programmez!
#194 - Mars 2016
le magazine du développeur

Choisir son environnement de développement mobile multiplateforme

Les 2 commandements du développeur :
1 / faire un code propre et clair
2 / commenter son code

Les architectures lambda

Monter et configurer son imprimante 3D

Maîtrisez les applications Windows universelles

Java
Utiliser OrmLite

Créer un ruban Windows en C++

© eternalcreative

M 04319 - 194 - F: 5,95 € - RD

Printed in EU - Imprimé en UE - BELGIQUE 6,45 € SUISSE 12 FS - LUXEMBOURG 6,45 € - DOM Surf 6,90 € Canada 8,95 $ CAN - TOM 940 XPF - MAROC 50 DH

LE CLOUD GAULOIS, UNE RÉALITÉ ! VENEZ TESTER SA PUISSANCE

EXPRESS HOSTING

Cloud Public
Serveur Virtuel
Serveur Dédié
Nom de domaine
Hébergement Web

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

ENTERPRISE SERVICES

Cloud Privé
Infogérance
PRA/PCA
Haute disponibilité
Datacenter

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

EX10

Cloud Hybride
Exchange
Lync
Sharepoint
Plateforme Collaborative

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

If ou while(true)

Le développeur a-t-il réellement le libre choix des langages, des outils et des technologies ? La réponse est clairement non.

En entreprise, en SSII, chez un éditeur logiciel, notre dév aura toujours un cadre technique et technologique. Je ne dis pas figé et rigide, car il aura, parfois, une marge de manœuvre pour conseiller et proposer, mais les fondamentaux seront déjà choisis pour lui.

Finalement, on pourrait risquer une analogie avec la langue française. Elle possède environ 100 000 mots, mais au quotidien, notre vocabulaire se limite à 100, 150 mots. En développement, c'est un peu pareil. Au quotidien, le dév se limitera à une dizaine d'outils et de langages...

Cet environnement limité ne doit cependant pas vous empêcher de regarder ce qu'il se passe ailleurs pour apprendre de nouvelles choses et les proposer pour un futur projet. Une des erreurs à ne pas commettre est d'enfermer son environnement et refuser toute évolution majeure.

Parfois les ruptures sont très dures, regardez la rupture radicale entre Angular 1 et Angular 2. Mais le projet était finalement dans une impasse et cette radicalité s'imposait d'elle-même. PHP pour suivre les évolutions et se dépoussiérer a dû faire des choix, quitte à ne pas être totalement rétro-compatible, tout en faisant l'impasse sur la v6 (et faire oublier l'échec du projet). Les mêmes problèmes se posent à Java. Les refontes profondes sont découpées, repoussées, parfois moins ambitieuses. Apple n'a pas fait autrement en décidant de bâtir un nouveau langage, Swift, pour déprécier Objective-C qui aurait demandé beaucoup trop d'efforts pour le faire évoluer et qui était critiqué (souvent à juste raison) pour son apprentissage.

L'art du compromis est une des bases de l'informatique. Pour tenir les délais, sortir une version, il faut arrêter les développements et couper dans le vif pour décaler des fonctions, un module. J'avoue que je me dis en voyant un nouveau matériel ou logiciel : pourquoi n'ont-ils pas mis cette fonction, changé ce bloc, adapté ceci... On fonctionne toujours par itération. Oui c'est un choix marketing, mais il y a aussi un choix technique.

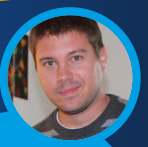
Sinon, vous serez dans une boucle infinie while (true)... et là, vous vous dites, euh, où est ma condition de sortie ?

François Tonic /
dresseur de codes
ftonic@programmez.com

sommaire







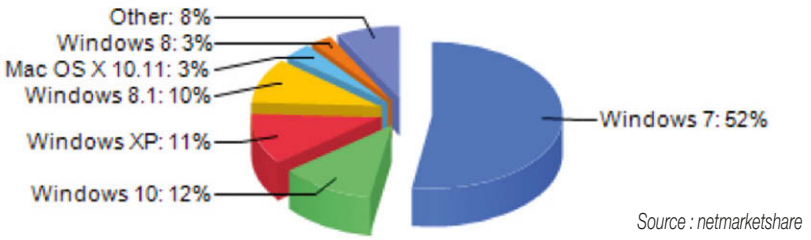



À lire dans le prochain numéro n°195 en kiosque le 31 mars 2016

PHP 7 par la pratique
Migration, optimisation et performances, gestion des erreurs, focus sur le typage et le moteur interne.

Bienvenue dans l'APIconomie
Les API sont partout. Elles envahissent tout. Qu'est-ce qu'une API ? Comment les créer ? Quel modèle économique ? Comment les utiliser et les consommer ?

WINDOWS 10 DEVANT TOUT LE MONDE, MAIS DERRIÈRE WINDOWS 7



Source : netmarketshare

Mini-sondage Programmez ! :

Que vous inspire le nouveau langage d'Apple, SWIFT ?

44 %, bah je ne suis pas un fanboy
21 %, un vrai développeur utilise uniquement l'assembleur
20 %, cool, je le teste ou je vais le tester
15 %, cool, c'est open source

Total des votes : 231

Et si le plénoptique devenait réellement utilisable ?

Souvenez-vous de la société Lytro et de sa promesse de prendre des photos et de choisir ensuite la zone que vous souhaitez voir pour supprimer le flou des photos. La technologie permet de faire le focus après sa prise. Malheureusement, elle imposait des applications dédiées et surtout, les appareils photo étaient vendus assez cher. Des rumeurs voudraient qu'Apple prépare un système photo à double capteur pouvant éventuellement permettre cette manipulation...

L'avenir des datacenter est-il à 10 000 lieux sous la mer ?

Avec le projet Natik, Microsoft tente de le prouver

Google

travaille toujours sur Android Studio 2.0. Le 29 janvier dernier, une 9e preview a été dévoilée !

James Cameron

va-t-il écrire et réaliser le prochain Terminator ?

Pyjion,

un compilateur JIT Python, signé Microsoft

https://twitter.com/stroughtonsmith

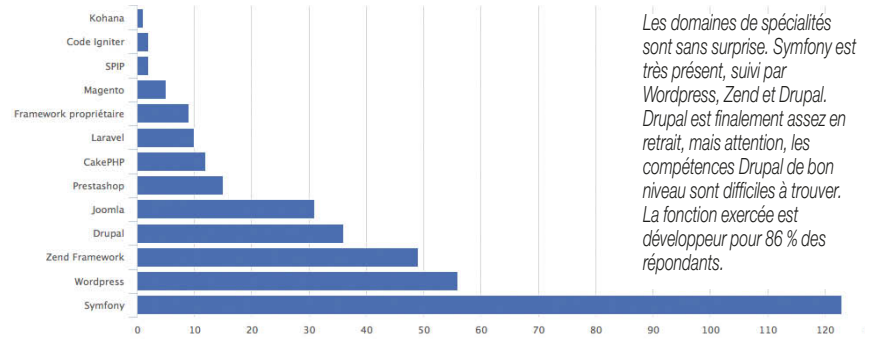
est le compte twitter que tout développeur iOS doit suivre. Ce dév indépendant scrute le moindre recoin du système et découvre des choses étonnantes...

LE DÉV PHP GAGNE-T-IL BEAUCOUP ?

L'AFUP et Human Coders ont dévoilé leur étude 2015. Les salaires sont finalement dans les grandes moyennes que l'on constate depuis plusieurs années.

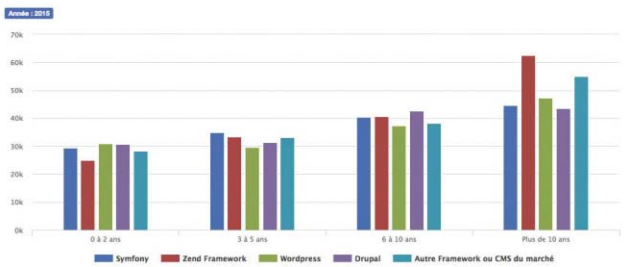
Expérience	Paris	Régions
0 à 2 ans	34 600 €	28 600 €
3 à 5 ans	39 800 €	31 500 €
6 à 10 ans	45 400 €	35 800 €
Plus de 10 ans	56 700 €	41 300 €

Le salaire des « débutants » est dans la moyenne haute. Il faudra parfois assez fortement pondérer ce niveau.



Les domaines de spécialités sont sans surprise. Symfony est très présent, suivi par Wordpress, Zend et Drupal. Drupal est finalement assez en retrait, mais attention, les compétences Drupal de bon niveau sont difficiles à trouver. La fonction exercée est développeur pour 86 % des répondants.

Répartition du salaire par spécialité



Le salaire par spécialité est intéressant à observer. Nous trouvons assez peu de différences sur les expériences de 0 à 10 ans. On constate tout de même un salaire très bas pour un débutant Zend, mais il pourra rapidement monter en compétence et en salaire.

L'INDEX TIOBE DU MOIS

02/16	02/15	tendance	langage	%	évolution en %
1	2	^	Java	21.145 %	+5.80 %
2	1	v	C	15.594 %	-0.89 %
3	3	=	C++	6.907 %	+0.29 %
4	5	^	C#	4.400 %	-1.34 %
5	8	^	Python	4.180 %	+1.30 %
6	7	^	PHP	2.770 %	-0.40 %
7	9	^	Visual Basic .NET	2.454 %	+0.43 %
8	12	^^	Perl	2.251 %	+0.86 %
9	6	v	JavaScript	2.201 %	-1.31 %
10	11	^	Delphi/Object Pascal	2.163 %	+0.59 %

Java reprend la tête du classement tandis que C recule d'une place. Mais C et C++ font toujours jeu égal avec Java. Python continue à être populaire. Pour le reste, quelques progressions, mais relativement minimes. Objective-C pointe à 14e place et Swift à la 16. Mais attention, l'index ne mesure pas l'usage réel des langages, mais uniquement les requêtes de recherches.

DOMAINES | MAIL | **HÉBERGEMENT** | E-COMMERCE | SERVEURS

~~4,99~~ **0,99** € HT/mois (1,19 € TTC)*
À partir de

MANAGED WORDPRESS

100 % PERFORMANCE

- **NOUVEAU !** Espace Web **illimité avec SSD**
- **NOUVEAU !** Bases de données **illimitées sur SSD**
- **NOUVEAU ! PHP 7 avec OPcache**
- Trafic **illimité**
- Comptes email **illimités**
- **2 Go de RAM** garantis

100 % SÉCURITÉ

- **NOUVEAU ! Protection contre les attaques DDoS avec NGINX** pour encore plus de performance, de fiabilité et de sécurité
- **Géo-redondance :** hébergement simultané dans des data centers distincts
- 1&1 CDN avec Railgun™
- 1&1 SiteLock

100 % CONFORT

- **NOUVEAU !** L'**Assistant WP** vous guide dans l'installation et dans le choix du design
- **Inclus : thèmes prêts à l'emploi**
- **Assistance 24/7**
- Support Expert WordPress
- 1&1 Community

1 MOIS POUR ESSAYER

1 CLIC POUR CHANGER DE PACK

1 APPEL UN EXPERT VOUS RÉPOND

0970 808 911
(appel non surtaxé)

1&1

1and1.fr

*Les packs Managed WordPress sont à partir de 0,99 € HT/mois (1,19 € TTC) pour un engagement minimum de 12 mois. À l'issue des 12 premiers mois, les prix habituels s'appliquent. Certaines fonctionnalités citées ne sont pas disponibles dans tous les packs. Offres sans durée minimum d'engagement également disponibles. Conditions détaillées sur 1and1.fr. Rubik's Cube® utilisé avec l'accord de Rubik's Brand Ltd. 1&1 Internet SARL, RCS Sarreguemines B 431 303 775.

mars

NIDays 2016 : 10 mars

National Instruments organise sa journée annuelle à Paris le 10 mars prochain. Occasion pour faire le point sur les différents outils, l'instrumentation, l'informatique industrielle et embarquée. La journée sera rythmée de nombreuses conférences.
Grand rendez-vous de la journée, deux coupes seront organisées, la Coupe NXT et la Coupe RIO, dont le thème commun est la robotique.
Pour plus de détails : http://france.ni.com/nidays/coupes-robotiques
Programmez ! sera présent. Venez nous rencontrer.
Site officiel : http://france.ni.com/nidays

avril

Devoxx 2016 : du 20 au 22 avril 2016

Réservez déjà les dates pour venir au plus grand événement Java en France. Cette année encore, la conférence promet beaucoup, avec de nombreux thèmes abordés : architecture, sécurité, Cloud, core Java, les langages de la JVM, méthodologie, mobilité, html 5...
Site : http://www.devoxx.fr/

quelques dates à retenir

Meetup Swift – RXSwift & TBD :
Le 15 mars, meetup à Paris autour du langage Swift et de RXSwift. Site : http://www.meetup.com/fr-FR/swiftparis/events/227455210/

Evolve16 :
La conférence développeur de Xamarin se déroulera en Floride du 24 au 28 avril. De nombreux thèmes seront abordés : design, compilation, intégration, sécurité, tests, monitoring... https://evolve.xamarin.com

La conférence **Ncrafts 2016** aura lieu cette année les 12 et 13 mai. Cette conférence ravira les développeurs les plus exigeants.

PHP Tour 2016 à Clermond-Ferrand
L'AFUP pose les Elephpants à Clermont-Ferrand en mai prochain. Une occasion de réunir la communauté, l'écosystème et les acteurs du monde PHP. Site : http://event.afup.org

ReSeT #28 à Coutances : la scène Amstrad CPC du 24 au 26 juin à Coutances.
Comme chaque année, la scène Amstrad CPC (& friends) se réunira à Coutances, du 24 au 26 Juin 2016. C'est l'un des plus gros événements de la démoscène française. Compétition, l'actualité récente sur CPC, Atari, ZX et surtout de la bonne humeur !
Site : http://reset.cpcscene.net/index.php?lang=fr

TOUR DE FRANCE

Comme chaque année, PC Soft fait le show avec une journée autour de son outil phare : WinDev. L'événement tournera dans 11 villes avec de nombreuses sessions, des formations dédiées.
Voici les dates :

- Montpellier 5 mars
- Toulouse 15 mars
- Bordeaux 16 mars
- Nantes 17 mars
- Bruxelles 22 mars
- Lille 23 mars
- Paris 24 mars
- Strasbourg 29 mars
- Lyon 30 mars
- Marseille 31 mars
- Genève 5 avril

Site : http://www.pcsoft.fr/pcsoft/tdftech/2016/

MAKERFAIRE PARIS 2016 : les 30 avril & 1er mai

Cette année encore, MakerFaire se tiendra durant la Foire de Paris, à la porte de Versailles. Tout Maker peut proposer son projet et être présent durant l'événement. En 2015, ce sont plus de 35 000 visiteurs qui ont parcouru les allées.
Site : http://www.makerfaireparis.com
Cette année, deux mini Maker Faire seront organisées en France :

- Saint-Malo : 9 & 10 avril
- Rouan : 3 & 4 juin

Dev/Test avec Azure

Cellenza propose le 7 avril une conférence DevTest avec Azure au campus de l'éditeur près de Paris. Cet événement s'adresse aux développeurs, testeurs, architectes, et responsables IT, travaillant sur des environnements de développement. La journée débutera à 9h30. Plusieurs sessions techniques seront proposées : automatisation d'un infrastructure dev/test, Chef, gouvernance du dev/test, etc.
inscription : https://goo.gl/u5EeRl

LES MATINALES DE ZENIKA

Matinale Angular 2 / 10 mars (Zenika Bordeaux)

Vous souhaitez commencer une nouvelle application et vous hésitez entre AngularJS et Angular 2 ? Vous désirez avoir un aperçu de la prochaine version du framework ? Ou peut-être faites-vous partie des personnes qui ont des interrogations par rapport à ce qu'elles ont déjà entendu sur l'avenir d'Angular ?
Programme et inscription : http://www.zenika.com/zenika-ng2-tour-bordeaux.html

Matinale Agile Wake Up - AWU #2 / 22 mars (Zenika Paris)

Notre équipe Agile By Zenika, vous propose d'aborder trois nouvelles thématiques qui lient organisations et Agilité : les entreprises libérées, les nouvelles méthodes de management et la mesure de la valeur.

Programme et inscription :
http://www.zenika.com/agile-wake-up-2.html

MUG Strasbourg : les prochaines réunions

Le Microsoft User Group de Strasbourg organise régulièrement pour la communauté des conférences autour des technologies Microsoft. Les prochains événements prévus sont :

- Mars : conférence sur TypeScript et Angular 2

Facebook : https://www.facebook.com/groups/MugStrasbourg/?fref=ts
Twitter : https://twitter.com/MUGStrasbourg

FAÇONNONS ENSEMBLE L'INTERNET DES OBJETS

NIDays

Rejoignez plus de 1 200 innovateurs issus de secteurs industriels variés, venez échanger avec les équipes NI et découvrez comment les avancées technologiques convergent pour créer un monde plus intelligent et plus connecté, reposant sur des systèmes conçus par logiciel.

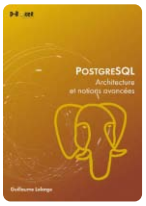
Paris, Palais des Congrès
Jeudi 10 mars 2016

Inscription gratuite sur **ni.com/nidays.**
Suivez-nous sur Twitter : **@NIFrance** et en live avec **#NIDays.**

NATIONAL INSTRUMENTS

PostgreSQL 9.5 : les nouveautés

Début mai 2014, les développeurs du moteur de bases de données libre PostgreSQL débutent le travail sur cette nouvelle version. Après 21 mois de développement acharné et de discussions à n'en plus finir, la version finale est sortie le 7 janvier 2016. Un grand nombre de nouvelles fonctionnalités a été ajouté à un moteur déjà très fonctionnel.



Guillaume LELARGE
Consultant au sein de la société d'expertise PostgreSQL Dalibo, coordinateur de la traduction en français du manuel officiel de PostgreSQL, auteur du livre PostgreSQL – Architecture et notions avancées paru aux éditions D-BookeR
http://blog.guillaume.lelarge.info

SQL

Les nouveautés au niveau SQL comblent deux besoins très forts. Le premier concerne les instructions SQL sur les grosses volumétries. Le second est une fonctionnalité très demandée depuis plusieurs années. Nous allons commencer par les instructions SQL.

UPSERT

Les utilisateurs de PostgreSQL remontent souvent l'absence d'une instruction MERGE, aussi appelée UPSERT. L'idée est de pouvoir, en une requête, insérer une nouvelle ligne et, en cas de conflit avec une ligne existante, modifier cette ligne. Le développement de cette fonctionnalité a été particulièrement long à cause de la complexité de cette fonctionnalité. De nombreuses discussions ont fait varier la proposition initiale. Au final, PostgreSQL dispose maintenant d'une instruction INSERT … ON CONFLICT... La clause ON CONFLICT permet de transformer l'instruction INSERT en un UPDATE. Il est même possible d'ignorer simplement l'insertion. Voici un exemple des deux cas :

```
CREATE TABLE villes (id integer, nom text, PRIMARY KEY (id));
INSERT INTO villes VALUES (1, 'Paris');
INSERT INTO villes VALUES (2, 'lilles');
INSERT INTO villes VALUES (2, 'Lille')
ON CONFLICT ON CONSTRAINT villes_pkey DO UPDATE SET nom=excluded.nom;
SELECT * FROM villes;
 id  | nom
 ----+-------
  1  | Paris
  2  | Lille
(2 rows)
INSERT INTO villes VALUES (2, 'Valenciennes')
ON CONFLICT ON CONSTRAINT villes_pkey DO NOTHING;
SELECT * FROM villes;
 id  | nom
 ----+-------
  1  | Paris
  2  | Lille
(2 rows)
```

Ceci devrait faciliter grandement le développement de certaines applications et surtout le portage d'une application prévue pour un autre moteur de bases de données vers PostgreSQL.

CUBE, ROLLUP et GROUPING SETS

Ces trois clauses étendent les possibilités de la clause GROUP BY, permettant le regroupement de lignes à des fins de calculs statistiques et analytiques. La clause GROUP BY ne regroupe les données que sur une seule dimension. Par exemple, un GROUP BY col1, col2, col3 regroupera les données sur toutes les colonnes et en suivant l'ordre des colonnes (à savoir col1, puis col2, puis col3). Il ne fera pas de regroupement de col2 avec col1 et col3 pour donner un exemple. Si différents regroupements étaient nécessaires, il fallait exécuter les requêtes séparément et faire en sorte que l'applicatif assemble leurs résultats ou plus simplement exécuter les différentes requêtes en assemblant leur résultat avec une ou plusieurs clauses UNION ALL. Ce dernier cas est plus simple à écrire mais la requête devient complexe et difficile à interpréter. Arrivent donc avec la version 9.5 les clauses GROUPING SETS, CUBE et ROLLUP. Avec GROUPING SETS, un regroupement sur col1 et col2 donne le regroupement pour la colonne col1, puis celui pour la colonne col2. Autrement dit, cette requête :

```
SELECT col1, col2, count(*) FROM tests GROUP BY GROUPING SETS(col1, col2);
```

est identique à

```
SELECT col1, NULL AS col2, count(*) FROM tests GROUP BY col1
UNION ALL
SELECT NULL AS col1, col2, count(*) FROM tests GROUP BY col2;
```

Ceci étant dit, la durée d'exécution n'est pas identique. Dans le premier exemple, la table n'est lue qu'une fois alors que, dans le second, elle est lue deux fois. Voici un exemple de l'exécution d'un GROUPING SETS :

```
SELECT col1, col2, count(*) FROM tests GROUP BY GROUPING SETS(col1, col2);
 col1 | col2 | count
------+------+-------
    1 |      |     3
    2 |      |     3
    5 |      |     4
      |    1 |     3
      |    3 |     2
      |    5 |     1
      |   10 |     2
      |   11 |     1
      |   13 |     1
(9 rows)
```

La clause ROLLUP va plus loin en réalisant plusieurs regroupements qui respectent l'ordre indiqué des colonnes. Avec la clause ROLLUP (col1, col2), le regroupement se fera sur la paire col1 et col2, puis sur la colonne col1 seule puis sur tout, ce qui pourrait donner ceci par exemple :

```
SELECT col1, col2, count(*) FROM tests GROUP BY ROLLUP(col1, col2);
 col1 | col2 | count
------+------+-------
    1 |    1 |     3
    1 |      |     3
    2 |    3 |     2
    2 |    5 |     1
    2 |      |     3
    5 |   10 |     2
    5 |   11 |     1
    5 |   13 |     1
    5 |      |     4
      |      |    10
(10 rows)
```

Enfin, la clause CUBE va encore plus loin en ajoutant le regroupement par la colonne col2 :

5e édition

Congrès & Expo

BIGDATA

Turning the data deluge into decisions

Rendez-vous les
7 & 8 mars 2016
Palais des Congrès de Paris

Les supers pouvoirs du Big Data !

Participez à ***l'événement leader du Big Data*** en France

- 10 000 participants
- 150 exposants
- 100 experts à la tribune
- 5 salles de conférences

Réservez dès à présent vos ***7 & 8 mars 2016*** et préparez ***l'avenir*** de votre entreprise avec ***Big Data Paris !***

Votre badge gratuit sur www.bigdataparis.com

by Cap in Corporations we Trust

```
SELECT col1, col2, count(*) FROM tests GROUP BY CUBE(col1, col2);
 col1 | col2 | count
------+------+-------
   1  |   1  |   3
   1  |      |   3
   2  |   3  |   2
   2  |   5  |   1
   2  |      |   3
   5  |  10  |   2
   5  |  11  |   1
   5  |  13  |   1
   5  |      |   4
      |      |  10
      |   1  |   3
      |   3  |   2
      |   5  |   1
      |  10  |   2
      |  11  |   1
      |  13  |   1
(16 rows)
```

Tout ceci permet des calculs analytiques poussés, sur des petites comme des grosses volumétries, le nombre de lectures de la table étant diminué.

TABLESAMPLE

Le rôle de cette clause SQL est de récupérer un échantillon d'une table, principalement dans le but de calculer rapidement le résultat qui, par voie de conséquence, ne sera qu'approximatif. Le ratio de lecture est un argument de la clause TABLESAMPLE. Évidemment, plus le pourcentage est petit, plus le calcul est rapide. Mais aussi plus le résultat est approximatif.

```
CREATE TABLE echantillon
  AS (SELECT trunc(random() * 90) AS i FROM generate_series(1,100000000));
Time: 90898.411 ms

SELECT avg(i) FROM echantillon;
44.49821317
Time: 13485.667 ms
```

Le calcul de la moyenne sur la table complète prend 13,4 secondes. Avec la clause TABLESAMPLE, il est possible de n'utiliser qu'un échantillon de la table. Voici ce que cela donne avec 10% de la table :

```
SELECT avg(i) FROM echantillon TABLESAMPLE SYSTEM(10);
44.481114321303
Time: 1977.018 ms
```

La durée d'exécution passe à un peu moins de 2 secondes, soit pratiquement sept fois plus rapide. Deux méthodes d'échantillonnage existent. Celle utilisée ci-dessus, appelée *system*, provoque la lecture d'un nombre de blocs équivalent au pourcentage indiqué et le calcul se fait sur toutes les lignes trouvées dans ces 10 % de blocs. La méthode *bernoulli* lit tous les blocs et ne récupère que 10 % des lignes pour le calcul. Cette deuxième méthode est donc plus lente que la méthode *system* vu qu'elle lit la table entière, mais reste bien plus rapide que l'exemple sans clause TABLESAMPLE, étant donné que le calcul ne se fait pas sur toutes les lignes.

```
SELECT avg(i) FROM echantillon TABLESAMPLE bernoulli(10);
44.4829093062548
Time: 4246.839 ms
```

En conclusion, la clause TABLESAMPLE permet d'obtenir un résultat plus rapidement, mais approximatif et rarement identique même si la table n'évolue pas entre deux exécutions.

SKIP LOCKED

Cette clause est une amélioration de l'instruction SELECT … FOR UPDATE.
Quand une transaction fait appel à un SELECT … FOR UPDATE, elle place un verrou sur cette ligne pour être sûre qu'aucune autre transaction ne puisse la mettre à jour en même temps. Si cette autre transaction exécute elle aussi un SELECT … FOR UPDATE, elle se verra bloquée, en attente de la fin de la transaction qui a le verrou sur cette ligne.
Parfois, plutôt que de rester bloqué, on préférerait passer à la ligne suivante. Un exemple typique est le traitement d'une table de travail. Plusieurs processus exécutés en parallèle cherchent tous à traiter les lignes qui se trouvent dans une table de travail. Il ne faut pas qu'un processus soit bloqué parce qu'un autre a verrouillé la ligne qui l'intéressait. Il est préférable qu'il passe à la ligne suivante. C'est ce problème qui est à l'origine de la clause SKIP LOCKED. En ajoutant celle-ci à la requête, toutes les lignes déjà verrouillées seront ignorées, et ce sera la première ligne libre qui sera récupérée. Dans cet exemple, la table jobs est traitée par deux processus. Le premier verrouille la ligne 1 :

```
BEGIN;
SELECT * FROM jobs ORDER BY id LIMIT 1 FOR UPDATE;
 id  | libelle
-----+---------
  1  | job 1
(1 row)
```

Le deuxième essaie d'obtenir le job suivant.

```
BEGIN;
SELECT * FROM jobs ORDER BY id LIMIT 1 FOR UPDATE;
^CCancel request sent
ERROR:  canceling statement due to user request
CONTEXT:  while locking tuple (0,1) in relation "jobs"
```

S'il utilise la même requête que l'autre transaction, il se retrouve bloqué jusqu'à la fin de la transaction précédente. Essayons maintenant avec la clause SKIP LOCKED :

```
BEGIN;
SELECT * FROM jobs ORDER BY id LIMIT 1 FOR UPDATE SKIP LOCKED;
 id  | libelle
-----+---------
  2  | job 2
(1 row)
```

Cette fois, la ligne 1 a été ignorée car elle est verrouillée, et la ligne suivante étant la ligne 2, cette dernière a été renvoyée (et verrouillée à son tour).

PERFORMANCE

Chaque nouvelle version majeure de PostgreSQL est l'occasion d'améliorer les performances. Cela se fait généralement en poussant encore plus loin les possibilités de l'optimiseur de requêtes. Bien qu'il y ait eu des améliorations en ce domaine, ce n'est pas ce qui va retenir notre attention cette fois.

BRIN

PostgreSQL dispose de plusieurs algorithmes d'index suivant le type de données à indexer : le très connu et très commun index Btree, l'index GIN pour des données souvent répétées, l'index multifonctions GiST, etc. Les travaux sur les grosses volumétries apportent l'index BRIN en version 9.5. Le but de BRIN est de pouvoir indexer facilement et rapidement un très grand volume de données. Pour diminuer sa taille, un index BRIN ne contient pas toutes les valeurs de la colonne indexée. Il ne contient que des valeurs particulières, comme les bornes inférieure et supérieure des lignes d'un ensemble de blocs. Il enregistre donc deux valeurs au lieu de plusieurs centaines, ce qui diminue considérablement la taille de l'index. Sur une table

WEBDEV®

NOUVELLE VERSION 21

CRÉEZ FACILEMENT DES SITES «RESPONSIVE WEB DESIGN»

ACCÉDANT À VOS BASES DE DONNÉES

Fournisseur Officiel de la Préparation Olympique

Rendez vos sites «Mobile Friendly».

WEBDEV 21 vous permet de rendre facilement vos sites «Mobile Friendly».
Les sites que vous créez sont ainsi mieux référencés par Google.
Responsive Web Design et Dynamic Serving sont à votre service dans WEBDEV 21.

WEBDEV** est compatible avec **WINDEV

DÉVELOPPEZ 10 FOIS PLUS VITE

www.pcsoft.fr

Des centaines de témoignages sur le site

comportant 100 millions d'entiers sur quatre octets, un index Btree sur cette colonne pèse un peu plus de 2 Go alors qu'un index BRIN arrive à peine à 128 Ko. Ça diminue aussi le coût de maintenance de l'index. Quant à son parcours lors d'une recherche, comme l'index est très petit, il est vite parcouru. Mais ensuite, une opération de vérification est nécessaire au niveau de la table pour trouver la ligne dans l'ensemble de blocs qui a correspondu à la recherche. En termes de performances, en partant du même exemple que ci-dessus, pour des données réparties de façon aléatoires, la recherche d'une valeur dans un index Btree prend 0,3 ms, alors que la même recherche dans l'index BRIN prend 1,7 ms. La différence entre les deux peut sembler énorme mais il faut comparer cela à une recherche sans index qui, elle, prend plus de deux minutes. De ce fait, la différence de performance à la recherche entre l'index Btree et l'index BRIN est minime. La vitesse de mise à jour ou de création de l'index est par contre très différente. Un index BRIN est bien plus rapide à construire et à maintenir. De plus, il est à savoir que les performances sont d'autant plus intéressantes que l'ordre logique des données est corrélé à leur ordre physique (sur disque). Le regroupement se fait par ensemble de 128 blocs par défaut, mais le paramètre pages_per_range permet de personnaliser cette valeur. Plus la valeur de ce paramètre est basse, et plus l'index est précis mais volumineux.

Optimisation des tris

Les tris sont des opérations très fréquentes dans un moteur de bases de données. Un ORDER BY, un GROUP BY, un DISTINCT, certaines jointures sont gérées avec des tris. La création d'index Btree (le type le plus commun d'index) demande un tri préalable. Donc optimiser les tris est un moyen d'augmenter les performances globales d'un serveur. Un gros travail a été fait dans ce cadre. Le tri accéléré se base sur une version abrégée de la donnée. Cela a un impact fort quand on compare des données de type textuel ou numérique.

ADMINISTRATION

Row Level Security

Le moteur PostgreSQL dispose de droits sur les différents objets qu'il gère depuis de très nombreuses versions. Il est même possible de gérer les droits au niveau des colonnes depuis la version 8.4. Néanmoins, un manque restait flagrant : l'absence de droits sur les lignes. Dans certaines applications, il est essentiel de pouvoir s'assurer que les utilisateurs n'aient droit d'accéder qu'à certaines lignes d'une même table, soit en lecture, soit en lecture/écriture. Prenez par exemple une application de gestion de ressources humaines. Un responsable d'équipe ne doit avoir accès qu'aux informations concernant les personnes de son équipe, mais pas à celles d'une autre. Gérer ce type de droits au niveau de la base de données était très difficile jusqu'à la version 9.5. Maintenant, il suffit de déclarer une politique qui ne sera appliquée qu'aux tables et qu'aux utilisateurs dûment sélectionnés. Soit une table employes contenant les lignes suivantes :

```
SELECT * FROM employes;
 id  | nom   | salaire | responsable
-----+-------+---------+-------------
  1  | emp1  |  10000 | resp1
  2  | emp2  |  10000 | resp2
  3  | emp3  |  10000 | resp1
  4  | emp4  |  10000 | resp1
  5  | emp5  |  10000 | resp3
  6  | emp6  |  10000 | resp3
(6 rows)
```

Nous allons créer une politique de sécurité pour que les responsables ne puissent voir que les lignes de cette table qui les concernent. Voici à quoi ressemble la requête de création de la politique :

```
CREATE POLICY regles_resp ON employes FOR ALL USING (current_user = responsable);
```

Nous avons été ici très permissifs vu que nous autorisons tout (clause FOR ALL). La table doit être configurée pour vérifier les politiques de sécurité, ce qui se fait avec cette requête :

```
ALTER TABLE employes ENABLE ROW LEVEL SECURITY;
```

Pour que la vérification puisse se faire, l'utilisateur doit être connecté en tant que respX. Il nous faut donc les utilisateurs au niveau du serveur PostgreSQL :

```
CREATE ROLE resp1 LOGIN;
CREATE ROLE resp2 LOGIN;
CREATE ROLE resp3 LOGIN;
```

Ces utilisateurs doivent avoir le droit de lire et écrire la table. Cela se fait avec l'ordre GRANT habituel :

```
GRANT ALL ON TABLE employes TO resp1, resp2, resp3;
```

Maintenant, en se connectant en tant qu'un des trois responsables, l'utilisateur ne verra que les lignes qui lui sont destinées :

```
postgres=# \c - resp2
You are now connected to database "postgres" as user "resp2".
postgres=> SELECT * FROM employes;
 id  | nom   | salaire | responsable
-----+-------+---------+-------------
  2  | emp2  |  10000 | resp2
(1 row)
postgres=> \c - resp1
You are now connected to database "postgres" as user "resp1".
postgres=> SELECT * FROM employes;
 id  | nom   | salaire | responsable
-----+-------+---------+-------------
  1  | emp1  |  10000 | resp1
  3  | emp3  |  10000 | resp1
  4  | emp4  |  10000 | resp1
(3 rows)
```

En fait, il faut bien voir cela comme une clause WHERE supplémentaire. D'ailleurs, la commande EXPLAIN montre le filtre en question (en plus d'autres filtres le cas échéant) :

```
EXPLAIN SELECT * FROM employes WHERE nom LIKE 'e%3';
                        QUERY PLAN
-----------------------------------------------------------------------------
 Subquery Scan on employes  (cost=0.00..21.06 rows=1 width=100)
   Filter: (employes.nom ~~ 'e%3'::text)
   ->  Seq Scan on employes employes_1  (cost=0.00..21.02 rows=3 width=100)
         Filter: (("current_user"())::text = responsable)
(4 rows)
```

Les superutilisateurs ne sont pas sensibles aux politiques de sécurité (ni à aucun autre droit). Par ailleurs, il est possible d'autoriser spécifiquement un utilisateur à contourner les règles de sécurité en lui accordant l'attribut BYPASSRLS :

```
postgres=# ALTER ROLE resp2 BYPASSRLS;
ALTER ROLE
postgres=# \c - resp2
You are now connected to database "postgres" as user "resp2".
postgres=> SELECT * FROM employes;
 id  | nom   | salaire | responsable
-----+-------+---------+-------------
  1  | emp1  |  10000 | resp1
  2  | emp2  |  10000 | resp2
  3  | emp3  |  10000 | resp1
```

Tous les mois, faites le plein de codes

Abonnez-vous à programmez!

le magazine du développeur

1 an 49 €
11 numéros

PDF 30 €(*)
1 an - 11 numéros
(*) Souscription sur le site internet

2 ans 79 €
22 numéros

Etudiant 39 €
1 an - 11 numéros

© eternalcreative

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

Toutes nos offres sur www.programmez.com

Tarifs France métropolitaine

Oui, je m'abonne

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

☐ ***Abonnement 1 an au magazine :*** 49 €

☐ ***Abonnement 2 ans au magazine :*** 79 €

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

PROG 194

Offre limitée, *valable jusqu'au 31 mars 2016*

☐ M. ☐ Mme ☐ Mlle 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*

```
 4 | emp4 |  10000 | resp1
 5 | emp5 |  10000 | resp3
 6 | emp6 |  10000 | resp3
(6 rows)
```

Cette nouvelle fonctionnalité devrait séduire les personnes ayant besoin de plus de sécurité.

SET LOGGED/SET UNLOGGED

Les tables non journalisées sont arrivées à PostgreSQL avec la version 9.1. Cependant, une contrainte existait : une table non journalisée ne pouvait être transformée en table journalisée. L'inverse était aussi vraie. Un projet GsoC (*Google Summer of Code*) a permis d'avoir cette fonctionnalité.

Quand une table passe de journalisée à non journalisée grâce à l'ordre ALTER TABLE … SET UNLOGGED, la bascule est immédiate. Par contre, lorsqu'elle passe de non journalisée à journalisée (SET LOGGED), toutes les données en cours de la table doivent être placées dans les journaux de transactions pour que les données soient envoyées aux serveurs secondaires ou enregistrées dans le cadre d'une sauvegarde PITR (*Point In Time Recovery*). Cet envoi prendra d'autant plus de temps que la table est volumineuse.

FDW

La norme SQL/MED indique comment accéder à des données externes au moteur de bases de données. PostgreSQL a commencé l'intégration de cette norme avec la version 8.4. L'intégration a été améliorée en version 9.1 avec l'ajout des tables externes, uniquement disponibles en lecture. D'autres améliorations ont suivi, comme la possibilité d'écrire dans ces tables.

La version 9.5 ajoute d'autres possibilités. L'instruction IMPORT FOREIGN SCHEMA permet la création des objets externes provenant du schéma d'une base distante. Disons que nous avons une base Oracle contenant 100 tables et 20 vues pour un schéma particulier. Auparavant, nous devions créer manuellement chaque table et chaque vue, en convertissant également manuellement les types de données. Avec IMPORT FOREIGN SCHEMA, le connecteur de données (appelé un *Foreign Data Wrapper*) récupère les définitions des relations (tables et vues) pour les créer au niveau de PostgreSQL en tant que table externe. Cela facilite grandement la mise en place d'un tel connecteur. Évidemment, le connecteur de données doit être compatible avec cette nouvelle instruction et seuls les connecteurs PostgreSQL et Oracle le sont pour l'instant. Il est aussi possible de mixer des tables locales et des tables externes dans le cadre d'une table partitionnée. Cela permet de mettre en place un partitionnement horizontal où les données ne se trouvent pas toutes sur le même serveur mais, au contraire, réparties sur plusieurs serveurs (ce qu'on appelle habituellement du *sharding*).

pg_rewind

pg_rewind est un outil qui existait auparavant en dehors du code de PostgreSQL. Son but est de synchroniser une instance avec une autre copie de la même instance, alors que les timelines divergent. Un cas classique consiste à vouloir synchroniser un ancien primaire après un failover pour qu'il puisse se rattacher au nouveau primaire en tant que serveur secondaire. L'ancien moyen de le faire consistait à recopier le répertoire des données du nouveau primaire, soit complètement soit partiellement (avec un outil comme *rsync*, capable de détecter les fichiers modifiés et de n'envoyer que ceux-là). Cependant, avec ce type d'outils, il est nécessaire de vérifier tous les fichiers. Là où *pg_rewind* montre son intérêt, c'est qu'il connaît les fichiers modifiés sans avoir à les parcourir, simplement en consultant les anciens journaux de transactions. Cela rend l'opération bien plus rapide, notamment sur les instances à grosse volumétrie.

pg_rewind consulte l'historique des timelines des instances source et cible pour déterminer le moment où elles ont divergé. À partir de là, il rejouera les journaux de transactions se trouvant dans son répertoire pg_xlog. Si jamais certains venaient à manquer (notamment si un long moment s'est passé entre le moment de divergence et le moment où *pg_rewind* est exécuté), il faudrait copier manuellement les journaux de transactions de leur répertoire d'archivage vers le répertoire pg_xlog de l'ancien serveur primaire. En effet, *pg_rewind* n'est pas capable de les récupérer sur le nouveau primaire. Une fois *pg_rewind* exécuté, en démarrant le nouveau secondaire, ce dernier va terminer l'opération en récupérant les journaux manquants à partir du répertoire d'archivage (si cela a été configuré dans le fichier recovery.conf) ou directement auprès du serveur primaire (en espérant que ce dernier les a toujours). Le nouveau secondaire se comporte donc comme n'importe quel autre serveur secondaire. L'utilisation de *pg_rewind* nécessite néanmoins une configuration préalable à la bascule :

- Le paramètre full_page_writes doit être activé (il l'est par défaut mais une vérification est hautement préférable) ;
- Le paramètre wal_log_hints ou les sommes de contrôle sur les fichiers de données doivent être activés (les deux sont désactivés par défaut).

Gestion des journaux de transactions

Avant la version 9.5, le paramètre wal_keep_segments servait à gérer deux comportements bien différents, ce qui rendait sa configuration particulièrement difficile. En effet, il était utilisé à la fois pour déclencher des opérations checkpoints en avance en cas de brusque augmentation du volume d'écriture (pensez batchs d'écriture) et pour connaître le nombre maximum de journaux de transactions conservés par PostgreSQL avant recyclage (ce qui était généralement détourné pour connaître la taille minimale de la partition dédiée aux journaux de transactions). La version 9.5 remplace ce paramètre par deux paramètres, chacun gérant un type de comportement :

- min_wal_size permet d'indiquer le volume de données minimal avant de lancer un checkpoint (tout en conservant des checkpoints réguliers grâce au paramètre checkpoint_timeout) ;
- max_wal_size indique la limite maximale, en octets, de journaux de transactions à conserver (donc avant recyclage)... Il est cependant toujours possible d'avoir plus de journaux pour différentes raisons comme un batch très important, une configuration haute du paramètre wal_keep_segments ou une commande d'archivage en échec.

Les valeurs par défaut des deux nouveaux paramètres sont déjà plus importantes que l'ancienne valeur de checkpoint_segments. min_wal_size vaut 80 Mo, ce qui équivaudrait plutôt à un checkpoint_segments à 5 (anciennement 3 par défaut). Quant à max_wal_size, il est configuré à 1 Go par défaut (correspondant à un checkpoint_segments à 64).

ET LA SUITE ?

La version 9.5 est sortie début janvier 2016. La version suivante, numérotée pour l'instant 9.6, est en cours de développement depuis mai 2015. Un grand nombre de nouvelles fonctionnalités ont déjà fait leur apparition. Néanmoins, l'ajout qui semble le plus intéressant actuellement est l'exécution parallélisée d'une même requête. Cet ajout est le résultat d'un travail considérable initié et maintenu par la société EnterpriseDB. Pour l'instant, cela concerne les parcours séquentiels et les jointures. Cependant, d'autres types d'opérations sont en cours d'étude, comme par exemple le tri. Avec ce type de fonctionnalité, il n'est pas du tout improbable que la version soit numérotée 10.0. Autre fonctionnalité dont on attend beaucoup : la réplication logique. Un autre bout d'infrastructure a été ajouté avec la version 9.5, à savoir la disponibilité de l'horodatage des COMMIT des transactions via l'appel à une procédure stockée. Le but d'une telle fonctionnalité est de résoudre les conflits dans le cadre de la réplication bidirectionnelle (multi-primaire). Cependant, la version 9.6 à l'heure actuelle n'intègre pas de nouveautés à ce niveau-là. ⊞

PHARAON + programmez! le magazine du développeur

Défi de programmation 2016 : à partir du 15 mars

Comment concilier archéologie, histoire, voyage et programmation ? Les magazines Pharaon et Programmez ! lancent un défi de programmation pour les développeurs, les étudiants en informatique et tous les passionnés !

L'objectif est simple :

Créer une app mobile capable de prendre une photo des cartouches royaux des pharaons pour lire automatiquement les noms écrits en hiéroglyphes et afficher le nom du pharaon.

L'app doit fonctionner sur smartphone ou tablette et être disponible, au moins, sur une des 3 plateformes du marché : **Android, iOS, Windows Mobile.**

Agenda

Le défi sera ouvert à partir du 15 mars pour une durée de 3 mois.
Le 15 juin, les développeurs devront envoyer leurs apps pour que la rédaction les teste et choisisse les gagnants qui seront annoncés dans le numéro d'été et/ou sur le site Web du magazine fin juin.
Trois apps gagnantes seront retenues.

Les gagnants recevront :

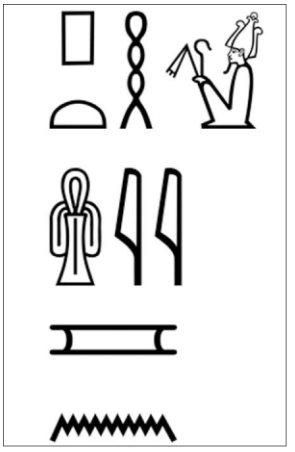
- 1 Intel NUC,
- 1 clé USB Programmez !
- 1 clé USB Pharaon Magazine,
- 1 visite du département égyptien du Musée du Louvre avec François Tonic (historien, éditeur & rédaction en chef de Programmez ! et de Pharaon Magazine),
- 2 livres : la tombe de Ramose, la tombe royale d'Akhenaton,
- présentation et article dans Programmez ! et Pharaon Magazine.

TOUS LES DÉTAILS SUR
WWW.PROGRAMMEZ.COM

EXEMPLE

1 - *L'utilisateur prend la photo du cartouche :*

2 - *L'app reconnaît les deux cartouches et signes hiéroglyphiques :*



3 - *Réponse donnée à l'utilisateur :*
Menmaâtrê
L'Osiris, celui de Seth, l'aimée de Ptah
= le pharaon Séthy Ier
(hiéroglyphes réalisés avec l'outil JSesh)

Forensic tools for in-depth performance investigation

"40% des personnes abandonnent un site qui met plus de 3 secondes à se charger. Une seconde de délai entraîne jusqu'à 7% de réduction de conversion. Si un site e-commerce génère 100 000€ de chiffre d'affaires par jour, 1 seconde de délai induit potentiellement une perte de 2.5 millions d'euros chaque année".

Katherine Daniel *VelocityConf 2015 @beerops*

Adrien Le Priol
Ingénieur Technique WebOps chez **Oxalide**
Adrien.LePriol@oxalide.com

Fort de ce constat, il est nécessaire d'adresser les problématiques de performance au cœur des enjeux business.

Des connexions inégales face aux performances des sites

La première cause de lenteurs est le poids des pages du site. Théoriquement, voici le temps de chargement des pages en fonction de la connexion à Internet.

Taille	ADSL (2Mbs/s)	3G	ADSL (24Mbs/s)	Fibre
500Ko	2s	0s	0s	0s
1Mb	4s	1s	0s	0s
5Mb	20s	5s	1s	0s

Mais ce n'est que théorique, car pour apprécier la performance d'une page, la qualité de connexion à Internet est essentielle dans cet exercice. Tout le monde n'est pas en plein centre d'une grande ville où l'accès au haut-débit ou très haut-débit (ADSL, Fibre ou encore la 4G) est un standard. Et on l'oublie souvent, on oublie ces zones où les connexions ADSL et mobiles sont présentes, mais où la vitesse et/ou la stabilité d'accès au haut-débit font parfois défaut.

Sans compter leur temps de génération, une bonne pratique DevOps vise à s'octroyer une charge utile (payload) maximum sur une page 500Ko. Au-delà de cette préconisation, mesurer et surveiller cet indicateur est nécessaire pour apprécier la balance "bénéfice et risque".

Optimiser les performances d'une page Web : limiter les transactions avec le serveur

Trouver les leviers de performance : s'outiller

L'optimisation du poids peut se faire autour de fonctionnalités à optimiser. Elle se base sur de nombreux outils pour analyser la construction complète des pages. Les outils nous aident à déceler des problèmes évidents de performances comme des fichiers statiques volumineux tels que les images, mais également les ressources externes comme des Javascripts ou les bannières publicitaires. Des sites comme www.webpagetest.org, gtmetrix.com ou encore https://developers.google.com/speed/pagespeed/insights

Une autre aide précieuse, celle du navigateur, disposant d'un mode développement qui analyse le temps de chargement d'une page. L'analyse de la construction des pages nous oriente dans les axes d'amélioration mais également de performance technique dans le choix des prestataires.

Gratter quelques millièmes de seconde : le poids insoupçonné des requêtes DNS et des redirections

Dans ce monde où chaque transaction, chaque bit, chaque appel compte, il y a des points dont on ne mesure pas suffisamment l'impact : les requêtes DNS et les redirections.

Les requêtes **DNS** : l'idée est de réduire au maximum l'appel à des ressources extérieures. Il est également possible en HTML, que le navigateur n'effectue la résolution DNS qu'une fois pour une liste de ressources données. En effet, une requête DNS par ressource externe induit des requêtes inutiles et donc contre-performantes.

L'idée étant d'indiquer au navigateur la liste des domaines présents dans la page afin de pré-résoudre les requêtes une fois pour toutes.

```
<html>
  <head>
    <link rel="dns-prefetch" href="//www.domain1.com">
    <link rel="dns-prefetch" href="//www.domain2.com">
  </head>
  <body>
    <img src="www.domain1.com/image1.jpeg">
    <script src="www.domain2.com/script1.js">
  </body>
</html>
```

Les **redirects** : on ne s'en rend pas forcément compte quand on navigue sur Internet, mais bien souvent une simple page Web est composée d'énormément de liens qui finissent parfois par des enchaînements de redirections. Par exemple les liens de tracking de Google peuvent rediriger jusqu'à 4 fois l'internaute.

Grâce aux graphiques de bande passante et d'utilisation du CPU fournis par www.webpagetest.org, il est possible de voir à quel moment du chargement de la page ces ressources ne sont pas correctement utilisées.

D'une manière générale ce sont les ressources Javascript qui consomment le plus de CPU.

C'est pour cela qu'il est conseillé, dans la mesure du possible, de charger ces éléments à la fin de la page et ainsi privilégier le rendu client.

Un site Web non optimisé Fig.1.

Un site web optimisé Fig.2.

Au-delà des performances brutes, il faut bien prendre en considération que ces petites améliorations permettent bien souvent de mettre en avant son site sur les moteurs de recherches.

Communément appelé référencement naturel, il s'agit bien là de trouver sa place sur ces moteurs, gratuitement.

Ne transférez pas plusieurs fois la même donnée : accélérez avec le cache

Le cache est le mécanisme qui permet de livrer rapidement un élément sans repasser par la case transfert dans notre cas. Il existe deux niveaux de cache utilisables : le cache navigateur (Chrome, Firefox, Safari, Opéra) et Varnish.

- Le cache navigateur : " La mise en cache de documents Web (ex : page Web, images) est utilisée afin de réduire la consommation de bande passante, la charge du serveur Web (les tâches qu'il effectue), ou d'améliorer la rapidité de consultation lors de l'utilisation d'un navigateur Web. Un cache Web conserve des copies de documents transitant par son biais. Le cache peut, dans certaines conditions, répondre aux requêtes

ultérieures à partir de ses copies, sans recourir au serveur Web d'origine." (source Wikipédia).

- Le cache Varnish : " Varnish est un serveur de cache HTTP apparu en 2006 et distribué sous licence BSD. Déployé en tant que proxy inverse entre les serveurs d'applications et les clients, il permet de décharger les premiers en mettant en cache leurs données, selon des règles définies par l'administrateur système et les développeurs du site, pour servir plus rapidement les requêtes, tout en allégeant la charge des serveurs." (source Wikipédia).

Pour le navigateur : c'est le "header expire" défini dans la RFC 2616 qui permet de définir la validité dans le temps d'une ressource. Interprétée par les navigateurs, elle permet aux clients Web de conserver les éléments dits statiques (images, css, html, javascript) dans le cache (stocké sur le disque dur).

Ainsi, si une ressource est appelée dans une autre page, le navigateur n'ira pas interroger le serveur et économisera ainsi énormément de temps.

Par exemple, lorsque l'index d'un site Magento avec les images, les feuilles de style et Javascript est téléchargé une fois. Au prochain appel si le "header expire" indique une date supérieure à l'instant présent, alors mon navigateur n'ira pas redemander la ressource mais utilisera celle qu'il a mémorisée.

Pour ce faire, il faut activer le mod_expire dans Apache. La configuration peut se réaliser de 2 manières soit dans la configuration d'Apache ou ajouter dans le .htaccess à la racine du site :

/etc/apache2/sites-enables/monsite.fr ou .htaccess

```
<IfModule mod_expires.c>
 # Enable expirations
 ExpiresActive On
 # Default directive
 ExpiresDefault "access plus 1 month"
 # My favicon
 ExpiresByType image/x-icon "access plus 1 year"
 # Images
 ExpiresByType image/gif "access plus 1 month"
 ExpiresByType image/png "access plus 1 month"
 ExpiresByType image/jpg "access plus 1 month"
 ExpiresByType image/jpeg "access plus 1 month"
 # CSS
 ExpiresByType text/css "access plus 1 month"
 # Javascript
 ExpiresByType application/javascript "access plus 1 year"
 </IfModule>
```

Il est plus commode d'utiliser un fichier .htaccess car le fichier est directement accessible à un développeur pour modifier la configuration des durées de rétention.

Imaginons, nous sommes le 10 février 2016 avec la configuration ci-dessus, le header renvoyé pour toutes images et css sera : Expires : Tue, 10 Mar 2016 12:08:08 GMT

Pour Varnish, le "header expire" est également interprété ; lorsqu'un client A télécharge depuis le serveur, une image sera immédiatement stockée en RAM dans Varnish le temps du expire pour les clients suivants. Cela garantit une réponse rapide des statiques, comme un cache navigateur mutualisé. Lorsqu'un premier visiteur télécharge une page produit sur un site marchant avec ses images, feuilles de style, Javascripts, Varnish mémorise le retour que lui fait le serveur Web sur les requêtes du type GET sans paramètres le temps du expire indiqué. Ainsi le second visiteur qui télécharge la même page produit verra les éléments statiques renvoyés par Varnish sans computing (intelligence CPU) que génèrent les serveurs Web. Aucune configuration "header expire" est nécessaire car il est native dans Varnish. En revanche, il est possible de récupérer le header cache-control renvoyé par certains applicatifs comme Drupal :

/etc/varnish/default.vcl

```
if (resp.http.cache-control ~ "max-age=" ) {
    set resp.http.Expires = "" + (now + std.duration(regsub(resp.http.cache-control,"max
-age=([0-9]+).*$","\1")+"s", 0s));
 }
```

Des solutions coté serveurs : le HTTP/2 et le SaaS

Ces modifications sont bien souvent à appliquer côté client (développement), notre position de DevOps nous oblige à travailler avec le client pour améliorer ces points.

Le HTTP/2 issu du travail sur le protocole parent SPDY est de plus en plus implémenté sur les navigateurs modernes. En revanche Apache et Nginx, les serveurs Web les plus populaires, ne supportent pas officiellement ce protocole sauf par le biais de modules tiers expérimentaux.

Il est pour le moment risqué d'implémenter uniquement ce protocole directement en production, mais néanmoins nécessaire de préparer dès à présent la transition.

Le module Apache/Nginx PageSpeed développé par Google permet à la volée, de compresser et minimiser les images, Javascripts, CSS. Il est intéressant et même indispensable de coupler ce module avec un système de cache HTTP type Varnish. D'autant plus qu'il intègre une gestion des headers expirée utilisée par défaut par Varnish pour la durée du cache des objets. Si le souci de performance est au centre de la préoccupation client et que malheureusement il n'a pas les moyens d'améliorer la situation, il existe des services comme Fasterize qui est un reverse proxy - cache qui fait de la compression de données et l'optimisation de réseau avec le support du protocole HTTP2 par exemple.

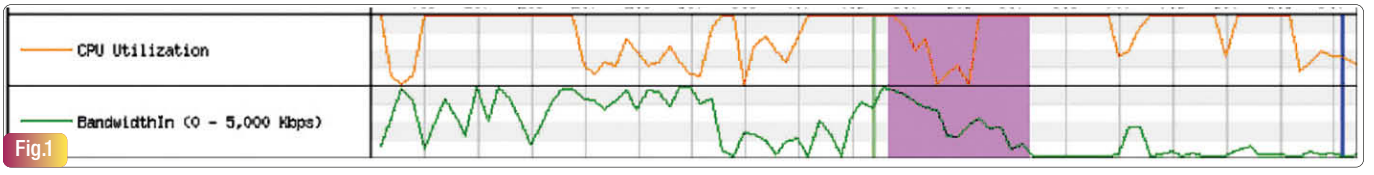


Fig.1

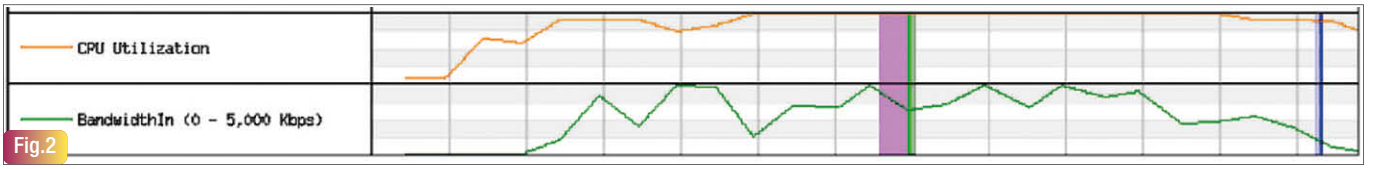


Fig.2

Blague 01

Combien de développeurs faut-il pour changer une ampoule ?

Aucun, c'est un problème matériel.
(merci à CodeChef)

Définition

Dictionnaire : robot

D.R.

Robot vient de robota, mot tchèque inventé par Karel Capek signifiant « corvée », « travail », « servage ». Ce mot apparaît dans sa pièce de théâtre « R.U.R. » en 1920. Il faut comprendre « robot/robota » comme une force de travail. Dans cette première apparition, robot n'a pas le sens qu'on lui donne aujourd'hui. Il s'agit d'une personne artificielle non pas mécanique, mais chimique. Le robot mécanique supplante totalement la définition première de robot dans les années 1950.

D.R.

Série

Halt and catch fire : bientôt la ~~saison~~ 3

photo : Tina Rowden / AMC

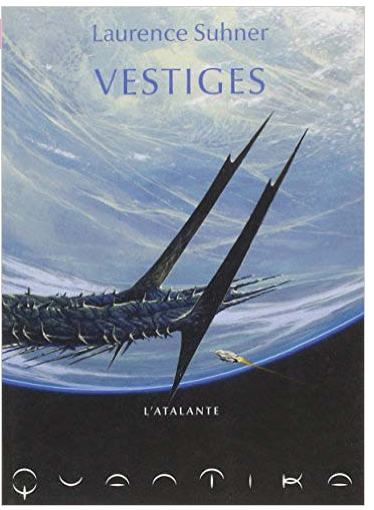
Avez-vous vu cette série ? Halt and catch fire revient sur les coulisses de l'informatique durant les années 80 avec des sociétés fictives pour les « héros », mais se confronte à de vrais constructeurs et éditeurs de l'époque : IBM, Apple, Microsoft, Atari... La première saison évoque un PC portable que MacMillan veut imposer à une société qui fait surtout du mainframe. Trahisons, tensions, mensonges, de bons ingrédients. Comment faire rentrer les composants dans un espace réduit ? Quid du design ? Quid du système ? Une jeune développeuse, un peu hackeuse, va porter le projet mais elle ira de déconvenues en mésaventures. De nombreux passages rappellent la véritable histoire de l'informatique : Apple qui va présenter son Macintosh, le rôle du design, IBM et Microsoft et le fameux MS-DOS,... C'est aussi l'époque des Dell et Compaq. Dell est créée au Texas (action de la série) en 1984.
Nous avons adoré la saison 1. La saison 2 est un cran en-dessous, à vouloir mêler trop d'histoire, l'origine des forums et des réseaux communautaires. Elle manque de cohésion par rapport à la saison 1.
La saison 3 devrait arriver en mai ou juin et se déroulera dans la Silicon Valley.

P.S. : le nom de la série fait référence à une inside joke sur une instruction pouvant faire chauffer les composants pour qu'ils prennent feux...

Littérature

La saga Quantika

Laurence Suhner, auteure suisse de son état, nous livre avec Quantika une trilogie de science-fiction digne des plus grands écrivains anglo-saxons. Trois tomes, plus de 1 500 pages d'une écriture nerveuse, nous font vivre un planet opera somptueux où l'accent est mis sur des personnages attachants emportés dans une aventure fascinante et trépidante.
L'action se déroule principalement sur Gemma, une planète glacée à quelques dizaines d'années-lumière de la Terre. Une colonie humaine s'y est installée, malgré la rigueur du climat, afin d'exploiter les ressources de la planète. En orbite, un artefact extra-terrestre impénétrable veille sans rien dévoiler de sa vraie nature.
Ambre, une scientifique de renom débute des fouilles sur Gemma et découvre des vestiges pouvant s'apparenter à ce mystérieux vaisseau. Avec son équipe, elle « réveille » une entité enfouie au cœur du glacier depuis des milliers d'années et comprend peu-à-peu ce que sont les fameux « Bâtisseurs ». Tokalinan revient à la vie, lui qui s'est écrasé avec son vaisseau 12 000 ans auparavant, et va entraîner Ambre et ses compagnons dans la découverte du Grand Arc, le vaisseau-monde dans le ciel de Gemma.
Tour à tour haletante, mystique, émouvante, cette saga se lit d'une traite et ne révèle ses mystères que tardivement. La force de l'auteure est d'être restée focalisée principalement sur une petite dizaine de personnages, en explorant leurs peurs, leurs espérances et leurs relations avec Tokalinan, cet être venu d'ailleurs. Une réussite, à dévorer sans modération !



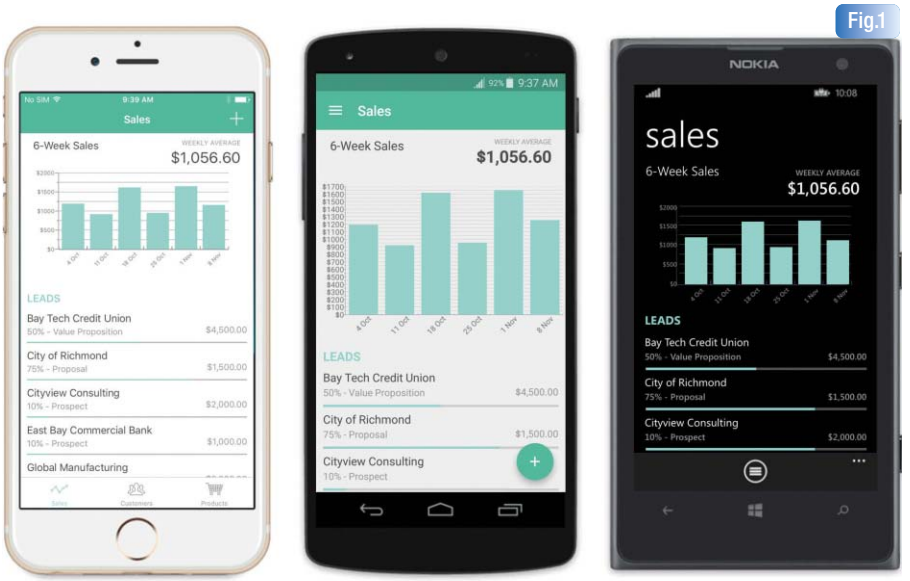
Quantika, aux éditions Atalante : Vestiges (2012), L'ouvreur des chemins (2013), Origines (2015)

Les nouveautés de Xamarin 4

Accessible depuis la fin du mois de Novembre 2015, la nouvelle version majeure de Xamarin (Xamarin 4) introduit son lot de nouvelles fonctionnalités que nous allons découvrir tout au long de cet article.

Thomas LEBRUN
Infinite Square
Consultant
tlebrun@infinitesquare.com
http://blogs.infinitesquare.com/b/tom
http://blog.thomaslebrun.net

Fig.1



La plateforme Xamarin

Au sein de la plateforme Xamarin, composée elle-même de plusieurs produits, on retrouve tout un ensemble de nouvelles fonctionnalités, à commencer par certaines qui touchent à Xamarin.Forms, avec l'arrivée de Xamarin.Forms 2.0 ! Fig.1.

Au programme, on y retrouve des optimisations de performances (qui faisaient partie des points noirs sur les premières versions), une version préliminaire du support des applications universelles de la plateforme Microsoft (Universal Windows Platform), l'introduction du support du « Material Design », si cher à Android et, également, de nouvelles gestures comme le « Pinch » et le « Pull-To-Refresh ».

Mais, avec Xamarin.Forms 2.0, on découvre une fonctionnalité importante sur la partie performance : le support des écrans précompilés. En effet, dans les versions précédentes des applications Xamarin.Forms, les écrans étaient développés en utilisant le langage XAML et, au lancement de l'application, ils étaient recompilés à la volée pour être affichés à l'utilisateur. Avec cette nouvelle version de Xamarin.Forms, le XAML est compilé directement en langage intermédiaire (IL) par le compilateur XAML. Cette différence, pouvant sembler mineure, permet cependant de bénéficier des améliorations suivantes :

- Une validation, à la compilation, du code XAML (afin de notifier le développeur des éventuelles erreurs) ;
- Une réduction du temps de chargement et d'instanciation des éléments XAML ;
- Une réduction du poids final de l'assembly généré en faisant en sorte que les fichiers XAML ne soient plus inclus dans le binaire.

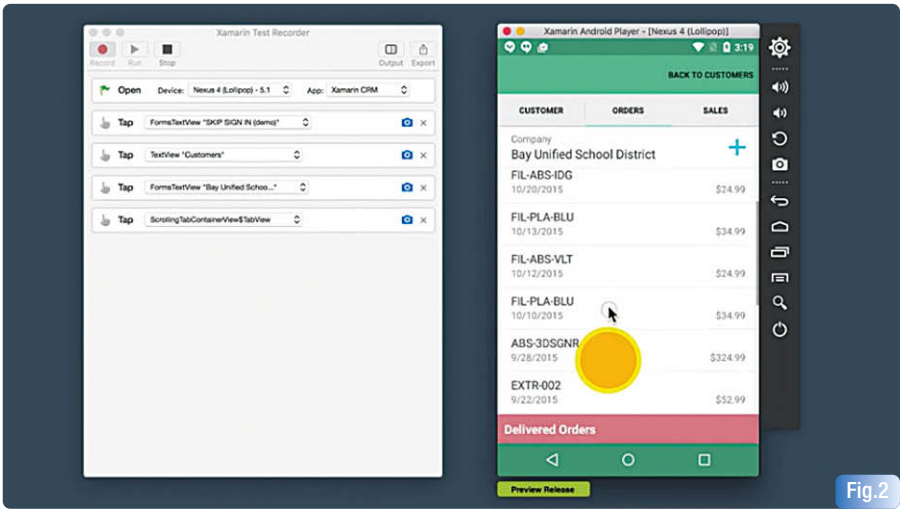


Fig.2

Côté Visual Studio, les équipes de Xamarin ont réécrit le support des applications iOS dans l'IDE Microsoft. Ainsi, si le Mac reste toujours obligatoire, il n'est plus nécessaire d'installer le Xamarin Build Host : l'ensemble de la communication entre le PC et le Mac passe désormais par des connexions SSH, permettant également de faire en sorte que plusieurs utilisateurs puissent se connecter en même temps (là encore, il s'agissait d'une forte limitation lors du développement d'application Xamarin.iOS). De plus, le designer iOS est maintenant en mesure de charger et sauvegarder des fichiers XIB en plus des fichiers de Storyboard.

Le designer Android a, lui aussi, été amélioré avec le support du Material Design et pas mal d'autres petites fonctionnalités (amélioration de l'interface graphique, interface d'édition plus rapide et fluide, etc.).

Au niveau du Framework Mono, les équipes de Xamarin ont profité de la mise à disposition d'une plus large partie du Framework .NET (de la part de Microsoft), pour y intégrer certaines parties. En effet, certaines briques de Mono étaient, à l'heure actuelle, potentiellement erronées (car basées sur du code « déduit » depuis le code du Framework .NET), ce qui n'est plus le cas dorénavant et assure donc une meilleure compatibilité et de meilleures performances !

Xamarin Test Cloud

L'offre de tests de Xamarin, Xamarin Test Cloud, a également évolué avec l'apparition d'un nouveau produit, Xamarin Test Recorder : Fig.2. Pour le moment uniquement disponible sur Mac (on attend tous la version Windows avec impatience), l'objectif de l'outil est d'enregistrer l'ensemble des actions/interactions que vous réalisez avec votre application iOS ou Android. Une fois votre séance terminée, l'ensemble des cas de tests sont immédiatement créés pour pouvoir être

joués dans Xamarin Studio ou Visual Studio ! L'ensemble des tests sont enregistrés avec le Framework de tests de Xamarin, UITest, afin que vous puissiez les exécuter de manière automatique dans un processus d'intégration continue (de manière locale ou dans le Cloud, avec Xamarin Test Cloud).
De son côté, ce fameux Framework de tests, UITest, passe en version 1.0 et se trouve être gratuit pour l'ensemble des utilisateurs, sans limitation sur la durée d'exécution du test (c'était une des limite de la version préliminaire) ou sur l'utilisation d'un vrai périphérique ou d'un simulateur.

Xamarin Insights

Pour finir dans les nouveautés de cette nouvelle version majeure, il est nécessaire de regarder ce qui a été annoncé autour de Xamarin Insights, l'outil de Xamarin permet de mettre en place et suivre le monitoring de vos applications (quelles sont les pages visitées, quels est/sont le(s) bouton(s) le(s) plus cliqué(s) ?, etc.) : Fig.3.
Disponible de manière générale, la mise en place de Xamarin Insights peut se faire en quelques lignes de code et se trouve accessible à l'ensemble des abonnés Xamarin, sans supplément de prix. En fait, cette offre est bien plus étoffée que cela et comprend :

- L'accès aux crashs et erreurs, au travers de Xamarin Insights, sur une durée de 30 jours (avec accès à l'ensemble des informations nécessaires et détaillées) ;
- La mise à disposition de 60 minutes de location de périphériques, pour Xamarin Test Cloud ;
- L'accès complet au Framework Xamarin.UITest, pour la rédaction des tests ;
- Une période de 30 jours pour tester et évaluer les bénéfices de la Xamarin University (les cours de formation proposés par Xamarin à ses abonnés).

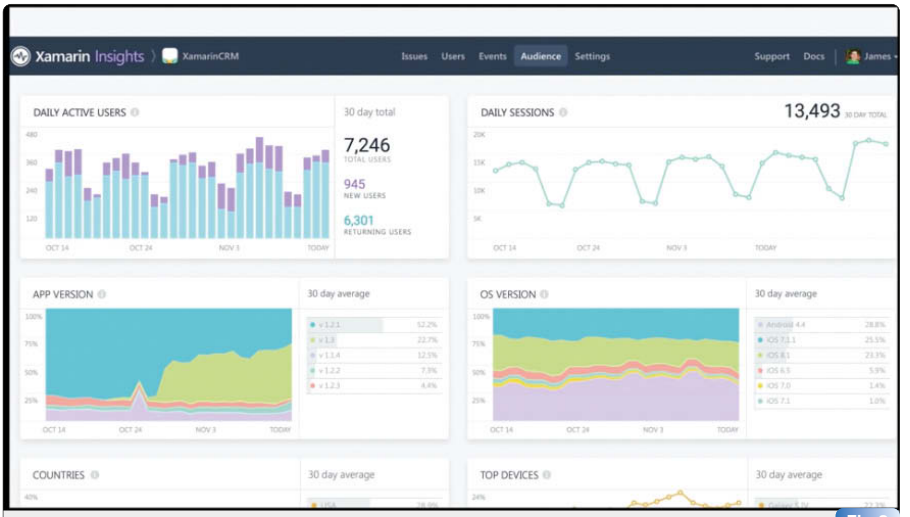


Fig.3

Si vous n'êtes pas un utilisateur de Xamarin mais que vous souhaitez tout de même utiliser Xamarin Insights dans vos applications, vous pouvez souscrire à une offre dont les prix, qui varient en fonction des besoins, commencent à 99$ par mois. N'hésitez pas à visiter la page dédiée pour en savoir plus : https://xamarin.com/insights#pricing

Xamarin Ultimate

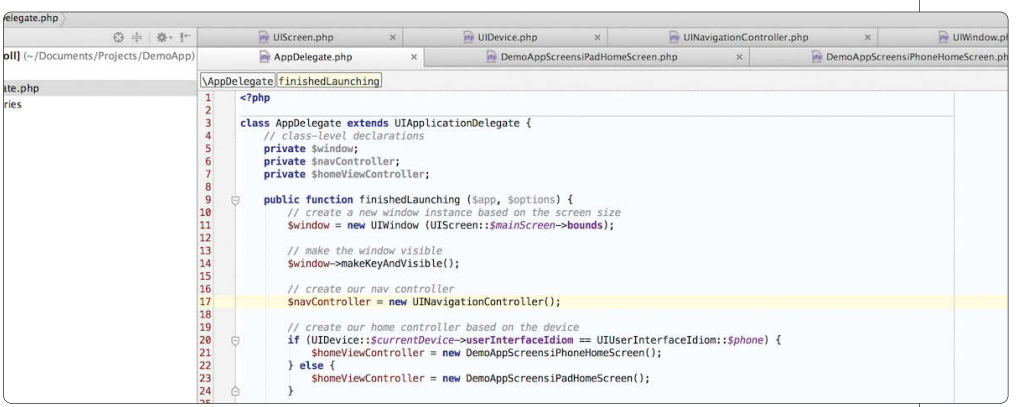
Parce qu'il ne faut pas laisser de côté l'aspect commercial des produits, une nouvelle offre fait son apparition dans le catalogue des équipes de Xamarin : Xamarin Ultimate.
Comme il est possible de le deviner via son nom, l'idée de cette offre est de permettre la mise à disposition de l'ensemble des produits de la gamme Xamarin (soit l'accès à la plateforme Xamarin, Xamarin Test Cloud, Xamarin Insights et la Xamarin University) pour l'ensemble des collaborateurs de votre équipe.
Comme il est possible de s'en douter, le prix d'une telle offre est extrêmement compliqué à estimer car il dépend, principalement, du nombre de collaborateurs qui seront concernés. Du coup, si cette offre vous intéresse, vous devez simplement envoyer un email à l'adresse email : hello@xamarin.com
Pour en savoir plus sur Xamarin Ultimate, vous pouvez vous rendre sur la page suivante : https://xamarin.com/ultimate

Conclusion

La plateforme Xamarin ne cesse d'évoluer, tous les ans, pour s'enrichir de nouvelles fonctionnalités et permettre aux développeurs d'améliorer leur expérience. La conférence Evolve 2016 (https://evolve.xamarin.com), qui aura lieu à la fin du mois d'Avril 2016, permettra sans doute de découvrir les nouveautés qui nous ont été réservées par les équipes de développement d'un produit en constante évolution !

PHAP : développer en PHP des apps mobiles

Voilà un projet que l'on n'attendait pas : PHAP. Le but est très simple : développer des apps mobiles avec le langage PHP. Un même code pour iOS, Android et Windows Mobile. Pour la partie interface, la solution s'appuie sur HTML5 et CSS3. Un wrapper JavaScript, PHAPJS est même disponible ! Les créateurs mettent en avant : la simplicité du langage, une syntaxe rapide à maîtriser, l'utilisation de Composer pour gérer les dépendances, l'utilisation de PHPUnit pour les tests. Un IDE dédié, basé Eclipse, sera prochainement disponible : PHAP. La solution est toujours en développement mais une version sera bientôt disponible. Site : http://phap.landingpage.io



La rédaction

Choisir sa solution de développement mobile

Le développement mobile, nous vous en parlons dans le magazine. Régulièrement, une question se pose : comment choisir son environnent de développement pour apps mobiles ? Natif ou multiplateforme ?

La question reste plus que jamais d'actualité avec l'explosion des apps et une utilisation toujours plus intensive. Pour le développeur, surtout quand il est indépendant ou dans une petite structure, le choix de la technologie est un critère crucial. Sans oublier la complexité de ces solutions et du coût financier de certaines d'entre elles...

Ce choix repose sur de nombreux critères et pas uniquement techniques. Il faut trouver le bon compromis. Basiquement, il y a deux catégories de développements mobiles :

- *Natif : avec portage du code sur x plateformes si nécessaire*
- *Multiplateforme : un code de base ciblant x plateformes, avec adaptation de l'interface et de morceaux de codes si nécessaire.*

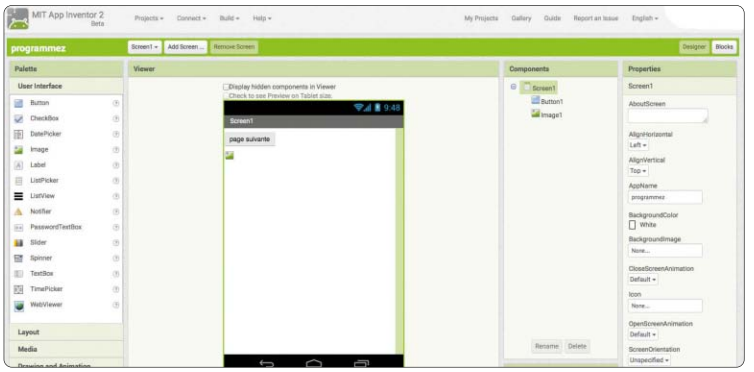
Dans l'approche multiplateforme, vous avez le choix les solutions HTML / JavaScript / CSS packageant des apps et les solutions un langage de type C# sur lequel on cible les différents systèmes.

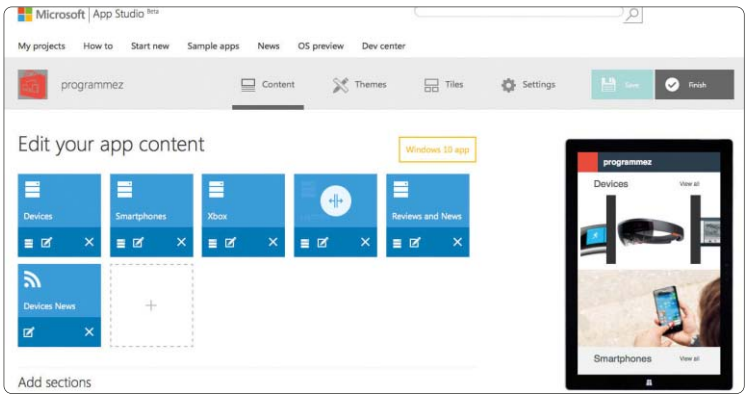
Dans ce mini-dossier, nous allons vous aider à comprendre les enjeux et à établir les critères de choix objectifs.

La rédaction

App mobile : approche IDE ou approche métier ?

Les technologies évoluent très vite et le développement mobile a explosé lorsque Apple a mis en place l'App Store et que Google a imposé Android. Les solutions de développement apparaissent, font le buzz puis retombent dans le train-train, voire disparaissent. Aujourd'hui, il y a grosso modo 3 manières d'aborder le développement mobile : natif, hybride (mêlant un peu de natif et beaucoup de JavaScript, CSS, HTML5), et le vrai multiplateforme de type Xamarin (1 code pour x systèmes cibles). Mais, n'oublions pas un élément important : en entreprise, une App mobile peut être vue de différentes manières et surtout peut s'intégrer avec une multitude de technologies, de protocoles. Il faut aussi se rendre compte que développer un jeu mobile est à la fois proche et éloigné d'une App métier/business.





François Tonic

Êtes-vous pour une approche « IDE » ou « métier/services » ?

Pour comprendre cette réflexion, ayons à l'esprit qu'un projet se comprend dans sa globalité, c'est-à-dire avec une vision à 360°. Quand on reste au niveau du développeur, souvent, on regarde le code, la base de données. Mais dans un projet d'entreprise, ce développement va s'intégrer, souvent, dans un espace plus large ; cela signifie qu'il va se connecter, interagir avec d'autres services, d'autres systèmes, d'autres protocoles et technologies. La direction, le département informatique et le développeur pourront avoir 3 visions différentes d'un seul et même projet, avec des mots différents que l'autre ne comprendra pas ou mal.

> *« Une App n'est pas que du code, surtout en entreprise »*

L'un parlera de service d'entreprise à délivrer en interne ou aux clients, un autre de répondre à la demande de l'entreprise et enfin, le dernier verra du code et un développement à réaliser selon des spécificités plus ou moins bien définies (quand elles existent).

Des enjeux pour les App mobiles d'entreprise et les services d'entreprise

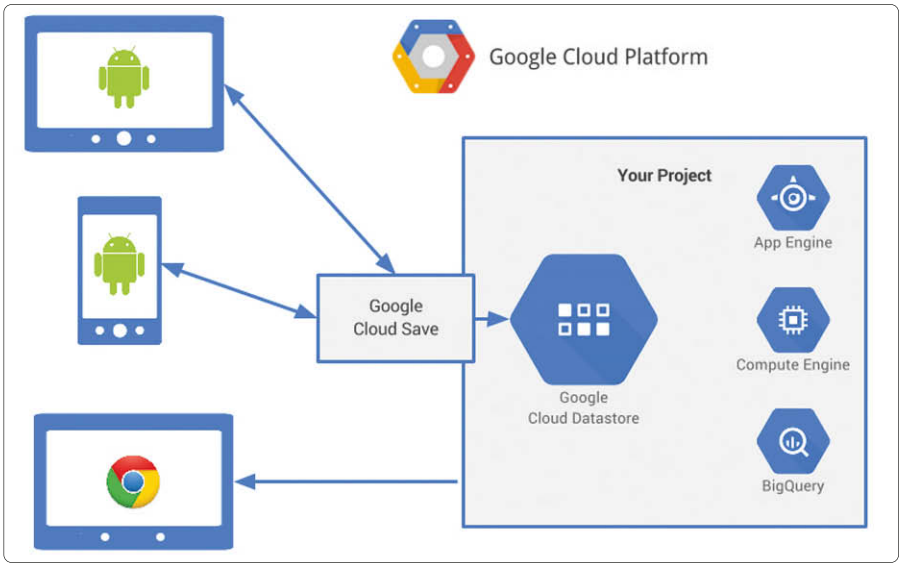
Vous me direz que développer une App métier/d'entreprise est la même chose que de développer un jeu mobile. Fondamentalement, vous auriez raison. Mais les contraintes seront plus ou moins fortes selon l'activité de l'entreprise et le service à proposer.
Prenez par exemple un système de service bancaire ou un service de réservation de voyages. L'app aura une partie front-end (interface, logique minimale) s'appuyant sur un back-end beaucoup plus important : multiples requêtes pour les données sur les hôtels, les transports, etc., gestion de l'authentification, gestion du paiement, gestion de la confirmation du voyage. Le paiement est l'un des services les plus sensibles. Pour le bancaire, les protocoles et systèmes sont parfois anciens, mais sont toujours là comme le CICS que l'on attaque via des API, des gateways. Chaque banque fournira des API spécifiques à elle et selon le type de développement que l'on fait. PayPal fournit aussi des API dédiées mobiles.
Vous souhaitez utiliser les paiements NFC ? Pas de souci, mais quelles technologies choisir ? Je veux utiliser Apple Pay ou un système équivalent ? OK, mais ces systèmes sont-ils disponibles dans les pays où mon App mobile sera utilisée ? Par exemple, en France, Apple Pay n'est pas déployée... Sans compter la gestion des taux de taxes. En France, vous n'aurez que l'embarras du choix...

Les Apps mobiles internes se multiplient

Avec des parcs mobiles de plus en plus importants en entreprise, la mobilité croissante des salariés, il est aujourd'hui nécessaire de déployer des Apps mobiles pour répondre aux usages et aux demandes internes. Cela signifie que ces Apps n'ont pas vocation à sortir de l'entreprise. Et là, vous aurez une multitude de situations et finalement, chaque développement sera spécifique.
On ne demande pas forcément de développer des belles Apps et que ces développements demandent au moins 5-6 mois. Non, on parle de quelques semaines, un trimestre maximum pour avoir quelque chose d'utilisable. Bref, c'est l'App mobile « juste à temps »(1). L'App doit répondre à un besoin précis et tout de suite. En PME ou dans des départements de grosses sociétés, ce sont des enjeux cruciaux pour faciliter (?) la vie du salarié et du service. Il faut aller vite. Parfois, pour répondre à ces demandes urgentes, les départements IT ne

(1) Le terme JIT, Just In Time, est bien plus joli

sont même pas sollicités, on conçoit des macros, des Apps assemblées. C'est une partie de ce que l'on appelle le shadow IT.
On assemble des briques, des composants. On « compile » et on utilise. Ou alors, on utilisera des environnements très métiers comme WinDev, force.com ou encore Kony, pour répondre aux besoins de ses commerciaux, des secrétaires, de son patron, etc. Le développeur aura sa place, mais dans des développements de type Xamarin, Java, etc.
Nous sommes dans le fonctionnel pur et dur. C'est comme si, un jour, le responsable marketing/clientèle d'une PME, venait me voir : « bon j'ai besoin d'un tableau de bord pour mieux suivre les ventes et le service client. Nous avons un peu de Salesforce, de l'Excel. Tu me fais ça pour vendredi... Ah oui, j'en ai besoin sur mon smartphone et ma tablette... ». Euh, nous sommes mardi... Là typiquement, il faut de l'assemblage, des composants, des connecteurs.
Souvenez-vous de la notion de mashup d'il y a quelques années. C'est totalement cela.
Une App de gestion de stock utilisée directement dans les entrepôts devra se connecter à une base de données dédiée elle-même sans doute interrogée par différentes applications. En entrepôt, il faudrait utiliser des douchettes de lectures, parfois des fonctions de géolocalisations. C'est dans le spectre fonctionnel des solutions citées ci-dessus. Un développement C#, Java, pur natif, sera sans doute plus long.

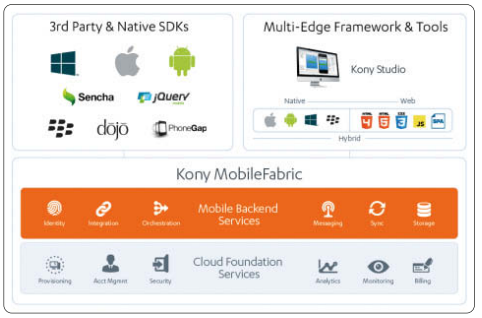


Des outils bien loin de l'IDE et autres Cordova, Xamarin...

Sans IDE, point de développement ? En réalité, il existe de nombreux environnements de conceptions d'Apps mobiles, sans codes ou très peu. On ajoute des composants/modules, on configure/paramètre et très rapidement, on peut créer une App mobile présentable. Pour une PME, un indépendant, une boutique, une école, cette approche a plusieurs intérêts :

- Aller vite et avoir une vitrine ;
- Utiliser éventuellement des données, un catalogue existant ;
- Répondre à des besoins métiers/fonctionnels et business ;
- Palier aux manques de compétences internes ;
- Ne pas utiliser des solutions et outils disproportionnés pour des besoins « basiques ».

Comme nous le disons depuis le début, de nombreuses Apps n'ont pas besoin de sortir toute la panoplie du parfait développeur mobile (Visual Studio, XCode, Android Studio, Xamarin, Cordova, etc.). C'est disproportionné et inutile, surtout quand vous mettez en face le budget et les délais... Et là, c'est la panique assurée.



Avec un outil en ligne de type App Studio, en quelques heures, il est possible de créer une App mobile, mais uniquement dédiée aux environnements Windows. Toutes les fonctionnalités ne sont pas disponibles, mais vous pouvez créer une App vitrine sans code.
Sur Android (et uniquement pour Android), vous pouvez utiliser MIT App Inventor 2 (http://ai2.appinventor.mit.edu). Il possède une vue design pour créer l'interface et une vue blocs. Cela ressemble beaucoup au modèle Scratch où l'on crée les interactions avec des blocs.
Pour les développeurs indépendants, ces outils ne sont pas à négliger quand vous travaillez avec des petites sociétés, des artisans, les budgets y sont souvent faibles et les délais inexistants,ou presque. Si vous travaillez en entreprise, ces outils permettent, vous l'aurez compris, de répondre rapidement à un besoin, une demande et vous serez capable de prototyper une première version en quelques heures contre, souvent, plusieurs jours, avec un environnement « standard ».
Bien entendu, il faut être conscient des limitations de ces solutions et ne pas espérer tout faire ou faire espérer des killer Apps qui nécessiteraient plusieurs semaines/mois de développement.

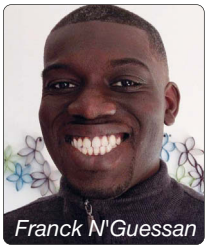
NE PAS OUBLIER LES MBAAS

Le MBAAS est le « Mobile Backend As A Service ». C'est à dire des services disponibles chez des fournisseurs de Cloud Computing pour les Apps mobiles. Parfois, on utilise aussi le terme BaaS. Ces services peuvent être plus ou moins divers selon le fournisseur. On y trouve souvent du stockage, de l'authentification, des mécanismes de notifications... Ils s'utilisent via des SDK et des API. Ils sont souvent payants (selon les ressources consommées), mais un usage gratuit est parfois disponible (avec des quotas).

Le MBaaS permet de déporter sur le Cloud une partie du stockage et des fonctionnalités. Cela doit simplifier le travail du développeur et être le plus indépendant possible du terminal et de son système pour rapidement porter, si nécessaire, son App. Mais les MBaaS utilisent parfois des API dédiées à chaque plateforme (cela dépend du fournisseur MBaaS). Ainsi, vous devrez utiliser les API Android pour l'App Android, iOS pour iOS, etc.

Ces services fournissent désormais de nombreuses fonctions que le développeur peut, ou non, utiliser. Microsoft, Amazon, Axway, Google, et d'autres, proposent leur MBaaS. À vous de choisir... Aujourd'hui, il existe plusieurs dizaines de fournisseurs MBaaS, avec tous les niveaux fonctionnels, gratuits, payants, freemiums...

Le développement mobile, est-ce rentable ?


Franck N'Guessan

Je suis un développeur hobbyiste depuis 2011. Autrement dit, le développement est une passion dans laquelle j'ai envisagé un business plan afin de dégager des fonds. Je développe exclusivement pour Windows Phone (7.x, 8.x, et 10). Je tiens à préciser qu'initialement j'ai suivi une formation en électronique et que le développement d'application mobile est uniquement une envie personnelle provoquée par un attachement aux produits Nokia/Microsoft et un besoin de continuer une autoformation débutée avec des cours du soir.

L'apprentissage de la plateforme

L'apprentissage de la plateforme est un combat quotidien et difficile. La documentation de Microsoft est trop succincte dans certains cas. Nonobstant ces lacunes, il était important d'intégrer une communauté de développeurs afin de pouvoir poser des questions et surtout de profiter des astuces de chacun. Cela m'a par ailleurs donné l'envie de partager également mon expérience via un blog. La plateforme Windows est intéressante car jeune. Il est donc très facile de se faire une place dans un domaine. J'avoue avoir été séduit par Visual Studio le duo de langages C#/XAML.

L'investissement

Durant ces 4 dernières années afin de développer dans de bonnes conditions, j'ai investi dans un smartphone (Nokia Lumia 820 dès sa sortie) et une Surface Pro 3. Pour le reste j'ai surtout « profité » des prêts à durée déterminée de matériel par Nokia (avant son rachat) et du programme MSTechRewards (anciennement DVLUP) afin de pouvoir étoffer ma gamme d'appareils de test.
Malheureusement ce programme a été lâché et avec Windows 10 Mobile qui pointe le bout de son nez, il est difficile de « se mettre » à la page sans utiliser son argent personnel qui n'est en aucun cas un bénéfice d'application. J'ai également payé 3 fois 60 € /an de licence de développement Windows avant que cela soit à vie.

Les revenus

Pour la majorité de mes applications, j'ai choisi d'opter pour le modèle suivant : gratuit, avec publicité, et possibilité de la retirer avec un achat de l'application. J'utilise naturellement (à tort ?) la régie Pubcenter de Microsoft. Les revenus sont très faibles pour ne pas dire nuls pour chaque application ; en résumé sur Windows, il faut multiplier les concepts inédits si l'on souhaite dégager des bénéfices. En ce qui me concerne, je ne souhaitais absolument pas vivre du développement, mais je voyais ceci plutôt comme un complément de salaire (une sorte de beurre dans les épinards).

Temps de développement

Il est difficile de trouver du temps quand on ne peut réaliser que ça le soir ou le weekend quand on est père de famille. Cela ralentit fatalement le délai de sortie de corrections d'une application et donc forcément le temps investi. Ceci étant dit, développer tout en ayant de telles contraintes reste tout à fait possible. En effet ma vie personnelle a orienté mes choix de thèmes d'applications. Choix qui sont encore « payants » aujourd'hui.

Le marketing

Sans le marketing, votre application n'est pas connue. Il est donc important de ne pas négliger cet aspect. Microsoft avait un programme « Accélérateur » qui permettait la mise en avant dans les stores et un placement intéressant dans certaines collections. Aussi ne faut-il pas négliger les blogs spécialisés. Ces derniers (surtout ceux qui sont francophones) mettent avec plaisir en avant vos œuvres ne moyennant aucune finance. Et puis il y a également des régies publicitaires moyennant investissement. Mais cela s'adresse plus aux grandes entreprises ou développeurs qui ont le budget adéquat

Quid des autres plateformes ?

J'ai tenté l'expérience Android que ce soit avec Android Studio ou Eclipse mais il est clair que le confort de Visual Studio est unique. Etant très attaché au C#, j'ai tenté l'expérience Xamarin en portant certaines de mes applications sur le Google Play Store. J'en ai un avis mitigé. En effet, Xamarin sur le papier permet de « porter « des applications sur d'autres supports via un langage unique C# ; la réalité en ce qui me concerne en est toute autre. La documentation est perfectible et l'intégration de service Google nécessite une ENORME patience. Xamarin coûte près de 1000€/an. Et il aurait fallu investir dans des smartphones/tablettes Android. Quant à iOS, le débat fut vite clos : je ne possède pas de Mac.
En résumé je n'ai pas cédé aux sirènes d'Android et iOS par faute de moyens (venant d'investir dans une Surface Pro, je ne pouvais me permettre d'acheter un iMac et un iPhone !) ou d'attrait. Enfin ayant mis 3 / 4 ans à bien maîtriser la technologie Microsoft (de Silverlight à Universal Windows Program), je ne me voyais pas repartir de zéro pour les autres plateformes. Au-delà du développement mobile, le langage Java ne m'a jamais séduit. Et XCODE/SWIFT... je n'ai pas de Mac donc tout est écrit. Néanmoins je projette de me lancer dans le développement via Unity et donc de quoi toucher toutes les plateformes mobiles.

Conclusion

En résumé, développer dans les applications mobiles demande du temps, de l'investissement, et, si l'on souhaite tout faire seul, il faut être conscient que plusieurs clés de la réussite ne sont pas à négliger : un bon concept et un bon « business model ». Le développement mobile peut être rentable si l'on se fixe des limites. En ce qui me concerne je suis assez satisfait de mon plan produit. En effet j'ai peu investi financièrement dans cette plateforme. Avec toutes les aides reçues de la part de Microsoft et le fait de m'être uniquement concentré sur Windows (Phone), je peux dire que mon investissement dans le développement mobile a été rentabilisé au fil du temps.

Quelle approche pour mon application multiplateforme mobile ?

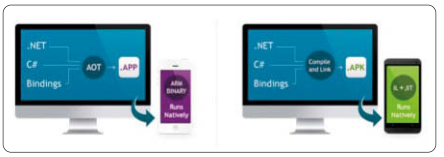
Disposer d'applications mobiles relève de l'évidence même pour la plupart des entreprises. En plus de répondre à des besoins elles sont aussi des vecteurs de communication véhiculant l'image de marque de votre société. On aurait par ailleurs tort de sous estimer l'impact désastreux qu'une mauvaise application peut avoir pour la société la distribuant.

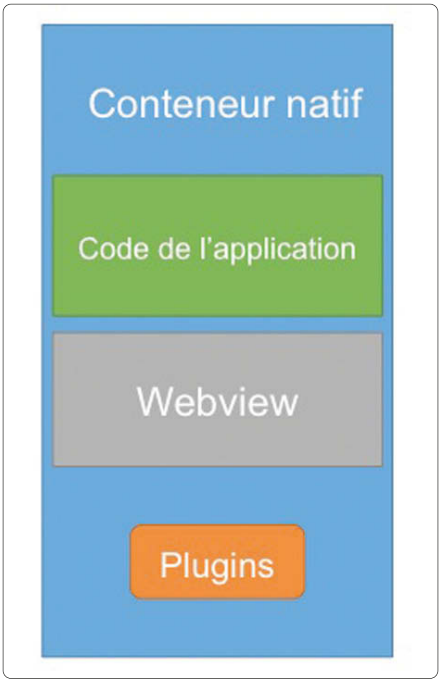
John Thiriet
Consultant / Formateur à **Cellenza**
@johnthiriet
Microsoft Windows Development MVP
Xamarin MVP

Nous sommes dans un monde dans lequel les utilisateurs ont des attentes supérieures en termes d'ergonomie et de facilité d'utilisation. Le simple service rendu n'est bien souvent plus un objectif en soi mais plutôt le strict minimum à fournir. Une application mobile efficace est désormais un critère dans le choix d'une banque pour une partie des utilisateurs.
Si les utilisateurs ne sont pas satisfaits d'une application, ils peuvent la désinstaller et utiliser une application concurrente. Pire encore, ils peuvent laisser des commentaires et des notes négatives qui ôteront l'envie aux autres utilisateurs d'installer cette application. Ainsi, si elle ne répond pas à des critères de qualité élevée tant en termes de design que de fiabilité, elle risque d'être condamnée à court terme.
Il existe une grande variété de terminaux et de systèmes d'exploitation permettant de les utiliser. Certains ont de grands écrans, des claviers physiques ou encore des appareils photo digne du milieu de gamme des appareils photo numériques.
Pour le développeur, l'équation est difficile à résoudre : comment répondre à ces développements avec les outils les mieux adaptés ? Quel modèle de développement adopter ? Et surtout, quelles solutions utiliser ?

Le développement natif

Le développement natif est la solution proposée et supportée par chaque plateforme. Il impose un développement spécifique sur chacune d'elles. Ce faisant, il est un gage de performance et de qualité par rapport aux approches hybrides, à défaut de permettre un partage de code entre toutes les plateformes ciblées. Une application est dite native lorsqu'elle utilise les langages, les API et les outils proposés par les éditeurs de la plateforme ciblée.
Pour créer une application iOS il faut utiliser





XCode et développer en Objective-C ou en Swift sur OS X. On y utilise des "Storyboard" ou des "Xib" pour définir les interfaces graphiques.
Une application Android se développe avec Android Studio (ou autre IDE équivalent) en Java sur n'importe quel système d'exploitation. On utilise des "Activity" et des "Fragment" couplés à des fichiers AXML pour définir des interfaces graphiques.
En ce qui concerne Windows on travaille sur Windows 10 avec Visual Studio 2015 en C# ou VB.NET, et on utilise des "Page" ou "Window" ainsi que des "UserControl" dans des fichiers XAML pour définir l'interface graphique. Sous Windows 10 il est également possible d'utiliser de manière native un développement basé sur HTML/CSS et Javascript.
Premier constat : chaque développement natif utilise des langages et outils propres à chaque plateforme.
Outre ces différences de langages ou de formats graphiques les approches même de développement varient. Sur iOS et Android on retrouve des variantes du modèle MVC alors que sur Windows on utilise plus naturellement le modèle MVVM.
Faire du développement natif demande donc un bon niveau d'expertise sur chacune des plateformes que l'on souhaite cibler.

Pourquoi faire du natif ?

Le choix d'un développement natif doit être étudié de près, car, par essence à chaque plateforme, on va ajouter une nouvelle base de code.
Il faut déjà savoir si être présent sur plusieurs plateformes est intéressant, et indispensable pour le projet. En effet, si la cible d'une application se situe sur une seule plateforme il n'y a aucun intérêt à en cibler une autre. C'est d'ailleurs le cas pour toutes les entreprises qui maîtrisent leurs parcs mobiles et qui souhaitent développer une application interne. Si une entreprise ne fournit à ses employés que des terminaux Windows, il n'y a aucun intérêt à développer une version iOS et Android. De plus, on bénéficie dans cette situation d'un support de l'éditeur de la plateforme qui peut être rassurant pour une entreprise.
Cette approche du développement mobile est connue et capitalise sur une expérience de plusieurs années qui permet de mettre en route rapidement de nouveaux projets. De plus, les compétences natives, qu'elles soient iOS, Android ou Windows sont relativement faciles à trouver sur le marché du travail. Il est également aisé de trouver des prestataires capables de réaliser ces applications en forfait ou en régie.
Grâce au développement natif on peut aussi adapter facilement notre application en fonction des utilisateurs de chaque plateforme car les possesseurs de terminaux iOS ou Android ne sont pas les mêmes.
iOS est souvent considéré comme un produit haut de gamme. On a d'ailleurs constaté que les utilisateurs de cette plateforme achètent plus facilement des applications ou utilisent les achats intégrés aux applications.
Sur Android, on a, au contraire, constaté que les utilisateurs sont très réticents à acheter des

applications. Le modèle de monétisation est ici plutôt centré autour de la publicité ou de l'abonnement à des services en ligne. De part le faible prix de certains appareils, c'est aussi un bon point d'entrée vers les adolescents des pays développés et des pays émergents.
Le dernier argument pouvant faire pencher la balance en faveur du développement natif est la performance. Il est indispensable d'avoir une application réactive lorsque l'on sait qu'environ un tiers des applications sont utilisées moins de 60 secondes. Cela ne laisse pas beaucoup de temps pour fidéliser les utilisateurs, il faut donc que l'impression qu'elle laisse soit impeccable durant ce court laps de temps.
Avec le développement natif, on est en mesure d'accéder à toutes les ressources et toutes les APIs offertes par la plateforme. Grâce à cela, il est possible d'offrir aux utilisateurs les meilleures performances et la meilleure réactivité possible.

Développement hybride

Le multiplateforme hybride est une des premières réponses qui ait été apportée à cette problématique de développement multiplateforme. Le principe est d'opter pour des technologies déjà connues pour fonctionner sur de multiples plateformes, les technologies Web.
Une application hybride est donc une application basée sur HTML, CSS et Javascript. Chaque plateforme native fournit un composant permettant d'afficher un navigateur Web appelé Webview. Cette Webview est hébergée dans une application native appelée conteneur. L'application en elle-même s'exécute au sein de la Webview et le conteneur natif permet à cette dernière d'effectuer des appels aux API natives. Le terme hybride est donc utilisé car l'application n'est ni complètement native, ni purement basée sur des technologies Web.
L'inconvénient principal de cette approche réside dans le concept même d'une application hybride. Comme l'intégralité de l'affichage et du code métier est exécutée par une Webview, l'application aura des performances inférieures à son pendant natif. De plus certaines fonctionnalités leur seront inaccessibles comme l'accès à la carte graphique. Une application hybride n'hérite pas non plus du style graphique du système. En effet, une application hybride n'utilise pas de composants visuels natifs et doit donc gérer son propre affichage. Il existe de nombreux frameworks graphiques tels que Ionic rendant les applications hybrides plus attrayantes pour l'œil, mais elle n'en reste pas moins une application hybride par défaut non intégrée au système.
Choisir le développement hybride c'est donc choisir de développer une application non-native et qui peut être ressentie comme telle par les utilisateurs.

Pourquoi choisir l'approche hybride ?

En général l'approche hybride est plébiscitée grâce à son coût d'entrée faible et une grande rationalisation des développements. Avec les approches hybrides, on arrive très rapidement à des applications certes non-natives, mais fonctionnelles, et ce sur un très grand nombre de plateformes. A la différence des approches natives, le rendu d'une application hybride sera quasiment identique sur toutes les plateformes et ne dépendra finalement que du navigateur utilisé. Les mêmes problèmes et avantages que le développement Web se retrouvent ainsi donc transposés dans les applications mobiles. De plus la majeure partie des frameworks Javascript existants sont utilisables directement dans une application hybride, ce qui représente un gain non négligeable de productivité.
De plus, le fait de ne dépendre que des technologies Web permet de rationaliser les coûts également en termes d'effectifs. Seuls des développeurs HTML/CSS/Javascript sont nécessaires à la différence des approches natives qui nécessitent des compétences sur chaque plateforme ciblée. Il est très facile à un développeur Web de devenir efficace avec les approches hybrides. La courbe d'apprentissage est relativement faible et il faudra juste un peu de temps pour maîtriser les API spécifiques servant à interagir avec la plateforme.

Cordova

Le framework principal permettant le développement d'applications hybrides est Cordova. Cordova est historiquement connu sous le nom de PhoneGap, lequel avait été racheté par Adobe à son créateur puis libéré en Open Source sous son nom actuel. Adobe se concentre maintenant à la création de composants et d'offres payantes autour de Cordova. Ces derniers temps, le développement d'applications Cordova est devenu plus facile et est pris en charge nativement dans de nombreux IDE comme Visual Studio 2015.
Cordova fonctionne avec une architecture modulaire à base de plugins. Chaque plugin permet aux applications Cordova d'accéder à de nouvelles fonctionnalités. Ce sont ces derniers qui font le pont entre Javascript et une implémentation native spécifique à chaque plateforme. Pour qu'un plugin fonctionne sur trois plateformes, il faut donc qu'il existe trois implémentations natives de ce dernier en plus de l'interface Javascript qui le définit.
Cordova met à disposition de nombreux plugins de base, mais il faudra bien faire attention au tableau de compatibilité de ces plugins. Ce qu'il faut retenir c'est que plus une application doit fonctionner sur de nombreuses plateformes, plus il y a de chances qu'un plugin ne soit pas compatible avec une des plateformes ciblée.
Cordova n'est qu'un framework d'applications hybrides. Il ne contient rien par défaut permettant de développer une interface graphique complète ou bien de frameworks structurant le code de l'application. A ce titre il est de bon ton de lui adjoindre divers frameworks comme Ionic pour la partie graphique et Angular, ReactJS ou Aurelia pour la partie architecture. Attention cependant car Ionic est lui-même construit sur Angular. Ionic met à disposition de nombreux contrôles graphiques simulant une ergonomie native. Ils ne sont pas complètement intégrés à la plateforme mais atténuent néanmoins la différence entre natif et hybride. Dans Ionic 2, il existe deux CSS séparés pour les thèmes graphiques iOS et Material Design d'Android. Cela permet là aussi de simuler une bien meilleure intégration au système.

Les points clefs

Le développement hybride permet de fortement rationaliser les coûts de développement de la couche métier mais aussi et surtout de la couche graphique. En utilisant en complément les framework appropriés tels que Ionic, on s'approche d'une interface intégrée au système sans toutefois en avoir toutes les performances. Cette interface partagée entre toutes les plateformes peut être vécue comme un avantage ou un inconvénient en fonction du contexte.
Ces défauts en termes de performance et de non-intégration au système sont souvent sanctionnés par les utilisateurs sur les magasins d'applications, mais ne sont pas critiques dans le cas d'applications internes ou orientées business.
Les différences de rendu entre les navigateurs peuvent rendre le test des applications complexes si elles ciblent de nombreuses plateformes ou versions du système. Il existe en effet différentes versions des Webview en fonction des versions d'iOS ou d'Android. De manière générale, plus on ciblera de terminaux,

plus le temps de test alloué aux applications devra être important afin de s'assurer d'avoir un rendu correct sur tous les navigateurs embarqués.
L'utilisation des technologies Web induit une montée en compétence rapide des équipes et on peut s'attendre à avoir rapidement des applications fonctionnelles sur de multiples plateformes avec une approche hybride.
L'approche hybride n'est donc pas une solution miracle, très intéressante sur un plan financier à court terme, elle induit de plus faibles performances et des fonctionnalités limitées graphiquement à ce qu'un navigateur est capable de faire. Le coût à long terme des applications hybrides dépend quant à lui essentiellement de la maturité des équipes. En effet, les technologies Web et les frameworks évoluent très rapidement, beaucoup plus que les technologies natives. Une application hybride acquiert donc rapidement de la dette technique et nécessitera un entretien régulier du code pour que le gain financier initial ne se perde pas complètement.

Le multiplateforme natif

Lorsque l'on parle de développement multiplateforme, on pense tout d'abord aux solutions basées sur des technologies Web comme Cordova.
Ces solutions permettent de rationaliser les coûts de développement en partageant un maximum de code, qu'il soit graphique ou métier, au prix d'un sacrifice de l'expérience utilisateur et de performances moindres que les approches natives.
Partant de ce constat, certains ont réfléchi au meilleur moyen de partager du code entre toutes les plateformes tout en proposant des interfaces natives aux utilisateurs. Le concept général de ces solutions, est d'utiliser un seul et même langage de programmation pour toutes les plateformes, tout en utilisant les API natives à chacune de ces plateformes.

Xamarin

Xamarin est probablement la solution la plus populaire entrant dans cette catégorie. Xamarin permet le développement d'applications natives en C# ou en F#. Les applications développées avec Xamarin bénéficient de performances quasi-identiques aux applications natives et d'une interface graphique riche et intégrée au système. Ce framework permet d'accéder à 100% des APIs natives offertes par chaque plateforme. L'utilisation d'un langage unique, connu, riche, performant et maîtrisé comme C# permet à l'approche Xamarin de réduire fortement les coûts de développement et d'améliorer maintenabilité et taux de réutilisabilité du code. En résumé, Xamarin propose tout simplement de conjuguer les avantages du natif et de l'hybride.

Deux approches

Le développement d'applications Xamarin peut se faire de deux manières différentes, l'approche classique ou Xamarin Forms.
L'approche classique permet de développer une application iOS et Android native en C#. Dans cette approche on utilise les fichiers graphiques natifs de la plateforme (Storyboard, Xib ou Axml), les mêmes API ainsi que les mêmes paradigmes de développement. En utilisant cette approche on est donc bien en train de développer une application Android ou iOS native. De cette manière on partage la logique métier entre les différentes plateformes tout en proposant aux utilisateurs de l'application une expérience 100 % native. On estime généralement pouvoir arriver à un partage de code de l'ordre de 70 à 85 % entre chaque plateforme. Certaines applications ne nécessitent pas une expérience personnalisée sur chaque plateforme. L'idée derrière Xamarin Forms est relativement simple : proposer une API graphique indépendante de la plateforme et la transposer nativement sur chaque plateforme. Par ce fonctionnement, Xamarin Forms permet de partager près de 90 à 98 %. Ici, on parle bien de code d'interface partagée et non de l'interface elle-même. Dans une application Xamarin Forms, lorsque l'on affiche un bouton, c'est un bouton natif iOS, Android ou Windows en fonction de la plateforme.
Par ailleurs, Xamarin Forms utilise une version de XAML pour définir son interface graphique ce qui le rend attractif et facile d'accès pour les développeurs issus de l'écosystème Microsoft étant déjà familiers avec XAML.

Pourquoi choisir Xamarin ?

Il existe de nombreuses raisons pouvant mener au choix de Xamarin :

- Vous avez des équipes de développement C#/XAML productives avec Visual Studio et vous devez développer des applications sur iOS et/ou Android ;
- Votre entreprise change régulièrement de fournisseur de terminaux ou bien fonctionne en BYOD (Bring Your Own Device) et vous ne pouvez donc pas limiter votre développement à une plateforme en toute sérénité ;
- Vous souhaitez maximiser le partage de code tout en proposant à vos utilisateurs une expérience 100% native ;
- Vous avez une application Windows que vous souhaitez porter sur iOS, Android ou OS X ;

Si vous pensez que Xamarin est le bon choix il vous reste maintenant à choisir entre l'approche classique ou Xamarin Forms.

Approche classique ou Xamarin Forms

Dans le cadre d'applications B2B, on cherche généralement à rationaliser au maximum les coûts de développement à niveau de fonctionnalité égal par rapport à leurs pendants B2C.
Xamarin Forms, de par son orientation formulaire et son code graphique mutualisé est donc tout à fait adapté aux applications B2B. Les applications B2C elles répondent à une autre logique. De par le niveau d'exigence du public il faut privilégier une approche native du développement afin de répondre à leurs attentes en termes d'ergonomie et de performance. En procédant de cette manière, on augmente significativement les chances de récolter de bons retours utilisateurs et donc de démarrer un cercle vertueux d'augmentation des téléchargements et des commentaires positifs. Pour ce type d'applications, l'approche classique de Xamarin est tout à fait indiquée.

Combien ça coûte ?

Xamarin est une solution commerciale qui est facturée par plateforme et par développeur.
Il existe trois licences possibles pour Xamarin dont voici les caractéristiques principales :

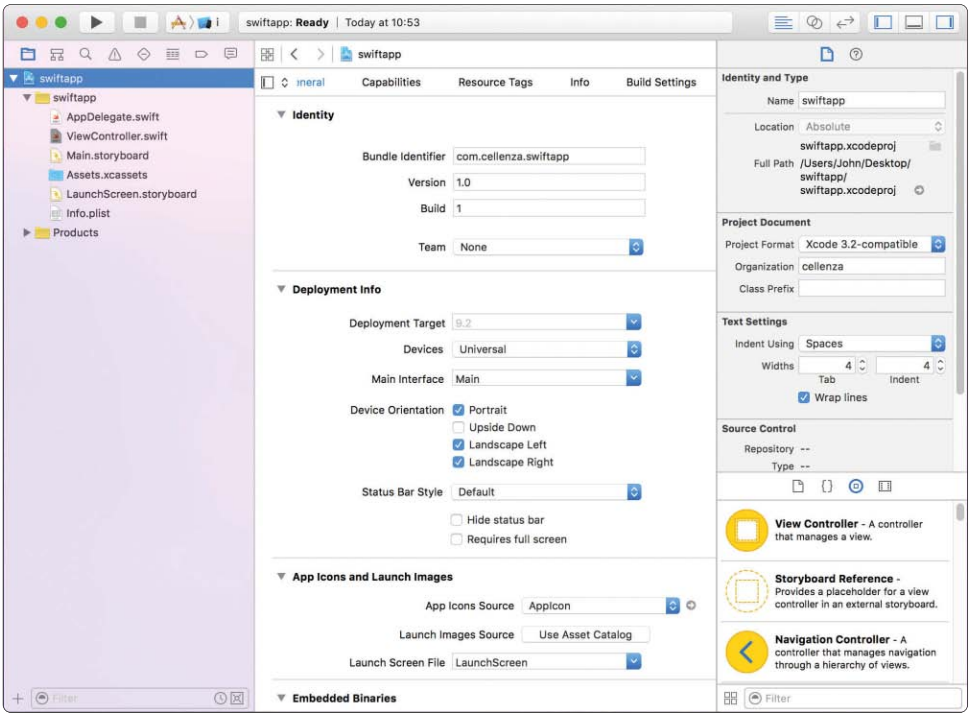
	Indie	Business	Entreprise
Coût	25 $ par mois	999 $ par an	1899 $ par an
Avantage principal	Coût le plus faible	Support de Visual Studio	Support technique avancé, techniques de compilation avancées
Inconvénient principal	Pas de support de Visual Studio	Prix élevé pour un indépendant	Prix élevé

- La licence Indie permet aux développeurs indépendants ou aux entreprises de moins de 5 employés d'utiliser Xamarin à moindre frais.
- La licence Business proposée est la licence qui répond à la majorité des besoins rencontrés. C'est vers cette licence qu'il faut se tourner au minimum en entreprise.
- La licence Entreprise offre de nombreux avantages en termes de support technique de la part de Xamarin ou l'ajout d'une fonctionnalité de compilation spéciale utile pour bien protéger son code et augmenter les performances.

Prenons l'exemple d'un développeur développant depuis Visual Studio pour Android et iOS. Il aura besoin de deux licences

Business ce qui reviendra environ à 2000 $. Ce coût est à remettre en perspective avec le coût par jour d'un développeur mobile. On estime généralement que le coût de Xamarin correspond de 3 à 5 jours du coût d'un développeur mobile.
Outre le coût en termes de licence vient le coût en termes de formation des équipes de développement qui vont varier en fonction des profils. Le tableau ci-dessous donne un ordre d'idée des compétences à acquérir et de l'effort nécessaire à fournir afin de devenir un développeur efficace avec Xamarin.

Expérience du développeur	Effort de formation	Type d'apprentissage
iOS	Moyen	Apprendre C#
Android	Simple	Apprendre C#
Windows Phone / Windows 8+	Simple	Apprendre iOS / Android
ASP.NET	Complexe	Apprendre le développement mobile



Bien qu'il soit possible de s'auto-former, devant une technologie comme Xamarin, il peut être intéressant d'opter pour des formations. Plusieurs sociétés proposent de telles formations en langue française. Il est aussi possible d'utiliser Xamarin University qui propose d'excellents cours en ligne en anglais afin de monter en compétence.
Le coût d'une telle formation va bien entendu dépendre du développeur et de la formation choisie. Xamarin University est proposée au tarif de 1995 dollars par an, à vous ensuite de voir ce qui correspond le mieux à votre besoin.

Les points clefs

Le développement multiplateforme natif comme le propose Xamarin est en quelque sorte, sur un plan technique, le meilleur de tous les mondes. L'application est native, ergonomique, performante et intégrée à la plateforme. Dans le même temps, son code est partagé, ce qui permet de s'assurer qu'une correction disponible pour une plateforme le sera pour toutes les autres. Il permet également de proposer des déploiements presque simultanés sur tous les magasins d'applications, ce qui s'avère très pratique dans le cadre de corrections urgentes ou de campagnes marketing. L'investissement initial en licence et formation est important mais il sera rentabilisé à court ou moyen terme une fois les développeurs formés et efficaces avec la technologie.

Les critères	Natif	Cordova	Xamarin
Coût immédiat	Moyen	Faible	Moyen à élevé
Coût à long terme	Elevé	Elevé	Moyen
Facilité à trouver des profiles formés	Facile	Moyen	Difficile
Temps de formation	Moyen	Faible	Moyen à élevé selon profil
Partage d'interface graphique	Non	Oui	Code partagé avec Xamarin Forms
Partage de code métier	Non	Oui	Oui
Accès aux API du système	Oui	En partie via des plugins	Oui
Taux de partage de code	N/A	Elevé	Elevé
Performance	Elevé	Faible à moyenne	Elevé
Tests unitaires	Non mutualisés	Mutualisés	Mutualisés
Ecosystème	Spécifique par plateforme	Html/CSS/Javascript	.NET, C#, F#
Particulièrement approprié pour	Projets très graphiques à haute performance en développement au forfait.	Projets internes ou B2B de type formulaires avec un faible besoin de performance	Projets avec fort taux de partage de code sans compromission sur les niveaux de performances
Atouts principaux	Grande performance et support des éditeurs	Faible coût de mise en œuvre et partage de rendu graphique	Taux de partage de code élevé et performances natives
Frein principal	Autant de bases de code que de plateformes	Performances dégradées	Coût des licences

Conclusion

Le marché mobile, à la différence du marché des ordinateurs de bureau, semble parti pour rester fragmenté entre plusieurs systèmes d'exploitation, langages et constructeurs.
Nous avons vus les intérêts et inconvénients principaux de chaque de approche de développement, qu'ils soient techniques et/ou économiques.
Performance, compétences techniques, taux de partage de code, rapidité des développements, facilité de la maintenance et techniques de contrôle de code, sont les critères principaux auxquels vous allez devoir penser pour faire votre choix. Rappelez-vous cependant qu'aucune des approches exposées dans cet article ne vous permettra de remplir tous les critères ci-dessus.
A chaque approche sa force, la votre est celle de faire le bon choix !
Pour finir cet article et vous aider dans votre décision je vous propose ce dernier tableau. Il récapitule succinctement les axes principaux menant au choix d'une approche de développement multiplateforme particulière évoqués tout au long de cet article, sans toutefois se vouloir exhaustif de tous les points abordés.

Les 2 commandements du développeur : faire un code propre et commenter son code

Il y a des fondamentaux que tout développeur ne devrait jamais oublier. Un code propre facilite sa lecture mais aussi son évolution et sa maintenance. Un débat existe sur l'utilité de commenter son code. Pour nous, c'est indispensable mais pas question de commenter chaque ligne... Un petit dossier pour comprendre ces 2 commandements, avec de bonnes pratiques et des outils...

La rédaction.

Un code propre est indispensable

Vous vous êtes déjà retrouvés devant un code illisible ? Un bug que vous devez corriger, ou une fonctionnalité à rajouter, et rien que la lecture du code vous pique les yeux ?
C'est la dure loi de l'écriture de logiciel. Le temps passe, des fonctionnalités sont ajoutées, des changements sont faits, des corrections par-ci par-là... Les développeurs ont changé d'équipe ou de projet et la moindre modification de code devient de plus en plus risquée et de plus en plus complexe. C'est bien connu, la productivité diminue au fur et à mesure que le temps passe.

Nicolas Hilaire
Expert.NET - Neotech Solutions

Martin Fowler disait que n'importe qui peut écrire du code qu'un ordinateur comprend. Les bons développeurs écrivent du code que les humains peuvent comprendre.

Vous devez donc écrire votre code comme un auteur de livres, faire en sorte qu'il raconte une histoire claire, qu'il soit fluide et auto-descriptif. Il faut qu'un futur relecteur comprenne très rapidement l'intention de votre code et ainsi lui faciliter sa maintenance ou ses évolutions.
Je tiens à vous dire tout de suite qu'écrire un code propre est compliqué. Cela demande peu de connaissances, mais beaucoup d'entraînement. Mais si vous n'avez pas encore tourné la page, c'est que vous avez envie de vous améliorer et c'est un très bon point. Alors, comment améliorer la lisibilité de son code ? Vous trouverez dans cet article quelques bons principes éprouvés qui pourront faire de vous un meilleur auteur de code.
Pour la suite des exemples, j'utiliserai le C#, langage que j'apprécie particulièrement, mais les conseils valent pour n'importe quel langage.

Un nommage significatif :

Exprimer l'intention

Cela paraît évident, mais utilisez des noms qui expriment l'intention, qu'il s'agisse de variables, de méthodes ou de classes.
Pourquoi écrire le code suivant :

```
string adr1; // adresse d'expédition
string adr2; // adresse de facturation
```

Alors que nous pourrions directement écrire :

```
string adresseExpedition;
string adresseFacturation;
```

Ainsi, toute utilisation future d'une de ces deux variables indiquera bien avec quelle adresse nous travaillons. C'est cela, exprimer l'intention de la variable. Et c'est parfois très complexe.

Ne pas utiliser de nombre ou chaîne magique

De la même façon, si je tombe sur la méthode suivante, je ne suis pas sûr de comprendre vraiment ce qu'elle fait :

```
public bool Tirage(int[] tab1, string[] tab2)
{
   int i = new Random().Next(53);
   if (tab1[i] == 0 && tab2[i] == "R")
      return true;
   return false;
}
```

On se doute bien qu'on cherche un truc aléatoirement dans des tableaux ... mais quoi ? C'est quoi ce 53 ? Et ce "R" ?
N'est-ce pas plus clair avec un nommage plus adapté de ce genre ?

```
public bool EstDameDeCoeur(int[] couleurs, string[] hauteurs)
{
   const int NombreMaxDeCartes = 52;
   const int Coeur = 0, Pique = 1; // ...
   const string Reine = "R";

   int tirageAleatoire = new Random().Next(NombreMaxDeCartes + 1);
   if (couleurs[tirageAleatoire] == Coeur && hauteurs[tirageAleatoire] == Reine)
      return true;
   return false;
}
```

Les nombres magiques dans le code sont une vraie plaie. C'est une très bonne pratique de les remplacer par des constantes aux noms bien explicites. Bien sûr, on pourrait rajouter un commentaire à côté, mais c'est tellement plus clair quand la variable est bien nommée.

Évitez les noms fourre-tout

Soignez le nom de vos classes. Évitez les noms fourre-tout comme Mana-

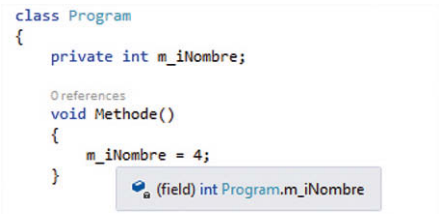
ger, Processor, Provider, Service, etc. Oui - je sais - vous l'avez déjà fait ! Une classe doit être un nom, et sûrement pas un verbe.
Vous gagnerez aussi à faire en sorte que vos noms puissent se chercher facilement dans le code avec un Control+F. Les éditeurs sont de plus en plus habiles pour naviguer dans le code, mais une bonne vieille recherche textuelle peut parfois vous sauver la vie.

Pas de notation hongroise

Par pitié, enlevez-moi tous les préfixes de la notation hongroise https://fr.wikipedia.org/wiki/Notation_hongroise qui ne servent à rien quand on a un IDE digne de ce nom !

```
int m_iNombre = 4;
```

C'est quand même évident que c'est un entier. Ça avait peut-être un sens en 1960 ... mais là, on est en 2016 ! Et puis en cas de doute, il y a toujours l'info-bulle pour nous rappeler de quel type est la variable :



Remplacer les commentaires par des méthodes bien nommées

Avec un bon nommage de vos méthodes, vous pouvez aussi remplacer des potentiels commentaires par des noms de méthodes explicites, voyez par exemple :

```
int i = int.Parse(Console.ReadLine());
if (i % 2 == 0) // si multiple de 2
   Console.WriteLine("{0} est pair", i);
if (i % 5 == 0) // si multiple de 5
   Console.WriteLine("{0} est un multiple de 5", i);
```

Il suffit d'extraire les conditions des tests et d'utiliser le commentaire pour nommer la méthode :

```
static void Main(string[] args)
{
   int i = int.Parse(Console.ReadLine());
   if (EstPair(i))
      Console.WriteLine("{0} est pair", i);
   if (EstMultipleDe5(i))
      Console.WriteLine("{0} est un multiple de 5", i);
}

private static bool EstPair(int i)
{
   return i % 2 == 0;
}

private static bool EstMultipleDe5(int i)
{
   return i % 5 == 0;
}
```

C'est tout de même beaucoup plus lisible, non ?

Découpez votre code

Des méthodes ou des classes doivent bien sûr être correctement nommées, mais elles ne doivent pas contenir trop de code. Quand une méthode est trop longue, il faut la découper en plusieurs méthodes. C'est un peu ce que nous avons fait juste au dessus.
Idéalement, chaque méthode ne doit faire qu'une seule chose (et doit le faire bien !). Par exemple vous pouvez changer un code de ce genre :

```
private bool VerifieUtilisateur(string nom, int age)
{
   if (string.IsNullOrWhiteSpace(nom))
      return false;
   if (nom.Length > 20)
      return false;

   if (age < 18)
      return false;

   return true;
}
```

par :

```
private bool VerifieUtilisateur(string nom, int age)
{
   if (!VerifieNom(nom))
      return false;

   if (!VerifieAge(age))
      return false;
   return true;
}
```

Bien découper une grosse méthode en plusieurs petites permet également de mieux exprimer l'intention de votre code en faisant apparaître un algorithme explicite :

```
private bool ValideLePanier()
{
   if (!VerifieStockProduits())
      return false;
   if (ADesProduitsEnReductions())
   {
      if (VerifieReductions())
         montant = CalculeLeNouveauMontant();
   }
   if (EstJeudi())
      DoublePointsFidelites();

   return ReserveStock();
}
```

Cependant, lorsque vous découpez vos méthodes (ou que vous en créez), prenez garde aux paramètres. Un relecteur de code va s'attendre à ce que les paramètres d'une méthode soient des paramètres d'entrée. Les paramètres de sortie sont plus difficiles à appréhender. Si vous devez modifier

quelque chose dans la méthode, faites en sorte de renvoyer une valeur.
De même, n'utilisez pas de booléen en paramètre d'une méthode. Ils sous-entendent que la méthode fait 2 choses différentes ; une quand le booléen vaut vrai, une autre chose quand il est égal à faux.
Plutôt que :

```
private bool Faire(bool flag)
{
   if (flag)
   {
      // fait ceci
   }
   else
   {
      // fait cela
   }
}
```

Faites deux méthodes FaitCeci() et FaitCela() et utilisez les indépendamment. Et puis que diriez-vous d'une méthode comme celle-ci, que l'on retrouve dans beaucoup de frameworks ?

```
CreateFile(path, true, false, true);
```

Alors elle fait quoi ? Un mode append, un mode overwrite ? Les deux ? Encodage par défaut ? Bref, on en sait rien tant qu'on ne va pas voir la documentation !

Vos classes aussi ne doivent faire qu'une seule chose, et pour cela n'hésitez pas à suivre le principe de responsabilité unique (https://fr.wikipedia.org/wiki/Principe_de_responsabilit%C3%A9_unique) que l'on retrouve dans les principes SOLID (https://fr.wikipedia.org/wiki/SOLID_(informatique)).

Ce principe de responsabilité unique vient souvent avec un autre principe, à savoir qu'une classe doit être ouverte à l'extension et fermée à la modification (https://fr.wikipedia.org/wiki/Principe_ouvert/ferm%C3%A9).

Et évidemment, factoriser le code qui est identique. Suivez le principe DRY https://fr.wikipedia.org/wiki/Ne_vous_r%C3%A9p%C3%A9tez_pas.
Oui, - je sais - comme vous, ça m'est déjà arrivé de faire un copier-coller plutôt qu'un refactoring adéquat. Honte à moi.

Bien commenter

Un commentaire utile est un commentaire absent. Si le code est auto-descriptif, alors il n'y a pas besoin de commentaire. Quand vous êtes tentés d'écrire un commentaire pour clarifier votre code, voyez plutôt s'il n'aurait pas besoin d'un petit refactoring pour le rendre plus lisible.
Comme on l'a déjà vu, qu'est-ce qui est le plus clair ?

```
if (i % 2 == 0) // si multiple de 2
   Console.WriteLine("{0} est pair", i);
```

ou

```
if (EstPair(i))
   Console.WriteLine("{0} est pair", i);
```

Bien sûr, bannissez les commentaires inutiles et redondants :

```
/// <summary>
/// Classe représentant un client
/// </summary>
public class Client
{
   /// <summary>
   /// Nom du client
   /// </summary>
   public string Nom { get; set; }
   /// <summary>
   /// Age du client
   /// </summary>
   public int Age { get; set; }

   /// <summary>
   /// Constructeur par défaut
   /// </summary>
   public Client()
   {
      // ...
   }
}
```

La classe et ses propriétés sont suffisamment bien nommées pour ne pas avoir à répéter un commentaire redondant qui vient polluer le code et sa lisibilité. Et ne me faites pas croire que vous n'avez jamais écrit un commentaire de ce genre, je ne vous croirais pas :).
Je ne dis bien sûr pas qu'il ne faut pas commenter son code, mais soyez certains que votre commentaire apporte quelque chose et qu'il clarifie bien votre intention ou pourquoi vous avez choisi telle solution plutôt qu'une autre. Par exemple :

```
private long Factorielle(int nombre)
{
   // J'ai choisi d'implémenter la factorielle avec une boucle plutôt qu'une méthode récursive
   // car cela posait des problèmes de performance quand le nombre était trop élevé
   ...
}
```

Enfin, et cela sera mon dernier mot sur les commentaires : ne laissez pas du code commenté. Non, ne le faites pas, c'est mal ! Si je viens à travailler sur le même code que vous, je vais me demander pourquoi le code est commenté. Est-ce que je dois le supprimer ? Mais s'il n'est pas supprimé, c'est que le code est important ? Oui ? Alors pourquoi est-il commenté ? Si c'est juste pour garder une trace, le contrôle de code source est fait pour ça !

Conclusion

On pourrait parler de propreté de code pendant des pages et des pages. Certains l'ont fait. Si le sujet vous intéresse, je ne peux que vous encourager à lire l'excellent livre de Robert C. Martin "Clean Code", ou sa traduction en français "Coder proprement", livre que tout développeur devrait avoir lu. Il y a un principe intéressant que vous pouvez commencer à appliquer dès maintenant : le principe du boy scout. "Après votre passage, laissez toujours un code plus propre que vous ne l'avez trouvé". Pensez-y la prochaine fois. Et comme le disait Martin Golding, écrivez votre code comme si la personne qui allait devoir le maintenir était un psychopathe violent qui sait où vous habitez. ⊞

Les règles de base pour développer du code propre

Écrire du code propre ou clean code permet de mieux gérer la dette technique et de rendre plus simple l'évolution, mais aussi la maintenance des applications. Cette simplicité se traduit par un développement de logiciels plus rapide, mieux maîtrisé ainsi que des équipes plus productives.

Houssam Fakih
Software Engineer Arolla
@houssamfakih

Une condition nécessaire, voire obligatoire, pour fournir du code propre est notre capacité de :

- Se détacher complètement de notre code ;
- De savoir évaluer ses qualités et identifier ses défauts.

Ce détachement évite que toute critique envers le code soit perçue comme une attaque personnelle et favorise des discussions constructives qui nous permettent de nous améliorer. D'autre part, les développeurs sont souvent très critiques à l'égard des codes qu'ils n'ont pas écrit. Ils ne comprennent pas forcément les contraintes et les choix qui ont été faits par leurs confrères. Ces critiques souvent sévères ne favorisent pas la communication entre eux.

Par quoi se caractérise le code propre ?

Il y a plusieurs propriétés qui caractérisent le clean code. Ces propriétés sont communes à tous les développeurs, peu importe le langage de programmation utilisé :

- Le développement **itératif et incrémental** : toujours avancer par des petites étapes (baby steps) est le meilleur moyen pour tacler la complexité et avoir un feedback rapide sur notre travail. Le développement dirigé par les tests (TDD) est l'approche la plus connue pour ce mode de travail. Avec un cycle basé sur trois états : Red, Green et Refactor (souvent enrichi sous la forme : Think, Red, Green, Refactor & Repeat), le développeur commence par écrire un test qui devrait échouer dans un premier temps. Il écrit ensuite le code nécessaire pour faire passer ce test au vert. Puis il refactorise le code écrit pour améliorer sa qualité et le rendre plus propre. Il faut faire attention à ne pas négliger la qualité des tests et considérer que les normes de qualité sont les mêmes pour le code de production et les tests (clean tests).
- L'**autodocumentation** : un code autodocumenté doit parler de lui-même. Le développeur n'a pas besoin de l'expliquer en ajoutant des commentaires. D'où l'importance de bien choisir les noms des différentes briques à commencer par le nom des variables, mais aussi le nom des méthodes, des objets, des classes, etc. Un autre aspect de la documentation est l'écriture des tests qui, en plus d'aider à faire émerger le design et s'assurer que le code fonctionne comme prévu, permet de fournir une documentation utile pour la compréhension de nos programmes, leur fonctionnement et leur utilisation. Comparés aux commentaires qui deviennent rapidement obsolètes et rarement mis à jour, les tests échouent quand le code est modifié et n'est plus en phase avec le test. Si les développeurs prennent soin de faire passer tous leurs tests au vert après avoir modifié le code, ils vont régulièrement corriger les tests devenus obsolètes et avoir la garantie que leur documentation soit à jour.
- La **simplicité** et le **minimalisme** : un code simple est agréable à lire et se comprend facilement. Une structure simple (qu'elle soit une méthode, une classe, etc.) définit une et une seule fonctionnalité à chaque niveau de granularité. KISS (Keep It Simple Stupid) et YAGNI (You Ain't Gonna Need It) représentent les deux lignes directrices pour éviter la complexité dans le code. KISS rappelle aux développeurs qu'un code simple est toujours plus facile à maintenir et à comprendre qu'un code complexe tandis que l'approche YAGNI, qui est un principe de XP, encourage une attitude minimaliste en encourageant les développeurs à ne pas ajouter de nouvelles fonctionnalités tant que celles-ci ne sont absolument pas nécessaires. Un des moyens efficaces pour vérifier la simplicité de code est de le « reviewer » avec nos collègues. Si les reviewers n'arrivent pas à le lire et le comprendre au bout d'une à deux minutes, c'est qu'il n'est pas assez intuitif. Il faut partir du principe que c'est toujours le lecteur qui a raison et non pas celui qui écrit le code.
- La **localisation** : rien de pire qu'un code mélangé ou dispersé. Par souci de clarté, un clean code doit être localisé à un seul et même endroit. Ce critère doit aussi s'appliquer aux services rendus par l'application pour l'utilisateur final. Ce dernier, par exemple, ne devrait avoir à ouvrir qu'un seul écran pour configurer le lancement d'un calcul donné. Si cette configuration nécessite l'ouverture de plusieurs écrans, nous sommes dans le cas d'une dispersion. De même si un seul écran permet à l'utilisateur de configurer plusieurs options sans rapport entre elles, nous sommes dans un cas de mélange. Ces défauts rendent les applications complexes et alourdissent le temps d'apprentissage aussi bien pour les développeurs que pour les nouveaux utilisateurs.
- La **non-duplication** (DRY — Do not Repeat Yourself) : certains développeurs pratiquent la duplication en espérant gagner du temps. C'est un leurre, car il faudra systématiquement revenir sur toutes les duplications pour la moindre modification. La duplication représente donc un coût important pour la maintenance, l'évolution, mais aussi le chiffrage.

Ces différentes propriétés du clean code devraient être appliquées également aux tests avec la même exigence demandée pour le code qui part en production. Connaître les propriétés du clean code est une première étape nécessaire pour produire des logiciels de qualité. Le développeur doit s'habituer à vérifier au quotidien que son code respecte ces propriétés avant de le livrer. Avec des délais de livraison de plus en plus courts et des clients toujours plus exigeants, la tentation de tomber dans de fausses facilités s'entend et les développeurs ont souvent tendance à transgresser les règles du clean code parce qu'ils pensent que cela leur permet de gagner du temps. La transgression de ces règles engendre deux conséquences :

- La dégradation des compétences du développeur (prise de mauvaises habitudes) ;
- La dégradation de la qualité du code livré (risque d'introduire des anomalies ou de la complexité).

Néanmoins, quels que soient les délais et les contraintes, tout développeur devrait veiller au respect des règles du clean code par souci de professionnalisme, de clarté et d'optimisation de son travail.

Pour aller plus loin sur le sujet, les deux livres d'Uncle Bob : clean code et the clean coder sont un "must read".

Quel est le vrai coût d'une ligne de code ?

Quel développeur n'a jamais été agacé face à cette question récurrente ? Et systématiquement derrière ce type d'interrogation se cache un responsable marketing ou un client. Ces derniers attendent un montant sonnant et trébuchant, qu'ils pourront comprendre. Or, nous le savons, il n'est pas possible et pertinent de faire ce calcul. Cet article a pour vocation de vous donner des arguments pour expliquer à vos différents interlocuteurs, comment évaluer le coût du code dans sa globalité, c'est-à-dire en prenant en compte les aspects humains et les coûts cachés.

Aurélie GUILLAUME
Responsable du développement
Web chez Creads
@slig36
www.creads.fr

Partons donc de l'élément dont on nous demande le coût, à savoir "une ligne de code". Comme vous l'avez probablement constaté, il existe plusieurs manières d'écrire un test ou une fonction dans chaque langage.

Dans l'exemple ci-dessous, codé en PHP, les internautes doivent donner leur âge pour être autorisés à consulter tout ou partie des contenus du site. Voici 4 manières différentes d'écrire le code pour solliciter cette requête :

Exemple 1: 6 lignes de code

```php
$age = 42;
if ($age >= 18) {
    $majeur = true;
} else {
    $majeur = false;
}
```

Exemple 2: 5 lignes de code

```php
$age = 42;
$majeur = false;
if ($age >= 18) {
    $majeur = true;
}
```

Exemple 3 : 2 lignes de code

```php
$age = 42;
$majeur = ($age >= 18) ? true : false;
```

Exemple 4: 10 lignes de code

```php
$age = 42;

function isMajeur($age)
{
    $majeur = false;
    if ($age >= 18) {
        $majeur = true;
    }

    return $majeur;
}

$majeur  = isMajeur($age);
```

Avec ces 4 exemples, on pourrait en déduire que le coût de production de l'exemple 4 est le moins important tandis que l'exemple le plus cher serait l'exemple 3. Néanmoins l'exemple 3 peut manquer de lisibilité si la condition de test est plus compliquée. En outre, sa syntaxe n'est pas employée par tous les développeurs.

Imaginons également qu'au sein de votre application vous ayez besoin de faire ce test à 15 endroits différents, l'intérêt d'une fonction générique utilisable depuis plusieurs endroits du site vous fera gagner de précieuses minutes et lignes de code. On le voit ici, on ne peut pas affirmer que moins il y a de lignes de code plus leur coût sera élevé. Le temps de conception est à prendre en compte.

En effet, reprenons notre besoin qui est de savoir si une personne est majeure ou non. L'emploi du code de l'exemple 1 ou 2 de manière récurrente mènerait inéluctablement à la nécessité de factoriser ce code dans une fonction pour en éviter la duplication. Or, cela prend plus de temps de factoriser du code déjà produit et testé que d'avoir pris un peu plus de temps en conception pour avoir anticipé ce besoin. On constate donc que le temps de la conception ne se traduit pas dans le coût chiffré d'une ligne de code.

Et l'humain dans tout cela ?

L'anticipation du besoin devrait être traitée en amont de la production d'une ligne de code. En effet, c'est à ces moments là, qu'interviennent des acteurs (architecte logiciel, data scientist, ux designer, chef de projet, Scrumaster...) qui n'écrivent pas à proprement parler la ligne de code dont on souhaite connaître le coût. Ces acteurs produisent également de la valeur. Ils interviennent sur un périmètre donné. Ils ne produisent pas de code mais sont bien souvent essentiels à la ligne de code qui sera produite en définitive.

Après la réalisation de cette ligne de code unitaire, il y a également d'autres coûts humains cachés qui peuvent être ceux des testeurs, d'autres développeurs qui reliront la ligne produite ou un admin sys / devOps qui sera en charge de mettre en ligne cette ligne de code. Tous ces acteurs ont un coût qui ne peut être calculé par le simple ratio : nombre de lignes total du projet / nombre de développeurs sur le projet x coûts de ces développeurs.

Quick and dirty ?

Imaginons maintenant que l'on vous demande de développer un site facturé 10 000 € au client et estimé 30 jours-homme. Vous n'avez pas de spécifications, pas de chef de projet ou de référent fonctionnel, et que le projet s'avère être un projet urgent pour lequel vous ne disposez plus que de 20 jours-homme. La tentation du quick & dirty (production rapide du code mais sale / mal fait) deviendra très forte surtout si l'on vous dit qu'il ne faudra pas maintenir le site après l'avoir mis en ligne.

Si nous avions réalisé ce développement en 30 jours comme prévu initialement la somme de nos lignes de code n'aurait pas été un tiers supérieur au nombre de lignes de code produites sur les 20 jours. Cependant, dans 6 mois, la version 30 jours aura un coût de maintenance plus faible. Et pourtant dans notre exemple, le client paiera toujours 10 000 € son projet et ne connaîtra jamais cette différence qualitative de coût de conception qui restera dans les coulisses de l'agence Web ou la SSII. Il se demandera juste dans 8 mois pourquoi les coûts de maintenance (TMA) sont aussi chers...

Cost of Legacy code ?

La factorisation des lignes de code, au-delà de la conception en amont du projet, est un sujet important chez les développeurs. En effet, nous avons chacun notre manière de faire et donc notre manière de factoriser. Donc forcément lorsque l'on reprend le code d'un autre, a fortiori si c'est du "legacy code" (code non évolutif historique d'un projet), nous avons très envie de le faire évoluer, ou carrément de le réécrire entièrement. Cela ne doit pas être perçu par le client comme un frein ou une perte de temps et de productivité immédiate du projet, mais comme un investissement qualitatif à long terme. Cette réécriture va forcément faire varier le coût de notre ligne de code unitaire qui valait N € avant notre intervention, et vaudra Y € après notre passage.

La reprise de legacy code fait accroître le coût de celle-ci. En effet, plusieurs facteurs vont entrer en compte. Depuis combien de temps entretient-on ce code ? Qui en est à l'origine et qui est intervenu dessus ? Dans quel langage a-t-il été écrit ?

La notion de l'intervenant sur une ligne de code est importante et va fortement faire varier un coût d'intervention. En effet, si le développeur

intervenant est un développeur expérimenté, qui connaît l'historique du projet et de la techno, il sera en mesure d'intervenir de manière chirurgicale. Un jeune padawan (le nouveau stagiaire), passera son temps à essayer de comprendre le sens du code, et ne pourra pas intervenir aussi efficacement. Il devra être aidé par un senior assistant ou un autre développeur ce qui va aussi gonfler le coût du code.
De la même manière si un projet est passé entre de nombreuses mains sans avoir été un minimum géré par un Responsable Technique ou des gardes fous type "code Review" (relecture du code produit par d'autres développeurs de l'équipe), garants des grandes règles et principes de code du projet, il sera plus coûteux de le maintenir au fil du temps.

Processus qualité, un investissement utile ?

La notion de qualité entrevue au travers des revues de codes (Code Review) peut amener bien plus de coûts supplémentaires cachés qu'il faut être en mesure d'expliquer à un manager ou un client. En effet, la mise en place d'un process de contrôle qualité fonctionnel ou technique (test fonctionnel &/ou unitaire) participera au bon développement du projet mais reste un investissement initial à faire. Cela va forcément produire un plus grand nombre de lignes de code associées au projet mais facilitera sa maintenance. Il en va de même pour l'organisation et l'automatisation du processus de déploiement qui ne rentre pas directement dans le coût d'une ligne de code mais qui est à prendre en compte dans le coût global du projet.
La dette technique enfin est une notion qui n'est jamais budgétée par les entreprises. La dette technique comprend le code qui a été produit, mais qui n'est pas évolutif, ou les raccourcis de développement qui ont pu être utilisés dans l'urgence. Les entreprises tendent à ne pas prendre en compte les défauts de conception, de failles de sécurité ou d'obsolescence de la techno utilisée. Une étude réalisée par l'éditeur Cast a conclu qu'en moyenne le coût de la dette technique revenait à 3.61 $ par ligne de code et que le langage qui serait le plus touché serait le Java. Cela démontre encore une fois que plus la problématique de la qualité de code est prise au sérieux par le client et l'équipe en amont, plus ce coût diminuera et permettra des économies sur le long terme.

Les failles de sécurité ont aussi un coût !

La sécurité et le respect des lois sont également des enjeux à garder en tête pour une grande majorité de clients. Pour vous aider à les comprendre, nous allons vous mettre dans leur situation. Imaginons que vous êtes l'un des décideurs de Volkswagen et que vous demandez à l'un de vos développeurs de rajouter quelques lignes de code pour flooder le système de contrôle anti-pollution aux États-Unis. Le coût de cette modification en termes strict de lignes de code sera très faible. Mais le contre coup du non-respect de la législation américaine est énorme en termes de procès devant les tribunaux et d'image de marque.
Autre exemple, vous êtes maintenant à la tête d'un gros site e-commerce français et certains de vos clients un peu bidouilleurs arrivent à payer un panier de 800 €, 1 € symbolique... A ce rythme-là vous allez mettre la clé sous la porte rapidement et tout cela car un développeur n'a pas correctement sécurisé le système de paiement en ligne. Cela vous paraît possible car vous êtes développeur, mais si vous prenez des exemples réels pour votre interlocuteur, il s'intéressera beaucoup plus attentivement au temps supplémentaire pour développer son produit ou à l'audit sécurité que vous lui conseillez à la fin du développement. Et tout ceci n'aura pas changé le coût de la ligne de code en soit !

Et l'hébergement ?

Il en va de même pour les coûts cachés liés à l'infrastructure. Reprenons l'exemple de notre site e-commerce Français. Leur service de communication a décroché un passage TV dans l'émission Capital. Appel en panique du DG à 21h52 le dimanche soir : *"Le site ne fonctionne plus, je vais perdre des milliers de commandes à cause de vous !!!".* En fait non, c'est plutôt à cause de lui. En effet, avant une augmentation prévue de l'audience, il est judicieux de faire faire des tests de montée en charge pour déceler des problèmes, et les résoudre grâce à la mise en place d'outils spécifiques (ajout de serveurs, systèmes de cache, CDN). Au delà des lignes de code supplémentaires nécessaires à la mise en place de ces outils, le coût en temps-homme pour la configuration de ces derniers n'est pas négligeable. Ces nouvelles lignes de code auront un coût. Cependant, ce coût supplémentaire sera moindre face au manque à gagner d'un site en panne suite à un pic d'audience mal anticipé.

.Net vs PHP, Objective C vs Java ?

Le choix de la techno de développement du projet est également un facteur influent sur le coût du projet. En effet, si le choix se porte vers du .Net vs du PHP, le coût du projet sera supérieur. En effet, le développeur .Net est plus rare que le développeur PHP donc plus cher, CQFD. De plus, un développement dans un langage propriétaire induira aussi une augmentation des coûts annexes (qui ne sont pas directement liés au développeur produisant du code) tel que l'hébergement (ici IISS), le coût d'utilisation de licence tierce (SQL server, Oracle...).
De plus, l'écriture en soit d'un test ou d'une fonction ne produira probablement pas le même nombre de lignes que l'on soit en python, en C++, ou en PHP.
En 2016, le choix de la techno n'est plus exclusivement de la responsabilité du développeur. En effet, dans le cadre d'une demande d'une application mobile, il est possible que le client ait besoin d'une application développée en langage natif, donc spécifique pour le type de mobile. Ou bien on pourra lui proposer le développement d'une application "universelle" grâce au framework d'encapsulation type "Phonegap", Cordova ou Ionic. Ce choix orientera forcément le coût unitaire de la ligne de code, car le développement natif est plus onéreux que le développement "cross platform'. De plus, si l'application demande un développement de fonctionnalités spécifiques sur les 2 platform leaders , le coût total de l'application sera au minimum 2 fois supérieur à celui d'un développement "hybride".

CMS, from scratch

Dans le questionnement des outils et technos à utiliser pour développer un projet, vous pouvez également être amené à vous poser la question : *"Dois-je utiliser un développement "From scratch", un framework ou un CMS (pour du Web) ?"*
En termes de nombre de lignes de code pur, le framework aura un nombre de fichiers et donc de lignes de code assez important pour faire tourner son coeur (routeur, moteur graphique, base de données...). Il en est de même pour le CMS qui aura des fichiers pour son fonctionnement mais qui embarquera probablement un back office d'édition du contenu plus ou moins complexe, ce qui va produire aussi beaucoup de lignes de code supplémentaires qui viendront noyer le véritable coût du développement et de la valeur ajoutée du développeur sur le projet.
Car oui, dans l'absolu produire un projet qui a 350 000 lignes de code dont 98% ont été produites indirectement, que ce soit par un Framework ou un CMS cela ne reflétera pas le coût réel du projet.
Prenons le site creads.fr, à titre d'exemple, nous utilisons le Framework Symfony2 pour nos développements. Cela fait 2,5 ans que nous travaillons sur le site. Le ratio de notre dev vs la base de Symfony est totalement différent et reflète plus la valeur ajoutée de nos développements. En effet, si l'on prend un projet Symfony 2.3 fraîchement installé, il contiendra 306 851 lignes de code pur PHP. Par contre, sur notre

projet, le nombre de lignes de code PHP comprend 482 643 lignes produites par nos soins. Faut-il pour autant déprécier la valeur des lignes du Framework utilisé ? Bien sûr que non. Le code du Framework, nous permet de gagner un temps précieux pour nous concentrer sur l'essentiel : le code métier du client, celui qui aura une forte valeur ajoutée mais également celui qu'il paye au final.

Si vous souhaitez connaître le nombre de lignes de code de vos projets, je vous invite à installer Cloc. Cloc est un utilitaire open source qui va permettre de connaître le nombre de lignes de code de vos projets par langage, tout en ignorant les fichiers types .git et les commentaires.

En conclusion

Désormais, lorsque l'on vous posera la question *"Quel est le vrai coût d'une ligne de code ?"*, vous pourrez énumérer un certain nombre d'arguments à votre interlocuteur et lui expliquer en quoi sa question n'est peut-être pas pertinente. Car, en effet, le coût d'une ligne de code ne veut pas dire grand chose. L'intégralité des coûts visibles et cachés doivent être pris en compte et non juste appliquer une logique de coûts unitaires. En somme, on pourra retenir que ce n'est pas le nombre de lignes de code d'un projet qui en définira son coût final pour le client mais plutôt le temps, les moyens humains, fonctionnels et technologiques affectés au projet qui devront être pris en compte.

```
http://cloc.sourceforge.net v 1.60  T=226.57 s (64.6 files/s, 7055.3 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
PHP                          10835         159391         310669         789494
Ruby                          1035           9674           2203          65952
Javascript                     479           9446           9504          59571
XML                            431           1472            119          34639
LESS                            85           3393            938          25715
Pascal                         600           3468          12325          22063
CSS                            258           2274            810          21631
HTML                           144           2785            265          18112
YAML                           626           1438            397          15617
XSD                             19            317             88           4297
Bourne Shell                    70            695            510           3342
Bourne Again Shell              20            189            193           1190
C                                1            166            150            793
Perl                             1             56            126            784
SQL                              9            169              2            731
Ant                              4            111             65            434
make                             4             60             28            178
DOS Batch                        2             17              1            104
Python                           2             69            144             81
C/C++ Header                     1              7             13             11
m4                               1              2              1              5
ASP.Net                          1              0              0              1
-------------------------------------------------------------------------------
SUM:                         14628         195199         338551        1064745
-------------------------------------------------------------------------------
```

(Nombre de lignes de code de creads.fr)

SOURCES

La dette technique qui coûte le plus chère est en JAVA :
http://www.lemagit.fr/actualites/2240198129/Les-defauts-logiciels-coutent-en-moyenne-361-par-ligne-de-code

CLOC : https://github.com/AlDanial/cloc#Overview

Google 2 000 000 lignes de codes
http://www.abondance.com/actualites/20150918-15577-google-cest-2-milliards-de-ligne-de-code.html

Infographie ligne de code des grands projets mondiaux :
http://www.numerama.com//content/uploads/2013/10/lignesdecodeinfographie.png

Moi je code m'ADOC

En tant que développeurs, nous sommes évidemment amenés à produire du code, mais également de la documentation : fichiers README, spécifications techniques/fonctionnelles, guide d'installation, manuel utilisateur, wiki, ... N'oublions pas nos amis étudiants et le fameux rapport de stage. Le but de cet article est de montrer comment mieux écrire sa « doc » en choisissant les bons outils collaboratifs provenant du monde Open Source. Préférons des outils simples, ouverts, gratuits, maintenus à jour par la communauté.

Mehdi Rebiai
mrebiai@sqli.com
Architecte technique SQLI. Speaker Bdx.io 2015 « moi je code ma doc ». Plus de 10 ans d'expérience dans le développement d'applications Java/JavaEE.
Curieux d'apprendre de nouveaux sujets et aimant transmettre au plus grand nombre.

Attention au grand écart entre le monde du développement et les outils de la sacro-sainte documentation ! À ma gauche, un univers industrialisé (IDE, GIT, releases), et à ma droite une suite bureautique (MS Office, OpenOffice ...) complétée d'un modeleur UML et d'outils annexes. À moins d'utiliser une GED ou des outils collaboratifs genre « GoogleDoc », la documentation est souvent stockée sur un répertoire réseau d'entreprise avec la fameuse technique du « copier-coller-suffixer »! MonFicher-v1.doc deviendra MonFicher-v2.doc et un dossier « archives » sera créé pour gérer l'historique des anciennes versions. Drôle de versioning.

La figure 1 illustre ce grand écart « Doc vs Code » pour une application, chaque branche ayant son propre cycle de vie. Et pourtant, lors d'une livraison, on se doit de réconcilier l'artéfact applicatif (WAR, ZIP) et l'artéfact documentaire (DOC, ZIP). À noter également que le code contient sa propre documentation interne (commentaire, JavaDoc). Nous y reviendrons un peu plus tard. **Fig.1.**

Et si la « doc » était du code? La figure 2 prend cette hypothèse. Et maintenant tout devient plus simple. Il faut néanmoins changer certains outils. En allant plus loin, tout code est documentation, dixit le dicton du développeur « la vérité est dans le code »! **Fig.2.**

Vive le fichier plat!

Le principe du MDA (Model Driven Architecture) a longtemps été mis en avant pour son côté rigoureux. Un modèle UML source contient la conception UML complète de l'application ainsi que sa documentation. L'application est ainsi générée à partir de ce "gros" modèle source. Cette approche,

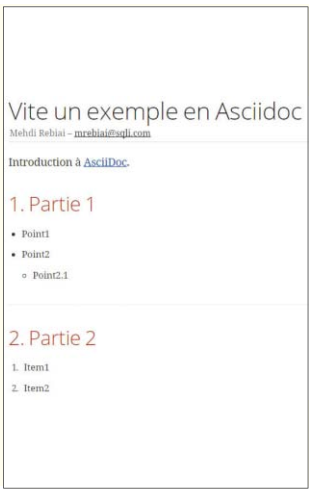
a priori séduisante, se confronte vite au problème du travail collaboratif et au versioning/branching/merging quasiment impossible. Ce frein limite le refactoring du code et de la documentation.
Le choix de MS Office pour la documentation signifie des fichiers binaires, au mieux XML, ne permettant que difficilement le travail collaboratif, à moins d'investir dans les outils Microsoft adéquats : SharePoint, Office365. Rien de mieux qu'un bon fichier plat. En effet, il est éditable par n'importe quel éditeur de texte. On peut l'intégrer sous sGit pour obtenir gratuitement le versioning et le merging. Notre code Java, JavaScript ... est contenu dans des fichiers plats.
Pourquoi ce concept, admis pour le code, ne le serait-il pas pour la documentation? Par exemple, chez SQLI ISC (Innovative Service Center) France, nous repensons complètement notre SI et nos outils, dans le but d'améliorer la qualité de nos services et de nos livrables. L'approche "As Code" prend un rôle central, au travers de l'Infrastructure As Code, entre autres. Tout doit être scriptable afin de garantir une reproductibilité. Notre SI se décrit ainsi par un ensemble de fichiers plats. Notre documentation, en asciidoc, rentre également dans cette philosophie via la "Documentation As Code".

Asciidoc(tor)

Le bon outil pour écrire de la doc serait bien l'asciidoc. Il s'agit d'un langage de balisage léger, créé en 2002. À partir d'un fichier texte source (ADOC), on peut générer des documents publiables (HTML, PDF ...). Fig.3.
Le processeur initial était écrit en Python. "Le contenu avant la forme", voici la devise d'asciidoc. Regardons ci-dessous un premier exemple de document avec son rendu HTML.

Table 1. Premier asciidoc

```
= Vite un exemple en Asciidoc
Mehdi Rebiai <mrebiai@sqli.com>

Introduction à http://asciidoc.org[AsciiDoc].

== Partie 1

* Point1
* Point2
** Point2.1

== Partie 2

. Item1
. Item2
```



La syntaxe asciidoc qui a eu du mal à percer au début, contrairement à son rival Markdown, semble bien rattraper son retard depuis l'arrivée d'asciidoctor (http://asciidoctor.org). Il s'agit d'un nouvel écosystème, basé sur la syntaxe asciidoc. Ce projet Open Source, dirigé par Dan Allen son "project Lead", a remplacé l'ancien processeur Python par une implémentation Ruby. De nombreux outils et intégrations sont proposés afin de simplifier l'écriture et la publication de documentation. Afin de prouver le succès grandissant de l'asciidoc, ce format est désormais géré par de nombreux acteurs du Web, dont GitHub et GitLab. Pour information, la documentation du célèbre SpringFramework a été écrite en asciidoc.
Continuons le parcours de nombreuses possibilités offertes par la syntaxe asciidoc, illustrées par l'outillage asciidoctor. La figure ci-dessous montre le mécanisme d'inclusion d'image, de vidéo, de code et d'autres fichiers asciidoc :

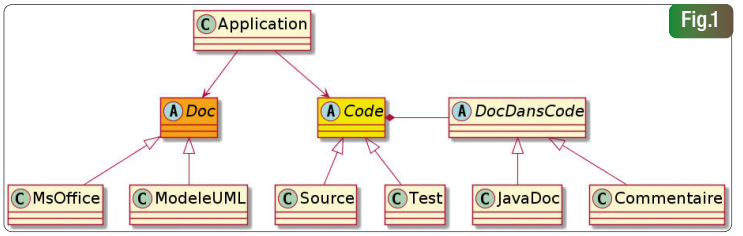


Fig.1
Doc vs Code ?

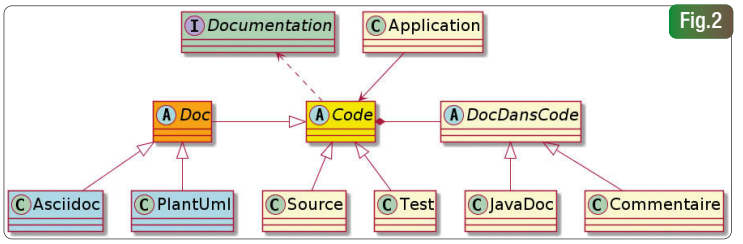


Fig.2
Et si Doc = Code ?

```
= Include
image::monImage.png[height=100]
vidéo::monVideo.avi[]

[source,JavaScript]
----
include::HelloWorld.js[lines=2..4]
----

:val: cas1
include::test.adoc[]
:val: cas2
include::test.adoc[]
```

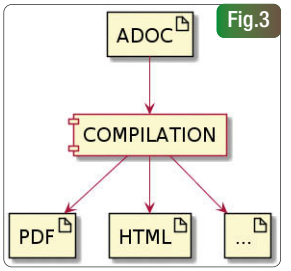


Fig.3
Une source, plusieurs documents publiables

Personne n'imagine écrire une classe Java et des milliers de lignes. De la même façon, n'écrivez plus de gros documents. Découpez-les et utilisez les inclusions. Autre point très intéressant, lorsque l'on souhaite inclure du code dans de la documentation, fini le "copier-coller" ou le "screenshot" qui provoque obligatoirement une décorrélation dans le temps. Utilisez l'inclusion de fichier code, en profitant même de la coloration syntaxe (via les outils coderay, pygments, highlightjs). L'inclusion peut être totale ou partielle, en précisant les lignes à extraire ou via des tags commentaire dans le source. On peut également définir un sommaire automatique, créer des variables et les utiliser, créer des tableaux (inclusion CSV possible), inclure des diagrammes (voir chapitre PlantUML), créer des formules mathématiques et plus encore. Le principe du "Doc As Code" nécessite une phase de compilation de la documentation. Eh oui, on compile sa doc, tout comme son code. La commande ci-dessous propose la compilation d'un fichier monfichier.adoc et la création d'un fichier monfichier.html (HTML5 par défaut).

```
asciidoctor monfichier.adoc
```

La commande peut être complétée de paramètres (non exhaustifs) afin de "customiser" la compilation :

```
asciidoctor -D <output> -T <template> -r <requirements> monfichier.adoc
```

Il faudra préalablement avoir installé Ruby et la gem asciidoctor. Évitez-vous cette étape en utilisant Docker. Ce logiciel libre au succès grandissant est en passe de devenir un incontournable du monde informatique.
Asciidoctor propose une image officielle Docker. Le développeur peut ainsi déléguer la compilation de sa documentation à un conteneur "boîte à outils".
La commande vue ci-dessus est à positionner au niveau de <CMD> :

```
docker run --rm -v <WORKSPACE>:/documents/ asciidoctor/docker-asciidoctor <CMD>
```

Pour plus d'informations sur la syntaxe asciidoc ou sur l'outillage asciidoctor, n'hésitez pas à consulter le site http://asciidoctor.org, réalisé évidemment en asciidoc. Cette syntaxe asciidoc peut faire peur au début, mais elle s'apprend très rapidement. Les mots clés à retenir sont **Simplicité** & **Productivité**. Maintenant, demandons-nous comment réaliser des schémas, des diagrammes avec cette approche fichier plat.

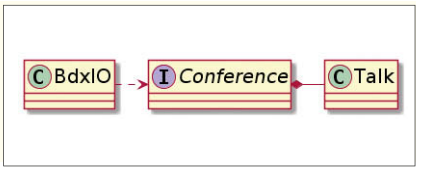
PlantUML

Pourquoi PlantUML (http://fr.plantuml.com) ? Tout simplement, car c'est LA solution pour faire de l'UML à partir de fichiers plats. Cette solution Open Source s'intègre parfaitement avec l'outillage asciidoctor. On écrit des fichiers puml et on génère en sortie des images png, svg, ... Ajoutons le besoin d'installer l'outil graphviz (http://www.graphviz.org) pour le rendu de certains diagrammes.

Le fichier puml commence par @startuml et se termine par @enduml. La suite repose sur le principe de l'"ascii-art" pour dessiner les relations UML. De nouveau, on se focalise plus sur le contenu que sur la forme. On peut tout de même jouer sur le style et positionnement des formes.

Table 2. Premier exemple PlantUML

```
@startuml
interface Conference
Conference *- Talk
BdxIO .> Conference
@enduml
```



Tout comme pour asciidoc, avec PlantUML il y a une phase compilation. Cette dernière peut être exécutée de façon isolée via l'utilisation du jar plantuml.jar.

Génération de monfichier.png

```
java -jar plantuml.jar monfichier.puml
```

Mais on préfèrera la solution offerte par asciidoctor. Il suffit d'inclure le PlantUML dans un fichier adoc. On notera l'activation de la dépendance asciidoctor-diagram :

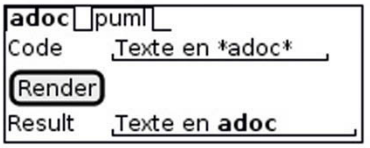
Génération de monfichier.html contenant le png du diagramme

```
asciidoctor -r asciidoctor-diagram monfichier.adoc
```

Mais de nouveau, l'image Docker officielle d'asciidoctor propose tout le nécessaire.

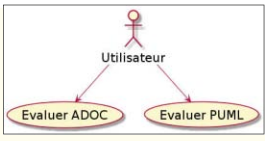
```
docker run --rm -v <WORKSPACE>:/documents/ asciidoctor/docker-asciidoctor <CMD>
```

Afin d'illustrer les possibilités de PlantUML, prenons un cas concret d'une application Web. Cette application réalise du "live rendering" de texte asciidoc et en PlantUML :



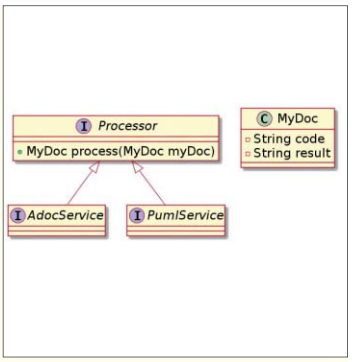
Par où débuter lorsque l'on réalise une conception technique? Et bien cela commence souvent par la création de diagrammes de cas d'utilisation :

```
@startuml
:Utilisateur: --> (Evaluer ADOC)
:Utilisateur: --> (Evaluer PUML)
@enduml
plantuml::puml/Usecase.puml[nom-fichier-image, format="png", align="center"]
```



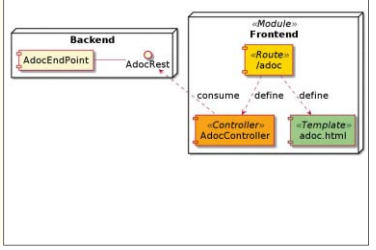
Continuons par la réalisation du modèle de données et de l'API de services :

```
@startuml
class MyDoc {
 - String code
 - String result
}
interface Processor {
 +MyDoc process(MyDoc myDoc)
}
interface AdocService -up-> Processor
interface PumlService -up-> Processor
@enduml
```



Ensuite le diagramme de composants permet une bonne visualisation globale d'une application :

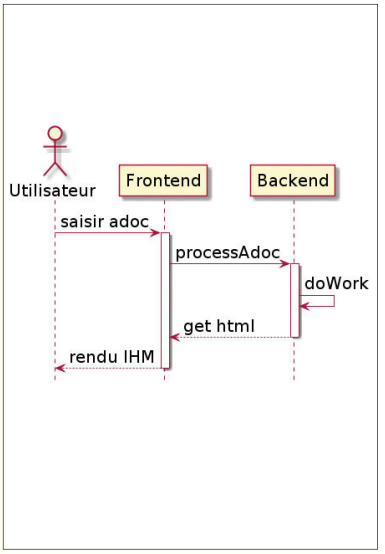
```
@startuml
node "Frontend" <<Module>> {
 [/adoc] <<Route>> as route_adoc #Gold
 [AdocController] <<Controller>> as contr_adoc #orange
 [adoc.html] <<Template>> as templ_adoc #LightGreen

 route_adoc ..> contr_adoc : define
 route_adoc ..> templ_adoc : define
}

node "Backend" {
 [AdocEndPoint] - AdocRest
}
contr_adoc .up.> AdocRest : consume
@enduml
```



Le classique diagramme de séquence n'est pas oublié. Vous noterez l'extrême simplicité de sa réalisation :

Table 3. Sequency

```
@startuml
actor Utilisateur
participant Frontend
participant Backend

Utilisateur -> Frontend : saisir adoc
activate Frontend
Frontend -> Backend : processAdoc
activate Backend
Backend -> Backend : doWork
Backend --> Frontend : get html
deactivate Backend
Frontend -->Utilisateur : rendu IHM
deactivate Frontend

hide footbox
@enduml
```



Pourquoi investir financièrement dans des gros modeleurs UML alors qu'on les sous-exploite bien souvent. De plus, un modèle UML est rarement maintenu à cause de la lourdeur de la mise à jour. Le fichier plat et

PlantUML semblent bien répondre au problème. Ce type d'outillage n'ira jamais aussi loin qu'un véritable modeleur UML. Mais la force de PlantUML n'est pas là. En plus d'être gratuit, il rend facile le travail collaboratif et le refactoring. N'hésitez pas à vous référer à la documentation de PlantUML (http://fr.plantuml.com) pour plus d'informations.

Workflow de génération

Remplacez MS Office et un modeleur UML par asciidoc et PlantUML ne fonctionnera que si le processus de génération est simplifié. En effet sans automatisation du "workflow de génération", ces outils resteraient confidentiels et réservés aux plus technophiles d'entre nous. Pour le poste du développeur, de nombreux outils proposent du "live rendering": plugins dans IDE (Eclipse, IntelliJ), éditeur (atom.io), extensions navigateur (Chrome, Firefox), ...

Dans le cas classique d'un projet Java, la génération de la documentation peut s'intégrer au cycle de vie Maven/Gradle, grâce à l'outil JRuby qui concilie Java et Ruby. À partir d'un répertoire ***src/main/asciidoc***, un dossier ***target/generated-docs*** sera créé. Désormais, la compilation d'un projet impacte à la fois le code source et la documentation. Vous trouverez de très bons exemples Maven/Gradle sous GitHub :

https://github.com/asciidoctor/asciidoctor-maven-examples

https://github.com/asciidoctor/asciidoctor-gradle-examples

Les développeurs JavaScript ne sont pas oubliés. Le projet asciidoctor.js propose un portage d'asciidoctor en JS, grâce à l'outil Opal. Des plugins Grunt/Gulp permettent ainsi de réaliser le même type d'intégration que Maven/Gradle. Poussons la logique un peu plus loin: et si l'on écrivait un wiki en asciidoc, avec la contrainte d'avoir les fichiers des pages sous Git? Comment dynamiser la génération? Où stocker les pages HTML générées? Comment maîtriser les droits d'accès à ce wiki? L'outillage asciidoctor propose un plugin poussant les fichiers HTML générés vers le portail Confluence d'Atlassian. Mais n'y a-t-il pas une solution plus simple et surtout gratuite et basée que sur des outils Open Source? Que de questions, mais rassurez-vous, des outils existent! Le projet Hubpress permet de réaliser rapidement des blogs sous Github à partir de source asciidoc. Malheureusement cela ne pourra pas convenir pour les entreprises à cause du côté public. L'outil awestruct permet la génération d'un site Web statique. Il est développé en Ruby et propose un plugin asciidoctor permettant de compiler en live les fichiers adoc. Mais n'y a-t-il une solution plus simple encore?

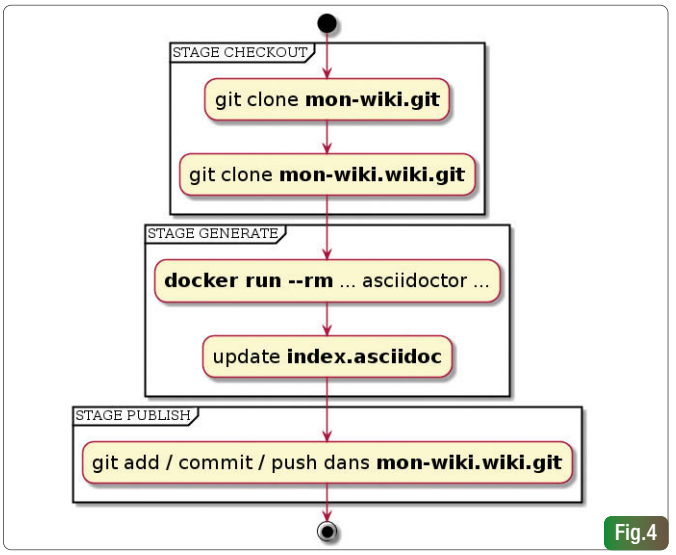


Fig.4

Voici l'idée pour répondre au stockage et à la publication : l'outil Gitlab est utilisé comme serveur Git pour héberger les sources asciidoc et sa partie "wiki" (repository fils de celui des sources) être utilisé pour héberger les fichiers HTML générés. Sur un projet "MyProject", Gitlab propose une IHM d'administration des droits d'accès. Le projet fils "MyProject.wiki" héritera des mêmes droits. Si je vois le fichier adoc, je vois le fichier HTML. À noter que Gitlab propose une interprétation propre de la syntaxe asciidoc, sans toutefois rivaliser avec asciidoctor. Par exemple, Gitlab n'interprète que le code PlantUML. Utilisons plutôt l'image Docker pour générer nos pages HTML. L'outil Open Source Jenkins propose le "Jenkins Worflow Plugin" qui permet de coder son workflow via le langage Groovy. Cette solution est très intéressante, car on peut sortir du schéma classique d'un job. Un job "Workflow Plugin" peut créer à la volée un conteneur Docker pour la génération de la documentation. Voici un schéma explicatif d'un job Jenkins permettant de générer les pages HTML d'un wiki : **Fig.4.**

On imagine maintenir un fichier index.asciidoc listant l'ensemble des pages du wiki. Ce job peut s'exécuter à la suite de tout changement sous Git des sources asciidoc. **Fig.5**. Cette solution permet d'avoir un wiki toujours à jour. Un projet de documentation doit être vu comme tout projet de développement, avec la mise en place d'une plateforme d'intégration continue (ici Jenkins) possédant des jobs validant régulièrement de la bonne bonne santé du projet : compilation, publication, ...

Au final

L'outillage asciidoctor permet de faire encore beaucoup plus de choses :

- Générer de belles présentations RevealJS, DeckJS, ... afin de remplacer notre ancien MsPowerpoint ;
- Ecrire sa JavaDoc en asciidoc à l'aide d'asciidoclet.

Maintenant que les outils ont été présentés, le sujet de "Living documentation" peut être abordé. TODO

Vous retrouverez l'intégralité des ressources présentées (PlantUML, Docker, Groovy) sous GitHub : https://github.com/mrebiai/moijecodemadoc

Remerciements

Je tiens à remercier Benoit Prioux qui m'a initié à asciidoctor, Christian Chavez qui m'a sensibilisé à docker et à Rémi Goyard pour m'avoir convaincu de partager ce sujet au plus grand nombre.

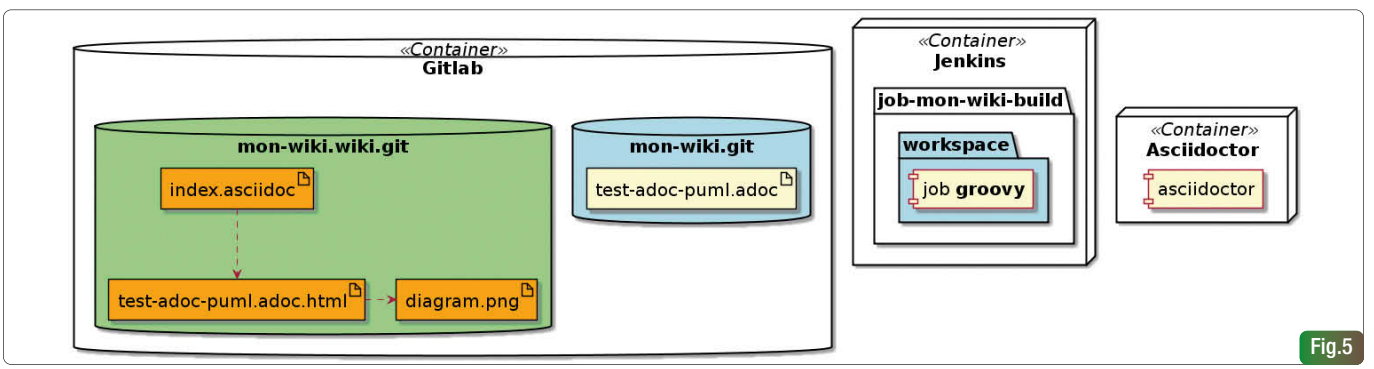


Fig.5

Le retour de la démoscène

Après avoir consacré un dossier vintage (n° 192), il était important de voir ou revoir les animations qui vous faisaient rêver dans les années 1980 / 90, portées sur le Web. Pourquoi ne pas en créer de nouvelles en HTML 5 ?

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

Approche

La scène démo ou appelée 'demoscene' est une autre culture de l'informatique ayant pour but la création artistique sous la forme de programmes autour de trois domaines : la musique, le graphisme et la programmation.
L'objectif est de réaliser des animations appelées « démos », en associant les performances autant techniques que artistiques. Tout en s'appuyant sur les astuces de programmation et de performances.
L'origine des démos était de petites animations distribuées avec les jeux et les logiciels piratés. Il s'agissait souvent de la signature des groupes qui ont réussi à cracker (pirater) un jeu ou une application.
Ces réalisations s'appelaient cracktro, un acronyme qui mélangeait 'crack' et 'intro'. Rapidement, les groupes rivalisèrent au niveau de la technique pour montrer leurs talents. Cette mutation a permis de voir apparaître un nouveau courant et d'organiser des concours, des coding party pour laisser apparaître la démoscène et se désolidariser totalement du piratage des années 90. Au fil des années, les exploits techniques sont toujours apparus quel que soit le matériel (ordinateur, GPS, Minitel, smartphone), et ont toujours vu des tentatives de hacks pour voir les possibilités d'améliorer le produit et de réaliser des démos. Enfin, dans cet univers underground, le web n'a pas échappé à la règle de voir réaliser des animations, quels que soient le langage et la technologie (HTML, Flash, Java, JavaScript...).

La mutation

L'Internet n'a pas chamboulé les technologies d'avant, elle offre d'autres horizons et possibilités de détourner ou de modifier (hacker) l'utilisation d'origine d'un matériel. Au niveau des langages de l'époque, l'Assembleur, le Basic, le C, le Pascal... ont laissé la place au HTML 5 et JavaScript. Ce changement a été facilité par l'arrivée de nouvelles balises comme Canvas, WebGL... avec lesquelles les navigateurs modernes sont compatibles.
La contrepartie de cette technologie de plus en plus puissante, touche la notion d'exploit qui s'est retrouvée en second plan, car le matériel hardware n'est plus pris en compte. C'est pourquoi les concours de l'époque ont subi quelques changements au niveau des règles comme par exemple la taille d'un fichier, réaliser une animation en 4Ko qui correspond à la taille d'un document Word vierge.
Toutefois avec les smartphones, les objets connectés, les exploits sont de nouveau possibles, car le défi hardware est de nouveau présent.

CODEF

CODEF signifie Canvas Oldshool Demo Effect Framework. Il s'agit d'un Framework Open Source à destination des programmeurs qui aiment les démos animées et les faire tourner dans un navigateur Web.

But

L'existence d'un Framework comme CODEF, permet de réaliser des animations sans besoin de tout reconstruire.

Fig.1

Même s'il existe de nombreux frameworks JavaScript, celui-ci est spécifique. Il bénéficie de l'ensemble des fonctionnalités présentées ci-dessus.

We Are Back

Pour montrer la puissance de ce framework, son concepteur propose de retrouver les nombreuses réalisations dans une galerie appelée 'We Are Back' (http://wab.com/). Ainsi, vous retrouverez différentes catégories classées par machines. Tout d'abord, les démos réalisées sur Atari, Amiga, C64 et portées sur le framework. Ensuite, vous trouverez des rubriques GL, Compo, Jeu (Game). Elles correspondent à des nouveaux écrans jamais créés dans le passé.

Cas simple

Pour réaliser une simple animation en 2 dimensions, nous pouvons nous appuyer sur les fonctionnalités 'canvas' du HTML 5. Le framework va vous permettre de réaliser cette opération facilement ; par exemple nous effectuons un déplacement horizontal d'une image sur le principe d'un scroll, Fig.1.

Etape 1 : Déclaration

```
<script src="codef/codef_core.js"></script>
<script>
 var myimage=new image('logo.png');
 var mycanvas;
 var myimageX=0;
 var myimageMove=1;

 function init(){
  mycanvas=new canvas(640,480,"main");
  go();
 }
</script>
```

Le script déclare une balise HTML5 'canvas' avec une taille précise. Nous utiliserons le 640 par 480. Ensuite, il est important de déclarer certaines variables qui nous seront utiles plus tard.

Etape 2 : Préparation

Tout d'abord il est indispensable de lui attribuer un fond, pour cela la couleur noire est utilisée pour donner un rendu de vide. Nous utilisons la fonction fill.

```
fill('black');// en hexadécimal #000000
```

Pour afficher notre logo 'Programmez', nous lui attribuons une position en X et Y. Pour cela, la fonction draw sera utilisée.

```
draw(mycanvas,myimageX,myimageY) ;
```

Etape 3 : Animation

Pour créer l'effet de mouvement, nous prévoyons un déplacement horizontal avec la position X qui sera incrémentée de 1 pour aller à droite et de -1 pour aller à gauche. Le changement de mouvement se détermine par les limites de la zone canvas.

```
<script>
    function go(){
     mycanvas.fill('#000000');
     myimage.draw(mycanvas,myimageX,240) ;
     myimageX+=myimageMove;

     if(myimageX>640-myimage.img.width)
       myimageMove=-1;
     if(myimageX<0)
       myimageMove=1;


     requestAnimFrame( go );
    }
  </script>
```

La fonction 'requestAnimFrame' permet de répéter notre boucle pour obtenir comme résultat l'animation.

Etape 4 : Exécution

Pour charger et lancer notre animation, nous utiliserons dans la balise div que nous appelons « main », pour obtenir ceci :

```
<HTML>
<head></head>
<body onLoad="init();">
<div id="main"></div>
</body>
</HTML>
```

Cas avancé

Le framework peut aussi s'appuyer sur les autres fonctionnalités du HTML 5, comme WebGL, Webkit... Grâce à cela, il utilise la librairie three.js pour faciliter l'utilisation de la 3D. Fig.2.

Fig.3

Fig.2

Etape 1 : Préparation

Pour réaliser l'animation, nous utilisons cette image que nous avons réalisée. Fig.3.

Etape 2 : Les déclarations

L'étape de la déclaration est importante, car nous réalisons la forme graphique qui sera représentée. L'article montrera un losange en 3 dimensions. Pour cela, nous construisons un objet 'myobjvert' à partir des différents points de conception.

```
<script>
    var mycanvas;
     var my3d;

     var myobj = new Array();
     var myobjvert = new Array();
     myobjvert=[
      {x:-100, y:0, z: 100},
      {x: 100, y:0, z: 100},
      {x: 100, y:0, z: -100},
      {x:-100, y:0, z: -100},
      {x:0, y: 200, z: 0},
      {x:0, y:-200, z: 0},
     ];
</script>
```

Ensuite, nous remplissons les différentes faces de notre objet par l'image que nous avons réalisée précédemment dans un objet appelé 'myobj'.

```
<script>
var myimage=new image('programmez3.png');
 myobj=[
{p1:1, p2:4, p3:0, u1:1,v1:1, u2:0.5,v2:0, u3:0,v3:1,
params:new MeshBasicMaterial({ map: new Texture( myimage.img ), opacity:0.4 })},
{p1:2, p2:4, p3:1, u1:1,v1:1, u2:0.5,v2:0, u3:0,v3:1,
params:new MeshBasicMaterial({ map: new Texture( myimage.img ), opacity:0.4 })},
{p1:3, p2:4, p3:2, u1:1,v1:1, u2:0.5,v2:0, u3:0,v3:1,
params:new MeshBasicMaterial({ map: new Texture( myimage.img ), opacity:0.4 })},
{p1:0, p2:4, p3:3, u1:1,v1:1, u2:0.5,v2:0, u3:0,v3:1,
params:new MeshBasicMaterial({ map: new Texture( myimage.img ), opacity:0.4 })},
{p1:0, p2:5, p3:1, u1:1,v1:1, u2:0.5,v2:0, u3:0,v3:1,
params:new MeshBasicMaterial({ map: new Texture( myimage.img ), opacity:0.4 })},
{p1:1, p2:5, p3:2, u1:1,v1:1, u2:0.5,v2:0, u3:0,v3:1,
params:new MeshBasicMaterial({ map: new Texture( myimage.img ), opacity:0.4 })},
{p1:2, p2:5, p3:3, u1:1,v1:1, u2:0.5,v2:0, u3:0,v3:1,
params:new MeshBasicMaterial({ map: new Texture( myimage.img ), opacity:0.4 })},
{p1:3, p2:5, p3:0, u1:1,v1:1, u2:0.5,v2:0, u3:0,v3:1,
params:new MeshBasicMaterial({ map: new Texture( myimage.img ), opacity:0.4 })},
];
</script>
```

Chaque intercession représente une face et définit comme une position avec la définition de mouvements et de rapidité. De plus nous appliquons notre structure image dans la forme géométrique avec la fonction 'MeshBasicMaterial' avec un fond transparent.

Etape 3 : Porte d'entrée

La porte d'entrée pour afficher l'animation passe par la fonction init() que nous plaçons dans le corps de la page pour l'exécuter.

```
<!DOCTYPE HTML>
<HTML>
 <head>
  <script src="codef/codef_core.js"></script>
  <script src="codef/codef_3d.js"></script>
 </head>
 <body>
  <div id="main"></div>
  <script> init();</script>
 </body>
</HTML>
```

Cette fonction init() se compose de différentes étapes :

- Déclaration de la zone d'affichage 'canvas' ;
- Initialiser les effets 3D ;
- Préparation des faces ;
- Exécuter l'animation.

```
function init()
{
    mycanvas=new canvas(640,480,"main");
    my3d=new codef3D(mycanvas, 900, 40, 6, 1600 );
    my3d.faces(myobjvert,myobj, true, true );
    go();
}
```

Etape 4 : L'animation

Après avoir déclaré l'ensemble des objets et la zone d'affichage, nous animons celui-ci, en définissant des valeurs différentes en X, Y et Z pour déterminer la position.

```
function go(){
mycanvas.fill('#000000');
my3d.group.rotation.x+=0.01;
my3d.group.rotation.y+=0.03;
my3d.group.rotation.z+=0.06;
my3d.draw();
requestAnimFrame( go );
}
```

Chaque image sera définie comme une frame (image pleine) et les transitions seront animées par la fonction RequestAnimFrame.
Cette fonctionnalité s'appuie sur le framework. Il permet ainsi d'être compatible avec les différentes versions des navigateurs du marché Firefox, Chome, Opera, Edge, Safari.

Communauté

Les acteurs adeptes de ces techniques sont souvent des anciens démomakers qui retrouvent de nouvelles sensations avec le Web d'aujourd'hui. Cependant la nouvelle génération de développeurs a pu s'en inspirer pour reproduire ces effets d'animations
Par ailleurs, les coding parties permettent de rassembler les différentes communautés. Tout d'abord l'ancienne génération qui a développé en Assembleur, Pascal, C... Et les nouveaux qui connaissent l'Internet, le HTML 5.

Plus loin

Il est possible de retrouver de nombreuses animations et réalisations sur différents sites Internet.
Pouet : http:///www.pouet.net/
Animations en 1 Ko : http://js1k.com
Réalisations avec le framework Codef : http://www.codef.
Librairie Three.js : http://threejs.org
Les scripts d'animations présentés dans l'article sont disponibles sur le site du magazine, les sources originales proviennent du Framework Codef.

Conclusion

Ces réalisations sont souvent de petits exploits, car elles permettent de mesurer le niveau de maîtrise du langage HTML 5 / CSS / JS. Ces personnes peuvent être retrouvées dans les équipes qui réalisent des jeux ou des animations pour Internet.

Débuter avec Visual Studio Online

Depuis 2012, la version « Cloud » de Team Foundation Server, initialement baptisée Team Foundation Service, permet à tous les développeurs de s'affranchir du maintien de l'infrastructure d'un serveur local. C'est en 2013 que l'offre passe en production et assure ainsi un taux de disponibilité de 99,9%. Cette plateforme permet ainsi à toutes les équipes de développement d'accéder facilement et rapidement à un outil d'ALM (Application Lifecycle Management) complet et adaptable à vos besoins.

Vivien FABING
Consultant ALM
Jérémy LANDON
Consultant et MVP ALM
INFINITE SQUARE

Création du compte

Visual Studio Online est une offre en mode SAAS, ainsi le commencement se fera entièrement en ligne. L'avantage ici est que tout se fera extrêmement facilement, il suffira de s'armer d'une adresse Microsoft et de 5 minutes. Allez sur visualstudio.com -> produit -> Visual Studio Online -> Inscrivez-vous. Il sera demandé de se connecter avec une adresse email Microsoft, et à ce moment-ci laissez-vous guider. Fig.1. La souscription Visual Studio Online est finalisée. Il est temps de passer à son utilisation.

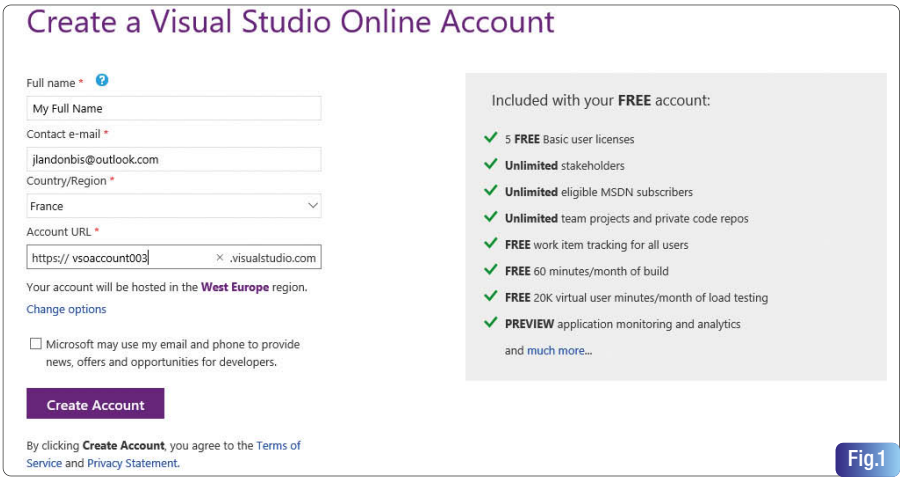


Fig.1

Création du premier Team Project

Dans Team Foundation Server et ainsi Visual Studio Online, tout commence par un Team Project. Un Team Project est un conteneur rassemblant les diverses briques qui seront détaillées plus tard : Contrôle de source, Gestion de projet, Build, Test...
Comme la souscription vient d'être créée, Visual Studio Online propose de suite de créer son premier Team Project. A la différence de Team Foundation Server où nous devons passer par le client Team Explorer, ici tout se fera en ligne. Lors de la création d'un projet, 3 informations seront demandées :

- Le nom du projet : aucune surprise de ce côté, il est tout de même à noter qu'il pourra facilement être changé après coups.
- Le process Template : au nombre de 3 (Agile, CMMI, Scrum) ils vont permettre de guider le projet dans une méthodologie au choix. C'est un choix important car contrairement au nom du projet, le process template ne pourra pas être modifié par la suite. Chacun de ces process possède de nombreuses subtilités qui s'adapteront ou non à votre équipe. Mais pour faire très simple nous pouvons les décrire ainsi :
  - Scrum : comme son nom l'indique, permet au mieux de se calquer à la méthodologie Agile Scrum. Microsoft apporte une attention particulière au fait que le process est adapté au Scrum défini par l'organisation Scrum (Scrum.org). L'estimation des besoins sera effectuée avec une métrique d'effort.
  - Agile : processus Agile qui permet de ne pas être limité par Scrum. Cette méthodologie permettra entre autres de pouvoir comparer le temps estimé et le temps complété pour une tâche donnée.
  - CMMI : à utiliser dans le cadre d'une gestion de projet plus « formelle ». CMMI a pour objectif de servir de base pour l'initiative d'amélioration des processus, et permettra au mieux de suivre chacune des décisions du projet.
- Le contrôle de sources : 2 sont présents :
  - TFVC (Team Foundation Version Control) : le contrôle de sources historique de Microsoft, il est centralisé et chaque modification devra obligatoirement être envoyée au serveur pour être archivée.
  - GIT : contrôle de sources décentralisé extrêmement puissant et souple, ce contrôle de source est le plus populaire actuellement. A noter que dans le cadre de Visual Studio Online, le repository GIT sera toujours privé et demandera des permissions pour y accéder. Avec les 5 utilisateurs gratuits fournis à chaque souscription, Visual Studio Online permet de manière déguisée de posséder gratuitement un nombre illimité de repositories GIT privés.

Ajouter des membres dans le projet d'équipe

Une fois le projet d'équipe créé, il est maintenant temps de rajouter les membres de votre équipe. Pour ce faire, il suffit de cliquer sur le lien de gestion des membres de l'équipe, présent sur la page d'accueil de votre nouveau projet d'équipe. Dans le cas d'une petite équipe, une simple adresse email suffit. Pour les utilisateurs d'Azure Active Directory, l'intégration avec Visual Studio Online vous demandera de déclarer les utilisateurs externes au préalable. À ce niveau, pas de rôle associé à un membre de l'équipe : ses différentes activités seront déclarées plus tard au niveau du Sprint Backlog. Pour un projet d'équipe qui serait développé par plus d'une seule équipe simultanément, il est possible d'ajouter au projet une équipe supplémentaire via l'interface d'administration de Visual Studio Online. Cette équipe pourra ainsi bénéficier de configurations personnelles des différents Rapports de suivi, Backlogs et autres Boards.

Gestion de projet

Pour un projet réalisé en équipe, une des étapes fondamentale reste celle de la définition des fonctionnalités à implémenter.
Très souvent, le **Product Backlog** est le point d'entrée principal du démarrage d'un projet. On y ajoute des User Stories en précisant leur titre et en les organisant par priorité via simple glisser-déposer. Pour avoir une vision plus macro du projet, il est possible de regrouper

nos User Stories par Features, voire de regrouper les Features par Epic (Note : la notion d'Epic n'est pas activée par défaut sur Visual Studio Online, et il suffira d'activer cette option dans les paramètres du Backlog, dans la section General > Backlogs. Une fois nos User Stories définies, il suffit de les glisser-déposer dans le premier **Sprint** pour pouvoir commencer la configuration de celui-ci. Si l'onglet Backlog de ce Sprint est très similaire au Product Backlog, on note néanmoins l'apparition d'un nouvel onglet **Capacity**. Ce dernier permet d'attribuer pour chaque membre du projet un ou plusieurs domaines d'activité (Développement, Test, Design, etc.) et pour chaque domaine, une capacité exprimée en heures par jour. Dans le cas d'une répartition des tâches au préalable, cette interface permettra d'obtenir simplement un aperçu de la charge globale par domaine d'activité (et ainsi prévoir éventuellement de réduire la charge de travail dans un domaine bien précis). À noter que cette fonctionnalité n'est utilisable qu'après avoir configuré les dates de début et de fin du Sprint. Fig.2 et 3. Le découpage des Users Stories en tâches peut s'effectuer aussi bien via l'onglet Backlog que via l'onglet **Board**. Ce dernier se veut être un remplacement efficace au bien connu « Tableau de post-it » des méthodes Agiles. D'un simple glisser-déposer, il est possible de passer l'état d'une tâche de l'état « To do » à « In Progress » puis « Done ». D'un simple clic il est possible de changer le nombre d'heures restantes pour terminer la tâche ou encore changer la personne responsable de la tâche.

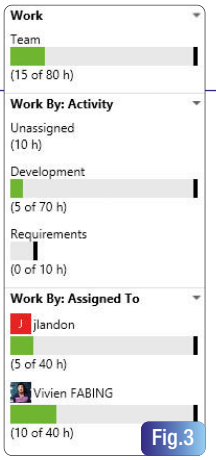


Fig.3

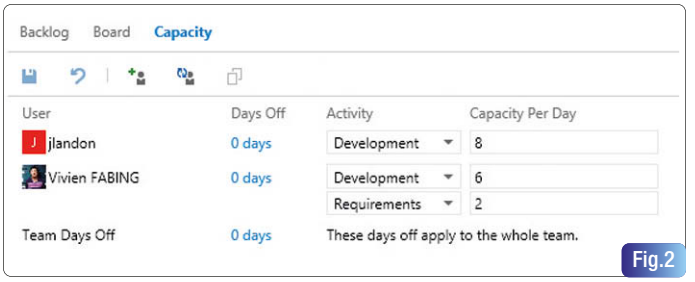


Fig.2

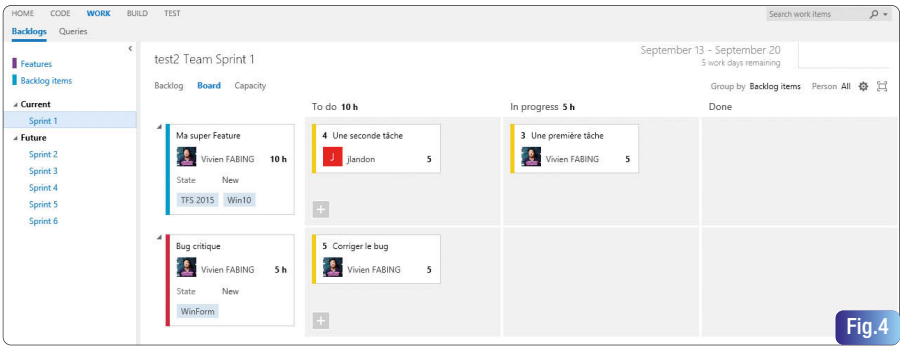


Fig.4

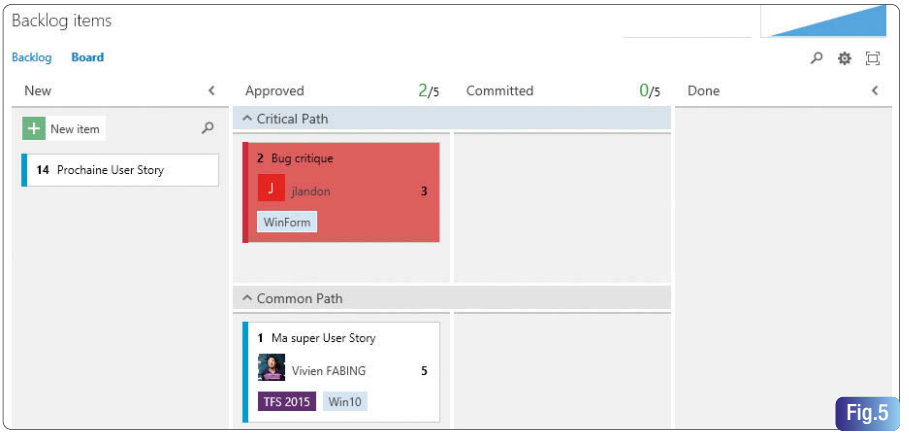


Fig.5

Ce tableau, toujours en évolution constante depuis le lancement de Visual Studio Online, est devenu de plus en plus paramétrable, de la sélection des informations à afficher, à la coloration des éléments ou tags répondant à certains critères. La Roadmap officielle de Microsoft nous laisse prévoir encore plus de customisation dans les mois à venir. Fig.4.

Pour des scénarios plus avancés, notamment dans le cas où l'on voudrait restreindre l'accès à certains Work Items, ou plus simplement pour classer nos Work Items par catégories afin de générer des rapports prenant en compte ces dernières, il est possible de configurer des **Area** (zones). On pourrait par exemple vouloir distinguer les bugs concernant des problèmes relatifs aux bases de données des bugs concernant l'interface utilisateur. La configuration de ces Area s'effectue via la partie administration de Visual Studio Online, ou directement via la page d'accueil du projet d'équipe. La page d'accueil du projet d'équipe est d'ailleurs l'un des moyens le plus facile d'obtenir un **tableau de bord** de suivi du projet. Chaque équipe peut alors décider des « tuiles » de suivi à afficher : état général des Bugs, évolution des Builds, exécution du plan de test, suivi du Sprint en cours, etc. Tout est disponible pour effectuer un suivi simple et efficace du projet en cours. Pour finir avec ce tour d'horizon des fonctionnalités de suivi de projet de Visual Studio Online, il est à noter que le tableau des gestions des User Stories, Features et Epic peut-être très facilement configuré de telle manière à fonctionner tel un tableau **Kanban**. On se retrouve ainsi à pouvoir définir les différentes étapes qui séparent l'état « To do » de l'état « Done », avec la possibilité de préciser une capacité pour chaque étape pouvant ainsi refléter la capacité de traitement de cette étape. Il est ainsi très facile de visualiser d'un simple coup d'œil sur ce tableau quelles étapes sont en sous effectifs et auraient besoin d'être renforcées, etc.

De plus, un récent ajout de la possibilité de préciser différents « Lane » (chemins) pour arriver à l'état final permet d'implémenter la bien connue « chaîne critique » des utilisateurs de la méthode Kanban. Fig.5.

Contrôle de source

Après la configuration de la partie gestion de projet, l'étape suivante de la mise en place d'un projet sous Visual Studio Online consiste bien entendu à bien paramétrer le **contrôle de code source**. Que l'on ait choisi le classique TFVC (Team Foundation Version Control) ou le plus récent Git, l'ajout de code source se fait en général via l'IDE Microsoft Visual Studio. Dans le cas de TFVC il faudra configurer un premier Workspace alors que pour Git il sera question de clôner un Repository. Dans les deux cas, les fonctionnalités de Visual Studio Online vous permettront d'effectuer des actions de base de tout contrôle de code source qui se respecte, à savoir visualiser l'historique, comparer des fichiers, comprendre les modifications grâce à la forte traçabilité due à la liaison aux Work Items, etc. Outre ces manipulations standards,

il est également possible via Visual Studio Online d'éditer les fichiers de code source directement depuis l'interface Web ! Cette fonctionnalité bien que relativement puissante (elle bénéficie tout de même d'auto-complétion partielle, de la possibilité de créer un fichier, etc.) est à réserver à un usage plutôt ponctuel (modification d'un fichier de configuration, changement d'une variable dans un fichier de Script, etc.).

Un système de **revue de code** assez simple est également disponible directement depuis l'interface Web. Il suffit de partager par email ou tout autre moyen de communication l'adresse Web des changements à revoir pour permettre au destinataire d'apporter un commentaire global, sur un fichier, voire même dans le code source.

Dernière fonctionnalité à connaître pour bien démarrer avec l'utilisation de Git en tant que contrôleur de code source : le **Pull-Request**. En effet cette fonctionnalité bien connue des utilisateurs de Github et autres repositories Web de Git est également disponible depuis l'interface Web de Visual Studio Online. Le principe étant toujours le même : faire valider ses modifications avant de pouvoir merger dans la branche Master du repository. Fig.6.

Build

Développer une application et la gérer c'est bien, mais il faudra à un moment la déployer ou simplement la générer souvent pour pouvoir la tester au cours de son développement. Pour faire ceci, il faudra passer par un serveur de builds, son rôle sera très simple : remplacer le travail que ferait normalement une ou plusieurs ressources au travers de différentes tâches. Exemple : générer l'application pour vérifier les archivages partiels, jouer les tests unitaires/intégration pour simplement vérifier la qualité, déployer l'application... Il existe de nombreuses solutions de build sur le marché, pour les plus connus nous pouvons citer : Jenkins, TeamCity ou encore CruiseControl. Visual Studio Online propose ses propres systèmes de build qui offrent l'avantage d'être en totale synergie avec le reste des modules. Visual Studio Online propose deux systèmes de builds :

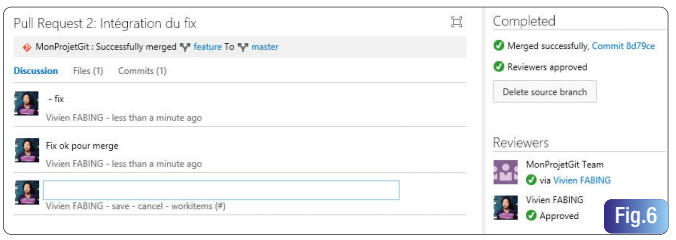
- Les builds 1.0 : build historique de Team Foundation Server, celles-ci basent la réalisation du workflow à travers du Workflow Foundation. Elle est très puissante et doit être éditée en local puis archivée sur Visual Studio Online pour être prise en compte. Entièrement personnalisées et évolutives, de nombreuses activités sont présentes de base (exemple : récupérer les sources, lancer msbuild, placer du conditionnel (if, switch)...) mais il est tout à fait possible de créer ses propres activités à travers du code C#. Même si de nombreuses activités sont présentes de base, il est tout de même parfois important d'utiliser les « Community TFS Build Extension » présentes sur CodePlex qui offrent bon nombre d'activités utiles (exemple : transfert FTP, zip, vérification via StyleCop...) Fig.7.
- Les builds 2.0 (ou build VNext) : nouveau système de build, il offre l'avantage d'être extrêmement simple à utiliser et surtout cross plateforme. Son édition est entièrement réalisable en ligne, et il se présente comme un ordonnanceur de tâches. Cross Plateforme signifie qu'il sera possible contrairement aux builds 1.0 de générer des applications à destination sur Windows, Linux et OSX. Fig.8.

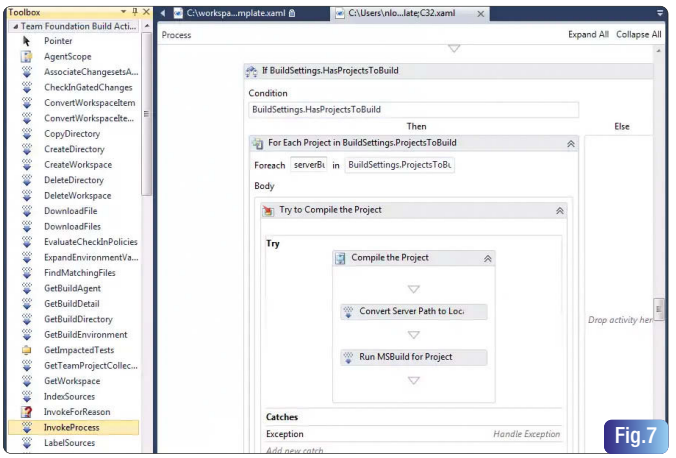
Chaque système de build demande d'avoir un serveur de builds sur lequel vont s'exécuter les actions. Microsoft propose de s'occuper de cette partie en mettant à disposition un serveur de builds à la demande. Le système Hosted permet de ne se soucier que de la partie Workflow, lors de l'exécution Visual Studio Online va mettre à disposition de manière automatique et pour la durée de la machine un serveur de builds provisionné dans Azure.

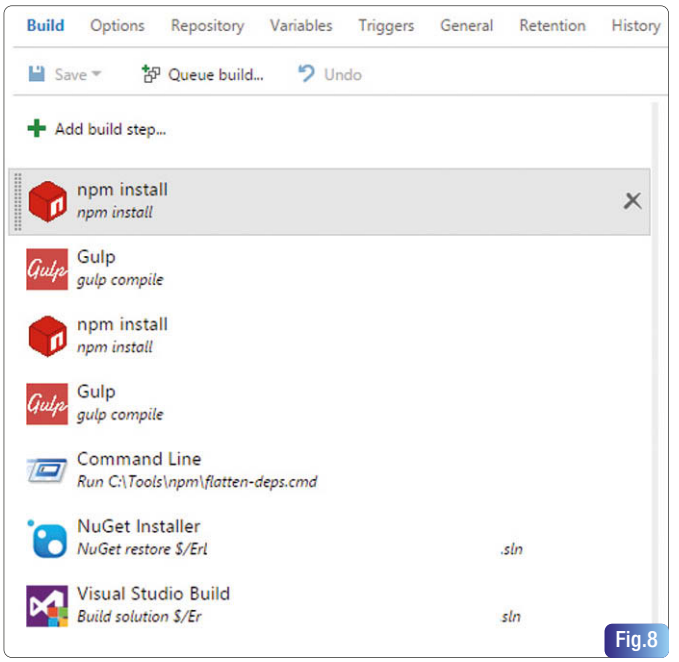
Test

Côté **test**, Visual Studio Online permet un démarrage en douceur via son interface Web. Il est ainsi possible de composer un plan de test directement depuis l'interface Web, puis de décrire des cas de tests pour vérifier une des fonctionnalités de la solution. Il est ensuite possible d'exécuter ces cas de tests toujours depuis l'interface Web, ou même d'exécuter le client lourd Microsoft Test Runner (fourni avec l'installation de Visual Studio Enterprise ou Test Professional) qui permet de collecter automatiquement de nombreuses informations concernant le contexte d'exécution du cas de test (les actions effectuées, l'utilisation du système, les détails de l'environnement, etc.). Fig.9. Pour les heureux détenteurs d'une licence Visual Studio Enterprise avec MSDN, il est également possible d'exécuter de manière très simple un **test de charge dans le Cloud** assez basique directement depuis le site Web de Visual Studio Online.

L'onglet Load test est disponible au niveau de la page d'accueil de votre site Visual Studio Online et vous permet de configurer en







quelques clics un scénario de test de charge. Les résultats du test sont ensuite visibles depuis le site Web. Fig.10.

L'ouverture

Microsoft s'ouvre de plus en plus vers les autres systèmes et solutions. Le nouveau système de builds 2.0 offre déjà la possibilité de s'entendre sur des systèmes tiers. Microsoft pousse l'ouverture plus loin en offrant la possibilité de se connecter avec d'autres solutions du marché à travers les Services Hooks. Personnalisables pour chaque projet d'équipe les services hooks sont éditables dans les configurations du projet d'équipe. Il sera alors possible par exemple de passer par la solution Jenkins et de déclencher une build à chaque archivage sur le contrôle de source. Ou encore il sera possible de se mapper sur un projet Trello afin de profiter d'une synergie entre les tâches Trello et les work items présents dans le projet Visual Studio Online. Certains services sont présents de base, mais les services hook proposent aussi via les Web Hooks de s'abonner à divers évènements : build complété, code archivé, work item modifié, message posté sur la Team Room (centre de communication au sein du projet) se produisant sur le projet d'équipe afin de pouvoir créer ses propres outils liés à Visual Studio Online. Fig.11. Visual Studio Online offre aussi la possibilité de personnaliser la plateforme à travers des extensions. L'avantage ici est que ces extensions sont présentes dans un market. Quelques extensions sont présentes actuellement, la plus notable étant le calendrier permettant d'avoir une vue sur le planning général du projet en incluant les données de ressources précisé dans le projet d'équipe. Fig.12.

Gérer son budget

Visual Studio Online peut être entièrement gratuit, mais certaines fonctionnalités sont payantes si elles sont utilisées :

- Ajouter des intervenants à la souscription, ici 3 possibilités :
  - L'intervenant possède une souscription MSDN, dans ce cas cet utilisateur est totalement gratuit.
  - L'intervenant n'a besoin d'accéder qu'au backlog afin d'avoir une vision sur la gestion du projet. Dans ce cas l'utilisateur est gratuit et se verra affecter le rôle de Stakeholder.
  - L'intervenant ne possède pas de souscription MSDN et souhaite participer au développement du projet ou réaliser des tests, Microsoft propose 2 licences :
    - Basic : offre l'accès au contrôle de source, au backlog, la build ;
    - Professional : basic + une licence mensuelle Visual Studio Professionnal ;
    - Advanced : basic + une licence mensuelle Visual Studio Test Professionnal + la possibilité d'accéder à la gestion des tests.
- La build hosted : facturé à la minute de build (60 min offertes).
- Les tests de charges dans le Cloud : facturés à la minute utilisateur (20 000 minutes utilisateurs offertes).

Conclusion

Outre un avantage au niveau infrastructure, Visual Studio Online est également une plateforme qui permet de bénéficier des toutes dernières mises à jour en avance de phase des releases annuelles de Team Foundation Server. Si la plupart des fonctionnalités présentées dans cet article sont d'ores et déjà disponibles dans la version 2015, quelques-une d'entre elles ne le sont pas encore et seront l'objet d'une future Update 2 comme l'on peut le voir dans le planning officiel de Microsoft (https://www.visualstudio.com/en-us/news/release-archive-vso)

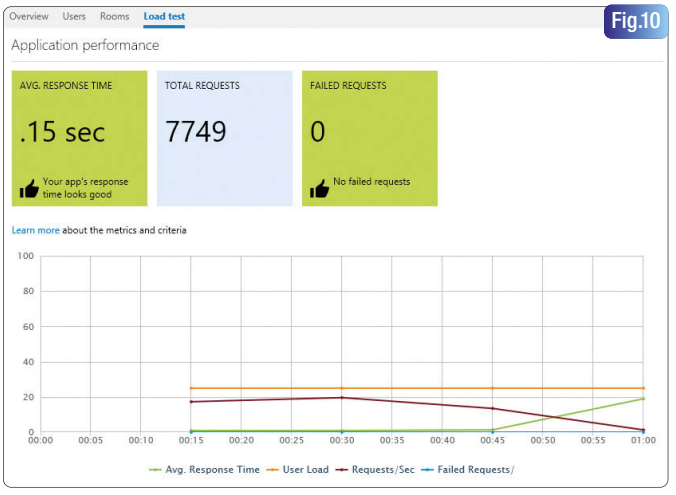


Fig.10

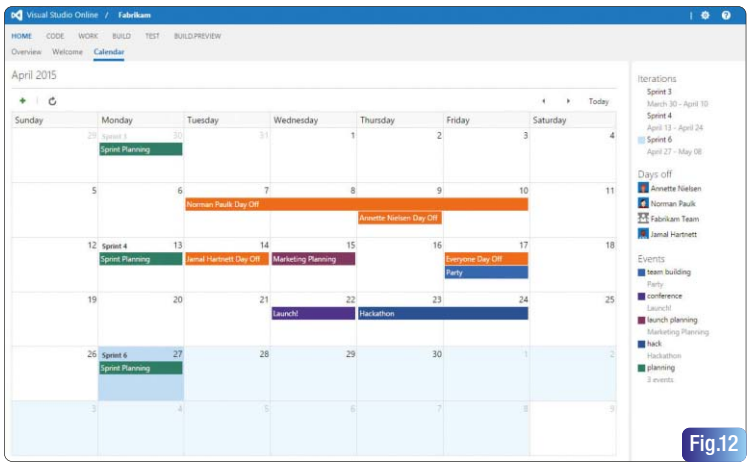


Fig.12

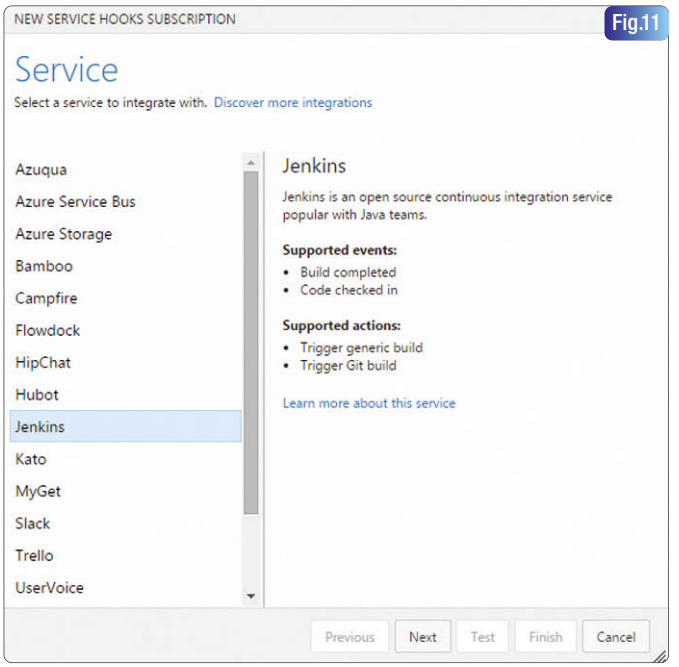


Fig.11

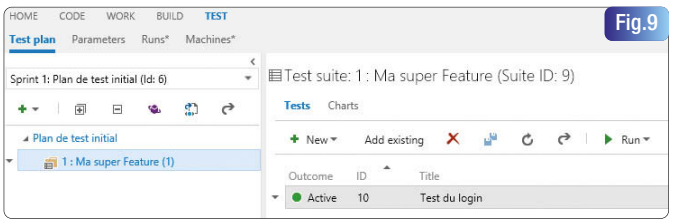


Fig.9

L'INFORMATICIEN + PROGRAMMEZ
versions numériques

PDF

OFFRE SPÉCIALE DE COUPLAGE

2 magazines mensuels,
22 parutions / an
+ accès aux archives PDF

PRIX NORMAL POUR UN AN : 69 €
POUR VOUS : 49 € SEULEMENT*

Souscription sur www.programmez.com

* Prix TTC incluant 1,01€ de TVA (à 2,10%).

Construire et coder son imprimante 3D

Vous entendez partout parler de l'impression 3D et souhaitez vous lancer dans l'aventure ? Toutefois vous ne savez pas par où commencer et souhaitez surtout commencer d'une manière différente avec les mains dans les fils. Vous voulez monter votre machine, définir ses dimensions et la programmer grâce à des outils et des logiciels disponibles sur le net ? Suivez le guide...

Mike FAHRASMANE
Titulaire d'un master en génie électrique et acteur du mouvement maker et DIY (Do-It-Yourself) passionné d'innovation. Il est le fondateur de GEEKOFYOU (http://www.geekofyou.fr) société de formation en impression 3D FDM (dépôt de fil fondu) et nouvelles technologies.
www.geekofyou.fr | contact@geekofyou.fr


Fig.1


Fig.2


Fig.3

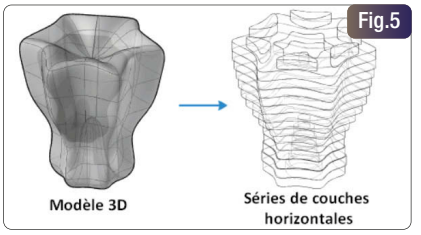


Fig.5

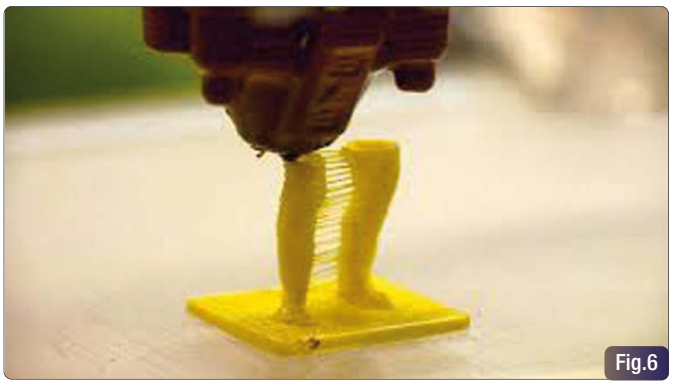


Fig.6

Depuis quelques années, l'impression 3D ne cesse de faire parler d'elle. Que ce soit au niveau des médias (télévision, radio, journaux...), les réseaux sociaux ou même le bouche-à-oreille, ça discute partout de cette technologie et des dernières nouveautés dans le secteur.
Il faut savoir que c'est une technologie qui existe depuis 30 ans, qui n'est donc plus toute jeune ; elle a été créée au départ pour la conception de prototypes et maintenant pour divers types d'objets: **Fig.1**.
Depuis ces débuts, différents types de technologies ont vu le jour permettant toujours de réaliser des objets par superposition de couches (plastique, céramique, verre, nylon, métal...).
Les différents types de techniques étant :

- Le **SLA** (stereolithographie) utilisant de la résine liquide avec une bonne précision et une solidité qui ne se dément plus **Fig.2**.
- La techno **SLS** (frittage sélectif par laser) qui utilise une poudre polymère réagissant au laser en se durcissant, toujours dans le principe de la superposition des couches. Elle est très précise (de l'ordre de quelques microns), est multicolore à l'impression si besoin et multi-matériaux (titane, plastique, plomb...).

Les technologies SLS et SLA ont la particularité d'avoir une très bonne précision et finition. Le seul point noir étant le prix des machines : plusieurs dizaines de milliers d'euros la plupart du temps. Ceci dit, nous nous intéresserons à un autre type de technologie dite FDM.
Qu'est-ce que la FDM ? C'est tout simplement l'impression 3D la plus répandue et celle qui fait le plus parler dans les médias. C'est un peu grâce à ce principe si l'impression 3D est aussi connue de nos jours.
FDM, voulant dire "Fused Deposition Modeling", consiste à faire fondre du fil (Filament PLA ou ABS majoritairement) grâce à une cartouche chauffante et à entraîner le plastique fondu à travers une buse de quelques microns (0,5; 0,4; 0,3 mm) pour empiler les couches les unes au-dessus des autres **Fig.3**. Ce qui revient à nous poser la question: Comment fonctionne une imprimante FDM? **Fig.4**
Le principe de l'impression 3D est l'empilement de couches de matière en commençant par la base **Fig.5**. Un bon exemple peut-être celle de la construction d'une maison en commençant par les fondations puis en empilant brique par brique les différentes couches jusqu'à arriver à la toiture.
L'imprimante grâce à ses moteurs pas-à-pas déploiera la buse sur les axes X et Y afin de créer la première couche (les fondations) de notre objet sur le plateau d'impression **Fig.6**. Celle-ci remplit et finit la 1ere couche), la tête d'impression se soulèvera alors sur l'axe z de quelques dizaines de micromètres « ¦µm'' pour réaliser la seconde couche et se déplacer aussi sur X et Y.

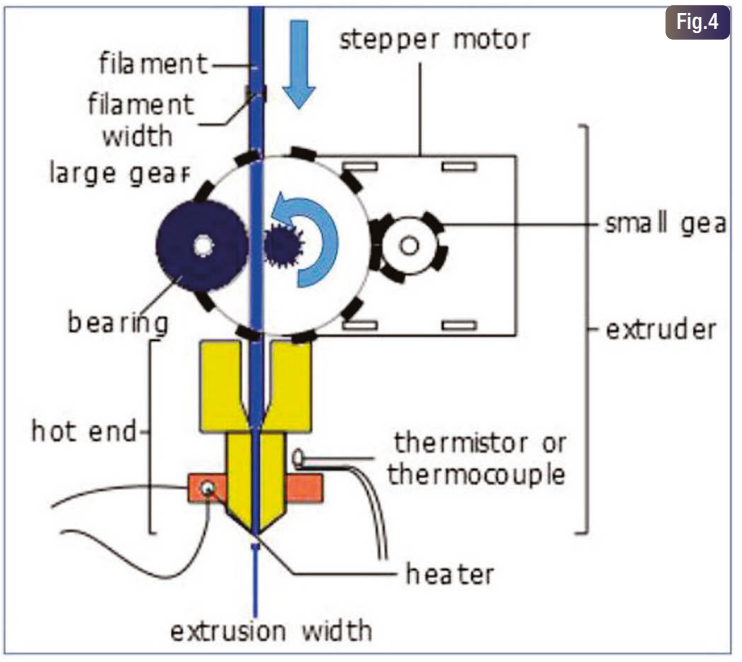


Fig.4

Les différents types de technologies d'imprimantes à fil fondu

Parmi les technologies d'imprimantes à fil fondu, il existe différentes formes d'imprimantes sur le marché :

- Type **cartésienne** Fig.7 ;
- Type **delta** Fig.8.

La plupart des imprimantes open source sont répertoriées sur le site www.reprap.org, le site étant pionnier pour la construction et le partage autour des imprimantes 3D FDM open source et open-hardware.
L'imprimante cartésienne a la particularité de déplacer sa tête chauffante de façon linéaire sur les axes X, Y et Z pour le dépôt de filament chaud Fig.9. Son boîtier étant généralement de forme cubique.
L'imprimante delta, quant à elle, a une forme de boîtier plus cylindrique. Elle utilise le principe du "Pick and place", un peu comme les robots bras manipulateurs sur les chaînes de montage pour le déplacement de sa tête chauffante. De façon mathématique, on parlera de coordonnées polaires.

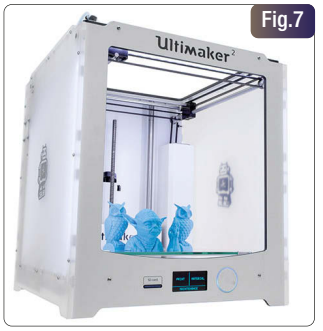


Fig.7


Fig.9

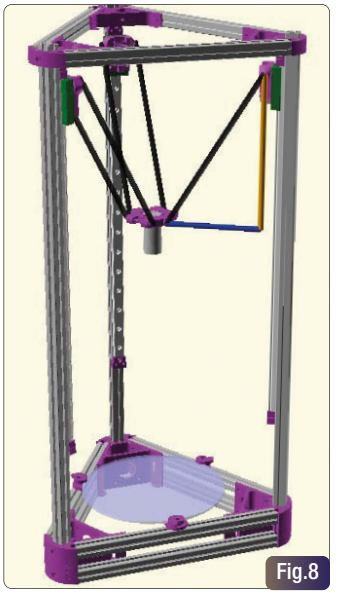


Fig.8

L'imprimante delta

Nous nous consacrerons principalement à l'étude de l'imprimante delta. Elle a la particularité de ne pas déplacer ses objets durant l'impression (moins de variation de température pour la pièce), un déplacement plus rapide de la tête d'impression (moteur d'extrusion déporté sur le boîtier de la machine) grâce à une tête plus légère Fig.10 et 19.
Contrairement aux imprimantes cartésiennes qui utilisent généralement 4 moteurs pas-à-pas pour les déplacements de la tête incluant 2 pour le déplacement de l'axe Z, la delta utilise 3 moteurs pour les déplacements de la tête d'impression. Chacun représentant les axes X, Y et Z.
Les différents moteurs sont au nombre de quatre (X, Y, Z et l'extrudeur) ; l'extrudeur servant à tirer le fil et l'aiguiller dans un tube « PTFE » vers la résistance chauffante et la buse, supportées elles-mêmes par des biellettes Fig.11.
Ces biellettes sont déplacées indépendamment de façon à garder la tête d'impression parallèle au plateau grâce aux moteurs X, Y et Z. Ces moteurs pilotent des courroies qui sont elles aussi reliées aux biellettes Fig.12.
Le cerveau de la machine est une carte Arduino 2560 dans laquelle est flashé le firmware (Repetier firmware, Marlin, Sprinter...) Fig.13.


Fig.10


Fig.11

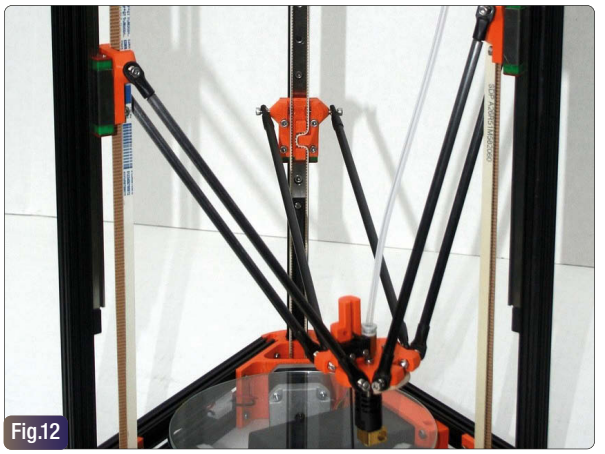


Fig.12

Dimensionner son imprimante 3D

L'objectif ici est de monter son imprimante et de la configurer côté logiciel.
Pour cela, nous allons travailler avec une imprimante de type delta en Kit. Le plus de cet article est que nous allons pouvoir dimensionner le volume d'impression de notre machine et par la même occasion définir ses dimensions extérieures.
L'imprimante en kit que nous utiliserons est la « **Kossel** » Fig.14.
Les différentes parties qui la composent sont :

- La partie électronique ;
- La partie chauffante ;
- La partie mécanique ;
- Le châssis ;
- Les parties imprimées en 3D.

La partie mécanique se compose :

- Des rails ;
- De moteurs pas-à-pas ; Fig.16
- De biellettes ;
- D'un moteur pour l'extrudeur ;
- De vis ;

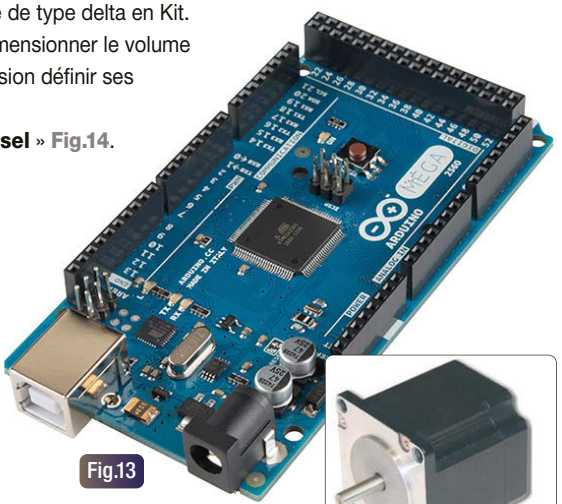


Fig.13

Fig.16


Fig.14


Fig.15

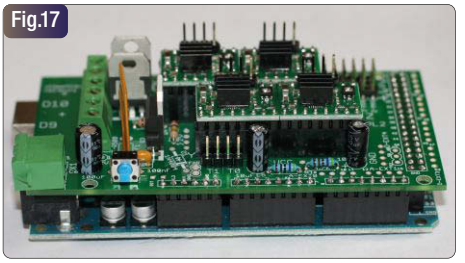


Fig.17

- D'écrous ;
- De rondelles ;
- De profilés aluminium (**Fig.15).**

La partie électronique contient :

- Une carte Arduino 2560 (que l'on ne présente plus, ;-)) ;
- Une carte de puissance de type "RAMPS" venant se brancher sur l'Arduino pour piloter les différents moteurs. **Fig.17 ;**
- 4 drivers pour contrôler les différents moteurs ;
- Des câbles pour les connecter aux moteurs ;
- Une alimentation de 120 Watts ;
- 3 endstops **Fig.18.**

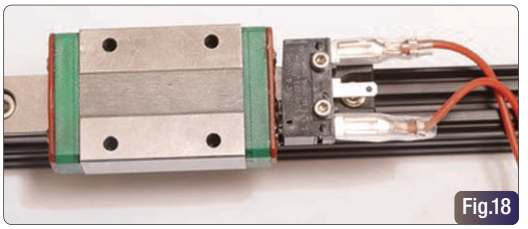


Fig.18


Fig.19

La partie chauffante est composée de :

- Une résistance chauffante ;
- Une buse ;
- Un kit J-Head. **Fig.19 ;**
- Un tube PTFE ;
- Une sonde de température.

Les parties imprimées en 3D composées de :

- Les parties imprimées en 3D servant de logement aux moteurs pas-à-pas. **Fig.20 ;**
- La pièce servant à supporter le kit J-Head ;
- Les pièces 3D servant à brancher les rails entre eux.


Fig.20

Il vous faudra pour monter la machine :

- Des tournevis plats et cruciformes ;
- Des clés allen ;
- Des clés plates ;
- Une pince plate ;
- Une pince coupante.

Pour le montage de la Kossel, toutes les étapes du montage de la machine ainsi que tous les détails évoqués sur les différentes parties de celle-ci sont sur le site : http://reprap.org/wiki/Kossel

Le but dans cette partie de l'article est de pouvoir maîtriser les outils permettant de redimensionner le volume d'impression de l'imprimante à notre guise.

Nous évoquons là le diamètre et la hauteur d'impression, parce que le volume d'impression de la delta est un cylindre comparé à la cartésienne qui est un cube.

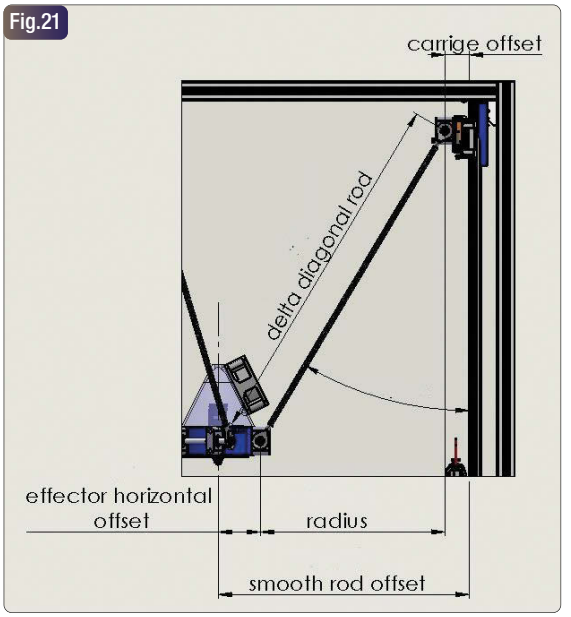


Fig.21

Outils pour la gestion des dimensions de la machine

Présentation des dimensions de la machine

L'imprimante 3D de type delta est une machine avec un volume d'impression de forme cylindrique avec des biellettes comme support pour la tête d'impression.

Nous allons vous montrer dans les images à venir les différentes valeurs nécessaires à la configuration de la machine et les différents décalages à prendre en compte en configurant la machine (**Fig.21).**

Il est aussi important de savoir situer le point de départ de notre tête d'impression sur notre plateau d'impression. Nous entendons par là les points (X, Y) = (0,0) et le sens d'évolution positif et négatif de ceux-ci (**Fig.22).**

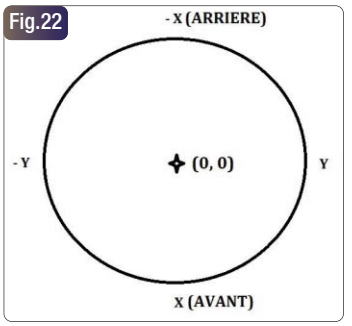


Fig.22

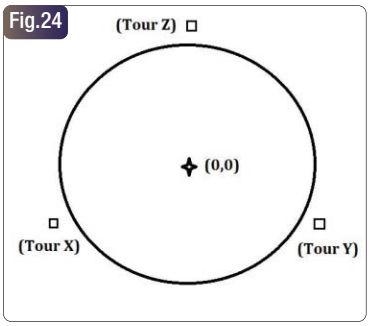


Fig.24

Le sens des coordonnées fixé, il est important de pouvoir bien placer les différents piliers autour de ce plateau d'impression à 120° d'intervalle. Ces piliers (tours) servant aussi de boîtier à l'imprimante (**Fig.24).**

Gestion des dimensions de la machine

Maintenant que nous connaissons les différentes parties de la machine, il est maintenant temps de passer au calcul des parties que nous souhaitons pour notre machine. Pour cela, nous pouvons télécharger un

tableau Excel où nous rentrerons les différentes valeurs ou via une représentation graphique de notre imprimante grâce au fichier "openscad**" kossel_1515** (le logiciel Openscad) que vous pouvez télécharger sur le site : www.openscad.org. **Fig.25, Fig.26 et Fig.27**
Le dossier contenant les fichiers Excel et openscad est téléchargeable via le lien : https://github.com/Jaydmdigital/mk_visual_calc
Nous pouvons reconnaître différentes valeurs sur l'image et le tableau Excel. **Fig.21, Fig.26 et Fig.27**
Parmi ces valeurs, nous rentrerons les valeurs pour l'input :

- Longueur du rail horizontal ;
- Longueur du rail vertical ;
- Longueur du kit d'extrusion ;
- Offset du carriage au milieu du rail ;
- Offset du support du kit d'extrusion au milieu du kit d'extrusion ;
- Epaisseur du plateau d'impression ;
- Angle minimum quand l'extrudeur touche l'autre bout du plateau.

Nous retrouvons en sortie les valeurs suivantes vues à l'image (**Fig.28**) :

- delta radius ;
- delta smooth rod offset ;
- delta diagonal rod ;
- delta rod angle ;
- delta vertical length ;
- print height ;
- build surface diamètre ;
- delta rod angle.

Notons bien que les valeurs auxquelles on doit couper les rails :

- verticaux : 600 mm ;
- horizontaux : 240 mm.

Afin d'avoir un **diamètre d'impression** de 16,9 cm et une **hauteur d'impression** de 24,4 cm.

Programmation de la carte

Maintenant que nous avons récupéré les valeurs nécessaires aux différentes mesures de notre imprimante, nous pouvons maintenant entrer ces valeurs dans le firmware et les charger sur notre carte Arduino.
Mais avant tout cela, nous devons télécharger le logiciel "**Repetier Host**" qui est une interface pour la configuration de l'imprimante et "**Repetier Firmware**" le programme que l'on téléverse dans la carte électronique (Arduino + ramps). Le site : www.repetier.com puis l'onglet "Download". De là, vous pourrez télécharger "Repetier Host" (version Windows). Ensuite, cliquez sur l'onglet Repetier firmware puis téléchargez (version Windows **Fig.23 et 29**). Ceci fait, vous devez télécharger la dernière version de l'IDE Arduino.

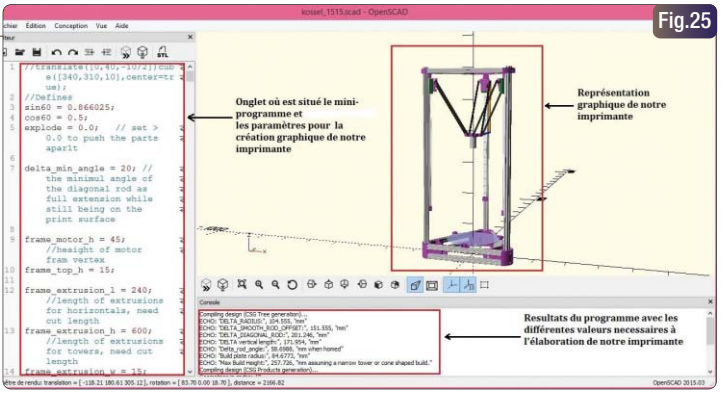


Fig.25

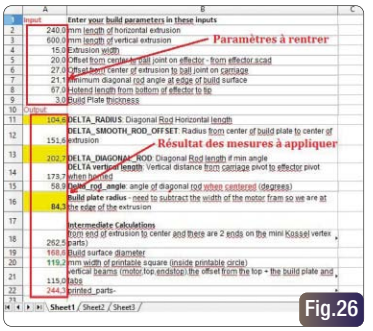


Fig.26

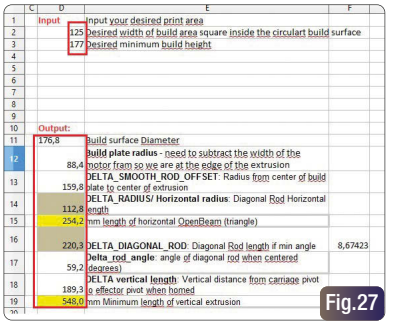


Fig.27

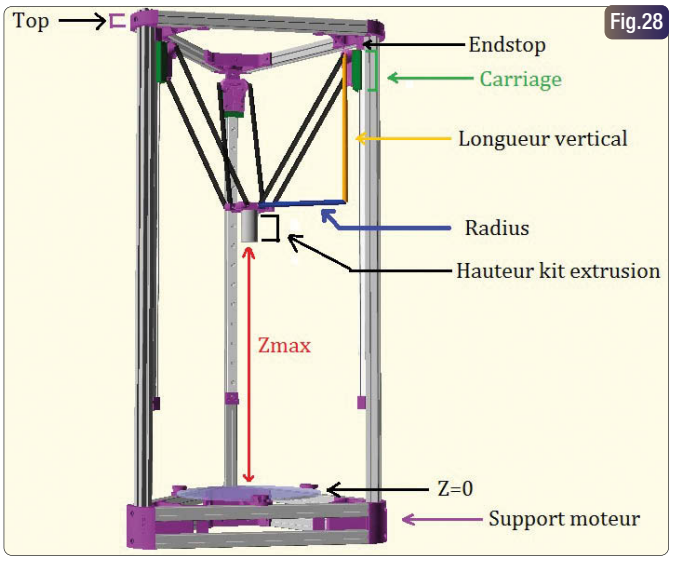


Fig.28

Fig.29

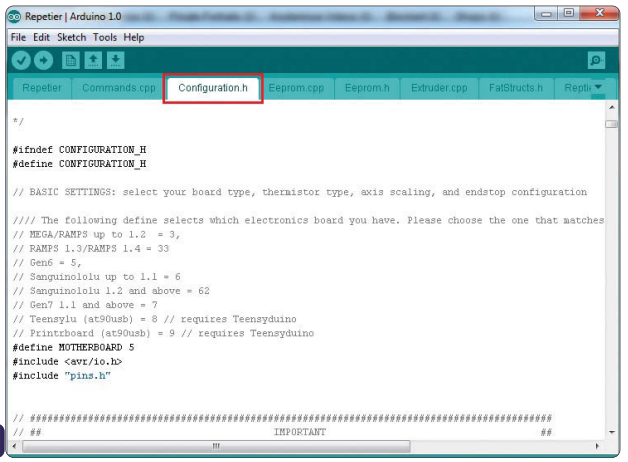


Fig.30

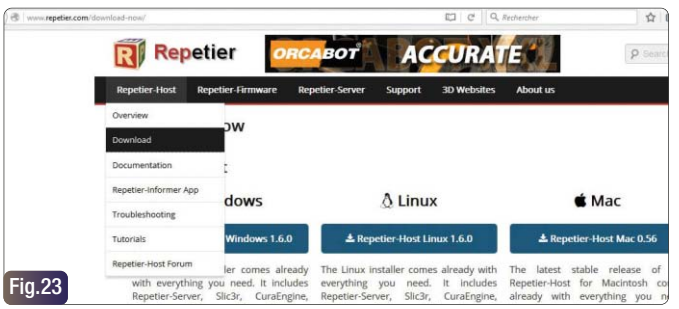


Fig.23

L'IDE Arduino est disponible sur le site : https://www.arduino.cc/en/Main/Software. De nombreux tutoriels sont disponibles sur le site pour ceux n'ayant pas encore eu l'occasion d'utiliser une carte Arduino auparavant.
Décompresser le fichier "Repetier firmware" que vous venez de télécharger et cliquez sur l'icône "configuration.h". La fenêtre de l'IDE Arduino s'ouvrira. Cliquez alors sur l'onglet "Configuration.h" (**Fig.30).**
La prochaine étape est la configuration de certaines parties du programme de façon à configurer l'imprimante à notre sauce en faisant un petit tour sur le fameux tableau Excel. **Fig.26 et 27.**
Grâce au raccourci "Ctrl + F", cherchez dans le programme les variables suivantes pour y mettre les valeurs suivantes :

- #define NUM_EXTRUDER 1 // 1 représentant le nombre d'extrudeurs de la machine ;
- #define MOTHERBOARD 33 // 33 qui représente la carte électronique RAMPS 1.4 ;
- #define DRIVE_SYSTEM == 3 // 3 pour choisir une imprimante de type delta (Rostock,Kossel…) ;
- #define INVERT_X_DIR 0 ;
- #define INVERT_Y_DIR 0 ;
- #define INVERT_Z_DIR 0 ;
- #define X_HOME_DIR -1 // Position HOME située en haut de l'imprimante ;
- #define Y_HOME_DIR -1 ;
- #define Z_HOME_DIR 1 ;
- #define ENDSTOP_X_BACK_MOVE 5 // Après avoir atteint la position HOME, le moteur recule de 5mm ;
- #define ENDSTOP_Y_BACK_MOVE 5 ;
- #define ENDSTOP_Z_BACK_MOVE 5 ;
- #define ENDSTOP_X_RETEST_REDUCTION_FACTOR 3 // Réduit la vitesse des moteurs par trois en se plaçant en position "HOME" ;
- #define ENDSTOP_Y_RETEST_REDUCTION_FACTOR 3 ;
- #define ENDSTOP_Z_RETEST_REDUCTION_FACTOR 3 ;
- #define X_MAX_LENGTH 169 ;
- #define Y_MAX_LENGTH 169 ;
- #define Z_MAX_LENGTH 244 ;
- #define XAXIS_STEPS_PER_MM 98,4251 // Pas par mm pour le moteur X ;
- #define YAXIS_STEPS_PER_MM 98,4251 ;
- #define ZAXIS_STEPS_PER_MM 98,4251 ;
- #define EXT0_STEPS_PER_MM 370 // Pas par mm pour le moteur d'extrudeur ;
- #define ENDSTOP_PULLUP_X_MIN true ;
- #define ENDSTOP_PULLUP_X_MAX true ;
- #define ENDSTOP_PULLUP_Y_MIN true ;
- #define ENDSTOP_PULLUP_Y_MAX true ;
- #define ENDSTOP_X_MIN_INVERTING true ;
- #define ENDSTOP_X_MAX_INVERTING true ;
- #define ENDSTOP_Y_MIN_INVERTING true ;
- #define ENDSTOP_Y_MAX_INVERTING true ;
- #define MIN_HARDWARE_ENDSTOP_X true ;
- #define MAX_HARDWARE_ENDSTOP_X true ;
- #define MIN_HARDWARE_ENDSTOP_Y true ;
- #define MAX_HARDWARE_ENDSTOP_Y true ;
- #define DELTA_DIAGONAL_ROD 202,7 // Longueur de la biellette ;
- #define BAUDRATE 115200 // Vitesse transfert données ;
- #define EEPROM_MODE 0 // Pas de carte SD installé sur l'imprimante ;
- #define MAX_FEEDRATE_X 200 ;
- #define MAX_FEEDRATE_Y 200 ;
- #define MAX_FEEDRATE_Z 200 ;
- #define HOMING_FEEDRATE_X 40 ;
- #define HOMING_FEEDRATE_Y 40 ;
- #define HOMING_FEEDRATE_Z 2 ;
- #define DELTA_SEGMENTS_PER_SECOND_PRINT 200 // Précision des mouvements en impression ;
- #define DELTA_SEGMENTS_PER_SECOND_MOVE 70 // Précision des mouvement sans impression ;
- #define PRINTER_RADIUS 104,6 // Delta_smooth_rod, calculé sur le tableau Excel ;
- #define END_EFFECTOR_HORIZONTAL_OFFSET 20 // Décalage du chariot portant les biellettes ;
- #define CARRIAGE_HORIZONTAL_OFFSET 27 // Décalage effecteur extrudeur
- #define BELT_PITCH 2 // Courroie GT2, pas de 2mm ;
- #define PULLEY_TEETH 20 // Nombre de dents de la poulie ;
- #define PULLEY_DIAMETER 10 // Diamètre de la poulie en mm ;
- #define STEPS_PER_ROTATION 200 // Nombre de pas par rotation de la poulie ;
- #define MICRO_STEPS 16 // Nombre de micro pas du moteur pas-à-pas.

Tout ceci fait, vous devez uploader le firmware dans la carte de votre imprimante en branchant le port USB à l'ordinateur (**Fig.31).**

N.B : Pensez à installer le driver nécessaire afin de faire reconnaître la carte par l'ordinateur. Les détails de l'installation sont disponibles sur le site : https://www.arduino.cc/en/Guide/Windows

Calibration de la Kossel

Pour commencer notre calibration, nous évoquerons déjà les positions des différentes tours qui sont espacées de 120° (**Fig.32)** :

- Position tour X : 210°
- Position tour Y : 330°
- Position tour Z : 90°

Les points de calibrage (coordonnées X et Y) des différentes tours se calculent de la manière suivante :

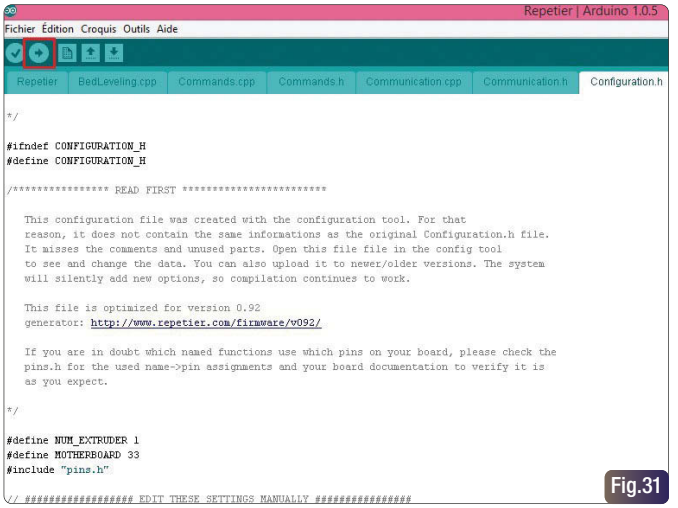


Fig.31

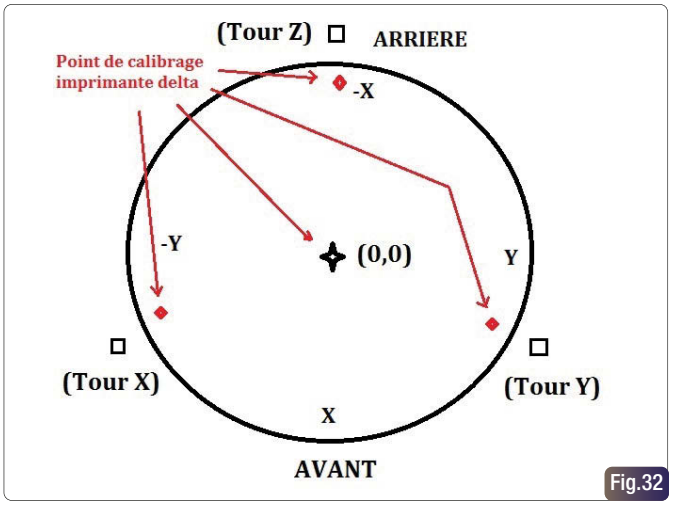


Fig.32

Tour X :

- X = Rayon (en mm) x cos (position tour X) = 88.cos (210) = -76,2
- Y = Rayon (en mm) x cos (position tour X) = 88.sin (210) = -44

Tour Y :

- Rayon (en mm) x cos (position tour Y) = 88.cos (330) = 76,2
- Rayon (en mm) x cos (position tour Y) = 88.cos (330) = - 44

Tour Z :

- Rayon (en mm) x cos (position tour Z) = 88.cos (90) = 0
- Rayon (en mm) x cos (position tour Z) = 88.cos (90) = 88

Maintenant que nous avons en notre possession les points de calibration des différentes tours, nous pouvons connecter l'imprimante à l'ordinateur via un câble USB. Pour cela, rendez-vous dans : Panneau de configuration -> gestionnaire de périphériques -> "Ports (COM et LPT)" afin de vérifier sur quel port est connecté notre machine. Ouvrez Repetier Host, dirigez-vous vers l'onglet "configuration" puis "Réglages imprimantes" (**Fig.33).**

Choisissez le port COM trouvé dans votre gestionnaire de périphériques et cliquez "OK".

Toutes les mesures faites, vous devez procéder à la calibration de la machine. Ceci permettra de déterminer la planéité du plateau d'impression. Pour cela, nous devrons calibrer le Zmax (la distance que l'on a calculé auparavant Fig.28) à 4 points différents du plateau. Un calibrage au point (X,Y)=(0,0) et à 3 autres points différents qui se situent devant chaque tour (X, Y et Z). **Fig.32.**

Nous utiliserons dans cette partie de l'article du **Gcode** afin de déplacer la tête d'impression à l'endroit voulu. Pour cela, vous devrez vous rendre dans "Control manuel" puis Gcode dans l'interface Repetier Host (**Fig.34).**

Nous devons entrer les valeurs des différentes tours (X, Y, Z) ainsi que le centre du plateau dans la partie Gcode et l'envoyer. **Fig.34 et 35**

Pour la tour X, elle prendra cette forme : **G1 X-76,2 Y-44 Z15 F7000**

Définition :

- G1: Représente un déplacement linéaire de notre extrudeur ;
- X, Y : Ce sont les coordonnées sur les axes X et Y en mm ;
- Z: La coordonnée sur Z, en l'occurrence la buse sera située à 15 mm de notre plateau en fin de mouvement ;
- F7000 : représente la vitesse de déplacement en mm/min, ce qui équivaut à 116 mm/s.

Cliquez sur "Envoyer" pour transmettre les paramètres dans la machine. Cela fera s'arrêter notre extrudeur à environ 15 mm de la zone d'impression. On placera une feuille de papier sur le plateau d'impression. Puis, grâce aux flèches directionnelles dans l'interface Repetier, on déplacera notre tête sur l'axe Z vers le plateau avec un pas

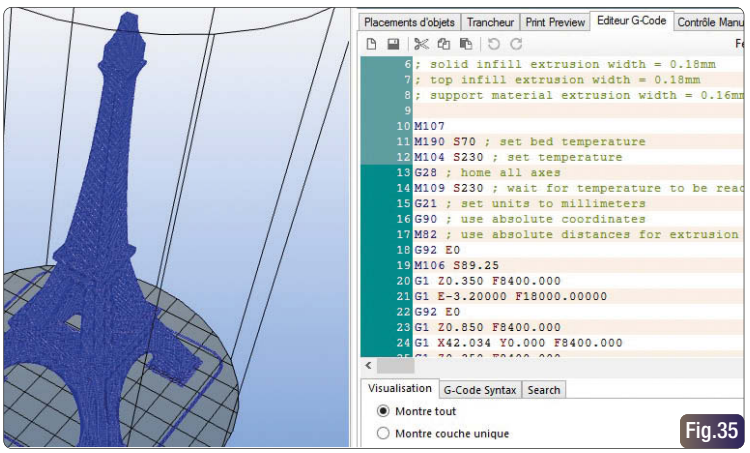
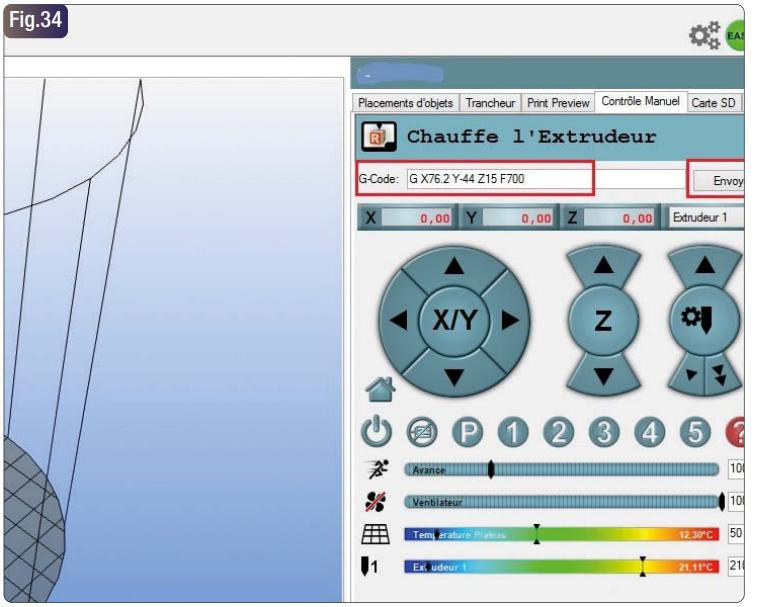


Fig.35

Fig.34



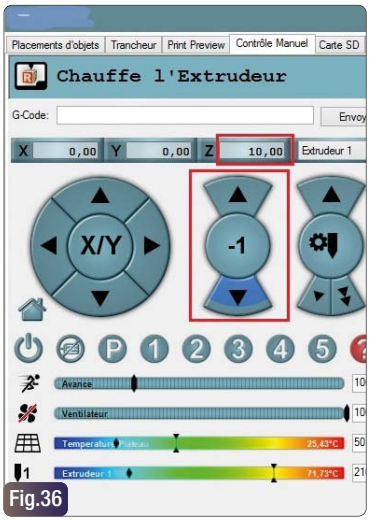


Fig.36

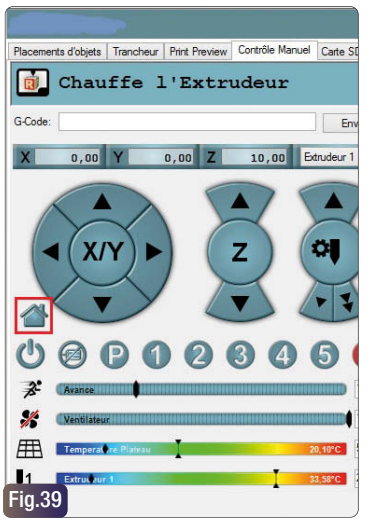


Fig.39

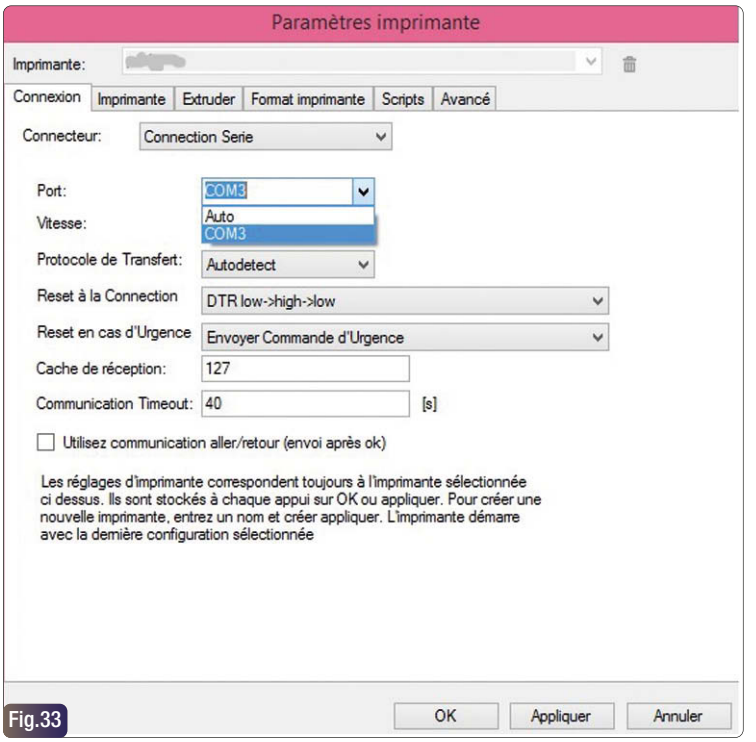


Fig.33

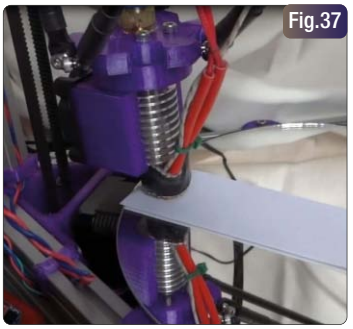


Fig.37

de 1 mm jusqu'à arriver assez proche du plateau. On continue ensuite avec un pas de 0,1 mm (**Fig.36 et 37).** Quand il se fera sentir une petite résistance en tirant ou poussant la feuille, ne touchez plus à rien. A ce moment, jetez un œil à la position de Z dans l'interface Repetier. Elle devrait normalement être à zéro (**Fig.36).** Si tel n'était pas le cas, vous devrez rajouter ou retirer la différence à "endstop_tower" (**Fig.38).** Ex: si en touchant le plateau vous êtes à z = 0,5, vous devez retirer 0,5 mm à "endstop_tower". L'endstop_offset a été mis à 800 pour les 3 tours (X, Y, Z). 800 représente l'offset (décalage) que l'on ajoute à "Tower endstop offset". Ce qui représente environ 5 mm.

Calcul : 5 mm x 160 = 800 ; 160 étant le nombre de pas du moteur pour déplacer la tête d'impression de 1 mm. Nous disions que nous devions retirer 5 mm de 800. Le calcul est 0,5 x 160 ce qui représente 80 pas. Ces 80 pas seront soustrait de 800: 800 − 80 => 720. Votre valeur en main, rendez-vous dans l'onglet "Configurations" puis "firmware EEPROM". De là, cherchez "Tower X endstop offset" et entrez la valeur de 720 calculée plus tôt en validant par "OK". Mettez l'imprimante en position "Home" et tapez ensuite le même Gcode qu'auparavant **(Fig.39)** :

G1 X-76,2 Y-44 Z15 F7000

et appuyez sur "Envoyer".

Répétez la même manipulation avec le papier sous la buse et vérifiez que notre Z est bien égal à zéro. Si la tour X est calibrée, répétez la même manipulation avec les tours Y et Z en prenant garde de changer les coordonnées X et Y dans le Gcode.

- Tour Y : **G1 X 76,2 Y− 44 Z15 F7000**
- Tour Z : **G1 X0 Y88 Z15 F7000**
- Centre : **G1 X0 Y0 Z15 F7000**

Faites l'ajustement pour les 3 tours et le centre en n'oubliant pas de rajouter

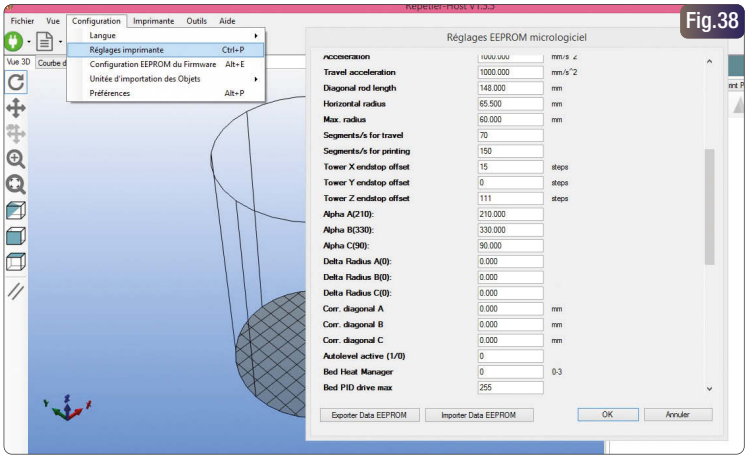


Fig.38

l'offset et d'effectuer la sauvegarde dans l'EEPROM. De nombreux essais seront nécessaires. Un peu de patience jusqu'à votre 1ere impression.

Nous joindre

L'impression 3D étant en constante évolution, il est important d'avoir de bonnes bases afin de bien maîtriser toutes les parties nécessaires pour le bon déroulement de vos impressions. Gardez en tête que cette technologie s'incruste de plus en plus dans tous les domaines. Vous serez amené à croiser ou à en entendre parler demain ou après-demain.

Si comme tous les passionnés vous souhaitez plus d'informations ou assister à un stage, contactez-nous : www.geekofyou.fr/formation/ ou par mail : contact@geekofyou.fr

Sources :

www.geekofyou.fr
www.reprap.org
www.ultimaker.com
www.formlabs.com
www.kosselpro.com
www.charlyrobot.fr
www.youtube.com : reprap mini kossel
www.makergeeks.com
www.thingiverse.com
www.aliexpress.com
www.staples.ca
www.7en3d.com
https://sites.google.com/site/3dprinterlist/delta-3d-printers/delta-robot-calculator

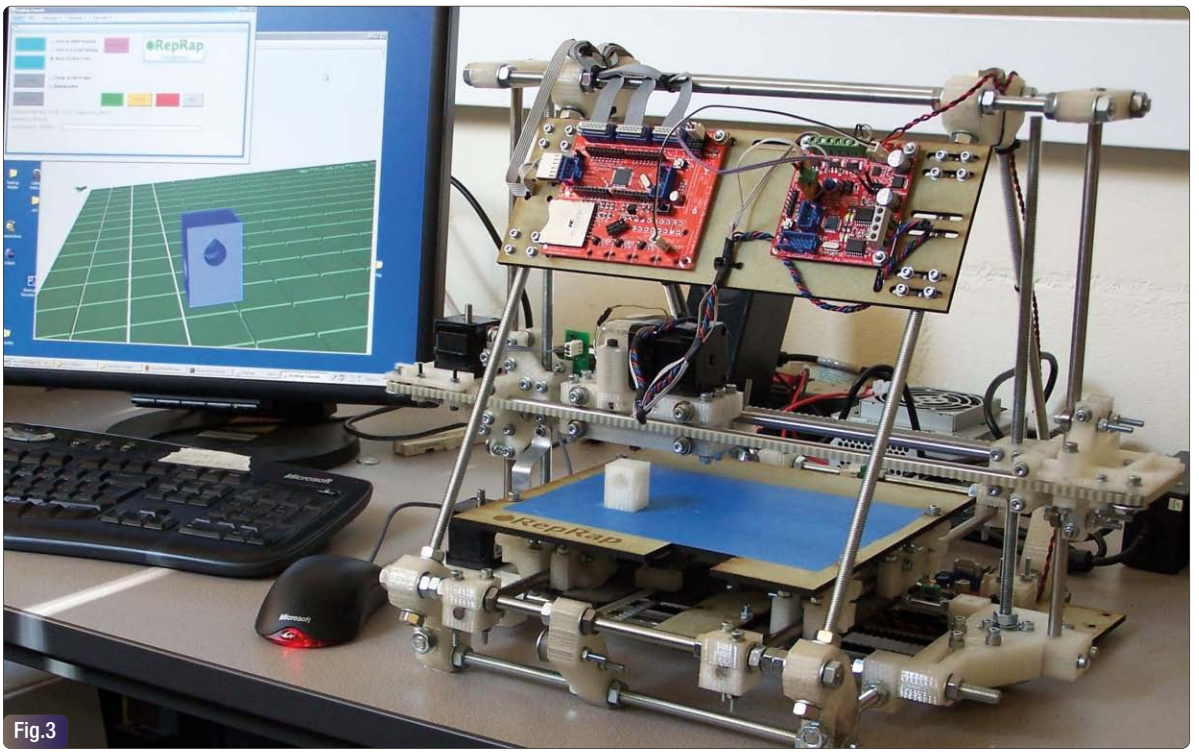


Fig.3

OrmLite : le framework ORM léger pour Android

Le stockage des données au sein des applications est un enjeu crucial. Pour adresser cette problématique, la plateforme Android met à disposition du développeur un certain nombre de solutions. Parmi celles-ci, on retrouve une base de données légère SQLite accessible pour chaque application. Bien que parfaitement fonctionnelles, les APIs fournies par le SDK imposent de passer par du SQL. Afin de proposer une alternative se rapprochant de ce que peuvent connaître les développeurs du monde Java, OrmLite a été adapté spécifiquement pour Android. Tour d'horizon d'un framework qui promet d'accroître la productivité des développements Android.

Sylvain SAUREL
Ingénieur d'Etudes Java / JEE
sylvain.saurel@gmail.com

Bien connu du monde des développeurs Java car popularisé par le framework Hibernate, le concept d'ORM s'avère essentiel lors du développement d'applications manipulant une base de données afin d'accroître la productivité des développements.

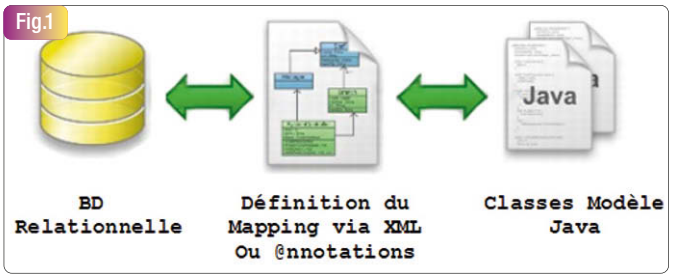
Concrètement, l'ORM (pour Object-Relational Mapping) est une technique permettant d'établir un mapping entre des données de systèmes incompatibles. Appliqué au monde des bases de données, les ORMs permettent le mapping entre des données relationnelles et des classes Java dans le but de faciliter les opérations liées à la base de données (figure 1). Définies à l'origine au sein de fichiers XML, les informations de mapping sont désormais manipulables grâce aux annotations Java, et c'est cette approche qui est privilégiée. Fig.1.

Au sein du monde des ORM Java, OrmLite est un framework open source moins connu qu'Hibernate et MyBatis, se démarquant par sa légèreté et sa puissance. Se voulant avant tout léger, OrmLite a été créé avec l'objectif de fournir une abstraction autour des fonctions amenées par JDBC sans la complexité inhérente aux frameworks cités précédemment. OrmLite supporte les connexions JDBC pour un grand nombre de bases de données parmi lesquelles MySQL, PostgreSQL, Microsoft SQL Server ou encore SQLite tout en étant assez extensible pour supporter de nouvelles bases. Dans notre cas, c'est le support de SQLite qui nous intéresse. Peu connu, SQLite est une base de données légère qui, en dépit d'un nom modeste, se révèle performante et extrêmement stable tout en possédant un ensemble de fonctionnalités robuste. Au sein du SDK Android, les APIs liées à JDBC sont bien existantes puisque les classes du package java.sql.* sont présentes mais elles restent non supportées. Au lieu de ça, Android pousse à l'utilisation de son API contenue au sein des classes du package android.database.sqllite.* mais l'utilisation de ces dernières se révèle fastidieuse et nécessite une manipulation du SQL. Pour des développeurs issus du monde Java, cette contrainte s'avère souvent contre-productive. Heureusement, depuis Septembre 2010 OrmLite prend en charge les appels natifs aux APIs de bases de données d' Android rendant accessible l'utilisation de ce framework ORM léger au sein de vos applications.

Installation

Activement maintenu, OrmLite est actuellement en version 4.42 et se présente sous la forme de 3 modules nommés core, jdbc et Android. Ils peuvent être ajoutés à un projet via des dépendances Maven ou directement en les téléchargeant depuis le site suivant : http://ormlite.com/releases/ . Quelle que soit l'utilisation faite d'OrmLite, le module core sera utilisé. Les archives JAR jdbc et Android s'utilisant quant à elles en alternance suivant le contexte. Dans notre cas, nous téléchargeons ainsi le module Android. Ajoutés au classpath d'un projet Android, ces 2 archives JAR (ormlite-core-4.42.jar et ormlite- android-4.42.jar) ne représentent qu'un poids total d'un peu plus de 300 Ko ce qui reste relativement faible comparé à la plus-value apportée par le framework.

Fig.1



Fonctionnement général d'un ORM

Fig.2

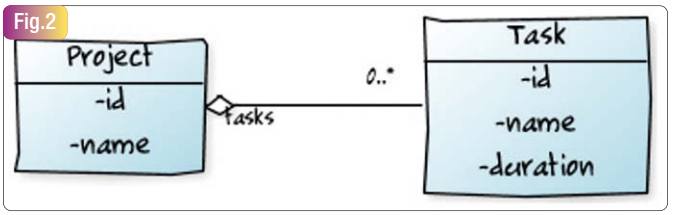


Diagramme de classes des objets du domaine

Modélisation Base de Données

Pour illustrer la prise en main d'OrmLite, nous considérons un modèle de données simplifié représentant des projets et des tâches. Ce modèle est présenté à la figure 2 sous la forme des objets du domaine Java correspondants. Fig.2. OrmLite adopte une approche descendante puisqu'il s'appuie sur la définition des objets du domaine Java pour créer la base de données relationnelle SQLite associée. La définition du mapping se fait au niveau des classes Java grâce à l'utilisation d'annotations. Parmi celles-ci, on dénote 3 annotations principales qui s'avèrent suffisantes dans la majorité des cas d'utilisation :

- **@DatabaseTable** se plaçant devant la déclaration d'une classe Java modélisant un objet du domaine, et prenant en paramètre le nom de la table sur laquelle on fait le mapping.
- **@DatabaseField** permettant de configurer le mapping entre une propriété d'un objet du domaine et la colonne d'une table. Les propriétés de la colonne de base peuvent être configurées finement via l'emploi des différents paramètres de l'annotation que nous détaillerons par la suite.
- **@ForeignCollectionField** placée devant une collection d'objets et utilisée dans le cadre de la modélisation de relations 1-N.

Appliquée à nos classes Project et Task, la mise en place des annotations du mapping avec la base de données donne le résultat suivant :

```
@DatabaseTable(tableName = Project.TABLE_NAME)
public class Project implements Serializable {
```

```
  public static final String TABLE_NAME = "projects";
  @DatabaseField(generatedId = true)
  private int id;
  @DatabaseField(canBeNull = false, unique = true)
  private String name;
  @ForeignCollectionField
  private ForeignCollection<Task> tasks;
  // … getters / setters …
}

@DatabaseTable(tableName = Task.TABLE_NAME)
public class Task implements Serializable {
  public static final String TABLE_NAME = "tasks";
  @DatabaseField(generatedId = true)
  private int id;
  @DatabaseField(canBeNull = false)
  private String name;
  @DatabaseField
  private int duration;
  @DatabaseField(foreign = true, foreignAutoRefresh = true, columnName = "project_id")
  private Project project;
  // … getters / setters …
}
```

Ce code met en avant l'utilisation de champs indexés auto-incrémentés pour les colonnes id via le positionnement à true de la propriété generatedId de l'annotation @DatabaseField. On remarquera également la configuration plus fine de certaines colonnes de la base de données via les propriétés de cette même annotation parmi lesquelles on citera :

- **columnName** pour la définition du nom de la colonne de base sur laquelle le mapping est fait. Inutile dans le cas où l'on souhaite que le nom du champ et celui de sa colonne associée soient identiques.
- **canBeNull** précisant si un champ peut être nullable.
- **defaultValue** affectant une valeur par défaut à un champ avant enregistrement en base.
- **unique** définissant une colonne unique.
- **foreign** précisant si un champ est une clé étrangère de type 0-1. C'est le cas du champ project de la classe Task.
- **foreignAutoRefresh** définissant le comportement que doit adopter OrmLite pour le champ annoté lors de la remontée de l'objet parent. Dans le cas où la propriété est à true, l'objet lié est automatiquement rafraîchi.

Enfin, on signalera la présence d'un @ForeignCollectionField sur la collection de tâches contenues au sein d'un projet. Cette collection devant être de type ForeignCollection qui est une interface étendant l'interface Collection et managée par le framework.

Création du DatabaseHelper

La création et la gestion de la base de données par OrmLite nécessite de créer son propre DatabaseHelper. Ce dernier doit étendre OrmLiteSqlite OpenHelper pour autoriser OrmLite à prendre la main. Au sein du DatabaseHelper, des méthodes callbacks seront ainsi appelées lors de la création de la base de données à la première exécution et lors de la migration de la base lors des mises à jour de l'application suite à un changement de version. Il s'agit respectivement des méthodes onCreate et onUpgrade. L'implémentation doit globalement avoir l'allure suivante :

```
public class DatabaseHelper extends OrmLiteSqliteOpenHelper {
  private static final String DATABASE_NAME = "ormlitesample.db";
  // A incrémenter à chaque nouvelle version
  private static final int DATABASE_VERSION = 1;
  private ProjectDao projectDao;

  public DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION, R.raw.ormlite_config);
  }

  public void onCreate(SQLiteDatabase db, ConnectionSource connectionSource) {
    try {
      TableUtils.createTable(connectionSource, Project.class);
      TableUtils.createTable(connectionSource, Task.class);
    } catch (SQLException e) {
      Log.e(TAG, "Can't create database", e);
      throw new RuntimeException(e);
    }
  }

  public void onUpgrade(SQLiteDatabase db, ConnectionSource connectionSource, int old
Version, int newVersion) {
    try {
      TableUtils.dropTable(connectionSource, Project.class, true);
      TableUtils.dropTable(connectionSource, Task.class, true);
      onCreate(db, connectionSource);
    } catch (SQLException e) {
      Log.e(TAG, "Can't drop databases", e);
      throw new RuntimeException(e);
    }
  }

  public ProjectDao getProjectDao() throws SQLException {
    if (projectDao == null) {
      projectDao = DaoManager.createDao(getConnectionSource(), Project.class);
    }

    return projectDao;
  }

  @Override
  public void close() {
    super.close();
  }
}
```

Le DatabaseHelper est un endroit parfaitement adapté au stockage de DAOs spécifiques comme cela est fait avec le ProjectDao. On remarque également les méthodes utilitaires de la classe TableUtils fournies par le framework facilitant les opérations de création et de drop des tables.

Au sein du constructeur du DatabaseHelper, l'appel au constructeur parent contient une référence à un mystérieux fichier contenu au sein du répertoire res/raw et nommé ormlite_config. Ce fichier sert à améliorer le temps de chargement initial des DAOs qui peut s'avérer très long dans certains cas. En effet, pour construire les DAOs, OrmLite doit parcourir les objets du modèle à la recherche des annotations de mapping en recourant à la réflexion ; technique demeurant extrêmement pénalisante avec Dalvik. Afin de résoudre ces potentiels problèmes de performances en s'abstrayant du travail de recherche des annotations, OrmLite fournit des classes utilitaires générant un fichier plat de description du schéma de base de données utilisé ensuite au démarrage de l'application par le DatabaseHelper. Ultra simple, leur mise en place est réalisée de la sorte :

```
public class DatabaseConfigUtil extends OrmLiteConfigUtil {
  // Objets du domaine
  private static final Class<?>[] classes = new Class[] {
    Project.class, Task.class
  };

  public static void main(String[] args) throws SQLException, IOException {
    writeConfigFile("ormlite_config.txt", classes);
  }
}
```

L'exécution de code produit un fichier ormlite_config au sein du répertoire res/raw d'un projet qui sera ensuite utilisé au sein du DatabaseHelper. Il est bon de souligner que l'exécution de l'utilitaire doit se faire au sein de votre environnement de développement et non sur un périphérique Android. Enfin, à chaque modification du modèle de données il sera nécessaire de le générer de nouveau pour qu'il reste à jour.

DAO

OrmLite gère le cycle de vie des DAOs tant au niveau de leur création via le DaoManager qu'au niveau de leur réutilisation une fois créés. La création d'un DAO peut se faire directement à partir du DaoManager de la sorte :

```
Dao<Project, Integer> projectDao = DaoManager.createDao(connectionSource, Project.class);
```

Néanmoins, il est plus commode de passer par le DatabaseHelper pour accéder aux DAOs puisqu'alors leur création est réalisée de manière transparente pour le développeur. Le Helper propose une méthode générique getDao renvoyant une instance de Dao typée avec l'objet du modèle concerné. Les DAOs de base fournissent des méthodes génériques facilitant la réalisation d'opérations CRUD. Pour des besoins plus spécifiques, il est possible et même conseillé d'implémenter ses propres DAOs comme cela est fait avec le ProjectDao :

```
public class ProjectDao extends BaseDaoImpl<Project, Integer> {

  public ProjectDao(ConnectionSource connectionSource) throws SQLException {
    super(connectionSource, Project.class);
  }

  public List<Project> findAll() throws SQLException {
    return queryForAll();
  }

  public Project findById(int id) throws SQLException {
    return queryForId(id);
  }
}
```

Pour l'utilisation de DAOs spécifiques, OrmLite propose la propriété dao au sein de l'annotation @DatabaseTable. Cependant, cette pratique est à déconseiller puisqu'elle induit un couplage fort entre un objet du modèle et l'implémentation de son DAO ! Le remède à ce mal va consister à stocker une instance du ProjectDao au sein du DatabaseHelper comme cela a été précédemment.

Mise en place au sein d'une Activité

La gestion des connexions simultanées à la même base devant être surveillée avec extrême précaution, il est préférable de déléguer ce travail à OrmLite. Ainsi, nous utilisons la classe OpenHelperManager chargée de surveiller les appels au DatabaseHelper. L'accès à une instance de cet objet se faisant via un appel à getHelper() au sein des activités. Bénéficier de cette dernière implique de créer des activités héritant de classes managées par OrmLite et gérant les accès à la couche de persistance. Les classes à utiliser sont OrmLiteBaseActivity, OrmLiteBaseListActivity, OrmLiteBaseService ou OrmLiteBaseTabActivity et couvrent les différents cas d'utilisation des activités Android. Dans le but de lister les projets stockés en base de données, nous créons une activité héritant de OrmLiteBaseListActivity :

```
public class MainActivity extends OrmLiteBaseListActivity<DatabaseHelper> {
  private Button addProject;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    // ... code lien vers AddProjectActivity
  }

  @Override
  protected void onResume() {
    super.onResume();
    loadProjects();
  }

  private void loadProjects() {
    List<Project> projects = new ArrayList<Project>();

    try {
      ProjectDao projectDao = getHelper().getProjectDao();
      projects = projectDao.findAll();

    } catch (SQLException e) {
      Log.i(TAG, e.getMessage());
    }

    ListProjectAdapter adapter = (ListProjectAdapter) getListAdapter();
    adapter.setProjects(projects);
    adapter.notifyDataSetChanged();
  }

  private void deleteProject(final Project project) {
    try {
      final DatabaseHelper helper = getHelper();
      TransactionManager.callInTransaction(helper.getConnectionSource(), new Callable<Void>() {
        @Override
        public Void call() throws Exception {
          Dao<Task, Integer> taskDao = helper.getDao(Task.class);
          taskDao.delete(project.getTasks());
          ProjectDao projectDao = helper.getProjectDao();
          projectDao.delete(project);
          loadProjects();
          return null;
        }
      });
    } catch (SQLException e) {
      Log.i(TAG, e.getMessage());
    }
  }
```

```
  // ... code de l'alert dialog appelant la méthode deleteProject
}
```

Lors de son affichage, l'activité présente à l'utilisateur l'ensemble des projets stockés dans sa base de données en s'appuyant sur le ProjectDao. Lors du clic prolongé sur un élément de la liste, on propose à l'utilisateur la suppression du projet sélectionné. Celle-ci est réalisée au sein d'une transaction encapsulée dans un TransactionManager mettant à disposition la méthode callInTransaction. Se voulant le plus simple et léger possible, OrmLite ne supprime pas automatiquement en cascade les objets liés ce qui implique de supprimer explicitement les tâches associées à un projet via un appel à delete du DAO de l'objet Task.
Une seconde activité nommée AddProjectActivity donne la possibilité à l'utilisateur de créer des projets avec des tâches. Le contenu de cette activité n'est pas particulièrement intéressant dans le cadre de cet article si ce n'est sa méthode save() qui persiste en base un projet ainsi que ses tâches associées en utilisant pour cela la gestion des objets liés d'OrmLite :

```
Project project = new Project();
project.setName(name.getText().toString());
ProjectDao projectDao = getHelper().getDao(Project.class);
CreateOrUpdateStatus createStatus = projectDao.createOrUpdate(project);
// refresh du projet
project = projectDao.findById(project.getId());
// création des tâches
String tmp = tasks.getText().toString();
String[] tabTasks = tmp.split(";");
ForeignCollection<Task> listTasks = project.getTasks();

for (String tabTask : tabTasks) {
  Task task = new Task();
  task.setName(tabTask);
  task.setProject(project);
  listTasks.add(task);
}

CreateOrUpdateStatus updateStatus = projectDao.createOrUpdate(project);

if (createStatus.isCreated() && updateStatus.isUpdated()) {
  Toast.makeText(getApplicationContext(), "Projet créé", Toast.LENGTH_SHORT).show();
}
```

Requêtage

OrmLite donne également accès au développeur à un QueryBuilder afin de construire des requêtes SQL simplement de manière lisible sans utiliser de SQL. Pour ce, le QueryBuilder expose un ensemble de méthodes à combiner via des appels chaînés. Nous pouvons ainsi rajouter une méthode de sélection des tâches dont la durée est égale à 5 pour le projet d'id 1 :

```
Dao<Task, Integer> taskDao = helper.getDao(Task.class);

Where<Task,Integer> where = taskDao.queryBuilder().where().eq(Task.DURATION, 5).and()
.eq(Task.PROJECT_ID, 1);
PreparedQuery<Task> prepare = where.prepare();
List<Task> tasks = taskDao.query(prepare);
```

En outre, l'exécution de requêtes SQL demeure accessible via la méthode executeRaw du Dao générique d'OrmLite.

Conclusion

Léger et complet, OrmLite s'avère être une solution particulièrement efficace pour manipuler des bases des données dans les applications Android. Sa mise en œuvre est aisée comme nous avons pu le constater, et son approche de mapping par annotations le rapproche de ce que les développeurs Java connaissent côté serveur. Les âmes les plus chagrines mettront en avant un certain manque de performances au chargement initial des DAOs, mais là encore OrmLite amène une réponse concrète et efficace avec la génération d'un fichier plat décrivant le modèle de données associé à une application. De fait, il s'impose comme une solution de référence pour la gestion des bases de données augmentant nettement la productivité des développements Android.

Construire sa propre borne d'arcade avec Raspberry Pi

Tout est parti d'un constat simple : le jeu vidéo ne me divertissait plus autant qu'avant et malgré l'aspect visuel en constante évolution, je ne retrouvais plus la passion que j'avais eue pendant de nombreuses années.

Fig.1

Thomas Brouwer
SQLi

Je me dis souvent que je suis né trop tard, et que je n'ai pas eu la chance de connaître les parties endiablées au fond d'un bar de quartier ou la simplicité des jeux des années 80. Né en 1990, j'ai été bercé par les premières grosses productions en 3D et au rythme effréné des sorties, qui ne nous laissent plus le temps de savourer chaque nouveau jeu. Au fil du temps je me suis plongé dans le retrogaming en cherchant d'abord à posséder le plus de consoles possible et en finissant par me focaliser sur quelques consoles précises. Le budget n'étant pas extensible, j'ai eu l'idée de construire ma propre borne d'acarde de type "Bartop" que l'on pouvait trouver à l'époque sur les comptoirs de bars, mais en utilisant des technologies actuelles avec un Raspberry Pi que je venais d'acquérir. Le choix d'un tel type de borne a été conduit dans l'idée de pouvoir la caser sur ma table de salon dans mon petit appartement de 35m2, en me fixant un budget de 100€ maximum.

Matériel

- Un écran de 17 pouces au format 4:3, récupéré dans une association de recyclage de matériel informatique.
- Des panneaux de Médium 3mm.
- Visserie :
  - Vis de 5 x 20 mm ;
  - Ecrous de 5 mm ;
  - Vis de 4 x 20 mm ;
  - Ecrous de 4 mm ;
  - Charnières 70 x 40 ;
  - Charnières 50 x 50 ;
  - Équerres en acier.
- Boutons et sticks :
  - 7 boutons de 30 mm ;
  - 2 boutons de 24 mm ;
  - 1 stick et sa plaque de montage ;
- Connectique :
  - Cosses femelles de 2,8 mm ;
  - Câbles conducteurs avec jumper (pour se brancher au port GPIO du Raspberry Pi) ;
  - Connecteur 5 broches pour brancher le Joystick ;
  - Un câble HDMI avec adaptateur VGA ou DVI selon l'écran.
- Un Raspberry Pi ;
- Une carte SD de catégorie 10 de préférence ;
- Des enceintes d'ordinateurs recyclées.

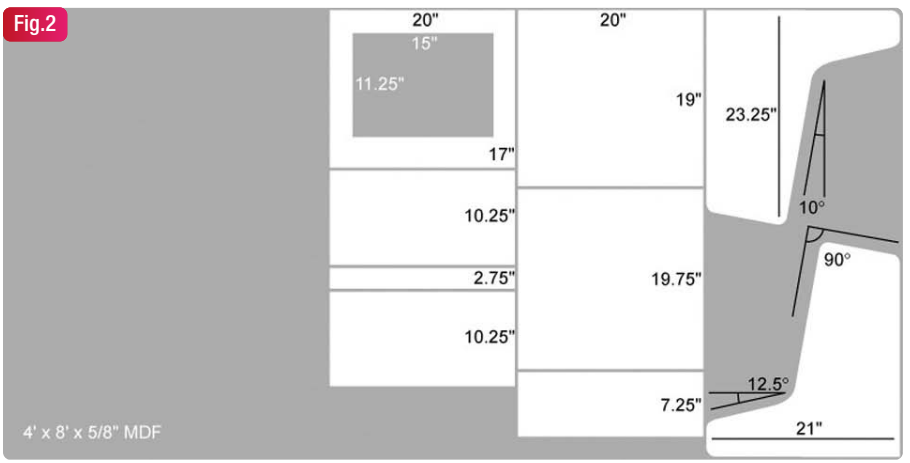
La liste du matériel est assez longue et le nombre de vis, écrous, câbles etc., dépendra du degré de finition que vous voudrez apporter à votre borne, ainsi que de la solidité globale attendue. Fig.1.

Structure et découpage

La première partie du travail fut de se décider sur la structure globale de la borne. Pour cela, j'ai utilisé un modèle trouvé sur Internet qui convenait parfaitement à mes attentes. Celui-ci est tiré du site BartopArcade (http://bartoparcade.katorlegaz.com/) et nous fait même l'honneur de contenir les différentes côtes à respecter. Fig.2.

Le plus gros du travail a été de vectoriser ces éléments en utilisant Inkscape, pour ensuite envoyer le fichier à une découpeuse laser qui s'est chargée de sortir l'ensemble de mes pièces aux dimensions exactes. Vous pouvez également réaliser la découpe à la main avec une scie circulaire et beaucoup de précision. Ensuite, la seule partie (et pas la moindre) que j'ai entièrement réalisée à la main est le panel de jeu, qui contient l'ensemble des boutons et le joystick. Pour décider quelles proportions respecter, plusieurs tentatives ont été réalisées, avec différents espacements entre les boutons en premier lieu, puis en faisant varier la place allouée au repos du poignet sur la planche.

Fig.2



Connectique

Une fois l'enrobage terminé et convenable, les branchements sont entrés en jeu. Le choix du Raspberry n'a pas été fait par hasard. Au début je pensais utiliser un ordinateur de bureau avec un format réduit, ou bien en désosser un complètement pour ne conserver que les composants et ainsi réduire le poids général. Le premier problème est que j'allais avoir besoin d'une alimentation beaucoup plus importante et que le bruit allait également être plus présent. Le second est que les composants allaient être posés au fond de la borne et que visuellement le résultat ne serait pas convaincant. Enfin, le coût était à prendre en compte, puisque même un ordinateur d'occasion et relativement correct pour un tel projet m'aurait demandé un investissement financier plus important que l'achat d'un Raspberry que par chance je possédais déjà. Le Raspberry a plusieurs avantages : sa faible

consommation en énergie sans pour autant avoir des capacités ridicules, son encombrement minimum puisqu'il fait à peu près la taille d'un téléphone, et enfin son émission de bruit nulle puisque seule la LED dessus permet d'attester de son fonctionnement.

Pour le branchement des boutons et du joystick, j'ai d'abord pensé à récupérer le squelette d'un vieux clavier pour en extraire son circuit et ainsi mapper chaque élément sur une touche préalablement choisie. Le problème est qu'un circuit de clavier est un fouillis monstre et qu'il faut suivre chaque ligne pour tomber sur la bonne broche à la fin.

L'idée a été abandonnée après avoir trouvé deux touches en 30 minutes. La solution de brancher en entrée directe sur le port GPIO du Raspberry a donc été choisie puisqu'elle permet de brancher, sans intermédiaire, un bouton sur une broche du GPIO.

Il suffit alors de posséder un ensemble de fils avec une cosse d'un côté pour le brancher aux bornes du bouton et un jumper de l'autre pour l'insérer sur une broche. Fig.3.

Le port GPIO d'un Raspberry se présente comme suit : chaque borne correspond à un numéro, une tension d'alimentation ou une terre. Par exemple pour le joystick qui possède une sortie 5 broches, il faudra utiliser 4 bornes numérotées, ainsi qu'une borne correspond à la terre (GRD pour ground), un bouton quant à lui utilisera uniquement une borne numérotée et une borne GRD. Pour optimiser l'utilisation des bornes GRD, j'ai réalisé un montage en série à ce niveau là.

Pour que cette installation fonctionne, il a fallu compiler la distribution avec le programme Retrogame fourni par Adafruit. Il faut télécharger le programme depuis le dépôt Github du projet (https://github.com/adafruit/Adafruit-Retrogame) puis décompresser le contenu à la racine de la carte SD. Modifier le fichier "retrogame.c" pour mapper chaque bouton avec la touche correspondante.

```
ioStandard[] = {
  { 25, KEY_LEFT    },
  {  9, KEY_RIGHT   },
  { 10, KEY_UP      },
  { 17, KEY_DOWN    },
  { 23, KEY_LEFTCTRL },
  {  7, KEY_LEFTALT },
  // ...
  { -1, -1 }
};
```

Ce fichier détaille la relation entre un numéro d'élément (le bouton ou joystick branché sur le port GPIO) et l'action associée. Pour trouver les actions possibles, il faut regarder dans le fichier /usr/include/linux/input.h. Une fois les changements effectués, il faut lancer la commande suivante pour compiler le programme :

```
make retrogame
```

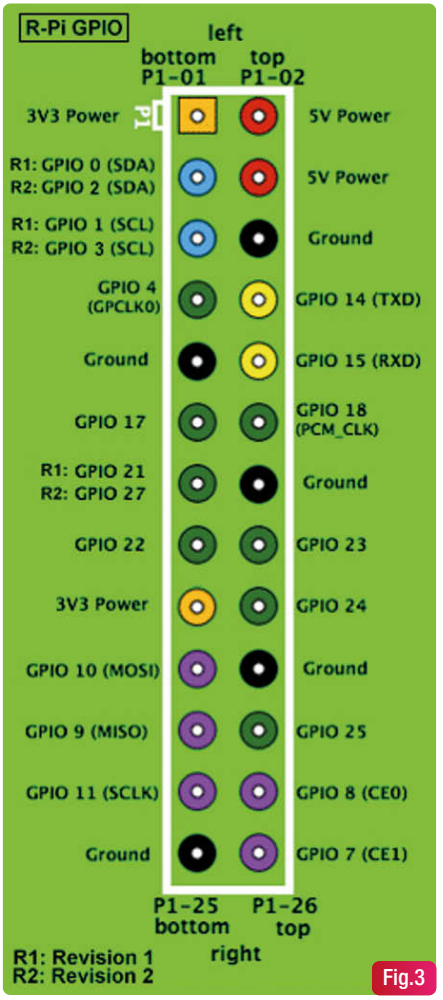


Fig.3

Système

Tout est alors fonctionnel, il ne manque plus qu'une distribution pour faire tourner tout ça sans perte de performance. J'ai choisi PiMame dans un premier temps, mais c'était avant que je ne découvre Lakka (http://lakka.tv) qui propose une interface bien plus agréable et facile à configurer. En effet, son interface se rapprochant d'un XMB (interface semblable à celle d'une PlayStation 3 par exemple) cela lui donne un côté accessible tout en restant élégant. Le principal problème de PiMame est que la navigation dans les menus ne peut pas se faire en utilisant le joystick, ce qui imposait d'intégrer une souris à l'installation. L'avantage du choix de Lakka est qu'aucune programmation n'a été nécessaire. Il suffit de déziper la distribution sur la carte SD puis de lancer le Raspberry pour pouvoir profiter de ses jeux. Fig.4.


Fig.4

Conclusion

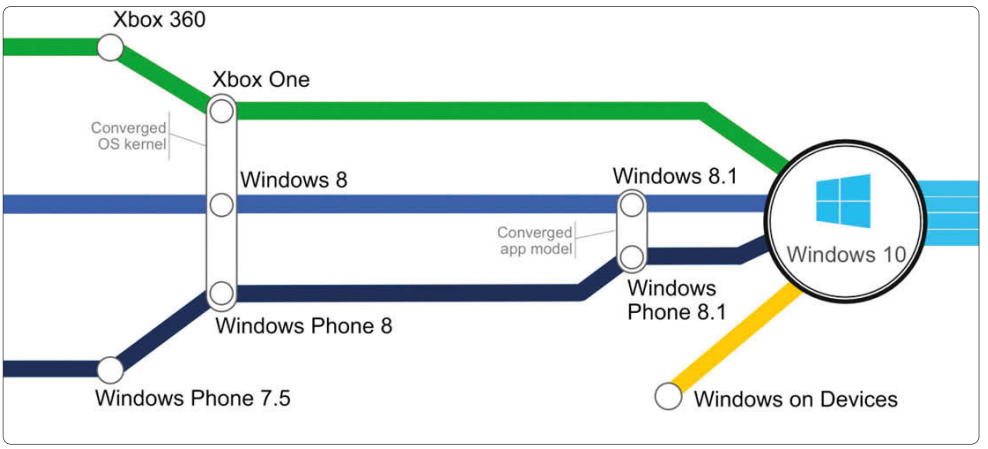
Autant l'avouer, le budget plafonné au départ à 100€ n'a pas été respecté. Au total la création de la borne complète m'est revenue à 150€. J'avais principalement sous-estimé le prix du joystick que je pensais tourner aux alentours de 15€ mais qui finalement coûtait 24€, et surtout l'écran que je pensais récupérer sur un vide grenier ou dans une brocante pour 10€, mais que j'ai payé 25€. Le prix final n'est pas non plus exorbitant, mais la réalisation d'une telle arcade permet de retrouver partiellement le confort d'une vraie borne sans débourser des sommes astronomiques chez des fabricants professionnels, avec en prime la satisfaction personnelle de l'avoir réalisée soi-même. Je parle de confort partiel car le médium ne remplacera jamais du bois brut et l'écran ne correspond pas aux cathodiques utilisés à l'époque, mais le bonheur de pouvoir jouer à une multitude de jeux - contrairement aux anciennes bornes qui ne proposaient qu'un jeu à la fois - fait oublier ces quelques aspects négatifs. Au palmarès des aspects positifs on peut principalement citer le cliquetis du joystick qui est vraiment similaire à ceux des vraies bornes et qui flatte l'oreille à chaque mouvement, et le confort des boutons qui ne seront jamais battus par les manettes actuelles.

Pour une prochaine version, je pense conserver le choix du Raspberry en prenant cependant la dernière version pour de meilleures performances, mais ce micro-ordinateur permet à faible coût et encombrement, de concevoir un système robuste.

Universal Windows Platform avec Desktop, Mobile, Xbox, IoT : Développement d'applications UWP (Windows 10 et VS 2015)

Windows 10 est sorti cet été, il est arrivé avec la fonctionnalité dont vous rêviez tous, le menu « démarrer » est de retour ! Non, plus sérieusement, c'est un véritable tour de force de la part de Microsoft. Un seul OS Windows qui peut faire tourner sur tous les dispositifs Windows 10. Une seule plateforme logicielle pour de véritables applications universelles.

Nicolas Cornet,
Développeur senior **VISEO**



Il est impossible de vous parler des applications universelles sans d'abord vous expliquer la genèse de Windows 10. En 2011, Microsoft avait 3 plateformes avec 3 systèmes d'exploitation différents :

- Les PC et les serveurs construits sur le noyau Windows NT ;
- Windows Phone bâti sur un fork de Windows CE avec des similarités avec Windows NT, mais un code source différent ;
- La Xbox 360 basée sur Windows NT, mais, depuis 10 ans, le fork avait évolué aboutissant une fois de plus à un code source bien différent.

La direction était donnée, la convergence était la cible. Windows Phone 8 en 2012 démarrait le bal et migrerait vers un noyau NT. En 2013 la Xbox One utilisait une version épurée de Windows 8. Cette fois-ci, les plateformes utilisent le même noyau, mais les codes sources restent différents. Windows 10 apporte cette convergence tant attendue. Non seulement pour les 3 plateformes existantes, mais aussi pour les nouvelles puisque le nouveau noyau fonctionne aussi sur des nano-ordinateurs (Raspberry Pi 2, Arduino, etc.), Hololens, la surface hub et tous les futurs dispositifs qui supporteront Windows 10.

Nouveautés du développement d'applications universelles

Les applications universelles ont été introduites à partir de Windows 8.1. Lors de la création d'un nouveau projet d'application universelle, le template commun de Visual Studio créait 3 projets : Windows, Windows Phone et Shared. Le projet Shared ne représentait pas une assembly à part entière, mais un ensemble de fichiers de code partagé entre les 2 applications. Les projets Windows et Windows Phone venaient alors lier ces fichiers au moment de la compilation pour construire leur exécutable respectif. Sous Windows 10, il n'y a plus qu'un seul et unique exécutable et donc un seul projet pour tout le code source, quelle que soit la famille de dispositif ciblée. L'application peut donc désormais s'exécuter partout où fonctionne Windows 10.

D'un point de vue marketplace, il y avait deux « stores » Windows 8.1, un pour les applications pour téléphones et un second pour les applications pour PC. Un mécanisme d'association fonctionnait grâce à l'association du compte du développeur et du nom de l'application. Cette gymnastique est désormais terminée, il n'y a qu'un seul store pour toutes les applications. De même, il n'est plus nécessaire de dupliquer les achats « In-App » sur chacun des stores.

Evidemment, même si l'application peut être déployée sur tous les dispositifs Windows 10, le développeur peut choisir de cibler la ou les plateformes qu'il souhaite. Ce store unique n'est pas dépourvu de sens ; une application peut cibler un écran de 5' ou une télévision de 42'. Les « *assets* » (images, graphismes, etc.) peuvent aussi cibler leur plateforme de prédilection, ils ne seront packagés que pour leur plateforme.

Les applications HTML/JavaScript fonctionnent toujours sur Windows 10, de même que les applications C++/DirectX pour celles qui ont besoin de 3D comme les jeux vidéo. Le XAML reste pourtant la couche UI privilégiée par Microsoft. Le nouveau menu « démarrer » ou encore Office 2016 utilisent cette technologie pour leur interface graphique, c'est dire si Microsoft mise dessus. De nouveaux composants viennent agrémenter le langage pour faciliter et optimiser le développement. Pour une liste exhaustive, vous pouvez aller sur le site MSDN (Microsoft Developper Network). Voici ceux que je retiendrai :

- **RelativePanel** : un nouvel élément de layout fait son apparition. Pour faciliter le développement du responsive design (une seule interface qui s'adapte suivant la taille de l'écran), ce panel permet de positionner les éléments les uns par rapport aux autres et non plus de manière linéaire comme dans une grid ou un stackpanel. Par exemple un champ texte va pouvoir se positionner à gauche d'un autre élément, un bouton en dessous d'un autre, etc.
- **SplitView** : un contrôle qui permet de montrer ou cacher un contenu transitoire. Il permet notamment d'implémenter le fameux menu "hamburger" qui montre ou cache la navigation pour gagner sur l'espace de contenu.
- **VisualState.StateTrigger et VisualState.Setter** : encore de nouveaux éléments pour faciliter, entre autres, le responsive design. Ils permettent de décrire et de commander des transitions directement dans le XAML sans passer par le code et sans StoryBoard. On pourra ainsi par

exemple changer l'orientation d'un StackPanel quand la largeur d'écran se réduit, adapter les lignes et les colonnes d'un élément dans une grille ou affecter n'importe quelle autre propriété au déclenchement de n'importe quel trigger.

- **x:Bind** : cet attribut apporte un mécanisme de binding compilé. Le traditionnel {Binding} était évalué à l'exécution par réflexion, il pouvait donc être relativement lent. De plus, rien n'empêchait de se binder sur quelque chose qui n'existait pas, car le compilateur ne réagissait pas. Le nouveau binding compilé, lui, permet au compilateur d'émettre une erreur si le binding est incorrect et accélère le binding lors de l'exécution. Un inconvénient tout de même, le binding doit se faire sur un élément du *code-behind* fortement typé. Les adeptes du pur MVVM pour lesquels la view doit être complètement indépendante du viewmodel n'y trouveront pas leur compte. Je pense néanmoins que les avantages dépassent les inconvénients.
- **x:Phase** : ce nouvel attribut permet un rendu incrémental (ou phasé) des listes en XAML sans passer par le code. Afin d'améliorer les performances et le ressenti de l'utilisateur, une longue liste d'objets complexes pourra être rendue au fur et à mesure. Utilisé avec x:Bind, cet attribut permet de ne plus du tout passer par l'événement ContainerContentChanging comme sous Windows 8.1.
- **x:DeferLoadingStrategy :** cet attribut permet d'implémenter une stratégie de chargement et d'améliorer le chargement d'un écran. Par exemple un contrôle de validation d'input n'a besoin d'être rendu que lorsque l'utilisateur interagit avec cet input. On peut alors décider de ne pas le charger avec le reste de l'écran, mais seulement quand l'utilisateur utilise l'input, ce qui permet d'alléger l'arbre des contrôles XAML.

Bien d'autres nouveautés attendent le développeur d'applis universelles, en général ces nouveautés sont là pour plusieurs raisons. Premièrement pour unifier le développement des applications sur toutes les familles de dispositifs. Sous Windows 8.1, il y avait un composant Map sur Windows et un autre sur Windows Phone. Le nouveau MapControl devient le composant unique. Les gridview, pivot, hub, ont aussi un comportement constant, quelle que soit la famille de dispositif. Deuxièmement pour améliorer les performances, on peut citer notamment l'événement ChoosingItemContainer ou les optimisations du rendu du texte.

Enfin, il y a aussi toutes les nouveautés apportées par Windows 10 comme les nouveaux contrôles (CalendarView, CalendarDatePicker...), les vector icons, le nouveau Windows.UI.Text.Core, Cortana, le développement sur IoT, l'API de communication inter application, l'impression depuis les mobiles, etc.

Développement d'une application universelle : 500px Gallery

Commençons sans plus attendre notre application universelle. Le « Hello World » traditionnel sera basé sur la réalisation d'une galerie d'images provenant des API de « www.500px.com ».

Intro

Lorsque l'on développe une nouvelle application, on cherche les outils les plus adaptés à ce que l'on va entreprendre. Qui dit application XAML dit MVVM, et qui dit MVVM dit Framework. Est-on obligé de s'encombrer d'un Framework pour faire du MVVM ? Pas du tout, mais il faut avouer que lorsque l'on part d'une page blanche, cela peut accélérer le processus. Cela permet aussi de définir des conventions qui guideront le développement. Prism, Caliburn.Micro, MVVM Light, Okra... Tous ces Framework ont leurs mérites, comment choisir ? Il faut les tester. Pour ma part, j'ai choisi d'utiliser MVVM Light. Prouvé et éprouvé, c'est aussi peut-être celui qui a pris le plus tôt le virage Windows 10. Une nouvelle version de Prism est en cours d'écriture et je suis curieux de pouvoir l'essayer, mais pour l'heure elle n'est pas finalisée.

Une nouvelle application c'est aussi le moment de tester de nouveaux outils. Jerry Nixon, un évangéliste Microsoft assez visible puisqu'il anime beaucoup de cours sur la Microsoft Virtual Academy propose un template d'application appelé Template10. Il se compose d'un template d'application qui vise à éliminer le *boilerplate* d'une application Universal Windows, c'est-à-dire le code récurrent à toute nouvelle application. S'adjoint à ce template une librairie comprenant des services (gestion du clavier, de la navigation, des préférences utilisateurs), des converters pour la partie XAML ainsi que quelques contrôles (un menu Hamburger et une barre de titre). Si l'anglais ne vous fait pas peur, je vous engage à lire « Template10 : a new template to create Universal Windows apps – The basics »

Concernant l'environnement, Windows 10 et Visual Studio 2015 sont évidemment requis.

Installation des packages

Commençons par créer un nouveau projet dans Visual Studio. Vous allez voir que le nombre de templates disponibles pour une Application Universelle est assez restreint. Choisissons une application vierge. Ajoutons maintenant les outils que nous avons décidé d'utiliser.

Dans le Package Manager Console, taper ces deux commandes :

```
PM> Install-Package Template10
[...]
PM> Install-Package MvvmLightLibs
[...]
```

Vous pouvez également ajouter ces paquets Nuget à l'aide du Nuget Package Manager. Si vous n'avez pas l'habitude de Visual Studio 2015, vous pouvez remarquer que les dépendances Nuget utilisent maintenant la version 3.0. Plus de Package.config, mais un Package.json. Le changement le plus visible est le changement de format : du XML au JSON. Le changement le plus important est pour moi le support des dépendances transitives. Finies les dépendances qui ramènent toutes leurs sous-dépendances. En utilisant seulement quelques Nugets on pouvait se retrouver avec un long fichier Package.config confus. Ce changement de format permet aussi d'utiliser le framework .Net comme un package. Sur les applications universelles Windows 10 on ne code plus avec le Framework .Net mais avec le package .Net Core.

Pour plus de détails sur les nouveautés, rendez-vous sur le blog de Nuget : Nuget 3 – What and why ? (http://blog.nuget.org/20151008/NuGet-3-What-and-Why.html)

Pour .Net Core : http://blogs.msdn.com/b/dotnet/archive/2014/12/04/introducing-net-core.aspx

Intégrer MvvmLight et Template10

Template10 a été créé pour fonctionner en MVVM, il s'intègre sans problème à ce pattern de développement. Le template de base nous crée un App.xaml, point d'entrée du programme et son code behind App.xaml.cs. Il crée aussi une page MainPage.xaml et son code behind MainPage.xaml.cs. Comme nous avons décidé de faire du MVVM, commençons par créer un dossier ViewModels et créons cette classe dedans :

```
public class MainViewModel : Template10.Mvvm.ViewModelBase
{
    public string Message { get; }

    public MainViewModel()
```

```
  {
    Message = "Hello from MainViewModel!";
  }

  public override void OnNavigatedTo(object parameter, NavigationMode mode,
    IDictionary<string, object> state)
  {
    base.OnNavigatedTo(parameter, mode, state);
  }
}
```

C'est le ViewModel de notre MainPage.xaml. La philosophie du MVVM demande de limiter au maximum le code dans le fichier code behind de la page, certains ne veulent d'ailleurs aucun code dedans. C'est donc ce ViewModel qui coordonnera les échanges entre la page et le modèle.
Créons aussi un dossier Views dans lequel nous déplaçons la MainPage.xaml. Ne pas oublier d'adapter le namespace de cette page.
Maintenant, intégrons le code de base de Template10. A la place du App.xaml.cs coller cette classe :

```
sealed partial class App : Template10.Common.BootStrapper
{
  public App()
  {
    InitializeComponent();
  }

  public override async Task OnStartAsync(StartKind startKind, IActivatedEventArgs args)
  {
    NavigationService.Navigate(typeof(Views.MainPage));
    await Task.Yield();
  }
}
```

Celle-ci est beaucoup plus simple que le code de base. Toute la logique est intégrée dans la classe de base BootStrapper.
Ce bootstrapper intègre comme on le voit une méthode OnStartAsync. C'est ici que sera géré le démarrage de l'application. On pourra par exemple déterminer si l'application s'est lancée parce que l'on a cliqué sur sa tuile, parce que l'on a cliqué sur une notification ou encore parce que l'application redémarre suite à une suspension. Auquel cas, on pourra restaurer le contexte dans lequel on était lorsque l'application a été suspendue. De même, nous allons remplacer le App.xaml avec ce code :

```
<common:BootStrapper
  x:Class="FiveHundredPxGallery.App"
  xmlns:common="using:Template10.Common"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:FiveHundredPxGallery"
  xmlns:vm="using:FiveHundredPxGallery.ViewModels">

</common:BootStrapper>
```

Bien sûr, adaptez le namespace à votre application. Maintenant il faut créer une classe qui va permettre à la page de trouver son viewmodel. C'est le rôle du ViewModelLocator:

```
public class ViewModelLocator
{
  static ViewModelLocator()
  {
    ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

    SimpleIoc.Default.Register<MainViewModel>();
  }

  public MainViewModel Main => ServiceLocator.Current.GetInstance<MainViewModel>();
}
```

Nous utilisons ici SimpleIoc, c'est l'injection de dépendance fournie avec MvvmLight. Si vous ne savez pas ce qu'est l'injection de dépendance, je vous conseille de vous renseigner dessus. De manière globale, cela correspond au D des principes SOLID que tout développeur de programmation orientée objet doit connaître. Cela n'étant pas l'objet de cet article, je vous laisse vous informer par vous-même sur ce sujet.
Intégrons ce ViewModelLocator dans le App.xaml

```
<common:BootStrapper
  [...]>

  <Application.Resources>
    <ResourceDictionary>
      <vm:ViewModelLocator x:Key="Locator" />
    </ResourceDictionary>
  </Application.Resources>

</common:BootStrapper>
```

Si vous avez fait attention, nous avons défini une propriété Message dans notre MainPageViewModel, nous allons l'utiliser maintenant dans notre MainPage.

```
<Page
  x:Class="FiveHundredPxGallery.Views.MainPage"
  [...]
  DataContext="{Binding Source={StaticResource Locator}, Path=MainViewModel}">

  <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <TextBlock Text="{Binding Message}" HorizontalAlignment="Center"
        VerticalAlignment="Center" />
  </Grid>
</Page>
```

Vous pouvez maintenant lancer votre application et vous devriez voir le message au centre de la fenêtre.
Le TextBlock utilise le binding classique. Si vous voulez utiliser le nouveau x:Bind, il faut que votre code DataContext soit fortement typé. Allez dans le fichier MainPage.xaml.cs et ajoutez cette ligne :

```
public MainViewModel ViewModel => this.DataContext as MainViewModel;
```

Notez au passage la nouvelle notation C# 6, on peut désormais affecter une propriété readonly avec une expression lambda. Avec ce ViewModel fortement typé nous pouvons maintenant modifier le binding du TextBlock :

```
<TextBlock Text="{x:Bind ViewModel.Message}" HorizontalAlignment="Center" Vertical
Alignment="Center" />
```

Intégrons maintenant une couche service. Celle-ci doit nous abstraire l'accès aux données. Tout d'abord ajoutez un dossier Mvvm et collez-y cette classe qui sera notre nouveau ViewModelBase :

```
public abstract class ViewModelBase : GalaSoft.MvvmLight.ViewModelBase, Template10.Services.NavigationService.INavigable
{
  public virtual void OnNavigatedTo(object parameter, NavigationMode mode, IDictionary<string, object> state) { /* nothing by default */ }
  public virtual Task OnNavigatedFromAsync(IDictionary<string, object> state, bool suspending) => Task.FromResult<object>(null);
  public virtual void OnNavigatingFrom(Template10.Services.NavigationService.NavigatingEventArgs args) { /* nothing by default */ }

  [JsonIgnore]
  public Template10.Services.NavigationService.INavigationService NavigationService { get; set; }

  [JsonIgnore]
  public Template10.Common.IDispatcherWrapper Dispatcher { get; set; }

  [JsonIgnore]
  public Template10.Common.IStateItems SessionState { get; set; }
}
```

Vous pouvez utiliser celle-ci dans votre MainViewModel. Elle remplace celle de Template10 en reprenant celle de MvvmLight et y intégrant les éléments importants de Template10 (notamment le NavigationService). Ajoutons un dossier Models et un dossier Services à notre arborescence. Ajoutez cette classe dans Models :

```
public class Item
{
  public string Text { get; private set; }

  public Item(string text)
  {
    Text = text;
  }
}
```

Ajoutez dans Services cette interface ainsi que ces deux classes :

```
public interface IItemService
{
  Task<Item> GetItem();
}

public class ItemService : IItemService
{
  public Task<Item> GetItem()
  {
    return Task.FromResult(new Item("Real live data"));
  }
}

public class DesignItemService : IItemService
{
  public Task<Item> GetItem()
  {
    return Task.FromResult(new Item("Fake design data"));
  }
}
```

Nous avons un modèle et un service pour récupérer ces données. Ajoutez ce service à notre injection de dépendance pour pouvoir l'utiliser :

```
public class ViewModelLocator
{
  static ViewModelLocator()
  {
    ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

    if (ViewModelBase.IsInDesignModeStatic)
    {
      SimpleIoc.Default.Register<IItemService, DesignItemService>();
    }
    else
    {
      SimpleIoc.Default.Register<IItemService, ItemService>();
    }
    SimpleIoc.Default.Register<MainViewModel>();
  }

  public MainViewModel Main => ServiceLocator.Current.GetInstance<MainViewModel>();
}
```

Vous pouvez maintenant utiliser ce service dans notre MainViewModel :

```
public MainViewModel(IItemService itemService)
{
  _itemService = itemService;

  Task.Run(() => Initialize()).Wait();
}

private async Task Initialize()
{
  try
  {
    var item = await _itemService.GetItem();
    Message = item.Text;
  }
  catch (Exception ex)
  {
    Message = ex.Message;
  }
}
```

Le message qui s'affiche maintenant dans notre interface provient de notre service. Attention si vous utilisez x:Bind, celui-ci n'est pas compatible avec les données design, et le "Fake design data" n'apparaît pas. C'est une limitation connue du x:Bind. Pour l'instant on ne sait pas si celui-ci pourra les utiliser. Vous pouvez utiliser {Binding Message} à la place et vous verrez vos données design. Un uservoice a été lancé pour l'occasion : https://visualstudio.uservoice.com/forums/121579-visual-studio-2015/suggestions/9883587-add-design-time-support-for-x-bind-syntax

Voilà, tout est en place pour construire une application digne de ce nom. Dans le prochain numéro, nous verrons comment implémenter notre client 500px. Nous mettrons aussi en place une collection d'images à chargement incrémentiel et une page de détail pour chacune des images.

La programmation Windows, le retour Partie 1/5 : Le ribbon Windows

Nous vous proposons une série d'articles sur la pratique de la programmation Windows en C++. Cette nouvelle série se veut accessible à tous dans le but de faire des applications modernes. Ce premier article est consacrée au « Ribbon » alias le Ruban Windows que vous trouvez dans les applications comme MSPaint ou la suite Office.

Christophe PICHAUD | .NET Rangers by Sogeti
Consultant sur les technologies Microsoft
christophepichaud@hotmail.com | www.windowscpp.net

Le Ruban est apparu avec Microsoft Office 2007 et c'est une nouvelle expérience utilisateur qui permet de changer la présentation des anciens menus et des barres d'outils (toolbars). Avec le temps, Microsoft a constaté que les menus devenaient de plus en plus gros et que trouver une fonctionnalité devenait parfois difficile. Ils ont donc introduit ce nouveau paradigme sur l'interface UI. Et là, vous vous dites, est-ce que je peux mettre un ruban dans mon application ? Oui c'est possible et en natif. Il existe même deux solutions natives pour faire un ruban. Il y a la façon Win32 avec des API COM et il y a la version C/C++ disponible dans les MFC aka Microsoft Foundation Classes. Fig.1 et 2.

En effet depuis Visual Studio 2008 SP1, Microsoft a introduit le MFC Feature Pack qui contient de nombreuses classes C++ pour les MFC qui permettent de faire des applications exceptionnelles : support du Ruban, support des Docking Panes, Properties Windows, Options Windows. Bref, toutes ces fonctionnalités qui permettent de faire des applications modernes sont disponibles à portées de main avec leur code source car les MFC contiennent les headers .h et leur code source .cpp pratique pour le debugging avancé. On sait ce que fait le code. Pas de mystères. Pour montrer à quel point les API MFC sont poussées au maximum au niveau des détails, voici le ruban de la démo de l'application MSOffice2007Demo. Voici un extrait du ruban. On ne remarquera pas la différence avec le vrai. Fig.3.

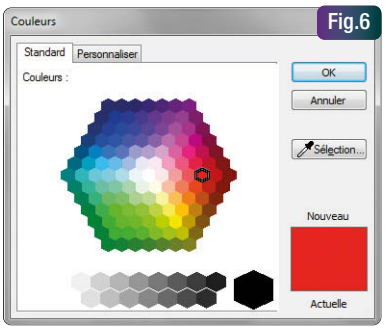


Fig.6

Avant de voir le code, il faut parler de l'architecture du ruban. C'est plus qu'une grosse toolbar. C'est un élément contextuel qui permet de tester des opérations sans applications définitives. Je m'explique : vous passez la souris au dessus d'une couleur, et votre objet graphique sélectionné change automatiquement de couleur rien qu'avec le déplacement de la souris. Et si vous clickez, le changement s'opère. Fig.4 et 5.

Regardez comme ces panneaux graphiques apportent de multiples fonctionnalités. Il existe même une fonctionnalité « Plus de Couleurs... » pour afficher une boîte dédiée à la sélection de couleur. Fig.6.

Tout est concentré dans des éléments graphiques simples à visualiser mais complexes à implémenter sans un léger coup de main. Et c'est là que la magie du C++ entre en scène. Il existe une classe C++ par fonctionnalité de ruban. Eh oui, il est possible de coder tout cela en C++ sans avoir besoin de développer des montagnes de code. Comment ?

Commençons par lancer Visual Studio et demandons un nouveau projet MFC. Au niveau des options, il faut cocher un projet de type Office avec un Ruban. Fig.7. Une fois l'assistant fini, il suffit de compiler le code qui a été produit et voilà le résultat : Fig.8.

Ouvrons le fichier de ressources RC et admirons le ruban en mode édition : Fig.9. Et là vous me dites : « cool c'est simple y a un éditeur ! ». Je vous arrête de suite... Ce n'est pas comme cela que l'on fera le ruban de Word,

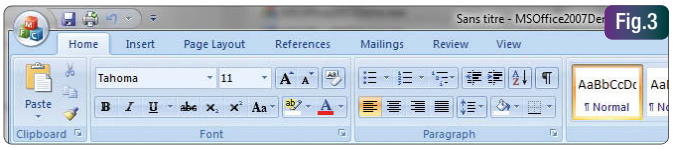


Fig.3

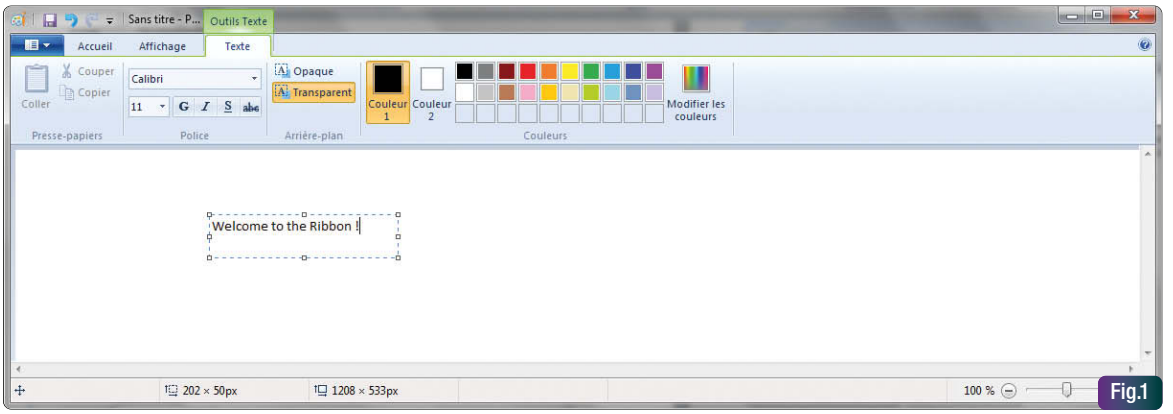


Fig.1

Fig.4

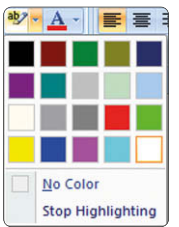


Fig.5

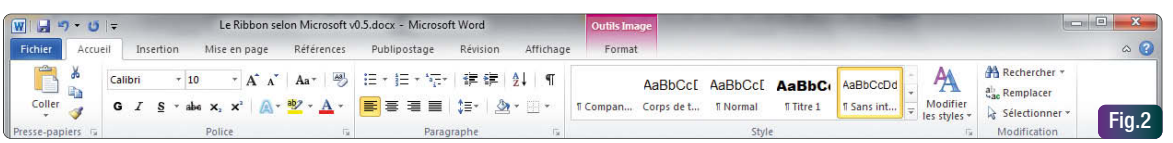


Fig.2

via l'éditeur. Nous allons construire le ruban tout en programmation pour supporter l'ensemble des gabarits disponibles. On va faire du code ! Je vous avais prévenu, le code Level 100 ce n'est pas aujourd'hui ! Et puis avouez que c'est plus marrant de faire du code non ?

Coder le ruban

Voici le ruban que nous allons créer : c'est une application existante que j'ai écrite pour que ma fille joue à la souris en faisant du dessin. Fig.10.
On commence par où ? Eh bien par le seul item qui ressemble de près ou de loin à un menu. Fig.11.
La première chose à faire c'est de créer l'objet ribbon : c'est le parent de tous les objets que nous allons créer.

```
CMFCRibbonBar    m_wndRibbonBar;
if (!m_wndRibbonBar.Create(this))
{
    return FALSE;
}
```

Ensuite, pour chaque élément CMFCRibbonButton, on fait un CMFCRibbonMainPanel::Add.

```
CMFCRibbonMainPanel* pMainPanel = m_wndRibbonBar.AddMainCategory(_T("File"), IDB_RIBBON_FILESMALL, IDB_RIBBON_FILELARGE);
pMainPanel->Add(new CMFCRibbonButton(ID_FILE_NEW, _T("&New"), 0, 0));
pMainPanel->Add(new CMFCRibbonButton(ID_FILE_OPEN, _T("&Open..."), 1, 1));
pMainPanel->Add(new CMFCRibbonButton(ID_FILE_OPEN_AS_XML, _T("Open As XML..."), 1, 1));
pMainPanel->Add(new CMFCRibbonButton(ID_FILE_SAVE, _T("&Save"), 2, 2));
pMainPanel->Add(new CMFCRibbonButton(ID_FILE_SAVE_AS, _T("Save &As..."), 3, 3));
pMainPanel->Add(new CMFCRibbonButton(ID_FILE_SAVE_AS_XML, _T("Save As & XML..."), 3, 3));
pMainPanel->AddSeparator();
pMainPanel->Add(new CMFCRibbonButton(ID_FILE_CLOSE, _T("&Close"), 8, 8));
pMainPanel->AddRecentFilesList(_T("Recent Documents"));
pMainPanel->AddToBottom(new CMFCRibbonMainPanelButton(ID_APP_EXIT, _T("E&xit"), 27));
```

Les numéros indiquent l'icône à utiliser par rapport à la ressource graphique IDB_RIBBON_FILESMALL et IDB_RIBBON_FILLARGE. Voilà pour le pave carré en haut à gauche du ruban.

Exemple du Ruban

Maintenant, il faut coder chaque bloc avec le détail nécessaire à l'architecture du ruban. Fig.12.
Dans cet exemple, « Home » est une catégorie de type CMFCRibbonCategory.
« Design » est un panel de type CMFCRibbonPanel.

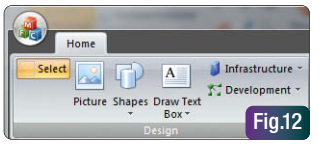


Fig.12

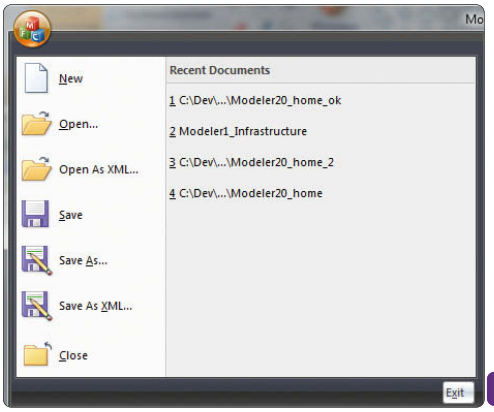


Fig.11

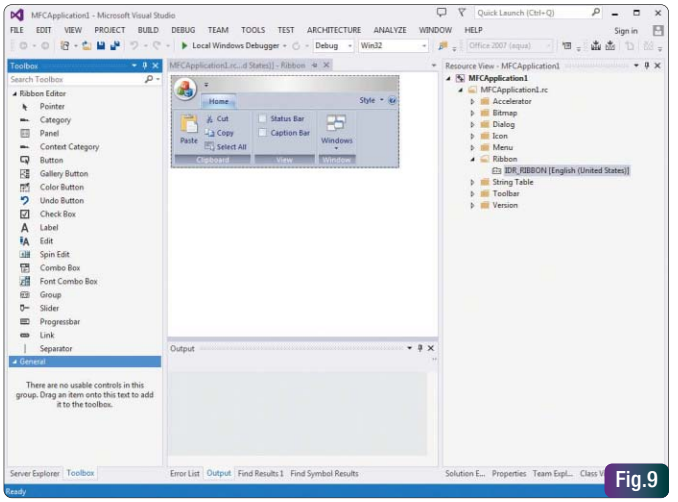


Fig.9

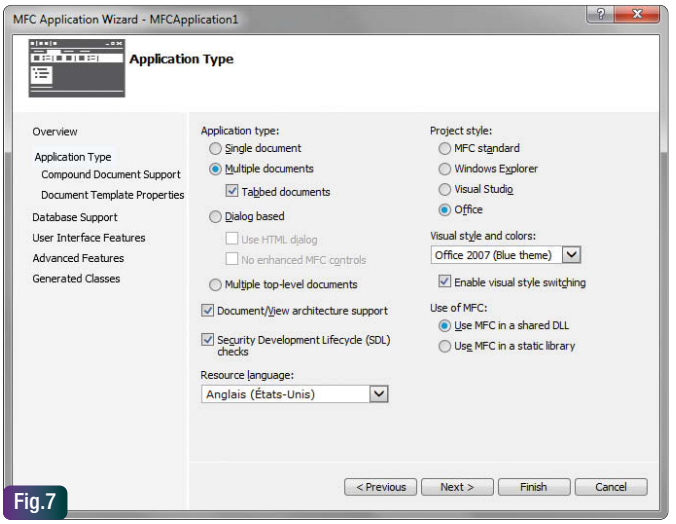


Fig.7

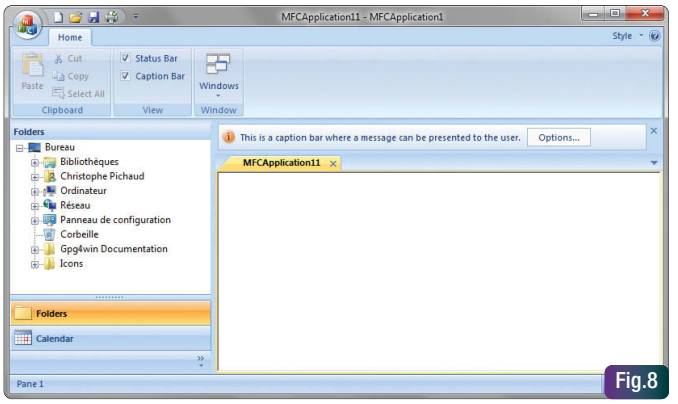


Fig.8

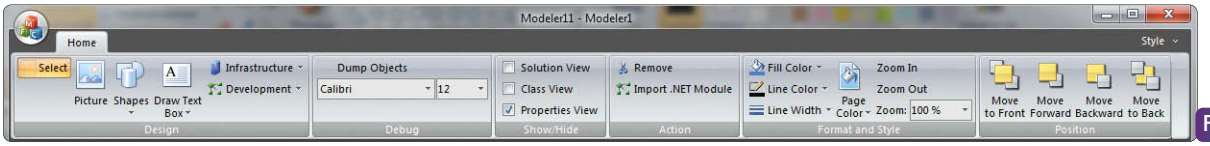


Fig.10

```
CMFCRibbonCategory* pCategory = m_wndRibbonBar.AddCategory(_T("&Home"), IDB
_RIBBON_WRITESMALL, IDB_RIBBON_WRITELARGE);
CMFCRibbonPanel* pPanelDesign = pCategory->AddPanel(_T("Design\nzd"), m_PanelI
mages.ExtractIcon(2));
```

Dans le panel "Design" on va ajouter des éléments. Par exemple, le bouton « Picture » :

```
CMFCRibbonButton* pInsertPictureBtn = new CMFCRibbonButton(ID_DESIGN_IMAGE, _T
("Picture"), 4, 4);
pPanelDesign->Add(pInsertPictureBtn);
```

L'icône associée au bouton provient de l'icône n°4 (index 0-based) de la ressource graphique IDB_RIBBON_WRITELARGE associée à writelarge.png dans le fichier .RC :

```
IDB_RIBBON_WRITELARGE  PNG              "res\\writelarge.png"
```

Fig.13.

Ce sont des images 32x32 pixels dans le même fichier. Un conseil : manipulez GIMP avec ce genre de chose car MSPaint devient très vite limité quand cela se joue au pixel prêt... Un conseil...

Ajoutons le bouton Shapes qui est une galerie. Fig.14.

C'est une galerie de type CMFCRibbonGallery.

```
CMFCRibbonGallery *pInsertShapesBtn = new CMFCRibbonGallery(ID_DESIGN_SHAPES,
_T("Shapes"), 6, 6);
pInsertShapesBtn->SetButtonMode();
pInsertShapesBtn->SetIconsInRow(12);
pInsertShapesBtn->AddGroup(_T("Recently Used Shapes"), IDB_SHAPE1, 20);
pInsertShapesBtn->AddGroup(_T("Lines"), IDB_SHAPE2, 20);
pInsertShapesBtn->AddGroup(_T("Basic Shapes"), IDB_SHAPE3, 20);
pInsertShapesBtn->AddGroup(_T("Block Arrows"), IDB_SHAPE4, 20);
pInsertShapesBtn->AddGroup(_T("Flowchart"), IDB_SHAPE5, 20);
pInsertShapesBtn->AddGroup(_T("Callouts"), IDB_SHAPE6, 20);
pInsertShapesBtn->AddGroup(_T("Stars and Banners"), IDB_SHAPE7, 20);
pInsertShapesBtn->AddSubItem(new CMFCRibbonButton(ID_DESIGN_SHAPES_NEW,
_T("&New Drawing Canvas"), 29, -1));
pPanelDesign->Add(pInsertShapesBtn);
```

Chaque ligne est une série de bitmaps enregistrés dans les ressources. Ex : IDB_SHAPE1 :

```
IDB_SHAPE1      BITMAP DISCARDABLE   "res\\shape1.bmp"
```

Ajoutons un menu sous forme de bouton : Fig.15.

Il suffit d'utiliser la classes CMFCRibbonButton et de faire appel à la méthode SetMenu().

```
CMFCRibbonButton* pBtnDrawText = new CMFCRibbonButton(ID_DESIGN_TEXTBOX,
_T("Draw Text Box\nx"), 15, 15);
pBtnDrawText->SetMenu(IDR_DRAW_TEXT_MENU);
pPanelDesign->Add(pBtnDrawText);
```

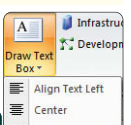


Fig.15

Fig.13

IDR_DRAW_TEXT_MENU est un menu défini dans le fichier de ressource RC. Ajoutons une liste de gabarits : Fig.16.

Il suffit d'utiliser la class CRibbonListButton et d'y ajouter une image et une liste des chînes de caractères. Fig.17.

```
CStringArray m_arInfraShapes;
m_arInfraShapes.Add(_T("AD Server"));
m_arInfraShapes.Add(_T("Server"));
m_arInfraShapes.Add(_T("Web Server "));
m_arInfraShapes.Add(_T("Database Server"));
m_arInfraShapes.Add(_T("Workstation"));
m_arInfraShapes.Add(_T("Laptop"));
m_arInfraShapes.Add(_T("Firewall"));
m_arInfraShapes.Add(_T("Network"));
m_arInfraShapes.Add(_T("Virtual Server"));
m_arInfraShapes.Add(_T("Virtual Web Server"));
m_arInfraShapes.Add(_T("Virtual Database Server"));
m_arInfraShapes.Add(_T("Virtualization Server"));
m_arInfraShapes.Add(_T("AD Server"));
m_arInfraShapes.Add(_T("Server"));
m_arInfraShapes.Add(_T("Database Server"));
m_arInfraShapes.Add(_T("Server Farm"));
m_arInfraShapes.Add(_T("Workstation"));
m_arInfraShapes.Add(_T("laptop"));
```

```
CRibbonListButton *pListBtnInfra = new CRibbonListButton(ID_DESIGN_SHAPESINFRA,
_T("Infrastructure\nti"), 20, -1, FALSE);
pListBtnInfra->AddGroup(_T("Built-In"), IDB_SHAPES_INFRA, 64, m_arInfraShapes);
pListBtnInfra->SetIconsInRow(4);
pListBtnInfra->EnableMenuResize();
pPanelDesign->Add(pListBtnInfra);
```

L'élément suivant du ruban est aussi un ensemble de gabarits. Il suffit de garder le même code mais de changer la liste de chaînes de caractères et l'image. Fig.18 et 19. Ajoutons la combobox des polices et la combobox de taille : Fig.20. Il suffit d'utiliser la classe CMFCRibbonFontComboBox pour afficher les polices. La combobox pour les tailles est une simple utilisation de la classe CMFCRibbonComboBox.

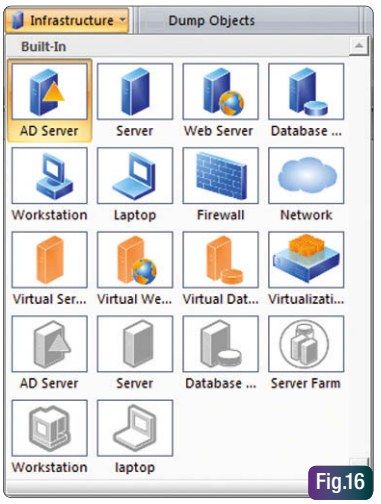


Fig.16

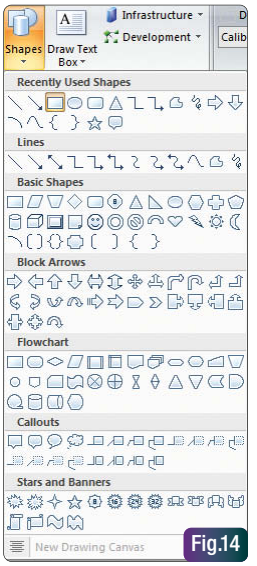


Fig.14

Fig.17

```
// Create "Debug" panel
CMFCRibbonPanel* pPanelDebug = pCategory->AddPanel(_T("Debug\nzd"), m_PanelImages.
ExtractIcon(2));

CMFCRibbonButtonsGroup * apFontGroup = new CMFCRibbonButtonsGroup();

CMFCRibbonFontComboBox::m_bDrawUsingFont = TRUE;

m_pFontCombo = new CMFCRibbonFontComboBox(ID_FONT_FONT, TRUETYPE_FONTTYPE );
m_pFontCombo->SetWidth(55, TRUE); // Width in "floaty" mode
m_pFontCombo->SelectItem(_T("Arial"));
apFontGroup->AddButton(m_pFontCombo);

m_pFontSizeCombo = new CMFCRibbonComboBox(ID_FONT_FONTSIZE, FALSE, 39);
m_pFontSizeCombo->AddItem(_T("8"));
m_pFontSizeCombo->AddItem(_T("9"));
m_pFontSizeCombo->AddItem(_T("10"));
m_pFontSizeCombo->AddItem(_T("11"));
m_pFontSizeCombo->AddItem(_T("12"));
m_pFontSizeCombo->AddItem(_T("14"));
m_pFontSizeCombo->AddItem(_T("16"));
m_pFontSizeCombo->AddItem(_T("18"));
m_pFontSizeCombo->AddItem(_T("20"));
m_pFontSizeCombo->SetWidth(20, TRUE); // Width in "floaty" mode
m_pFontSizeCombo->SelectItem(3);
apFontGroup->AddButton(m_pFontSizeCombo);

pPanelDebug->Add(apFontGroup);
```

Ajoutons une combobox : **Fig.21.**
Il suffit d'utiliser la classe CMFCRibbonCheckBox.

```
// Create "Show/Hide" panel:
CMFCRibbonPanel* pPanelShow = pCategory->AddPanel(_T("Show/Hide\nzs"), m_PanelImages.
ExtractIcon(4));
pPanelShow->Add(new CMFCRibbonCheckBox(ID_VIEW_FILE_VIEW, _T("Solution View\nc")));
pPanelShow->Add(new CMFCRibbonCheckBox(ID_VIEW_CLASS_VIEW, _T("Class View\nc")));
pPanelShow->Add(new CMFCRibbonCheckBox(ID_VIEW_PROPERTIES, _T("Properties View\np")));
```

Ajoutons un bouton pour choisir la couleur de fond de l'écran : **Fig.22.**
Il suffit d'utiliser la classe CMFCRibbonColorButton et d'y passer en paramètres des couleurs de palette. Il y a 3 palettes mais le code ici n'en présente qu'une. Le code est identique sauf qu'il y a 50 couleurs en plus...

```
CList<COLORREF,COLORREF> m_lstMainColors;

static ColorTableEntry colorsMain [] =
{
    { RGB(255, 255, 255), _T("White, Background 1") },
    { RGB(0, 0, 0), _T("Black, Text 1") },
    { RGB(238, 236, 225), _T("Tan, Background 2") },
    { RGB(31, 73, 125), _T("Dark Blue, Text 2") },
    { RGB(79, 129, 189), _T("Blue, Accent 1") },
    { RGB(192, 80, 77), _T("Red, Accent 2") },
    { RGB(155, 187, 89), _T("Olive Green, Accent 3") },
    { RGB(128, 100, 162), _T("Purple, Accent 4") },
    { RGB(75, 172, 198), _T("Aqua, Accent 5") },
    { RGB(245, 150, 70), _T("Orange, Accent 6") }
};

nNumColours = sizeof(colorsMain) / sizeof(ColorTableEntry);

for (i = 0; i < nNumColours; i++)
{
    m_lstMainColors.AddTail(colorsMain [i].color);
    CMFCRibbonColorButton::SetColorName(colorsMain [i].color, colorsMain [i].szName);
};

CMFCRibbonColorButton * pBtnPageColor = new CMFCRibbonColorButton(ID_FORMAT_PAGECOLOR,
_T("Page Color\npo"), TRUE, -1, 26);
pBtnPageColor->SetDefaultCommand(FALSE);
pBtnPageColor->EnableAutomaticButton(_T("&Automatic"), RGB(0, 0, 0));
pBtnPageColor->EnableOtherButton(_T("&More Line Colors..."), _T("More Line Colors"));
pBtnPageColor->SetColumns(10);
pBtnPageColor->SetColorBoxSize(CSize(17, 17));
pBtnPageColor->AddColorsGroup(_T("Theme Colors"), m_lstMainColors);
pBtnPageColor->AddColorsGroup(_T(""), m_lstAdditionalColors, TRUE);
pBtnPageColor->AddColorsGroup(_T("Standard Colors"), m_lstStandardColors);
pPanelFormat->Add(pBtnPageColor);
```

Ajoutons les deux boutons « ZoomIn » et « ZoomOut » :
Il suffit d'utiliser la class CMFCRibbonButton.

```
pPanelFormat->Add(new CMFCRibbonButton(ID_FORMAT_ZOOM_IN, _T("Zoom In\ni"), -1));
pPanelFormat->Add(new CMFCRibbonButton(ID_FORMAT_ZOOM_OUT, _T("Zoom Out\no"), -1));
```

Vous remarquerez que le -1 indique ne pas prendre d'icônes.

Conclusion

Construire un Ruban par programmation n'est pas si compliqué que cela car nous disposons de classes MFC qui font le job. La mécanique est toujours la même : on passe un identifiant d'image ou un numéro d'index sur un bitmap multiple et le tour est joué. Le codage d'un Ribbon comme celui de Office Word 2007 par exemple est disponible dans « Microsoft Visual C++ 2008 SP1 Sample Library »:
http://www.microsoft.com/fr-fr/download/details.aspx?id=5718
Vous télécharger vc_samples qui fait 8 MB et une fois installé, vous allez dans le répertoire C++\MFC\Visual C++ 2008 Feature Pack et vous tomberez sur les exemples suivants : MSOffice2007Demo, VisualStudioDemo, OutlookDemo, etc. Le code complet du Ribbon de cet article est accessible sur codeplex à l'URL suivante : http://ultrafluid.codeplex.com/

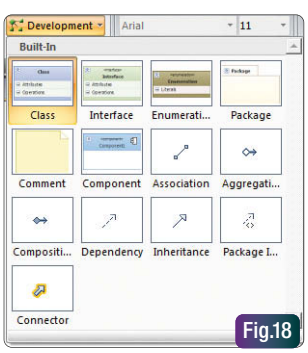


Fig.18

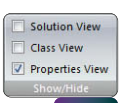


Fig.21

Fig.20

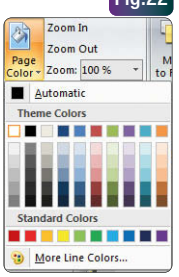


Fig.22


Fig.19

Construire son drone

3e partie

Si vous avez bien suivi les deux premières parties de cet article, votre drone est désormais quasiment prêt pour le décollage ! Vous pouvez ainsi le contrôler manuellement, et ses capacités par défaut en font un appareil autonome dans certaines situations. Dans cette troisième et dernière partie, vous découvrirez comment choisir et installer votre batterie, et enfin, comment interagir avec le multicopter.

François Chevalier
Concepteur développeur chez SQLI Enterprise à Nantes depuis 2011. Passionné par les nouvelles technologies et l'aéronautique, il s'intéresse au concept du Do It Yourself et essaye de l'appliquer dans son quotidien.

UNE LIPO ET ÇA DÉCOLLE...

Afin d'alimenter l'ensemble des systèmes de l'appareil, vous devez équiper votre drone d'une batterie. L'idéal, particulièrement lorsque l'on utilise des moteurs de type *brushless*, est d'utiliser le Lithium Polymère, plus communément appelé LiPo. Ce type de batterie a pour avantage de conserver l'intégralité de sa puissance tout au long de la décharge sans avoir d'effet palier.

Les cellules ("S")

Les batteries de type LiPo sont composées d'une ou plusieurs cellules nommées "S". Chacune de ces cellules a une tension nominale (c'est-à-dire une tension minimale utile, soit 80% de la décharge de la batterie) de 3,7V. Par exemple, pour une batterie LiPo de type 3S, la tension nominale est de 3 cellules x 3,7V, soit 11,1V.

Soyez vigilants : la tension maximale d'une cellule d'une batterie LiPo est de 4,20V lorsque la batterie est chargée à 100%. Au-delà de cette tension ou si la tension d'une cellule est trop basse (en général en dessous de 3,3V), vous risquez d'endommager la batterie.

J'insiste sur ce point car il est important de comprendre que l'utilisation de batterie de type LiPo n'est pas anodin : un dépassement de la charge maximale peut entraîner un incendie de la batterie. A l'inverse, une décharge trop importante peut rendre inutilisable la batterie ou entraîner un phénomène de surchauffe qui endommagerait l'appareil sur lequel elle est fixée. Pour ces différentes raisons, je conseille d'utiliser un chargeur spécifique pour charger une batterie LiPo, mais j'y reviendrai un peu plus tard. Attention enfin, soyez précautionneux lors de sa manipulation : une batterie qui a reçu un choc peut devenir inutilisable.

Le nombre de cellules de votre batterie dépendra des moteurs que vous avez choisis. Vous trouverez dans les caractéristiques de ceux-ci le nombre de cellules qu'il leur faut pour fonctionner. Dans mon cas, les moteurs que j'ai choisis doivent fonctionner avec une batterie ayant au minimum 4 cellules et au maximum 8 (4S - 8S). J'ai choisi d'utiliser une batterie LiPo 4S car le système que j'utilise pour alimenter mon contrôleur de vol ne supporte pas des tensions supérieures à ce que fournissent les batteries avec plus de 4 cellules.

La capacité, la clé de votre autonomie

La deuxième caractéristique pour choisir votre batterie est sa capacité, exprimée en mAh. Plus la capacité est élevée, plus l'autonomie de votre batterie sera grande... et plus elle sera lourde. A vous de jouer pour trouver le bon compromis entre autonomie et poids. Pour ma part, j'ai fait le choix d'une batterie d'une capacité de 5100 mAh et je pense, d'après mes calculs, pouvoir dépasser les 10 minutes d'autonomie. En effet, partant du principe que la consommation des systèmes électroniques tels que le contrôleur de vol et autres GPS ne représente qu'un infime pourcentage de la consommation réelle du drone (et donc que les moteurs consomment 99% de l'électricité) je fais le calcul suivant :

- Un moteur (T-Motor Mt4008-18) qui tourne à 85% de sa puissance avec une hélice de 15" de diamètre consommera 5,6A selon les données du constructeur.
- Mon drone est composé de 4 moteurs donc la consommation totale sera d'environ 4 x 5,6A = 22,4A.
- Si on prend la formule temps(t) = Capacité(Q) / Intensité(I) on obtient le calcul t = 5,1(Ah) / 22,4(A) = 0,23(h), ce qui nous donne en minute : 13,8 minutes (0,23 x 60).

Cette durée n'est évidemment que théorique : chaque moteur ne vas pas forcément tourner constamment à 85% de sa puissance maximum et surtout la batterie ne doit pas être déchargée à plus de 85% de sa capacité.

Le taux de décharge

La dernière caractéristique à prendre en compte dans le choix de votre batterie est son taux de décharge. Il est, en général, indiqué sur la batterie en C (capacité), mais il est aussi parfois exprimé explicitement en Ampère. Il correspond au courant (A) que la batterie peut fournir en continu lors de sa décharge. Ma batterie LiPo de 5100 mAh de capacité, dont la valeur est de 35C, fournit un courant égal à 35 fois sa capacité soit 178,5A (35 x 5100 mA = 178,5A).

Le taux de décharge est un élément à prendre en compte car il permet de s'assurer que la batterie sera capable de fournir un courant continu suffisant pour faire fonctionner les moteurs et l'électronique. En effet, imaginez la mauvaise surprise en vol si la batterie ne peut plus fournir suffisamment de courant !

Vous trouverez chez les différents vendeurs des batteries de toutes les marques avec des connecteurs différents. Veillez à vérifier quel type de connecteur est soudé sur votre batterie afin de pouvoir adapter votre montage avec le bon connecteur. En général, vous avez deux choix qui s'offrent à vous si votre batterie a un connecteur différent du reste de votre montage : soit vous coupez et soudez un nouveau connecteur que vous aurez pris le soin d'acheter, soit vous achetez un adaptateur qui fera la jonction entre les deux types de connecteurs différents.

En ce qui concerne les marques de batterie, les

mieux réputées mais aussi les plus chères sont les marques Tattu, Hyperion et Kypom. Ces batteries ont une bonne durée de vie, et les valeurs annoncées au niveau de la capacité et du taux de décharge sont fiables. D'autres marques très répandues car moins chères comme Turnigy ou Zippy semblent avoir un bon rapport qualité / prix, mais il semblerait que les valeurs annoncées soit un peu surestimées. Je vous conseille donc de vous renseigner avant d'acheter une batterie d'une marque inconnue afin d'éviter les mauvaises surprises.

Un chargeur sachant charger...

Pour charger votre batterie LiPo, équipez-vous d'un chargeur spécifique, permettant au notamment de surveiller le voltage au fur et à mesure de la charge, ceci afin de ne pas dépasser les limites critiques d'une part, et d'équilibrer la charge entre les cellules de la batterie d'autre part.

Il existe de nombreuses marques de chargeur avec plus ou moins de fonctionnalités et de sécurité, pour toutes les bourses. Pensez, lors de l'achat, à vérifier deux aspects :

- Le chargeur doit être compatible avec votre batterie, autrement dit capable de charger le nombre de cellules de votre batterie,
- La connectique doit être la bonne si par défaut votre chargeur n'a pas les bonnes fiches.

La charge doit s'effectuer de préférence en 1C, c'est-à-dire que l'ampérage lors de la charge doit correspondre à la capacité de la batterie. Pour ma batterie d'une capacité de 5100 mAh, je charge à 5,1A. Certains constructeurs annoncent que l'on peut charger jusqu'à 5C. Cependant, il est conseillé de ne pas le faire afin de préserver la durée de vie de votre batterie.

Enfin, gardez toujours un œil sur le processus de charge afin de pouvoir agir en cas de problème : n'oubliez pas que l'utilisation de ces batteries n'est pas sans risque !

4...3...2...1... Décollage !

Désormais équipés d'une batterie chargée, vous devez lui trouver une place sur (ou sous) votre multicopter. Vérifiez bien les dimensions de votre batterie d'une part, et les dimensions de la place que vous lui avez réservée d'autre part. Ayant mal anticipé ce point, j'ai dû improviser un bricolage pour pouvoir installer la batterie sous mon drone.

Une fois la batterie fixée, il est temps de vous préparer pour votre premier vol ! Voici les étapes-clés de calibrage et paramétrage à respecter :

- Assurez-vous que tous les éléments électroniques ont bien été calibrés (Accéléromètre, boussole, gyroscope, etc.),
- Vérifiez que le calibrage de la radiocommande à bien été réalisé et vérifiez si vous avez besoin ou non d'inverser la commande correspondant au pitch. En effet, selon les modèles de contrôleurs de vol, il est possible que le drone entame un déplacement vers l'arrière lorsque vous pousserez la manette pour avancer,
- Vérifiez que les ESC sont tous bien calibrés : lorsque vous jouez avec la manette des gaz, les moteurs doivent réagir en même temps et tourner à la même vitesse. On peut le voir facilement lorsque l'on remet les gaz à zéro : si les ESC ne sont pas tous calibrés de la même façon, un ou des moteurs tourneront pendant que les autres seront à l'arrêt. Si votre contrôleur de vol le permet, effectuez le calibrage sur les 4 ESC simultanément. Sinon, faites le ESC par ESC. Des tutoriels disponibles sur Internet vous aideront à réaliser le calibrage, des différences existent selon que vous passiez par le contrôleur de vol ou que vous branchiez le récepteur de la radio commande directement sur l'ESC à calibrer.
- Si votre contrôleur de vol implémente les fonctions Return To Launch (RTL) ou Loiter, paramétrez-les sur votre radiocommande : lors des premiers vols, ces deux fonctions peuvent vous être très utiles en cas de perte de repère ou si le contrôle devient un peu trop difficile.

Ainsi, vous êtes prêts pour votre premier vol de test ! Si vous voulez voir le mien, ça se passe à l'adresse suivante : http://youtu.be/xLndW0g0q5o

ALLO HOUSTON, ICI LE DRONE !

Dès lors que notre drone se retrouve dans les airs, on ne peut se fier qu'à nos yeux pour le diriger et évaluer les situations dans lesquelles il se trouve. De plus, construire un drone implique une recherche d'autonomie. J'aborde donc dans cette partie le sujet de la télémétrie, qui nous permettra non seulement d'avoir un feedback du contrôleur de vol, mais également de créer des missions autonomes... et de faire un peu de programmation !

Il existe plusieurs façons de recevoir et d'envoyer des informations à votre drone depuis votre contrôleur de vol. Je n'en évoquerai que deux, en lien avec le contrôleur de vol que j'ai décidé d'utiliser, en l'occurrence le contrôleur *Pixhawk*. Je vous invite à vous renseigner sur les possibilités qu'offre le contrôleur que vous aurez choisi.

Ceci étant dit, voici les options de télémétrie qui s'offrent à nous :

- Le module de télémétrie radio : ce module est composé de deux parties, une première partie destinée à être installée sur votre multicopter et qui se branche sur le port "Telem 2" de votre contrôleur *Pixhawk*. Ce module sera alimenté par le contrôleur et vous n'aurez rien d'autre à faire que de le brancher et de le fixer. La deuxième partie est équipée d'une prise USB à brancher sur votre ordinateur. Lorsque les deux parties sont alimentées respectivement par le drone et par l'ordinateur, l'appairage se fait automatiquement et vous aurez ainsi un canal de communication ouvert entre les deux. Libre à vous d'utiliser un logiciel dédié ou de tenter de programmer directement via le langage MavLink via ce canal. Une chose

importante à savoir sur ce module est qu'il existe deux versions avec des fréquences différentes. Une version à 915 Mhz pour les États-Unis et une version à 433 Mhz pour le reste du monde. Ce n'est pas un détail : j'ai fait l'erreur de commander un module avec une mauvaise fréquence, et je possède donc des appareils que je ne peux utiliser sous peine de brouiller certaines fréquences de téléphonie portable.

- Un Raspberry Pi : cette carte a de nombreux avantages et permet, par son interface GPIO, de se connecter elle aussi au port "Telem 2" afin d'échanger avec le contrôleur de vol. De préférence, utilisez un module USB WiFi pour communiquer avec un ordinateur au sol. Vous pouvez également communiquer avec un autre type de périphérique tel qu'un smartphone, lui aussi équipé du WiFi (la procédure pour se connecter et communiquer est un peu plus complexe dans ce cas).

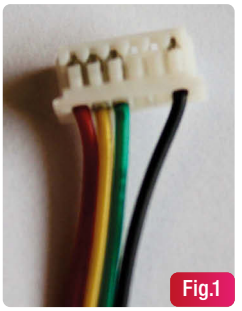


Fig.1

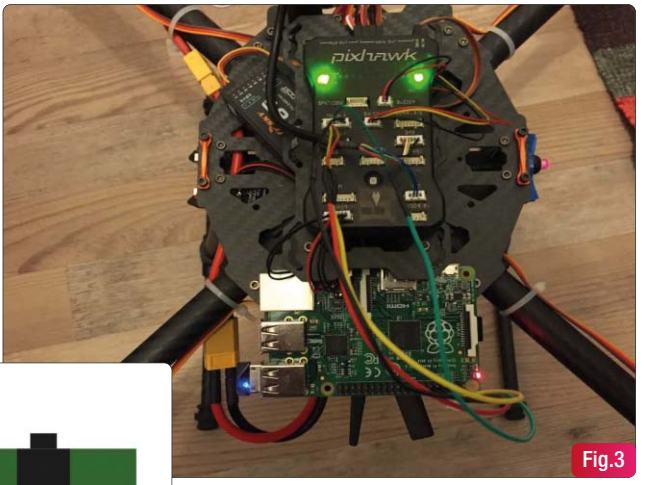


Fig.3

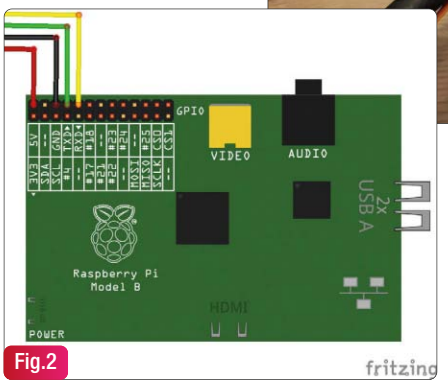


Fig.2

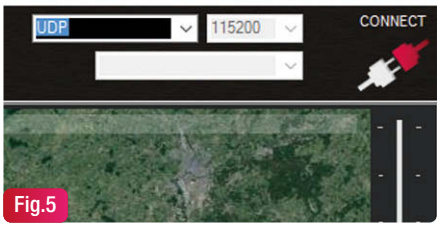


Fig.5

Raspberry Pi, Wifi, ordinateur : équipez votre station de contrôle

Pour faire communiquer notre drone avec notre ordinateur (sur lequel on aura préalablement pris le soin d'installer le logiciel "Mission Planner" offrant toutes les fonctionnalités d'une station de contrôle), 5 grandes étapes sont nécessaires.

Pré-requis :

- Un dongle USB WiFi opérationnel,
- Un Raspbian wheezy minimum d'installé,
- Une connexion Internet,
- Des câbles breadboard, Arduino, Pi,
- Un connecteur DF13 de 6 positions.

Etape 1 : la fabrication du câble de connexion

La première étape consiste à fabriquer le câble qui va relier le contrôleur de vol avec la carte Raspberry. Pour le réaliser, j'ai utilisé des câbles classiques avec connecteur femelle pour breadboard pour réaliser les branchements sur les pins du Raspberry, et de l'autre côté un câble avec un connecteur DF13 (6 positions) pour le branchement dans le connecteur du contrôleur de vol. Vous n'aurez besoin que de 4 positions sur les 6. Pour le sens de branchement, il faut respecter les branchements suivants : Fig.1.

- Pour le connecteur DF13 (6 positions) :
- GPIO Raspberry Pi : Fig.2.

Pour résumer ces branchements, le fil rouge correspond à la sortie 5V du contrôleur qui vient se brancher sur l'entrée 5V du Raspberry. Dans ce cas, c'est le contrôleur qui alimente la carte. Le fil noir correspond au ground, et les fils jaune et vert correspondent respectivement au signal de transmission et de réception. Une fois votre câble fabriqué, place désormais à l'installation des librairies qui permettront au Raspberry de communiquer avec le contrôleur de vol, mais aussi de renvoyer l'information vers l'ordinateur distant.

Etape 2 : la désactivation du contrôle du port série par l'OS

Pour réaliser la désactivation du contrôle du port série par l'OS, vous devez vous connecter en SSH à votre Raspberry, puis entrer la commande suivante :

```
sudo raspi-config
```

Lorsque la fenêtre de configuration s'affiche, sélectionnez la 8ème ligne : "Advanced Option". Dans l'écran suivant, sélectionnez également la 8ème ligne "A8 serial".

Sélectionnez ensuite "No" dans l'écran suivant. Un reboot s'impose maintenant pour prendre en compte le nouveau paramétrage.

Etape 3 : la récupération et l'installation des librairies MAVLink et MAVproxy

Cette étape nécessite en premier lieu de mettre à jour la liste des librairies :

```
sudo apt-get update
```

Puis installez les librairies et utilitaires nécessaires à l'installation de MAVLink et MAVproxy :

```
sudo apt-get install screen python-wxgtk2.8 python-matplotlib python-opencv python-pip python-numpy python-dev libxml2-dev libxslt-dev
```

Enfin, récupérez et installez les librairies MAVLink et MAVproxy :

```
sudo pip install mavproxy
```

Etape 4 : le test de connexion

Vous êtes désormais prêt à faire un test de connexion entre le Raspberry Pi et le contrôleur de vol. Pour cela, il est nécessaire de connecter la carte au contrôleur sur le port "Telem 2" grâce au câble précédemment fabriqué, puis de mettre sous tension le drone, et, par la même occasion, le contrôleur de vol et le Raspberry Fig.3. Une fois connecté en SSH au Raspberry, entrez la ligne de commande suivante :

```
sudo -s mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --aircraft monDrone
```

Si tout se passe bien, vous obtiendrez le résultat suivant : Fig.4.

Dans la commande précédente, l'argument master correspond au port utilisé pour se connecter au contrôleur de vol. Dans le cas d'un Raspberry, on utilise le port série qui est /dev/ttyAMA0. L'argument baudrate correspond à la vitesse de transmission sur le port. N'essayez pas de mettre plus 57600 lorsque vous utilisez le port série du Raspberry car cela ne fonctionne pas ! Le dernier argument correspond au nom de dossier qui sera créé pour stocker les logs et données de télémétrie.

Etape 5 : l'envoi des données vers Mission Planner

La connexion étant fonctionnelle, vous allez pouvoir faire transiter les informations du contrô-

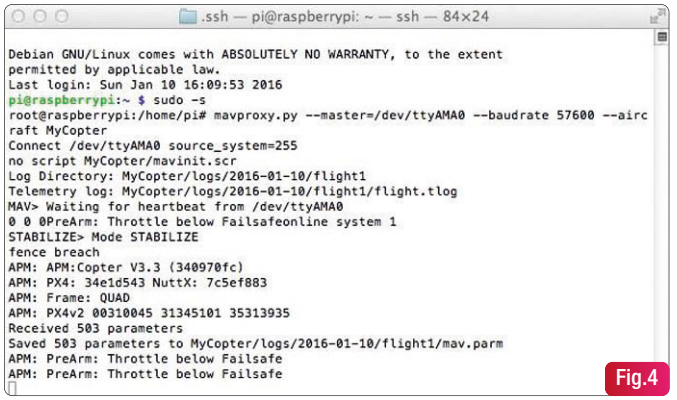


Fig.4

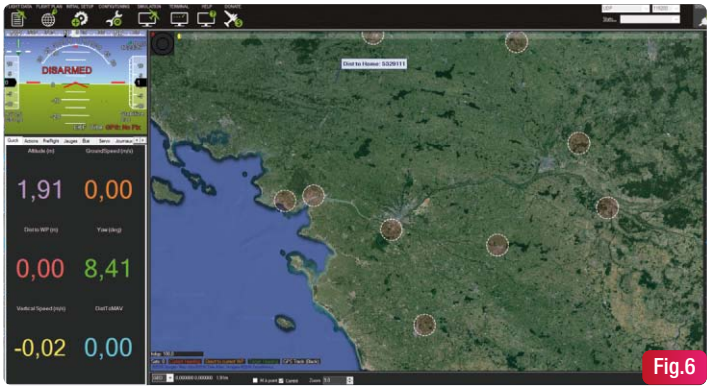


Fig.6

leur vers le logiciel Mission Planner en passant par la carte Raspberry Pi.

Quelques mots sur Mission Planner en préambule : ce logiciel, si vous avez choisi d'utiliser un contrôleur de vol de type Pixhawk ou APM, s'avère indispensable car il vous permet, entre autres choses, de mettre à jour le firmware du contrôleur, de faire les calibrages de l'accéléromètre, du compass, de la radio commande, etc. C'est une véritable « boite à outils » qui implémente aussi les fonctionnalités relatives à une station de contrôle, c'est-à-dire qu'il affiche plusieurs paramètres concernant votre drone, comme son assiette, son altitude, sa vitesse, son orientation, sa position GPS, etc. Son seul défaut est qu'il n'est disponible que sous Windows !

Voici le lien de téléchargement du logiciel, si vous voulez vous faire une idée de ses fonctionnalités : http://ardupilot.com/downloads/?did=82

Pour pouvoir rediriger les informations vers l'ordinateur distant où Mission Planner est installé, il faut récupérer son adresse IP avec une invite de commande par exemple et la commande *ipconfig*

Une fois l'adresse IP récupérée, connectez-vous en SSH sur le Raspberry connecté au contrôleur de vol, puis tapez la commande suivante :

```
mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --out 192.168.1.11:14550 --aircraft MonDrone
```

Ici l'argument "out" correspond à l'adresse IP de l'ordinateur distant sur lequel Mission Planner est installé.

Sur l'ordinateur distant, lancez Mission Planner, puis en haut à droite de la fenêtre, sélectionnez

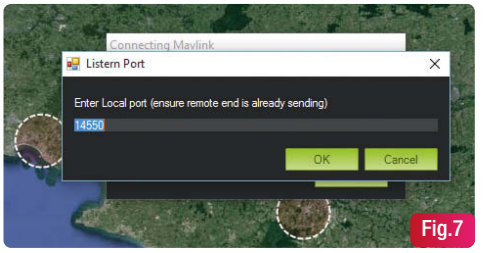


Fig.7

dans le menu déroulant "UDP", laissez la vitesse par défaut puis cliquez sur "Connect". Une fenêtre *popup* vous demandera le numéro de port UDP, il suffit alors d'entrer le port 14550 Fig.5 et 6.

Votre drone est ainsi connecté à la station de contrôle. Vous pouvez prendre votre drone dans vos mains et le bouger pour voir l'indicateur d'assiette évoluer en même temps ! Fig.7.

Pour aller plus loin…

Le fait de pouvoir communiquer avec le contrôleur de vol sans passer par la radio-commande nous donne la possibilité de faire des scripts, voire des programmes, pour diriger notre multicopter. La société qui développe le contrôleur de vol Pixhawk a décidé de mettre à disposition une API afin de multiplier les possibilités d'usage de notre drone. Je vous propose ici de voir ensemble un exemple très simple d'utilisation de cette API, avant de vous laisser la main pour des projets plus complexes.

Le principe

L'exercice vise à utiliser l'API DroneKit en Python, dont vous trouverez la documentation complète à l'adresse suivante : http://dronekit.io

Avec cette API, vous pourrez réaliser un script qui fera décoller le multicopter pour atteindre une altitude donnée, puis déplacer le drone à 2 points GPS différents, puis à le faire atterrir. Pour réaliser ce vol, il est évidemment nécessaire que votre multicopter soit équipé d'un GPS.

L'API DroneKit est utilisable pour plusieurs plateformes : Cloud, Android, et Python. Puisque le programme a pour vocation d'être exécuté sur un Raspberry, il me paraissait logique d'utiliser la librairie Python pour réaliser le script.

La récupération et l'installation de l'API DroneKit

Connectez-vous en SSH sur le Raspberry puis entrez la commande suivante :

```
sudo pip install dronekit
```

La création du script python

A l'aide de votre éditeur de texte préféré, nous allons commencer notre script par l'import des classes dont nous aurons besoin :

```
from dronekit import connect, LocationGlobalRelative
from dronekit.lib import VehicleMode
import time
```

- *connect* : classe qui permet d'initialiser la connexion avec le contrôleur de vol :
- *LocationGlobalRelative* : classe qui fait des manipulations sur les coordonnées GPS ;
- *time :* car nous allons user de timer dans ce script.

Nous allons ensuite déclarer une variable "vehicle" qui sera initialisée lors de la connexion au contrôleur de vol :

```
vehicle = None
```

Nous allons maintenant créer une méthode dans laquelle nous allons effectuer la connexion vers le contrôleur de vol, puis attendre que le processus d'initialisation aille jusqu'au bout :

```
def init_connexion():
```

```
vehicle = connect('/dev/ttyAMA0', wait_ready=True, baud=57600)

while vehicle.gps_0.fix_type < 2:
  print "Etat du gps :", vehicle.gps_0.fix_type
        time.sleep(1)

while not vehicle.is_armable:
  print "Attente de la fin du process d'initialisation"
        time.sleep(1)

vehicle.mode = VehicleMode("GUIDED")
```

Ici, plusieurs tâches sont effectuées :

- La connexion via le port série du Rapsberry vers le contrôleur ;
- La vérification que le signal GPS est suffisamment fort pour pouvoir avoir une idée précise de la position du drone ;
- La vérification que le contrôleur de vol est prêt à armer les moteurs, cela veut dire que la check-list du contrôleur est OK et que rien n'empêche le décollage du multicopter ;
- Enfin, nous forçons le mode "GUIDED" qui est le mode compatible avec le guidage du drone par la programmation.

La méthode qui suit permet de gérer la phase de décollage du drone :

```
def arm_and_takeoff(altitude):
  vehicle.armed = True

  while not vehicle.armed:
    print "Attente de l'armement des moteurs"
        time.sleep(1)

  if vehicle.mode.name == "GUIDED":
        vehicle.simple_takeoff(altitude)

        while True:
    print " Altitude: ", vehicle.location.global_relative_frame.alt

        if vehicle.location.global_relative_frame.alt >= altitude * 0.95:
      print "Altitude initiale atteinte"
        break
        time.sleep(1)
  else:
        vehicle.armed = False
```

Les étapes suivies dans cette méthode sont :

- L'armement des moteurs, puisque normalement tout est OK pour le décollage ;
- L'attente que le processus d'armement des moteurs soit réalisé ;
- La Vérification que le mode "GUIDED" est toujours le mode actif. Cela permet, en cas de problème, d'annuler le décollage en changeant le mode via votre radio commande ;
- Le Décollage avec un objectif d'altitude initial ;
- Enfin, on attend que le multicopter ait atteint son altitude initiale pour aller plus loin dans le script.

Il faut maintenant demander au multicopter de rejoindre un point GPS à une altitude que l'on aura définie, on crée donc la méthode suivante :

```
def go_to(lat, long, alt):
  if vehicle.mode.name == "GUIDED":
        tempPoint = LocationGlobalRelative(lat, long, alt)

        vehicle.simple_goto(tempPoint, groundspeed=3)
```

Ici, nous convertissons la latitude, la longitude et l'altitude pour en faire un point compréhensible par la méthode simple_goto. Puis nous donnons l'ordre de rejoindre le point avec cette même méthode en précisant une vitesse en m/s. Nous vérifions aussi ici si le mode est toujours le mode "GUIDED" pour éviter d'être court-circuité par le script si l'on décide de reprendre la main manuellement sur le drone.
Une précision sur la méthode simple_goto : cette méthode ne donne pas de feedback lorsque le point est atteint, et si l'on donne un autre ordre, alors la commande est annulée. Il faut donc mettre un timer pour laisser le temps au drone d'atteindre le point souhaité avant de donner un autre ordre.
Nous avons désormais nos méthodes, nous pouvons définir le scénario à suivre pour notre multicopter :

```
print "Initialisation de la connexion"
init_connexion()

print "Armement et Décollage avec une aaltitude initiale de 10 m"
arm_and_takeoff(10)

print "En route pour le premier point"
go_to(48.630150, -1.491171, 10)
time.sleep(10)

print "En route pour le deuxième point"
go_to(48.630007, -1.491560, 10)
time.sleep(10)

print "Retour au point de départ puis atterissage"
vehicle.mode = VehicleMode("RTL")

vehicle.close()
```

On laisse dix secondes pour atteindre le premier point, idem pour le deuxième point. Ensuite, nous activons le mode Return To Launch pour revenir au point de départ et faire atterrir le drone.
Les points GPS sont indicatifs et à changer en fonction d'où vous vous trouverez. Sachant que la vitesse définie est de 3 m/s, votre drone aura parcouru à peu près 30 mètres après 10 secondes. Ceci est à prendre en compte dans le choix du terrain pour effectuer vos tests. Il est évident qu'il faut réaliser ce genre de choses en rase campagne afin d'éviter les désagréments en cas de problème.
Le script complet donne cela : **code complet sur le site de programmez.**

Il ne vous restera plus qu'à copier le script sur le Raspberry, puis de l'exécuter avec la commande :

```
python nom_du_script.py
```

J'espère vous avoir donné envie de construire votre propre drone tout au long de ces articles. N'oubliez pas qu'il s'agit d'un hobby qui construit lentement sa bonne réputation, à pratiquer dans des conditions de sécurité maximales et dans le respect des règles de vol et de voisinage. Bon vol !

Les architectures Lambda

Les architectures Lambda dans le domaine de l'IT peuvent se définir comme étant une architecture générique et évolutive permettant le traitement en temps réel (ou pas) d'une grande quantité de données via des systèmes distribués. On retrouve ainsi la notion de Big Data qui y est fortement rattachée. En effet il s'agira par exemple d'analyse en temps réel des évènements ou logs enregistrés en temps réel sur un grand site de e-commerce avec pour objectif de prédire et/ou guider le comportement des internautes, ou potentiellement de détecter des fraudes ou anomalies dans le système. Microsoft au travers de sa plateforme Cloud Azure propose un ensemble de composants dédiés (PaaS) à ce type d'architecture, bénéficiant ainsi de toute sa puissante de calcul en provenance de son infrastructure (IaaS).

Georges DAMIEN
Consultant .NET
chez **Cellenza**
MCP, MCSD
@Georges_Damien

Michel Hubert
Directeur Technique
chez **Cellenza**
MVP Azure, Regional Director
@michelhubert

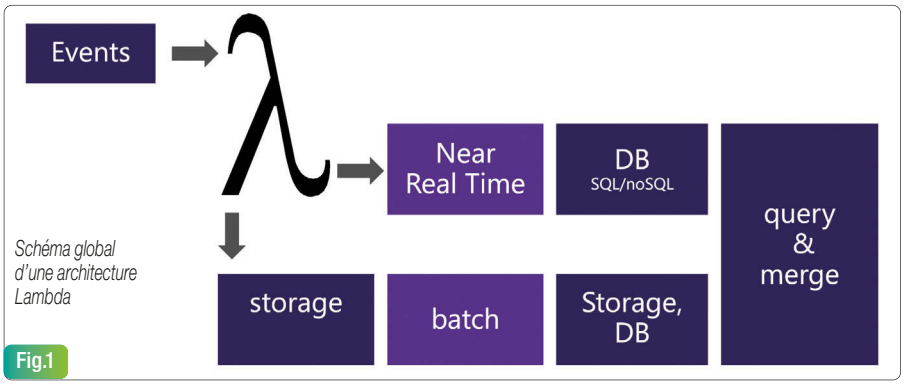


Schéma global d'une architecture Lambda

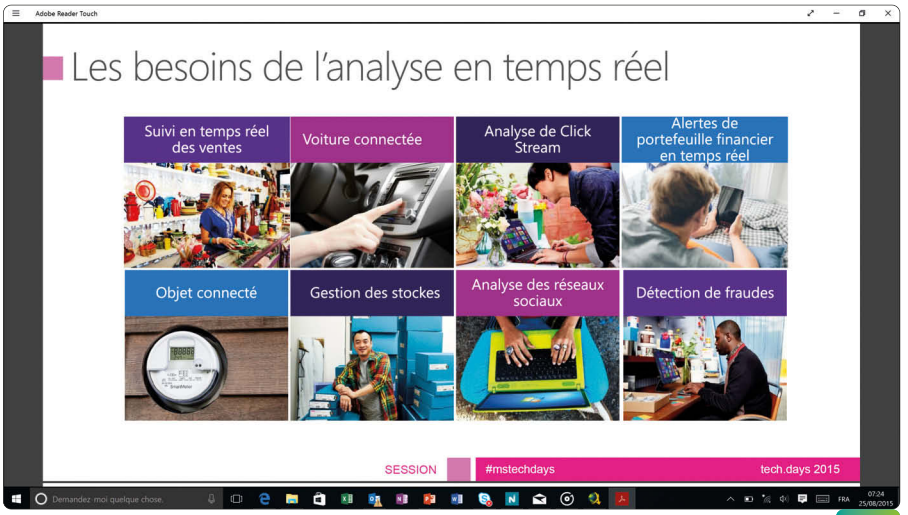
Fig.1

Comment ça fonctionne ? Fig.1

Ce type d'architecture se décompose de la façon suivante : en entrée, des événements (potentiellement de masse) sont générés et potentiellement collectés dans un concentrateur de données. Celui-ci distribue ensuite les données soit pour de l'analyse en temps réel soit pour de l'analyse statique. Au passage, la notion de stockage intervient à tous les niveaux à des dégrés différents : stockage de masse, base de données, ou simple rétention pendant un temps bien défini. En sortie, les données sont traitées (filtrées/agrégées) pour ensuite être analysées ou simplement utilisées pour des cas d'usage concret.
Voyons donc quelques cas d'usage.

Les cas d'usages

De nos jours, les flux de données transitant à travers la toile, les objets connectés, les terminaux mobiles, etc. sont importants. Ainsi, c'est à travers des millions d'informations que peuvent se noyer des informations clés. En exemple, on peut citer des cas d'usage comme da détection de fraude, la gestion de stocks en temps réel avec par exemple de la prédiction, etc. Fig.2. L'idée est donc de pouvoir, au travers d'informations **transitant via les différents canaux de communication,** effectuer des prélèvements de masse ou en temps réel afin d'y effectuer des analyses.



Exemples de cas d'usage d'architectures Lambda

Fig.2

La solution proposée par Microsoft

Microsoft, via sa plateforme de Cloud « Azure » propose une solution clé en main pour mettre en pratique les architectures Lambda, notamment au travers les briques suivantes :

- Event Hubs : concentrateur de données permettant la rétention d'informations en masse ;
- Stream Analytics : ce composant permet de l'analyse en temps réel via un système de requêtage de données avec agrégation temporelle ;
- HDInsight : composant pour la Big Data dans Azure.

D'autres briques plus ou moins annexes interviennent tout le long du processus d'utilisation des architectures Lambda comme par exemple :

- Les Blobs et Bases de données Azure : permettant du stockage de données ;
- Power BI : outil de reporting auquel on peut se « plugger » directement en sortie de certains composants Azure. Fig.3.

Quels avantages ?

Tout d'abord, l'abstraction de la partie infrastructure : en effet, vous n'avez pas à vous soucier des serveurs ou unités de calculs ; tout est géré dans le Cloud Azure. Vous pouvez intervenir pour régler la puissance de calcul souhaitée, mais ça ne se limite qu'à cela. Ensuite, les composants sont « on demand » : vous avez besoin d'effectuer un traitement sur un laps de temps ? Vous pouvez le faire et ensuite l'arrêter, le désactiver ou le supprimer. Vous avez donc une certaine maîtrise sur vos coûts. Par ailleurs, il faut se l'avouer, les composants mis à disposition sont assez simples d'utilisation et rapides à mettre en œuvre : en quelques clics, vous avez toute une plateforme d'analyse de données en temps réel ou en mode batch à disposition, et une architecture complète déployée dans le Cloud.
Voyons donc comment mettre en place ce type d'architecture en place.

Le temps réel

Afin de mettre en pratique une architecture Lambda dans Azure, nous allons dans un premier temps créer un « Event Hub » : il s'agit du concentrateur de données qui va collecter les informations en « input ». « Event Hubs » fait par défaut une rétention de 7 jours glissants. Ensuite en sortie du Hub, nous allons utiliser « Stream analytics » pour filtrer et agréger nos données en fonction de nos besoins d'étude. Puis la dernière étape consiste à extraite et afficher le résultat de nos traitements dans une base de données, ou dans une feuille Excel, ou dans un outil de reporting tel que PowerBI par exemple.

Event Hubs (Dans Azure Service Bus)

Comme son nom l'indique, Event Hubs sert de concentrateur de données : il va donc faire de la rétention de flux de données en entrée (en générale des données de masse) qui serviront ensuite à d'autres composants notamment des composants d'analyse de données (tels que Stream Analytics).
Event Hubs fonctionne par un mécanisme de « publication » d'évènements : il fait partie du module Azure « Service Bus ». Pour **publier** ou **recevoir** des données sur le hub, il faut s'y connecter via un mécanisme de « clé chiffrée » paramétrable depuis le composant Azure : vous pourrez ainsi définir un « canal » d'accès au Hub pour les entrées, sorties, ou les 2 en même temps. Fig.4.
Une fois l'Event Hubs créé et paramétré, vous pourrez envoyer ou recevoir des données depuis ou vers vos applications au travers de 2 protocoles : soit par AMQP, soit par HTTP.

Publication de messages

Exemple .net via AMQP

Pour vos applications .NET, des packages Nugget associés à Event Hubs sont disponibles et permettent une implémentation aisée d'envoi de données via le protocole AMQP. Fig.5.

Via HTTP

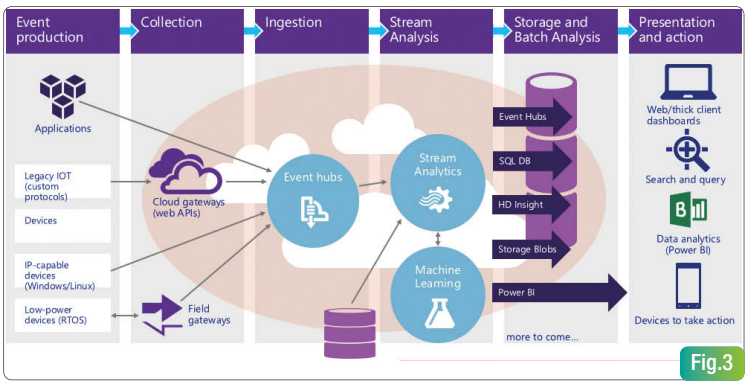
Vous pouvez également requêter l'Event Hubs via HTTP notamment depuis des clients Web. Fig.6.

Stream Analytics

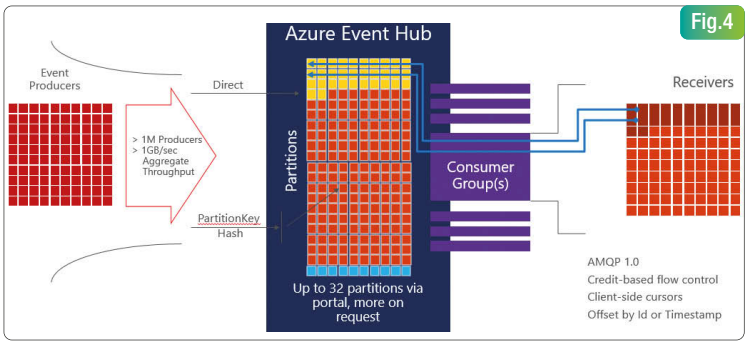
Le composant Azure Stream Analytics effectue des traitements de flux de données en temps réel. Il va donc servir de filtre et d'agrégateur d'informations. En effet, le traitement de données de Stream Analytics fonctionne via 3 grandes étapes :

- Le(s) entrée(s)
- Le(s) requête(s)
- Les sortie(s)

Les entrées :
Il s'agit du point d'entrée des données à traiter. Stream Analytics peut consommer en entrée un flux de données en provenance du concentrateur d'évènements d'Azure (Event Hubs), ou d'un Blob Azure.



Architecture Lambda By Microsoft



Event Hubs en action

```
public static void SendEvent(Tweetinvi.Core.Interfaces.ITweet tweet)
{
    // Create EventHubClient
    EventHubClient client = EventHubClient.Create("thatconference");

    // create event object to be sent
    TweetEvent myEvent = new TweetEvent() {
        Author = tweet.Creator.ScreenName,
        Text = tweet.Text,
        CreatedAt = tweet.CreatedAt
     };
    var serializedString = JsonConvert.SerializeObject(myEvent);
    EventData data = new EventData(Encoding.Unicode.GetBytes(serializedString))
    {
        PartitionKey = "WebSource"
    };

    // Set user properties if needed
    data.Properties.Add("Type", "TweetEvent");

    // Send the metric to Event Hub
    client.SendAsync(data);
}
```

Fig.5

Exemple de méthode c# d'envoi d'évènements via Amqp d'Event Hubs

```
<script type="text/javascript">
    function doCheckin() {
        // get selected Attendee
        var e = document.getElementById("ddlAttendee");
        var strAttendee = e.options[e.selectedIndex].value;

        // get selected Vendor
        var e = document.getElementById("ddlVendor");
        var strVendor = e.options[e.selectedIndex].value;

        // set up  parameters for EventHub
        var sas = "SharedAccessSignature sr=https%3a%2f%2fthatconference-ns.servicebus.windows.net%2fthatconference%2fpublishers%2fbstineman

        // set up request
        var xmlHttpRequest = new XMLHttpRequest();
        xmlHttpRequest.open('POST', 'https://thatconference-ns.servicebus.windows.net/thatconference/publishers/bstineman/messages', true);
        xmlHttpRequest.setRequestHeader('Content-Type', 'application/atom+xml;type=entry;charset=utf-8');
        xmlHttpRequest.setRequestHeader('Authorization', sas);
        // optional: set property on the message.
        xmlHttpRequest.setRequestHeader('Type', 'VendorCheckin');

        eventMsg = "{ VendorID: " + strVendor + ", BadgeID: " + strAttendee + " }";
        xmlHttpRequest.send(eventMsg);

        window.alert("Sent event: " + eventMsg);
    }
</script>
```

Fig.6

Exemple de requête HTTP d'envoi d'évènements Event Hubs

Les requêtes :
C'est la pierre angulaire de Stream Analytics. C'est là que les données seront agrégées/filtrées notamment via un langage de requêtes très proche du SQL : **le DML (Data Manipulation Language)**.
Lien en français de la documentation du DML : https://msdn.microsoft.com/fr-fr/library/azure/dn834998.aspx
Voici un exemple de requête en DML : Fig.7.
La requête effectue ici une agrégation permettant la remontée du nombre de hits sur un site Internet sous IIS (analyse de logs ETW) avec un fenêtrage de 5 minutes (toutes les 5 minutes si au moins un évènement intervient dans cette tranche de temps).
Ci-dessous les différents modes de fenêtrage de requête : Fig.8.

Les sorties :
Une fois le(s) requête(s) paramétrées, testées et exécutées, les données en sortie sont à définir. On pourra ainsi les stocker dans une base de données, des fichiers plats, les récupérer dans Power BI ou Excel, ou même les rediriger vers un « Event Hubs » par exemple.
Lien Microsoft de la documentation en français de Stream Analytics :
https://azure.microsoft.com/fr-fr/services/stream-analytics/

Le mode Batch

HDInsight

HDInsight est une distribution Apache Hadoop.
Elle permet de gérer dans le Cloud une quantité élevée de données et propose la création de clusters Hadoop, Spark, HBase et Storm. Fig.9.
Dans le cadre de notre architecture Lambda, et tout particulièrement sur le traitement « batch », nous pouvons créer un cluster HDInsight (Hadoop), spécifier la taille de cluster (8 nœuds par exemple) en fonction de la puissance de calcul souhaité (on peut à tout moment augmenter ou diminuer cette taille).
La création du cluster prend quelques minutes (5 à 15minutes en moyenne selon la taille des nœuds choisie). Une fois ce cluster créé, depuis le portail Azure, dans la section « HDInsight », il nous est donné la possibilité de se connecter au 1er nœud (entête). A partir d'un accès à distance, nous pouvons nous y connecter et lancer la console « Hadoop Command Line ».
Depuis cette console, nous pourrons exécuter nos traitements de calcul via des commandes « **PIG** » le langage procédurale **Pig Latin**.
Une fois les traitements effectués, les données en sortie peuvent être déposées dans un conteneur Azure afin d'y être exploitées.

On peut imaginer une utilisation simple du résultat en sortie depuis une feuille Excel par exemple (nous pouvons de surcroît effectuer une connexion directe depuis Excel à HDInsight).

Conclusion

De nos jours, les tendances technologiques sont en faveur des architectures Lambda : en effet, au travers de l'utilisation des applications riches sur les terminaux mobiles, la vulgarisation des objets connectés, la forte utilisation des réseaux sociaux et d'Internet en général, de grandes quantités de données sont échangées et traitées.
Ainsi, la mise en place d'architectures Lambda se justifie dans énormément de cas et dans une multitude de fonctions métiers.
Que ce soit pour de la détection de fraude, de la gestion de stock en temps réel, de l'analyse prédictive de comportements humains sur les sites Internet de vente en ligne, ou dans les systèmes embarqués des voitures récentes, ce type d'architecture devient omniprésent.
Microsoft, via Azure, propose un ensemble de composants répondant parfaitement aux problématiques de gestion et d'analyses de données en masse en temps réel ou en mode batch.
Bénéficiant donc d'une infrastructure dans le Cloud dont on peut faire abstraction à l'usage, l'utilisateur peut mettre en place une architecture Lambda au travers de quelques clics, et se focaliser uniquement sur l'analyse ou le traitement de ses données en sortie, donc sur son métier.

Fig.7

```
SELECT DateAdd(minute,-5,System.TimeStamp) AS WinStartTime,
       System.TimeStamp AS WinEndTime,
       [event_s_sitename] as Site,
       [event_cs_uri_stem] as URL,
       COUNT(*) as Count
FROM [iis-etw-hub]
WHERE [event_sc_status] < 400
GROUP BY [event_s_sitename], [event_cs_uri_stem], SlidingWindow(minute, 5)
```

Exemple de requête DML

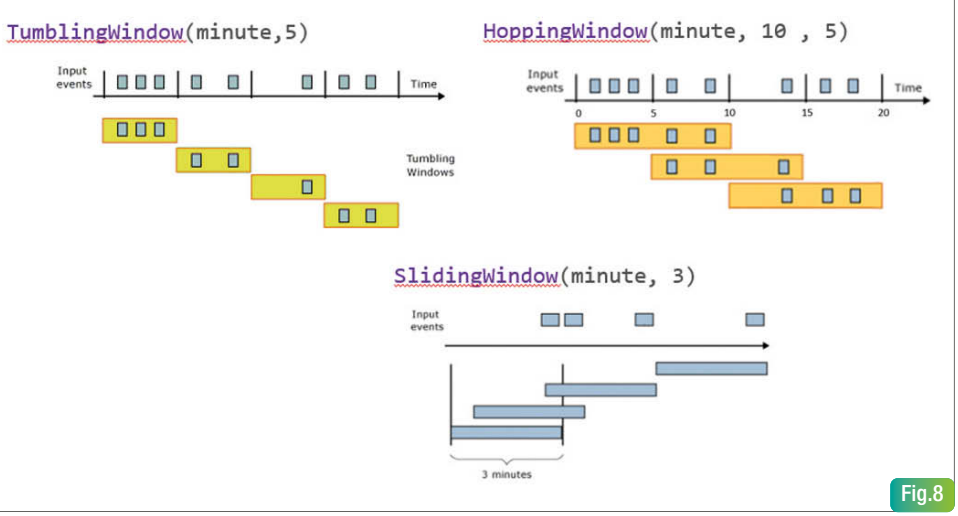


Fig.8

Les différents types de fenêtrages dans Stream Analytics

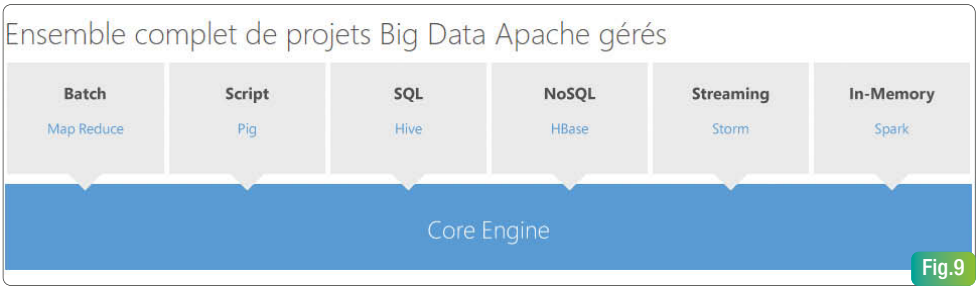


Fig.9

Introduction à Gearman pour le multitâche en PHP

2e partie

Gearman (anagramme de « Manager ») est un framework applicatif open source conçu pour distribuer des tâches vers plusieurs machines ou processus, en supportant le changement d'échelle. Il permet d'effectuer des traitements en parallèle, de faire de la répartition de charge, et d'appeler des fonctions entre différents langages. En utilisant la redondance et la persistance, il peut être utilisé sans présenter de point individuel de défaillance et en assurant la reprise après erreur.

Guillaume Burlot,
ingénieur développement **Osaxis**

Problème

Afin d'éviter ce long blocage de 15 secondes durant lesquelles l'utilisateur ne peut rien faire (à part stopper la requête ou quitter la page), nous souhaiterions lancer le traitement en arrière-plan, rendre la main à l'utilisateur (au navigateur) et laisser les calculs se terminer de façon asynchrone. De plus, comme le traitement d'une phrase est indépendant des autres, nous aimerions que plusieurs calculs puissent s'effectuer en même temps en parallèle, éventuellement sur des machines distinctes.

Mise en application : ajout de Gearman

Nous utilisons des machines supplémentaires à installer et configurer :

- « gearman1 » et « gearman2 » (192.168.0.201 et 202) : serveurs de jobs Gearman (il en faut au moins un, le deuxième prend le relai si le premier ne répond plus).
- « workers1 », « workers2 » et « workers3 » (192.168.0.211, 212 et 213) : hôtes des workers Gearman.

Installation

Sur toutes les machines, le système d'exploitation est Ubuntu Server 14.04 LTS (« trusty ») 64 bits.

- Sur chacun des serveurs gearman1 et 2, on installe le paquet gearman, qui installe en fait les paquets gearman-job-server (qui fournit le serveur gearmand) et gearman-tools (qui fournit l'outil d'administration gearadmin et une interface en ligne de commande pour le client gearman) :

```
sudo apt-get install gearman
```

- Sur chacun des serveurs workers1, 2 et 3, on installe PHP (seule la version ligne de commande est nécessaire) et l'extension Gearman pour PHP (pour l'interface worker/serveur) :

```
sudo apt-get install php5-cli php5-gearman
```

- Et sur www, on installe aussi l'extension php5-gearman (pour l'interface client/serveur) :

```
sudo apt-get install php5-gearman
```

Fonctionnement choisi

Schématisons comment fonctionne notre nouvelle architecture (Fig.6).
Un worker, répliqué sur chacun des hôtes workers1 à 3, déclare une fonction sous le nom « uppercase » auprès des serveurs gearman1 à 2, et définit le traitement associé (correspondant à doUppercase).
L'application Web, lors de la soumission de phrases, envoie pour chaque phrase une demande d'exécution asynchrone de la fonction « uppercase » à un des serveurs gearman1 à 2. Chacun des serveurs gearman1 à 2 gère une file de tâches, et pour chaque tâche transmet un job à un des workers connus disponibles pouvant effectuer la fonction demandée (s'ils sont tous occupés, il attend qu'un se libère). Tout cela est fait de manière automatique par Gearman. Chaque instance de worker effectue les jobs reçus des serveurs, cela étant fait de manière automatique par Gearman. Pour la fonction « uppercase », chaque exécution crée un fichier sur le serveur files. L'application Web, sur la page liste, continue de lister les fichiers créés sur le serveur files.

Le code

Nous écrivons le worker pour la fonction « uppercase » sous la forme d'un script PHP. Nous aurions pu choisir un autre langage supporté et installer les paquets correspondants, mais ici nous allons réutiliser notre fonction PHP doUppercase :

uppercase_worker.php :

```
#!/usr/bin/php
<?php

function doUppercase($input, $resultPath)
{
  /* … fonction copiée depuis form.php … */
}

$worker = new GearmanWorker();

$worker->addServer('gearman1');
$worker->addServer('gearman2');

$worker->addFunction('uppercase', function (GearmanJob $job) {
  $workload = json_decode($job->workload(), true);
  doUppercase($workload['input'], $workload['result_path']);
});

while ($worker->work()) {
  $returnCode = $worker->returnCode();
  if ($returnCode !== GEARMAN_SUCCESS) {
    echo 'Bad return code: ' . $returnCode . "\n";
    break;
  }
}
```

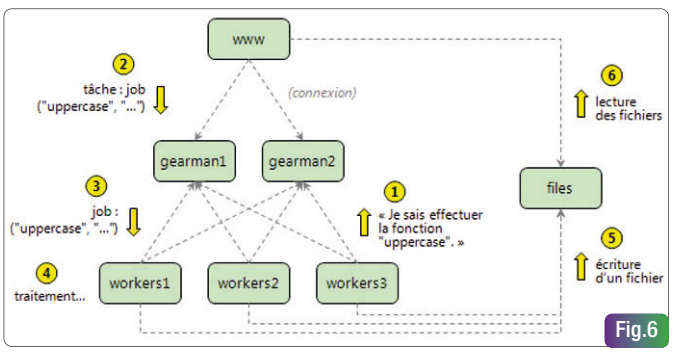


Schéma de fonctionnement

Le fichier reprend d'abord la définition de la fonction doUppercase de form.php. En dessous, le code crée un objet worker Gearman qui peut se connecter aux serveurs gearman1 et gearman2, et enregistre auprès de ces serveurs une fonction « uppercase », en y associant un code de traitement, ici une fonction anonyme, qui reçoit un objet job Gearman, décode le workload (argument de la fonction « uppercase », ici un tableau associatif encodé en JSON) et exécute la fonction doUppercase avec les paramètres « input » et « result_path » du workload. Puis le code appelle la méthode « work » de l'objet, qui attend la réception d'un job d'un serveur, appelle alors le code de traitement associé à la fonction demandée, puis une fois le traitement terminé (ou en cas d'échec), retourne un booléen de succès. Alors, si le booléen indique un échec ou si le code de retour Gearman vaut autre chose que la valeur de succès, le script se termine (sortie de la boucle while), sinon le code rappelle la méthode work (boucle while), qui attend le prochain job, etc.

Nous copions ce script sur chacun des hôtes de workers, par exemple dans un sous-dossier « workers » de notre dossier home. Puis, sur chaque hôte, nous lançons un processus de worker (en arrière-plan avec redirection des sorties dans un fichier de log), avec la commande :

```
php ~/workers/uppercase_worker.php > ~/logs/uppercase-$BASHPID.log 2>&1 &
```

Puis sur chacun des serveurs Gearman, on vérifie qu'il y a bien eu enregistrement d'une fonction « uppercase » auprès du serveur par trois instances de worker, avec la commande :

```
gearadmin --status
```

qui affiche :

```
uppercase     0     0     3
.
```

et on peut voir le détail des workers avec la commande :

```
gearadmin --workers
```

qui affiche quelque chose comme :

```
36 : : 3530 : 3738 : 3400 : 0 % 1470921328 - :
33 192.168.0.211 - : uppercase
34 192.168.0.212 - : uppercase
35 192.168.0.213 - : uppercase
.
```

Nous modifions ensuite l'application Web sur www. Dans form.php, nous supprimons la définition de la fonction doUppercase, qui est maintenant dans le code du worker, et modifions le code de gestion de la soumission du formulaire :

form.php :

```
<?php

const FOO_INPUTS_COUNT = 5;

if ($_SERVER['REQUEST_METHOD'] === 'POST') {

  $client = new GearmanClient();

  $client->addServer('gearman1');
  $client->addServer('gearman2');

  require_once __DIR__ . '/inc/config.php'; // for FOO_FILES_DIR
  $outDir = FOO_FILES_DIR;
  $uniq = uniqid('', true);
  for ($i = 0; $i < FOO_INPUTS_COUNT; $i++) {
    if (isset($_POST['inputs'][$i]) && $_POST['inputs'][$i] !== '') {

      $client->doBackground('uppercase', json_encode(array(
        'input' => $_POST['inputs'][$i],
        'result_path' => "$outDir/$uniq-$i.json",
      )));
    }
  }

  // Redirect to the list page
  header('Location: list.php');
  exit;
}

/* … la suite inchangée par rapport à la version initiale … */
```

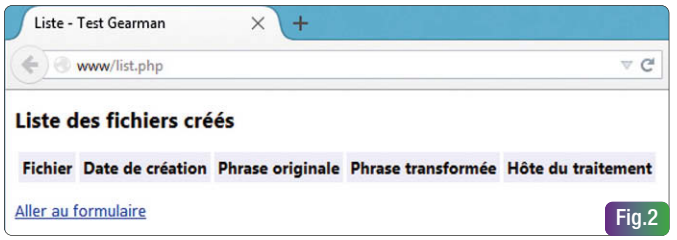


Fig.2

Page liste de notre application web

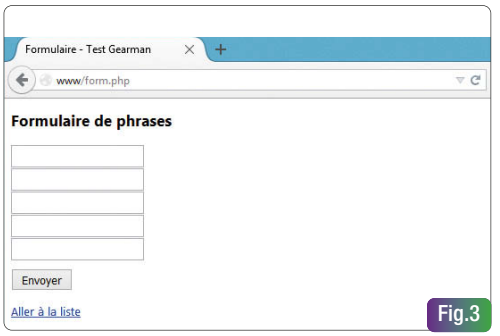


Fig.3

Page formulaire de notre application web

Fig.4

Soumission de 5 phrases à l'instant T

Fichier	Date de création	Phrase originale	Phrase transformée	Hôte du traitement
55d1d60c7560f5.85162702-0.json	17/08/2015 16:39:43	bonjour	BONJOUR	www
55d1d60c7560f5.85162702-1.json	17/08/2015 16:39:46	enchanté	ENCHANTÉ	www
55d1d60c7560f5.85162702-2.json	17/08/2015 16:39:49	comment allez-vous ?	COMMENT ALLEZ-VOUS ?	www
55d1d60c7560f5.85162702-3.json	17/08/2015 16:39:52	ça va, merci !	ÇA VA, MERCI !	www
55d1d60c7560f5.85162702-4.json	17/08/2015 16:39:55	à bientôt...	À BIENTÔT...	www

Fig.5

Liste affichée après redirection à T+15s

Fichier	Date de création	Phrase originale	Phrase transformée	Hôte du traitement

Fig.7

Liste rechargée entre T+0s et T+3s

Fichier	Date de création	Phrase originale	Phrase transformée	Hôte du traitement
55d1d76b34f513.17098923-0.json	17/08/2015 16:45:34	bonjour	BONJOUR	workers2
55d1d76b34f513.17098923-1.json	17/08/2015 16:45:34	enchanté	ENCHANTÉ	workers3
55d1d76b34f513.17098923-2.json	17/08/2015 16:45:34	comment allez-vous ?	COMMENT ALLEZ-VOUS ?	workers1

Fig.8

Liste rechargée entre T+3s et T+6s

Fichier	Date de création	Phrase originale	Phrase transformée	Hôte du traitement
55d1d76b34f513.17098923-0.json	17/08/2015 16:45:34	bonjour	BONJOUR	workers2
55d1d76b34f513.17098923-1.json	17/08/2015 16:45:34	enchanté	ENCHANTÉ	workers3
55d1d76b34f513.17098923-2.json	17/08/2015 16:45:34	comment allez-vous ?	COMMENT ALLEZ-VOUS ?	workers1
55d1d76b34f513.17098923-3.json	17/08/2015 16:45:37	ça va, merci !	ÇA VA, MERCI !	workers1
55d1d76b34f513.17098923-4.json	17/08/2015 16:45:37	à bientôt...	À BIENTÔT...	workers3

Fig.9

Liste rechargée après T+6s

Le nouveau code crée un objet client Gearman qui peut se connecter aux serveurs gearman1 et gearman2.
Puis, pour chaque phrase soumise, l'application envoie automatiquement au premier serveur disponible une tâche en arrière-plan, sans attendre le résultat, demandant d'exécuter la fonction « uppercase » (enregistrée sous ce nom auprès du serveur) avec les arguments « input » et « result_path » (sous la forme d'un tableau associatif encodé en JSON car le workload doit être une string). Puis le code redirige vers la page liste. Les autres fichiers restent inchangés.

Test de l'application

Avant tout, nous supprimons sur le serveur files les fichiers créés par la version initiale. On recharge la page liste dans le navigateur (on retrouve la liste vide comme sur la Fig.2).
Puis à nouveau, on accède au formulaire, et on soumet les mêmes phrases (comme sur la Fig.4).
On constate que la requête reçoit maintenant une réponse quasi immédiate, qui redirige vers la page liste, sur laquelle la liste est pour l'instant toujours vide (Fig.7).
On recharge la page dans le navigateur toutes les secondes, et au bout de 3 secondes on voit apparaître les trois premiers fichiers (Fig.8), puis 3 secondes plus tard les deux autres (Fig.9).
On peut remarquer au passage que le serveur Gearman n'a pas suivi d'ordre particulier pour les workers : bien qu'il traite les tâches dans l'ordre de leur réception, chaque job est soumis à un worker disponible quelconque.

Analyse des résultats

Nous avons bien obtenu le comportement souhaité : toutes les tâches sont lancées en arrière-plan, l'utilisateur est libéré presque immédiatement, et les N workers actifs permettent de générer N fichiers en même temps en parallèle. Cet exemple était volontairement très simple.
Pour une utilisation réelle on lancerait plusieurs instances de worker sur chaque hôte (sous forme de processus indépendants ou de threads par exemple).
On aurait aussi probablement plusieurs workers (par exemple un autre script lowercase_worker.php pour une fonction « lowercase »).
En outre, un même worker peut déclarer plusieurs fonctions (mais bien sûr chaque instance ne pourra en traiter qu'une à la fois).

Conclusion

Gearman nous a permis d'exploiter de manière automatique nos ressources de calcul en parallèle, via une API très simple à utiliser.
L'architecture offerte est rapide à mettre en place et modulable : elle permet de facilement ajouter ou supprimer des machines, des workers et des fonctions, en laissant le serveur de jobs s'adapter à l'évolution.

Pour plus d'informations sur Gearman en général et sur les points non abordés, par exemple la communication entre des langages différents ou les files persistantes, on peut consulter :
Le site officiel (avec documentation) : http://gearman.org (en anglais)
La documentation de l'extension PHP : http://php.net/gearman

Attention : certaines images sont parues dans la 1ère partie.

10 façons de renforcer la sécurité dans Drupal 8

Implémenter les règles de sécurité sur un logiciel ou un produit après sa conception s'avère très compliqué. C'est pourquoi il est crucial d'intégrer les questions de sécurité au cœur de la conception technique afin d'éviter toute erreur irrémédiable. C'est le cas pour Drupal 8, où une sécurité par défaut a été prévue, tout en restant ouverte pour les développeurs et gestionnaires de sites.

Peter Wolanin,
Ingénieur principal Cloud Security chez **Acquia**

Dans cette liste non exhaustive d'actions pour renforcer la sécurité, certaines se résument à une ou deux lignes de code visant à optimiser certaines configurations – et il y a bien sûr d'autres enjeux qui nous semblent moins essentiels que nous n'avons pas mentionnés dans cet article. Néanmoins, ces points d'amélioration font l'objet d'un consensus au sein de la communauté et répondent à des problèmes de sécurité clairement identifiés qui nécessitaient des correctifs pour la version *core* de Drupal ou certains modules additionnels des versions antérieures. Ces 10 améliorations sont importantes, elles résultent d'une analyse des « security advisories » (SA) identifiés par le passé ainsi que des questions soulevées auprès d'Acquia par les entreprises qui envisagent d'adopter Drupal.

1. Les modèles Twig utilisés pour la production HTML

Ce point est sans doute celui qui vient en premier quand on évoque la sécurité dans Drupal 8 ; il correspond aussi à l'une des fonctionnalités les plus utilisées chez les développeurs de thèmes (ou « themeurs »).

Un gain de sécurité provient de la séparation stricte qui a été mise en place entre la logique métier et la couche de présentation – ce qui rend plus facile la validation des thèmes tiers et permet de déléguer le travail de présentation pure. Vous ne pouvez pas exécuter de requêtes SQL ou accéder à l'API Drupal à partir de Twig. En outre, Drupal 8 permet à Twig un *auto-escaping*, ce qui signifie que toute donnée qui n'est pas spécifiquement identifiée comme sûre sera sécurisée en utilisant les fonctions PHP *htmlspecialchars* (par exemple c'est la même chose avec Drupal 7 check_plain []). L'*Auto-escaping* de variables permettra d'éviter de nombreuses vulnérabilités XSS qui se sont accidentellement introduites dans les thèmes de site personnalisés. C'est la raison pour laquelle on peut classer ce point au rang de N° 1 des enjeux de sécurité, XSS étant la vulnérabilité la plus fréquente dans le code Drupal. Nous ne disposons pas de beaucoup de données mais au vu des audits de sites menés dans le passé, nous pouvons dire que près de 90% des vulnérabilités sont issues de thèmes personnalisés. Pour comprendre pourquoi les « *themeurs* » aiment Twig, comparer le code block.tpl.php de Drupal 7 à celui de Drupal 8 Twig.

Drupal 7 block.tpl.php :

```
<div id="<?php print $block_html_id; ?>" class="<?php print $classes; ?>"<?php print $attributes; ?>>

  <?php print render($title_prefix); ?>
<?php if ($block->subject): ?>
  <h2<?php print $title_attributes; ?>><?php print $block->subject ?></h2>
<?php endif;?>
  <?php print render($title_suffix); ?>

  <div class="content"<?php print $content_attributes; ?>>
    <?php print $content ?>
  </div>
</div>
```

Drupal 8 block.html.twig :

```
<div{{ attributes }}>
  {{ title_prefix }}
  {% if label %}
    <h2{{ title_attributes }}>{{ label }}</h2>
  {% endif %}
  {{ title_suffix }}
  {% block content %}
    {{ content }}
  {% endblock %}
</div>
```

- Le manuel : https://www.drupal.org/theme-guide/8/twig
- Les changements d'entrées : https://www.drupal.org/node/2296163
- La présentation de l'audit de site : http://szeged2008.drupalcon.org/files/Hack-proof%20Your%20Drupal%20App.pdf
- Davantage d'info à lire sur les XSS : https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

2. Supprimer le filtre d'entrée PHP et interdire l'import de configuration via PHP

Effectivement, ce point aurait pu constituer le point N° 1. Drupal 8 n'intègre pas le format d'entrée PHP dans le cœur. En plus de favoriser les bonnes pratiques (gestion du code dans un système de contrôle de révision comme Git), cela signifie que Drupal ne permet plus de remonter au login administrateur pour exécuter arbitrairement du code PHP ou une commande *shell* sur le serveur. Concernant Drupal 7, l'import d'une vue via le module Views est fait via code du PHP exécutable, pareil pour certains réglages types visibilité des blocks personnalisés, peuvent nécessiter du code. Ces usages de PHP évalués (qui exposent à des vulnérabilités lors de l'exécution du code) n'existent plus – voir le point ci-dessous sur la gestion de la configuration.

Voir plus : https://dev.acquia.com/blog/drupal-8/10-ways-drupal-8-will-be-more-secure/2015/08/27/6621#sthash.IQEZSqUW.

Ces deux points me semblent essentiels, l'ordre d'importance des points suivants est plus arbitraire.

3. Exporter vos configurations de site et les versionner comme du code

L'Initiative de Gestion de Configuration (Configuration Management Initiative – CMI) a transformé la façon dont Drupal 8 gère les configurations par rapport à Drupal 7 (ex : code PHP, les variables Drupal ou ctools exportables). La CMI utilise YAML comme le format d'export et d'import, ces fichiers YAML peuvent être versionnés et vérifiés avec votre code et votre outil de versionning (Git).

Pourquoi est-ce une amélioration de la sécurité ? Eh bien, en plus d'éliminer l'utilisation de code PHP comme un format d'import, la traçabilité des configurations est plus simple grâce à un historique des changements. Drupal devient plus intéressant et adapté aux entreprises qui ont besoin de contrôles drastiques des changements de configuration mis en place. En outre, les fichiers de configuration pourront être testés puis déployés en même temps que les modifications du code correspondantes (afin d'éviter des erreurs manuelles de configuration). Il est en outre possible d'interdire les modifications via l'interface utilisateur et d'imposer une mise à jour des configurations par déploiement de code.

- Le manuel sur le CMI : https://www.drupal.org/documentation/administer/config

4. Contenu utilisateur et filtrage amélioré

Alors que l'intégration de système d'édition WYSIWYG (What You See Is What You Get) dans la version cœur de Drupal est une grande avancée en termes d'ergonomie, des mesures supplémentaires ont été prises pour limiter les mauvaises pratiques issues de l'éditeur WYSIWYG ayant été constatées dans les versions antérieures de Drupal. En particulier, les

utilisateurs d'éditeur WYSIWYG ont souvent été autorisés à accéder au format de texte HTML complet, ce qui a ouvert la porte à des attaques XSS sur d'autres sites utilisateur. Afin de favoriser l'utilisation de format HTML filtré en ligne avec les bonnes pratiques actuelles, la configuration de l'éditeur WYSIWYG Drupal 8 est intégrée avec le filtre de texte correspondant. Quand un bouton est ajouté à la configuration active, la balise HTML correspondante est ajoutée à la liste autorisée pour le filtre de texte.

Faites glisser un nouveau bouton de disponible à la barre active dans la configuration de l'éditeur :

La balise HTML correspondante (la balise U) est ajoutée à la liste autorisée :

Une amélioration supplémentaire en termes de sécurité est que le filtrage de texte de base contraint les utilisateurs à utiliser seulement des images locales sur le site ce qui aide à prévenir le *cross-site request forgery* (CSRF) et d'autres attaques ou l'abus d'utilisation d'image.

5. La session utilisateur et la gestion des sessions plus sécurisées

Il y a trois améliorations distinctes dans la gestion des sessions et des *cookies*. Tout d'abord, la sécurité des identifiants de session a été grandement améliorée en termes d'exposition à des sauvegardes de bases de données ou d'injection SQL (7.x arrière-port). Auparavant, dans Drupal, les identifiants de session étaient stockés et vérifiés directement avec le *cookie* de session à partir du navigateur. Le risque est que la valeur de la base de données soit utilisée pour remplir les *cookies* du navigateur et ainsi de supposer que la session et l'identité de tout utilisateur disposant d'une session valide dans la base de données, soient les mêmes. Dans Drupal 8, les identifiants sont « hash-és » (chiffrés) avant d'être stockés, empêchant ainsi la valeur d'être utilisée pour reconnaître une session utilisateur en base de données. La valeur du cookie entrante est donc « hash-ée » avant d'être recherchée en base de données.
Un *mixed-mode SSL session* a été ajouté au cœur pour supporter les sites qui utiliseraient des modules additionnels d'authentification sur protocol SSL alors que d'autrespages ne sont pas chiffrées. Il faudra remplacer le service de gestion de session si vous en avez vraiment besoin. Ce choix par défaut encourage l'utilisation du SSL sur l'intégralité du site (améliorant aussi le classement dans les moteurs de recherche).
Le dernier changement est que le « www. » n'est plus supprimé du cookie de session relatif au domaine afin de ne plus envoyer le cookie de session à tous les sous domaines. (7.x arrière-Port)

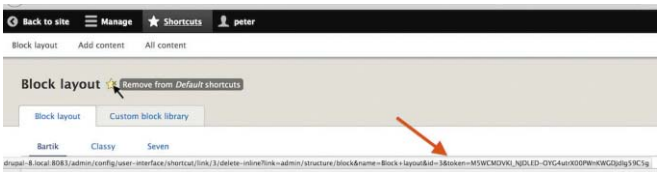
- Les changements d'entrées sur les principaux « www » : https://www.drupal.org/node/2523826
- Les changements d'entrées en mode mixte : https://www.drupal.org/node/2384903
- Davantage d'info à lire sur la manipulation de session : https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

6. Protection automatique via un jeton CSRF (Cross-Site Request Forgery) dans la définition des routes

Les liens (requêtes GET) qui provoquent des actions destructives ou des changements de configuration doivent être protégés contre les attaques CSRF, généralement grâce à un jeton spécifique à utilisateur prévu dans l'URI. Ce jeton étant vérifié avant d'effectuer l'action. Ce changement améliore l'expérience de développeur et la sécurité en automatisant un processus souvent oublié dans les modules communautaires. Par ailleurs, la centralisation du code facilite la vérification et offre une garantie de test.
Drupal 8 rend cela beaucoup plus simple. Le développeur doit simplement indiquer que la route (un chemin d'accès système [hook_menu*] en termes Drupal 7) nécessite un jeton CSRF. Ci-dessous un exemple de définition YAML d'une route protégé dans Drupal 8 :

```
entity.shortcut.link_delete_inline:
  path: '/admin/config/user-interface/shortcut/link/{shortcut}/delete-inline'
  defaults:
    _controller: 'Drupal\shortcut\Controller\ShortcutController::deleteShortcutLinkInline'
  requirements:
    _entity_access: « shortcut.delete »
    _csrf_token: 'TRUE'
```

7. Une seule ligne dans « **requirements**: » suffit à protéger le lien via un jeton CSRF. Le lien pour supprimer le raccourci et le jeton correspondant dans la requête :



8. Vérification des URLs via des patterns

De nombreux sites Drupal répondront à une requête de page grâce à une entête de page arbitraire envoyée à l'adresse IP correcte. Cela peut conduire à la saturation du cache (cache poisonning), des emails malveillants, de fausses demandes de récupération de mot de passe, et à autres failles de sécurité.
Pour les versions antérieures, la configuration du serveur Web peut s'avérer un casse-tête pour prévenir les attaques par type « host header spoofing » pour un site unique utilisant sites/default comme répertoire par défaut. Drupal 8 propose un outil simple pour configurer les patterns de domaine attendus dans settings.php et avertira dans les rapports du site les patterns non autorisés.

- Le manuel sur les problèmes liés à l'usurpation d'identités : https://www.drupal.org/node/1992030
- Les changements d'entrées : https://www.drupal.org/node/2221699

9. L'extension MySQL PDO (PHP Data Objects) limitée à l'exécution des instructions simples

Drupal 8 limite PHP afin d'envoyer une seule et unique instruction SQL à la fois. Ce changement réduit l'impact de SA-CORE-2014-005 (une vulné-

rabilité d'injection SQL qui a été facilement exploitée par des utilisateurs anonymes) (7.x arrière-port). Pour réaliser ce changement dans Drupal 8 il a fallu d'abord que j'opère en amont un petit changement dans le langage PHP en lui-même, ainsi que dans la bibliothèque PDO MySQL disponible dans les versions PHP 5.5.21 ou 5.6.5, et versions suivantes.
Il y a aussi un patch actif pour mettre en place cette protection quel que soit le pilote de la base de données utilisé.

10. Une protection Clickjacking activée par défaut

Un petit changement, Drupal 8 envoie maintenant des entêtes ***X-Frame-Options*** **: SAMEORIGIN** par défaut dans toutes ses réponses. Cette entête est respectée par la plupart des navigateurs et empêche le site d'être accessible dans un *iframe* sur un autre domaine. Cela bloque les attaques de type *clic-jacking* (par exemple, des formulaires ou des liens présentés de façon déguisés sur le site d'un attaquant à l'intérieur d'une iframe), ainsi que la réutilisation non autorisée du contenu du site via les *iframes*. (7.x arrière-port).

- Les changements d'entrée : https://www.drupal.org/node/2514152
- Davantage d'infos à lire sur le clickjacking : https://www.owasp.org/index.php/Clickjacking

11. L' API JavaScript du core Drupal compatible avec la spécification CSP (sécurité stricte des contenus)

Le support pour l'*inline JavaScript* a été retiré des propriétés de *#attached dans la Drupal render* API. Par ailleurs, les variables Javascript (settings) sont maintenant ajoutées à la page comme des données JSON puis chargées dans une variable au lieu d'être incluses directement dans la page dans des balises Javascript. Les settings Javascript étaient l'utilisation d'*inline javascript* dans Drupal 8. Cela signifie que les gestionnaires de sites peuvent beaucoup plus facilement activer une politique CSP — un nouveau standard Web pour communiquer des restrictions par site aux navigateurs et atténuer ainsi les risques de failles XSS et autres vulnérabilités.

- Davantage d'info à lire sur les CSP : https://www.owasp.org/index.php/Content_Security_Policy
- Les standard CSP W3C : http://www.w3.org/TR/CSP/

Une dernière note de précaution : la refonte et la réorganisation substantielle du code de Drupal 8 ainsi que la dépendance à des composants PHP tiers représentent certains nouveaux risques. La réorganisation du code peut avoir introduit des bugs qui n'ont pas été détectés par les tests du cœur Drupal existants. Les composants tiers eux-mêmes peuvent avoir des failles de sécurité qui affectent Drupal, c'est pourquoi nous avons besoin de suivre leurs évolutions et de les mettre à jour et réparer leur intégration pour toutes modifications d'API correspondantes. Pour diminuer le risque, l'Association Drupal a créé le premier *Drupal security bug bounty* pour toutes les versions du cœur de Drupal. Cela a permis de découvrir et réparer plusieurs failles de sécurité avant la sortie officielle de Drupal 8.
Je suis fier que nous ayons réussi à renforcer la « sécurité par défaut » de Drupal 8,

Code génial ou code débile

Abonnement : Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex. - Tél. : 01 55 56 70 55 - *abonnements.programmez @groupe-gli.com* - Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30. **Tarifs** abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 € Autres pays : nous consulter.
PDF : 30 € (Monde Entier) souscription sur www.programmez.com

Une publication **Nefer-IT**
7 avenue Roger Chambonnet
91220 Brétigny sur Orge
redaction@programmez.com
Tél. : 01 60 85 39 96

Directeur de la publication & rédacteur en chef : François Tonic

Secrétaire de rédaction : Olivier Pavie

Ont collaboré à ce numéro : S. Saurel

Experts : G. Lelarge, A. Le Priol, T. Lebrun, F. N'Guessan, J. Thiriet, A. Guillaume, M. Rebiai, N. Hilaire, H. Fakih, C. Villeneuve, V. Fading, J. Landon, M. Fahrasmane, Thomas Brouwer, N. Comet, C. Pichaud, F. Chevalier, G. Damien, M. Hubert, G. Burlot, P. Wolanin

Couverture : © eternalcreative

Maquette : Pierre Sandré

Publicité : PC Presse,
Tél.: 01 74 70 16 30, Fax : 01 41 38 29 75
pub@programmez.com

Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.

Marketing et promotion des ventes :
Agence BOCONSEIL - Analyse Media Etude

Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD
Téléphone : 09 67 32 09 34

Contacts
Rédacteur en chef : ftonic@programmez.com
Rédaction : redaction@programmez.com
Webmaster : webmaster@programmez.com
Publicité : pub@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

DÉVELOPPEZ 10 FOIS PLUS VITE

WINDEV 21

TDF TECH 2016

VOUS ÊTES INVITÉ !

100% TECHNIQUE

Inscrivez-vous vite !

Montpellier	mardi 8 mars
Toulouse	mardi 15 mars
Bordeaux	mercredi 16 mars
Nantes	jeudi 17 mars
Bruxelles	mardi 22 mars
Lille	mercredi 23 mars
Paris	**jeudi 24 mars**
Strasbourg	mardi 29 mars
Lyon	mercredi 30 mars
Marseille	jeudi 31 mars
Genève	mardi 5 avril

35 sujets techniques sur WINDEV 21, WEBDEV 21 et WINDEV Mobile 21.

11 villes

du 8 mars au 5 avril

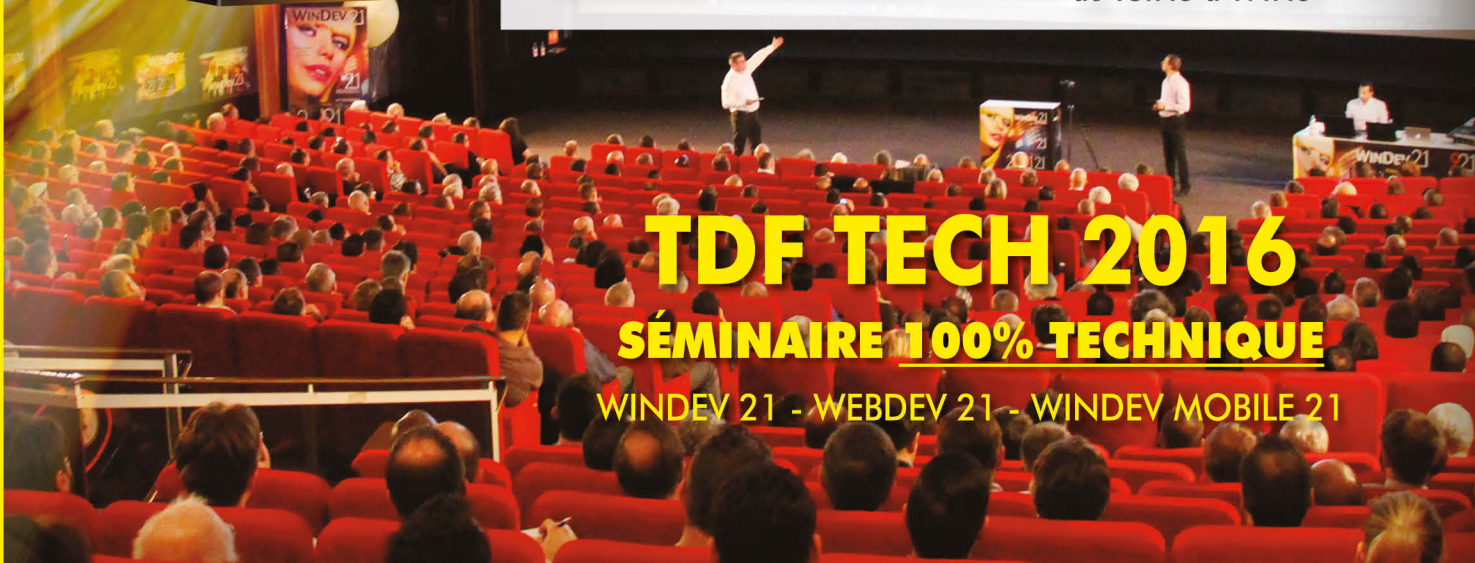
10.000 places

inscrivez-vous vite !

(gratuit)

www.pcsoft.fr

de 13h45 à 17h45



TDF TECH 2016

SÉMINAIRE 100% TECHNIQUE

WINDEV 21 - WEBDEV 21 - WINDEV MOBILE 21