PROGRAMMEZ!

#205 - mars 2017

le magazine des développeurs

Souriez !

Vous êtes analysés !

Tout savoir sur les technologies cognitives.
Inclus : des exemples à réutiliser immédiatement

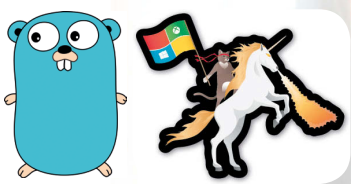
OpenData

La RATP ouvre (enfin) les données des horaires des transports !

Docker

Pourquoi il est urgent de l'utiliser ?

Des histoires de peluches et de mascottes



Vanilla.JS :

pour un JavaScript pur et ultra rapide !

Office 365 c'est aussi pour les développeurs

IoT Android Things : Google réinvestit les objets connectés
Contrôler son robot en Bluetooth

M 04319 - 205 - 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

PRÉSENTE

CLOUD**IKOULA**ONE

Le succès est votre prochaine destination

MIAMI SINGAPOUR PARIS
AMSTERDAM FRANCFORT _ _ _

*CLOUD**IKOULA**ONE* est une solution de Cloud public, privé et hybride qui vous permet de déployer en 1 clic et en moins de 30 secondes des machines virtuelles à travers le monde sur des infrastructures SSD haute performance.

www.ikoula.com sales@ikoula.com 01 84 01 02 50

ikoula HÉBERGEUR CLOUD

CLOUD | INFOGÉRANCE | SERVEUR DÉDIÉ | VPS | MESSAGERIE

“… de l’audace, encore de l’audace, toujours de l’audace…”

Ce morceau de phrase a été prononcé par Danton le 2 septembre 1792. Nous ne reviendrons pas sur le pourquoi du comment de ce discours à l’Assemblée. Danton fut l’un des plus grands orateurs de la Révolution et, au final, grand ennemi de Robespierre.
Ce dernier causa sa chute.

Pourquoi (reprendre) cette déclaration révolutionnaire ? On peut parfaitement appliquer celle-ci à l’industrie et aux technologies. Sans audace, où en serions-nous, même si parfois elle mène à la guerre et aux plus sombres heures de notre Histoire ? L’audace a permis de marcher sur la lune. L’audace a permis l’apparition de nouvelles industries, de nouveaux transports. Si Collomb n’avait eu un peu d’audace, aurait-il découvert le Nouveau Monde ? Ben ok, il a mal compris le géographe Ptolémée, mais cette erreur n’en a pas été une finalement. Le génie, la bonne idée ne suffisent pas. L’audace est indispensable. Steve Jobs, que l’on aime ou pas l’homme, l’avait bien compris.

Combien de bonnes idées et de prometteuses technologies n’ont jamais percé faute d’audace, avec un soupçon d’opportunité et de chance ?

Prenons Elon Musk. Il est à la fois admiré, honni et détesté. Mégalomane, invivable, ambitieux ? Oui sans aucun doute. Tesla serait un flop industriel et serait incapable d’être viable pour concurrencer les constructeurs automobiles ? Son usine géante, inutile et démesurée ? SpaceX et Hyperloop ? Des projets fous ? Alors, pourquoi investir des milliards et lancer des projets aussi ambitieux ? Justement, Musk lance ces projets, car ils sont ambitieux, bousculent l’industrie et font rêver le monde. Et le rêve est une composante essentielle.
Quand Kennedy défie l’Amérique à aller sur la Lune dans les dix ans le 12 septembre 1962, il donne une ligne directrice, mais surtout, il fait rêver. Musk veut envoyer l’Homme sur Mars. Alors que beaucoup vont critiquer et dire que c’est impossible, ou presque, d’autres vont rêver et applaudir. J’ai applaudi.

Oui, aller sur Mars, avec l’impossibilité de revenir, du moins pour les premières missions est un défi aussi bien humain que technologique. Certaines critiques ne veulent pas que des entreprises privées s’approprient l’espace. Oui il y a un danger, mais pourquoi les agences spatiales n’ont-elles pas créé des missions humaines pour retourner sur la Lune et s’y installer avant d'aller sur Mars ? La station ISS a coûté une centaine de milliards en 19 ans. Et les ambitions au-delà demeurent floues. On reparle ici et là de la Lune, mais nous attendons du concret.

Dernier exemple, le Concorde a été une audace technologique et industrielle. Depuis la fin de son exploitation en novembre 2003, le transport a régressé et on attend toujours son remplaçant qui pourrait arriver vers 2020 – 2025. La même question se pose pour notre TGV dont la première ligne date de 1981. Aujourd’hui, le modèle TGV atteint ses limites avec un coût d’exploitation important, notamment pour la construction des lignes. L’avenir du transport à haute vitesse en France et en Europe n’est plus le TGV.

De l’audace, encore de l’audace, toujours de l’audace.

ftonic@programmez.com

Dans le prochain numéro !
Programmez! #206, dès le 1er avril 2017

Visual Studio 2017
La nouvelle version de l’IDE de Microsoft est disponible. Test & présentation complète.

Dossier : les tests
Les tests sont plus que jamais indispensables pour stabiliser une application et son code. Mais il ne faut pas les faire n’importe comment.

Guide des cartes Maker
Notre guide complet des cartes de prototypages, IoT et nano-PC. Comment choisir ? A quel prix ? Programmez! vous dit tout.

WebKit dévoile une nouvelle API 3D pour les navigateurs qui ne remplace pas forcément WebGL.

Java 9 est toujours prévu pour le 27 juillet 2017…

Oculus, filiale de Facebook, perd son procès contre Zenimax. Zenimax avait porté plainte pour vols technologiques. Montant du dédommagement : 500 millions $. Le vol n'a pas été retenu.

Waymo, filiale de Google, va développer le système de conduite autonome.

OVH aura un 2e datacenter aux USA, capacité : 80 000 serveurs.

Ford annonce un investissement d'un milliard dans Argo AI.

Apple collabore officiellement avec le Wireless Power Consortium, la recharge sans fil devrait donc apparaître en 2017 et a rejoint l'OpenAI, sur l'intelligence artificielle.

Non, un geek n'est pas forcément développeur et un développeur n'est pas forcément un geek. Marre de mettre le mot geek a toutes les sauces médiatiques.

La politique américaine de Trump est-elle une bonne occasion pour l'Europe d'émerger (réellement) sur les nouvelles technologies ?

iPhone 2017, 2 ou 3 modèles, recharge à distance ou pas, +1000 € pour le haut de gamme ou pas, plus de bouton Home ou pas… Bref, attendons le mois de septembre…

Nintendo Switch, la nouvelle console Nintendo, disponible le 3 mars en France, gros succès et pénurie dès son lancement ?

La 5G a son logo et méga surprise, il s'agit d'un 5 !

SpaceX veut accélérer les lancements dès cette année.

Postscript a des failles et compromet la sécurité des imprimantes.

Planet of the Apps, la série créée par Apple, sera diffusée sur Apple Music au printemps. Quel développeur gagnera ?

Gmail ne sera plus disponible sur XP et Vista.

Chômage dans le monde informatique : on en parle ?

En décembre 2016, Pôle emploi annonçait qu'il y avait 48 800 personnes inscrites dans la catégorie "systèmes d'information et de télécommunication". D'un mois sur l'autre, ce chômage IT n'avait pas évolué. Pas de quoi être satisfait car 2016 a été une mauvaise année avec plusieurs mois de hausses. Pis, depuis septembre 2013, ce chiffre n'a jamais pu descendre sous les 40 000 (stats de Pôle emploi).

On parle de pénurie de certains profils, de problèmes de recrutement. Ce qui est vrai. Mais alors comment expliquer cette persistance du chômage IT qui, au mieux est stable, mais à un niveau élevé, au pire, en progression ? Il serait peut être temps de s'y pencher sérieusement.

Interrogation : dans ce chiffre, nous trouvons aussi celles et ceux qui lancent une activité en indépendant ou montent une société, ce % n'est pas précisé dans les statistiques.

QUELLES TENDANCES POUR LES LANGAGES : INDEX TIOBE VS LES AUTRES

A Programmez!, nous utilisons beaucoup l'index Tiobe pour les tendances de recherches. Cet index, et les % fournis, vaut ce qu'il vaut mais peut donner des indices sur les tendances réelles. Nous allons comparer quelques chiffres du State of Octoverse 2016 datant de septembre 2016 (GitHub) et du Tiobe, pour la même période.

Octoverse indique l'activité des langages, requêtes, patchs, etc.

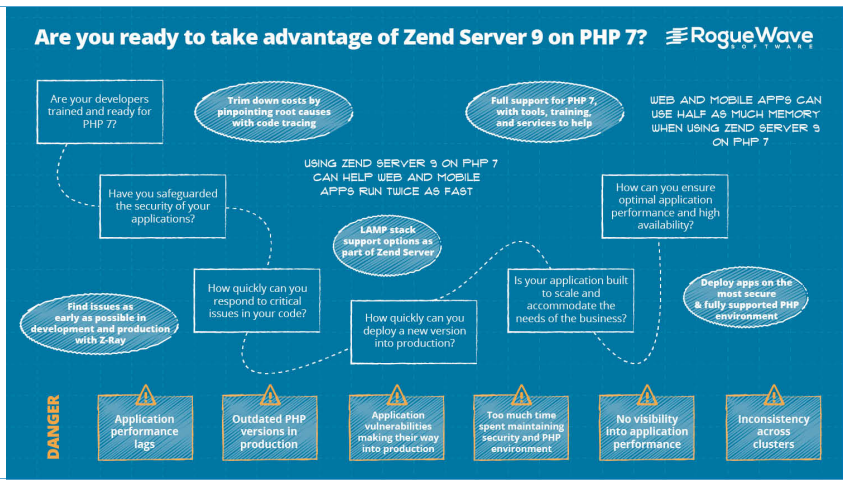
Langage	Octoverse 2016	Tiobe septembre 2016
JavaScript	1 604 219	6e 2,929 %
Java	763 783	1e 18,2 %
Python	744 610	5e 4,3 %
Ruby	740 610	12e 1,965 %
PHP	478 153	7e 2,8 %
C++	330 259	3e 6,6 %
CSS	271 782	-
C#	229 985	4e 5,49 %
C	202 295	2e 10,955 %
Go	188 121	19e 1,6 %
Shell	143 071	-
Objective-C	75 478	14e 1,8 %
Scala	70 216	au-delà de 20e
Swift	62 284	13e 1,9 %

Cette comparaison est intéressante et montre bien l'activité réelle des développeurs sur les projets et requêtes GitHub. JavaScript est sans conteste le plus actif. Java, Python et Ruby se tiennent, alors que sur Tiobe, Ruby est très loin. Pour Objective-C et Swift, nous retrouvons grosso-modo les mêmes places.

Par contre, C++, C#, C sont actifs mais représentent à peine le tiers de l'activité d'un Python ou Java.

ZEND, OÙ ES-TU ?

En octobre 2015, l'éditeur Rogue Wave, connu pour ses outils C++, rachetait Zend Technologies. Il faut bien avouer que depuis cette date, Zend a été plutôt discret, tout comme Rogue Wave. Les principales annonces 2016 ont été Zend Server + PHP 7, Zend Studio 13.5, Zend Framework 3 et la ZendCon 2016. Nous avons connu éditeur plus actif surtout dans le monde PHP et du développement. Ce rachat avait suscité de nombreuses questions. Et 18 mois après le rachat, on ne peut pas dire que la situation et la stratégie soient claires. Espérons que 2017 soit le renouveau de Zend. Faut-il sauver le soldat Zend ?



[2017]

WINDEV 22

922 NOUVEAUTÉS TECHNOLOGIQUES INDISPENSABLES

La nouvelle version 22 de WINDEV est la version **la plus riche** en nouveautés.

Parmi les 922 nouveautés vous bénéficierez de nouvelles technologies indispensables : ● **Le nouveau champ Traitement de Texte** permet de créer et manipuler des documents **sans sortir de l'application** (et également de les gérer en WLangage) ● Le nouvel **éditeur d'images** orienté développeur permet de retoucher vos images, créer vos icones et vos images «5 états» **sans quitter l'environnement** ● 22 nouveautés boostent les **tables** ● 11 nouveautés boostent vos **plannings** ● **Les nouveaux graphes** ● **IOT** (Objets connectés): support de la norme MQTT ● Pour améliorer la **vitesse des requêtes, HFSQL** trouve les meilleures clés d'index de chaque serveur en exploitation ● **HFSQL** bénéficie d'un **nouveau tableau de bord** ● Installez votre **GDS dans le cloud** en 3 clics pour **2 Euros par mois** ● **Nouveau GDS visuel:** gérez les branches d'un clic ● Les centaines de nouvelles fonctions **WLangage** ● **Le Code Coverage** affiche le pourcentage de code testé ● Le **GO** de projet WINDEV Mobile dans WINDEV ● Dans **WINDEV Mobile 22**, vous êtes averti immédiatement si une ligne de code que vous tapez ne fonctionnera pas sur un système ● Depuis WEBDEV, utilisez des composants **Angular JS**, **Bootstrap**, **jQuery UI**... ● **Télémétrie sur mobile** ● Créez des **Webservices REST** ● Créez des sites complets dans une seule page (**Single Page Application**) ● Utilisez les mots de passe de **Facebook**,... comme identifiants de vos applications et vos sites ● **WebSocket**: c'est le serveur Web qui envoie lui-même les données modifiées aux pages ● Intégrez automatiquement les **trackers Google Analytics** dans vos sites ● Etc, Etc... (liste exhaustive des nouveautés dans la revue de 92 pages accessible sur pcsoft.fr)

Tél Paris: 01 48 01 48 88
Tél Montpellier: 04 67 032 032

WWW.PCSOFT.FR

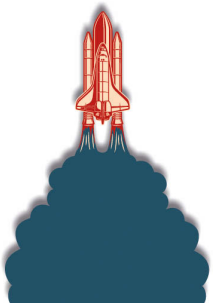
AFFLUENCE RECORD DANS LES 12 VILLES DU «WINDEV TOUR 22»

Merci de votre fidélité

mars

BIG DATA PARIS 2017

6 & 7 mars / Paris



Cette 6e édition parlera beaucoup d'Intelligence Artificielle et de Machine Learning. 170 exposants, des labs orientés IA, un village startup, de nombreuses conférences.
Pour en savoir plus : www.bigdataparis.com

GAME OF CODE

11 & 12 mars / Luxembourg

Relevez le défi de programmation du Game of Coding, un grand hackathon qui se déroulera au Luxembourg. C'est une compétition de 24h avec défis, coaching mais aussi l'occasion de voir du monde, d'autres dévs et de s'amurer un peu (ou beaucoup).
Toutes les infos : http://www.gameofcode.eu

LAVAL VIRTUAL 2017

du 22 au 26 mars / Laval

Le grand salon sur les réalités alternatives revient à Laval, pour la 19e édition. C'est le lieu incontournable pour parler réalité virtuelle / augmentée / mixte. On parle matériel, formation, logiciel mais aussi et surtout usages qu'ils soient grand public ou industriels.
Attention : les 22, 23 et 24 mars sont réservés aux professionnels.
Informations : http://www.laval-virtual.org

LES ÉVÈNEMENTS ZENIKA

- Matinale Agile le 7 mars : le lancement et le quotidien d'un projet Agile, Zenika Lille.
- Matinale Réalité virtuelle le 21 mars : quels usages en 2017 ? Zenika Paris.
- Agile Wake Up le 30 mars : les tests et l'agilité, Zenika Paris.

Informations et inscription sur notre site : https://zenika.com/agenda/

- Forum recrutement "Code in the region " le 14 mars, Zenika Paris.

Cet évènement est dédié à tous les Parisiens qui souhaitent partir en région : informations et inscription : https://jobs.zenika.com/codeintheregion/

Avril

JOURNÉE FRANÇAISE DES TESTS LOGICIELS

11 avril / Montrouge

Cette 9e édition se tiendra le 11 avril à Montrouge. 16 conférences seront organisées durant la journée.
Le 10 avril se déroulera une journée complète de tutoriels. "Afin de permettre aux professionnels d'approfondir certaines problématiques du test à travers des enseignements pratiques, le CFTL propose, dans une approche pédagogique, différents tutoriels d'une durée de 3 heures, dispensés par des praticiens expérimentés, sur les thèmes de l'utilisabilité, du test en contexte agile, de l'automatisation pour le Web et le mobile, des processus métier et test de recette, et des environnements de test. » dixit les organisateurs.

REBUILD 2017

27 avril / Nantes

Les communautés Microsoft seront à l'honneur le 27 avril pour la grande journée Rebuild. Plus de 40 sessions seront jouées. Elles seront réparties en 3 tracks : décideurs, IT et développeurs. Une occasion d'échanger et de parler technologies et tendances actuelles.

mai

PHP TOUR 2017

18 et 19 mai / Nantes

La grande conférence itinérante PHP s'arrêtera cette année à Nantes !

juin

DEVFEST LILLE 2017

9 juin / Lille

Le Google Developer Group de Lille (@GDGLille) organisera la première édition du DevFest Lille. Au programme, des conférences et des codelabs autour des technologies que nous aimons tous : Web, Mobilité, Cloud, IoT, … Les inscriptions et le CFP sont ouverts. Rendez-vous sur https://devfest.gdglille.org/ .

MAKER FAIRE PARIS 2017

du 9 au 11 juin / Paris

Cette nouvelle édition du Maker Faire se déplace à la Villette et se tiendra en dehors de la foire de Paris comme ce fut le cas les années passées. Comme chaque année, ce sera l'occasion de découvrir des centaines de makers et de projets divers et variés. L'appel aux makers a été lancé il y a quelques jours.

Communautés

JUG PARIS

14 mars, soirée Lagom & Hand on lab JDK 9.

- **JUG Toulouse :**
23 mars, programmation concurrente dans l'univers Java.

- **Python Grenoble :**
15 mars, langage Julia ;
12 avril : Docker et Python.

- **CocoaHeads Strasbourg :**
16 mars, meetup
sur Realm, base de données mobile.

- **GDG Nantes :**
8 mars, soirée Cloud / Next'17.

- **Software Craftsmanship Bordeaux :**
13 mars, Coding Dojo.

BIGDATA by CORP.
PARIS 2017

Congrès & Expo

6e édition

Rendez-vous les
6 & 7 mars 2017
Palais des Congrès

+12 000 participants
+200 exposants
+250 intervenants

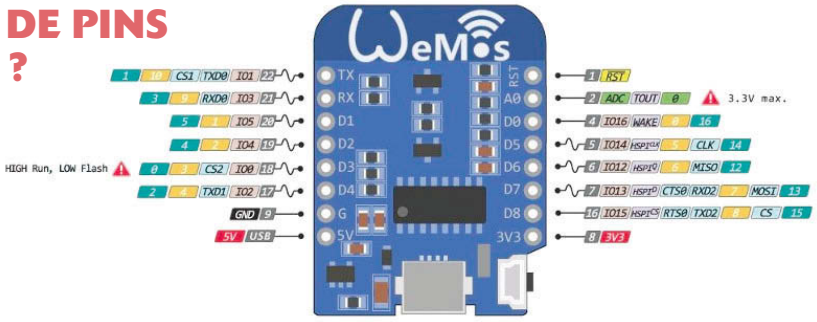
Participez à l'**événement leader** en France et **préparez l'avenir** Big Data de **votre entreprise**

Master the Data Galaxy

www.bigdataparis.com by CORP. in Corporations we Trust

UN PROBLÈME DE MAPPING DE PINS SUR VOTRE WEMOS D1 MINI ?

D'une manière générale, les ESP8266 ne possèdent pas autant de GPIO qu'une Arduino, et quand vous souhaitez utiliser des capteurs ou un projet Arduino, vous allez peut-être vous confronter à un problème simple : comment connecter mon capteur. C'est bête mais vous perdrez beaucoup de temps.



Nous avons eu ce problème en testant un module RFID, modèle MFRC522. En Arduino, nous aurions les GPIO 9, 10, 11, 12, 13 pour le SDA, RST, SCK, MOSI et MISO. Or, en D1 Mini, nous n'avons pas exactement ces pins. Cela fonctionne avec la board D1 qui possède un GPIO étendu mais pas en D1 Mini. Pour résoudre le problème, nous avons pris le branchement utilisé sur un ESP8266 (type ESP-12).

Ensuite, nous prenons le schéma des pins de la WeMos pour les correspondances, ce qui donne :

RFID	ESP8266	WeMoS D1 Mini
RST	GPIO15	D8
SDA (SS)	GPIO2	D4
MOSI	GPIO13	D7
MISO	GPIO12	D6
SCK	GPIO14	D5
GND	GND	GND
3,3V	3,3V	3,3V

En théorie, nous sommes bons. Sauf que dans cette configuration, vous allez avoir des erreurs dans le téléversement du code depuis Arduino IDE. Le problème vient de la pin RST qui ne doit pas être en D8. Nous l'avons mis en D1 et le SDA en D2. Et là, tout fonctionne.

Autre subtilité du mapping, dans le code RFID, ne pas oublier de mettre la désignation D et non uniquement le numéro GPIO.

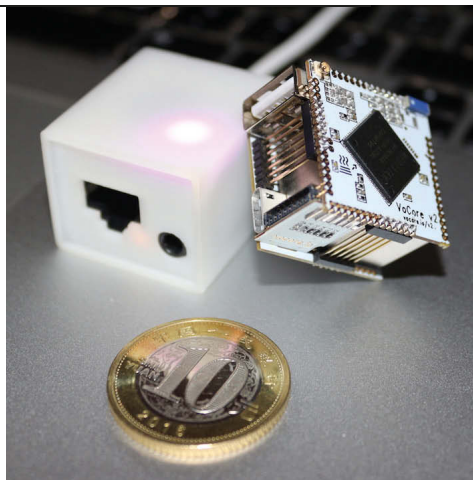
Sur la partie mapping entre Arduino et WeMos / ESP, vous avez des librairies dédiées : pins_arduino.h pour faire le mapping et l'étendre.

Nous reviendrons rapidement sur ce problème avec une solution matérielle : les expenders I/O.

Merci à Sébastien Warin pour nous avoir donné des pistes. •

VOCORE2 : EH MEC OÙ EST MON PC ?

VoCore ? Pour beaucoup d'entre vous, ce nom ne dira rien. Créée par VoCore Studio, cette carte est assez impressionnante par sa taille et surtout par la puissance matérielle sur un circuit de 2,5 cm de côté ! Nous avons été impressionnés par cette board.



La VoCore est ce que l'on appelle en Anglais, un "coin-sized Linux Computer" que l'on peut traduire par "ordinateur (sous Linux) de la taille d'une pièce de monnaie ! Les créateurs ont réussi un pari assez fou.

Les principales caractéristiques sont :

- SoC Mediatek MT728AN
- 128 Mo de mémoire vive (DDR2)
- Stockage interne de 16 Mo, extensible via SD
- WiFi 802.11n
- Support Ethernet 10/100
- Support USB 2
- Support PCIe 1.1
- 30 GPIO
- 3 UART
- Broches PWM
- Consommation de 74 mA (wifi en pause), 230 mA en pleine puissance

A la rédaction, nous avons opté pour la version Ultimate qui est un module monté incluant le support SD, Ethernet, le WiFi et un boîtier pour loger le serveur. Les dimensions font rêver : 2,4 cm de hauteur, 2,8 cm de largeur et 3,2 cm de profondeur !

Honnêtement, c'est assez bluffant. Nous vous conseillons vivement une édition prépackagée et soudée. Les soudures à faire pour fixer les headers sont très fines, si vous n'êtes un bon soudeur, nous vous déconseillons vivement de tenter l'aventure.

Le VoCore2 est prêt à l'emploi car OpenWRT est pré-installé sur le stockage interne. OpenWRT est une distribution Linux adaptée aux environnements embarqués, aux objets connectés un peu complexes et aux routeurs. Actuellement, 1150 périphériques utilisent OpenWRT.

Notre premier réflexe est de transformer le VoCore 2 en nano serveur, et il remplit parfaitement le travail. Nous l'avons testé en serveur de fichiers SAMBA et en mini serveur Web. C'est largement suffisant pour un usage personnel, à la maison ou même dans une petite entreprise. Par contre, attention à ne pas perdre la petite boîte.

L'administration de base d'OpenWRT se fait via une connexion SSH à la machine depuis Putty ou un Terminal. Ensuite, c'est un banal Linux. Pour l'administration simple et rapide, vous pouvez utiliser l'interface Web, via LuCi. Il suffit de taper dans son navigateur l'adresse IP de la machine. Bien entendu, le VoCore doit être actif et connecté. Si vous êtes habitué à l'admin Linux aucun souci pour vous ! Pour les autres, il faudra un peu de pratique.

Bien entendu, comme OpenWRT est orienté réseau / routeur, vous pouvez très bien aller dans l'admin réseau, la gestion IP, du DNS, etc. Un des usages courants de cette distribution est de monter son propre routeur. Pour développer, vous devrez monter une VM ou conteneur avec OpenWRT ou passer par une VM Ubuntu (ou autre) puis installer les extensions et les outils nécessaires. Par contre, cette board n'est pas "interface friendly" donc si vous n'êtes pas très à l'aise avec la ligne de commandes Linux, nous vous la déconseillons. Si vous souhaitez vous lancer dans le monde réseau, Linux, et même IoT, pour nous, c'est un excellent choix, moins facile qu'un choix classique tel qu'une CHIP ou une Raspberry Pi mais le potentiel est là !

Côté tarifs, le VoCore2 seul est proposé sur le site officiel à 17,99 $, la version Ultimate à 44,99 $. La version intermédiaire, VoCore + Docker est proposée à 36,99 $.

Vous trouverez des tutoriels et des idées sur : http://vonger.cn •

Abonnez-vous à programmez!

le magazine des développeurs

Nos classiques

1 an ..... **49€***
11 numéros

2 ans ..... **79€***
22 numéros

Etudiant ..... **39€***
1 an - 11 numéros * *Tarifs France métropolitaine*

Abonnement numérique

PDF ..... **35€***
1 an - 11 numéros

Souscription uniquement sur
www.programmez.com

Option :
accès aux archives 10€

Nos offres spéciales printemps 2017

1 an ..... **59€**
11 numéros + 1 vidéo ENI au choix :
- Big Data avec Hadoop
- Framework Spring

(Valeur de la vidéo de 29,99 à 59,99 €)

2 ans ..... **89€**
22 numéros + 1 vidéo ENI au choix :
- Big Data avec Hadoop
- Framework Spring

(Valeur de la vidéo de 29,99 à 59,99 €)

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

☐ **Abonnement 1 an :** 59 €
11 numéros + 1 vidéo ENI au choix :
☐ **Abonnement 2 ans :** 89 €
22 numéros + 1 vidéo ENI au choix :

☐ Vidéo : Big Data avec Hadoop
☐ Vidéo : Framework Spring

PROG 205
Offre limitée, valable jusqu'au 31 mars 2017

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

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

E-mail : @

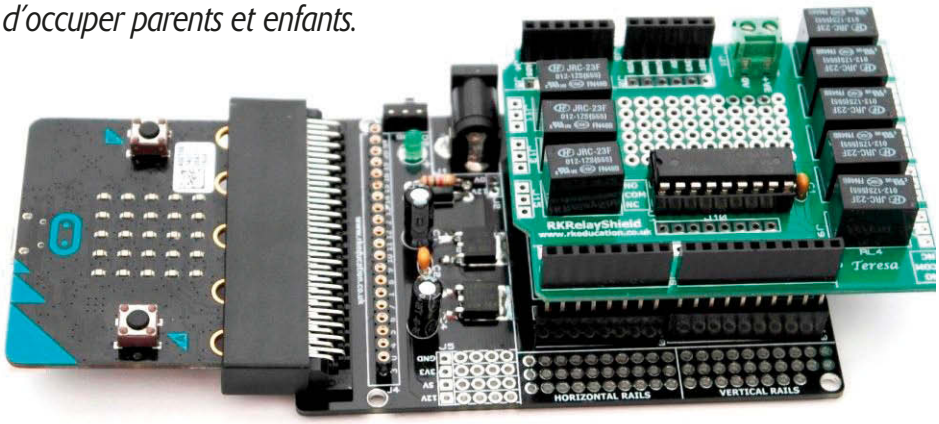
☐ Je joins mon règlement par chèque à l'ordre de Programmez !
☐ Je souhaite régler à réception de facture

*** Tarifs France métropolitaine**

Découverte de la carte Micro:bit

• Thierry Chantier
Geek de toujours, artisan du code depuis plus de 20 ans.
Co-Fondateur de l'association Mixteen, proposant de façon bénévole des ateliers de découvertes du code et du numérique (Scratch, Python, ateliers déconnectés, Thymio, micro:bit, Poppy…) : https://www.mix-it.fr/mixteen.
Email : titimoby@gmail.com
Github de cet article :
https://github.com/titimoby/microbit4all/tree/master/Programmez

Je ne sais pas pour vous, mais s'il y a quelque chose qui m'agace rapidement, ce sont les dimanches pluvieux. Il faut alors toute la ruse et le passé d'un vieux geek pour trouver des ressources permettant d'occuper parents et enfants.



Un passé geek qui ressurgit

Ce dimanche là, la discussion a rapidement dérivé sur ce qu'on pourrait bien tenter de plus avec une des Raspberry Pi de la maison. De fil en aiguille, nous en sommes venus à comparer ce que l'école propose de nos jours et ce qu'elle a proposé dans le domaine informatique. Comme souvent, la discussion s'illustre de quelques pages Web, recherchant quelques traces de ce passé pas si lointain. Lorsque je me suis mis à évoquer le Plan Informatique Pour Tous des années 1980 (oui, du siècle dernier) un nom est revenu me hanter : BBC Micro.

La BBC avait supervisé la production d'un micro ordinateur visant l'éducation. La conception et la production furent confiées à Acorn qui proposa une gamme dans les années 80 aboutissant à des machines qui faisaient bien plus rêver que ce que ma salle de Thomson MO5 et TO7 pouvait me proposer. (Ah, l'Archimedes !). C'est à ce moment que le passé a rejoint le présent : un site nous annonçait fièrement le lancement de la carte micro:bit qui serait distribuée aux écoliers anglais dès la rentrée scolaire 2016.

La BBC s'est vue confier un bien beau dossier par le gouvernement Britannique : concevoir une carte susceptible d'encourager les jeunes anglais à transiter de consommateur à acteurs du monde numérique. Tout ceci devant s'appuyer sur une carte leur permettant de concevoir des programmes informatiques et de construire des objets électroniques de toutes sortes.

Le projet visait d'ailleurs initialement la rentrée scolaire 2015, mais l'ampleur de la tâche et le fait d'avoir 29 partenaires impliqués ont retardé l'arrivée des premiers modèles en mars 2016. Ce n'est que pour la rentrée 2016 que la quantité pharaonique d'un million de cartes a pu être livrée.

Le dernier point important est que tout l'ensemble de ce projet est à présent sous la tutelle d'une fondation qui est dédiée au développement de cet écosystème.

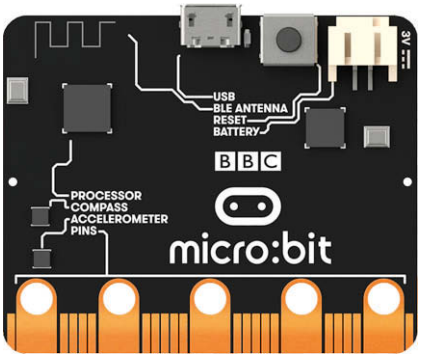
La porte est ouverte à toutes sortes de collaborations, n'hésitez pas à vous documenter par http://tech.microbit.org/ et venez en discuter sur les différents channels Slack.

La carte et ses extensions

Concrètement, à quoi ressemble cette carte et de quoi est elle capable ?

La BBC a réussi le tour de force de rassembler sur une carte de 4x5 cm une belle série de composants et de fonctions :

- Une série de 25 Leds disposées en un carré de 5 par 5 ;
- Deux boutons poussoirs ;
- Un capteur de température ;
- Une boussole, indiquant la direction et les changements d'orientation ;
- Un accéléromètre, indiquant les vitesses des mouvements sur chaque axe ;
- Trois gros connecteurs numérotés 0, 1 et 2, utilisables avec des pinces crocodiles ou des fiches bananes, points d'entrée/sortie de la micro:bit ;
- Un connecteur 3V et un connecteur GND pour alimenter les montages ;
- 20 petites broches, nécessitant un connecteur additionnel pour être utilisées simplement ;
- Un connecteur pour une batterie ;
- Un composant Bluetooth/Radio : la micro:bit est un bel objet connecté ;
- Un connecteur micro USB, permettant la programmation et l'alimentation ;



Pour une si petite carte, avec des jeunes comme cible, le travail est vraiment complet et bien intégré.

En plus de la carte elle-même, un certain nombre d'accessoires permettent d'aller plus loin. Des kits proposent d'utiliser l'ensemble des broches disponibles, il y a des batteries externes et même un kit pour réaliser un petit buggy pilotable depuis son smartphone.

Les outils disponibles et leur approche

Pour réaliser tous ces projets, il fallait bien entendu de quoi programmer la carte micro:bit.

Et là, la fondation MicroBit voit les choses en grand et surtout en grande diversité.

Il y a tout d'abord la possibilité de programmer à l'aide d'outils très visuels, proches de ce que propose Scratch. Dans ces outils, il s'agit d'agencer des blocs représentant les différentes actions envisageables.

- Code Kingdoms Javascript : les blocs sont convertis en Javascript, permettant de progressivement passer de programmes purement visuels à des scripts entièrement en Javascript ;
- Microsoft Block Editor : l'outil est purement visuel, mais il propose la conversion pour un second outil de Microsoft, Touch Developp ;

- Microsoft Touch Develop : une approche par blocs permet de générer une partie du code, il suffit de compléter ensuite ;
- Microsoft PXT : dernier outil en date, il est également celui qui est le plus dynamique. Les mises à jour sont nombreuses et la collaboration hors de murs de Microsoft est très facile (une partie de la traduction française vient de votre serviteur ;)). En outre, le même outil permet de passer du mode programmation par blocs à un éditeur de code Javascript. C'est un redoutable outil d'apprentissage.

Ces outils proposent des projets complets avec des explications détaillées, de quoi se lancer sereinement dans l'aventure.

Il est également possible d'utiliser le langage Python pour coder sa carte micro:bit. Pour être précis, il s'agit de la version Micro Python, développée pour permettre l'usage de Python avec des cartes basées sur des micro contrôleurs comme les ESP8266 par exemple.

Ici encore, il y a plusieurs possibilités :

- L'éditeur proposé par la fondation, hébergé sur le site. Il permet de produire du code Python et de le flasher sur la carte sans rien installer de plus sur son ordinateur que son navigateur habituel.
- Un éditeur tiers, le Mu Code Editor : celui-ci s'installe sur la machine propose une interface simple mais complète pour coder sa carte micro:bit. Quelques outils supplémentaires comme un explorateur de fichiers sur la carte et un shell dédié en font un superbe outil.

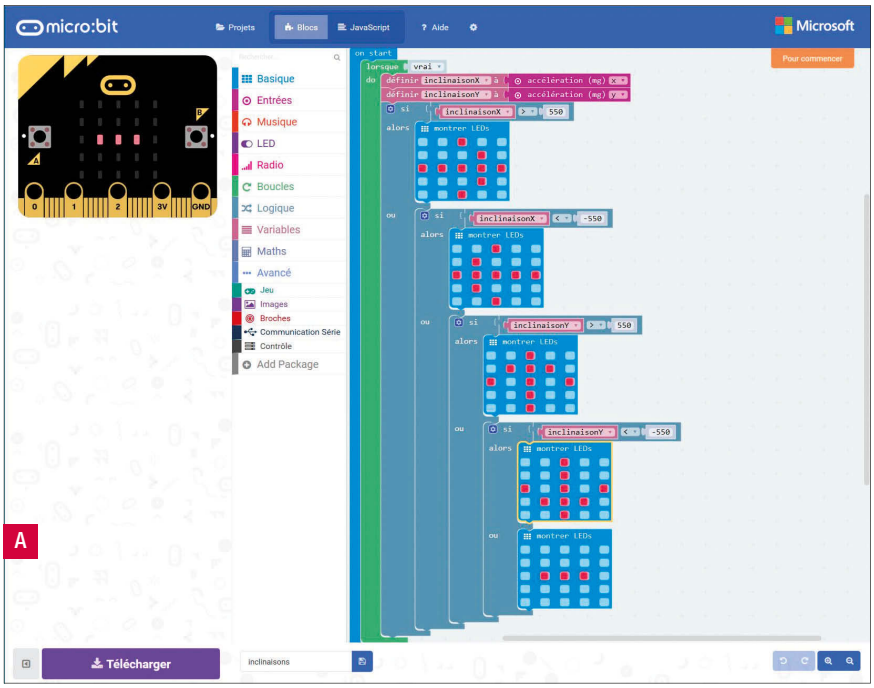
Exemple concret : affichage de flèches en fonction de l'inclinaison de la carte

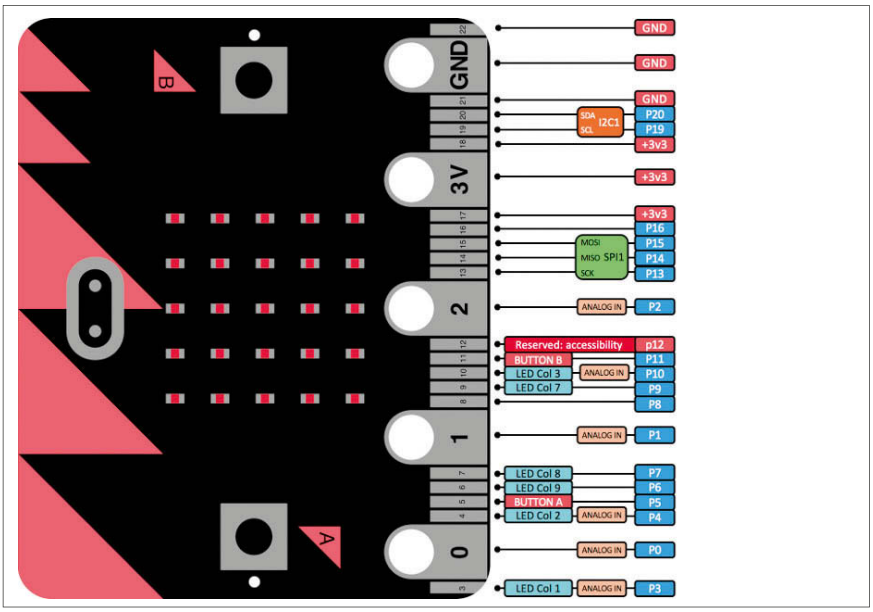
Tout cela est bien beau, mais pour occuper toute la famille, il faut plus qu'un conte de fée avec des marraines issues du passé numérique d'un vieux geek. Nous avons la carte (oui, le temps de vous parler de la carte et j'en ai reçu une, et même plusieurs tant qu'à faire) et nous avons les outils. Pour commencer, je vous propose le programme suivant :

Suivant l'inclinaison de la carte micro:bit, affichons une flèche avec les Leds indiquant l'inclinaison détectée.

Nous ferons cela à l'aide de l'éditeur PXT. Vous pourrez ainsi apprendre à l'aide des blocs graphiques tout en regardant le code Javascript accessible.

Bien entendu, tout ce que nous allons faire sera accessible par un dépôt sur le site Github : https://github.com/titimoby/microbit4all/tree/master/Programmez





Le meilleur exercice sera de reproduire ce que vous voyez sur l'image illustrant cet article. Reproduire, c'est déjà apprendre. [A]

Lorsque vous démarrez, l'éditeur vous montre 2 blocs bleus : "On start" et "Toujours" (oui, toute la traduction n'est pas encore faite).

Enlevez le bloc "Toujours" en le déplaçant dans la colonne proposant tous les autres blocs.

Un symbole de corbeille apparaît, vous pouvez relâcher la souris.

Dans le menu "Boucles" allez chercher le bloc vert "Lorsque vrai… do" et positionnez le à l'intérieur du bloc "On start".

Tout ce que vous ajouterez alors dans le bloc vert sera répété indéfiniment et dans cet ordre. C'est cette boucle que nous allons remplir.

Depuis le menu "Variables", créez 2 variables que vous nommerez "InclinaisonX" et "InclinaisonY".

Imaginez que ces variables sont des tiroirs que vous étiquetez et dans lesquels vous rangerez ce dont vous aurez besoin.

C'est ce que vous ferez ensuite en positionnant 2 blocs "Définir " du menu Variables"

A la place des valeurs "0", allez chercher dans le menu le bloc "accéléromètre (mg)".

Vous changerez la valeur "x" par "y" pour le bloc "Définir" de "InclinaisonY".

Cela signifie qu'à chaque fois que nous exécutons notre boucle, nous récupérons les valeurs de l'accéléromètre sur l'axe des X (gauche / droite) et l'axe des Y (avant / arrière).

Découvrez tangente le magazine des mathématiques

Pour mieux comprendre le monde

tangente
l'aventure mathématique

Aussi en version numérique !

Tous les deux mois chez votre marchand de journaux

tangente
l'aventure mathématique

Politique, économie, finance, jeux, musique, littérature, arts plastiques, architecture, **informatique**, physique, biologie, géographie : **les mathématiques sont partout !!!** Le magazine *Tangente* et ses hors séries vous aident à redécouvrir notre quotidien.

Les hors séries « kiosque »

4 fois par an, un hors série d'au moins 56 pages, explore un grand dossier de savoir ou de culture. **Derniers parus** : Les angles - Les graphes - Les démonstrations - Les fonctions - Maths des assurances - Maths et médecine - La Droite - Maths et Architecture - Les ensembles

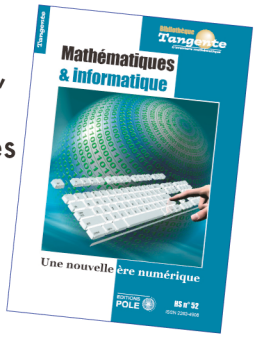
Disponibles chez votre marchand de journaux ou avec **l'abonnement PLUS.**

Vous voulez vous rendre compte de ce qu'est la consultation numérique d'un numéro de *Tangente* ?

L'accès au numéro 167 vous est offert !

Rendez-vous sur http://tangente-mag.com (identifiez-vous)

La « Bibliothèque Tangente »



Pour les lecteurs les plus curieux, les articles des hors séries de *Tangente* sont repris et complétés dans la Bibliothèque Tangente, avec de magnifiques ouvrages d'environ 160 pages (prix unitaire 20 à 22 €), richement illustrés, disponibles

- sur la boutique du site **www.infinimath.com**
- chez votre libraire
- avec **l'abonnement SUPERPLUS.**

http://tangente-mag.com

DEMANDEZ UN ANCIEN NUMÉRO de *Tangente* - **Joignez juste 3 € de timbres**

À adresser à *TANGENTE* - 80 BD SAINT-MICHEL - 75006 PARIS avant le 31/03/17

NOM* . . . . . PRÉNOM* . . . . .

ADRESSE* . . . . .

CODE POSTAL* . . . . . VILLE* . . . . . PAYS . . . . .

MAIL* . . . . . TÉLÉPHONE* . . . . .

ABONNÉ À PROGRAMMEZ ❒ OUI ❒ NON PROFESSION . . . . . .

Nous allons ensuite tester les valeurs reçues et en fonction de certains seuils, nous allumerons les Leds nécessaires pour faire apparaître une flèche directionnelle. Les tests se font à l'aide des blocs du menu "Logique". Nous allons utiliser le bloc "Si...Alors..Ou".

Le principe est le suivant : si la condition indiquée dans la partie "Si" est vraie, la carte exécutera ce qui se trouve dans le "Alors". Dans le cas contraire, elle exécutera le code se trouvant dans le "Ou".

Nous allons les enchaîner avec l'idée suivante :

- Si (un seuil sur X est atteint à droite) ;
- Alors affichons (une flèche pointant à droite) ;
- Ou
  - Si (un seuil sur X est atteint à gauche)
  - Alors affichons (une flèche pointant à gauche).
  - Ou
    - Si (un seuil sur Y est atteint à l'avant)
    - Alors affichons (une flèche vers l'avant)
    - Ou
      - Si (un seuil sur Y est atteint à l'arrière)
      - Alors affichons (une flèche vers l'arrière)
      - Ou affichons (un tiret)

Les seuils que je vous propose (550 et -550) ont été établis de manière empirique sur mes cartes. Mais vous pouvez changer les valeurs plus tard pour expérimenter la sensibilité de l'accéléromètre. Revenons à notre éditeur PXT. Pour chaque bloc "Si..Alors..Ou" vous ajouterez :

- Un bloc "Logique" indiquant qu'une valeur est supérieure ou inférieure à une autre ;
- Dans la première valeur, insérez depuis le menu "Variables" l'inclinaison X ou Y ;
- Dans la seconde, saisissez soit 550 soit -550 selon les cas ;
- Dans le bloc "Alors", ajoutez un bloc "Montrer Leds" du menu "Basique" et dessinez la flèche qui correspond au cas ;
- Imbriquez dans le bloc "Ou" le bloc "Si..Alors..Ou" suivant.
- Quand vous avez géré les 4 cas (InclinaisonX > 550 , InclinaisonX < -550, InclinaisonY > 550 et InclinaisonY < -550), insérez directement un bloc "Montrer Leds" et dessinez un tiret.

Le programme est à présent complet. Avant de l'enregistrer sur la carte, vous pouvez le tester à l'aide du simulateur inclus dans l'éditeur PXT.

Par défaut, le code est d'ailleurs déjà à jour et en route.

L'éditeur PXT se charge de tenir le simulateur dans l'état le plus à jour possible.

Si vous cliquez sur le visuel représentant la carte micro:bit, vous pouvez simuler l'inclinaison en déplaçant votre souris dans les 4 directions. La flèche voulue doit alors s'afficher.

En cas de besoin, le simulateur permet également d'utiliser les boutons A et B. Vous êtes alors prêts à compiler votre programme et le copier sur la carte. Tout d'abord, branchez la carte micro:bit. Elle apparaît comme une clé USB appelée « MICROBIT ».

L'explorateur de fichier standard vous montre un fichier « DETAILS.TXT » contenant quelques détails sur le firmware actuellement utilisé par votre carte ainsi qu'un fichier « «MICROBIT.HTM ». Ne vous préoccupez pas de ces fichiers, laissez-les ainsi.

La suite des opérations est gérée par l'éditeur PXT de manière fort simple :

- Cliquez le gros bouton « Télécharger ».
- Votre navigateur vous propose alors de sauvegarder un fichier « nom de votre programme.hex »
- Enregistrez-le directement sur la carte.

Et c'est tout : la carte possède une Led jaune qui va clignoter au niveau de la prise micro USB le temps de sauvegarder votre programme. La carte redémarre ensuite toute seule.

Pas de panique, cela fera disparaître le lecteur correspondant, mais il réapparaîtra seul.

Le programme se lance ensuite : vous voilà officiellement développeur !

Manipulez votre carte, inclinez-la et les flèches doivent changer au fur et à mesure des changement d'inclinaison.

Si vous avez du mal à reproduire ce qui est montré sur la copie d'écran, vous pouvez tenter une des deux autres approches possibles :

La première possibilité consiste à ouvrir le fichier « Programmez_Inclinaisons_PXT.js » (disponible sur le github de l'article) ;

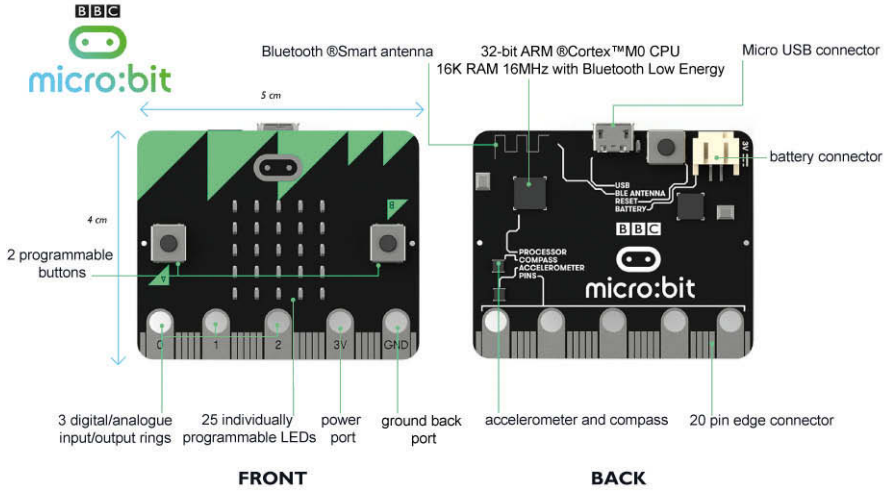
Dans l'éditeur PXT, sélectionnez le mode JavaScript. Copiez le contenu du premier fichier et collez le dans l'éditeur PXT.

Vous pouvez alors repasser en mode Blocs.

La seconde possibilité est encore en test au moment où j'écris ces lignes. Il s'agit d'utiliser le menu "Projets" puis l'option "Import file" ;

Vous pouvez alors choisir le fichier « Programmez_Inclinaisons_PXT.hex » ;

Votre dernière chance est de me contacter ou de rejoindre les différentes communautés existantes.



Pour aller plus loin

A partir de là, plus rien ne peux vous arrêter !

Si vous le souhaitez, de nombreux exemples sont disponibles :

- Depuis le menu « Projets » de l'éditeur PXT ;
- Depuis le site microbit.org.

Si vous ou un de vos amis possédez une seconde carte, vous pouvez imaginer une suite à notre programme d'inclinaisons.

En effet, les cartes micro:bit ont un module radio encore plus simple à utiliser que le Bluetooth. Une des cartes peut alors envoyer un code correspondant au cas de flèches à afficher et la seconde affichera la flèche demandée. Je vous fournis quelques indices sur le Github de l'article. Si vous le souhaitez, vous pouvez également vous lancer dans des programmes écrits en Python à l'aide de l'éditeur en ligne ou bien avec Mu Code Editor.

Là encore, des indices et exemples sont disponibles sur les sites officiels de la fondation micro:bit ainsi que sur le Github de cet article.

Vous pouvez par exemple reproduire le même programme d'inclinaison mais cette fois écrit en Python.

Vous pourrez même améliorer l'original et déduire un angle d'inclinaison combinant l'axe X et Y et afficher bien plus que les 4 cas étudiés ici. Et maintenant, soyez curieux, rejoignez les communautés micro:bit, envoyez-moi des emails, n'attendez plus : lancez-vous, accompagnez vos enfants !

Liens

Fondation Micro:bit : http://microbit.org/fr/

Site de la communauté technique Micro:bit : http://tech.microbit.org/

Mu Code Editor : https://codewith.mu/

PIXEL, la nouvelle interface graphique du Raspberry Pi

Thomas Gayet,
Creative Technologist,
SQLI

SQLI
Digital Performance

Le Raspberry Pi, qui s'est vendu à plus de 10 millions d'exemplaires, s'offre une mise à jour de son interface graphique. Disponible depuis le 28 septembre dernier sous le nom de PIXEL (Pi Improved Xwindows Environment, Lightweight).

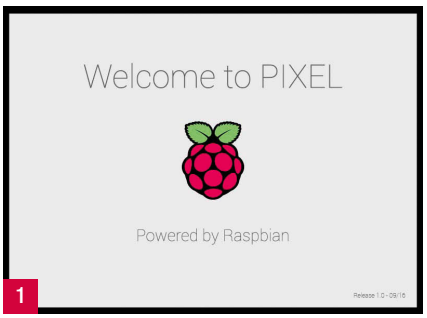
Ce nouveau bureau offre une approche plus grand public avec une interface retravaillée et plus de fonctionnalités disponibles.

« La majorité des ajouts de cette mise à jour se concentrent sur l'apparence du bureau : quelques changements fonctionnels et nouvelles applications sont proposés, mais il s'agit avant tout de rendre l'ensemble plus agréable à regarder » explique Simon Long – l'UX engineer qui travaille sur le projet PIXEL depuis 2 ans.

Au démarrage, un Splashscreen fait son apparition en lieu et place du traditionnel défilement des lignes de commandes [1]

D'autres améliorations sont embarquées dans cette nouvelle version : les fenêtres ont ainsi été revues et corrigées, ainsi que les menus. Raspbian propose maintenant 16 fonds d'écran, et différentes icônes et typos revues et corrigées, visant à les rendre plus claires et plus faciles à comprendre. De nouveaux outils de surveillance matérielle font aussi leur apparition : température de la carte mère, charge du processeur, etc. [2]

Le navigateur Epiphany, installé par défaut dans toutes les versions précédentes de Raspbian,

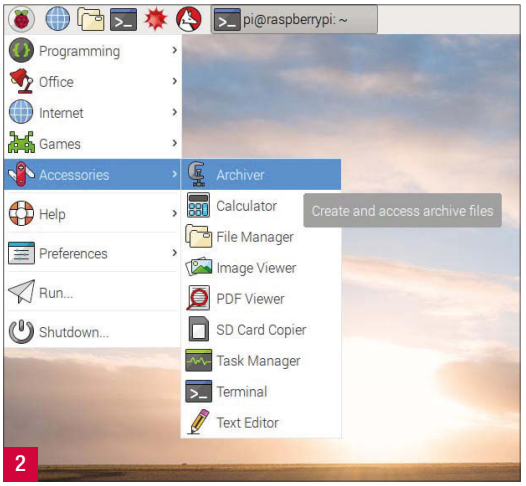


1

cède sa place à Chromium. Constitué du code libre et gratuit du navigateur Chrome de Google, Chromium profite un peu de la puissance de ce dernier, notamment en ce qui concerne l'utilisation de la carte graphique. Tandis qu'Epiphany luttait bien souvent pour décoder la moindre vidéo YouTube, Chromium est capable de tirer parti de la décompression matérielle h.264/AVC intégrée dans le processeur graphique. Cela devrait rendre le surf et la consultation multimédia plus agréable pour les versions les plus récentes du nano ordinateur.

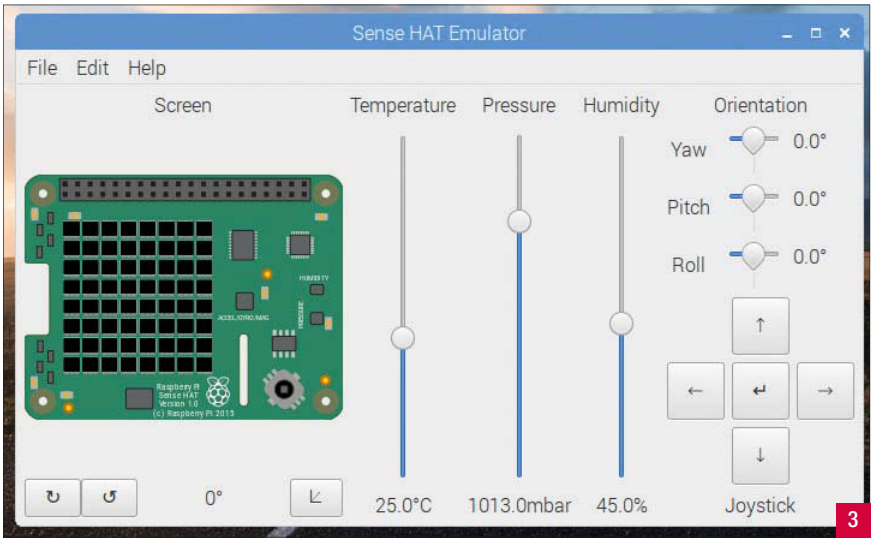
Epiphany ne disparaît cependant pas, il reste installé pour les versions les moins puissantes de Raspberry Pi (version A et Zero).

S'ajoutent à cela de nouvelles applications installées par défaut, telles que :



2

- RealVNC qui permet d'accéder au Raspberry Pi et à son interface depuis un ordinateur distant ;
- Un émulateur de Sense HAT, une extension du Raspberry Pi qui permet notamment de contrôler plusieurs capteurs ainsi qu'un affichage de 8x8 LED ;
- Des fonctions d'arrêt des connexions sans fil, qui permettent de désactiver le Wi-Fi et/ou le Bluetooth ;
- Des mises à jour des programmes existants ;
- Ainsi que de nombreuses corrections de bugs. [3]



3

La dernière version de Raspbian est la déclinaison 23.9.2016. Elle est déjà disponible en téléchargement avec PIXEL. Pour son activation, voici les lignes de commande :

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install -y rpi-chromium-mods
sudo apt-get install -y python-sense-emu python3-sense-emu
sudo apt-get install -y python-sense-emu-doc realvnc-vnc-viewer
```

Docker outil indispensable du Devops

Paquet Judicaël
DSI chez **batiwiz** *et*
Coach Agile & Devops
chez **Myagile Partner**

De plus en plus populaire ces dernières années, nous allons comprendre pourquoi docker est de plus en plus utilisé par les développeurs, et pourquoi il est devenu une arme indispensable dans le monde du Devops. Nous allons ensuite le prendre en main avec des exemples concrets afin d'apprendre à l'utiliser sur des cas concrets.

Docker est un logiciel libre qui permet d'automatiser le déploiement d'applications et de leurs dépendances dans différents conteneurs isolés sur n'importe quel serveur Linux. Docker a été développé par Solomon Hykes à la base comme projet interne de DotCloud en tant qu'évolution de solutions open-source déjà existantes au sein de la société. Docker a été distribuée en open-source la première fois le 13 mars 2013. Au moment où j'écris ces lignes, il y a 1600 contributeurs sur le Github de l'application.

Les conteneurs

Si la notion de conteneurs se démocratisa avec l'arrivée de Docker, il faut savoir que ce concept en réalité n'a pas été créé par Docker. Linux a en fait déjà un système de conteneurs qui s'appelle LXC (Linux Containers) sur lequel Docker s'est d'ailleurs initialement basé. Un conteneur est une boîte qui va être intégralement isolée du système d'exploitation dans laquelle on installera une application ainsi que toutes les librairies nécessaires au bon fonctionnement de celle-ci. Il sera alors facile de distribuer son application qui ne sera pas dépendante de votre système d'exploitation.

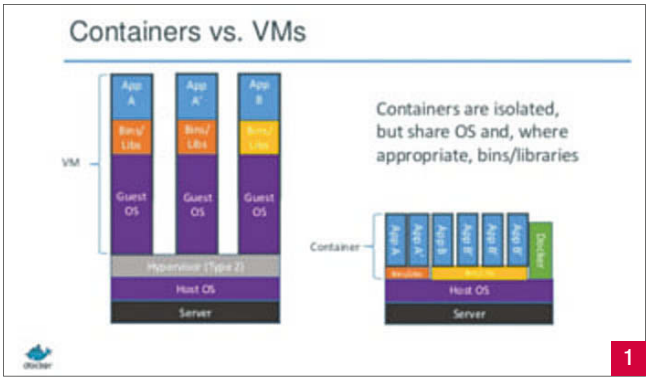
Les avantages de ce concept

Nos applications se complexifient avec le temps et nous sommes contraints d'installer de nombreuses librairies pour faire fonctionner correctement nos applications.

Pour prendre un exemple simple, mon futur site Internet aura plusieurs besoins pour fonctionner : mettre un Apache en serveur Web, un redis en serveur de cache, un MySQL pour avoir une base de données, un PHP 7 avec différentes librairies en interpréteur de code.

Sans conteneur, ceci va avoir des conséquences pour les administrateurs système qui vont devoir refaire la configuration des machines (éventuellement via un infogéreur externe) et tous les développeurs vont devoir installer l'environnement (ce qui se complexifie quand les développeurs ne travaillent pas sur le même système d'exploitation).

En effet il existe des outils comme Vagrant, Chef, Puppet qui aident les administrateurs système mais pas du tout au niveau d'automatisations qu'on aimerait avoir. Les conteneurs vont permettre d'éviter toute cette complexité en proposant une installation commune pour tout le monde. Un gain de temps phénoménal quand on maîtrise un minimum Docker.



1

Quelle différence avec des machines virtuelles ?

Au premier abord, nous pourrions ne pas comprendre ce qu'apportent les conteneurs par rapport aux machines virtuelles.

Voici un schéma proposé par Docker pour expliquer la différence entre des conteneurs et des machines virtuelles : [1]

Les machines virtuelles sont des systèmes d'exploitation qui vont fonctionner par-dessus le système d'exploitation du serveur. Pour faire simple, on a des ordinateurs dans un ordinateur.

Le souci de ce type d'architecture, c'est qu'on est obligé de faire tourner 4 systèmes d'exploitation différents pour mettre 3 machines virtuelles (3 applications différentes) et dédier de la ressource à celles-ci (ram, cpu, disque dur) ; ceci prend énormément de ressources et limite le nombre de machines virtuelles que nous pouvons mettre sur un serveur. Les conteneurs vont permettre de faire tourner les 3 applications directement sur 1 seul système d'exploitation et donc d'alléger considérablement le besoin en ressources.

Du coup au lieu de transférer des systèmes complets comme on est obligé de le faire avec des machines virtuelles pour porter notre application à différents endroits, les conteneurs seront très légers à transférer.

Les conteneurs permettent en conséquence d'être beaucoup plus scalables et peuvent beaucoup plus facilement être déployés.

Avec Docker, les conteneurs se popularisent

Si le système des conteneurs n'est pas nouveau, Docker facilite considérablement son utilisation en proposant de gérer vos conteneurs avec de simples fichiers plats contenant l'ensemble des installations complémentaires à réaliser et en vous offrant des lignes commandes très simples à utiliser pour vous aider dans la gestion de vos conteneurs.
De plus comme Github, Docker propose un espace de partage de conteneurs privés ou publics très utilisés par les différents éditeurs de logiciels. Ces derniers y exposent des conteneurs complets et configurés prêt à l'emploi. Une mine d'or pour travailler plus rapidement.

Fonctionnement de Docker

Docker, c'est un démon qui tournera en permanence pour gérer les conteneurs en cours d'exécution et un client qui propose de nombreuses lignes de commande que nous verrons à la suite de ce dossier.
Afin de faire profiter de cette technologie aux utilisateurs de Windows et macOS, Docker propose également le Boot2Docker (Docker Toolbox) qui se chargera de monter une machine virtuelle sur ces systèmes d'exploitation avec l'ensemble des outils docker nécessaires pour son bon fonctionnement. Docker travaille également sur la possibilité de faire des conteneurs sur Windows dans le futur ; cela qui apportera également un gain considérable pour tous les systèmes spécialisés sous Windows.

Installation

Pour nos premiers pas, je vais prendre une distribution Linux Ubuntu 16.04 qui est l'une des plus populaires dans le monde des développeurs. Si vous utilisez d'autres distributions, vous trouverez facilement les démarches pour installer votre Docker.
Commençons par installer Docker sur notre serveur Ubuntu :

```
sudo apt-get install -y docker.io
```

Sous Windows et macOS, il faudra installer le logiciel Docker Toolbox que vous pouvez trouver sur le site officiel de Docker. Dès que vous aurez lancé votre application Docker Toolbox, vous pourrez utiliser Docker et toutes les fonctions docker de cet article de la même façon que sous Linux

Lançons notre premier conteneur Docker

Sur Linux, il faudra exécuter le démon de Docker (s'il n'est pas déjà lancé) de cette façon :

```
/etc/init.d/docker start
```

Si le démon se lance correctement ce qui devrait être le cas, vous aurez le message suivant : *[OK] Starting docker (via systemctl): docker.service*.
Sous Windows, le démon est déjà démarré au lancement de l'application Docker Toolbox. Si vous avez besoin un jour de connaître la version de votre Docker, voici la commande à faire :

```
docker version
```

Attention : selon la configuration de votre Linux Ubuntu, il sera peut-être nécessaire de mettre sudo avant chaque commande docker ou de revoir la configuration de vos utilisateurs.
Nous commençons par lancer un premier conteneur qui sert uniquement à afficher "Hello world" :

```
docker run hello-world
```

La première fois, Docker vous prévient qu'il ne trouve pas l'image docker du nom de hello-world en local sur la machine ; il va donc la télécharger automatiquement, l'installer et lancer ce premier conteneur.
Si tout va bien, vous verrez un « *Hello from Docker* » apparaître. Ce conteneur est simple et ne sert qu'à faire ce petit test ; cependant vous venez de lancer votre premier conteneur. Afin d'avoir une vision des conteneurs qui sont en cours d'exécution, il suffira d'utiliser cette commande :

```
docker ps
```

Notre conteneur s'est donc arrêté car il n'est pas dans la liste. En ajoutant l'option -a, vous pourrez le retrouver et ainsi voir tous les conteneurs qui ont été lancés même ceux qui se sont arrêtés.

Les images Docker

Une image Docker est une configuration d'un conteneur que nous pouvons récupérer ou que nous pouvons partager. Cet échange de conteneur se fait par le site Docker Hub fortement inspiré de GitHub que nous avons vu brièvement lors de la présentation de Docker.
Vous pourrez y récupérer un grand nombre d'images dont certaines directement proposées par les éditeurs pour vous mâcher une partie du travail. Vous pourrez sans soucis aller chercher une image sur leur site : https://hub.docker.com/ ou directement en ligne de commande.
Nous allons utiliser une commande proposée par Docker pour chercher une image PHP dans le but de la télécharger :

```
docker search php
```

Le premier résultat retourné est parfait pour mon test (image avec comme nom « php »). Nous allons en profiter pour la télécharger afin de l'utiliser :

```
docker pull php
```

Si ce terme vous est familier c'est parce que Docker a copié Git pour la gestion de partage d'images (docker commit, docker pull, docker push…) afin de rendre l'utilisation de Docker Hub la plus intuitive possible. Si vous avez bien suivi ce tutoriel, j'ai normalement 2 images à présent sur mon ordinateur : hello-world et php. Nous allons vérifier cela grâce à la commande suivante qui me permet de connaître le nom de toutes les images disponibles en local :

```
docker images
```

Vous pourrez à présent lancer un conteneur de cette nouvelle image PHP.

Créer sa propre image

Docker nous propose de créer nos images nous-même en utilisant le concept du fichier Dockerfile. Nous allons commencer par créer notre propre image pour bien comprendre les principes de base.
Nous allons créer un dossier dans lequel on va aller créer le fichier Dockerfile ci-dessous :

```
FROM ubuntu:16.04
MAINTAINER Judicael paquet
```

Pour expliquer brièvement, je commence par définir le Linux sur lequel va se créer mon image et l'auteur de celle-ci (autant se faire de l'auto promo). A présent, nous allons builder (construire en mode pro) notre image. Le point en fin de ligne de ma commande n'est pas une erreur car il permet de définir le lieu où se situe mon fichier (soit dans le dossier courant dans notre cas).

```
docker build .
```

Il télécharge puis exécute les deux étapes que je lui ai définies dans le Dockerfile. Si tout va bien, le script se termine sur un Successfully Build XXX. Nous allons à présent rajouter l'installation d'apache 2 en mettant à jour la liste de packages et en installant le package apache2 pour créer une image plus intéressante qu'un Linux vide. Voici le Dockerfile mis à jour :

```
FROM ubuntu:16.04
MAINTAINER Judicael paquet

RUN apt-get update && \
      apt-get install -y apache2
```

La commande RUN nous permettra comme vous devez le deviner de lancer des commandes lors de la création de notre image et le caractère \ permet d'écrire notre commande sur plusieurs lignes.
Je vous laisse builder votre image comme tout à l'heure afin de la mettre à jour. Vous constaterez que Docker n'exécute que l'étape 3 car il a gardé les étapes précédentes en mémoire.
Un jour cela pourra cependant vous poser des soucis donc pensez qu'il existe un paramètre *--no-cache* pour refaire à 100% le build de votre image.
Vous pourrez également nommer votre image avec le paramètre *-t monapache* qui vous permettra quand vous listerez vos images d'avoir écrit "monapache" au lieu de "<none>" dans la colonne REPOSITORY.

```
docker build --no-cache -t monapache .
```

Maintenant que notre image est complète, nous allons pouvoir en faire un conteneur en lançant la commande suivante que nous expliquerons juste en-dessous :

```
sudo docker run -d -p 8000:80 monapache /usr/sbin/apache2ctl -D FOREGROUND
```

-d : le mode détaché permet de ne pas bloquer votre terminal. Dès que le conteneur est lancé, docker vous rend la main
-p : permet de définir le port à appeler pour qu'il atteigne le port 80 sur le conteneur. C'est très pratique car nous pourrons du coup avoir un conteneur par port.
-D FOREGROUND : permet de lancer le processus dans le conteneur et d'attacher la console au processus d'entrée classique.
Nous pouvons à présent tenter de mettre http://localhost:8000 dans notre navigateur. En effet, nous arrivons bien à afficher la page par défaut d'Apache sur Ubuntu. Voici un autre exemple intéressant où nous allons copier des fichiers dans notre Docker en créant un dossier docker/ dans lequel nous allons créer les fichiers suivants.
host-apache2.conf :

```
<VirtualHost *:80>
   ServerAdmin webmaster@localhost
   DocumentRoot /var/www/test
</VirtualHost>
```

Index.html :

```
Hello World
```

On termine en mettant à jour notre Dockerfile avant de builder notre application :

```
FROM ubuntu:16.04
MAINTAINER Judicael paquet

RUN apt-get update && \
      apt-get install -y apache2

ADD docker/host-apache2.conf /etc/apache2/sites-enabled/000-default.conf
ADD docker/index.html /var/www/test/index.html

ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Vous constaterez que nous avons d'ailleurs utilisé la méthode Entrypoint qui permet de lancer une commande au lancement du conteneur. Cela nous évitera de le faire lors de notre docker run que nous ferons comme cela :

```
docker build -t monapache .
docker run -d -p 8002:80 monapache
```

Si vous avez bien suivi l'ensemble de la procédure, vous devriez avoir un joli « Hello World » qui apparaît sur votre navigateur quand vous y appelez http://localhost:8002.
Pour vous éviter de nombreuses recherches, si vous avez besoin de faire un mapping sur deux ports différents vous pourrez rajouter une deuxième fois un –p 3306 :3306 par exemple si vous y installez aussi un MySQL. Nous allons à présent créer un fichier index.html sur la machine « host » avec un mkdir –p /var/www/test2 en y mettant « Hello my friend ». Docker permet également de partager ce fichier de la machine « host » au conteneur en utilisant l'option –v [path_host]:[path_container] comme ceci :

```
Docker run –d –p 8003:80 –v /var/www/test2:/var/www/test monapache
```

Le fichier qu'on avait copié par le Dockerfile n'est plus pris en compte car on a demandé au lancement du conteneur de monter le dossier test2 en tant que dossier test.
Et en testant dans votre navigateur, vous aurez la bonne surprise de voir s'afficher « Hello my friend » et non plus le « Hello World » précédent. Si l'exemple ici est un peu bateau, cela peut-être fort utile pour les personnes qui désirent partager le code de la machine host à un conteneur qui en aurait besoin.

Ajout d'une image MySQL

Maintenant que vous êtes plus à l'aise avec Docker et les Dockerfile, nous allons voir comment faire communiquer les conteneurs entre eux.
Nous allons commencer par faire tourner un nouveau conteneur faisant

fonctionner MySQL de façon isolée. Pour cela, nous allons chercher sur Docker Hub (https://hub.docker.com), un conteneur déjà tout prêt que nous lancerons comme ceci :

```
docker run --name monmysql -e MYSQL_ROOT_PASSWORD=passe -d mysql
/mysql-server:5.7
```

L'option –e permet de définir une variable d'environnement lors du lancement de votre conteneur.
Notre nouveau conteneur est à présent lancé mais on va devoir vérifier cela. Docker propose une commande qui permet de se connecter à son conteneur et de l'utiliser :

```
docker exec –it monmysql mysql –uroot –p
```

Si vous suivez bien, le mot de passe demandé est celui que nous avions indiqué dans la précédente commande soit « passe ». Vous pourrez à présent faire toutes les opérations désirées sur votre MySQL.
Vous pourrez exécuter toutes les commandes que vous désirez avec cette commande docker exec dont la plus importante qui permet de se connecter de la même façon qu'en SSH :

```
docker exec –it monmysql /bin/bash
```

Linker les conteneurs entre eux

Nous allons maintenant voir comment faire communiquer deux conteneurs ensemble car cela pourrait vous aider pour monter vos futures applications avec docker.
Nous allons commencer par supprimer notre fichier /var/www/test2/index.html et nous allons créer un fichier /var/www/test2/index.php qui contiendra ceci :

```
<?php

$dbh = new PDO('mysql:host='.getenv('MYSQL_PORT_3306_TCP_ADDR').';dbname=
mysql', 'root', 'passe');

foreach($dbh->query('SHOW DATABASES') as $row) {
  echo $row[0]. '<br/>';
}
```

Nous allons utiliser le PHP pour faire une simple connexion sur le MySQL de l'autre conteneur et afficher le nom des bases.
Nous allons devoir par contre faire évoluer notre Dockerfile afin de rajouter toutes les librairies nécessaires pour faire la connexion à MySQL :

```
FROM ubuntu:16.04
MAINTAINER judicael paquet

RUN apt-get update && \
  apt-get install -y apache2 php libapache2-mod-php mysql-client php7.0-mysql

RUN sed -i 's/;extension=php_mysqli.dll/extension=php_mysqli.dll/' /etc/php/7.0/
apache2/php.ini
RUN sed -i 's/;extension=php_pdo_mysql.dll/extension=php_pdo_mysql.dll/' /etc/
php/7.0/apache2/php.ini

ADD docker/host-apache2.conf /etc/apache2/sites-enabled/000-default.conf

ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Il faudra par contre dans notre cas donner des droits à notre utilisateur root de MySQL pour pouvoir y accéder de l'extérieur avec la commande suivante :

```
docker exec -it monmysql mysql -uroot -p -e "CREATE USER 'root'@'%' IDENTIFIED
 BY 'passe';GRANT ALL PRIVILEGES ON *.* TO 'root'@'%';FLUSH PRIVILEGES;"
```

Cet article n'a pas pour but de travailler sur la sécurité de vos applications mais il est évident qu'il faudra créer des utilisateurs plus sécurisés pour vos futures applications.
Il ne reste plus qu'à rebuilder l'image monapache pour pouvoir la lancer avec l'option –link qui permettra de lier nos deux conteneurs :

```
docker run -p 8004:80 --name monapachephp --link monmysql:mysql -v /var/
www/test2:/var/www/test -d monapache
```

Normalement vous verrez la liste des bases de données disponibles qui s'afficheront sur votre navigateur Internet quand vous mettrez http://localhost:8004. J'espère que j'ai pu vous permettre de bien comprendre comment fonctionne docker et les Dockerfile qui vous seront très utiles pour créer vos futurs environnements.

Des petites choses à savoir sur le Dockerfile

Voici d'autres instructions que vous pourrez utiliser au sein de vos Dockerfile pour créer vos images :
ENV abc=demo permet de créer une variable d'environnement abc qui aura la valeur demo ;
CMD echo 'Test' permet d'exécuter du code au moment où le conteneur se lance. Vous pourrez surcharger cette commande dans le docker run. Si vous mettez plusieurs CMD au sein de votre Dockerfile, seul le dernier sera pris en compte ;
EXPOSE 3306 permet de définir le ou les ports qui pourront communiquer avec l'extérieur ;
ENTRYPOINT cmd permet d'exécuter une commande au démarrage du conteneur. Comme pour CMD, seul le dernier ENTRYPOINT sera pris en compte. Par contre cette méthode n'est pas surchargeable par docker run ;
USER ubuntu permet de définir l'utilisateur qui sera utilisé pour lancer tous les prochains RUN et ENTRYPOINT du fichier. Par défaut docker utilise le root ;
WORKDIR path permet de définir le répertoire dans lequel s'exécuteront les commandes CMD et ENTRYPOINT ;
ONBUILD est une notion un peu plus avancée qui permet de réaliser des triggers que nous ne verrons pas dans cet article.

Quelques commandes Docker supplémentaires

Voici des commandes qui pourront vous être utiles dans docker :
docker ps –last 10 permet de voir les dix derniers conteneurs lancés ;
docker kill monconteneur permet d'arrêter un conteneur en cours d'exécution. C'est très pratique quand on commence à avoir de nombreux conteneurs de tests ;

docker logs monconteneur permet de revoir les logs du conteneur ;
docker restart monconteneur permet de redémarrer si besoin un conteneur ;
docker rm monconteneur permet de supprimer un conteneur définitivement ;
docker stop monconteneur et **docker start monconteneur** permettent d'arrêter ou de démarrer un conteneur ;
docker top monconteneur permet d'avoir un top du conteneur.
Il existe un grand nombre d'autres commandes mais je n'aurai pas la place de toutes les décrire car docker propose vraiment un grand nombre de choses pour gérer vos conteneurs.

Docker-compose permet de packager

Docker propose un outil très intéressant de gestion de package docker. Cet outil permet de définir et de faire tourner plusieurs conteneurs grâce à un seul fichier de configuration. C'est un peu le apt-get ou le composer de docker pour faire simple. Cet outil vient simplifier la vie aux utilisateurs car comme vous l'avez vu dans cet article, maîtriser Docker n'est pas si simple.
Voici l'installation qu'il faudra faire pour installer docker-compose qui ne fait pas partie du docker.io qu'on avait installé en début d'article :

```
sudo curl -o /usr/local/bin/docker-compose -L "https://github.com/docker/compose/releases/download/1.8.1/docker-compose-$(uname -s)-$(uname -m)"
sudo chmod +x /usr/local/bin/docker-compose
```

Nous allons vérifier que l'installation s'est bien déroulée en tentant de vérifier la version du docker-compose que nous venons d'installer :

```
docker-compose –v
```

Nous allons à présent légèrement modifier notre fichier /var/www/test2/index.php comme ceci afin d'utiliser docker-compose pour recréer l'environnement que nous avions dans cet article avant d'entamer docker-compose :

```
<?php

$dbh = new PDO('mysql:host=db;dbname=mysql', 'root', 'passe');

foreach($dbh->query('SHOW DATABASES') as $row) {
  echo $row[0]. '<br/>';
}
```

Docker compose se base sur un fichier docker-compose.yml qui sera la description de l'ensemble des besoins pour créer une application complète potentiellement constituée de plusieurs conteneurs ; ce fichier est écrit sous le format Yaml.

```
db:
   image: mysql/mysql-server:5.7
   ports:
     - "3306:3306"
   environment:
     - "MYSQL_ROOT_PASSWORD=passe"
     - "MYSQL_USER=your_user"
     - "MYSQL_PASSWORD=your_user_password"
     - "MYSQL_DATABASE=your_database_name"
superapachephp:
   build: ./
   ports:
     - "9010:80"
   volumes:
     - "/var/www/test2:/var/www/test:rw"
   links:
     - "db:db"
   working_dir: "/home/docker"
```

Nous devons lancer la commande suivante pour mettre notre docker compose en action et qu'il fasse lui-même tout le travail pour lancer les différents conteneurs :

```
docker-compose up -d
```

En testant http://localhost:9010 dans votre navigateur vous aurez la liste des bases existantes dans votre base de données MySQL. N'oubliez pas la manipulation à faire sur les utilisateurs sous MySQL si rien ne s'affiche.
Vous pourrez arrêter tous les conteneurs de votre docker-compose automatiquement en une seule commande ce qui est très pratique quand on travaille sur des projets assez gros :

```
sudo docker-compose stop
```

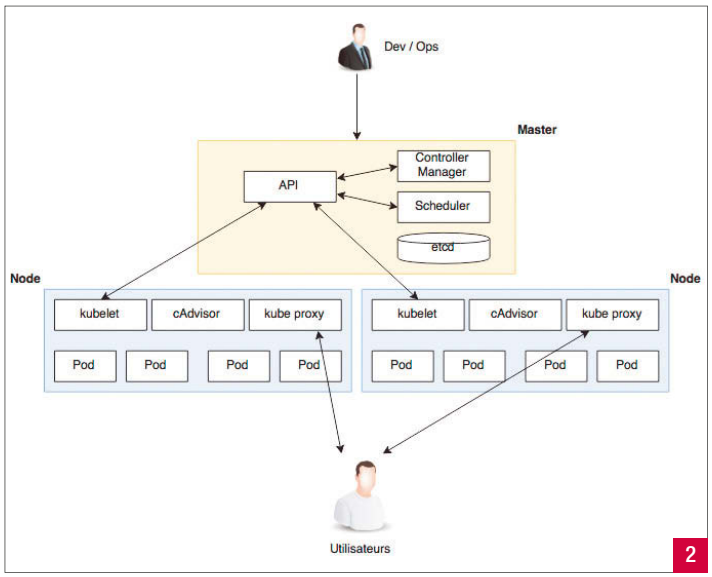
L'utilisation de docker-compose n'est pas toujours simple mais la communauté de Docker est tellement active que vous trouverez toujours une solution à vos différents problèmes.

Docker au service du déploiement

Docker et docker-compose permettent de préparer vos déploiements dans tous types d'environnements : dev, recette, pré-production, production.
Vous verrez de plus en plus dans les projets récents partagés sur Github des fichiers Dockerfile et docker-compose.yml à la racine des projets pour monter les environnements complets de l'application à la suite du git clone du projet. Cette pratique est très utile pour tester rapidement l'application proposée voire pour travailler dessus.

Orchestrer ses conteneurs avec Kubernetes

Kubernetes est une solution d'orchestration et de gestion de clusters de conteneurs open-source écrit en Go qui a été créé par Google.
Le projet est partagé sur Github et possède une communauté de plus de 1000 contributeurs.
Si l'utilisateur classique va simplement choisir un type d'instance, un OS et une région d'hébergement, l'hébergeur va, lui, avoir un véritable orchestrateur pour répondre à votre demande en provisionnant des VM sur des serveurs physiques ou en adaptant des VM déjà existantes.
SI Docker est un outil idéal dans la gestion de conteneurs, Kubernetes propose beaucoup plus que la gestion de conteneurs. [2]
L'idée de base de Kubernetes est de pouvoir déployer un ou plusieurs conteneurs dockers gérés par un master. On appellera des « Pods » (voir image sur l'architecture de Kubernetes), des unités de déploiement de Kubernetes qui peuvent être créées ou supprimées par l'intermédiaire de commandes ; c'est la notion la plus granulaire sur un cluster Kubernetes.



Les nodes (appelés aussi workers ou minions) sont des machines (ou machines virtuelles) où des conteneurs sont déployés.
Kubernetes fonctionne avec les différents hébergeurs Cloud comme AWS, Azure et Google Cloud.
Kubernetes n'est pas la seule solution d'orchestration mais est de loin la plus répandue. Vous pourrez également regarder du côté de docker-swam, Mezos ou Fleet.

Installation de kubernetes

Pour installer Kubernetes sur notre Ubuntu, nous allons appliquer cette liste de commandes :

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo chmod 777 /etc/apt/sources.list.d
cat <<EOF > /etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl kubernetes-cni
```

Kubernetes et Google Container Engine

Pour travailler avec Google Container Engine, nous devons installer l'outil proposé par Google comme ceci :

```
export CLOUD_SDK_REPO="cloud-sdk-$(lsb_release -c -s)"
echo "deb https://packages.cloud.google.com/apt $CLOUD_SDK_REPO main" |
sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo apt-get update && sudo apt-get install google-cloud-sdk
```

Il vous faudra également créer un compte gratuit sur Google Container Engine qui vous permettra de réaliser jusqu'à 300 de tests.
Dès que votre inscription est terminée, pensez à créer un nouveau projet vide sur l'interface. Ce projet va regrouper l'ensemble de vos clusters donc le cluster que nous allons installer.
Maintenant que gcloud est installé, nous allons le configurer comme ceci :

```
gcloud init
gcloud config set compute/zone europe-west1-b
gcloud container clusters create cluster-1
gcloud auth application-default login
gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project
stable-synapse-156722
```

Le paramètre à mettre en projet sera l'id associée au projet que vous avez créé sur l'interface de Google. Vous le trouverez dans la page d'accueil de Google Cloud Platform au sein du bloc projet.
Nous pourrons vérifier l'état des clusters avec la méthode suivante :

```
Kubectl cluster-info
```

Nous pourrons regarder à tout moment l'état de nos pods et de nos services avec les commandes suivantes :

```
kubectl get pod
kubectl get service
```

On va ainsi créer des pods et le service associé d'une nouvelle image que nous allons récupérer :

```
docker pull eboraas/apache-php
kubectl run apache --image=eboraas/apache-php --replicas=2 --port=80 --
expose --service-overrides='{ "spec": { "type": "LoadBalancer" } }'
```

En affichant vos pods et vos services, vous aurez au bout de quelques minutes, une IP publique pour votre application :

```
NAME                READY   STATUS    RESTARTS   AGE
apache-3670720257-1ll4t  1/1     Running   0          3m
apache-3670720257-1v7af  1/1     Running   0          3m

NAME         CLUSTER-IP    EXTERNAL-IP     PORT(S)      AGE
apache       10.55.254.17  104.198.150.151  80:32130/TCP  3m
kubernetes   10.55.240.1   <none>          443/TCP      14m
```

En vérifiant la nouvelle adresse dans le navigateur, vous pourrez voir la belle page d'accueil d'apache 2 qui nous confirme que nous venons de déployer notre premier projet qui contient un load balancer et deux enfants sur Google Cloud Platform.
On pourra supprimer notre déploiement avec les deux méthodes suivantes car il ne faut pas oublier qu'on a créé un service et des pods :

```
kubectl delete deployment my-nginx
kubectl delete svc my-nginx
```

CONCLUSION

Docker et Kubernetes sont devenus des outils indispensables dans l'univers devops mais aussi dans le monde de l'open source.
Si cet article vous permet de vous lancer avec ce type d'outils, sachez qu'ils ne sont pas évidents à maîtriser. N'hésitez pas à parcourir le net qui propose de nombreux articles sur les sujets qui viendront compléter le contenu de cet article. •

Souriez ! Vous êtes analysé(e)s !

Les technologies cognitives envahissent notre quotidien

On parle beaucoup de technologies et d'API cognitives, sans toujours savoir ce qu'il se cache réellement derrière ce terme. Il s'agit de services, de SDK, d'API, de technologies qui vont permettre de capturer, d'analyser, d'interpréter des comportements, des émotions, des environnements, de la parole, des langues, des objets, etc. Ces services peuvent être intégrés à des apps, des sites, du matériel et systèmes embarqués.

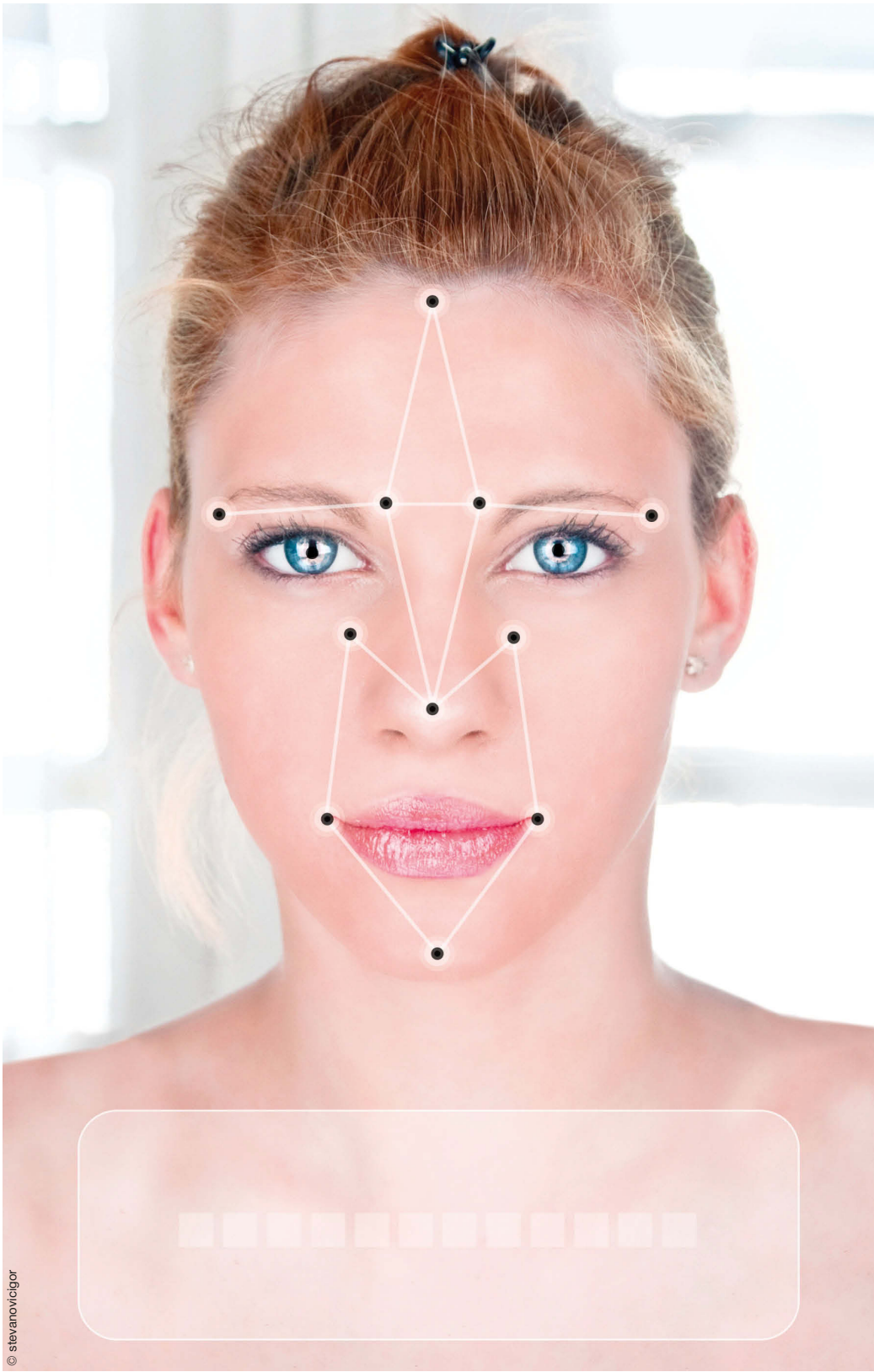
Les domaines d'action sont très larges :

- *Analyses d'images fixes ou live ;*
- *Analyses de textes, de langues pour la traduction à la volée ;*
- *Reconnaissance et analyse des émotions, des visages, des personnes, des mouvements.*

Ces services étendent, ou complètent le Machine / Deep Learning, et font partie de la nébuleuse de l'intelligence artificielle. Ils peuvent servir à beaucoup de choses : sécurité, jeux, réseaux sociaux, préventions de risques, traductions de contenus, rapprochement des contenus, dans le commerce, etc.

Ces services ouvrent des perspectives et opportunités très larges pour les développeurs, les éditeurs, industriels, les utilisateurs. Ce qui peut être pertinent est la traduction à la volée des vidéos et des textes, d'une conversation. Nous ne sommes pas encore à créer un traducteur universel en temps réel, mais on commence à s'en rapprocher même si le vocal est toujours très difficile à traduire à la volée, notamment avec les variétés de sons, des langues, des accents, etc. Mais la retranscription d'un flux live a fait énormément de progrès en quelques années.

François Tonic



© stevanovicigor

Introduction au **Machine Learning** sous Android avec l'API Mobile Vision de Google

• Sylvain Saurel
sylvain.saurel@gmail.com
Développeur Android
https://www.ssaurel.com

Les débats sur l'Intelligence Artificielle ont atteint la sphère publique et tout le monde a désormais entendu parler des futures prouesses que nous réserveront les machines dans les années à venir. Champ d'étude de l'Intelligence Artificielle, le Machine Learning est désormais un domaine à la portée de tous les développeurs grâce à des APIs dédiées proposées par certains géants du Web. Dans cet article, nous vous proposons de découvrir l'API Mobile Vision de Google et sa mise en oeuvre sous Android.

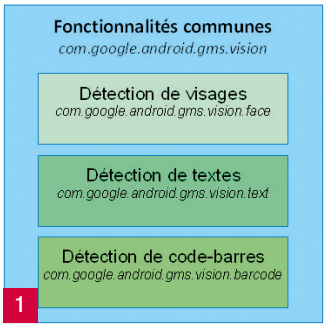
Champ d'étude de l'Intelligence Artificielle, le Machine Learning, ou apprentissage automatique pour nous autres héritiers de la langue de Molière, confère notamment la possibilité à des machines de remplir des tâches difficiles ou problématiques grâce à l'implémentation d'algorithmes spécifiques dans des programmes informatiques. Parmi ces tâches difficiles, nous allons plus particulièrement nous intéresser dans cet article à la détection de visages au sein d'images, à la détection de textes et enfin à la détection de codes-barres. Pour adresser cette problématique et permettre au plus grand nombre de développeurs d'intégrer ce type de fonctionnalités au sein de leurs programmes, Google propose l'API Mobile Vision. Disponible sur Android, mais également sous iOS, elle offre aux développeurs un framework simple et puissant pour trouver des objets au sein d'images et de vidéos. Ainsi, elle permet de détecter des visages sur des images mais également les différentes parties présentes sur ces visages comme par exemple les yeux, le nez ou la bouche . En sus, elle propose la détection de textes et de codes-barres.

Architecture

L'API Mobile Vision propose un framework découpé en 4 parties majeures (figure 1) :

- Fonctionnalités communes regroupées au sein du package com.google.android.gms.vision ;
- Fonctionnalités liées à la détection de visages présentes au sein du package com.google.android.gms.vision.face ;
- Fonctionnalités liées à la détection de textes au sein du package com.google.android.vision.text ;
- Fonctionnalités liées à la détection de codes-barres regroupées au sein du package com.google.android.gms.vision.barcode.

Les différentes fonctionnalités offertes par l'API Mobile Vision pouvant être utilisées ensemble ou de manière séparée.



Architecture de l'API Mobile Vision

Installation

Afin d'utiliser l'API au sein d'une application Android, il est nécessaire de cibler des terminaux fonctionnant au moins sous Android Gingerbread 2.3 ce qui est assez aisé puisque plus de 95% des appareils Android tournent sur des versions supérieures à la 2.3 selon les dernières statistiques publiées par Google en Janvier 2017. La seconde étape consiste à ajouter la dépendance à l'API au sein du fichier de build Gradle de votre projet d'application Android. L'API étant distribuée au sein du SDK des Google Play Services, il faudra ajouter la dépendance suivante :

```
compile 'com.google.android.gms:play-services-vision:10.0.1'
```

Pour activer la bibliothèque Mobile Vision, il faudra enfin ajouter une métadonnée au sein du manifest Android de l'application précisant le type de détection que l'on souhaite réaliser et donc les composants Mobile Vision à utiliser. Pour la détection de visages, la ligne suivante est à ajouter au sein de la balise application :

```
<meta-data android:name="com.google.android.gms.vision.DEPENDENCIES"
  android:value="face"/>
```

Notons que pour faire de la reconnaissance de textes, on appliquera la valeur "ocr" et pour la détection de codes-barres on appliquera la valeur "barcode". Pour utiliser ensemble ces fonctionnalités de détection au sein d'une même application, il suffira alors d'ajouter ces valeurs séparées par des virgules comme suit :

```
<meta-data android:name="com.google.android.gms.vision.DEPENDENCIES"
  android:value="barcode, face, ocr"/>
```

Détection de visages

Pour commencer notre exploration de l'API Mobile Vision, nous allons réaliser une application détectant un visage sur une image, détaillant les points d'intérêts de ce visage tels que les yeux, la bouche ou le nez et permettant enfin de remplacer une partie des points d'intérêts du visage par des éléments personnalisés. La première étape va consister à charger l'image d'entrée au sein d'un Bitmap. Pour travailler sur cette image, nous allons définir une image temporaire via un second Bitmap qui aura les mêmes dimensions que l'image en entrée. Cette initialisation des Bitmaps est réalisée comme suit :

```
BitmapFactory.Options bitmapOptions = new BitmapFactory.Options();
bitmapOptions.inMutable = true;

defaultBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.bill_gates,
      bitmapOptions);
temporaryBitmap = Bitmap.createBitmap(defaultBitmap.getWidth(), defaultBitmap
      .getHeight(), Bitmap.Config.RGB_565);
```

Pour dessiner les éléments autour du visage tels que les points d'intérêts détectés, nous utiliserons un objet Canvas prenant en entrée le Bitmap temporaire créé précédemment. L'étape suivante consiste à initialiser l'ob-

jet FaceDetector en charge de réaliser la détection de visages. La construction de cet objet se réalise à l'aide du classique pattern Builder et nous précisons que nous souhaitons obtenir tous les points d'intérêts détectés par l'API via le passage en entrée de la constante FaceDetector.ALL_LANDMARKS.

Une fois l'objet FaceDetector construit, les dépendances supplémentaires éventuellement nécessaires sont téléchargées. Pour vérifier que la détection est prête à être réalisée, il faut appeler la méthode isOperational() de l'objet FaceDetector. Lorsque la méthode renvoie true, nous pouvons commencer la détection de visages. Ainsi, nous créons un élément Frame proposé par l'API Mobile Vision au sein duquel nous passons l'image à traiter. La suite va consister à appeler la méthode detect() du FaceDetector avec cet objet Frame en entrée. En retour, on obtient une liste de visages modélisés au sein de la classe Face.

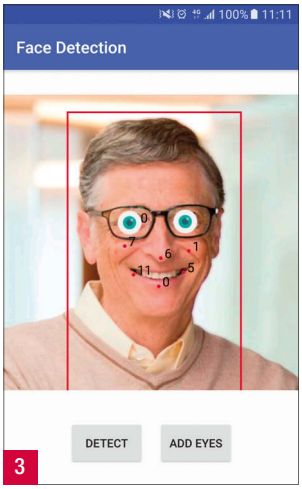
Pour chaque visage détecté, nous allons l'entourer avec un rectangle dessiné à l'aide de l'API Canvas du SDK Android. Ensuite, nous récupérons les points d'intérêts de chaque visage. Chaque point d'intérêt est alors dessiné à l'aide d'un cercle et numéroté suivant la numérotation proposée par l'API. Il ne reste alors plus qu'à afficher le Bitmap sur lequel nous avons travaillé au sein d'un composant visuel ImageView, puis à appeler la méthode release() du FaceDetector pour libérer les ressources qui ont été allouées. En prenant en entrée une photo du bien connu Bill Gates, on obtient le résultat proposé à la figure 2.

Pour aller légèrement plus loin, nous pouvons même choisir de dessiner des éléments personnalisés sur des parties spécifiques du ou des visages détectés. Ici, nous allons rajouter des yeux un peu plus fantaisistes à notre très cher Bill Gates. Pour ce faire, il suffit de récupérer la position des yeux via la liste des points d'intérêts retournés pour un visage donné. Chaque point d'intérêt détecté ayant un type spécifique, il suffira de traiter uniquement les points d'intérêts ayant pour type Landmark.LEFT_EYE ou Landmark.RIGHT_EYE correspondant respectivement à l'oeil gauche et à l'oeil droit. Enfin, il reste à dessiner les yeux fantaisistes à ces positions données. Le résultat est proposé à la figure 3.

Le code réalisé pour la détection de visages et l'ajout d'éléments fantaisistes sur les yeux est présenté dans ce qui suit :

2

Détection de visages et des points d'intérêts associés



3

Ajout d'éléments personnalisés sur le visage détecté

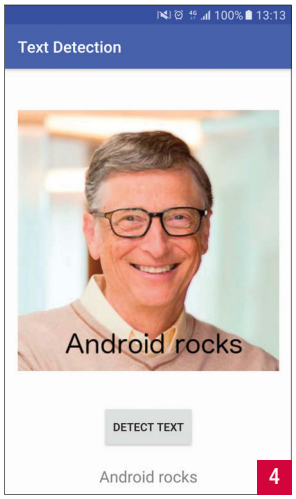
```
public class MainActivity extends AppCompatActivity {

 private ImageView imageView;
 private Paint txtPaint;
 private Paint roundPaint;
 private Paint rectPaint;
 private Bitmap defaultBitmap;
 private Bitmap temporaryBitmap;
 private Bitmap eyePatchBitmap;
 private Canvas canvas;
 private boolean withEyes;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  imageView = (ImageView) findViewById(R.id.image_view);
  initPainters();
  initBitmaps();
 }

 public void processImage(View view) {
  withEyes = false;
  process();
 }

 public void processImageWithEyes(View view) {
  withEyes = true;
  process();
 }


 public void process() {
  canvas = new Canvas(temporaryBitmap);
  canvas.drawBitmap(defaultBitmap, 0, 0, null);

  FaceDetector faceDetector = new FaceDetector.Builder(this)
  .setTrackingEnabled(false)
  .setLandmarkType(FaceDetector.ALL_LANDMARKS)
  .build();

  if (!faceDetector.isOperational()) {
   new AlertDialog.Builder(this)
   .setMessage("Face Detector couldn't be set up on your device")
   .show();
  } else {
   Frame frame = new Frame.Builder().setBitmap(defaultBitmap).build();
   SparseArray<Face> sparseArray = faceDetector.detect(frame);
   detectFaces(sparseArray);
   imageView.setImageDrawable(new BitmapDrawable(getResources(), temporaryBitmap));
   faceDetector.release();
  }
 }

 private void initBitmaps() {
  BitmapFactory.Options bitmapOptions = new BitmapFactory.Options();
  bitmapOptions.inMutable = true;

  defaultBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.bill_gates,
  bitmapOptions);
  temporaryBitmap = Bitmap.createBitmap(defaultBitmap.getWidth(), defaultBitmap
  .getHeight(), Bitmap.Config.RGB_565);
  eyePatchBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.eye,
  bitmapOptions);
```

```
}

private void initPainters() {
  roundPaint = new Paint();
  roundPaint.setColor(Color.RED);
  roundPaint.setStyle(Paint.Style.FILL);

  rectPaint = new Paint();
  rectPaint.setStrokeWidth(10);
  rectPaint.setColor(Color.RED);
  rectPaint.setStyle(Paint.Style.STROKE);

  txtPaint = new Paint();
  txtPaint.setColor(Color.BLACK);
  txtPaint.setStyle(Paint.Style.FILL);
}

private void detectFaces(SparseArray<Face> sparseArray) {
  for (int i = 0; i < sparseArray.size(); i++) {
    Face face = sparseArray.valueAt(i);
    float left = face.getPosition().x;
    float top = face.getPosition().y;
    float right = left + face.getWidth();
    float bottom = right + face.getHeight();

    RectF rectF = new RectF(left, top, right, bottom);
    canvas.drawRect(rectF, rectPaint);

    detectLandmarks(face);
  }
}

private void detectLandmarks(Face face) {
  for (Landmark landmark : face.getLandmarks()) {
    int cx = (int) (landmark.getPosition().x);
    int cy = (int) (landmark.getPosition().y);
    canvas.drawCircle(cx, cy, 10, roundPaint);

    drawLandmarkType(landmark.getType(), cx, cy);

    if (withEyes) {
      drawEyeBitmap(landmark.getType(), cx, cy);
    }
  }
}

private void drawLandmarkType(int landmarkType, float cx, float cy) {
  String type = String.valueOf(landmarkType);
  txtPaint.setTextSize(70);
  canvas.drawText(type, cx + 20, cy, txtPaint);
}

private void drawEyeBitmap(int landmarkType, float cx, float cy) {
  if (landmarkType == Landmark.LEFT_EYE  ||  landmarkType == Landmark.
RIGHT_EYE) {
    int scaledWidth = eyePatchBitmap.getScaledWidth(canvas);
    int scaledHeight = eyePatchBitmap.getScaledHeight(canvas);
    canvas.drawBitmap(eyePatchBitmap, cx - (scaledWidth / 2), cy - (scaledHeight / 2), null);
  }
 }
}
```

Détection de textes

Notre second exemple va concerner la détection de textes sur une image puisque la solution Mobile Vision offre également des fonctionnalités de reconnaissance de caractères. En entrée, nous allons considérer une image de Bill Gates avec un texte écrit en bas de l'image. La première étape va être de charger l'image contenant le texte à décoder au sein d'un Bitmap. La suite consiste à initialiser un objet TextRecognizer implémentant la fonctionnalité de reconnaissance de textes. On vérifie également que toutes les bibliothèques nécessaires au bon fonctionnement sont à disposition via un appel à la méthode isOperational() sur l'objet construit. Si le TextRecognizer est prêt à fonctionner, on va créer un objet Frame au sein duquel on passe le Bitmap contenant l'image d'entrée à traiter.

Il faut ensuite appeler la méthode detect() du TextRecognizer avec en entrée l'objet Frame nouvellement créé. En retour, l'API nous renvoie une liste d'objets TextBlock qui sont des blocs de textes reconnus par le framework Mobile Vision sur l'image en cours de traitement. En itérant sur cette liste, on peut ainsi reconstruire les différentes lignes du texte reconnu sur l'image, et afficher le résultat final au sein d'un composant TextView (figure 4).



4
Détection de textes

Le code nécessaire pour implémenter une reconnaissance de caractères simple sous Android a l'allure suivante :

```
public class MainActivity extends AppCompatActivity {

  ImageView imageView;
  TextView detectedTextView;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    imageView = (ImageView) findViewById(R.id.image);
    detectedTextView = (TextView) findViewById(R.id.detected_text);
  }

  public void detectText(View view) {
    Bitmap textBitmap = BitmapFactory.decodeResource(getResources(), R.drawable
.bill_gates);
    TextRecognizer textRecognizer = new TextRecognizer.Builder(this).build();

    if (!textRecognizer.isOperational()) {
      new AlertDialog.Builder(this)
      .setMessage("Text recognizer couldn't be set up on your device")
      .show();
      return;
    }
```

```
  Frame frame = new Frame.Builder().setBitmap(textBitmap).build();
  SparseArray<TextBlock> texts = textRecognizer.detect(frame);
  StringBuilder builder = new StringBuilder();

  for (int i = 0; i < texts.size(); ++i) {
   TextBlock item = texts.valueAt(i);

   if (item != null && item.getValue() != null) {
    builder.append(item.getValue()).append("\n");
   }
  }

  detectedTextView.setText(builder.toString());
  textRecognizer.release();
 }
}
```

Détection de Code-barres

Pour notre dernier exemple d'utilisation de l'API Mobile Vision, nous allons nous concentrer sur la détection et la lecture de code-barres au sein d'une image. Pour cela, nous initialisons un objet BarcodeDetector et précisons durant la construction que nous souhaitons reconnaître tous les types de codes-barres en appliquant la constante Barcode.ALL_FORMATS en entrée. Comme vu précédemment, on prend bien garde de vérifier que l'objet est opérationnel via un appel à la méthode isOperational(). Si c'est le cas, on construit un objet Frame au sein duquel on passe en entrée l'image à traiter chargée au sein d'un Bitmap. Il ne reste alors plus qu'à appeler la méthode detect() de l'objet BarcodeDetector avec l'instance d'objet Frame que l'ont vient de créer. Comme pour la détection de visages et de textes, l'API nous renvoie une liste d'objets correspondant à la détection réalisée. Ici, il s'agit d'objets Barcode qui sont associés aux codes-barres détectés. Nous itérons sur les codes-barres détectés et pour chacun d'entre eux nous récupérons la valeur qui y est codée via l'appel à la propriété displayValue. Enfin, il ne reste plus qu'à afficher les valeurs décodées au sein d'un composant TextView et à libérer les ressources allouées à l'objet BarcodeDetector en appelant sa méthode release(). Le résultat de l'exécution de l'application implémentant cette détection est présenté à la figure 5 : Le code de l'application permettant la détection de code-barres a l'allure suivante :

Détection de code-barres

```
public class MainActivity extends AppCompatActivity {

 private Bitmap barcodeBitmap;
 private TextView textView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  ImageView imageView = (ImageView) findViewById(R.id.image_view);
  textView = (TextView) findViewById(R.id.text_view);
  barcodeBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.qrcode);

  imageView.setImageBitmap(barcodeBitmap);
 }

 public void processBarcode(View view) {
  BarcodeDetector barcodeDetector = new BarcodeDetector.Builder(this)
  .setBarcodeFormats(Barcode.ALL_FORMATS)
  .build();

  if (!barcodeDetector.isOperational()) {
   new AlertDialog.Builder(this)
   .setMessage("Barcode detector couldn't be set up on your device")
   .show();
   return;
  }

  Frame frame = new Frame.Builder().setBitmap(barcodeBitmap).build();
  SparseArray<Barcode> barcodes = barcodeDetector.detect(frame);

  int size = barcodes.size();

  if (size == 0) {
   textView.setText("No information available");
  } else {
   StringBuilder barcodeValue = new StringBuilder();

   for (int i = 0; i < size; i++) {
    barcodeValue.append(barcodes.valueAt(i).displayValue).append("\n");
   }

   textView.setText(barcodeValue.toString());
  }

  barcodeDetector.release();
 }
}
```

CONCLUSION

Simple d'usage et puissante à la fois, l'API Mobile Vision de Google met à disposition des développeurs d'applications mobiles Android (iOS également) un framework complet permettant d'offrir des fonctionnalités de Machine Learning à leurs utilisateurs.

Dans cet article, nous nous sommes limités à travailler sur des images fixes mais il est également tout à fait possible de travailler directement sur un flux vidéo capturé depuis la Caméra d'un appareil mobile. L'API permet alors de réaliser du suivi de visages puisqu'elle permet de réaliser des détections sur des séquences vidéo. La possibilité d'accéder précisément aux différents points d'intérêts détectés sur les visages tout comme celle d'obtenir une classification sur l'expression d'un visage (yeux ouverts ou fermés / visage souriant ou non) ouvre des perspectives aux développeurs en termes de cas d'usages innovants qu'ils pourront proposer à leurs utilisateurs. Pouvoir associer cette détection de visages à de la reconnaissance de caractères s'avère également très intéressant. Il ne reste plus aux développeurs qu'à s'approprier pleinement l'API Mobile Vision pour en tirer partie au sein de leurs applications ce qui ne devrait pas tarder dans les mois à venir. •

Cognitive Services de Microsoft

• Matthieu COSTABELLO, *Développeur*
• Alexandre JACOB, *Project Lead*
• Michaël FERY, *Tech Lead IoT*

Infinite Square
http://blogs.infinitesquare.com/

Dans un monde de données, le traitement des images et de la voix sont devenus primordiaux. De nombreux algorithmes existent mais face à la masse de données et aux contraintes de performance, le Machine Learning est devenu une clef pour maîtriser ces sujets. Conscient de ces enjeux, Microsoft a depuis longtemps dédié une partie de ses équipes pour tenter d'adresser les problématiques de Machine Learning et les applications qui en découlent.

Quels sont ces services ?

Parmi les API proposées par Microsoft, des sujets bien divers sont couverts mais voici les principaux :

Vision : La détection d'image n'a jamais été aussi fine, puisque l'API de Vision permet de détecter des visages, de reconnaître les éléments sur une photographie mais également d'y détecter les sentiments de la personne photographiée.

Speech : Ecouter et reconnaître les phrases énoncées ainsi que les personnes qui parlent

Language : Utiliser un langage naturel pour déterminer les demandes des utilisateurs

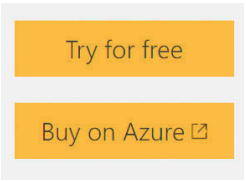
Knowledge : Proposer des bases de connaissances basées sur les données Web ou des données propres agrégées dans le service

Search : Basé sur les API de Bing, ce service propose de rechercher parmi les milliards de pages web, images, vidéos et actualités.

Comment s'en servir ?

Ces services sont disponibles via des APIs et SDKs fournis par Microsoft. La première étape consiste à s'inscrire sur le site dédié : https://www.microsoft.com/cognitive-services/en-us/sign-up

S'il existe des paliers d'utilisation payants, tester les versions gratuites ne nécessite qu'un compte Microsoft, GitHub ou LinkedIn. Il n'y a donc pas de barrière à l'entrée pour commencer à tester et éprouver les APIs. Pourquoi se priver alors ?



Prise en main du service Speech

Le service Speaker Recognition se divise en 2 catégories :

• ***Speaker vérification :***

La vérification permet via une phrase préétablie de vérifier et d'authentifier une personne, un peu comme Windows Hello fait avec les visages.

• ***Speaker Identification :***

L'identification permet de reconnaître et d'identifier la voix d'une personne indépendamment des phrases qu'elle prononce.

Pour commencer nous nous rendons à l'adresse https://www.microsoft.com/cognitive-services/en-US/subscriptions?mode=NewTrials où l'on peut s'inscrire au service désiré, ici Speaker Recognition – Preview [1]

Une fois l'inscription validée, on peut récupérer la clé qui va nous permettre d'utiliser le service. [2]

En mode gratuit nous sommes limités à 20 appels à chaque API par minute et à 10k appels par mois.

En mode premium cette limite passe à 5 appels par seconde pour un coût d'utilisation de 10 les 50k appels aux API d'identification et de 5 les 50k appels aux API de vérification. Ces prix sont dégressifs en fonction de l'utilisation.

Du code, du code !

Passons dans Visual Studio. Nous commençons par télécharger le package NuGet Microsoft.ProjectOxford.SpeakerRecognition fourni par Microsoft. [3]

Le SDK possède 2 classes très importantes :

• SpeakerIdentificationServiceClient, chargée de l'identification
• SpeakerVerificationServiceClient, chargée de la vérification

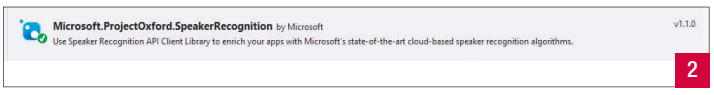
Ici nous allons tester l'identification.

Le service se découpe en 3 étapes distinctes :

Etape 1 : Création d'un profil (identifié par un GUID) ;
Etape 2 : Entrainement d'un profil avec des échantillons d'une voix ;
Etape 3 : Identification du profil.

L'utilisation est très simple, on commence par instancier notre SpeakerIdentificationServiceClient avec la clé récupérée lors de l'inscription sur le site :

```
SpeakerIdentificationServiceClient sis = new
SpeakerIdentificationServiceClient("{YOUR KEY}");
```



Création du profil

Ensuite il faut créer un profil pour chaque personne dont l'identification est souhaitée.
Pour cela, nous appellerons 3 fois la méthode **CreateProfileAsync** pour créer un jeu de données de 3 personnes pouvant être identifiées.

```
var Profils = new List<Guid>();

        for (int i = 0; i < 3; i++)
        {
          //pour l'instant seul le paramètre "en-US" est accepté, mais l'api gère
assez bien le français.
          var profileResponse = await sis.CreateProfileAsync("en-us");

          Profils.Add(profileResponse.ProfileId);
    }
```

Cette fonction retourne un GUID qui sera l'identifiant de notre speaker.

Entrainement d'un profil avec des échantillons d'une voix

Il va maintenant falloir entraîner ces profils avec des streams audio.
Le service demande au moins 15 secondes d'échantillons de voix pour réussir à identifier une personne.
De plus, comme précisé dans la documentation, il est important de noter que les flux audio envoyés à l'API doivent avoir les caractéristiques suivantes :

Container	WAV
Encoding	PCM
Rate	16K
Sample Format	16 bit
Channels	Mono

Il nous suffit maintenant d'envoyer le stream audio et l'id du speaker à la fonction d'entraînement EnrollAsync

```
var operationLocation = sis.EnrollAsync(streamAudio, Profils[x])
```

Cette méthode nous renvoie un objet contenant les informations nécessaires pour connaitre le statut de l'opération.
La fonction suivante nous renseigne sur l'état de notre opération :

```
var statut = await _sis.CheckEnrollmentStatusAsync(operationLocation);
```

Le statut retourné sera « **Running** » pendant un moment puis passera à « **Succeeded** » en cas de réussite ou à « **Failed** » en cas d'échec. Sachez que dans ce cas-là un message d'erreur se trouve dans la variable Message. Pour savoir si la reconnaissance de voix est opérationnelle sur un speaker on appelle le code suivant :

```
        var profil = await sis.GetProfileAsync(Profils[x]);

        EnrollmentStatus = profil.EnrollmentStatus.ToString();
      RemainingEnrollmentSpeechTime = profil.RemainingEnrollmentSpeechSeconds;
```

Le statut nous indique si cette personne peut être utilisée pour l'identification. Tant que le statut est à **Enrolling** ou **Training,** ce GUID doit encore être entraîné. En revanche, si le statut est **Enrolled** alors l'identification peut commencer.

Nous avons par ailleurs la propriété **RemainingEnrollmentSpeechSeconds** qui nous donne une indication du temps de voix encore nécessaire à l'entraînement de la personne.

Identification du profil

Tout étant configuré nous pouvons désormais passer à l'identification à proprement parler.
Encore une fois nous nous contentons d'envoyer un stream contenant la voix de la personne à identifier et une liste de GUID de nos profils créés et entraînés précédemment.

```
    var operationLocation = await _sis.IdentifyAsync(stream.AsStreamForRead(),
Profils.ToArray());
```

Puis de la même manière que pour vérifier le statut de l'opération entraînement :

```
var statut = await _sis.CheckIdentificationStatusAsync(result);

        CurrentOperationStatut = statut.Status.ToString();
        CurrentOperationMessage = statut.Message;
```

Un fois l'opération terminée on récupère le résultat comme suit :

```
        if (statut.ProcessingResult != null)
        {
          CurrentOperationMessage = $" GUID identifié : {statut.ProcessingResult
.IdentifiedProfileId.ToString()} avec un degré de confiance {statut.ProcessingResult.
Confidence.ToString()}";
        }
```

Nous obtenons finalement le GUID de la personne identifiée dans le stream envoyé précédemment avec un certain niveau de confiance **Low**, **Normal** ou **High**.
Voilà donc comment avec quelques lignes de code nous avons accès au service de reconnaissance de speaker fourni par Microsoft.

Un cas d'usage concret avec Vision

Nous sommes en droit de nous demander si ces Cognitives Services sont réellement utilisables en Production ou s'il s'agit de simples preuves de concept. Voyons ensemble un cas d'usage concret.

Problématique

Un client nous a demandé d'imprimer la photo du visiteur sur des badges pour un salon évènementiel
Cependant le visiteur s'inscrivait en ligne via un service tiers, qui ne faisait aucun contrôle sur la photo demandée.
Donc c'est tout naturellement que nous avons reçu un package de 2500 photos dont une bonne partie ne représentaient pas le visage d'une personne, par exemple il y avait un koala, des photos de groupe, des avatars, … Et d'autres dont le sujet n'était pas du tout centré dans l'image, alors que nous devions les imprimer dans un cadre de 350 par 350 pixels.
Plusieurs problématiques en découlaient :

- Quelles sont les photos qui ne comportent aucun visage ?
- Quelles sont les photos qui comportent plus d'un visage ?
- Quelles sont les photos de taille ou de qualité insatisfaisante ?

• Comment recadrer les photos pour centrer sur le visage ?

Dans les 3 premiers cas l'idée était de lister les utilisateurs ne répondant pas aux critères pour leur redemander le téléchargement de leur photo.

L'API Face

L'utilisation de l'api *Face* des Cognitive Services nous a paru la meilleure façon de répondre à ces 4 questions. En effet, cette dernière permet à partir d'une image de donner des informations sur des visages :

- Nombre de visages
- Position de chacun d'eux (coordonnées du coin en haut à gauche + longueur et largeur)
- Position et taille des yeux, des sourcils, du nez, de la bouche
- Pilosité faciale (moustache, barbe, …)
- Age de la personne, et son sexe
- Si elle sourit
- Si elle porte des lunettes

Il est également possible d'envoyer deux photos pour comparer s'il y a des personnes similaires sur les deux.

Vous pouvez essayer ici : https://www.microsoft.com/cognitive-services/en-us/face-api. Microsoft propose un ensemble de photos de démonstration, mais propose d'envoyer ses propres photos pour tester. [4]

Dans notre cas seules les 2 premières informations nous intéressent :

- Le nombre de visages
- La position du visage

Implémentation technique

Comme pour Speaker Recognition, il faut d'abord demander une clef d'utilisation aux Cognitive Services, pour ce faire, simplement aller sur le site https://www.microsoft.com/cognitive-services/en-US/subscriptions?mode=NewTrials et demander l'accès à « Face – Preview » [5]

Et récupérer la clef masquée ici en cliquant sur le bouton « Show ». Et en allant ensuite dans Visual Studio.

Il faut ensuite télécharger le package Nuget Microsoft.ProjectOxford.Face [6]

4

6

Puis commencer à implémenter via l'objet FaceServiceClient

```
using (FaceServiceClient fsc = new FaceServiceClient("XXX"))
```

Où XXX correspond à la clef récupérée précédemment.

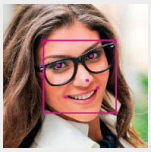
De là, en utilisant un fileStream d'une image à tester, on appelle l'API :

```
var faces = fsc.DetectAsync(imageFileStream);
var faceRects = faces.Result.Select(face => face.FaceRectangle);
```

Ainsi l'objet faceRects contient l'ensemble des visages repérés sur la photo. Dans ce cas il suffit déjà de tester les conditions :

- Si l'objet est vide, il n'y a pas de visage : je rejette la photo ;
- Si l'objet a plus d'un élément j'ai trop de visages : je rejette la photo.

A noter qu'il faut au moins un visage de 16 x 16 pixels pour être repéré. S'il n'y a qu'un seul visage, nous pouvons récupérer les coordonnées et la taille de celui-ci :



```
var firstface = faceRects.First();
    int left = firstface.Left;
    int top = firstface.Top;
    int width = firstface.Width;
    int height = firstface.Height;
```

Attention toutefois, car comme visible sur la photo d'exemple, le rectangle est plus petit que le visage donc pour découper l'image il faut augmenter la taille du carré du double environ. C'est pourquoi nous ne rejetions que les photos dont la width ou la height étaient inférieures à 150 px pour éviter une trop grosse perte lors d'un redimensionnement grossissant.

L'API peut renvoyer des exceptions de type **FaceAPIException**. Typiquement si les photos envoyées sont trop grosses, trop petites, ou que la limite des quotas fixés par le mode de souscription est dépassée.

Tarifs

Nous évoquons ici l'utilisation de Cognitive Service de manière concrète et sur des données volumineuses. La question du prix se pose tout naturellement.

Dans notre exemple, il n'y avait que 2500 photos, ce qui est largement suffisant pour effectuer plusieurs traitements dans la limite des 30000 transactions par mois. Il faut toutefois bien respecter un temps d'attente entre chaque transaction pour éviter de dépasser les 20 transactions par minute.

Le modèle Standard (S0) s'achète sur Microsoft Azure et baisse cette limite à 10 transactions par seconde. Le prix est d'environ 1,26 pour 1000 transactions. •

5

L'exemple des **API Cognitives** d'IBM

• Romain Willmann
Enseigne actuellement le développement Web à emlyon business school. Il conseille et forme également des entrepreneurs du digital.
Twitter : @rwillmann
Github : https://github.com/rwillmann
Linkedin : https://www.linkedin.com/in/rwillmann

Définir l'informatique cognitive n'est pas une chose aisée. L'industrie informatique assiste en effet depuis une dizaine d'années à une véritable prolifération de buzzwords qui recouvrent parfois des réalités difficiles à distinguer : si le Big Data est assez simple à différencier de l'intelligence artificielle, que dire de la comparaison parfois faite entre machine learning et deep learning ? Plutôt que de donner une définition subjective et nécessairement incomplète de l'informatique cognitive, nous essayerons ici d'en préciser le plus possible les contours.

Le mot cognitif provient du terme latin *cognosco*, qui désigne l'acte intellectuel par lequel on acquiert une connaissance ou la manière dont on apprend à connaître. L'informatique cognitive s'intéresse ainsi aux outils et aux technologies qui permettent à un ordinateur d'acquérir une connaissance. Ce champ est par définition très large et englobe des technologies diverses, parfois anciennes. Ainsi, la reconnaissance optique de caractères, parce qu'elle permet de transformer un écrit en une information qui peut être exploitée par un ordinateur, entre dans le champ de l'informatique cognitive. Le machine learning, parce qu'il permet à un programme d'analyser empiriquement des informations, peut également entrer dans la définition de l'informatique cognitive. Contrairement à ce que pourraient laisser croire certains discours marketing approximatifs, l'informatique cognitive n'est ainsi pas une révolution, mais d'avantage une combinaison de technologies déjà existantes, dont le but est de rapprocher au maximum le comportement d'un programme informatique de celui d'un humain.

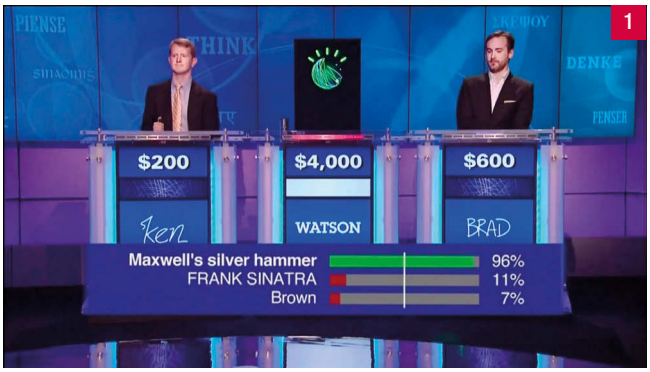
Pour y parvenir, les technologies cognitives cherchent à améliorer la manière dont un programme parvient à traiter de l'information non structurée, c'est-à-dire produite par l'homme et qui n'a pas de signification pour une machine (une image, un son, un fichier binaire). Même si les technologies s'améliorent constamment, il n'est, à l'heure actuelle, pas possible d'avoir une machine totalement cognitive : l'informatique cognitive permet seulement de *simuler* l'acquisition de connaissance.

La bonne nouvelle pour les développeurs est qu'il est possible de jouer dès maintenant avec des APIs cognitives de pointe !

CHOISIR UNE PLATEFORME

L'écosystème IBM Watson

IBM est l'entreprise la plus active dans le domaine de l'informatique cognitive(1). Elle a en effet été la première à montrer un démonstrateur concret de sa technologie cognitive. En 2011, son programme d'intelligence artificielle Watson est en effet parvenu à battre deux anciens champions américains du jeu télévisé Jeopardy ! (une sorte de version outre-Manche de Question pour un champion) Pour réussir cet exploit, Watson était capable d'analyser les questions, de chercher les réponses dans un corpus, et de produire une synthèse vocale de la réponse. Au-delà d'être une réactualisation de l'affrontement entre Deep Blue et Kasparov en 1997, Watson est également une activité stratégique d'IBM. IBM a construit un écosystème autour de sa technologie cognitive, notamment en faisant l'acquisition d'entreprises spécialisées dans ce domaine (AlchemyAPI en mars 2015, Truven Health Analytics en février 2016…). Depuis 2014, Watson est essentiellement utilisé à des fins médicales. Le programme analyse l'ensemble des publications scientifiques disponibles, notamment sur le cancer, et les compare avec le dossier médical d'un patient afin de rendre possible la création de traitements personnalisés. [1]



Watson au cours de l'émission Jeopardy. Les réponses étaient assorties d'un score de pertinence (crédit : YouTube)

IBM rend accessible ses outils cognitifs sur sa plateforme Cloud IBM Bluemix. Ils prennent la forme d'une vingtaine d'APIs REST spécialisées sur des tâches précises comme la synthèse vocale, l'analyse d'image ou l'analyse d'un texte. Bluemix propose de loin le catalogue le plus varié d'APIs cognitives et la plupart d'entre elles comportent un niveau gratuit assez généreux. Au-delà d'un certain seuil, les APIs sont facturées à l'usage. Les APIs reposent sur l'architecture REST et peuvent donc être accessibles à l'aide de requêtes HTTP. IBM propose également des *wrappers* pour faciliter leur intégration avec les principaux langages et outils de développement (Node.js, Java, iOS…). L'offre d'IBM évolue constamment et le catalogue d'APIs est fréquemment mis à jour. Cela se ressent parfois sur la qualité de la documentation des APIs, qui est assez fragmentée et parfois aléatoire(2).

Nous utiliserons quatre des services cognitifs d'IBM dans la seconde partie de cet article. Les exemples d'applications sont réalisés avec Node.js mais les APIs peuvent être utilisées avec la majorité des autres outils et langages de programmation.

Les Microsoft Cognitive Services

Microsoft a développé un ensemble de technologies cognitives que la firme de Redmond met en avant depuis 2010. Certaines d'entre elles sont d'ailleurs directement incorporées dans ses produits, comme l'assistante virtuelle Cortana. Les développeurs peuvent accéder à ces technologies

(1). http://www.lesechos.fr/03/06/2015/LesEchos/21950-086-ECH_watson-d-ibm--l-une-des-premieres-applications-grand-public.htm
(2). https://watson-api-explorer.mybluemix.net/

de deux manières. Microsoft propose depuis janvier 2016 un SDK Open Source autour du cognitif, le Microsoft Cognitive Toolkit(3). Il existe également une collection d'APIs REST, les Cognitive Services APIs, accessibles depuis Microsoft Azure. Ces dernières sont utilisables en mode SaaS, sont facturées à l'usage et comportent la plupart du temps un niveau gratuit suffisant pour découvrir les technologies cognitives ou créer une petite application. Microsoft a été la première entreprise à montrer les limites de l'informatique cognitive, mais aussi de ses propres technologies. En mars 2016, Microsoft a lancé Tay, un logiciel capable de communiquer de manière autonome via Twitter. Le *chatbot* était initialement conçu pour répliquer le comportement d'un jeune adulte et d'apprendre de ses échanges avec les autres utilisateurs. Le programme a rapidement multiplié de manière alarmante les propos racistes, antisémites et négationnistes quelques heures après son lancement. Tay a été arrêté moins de seize heures après son lancement(4). Ironie de l'histoire, les Cognitive Services APIs proposent des modules consacrés à la modération de contenu…

La plateforme Haven OnDemand

Troisième entreprise mondiale sur le marché des logiciels professionnels, Hewlett Packard Enterprise propose également des APIs cognitives qui peuvent être intégrées dans des applications. L'essentiel des technologies cognitives d'HPE provient de l'acquisition d'Autonomy en 2011. Son catalogue d'APIs cognitives porte principalement sur l'analyse de flux vidéo ou audio et du *data mining*. On y trouve également quelques APIs très spécialisées, qui permettent par exemple d'extraire des numéros de plaque d'immatriculation d'une vidéo.

Pour pouvoir utiliser les APIs cognitives d'HPE, il faut utiliser sa plateforme Cloud dédiée au Big Data, Haven OnDemand(5). Accessible à tous depuis la fin de l'année dernière, cette dernière propose une cinquantaine d'APIs cognitives mais également des outils analytiques plus classiques qui reposent pour la plupart sur le framework Hadoop.

Contrairement aux deux plateformes précédentes, Haven OnDemand repose sur un modèle d'abonnement. Le niveau gratuit permet 10 000 appels d'APIs par mois. Il faut ensuite compter de 10 à 350 dollars pour disposer de plus d'appels. Les APIs d'HPE reposent toutes sur l'architecture REST et l'entreprise propose également des *wrappers* pour les principaux langages ainsi qu'une interface graphique qui permet de les prendre en main rapidement.

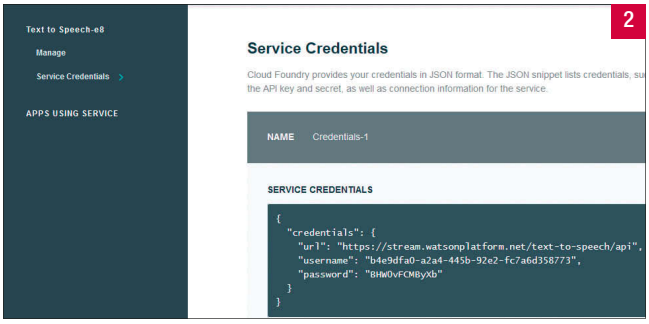
INTÉGRER UNE API COGNITIVE À UNE APPLICATION

Les APIs cognitives permettent à un programme de mieux traiter de l'information non structurée. Nous verrons ici les deux principaux cas d'usage de ces technologies : l'analyse de texte et l'interprétation de fichiers multimédia.

Analyser du texte

Etablir un profil psychologique

Comme mentionné précédemment, l'un des objectifs poursuivis par les technologies cognitives est d'améliorer la compréhension de données non structurées par un ordinateur. Parmi celles-ci, le langage naturel, c'est-à-dire utilisé par des humains, occupe une place importante. On trouve sans surprise de nombreuses APIs autour de l'analyse de texte. Watson intègre ainsi un module, l'API Personality Insights, qui tente d'établir le profil psychologique d'un individu à partir de ses écrits. Si la fiabilité et la pertinence des méthodes de *profiling* prêtent à discussion, elles



Il faut activer un service dans IBM Bluemix pour pouvoir accéder aux credentials

n'en restent pas moins fréquemment utilisées dans le domaine du marketing ou des ressources humaines.

L'API Personnality Insight repose sur des méthodes d'analyse sémantique(6) qui permettent de déterminer les traits de personnalité saillants exprimés dans un texte. Pour fonctionner, l'API nécessite un texte d'au moins cent mots, écrits dans l'une des dix langues compatibles avec le service. Le niveau gratuit de l'API permet cent appels par mois, facturés aux alentours de 0,02 $ par appel au-delà.

Pour tester les services, nous créerons une application Node.js simple, qui utilise le module watson-developer-cloud(7). Ce dernier est un *wrapper* qui permet d'intégrer plus simplement les services cognitifs d'IBM dans une application Node.js. L'API Personality Insight s'utilise en trois étapes. Il faut d'abord s'authentifier auprès du service, en lui fournissant des *credentials*. Pour les trouver, il faut tout d'abord activer le service dans Bluemix, puis aller dans l'onglet intitulé Service Credentials. Il faut ensuite envoyer à l'API un texte d'au moins cent mots. Elle retourne alors un objet JSON qui contient les résultats détaillés de l'analyse. [2]

Nous créons ici une application minimaliste, qui transmet un texte à l'API. Elle nécessite au moins cent mots pour produire un résultat mais ce dernier devient significatif à partir de 600 mots ou plus. L'API est à l'heure actuelle capable d'analyser onze langues, dont le français.

```
var PersonalityInsightsV3 = require('watson-developer-cloud');

// Les identifiants se trouvent sur Bluemix
var personality_insights = new PersonalityInsightsV3({
  username: '<username>',
  password: '<password>',
  version_date: '2016-10-19'
});


personality_insights.profile({
  text: 'Tapez au moins 100 mots ici'
  },
  function (err, response) {
    if (err) {
      console.log('erreur:', err);
      }
    else {
```

(3). https://github.com/Microsoft/CNTK/
(4). https://www.theguardian.com/technology/2016/mar/24/tay-microsofts-ai-chatbot-gets-a-crash-course-in-racism-from-twitter
(5). https://www.havenondemand.com/
(6). http://www.ibm.com/watson/developercloud/doc/personality-insights/science.shtml
(7). https://www.npmjs.com/package/watson-developer-cloud

```
    console.log(JSON.stringify(response, null, 2));
  }
});
```

Le document JSON retourné par l'API est un arbre, qui identifie les traits de caractère qui peuvent être identifiés dans le texte. L'analyse est pertinente si le texte fourni est relativement long et écrit par une seule personne. À noter que l'API peut également analyser une page HTML ou encore un document JSON.

```
        ...
  "personality": [{
    "trait_id": "big5_openness",
    "name": "Openness",
    "category": "personality",
    "percentile": 0.7443507132159689,
    "raw_score": 0.7696218324525445,
    "children": [{
      "trait_id": "facet_adventurousness",
      "name": "Adventurousness",
      "category": "personality",
      "percentile": 0.9706118681052658,
      "raw_score": 0.5720327382086956
    }, {
      "trait_id": "facet_artistic_interests",
      "name": "Artistic interests",
      "category": "personality",
      "percentile": 0.4063590784034766,
      "raw_score": 0.6533310097107441
    }, {
        ...
```

Identifier des concepts

La deuxième API que nous allons voir permet également d'analyser du texte, pour cette fois être capable d'en extraire les principaux concepts et le ton utilisé. L'API en question est AlchemyLanguage, qui est en partie intégrée à l'écosystème Watson. L'un des clients de ce type de service est l'entreprise SoftBank(8), qui a signé un partenariat avec IBM pour rendre Watson capable de traiter le Japonais.

L'API AlchemyLanguage peut se faire à l'aide du module utilisé précédemment. Nous allons ici créer une application Node.js qui permet d'analyser le contenu d'un fichier texte. Les résultats retournés prendront la forme d'un objet JSON. Il convient de noter qu'AlchemyLanguage fonctionne essentiellement sur des textes en anglais. L'envoi d'un texte en français retourne en effet une erreur.

```
// Le module fs est ici utilisé  pour lire les fichiers
var watson = require('watson-developer-cloud');
var fs = require('fs');

// Alchemy nécessite une clef API, qui peut être récupéré depuis Bluemix
var alchemy_language = watson.alchemy_language({
  api_key: 'API_KEY'
});

// L'API analyse les fichiers TXT, JSON ou HTML
var parameters = {
  'txt': data
};

fs.readFile('demo.txt', 'utf8', function (err, data) {
  if (err) {
    return console.log(err); //  veentuelle erreur   l'ouverture du fichier
  }
  alchemy_language.concepts(parameters, function (error, response) {
    if (error) {
      console.log('erreur:', error);
    }
    else {
      console.log(JSON.stringify(response, null, 2));
    }
  })
});
```

AlchemyAPI peut également analyser une page Web. Il suffit alors de passer une URL en paramètre lors de l'appel de l'API.

```
var parameters = {
  url: 'http://www.programmez.com/'
};
```

Tout comme pour l'API précédente, l'analyse prend la forme d'un fichier JSON assez dense.

```
          ...
  "language": "english",
  "entities": [
    {
      "type": "Company",
      "relevance": "0.953995",
      "sentiment": {
        "type": "positive",
        "score": "0.633616"
      },
      "emotions": {
        "anger": "0.572905",
        "disgust": "0.114106",
        "fear": "0.182145",
        "joy": "0.101376",
        "sadness": "0.108572"
      },
      "count": "2",
      "text": "Bluemix"
          ...
```

L'API compte une dizaine de fonctionnalités qui permettent d'étudier en profondeur un texte, en identifiant d'autres éléments (sentiments, mots-clefs…). Dans l'exemple ci-dessous, nous demandons à l'API de chercher à identifier l'auteur d'un texte en utilisant la méthode authors. Elle n'est bien sûr pertinente que si l'auteur est connu.

```
alchemy_language.authors(parameters, function (error, response) {
  if (error) {
    console.log('error:', error);
  }
  else {
    console.log(JSON.stringify(response, null, 2));
```

(8). http://www-03.ibm.com/press/us/en/pressrelease/46045.wss

Facebook fait un usage intensif de la classification d'images

```
  }
});
```

Voir et parler

L'une des applications majeures des APIs cognitives est de rapprocher le plus possible un ordinateur d'un humain, en le rendant capable de manipuler de l'information non-structurée. On trouve ainsi un certain nombre d'APIs qui rendent un programme capable d'analyser des images, un discours, ou encore de parler. Nous verrons ici comment utiliser des APIs pour rendre une application capable de voir et de parler.

Analyser une image

Nous utiliserons ici l'API Visual Recognition, qui est une composante de Watson. Comme toutes les autres APIs utilisées dans cet article, elle repose sur l'architecture REST. Il faut donc disposer d'un compte IBM Bluemix pour avoir une clef et pouvoir l'utiliser. Le niveau gratuit de l'API permet de traiter jusqu'à 250 images par jour. Au-delà, le service est facturé entre 0,004 $ et 0,002 $ par requête(9).

L'API fonctionne simplement : elle reçoit une image (ou son URL), l'analyse et retourne un résultat. La documentation de l'API(10) montre qu'elle peut effectuer une dizaine d'analyses différentes : identification de visages, recherche d'images par similarité, ou encore classification des éléments d'une image. Nous nous intéresserons ici à cette dernière fonctionnalité. [3]

Les images sont des données non structurées et non textuelles, omniprésentes dans la vie quotidienne. Leur classification est une tâche importante, car elle permet de rendre un programme capable de les manipuler, afin par exemple de les filtrer ou encore d'améliorer leur référencement. Les APIs cognitives autour de la reconnaissance visuelle sont ainsi très prisées des réseaux sociaux. Nous réaliserons ici une petite application Node.js qui permet à un utilisateur de soumettre une image. Cette dernière sera alors analysée par l'API Visual Recognition et nous afficherons les résultats de la classification. Côté client, une application minimaliste comportera au moins une page Web avec un formulaire qui permet à un utilisateur d'envoyer un fichier. Celui-ci sera placé dans un fichier que nous nommerons par convention index.html.

```
<form action="/post" enctype="multipart/form-data" method="post">
  <input id="fileToUpload" name="fileToUpload" type="file">
  <input name="submit" type="submit" value="Envoyer l'image">
</form>
```

Côté serveur, nous créerons une application avec Express, qui comporte deux routes : l'une affichera la page Web créée précédemment et l'autre permettra de récupérer l'image envoyée. Cette dernière sera alors envoyée à l'API de reconnaissance visuelle.

```
// Chargement des dépendances : express, body-parser et multer sont requis pour
 créer notre application
var express = require('express');
var bodyParser = require('body-parser');
var multer  = require('multer');

// Chargement de la librairie de l'API Visual Recognition. Cette dernière nécessite
un accès au système de gestion de fichier de Node.js
var fs = require('fs');
var watson = require('watson-developer-cloud');

var app = express();

// Configuration de l'application pour qu'elle puisse recevoir des fichiers
app.use(bodyParser.urlencoded({ extended: true }));
app.use(multer({ dest: './uploads/'}));

// La clef API est à récupérer sur Bluemix
var visual_recognition = new VisualRecognitionV3({
  api_key: '<clef>',
  version_date: '2016-05-19'
});

// Page d'accueil de l'application
app.get('/', function (req, res) {
  res.sendFile(__dirname + "/index.html");
});

// Les images fournies par l'utilisateur sont envoyées sur la route /post
app.post('/post', function (req, res) {

  var params = {
    // On prépare l'envoi de l'image vers l'API
    image_file: fs.createReadStream(req.files.fileToUpload.path)
  };

  // Affichage des résultats
  visual_recognition.recognize(params, function(err, res) {
    if (err) {
      //On loggue les erreurs éventuelles
      console.log(err);
    } else {
      // On récupère sinon les classificateurs de l'image
      console.log(JSON.stringify(res, 'labels', 2));
    }
  });
});

app.listen(8080);
```

Une fois que l'image a été envoyée sur les serveurs d'IBM, elle est analysée puis les résultats sont renvoyés sous forme d'un objet JSON. Ce dernier contient le plus souvent une dizaine de termes qui décrivent l'image, accompagnés d'un intervalle de confiance qui permet de déterminer la fiabilité de l'analyse. [4]. Ces chiffres sont très importants, car

(9). https://console.ng.bluemix.net/catalog/services/visual-recognition
(10). https://watson-api-explorer.mybluemix.net/apis/visual-recognition-v3

l'API Visual Recognition est capable du meilleur comme du pire. Ainsi, l'image de canard [4] produit le résultat suivant.

```
"classes": [
{
  "class": "mallard",
  "score": 0.918,
  "type_hierarchy": "/animal/aquatic bird/waterfowl/duck/mallard"
},
{
  "class": "duck",
  "score": 0.925
},
   ...
{
  "class": "cockatoo parrot",
  "score": 0.554,
  "type_hierarchy": "/animal/bird/cockatoo parrot"
},
   ...
```

On constate ici que l'API est parvenue à identifier avec précision que l'image contenait un canard et même à identifier son espèce. Toutefois, le fichier JSON contient des informations non pertinentes, qui sont ici les mots "cockatoo parrot" et "parrot". Ce sera donc au développeur de filtrer les résultats. Pour y parvenir, il est possible de parser l'objet JSON reçu ou encore de définir un attribut threshold lors de l'appel de l'API. Comme son nom l'indique, il permet de définir l'intervalle de confiance minimum qu'un résultat doit avoir pour être retourné par l'API.

```
var params = {
  image_file: fs.createReadStream(req.files.fileToUpload.path),
  threshold : 50
};
```

Il convient de noter que l'API apprend au fil du temps. Elle devient ainsi plus précise dans ses identifications et dans les classificateurs utilisés.

Une synthèse vocale

La seconde API que nous allons voir permet de faire de la synthèse vocale. Nous utiliserons ici l'API Text to Speech, disponible également sur IBM Bluemix. Il s'agit du module utilisé par Watson en 2011 et qui a depuis évolué, puisqu'il est désormais capable de « parler » dans sept langues différentes, dont le français. La synthèse vocale mobilise de nombreuses techniques issues de différents domaines, comme la linguistique ou l'acoustique. Créer une synthèse vocale de qualité est donc une tâche complexe qui mobilise de nombreuses ressources scientifiques et techniques. L'introduction des technologies cognitives dans un processus de synthèse vocale a pour but d'en améliorer la qualité et de gagner en rapidité. L'API d'IBM permet ainsi de faire de la synthèse vocale dans le Cloud, en mode SaaS. Le niveau gratuit de l'API permet de synthétiser un million de caractères par mois (environ 250 pages A4). Au-delà, il faut compter 0,02 $ par millier de caractères traités(11).

En nous appuyant sur la documentation de l'API, nous allons créer une petite application qui enverra un texte vers les serveurs d'IBM. La synthèse vocale sera alors enregistrée dans un répertoire de l'application.

```
var watson = require('watson-developer-cloud');
var fs = require('fs');
```

Watson détecte un canard colvert mais également un perroquet (crédit : Mallard landing in approach, Bert de Tilly, CC BY-SA 4.0)

```
// On remarque ici que l'identification se fait au moyen d'un nom d'utilisateur et
d'un mot de passe
var text_to_speech = new TextToSpeechV1({
  username: '<username>',
  password: '<password>'
});

var params = {
  text: " Entrez un texte ici" ,
  voice: 'fr-FR_ReneeVoice',// Il n'y a à l'heure actuelle qu'une seule voix française
  accept: 'audio/wav'
};

// Le texte synthéthisé est enregistré dans le répertoire de l'application au moyen
 d'un stream
text_to_speech.synthesize(params).pipe(fs.createWriteStream('output.wav'));
```

CONCLUSION

Les technologies cognitives sont prometteuses et tout développeur sérieux devrait se pencher dessus, d'autant plus que leur prise en main peut être assez simple. Pourtant, s'il est possible de parvenir à réaliser une petite application cognitive, de nombreuses questions se posent pour des projets de plus grande envergure.

En effet, l'informatique cognitive est hautement stratégique pour les entreprises. Elles dévoilent assez rarement leurs algorithmes et méthodes. Les solutions cognitives sont donc essentiellement construites sur des technologies propriétaires. Ceci peut constituer un frein important dans leur adoption, car elles sont en concurrence avec les technologies autour du Big Data, qui sont la plupart du temps Open Source (Hadoop, NLTK, etc). Les DSI et les entreprises auront un arbitrage à faire entre l'utilisation d'une technologie cognitive tierce, susceptible d'évoluer et propriétaire, et le développement de solutions *ad hoc* à partir d'outils Open Source. Enfin, l'utilisation des technologies cognitives a un coût qui demeure pour l'instant élevé : la plupart des APIs présentées dans ce dossier ont un prix qui peut dépasser 0,01 par appel. Il est donc nécessaire de trouver un business model suffisamment rentable pour amortir le coût des fonctions cognitives. Il faut également garder en tête qu'elles peuvent avoir des résultats parfois approximatifs, voire décevants. En l'état, les APIs cognitives sont donc des produits de niche, qui peuvent néanmoins apporter énormément de valeur ajoutée à un projet. •

(11). https://console.ng.bluemix.net/catalog/services/text-to-speech

La librairie OpenCV

• Bertrand Lanneau

Bonjour ! Voici un article visant à décrire la librairie openCV, son fonctionnement et son intégration. Opencv(Open Source Computer Vision) est une librairie open source (BSD) de traitements d'images qui offre une grande palette d'outils initialement développée par Intel. Depuis 2008, la société Willow Garage met à jour, enrichit et ouvre à de multiples plateforme le déploiement de cette librairie.

Plateformes

A l'heure actuelle, cette librairie est déployée pour les plateformes Windows, Linux, iOS et Android. Il est également possible de l'intégrer sur des supports Arm type Raspberry Pi. Côté langage, initialement développé en C, nous trouvons aujourd'hui cette librairie déclinée en C++, Python et Java. L'ensemble est très bien documenté et il est assez aisé de se lancer sans être érudit grâce à des samples et autres démos opérationnelles. Pour les ressources, il est évident que nous ne pouvons pas nous passer d'une configuration µP/Mémoire vive suffisante. Le traitement d'image, et, de plus en live, nous impose une puissance de calcul certain. Je n'ai pas testé OpenCV sur toutes les plateformes mais pour vous donner un ordre d'idée, j'ai utilisé cette librairie sur un Raspberry Pi B+ en Java. Pour un projet de détection via la caméra du Pi, j'avais un temps de traitement de l'ordre de la seconde avec une résolution réglée au minimum. Pour des plateformes type ordinateur de bureaux actuels, vous pourrez traiter bien plus de 10 images/secondes.

Installation

Sous forme d'un setup, et une fois installée, la librairie vous propose une arborescence de répertoires où vous trouverez l'ensemble des fichiers nécessaires à son utilisation. [1]

Je ne vais pas entrer dans le détail des imports de la librairie OpenCV dans les différents outils de développement. Toutefois j'ai pu l'utiliser avec Borland C++ et Eclipse (Java) sans problème.

Pour les novices, vous trouverez beaucoup de documentation sur le Net pour vous aider à l'importer dans votre plateforme de développement préférée. Les sources nécessaires pour découvrir et tester cette librairie sont stockées dans le répertoire « samples » et je dois dire que c'est assez complet. Après ce tour d'horizon, nous allons passer à la description des grandes fonctionnalités d'OpenCV.

Les fonctionnalités

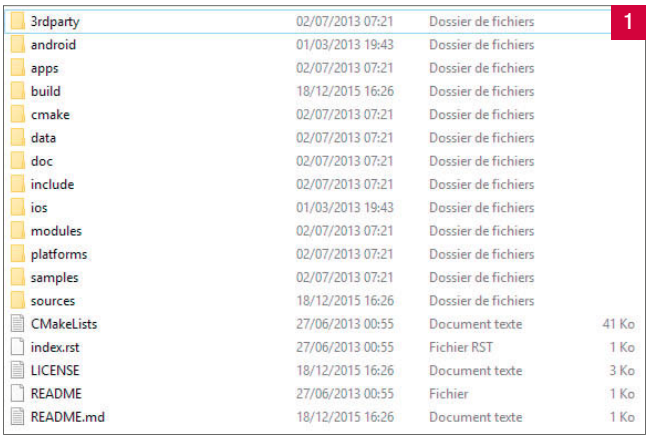
Voici les principaux traitements que je vais vous décrire :

- Capture, Lecture/écriture et affichage de photos et vidéos.
- Traitements de l'image (contrastes, nuances de gris, de couleurs, filtrage, …).
- Détection de formes et de mouvements.

Il est à noter que les concepteurs de cette librairie ont réalisé un travail fastidieux et très complet. Tous les éléments nécessaires aux traitements des images, et même les plus basiques, sont à notre disposition.

1 : Acquisition Vidéo

Quel que soit votre périphérique d'acquisition d'images, OpenCV scrute votre matériel et vous en donne un accès direct. Et ceci grâce à la classe VideoCapture. A l'instanciation de cette classe, Opencv nous propose plusieurs sources de flux vidéo. Depuis votre matériel, en pointant vers celui-ci avec un Integer en paramètre (0 : première source d'acquisition


1

3rdparty	02/07/2013 07:21	Dossier de fichiers	
android	01/03/2013 19:43	Dossier de fichiers	
apps	02/07/2013 07:21	Dossier de fichiers	
build	18/12/2015 16:26	Dossier de fichiers	
cmake	02/07/2013 07:21	Dossier de fichiers	
data	02/07/2013 07:21	Dossier de fichiers	
doc	02/07/2013 07:21	Dossier de fichiers	
include	02/07/2013 07:21	Dossier de fichiers	
ios	01/03/2013 19:43	Dossier de fichiers	
modules	02/07/2013 07:21	Dossier de fichiers	
platforms	02/07/2013 07:21	Dossier de fichiers	
samples	02/07/2013 07:21	Dossier de fichiers	
sources	18/12/2015 16:26	Dossier de fichiers	
CMakeLists	27/06/2013 00:55	Document texte	41 Ko
index.rst	27/06/2013 00:55	Fichier RST	1 Ko
LICENSE	18/12/2015 16:26	Document texte	3 Ko
README	27/06/2013 00:55	Fichier	1 Ko
README.md	18/12/2015 16:26	Document texte	1 Ko

trouvée par Opencv, 1 la deuxième,…). Vous avez aussi la possibilité de renseigner une String pour pointer vers un flux IP ou un fichier.

Voici l'exemple issu de la documentation officielle :

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <stdio.h>
using namespace cv;
using namespace std;
int main(int, char**)
{
    Mat frame;
    //--- INITIALIZE VIDEOCAPTURE
    VideoCapture cap;
    // open the default camera using default API
    cap.open(0);
    // OR advance usage: select any API backend
    int deviceID = 0; // 0 = open default camera
    int apiID = cv::CAP_ANY; // 0 = autodetect default API
    // open selected camera using selected API
    cap.open(deviceID + apiID);
    // check if we succeeded
    if (!cap.isOpened()) {
        cerr << "ERROR! Unable to open camera\n";
        return -1;
    }
    //--- GRAB AND WRITE LOOP
    cout << "Start grabbing" << endl
    << "Press any key to terminate" << endl;
    for (;;)
    {
    // wait for a new frame from camera and store it into 'frame'
    cap.read(frame);
```

```
    // check if we succeeded
    if (frame.empty()) {
        cerr << "ERROR! blank frame grabbed\n";
        break;
    }
    // show live and wait for a key with timeout long enough to show images
    imshow("Live", frame);
    if (waitKey(5) >= 0)
        break;
    }
    // the camera will be deinitialized automatically in VideoCapture destructor
    return 0;
}
```

2 : Acquisition d'une image :

La classe Huighui nous propose une fonction imRead qui permet de charger une image en mémoire.

```
imRead(const String filename, int flags)
Parameters
filename  Name of file to be loaded.
flags  Flag that can take values of cv::ImreadModes
```

Comme vous pouvez le constater dans ces exemples, les images capturées sont stockées dans un format particulier de type Mat. L'ensemble des manipulations des images par Opencv est traité grâce à ce type. C'est une matrice composée d'un tableau défini par trois éléments :

```
Cv ::Mat ::Mat(int rows, int cols, int type)

Parameters
rows  Number of rows in a 2D array.
cols Number of columns in a 2D array.
type Array type. Use CV_8UC1, CV_64FC4 to create 1-4 channel matrices, or CV_8UC(n),
 ..., CV_64FC(n) to create multi-channel (up to CV_CN_MAX channels) matrices.
```

Voici ce que nous décrit la documentation.
Une fois les images en mémoire, passons maintenant à quelques traitements mis à notre disposition. Pour cela, nous allons utiliser la classe Imgproc. Cette classe nous donne accès à des fonctionnalités tels que le redimensionnement, le flou, la détection des contours, la dilatation, les nuances de gris, le dessin de formes, et bien d'autres. Par exemple, pour générer un flou, rien de plus simple (Java) en mode static: [2]

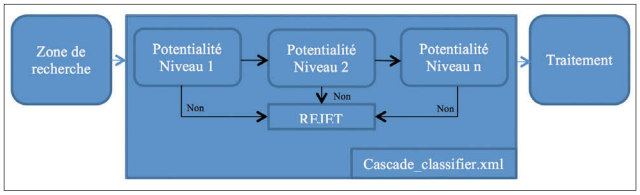
```
Mat img=Highgui.imread("chemin_de_votre_image_d'entrée.png");
Imgproc.blur(img, img, new Size(30,30));
Highgui.imwrite("chemin_de_votre_image_de_sortie.png", img);
```

Pour terminer, les fonctions imRead et imwrite comme illustré ci-dessus, permettent les accès en lecture et écriture sur votre disque dur.
Maintenant nous allons évoquer une des autres nombreuses fonctionnalités : la détection de formes.

Image d'entrée *Image de sortie*

La détection de formes

La méthode utilisée par Opencv est celle proposée par Viola et Jones. Elle consiste à reconnaître une cible sur une image en s'appuyant sur un fichier généré à partir de milliers de clichés. Suite à une compilation de fonds ou « backgrounds » et d'images à détecter « foregrounds », un fichier xml déterminera la procédure à suivre pour détecter la forme voulue. On parle d'apprentissage pour la génération de ce fichier, que l'on peut assimiler au Machine Learning. Il est basé sur un fonctionnement en cascade, ce qui permet une plus grande rapidité de calcul. L'image à traiter est segmentée en plusieurs zones auxquelles s'applique le processus de détection.



Voici le développement d'une application pas à pas en Java pour détecter un visage. En premier lieu nous allons mettre en place une classe qui sera appelée depuis le « main » de votre application.

```
public final class FaceDetect {
        run()
        {
  //Insérez votre code ici
        }
}
```

Intégrez le chargement de la librairie.
Puis instanciez les classes et variables suivantes :

```
// Classe fournissant les séquences d'images à traiter depuis votre caméra.
 L'argument de type integer permet de pointer vers la caméra souhaitée.
final VideoCapture videoCapture = new VideoCapture(0);
//Classe permattant d'effectuer les traitements sur images(matrice).
final Mat mat = new Mat();
//Définissez une taille à vos images capturées. En ajustant ces valeurs, vous
pouvez gagner un temps considérable de calcul.
final Size taille = new Size(260, 180);
//Compteur d'images avec visages détectés.
int nbImagesAvecFace = 0;
//Classe invoquant le fichier xml cascade pour la détection. En argument, le
chemin de votre fichier.
CascadeClassifier faceDetector = new CascadeClassifier("/opencv/data/haar
cascades/haarcascade_frontalface_alt.xml");
//Cette classe récupère les images avec les objets détectés
MatOfRect faceDetections = new MatOfRect();
```

Maintenant, nous allons utiliser une boucle qui permettra de capturer une nouvelle image à chaque fois qu'elle sera disponible :

```
while (videoCapture.read(mat))
{
        //Traitement des images
}
```

Dans cette boucle, nous allons dans un premier temps retravailler l'image capturée.

```
Mat matTraitement = new Mat();
matTraitement.convertTo(matTraitement, CvType.CV_8U);// Paramétrage de 8bits/pixel
Imgproc.resize(mat, matTraitement,taille);//Copie de l'image capturée dans sa coquille avec la dimension voulue.
```

Et voilà la fonction qui va pouvoir effectuer les traitements de détections. Je vais décrire plus bas les arguments de cette fonction.

```
faceDetector.detectMultiScale(matTraitement,faceDetections,1.1,1,1,new Size(10,10),new Size(40,40));
```

Suite à l'appel de cette fonction, "faceDetection" contiendra les éventuelles images contenant un visage. On va donc pouvoir récupérer les captures qui ont matché dans une boucle « for ».

```
for (Rect rect : faceDetections.toArray())
 {
      System.out.println("DETECTION!");
      nbImagesAvecFace ++;
 }
```

Si vous souhaitez dessiner un rectangle autour de la cible et récupérer les coordonnées du visage sur l'image ainsi que sa taille, la librairie opencv a mis à notre disposition la fonction suivante :

```
Core.rectangle(videoMatGray, new Point(rect.x, rect.y), new Point(rect.x +rect.width, rect.y + rect.height),new Scalar(0, 255,0));
```

Enfin, si vous le souhaitez, enregistrez l'image sur votre disque.

```
Highgui.imwrite("Capture_"+ nbImagesAvecFace +".jpg",mat);
```

Sans avoir défini un chemin au nom de votre image, elle sera enregistrée par défaut dans le répertoire de votre application. Pour terminer votre classe, après votre boucle while, libérez votre caméra.

```
videoCapture.release();
```

Enfin, lancez votre classe FaceDetect dans votre « main » :

```
public static void main(String[] args) throws Exception
{
      FaceDetect.run();
}
```

Compilation de votre code :

Pour tester l'installation d'OpenCV, vous avez deux possibilités.
Méthode 1 ;
Compilez et lancez directement votre code depuis votre invite de commande en utilisant les commandes suivantes :

```
javac –cp .:* OpencvTest.java
java –cp .:* OpencvTest
```

Méthode 2 :
Utilisez votre plateforme de développement et exportez votre projet en un .jar exécutable. Export de votre projet depuis Eclipse:
Cliquez droit sur votre projet puis sélectionnez **Run/Debug Setting**
Ensuite dans la zone **Main Class** cliquez sur **Search** et sélectionnez votre classe principale et validez. De nouveau clic droit sur votre projet puis **Exporter...** Déroulez java et choisissez **Runnable JAR file** [3]
Cliquez sur **Next** et dans la zone « **Launch configuration** », récupérez votre configuration générée à l'étape précédente. Donnez un nom et un chemin à votre .jar dans la zone **Export Destination.**

Enfin, cliquez sur **finish**. Lancez votre application avec cette commande :

```
sudo java -Djava.library.path=/usr/lib  -jar opencv/OpencvTest.jar //OpencvTest étant le nom que vous avez donné à votre jar exécutable.
```

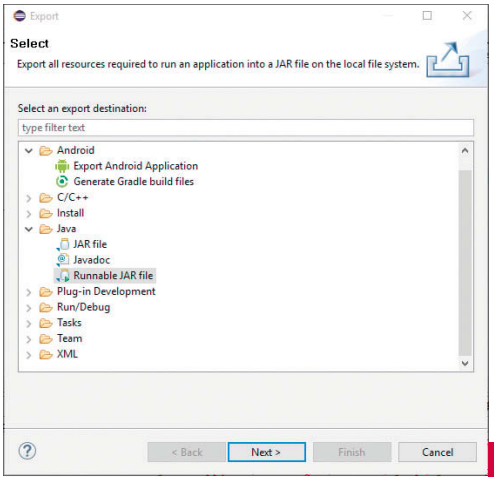
Si tout s'est bien passé, vous pouvez tester votre programme et l'améliorer en modifiant quelques paramètres de la fonction « detectMultiScale() ».

La méthode « detectMultiScale() »

Nous allons maintenant nous intéresser aux arguments de la fonction principale de détection de forme **detectMultiScale()**. Vous avez la possibilité de modifier certains paramètres afin d'affiner la détection en fonction de votre environnement.
Cette fonction permet donc de nous retourner un tableau de MatOfRect de taille égale au nombre de détections par image analysée.

```
detectMultiScale(Mat image, MatOfRect objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size(), Size maxSize=Size());
```

Argument 1 :**image**, type Mat() :
Image d'entrée à analyser par la fonction.
Argument 2 :**object**,type MatOfRect()
Vecteur de rectangles ou chaque rectangle contient l'objet détecté.
Argument3 :**scaleFactor**, type Double.
Facteur de redimensionnement de l'image source. Lors du traitement, l'image est redimensionnée à plusieurs reprises (type pyramidale).
Plus vous augmentez ce paramètre, plus le temps de calcul sera long mais plus vous augmentez vos chance de détecter plusieurs sujets sur la même image.
Argument 4 : **minNeighbors**, type integer
Paramètre spécifiant le nombre de rectangles voisins que chaque rectangle candidat devrait avoir à conserver.
Argument 5 : **flags**, type integer.
Non utilisé pour les versions récentes. Mettre 1.
Argument 6 : **minSize**, type Size.
Définition de la taille minimum de la cible. En dessous de cette taille, tous les éléments seront rejetés.
Argument 7 : **maxSize**, type Size.
Définition de la taille maximale de la cible. Au-dessus de cette taille, tous les éléments seront rejetés.
Nous voilà arrivés au terme de cet article. En espérant qu'il vous sera utile pour vos projets de détection. Vous pouvez aller jeter un coup d'œil sur mon blog où j'ai intégré Opencv sur Raspberri Pi. https://myexperimentalrobot.wordpress.com/ A bientôt. •



3

Taxer les robots : l'émergence d'une société hybride

François Tonic

Il y a quelques semaines, les médias ont parlé de la possibilité de taxer les robots. Cette taxation pourrait, selon les auteurs de cette idée, financer la sécurité sociale, le revenu universel ou d'autres choses. Une des logiques de cette proposition est que le robot détruit les emplois humains donc il faut bien taxer quelque chose et aider l'humain à vivre. Comme toujours, il y a les partisans et les opposants. Mais aujourd'hui, les robots équipent les usines et ces chaînes de fabrication ne sont pas nouvelles ; les robots existent depuis longtemps.

L'automatisation serait beaucoup plus forte aujourd'hui, à cause des robots ; on détruirait ainsi de nombreux emplois et de nombreux métiers, 90-96 % selon les études seront concernés dans un futur proche. Donc, moins de salariés humains, moins de taxes et d'impôts pour les États, un appauvrissement généralisé. Donc, taxons les robots pour récupérer l'argent manquant.

Pas aussi simple

Je simplifie le débat de ces dernières semaines. Mais personne n'est forcément d'accord sur l'évolution de l'emploi salarié vs l'emploi robotique. Le robot va-t-il réellement prendre la place d'une personne réelle dans 90 % des métiers, des usines et même dans l'espace public (gares, rues, accueils, etc.) ? Il y a un an, la SNCF commençait à déployer des robots, et à les expérimenter dans les espaces d'accueil de plusieurs gares mais aussi dans l'entretien, en précisant bien qu'ils n'allaient pas être utilisés au détriment des emplois. Mais au final, un robot coûte-t-il moins cher qu'un humain ? Et ne me dites pas oui immédiatement car de nombreux coûts doivent être inclus.

Cette effervescence vient aussi des progrès rapides de l'Intelligence artificielle (IA). Sur l'IA, nous entendons parler de l'IA forte et de l'IA faible. Cette dernière est largement déployée dans les systèmes actuels et la robotique. Le robot réalise des tâches répétitives et une absence, totale ou quasi, d'autonomie. Mais, nous voyons apparaître des robots plus autonomes, capables d'agir par eux-mêmes ; toutefois cette "intelligence" demeure limitée à ce qui a été programmé. Dans l'IA forte, le robot serait réellement indépendant et autonome, capable d'agir, de réagir, d'interagir, etc. Un peu comme un Terminator ou un Cylon hybride, à ne pas confondre avec les centurions par exemple.

Effectivement, dans ce dernier cas, des questions sociétales, éthiques, morales et philosophiques se poseront : quelle capacité juridique pour ces robots, seront-ils payés pour travailler, peuvent-ils être juridiquement émancipés, s'ils perçoivent un salaire doivent-ils payer des taxes et des impôts, peuvent-ils détenir un compte bancaire, acheter des marchandises, etc.

Certes, nous sommes encore dans l'imaginaire et la Science-Fiction, mais est-ce pour autant qu'il faut rester aveugle sur l'évolution rapide de l'IA forte et de la robotique ? Nous devons d'ores et déjà réfléchir à ces questions ; certains pays ont déjà adopté des lois robotiques ou en discutent. Oui, ces robots totalement indépendants et pensants ne sont pas encore là et n'existent pas, pas encore. Mais dans 20 ou 30 ans, où en serons-nous ? Les progrès dans l'IA, les algorithmes, les réseaux neuronaux sont considérables.

Parmi les étapes cruciales, il y a les blocs processeurs capables de penser, de rêver, de réfléchir, de créer des émotions ou encore la capacité illimitée à apprendre, voire, à générer ses propres souvenirs, émotions et programmes. Des cas de conscience sont soulevés par les voitures à conduite autonome. Faut-il sauver la vie des passagers du véhicule ou éviter un groupe de x personnes quitte à tuer le conducteur ? Que doit choisir l'IA ? Et qui est responsable en cas d'accident ?

Et si on frappe ou tue un robot ?

En été 2015, le robot auto-stoppeur HitchBOT fut victime d'une violente attaque alors qu'il était en vadrouille à Philadelphie, et pourtant, il avait pu se balader sans encombre dans le Canada. Acte d'une délinquance banale ou acte anti-robot ?

Si je détruis un robot d'une usine ou un robot d'un espace public, je serai sans doute accusé de vandalisme ou de destruction de bien public, avec dépôt de plainte contre moi. Si je croise un robot policier et que je le frappe ou le fais tomber ou l'insulte, qu'est-ce que je risque réellement ?

Autre question sociétale forte : quand un robot-tueur est utilisé par la police (ou l'armée) sur quelle base légale agit-il ? Est-ce l'humain qui le manipule qui est responsable ou le robot ?

Une question d'avenir

Notre société est-elle uniquement une société humaine ? On peut se poser la question avec le transhumanisme, l'homme augmenté. Est-il uniquement un humain ou un peu plus ? Et s'il est un peu plus qu'un humain, quelle est sa place réelle et concrète dans la société ? Faut-il créer un nouveau genre propre aux robots et aux humains augmentés ?

La pire des erreurs serait d'ignorer ces évolutions et de jouer (uniquement) sur la peur de la population. Diaboliser l'IA ou les robots est un réflexe simpliste. D'autres enjeux se jouent actuellement sur l'ADN, le séquençage, sa manipulation par les groupes industriels. En France, ce sujet n'est jamais abordé ou si peu. Et pourtant, ces travaux nous touchent directement. Et le risque d'un nouvel eugénisme n'est pas exclu. •

Histoire & futur de l'informatique (fin)

• Sylvain Abélard
(@abelar_s)
Architecte logiciel
Faveod

Les principes de l'innovation sont toujours les mêmes : une société, une économie se posent des problèmes. On imagine, applique, croise et assemble des siècles d'idées et techniques.

Dans les deux premières parties, nous avons fait un état de ces "lois de l'évolution" et de l'état de l'art actuel. Que peuvent bien nous promettre le futur proche et lointain ?

L'exercice de la prédiction est toujours difficile, et "le meilleur moyen de prédire le futur, c'est de l'inventer" (attribué selon les sources à Alan Kay ou Peter Drucker). Il est difficile de dire si un ensemble d'améliorations modestes, mises ensemble, constitue une révolution. Si l'innovation est la technique qui ouvre la voie, ou l'usage qui le confirme.

Énergie, transports, médecine... le numérique a-t-il déjà fait sa révolution, ou en garde-t-il quelques autres dans la manche ?

DEMAIN

Le futur proche

Comme toujours, il sera fait d'un croisement entre l'état de l'art, les nouvelles possibilités, la société et l'économie. Ces forces s'appliquent différemment selon votre contexte : les outils employés ne sont pas les mêmes partout. Pour autant, vivez-vous dans le passé ? Le futur ? Non, dans votre présent, vous êtes toujours dans un mélange des deux, fait de plusieurs arbitrages qui ont chacun eu un sens au moment où ils ont été faits.

Et chacun tente de ne pas se laisser distancer, essaie de capitaliser sur ses succès passés (qui ne veut pas dire investissements passés, la nuance est douloureuse), naviguer sur les modes, et anticiper le futur. Que pourront donner les croisements des mondes du développeur, du manager et de la mode, à court, moyen ou long terme ?

Mieux coder, moins coder

"The Best Code is No Code" : le meilleur code, c'est celui qui n'existe pas. Le code qui n'est pas écrit c'est celui qui ne bogue pas, ne doit pas être maintenu...

Chaque génération apportait son confort, son sucre syntaxique, ses moyens d'expression plus puissants, concis et flexibles... et ses levées de bouclier par les "vrais programmeurs".

En ayant démystifié ce qui dirige l'innovation, il semble impossible que la manière d'écrire du code, dans un éditeur de texte puis dans un IDE (en français EDI, Environnement de Développement Intégré) ne subisse pas elle aussi des mutations cosmétiques (langages, outils de formatage et validation) à structurelles.

Mieux exploiter

"Ce qui est douloureux, faites-le souvent, ça ira mieux" : cet adage qui semble évident s'applique aux tests, au déploiement continu, mais aussi à la communication avec le client que prônent les méthodes Agile. On peut bien entendu outiller nos outils, mesurer ce qui va mieux ou moins bien, en causant si possible un minimum de surcoût... sans quoi toute l'opération perd son sens, et sera mise au placard jusqu'à ce qu'une autre innovation permette de ressusciter l'idée. On peut s'attendre, espérons-le, à des IDEs de plus en plus intelligents.

Attention toutefois à l'effet "magie" : les développeurs n'ayant plus à apprendre les bases, sauront-ils encore développer ou sauront-ils quels impacts "plus bas" dans le mille-feuilles de couches architecturales, ils sont en train de causer par une ligne de code qu'ils insèrent dans une conception un peu complexe ?

Il est encore possible de gagner quelques pourcents, mais jusqu'à quelques ordres de magnitude en performance ? Les langages dits "lents" comme JavaScript ou Ruby voient les nouvelles versions embarquer des innovations comme des VM, les compilations JIT et AOT (JS V8, Ruby 1.9 et bientôt 3x3 pour "Ruby 3, 3 fois plus vite"), voire la traduction ou le portage vers d'autres plateformes (LLVM ou JVM), levant encore des barrières à leur utilisation.

On peut enfin rêver tout simplement d'une utilisation appropriée et plus vaste de choses qui existent depuis longtemps : Erlang, langage conçu en 1986 chez Ericsson pour gérer les affres du réseau et de la téléphonie (concurrence, tolérance, disponibilité...) se retrouve ainsi au devant de la scène en 2016 car chacune de ses propriétés le rend plus désirable pour le multi-processeur, l'informatique distribuée, les applications Web.

Le code n'est pas (encore ?) de la science

L'université ne semble que rarement écoutée, et certes quand elle l'est, ses conseils ne sont pas toujours pragmatiques aux yeux des développeurs en poste.

Parfois, les solutions qui en sortent peuvent ne pas avoir d'application immédiate. Mais surtout, l'académie n'est pas très "vendeuse" : les développeurs et décideurs se moqueront d'elle et des modes qu'elle suit, et s'offusqueront quand elle s'attaque à leurs modes fétiches. La divergence était comme souvent cachée sous notre nez : on croit alors travailler sur quelque chose de scientifique, très clair et rationnel comme une formule de mathématiques, alors que tous les aspects, des besoins à la réponse, sont encadrés par des humains qui devinent plus qu'ils ne cherchent.

Mais les outils informatiques sont... outillés eux aussi ! On a des outils d'analyse statique (complexité, performance, "style guides", sécurité) et du monitoring dynamique (performance et sécurité de nouveau, erreurs, etc).

Remettre cette science au coeur du process

Ne serait-il pas intéressant de profiter de ces métriques ? Quelles instructions, quelle manière de coder, quel langage donnent à long terme moins d'erreurs, de bugs et de maintenance ? Au sein de votre projet et de votre équipe, qui est performant sur quel sujet et où se trouvent le plus de changements, de remise en cause de l'architecture, d'interventions et de bugs ? Pour l'instant, on en reste à des méthodes plus simples : revues de code, programmation en binôme, fichiers journaux... On peut saluer l'outil Science, publié par GitHub, qui fait tourner les nouvelles versions en parallèle à la production actuelle pour comparer les retours et la performance, leur donnant une sérénité accrue pour leurs déploiements continus.

Fini l'empirique, vive la science !

APRÈS-DEMAIN

La prochaine révolution

L'informatique a révolutionné tant d'autres secteurs, il est illusoire de croire qu'elle ne subira jamais de révolution.

Il est tentant de savoir ou deviner d'où viendra le prochain changement radical pour changer le paysage, digne du nom de révolution : il y a de l'argent à se faire à surfer sur la vague, et de l'argent à perdre quand on se fait "disrupter" (subir un changement, une innovation de rupture) ou quand on n'a pas su anticiper.

"Les prédictions sont difficiles, surtout dans le futur", a fortiori dans les 10 ans ou davantage.

Langages et outils

Tout d'abord, on peut trouver des améliorations incrémentales à tous les aspects qui nous rendent

aujourd'hui la vie difficile : performance, productivité, confort, réduction des erreurs... Gagner 10% par ci, 5% par là, cela s'additionne !
Cela permet surtout réduire les goulots d'étranglement, de voir ce qui devient la nouvelle ressource limitante, et, une fois les petits soucis résolus et oubliés, d'appliquer ses efforts à autre chose de plus douloureux ou plus important. C'est le premier des bienfaits de l'amélioration continue.

Métiers et méthodes

De même, toute modification dans la manière de travailler, de collaborer, de partager les objectifs et d'y répondre ensemble est une opportunité de s'améliorer, marginalement puis significativement.
Les générations qui arrivent désormais en entreprise sont nées et ont grandi dans un monde numérique. L'outil fait moins peur, il est mieux maîtrisé, peut-être moins compris dans le détail aussi.
Toute organisation a deux natures : celle de se maîtriser, rationaliser ses process, et celle d'évoluer en continu pour répondre ou anticiper les contraintes.
Entre agitation et calcification, sachons accepter le changement et même en tirer profit. On ne pourrait être prêts à accueillir une innovation salvatrice, si tout changement est d'abord vu comme une source de peur et de douleur !

Big Data et Machine Learning

Après des années de Data Mining, et avant le Machine Learning, le Big Data semble être la tendance du moment.
Il a fallu d'abord avoir les données à disposition, et ça n'était pas simple, puis chercher les parties utiles, et enfin les organiser. Parfois, il faut avant tout essayer de faire sens d'une masse nébuleuse qu'on ne comprend pas.
Très souvent, ce qu'on croit être Big Data tiendrait très bien dans les bases de données du marché. Très souvent, on a juste besoin de classifier la donnée. Pour ceux qui vont vraiment plus loin, le Machine Learning permet une classification automatique, permettant de ressortir les signaux forts.
Numériser l'entreprise, c'est comme découvrir son ADN : trouver les blocs qui la constituent. Qualifier la donnée d'entreprise, c'est comme reclasser l'arbre du vivant : un chantier gigantesque, toutefois facilité par l'analyse des blocs constitutifs. Comprendre l'entreprise, c'est comme découvrir les prédispositions par analyse statistique : disponible à celui qui a, travaille et questionne correctement la donnée. Mais transformer l'entreprise ? C'est comme proposer des thérapies géniques : c'est possible aujourd'hui, ça promet des miracles demain, mais il faut d'abord maîtriser les étapes d'avant.
Mais la génétique a un avantage clé : ces données sont par nature structurées, ce qui n'est pas le cas de l'entreprise. De plus, les volumes explosent, et on ne souhaite plus avoir simplement un rapport du trimestre précédent, mais les tendances en temps réel. La valeur principale reste alors de se poser les bonnes questions, et se donner les moyens d'agir en fonction des réponses !

Informatique quantique

L'informatique quantique est la promesse de nouvelles portes logiques, de n'être plus limité par des bits 0 ou 1 mais de pouvoir avoir d'autres états.
Il faut alors séparer deux choses : soit on a des problèmes adaptés à ce genre de logique (auquel cas, vous êtes probablement au courant), soit on essaie simplement de l'utiliser pour travailler comme avant, mais plus vite : en améliorant les performances tout en bas de ce mille-feuilles de technologies, on espère des gains exponentiels. Un tel gain de puissance de calcul et de vitesse serait une innovation marquante jusque dans les usages.
Cela ouvrirait des portes à toutes les tâches intensives en calcul qui ne sont aujourd'hui pas encore assez praticables : calcul scientifique (attendons-nous à des découvertes), simulations, 3D...
... et attaques en "force brute", comme craquer des mots de passe. La réponse devra alors venir des fondements théoriques (mathématiques) et des usages.

IA

Récemment, de nombreux grands noms ont appelé à la plus grande attention sur les sujets liés à l'intelligence artificielle : avenir de l'humanité ? Fin de l'espèce ?
La réponse est probablement entre les deux, ou plutôt comme avec tous les outils, le sens de la chose sera donné par celui qui le manie.
Le problème de l'Intelligence Artificielle n'est pas Artificielle, c'est Intelligence : si l'IA est quelque chose qui est différent de notre intelligence, saurons-nous reconnaître qu'il s'agit de quelque chose d'intelligent, et non de décisions absurdes et aléatoires ? On peut toutefois, comme dans le monde du jeu de Go, bénéficier grandement d'une puissance de calcul, de classification et d'approches que les humains n'avaient jamais tentées.
Ainsi, plus le problème sera défini et maîtrisé, plus il y a de chances qu'une certaine forme d'IA, disons un assistant évolué, nous permette d'améliorer significativement tel ou tel aspect de notre travail ou de notre société.

Management de l'innovation

Dans tous les cas, chacune de ces expériences se heurtera à la résistance au changement. À tort ou à raison, changer c'est risquer. Certaines entreprises ou plutôt certaines équipes savent laisser la bride suffisamment lâche pour laisser les nouveautés émerger. D'autres en meurent ou périclitent, avec des citations qui font rire des années après : Bill Gates ne croyait pas à Internet, Kodak refusait l'idée d'appareil photo numérique, mal alignée avec son business de vente de pellicules... À l'inverse, Steve Jobs souhaitait voir ses propres nouveautés cannibaliser ses produits existants, de peur qu'un concurrent ne le fasse. Mais bien évidemment, cela a un coût.
Pour les autres, seule la menace d'une crise imminente suffit : Bletchley Park n'a probablement existé que par la menace de la guerre : terrible, visible, immédiate. Combien ne découvrent que trop tard qu'il aurait fallu évoluer voilà dix ans ? Combien utilisent cette justification pour se ruiner dans des expérimentations instables ?

Management de la mode

Comme toujours, difficile de savoir si une mode prendra, ou durera.
Dans les deux cas, elle restera rarement sous la même forme : la partie de ses enseignements qui auront été les plus entendus persisteront probablement en tant que "bonnes pratiques", la plupart des équipes et entreprises gagneront en maturité sur le sujet puisqu'il a été médiatisé, et la fuite en avant des buzzwords et autres nouvelles tendances trouvera probablement une approche similaire et un nom nouveau sous lequel tenter de relancer une mode. Cas d'école : client-serveur, client léger, ASP, SaaS, Cloud, Serverless.
Chacun correspond à une itération de plus sur le même concept, et pourtant, presque chacune de ces vagues a été séparée par une vague dans le mouvement inverse : client lourd, RDA (Rich Desktop Application), applications mobiles...
L'informatique est cyclique, et ne pas comprendre l'histoire, c'est être condamné à la répéter.

Et vous ?

Notre secteur et nos métiers vont, et doivent, changer. Ceux qui survivent ne sont pas les plus forts de tel ou tel domaine, à tel moment, mais bien les plus adaptables.
Depuis des milliers d'années, notre espèce évolue et notre cerveau est probablement l'outil le plus fantastique, mais aussi le plus faillible. Nous sommes prêts à oublier le rationnel et à partir dans l'émotionnel au quart de tour.
Le numérique a de plus en plus de place dans notre vie, il est partout. Il ne tient qu'à nous d'en faire un outil de qualité, un assistant stable et fiable, ou un outil qui ne ferait que multiplier nos folies. •

Le **bestiaire** de l'Open Source : vive les mascottes !

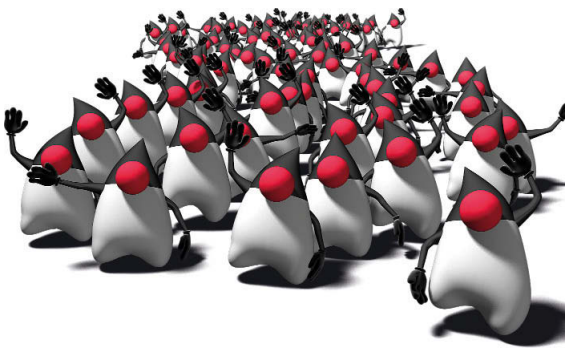
• Christophe Chervy
Open Goodies, elePHPant.com

• Christophe Villeneuve
Consultant IT pour Ausy, Mozilla Reps, auteur du livre "Drupal avancé" aux éditions Eyrolles et auteur aux Editions ENI, membre des Teams DrupalFR, AFUP, LeMug.fr (MySQL/MariaDB User Group FR), elePHPant.com, Drupagora...

À l'inverse des très sérieuses et représentatives fenêtres, pommes croquées ou lettres stylisées façon années 80, les projets Open Source ont toujours préféré utiliser comme mascottes tout un bestiaire d'animaux, réels, fabuleux ou créés spécifiquement. Le choix de ces animaux est rarement anodin et est censé refléter les qualités intrinsèques de chacun de leurs logiciels. De nombreuses légendes ont été répandues par ceux qui ont choisi ou conçu ces mascottes. Nous ferons ici le point sur les mascottes que nous connaissons tous et la petite histoire de leur naissance.

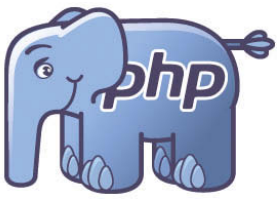
Le Duke de Java

L'un des langages les plus répandus, en Open Source depuis 2006, est représenté par une tasse à café ("Java" est l'équivalent de notre "Kawa"). Mais celui qui personnifie réellement le langage est Duke, un petit personnage de cartoon. Créé par Joe Palrang, qui faisait partie de l'équipe de designers des débuts, Duke était un agent qui devait aider l'utilisateur dans ses tâches. Par la suite, Sun et Oracle ont continué à l'utiliser dans leurs conférences et leur communication. Le petit personnage sympathique ayant une nouvelle version à chaque conférence JavaOne.

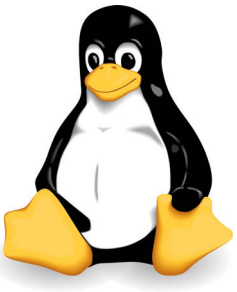


L'éléPHPant de PHP

Le très connu et adoré éléPHPant, créé en 1998 par Vincent Pontier, designer français (aka El Roubio), s'est, comme de nombreuses mascottes, imposé aux développeurs. Créé à l'origine pour un ami fan du langage, le petit animal a fait son chemin auprès d'une communauté grandissante pour finalement être adopté officiellement 7 ans plus tard et être décliné en version peluche en 2006. L'éléPHPant avait été choisi pour sa force et sa fiabilité, et bien sûr, pour le jeu de mot qu'il permet de faire. Il est toujours un mastodonte du Web 20 ans plus tard.



Le Tux de Linux



Probablement la mascotte la plus connue de tout l'écosystème Open Source. La notoriété du fameux manchot Tux a même dépassé les frontières de la communauté Linux. Dessiné avec GIMP en 1996 par Larry Ewing, le manchot avait été choisi par Linus Torvalds lui-même. Les raisons de ce choix sont en revanche plus obscures. Linus se plaisant à raconter des histoires différentes, nous retiendrons la plus amusante et improbable anecdote : la morsure qu'il aurait reçue par un manchot lors d'une visite au zoo. Après maintes discussions avec la communauté, Linus a donc demandé un manchot légèrement en surpoids, avec un sourire béat, comme après une trop grosse consommation de harengs. Son nom, TUX, comme « Torvalds UniX » colle tout autant au manchot puisque c'est l'abréviation de "tuxedo", ("smoking" en anglais) qui rappelle les couleurs de l'animal.

Le Druplicon de Drupal

La fameuse goutte du CMS belge Drupal. Cette fois, le logo est en rapport avec le nom du logiciel, même si le rapport entre les deux est dû au hasard. Au départ, le logiciel devait se nommer Dorp ("village" en néerlandais) qui, par une maladresse lors de l'enregistrement du nom de domaine, est devenu Drop ("goutte" en anglais). De là, la forme générale était trouvée, ne restait plus qu'à la personnaliser en lui donnant un visage. Les yeux sont en fait deux gouttes qui forment le symbole infini. Et l'ensemble est une goutte dans un cercle et un cercle dans une goutte.

L'Octocat de GitHub

GitHub a trouvé son logo sur iStock ! La mascotte de GitHub a une histoire peu banale : Simon Oxley, designer britannique, avait déposé sur la plateforme de photos plusieurs créations. L'une allait devenir le premier logo de Twitter et la seconde, le logo officiel de GitHub. Comme les règles d'IStock interdisent aux sociétés d'utiliser les créations disponibles sur la plateforme comme logo, GitHUb et Simon Oxley négocièrent donc en dehors de celle-ci, et la créature mi-chat mi-pieuvre reçut le succès qu'on lui connaît.

Le GNU de GNU

Basé sur le fameux acronyme récursif, "GNU's Not Unix!" créé par Richard Stallman en 1983. Dessiné à l'origine par Etienne Suvasa, une version plus moderne refaite par Aurelio Heckert lui a été préférée. La relation entre le nom du

projet GNU et sa représentation graphique est évidente, mais les explications de Stallman sur son choix de ce nom sont plus difficiles à cerner. Il mentionne néanmoins une chanson intitulée "The Gnu" où le duo britannique Flanders and Swann fait tout un travail comique sur les lettres non-prononcées de certains mots anglais. Stallman a clairement voulu jouer sur les mots en faisant de GNU un acronyme récursif.

Firefox en panda, oiseau et renard…

Chez Mozilla, tous les produits proposés par la fondation sont représentés par un animal. Le plus connu étant le navigateur Firefox, qui a, par ailleurs, subi le plus grand nombre de changements. À l'origine, celui-ci se nommait "phoenix". Ce nom a dû être abandonné car appartenant à Phoenix Technologies. Il a été remplacé par "firebird" dont le visuel était acceptable pour l'époque mais ne remportait pas un grand succès. De plus, ce nom était déjà utilisé par un autre logiciel libre. La pression de la communauté a forcé le changement.

C'est en 2004, que le nom Firefox est apparu et a été déposé comme marque aux Etats-Unis. Par la suite, c'est l'identité visuelle et l'interprétation du mot "Firefox" qui a subi de nombreuses évolutions.

La première version du logo a dévoilé un animal représenté par un renard stylisé en panda roux, qui est un petit animal d'origine asiatique dont le surnom anglais signifiait panda rouge. Au fil des années, celui-ci s'est vu transformer pour augmenter sa notoriété en laissant sa place au renard de feu.

L'animal a vu apparaître une longue queue lors du lancement du système embarqué pour mobile appelé Firefox OS.

Sakila, le dauphin MySQL

La mascotte de la fameuse base de données est un dauphin sauteur qui symbolise la vitesse, la puissance, la précision et la "bonne nature". Quatre caractéristiques que Monty, le créateur de la base, a toujours mises en avant et qui s'appliquent également à la communauté.

Ce dauphin s'appelle "Sakila", prénom féminin d'origine Swazi (du Swaziland) donné à la suite d'un concours organisé par MySQL pour nommer sa mascotte.

Cet article aurait pu vous parler de dizaines d'autres mascottes comme l'otarie de la base de données de MariaDB, le serpent du langage de Python, l'oiseau tonnerre de "Thunderbird" de Mozilla, ou encore le dromadaire du langage Perl. Chacun a sa propre histoire et parfois le lien est plus que lointain entre le logiciel et sa mascotte. Malgré tout, ces petits animaux réels ou imaginaires sont un véritable allié pour les développeurs, et ils renforcent le sentiment d'appartenance à une communauté très souvent virtuelle. •

Ninjacat : pourquoi Microsoft est cool de nouveau (dixit The Verge)

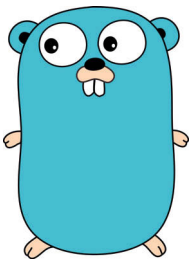
Un cat ninja, une licorne crachant du feu. Il n'en fallait pas plus pour rendre ce Ninjacat très sympathique; la communauté l'a immédiatement adopté. Comme le raconte le post sur les origines de cette surprenante image, rédigé par Raymond Chen, l'origine de l'histoire est le hasard lié aux origines de Windows 10 et à sa présentation officielle. L'idée était de faire une blague en utilisant des animaux, un arc-en-ciel, une licorne. L'inspiration est venue du très amusant Welcome to the Internet de Jason Heuser. L'image s'est fixée et l'équipe a sondé autour d'elle pour savoir si le ninjacat plaisait. Réponse : oh que oui ! Et en janvier 2015, avec l'annonce officielle de Windows 10, le ninjacat a commencé à envahir les communautés.

Pour nous, la mascotte la plus cool de Microsoft depuis… ben, depuis toujours. Désolé Clippy.

Clarus, le chien-vache d'Apple

Pour les utilisateurs Mac, Clarus était une des images les plus familières des premiers Mac et est rapidement devenue la mascotte des utilisateurs. Rapidement, Clarus est devenu iconique dans la communauté, aussi populaire que le Sad Mac. La mascotte était même décrite dans la Technote du 31 d'avril 1986. Peu à peu, une histoire c'est construite autour de lui et son fameux cri était "moof". Clarus a eu droit son modèle grandeur nature dans les jardins de Cupertino, retiré en 1998.

Gopher du langage Go



Certains y voient une marmotte, d'autres un castor. La drôle de mascotte du langage Go surprend. Avant d'être l'icône de Go, il s'agissait d'un des logos de la radio WFMU puis il est réapparu chez Bell Labs. Puis en 2009, la joyeuse bouille a été adaptée par l'open source du langage Go.

Le démon de BSD

BSD Daemon est une des mascottes les plus connues du monde open source. Il s'agit d'un petit diablotin rouge avec fourche. Daemon est une référence aux systèmes et non au Démon. Une variante a été créée spécialement pour la couche open source de macOS, Darwin. Il porte le nom de Hexley.

Par contre le projet OpenBSD utilise lui le surprenant et très jaune poisson-hérisson.

L'éléphant de PostgreSQL

L'éléphant est une valeur sûre de l'open surce. PostgreSQL utilise une imposante tête d'éléphant avec ses imposantes défenses. La mascotte fut introduire en 1997. Autre éléphant, cette fois-ci jaune, celui du projet Hadoop.

Le dragon de LLVM

Voici une des plus originales mascottes de l'open source : le wyvern du projet LLVM. Le wyvern est une variété de dragon. Il est stylisé. Ses premières apparitions datent de 1977. Autre dragon, celui du projet KDE même si à l'origine, il s'agit de Kandalf, une sorte de sorcier ressemblant à Gandalf.

• François Tonic

Nous recrutons des formateurs !

Vous êtes développeuse / développeur et vous rêvez d'être formatrice / eur sur votre langage ou technologie ?

OpenBrain IT et Programmez! s'associent pour proposer dès le printemps 2017 des formations techniques partout en France.

Venez rejoindre une communauté jeune et dynamique.

Envoyez dès aujourd'hui votre C.V., vos compétences : formateur@openbrainit.com

OpenBrain IT

PROGRAMMEZ!
le magazine des développeurs

Oculus Touch + VR

• Romain François
Membre de la virtual association
& **Jimmy Fenollar**

Début décembre 2016 sortaient les "Touchs", les contrôleurs si attendus pour les possesseurs d'un Oculus Rift CV1. Qu'est-ce que ces nouveaux contrôleurs apportent au casque de Facebook par rapport à son concurrent direct le HTC Vive ? Quel impact pour la RV et son développement ?

Présentation des Touchs

Les touchs sont disponibles pour la somme de 199 . Question ergonomie, ils sont plus légers et leur poids est mieux réparti que les contrôleurs du Vive. La forme épouse parfaitement la main permettant une ergonomie quasiment parfaite. Il faut noter également que contrairement au Vive, ils ne sont pas interchangeables. En effet chaque Touch a une forme spécifique pour épouser chaque main. Lors de la première prise en main, on retrouve une sensation agréable au toucher, très proche de celle de la manette Xbox : le design et l'ergonomie ont été très bien pensés, la prise est intuitive, on retrouve la marque de fabrique de Microsoft (ndlr: partenaire de Facebook) dans le design des touchs.

Chaque contrôleur dispose d'ailleurs d'un stick et de plusieurs boutons, ainsi qu'une fonction de tracking à 3 états pour les doigts : touché, pressé, relevé. On peut noter une très légère latence lors de la mise à jour visuelle (virtuellement) des différents états.

Les touchs ne disposent pas de batterie mais fonctionnent avec une pile AA chacun (fournies lors de l'achat). Choix qui peut paraître étonnant au début mais qui semble pertinent quand on voit que certains contrôleurs du Vive perdent la charge au bout de 200 heures d'utilisation. Pas d'inquiétude, on peut raisonnablement s'attendre à des batteries supplémentaires à venir pour les plus réticents.

Néanmoins la première configuration des contrôleurs avec le software du casque relève d'une expérience bien moins agréable. Etant livré avec un capteur Oculus sensor, vous devrez positionner les deux sensors très précisément (quasiment à l'angle près) pour pouvoir profiter des touchs dans des expériences à 180°, recommandées par Facebook. Pour pouvoir profiter d'une expérience roomscale debout à 360°, Un troisième sensor devra être acheté au prix de 89 , occupant 3 ports USB et rendant la configuration plus chère qu'un Vive actuellement.

Concernant le contenu, Facebook a annoncé pas moins de 53 jeux sur le store Oculus compatibles touch dès le lancement. Valve a également mis à jour sa plateforme SteamVR pour accueillir les nouveaux contrôleurs.

HTC VIVE ou Oculus CV1 ?

On peut maintenant comparer rapidement les deux casques. Grâce aux touchs, le Rift dispose désormais d'une zone de roomscale à 180° (voire 360° avec les 3 caméras) sur du 2m², cependant cette zone reste bien inférieure aux 4m² minimum recommandés par SteamVR et cette configuration occupe 2 à 3 ports USB.

Le système de tracking est quant à lui totalement différent : sur le CV1 c'est le casque qui émet, via un nuage de rayons IR. On perd donc toute détection en sortant du FOV (champs de vision) des Oculus sensors. A contrario, le Vive est un système récepteur, l'occlusion est minime grâce aux balayages des stations, et donc, il n'y a quasiment aucun problème de détection.

Concernant l'ergonomie et le confort, comme décrit en début d'article, Oculus, avec ses touchs marque un très beau point face à HTC.

C'est pour le moment, sans doute, les meilleurs contrôleurs du marché. On relève également un meilleur confort sur la durée avec Oculus, le poids du casque étant mieux réparti que celui du Vive, trop porté sur l'avant.
Les lentilles sont légèrement différentes entre les deux systèmes, avec un léger flou gommant la grille et les pixels pour le CV1 et un rendu plus piqué pour le Vive. Le CV1 dispose également d'un très bon micro intégré alors que le Vive dispose d'un module bluetooth et d'une caméra frontale.
Enfin au niveau du contenu, le Vive bénéficie de la plateforme Steam ainsi que de sa communauté, le CV1 ayant l'Oculus store de son côté. Malgré une compatibilité annoncée de la part des deux constructeurs, un grand nombre de jeux techniquement compatibles sont injouables en termes de gameplay avec l'un ou l'autre des contrôleurs disponibles. Pourquoi ? Tout simplement car tous les jeux n'ont pas été pensés et développés pour utiliser à la fois les touchs et les contrôleurs du Vive, contrôleurs qui sont foncièrement différents sur leur utilisation en termes de gameplay.
On pense notamment à "Paddle Up" sur SteamVR, injouable actuellement avec des touchs, et pourtant bien compatible. La seule solution actuellement qui ne semble pas en être une est de créer deux jeux pour les deux différents stores, comme c'est le cas de "Pole nation" disponible sur SteamVR et de "Sport Bar VR" disponible sur Oculus Store. Cette séparation numérique pose aussi problème au niveau de certains contenus comme "Arizona Sunshine", titre sorti récemment sur les deux stores. En effet si vous achetez le jeu sur l'Oculus store, vous ne pourrez y jouer que sur ce dernier, alors qu'en l'achetant sur Steam vous aurez la possibilité d'y jouer avec un Vive ou un Oculus. Pourtant ici le jeu est entièrement compatible et fonctionnel avec les deux types de contrôleurs.
Au terme de cette comparaison, on peut résumer qu'il n'y a pas à proprement parler de meilleur casque de réalité virtuelle pour le moment. Que ce soit le CV1 ou le Vive, chacun a ses spécificités, avantages, et défauts. Le choix de l'un ou de l'autre portera plus sur vos préférences de jeux, d'espaces ainsi que du contenu, de sa qualité sur la durée et de son rapport qualité prix.

Quelle situation et quel avenir pour la VR ?

Pour le moment, le monde de la VR est en plein essor, il y a plusieurs concurrents, chacun développant sa propre technologie. La communauté est grosso modo divisée en deux : le clan Oculus et le clan Vive. Les deux plateformes proposant chacune sa bibliothèque. Les compatibilités sont variées ainsi que les prix en fonction des titres. Sur plusieurs points cette situation est avantageuse car elle offre une grande compétitivité et beaucoup d'innovation. Cependant sur long terme, on constate qu'il n'y pas de standard réel pour la réalité virtuelle actuellement.
Les Oculus touchs ne sont pas compatibles à 100% avec les jeux annoncés car ces jeux n'ont pas été pensés pour les utiliser et n'incorporent donc aucune interface viable. Les différents stores sont séparés et le cross plateforme est rendu difficile (on doit installer ReVive pour jouer aux jeux Steam avec un rift). Le collectif OpenVR pourrait être une solution pour uniformiser toutes ces technologies afin de toutes les faire avancer vers un même chemin. Le groupe Khronos (ndlr: les créateurs d'OpenGL), constitué des plus grands comme Google, HTC, Valve, OSVR, Oculus est également une idée. Cependant Microsoft reste absent de ce groupe, pourtant très présent dans le milieu. On peut penser notamment à la technologie "Microsoft Windows Holographic", lancée initialement pour le Hololens. Microsoft, veut ainsi démocratiser la réalité virtuelle et la réalité augmentée. Holographic sera compatible pour tous les possesseurs de casques Oculus et HTC et disponible sur toutes les machines disposant de Windows 10. Des casques OEM Microsoft utilisant la même technologie et construits par ses partenaires tels que LG, Acer ou encore Lenovo, ont été également présentés lors du CES 2017 à Vegas.

Ainsi on pourrait se retrouver avec un standard Vive/Oculus contre un standard Microsoft, nous rappelant le combat d'aujourd'hui.
Si chaque casque apporte ses spécificités, il ne faut pas oublier d'uniformiser les technologies, de les rapprocher voire de les combiner pour offrir au public, à la communauté, un résultat performant et générique.
Dans ce domaine on ne peut que féliciter les efforts de Unity et d'Unreal Engine. Au fil des années leurs moteurs se sont affinés, améliorés et proposent désormais des solutions rapides et faciles pour le développement de jeux vidéos VR. Il reste encore du chemin avant d'arriver à une réalité virtuelle mature et efficace mais la plupart des outils et technologies arrivent à grands pas !

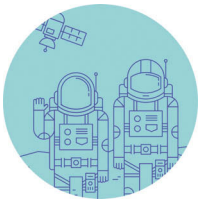
SHARED REALITY : HTC VIVE ET MICROSOFT HOLOLENS DANS LA MÊME RÉALITÉ

Un développeur, Drew Gottlieb, a conçu un prototype surprenant utilisant HTC Vive et Hololens. Il a appelé le projet « shared reality ». L'idée est simple : créer un monde mixte partagé entre un Vive et un Hololens. Cela signifie que les deux utilisateurs partagent et interagissent dans la même réalité. Ils peuvent utiliser les objets, les manipuler, collaborer ensemble. C'est assez impressionnant à voir dans les démos proposées. Et surtout, cela montre le potentiel de cette technique. Le Vive est une technologie de réalité virtuelle et non mixte.
Présentation du projet : http://drewgottlieb.net/2017/01/31/mixing-reality-with-vr.html
Projet sur Github : https://github.com/dag10/HoloViveObserver F.T.

Les Progressive Web Apps

Partie 3

La seconde chance du Web dans l'univers du mobile



Suite et fin de notre dossier sur les PWA

• Simon Gombaud
Directeur artistique **Made on Mars**

• Thomas Foricher
Directeur Technique **Made on Mars**

Favorise le réengagement

Sûrement l'une des fonctionnalités qui manquait le plus au Web face aux applications natives - les Push Notifications. Ce système est enfin disponible pour le Web via Push API et Notification API. Le Push permet au serveur de communiquer avec l'application et plus précisément le service worker. La Notification API permet au SW d'afficher la notification.
Dans notre application d'exemple nous allons intégrer les push notifications pour indiquer aux utilisateurs que le compteur a atteint des jalons particuliers, disons à chaque centaine tout rond. Tout d'abord, nous allons créer un projet dans Firebase Developer Console pour récupérer des clés afin d'utiliser l'API d'envoi de Push Notification de Google.

- Cliquez sur "créer un projet" et choisissez le nom que vous voulez
- Allez dans paramètres du projet en haut à gauche.
- Cliquez sur l'onglet "Cloud Messaging"
- Conservez votre clé de serveur ainsi que l'ID de l'expéditeur qui vont nous permettre de communiquer avec le système de messaging Firebase de Google.

Nous allons ajouter dans le code du serveur Web une méthode pour envoyer les notifications et un test qui le déclenche lorsque le compteur atteint des jalons de centaines (100, 200, 300, etc…).
Tout d'abord nous allons devoir ajouter un nouveau fichier de base de données pour enregistrer les identifiants des terminaux afin de leur envoyer les PNs. On créé donc un nouveau fichier clients.json à la racine du projet avec un tableau vide à l'intérieur pour le moment.

```
[]
```

Maintenant il ne nous reste qu'à ajouter une route d'API pour ajouter les identifiants des terminaux dans le fichier clients.json De la même manière que pour le compteur, on écrit directement sur le disque via le module fs.

```
app.get('/client/:id', function(req, res){
  var clientId = req.params.id || undefined;
  if(!clientId){
    res.status(500).json({status: 'err'});
    return;
  }
  fs.readFile('./clients.json', 'utf8', function(err, data){
    if (err) {
      res.status(500).json({status: 'err'});
      return;
    }
    var clients = JSON.parse(data);
    if(clients.indexOf(clientId) == -1){
      clients.push(clientId);
      fs.writeFile('./clients.json', JSON.stringify(clients, null, 4), function(err, data){
        if (err) {
          console.log(err);
        }
      });
    }
    res.json({status: 'ok'});
  });
});
```

On ajoute le module request qui va nous permettre de communiquer avec Firebase. On l'installe via la commande :

```
npm install request --save
```

Puis dans notre fichier app.js, on inclut le module et on spécifie notre clé serveur :

```
var request = require('request');
var gcmAPIendpoint = 'https://fcm.googleapis.com/fcm/send';
var gcmAPIKey = 'INSERER_VOTRE_CLE_SERVEUR_ICI';
```

Plus bas avant l'écriture dans le fichier clients.json, si la valeur du compteur est divisible par 100, on envoie une PN. Voir code complet sur le site de Programmez! Côté client, plusieurs choses sont à faire. Tout d'abord, dans le fichier manifest.json, on indique le gcm_sender_id que nous avons copié sur Firebase.

```
{
  "name": "Love love love",
  "gcm_sender_id": "INSERER_VOTRE_SENDER_ID"
}
```

N'oubliez pas d'ajouter le lien au manifest.json dans la section HEAD votre fichier public/index.html

```
<link rel="manifest" href="manifest.json">
```

Dans le fichier public/js/app.js nous allons ajouter l'interaction pour obtenir l'autorisation d'envoi de push notification
On ajoute des variables globales pour suivre si l'utilisateur accepte les PNs

```
var subscribeButton = document.getElementById('subscribeBtn');
var sub;
var isSubscribed = false;
```

On ajoute un écouteur sur le bouton d'inscription puis en fonction de l'état inscription, inscrit ou désinscrit, on informe le SW et on modifie l'interface de l'utilisateur.

```
// Au clic sur le bouton on inscrit ou désinscrit l'utilisateur
subscribeButton.addEventListener('click', function() {
  if (isSubscribed) {
    unsubscribe();
  } else {
    subscribe();
  }
});

function subscribe() {
```

```
  reg.pushManager.subscribe({userVisibleOnly: true}).
  then(function(pushSubscription) {
    sub = pushSubscription;
    var clientId = pushSubscription.endpoint.split('/').pop();
    xhr.onreadystatechange = function () {
      if (xhr.readyState === 4 && xhr.status === 200) {
        var response = JSON.parse(xhr.responseText);
        if(response.status === 'ok'){
          subscribeButton.textContent = 'Bloquer PNs';
          isSubscribed = true;
        }
      }
    };
    xhr.open('GET', '/client/'+clientId);
    xhr.send();
  });
}

function unsubscribe() {
  sub.unsubscribe().then(function(event) {
    subscribeButton.textContent = 'Recevoir PNs';
    isSubscribed = false;
  }).catch(function(error) {
    subscribeButton.textContent = 'Recevoir PNs';
  });
}
```

Une dernière chose à ajouter dans le fichier public/js/app.js est lors de l'enregistrement du service worker. Si l'enregistrement s'est passé avec succès, on débloque le bouton d'inscription aux PNs et on appelle la fonction subscribe pour proposer à l'utilisateur de recevoir les notifications. Le code devient comme ci-dessous :

```
// On regarde si le service worker est supporté par le navigateur
if ('serviceWorker' in navigator) {
  // On enregistre notre script sw.js
  navigator.serviceWorker.register('sw.js').then(function() {
    return navigator.serviceWorker.ready;
  }).then(function(serviceWorkerRegistration) {
    reg = serviceWorkerRegistration;
    subscribeButton.disabled = false;
    subscribe();
    console.log('Enregistrement du SW avec succès.');
  }).catch(function(error) {
    console.log('Une erreur est survenue.', error);
  });
}
```

Enfin dans le service worker nous rajoutons les écouteurs pour les évènements de push c'est-à-dire lorsque le serveur Firebase envoie l'information qu'une push notification est disponible.

```
self.addEventListener('push', function(event) {
  var title = "Nous avons atteint un nouveau jalon.";
  var body = "Viens voir le compteur s'affole !";
  var icon = 'images/icons/icon-android-152x152.png';
  event.waitUntil(
    self.registration.showNotification(title, {
      'body': body,
      'icon': icon
    }));
});
```

Vous noterez que nous indiquons le texte de la Push Notification en dur. Il est possible de passer des données entre notre serveur et le service worker qui nous permettraient d'ajouter la valeur du compteur. Ceci ajoute une complexité qu'il serait difficile de couvrir dans cet article, mais vous pouvez trouver plus d'informations à cette URL : **https://developers.google.com/web/updates/2016/03/web-push-encryption**

Aussi nous ajoutons un écouteur pour l'événement du clic sur la notification. Lors de son déclenchement, on ouvre la page Web en question ou on met le focus sur celle-ci si elle est déjà ouverte.

```
self.addEventListener('notificationclick', function(event) {
  event.notification.close();
  event.waitUntil(clients.matchAll({
    type: 'window'
  }).then(function(clientList) {
    for (var i = 0; i < clientList.length; i++) {
      var client = clientList[i];
      if (client.url === '/' && 'focus' in client) {
        return client.focus();
      }
    }
    if (clients.openWindow) {
      return clients.openWindow('/');
    }
  }));
});
```

Désormais à chaque fois que le compteur atteint une centaine, il va envoyer une push notification à tous les appareils autorisés.

CONCLUSION

Les PWA sont une initiative de Google et donc très poussées par la société de Mountain View. Ils ont très récemment annoncé que désormais la recherche sur mobile allait être leur souci principal face à la recherche classique, les PWA semblent donc avoir un bel avenir devant elles. Maintenant que nous avons notre Progressive Web App terminée, il est intéressant de la tester pour voir son score auprès de l'outil Google Lighthouse. Cet outil est disponible sur Github : **https://github.com/GoogleChrome/lighthouse** ou via une extension de Google Chrome disponible sur le Web Store Chrome.

Notre PWA est disponible à l'adresse **https://lovelovelove.xyz** et reçoit un score de 79/100. Ce qui n'est pas trop mal pour une mini PWA. Beaucoup d'améliorations peuvent être apportées à ce projet et nous vous invitons à venir collaborer sur sa version Github : **https://github.com/MadeOnMars/lovelovelove**

Merci d'avoir suivi cet article et n'hésitez pas à nous suivre sur les différents réseaux sociaux. •

Made On Mars est une agence digitale rennaise fondée en 2016 par Thomas Foricher (directeur technique) et Simon Gombaud (directeur artistique). Passionnés de digital, ils conçoivent des applications mobiles et des sites Web pour aider les entreprises à développer leur e-réputation. **https://www.made-on-mars.com/fr/**

Utiliser l'API RATP temps réel dans une application UWP

• Sébastien Putier
Consultant .NET
Ai3 / Agence Est

Au tout début de cette année, la RATP a mis à disposition du public une API temps réel permettant d'obtenir les horaires des lignes qui composent son réseau. Nous allons voir comment l'intégrer dans une application Universal Windows Platform.

La loi n° 2015-990 du 6 août 2015 pour la croissance, l'activité et l'égalité des chances économiques, dite « Loi Macron », impose aux opérateurs de transports publics de mettre à disposition du public certaines données relatives notamment aux horaires définis pour leurs réseaux. Ces informations doivent être rendues disponibles gratuitement par le moyen d'un canal de communication électronique. Dans ce cadre, la RATP, organisatrice des transports publics en Ile-de-France, a ouvert le 3 janvier 2017 l'accès à une API Web qui permet d'obtenir les informations suivantes :

- Structure du réseau : lignes et stations ;
- Horaires théoriques et prochains passages en temps réel ;
- Perturbations sur le réseau.

Toutefois, la RATP contrôle l'utilisation du service de manière à éviter les consommateurs trop gourmands. Certaines limitations sont en place au niveau du nombre de requêtes que le service peut accepter : au maximum, 200 par seconde, et au-delà de 30 000 000 en moins d'un mois, l'accès est coupé jusqu'au début du mois suivant. Afin de garder la maîtrise de l'utilisation du service, la RATP réserve par conséquent son accès aux seuls utilisateurs enregistrés.

Pour obtenir ce statut, il faut remplir le formulaire d'inscription : nom, prénom, adresse, informations relatives à la société si l'utilisateur est un professionnel, et surtout, adresses IP (3 au maximum) à partir desquelles l'utilisateur désire envoyer les requêtes. Le formulaire est à renvoyer par mail au service Open Data de la RATP, accompagné d'un justificatif de domicile. L'enregistrement est confirmé par retour de mail quelques jours après (6 jours, dans mon cas).

Le SDK

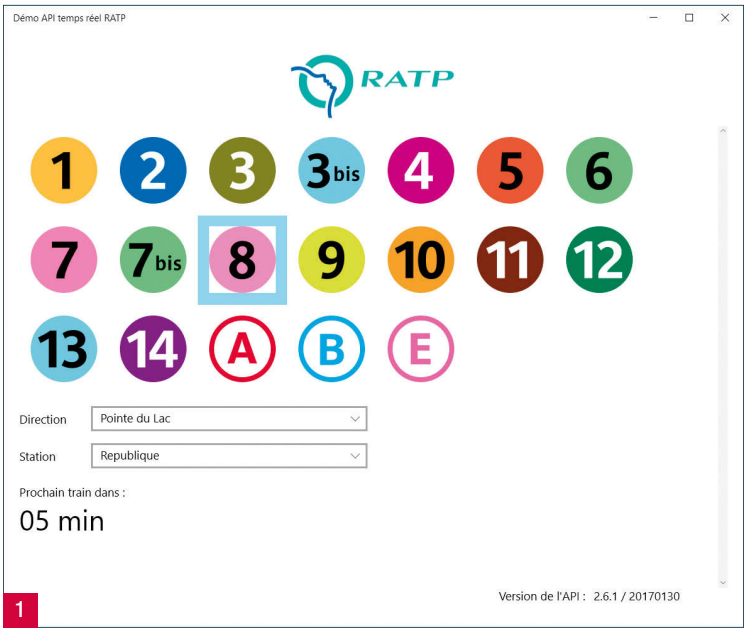
La RATP fournit un SDK téléchargeable à partir d'une page de son site dédié à l'Open Data : https://data.ratp.fr/page/temps-reel/. Ce kit de développement est tout simplement une archive au format zip qui contient 5 fichiers :

- doc_wsiv.html : la documentation technique de l'API ;
- exemple.pdf : une documentation qui montre comment manipuler l'API à l'aide de requêtes SOAP ;
- index.html : une page Web très simple qui pointe vers les deux fichiers précédents ;
- Ratp.gif : le logo de la RATP ;
- Wsiv.wsdl : le fichier de description de l'API Web, au format WSDL.

Ces éléments permettent de démarrer la phase de développement avant même d'avoir été enregistré dans le système de la RATP.

L'application UWP

L'application que nous allons construire ici a pour objectif de déterminer le temps d'attente avant le prochain passage d'une rame pour une ligne, une direction et une station donnés (Fig. 1). Les différentes lignes affichées correspondent au réseau du métro ainsi qu'à la portion du réseau RER géré par la RATP.



1

La première étape de son développement est bien évidemment la création du projet. Le template « Application vide (Windows Universel) » de Visual Studio 2015 convient parfaitement. Pour ma part, j'ai naturellement structuré le projet avec un dossier Views, contenant la vue principale (MainPage.xaml), et un dossier ViewModels, qui contient la classe MainViewModel. J'ai également installé les packages NuGet MvvmLightLibs et Microsoft.Xaml.Behaviors.Uwp.Managed pour tirer parti du pattern MVVM.

MainViewModel hérite de GalaSoft.MvvmLight.ViewModelBase. MainPage a quelques using supplémentaires pour utiliser des Behaviors, et son DataContext est valorisé avec une instance de MainViewModel directement en XAML

MainPage.xaml :

```
<Page
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HorairesRatp"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    x:Class="HorairesRatp.Views.MainPage"
    xmlns:viewModel="using:HorairesRatp.ViewModels"
    xmlns:interactivity="using:Microsoft.Xaml.Interactivity"
    xmlns:core="using:Microsoft.Xaml.Interactions.Core">
```

```
  <Page.DataContext>
    <viewModel:MainViewModel />
  </Page.DataContext>

  <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}"


  </Grid>
</Page>
```

Intégration du service Web

Maintenant que la mise en place est faite, il est temps de passer aux choses sérieuses. Pour la plateforme UWP, Visual Studio est capable de générer un proxy à partir d'un fichier WSDL. L'ajout d'une référence de service vers le fichier WSDL fourni effectue cette opération sans le moindre problème (Fig. 2).

Une fois le proxy généré, l'envoi de requêtes est réalisé à travers un objet de type ServiceRatp.WsivPortTypeClient.

La structure du réseau

D'un point de vue général, différents réseaux composent le maillage RATP : RER, Metro, Bus, etc. Chacun des réseaux est constitué de plusieurs lignes.

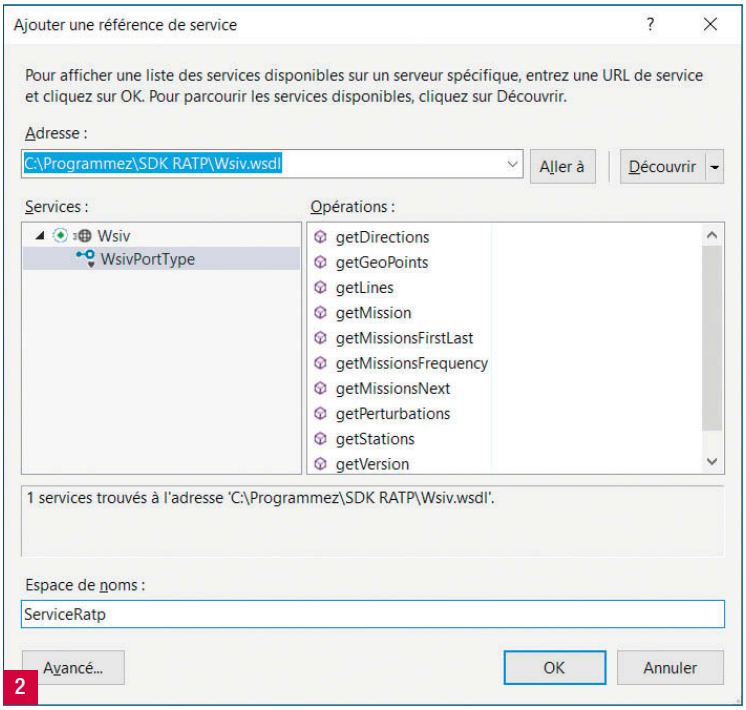
La méthode getLinesAsync exposée par le service permet de lister les lignes de chaque réseau. Elle attend un paramètre de type Line qui doit être (au moins) partiellement renseigné : pour une recherche par réseau, il faut valoriser sa propriété réseau. Il est également possible de rechercher par id, code (n° de ligne), code STIF (identifiant interne du STIF).

Différents codes permettent de cibler chacun des réseaux disponibles, ainsi que certains groupements de réseaux prédéfinis. Dans le cas présent, ceux qui nous intéressent sont « rer » et « metro ». Vous pouvez retrouver l'ensemble des valeurs acceptées à la section « 2.4.1.1 reseauOrGroup » de la documentation.

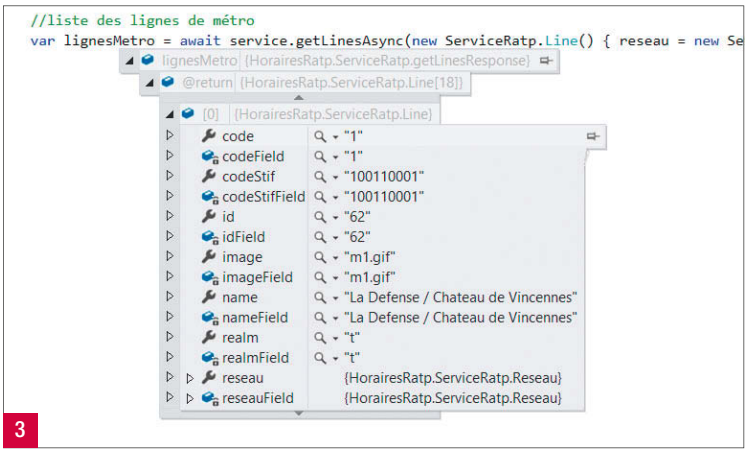
```
var lignesMetro = await service.getLinesAsync(new ServiceRatp.Line() { reseau = new ServiceRatp.Reseau() { code = "metro" } });

var lignesRER = await service.getLinesAsync(new ServiceRatp.Line() { reseau = new ServiceRatp.Reseau() { code = "rer" } });
```

Les valeurs retournées sont de type getLinesResponse. Cette classe expose un tableau d'objets Line à travers sa variable membre @return (Fig. 3). Pour afficher ces lignes dans une ListBox, nous allons les charger dans une collection au niveau du ViewModel. L'interrogation du service est faite au chargement de la Page principale par l'exécution d'une commande. Les lignes de RER retournées sont filtrées pour ne tenir compte que des lignes gérées par la RATP.

```
private IEnumerable<ServiceRatp.Line> _lines;
public IEnumerable<ServiceRatp.Line> Lines
{
  get { return _lines; }
  set
  {
    _lines = value;
    RaisePropertyChanged();
```



2



3

```
  }
}

public RelayCommand InitializeCommand { get; set; }

public MainViewModel()
{
  InitializeCommand = new RelayCommand(InitializeAction);
}

public async void InitializeAction()
{
  var service = new ServiceRatp.WsivPortTypeClient();

  //liste des lignes de métro
  var lignesMetro = await service.getLinesAsync(new ServiceRatp.Line() { reseau = new ServiceRatp.Reseau() { code = "metro" } });
  //liste des lignes RER
  var lignesRER = await service.getLinesAsync(new ServiceRatp.Line() { reseau = new ServiceRatp.Reseau() { code = "rer" } });
```

```
    var allLines = lignesMetro.@return
        .Where(l => l.code[0] >= '1' && l.code[0] <= '9')
        .Concat(lignesRER.@return.Where(l => l.codeStif != "0"))
        .ToList();

    Lines = allLines;
}
```

La commande est quant à elle associée à l'événement Loaded de la Page par l'utilisation d'un EventTriggerBehavior.

```
<interactivity:Interaction.Behaviors>
    <core:EventTriggerBehavior EventName="Loaded">
        <core:EventTriggerBehavior.Actions>
            <core:InvokeCommandAction Command="{Binding InitializeCommand}" />
        </core:EventTriggerBehavior.Actions>
    </core:EventTriggerBehavior>
</interactivity:Interaction.Behaviors>
```

Pour la suite, rien de bien compliqué. Une ListBox, un binding et tout s'affiche. Pour rendre le résultat un peu plus présentable, il est possible de récupérer les indices des lignes du réseau ferré sur le site de l'Open Data RATP : https://data.ratp.fr/explore/dataset/indices-et-couleurs-de-lignes-du-reseau-ferre-ratp. Avec quelques manipulations pour associer la bonne image à chaque ligne et la modifications de l'ItemsPanel de la ListBox, le rendu est bien plus sympathique.
Pour ma part, j'ai intégré le code du WrapPanel du Community Toolkit UWP dans le projet : https://github.com/Microsoft/UWPCommunityToolkit/tree/dev/Microsoft.Toolkit.Uwp.UI.Controls/WrapPanel
Le code qui permet de générer le bon nom d'image pour chacune des lignes affichées est relativement court :

```
private string GetLineImagePath(ServiceRatp.Line line)
{
    if (line.reseau.code == "metro")
        return "M_" + line.code + (line.code.EndsWith("b") ? "is" : "") + ".png";
    else if (line.reseau.code == "rer")
        return "RER_" + line.code + ".png";

    return null;
}

allLines.ForEach(l => l.image = GetLineImagePath(l));
```

Pour la suite, une commande est exécutée à chaque déclenchement de l'événement SelectionChanged de la ListBox. Son rôle est la récupération des directions associées à la ligne sélectionnée : les directions.
La méthode exposée pour la recherche des directions, getDirectionsAsync, est très simple puisqu'elle n'attend qu'un argument : un objet Line. En retour, on obtient un objet getDirectionsResponse dont le membre @return contient les données attendues.

```
var service = new ServiceRatp.WsivPortTypeClient();

var directions = await service.getDirectionsAsync(_selectedLine);
LineDirections = directions.@return.directions;
```

Ici, même mécanique que précédemment : une ComboBox bindée sur la propriété LineDirections, une commande associée à son événement SelectionChanged. Celle-ci exécute le chargement des différentes stations parcourues sur le trajet choisi.

```
<ComboBox ItemsSource="{Binding LineDirections}" SelectedItem="{Binding Selected
LineDirection, Mode=TwoWay}" Width="352" Margin="15,0,0,0">
    <interactivity:Interaction.Behaviors>
        <core:EventTriggerBehavior EventName="SelectionChanged">
            <core:InvokeCommandAction
                Command="{Binding SelectedDirectionChangedCommand}" />
        </core:EventTriggerBehavior>
    </interactivity:Interaction.Behaviors>
    <ComboBox.ItemTemplate>
        <DataTemplate>
            <TextBlock Text="{Binding name}" />
        </DataTemplate>
    </ComboBox.ItemTemplate>
</ComboBox>
```

La méthode GetStationsAync est nettement plus flexible. La recherche peut se baser sur des critères variés, :

- Id de station
- Ligne & direction (avec direction optionnelle)
- Nom de station
- Coordonnées géographiques

```
public Task<getStationsResponse> getStationsAsync(
        Station station,
        GeoPoint gp,
        int[] distances,
        int limit,
        bool sortAlpha)
```

Notre recherche s'appuie sur les lignes et directions choisies. Ces valeurs sont passées par l'intermédiaire du paramètre station. Le nombre de stations retournées est limité à 100 par sécurité, au lieu de 500 par défaut (quand limit = 0).

```
var service = new ServiceRatp.WsivPortTypeClient();

var stations = await service.getStationsAsync(
        new ServiceRatp.Station()
        {
            line = _selectedLine,
            direction = _selectedLineDirection
        },
        null, null, 100, false);

LineStations = stations.@return.stations;
```

D'autres cas d'utilisation nécessitent de se baser sur un critère géographique. Le second paramètre de la méthode getStationsAsync a pour objectif de gérer ce cas. En revanche, il sera bien difficile de le valoriser avec une latitude et une longitude… Les coordonnées doivent en effet être au format Lambert II étendu. Pour effectuer cette conversion, vous pouvez utiliser le type GpsLambert2Converter, disponible sur GitHub : https://github.com/sputier/SPR.Geo

La récupération des coordonnées géographiques de l'utilisateur est exécutée à l'aide du type Windows.Devices.Geolocation.Geolocator. Pour que la géolocalisation soit fonctionnelle au sein d'une application UWP, il est nécessaire d'indiquer son utilisation au travers du manifeste de l'application. Dans Visual Studio, il faut pour cela ouvrir le fichier Package.appxmanifest par un double clic, puis cocher la capacité Emplacement dans l'onglet Capacités. Au niveau du code, il faut également demander explicitement l'autorisation de cette fonctionnalité par l'utilisation de la méthode de Geolocator.RequestAccessAsync.

```
var accessStatus = await Geolocator.RequestAccessAsync();

if (accessStatus != GeolocationAccessStatus.Allowed)
   return;

Geolocator geolocator =
   new Geolocator { DesiredAccuracyInMeters = 0 };

Geoposition pos = await geolocator.GetGeopositionAsync();

var lambert2Position = new GpsLambert2Converter().GpsToLambert2(
   pos.Coordinate.Point.Position.Latitude,
   pos.Coordinate.Point.Position.Longitude);
```

Une fois ces coordonnées obtenues, il faut les passer à la méthode getStationsAsync sous la forme d'un objet GeoPoint. Pour rechercher l'ensemble des stations à proximité, le premier paramètre doit avoir pour valeur null, mais il est possible limiter le nombre de stations retournées en fournissant un objet Line renseigné. L'exemple suivant recherche les stations de la ligne sélectionnée dans un rayon de 2000m autour de l'utilisateur.

```
var stationsAProximite = await service.getStationsAsync(
   new ServiceRatp.Station()
   {
      line = _selectedLine,
      direction = _selectedLineDirection
   },
   new ServiceRatp.GeoPoint()
   {
      x = lambert2Position.X,
      y = lambert2Position.Y,
      xSpecified = true,
      ySpecified = true
   },
   new int[] { 2000 },
   0,
   false);
```

Une fois la liste des stations obtenues, il suffit d'appliquer la même recette que pour les directions : ComboBox, binding, commande qui déclenche la recherche des horaires.

Les horaires

Les horaires sur le réseau sont étroitement liés au concept de mission. Chaque mission représente une course, d'un point de départ à un point d'arrivée, avec des arrêts intermédiaires. La récupération des prochains passages à une station est faite par un appel à la méthode getMissionsNextAsync. Les deux premiers arguments passés correspondent à la station et à la direction, le troisième est la date à partir de laquelle l'on souhaite les horaires (null = maintenant), et le dernier permet de limiter le nombre de résultats. Les dates manipulées par le service sont représentées sous forme de chaînes de caractères au format yyyyMMddHHmm, il est donc nécessaire de faire un peu de parsing pour obtenir un objet DateTime en retour.

```
var service = new ServiceRatp.WsivPortTypeClient();

var nextMission = await service.getMissionsNextAsync(
      _selectedLineStation,
      _selectedLineDirection,
      null
      1);

string result = null;
var dateString = nextMission.@return.missions.FirstOrDefault()?.stationsDates?[0];

if (dateString == null)
   return;

var date = new DateTime(int.Parse(dateString.Substring(0, 4)),
               int.Parse(dateString.Substring(4, 2)),
               int.Parse(dateString.Substring(6, 2)),
               int.Parse(dateString.Substring(8, 2)),
               int.Parse(dateString.Substring(10, 2)),
               0);

var waitTime = date - DateTime.Now;

if (waitTime.TotalHours < 1)
   result = string.Format("{0:00} min", (int)waitTime.TotalMinutes);
else
   result = string.Format("+{0} h", (int)waitTime.TotalHours);

NextTrainWaitTime = result;
```

Le détail d'une mission est accessible par un appel à la méthode getMissionAsync. Le code suivant crée une séquence d'objets [station, date] correspondant à chaque arrêt passé et à venir du prochain train.

```
var mission = await service.getMissionAsync(nextMission.@return.missions.
First(), null, false, false);

var missionDetails = mission.@return.mission.stations.Zip(
   mission.@return.mission.stationsDates,
   (a, b) => new { Station = a.name, DateString = b });
```

En résumé

L'API Temps réel de la RATP est simple et plutôt efficace. Avec une documentation relativement claire et complète, ce service permet d'intégrer des fonctionnalités de recherche d'horaires très rapidement dans une application UWP. On pourra toutefois regretter la phase d'inscription très « administrative » ainsi que l'utilisation de SOAP, qui impose presque de devoir générer un proxy pour conserver le caractère utilisable de l'API.
Le code source utilisé est disponible sur Github :
https://github.com/sputier/Demo_APITempsReel_RATP •

EmbedXcode : comment programmer un Arduino à l'aide d'Xcode sous Mac Partie 2

• Renaud Pinon

Dans cette 2e partie, faisons notre premier déploiement de XCode vers Android.

Diviser pour mieux régner

Evidemment, maintenant que l'on a une arborescence, la gestion des fichiers multiples dans le projet s'en retrouve extrêmement facilitée. Alors que sous *Arduino IDE* l'affichage multi-onglets des fichiers relevait de la torture (impossible de déplacer les onglets, manque de lisibilité en cas de nombreux fichiers, …), nous allons désormais pouvoir, sous Xcode, utiliser le bon vieil adage « diviser pour mieux régner », c'est-à-dire utiliser des fichiers *.h* regroupés par thématiques. Par exemple un fichier pour les structures, un pour les énumérations, un autre pour les prototypes, etc.
Pour l'exemple, créons simplement un fichier .h qui contiendra une bête fonction et son prototype. Faites un clic droit sur l'arborescence et choisissez « *New group* ». Nommez-le par exemple « *Headers* », puis faites un clic droit dessus et choisissez « *New File…* ». Choisissez à gauche la catégorie *OS X / Source*, puis à droite l'icône *Header File* et enfin cliquez sur *Next*. [7] Choisissez ensuite le dossier source de votre projet, créez optionnellement un dossier intitulé *Headers*, nommez le fichier à créer *MyFunctions.h*, puis cliquez sur *Create*. Une fois présent dans l'arborescence de gauche, sélectionnez ce fichier pour afficher le contenu dans la partie centrale.

```
#ifndef MyFunctions_h
#define MyFunctions_h

// Prototype:
int Somme(int a, int b);

// Implémentation:
int Somme(int a, int b)
{
   return a+b;
}

#endif /* MyFunctions_h */
```

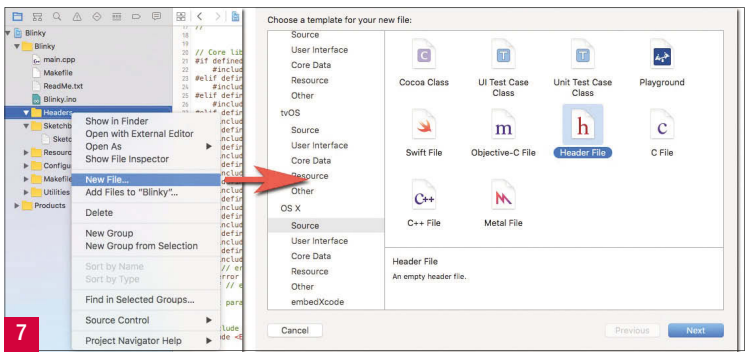
Copiez le contenu suivant à l'intérieur :
Enfin dans le fichier *Blinky.ino*, remplacez tout le code après les « *#define magiques* » par ce code :

```
// Fichiers en-tête:
#include "MyFunctions.h"

// Add setup code
void setup()
{
  Serial.begin(9600);
}

// Add loop code
void loop()
{
  int a = random(20);      // Nombre entre 0 et 20.
  int b = random(20);

  // On écrit la somme des deux nombres:
  Serial.print("Somme de "+String(a)+" et "+String(b));
  Serial.println(" = "+String(Somme(a, b)));
  delay(1500);    // Délai de 1,5 seconde.
}
```

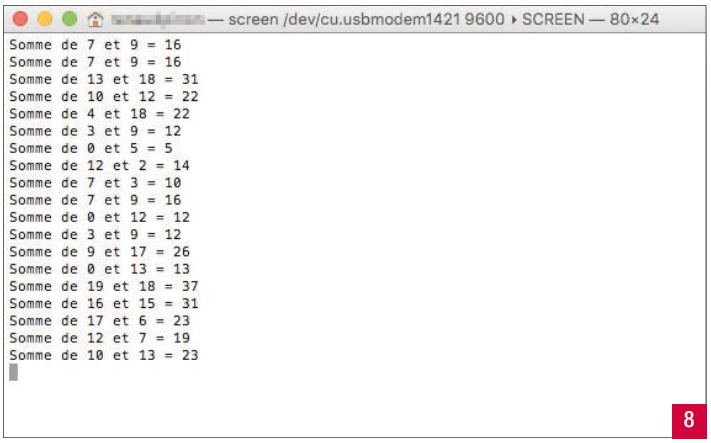
Lancez la compilation : une fenêtre Terminal doit s'ouvrir avec l'écriture toutes les 1,5 secondes d'une ligne semblable à ce que vous pouvez trouver sur cette capture : [8]
Bien évidemment, si vous vous amusez à taper « *Som* » sur une nouvelle ligne dans la fonction *loop()*, vous constaterez qu'*Xcode* est capable de vous afficher la proposition de code et de faire l'auto-complétion vers notre fonction *Somme()*. Joie :)

Bibliothèques

La communauté *Arduino* étant très active, il est très fréquent qu'elle développe des bibliothèques permettant de faire fonctionner nos modules préférés. Plus généralement, un certain nombre d'entre elles sont déjà inclues par défaut dans le logiciel *Arduino*.



7



8

EmbedXcode permet évidemment de les utiliser. Il y a toutefois une petite subtilité en comparaison avec *Arduino IDE* : il va falloir indiquer au fichier *Makefile* du projet que nous souhaitons lier telle ou telle bibliothèque.
Deux types se distinguent dans *EmbedXcode* : les bibliothèques d'application et les bibliothèques utilisateur. Les premières sont intégrées d'office dans *Arduino IDE* : *SoftwareSerial*, *EEPROM*, *Ethernet*, *LiquidCrystal*, … Vous pouvez toutes les consulter dans le projet *Xcode* en dépliant le dossier *Resources/Arduino/arduino/libraries* de l'arborescence de gauche.
Les bibliothèques utilisateur sont, comme leur nom l'indique, téléchargées par l'utilisateur. Si on se place du point de vue d'*Arduino IDE*, ce sont les bibliothèques qui se trouvent dans le dossier *~/Documents/Arduino/libraries*. On peut soit les télécharger sur Internet, soit demander à *Arduino IDE* de le faire pour nous en allant dans son menu *Croquis/Gérer les bibliothèques*, puis en choisissant celle(s) à installer dans la liste.
Mais revenons à *Xcode* : nous allons tout d'abord voir comment utiliser la bibliothèque d'application *EEPROM* en modifiant notre projet exemple. Notre code tentera d'écrire une valeur à l'adresse 0x00 de l'*EEPROM* (mémoire de masse de quelques ko) intégrée à l'Arduino, puis de lire la valeur présente à cette même adresse pour voir si elle s'est correctement écrite. Pour cela, modifions de nouveau notre fichier *Blinky.ino* : remplacez tout ce qui se trouve après les « *#define magiques* » par ce code :

```
// [...] defines...
// Include application, user and local libraries
#include <EEPROM.h>

// Constantes:
const int _address = 0x00;

// Add setup code
void setup()
{
  Serial.begin(9600);    // Init connexion série.
  EEPROM.begin();        // Init EEPROM.
}

// Add loop code
void loop()
{
  // Nombre aléatoire entre 0 et 255:
  int rnd = random(255);

  // écrit la valeur dans l'EEPROM à l'adresse 0x00:
  EEPROM.write(_address, (uint8_t)rnd);

  // délai pour être sûr que la valeur est bien écrite dans l'EEPROM:
  delay(50);  // 50 ms

  // Lit la valeur à l'adresse 0x00.
  uint8_t readValue = EEPROM.read(_address);

  // Ecrit les valeurs:
  Serial.print("Valeur attendue: "+String(rnd));
  Serial.println(", valeur lue dans l'EEPROM: "+String(readValue));
  delay(1500);  // 1.5 sec.
}
```

Lancez ensuite la compilation. Enfer et damnation ! A la ligne « *#include <EEPROM.h>* », vous obtenez l'erreur :

```
EEPROM.h: No such file or directory
```

Rassurez-vous, c'est parce qu'il faut faire une dernière étape avant d'utiliser une bibliothèque. Sélectionnez le fichier *Makefile* dans l'arborescence de gauche (à ne pas confondre avec le dossier *Makefiles*) et repérez la ligne :

```
APP_LIBS_LIST = 0
```

Elle indique qu'aucune bibliothèque d'application ne doit être liée au moment de la compilation. Nous allons justement la modifier pour nous permettre d'utiliser la bibliothèque *EEPROM*. Pour cela nous allons tout simplement la remplacer par :

```
APP_LIBS_LIST = EEPROM
```

A noter que si nous voulions utiliser plusieurs bibliothèques, il suffirait de séparer leurs noms par un espace. Par exemple :

```
APP_LIBS_LIST = EEPROM Ethernet LiquidCrystal
```

permet d'utiliser dans le programme les bibliothèques *EEPROM*, *Ethernet* et *LiquidCrystal*. Enfin, si rien n'est indiqué après le signe égal (simple retour chariot) alors **toutes** les bibliothèques sont considérées au moment de la compilation. Cela peut être utile si vous avez du mal à faire reconnaître une bibliothèque un peu exotique, mais la compilation sera évidemment plus lente.
Maintenant que vous avez rajouté *EEPROM* dans la liste, relancez la compilation : tout devrait s'exécuter correctement et une fenêtre *Terminal* doit s'ouvrir avec un contenu ressemblant à celui-ci : [9]
Parfait. Mais qu'en est-il des bibliothèques créées par d'autres utilisateurs ? N'étant pas intégrées de base au logiciel *Arduino*, il convient de les télécharger soit à partir du menu *Croquis / Gérer les bibliothèques* d'*Arduino IDE*, soit sous forme de fichier *.zip* sur Internet et de les décompresser dans le dossier *~/Documents/Arduino/libraries*. Toutefois, lorsque ceci est fait, une autre étape nous attend dans *Xcode* avant que ces bibliothèques ne puissent être prises en compte : il faut inclure leurs fichiers au projet. Pour cela, ouvrez une fenêtre *Finder*, allez dans le dossier *~/Documents/Arduino/libraries* et déplacez-glissez tous les dossiers qu'il contient vers *Xcode*, dans le dossier *Sketchbook* de l'arborescence du projet.
Une boîte de dialogue va alors s'ouvrir : désactivez « *Copy items if needed* », sélectionnez « *Create groups* », et cochez la case « *Index* » dans le cadre « *Add to target* ». Cliquez enfin sur le bouton *Finish*.
Pour terminer, sélectionnez le fichier *Makefile* pour l'éditer, mais cette fois modifiez la ligne intitulée « *USER_LIBS_LIST = 0* » et remplacez « *0* » par le nom de la ou les bibliothèques qui vous intéressent.
Pour l'exemple, nous allons ici créer un programme qui utilise la bibliothèque *Keypad* et lira la valeur des boutons pressés sur un clavier numérique branchés aux broches *D2* à *D8* de notre *Arduino* : [10]
Bien-sûr, cela ne fonctionnera que si vous possédez un tel clavier :)

```
— screen /dev/cu.usbmodem1421 9600 ▸ SCREEN — 80×24
Valeur attendue: 232, valeur lue dans l'EEPROM: 232
Valeur attendue: 232, valeur lue dans l'EEPROM: 232
Valeur attendue: 19, valeur lue dans l'EEPROM: 19
Valeur attendue: 158, valeur lue dans l'EEPROM: 158
Valeur attendue: 38, valeur lue dans l'EEPROM: 38
Valeur attendue: 175, valeur lue dans l'EEPROM: 175
Valeur attendue: 197, valeur lue dans l'EEPROM: 197
Valeur attendue: 114, valeur lue dans l'EEPROM: 114
```

9

Voici la ligne à remplacer dans le fichier *Makefile* :

```
USER_LIBS_LIST = Keypad
```

Notez encore une fois que si nous voulons utiliser plusieurs bibliothèques dans notre programme il suffit de séparer leurs noms par un espace, et que si nous voulons que le compilateur considère toutes les bibliothèques installées sans avoir à donner de nom, il suffit de faire un retour chariot directement après le signe égal.
Et pour finir, voici notre programme exemple à insérer dans le fichier *Blinky.ino* (après les « *#define magiques* ») :

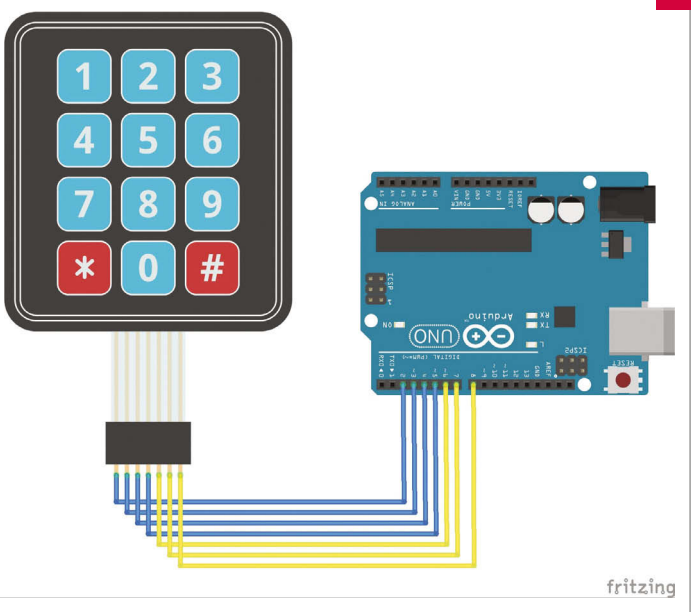
```
// Bibliothèques:
#include <Keypad.h>

// Defines, constantes et variables globales:
#define kRows     4     // nombre de rangées sur le clavier.
#define kCols     20    // nombre de colonnes sur le clavier.

byte _rowPins[kRows] = {2,3,4,5};  // broches Arduino rangées.
byte _colPins[kCols] = {6,7,8};    // broches Arduino colonnes.

// Tableau représentant le mapping des caractères sur le clavier:
char _keys[kRows][kCols] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'},
};

// Initialisation de l'objet global clavier:
Keypad _keypad = Keypad(makeKeymap(_keys), _rowPins, _colPins, kRows, kCols);

// Add setup code
void setup()
{
  Serial.begin(9600);    // Init connexion série.
}

// Add loop code
void loop()
{
  // Obtention de la touche appuyée (s'il y en a une):
  char key = _keypad.getKey();

  // Si une touche est appuyée:
  if (key != NO_KEY)
  {
     // On écrit la touche appuyée:
     Serial.print(key);
  }

}
```

Compilez votre programme : si tout se passe bien et que vous avez correctement raccordé un clavier numérique à l'*Arduino*, une valeur s'imprime à chaque fois que vous appuyez sur une touche

CONCLUSION

En définitive, *EmbedXcode* est-il exempt de tout défaut ? Certes, non.
Le principal problème, comme pour *Arduino IDE*, est que les erreurs ne s'affichent qu'au moment de la compilation. Bien sûr, la ligne incriminée est indiquée beaucoup plus clairement que dans l'habituel logiciel dédié à nos cartes électroniques, mais tout de même, la compilation ressemblera toujours à l'instant « loterie » où l'on saura enfin s'il y a des erreurs dans notre code.
De plus, ne rêvez pas : les points d'arrêts que vous placerez dans *Xcode* ne mettront pas l'exécution en pause (ils serviront tout au plus de signet). Toutefois, d'après la documentation, il semblerait que le *debug* soit possible avec la version « + » d'*EmbedXcode*, combinée à un boîtier de débogage matériel pour *Arduino* de type *Segger J-Link*. Cependant, n'ayant pu tester cela, je vous conseille de bien lire la documentation avant de vous lancer dans de potentiels achats.
Enfin, le dernier défaut est la gymnastique liée à la prise en compte des bibliothèques dans le fichier *Makefile*. Mais malgré cela, avouons que le pli est vite pris, et qu'à part quelques soucis avec des bibliothèques un peu anciennes ou exotiques, l'intégration reste relativement simple.
Avec ces défauts encore présents, se servir d'*Xcode* au lieu d'*Arduino IDE* vaut-il alors vraiment le coup ? La réponse est : mille fois oui ! Utiliser un environnement de développement moderne est une véritable révolution pour quiconque est habitué à *Arduino IDE*. Les fonctionnalités telles que la proposition de code, l'auto-complétion, la vue hiérarchique des fichiers ou encore une véritable coloration syntaxique sont autant d'atouts vous permettant enfin d'envisager des projets électroniques plus ambitieux et mieux structurés. Bref, l'essayer c'est l'adopter, et personnellement je me vois mal revenir à *Arduino IDE*, qui, même s'il fait le job (et c'est tout à son honneur), présente bien trop de lacunes pour le développeur habitué à des éditeurs évolués. Et cerise sur le gâteau : le contrôle de code source se faisant à partir d'*Xcode*, pourquoi ne pas songer au travail collaboratif sur un même projet grâce à un serveur GIT ? •

10



JavaScript : des origines à NodeJS Partie 2

• Teddy DESMAS
• Guillaume LEBORGNE
• Thomas OUVRE
Développeurs chez **Infinite Square**
http://blogs.infinitesquare.com/

JavaScript ou JS est un langage de programmation Web basé comme son nom l'indique sur des scripts. Il a été créé en 1995 par Brendan Eich, un informaticien américain travaillant alors chez Netscape en s'inspirant de Java (d'où son nom) mais en simplifiant la syntaxe de ce dernier pour faciliter son utilisation notamment pour les débutants. JavaScript fut standardisé 2 ans après, en juin 1997 sous le nom d'ECMASCRIPT. Il est alors devenu une implémentation de ce standard par la fondation Mozilla, et certaines entreprises comme Microsoft ou Adobe ont implémenté leur propre version (respectivement JScript et ActionScript).

Nous avons finalement parcouru une bonne partie des nouveautés introduites par ES6 et il semble plus qu'évident qu'elles vont simplifier le quotidien des développeurs JavaScript. Si toutefois toutes ces fonctionnalités ne vous suffisaient pas, il existe d'autres langages transpilés, offrant plus encore, comme le TypeScript !

TypeScript

De la même façon que Babel avec les fonctionnalités ES6 non supportées nativement, TypeScript offre de nouvelles fonctionnalités aux développeurs JavaScript en y étendant la syntaxe. Ainsi TypeScript s'apparente à un nouveau langage, très proche du JavaScript, et apporte des possibilités supplémentaires comme le typage statique. Il supporte aussi toutes les fonctionnalités ES6. Il s'agit lui aussi d'un langage que l'on doit transpiler en JavaScript pour son exécution.

Getting Started

Comme pour bénéficier des fonctionnalités ES6 avec Babel, l'utilisation de TypeScript nécessite un peu de configuration sur son environnement de développement. La première chose à faire consiste à en installer le package NPM :

```
npm install -g typescript
```

Il est ensuite possible de compiler un fichier TypeScript, dont l'extension est « .ts » avec la commande suivante :

```
tsc main.ts
```

Où « main.ts » serait constitué du code suivant :

```
class Hello {
    world() {
        console.log("hello world");
    }
}
Le fichier produit, « main.js » :
var Hello = (function () {
    function Hello() {
    }
    Hello.prototype.world = function () {
        console.log("hello world");
    };
    return Hello;
}());
```

Fonctionnalités

Typage

TypeScript introduit le typage statique et tous les bénéfices qui en découlent : un code plus sécurisé du point de vue de la compilation, plus facile à produire et à maintenir. Il faut toutefois conserver à l'esprit que le but final est de produire un fichier JavaScript, qui, lui, n'est pas fortement typé. Le typage de TypeScript n'a donc d'impact que lors de la compilation, pas durant l'exécution. Il est ainsi possible de définir le type d'une variable, le retour d'une fonction ou celui de ses paramètres :

```
let v1: Window = null;

class Demo {
    m1: string;

    f1():void {}

    f2(p1: number, p2: {m2:Date}) {
        console.log(p1);
        console.log(p2.m2);
    }
}
```

Il existe aussi un type particulier, « any », qui permet de représenter un objet dont le type n'est pas connu à la compilation. Ainsi, il est possible de retrouver un fonctionnement proche du JavaScript pur en termes de typage grâce à ce mot clef :

```
function useAny(value: any) {
    console.log(typeof value);
}

useAny(1);
useAny("string");
useAny({});
useAny(null);
useAny(undefined);
```

Classes, interfaces et enums

Comme avec ES6, TypeScript autorise la définition de classes, mais va aussi plus loin en permettant de créer des interfaces et des énumérations. Les interfaces permettent de définir les membres attendus d'un objet :

```
interface SampleInterface {
    member1: number;
    member2: string;
    member3: SampleInterface;
}

function useInterface(value: SampleInterface) {
    console.log(value.member1);
    console.log(value.member2);
```

```
    console.log(value.member3);
}
```

Voici la définition d'une énumération et son usage :

```
enum SampleEnum {
    v1,
    v2,
    v3 = 5
}
function useEnum(value: SampleEnum) {
    console.log(value == SampleEnum.v1);
}
```

Les classes permettent aussi de définir une visibilité pour les membres qui la composent :

```
class SampleClass implements SampleInterface {
    public member1: number;
    private _member4: SampleEnum;
    constructor(public member2: string, public member3: SampleInterface) {}
    protected member5() {}
}
```

On peut aussi voir dans ce précédent exemple la façon d'implémenter une interface à l'aide du mot clef « implements ». Les membres 2 et 3 sont aussi définis, dans le constructeur : ajouter une visibilité à un paramètre du constructeur en fait automatiquement l'un des membres de la classe. Une classe peut aussi hériter d'une autre (mot clef « extends ») :

```
class InheritedClass extends SampleClass {
    constructor() {
        super("value", null);
        this.member5();
    }
}
```

Le mot clef « super » permet de faire appel au constructeur de la classe mère, puis d'autoriser ensuite l'usage du mot clef « this » dans le constructeur, et donc de faire appel à un membre « protected » par exemple.

Generics

TypeScript apporte aussi une autre fonctionnalité très pratique : celle des types génériques. Il est ainsi possible de définir des classes, interfaces et fonctions génériques :

```
function useGeneric<T>(p1:T, p2:T) {
    console.log(p1, p2);
}

useGeneric(1,2);
useGeneric("1", "2");
useGeneric(new Date(), new Date());
useGeneric(1,""); // error
```

Il est possible de combiner plusieurs types génériques :

```
class WithGenerics<T, U> {
    constructor(public m1: T, public m2: U) {}
}

let v = new WithGenerics("", 0);
console.log(v.m1);
console.log(v.m2);
```

Les types génériques peuvent aussi être définis avec des contraintes :

```
interface IWithGenerics<T, U extends T> {
    sum():U;
}
```

Dans ce précédent exemple, le type générique U doit étendre de T, ce qui signifie que l'exemple suivant est valide :

```
let v2: IWithGenerics<HTMLElement, HTMLDivElement>;
```

En revanche, l'exemple suivant n'est pas valide :

```
let v3: IWithGenerics<HTMLDivElement, HTMLElement>; // error
```

Decorators

TypeScript implémente aussi parfois des fonctionnalités qui ne sont encore qu'au stade expérimental. Les décorateurs sont une proposition faite pour une prochaine évolution du langage JavaScript et permettent d'annoter et modifier des classes et ses membres lors de la compilation. Ainsi il est possible d'utiliser une syntaxe de ce type :

```
@decorator
class AClass {}
```

Les décorateurs peuvent être utiliser sur les membres d'une classe :

```
class AClass {
    @decorator
    m1: any;

    @decorator
    f1() {}
}
```

Ils peuvent aussi être utilisés sur les paramètres d'une fonction. Les décorateurs sont par exemple largement employés dans le framework Angular2 lorsque l'on développe avec TypeScript. Comme nous avons pu le voir, TypeScript apporte beaucoup de fonctionnalités auxquelles les développeurs C# (ou même Java) sont habitués. Bien que l'absence de typage soit généralement considérée comme une force de JavaScript, ce que propose TypeScript apporte une expérience de développement généralement plus commode : on sait rapidement les types d'arguments attendus par une fonction sans devoir en regarder l'implémentation. De plus, l'IntelliSense de TypeScript est très puissante : un IDE compatible comme VS Code reconnaît toujours le contexte dans lequel on se trouve et propose une auto-complétion exhaustive. Enfin TypeScript permet d'utiliser directement du code JavaScript si on le souhaite.
Passons maintenant à un des outils qui a bouleversé le développement Web ces dernières années, je parle bien sûr de Node.js.

Node.js

Node.js est un shell permettant d'exécuter des scripts JavaScript en ligne de commande. La version officielle (disponible sur https://nodejs.org) est basée sur V8, le moteur JavaScript de Chrome, mais il existe aussi une version expérimentale fonctionnant sous Chakra, le moteur utilisé dans Microsoft Edge (https://github.com/nodejs/node-chakracore). A ses débuts, Node.js a été vu par de nombreux développeurs comme une sorte d'OVNI, et même par certains comme une hérésie (du JavaScript côté serveur ?

quelle horreur, ça ne marchera jamais…). Pourtant, c'est aujourd'hui une plateforme performante, fiable, robuste, et très utilisée notamment dans le développement Web et l'IoT. Après avoir séduit les grands acteurs du Web, il est maintenant également utilisé par les entreprises.

Que fait-on avec node.js ?

Pour le développeur web, node.js est devenu quasi incontournable. Il permet d'avoir un langage unique pour écrire ses scripts de build, compiler et déployer son application, faire le scripting côté navigateur, et potentiellement d'exécuter le code serveur de son application Web. Tout cela en ayant la possibilité de partager du code entre ces différentes briques. A tel point qu'on a vu apparaître le concept d'applications Web « isomorphiques » ou « universelles ». Cette architecture applicative permet de partager le code et faire le rendu d'une page Web aussi bien côté serveur que côté client. L'objectif du rendu sur le serveur étant de faire des applications riches (SPA), sans sacrifier les temps de chargement et le référencement dans les moteurs de recherche. Les frameworks JavaScript récents comme React ou Angular 2 permettent maintenant de réaliser ce genre d'applications relativement aisément. Ces derniers temps, node.js est aussi utilisé pour réaliser des applications de bureau multiplateformes. Des frameworks comme Electron ou nw.js permettent de réaliser des applications dont l'IHM repose sur Chromium (le moteur de chrome), qui embarquent un moteur node.js pour réaliser les différents traitements. On trouve des applications de tous types réalisées avec cette approche. Par exemple, le client desktop de Slack, ou des éditeurs de texte comme Atom ou Visual Studio Code reposent sur ces outils. Malgré tout, node.js n'est pas limité au développement Web. On le retrouve dans d'autres domaines comme l'IoT. De nombreuses plateformes IoT supportent node.js, certaines même nativement. C'est un domaine pour lequel il existe plusieurs frameworks très complets pour réaliser facilement vos projets (comme Johnny 5 disponible ici : http://johnny-five.io/).

Premiers pas

Il est nécessaire tout d'abord d'installer node.js. Rendez-vous sur le site officiel (https://nodejs.org/) pour télécharger la dernière version stable, privilégiez celle indiquée par la mention LTS car c'est cette branche qui est officiellement supportée.
Une fois que vous aurez installé node, ouvrez une invite de commande et tapez simplement « node » puis sur la touche « entrée ». Vous passerez alors en mode shell. Dans ce mode vous pouvez écrire directement des commandes en JavaScript, par exemple le traditionnel « Hello world » :

```
console.log("hello world");
```

Il est possible également de lancer vos commandes dans un fichier de script. Créez un fichier « hello.js » et déplacez-y votre Hello World. Lancez ensuite la commande suivante :

```
node hello.js
```

Vous devriez voir votre message apparaître dans la console.

On passe la seconde !

Hello World est toujours intéressant mais allons tout de même un peu plus loin. Nous allons maintenant réaliser un Hello World sous la forme d'une API web. Pour cela, nous allons avoir besoin de quelques librairies supplémentaires. Vous allez voir que c'est extrêmement simple. Comme la plupart des plateformes de développement, node est fourni avec quelques outils. Celui que vous utiliserez probablement le plus est un outil nommé « npm » pour Node Package Manager. Comme son nom l'indique, npm vous permet de gérer les packages JavaScript, mais aussi de faire des scripts. Il est aujourd'hui très fréquent qu'une librairie JavaScript que vous aurez glanée sur le Web s'installe et s'initialise en utilisant npm. Découvrons son mode de fonctionnement. Pour commencer nous allons initialiser un package pour contenir notre code en lançant la commande suivante :

```
npm init
```

Npm va vous poser plusieurs questions sur votre package. Son nom, sa version, son point d'entrée, etc. A l'issue de cette commande, votre répertoire va contenir un fichier « package.json » qui contient les informations que vous aurez saisies. En plus de ces métadonnées, ce fichier va être utilisé par npm pour référencer les librairies utilisées par notre package, que ce soit pour les outils de développement ou pour les dépendances nécessaires pour son exécution. Pour notre tutorial, nous allons utiliser une librairie appelée « Express ». Elle permet de réaliser des sites et des API Web. Pour l'installer lancez la commande :

```
npm install express
```

Votre répertoire va maintenant contenir un sous répertoire « node_modules », lui-même contenant de multiples sous-répertoires, dont un répertoire « express ». Les autres répertoires présents dans le dossier « node_modules » correspondent aux librairies dont express dépend. Si vous consultez votre fichier package.json, vous verrez qu'il ne contient pas de trace de la librairie que nous avons ajoutée. C'est parce que npm n'ajoute pas les dépendances automatiquement (en tout cas par défaut). Pour ce faire, modifions légèrement notre commande :

```
npm install express --save
```

L'option « --save » permet de préciser à npm qu'on souhaite garder une référence à cette dépendance. Si l'on installe une dépendance vers un outil de développement, on peut utiliser « --save-dev » pour préciser que cette dépendance concerne un outil de développement. Installons par exemple un outil pour vérifier que notre code respecte les best practices de développement JavaScript :

```
npm install eslint --save-dev
```

Si vous consultez votre fichier "package.json" vous verrez que vous avez maintenant des noeuds "dependencies" et "devDependencies" qui comportent des références vers les librairies utilisées. L'intérêt de cette déclaration est de pouvoir déployer votre code sans nécessairement déployer les dépendances associées. Le test est très simple à faire : supprimez le répertoire node_modules et lancez la commande :

```
npm install
```

En l'absence d'arguments, npm va utiliser le fichier package.json pour lister et installer les dépendances de notre package. Pour un déploiement en mode production on peut également faire une installation sans les packages de développement en ajoutant un argument :

```
npm install -p
```

Npm propose de nombreuses options comme la possibilité d'installer un package de façon globale sur la machine, installer une version spécifique d'un package, etc. Reportez-vous à la documentation en ligne pour en savoir plus. Vous pouvez aussi utiliser la commande pour afficher l'aide :

```
npm - ?
```

Mise en place de Express

Nous allons maintenant mettre en place notre serveur Web. Commencez par créer un fichier « index.js » qui sera le point d'entrée de notre package. Dans votre fichier, ajoutez les lignes de code suivantes :

```
const port = process.env.PORT || 4242;
const express = require('express');
const server = express();
```

Nous commençons tout d'abord par définir le port sur lequel notre serveur Web va écouter. Nous prenons le port assigné au processus de node dans une variable d'environnement, ou le port 4242 si aucun port n'est assigné. Ce contrôle est important car cela vous permet de pouvoir définir le port par configuration. Ce sera nécessaire par exemple si vous hébergez votre site dans unCcloud en mode PaaS (dans ce genre d'hébergement, c'est l'hébergeur qui vous assigne un port). La deuxième ligne de code nous permet de découvrir une notion très importante en node.js et en JavaScript moderne : la notion de module. Cette ligne de code pourrait être transcrite en « importe le module nommé express dans une constante nommée express ». Le nom du module correspond ici au nom du package que nous avons installé avec npm. A l'exécution, node va voir que l'import du module n'est pas un chemin vers un fichier, et va donc aller chercher le module dans le répertoire « node_modules ». La notion de module est importante car elle permet de structurer votre code en isolant les différentes composantes de votre application. Un module permet de définir explicitement les parties du code qui seront exposées aux consommateurs du module. Les autres portions de code resteront internes au module et seront invisibles de l'extérieur. La troisième ligne de code nous permet de créer une instance de serveur Express. C'est cette instance que nous allons utiliser pour configurer et faire fonctionner notre serveur Web.

Création d'un module

Nous allons maintenant créer notre API dans un module séparé. Créez un fichier « api.js » et ajoutez le code suivant :

```
function helloworld(request, response){
  response.status(200).send({ hello: "world" });
}
function configureApi(server) {
  server.get('/helloworld', helloworld);
}
module.exports = configureApi;
```

Nous déclarons ici une fonction « helloworld » qui va traiter une requête en renvoyant une réponse json. Notre module comporte également une fonction « configureApi » qui va recevoir notre serveur Express en argument, et dans laquelle nous allons mettre en place les différentes routes de notre API. Dans notre cas, nous déclarons une route accessible en « GET » sur l'url [url du site]/helloworld. Notre module exporte ensuite cette fonction de configuration. Nous exportons ici une fonction mais vous pouvez tout aussi bien exporter un objet, un tableau, etc. Seuls les éléments exportés seront accessibles par les portions de code qui vont utiliser notre module. Dans notre cas par exemple, la fonction « helloworld » sera totalement invisible en dehors du module.

Configuration du serveur Web

Maintenant que notre module gérant l'API est prêt, retournez dans votre fichier index.js, et ajoutez l'import de votre module :

```
const api = require('./api');
api(server);
```

Nous récupérons ici notre module en donnant son chemin (notez que l'extension du fichier « .js » est optionnelle) et nous l'invoquons pour mettre en place notre API. Reste maintenant à mettre en place notre serveur Web. Ajoutez le code suivant :

```
server.listen(port, '0.0.0.0', function onStart(err) {
  if (err) {
    console.error(err);
   return;
  }
  console.log('==> 🌎  Listening on port %s. Open up http://0.0.0.0:%s/ in your
browser.', port, port)
});
```

Nous demandons ici à notre serveur d'écouter sur le port que nous avons déclaré, sans spécifier de nom d'hôte pour notre serveur Web. Nous passons également une fonction de callback nous permettant de nous assurer que notre serveur est bien démarré. Il ne reste plus qu'à lancer notre serveur. Ouvrez une invite de commande dans le répertoire et lancez la commande suivante :

```
node index.js
```

Vous devriez voir le message « Listening on port 4242 » dans la console. Ouvrez un navigateur et accédez à l'api que nous avons déclarée en utilisant l'url http://localhost:4242/helloworld.

Utilisation de npm pour lancer des scripts

Nous avons vu que npm permet de gérer les dépendances vers d'autres librairies. Npm permet aussi de déclarer des scripts qui vont pouvoir être lancés très facilement. Une convention sur les packages node est de proposé un script permettant d'initialiser et démarrer le package avec une commande start. Ouvrez votre fichier « package.json ». Il devrait contenir un nœud « scripts » avec une propriété « test ». Remplacez le nœud « scripts » avec le suivant :

```
"scripts": {
  "start": "node index.js"
 },
```

Ouvrez une invite de commande dans le répertoire de votre package et lancez la commande

```
npm start
```

Allez plus loin ...

Nous avons vu ensemble les principes fondamentaux pour bien démarrer avec node.js. Mais pour ceux qui souhaitent se lancer, vous pouvez faire un tour sur les plateformes d'apprentissage comme nodeschool (https://nodeschool.io) et faire quelques tutoriels pour découvrir tous les aspects de Node.js.

Le mot de la fin

Comme vous avez pu le voir, JavaScript est très riche de par ce qu'il propose techniquement, mais également et surtout de par sa communauté. Néanmoins, c'est une plateforme qui reste très facile d'accès. Pour bien commencer, n'hésitez pas à aller sur Internet, il existe beaucoup de site Internet très bien faits qui proposent des tutoriels soit pour découvrir JavaScript, soit pour aller plus loin (même pour les développeurs confirmés). Pour plus de facilités dans le développement, n'hésitez pas à installer un éditeur de code comme Atom (https://atom.io/) ou Visual Studio Code (https://code.visualstudio.com/) qui, en plus des fonctionnalités classiques, vous permettront de debugger votre code. •

Intégration des services Office 365 avec Microsoft Graph

Eric Vernié
Fier d'être développeur

Microsoft Office 365 est un ensemble de services et d'applications, mis à la disposition des particuliers ou des entreprises. Nous y retrouverons les services tels que Exchange Online, Skype For Business Online, SharePoint Online, Office Vidéo et d'autres comme Yammer et des applications traditionnelles telles que, Excel, Word, PowerPoint, Outlook, OneNote pour les plus connues. Ces dernières sont d'ailleurs disponibles à la fois en ligne, ou sur le bureau après téléchargement.

Le développement sur la plateforme Office 365, s'articule autour de deux axes principaux :

- Enrichir les applications Office 365 sous forme de compléments ;
- Enrichir sa propre application en y intégrant les services Office 365.

Dans ce dossier, nous porterons notre attention sur l'intégration des Services Office 365 au travers de l'utilisation du nouveau service **Microsoft Graph** et de son API associée. Office 365 étant une plateforme ouverte, vous pourrez développer n'importe quel type d'application, Web, Native, Mobile, avec n'importe quel type de langage et sur n'importe quel type d'OS.

Cloisonnement des Services Offices 365

Pourquoi Microsoft Graph ? Pour répondre à cette question, il faut se rappeler que les données d'entreprises sont cloisonnées, ayant pour conséquence directe que tous les services associés le sont aussi. Par exemple, pour accéder aux 10 premières personnes de mon organisation voici la requête Rest à utiliser.

https://graph.windows.net/myorganization/users?api-version=1.6&$top=10

De la même manière voici la requête pour accéder aux 10 premiers fichiers sur OneDrive For Business

https://[montenant]-my.SharePoint.com/_api/v2.0/drive/root/children?$top=10

Et enfin pour les 10 premiers Messages

https://outlook.office.com/api/v2.0/me/messages?$top=10

Comme vous pouvez le constater, **à chaque service correspond une ressource différente**, indépendante les unes des autres, cloisonnée dans un **environnement de sécurité** qui lui est propre. Ce que je veux souligner ici, c'est qu'une autorisation obtenue pour accéder à One Drive For Business ne peut être utilisée que pour One Drive For Business et non pas pour Exchange, ni pour SharePoint, ni pour l'annuaire d'entreprise. En tant que développeur, ce cloisonnement nous oblige à des gymnastiques algorithmiques lorsque l'on souhaite faire la liaison entre chaque entité. C'est la raison d'être de **Microsoft Graph** qui a vu je jour, pour décloisonner ces services, offrir **un seul point de terminaison**, accessible via **une seule autorisation d'accès**. https://Graph.Microsoft.com/ comme illustré sur la figure suivante : [1]

Utilisation du service de Microsoft Graph d'Office 365.

A vous d'imaginer comment vous pourriez simplifier la vie de vos utilisateurs et ce que pourrait devenir votre application, si vous pouviez accéder simplement à leurs calendriers, envoyer en leur nom des courriers électroniques, partager leurs fichiers avec d'autres utilisateurs, accéder à l'annuaire de l'entreprise, connaître l'état de présence des utilisateurs de l'entreprise, les groupes auxquels ils appartiennent, lire des vidéos, intégrer des flux de conversations etc. Mais avant d'imaginer, il est important de comprendre comment ça marche.

1

Pour utiliser les différents services Office 365, il faut **s'authentifier** et obtenir les **autorisations** nécessaires pour y accéder. A cette fin, Office 365 s'appuie sur **Azure Active Directory** qui est la plateforme de gestion des identités de Microsoft, exposée en tant que service. Azure Active Directory fournit aux développeurs un moyen efficace pour intégrer la gestion des identités dans leurs applications en utilisant les protocoles modernes et standard, **OAuth 2.0** ([RFC 6749], [RFC 6750]) et **OpenId Connect (**http://openid.net/specs/openid-connect-basic-1_0.html).

Note : *En souscrivant à Office365 vous avez de facto accès à Azure Active Directory, et si vous souhaitez tester les services, rendez-vous à cette adresse* http://dev.office.com/devprogram*, une souscription gratuite d'un an vous attend.*

Azure Active Directory étant un service dans le cloud, il faut qu'il puisse identifier votre application. C'est pourquoi il est nécessaire de **l'enregistrer** dans l'annuaire que vous allez utiliser. Attention, je prends ici un raccourci qui peut amener à se poser des questions du style : dois-je enregistrer cette application dans tous les Azure AD de mes clients ? Bien évidemment non, avec la notion d'application mutualisée.

Le développement autour d'Azure AD étant un sujet à part entière, et si vous souhaitez aller plus loin, je ne peux que vous conseiller le **Guide du développeur Azure Active Directory** http://aka.ms/aadfrdev

Toutefois, un moyen simple pour enregistrer votre application en mode découverte des APIs, et ceci sans passer par le portail Azure Active Directory, est d'utiliser l'application Web à cette adresse. http://dev.office.com/app-registration.

En simplifiant au maximum, accéder aux services Office 365 implique 4 phases

- Enregistrer l'application auprès d'Azure Active Directory
- L'application permet à l'utilisateur de s'authentifier auprès de l'annuaire de son entreprise.

- Une fois authentifié, l'utilisateur consent à votre application d'agir en son nom, (exemple envoyer un message).
- L'application reçoit **un jeton d'accès** l'autorisant à manipuler les données de l'utilisateur authentifié.

Note : *Selon le type et l'architecture de votre application, que ce soit de la simple page Web, de l'application native, ou d'une application Web, et autres, il existe différents scénarios à prendre en compte, que vous pourrez retrouver à cette adresse.* http://aka.ms/aadfrscenarios

Une fois le jeton d'accès obtenu, **la bonne nouvelle**, c'est que tous ces services sont accessibles via les **standards REST**. En conséquence de quoi, peu importe, le langage, la plateforme et les appareils que vous utilisez ou que vous ciblez, vous pourrez y accéder tout simplement via une requête http. Toutefois, il existe de nombreux SDK propres à chaque plateforme qui permettent de s'abstraire d'un certain nombre de contraintes. SDK disponibles à cette adresse : https://github.com/OfficeDev

Exemple de SDK disponibles :
https://github.com/OfficeDev/Office-365-SDK-for-iOS
https://github.com/OfficeDev/Office-365-SDK-for-Android
https://github.com/OfficeDev/Office-365-SDK-for-Java

Pour illustrer mes propos, voici la requête REST pour lire les 10 premiers messages d'un utilisateur authentifié.
Get https://Graph.Microsoft.com/v1.0/me/messages?$top=10
La réponse est au format JSON dont voici un extrait :


```
{
 "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users('admin%
40mondomaine.com')/messages",
 "@odata.nextLink": "https://graph.microsoft.com/v1.0/me/messages?$top=10&
$skip=30",
 "value": [
  {
   "@odata.etag": "W/\"CQAAABYAAAAaXZvJHrZF+yYFJsHLAABziosD\"",
   "id": "AAMkADVkNjA2Y2RI_1y0AAA=",
   "categories": [],
   "receivedDateTime": "2016-02-02T03:15:52Z",
   "sentDateTime": "2016-02-02T03:15:45Z",
   "hasAttachments": false,
   "subject": "2 Days Left to Clean up your Address Book!",
   "body": {
    "contentType": "html",
```


De la même manière pour requêter les fichiers sur OneDrive For Business
Get https://Graph.Microsoft.com/v1.0/me/drive/root/children?$top=10


```
{
 "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users('admin
%mondomaine.com')/drive/root/children
 "value": [
  {
   "createdBy": {
    "application": {
     "id": "678cc2c2-2a68-4710-b2a3-...",
     "displayName": "Zotus Jide"
    },
    "user": {
     "id": "0fa8f787-1372-43a2-627002....",
     "displayName": "Peter Parker"
    }
   },
    "user": {
     "id": "0fa8f787-1372-43a2-627002....",
     "displayName": "Peter Parker"
    }
   },
   "lastModifiedDateTime": "2016-02-08T16:29:40Z",
   "name": "My First Presentation.pptx",
```


Ou pour récupérer les utilisateurs de mon organisation
Get https://Graph.Microsoft.com/v1.0/users


```
{
 "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users",
 "value": [
  {
   "businessPhones": [],
   "displayName": "Peter Parker",
   "givenName": "Peter",
   "jobTitle": "Admin",
   "mail": "pp@mondomaine.com",
   "mobilePhone": "+33 11111111",
   "officeLocation": null,
   "preferredLanguage": "fr-FR",
   "surname": "Parker",
```


Mais Microsoft Graph ne s'arrête pas là, bien au contraire. Tout d'abord, dans Office 365 vous avez depuis quelques temps la notion de **groupe office 365**, qui est un espace de travail partagé pour le courrier électronique, les conversations, les fichiers, les blocs-notes et les événements de calendrier, au sein duquel les membres du groupe peuvent collaborer et accomplir leurs tâches rapidement comme illustré sur la figure suivante : [2]
Pour requêter un groupe c'est simple, il suffit de retrouver son id
Get https://Graph.Microsoft.com/v1.0/groups


```
{
 "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#groups",
 "value": [
  {
   "id": "01baf240-dd86....",
   "createdDateTime": null,
   "description": "Ce groupe a pour vocation de tester l'API Microsoft Graph",
   "displayName": "GroupeTravauxCollaboratifs",
   "groupTypes": [
```


2

```
    "Unified"
  ],
```

Ensuite il est possible de récupérer avec cet id, la liste des membres, la conversation, les fichiers partagés par le groupe, les évènements du calendrier, etc.

Get https://Graph.Microsoft.com/v1.0/groups/01baf240-dd86.../members
Get https://Graph.Microsoft.com/v1.0/groups/01baf240-dd86.../conversations/
Get https://Graph.Microsoft.com/v1.0/groups/01baf240-dd86.../drive/root/children
Get https://Graph.Microsoft.com/v1.0/groups/01baf240-dd86.../events

Tout ce que nous venons d'aborder est déjà disponible dans sa version v1.0 et vous pouvez commencer à l'intégrer dans du code de production. Néanmoins il serait réducteur d'en rester là. En effet Microsoft Graph est bien plus qu'une unification des APIs Office 365 ; c'est essentiellement un système **qui met l'utilisateur au centre de son organisation** et constitue **des relations** entre chaque entité, afin de réaliser un graphe autour de lui. [3]

Graphe, qui sera constitué des membres de son organisation, des relations de travail internes et externes avec lesquelles l'utilisateur travaille le plus, des données qui constituent son environnement de travail, les messages, les évènements d'un calendrier, les fichiers partagés, etc.

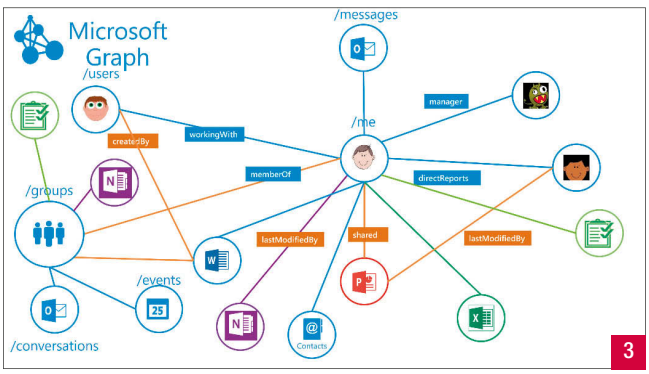
Microsoft Graph, utilise des techniques de Machine Learning, afin d'agréger dans le graphe de l'utilisateur et ceci de manière sécurisée, les données qui proviennent de tous les services Office 365.

Par exemple, vous pouvez retrouver pour un utilisateur, les documents les plus appropriés autour de lui, ceux sur lesquels il a travaillé, ceux qu'un autre utilisateur aurait partagés avec OneDrive For Business, voir ceux attachés comme pièces jointes dans un courrier électronique. C'est la relation **trendingAround** (en beta actuellement) qui permet ça.

Get https://Graph.Microsoft.com/beta/me/trendingAround

```
{
 "@odata.context": "https://graph.microsoft.com/beta/$metadata#users('admin
%mondomaine.com')/TrendingAround",
 "value": [
  {
   "@odata.id": "Drives('NWZiMjdkYmUtZGVmMC00ZjZmODAtO...YjU')/items
('01BCMELZXP6IYD...')",
   "id": "01BCMELZXP6IYD ",
   "name": "Travaux collaboratifs.pptx",
   "DateTimeCreated": "2016-01-11T20:28:14Z",
   "DateTimeLastModified": "2016-01-15T12:06:17Z",
   "webUrl": "https://MonTenant-my.sharepoint.com/personal/user2_www_
mondomaine_com/Documents/Travauxcollaboratifs.pptx"
```

Il est possible de requêter les personnes avec lesquelles il travaille le plus, avec la relation **workingWith**



3

Get https://Graph.Microsoft.com/beta/me/workingWith.

Remarque : Comme Microsoft Graph utilise une technique de machine learning, il est probable que vous ne voyiez pas l'effet immédiat de ces deux relations dans votre graphe. Il faut être patient et attendre que le système prenne le temps de l'analyser.

Note : *Une des 1eres applications à utiliser Microsoft Graph est l'application Delve disponible avec Office 365.*

Si vous n'avez pas le temps de vous plonger dans le code pour tester l'API, vous pouvez utiliser le lien suivant https://msgraph.azurewebsites.net/Home/GraphExplorer afin de la tester in situ.

Pour les autres, nous allons créer une simple application console sur la plateforme .NET, mais les mêmes principes sont peu ou prou identiques avec les autres plateformes. Si vous souhaitez en savoir plus, rendez-vous à cette adresse https://graph.microsoft.io/en-us/docs/platform/rest ou vous trouverez des exemples pour des plateformes telles que IOS, Android, Windows, pour des frameworks tels que ASP.NET MVC, UWP, Angular, .NET, node.js, pour des langages tels que Python, PHP, Ruby, C#.

Accéder aux services Office365

Pour utiliser les services Office 365, il vous faut donc une licence Office365. Vous pourrez en obtenir une à cette adresse gratuitement et ceci pour un an : http://dev.office.com/devprogram.

Enregistrer l'application dans Azure Active Directory

Comme je le disais précédemment, il faut enregistrer l'application dans Azure Active Directory. Si vous n'avez pas de souscription Azure vous pouvez procéder comme suit :

- Allez à l'adresse https://graph.microsoft.io/en-us/app-registration ;
- Connectez-vous à l'application **dev.office.com** avec un compte Admin Globale d'Office 365 ;
- Remplissez les différents champs comme suit :
  - App name : **Demo Programmez Microsoft Graph.**
  - App type : **Native App.**
  - Redirect URI : https://localhost:8080.
  - App permissions : **Tout cocher.**
- Enregistrez l'application, en actionnant la commande **Register App ;**
- Une fois l'application enregistrée, n'oubliez pas de copier le **Client Id**, qui a été retourné ainsi que l'url de redirection, car nous les utiliserons plus tard dans le code.

Note : *Des petites vidéos de 3 minutes sur l'intégration des applications dans Azure Active Directory avec le portail Azure sont disponibles ici :*
Partie 1 : https://youtu.be/IC4cLX6GmJ0
Partie 2 : https://youtu.be/j80VIRdF4s0
Partie 3 : https://youtu.be/PX6MZD5OFEU

Créer une application d'accès aux services Office 365

Dans Visual Studio 2015 (à télécharger à cette adresse http://aka.ms/FreeVS2015)

- Créez un nouveau projet de type **Console.**
- Ouvrez la console des packages nuggets : **Tools | Nugget Package Manager | Package Manage Console**.
- Installez le package pour Azure Active Directory :

PM> **install-package Microsoft.IdentityModel.Clients.ActiveDirectory.**

- Installez le package pour Microsoft Graph :

PM> `install-package Microsoft.Graph`

Authentification/Autorisation avec Azure Active Directory

Les principes fondamentaux de l'authentification avec Azure AD pour notre application native sont simples avec la librairie ADAL .NET que nous venons d'installer.

- L'ADAL présente à l'utilisateur une fenêtre de connexion pour qu'il s'authentifie, Azure AD retourne à l'application un code d'authentification.
- Si l'utilisateur n'a pas déjà approuvé que l'application pouvait agir en son nom, une phase de consentement est demandée.
- Avec ce code d'authentification, l'ADAL demande un jeton d'accès à la ressource.

Ce jeton est porteur (*bearer*) non seulement des autorisations pour accéder à la ressource en question, mais contient également un jeton (*Refresh token*) qui permettra à l'application de demander un nouveau jeton d'accès à Azure AD de manière silencieuse, c'est-à-dire sans interaction avec l'utilisateur. Ensuite voici comment procéder :

- Tout d'abord il faut instancier un contexte d'authentification, en lui précisant l'autorité que nous allons utiliser, en l'occurrence ici une URL qui pointe sur Azure Active Directory.

```
AuthenticationContext authenticationContext =
        new AuthenticationContext("https://login.microsoftonline.com/common", false);
```

Puis il faut demander le jeton d'accès à l'aide de la méthode *AcquireToken*, en lui passant comme paramètres,

- La ressource pour laquelle on souhaite obtenir les autorisations,
- Le Client ID obtenu lors de l'enregistrement de l'application et qui permettra d'identifier votre application auprès d'Azure AD.
- L'adresse de redirection, qui sera utilisée par l'ADAL pour récupérer le code d'authentification.
- Et enfin le paramètre *PromptBehavior.RefreshSession*, qui permet d'éviter à l'utilisateur de ré-entrer ses informations de connexion lorsque le jeton a expiré.

```
AuthenticationResult userAuthnResult =
            authenticationContext.AcquireToken(
                        "https://graph.microsoft.com/",
                        "[VOTRE CLIENT ID]",
                         new Uri("https://localhost:8000"),
                         PromptBehavior.RefreshSession);
```

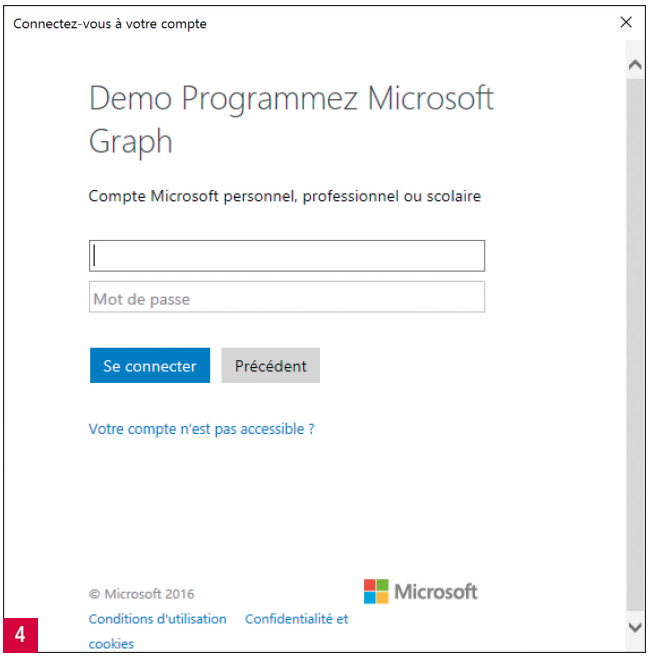
- Lorsque la méthode *AcquireToken* est invoquée, l'ADAL requête l'autorité avec une URL du type :

```
GET /common/oauth2/authorize?resource=https://graph.microsoft.com/&
client_id=[VOTRE CLIENT ID]&response_type=code&redirect_uri=https://local
hsot:8080/&prompt=refresh_session
```

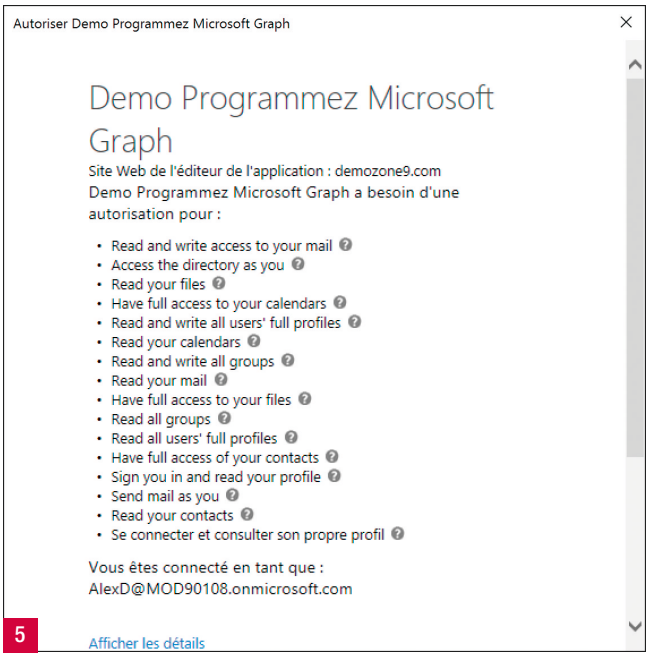
- Azure AD, retourne la page de connexion suivante, que l'ADAL affiche dans une simple WebView [4]
- Si l'utilisateur s'authentifie pour la 1ere fois, Azure AD retourne une page de consentement [5]
- Une fois approuvé par l'utilisateur, l'ADAL récupère le code d'authentification et demande le jeton d'accès avec une URL du type :

```
POST /common/oauth2/token et les paramètres suivants :

resource=https://graph.microsoft.com/&client_id=[VOTRE CLIENT ID]&grant_
type=authorization_code&code=AAABAAAAiL9.. &redirect_uri=https://Localhost:8080/
```

Voici le listing complet :



4



5

```
private static string TokenForUser;
private static DateTimeOffset expiration;
public static string GetToken()
{
 if (TokenForUser == null || expiration <= DateTimeOffset.UtcNow.AddMinutes(5))
 {
    var redirectUri = new Uri("https://localhost:8000");
    string clientId = "[VOTRE CLIENT ID]";
    string authority = "https://login.microsoftonline.com/common";
    string ResourceUrl = "https://graph.microsoft.com/";
    AuthenticationContext authenticationContext =
                new AuthenticationContext(authority, false);
    AuthenticationResult userAuthnResult =
                authenticationContext.AcquireToken(
                ResourceUrl,
                clientId,
```

```
                redirectUri,
                PromptBehavior.RefreshSession);
            TokenForUser = userAuthnResult.AccessToken;
            expiration = userAuthnResult.ExpiresOn;
    }
        return TokenForUser;
}
```

Dans la méthode *GetToken* avant de demander un nouveau jeton, on vérifie s'il n'a pas expiré.

Accéder aux services Office 365 Microsoft Graph

Maintenant que nous avons notre méthode pour récupérer un jeton valide, nous allons accéder à nos différents services Office 365 à l'aide de l'API .NET Microsoft Graph.

• Tout d'abord il faut instancier le service

```
GraphServiceClient graphClient = new GraphServiceClient(
        new DelegateAuthenticationProvider(
          (requestMessage) =>
          {
            requestMessage.Headers.Authorization =
                    new AuthenticationHeaderValue(
                        "bearer",
                        GetToken());
            return Task.FromResult(0);
          }));
```

La classe *GraphServiceClient*, prend en paramètre un **délégué**, qui sera invoqué à chaque demande de ressources.

Ce délégué a pour tâche, de récupérer le jeton d'accès et de l'inscrire dans l'entête http comme paramètre d'autorisation d'accès à la ressource.

• Ensuite il suffit d'accéder à la ressource que l'on souhaite.

Par exemple pour accéder aux informations de l'utilisateur authentifié.

```
var me = graphClient.Me.Request().Select("DisplayName,Department").GetAsync().Result;
Console.WriteLine($"Nom : {me.DisplayName}");
Console.WriteLine($"Departement : {me.Department}");
```

Ici on demande à l'API de ne récupérer que les champs DisplayName et Department du profile avec la méthode *Select* qui est en fait le reflet de paramètres optionnels OData (**http://graph.microsoft.io/en-us/docs/overview/query_parameters**).

Dans la même veine, si on souhaite les 10 premiers messages, nous utiliserons le paramètre *Top*.

```
var messages = graphClient.Me.Messages.Request().Top(10).GetAsync().Result;
foreach(var message in messages)
{
  Console.WriteLine($"Sujet :{message.Subject}");
}
```

Filtrer les fichiers en fonction du début de leur nom en utilisant la méthode *Filter*.

```
var OneDriveFiles = graphClient.Me.Drive.Root.Children.Request().Filter("startswith(name,'fusion')")
.GetAsync().Result;
foreach (var item in OneDriveFiles)
{
  Console.WriteLine($"Nom :{item.Name}");
}
```

Il est possible de combiner les différentes options, les 10 premiers fichiers, ordonnés par nom (*OrderBy*).

```
var OneDriveFiles = graphClient.Me.Drive.Root.Children.Request().Top(30).OrderBy("name")
.GetAsync().Result;
```

Enfin on peut pour récupérer les utilisateurs de l'organisation.

```
var users = graphClient.Users.Request().GetAsync().Result;
foreach(var user in users)
{
  Console.WriteLine($"Utilisateur : {user.DisplayName}");
}
```

Et ainsi de suite pour les autres services, vous avez compris le principe.

• Néanmoins, si pour une raison ou pour une autre, une fonctionnalité n'est pas couverte par le modèle, il est toujours possible d'utiliser la couche transport http de l'API Microsoft Graph, *HttpProvider*, en y injectant des requêtes REST directement.

• Dans cet exemple, on souhaite partager un fichier avec des membres de l'organisation :

```
string uriRequestUrl = "https://graph.microsoft.com/beta/me/drive/items/[ID]/microsoft.graph.invite";

HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Post, uriRequestUrl);
string content = "{ 'requireSignIn':false,'sendInvitation':true,'roles':['write'],'recipients':[{'email':ftonic@modomaine.com'}],'message':'Bonjour François, je vous partage le fichier'}";

request.Content = new StringContent(content, Encoding.UTF8, "application/json");
await graphClient.AuthenticationProvider.AuthenticateRequestAsync(request);

var response = await graphClient.HttpProvider.SendAsync(request);
```

L'astuce ici, est d'utiliser la méthode *AuthenticateRequestAsync*, pour que la requête bénéficie automatiquement du contexte de sécurité et de créer un contenu au format JSON permettant d'inviter une ou plusieurs personnes à collaborer sur le fichier.

Note : *Retrouvez également des vidéos de 3 minutes qui explique la manière de faire*
Partie 1: **https://youtu.be/CeKjHglnFmw**
Partie 2: **https://youtu.be/gULyiiU2qqE**
Partie 3: **https://youtu.be/iY3mWXJAYq8**
Partie 4: **https://youtu.be/l9n4zkP89Qg**
Partie 5: **https://youtu.be/_lokttobKcY**

CONCLUSION

L'API Microsoft Graph, est une librairie simple d'emploi, qui unifie les différents services Office 365, avec l'utilisation d'un seul point de terminaison (**https://microsoft.graph.com**) en lieu et place des autres points de terminaison. Conjuguée avec les librairies d'authentification et d'autorisation d'Azure Active Directory, elle se contente d'un seul Jeton d'accès pour accéder à tous les services Office 365. Aujourd'hui en version 1.0, elle évoluera très rapidement, car Microsoft Graph, c'est aussi un service qui collecte et analyse en continu les signaux que vous et vos collègues échangez lorsque vous travaillez dans Office 365. Par exemple, le signal que vous et un collègue modifiez ou affichez le même document indique à Microsoft Graph que vous êtes susceptibles de travailler ensemble. Les autres signaux analysés sont les personnes avec lesquelles vous communiquez par courrier électronique et avec lesquelles vous avez partagé des documents; ils déterminent qui est votre responsable et qui a le même responsable que vous. •

Pour aller plus loin une seule adresse : **http://dev.office.com**

Premiers pas avec **Android Things** la plateforme IoT de Google

• Sylvain Saurel
sylvain.saurel@gmail.com
Développeur Android
https://www.ssaurel.com

En fin d'année 2016, Google a dévoilé Android Things, sa version d'Android taillée sur mesure pour les objets connectés. Android Things propose aux développeurs Android d'utiliser tous leurs outils habituels pour réaliser des applications à destination des objets connectés. Cette initiative doit permettre de faciliter et populariser le développement pour l'Internet des Objets. Dans cet article, nous vous proposons d'effectuer vos premiers pas avec Android Things.

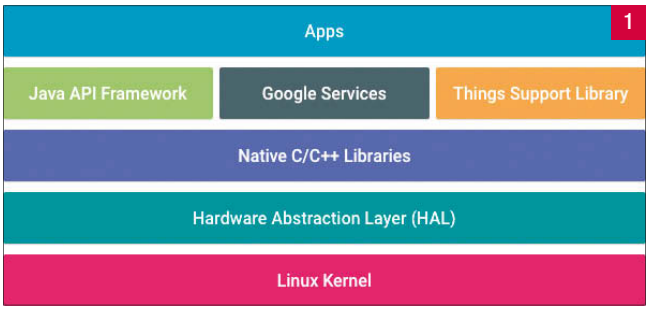
Lancé par Google en fin d'année 2016, Android Things est une plateforme à destination de l'Internet des Objets basée sur une version taillée sur mesure du système d'exploitation Android. Plus connu via l'acronyme IoT (pour Internet of Things), l'Internet des Objets sera le terrain de la prochaine bataille que vont se livrer les géants du Web. En effet, le monde des objets connectés est promis à un avenir brillant et Google veut que son système d'exploitation maison Android soit au coeur de cette prochaine vague technologique.

Pour ce faire, la firme de Mountain View a décidé de proposer une plateforme IoT basée sur Android et adaptée aux contraintes des objets connectés. Android Things va ainsi permettre de construire des objets intelligents et de les faire fonctionner avec les APIs d'Android et les services de Google. La stack complète des outils de développement Android est supportée, ce qui permettra aux développeurs d'utiliser Android Studio ou bien encore les Google Play Services pour développer des applications pour l'IoT comme ils le font quotidiennement en développant des applications Android pour smartphones et tablettes.

En outre, les développeurs pourront également utiliser le protocole Google Weave pour faire communiquer des appareils entre eux ou avec des services Cloud tels que Google Cloud Vision. Afin de rendre Android Things directement exploitable, Google a travaillé en commun avec plusieurs constructeurs de hardware pour que sa plateforme soit supportée. Ainsi, la carte Intel Edison, la NXP Pico et la Raspberry Pi 3 supportent d'ores et déjà Android Things.

Architecture

Android Things rend le développement pour les objets connectés facile en fournissant aux développeurs les mêmes outils qu'ils connaissent sous Android mais également via une architecture en couches simplifiée (figure 1). Ainsi, les développeurs peuvent accéder de manière plus directe aux périphériques matériels et aux drivers que ce qu'il est possible de faire dans la version classique d'Android. Android Things étend le coeur traditionnel d'Android via des APIs additionnelles fournies par la bibliothèque Things Support Library. Ces APIs donnent la possibilité aux applications d'interagir au mieux avec les différents types de hardware que l'on ne rencontre pas dans le développement pour appareils mobiles. Enfin, la plateforme Android Things a été conçue pour un usage mono application. Ainsi, les applications systèmes ne sont pas présentes, et une application est lancée automatiquement au démarrage pour plonger les utilisateurs directement dans l'expérience de l'application.



Architecture d'Android Things

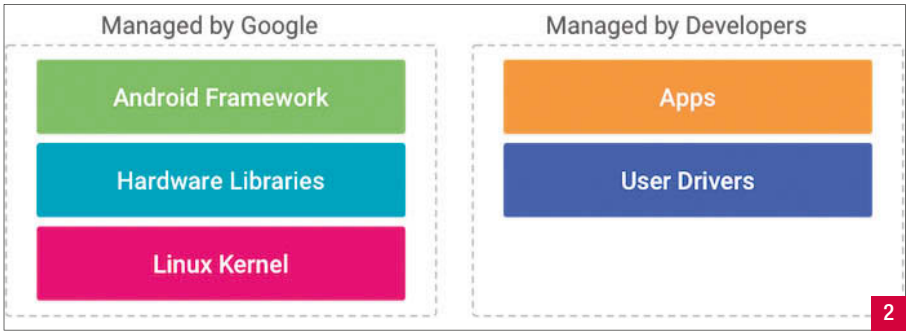
Things Support Library

Les APIs proposées au sein de la bibliothèque Things Support Library sont classées en 2 catégories. La première regroupe les APIs liées à la gestion des périphériques d'Entrée/Sortie (In/Out) qui vont permettre aux applications de communiquer avec les capteurs et les actionneurs en utilisant les protocoles standards et les interfaces de l'industrie. Les interfaces supportées sont :

- GPIO (General Purpose Input/Output) qui propose une API pour accéder à des capteurs simples comme les détecteurs de mouvement, les détecteurs de proximité ou les périphériques à état binaire tels qu'une LED.
- PWM (Pulse With Modulation) proposant une API pour les servomoteurs, les moteurs à courant continu ou les lumières nécessitant un signal proportionnel pour fournir un contrôle plus fin.
- I2C (Inter-Integrated Circuit) pour la communication en série synchrone avec une vitesse de transfert faible. Peut être utilisée avec des thermomètres, des accéléromètres ou des écrans LCD.
- SPI (Serial Peripheral Interface) pour la communication en série synchrone avec une vitesse de transfert rapide. Peut être utilisée pour communiquer avec de la mémoire externe non volatile ou des périphériques d'affichages graphiques.
- UART (Universal Asynchronous Receiver Transmitter) pour la communication série asynchrone avec une vitesse de transfert moyenne. Cela concerne la communication avec des modules GPS, des écrans LCD ou des radios XBee.

La seconde catégorie regroupe les APIs liées aux drivers utilisateurs. Ces derniers permettent d'étendre les services du framework Android et autorisent les applications à injecter des évènements hardware dans le

framework pour que d'autres applications puissent y accéder en utilisant les APIs Android (figure 2). Alors que dans la plupart des cas les APIs liées aux périphériques I/O suffiront pour communiquer directement avec des appareils externes, il se peut que les développeurs trouvent un avantage à intégrer leur hardware avec le reste du framework Android que ce soit pour des raisons de portabilité, de réutilisation ou d'intégration. Parmi les drivers utilisateurs, on pourra retrouver des modules GPS, HID venant offrir une interface utilisateur aux applications, ou des capteurs liés à l'environnement physique.



Stack des Drivers sous Android Things

Différences avec le SDK Android

Comme expliqué précédemment, Android Things ne supporte pas un certain nombre d'applications systèmes standards et de Content Providers puisqu'il a vocation à être installé sur des objets où ces applications n'ont pas de sens. Il faudra éviter d'utiliser un des Intents liés à l'horloge, au calendrier, à la téléphonie ou encore aux contacts. De même, les APIs standard d'Android liées au calendrier, aux contacts, au téléchargement ou encore à la téléphonie sont à éviter.

Android Things supporte les interfaces utilisateurs graphiques via le même UI Toolkit que les applications Android traditionnelles, à la différence près que la plateforme n'inclut pas de barre de statut système ou de boutons de navigation afin de donner aux applications un contrôle complet sur l'expérience visuelle de l'utilisateur. Cependant, il faut garder à l'esprit qu'Android Things ne requiert pas la présence d'un affichage graphique. Sur les périphériques dépourvus d'affichage graphique, les activités restent tout de même le point d'entrée d'une application Android Things puisque le framework va délivrer tous les évènements utilisateurs à l'activité en premier plan ayant le focus. Ainsi, une application ne peut recevoir d'évènements utilisateurs en entrée via d'autres composants applicatifs tels que les services.

Puisque la plateforme s'attend à avoir une activité comme point d'entrée d'une application, il faudra la définir au sein du manifest pour qu'elle soit automatiquement lancée au boot. Cette activité devra contenir un intent filter incluant à la fois l'intent CATEGORY_DEFAULT ainsi que l'intent IOT_LAUNCHER. Ceci permettant de lancer l'activité automatiquement au boot de l'appareil cible. En outre, pour des raisons de facilité de développement, il est intéressant d'inclure un intent filter incluant l'intent CATEGORY_LAUNCHER pour qu'Android Studio puisse lancer l'application. Au niveau du manifest, la déclaration de l'activité principale aura donc l'allure suivante :

```
<application
  android:label="@string/app_name">
  <activity android:name=".HomeActivity">
    <!-- Lancement via Android Studio -->
    <intent-filter>
      <action android:name="android.intent.action.MAIN"/>
      <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>

    <!-- Lancement au boot -->
    <intent-filter>
      <action android:name="android.intent.action.MAIN"/>
      <category android:name="android.intent.category.IOT_LAUNCHER"/>
      <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
  </activity>
</application>
```

Au niveau des services Google, Android Things supporte un sous-ensemble des APIs Google Services pour Android. De manière générale, les APIs nécessitant une entrée utilisateur ou une authentification ne sont pas disponibles pour les applications sous Android Things. Ainsi, des services Cast, Firebase Analytics, Firebase Storage ou encore Location sont accessibles. En revanche, AdMob, Android Pay, Firebase Notifications ou les services Play Games sont par exemple inaccessibles, ce qui apparaît comme logique.

Enfin, un mot sur les permissions qui nécessitent une demande d'autorisation auprès de l'utilisateur depuis Android Marshmallow. Puisque la présence d'une interface graphique n'est pas garantie avec les appareils tournant sous Android Things, il n'y a pas de gestion des permissions à l'exécution à effectuer, mais simplement une déclaration à réaliser au sein du manifest de l'application comme cela se faisait précédemment.

Préparation de l'Environnement

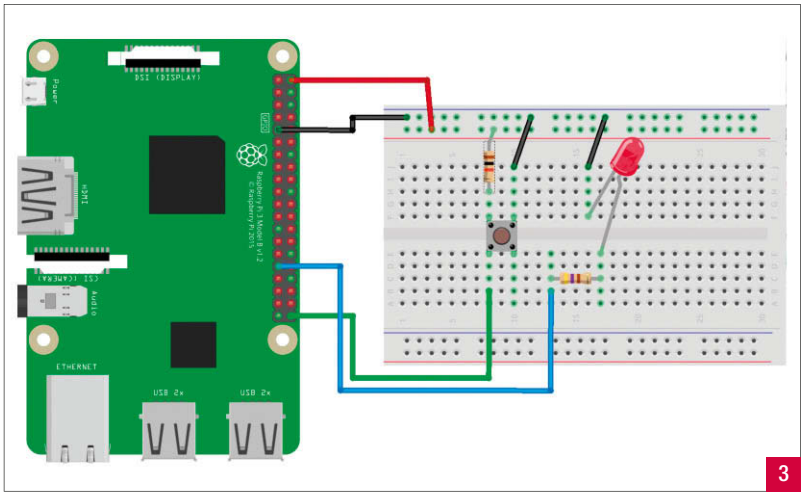
Pour pouvoir réaliser une première application tournant sous Android, vous devrez avoir à disposition un appareil supportant Android Things. La liste des objets supportant la plateforme peut être trouvée sur le site d'Android Things ici :
https://developer.android.com/things/hardware/developer-kits.html

Cette liste inclut le module de calcul Intel Edison, la carte NXP Pico i.MX6UL et le Raspberry Pi 3. D'autres appareils sont en préparation et devraient bientôt arriver sur le marché comme l'Intel Joule 570x et la carte NXP Argon i.MX6UL. Une fois une de ces cartes achetée, vous devrez la flasher pour y installer Android Things en suivant les instructions spécifiques qui vous seront détaillées sur le site du constructeur de votre carte. Ensuite, vous aurez besoin de la dernière version stable d'Android Studio sur votre machine de développement. A l'heure de l'écriture de cet article, il s'agit de la version 2.2. Le téléchargement se fait à partir du lien suivant : https://developer.android.com/studio/index.html

L'installation vous permettra également d'obtenir le SDK Android et tous les outils nécessaires au développement d'applications Android, et, par extension, d'applications pour Android Things. Cette phase d'installation terminée, nous allons pouvoir connecter notre carte compatible à notre machine de développement et exécuter la commande adb devices pour vérifier que l'appareil est bien reconnu par adb :

```
$ adb devices
List of devices attached
4560736843791520041    device
```

Si la carte est reconnue correctement par adb, cela signifie que l'installation s'est bien déroulée et que nous allons pouvoir poursuivre nos premiers pas avec Android Things. Pour s'amuser un peu plus, il peut être intéressant également d'acheter quelques périphériques externes à brancher sur votre carte. Sur le site de la plateforme, Google donne un lien vers un kit de projet Android Things à acheter et proposant tout un tas de périphériques qui permettent de s'initier en douceur au développement sous Android Things : https://www.adafruit.com/product/3227



Raspberry Pi 3 utilisée pour notre exemple

Avec le contenu de ce kit et avec une Raspberry Pi 3, nous sommes désormais prêts à réaliser une application Android Things permettant d'allumer une LED quand on pressera un bouton et d'éteindre cette même LED lorsque le bouton sera relâché (figure 3).

Réalisation d'une application

Une fois Android Studio lancé, il faut créer un projet de nouvelle application Android. Puisque nous allons utiliser un bouton LED qui sera allumé ou éteint, nous devons ajouter une dépendance Android Things spécifique pour le driver de bouton. En outre, il faut ajouter une dépendance à la Developer Preview d'Android Things et préciser que le projet doit être compilé avec l'API Android en version 24 ce qui nous donne le fichier build.gradle suivant :

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion '24.0.3'

    defaultConfig {
        applicationId "com.ssaurel.androidthings.button"
        minSdkVersion 24
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile 'com.google.android.things.contrib:driver-button:0.1'
    provided 'com.google.android.things:androidthings:0.1-devpreview'
}
```

Le manifest va permettre de définir le point d'entrée de l'application comme expliqué précédemment en ajoutant les intent filters adéquats pour l'activité ButtonActivity. Il faut également préciser que l'application va recourir à Android Things en ajoutant la ligne suivante au sein de la balise application :

```
<uses-library android:name="com.google.android.things"/>
On obtient ainsi le manifest suivant pour notre application :
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ssaurel.androidthings.button">

    <application android:allowBackup="true"
        android:icon="@android:drawable/sym_def_app_icon"
        android:label="@string/app_name">

        <uses-library android:name="com.google.android.things"/>

        <activity android:name=".ButtonActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.IOT_LAUNCHER"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Au niveau du code Java, nous commençons par créer un objet BoardDefaults avec des méthodes en charge de renvoyer le GPIO pour le bouton connecté et pour la LED connectée en fonction de la carte où l'application s'exécute. Ces méthodes utilisent une méthode getBoardVariant qui fait appel à l'API PeripheralManagerService contenue au sein de la bibliothèque Things Support Library d'Android Things. Tout ceci nous donne le code suivant :

```
package com.ssaurel.androidthings.button;

import android.os.Build;
import com.google.android.things.pio.PeripheralManagerService;
import java.util.List;

public class BoardDefaults {
  private static final String DEVICE_EDISON_ARDUINO = "edison_arduino";
  private static final String DEVICE_EDISON = "edison";
  private static final String DEVICE_RPI3 = "rpi3";
  private static final String DEVICE_NXP = "imx6ul";
  private static String sBoardVariant = "";

  /**
   * Renvoie le GPIO pin pour la LED connectée
   */
  public static String getGPIOForLED() {
    switch (getBoardVariant()) {
      case DEVICE_EDISON_ARDUINO:
        return "IO13";
      case DEVICE_EDISON:
        return "GP45";
      case DEVICE_RPI3:
        return "BCM6";
      case DEVICE_NXP:
        return "GPIO4_IO21";
      default:
        throw new IllegalStateException("Unknown Build.DEVICE " + Build.DEVICE);
    }
  }

  /**
   * Renvoie le GPIO pin pour le Bouton connecté
   */
  public static String getGPIOForButton() {
    switch (getBoardVariant()) {
      case DEVICE_EDISON_ARDUINO:
        return "IO12";
      case DEVICE_EDISON:
        return "GP44";
      case DEVICE_RPI3:
        return "BCM21";
      case DEVICE_NXP:
        return "GPIO4_IO20";
      default:
        throw new IllegalStateException("Unknown Build.DEVICE " + Build.DEVICE);
    }
  }

  private static String getBoardVariant() {
    if (!sBoardVariant.isEmpty()) {
      return sBoardVariant;
    }

    sBoardVariant = Build.DEVICE;

    // Exception pour gérer le cas particulier de  l'Intel Edison
    if (sBoardVariant.equals(DEVICE_EDISON)) {
      PeripheralManagerService pioService = new PeripheralManagerService();
      List<String> gpioList = pioService.getGpioList();
      if (gpioList.size() != 0) {
        String pin = gpioList.get(0);
        if (pin.startsWith("IO")) {
          sBoardVariant = DEVICE_EDISON_ARDUINO;
        }
      }
    }

    return sBoardVariant;
  }
}
```

Point d'entrée de l'application

Le point d'entrée de notre application Android Things sera l'activité ButtonActivity qui classiquement hérite de la classe Activity du SDK Android. A la création de l'activité, c'est-à-dire au boot de l'appareil cible, nous initialisons le service PeripheralManagerService. Puis, nous configurons la LED en appelant la méthode openGpio du service instancié précédemment en passant en entrée le GPIO pour la LED connectée, et nous fixons la direction de communication. Ensuite, nous répétons la même opération avec le bouton connecté en créant un objet ButtonInputDriver auquel nous passons à la construction le GPIO à utiliser, l'état logique considéré et le key event qui sera associé lorsqu'il sera pressé ou relâché. Enfin, nous appelons la méthode register sur cet objet pour valider son enregistrement.

Pour écouter les changements d'état du bouton, nous implémentons les méthodes onKeyDown et onKeyUp au sein de notre activité et nous réagirons lorsque le key event associé au bouton sera appelé. Si l'utilisateur presse le bouton, la méthode onKeyDown est appelée, et il est nécessaire d'allumer la LED. Si l'utilisateur relâche le bouton, la méthode onKeyUp est appelée et il est nécessaire d'éteindre la LED. Le contrôle sur l'état allumé / éteint de la LED connectée se faisant au travers d'un simple appel à la méthode setValue de l'instance d'objet Gpio retournée à l'établissement de la connexion à la LED via le service PeripheralManagerService.

La dernière étape consiste à bien libérer les connexions aux objets connectés à notre carte dans la méthode onDestroy de l'activité ButtonActivity. Au final, nous obtenons le code suivant :

```
package com.ssaurel.androidthings.button;

import android.app.Activity;
import android.os.Bundle;

import com.google.android.things.contrib.driver.button.Button;
import com.google.android.things.contrib.driver.button.ButtonInputDriver;
import com.google.android.things.pio.Gpio;
import com.google.android.things.pio.PeripheralManagerService;
import android.util.Log;
import android.view.KeyEvent;

import java.io.IOException;

public class ButtonActivity extends Activity {
  private static final String TAG = ButtonActivity.class.getSimpleName();
```

```
private Gpio mLedGpio;
private ButtonInputDriver mButtonInputDriver;

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 Log.i(TAG, "Démarrage");

 PeripheralManagerService pioService = new PeripheralManagerService();
 try {
  Log.i(TAG, "Configuration GPIO pour la LED");
  mLedGpio = pioService.openGpio(BoardDefaults.getGPIOForLED());
  mLedGpio.setDirection(Gpio.DIRECTION_OUT_INITIALLY_LOW);

  Log.i(TAG, "Enregistrement du driver pour le Bouton");
  mButtonInputDriver = new ButtonInputDriver(
  BoardDefaults.getGPIOForButton(),
  Button.LogicState.PRESSED_WHEN_LOW,
  KeyEvent.KEYCODE_SPACE);
  mButtonInputDriver.register();
 } catch (IOException e) {
  Log.e(TAG, "Erreur à l'initialisation", e);
 }
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
 if (keyCode == KeyEvent.KEYCODE_SPACE) {
  setLedValue(true);
  return true;
 }

 return super.onKeyDown(keyCode, event);
}

@Override
public boolean onKeyUp(int keyCode, KeyEvent event) {
 if (keyCode == KeyEvent.KEYCODE_SPACE) {
  setLedValue(false);
  return true;
 }

 return super.onKeyUp(keyCode, event);
}

private void setLedValue(boolean value) {
 try {
  mLedGpio.setValue(value);
 } catch (IOException e) {
  Log.e(TAG, "Erreur changement état de la LED", e);
 }
}

@Override
protected void onDestroy(){
 super.onDestroy();

 if (mButtonInputDriver != null) {
  mButtonInputDriver.unregister();
  try {
   mButtonInputDriver.close();
  } catch (IOException e) {
   Log.e(TAG, "Erreur libération du driver du Bouton", e);
  } finally{
   mButtonInputDriver = null;
  }
 }

 if (mLedGpio != null) {
  try {
   mLedGpio.close();
  } catch (IOException e) {
   Log.e(TAG, "Erreur fermeture du GPIO de la LED", e);
  } finally{
   mLedGpio = null;
  }
  mLedGpio = null;
 }
}
```

Déploiement

Sous Android Studio, le déploiement de l'application se fait en cliquant sur le bouton Run de manière assez classique. Pour les adeptes de la ligne de commande, il est possible d'effectuer le déploiement et le lancement de l'application comme suit :

```
./gradlew installDebug
adb shell am start com.ssaurel.androidthings.button/.ButtonActivity
```

Si la LED et le bouton sont correctement connectés à la carte, la LED va s'allumer lorsque l'on pressera le bouton et s'éteindra lorsqu'on le relâchera. Vous avez ainsi développé votre première application Android Things avec succès. Pour aller plus loin, vous pourriez développer une petite station météo ou une sonnette. Bien entendu, il faudra alors acheter de nouveaux périphériques compatibles avec votre carte pour avoir un affichage graphique pour la température notamment.

CONCLUSION

Encore récente, la plateforme Android Things se révèle plus que prometteuse et surtout simple d'accès pour les développeurs. En effet, ces derniers y retrouvent tous les outils de la stack Android qu'ils ont l'habitude d'utiliser quotidiennement pour le développement d'applications mobiles. Compte tenu du succès important de la plateforme Android, il est fort à parier que bon nombre d'entre eux se laissent tenter à tester Android Things et à proposer des applications intéressantes et innovantes dans le domaine de l'IoT. En ajoutant à cela la puissance de Google, notamment pour encourager des fabricants comme Intel ou Raspberry à proposer des cartes supportant Android Things, la plateforme devrait connaître un bel essor dans les mois et années qui viennent, et, pourquoi pas, prendre une place de choix dans le monde de l'Internet des Objets à l'instar de ce qu'Android connaît dans le monde du mobile. •

Mon premier projet sur **Symfony 2**

• Alexis Treton
Etudiant WebDesign
3ème année **ICAN**

Si vous êtes actuellement en train de lire cet article, c'est que vous avez enfin décidé de vous lancer dans l'aventure de Symfony 2, un des principaux frameworks de développement PHP qui vous permettra de réaliser des sites puissants et efficaces mais surtout très rapidement. Un rêve pour chaque développeur qui se respecte, donc. Cependant, même si ce tutoriel s'adresse aux débutants, car il consiste à mettre en place Symfony 2 et à créer son premier projet, il est nécessaire d'avoir d'ores et déjà de solides notions en PHP ainsi qu'en POO. Si ce n'est pas le cas, je vous invite à aller chercher des explications car sans ça, impossible de s'en sortir avec Symfony.

« Frame-quoi ? »

Framework ! Il s'agit d'un ensemble de composants utilisables comme des bibliothèques, qui vont permettre la réalisation de votre projet en gagnant du temps.

En clair, ici, on oublie la redondance. Plus besoin de recréer quelque chose de déjà existant. Si ça a déjà été fait par quelqu'un d'autre, ou par vous, c'est réutilisable facilement grâce à l'intégration du framework, et de ses bibliothèques ou de ses bundles (nous y reviendrons plus tard).

Cependant, en gagnant du temps de cette façon, on laisse totalement de côté l'apprentissage, c'est pour ça que je vous expliquais plus haut qu'il fallait avoir déjà des bases solides ainsi que des notions avancées en PHP et en POO. En utilisant Symfony 2, on quitte donc le monde de l'apprentissage pour passer dans celui de la production pure et dure.

Le seul inconvénient ici est le temps que vous prendra Symfony 2 pour être maîtrisé. En effet, ce framework est terriblement complet mais sa maîtrise est longue et fastidieuse. Cependant, ne désespérez pas, accrochez-vous, et une fois que vous maitriserez la bête, vous serez impressionné par vos propres capacités.

Un petit bout d'histoire

Loin d'être le seul framework PHP existant, il n'en demeure pas moins l'un des plus connus et l'un des plus utilisés. On le retrouve sur des plateformes de géants telles que Dailymotion pour la vidéo mais aussi Joomla ainsi que Magento pour l'e-commerce.

Développé par l'entreprise française SensioLabs, cette version 2 a vu le jour en 2015. Ce n'est pas la dernière actuellement, il existe des versions plus récentes comme la 3 qui ne sont pas encore totalement stables, et où les bundles n'ont pas été mis à jour.

MISE EN PLACE DU SERVEUR SOUS WINDOWS

Installation de Wamp

Si vous suivez ce tutoriel, comme expliqué au tout début, vous devez déjà avoir acquis certaines notions basiques dont fait partie l'installation de WAMP ou n'importe quel logiciel émulant un serveur Web sur votre machine. Si ce n'est pas le cas, je vous invite à reprendre les bases en cherchant différents tutoriels sur votre moteur de rechercher favori.

Paramétrage de PHP dans le Path

Afin de faciliter l'accès à php.exe nous allons l'inclure dans le « path ». En clair, il s'agit d'un alias qui vous évitera d'avoir à pointer sans cesse vers votre fichier php.exe stocké dans le dossier de votre serveur. Vous gagnerez ainsi en temps et en efficacité.

Suivez donc ce chemin pour accéder aux réglages du « path ».

- Menu démarrer
  - Panneau de configuration
    - Système et sécurité
      - Système
        - Paramètres système avancés

Rendez-vous ensuite dans les variables d'environnement système et trouvez l'entrée « path ». Vous devez ensuite saisir le répertoire dans lequel se trouve votre fichier php.exe. Si vous avez suivi à la lettre nos explications et laissé l'installation par défaut de WAMP, celui-ci devrait donc se trouver dans C:\wamp\bin\php\php*.*.*.

Les astérisques correspondent à la version PHP installée avec votre serveur WAMP.

Vous devez précéder le chemin d'accès par un « ; »

Exemple : ;C:\wamp\bin\php\php5.4.0

Attention, pour faire fonctionner Symfony 2, votre version de PHP doit au minimum être 5.4.

Afin de vérifier votre version, vous pouvez lire l'étape suivante. Si vous êtes sûr d'être suffisamment à jour, vous pouvez sauter l'étape suivante.

Connaître la version PHP installée sur son serveur

Pour cela rien de plus simple, il vous suffit d'ouvrir l'invite de commandes et tapez la commande « php –v ».

Un message apparaîtra vous indiquant la version actuelle avec la date de génération du build. Si vous avez une version inférieure, mettez à jour votre serveur WAMP avant de continuer ce tutoriel.

INSTALLER SYMFONY 2

Télécharger le framework

Avant de pouvoir vous amuser (il est encore temps de fuir !) avec Symfony 2, il va de soi qu'il faut d'abord que vous le téléchargiez. Pour cela rendez-vous à l'adresse suivante symfony.com/installer afin de récupérer le fichier PHAR nécessaire à l'installation du framework.

Afin de vous faciliter la tâche, copiez ce fichier à la racine Web de votre serveur (ici par exemple, c:\wamp\www). Cela vous évitera encore une nouvelle fois de devoir entrer un chemin d'une longueur déconcertante et vous pousser à l'abandon avant même d'avoir attaqué le plus dur.

Il n'y a pas vraiment d'installation à proprement parler pour Symfony 2. Ce fichier PHAR que vous venez de récupérer s'exécute via php.exe

et a plusieurs utilités. Ici nous nous en servirons pour générer un projet vierge pour Symfony 2.

Générer un nouveau projet

Maintenant que tout est en place, nous allons pouvoir générer notre premier projet. Pour cela, relançons une nouvelle fois l'invite de commandes. Pointez ensuite vers votre fichier php.exe grâce au raccourci créé précédemment dans le path, choisissez ensuite votre fichier symfony.phar, utilisez la commande new, indiquez le nom de votre projet suivi des lettres « lts ». Votre ligne devrait donc, si vous avez suivi à la lettre ce tutoriel, ressembler à celle-ci :

php symfony.phar new MonProjet lts

Revoyons ensemble en détails les différents points de cette ligne de commande :

Php : pointe vers votre fichier php.exe dans le path ;

Symfony.phar : pointe vers l'installer Symfony téléchargé précédemment ;

new : commande pour lui faire créer un nouveau projet ;

MonProjet : nom de votre projet ;

lts : Long Term Support.

Le paramètre « lts » permet de dire à l'installer de télécharger la dernière version « Long Term Support » de Symfony. Très utile si vous travaillez sur un projet qui durera dans le temps car c'est cette version qui continuera d'être corrigée et supportée pendant au minimum 3 ans. De quoi vous assurer une compatibilité sans soucis sur les années à venir.

Une fois la commande exécutée, laissez le temps de procéder aux différentes étapes jusqu'à atteindre la phrase finale :

OK Symfony X.Y.Z was successfully installed.

Si vous voyez cette phrase, félicitations, tous les fichiers pour votre projet ont été créés avec succès et vous êtes prêt à travailler sur votre premier projet en Symfony. Rendez-vous maintenant sur localhost/ MonProjet et vous devriez voir apparaître un joli message accompagné d'un symbole vert vous indiquant que tout fonctionne.

VOS PREMIERS PAS DANS SYMFONY 2

L'architecture

- Le dossier app: il contient toute la configuration Symfony qui concernera votre site Web, cependant, il ne contient pas les sources de vos pages.
- Le dossier src: le répertoire le plus important. Celui-ci comportera toutes les différentes pages de votre site Internet. Les différents fichiers déjà disponibles dans celui-ci sont des exemples de codes que vous pouvez lire afin de vous habituer à l'architecture du code.
- Le dossier vendor: il contient les différentes bibliothèques (rappelez-vous, nous en avons parlé au début !) qui seront liées à votre projet. Il s'agit de modules de codes déjà préparés que vous pouvez facilement intégrer à votre projet Symfony. En effet, comme expliqué au début de cet article, pourquoi réinventer la roue en recréant quelque chose de déjà existant ?
- Le dossier Web: Il contient toutes les ressources de votre site Internet, que ce soient les images, les scripts et les différents fichiers que l'on pourrait nommer comme publics. En fait, c'est ici que seront tous les fichiers accessibles par vos visiteurs.

Les environnements de travail

Là où Symfony se démarque des autres, c'est qu'il sait s'adapter en fonction de la personne à qui il s'adresse. Par exemple, un développeur travaillant sur le projet n'aura pas spécialement besoin des mêmes informations qu'un simple visiteur. C'est ce qu'on appelle les environnements de travail.

Il en existe deux, l'environnement dit de "dev" ou de "pré-production" et l'environnement de "prod" ou de "production".

Le premier s'adresse au développeur. En accédant au fichier app_dev, vous verrez donc les outils de développeur de Symfony alors que le fichier app classique ne vous offrira qu'une vue simple de votre projet sans aucune information complémentaire.

Si jamais des erreurs adviennent dans le mode de production, ne vous inquiétez pas, il existe un log qui garde toutes ces informations et qui vous sera accessible. En somme, avec Symfony, vous n'êtes jamais perdu.

CONCLUSION

Symfony est l'idéal si vous souhaitez vous investir davantage dans votre code et réaliser un véritable projet complet et fonctionnel. Grâce à lui, vous n'aurez plus besoin de commencer à zéro votre projet car différents modules existent afin que vous n'ayez jamais à réaliser des tâches fastidieuses et redondantes telles que la création d'un formulaire, ou la connexion à une base de données.

Cependant attention, la maîtrise d'un tel framework demande du temps, mais un de ses principaux avantages est qu'il est en français. Il existe un support et une communauté grandissante, vous trouverez de nombreux modules, une communauté prête à vous aider à n'importe moment.

Bon courage, et bienvenue dans l'aventure Symfony 2. •

Service Fabric : une plateforme PaaS sous stéroïdes Partie 2

• Maxime CAROUL
caroul@infinitesquare.com
• Thibaut RANISE
tranise@infinitesquare.com
INFINITE SQUARE

Derrière ce titre quelque peu marketing se dresse un constat simple, l'arrivée d'une nouvelle génération de solution Platform as a Service (PaaS) qui offre davantage de performance, de souplesse et d'évolutivité à nos applications. Cet article a pour objectif de vous présenter les avantages et inconvénients de cette offre avec, aussi souvent que possible, un élément de comparaison ou une mise en situation pour mieux comprendre le PaaS v2.

Modèle de service

Il existe plusieurs modèles de service pour créer rapidement vos applications : Service sans état ; Service avec état ; Acteur fiable ; Web API sans état ; ASP.NET Core ; Exécutable.
Dans cette partie, je vais présenter les trois premiers qui sont les principaux types. Les autres sont une variante du service sans état.

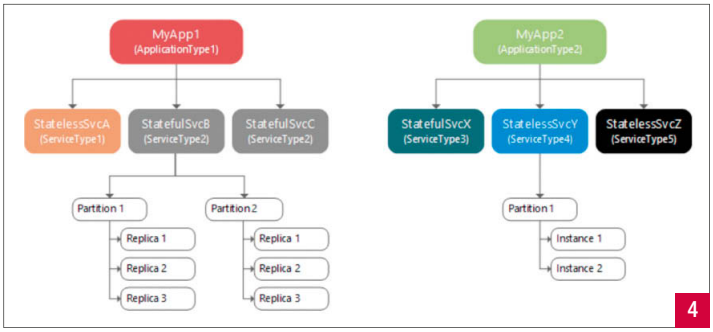
Service sans état

Un service sans état est comme son nom l'indique un type de service qui n'a pas besoin de maintenir d'état, c'est-à-dire des informations contextuelles, entre chaque appel.
Il existe deux types. Le premier au sens strict est un service qui n'a besoin que de ressources de calcul, celui-ci accepte en entrée les valeurs nécessaires à la production d'un résultat comme le redimensionnement d'une image et l'envoi d'un email. Le second, hybride, est un service qui délègue la gestion des données à un autre composant que lui-même comme une WebAPI connectée à une base de données ou à un cache distribué. Ainsi il n'est pas nécessaire d'avoir un mécanisme de basculement de nœud car il n'y a pas de risque de perte de données et la continuité de service peut être assurée avec d'autres instances saines.

Service avec état

- Ce type de service est historiquement plus complexe à gérer à cause des éléments suivants : Consommation de ressources plus importante : les données sont stockées en local ;
- Contrainte de routage du trafic : chaque instance gère son lot d'utilisateur ;
- Problématique de réplication : le service, ou du moins son état, doit être sauvegardé plusieurs fois ;
- Contrainte de performance : inhérente à la problématique de réplication ;
- Contrainte de qualité de service : lorsqu'une instance tombe, il faut être capable de basculer sur une nouvelle rapidement.

Pour résoudre ces problèmes, Service Fabric offre plusieurs solutions comme les collections fiables, la notion de réplica et le partitionnement. Les deux derniers seront abordés un peu plus loin dans cet article.
L'intérêt des collections fiables est de fournir un système de gestion de données répliqué, persistant, asynchrone et transactionnel avec une faible latence. Mais quel intérêt d'utiliser une telle solution alors qu'il en existe avec les mêmes caractéristiques comme Redis ou Azure DocumentDb permettant l'utilisation de service sans état ? La principale différence réside dans le fait que les données des collections fiables sont conservées localement au niveau de l'instance de service. Ainsi les lectures sont locales pour une faible latence et les écritures sont limitées au niveau du cluster. Concernant la réplication des états, Service Fabric propose deux types de collections fiables : les dictionnaires et les files d'attente.



4

Acteurs fiables

Le modèle acteur permet d'abstraire la complexité d'un système distribué par un ensemble d'agents autonomes qui communiquent entre eux par messages. Pour envoyer un message, il est nécessaire de connaître l'adresse du destinataire. (Wikipédia)
Pour faciliter son usage, Service Fabric fournit une API asynchrone et monothread. La gestion du cycle de vie d'un acteur est virtuelle et est entièrement gérée par le runtime, ce qui signifie qu'il n'a pas besoin d'être explicitement créé. Lorsqu'un message est envoyé à un acteur via son identifiant, la fabrique est responsable d'activer ce dernier si celui-ci n'est plus présent en mémoire (1ère création ou libéré par le ramasse miette). Si ce comportement vous dit quelque chose, cela vient du projet Orleans développé par Microsoft Research (disponible également sous GitHub). D'ailleurs pour information, celui-ci a été utilisé en production dans Azure pour les jeux Halo 4 et 5.

Partitionnement

C'est quoi ?

Une partition est une unité logique qui contient une ou plusieurs instances d'un service réparti à travers les nœuds du cluster. Dans le cas d'un service avec état, on ne parle pas d'instance mais de réplica car celui-ci possède en plus une copie de l'état du service. A un instant T, une ou plusieurs instances d'un service peuvent être actives dans un cluster. Pour un service avec état, chaque partition est responsable d'un sous-ensemble de l'état du service et possède un unique réplica primaire et potentiellement plusieurs secondaires actifs. [Fig.4].
Pour avoir un partitionnement efficace, il est conseillé de :

- Répartir les états sur l'ensemble des partitions ;
- Utiliser les mesures pour reporter la charge de chaque partition (expliqué dans la prochaine partie).

La différence entre primaire et secondaire actif réside dans le fait que c'est uniquement le réplica primaire qui supporte la charge utilisateur (lecture /écriture) et le ou les secondaires reçoivent la synchronisation des modifications apportées à l'état. Un nombre élevé de réplica améliore

donc la tolérance à la panne, dès que le nœud primaire tombe, il est possible de transférer la charge sur un autre nœud (devenant primaire à son tour) avec le même contexte de données. Attention cependant à ne pas tomber dans l'excès car trop de synchronisations peuvent avoir un impact négatif sur votre infrastructure.

« Un réplica secondaire peut également effectuer des tâches en lecture seule.. »

Service Fabric, trois modes de partitionnement :

- Par plage d'entier : défini par une valeur minimale, maximale et un nombre de partitions ;
- Partition nommée : utile dans le cas où les données sont gérées avec un discriminant du type client, pays ou année ;
- Singleton : mode par défaut d'un service sans état.

Par défaut, Service Fabric va s'assurer qu'il y a le même nombre de réplicas primaires et secondaires sur chaque nœud. Et en fonction de l'activité, des besoins en ressources, il sera sans doute amené à déplacer les réplicas d'un nœud à l'autre. Très bien, mais si les ressources disponibles au niveau du cluster ne suffisent pas, est-il possible d'ajouter un ou plusieurs nouveaux nœuds ? Oui mais attention, avant d'appliquer des changements à un cluster, il est important de prendre en considération le coût (CPU, réseau, espace disque) de déplacement d'une instance ou d'un réplica. Avant d'appliquer un changement, Service Fabric calcule le score correspondant aux différents scénarios. En cas d'égalité, il choisit le scénario avec le moins de déplacement. Donc quoi que je fasse, Service Fabric agira de la façon la plus optimale ? Encore une fois, attention ce système n'est pas parfait car le déplacement d'un service A n'est probablement pas égal au déplacement d'un service B. C'est pourquoi, il est également possible de définir de manière empirique (nulle, zéro, moyen ou élevé) le coût de déplacement associé à chaque service.

Les métriques

Une métrique est comparable aux compteurs de performance sous Windows. Elle peut être physique, tel que le pourcentage d'utilisation CPU du nœud ou logique comme un compteur métier défini dans le code du service. Lorsque vous créez un service, parfois il peut être difficile de connaître par avance les ressources qui seront sollicitées et/ou leur valeur de base. Dans le cas où aucune métrique n'est définie, Service Fabric utilise un ensemble par défaut (PrimaryCount, ReplicatCount et Count) avec les valeurs suivantes :

Mesure	Instance sans état	Instance avec état secondaire	Instance avec état primaire
PrimaryCount	0	0	1
ReplicaCount	0	1	1
Count	1	1	1

Prenons comme exemple un cluster de 6 nœuds avec un service sans état (3 instances) et un service avec état (3 partitions de 3 réplicas) avec les métriques par défaut de Service Fabric comme nous montre l'image ci-contre : [Fig.5].

On obtient une répartition qui semble homogène avec un nombre égal d'instances sur l'ensemble des nœuds. Cependant, celle-ci est purement arbitraire et ne prend pas en compte les besoins en ressources de chaque instance. Par conséquent, en laissant ces valeurs par défaut l'utilisation du cluster sera très certainement sous optimale.

Pour créer une métrique, il suffit de spécifier les éléments suivants :

- Le nom : identifiant unique ;
- La charge par défaut : entier représentant la charge de base. Dans le cas d'un service avec état il faut spécifier une valeur pour le réplica primaire et secondaire ;
- Le poids : valeur empirique (zéro, faible, moyenne ou élevée) permettant à Service Fabric de prendre des décisions plus optimales lors de la comparaison entre plusieurs métriques.

Il est possible de définir une charge par défaut pour chaque instance ou réplica. Cette valeur demeure statique jusqu'à ce que le gestionnaire de ressources reçoive une mise à jour. Cela est suffisant pour une charge de travail qui change peu. Néanmoins, dans un scénario réaliste, il est compliqué de définir une valeur fixe. C'est pourquoi, il est conseillé d'envoyer des rapports réguliers afin de mettre à jour la charge en fonction de l'activité courante. Ainsi Service Fabric est à même de prendre les meilleures décisions sur l'instant concernant la répartition des réplicas.

« Seul le réplica primaire envoie un rapport de métrique de charge. »

Même si vous envoyez les rapports de charge, il est conseillé de définir une charge par défaut. Cela permet à Service Fabric d'effectuer une répartition plus ou moins adaptée en attendant de recevoir les rapports. Dans le cas contraire, Service Fabric utilise un algorithme aléatoire ce qui n'est pas efficace.

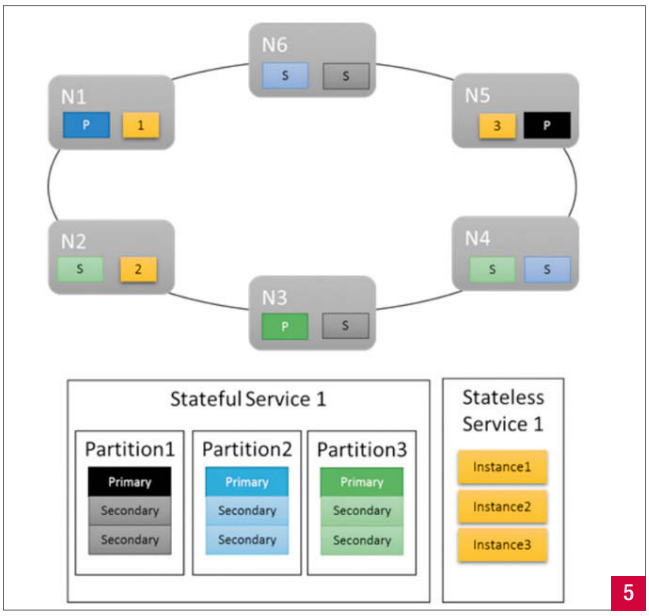
« A l'heure actuelle, la mise à l'échelle automatique ne prend pas en compte les rapports de charge. Par conséquent, seuls les compteurs de performance ont un effet sur le comportement de Service Fabric.. »

A savoir, Service Fabric prend en compte le poids de la métrique définie au niveau du réplica mais également sa moyenne à travers tous les réplicas. Pourquoi utiliser ces deux informations ? Cela permet de répartir les réplicas primaires sur l'ensemble des nœuds pour bénéficier d'une haute disponibilité.

Durant cette phase, plusieurs éléments sont à prendre en compte :

- Répartition des réplicas primaires afin d'assurer une haute disponibilité ;
- Répartition des réplicas secondaires ;
- Capacité du nœud ;
- Contraintes de placement.

Suite dans le prochain numéro



Retour aux sources du JavaScript avec Vanilla.js

• Fabien Pigere
Après avoir travaillé pour de grands comptes (Nokia, Suez, assurance, banque…) je suis devenu indépendant et spécialisé dans le front, tout particulièrement dans l'optimisation et l'algorithmie (canvas, svg, video, webgl,…) que cela soit sur portable (phonegap ou responsive design), que sur ordinateur de bureau

Depuis quelques années, les frameworks se développent dans la programmation Web, pour citer les plus connus, Angular ou React. En contresens de ces frameworks, un mouvement de résistance s'organise et grandit de jour en jour. Appelé un peu par dérision « vanilla.js », il repose sur l'existant de JavaScript dans les browsers ; là-dedans, même pas un petit jquery ! Les plus jeunes programmeurs prendront peur, mais allons-y, voyons le pourquoi, puis le comment faire.

Pourquoi Vanilla.js ?

N'utiliser que du JavaScript de base semble dur, mais ce n'est en fait pas le cas : sur les cas simples de manipulation de la DOM, les problèmes simples de comptabilité ont disparu depuis longtemps. Restent les cas assez complexes, comme les balises vidéo ou webgl, mais pour ceci, de toute façon, jquery ne fait rien, et ne le fera sûrement jamais. De plus, l'utilisation de polyfill est d'usage ; Polyfill est indépendant de Jquery. VanillaJS pourrait être comparé à l'assembleur du Web. En effet, que ce soit Angular ou Jquery, ils dépendent de Vanilla.js au final ! Rien de magique. Comprendre Vanilla.js permet donc de comprendre votre framework, voir où il est intéressant à utiliser, à identifier les goulots d'étranglement. Un bon programmeur ne se repose jamais sur la magie, il veut voir sous le capot ! Pourquoi utiliser du JavaScript natif ? Une réponse en un seul mot : vitesse ! Comparons deux cas classiques : on veut un div avec un id précis, où tous les éléments répondant à une classe, et comparons :

```
var res;
console.time("test natif");
for (var i = 0; i < 100000; i++)
   res = document.getElementById("editor");
console.timeEnd("test natif");

console.time("test jquery");
for (var i = 0; i < 100000; i++)
   res = $("#editor");
console.timeEnd("test jquery");
```

test natif: 34.9ms
test jquery: 122ms

Jquery est donc 4 fois plus lent sur ce simple test...

```
console.time("test natif 2");
for (var i = 0; i < 100000; i++)
   res = document.getElementsByClassName("test");
console.timeEnd("test natif 2");

console.time("test jquery 2");
for (var i = 0; i < 100000; i++)
   res = $(".test");
console.timeEnd("test jquery 2");
```

test natif 2: 36.2ms
test jquery 2: 300ms

...Et presque 10 fois plus lent ! N'oublions pas que nous n'avons RIEN téléchargé en plus, donc le gain est encore plus grand au démarrage de l'application, ce qui n'est pas négligeable non plus.
Convaincu ? Voyons comment utiliser le Vanilla.js à la place de Jquery.

Le natif pour l'utilisateur de Jquery

Ne nous leurrons pas la face, 90 % des requêtes Jquery sont simples, demandant un ID, une class, ou un tag, commençons donc par ces cas.

```
$(« #id») devient document.getElementById(« id »)
$(« .class ») devient document.getElementsByClassName(« class »)
$(«div ») devient  document.getElementsByClassName("div")
```

Très simple, pour un gain de temps de parfois *10 comme on l'a vu !
Qu'en est-il des requêtes complexes ? Depuis ie8, tout devient simple grâce à deux instructions :

```
document.querySelector(expr);
document.querySelectorAll(expr);
```

La 1er renvoie le 1er élément correspondant à la sélection, le 2eme quant à lui renvoie un tableau d'éléments correspondant. Au niveau de l'expression, elle est compatible à plus de 95 % avec la syntaxe de Jquery. Très peu de changement à faire donc, et les expressions non supportées sont, selon votre serviteur, exotiques ! Là aussi, faisons un test :

```
console.time("test natif 3");
for (var i = 0; i < 100000; i++)
   res = document.querySelectorAll("#test span");
console.timeEnd("test natif 3");

console.time("test jquery 3");
for (var i = 0; i < 100000; i++)
   res = $("#test span");
console.timeEnd("test jquery 3");
```

test natif 3: 51.8ms
test jquery 3: 308ms

Soit 6 fois plus rapide. Que demande de mieux le programmeur exigeant ? Bien sûr, nous voulons faire autre chose que des sélections ! Comment faire un fade sur un élément ?

```
var s = document.getElementById('thing').style;
```

```
s.opacity = 1;
(function fade(){(s.opacity-=.1)<0?s.display="none":setTimeout(fade,40)})();
```

Les plus fanatiques utiliseront un test sur « requestAnimationFrame » ! Comme on le voit, pas si compliqué.

Les class.

Voyons la manipulation de class. Cet exemple est basé sur ie10+, mais des versions ie8+ se trouvent sur le site [1] :

```
console.time("test natif 4");
 for (var i = 0; i < 100000; i++)
 {
    res = document.getElementById("test");
    res.classList.toggle("maClass");
 }
 console.timeEnd("test natif 4");

 console.time("test jquery 4");
 for (var i = 0; i < 100000; i++)
    $("#test").toggleClass("maclass");
 console.timeEnd("test jquery 4");
```

test natif 4: 129ms
test jquery 4: 1343ms

On a « juste » gagné 10 fois plus de temps, non négligeable !
Vous me direz donc, que dans la vrai vie, on modifie X éléments en une seul fois, voyons comment faire :

```
 console.time("test natif 5");
 for (var i = 0; i < 100000; i++)
 {
    res = document.querySelectorAll("#test span");
    for (var j = 0; j < res.length; j++)
       res[j].classList.toggle("maClass");
 }
 console.timeEnd("test natif 5");

 console.time("test jquery 5");
 for (var i = 0; i < 100000; i++)
    $("#test span").toggleClass("maclass");
 console.timeEnd("test jquery 5");
```

Comme on le voit, le code VanillaJS est moins concis, mais qu'en est-il de la vitesse ?

```
test natif 5: 1078ms
test jquery 5: 7913ms
```

On reste 8 fois plus rapide !
Pas convaincu ? Un dernier exemple, manipulons la CSS :

```
 console.time("test natif 6");
 for (var i = 0; i < 100000; i++)
 {
    res = document.getElementById("test");
    res.style.borderWidth = '20px';
 }
 console.timeEnd("test natif 6");

 console.time("test jquery 6");
 for (var i = 0; i < 100000; i++)
    $("#test").css('border-width', '20px');
 console.timeEnd("test jquery 6");
```

test natif 6: 160ms
test jquery 6: 1101ms

Et cette fois encore, Vanilla reste 7 fois plus rapide !
Et le reste ? Ceci dit, Jquery ne sert pas qu'à la manipulation de la DOM, voyons un appel AJAX en VanillaJS :

```
function loadbin(url, cb)
{
   var xhr = new XMLHttpRequest();
   xhr.onreadystatechange = function () {
      if (this.readyState === 4 && this.status === 200) {
         //this.response is what you're looking for
         cb(this.response);
      }
   }
   xhr.onprogress = function (evt)
   {
      var percentComplete = (evt.loaded / evt.total) * 100;
      console.log(percentComplete);
   };
   xhr.open('GET', url);
   xhr.responseType = 'blob';
   xhr.send();
}
```

Cette fonction appelle une callback en cas de succès. Bonus, le type demandé est blob (binary), ce qui n'est pas possible avec un appel Jquery, le type n'étant pas connu. A moins de bidouiller, sûrement, mais ce n'est pas de base. Autre bénéfice par rapport a Jquery : une fonction « onprogress » qui permet d'avoir, quand cela est possible, la notification pour le client du % de progression de la requête, très utile sur les BLOB. Bien sûr il y a quelques lignes de plus à taper par rapport à la version Jquery de cet appel, mais les avantages compensent à mon goût !

CONCLUSION

Vanilla.JS est la base de tous les frameworks actuels. Avec la disparition rapide des browsers non compatibles, il est devenu très utilisable de base, permettant les performances maximum, le tout sans aucun chargement, rendant le rendu de la page plus rapide, ce qui est loin d'être négligeable à une époque où la 4G n'est pas encore le standard. Il est aussi important pour avoir les bases en WEB, et permettre de savoir ce qui se trouve sous le capot, et ainsi comprendre « la magie » des grand frameworks actuels (Angular,React,Vue...). Une communauté de plus en plus grande permet d'obtenir des plug-ins ne dépendant que des bases, permettant une réactivité maximum en plus d'une indépendance complète. •

Pour plus amples informations, des sites « activistes » sur le sujet :
[1] http://youmightnotneedjquery.com/ [2] http://vanilla-js.com/
[3] https://plainjs.com/ [4] http://microjs.com/

Prenez le **contrôle d'un robot** par une page Web

Jean François Garreau,
Expert Technique
SQLI Nantes

J'ai acheté pour ma fille le robot MBot. C'est un robot basé sur un Arduino pouvant être programmé via Scratch. Le modèle que j'ai choisi est celui avec la version Bluetooth, car je savais que cela allait me laisser plus de possibilités pour le hacker plus tard. [1]

Il y a environ un an, j'ai aussi appris l'existence de deux APIs :

- L'API Web Bluetooth. Cette API permet de contrôler un appareil Bluetooth Low Energy (BLE) depuis une page Web !
- Le Physical Web.

Et si je pouvais enrichir mon Mbot pour qu'il me propose d'interagir avec lui mais sans avoir d'application à installer ?

Physical Web

Prenez un périphérique BLE. Faites-lui émettre une URL avec la norme EddyStone. Vous obtiendrez un appareil Physical Web !
Le principe du Physical Web est très simple. Il s'agit juste d'un appareil BLE qui émet une URL. Vous pouvez le comparer grossièrement à un QR Code sauf que ce dernier est Bluetooth.

Comment ça marche ?

- L'appareil doit émettre une trame EddyStone URL de façon à ce que votre téléphone puisse la capter.
- Le navigateur du téléphone (ou une application compatible Physical Web) va interroger son service pour vérifier si l'URL exposée est une URL HTTPS et non blacklistée.
- Le service interroge la page.
- Les métadonnées sont renvoyées au service.
- Le service va pouvoir répondre au téléphone pour que ce dernier affiche une notification native sur le téléphone.

Quel intérêt par rapport à un QR Code ?

En fait les intérêts sont nombreux :

- C'est aussi simple d'utilisation qu'un QR Code et ça permet de faire bien plus de choses !
- Contrairement à un QR Code, aucune application n'a besoin d'être installée pour capter la balise si ce n'est votre navigateur.
- Les sites malveillants ne seront pas exposés au public car ils auront été filtrés par le service.
- L'appareil qui émet l'URL pourra interagir avec le téléphone une fois que l'on y sera connecté.
- On peut mettre à jour l'URL de l'appareil si on le souhaite contrairement à un QR Code.
- Les notifications sont silencieuses ! En effet, ce n'est pas parce que l'on est proche d'un appareil Physical Web que notre téléphone va passer son temps à sonner. L'utilisateur ne verra la notification que si ce dernier regarde les notifications de son téléphone !

Rappel sur le fonctionnement d'un appareil Bluetooth Low Energy

Un appareil via un serveur GATT Bluetooth va exposer un ensemble de services GATT. Chaque service va lui-même exposer des caractéristiques sur lesquelles on pourra lire / écrire / s'abonner. L'appareil, les services et les caractéristiques sont identifiés par un UUID qui est unique.

1

Hack du protocole du Mbot

Afin de savoir quels services je dois appeler et quels types de données je dois transférer, je me suis lancé dans une opération de "reverse engineering" du Mbot pour comprendre comment l'utiliser. Je me suis appuyé sur cet article :Reverse Engineering a Bluetooth Low Energy Ligth Bulb qui m'a beaucoup aidé. Je vous conseille de le lire car il rentre un peu plus en détail que moi sur les étapes à suivre pour Hacker un appareil BLE. Je me contenterai ici simplement de lister les étapes principales que j'ai suivies et les résultats que j'ai obtenus.

0. Avoir un téléphone Android 4.4+

Un téléphone Android est obligatoire car nous allons analyser les trames Bluetooth émises par l'application officielle Mbot. La fonctionnalité qui permet de "sniffer" les trames Bluetooth n'est disponible que depuis Android 4.4.

1. Préparation du téléphone

Il faut installer l'application Mbot et aussi l'application nRF Connect for Mobile. Cette dernière va nous permettre de connaître les services disponibles et donc de trouver les bons UUIDs.

2. Détection des services

À l'aide nRF, je me suis connecté à mon Mbot : [2]
J'ai ensuite analysé les services Bluetooth qui étaient disponibles : [3]
On peut voir que les UUID des services sont disponibles. À ce moment-là, je ne sais pas lequel choisir. Il faut donc cliquer sur les deux services pour analyser leurs caractéristiques. [4]
En regardant les flèches sur la droite. On comprend facilement que le premier service expose une caractéristique en mode "notification" et une caractéristique en mode "écriture". J'ai donc supposé que les UUID qui m'intéressaient étaient :

Le Service avec l'UUID : 0000ffe1-0000-1000-8000-00805f9b34fb
La caractéristique avec l'UUID : 0000ffe3-0000-1000-8000-00805f9b34fb
J'ai aussi noté que le nom de mon appareil était "Makeblock_LE"

3. Écoute des trames

Il faut maintenant configurer son téléphone pour écouter les trames Bluetooth : Paramètres->Options de développement->Journal snoop HCI Bluetooth
Le fait d'activer cette option fait que le téléphone va écrire dans un fichier de log les trames Bluetooth. Sur mon Nexus le fichier est disponible sous /sdcard/btsnoop_hci.log mais sous mon Galaxy, le fichier est disponible sous /sdcard/Android/data/btsnoop_hci.log

4. Générer les fichiers de logs

Afin de comprendre et analyser au mieux les trames, j'ai procédé par étapes. J'ai ainsi généré plusieurs fichiers de logs afin d'isoler les instructions envoyées. Voici par exemple, des fichiers de logs générés :
Logs des moteurs : https://jef.binomed.fr/assets/2016-07-Mbot/btsnoop_hci_motor.log
Logs des leds RGB : https://jef.binomed.fr/assets/2016-07-Mbot/btsnoop_hci_rgb.log

5. Analyse des trames

Afin d'analyser les trames, je me suis servi de WireShark.
J'ai ensuite injecté mes fichiers dans WireShark pour faire ressortir les trames qui m'intéressaient. Contrairement à l'exemple fourni sur l'article de "reverse engineering". Les instructions envoyées ne sont pas des instructions BLE mais Bluetooth classiques. Heureusement pour moi, les instructions binaires restent les mêmes.
J'ai aussi eu la chance que le Mbot soit un projet open source. Ça m'a permis d'être sûr des instructions à regarder et comment les interpréter. J'ai pris pour exemple l'application Android : MeModule.java

6. Catalogue des instructions

Voici ce que j'ai pu comprendre. Une instruction Bluetooth du Mbot doit respecter le format suivant :

```
 /*
ff 55 len idx action device port  slot  data a
0  1  2   3   4      5      6     7     8
*/
```

Chaque message fait 13 bytes à répartir comme suit :

```
var byte0 = 0xff, // Static header
    byte1 = 0x55, // Static header
    byte2 = 0x09, // len
    byte3 = 0x00, // idx
    byte4 = 0x02, // action
    byte5 = type, // device
    byte6 = port, // port
    byte7 = slot; // slot
//dynamics values
var byte8 = 0x00, // data
    byte9 = 0x00, // data
    byte10 = 0x00, // data
    byte11 = 0x00; // data
//End of message
var byte12 = 0x0a,
```

Fonctionnement de l'API

Maintenant que nous savons comment contrôler notre robot, il nous faut nous intéresser à l'API Web Bluetooth. Avant toute chose, cette API est encore expérimentale et il faut encore l'activer dans Chrome : chrome://flags/#enable-web-bluetooth. Au moment où j'écris ces lignes, elle fonctionne sous Linux / Chrome OS / macOS / Android 5+ (Chromium) / Android 6+ (Chrome & Opera). Veuillez vous référer à cette page pour savoir ce qui est compatible : https://github.com/WebBluetoothCG/web-bluetooth/blob/gh-pages/implementation-status.md
Pour accéder à un appareil Bluetooth, il faut passer par plusieurs étapes :

- Rechercher l'appareil ;
- Se connecter ;
- Récupérer le service ;
- Récupérer la caractéristique ;
- Écrire / Lire.

Toute l'API Web Bluetooth fonctionne sur des promesses JavaScript. Chaque appel à l'API renverra une promesse.

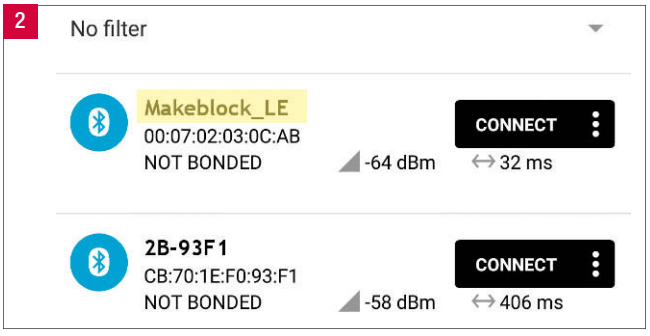
Rechercher l'appareil

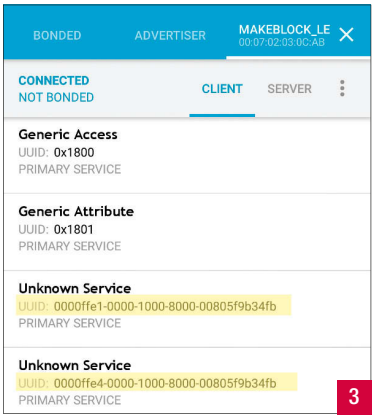
```
let options = {
    "filters": [{
        "name": "Makeblock_LE"
    }],
    "optionalServices": ["0000ffe1-0000-1000-8000-00805f9b34fb"]
};
navigator.bluetooth.requestDevice(options)
    .then(device => {
        return device;
});
```

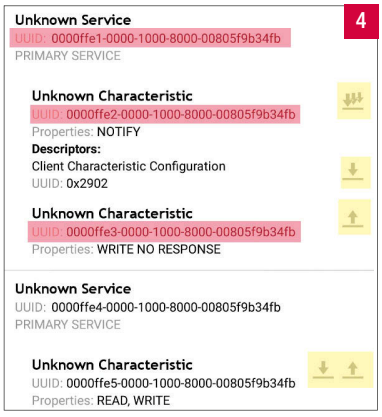
En précisant le nom du service dans le champ optionalServices, je pourrai me connecter au service. En effet, si on ne précise pas le nom du service, on ne pourra pas s'y connecter par la suite.

Connexion à l'appareil

```
device.gatt.connect()
.then(server => {
    return server;
})
```







Une fois connectés, nous récupérerons un serveur qui nous permettra de récupérer le service.

Récupération du service

Deux possibilités s'offrent à nous pour nous connecter à notre service : Depuis l'objet device ou à partir du serveur récupéré lors de la connexion.

```
// A partir du serveur
device.gatt.connect()
.then(server => {
    return server.getPrimaryService('UUID_Service');
})
.then(service => {
    return service
});

// A partir du device (on doit s'être connecté préalablement)
device.gatt.getPrimaryService('UUID_Service')
.then(service => {
    return service
})
```

/!\ Le UUID Service doit respecter le format suivant : 0000ffe1-0000-1000-8000-00805f9b34fb. En effet,0000ffe100001000800000805f9b34fb ne sera pas reconnu. Il est donc important lors des connexions de faire attention à ça !

Caractéristiques

La caractéristique doit se récupérer sur l'objet service :

```
// A partir du gatt (on doit s'être connecté préalablement)
service.getCharacteristic('UUID_Char')
.then(characteristic => {
    return characteristic
})
```

Lecture / Ecriture / Abonnement

On va pouvoir interagir de 3 façons avec une caractéristique :

```
// Lecture
characteristic.readValue()
.then(value => {
    return value
})

// Ecriture
characteristic.writeValue(bufferValue)
.then(_ => {})

// Notification start
characteristic.startNotifications().then(_ => {
    characteristic.addEventListener('characteristicvaluechanged', callback);
})
// Notification stop
characteristic.stopNotifications().then(_ => {
    characteristic.removeEventListener('characteristicvaluechanged', callback);
})
```

Helper

Tout ce code peut être généré grâce à François Beaufort qui a mis à disposition un générateur de code Web Bluetooth.

Attention aux données

Les données manipulées sont des données binaires. Il convient donc de faire les transformations nécessaires pour lire / écrire correctement. Pour les données textuelles il faut utiliser les objets TextEncoder et TextDecoder pour encoder et décoder correctement vos données !

Contrôle du Mbot

Maintenant que l'on a vu comment manipuler l'API, nous allons essayer de passer les instructions à notre robot pour l'animer.

Rappel sur le protocole du Mbot

Pour rappel, le Mbot répond aux contraintes suivantes :
Nom : **Makeblock_LE**
Service : **0000ffe1-0000-1000-8000-00805f9b34fb**
Caractéristique : **0000ffe3-0000-1000-8000-00805f9b34fb**
Format d'échange :

```
/*
ff 55 len idx action device port  slot  data a
0  1  2   3   4      5      6     7     8
*/
```

Comment donner à manger des données qui sont déjà binaires ?

Quand on regarde le format des instructions Bluetooth, on se rend compte qu'on ne manipule pas ici des chiffres ou des chaînes de caractères mais bel et bien une instruction binaire. Comment faire pour que cette donnée soit transférée correctement au Mbot ? C'est très simple, la fonction write Value d'une caractéristique attend un buffer et on va lui donner des données binaires. Prenons en exemple l'instruction suivante : ff:55:09:00:02:0a:09 :64:00:00: 00:00:0a . Il s'agit d'une instruction moteur qui fait 13 bytes. Le buffer étant **une puissance de 2**, il va falloir compléter le buffer avec des 0 et ainsi préparer un buffer de **16 bytes**. Une des subtilités de l'utilisation d'un buffer est qu'il faut inverser par paire les bytes à envoyer, sinon la donnée n'est pas reçue dans l'ordre ! Ainsi pour envoyer ff:55:09:00:02:0a:09:64:00:00:00:00:0a , je me retrouve à envoyer quelque chose comme ça : 0x55ff;0x0009;0x0a 02;0x0964;0x0000; 0x0000;0x000a;0x0000; Voici donc le code associé à l'écriture de l'instruction précédente : voir code complet sur **www.programmez.com**

Résultat

Pour conclure on peut voir que le Web prend une nouvelle direction et qu'il se positionne comme le compagnon idéal des objets connectés car ne nécessitant pas d'installation supplémentaire. Vous pouvez accéder à l'application à l'adresse suivante : https://jef.binomed.fr/mbot-webbluetooth/ (Attention à bien avoir configuré son navigateur pour autoriser le Web Bluetooth).

Crédits

Le code source du projet est disponible ici : **https://github.com/binomed/mbot-webbluetooth/blob/master/index.html**

Pour aller plus loin :

https://plus.google.com/communities/108953318610326025178 : la communauté officielle ;
https://googlechrome.github.io/samples/web-bluetooth/ : des exemples de code ;
https://www.w3.org/community/web-bluetooth/ : la communauté W3C dédié au Web Bluetooth ;
Spec : la spécification / documentation officielle ;
https://developers.google.com/web/updates/2015/07/interact-with-ble-devices-on-the-web : Article d'introduction de Google pour apprendre à utiliser l'API. •

Modules très utiles pour Python

Partie 1

Franck Ebel, *Expert R&D et Formations Serval-concept / serval-formation Responsable Licence CDAISI, UVHC Commandant de Gendarmerie réserviste, cellule Cyberdéfense*

Python est un langage riche. Les modules que nous vous proposons dans cet article s'avèreront très utiles quand vous souhaiterez vous attaquer à des bases de données ou pour gérer plusieurs clients qui vont se connecter à un serveur que vous aurez créé.
Nous allons nous intéresser dans cet article à des modules utiles pour la suite. Nous verrons donc en détail les modules re, Thread, les connexions aux bases de données, le module shelve et pickles.
Bien sur, ces modules ne sont pas écrits pour être utilisés pour du hacking ou du forensic mais nous seront bien utiles quand nous souhaiterons nous attaquer à des bases de données ou avoir par exemple plusieurs clients qui vont se connecter à un serveur que nous aurons créé. Nous aurons aussi besoin de pouvoir passer des pages et d'en extraire des données utiles pour par exemple construire un fichier de mots de passes ou récolter des mails. Commençons donc par le module re

Module re

Les expressions régulières sont très utiles lorsque nous recherchons des caractères, une expression ou autre, ou que nous voulions la remplacer.
Les expressions régulières ou regex ne sont pas spécifiques à Python, elles sont utilisées dans beaucoup de langages et, dès que nous aurons compris son fonctionnement, pourront nous aider dans beaucoup de domaines comme le forensic, la recherche simple de données, la sécurité informatique … Nous allons devoir créer ce que l'on appelle un motif, c'est à dire la chaîne de caractères que nous recherchons. Différents symboles vont nous aider dans la création de ce motif, symboles que nous allons parcourir. Nous allons juste parcourir l'essentiel, à vous de vous reporter aux ouvrages spécifiques aux regex pour de plus amples informations. Des caractères ont été définis dans les regex pour donner une signification particulière. Les adeptes de Linux ou bsd en ligne de commandes reconnaîtront certains de ces symboles utilisés avec la commande find ou grep par exemple. Nous allons donc voir dans un premier temps la signification des caractères spéciaux pour passer dans un deuxième temps à leur utilisation avec le module re.

. : n'importe quel caractère ;
^ : début d'une chaîne, sauf dans des crochets [] alors ^voudra dire qui n'est pas ;
$: fin de chaîne ;
{n} : le caractère précédent est répété n fois ;
{n,m} : le caractère précédent est répété entre n et m fois ;
** : le caractère précédent peut être répété aucune ou plusieurs fois ;*
+ : le caractère précédent peut être répété une ou plusieurs fois ;
?: le caractère précédent peut être répété zéro ou une fois ;
[] : permet d'indiquer une plage de caractères ;
Par exemple :
abc pourra nous retourner ab, abc, abcc, abccc …*
abc+ pourra nous retourner abc, abcc, abccc …
abc? pourra nous retourner ab ou abc
ab{2} nous retournera abb
ab{2,4} pourra nous retourner abb, abbb, abbbb ;
a.b pourra nous retourner acb, azb, agb … ;
[abc] nous retournera une des lettres abc ;
[a-z] nous retournera une lettre minuscule ;
[a-z]{3} nous retournera 3 lettres minuscules , par exemple abc, gfd, bht, vch … ;
[a-fA-F0-9]{8} nous retournera un nombre hexadécimal 32 bits ;
[^a-z] nous retournera tout sauf des lettres minuscules ;
D'autres caractères spéciaux nous seront aussi utiles :
\w : tout caractère alphabétique, identique à [a-zA-Z] ;
\W : tout ce qui n'est pas un caractère alphabétique ;
\d : tout caractère numérique, identique à [0-9] ;
\D : tout caractère qui n'est pas numérique ;
\s : désigne un caractère espace ;
\S : désigne un caractère non espace.

Mais comment utiliser tout cela avec le module re de Python ? Nous allons avoir plusieurs méthodes dans le module re. Match() va permettre de trouver exactement le motif défini.

```
>>> regex = re.compile('[a-z]+')
>>> result=regex.match('tempo')
>>> print result
<_sre.SRE_Match object at 0x10d7beb90>
>>> result.group()
'tempo'
>>> result=regex.match('tempo, toto, python')
>>> print result
<_sre.SRE_Match object at 0x10d7bec60>
>>> result.group()
'tempo'
>>> result=regex.match('java,1234, tempo, toto, python')
>>> result.group()
'java'
>>>
```

Nous commençons par déclarer notre expression régulière, ici dans la variable regex. Si nous sommes amené à utiliser une expression régulière plusieurs fois, la méthode re.compile() permet de compiler cette expression régulière afin d'accélérer le traitement. Cette méthode renvoie un objet « RegexObject » sur lequel on retrouve les méthodes : search(), match(), split() , sub() ... Nous testons ensuite grâce à la méthode match() si cette expression est présente dans les différentes phrases. Afin de visualiser le résultat nous utilisons la méthode group(). Nous remarquons que match() s'arrête à la première occurrence valide trouvée.
search() permet de savoir si une occurrence se retrouve au moins une fois dans une chaîne. findall() permet de trouver toutes les occurrences valides.

```
>>> regex="[a-z]{3}"
>>> reg=re.findall(regex,"Le livre Python pour developper ses propres outils de
 hacking et forensic")
>>> print(reg)
['liv', 'yth', 'pou', 'dev', 'elo', 'ppe', 'ses', 'pro', 'pre', 'out', 'ils', 'hac', 'kin', 'for', 'ens']
>>>
sub() permet de remplacer une expression par une autre.
>>> re.sub(r"(ab)",r" \1 ", "abcdef")
' ab cdef'
>>> re.sub("ruby","python","hacking et forensic en ruby")
'hacking et forensic en python'
>>>
```

Nous voyons que nous pouvons remplacer un mot par un autre, dans notre cas Ruby par Python.
Dans le premier exemple, nous avons défini un groupe (ab) que nous voulons remplacer par ' \1 '. C'est à dire que si l'occurrence ab est trouvée dans la phrase, elle sera remplacée par cette même occurrence mais avec un espace devant et derrière. \1 rappelle le groupe (ab) que nous avons créé. En plaçant un r avant le délimiteur qui ouvre notre chaîne, tous les caractères anti-slash qu'elle contient sont échappés.
Split() permet de séparer une expression en fonction d'une regex.

```
>>> var=re.split(r"\s+","hacking et forensic en Python")
>>> print var
['hacking', 'et', 'forensic', 'en', 'Python']
>>>
```

Nous passons en premier paramètre de split() le caractère espace 1 fois ou plus. Le deuxième paramètre est la phrase qui sera donc séparée par rapport à ces espaces. Nous retrouvons donc une liste avec chaque mot de la phrase.
Nous allons essayer de rechercher une adresse mac dans une chaîne de caractère. Nous recherchons un motif de la forme 11:22:33:44:55:66
Nous avons donc six fois deux chiffres hexadécimaux séparés par des : . ([a-fA-F0-9]{2}:?){6} : nous retrouvons donc 6 fois les symboles de a à f ou de A à F ou de 0 à 9 finissant ou pas par : .
Ce que nous recherchons est dans une liste (ou pas).
Nous parcourons donc cette liste éléments par éléments grâce au for... in... Chaque élément est ensuite testé avec la regex et si la regex existe, nous l'affichons.

```
>>> L=['Programmez','57/64/87:ef:87:a3','Python','87:98:09:ab:34:76']
>>> regex='([a-fA-F0-9]{2}:?){6}'
>>> for s in L:
...     a=re.compile(regex).search(s)
...     if a:
...             print s[a.start(): a.end()]
...
87:98:09:ab:34:76
>>>
```

Les méthodes acceptent des drapeaux pour déterminer comment doit se comporter la recherche, le découpage et le remplacement.
re.I ou re.IGNORECASE permet de ne pas tenir compte des majuscules et minuscules.
re.M ou re.MULTILINE permet de déterminer que le caractère ^ symbolise le début de la ligne et également le début de chaque ligne de la chaîne multiligne.
re.L ou re.LOCALE modifie le comportement des symboles \w \W \b \B \s en accord avec la locale.
re.U ou re.UNICODE permet que les symboles \w \W \b \B \d \D \s \S s'accordent avec les propriétés du caractère unicode.
re.X ou re.VERBOSE permet d'écrire des motifs d'expressions régulières plus lisibles, les espaces sont ignorés et nous pouvons mettre des commentaires en fin de ligne.
re.S ou re.DOTALL permet de définir que le caractère « . » signifie tous les caractères y compris les retours chariot.
re.DEBUG affiche des informations de déboguage sur l'expression compilée.

Module pickle et shelve

Le module pickle permet de sérialiser des variables, les types de ces variables sont conservés. La sérialisation (marshalling) permet d'enregistrer des variables dans un fichier, en vue de les recharger plus tard et/ou de les transmettre via le réseau.
La méthode dump() permet de sérialiser une variable.
La méthode load() permet de désérialiser une variable.
Les variables doivent être lues dans l'ordre ou elles ont été écrites dans le fichier et nous perdons le nom des variables.

```
import pickle

class Demo:
    def __init__(self):
        self.a = 6
        self.l = ('hello','world')
        print self.a,self.l


if __name__ == "__main__":
    f=Demo()
    pickle.dump(f, file('Demo.pickle','w'))

>>> f3=pickle.load(file("Demo.pickle"))
>>> f3.a
6
>>> f3.l
('hello', 'world')
>>>
```

Le module shelve permet de sérialiser des variables, les types de ces variables sont conservés et contrairement à pickle, l'accès à ces données peut être fait dans un ordre arbitraire. La méthode open() retourne un dictionnaire shelve. Toute insertion de variables dans ce dictionnaire est sérialisée. La clé stocke le nom de la variable, la valeur associée la valeur de la variable. Nous disposons des méthodes classiques d'un dictionnaire keys() et items() pour récupérer ces variables.
Le fichier créé par shelve est au format BDB (Berkeley DB).

```
import shelve

mabase=shelve.open('fichier','c')
mabase['cle1']=[2,1,9,7,[8,5,4,8],5,0]
```

```
mabase['cle2']=266
mabase['cle3']={2:456,5:789}
mabase['cle4']=(4,5,6,7,8,9)
mabase['cle5']="les chaînes sont permises"
mabase.close()

mabase=shelve.open('fichier','r')
for i in mabase:
  print mabase[i]
mabase.close()
```

Modules Bases de Données

Plusieurs types de bases de données existent, la plus populaire étant MySQL. Il existe plusieurs outils sous Python permettant de gérer ces différentes bases de données, nous allons en découvrir quelques-uns.

MySQLdb

Nous allons avoir besoin d'un serveur MySQL dans laquelle nous allons ajouter ces quelques données :

```
mysql -p
CREATE DATABASE python_db;
USE python_db;
CREATE TABLE IF NOT EXISTS USER(id int auto_increment PRIMARY KEY, pseudo text);
INSERT INTO USER (pseudo) VALUES ('Franck');
INSERT INTO USER (pseudo) VALUES ('Jerome');
INSERT INTO USER (pseudo) VALUES ('Sebastien');
INSERT INTO USER (pseudo) VALUES ('Guillaume');
```

Nous possédons donc maintenant une base de données et une table, la base étant destinée à contenir une ou plusieurs tables. Notre table contient deux colonnes (ATTRIBUTS). La première est un entier qui s'incrémente automatiquement à chaque insertion, la seconde contient le pseudonyme de l'utilisateur. Nous insérons quatre valeurs (les ids vont s'incrémenter de 1 à 4). Chaque ligne de cette table est appelée un tuple. Les principales actions que l'on peut faire sur une base de données sont les suivantes :
Se connecter ;
Lister les éléments d'une table (clause SELECT) ;
Ajouter un élément dans une table (clause INSERT) ;
Mettre à jour un tuple (clause UPDATE) ;
Mettre à jour une structure (clause ALTER) ;
Détruire des données ou des tables (clause DROP, DELETE) ;

```
import MySQLdb
lien_db=MySQLdb.connect(host=  "localhost", user="root", passwd="", db="test")
```

Nous importons le module puis nous appelons l'une de ses fonctions avec des paramètres labellisés (ce qui permet de les passer sans ordre précis). Nous nous connectons donc à une base de données située sur la machine localhost qui correspond à ma machine, nous devrons bien sûr adapter suivant l'endroit où se trouve la base de données et inscrire à la place l'adresse IP du serveur. Nous nous connectons donc en tant que root avec un mot de passe vide et à la base de données TEST. Vous adapterez ici aussi le login et le password en fonction de votre configuration ou des informations données par l'administrateur de la base de données. Par défaut, MySQL ne met pas de mot de passe root et laisse une base vide appelée « test ». Pensez donc à mettre un mot de passe et à supprimer cette base. Nous voilà connecté à notre serveur MySQL.
Nous allons maintenant lister la table USER.

```
import MySQLdb
lien_db=MySQLdb.connect(host=  "localhost", user="root", passwd="python",
 db="python_db")
lien_db.query("SELECT * FROM USER")
resultat= lien_db.store_result()
nb_tuple=resultat.num_rows()
while nb_tuple>0  :
   ligne=resultat.fetch_row()
   print(ligne)
   nb_tuple=nb_tuple – 1
```

Ce qui nous donne :

```
franck:~ franckebel$ ./scrip1.py
((1L, 'Franck'),)
((2L, 'Jerome'),)
((3L, 'Sebastien'),)
((4L, 'Guillaume'),)
franck:~ franckebel$
```

Nous nous connectons en créant un lien avec la base de donnée « lien_db » qui est en fait un objet dont nous allons nous servir des méthodes pour la manipuler.
La méthode query() acceptant une chaîne de caractères en paramètre va simplement envoyer la requête (query en anglais) au serveur sans se soucier du retour d'une quelconque réponse (sauf les erreurs).
La requête va sélectionner tous les tuples de la table USER et va garder ces données jusqu'à ce que nous le lui demandions grâce à l'appel à store_result() qui renvoie un objet de type « ressource MySQL ».
Nos données se trouvent alors dans la variable résultat.
Une petite astuce quand vous fonctionnez par tâtonnement ; faire un « print(variable)» affichera le type de la variable si celle-ci n'est pas une variable que nous pouvons afficher (c'est le cas dans l'exemple de « resultat »). Des méthodes de l'objet « resultat » vont nous permettre de ressortir les données. La méthode fetch_row() nous renvoie un tuple dans lequel se trouve d'autres tuples : le premier contient vos données, le second est un tuple vide. Sans paramètres, fetch_row() nous renvoie donc une seule ligne. Pour récupérer l'ensemble du résultat de notre requête, nous faisons une simple boucle avec comme intervalle le résultat d'une autre méthode de « resultat » nommée num_rows() (méthode qui nous dit combien de lignes contiennent le résultat de la requête).
fetch_row() peut préformater sa sortie, si jusqu'ici nous avons reçu un tuple de tuples, nous pouvons lui demander de récupérer non plus une mais toutes les lignes à la fois et nous les envoyer sous un format plus arrangeant comme un tuple de dictionnaire.

```
import MySQLdb
lien_db=MySQLdb.connect(host="localhost", user="root", passwd="eni",
db="python_db")
lien_db.query("SELECT * FROM USER")
resultat= lien_db.store_result()
nb_tuple=resultat.num_rows()
ligne=resultat.fetch_row(maxrows=nb_tuple,how=1)
print(ligne)
```

Ce qui donne :

```
franck:~ franckebel$ python3 script2.py
({'pseudo': 'Franck', 'id': 1L}, {'pseudo': 'Jerome', 'id': 2L}, {'pseudo': 'Sebastien,
'id': 3L}, {'pseudo': 'Guillaume, 'id': 4L})
```

Nous avons un tuple, dont nous connaissons le nombre d'éléments (nb_tuples) et qui contient des dictionnaires. Nous pouvons donc, en tant que propriétaire de la base de données, récupérer leurs données grâce aux entêtes de colonne (attributs).

```
print(ligne[0]["pseudo"])
```

Nous pouvons donc récupérer le pseudo du premier utilisateur à l'issue de la requête ci-dessus.
Il existe une autre méthode que store_result(), en effet, quand store_result() rapatrie depuis le serveur tous les tuples résultant de notre requête, use_result() ne les rapatrie qu'un par un. Dans le cas de requêtes qui renverront beaucoup de tuples, store_result() mettra un certain temps à tout récupérer mais permettra des accès plus rapides par la suite.
use_result() permettra un rapatriement rapide mais un accès ralenti par le devoir de contacter le serveur à chaque accès de tuple.
Pour l'insertion dans une base de données, qui correspond à une simple requête, nous ne voulons généralement pas savoir si elle s'est bien passée. Un simple appel à query() devrait nous suffire.

```
import MySQLdb
lien_db=MySQLdb.connect(host="localhost", user="root", passwd="eni",
 db="python_db")
requete="INSERT INTO USER(pseudo) VALUE('Robert')"
lien_db.query(requete)
lien_db.commit()
requete="SELECT * FROM USER"
lien_db.query(requete)
resultat= lien_db.store_result()
nb_tuple=resultat.num_rows()
for l in resultat.fetch_row(maxrows=nb_tuple,how=1):
      print(l["id"],l["pseudo"])
lien_db.close()
```

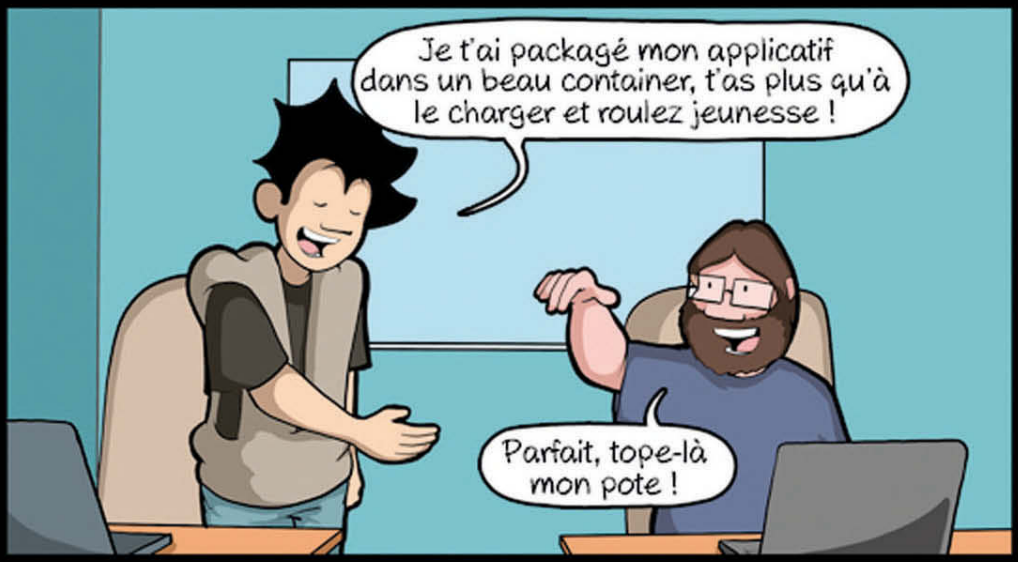
Ce qui donne :

```
franck:~ franckebel$ python3 script3.py
1 Franck
2 Jerome
3 Sebastien
4 Guillaume
5 Robert
```

Nous remarquons l'appel à la fonction commit(), cet appel est très fortement recommandé. Il met fin à ce que l'on appelle une transaction.
L'insertion par MySQL est gérée comme telle et si nous ne faisons pas cet appel, notre insertion risque de ne pas être sauvegardée ou de ne pas être prise en compte. Notre insertion deviendrait un tuple fantôme qui ne serait pas assimilée à 100% par le serveur qui ne le répercuterai pas dans les requêtes d'éventuels autres clients.
Il est recommandé également de procéder à un commit(), même après une requête SELECT , ne serait-ce que pour mettre à jour les statistiques et informations d'accès internes au serveur MySQL.
Une petite astuce consiste à lancer lien_db.auto-commit(true) qui demande à Python de lancer automatiquement la méthode commit(). Mais ceci peut engendrer des latences, notamment si nous exécutons plusieurs requêtes d'affilée, nous pourrions nous satisfaire d'un simple commit de fin de toutes les requêtes , alors que autocommit() fera la demande pour chaque requête.
La requête en elle-même est assez simple. Nous insérons un nouveau pseudo et l'id, comme convenu, va s'incrémenter automatiquement. Nous récupérons ensuite la liste des ids et des pseudos. Enfin, et chose que nous n'avons pas encore faite jusque-là, nous avons clos le lien entre notre script et MySQL. Dans le cas d'un script qui va durer après l'accès à la base de donnée, il faut délier le lien entre le script et MySQL. Nous devons avoir en tête que le serveur MySQL limite le nombre de connexions entrantes et , si nous ne fermons pas là le lien dès que possible, nous monopolisons une place pour rien.
Pour les mises à jour de tuples, de tables, les ajouts d'utilisateurs ou toute autre opération sur nos bases de données, il n'existe aucune différence au niveau du code Python. Il nous suffit de remplacer simplement notre requête INSERT par UPDATE, ALTER, DROP ou GRANT par exemple.
Les principales choses à connaître pour faire discuter Python et MySQL ensemble sont donc de savoir se connecter, envoyer une requête, récupérer les éventuels résultats et se déconnecter. •

La suite dans le 206

Comment le premier déploiement devait se passer



Comment le premier déploiement se passe

CommitStrip.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 & Rédacteur en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Ont collaboré à ce numéro : S. Saurel
Nos experts techniques : S. Potier, R. Pinon, T. Desmas, G. Leborgne, T. Ouvre, E. Vernié, A. Treton, F. Pigere, J-F Garneau, F. Ebel, M. Carol, T. Ranise, S. Gombaud, R. François, J. Fenollar, C. Chervy, C. Villeneuve, S. Abélard, B. Lanneau, R. Willmann, M. Costabello, A. Jacob, M. Fery, J. Paquet, T. Gayet, T. Chantier
Couverture : © stevanovicigor - **Maquette :** Pierre Sandré.
Publicité : PC Presse, Tél.: 01 74 70 16 30, Fax : 01 40 90 70 81 - 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 - © NEFER-IT / Programmez, avril 2017

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 : 35 € (monde entier) souscription sur www.programmez.com

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

[2017]

DÉVELOPPEZ 10 FOIS PLUS VITE

WINDEV®

WINDEV 22 : ATELIER DE DÉVELOPPEMENT PROFESSIONNEL, COMPLET EN STANDARD

Gestion du cycle de vie complet: Idée, Conception, Développement, Génération, Déploiement, Exploitation • Un code multi-plateformes Windows, Linux, Java, Internet, Mobiles • Environnement ALM complet • Toutes les bases de données sont supportées, Big Data • Inclus: HFSQL, base de données locale, Client/Serveur, cluster, embarquée et cloud • Puissant RAD • Intégration continue • Générateur de fenêtres (UI) visuel & intuitif • Utilisation • Audit statique & dynamique • Héritage et surcharge d'interface • Tous les champs facile de charte graphique • Tableau de bord de vos applications • Utilisation (contrôles) sont très puissants et livrés en standard: Champ de saisie, Tableau croisé dynamique (cube), Champ Planning, Champ Diagramme de Gantt, Champ Tableau de bord, Champ Table, Champ Graphe, etc. • FAA: chaque application bénéficie automatiquement de Fonctionnalités Automatiques: export vers Excel, vers Word, envoi d'email, etc • Sécurité : Mot de passe de vos applications • Puissant générateur de rapports et codes-barres • Langage de 5ème génération: WLangage • Editeur de code intuitif avec puissant débogueur • Tests unitaires et tests automatiques • Versioning (GDS/SCM) • Webservices SOAP et Rest • Modélisation Merise et UML • .NET, 3-Tier, MVP • Support de tous les standards: XML, USB, Bluetooth, NFC, J2EE, OLE, ActiveX, RPC, SaaS, SMTP, FTP, OPC, DLNA, IoT, Sockets, API, Webservices... • Lien avec Lotus Notes, SAP, Google, Outlook. • Multimédia, Domotique • Livré avec des centaines d'exemples et d'assistants • Génère le Dossier technique d'un clic • Télémétrie pour connaître l'utilisation réelle de vos applications • Support de 64 langues étrangères par applications • Générateur d'aide • Gestion des suggestions et incidents utilisateurs • Support Technique Personnalisé Gratuit* • ...

CONSULTEZ PLUS DE 100 TÉMOIGNAGES DE PROFESSIONNELS SUR LE SITE PCSOFT.FR

Elu
«Langage le plus productif du marché»

VERSION EXPRESS GRATUITE
Téléchargez-la !

VU À LA TÉLÉ EN 2017
SUR **TF1**, SUR **BFMTV** ET SUR **M6**

Logiciel professionnel. * 15 requêtes gratuites sur la version en cours de commercialisation

Tél Paris: 01 48 01 48 88 *Tél Montpellier: 04 67 032 032*

Plus de 100 témoignages
Dossier complet gratuit

WWW.PCSOFT.FR