes.

## Comment ça marche ?

NeoMAD fournit une API (com.neomades.content) qui, sur le même modèle que Volley, va permettre de consommer les data du réseau.
Pour envoyer une simple requête, nous allons créer un *ContentManager*. Celui-ci a pour mission de parser les réponses serveur et de les envoyer, d'où son nom : méthode *Volley*. Pour cela, il va gérer un pool de Threads d'opérations réseau. Chaque thread va lire/écrire dans le cache et parser les réponses retournées par le serveur. Pour envoyer une réponse nous utiliserons la méthode postQuery du content manager comme suit :

```
// Instantiation du content manager
ContentManager manager = new ContentManager();
manager.start();

// Création d'une requête
String url ="http://www.neomades.com";
// Request a response from the provided URL.
ContentQuery query = new ContentQuery(Method.GET, url);
query.setResponseParser(new Parser() {
  public Object deserialize(byte[] data) {
    return new String(data);
  }
});
// used for event type
query.setContentResponseType("neomadesCom");

//Ajout de la requête à la queue gérée automatiquement
manager.postContentQuery(query);

//Gestion des réponses
```

Le *'ContentManager'* propose systématiquement des réponses de type *'ContentResponse'*. Elles sont reçues sur le Thread principal par l'intermédiaire de la méthode *Screen.onContentResponse ()*. Le principal avantage est l'accès direct au Thread principal par l'interface utilisateur. La criticité repose sur la bonne gestion des requêtes d'annulation. Pour information, les informations coûteuses telles que le blocage des entrées/sorties (I/O), le parsing et décodage des flux sont effectués sur le Thread principal.

```
@Override
public void onContentResponse(ContentResponse response) {
// event that corresponds to neomades.com query
  if (response.hasType("neomadesCom")) {
    String responseString = (String)response.getValue();
    textLabel.setText("Response is: "+ responseString);
  }
}

//Gestion des erreurs de requête
@Override
public void onContentError(ContentResponse response) {
textLabel.setText("That didn't work!");
}

//Annulation d'une requête
```

Pour annuler une demande en attente (exemple : l'utilisateur clique sur le bouton retour du navigateur ou annule son action), il faut appeler la méthode OnDestroy()

```
// ...
public static final String TAG = "MyTag";
ContentQuery stringQuery; // Assume this exists.
ContentManager manager;  // Assume this exists.

// Set the tag on the query.
stringQuery.setTag(TAG);

// Add the request to the RequestQueue.
manager.postContentQuery(stringQuery);
```

```
@Override
protected void onDestroy() {
    super.onDestroy();
     postQuery(new CancelQuery(TAG));
}
```

Pour simplifier l'affichage des images dans les listes, NeoMAD propose une API de chargement des images en tant qu'URL. 2 classes servent à manipuler les images :

- ImageUrlLabel ;
- ImageLoader.

## ImageUrlLabel

Permet de télécharger des images à partir d'une URL et de les mettre en cache. Plusieurs avantages sont offerts :

- Optimisation de la taille de l'image;
- Affichage des images d'erreur et de chargement;
- Eviter les requêtes d'images non nécessaires.

En partant du constat que les opérations de gestion d'image sont particulièrement coûteuses (redimensionnement, décodage), elles sont gérées dans un Thread en arrière-plan. Lors du chargement d'une liste avec des images, il peut y avoir par défaut une image dite de chargement (ou autre). Si l'URL ne peut pas être chargée ou lue, alors elle est remplacée par une image d'erreur. ImageUrlLabel gère également l'annulation de demandes en attente si la vue a changé ou que la requête a été annulée. Ceci est très utile quand un utilisateur final fait défiler un ListView qui contient plusieurs vignettes.

## ImageLoader

ImageLoader est un 'chef d'orchestre' pour un grand nombre de ImageQuery, par exemple lors de l'affichage de plusieurs vignettes dans un ListView . ImageLoader fournit un cache en mémoire pour fonctionner en complément du cache traditionnel, ce qui est primordial si on veut éviter le scintillement au niveau de l'écran. ImageLoader permet de fournir des réponses multiples simultanément, ce qui améliore les performances.

## Utilisation

Vous pouvez utiliser ImageLoader et ImageUrlLabel de façon complémentaire pour gérer efficacement l'affichage de plusieurs images, comme par exemple dans un ListView. Dans votre fichier XML de mise en page, vous utilisez ImageUrlLabel de la même manière que vous utilisez imagelabel, par exemple :

```
<ImageUrlLabel
    id="@+id/imageUrlLabel"
    width="60dp"
    height="60dp" />
Affichage d'une image avec ImageLoader
// 20 is the number of image entries to keep in memory
ImageLoader imageLoader = new ImageLoader(new MemoryImageCache(20));
ImageLabel label;
...
label = (ImageUrlLabel) findView(Res.id.imageUrlLabel);

// Get the ImageLoader through your singleton class.
label.setDefaultImage(Res.image.myDefaultImage);
label.setErrorImage(Res.image.myErrorImage);
label.setImageUrl("http://...your image URL...", imageLoader);

mImageLoader = MySingleton.getInstance(this).getImageLoader();
mImageLoader.get(IMAGE_URL, ImageLoader.getImageListener(mImageView,
     R.drawable.def_image, R.drawable.err_image));
```

Voilà, pour une première présentation de la méthode volley qui offre de vraies solutions aux problématiques réseaux sur les différentes plate-formes mobiles supportées par NeoMAD (Android, iOS, Windows Phone, BlackBerry). Pour aller plus loin vous pouvez consulter la documentation en ligne sur le site neomades.com (http://neomades.com/fr/ressources/documentation-generale). N'hésitez pas à poser vos questions aux développeurs avec le skype ID : support.neomades.

# Créer une application de grille sans Excel en C#

*Ce mois-ci, la rubrique « Programmation Windows » ne se code pas en C++ mais en C#. Je vous promets du Level 300 car il y a des threads, de l'accès aux données SQL Server, des appels Web Services WCF et un Timer d'affichage pour le thread principale de l'application.*

Christophe PICHAUD | .NET Rangers by Sogeti
*Consultant sur les technologies Microsoft*
christophepichaud@hotmail.com | www.windowscpp.net


Le sujet de cet article est de créer une application qui représente une feuille Excel avec des données issues de source hétérogènes, des formules calculées mais sans Excel ! Oui, on va tout coder à la main.

## Choisir sa grille

J'entends déjà au loin les sirènes des acheteurs de grilles qui vont s'engouffrer pour me dire d'aller acheter la grille de Telerik, SyncFusion ou Component One... Désolé, le Framework .NET est suffisamment riche pour nous offrir des grilles de très grande qualité et entièrement customisables. Je veux pouvoir choisir la dimension d'une cellule, choisir une police de caractère, choisir une couleur d'arrière-plan ou principale. Et donc en choisissant de rester avec la grille standard du Framework .NET, je suis certain de trouver toute la documentation nécessaire à la construction de mon application. Je n'aurai pas besoin de me balader dans 250 samples de Grilles en cherchant comment faire telle ou telle fonctionnalités... Comme dirait Bill, « Simple is beautiful ». Je veux une rapidité totale d'exécution et en aucun cas un mécanisme de binding évolué. Je peux pouvoir décrire mes colonnes et coder le remplissage des cellules à la main. Et pour faire cela, le contrôle DataGridView fera l'affaire.

## Architecture WinForms ou WPF ?

Sur codeplex.com, il existe une DataGrid dans le projet WPFToolkit mais c'est pour WPF. Je n'ai pas besoin de WPF et de ses mécanismes de binding one-way two-way ou both, je veux du WinForms car c'est performant et qu'il n'y a jamais de trucs tordu. Tout ce qui doit arriver arrive en WinForms ; ce n'est pas le cas du XAML de WPF. Et puis Microsoft ne fabrique pas ses applications en WPF et nous savons pourquoi... Donc voilà, nous allons partir sur une application WinForms et la grille DataGridView du .NET Framework.

## L'application à réaliser

Voici à quoi ressemble ce que nous allons coder : Fig.1.
Je vais faire un zoom sur cette grille qui est moyennement compliquée :

- Il y a des accès aux données SQL Server ;
- Il y a des appels à un Web service Reuters ;
- Il y a des formules calculées ;
- Il y a un thème graphique.

## Par où commence-t-on ?

Tout d'abord, il faut déposer un composant DataGridView sur la Form. Ensuite, on clique sur la petite icône de la grille qui permet de gérer les colonnes. On se retrouve dans l'éditeur de colonnes. Fig.2.
Je positionne 4 choses pour chaque colonne :

- Le header text : le contenu de la cellule ;
- Le name : le nom de la cellule d'entête -> attention, cela sera déclaré dans le fichier de code du designer ;
- Le sort mode : je positionne à NotSortable ;
- La Width : je positionne à 50 ou 100, pas plus.

Une fois que la grille possède une définition de ses colonnes, je vais passer dans le code pour déclarer ses colonnes avec une énumération. Eh oui, dans le code, on va dire je veux la colonne « Devise » et non la colonne 8... On fait les choses bien ; même si on est en C#.

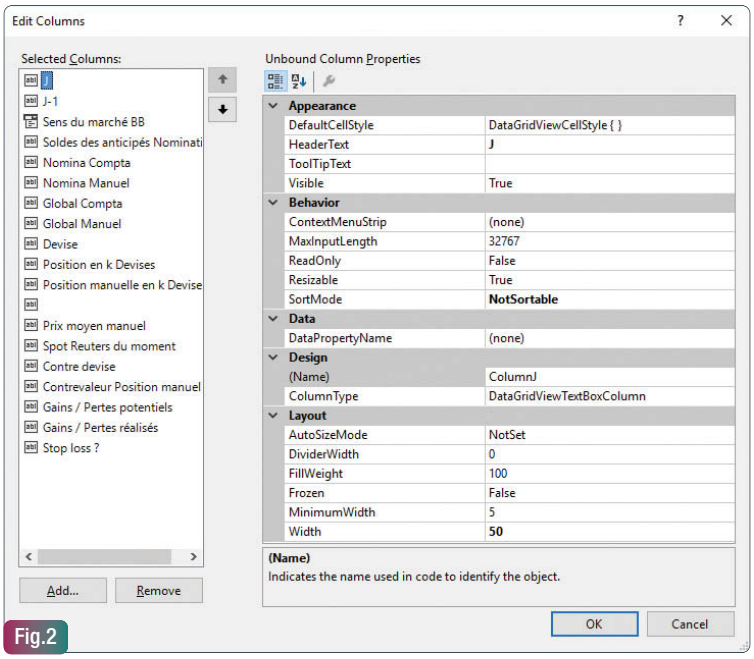
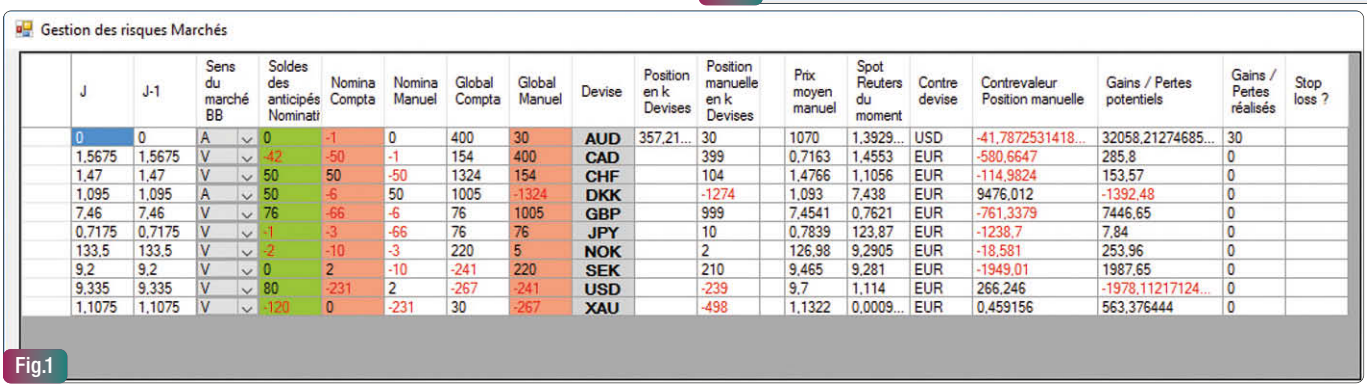


Fig.2




| | J | J-1 | Sens du marché BB | Soldes des anticipés Nominati | Nomina Compta | Nomina Manuel | Global Compta | Global Manuel | Devise | Position en k Devises | Position manuelle en k Devises | Prix moyen manuel | Spot Reuters du moment | Contre devise | Contrevaleur Position manuelle | Gains / Pertes potentiels | Gains / Pertes réalisés | Stop loss ? |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | A | 0 | -1 | 0 | 400 | 30 | AUD | 357,21... | 30 | 1070 | 1,3929... | USD | -41,7872531418... | 32058,21274685... | 30 | |
| | 1,5675 | 1,5675 | V | -42 | -50 | -1 | 154 | 400 | CAD | | 399 | 0,7163 | 1,4553 | EUR | -580,6647 | 285,8 | 0 | |
| | 1,47 | 1,47 | V | 50 | 50 | -50 | 1324 | 154 | CHF | | 104 | 1,4766 | 1,1056 | EUR | -114,9824 | 153,57 | 0 | |
| | 1,095 | 1,095 | A | 50 | -6 | 50 | 1005 | -1324 | DKK | | -1274 | 1,093 | 7,438 | EUR | 9476,012 | -1392,48 | 0 | |
| | 7,46 | 7,46 | V | 76 | -66 | -6 | 76 | 1005 | GBP | | 999 | 7,4541 | 0,7621 | EUR | -761,3379 | 7446,65 | 0 | |
| | 0,7175 | 0,7175 | V | -1 | -3 | -66 | 76 | 76 | JPY | | 10 | 0,7839 | 123,87 | EUR | -1238,7 | 7,84 | 0 | |
| | 133,5 | 133,5 | V | -2 | -10 | -3 | 220 | 5 | NOK | | 2 | 126,98 | 9,2905 | EUR | -18,581 | 253,96 | 0 | |
| | 9,2 | 9,2 | V | 0 | 2 | -10 | -241 | 220 | SEK | | 210 | 9,465 | 9,281 | EUR | -1949,01 | 1987,65 | 0 | |
| | 9,335 | 9,335 | V | 80 | -231 | 2 | -267 | -241 | USD | | -239 | 9,7 | 1,114 | EUR | 266,246 | -1978,11217124... | 0 | |
| | 1,1075 | 1,1075 | V | -120 | 0 | -231 | 30 | -267 | XAU | | -498 | 1,1322 | 0,0009... | EUR | 0,459156 | 563,376444 | 0 | |

Fig.1

```
public enum RowEnum : int
{
   J = 0,
   JMoins1 = 1,
   SensDuMarchéBB = 2,
...
   GainsPertesRealises = 17,
   StopLoss = 18
}
```

Maintenant que la grille est définie ainsi que la position des colonnes, il ne reste plus qu'à définir la classe qui va représenter une ligne de la grille. C'est le même code que l'énumération mais dans une classe :

```
public class DataRowNew
{
   public double J;
   public double JMoins1;
   public string SensDuMarchéBB;
...
   public double GainsPertesRealises;
   public string StopLoss;
}
```

Voilà tout ce dont j'ai besoin pour commencer à alimenter la grille... Maintenant je vous dévoile l'astuce pour que la grille prenne forme. Nous allons recopier l'ensemble des données de la feuille Excel que nous souhaitons remplacer dans la grille. Ensuite, nous allons appuyer sur un bouton qui sérialise la grille sous forme de liste d'éléments de ligne. Nous allons avoir une liste de DataRowNew dans un fichier XML. Nous allons aussi coder la méthode de desérialisation pour charger le fichier XML au démarrage de l'application.

## Load et Save

Le code n'est pas très compliqué ; pour Load, on dépile la liste de DataRowNew du XML et on remplit chaque cellule une par une...

```
   public void Load(string fileName, DataGridView dataGridView)
   {
      List<DataRowNew> dataRows = new List<DataRowNew>();
      XmlSerializer xmls = new XmlSerializer(typeof(List<DataRowNew>));
      StreamReader sr = new StreamReader(fileName);
      dataRows = (List<DataRowNew>)xmls.Deserialize(sr);
      sr.Close();

      int rows = dataRows.Count-1;
      dataGridView.Rows.Clear();
      dataGridView.Rows.Add(rows);

      int count = 0;
      foreach (DataRowNew row in dataRows)
      {
         DataGridViewRow r = dataGridView.Rows[count];
         r.Height = 15;

         dataGridView.Rows[count].Cells[0].Value = row.J;
         dataGridView.Rows[count].Cells[1].Value = row.JMoins1;
         dataGridView.Rows[count].Cells[2].Value = row.SensDuMarchéBB;
...
         dataGridView.Rows[count].Cells[17].Value = row.GainsPertesRealises;
         dataGridView.Rows[count].Cells[18].Value = row.StopLoss;

         count++;
      }
   }
```

La méthode Save fait le contraire :

```
   public void Save(string fileName, DataGridView dataGridView)
   {
      List<DataRowNew> dataRows = new List<DataRowNew>();

      foreach (DataGridViewRow row in dataGridView.Rows)
      {
         int count = 0;
         DataRowNew dataRow = new DataRowNew();
         foreach (DataGridViewCell cell in row.Cells)
         {
            switch (count)
            {
               case 0:
                  dataRow.J = GridHelper.GetDoubleValue(row, RowEnum.J);
                  break;
...
               case 18:
                  dataRow.StopLoss = GridHelper
.GetStringValue(row, RowEnum.StopLoss);
                  break;
            }

            count++;
         }
         dataRows.Add(dataRow);
      }

      XmlSerializer xmls = new XmlSerializer(typeof(List<DataRowNew>));
      StreamWriter sw = new StreamWriter(fileName);
      xmls.Serialize(sw, dataRows);
      sw.Close();
   }
```

Maintenant, on met deux boutons sur la Form pour faire les fonctionnalités de Load & Save. Ainsi, on lance l'application et la grille est vierge. On prend son temps et on recopie toutes les données de la feuille Excel dans la DataGridView .NET. Une fois que c'est terminé, on clique sur le bouton Save et on obtient un fichier XML :

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfDataRowNew xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:
xsd="http://www.w3.org/2001/XMLSchema">
 <DataRowNew>
  <J>0</J>
  <JMoins1>0</JMoins1>
  <SensDuMarchéBB>A</SensDuMarchéBB>
  <SoldeDesAnticipésNominatif>0</SoldeDesAnticipésNominatif>
...
  <GainsPertesPotentiels13>32059.154472969421</GainsPertesPotentiels13>
  <GainsPertesRealises>30</GainsPertesRealises>
  <StopLoss />
```

```
</DataRowNew>
<DataRowNew>
...
```

Au chargement de l'application, on appelle la méthode Load() et la grille est automatiquement remplie... Magic. Ok, la grille est moche. C'est blanc et noir... On va pouvoir attaquer les choses sérieuses...

## L'accès aux données SQL Server

Certaines colonnes vont être remplies par des appels SQL vers SQL Server toutes les 3 secondes. Donc on fait une couche DAL standard. On va faire des requêtes SQL et on va récupérer les données dans des classes C#. Oui Monsieur, à l'ancienne. Pour faciliter l'ouverture des connexions, la fermeture, les accès aux données en général, je vais utiliser le DAAB V2. C'est le Data Access Application Block 2.0 ; c'est en téléchargement sur le Microsoft Download Center. C'est une couche simple qui permet de faire rapidement des appels à SQL Server. C'est la version qui est sortie juste avant le Data Application Block de Enterprise Library... C'est du sans surprise. Allez-y ! On passe la connexion string et c'est tout. On s'applique à fermer le data reader et le tour est joué.

```
public static void LoadNominaCompta()
{
  DB.NominaComptaCollection.Clear();

  string cnx = Settings.GetConnectionString();
  SqlDataReader dr = SqlHelper.ExecuteReader(cnx, CommandType.Text,
"SELECT CODE_DEVISE, POSITION FROM NOMINA_COMPTA");
  while (dr.Read())
  {
    string devise = Convert.ToString(dr["CODE_DEVISE"]);
    double position = Convert.ToDouble(dr["POSITION"]);

    NominaCompta nc = new NominaCompta();
    nc.Devise = devise;
    nc.Position = position;
    DB.NominaComptaCollection.Add(nc);
  }
  dr.Close();
}
```

On remplit une collection en mémoire et il va falloir la charger dans la grille. Continuons de récupérer les données de sources hétérogènes...

## L'accès aux données Web Services Reuters

Pour aller chercher le cours Reuters, j'ai besoin d'un client proxy vers le fichier SVC. Bouton droit « Add Service Reference », je rentre l'URL du service et Visual Studio me génère la référence de service. La méthode CallReutersWS itère dans un dictionnaire de <devise, reutersData>. En effet, avant d'appeler le WS, je dois savoir sur quelles devises je fais une interrogation. Donc, j'ai une routine qui récupère cette information dans la grille.

```
public static void CallReutersWS()
{
  lock (Svc.LockReuters)
  {
    Svc.ReutersDataCollection.Clear();

    foreach (KeyValuePair<string, ReutersData> kvp
in ReutersDataCollectionPrepared)
    {
      try
      {
        FluxFinanciersServiceReference.FluxFinanciersClient ffc
= new FluxFinanciersServiceReference.FluxFinanciersClient();
        ffc.Open();

        string devise = kvp.Key;
        ReutersData rd = kvp.Value;
        FluxFinanciersServiceReference.ResponseStatutCours statusCours
= ffc.GetCoursDeChange(
rd.Devise,
rd.ContreDevise);

        if (rd.SensDuMarche == "A")
          rd.Bid = statusCours.Value.Bid.Value;

        if (rd.SensDuMarche == "V")
          rd.Ask = statusCours.Value.Ask.Value;

        ffc.Close();

        //
        // Store the data
        //

        ReutersDataCollection.Add(devise, rd);
      }
      catch (Exception ex)
      {
      }
    }
  }
```

Pour préparer les données Reuters, je parcours la grille et stocke en mémoire le sens du marché et les deux données de devises :

```
public void PrepareReutersData(DataGridView dataGridView)
{
  lock (Svc.LockReuters)
  {
    Svc.ReutersDataCollectionPrepared.Clear();

    int count = 0;
    foreach (DataGridViewRow row in dataGridView.Rows)
    {
      object objectDeviseInGrid = dataGridView
.Rows[count]
.Cells[Convert.ToInt32(RowEnum.Devise)].Value;
      string deviseInGrid = Convert.ToString(objectDeviseInGrid);
      object objectContreDeviseInGrid = dataGridView
.Rows[count]
.Cells[Convert.ToInt32(RowEnum.ContreDevise)].Value;
      string contreDeviseInGrid
= Convert.ToString(objectContreDeviseInGrid);
      object objectSensDuMarche
= dataGridView
.Rows[count]
```

```
.Cells[Convert.ToInt32(RowEnum.SensDuMarchéBB)].Value;
            string sensDuMarche = Convert.ToString(objectSensDuMarche);

            if (deviseInGrid == String.Empty)
            {
              break;
            }

            //
            // Store Data
            //
            ReutersData rd = new ReutersData();
            rd.SensDuMarche = sensDuMarche;
            rd.Devise = deviseInGrid;
            rd.ContreDevise = contreDeviseInGrid;
            rd.Ask = 0;
            rd.Bid = 0;

            Svc.ReutersDataCollectionPrepared.Add(deviseInGrid, rd);

            count++;
          }
        }
      }
```

Maintenant que nous avons codé les méthodes d'accès aux données, il faut relier cela au DataGridView... ces méthodes doivent être appelées toutes les 2 secondes. Il est improbable de la faire dans le thread principal de l'application. Nous allons utiliser un thread & un timer.

## Acquisition et affichage

Nous allons utiliser un thread pour récupérer les données hétérogènes en arrière-plan et utiliser un timer pour afficher ces informations dans la grille. Le thread principal de l'application est aussi le thread GUI qui possède la grille de message donc il faut faire les affichages dedans. Le thread fait ses appels à SQL Server (méthodes DB.xxx) et l'appel à Reuters (méthode Svc.xxx). Ces méthodes stockent les données en mémoire. Un timer se déclenche toutes les 2 secondes et si le booléen canDisplayData est vrai, il fait les affichages dans la grille DataGridView.

```
public static void ThreadProc(object data)
{
  while (true)
  {
    if (canCloseApp == true)
    {
      // App is closing so we stop the thread !
      break;
    }

    DB.LoadNominaCompta();
    DB.LoadGlobalCompta();
    DB.LoadSoldeAnticipesNominatif();
    DB.LoadNominaAnticipe();
    Svc.CallReutersWS();

    canDisplayData = true;

    Thread.Sleep(2000);
  }
}

public void EnableThread()
{
  Thread thread = new Thread(MainFrame.ThreadProc);
  thread.Start(null);
}


private void _timer_Tick(object sender, EventArgs e)
{
  View view = new View();
  view.PrepareReutersData(dataGridView1);

  if (canDisplayData == true)
  {
    View view2 = new View();
    view2.DisplayNominaCompta(dataGridView1);
    view2.DisplayGlobalCompta(dataGridView1);
    view2.DisplaySoldeAnticipesNominatif(dataGridView1);
    view2.DisplayNominaAnticipe(dataGridView2);
    view2.DisplayReutersData(dataGridView1);

    // Reset the boolean
    canDisplayData = false;
  }
}
```

OK pour le principe. Un thread acquière les données et le timer les affiche. Mais comment se fait l'affichage ? C'est très simple.

```
public void DisplayNominaCompta(DataGridView dataGridView)
{
  int count = 0;
  foreach (NominaCompta nc in DB.NominaComptaCollection)
  {
    dataGridView.Rows[count]
      .Cells[Convert.ToInt32(RowEnum.NominaCompta)]
      .Value = nc.Position;

    dataGridView.Rows[count]
      .Cells[Convert.ToInt32(RowEnum.Devise)]
      .Value = nc.Devise;

    count++;
  }
}
```

Chaque méthode d'affichage est externalisée et prend en paramètre un DataGridView. L'accès aux colonnes se fait via l'énumération RowEnum. A ce stade du développement, les données sont actualisées à intervalle régulier mais c'est toujours de la data en noir et blanc... Et puis, on est censé bypasser Excel et ses formules calculées ; comment faire ?

## Les formules calculées

Ça c'est un sujet intéressant ! Les formules doivent être réactives et il faut faire une liaison entre le code C# qui fait le calcul et la cellule du DataGridView. On fait comment ? La solution : on va se greffer sur l'évènement

OnChange de la grille. Et à chaque fois qu'une cellule est modifiée on va vérifier si une colonne entrant dans un ou des calculs a été modifiée. Si c'est le cas, on exécute la formule. Une formule ça ressemble à ça :

```
public class Formulas
{
    public static void Formula_PositionManuelleEnKDevises8(
DataGridView dataGridView, int col, int row)
    {
      if (col == Convert.ToInt32(RowEnum.NominaManuel)
        || col == Convert.ToInt32(RowEnum.GlobalManuel))
      {
        double nm = GridHelper.GetDoubleValue(dataGridView, row,
          RowEnum.NominaManuel);
        double gm = GridHelper.GetDoubleValue(dataGridView, row,
          RowEnum.GlobalManuel);

        double sum = nm + gm;
        GridHelper.SetDoubleValue(dataGridView, row,
          RowEnum.PositionManuelleEnKDevises8, sum);
      }
    }
```

La classe Formulas contient un ensemble de fonctions static qui prennent en paramètre :

- Un DataGridView ;
- Un indice de colonne ;
- Un indice de ligne .

On fait un test pour savoir si l'indice de colonne qui change est impliqué dans une formule. Si oui, on réapplique la formule et on positionne la valeur dans la grille. Voyons à présent comment est pris en charge l'évènement de modification d'une cellule :

```
    public void EnableCellChanged_DGV1(DataGridView dataGridView1)
    {
      dataGridView1.CellValueChanged += DataGridView1_CellValueChanged;
    }

    private void DataGridView1_CellValueChanged(
object sender,
DataGridViewCellEventArgs e)
    {
      DataGridView dataGridView = (DataGridView)sender;
      int col = e.ColumnIndex;
      int row = e.RowIndex;
      object objValue = dataGridView.Rows[row].Cells[col].Value;
      string value = Convert.ToString(objValue);
      string str = String.Format("row={0} col={1} value={2}", row, col, value);

      Formulas.Formula_PositionEnKDevises7Bis(dataGridView, col, row);
      Formulas.Formula_PositionManuelleEnKDevises8(dataGridView, col, row);
      Formulas.Formula_ContrevaleurPositionManuelle12(dataGridView, col, row);
      Formulas.Formula_CalculGainsPertesPotentiels(dataGridView, col, row);
      //
      // After all of these formulas, Reapply Thèmes.
      //
      ApplyColorTheme_DGV1(dataGridView);
      ApplyColorThemeOnValues_DGV1(dataGridView);
    }
```

L'évènement CellValueChanged permet de réappliquer les formules et aussi, à la fin, d'appliquer un thème graphique sur une cellule. Par exemple, la fonction ApplyColorTheme positionne des colonnes en vert et en rose et les devises en gris. La fonction ApplyColorThemeOnValues permet d'écrire en rouge les chiffres à virgule.

```
    public void ApplyColorThemeOnValues_DGV1(DataGridView dataGridView)
    {
      int rowCount = 10;

      for (int c = 0; c < rowCount; c++)
      {
        AddMinusSignOnRowCell(dataGridView, c, RowEnum.SoldeDesAnticipésNominatif);
        AddMinusSignOnRowCell(dataGridView, c, RowEnum.NominaCompta);
        AddMinusSignOnRowCell(dataGridView, c, RowEnum.NominaManuel);
...
        AddMinusSignOnRowCell(dataGridView, c, RowEnum.GainsPertesRealises);
        AddMinusSignOnRowCell(dataGridView, c, RowEnum.StopLoss);
      }
    }

    public void AddMinusSignOnRowCell(DataGridView dataGridView,
int c, RowEnum enumValue)
    {
      DataGridViewCell cell;

      cell = dataGridView.Rows[c].Cells[Convert.ToInt32(enumValue)];
      Object value = cell.Value;
      string strValue = Convert.ToString(value);
      if (String.IsNullOrEmpty(strValue) == true)
        return;

      double dValue;
      if (Double.TryParse(strValue, out dValue) == false)
        return;

      double doubleValue = Convert.ToDouble(value);
      if (doubleValue < 0)
        cell.Style.ForeColor = Color.Red;
      else
        cell.Style.ForeColor = Color.Black;
    }
```

Pour être capable de gérer les données incorrectes ; faute de frappe et autres ; il faut prendre des précautions lorsque l'on manipule les données de la grille. Il est possible de gérer des formats de cellules mais moi j'ai préféré gérer tout à la main. Dans la grille, tout est Object donc il faut gérer soit même les conversions de types.

# Vos premiers développements sur la blockchain Ethereum

*La blockchain est la technologie qui fait énormément parler d'elle en ce moment. Que vous n'en ayez jamais entendu parler, que vous soyez sceptique et tout simplement curieux, je vous propose d'explorer les possibilités de la blockchain Ethereum, par la pratique. Nous commencerons par survoler les principes de base de la blockchain, puis nous mettrons en place l'environnement de travail, de l'installation du client Geth à la configuration d'un réseau blockchain privé à 4 nœuds en local. Enfin, nous terminerons par la création d'un contrat Ethereum.*

Guillaume Nicolas,
*Ingénieur développeur à*
**SQLI Nantes**

## La blockchain ?

La blockchain est née en 2009 à la création de la crypto-monnaie Bitcoin. Son but était de combler les lacunes des précédentes crypto-monnaies et de mettre en place la gestion d'une monnaie de manière totalement décentralisée. La décentralisation nécessite l'entente entre les parties. En effet, le problème majeur lorsque l'on souhaite gérer un registre de données dans un système sans autorité centrale, c'est comment mettre d'accord les différents acteurs ? Le principe d'un système réparti, c'est que chaque partie possède les données, et ensemble, elles doivent se mettre d'accord sur les modifications de ces dernières. Alors comment être sûr qu'une partie ne va pas tricher en modifiant ses données ? Qu'est-ce qui légitime l'ajout de données dans le registre ? Sa validité ? C'est à ce niveau qu'intervient la blockchain. La chaîne de blocs est une succession de blocs horodatés, contenant des données et qui doivent être validés par consensus par les nœuds du réseau distribué.
Actuellement, sur la plupart des plateformes, la preuve de validité d'un bloc est appelée preuve de travail (Proof-of-Work). Trouver cette preuve demande énormément de ressources. C'est pourquoi, à chaque bloc ajouté à la blockchain, le mineur se voit verser un certain montant de crypto-monnaie, fraîchement créée. La complexité ainsi que le chaînage des blocs les uns à la suite des autres font qu'il est difficile avec les méthodes d'aujourd'hui de falsifier ou supprimer les données de la blockchain. Pour modifier une donnée d'un bloc, il faut refaire valider l'ensemble des blocs qui le suivent, avant qu'un autre nœud n'ajoute un nouveau bloc à la chaîne. Les nœuds qui cherchent la preuve de validité sont appelés les mineurs et l'opération s'appelle le minage.
Les données inscrites à l'intérieur des blocs peuvent être de différentes natures suivant la blockchain. Elles sont toutes stockées dans des transactions distinctes. Une transaction est un "échange" faisant intervenir plusieurs parties entres elles. Dans la première "version" des blockchains (type Bitcoin), les données sont des transactions financières entre plusieurs comptes. Il est également possible d'inscrire de la donnée en annexe (Cf. Namecoin). Mais c'est la seconde génération de blockchain qui possède le plus gros potentiel. Elle est née fin 2013 avec l'apparition d'Ethereum. Ethereum propose de stocker dans la blockchain, du code exécutable par une machine virtuelle (EVM) présente dans chaque nœud. Ce code exécutable est appelé contrat et ouvre la porte à de nombreuses possibilités.
De la même manière que Bitcoin permet de se passer d'intermédiaires financiers en connectant les demandeurs et bénéficiaires entres eux, Ethereum permet de se passer de l'ensemble des intermédiaires dont le travail peut être automatisé par du code fonctionnel. Il est possible d'imaginer le prochain Uber sur la blockchain, le prochain Airbnb, mais également d'automatiser les remboursements santé ou encore la déclaration et le paiement des impôts. Supprimant ainsi toute possibilité de fraude. En liant cette technologie avec le monde de l'IoT, c'est toute une société qui peut évoluer. Avec la blockchain, le monde de demain se veut plus transparent. Mais assez bavardé, passons à la pratique !

## Objectif et architecture

Avant de commencer à manipuler notre client Ethereum, je vais vous présenter l'objectif de cette mise en pratique. Par défaut, le client se connecte automatiquement sur le réseau principal : Homestead. Chaque action effectuée sur ce réseau nécessite de l'ether (la crypto-monnaie d'Ethereum). L'ether ne peut s'obtenir qu'en minant, ou en contrepartie d'un autre élément de valeur (service, produit, devise, ...). Le but n'étant pas de vous ruiner, ni de passer plusieurs années à miner pour gagner quelques unités, nous allons créer notre propre réseau privé, isolé de l'instance publique de la blockchain Ethereum.
Le réseau sera composé de plusieurs nœuds. Chaque nœud sera représenté par une instance de notre client. Pour avoir 4 nœuds, je vais donc lancer 4 fois le client, sur des processus différents. Nos 4 nœuds vont communiquer ensemble sur le réseau local de notre ordinateur. Nous pourrons interagir avec nos nœuds, et leur demander d'effectuer des actions, grâce à l'environnement Javascript en ligne de commande intégrée au client. Pour simuler le comportement de nœuds indépendants, chaque nœud travaillera dans un répertoire séparé. Ce répertoire contiendra la blockchain, ainsi que des informations liées au nœud. Comme son identité, ou les comptes qu'il héberge. Voici un schéma résumant ce que nous allons mettre en place (Fig.1)

## Installation

Commençons tout d'abord par l'installation du client qui servira à communiquer avec notre réseau privé. La communauté Ethereum étant très active, des clients sont implémentés sur bon nombre de technologies. On y retrouve Eth (C++), Geth (Golang), Pyethapp (Python), Parity (Rust), EthereumJS (Javascript) ou encore ethereum-haskell (Haskell). Dans cette introduction, nous utiliserons le client Geth, implémenté en Golang, et travaillerons sous Ubuntu 16.04 LTS. Geth est le client le plus utilisé sur le réseau principal d'Ethereum : Homestead. Pour l'installer, il suffit de mettre à jour le gestionnaire de paquet apt et d'ajouter les dépôts d'ethereum à la liste des paquets :

```
apt-get update
apt-get install -y software-properties-common
add-apt-repository -y ppa:ethereum/ethereum
```

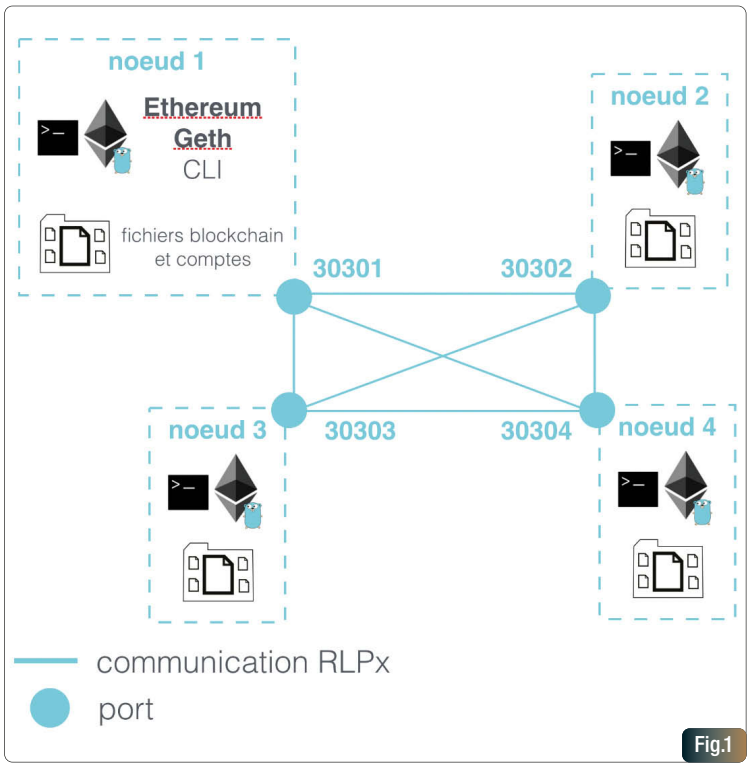



Fig.1

*Architecture de l'environnement*

```
add-apt-repository -y ppa:ethereum/ethereum-dev
apt-get update
```

Nous pouvons à présent installer geth ainsi que le compilateur solidity (solc) :

```
apt-get install –y ethereum solc
```

## Initialisation de la blockchain

Pour être sûr de pouvoir évoluer dans un environnement contrôlé, nous allons mettre en place une blockchain privée. À ce stade, la blockchain n'existe pas encore. Il faut donc créer manuellement le premier bloc de cette chaîne. Ce premier bloc contient l'ensemble des caractéristiques de départ de notre blockchain et doit être partagé à l'ensemble des noeuds. Pour définir ce bloc, il faut créer un fichier au format JSON contenant cette structure :

```
{
  "nonce": "0x00",
  "difficulty": "0x1000",
  "mixHash": "0x00",
  "timestamp": "0x00",
  "parentHash": "0x00",
  "extraData": "0x00",
  "gasLimit": "0x1000000000"
}
```

Plusieurs paramètres sont à renseigner :

- Le **nonce** est le hash que le mineur va devoir faire varier et trouver pour résoudre la preuve de travail. Il est complémentaire au *mixhash* et sert à vérifier que le bloc est valide.
- **Mixhash** est le hash de l'entête du bloc, sur lequel va se baser le mineur pour trouver le nonce.
- Le **timestamp** permet au client de réguler le temps entre la validation de 2 blocs successifs et de vérifier le bon enchaînement des blocs. Il est initialisé à 0 pour indiquer qu'il s'agit du premier bloc.
- Le **parentHash** est le hash du bloc précédent. Etant donné qu'il s'agit du premier bloc, ce hash vaut 0.
- **ExtraData** est facultatif et permet de stocker de la données (32 octets max), sur la blockchain.
- **GasLimit** est le montant maximum de *gas* que les contrats peuvent consommer. Au dessus de la limite, le contrat ne pourra pas être ajouté au bloc. Nous reverrons la notion de gas lors de la création de notre premier contrat.
- **Difficulty** correspond à la difficulté de la preuve de travail de ce bloc. Elle a un impact sur la rapidité de validation des blocs et donc la création d'ether. Nous mettons une valeur faible pour ne pas perdre de temps sur la validation de nos blocs et générer rapidement de l'ether. Cette valeur, couplée au timestamp, va permettre au client de fixer la difficulté de la prochaine preuve de travail, pour conserver un temps de validation linéaire.

Une fois ce fichier renseigné, nous pouvons demander au client de créer le dossier contenant la blockchain (*chaindata*) et de l'initialiser.

```
geth --datadir ./noeud1 --networkid "100" init genesis.json
```

*Networkid* est le flag qui va identifier notre réseau et permettre aux pairs de nous rejoindre. Étant donné que nous sommes sur un réseau privé, sa valeur n'a pas d'importance. Elle doit cependant être différente des identifiants des réseaux publics (> 2). À partir de maintenant, les flags *datadir* et *networkid* seront obligatoires pour appeler le client geth, car ils identifient respectivement le répertoire de travail et l'identifiant du réseau. Pour pouvoir faire tourner 4 nœuds sur le réseau local de notre poste, l'astuce est simplement de changer le répertoire des données (*datadir*). Nous initialisons donc la blockchain des 3 autres nœuds en exécutant 3 fois la commande précédente et en modifiant le datadir par ./noeud{2,3,4}.

## Création d'un compte

Réaliser des transactions, créer et interagir avec des contrats, nécessite un compte contenant de l'ether. Voici comment en créer un nouveau.

```
geth --datadir ./noeud{1-4} --networkid "100" account new
```

La console vous demande de saisir un mot de passe. Ce sera le seul moyen d'accès au porte-monnaie et pour réaliser des transactions. Le message retourné correspond à votre clé publique. C'est celle qui identifie votre compte et qui vous identifie sur la blockchain.

```
Address: {620a3e52cb988acc3b985475b0639bdb49577d44}
```

Le client crée le répertoire *keystore* qui sert à conserver les identifiants de votre compte (clé privée et clé publique). Ces données sont chiffrées (AES 128-bit) avec votre mot de passe. Votre mot de passe est donc la seule donnée que vous ne devez absolument pas divulguer.
Répétez cette opération dans les 3 répertoires de données pour créer un compte sur chaque nœud.

## Lancer le client

A présent, lançons nos nœuds (de 1 à 4) avec les différents paramètres relatifs au schéma Fig.1 :

```
geth --datadir ./noeud{1-4} --networkid "100" --port "3030{1-4}" console
```

- **Port** : définit le numéro de port sur lequel les autres nœuds viendront se connecter. On utilise un port différent pour chaque nœud car tous les nœuds seront sur la même adresse (127.0.0.1).
- **Console** : lance un environnement Javascript possédant l'ensemble des outils d'interaction avec le nœud.

**Info:** Il est également possible d'interagir avec un nœud via http, websocket et ipc.
Vous pouvez à présent demander des informations sur le nœud.

```
> admin.nodeInfo
{
  enode: "enode://ff581ffb37a1f387e50c8e17e103cfcf0caadc32cd9b6ff4f1762edff2a5d5ca317a186aab68714c8ec77db854de88fb6683f8361f27c9078461521869f1dff1@[::]:30301",
  id: "ff581ffb37a1f387e50c8e17e103cfcf0caadc32cd9b6ff4f1762edff2a5d5ca317a186aab68714c8ec77db854de88fb6683f8361f27c9078461521869f1dff1",
  ip: "::",
  listenAddr: "[::]:30301",
  name: "Geth/v1.5.0-unstable/linux/go1.6.1",
  ports: {
    discovery: 30301,
    listener: 30301
  },
  protocols: {
    eth: {
      difficulty: 4096,
      genesis: "0x5a65940292cbb5cefe79c39624d6dd228a696c96cc86ead9bde58f94cda96da3",
      head: "0x5a65940292cbb5cefe79c39624d6dd228a696c96cc86ead9bde58f94cda96da3",
      network: 100
    }
  }
}
```

L'objet Javascript retourné comporte plusieurs informations, comme l'identifiant du nœud (*enode*) ou encore les paramètres de la blockchain (*protocols.eth*).

## Connecter les nœuds

Pour le moment, nos 4 nœuds se trouvent sur le même réseau et possèdent la même base de blockchain. Mais ils ne se connaissent pas et ne sont pas capables de communiquer ou même de synchroniser leur blockchain. Étant donné qu'il s'agit d'un réseau pair à pair, il faut connecter les nœuds entres eux. Dans ethereum, chaque nœud est identifié par un *enode*. Pour que le client se connecte à un autre nœud sur le même réseau, il faut récupérer son identifiant (> admin.nodeInfo.enode) et appeler la méthode :

```
> admin.addPeer({enode du nœud sur lequel se connecter})
```

Vous pouvez vérifier que vous êtes bien connecté à des nœuds en affichant leurs informations.

```
> admin.peers
```

Une liste contenant les *admin.nodeInfo* des nœuds connectés doit être retournée.
A partir de maintenant, vos nœuds sont connectés, et vont pouvoir commencer à synchroniser leur blockchain.

## Miner

Vient le moment de gagner un peu de monnaie, pour pouvoir faire vivre notre blockchain privée.
Pour cela, il faut demander à notre client de démarrer le minage. Un bloc contient un nombre quelconque de transactions. Le mineur récupère les transactions en attente sur le réseau, et commence à chercher la preuve de validité du bloc. Pour simplifier, ce travail consiste à trouver une valeur qui respecte une condition particulière parmi un très grand ensemble de possibilités. Pour cela il réalise plusieurs tentatives, jusqu'à trouver la solution. Une fois qu'il pense avoir trouvé une solution, il la soumet au vote des autres nœuds du réseau. Par consensus, si le réseau affirme qu'il s'agit bien de la solution, alors le bloc est ajouté à la blockchain. Il est important de préciser qu'il est très difficile de trouver une solution, mais que la validation de la solution est immédiate.
Pour démarrer le mineur, on exécute donc la méthode

```
> miner.start()
I0513 13:52:57.431007 miner/miner.go:119] Starting mining operation (CPU=8 TOT=9)
I0513 13:52:57.431108 eth/backend.go:450] Automatic pregeneration of ethash DAG ON (ethash dir: /root/.ethash)
I0513 13:52:57.431131 eth/backend.go:457] checking DAG (ethash dir: /root/.ethash)
I0513 13:52:57.431141 miner/worker.go:559] commit new work on block 1 with 0 txs & 0 uncles. Took 112.309µs
I0513 13:52:57.431171 ethash.go:259] Generating DAG for epoch 0 (size 1073739904) (0000000000000000000000000000000000000000000000000000000000000000)I0513 13:52:58.091417 ethash.go:291] Generating DAG: 0%
I0513 13:53:00.671188 ethash.go:291] Generating DAG: 1%
…
I0513 13:57:41.525050 ethash.go:291] Generating DAG: 100%
I0513 13:57:41.526461 ethash.go:276] Done generating DAG for epoch 0, it took 4m44.095291362s

I0513 13:58:18.849764 miner/miner.go:119] Starting mining operation (CPU=8 TOT=9)
I0513 13:58:18.849875 miner/worker.go:559] commit new work on block 1 with 0 txs & 0 uncles. Took 85.301µs
I0513 13:58:19.926509 miner/worker.go:341]    Mined block (#1 / 901cea76). Wait 5 blocks for confirmation
> miner.stop()
```

Au premier démarrage du mineur, un fichier d'environ 1Go est généré par le client (le DAG), il sert au mineur dans sa recherche de solution. Cette phase est relativement longue. Pendant ce temps, le mineur ne travaille pas. Vous remarquerez que le temps pour construire un bloc est vraiment très court (quelques micro-secondes). Cela vient du fait que notre bloc genèse possède une difficulté extrêmement faible. Cela nous permet de réaliser n'importe quelle action sur notre réseau de test, très rapidement. L'argent gagné par le mineur est versé par défaut sur le premier compte créé sur le nœud. Comme nous n'avons créé qu'un seul compte sur chaque nœud, ce dernier recevra automatiquement la rémunération du mineur. Nous pouvons consulter le solde du premier compte du nœud.

```
> web3.fromWei(eth.getBalance(eth.accounts[0]),'ether')
5
```

Comme affiché dans la console, notre compte possède 5 unités de la monnaie (ether). Il s'agit d'une grosse somme sachant qu'une transaction coûte une très petite fraction d'ether.
Vous pouvez à présent faire miner vos autres nœuds.

## Echanger de la monnaie

De la même manière que pour la plateforme Bitcoin, la plateforme Ethereum est capable d'inscrire sur la blockchain des transactions financières.
Nous allons donc réaliser notre première transaction. Je vais envoyer 1 unité d'ether du compte sur le nœud 1 au compte sur nœud 2.
Afin de pouvoir envoyer une transaction, il est nécessaire de s'authentifier auprès du client. Pour cela, il faut saisir :

```
> personal.unlockAccount(eth.accounts[0],"monmdp",60)
```

Le dernier paramètre est facultatif et indique la durée de l'authentification en secondes.
Ensuite, je peux créer ma transaction et la soumettre au réseau.

```
> eth.sendTransaction({from:eth.accounts[0],to:"0x8baf922d903d3e3ef49082fb07d973
116effdbe0",value: web3.toWei(1,"ether"),data:"cadeau"})
0x9bf31717a6e78ae4b7b58ab25ab6fb60105a0b27c02c053e1dcfd2545c0c899c
```

À la création de la transaction, j'indique le compte débité (*from*), le compte crédité (*to*) et le montant (*value*). J'ai également ajouté une donnée personnelle (*data,* facultatif), qui sera inscrite dans la blockchain et que le compte crédité pourra lire.
La valeur retournée n'est autre que l'identifiant de la transaction. À partir de cet identifiant il est possible de récupérer toutes les informations de la transaction, ainsi que son état (présente sur la blockchain ou en attente). Si un mineur était actif sur le réseau, la transaction a été ajoutée à la blockchain. Si le nœud 2 synchronise sa version locale, il verra son solde passer de 0 à 1.

## Créer notre premier contrat

L'intérêt de la plateforme Ethereum est de pouvoir créer du code exécutable sur la blockchain, appelé *smart-contract*. Pour cela, je vous propose un exemple de contrat très simple : la satisfaction client. Qui n'a jamais suspecté que l'indice satisfaction client affiché sur un produit ou un service n'était pas exactement celui exprimé par les clients ? Grâce à la blockchain, et au contrat que nous allons rédiger, nous nous assurerons que le taux affiché est bien réel.
Pour commencer, rédigeons les lignes de notre contrat à l'aide de Solidity, le langage de programmation dédié.

```
contract SatisfactionClient {

  /* Hash de "service après ventes " */
  bytes32 constant SUJET = 0x428dddc2ccc007e2a6251a8e2d9ff928061f5fc5f475a9a4e3
a6d9ba8ddf4807;

  uint constant SEUILFAIBLE = 40;
  uint constant SEUILFORT = 80;

  string constant FAIBLE = "FAIBLE";
  string constant MOYEN = "MOYEN";
  string constant FORT = "FORT";

  uint nbClients = 0;
  uint satisfaction = 0;

  mapping(address => bool) aSoumis;

  function soumettreSatisfaction(uint s){
    if( aSoumis[msg.sender] || s<0 || s>100 ) throw;
    aSoumis[msg.sender] = true;
    satisfaction += s;
    nbClients += 1;
  }

  function satisfactionProduit() returns (uint256,string){
    uint256 taux = satisfaction/nbClients;
    if( taux < SEUILFAIBLE ) return (taux,FAIBLE);
    if( taux >= SEUILFORT ) return (taux,FORT);
    else return (taux,MOYEN);
  }

}
```

Solidity représente un contrat comme un objet comportant des variables d'état et des fonctions. La syntaxe est relativement simple, et similaire à un langage de programmation orienté objet. Chaque variable est statiquement typée.
La première variable du contrat est le *hash* du sujet de satisfaction client. Sa valeur est déterminée au préalable en passant la fonction web3.sha3 ("service après-vente") dans la console du client Geth. La valeur retournée fait 32 octets et est au format hexadécimal. De cette manière, on ne stocke pas la chaîne "service après-vente", mais son hash pour éviter le problème d'encodage et figer le sujet.
Deux autres constantes sont utilisées pour faciliter la compréhension. Les seuils (SEUILFAIBLE et SEUILFORT) permettent de retourner l'état de la satisfaction, sous la forme d'un des mots clés. Les 3 constantes suivantes sont les chaînes de caractères (string), associées à chaque état du taux de satisfaction (FAIBLE/MOYEN/FORT).
Nous stockons également le nombre de clients qui ont soumis une valeur et la somme des satisfactions. Ces variables sont de type unsigned int (uint) et initialisées à 0.

La dernière variable d'état est un dictionnaire, c'est-à-dire un tableau qui pour n'importe quelle clé, retourne une valeur. La particularité est que le *mapping* ne possède pas de taille. Il initialise virtuellement pour chaque clé possible (en fonction du type), une valeur initiale correspondante à 0. Il n'est donc pas possible d'ajouter ou de supprimer des clés. Il est simplement possible de modifier la valeur associée à une clé. Ici, notre mapping est de type (address => bool). Elle stocke donc pour chaque adresse (type spécial qui représente une adresse Ethereum), un booléen à vrai si l'adresse a déjà soumis sa satisfaction, ou faux dans le cas contraire. Ce système permet d'ôter la possibilité de soumettre plusieurs fois son avis (du moins avec la même adresse).
Nous n'avons pas défini de constructeur car aucune action particulière n'est nécessaire à l'initialisation du contrat. Le constructeur, comme dans beaucoup d'autres langages, est une fonction (Cf. paragraphe suivant), qui possède le même nom que celui défini après le mot clé "contract".
La première fonction soumettreSatisfaction permet à l'appelant de soumettre une valeur de satisfaction entre 0 et 100 qui sera enregistrée sur la blockchain et servira au calcul de la moyenne de satisfaction. Nous pouvons remarquer le mot clé **throw** qui est (au moment où j'écris cet article), la seule manière pour faire remonter une erreur à l'émetteur. Appelé, il engendre le remboursement des frais associés à la transaction et annule toutes les actions sur les données. Dans notre cas, si l'appelant

fournit un paramètre erroné, ou a déjà soumis son avis, alors il ne paiera pas les frais.
La seconde et dernière fonction satisfactionProduit permet de récupérer le taux de satisfaction ainsi que le mot clé associé. Le taux est donc calculé par la machine virtuelle Ethereum. Il est intéressant de remarquer qu'une fonction peut retourner plusieurs paramètres.
Le contrat étant à présent défini, il est temps de le partager avec l'un de nos nœuds. Pour cela commençons par ajouter notre code dans la console Javascript de l'un de nos nœuds. Nous définissons la variable "source" contenant le code source sous la forme d'une string. Attention, le code doit être sur une seule ligne.

```
> sources = 'contract SatisfactionClient {bytes32…'
```

Ensuite nous devons compiler le code Solidity avec un compilateur adapté. J'ai installé **solc** en tout début d'article. Je peux donc utiliser les outils prévus par le client et stocker le tout dans une autre variable "compiled".

```
> compiled = eth.compile.solidity(sources)
I0524 11:25:33.542102 common/compiler/solidity.go:114] solc, the solidity compiler commandline interface
Version: 0.3.2-0/RelWithDebInfo-Linux/g++/Interpreter

path: /usr/bin/solc
{
 SatisfactionClient: {
  code: "0x60[..]256",
  info: {
   abiDefinition: [{...}, {...}],
   compilerOptions: "--bin --abi --userdoc --devdoc --add-std --optimize -o /tmp/solc730488666",
   compilerVersion: "0.3.2",
   developerDoc: {
    methods: {}
   },
   language: "Solidity",
   languageVersion: "0.3.2",
   source: "contract SatisfactionClient{ [...] }",
   userDoc: {
    methods: {}
   }
  }
 }
}
```

Nous pouvons voir que l'objet Javascript retourné est composé entre autres du code hexadécimal, d'informations sur le compilateur et des sources.
A présent, nous pouvons instancier notre contrat de cette manière :

```
monContrat = eth.contract(compiled.SatisfactionClient.info.abiDefinition).new({from: eth.accounts[0], data: compiled.SatisfactionClient.code,gas:250000})
```

Cette opération est découpée en 2 parties. Dans un premier temps on crée un objet Javascript contrat (*eth.contract*), comportant toutes les spécificités d'un objet contrat mais également l'interface du contrat que nous avons écrit (*abiDefinition*). L'interface correspond à l'ensemble des éléments pouvant être appelés depuis l'extérieur du contrat (les 2 fonctions expliquées précédemment). Dans un second temps, il crée une nouvelle instance du contrat (mot clé **new**, similaire au Java). Nous passons en paramètre le compte permettant d'envoyer le contrat sur la blockchain (*from*), le code compilé du contrat (*data*) ainsi que le montant de gas à fournir pour faire fonctionner le contrat (*gas*). La valeur de gas donnée est relative à celle estimée par la fonction eth.estimateGas(). Le gas est une notion ajoutée à Ethereum pour éviter l'exécution de boucles infinies dans la machine virtuelle. Si une personne souhaite rendre les nœuds du réseau indisponibles pendant un long moment, il doit en payer le prix fort. Car une unité de gas possède un prix (variable) en ether, que le compte doit payer à l'appel d'une fonction d'un contrat.
Pour pouvoir envoyer la transaction, et donc exécuter la commande ci-dessus, il ne faut pas oublier de débloquer le compte spécifié par le paramètre "from" grâce à la commande "personal.unlockAccount()".
Comme il s'agit d'une transaction, pour appeler le contrat, il faut l'ajouter à la blockchain et donc le miner avec miner.start() et que les autres nœuds se synchronisent.
Pour appeler le contrat depuis un autre nœud, il faut récupérer l'instance, grâce à 2 informations : l'interface (monContrat.abi) et l'adresse du contrat (monContrat.address), disponibles sur le nœud qui l'a créé.

```
abi = [{constant: false, inputs: [{name: "s", ...
eth.contract(abi).at("0xb53fba34ddcfdbd642d52ac9dcc0ab8bce36a79e")
```

Nous pouvons à présent appeler l'une des fonctions disponibles, comme par exemple, soumettre notre avis à partir de n'importe quel noeud.

```
monContrat.soumettreSatisfaction.sendTransaction(70,{from:eth.accounts[0]})
```

Le premier paramètre de sendTransaction est la valeur du paramètre de soumettreSatisfaction, et le second est l'objet Javascript qui comporte seulement l'adresse du compte avec lequel on souhaite nous exprimer.
Ensuite, comme à chaque fois, il faut miner la transaction et la synchroniser sur nos autres nœuds pour pouvoir vérifier qu'elle s'est bien propagée.
Si maintenant nous appelons dans un autre nœud (qui n'a pas encore soumis son avis), qui a synchronisé sa blockchain et qui possède l'objet instance du contrat :

```
> monContrat.satisfactionProduit.call()
[70, "MOYEN"]
```

Nous faisons appel à notre contrat via un fonction différente cette fois ci : call. Elle permet de ne pas créer une transaction et d'exécuter la fonction sur notre blockchain locale sans payer de frais. À la fin de la fonction, les données qu'elle modifie seront remises à leur état avant exécution, comme si l'appel n'avait jamais eu lieu. Elle nous retourne un tableau contenant les 2 variables de retour. Nous pouvons voir que la satisfaction est moyenne. Répétons l'opération sur un autre nœud et revérifions.

```
> monContrat.soumettreSatisfaction.sendTransaction(10,{from:eth.coinbase})
> monContrat.satisfactionProduit.call()
[40, "MOYEN"]
```

Félicitations, vous êtes à présent prêt pour développer et déployer vos propres contrats sur la blockchain Ethereum !

# Lagom : framework de micro-services

*Le paysage du développement logiciel a été envahi ces derniers temps par une nouvelle tendance : « les micro-services ». Il existe plusieurs frameworks proposant la mise en oeuvre de ce type d'architecture. Dans cet article, nous allons découvrir le framework proposé par Lightbend : « Lagom ». Lagom met en oeuvre un certain nombre de concepts qu'il est important de comprendre pour aborder ce framework.*

Fabrice Sznajderman
*Développeur Scala/Java/web at* **Zenika**
@fsznajderman

Nous allons commencer cet article par un rappel sur les architectures microservices, l'approche CQRS et Event sourcing. En effet, ces deux derniers concepts font partie de l'ADN de Lagom. Ensuite, nous découvrirons le framework Lagom, la philosophie et les technologies mises en oeuvre. Enfin nous mettrons en oeuvre le framework au travers d'un exemple.

## Les concepts

Dans ce chapitre, nous allons faire quelques rappels sur des concepts clefs sur lesquels Lagom s'appuie. Il est important de bien les comprendre afin de pouvoir appréhender plus facilement le framework.

### Architecture micro-services

Depuis quelques temps, les architectures microservices sont apparues et se mettent en opposition aux architectures monolithiques classiques que l'on connaît bien.

*Pour rappel, les architectures monolithiques sont les bonnes vieilles applications livrées sous forme de WAR ou d'EAR.* Fig.1.

Le premier point que l'on peut reprocher à l'approche monolithique est la concentration de l'ensemble des fonctionnalités dans une seule application. Dans le temps, cette concentration fonctionnelle va accroître la complexité de l'application et poser des difficultés aux nouveaux arrivants qui mettront un temps supplémentaire pour appréhender le projet. La caractéristique monolithique a également comme effet de mettre en oeuvre une seule technologie (silver bullet?). Les points cités ci-dessus sont suffisamment significatifs pour que l'on puisse commencer à s'intéresser à de nouvelles architectures, comme par exemple les microservices.

### Définition

Pour commencer, nous allons voir une définition d'une architecture microservices. Cette définition est tirée du livre : Reactive Microservices Architecture: Design Principles for Distributed Systems :

> *"Microservices-Based Architecture is a simple concept : it advocates creating a system from **a collection of small**, **isolated services**, each of which **owns their data**, and is **independently isolated, scalable and resilient to failure**.*
> *Services integrate with other services in order to form a **cohesive** system that's far more flexible than the typical enterprise systems we build today."*
> *James Boners*

En substance, l'auteur explique qu'une architecture microservices est un ensemble de « petits » services isolés responsables de leurs données. Ils sont indépendants, scalables et résilients aux pannes. L'ensemble de ces services forme un système cohérent. Par rapport à cette définition, nous allons voir comment on peut déterminer si un microservice est correct. Pour déterminer cela au travers de 3 questions :

1 - Est-ce que mon service fait une chose et une seule?
2 - Est-ce que mon service est autonome?
3 - Est-ce que mon service possède ses propres donnés?

### Questions?

Est-ce que mon service fait une chose et une seule ? Fig.2.

A travers cette question, on va chercher à déterminer si l'objectif du microservice est d'adresser une fonctionnalité unique.

Pour répondre à cette question de manière positive, il faut être capable de d'exprimer la raison d'être du microservice en une phrase simple et concise :

*Ce service gère les utilisateurs.*

Par cette phrase on se rend compte que le périmètre du service est circonscrit à la gestion des utilisateurs et rien d'autre. Première condition validée pour notre service.

Est-ce que mon service est autonome? Fig.3.

Ici, nous allons chercher à savoir si notre service est autonome et ne dépend pas d'un autre service pour fonctionner. Un service est responsable de son comportement. Il ne doit en aucun cas, être dépendant de l'état d'un autre service.

Par exemple, imaginons que nous ayons un service qui gère des commandes. Pour pouvoir confirmer une commande, il faut avoir la confirmation du service de paiement que le règlement de la commande soit fait avec succès.

Au moment où la demande de confirmation arrive sur notre service de commande le service de paiement est KO.

Dans ce cas-là, le service de commande doit quand même pouvoir accepter la demande de confirmation bien que le service de paiement ne soit pas disponible. Ceci montre que le service de commande est autonome et que son état ne dépend pas de l'état d'un autre service.

Dans l'exemple donné ci-dessus, il faudra mettre

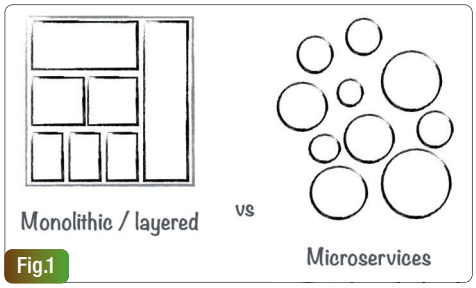


Fig.1

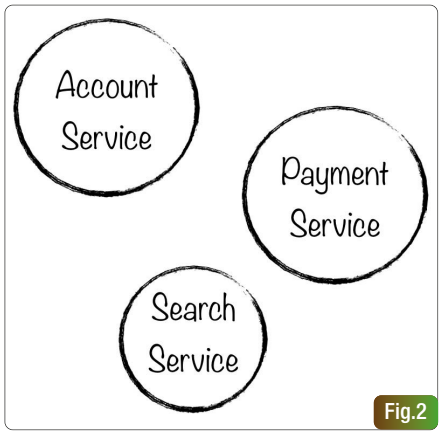


Fig.2

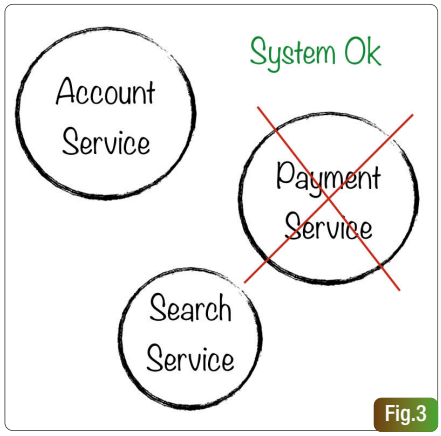


Fig.3

en place des mécanismes de retry et/ou d'intervention manuelle pour gérer a posteriori l'éventuel défaut de paiement.

Est-ce que mon service possède ses propres données? Fig.4. Un microservice est responsable de ses données. Ceci est vrai à condition qu'il soit le seul à pouvoir écrire ou lire les données contenues dans sa base de données.
Si l'on devait accéder aux données de notre service, alors c'est au service d'exposer une API d'accès aux données (afin d'en garder la maîtrise). En aucun cas, on doit pouvoir accéder aux données directement.
Si en répondant à ces questions vous apportez les bonnes réponses alors c'est que le microservice que vous avez développé est une bon microservice :).

## Communication inter microservice

Une des composantes essentielles des microservices est la localisation de ceux-ci. En effet, pour interagir entre eux, les microservices ont besoin de se connaître. En revanche ils ne peuvent pas se connaître directement, cela impliquerait du couplage et casserait le principe d'isolation. Fig.5.
Afin de préserver les propriétés d'isolation que l'on a vues précédemment, il faut mettre en place un service registry (ou service locator) auprès duquel chaque microservice viendra s'enregistrer et sera par conséquent accessible.
Aussi les échanges entre microservices se font de manière asynchrone et non bloquante. Fig.6.

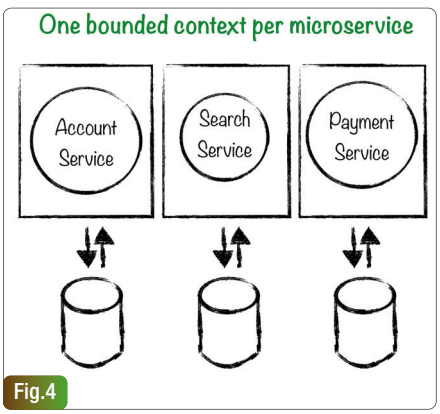


Fig.4

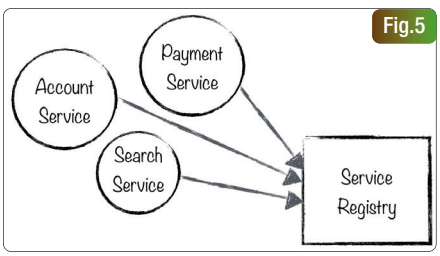


Fig.5

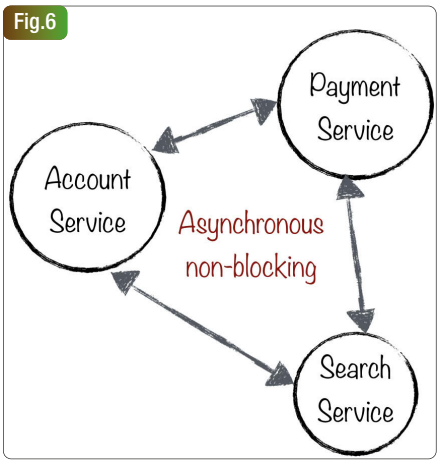


Fig.6

## Anti-patterns

Ce type d'architecture ne permet pas d'adresser toutes les problématiques et surtout n'autorise pas toutes les pratiques. Ci-dessous une liste d'anti-patterns qui ne sont pas compatible avec une architecture microservices ou bien qui représente un non-sens.

- Est-ce la bonne architecture pour mon use-case?
- Une approche transactionnelle entre les services va à l'encontre des principes évoqués ci-dessous (isolation) ;
- Prendre un monolithe que l'on va découper horizontalement couche par couche. Chaque couche applicative (présentation, service, dao) sera isolée dans un microservice. On appelle ce genre d'approche *Distributed monolithique layers*. Ce n'est évidemment pas une bonne pratique ;
- *"Don't Repeat Yourself"* : un microservice a une durée de vie courte. Ce composant doit pouvoir évoluer rapidement. Bien qu'il faille respecter les bonnes pratiques de développement, la répétition n'est pas une mauvaise pratique dans ce cadre-là.
- Partager un dépôt de données est, comme nous l'avons vu plus haut, une mauvaise pratique et va à l'encontre des principes qui définissent un bon microservice.

## Les challenges

Mettre en place une architecture microservices implique certains challenges. Nous allons voir une liste des principaux challenges à relever.

- Ce type d'architecture a un impact organisationnel. En effet, il est important de garder des équipes de taille réduite autour d'un ou deux microservices. Afin de garder une certaine vélocité ;
- La mise en place d'un monitoring du système de microservices ;
- La mise en place du service locator ;
- La capacité de pouvoir travailler dans un environnement d'architecture distribuée.

## CQRS

Avant toutes choses, que signifie les initiales CQRS : Command ; Query ; Responsability ; Segregation ;

## Approche traditionnelle

L'approche traditionnelle que l'on retrouve dans la plupart des projets se décrit selon le schéma suivant : Fig.7.

Nous retrouvons ci-dessous, une application traditionnelle avec la partie front-end et backend. La partie front-end va, via le modèle, faire des écritures et lectures. Nous constatons ici, l'utilisation du même modèle pour réaliser les deux opérations.
D'une manière générale, les besoins en lecture et en écriture ne sont pas les mêmes :

- **Lecture** : Dénormalisation, scalabilité, performance,...
- **Ecriture** : Normalisation, consistance, transactionnelle, ...

Utiliser un modèle unique pour ces deux besoins introduit de la complexité.

## Approche CQRS

L'approche CQRS propose de faire une séparation en deux modèles distincts. Un servirait exclusivement à la lecture et l'autre à l'écriture. Selon les termes employés par CQRS, le modèle dédié à l'écriture s'appelle **Command**, celui dédié à l'écriture est appelé **Query.** Chacun de ces modèles a des propriétés différentes :

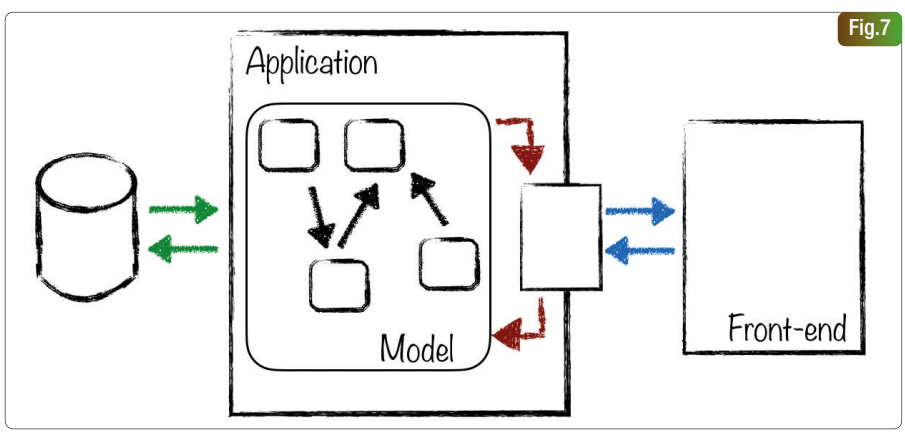


Fig.7

- **Command :** Provoque une changement d'état sur le système, sans retourner de valeur ;
- **Query :** Retourne une valeur, sans effet de bord. C'est-à-dire qu'aucun changement d'état du système ne doit être constaté.

Ci-dessous un schéma représentant l'approche CQRS au sein d'une application : Fig.8.

Cette nouvelle configuration permet d'optimiser chacune des parties selon les besoins.
Ci-dessous, un autre schéma présentant une séparation au niveau de la base de données avec un *dénormaliseur* permettant la translation des informations du côté écriture vers le coté lecture. Fig.9.

## Event Sourcing

En règle générale, l'approche CQRS va de pair avec l'approche Event sourcing.

### Approche traditionnelle

L'approche traditionnelle peut être décrite selon le schéma suivant : Fig.10.
L'idée de ce schéma décrit la manière de mettre en relation un modèle objet (au sens programmation) et un schéma de base de données relationnelle. Ce mapping impose une certaine complexité et n'est pas tout à fait naturel in fine.
Que se passe-t-il concrètement avec ce genre d'approche ? Au fur et à mesure de l'évolution du système, les données vont évoluer. A chaque changement d'état du modèle, les données vont être mises à jour.
On voit ci-dessous les évolutions d'un compte bancaire depuis sa création. A chaque changement d'état, on fait une mise à jour. A n'importe quel moment il est possible de demander l'état du compte. Jusque-là, pas de problème. Fig.11.
En revanche, la difficulté survient au moment où l'on pose la question :
*"mais comment en est-on arrivé à cet état-là ?"*
Cette question peut paraître futile, car ce qui compte c'est l'état courant de l'objet. Au moment d'un bug en production, l'idée d'avoir accès à l'historique des évènements qui ont amené l'objet dans cet état devient intéressante. Nous allons voir comment l'Event sourcing permet d'adresser ce problème.

### L'approche Event sourcing

L'idée principale derrière cette approche se résume en deux points :

- Pas de mise à jour de l'état courant de l'objet ;
- Sauvegarde de l'ensemble des évènements qui ont conduit à l'état courant.

Ce que l'on veut dire ici, c'est que le système va s'attacher à sauvegarder l'ensemble des évènements dans un commit logs. Chaque évènement décrira l'action qui aura un impact sur l'état du système.
A partir de cette collecte d'évènements, nous serons en mesure de pouvoir répondre à la question posée un peu plus haut :
*"mais comment en est-on arrivé à cet état-là ?"*
Le fait de pouvoir répondre à cette question, engendre pas mal d'autres choses positives. En effet. Par exemple, en cas de problème en environnement de production, il sera possible de récupérer l'ensemble des évènements et de les rejouer sur un environnement de tests. Le fait de rejouer les évènements permet de retrouver l'état dans lequel le système se trouvait lorsque le problème est apparu ; il sera plus facile de comprendre comment on en est arrivé là.

### Command et Event

Comment fonctionne ce principe. Sans rentrer dans les détails techniques, voici les règles à suivre pour la génération d'un event ;

- Une commande arrive avec pour objectif de changer l'état du système ;
- Chaque changement d'état donne lieu à un event ;
- L'ordre de création des évènements.

Ci-dessous un exemple de la création de plusieurs évènements lors de la création d'un compte bancaire : Fig.12.
Chaque évènement décrit un changement de l'état du système. Le premier évènement (*BankAccountCreated*) décrit la création du compte, ensuite (*DepositePerfomed*), indique le dépôt de la somme sur le compte, (*Owner Changed)* indique que le compte à changé de propriétaire, et, enfin, l'event (*WithdrwalPerformed)* montre un retrait d'une somme sur le compte.
Une chose que l'on peut noter est la manière de nommer chaque évènement. C'est-à-dire que le nom donné correspond à une action effectuée dans le passé. Sur le schéma ci-dessus la notion de commande n'apparaît pas, mais la convention de nommage dans ce cas-là est qu'une commande doit s'intituler avec un nom à la forme impérative. Maintenant que l'on a tous les évènements, et que l'on souhaite les restaurer (suite à un crash) comment cela se passe-t-il ?
Je vous propose le schéma suivant qui décrit comment les différents évènements vont être réappliqués au modèle : Fig.13. Chaque évènement (dans l'ordre chronologique) va être appliqué sur l'objet du modèle correspondant.

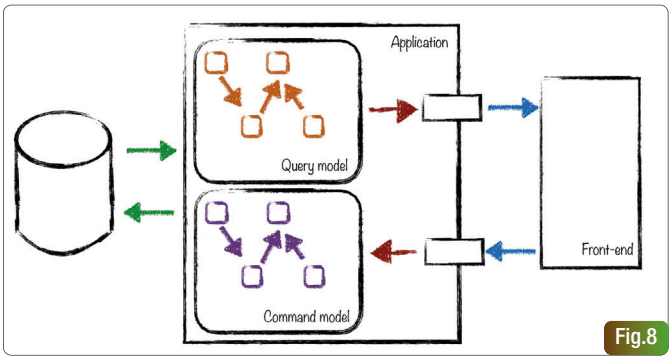


Fig.8

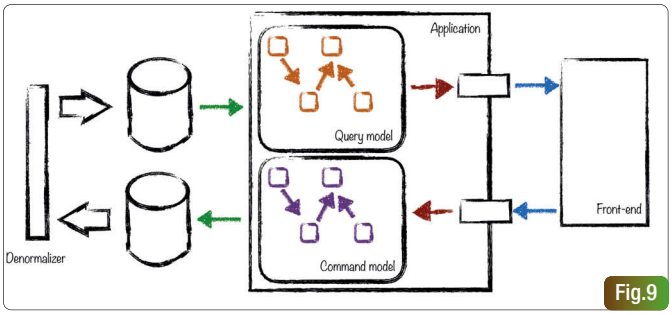


Fig.9

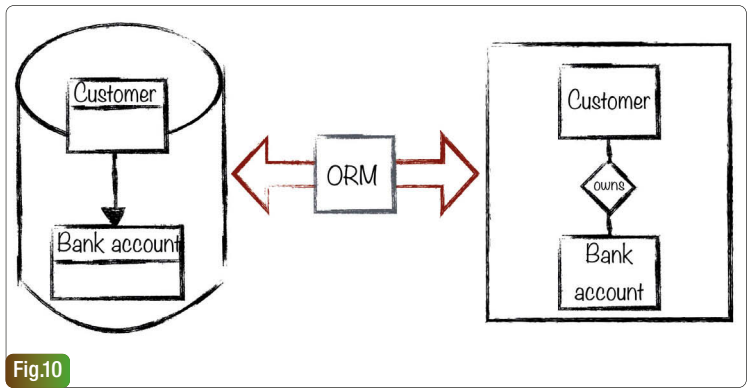


Fig.10

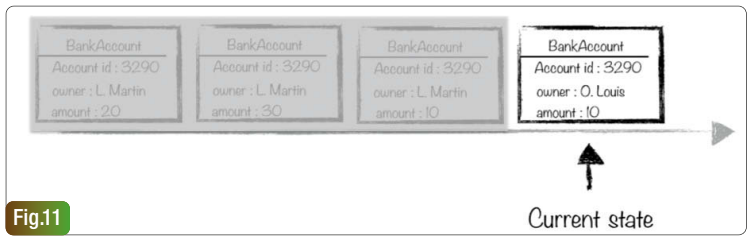


Fig.11

### Snapshot

La question qui peut se poser lorsque la volumétrie d'évènement commence à être importante est : *"comment gère-t-on la restauration avec un important volume d'évènement?"*. Effectivement, restaurer tous les évènements depuis le début peut paraître un peu brutal. C'est pour cette raison que la notion de *snapshot*. Le snapshot correspond à la pose d'un label sur un évènement en particulier pour indiquer au système que la restauration peut commencer à partir de celui-ci. Par ce mécanisme cela réduit le volume.

## LE FRAMEWORK LAGOM

Maintenant que nous avons introduit les concepts sur lesquels le framework Lagom se repose, nous allons entrer dans le vif du sujet.
Nous commencerons par une vue générale du framework, ensuite nous verrons la structure d'un projet type basé sur le Lagom, et, enfin nous verrons un exemple de code.

### Vue générale du framework

Lagom est un framework de microservice édité par Lightbend (ex Typesafe). L'objectif du framework est de proposer la mise en oeuvre d'un *système de microservices.*
Lagom se base sur les principes réactifs (http://www.reactivemanifesto.org/fr) et s'intègre complètement dans le workflow de développement. Nous verrons un peu plus loin comment se fait cette intégration.
Bien que le coeur de Lagom soit développé en Scala, l'API proposée aujourd'hui pour travailler avec le framework est en Java. Selon la communication qui est faite autour du framework, une version Scala devrait sortir très bientôt.
On notera tout de même qu'étant donné l'interopérabilité entre les deux langages, il est déjà possible de coder en Scala avec le framework.
Lagom repose sur 3 fonctionnalités majeures :

- Service API ;
- Persistance API ;
- Environnement de développement.

Chacune de ces fonctionnalités est un atout pour le framework.

### Service API

L'API de service est le point d'entrée lorsque l'on va commencer à écrire un microservice. En effet la première chose que l'on va s'attacher à faire est de décrire l'API que le microservice va exposer au reste du monde.
Pour cela, Lagom propose de faire cette description par le biais d'une interface (au sens Java). Cette interface étendra l'interface ***Service*** du framework et proposera d'implémenter la méthode descriptor().

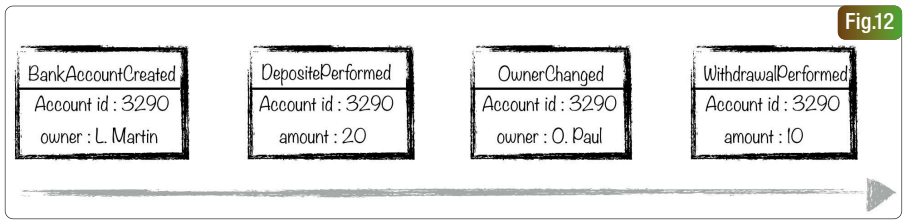


Fig.12

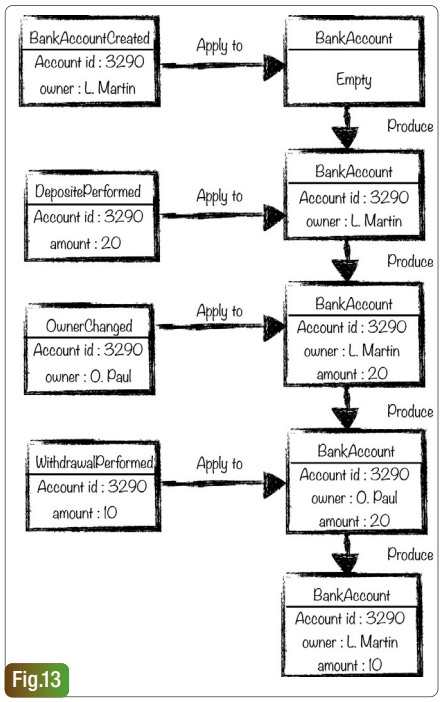


Fig.13

Cette méthode retournera une instance de l'object Descriptor qui contiendra l'ensemble des informations nécessaires au framework pour interagir avec le microservice. Lagom met à disposition un DSL pour décrire cette interface.
Un des avantages de passer par ce mécanisme est que la description du service est typée, ce qui limite les erreurs dans la configuration.
Nous verrons dans l'exemple à la fin de cet article comment tout cela fonctionne.

### Persistance API

Sur la partie persistance, Lagom est par défaut intimement lié avec les concepts que nous avons vus en introduction (CQRS, Event sourcing). Au sein du framework se trouve le composant *PersistentEntity*. Pour l'implémentation de base on retrouve les points d'entrées permettant d'enregistrer des handlers sur les commandes et les events. La notion d'état est aussi représentée. Lorsque l'on développe un microservice, il faut définir les commandes, events et états selon un design particulier (que l'on verra lors de l'exemple d'implémentation en fin d'article).
Aujourd'hui, l'approche par défaut est fortement orientée, mais dans un futur proche on pourra travailler avec une approche plus traditionnelle (même si bien sûr ce n'est pas recommandé par la philosophie de Lagom.

### Environnement de développement

L'idée principale qui émane vis-à-vis du développeur est de simplifier au maximum les étapes de développement.
Pendant la phase de développement, un certain de nombre de *facilitateurs* ont été mis en place.

- Le démarrage de l'ensemble du système à partir d'une seule commande ;
- Rechargement du code à chaud ;
- Ensemble des services disponibles par défaut et prêt à l'emploient (service locator, cassandra embedded, gateway) ;
- Une intégration complète dans les principaux IDEs.

Chacun de ces points améliore clairement la productivité du développeur.

### Protocoles de communication

Pour communiquer avec le monde extérieur, Lagom se base sur des standards.
Les protocoles utilisés sont :

- http ;
- Websocket ;
- JSON.

Lagom peut échanger avec n'importe quel système sans contrainte.

### Composants technologiques

Plutôt que de réinventer la roue, Lagom se base sur un ensemble de librairies éprouvées et reconnues, voici la liste exhaustive de la stack utilisée dans Lagom :

- **Java 8 (& Scala) ;**
- **Immutables** : gestion des objets immutables en Java ;
- **SBT** : système de build ;
- **Jackson** : sérialisation Json ;
- **Cassandra** : BDD embarquée par défaut ;
- **Play framework** : gestion des routes et UI ;
- **Guice** : injection de dépendances ;
- **Akka** : persistence, Pub/Sub, cluster ;
- **Akka Stream** : streaming part ;
- **SLF4J & Logback**.

## En conclusion

Cet article vient de sortir des cartons de Lightbend. Le design et les concepts utilisés en font un framework prometteur. Je trouve que l'on peut faire une analogie avec PlayFramework : efficacité avant tout! J'espère que cet article vous aura donné envie d'aller plus loin.

# Scapy

## 2e partie

*Dans le précédent article sur Scapy, nous avons découvert les bases, c'est à dire la façon de forger des paquets, de les envoyer et de traiter le retour de ces paquets. Nous allons ici un peu approfondir nos connaissances en regardant quelques modules existants pour ensuite écrire nos propres scripts sur des cas réels déjà rencontré lors de nos audits réseaux ou de la maintenance des réseaux informatiques.*

Franck Ebel
*Expert R&D et Formations*
*Serval-concept / serval-formation*
*Responsable Licence CDAISI , UVHC*
*Commandant de Gendarmerie réserviste,*
*cellule Cyberdéfense*

### Scapy en détail

L'opérateur / permet « d'assembler » deux couches entre elles, par exemple IP()/TCP(). La couche la plus basse peut avoir un ou plus de ses champs par défaut chargés dans la couche la plus haute, IP => TCP.

```
>>> IP()
<IP  |>
>>> IP()/TCP()
<IP  frag=0 proto=tcp |<TCP  |>>
>>> Ether()/IP()/TCP()
<Ether  type=0x800 |<IP  frag=0 proto=tcp |<TCP  |>>>
>>> IP()/TCP()/"GET / HTTP/1.0\r\n\n"
<IP  frag=0 proto=tcp |<TCP  |<Raw  load='GET / HTTP/1.0\r\n\n'
|>>>
>>> Ether()/IP()/IP()/UDP()
<Ether  type=0x800 |<IP  frag=0 proto=ipencap |<IP  frag=0
proto=udp |<UDP  |>>>>
>>> IP(proto=55)/TCP()
<IP  frag=0 proto=55 |<TCP  |>>
>>>
```

Nous allons maintenant travailler sur la façon d'afficher les résultats.

```
>>> str(IP())
'E\x00\x00\x14\x00\x01\x00\x00@\x00|\xe7\x7f\x00\x00\x01\x7f\x00\x
00\x01'
>>> IP(_)
<IP  version=4L ihl=5L tos=0x0 len=20 id=1 flags= frag=0L ttl=64
proto=ip chksum=0x7ce7 src=127.0.0.1 dst=127.0.0.1 |>
```

Nous pouvons afficher IP() au format chaîne de caractères.
Grâce à IP(_) nous pouvons visualiser les différents champs de la trame IP.

```
>>> a=Ether()/IP(dst="www.programmez.com")/TCP()/"GET /index.html
HTTP/1.0\n\n"
>>> hexdump(a)
0000  00 0B CD B1 24 33 00 26  B9 EB 6F 68 08 00 45 00  ....  $3.&..oh..E.
0010  00 42 00 01 00 00 40 06  50 67 C3 DD BD 9B 5A 53
.B....@.Pg....ZS
0020  4E 82 00 14 00 50 00 00  00 00 00 00 00 00 50 02
N....P........P.
0030  20 00 B3 64 00 00 47 45  54 20 2F 69 6E 64 65 78
..d..GET /index
0040  2E 68 74 6D 6C 20 48 54  54 50 2F 31 2E 30 0A 0A  .html
HTTP/1.0..
>>>
```

Nous pouvons aussi les visualiser en hexadécimal grâce à hexdump().

Nous avons, dans ce format d'affichage, les trames en hexadécimal à gauche et, à droite, la traduction en chaîne de caractères quand cela est possible. Nous retrouvons par exemple le GET /index HTTP/1.0.
Nous pouvons bien sûr visionner cela tout comme une chaîne de caractères (str()), ou avoir le détail de la trame (Ether()), et si cela est trop lourd à regarder, nous pouvons cacher les valeurs par défaut (hide_defaults()).

```
>>> b=str(a)
>>> b
'\x00\x0b\xcd\xb1$3\x00&\xb9\xeboh\x08\x00E\x00\x00B\x00\x01\x00\x
00@\x06Pg\xc3\xdd\xbd\x9bZSN\x82\x00\x14\x00P\x00\x00\x00\x00\x00\
x00\x00\x00P\x02 \x00\xb3d\x00\x00GET /index.html HTTP/1.0\n\n'
>>> c=Ether(b)
>>> c
<Ether  dst=00:0b:cd:b1:24:33 src=00:26:b9:eb:6f:68 type=0x800 |
<IP  version=4L ihl=5L tos=0x0 len=66 id=1 flags= frag=0L ttl=64
proto=tcp chksum=0x5067 src=195.221.189.155 dst=90.83.78.130
options=[] |<TCP  sport=ftp_data dport=www seq=0 ack=0 dataofs=5L
reserved=0L flags=S window=8192 chksum=0xb364 urgptr=0 options=[]
|<Raw  load='GET /index.html HTTP/1.0\n\n' |>>>>
>>> c.hide_defaults()
>>> c
<Ether  dst=00:0b:cd:b1:24:33 src=00:26:b9:eb:6f:68 type=0x800 |
<IP  ihl=5L len=66 frag=0 proto=tcp chksum=0x5067
src=195.221.189.155 dst=90.83.78.130 |<TCP  dataofs=5L
chksum=0xb364 options=[] |<Raw  load='GET /index.html
HTTP/1.0\n\n' |>>>>
>>>
```

Pour le moment, nous avons juste généré des paquets. Nous pouvons, si nous le désirons, personnaliser chaque champ du paquet. Nous pouvons, par exemple, définir l'IP de destination, changer le TTL, le port...

### Changement de la destination

```
>>> a=IP(dst="www.google.fr")
>>> a
<IP  dst=Net('www.google.fr') |>
>>> [p for p in a]
[<IP  dst=173.194.34.56 |>]
>>> a=IP(dst="www.google.com")
>>> a
<IP  dst=Net('www.google.com') |>
>>> [p for p in a]
[<IP  dst=173.194.34.52 |>]
>>> a=IP(dst="www.programmez.com")
>>> a
<IP  dst=Net('www.programmez.com') |>
>>> [p for p in a]
[<IP  dst=188.165.230.204 |>]
>>> a=IP(dst=Net("www.programmez.com"))
>>> a
<IP  dst=Net('www.programmez.com') |>
```

```
>>> a=IP(dst=Net("www.programmez.com/24"))
>>> a
<IP  dst=Net('www.programmez.com/24') |>
>>> [p for p in a]
[<IP  dst=188.165.230.0 |>, <IP  dst=188.165.230.1 |>, <IP  dst=188.165.230.2 |>,
 <IP dst=188.165.230.3 |>, <IP dst=188.165.230.4 |>, <IP dst=188.165.230.5 |>, <IP
dst=188.165.230.6 |>, <IP  dst=188.165.230.7 |>, <IP  dst=188.165.230.8 |>, <IP
dst=188.165.230.9 |>, <IP  dst=188.165.230.10 |>, <IP  dst=188.165.230.11 |>, <IP
dst=188.165.230.12 |>, <IP  dst=188.165.230.13 |>, …..
>>>
```

## Changement du TTL (Time To Live)

```
>>> b=IP(ttl=[1,2,(5,9)])
>>> [p for p in b]
[<IP  ttl=1 |>, <IP  ttl=2 |>, <IP  ttl=5 |>, <IP  ttl=6 |>, <IP
ttl=7 |>, <IP  ttl=8 |>, <IP  ttl=9 |>]
```

## Assemblage

```
>>> c=TCP(dport=[80,443])
>>> [p for p in a/c]
[<IP  frag=0 proto=tcp dst=90.83.78.128 |<TCP  dport=www |>>, <IP
frag=0 proto=tcp dst=90.83.78.128 |<TCP  dport=https |>>, <IP
frag=0 proto=tcp dst=90.83.78.129 |<TCP  dport=www |>>, <IP
frag=0 proto=tcp dst=90.83.78.129 |<TCP  dport=https |>>, <IP
frag=0 proto=tcp dst=90.83.78.130 |<TCP  dport=www |>>, <IP
frag=0 proto=tcp dst=90.83.78.130 |<TCP  dport=https |>>, <IP
frag=0 proto=tcp dst=90.83.78.131 |<TCP  dport=www |>>, <IP
frag=0 proto=tcp dst=90.83.78.131 |<TCP  dport=https |>>]
```

Dans l'exemple ci-dessus, nous couplons la configuration pour a et c ; nous aurions pu écrire :

```
>>> [p for p in IP(dst="www.programmez.com/30")/TCP(dport=[80,443])]
[<IP  frag=0 proto=tcp dst=90.83.78.128 |<TCP  dport=www |>>, <IP
frag=0 proto=tcp dst=90.83.78.128 |<TCP  dport=https |>>, <IP
frag=0 proto=tcp dst=90.83.78.129 |<TCP  dport=www |>>, <IP
frag=0 proto=tcp dst=90.83.78.129 |<TCP  dport=https |>>, <IP
frag=0 proto=tcp dst=90.83.78.130 |<TCP  dport=www |>>, <IP
frag=0 proto=tcp dst=90.83.78.130 |<TCP  dport=https |>>, <IP
frag=0 proto=tcp dst=90.83.78.131 |<TCP  dport=www |>>, <IP
frag=0 proto=tcp dst=90.83.78.131 |<TCP  dport=https |>>]
>>>
```

## Contenu du paquet

```
>>> a.show()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= ip
  chksum= None
  src= 195.221.189.155
  dst= Net('www.programmez.com/30')
  \options\
```

Maintenant que nous savons manipuler les paquets, voyons en détail comment les envoyer. La fonction send() permet d'envoyer les paquets sur la couche 3. La fonction sendp(), quant à elle, utilisera la couche 2. À vous donc de déterminer la fonction à utiliser suivant vos besoins.

```
>>> send(IP(dst="195.221.189.248")/ICMP())
.
Sent 1 packets.
>>> sendp(Ether()/IP(dst="195.221.189.248",ttl=(1,4)),iface="eth0")
....
Sent 4 packets.
```

Nous voyons dans l'exemple ci-dessus que si nous utilisons send(), nous devons définir que nous travaillons par exemple avec le protocole ICMP. En utilisant sendp(), nous pouvons si besoin fournir des options à IP() telles que l'interface utilisée, le TTL. Avec ttl=(1,4) nous envoyons ici 4 paquets.

```
>>> sendp("Programmez est sur le net",iface="eth0",loop=1, inter=0.2)
.....^C
Sent 15 packets
```

Nous pouvons utiliser la fonction sr() qui permet d'envoyer et de recevoir des paquets. La fonction sr1() en est une variante qui ne retourne qu'un paquet en retour au paquet envoyé. Le paquet doit être un paquet de la couche 3 (IP, ARP...). La fonction srp() fait de même, mais pour la couche 2.

## Test sur programmez.com

```
>>> p=sr1(IP(dst="www.programmez.com")/ICMP()/"PROGRAMMEZ")
Begin emission:
..Finished to send 1 packets.
.....*
Received 111 packets, got 0 answers, remaining 1 packets
>>>
```

Nous n'avons ici aucun paquet en retour (got 0 answers). Tentons sur une autre machine :

## Test en interne

```
>>> p=sr1(IP(dst="koala.univ-valenciennes.fr")/ICMP()/
"PROGRAMMEZ")
Begin emission:
...Finished to send 1 packets.
*
Received 4 packets, got 1 answers, remaining 0 packets
>>> p
<IP  version=4L ihl=5L tos=0x0 len=40 id=40214 flags=DF frag=0L
ttl=255 proto=icmp chksum=0xdb6e src=195.221.189.248
dst=195.221.189.155 options=[] |<ICMP  type=echo-reply code=0
chksum=0x4646 id=0x0 seq=0x0 |<Raw  load='PROGRAMMEZ' |<Padding
load='\x00\x00\x00\x00\x00\x00' |>>>>
>>> p.show()
###[ IP ]###
  version= 4L
  ihl= 5L
  tos= 0x0
  len= 40
  id= 40214
  flags= DF
  frag= 0L
  ttl= 255
  proto= icmp
  chksum= 0xdb6e
  src= 195.221.189.248
```

```
  dst= 195.221.189.155
  \options\   ###[ ICMP ]###
     type= echo-reply
     code= 0
     chksum= 0x4646
     id= 0x0
     seq= 0x0
###[ Raw ]###
        load= 'PROGRAMMEZ'
###[ Padding ]###
           load= '\x00\x00\x00\x00\x00\x00'
>>>
```

Nous voyons ici un paquet en retour, donc une réponse, nous pouvons donc voir le contenu de la réponse avec p.show() par exemple.
Nous pouvons observer qu'il y a eu une résolution DNS. Nous pouvons interroger ce DNS grâce à DNS().

```
>>> p=sr1(IP(dst="koala.univ-valenciennes.fr")/UDP()/
DNS(rd=1,qd=DNSQR(qname="www.univ-valenciennes.fr")))
Begin emission:
.Finished to send 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
>>> p
<IP  version=4L ihl=5L tos=0x0 len=268 id=40367 flags=DF frag=0L
ttl=255 proto=udp chksum=0xd9e1 src=195.221.189.248
dst=195.221.189.155 options=[] |<UDP  sport=domain dport=domain
len=248 chksum=0xf7ea |<DNS  id=0 qr=1L opcode=QUERY aa=1L tc=0L
rd=1L ra=1L z=0L rcode=ok qdcount=1 ancount=2 nscount=4 arcount=4
qd=<DNSQR  qname='www.univ-valenciennes.fr.' qtype=A qclass=IN |>
an=<DNSRR  rrname='www.univ-valenciennes.fr.' type=CNAME rclass=IN
ttl=86400 rdata='blade2.univ-valenciennes.fr.' |<DNSRR
rrname='blade2.univ-valenciennes.fr.' type=A rclass=IN ttl=86400
rdata='193.50.192.166' |>> ns=<DNSRR  rrname='univ-
valenciennes.fr.' type=NS rclass=IN ttl=86400 rdata='titan.univ-
valenciennes.fr.' |<DNSRR  rrname='univ-valenciennes.fr.' type=NS
rclass=IN ttl=86400 rdata='pulsar.univ-valenciennes.fr.' |<DNSRR
rrname='univ-valenciennes.fr.' type=NS rclass=IN ttl=86400
rdata='reserv1.univ-lille1.fr.' |<DNSRR  rrname='univ-
valenciennes.fr.' type=NS rclass=IN ttl=86400 rdata='reserv2.univ-
lille1.fr.' |>>>> ar=<DNSRR  rrname='titan.univ-valenciennes.fr.'
type=A rclass=IN ttl=86400 rdata='193.50.192.38' |<DNSRR
rrname='pulsar.univ-valenciennes.fr.' type=A rclass=IN ttl=86400
rdata='193.50.192.1' |<DNSRR  rrname='reserv1.univ-lille1.fr.'
type=A rclass=IN ttl=141972 rdata='193.49.225.15' |<DNSRR
rrname='reserv2.univ-lille1.fr.' type=A rclass=IN ttl=141972
rdata='193.49.225.90' |>>>> |>>>
>>>
```

Nous avons donc indiqué ci-dessus l'adresse du DNS et nous avons demandé les informations sur www.univ-valenciennes.fr. Nous pouvons donc visualiser ici toutes les informations reçues lors de la requête DNS. Nous pouvons récupérer indépendamment chaque élément de la liste ainsi obtenue :

```
>>> p[2][8][3]
<DNSRR  rrname='reserv2.univ-lille1.fr.' type=A rclass=IN
ttl=141972 rdata='193.49.225.90' |>
```

La fonction sr (send and receive) retourne deux listes. La première est une liste des couples paquets envoyés et reçus, et la deuxième une liste des paquets sans réponse.

```
>>> sr(IP(dst="195.221.189.248")/TCP(dport=[21,22,23]))
Begin emission:
..**.Finished to send 3 packets.
*
Received 6 packets, got 3 answers, remaining 0 packets
(<Results: TCP:3 UDP:0 ICMP:0 Other:0>, <Unanswered: TCP:0 UDP:0
ICMP:0 Other:0>)
>>> ans,unans=_
>>> ans.summary()
IP / TCP 195.221.189.155:ftp_data > 195.221.189.248:ftp S ==> IP /
TCP 195.221.189.248:ftp > 195.221.189.155:ftp_data SA / Padding
IP / TCP 195.221.189.155:ftp_data > 195.221.189.248:ssh S ==> IP /
TCP 195.221.189.248:ssh > 195.221.189.155:ftp_data SA / Padding
IP / TCP 195.221.189.155:ftp_data > 195.221.189.248:telnet S ==>
IP / TCP 195.221.189.248:telnet > 195.221.189.155:ftp_data SA /
Padding
>>>
```

Nous pouvons définir un temps d'attente entre deux paquets grâce au paramètre inter. Nous pouvons envoyer de nouveau les paquets sans réponses grâce au paramètre retry.
Nous pouvons envoyer et recevoir dans une boucle :

```
>>> srloop(IP(dst="www.univ-valenciennes.fr/30")/TCP())
fail 4: IP / TCP 195.221.189.155:ftp_data > 193.50.192.166:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.165:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.164:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.167:www S
RECV 1: IP / ICMP 193.50.192.66 > 195.221.189.155 dest-unreach
host-unreachable / IPerror / TCPerror
fail 3: IP / TCP 195.221.189.155:ftp_data > 193.50.192.166:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.165:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.167:www S
fail 4: IP / TCP 195.221.189.155:ftp_data > 193.50.192.166:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.165:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.164:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.167:www S
RECV 1: IP / ICMP 193.50.192.66 > 195.221.189.155 dest-unreach
host-unreachable / IPerror / TCPerror
fail 3: IP / TCP 195.221.189.155:ftp_data > 193.50.192.166:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.165:www S
        IP / TCP 195.221.189.155:ftp_data > 193.50.192.167:www S
send...
Sent 16 packets, received 2 packets. 12.5% hits.
(<Results: TCP:0 UDP:0 ICMP:2 Other:0>, <PacketList: TCP:14 UDP:0
ICMP:0 Other:0>)
>>>
```

Nous avons bien pris en main Scapy, nous pouvons maintenant créer nos propres scripts et nous attaquer à des applications utiles et réelles.

## Sécurité réseau

Traceroute est un outil réseau, disponible sous Linux et sous Windows, qui permet de suivre le chemin qu'un paquet de données (paquet IP) va prendre pour aller d'une machine A à une machine B.
Par défaut, le paquet est envoyé sur Internet mais le chemin emprunté par le paquet peut varier, en cas de panne d'un lien ou bien en cas de changement des connexions de l'un des opérateurs.
Après avoir été expédié au fournisseur d'accès, le paquet est transmis à des routeurs intermédiaires qui vont l'acheminer jusqu'à sa destination. Le paquet peut subir des transformations lors de son voyage. Il se peut aussi

qu'il n'arrive jamais à destination si le nombre de nœuds intermédiaires est trop important. Nous allons étudier les possibilités d'effectuer un Traceroute à l'aide de Scapy.

```
>>> ans,unans=sr(IP(dst="www.google.fr", ttl=(4,25),
id=RandShort())/TCP(flags=0x2))
Begin emission:
.****..Finished to send 22 packets.
.....
.....
.....
^C
Received 1115 packets, got 4 answers, remaining 18 packets
>>> for snd,rcv in ans:
...     print snd.ttl, rcv.src, isinstance(rcv.payload, TCP)
...
4 193.51.189.118 False
5 193.51.189.174 False
6 193.51.182.197 False
7 72.14.238.234 False
>>>
```

Mais Scapy a déjà sa fonction Traceroute intégrée. Contrairement aux autres programmes Traceroute, Scapy envoie tous ses paquets en même temps. L'avantage principal est que nous pouvons lui indiquer de multiples cibles à traiter en même temps.

```
>>> traceroute(["www.google.fr","www.programmez.com",
"www.univ-valenciennes.fr"])
Begin emission:
**********************************************************************
**.**.*.**Finished to send 90 packets.
**.....
Received 123 packets, got 75 answers, remaining 15 packets
   173.194.34.56:tcp80 193.50.192.166:tcp80 90.83.78.130:tcp80
1  195.221.189.115 11  195.221.189.115 11  195.221.189.115 11
2  195.221.189.254 11  195.221.189.254 11  195.221.189.254 11
3  193.51.250.129  11  192.168.206.2   11  193.51.250.129  11
4  -                   193.50.192.66   11  -
5  193.51.189.118  11  193.50.192.166  SA  193.51.189.118  11
6  193.51.189.174  11  193.50.192.166  SA  195.10.54.65     11
7  -                   193.50.192.166  SA  195.2.9.58       11
8  193.51.182.197  11  193.50.192.166  SA  -
9  72.14.238.234   11  193.50.192.166  SA  193.251.128.117 11
10 -                   193.50.192.166  SA  -
11 173.194.34.56   SA  193.50.192.166  SA  -
12 173.194.34.56   SA  193.50.192.166  SA  -
13 173.194.34.56   SA  193.50.192.166  SA  -
14 173.194.34.56   SA  193.50.192.166  SA  -
15 173.194.34.56   SA  193.50.192.166  SA  -
16 173.194.34.56   SA  193.50.192.166  SA  -
17 173.194.34.56   SA  193.50.192.166  SA  -
18 173.194.34.56   SA  193.50.192.166  SA  -
19 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
20 -                   193.50.192.166  SA  90.83.78.130    SA
21 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
22 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
23 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
24 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
25 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
26 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
27 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
28 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
29 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
30 173.194.34.56   SA  193.50.192.166  SA  90.83.78.130    SA
(<Traceroute: TCP:57 UDP:0 ICMP:18 Other:0>, <Unanswered: TCP:15
UDP:0 ICMP:0 Other:0>)
>>>
```

Nous pouvons ensuite récupérer le résultat pour le gérer .
Code complet sur www.programmez.com
Comme avec n'importe quel autre objet, nous pouvons, si nous le souhaitons, ajouter ensuite les résultats.
Code complet sur le site du magazine.
Nous allons maintenant nous attarder sur d'autres façons d'effectuer un Traceroute.

## TCP SYN traceroute

Code complet sur le site du magazine.
Nous effectuons un SYN car le flags S est activé dans notre commande.

## UDP traceroute

Code complet sur le site du magazine.

## DNS traceroute

Code complet sur le site du magazine.

## Conclusion

Nous venons de découvrir la simplicité de tester le réseau ou de mener des attaques simplement avec Scapy. Nous verrons dans un dernier article des scripts plus complexes nous permettant par exemple d'effectuer un Man in the middle en quelques lignes et en parallèle de sniffer les paquets circulant sur le réseau et de falsifier des paquets reçus avant de les transmettre.
A bientôt donc pour approfondir cet outil indispensable.

*Suite le mois prochain*

**Abonnement** : Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex. - Tél. : 01 55 56 70 55 - *abonnements.programmez @groupe-gli.com* - Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30. **Tarifs** abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 € Autres pays : nous consulter.
**PDF :** 30 € (Monde Entier) souscription sur www.programmez.com

Une publication **Nefer-IT**
7 avenue Roger Chambonnet
91220 Brétigny sur Orge
redaction@programmez.com
Tél. : 01 60 85 39 96
**Directeur de la publication :** François Tonic
**Secrétaire de rédaction :** Olivier Pavie

**Ont collaboré à ce numéro :** Sylvain Saurel

**Les experts du numéro :** F. Fadel, G. Serfaty, R. Pion, H. Meziani, J-M Billaud, K. Trelohan, C. Peugnet, P. Batty, Y. Chabeb, L. Le Goff, F. Santin, T. Ranise, T. Ranise, D. Djordjevic, M. Caroul, F. Botte, M. Roussel, J. De Oliveira, M. Bouilloux, V. Piard, A. Fourmi, A. Barralon, C. Pichaud, G. Nicolas, F. Sznajderman, F. Ebel

**Couverture :** © Nongkran_ch,
**Articles :** © HASLOO, © marekuliasz

**Maquette :** Pierre Sandré

**Publicité :** PC Presse,
Tél.: 01 74 70 16 30, Fax : 01 41 38 29 75
pub@programmez.com

**Imprimeur :** S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.

**Marketing et promotion des ventes :**
Agence BOCONSEIL - Analyse Media Etude

**Directeur :** Otto BORSCHA oborscha@boconseilame.fr
**Responsable titre :** Terry MATTARD
Téléphone : 09 67 32 09 34

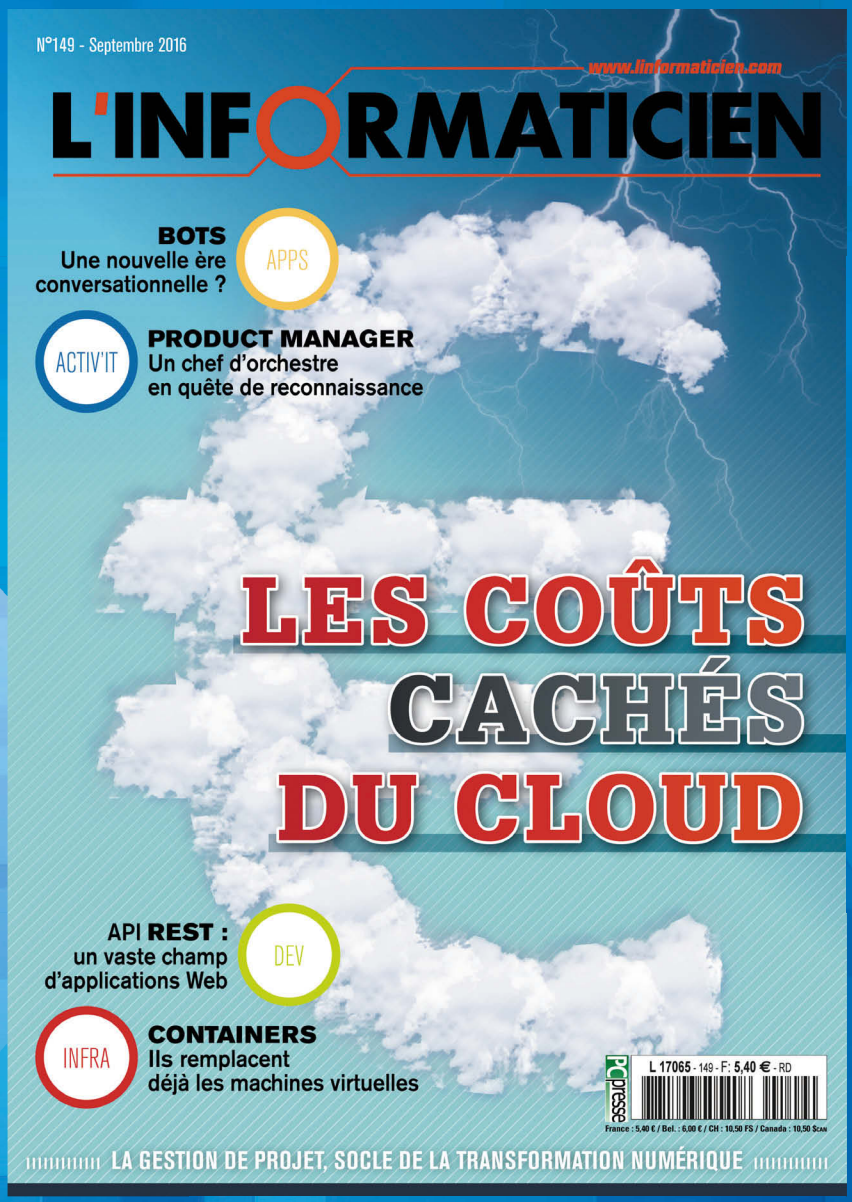
**Contacts**
**Rédacteur en chef :** ftonic@programmez.com
**Rédaction :** redaction@programmez.com
**Webmaster :** webmaster@programmez.com
**Publicité :** pub@programmez.com
**Evenements / agenda :** redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908

# LE MAGAZINE DES PROS DE L'IT

# Numéro spécial rentrée

**Sur abonnement ou en kiosque**




**Mais aussi sur le web**

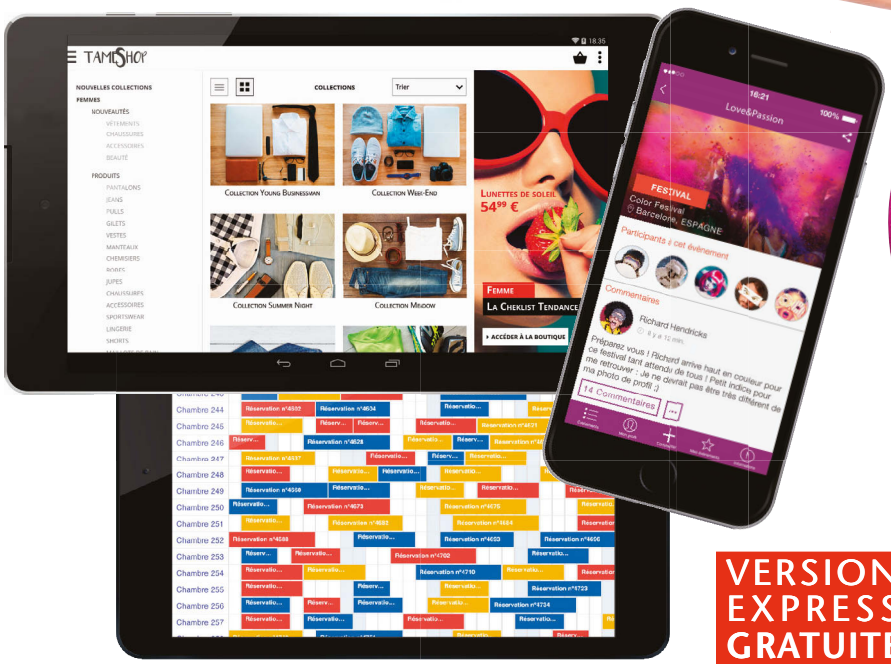


**Ou encore sur votre tablette**

www.linformaticien.com
L'INFORMATICIEN

# WINDEV
## NOUVELLE VERSION 21 MOBILE

# DÉVELOPPEZ MOBILE NATIF
# UN SEUL CODE, TOUTES LES CIBLES



VERSION EXPRESS GRATUITE
Téléchargez-la !
Version non limitée dans le temps

### DÉVELOPPEZ POUR TOUS LES MOBILES

**WINDEV MOBILE 21** vous permet de développer des applis mobiles **natives** pour tous les systèmes.
Le **code** et les **interfaces** sont **identiques**.
**Il suffit de recompiler votre source pour obtenir des applis natives** pour **Android**, **iOS**, **Windows 10 Mobile**, pour smartphones et pour tablettes...
Base de Données embarquée, Client/Serveur et Cloud inclus.

*Vous disposez d'applications WINDEV ? Elles sont déjà compatibles. Recompilez-les simplement sur mobile !*

**DÉVELOPPEZ 10 FOIS PLUS VITE**

**www.pcsoft.fr**

*+ de 100 de témoignages sur le site*
Dossier complet gratuit sur simple demande