SPÉCIAL RECRUTEMENT | SPÉCIAL RECRUTEMENT | SPÉ

programmez.com

[Programmez!]

Le magazine des développeurs

219 JUIN 2018

Choisir son framework

Nos astuces pour faire le bon choix

SPÉCIAL recrutement

+6 700 postes ouverts !

Les nouvelles méthodes de recrutement

UWP

La plateforme Microsoft pour les apps universelles

© StockPhotoAstu

TOP 10

Les meilleures astuces pour développer avec Kotlin

Kotlin

Xamarin Les nouveautés | Où va Xamarin ?

M 04319 - 219 - F: 6,50 € - RD

LE SEUL MAGAZINE ÉCRIT PAR ET POUR LES DEVELOPPEURS

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

EDITO

1 mise à jour = 1 serveur sauvé = 1 dév content

Il y a des comportements qui ne changeront jamais ou si peu. Depuis que je suis dans le monde informatique, depuis presque 35 ans, je me dis que tout change, rien ne change. Prenons les librairies, les systèmes, les composants. Quand on déploie un serveur, une application avec des dépendances techniques, nous utilisons des versions précises du composant X, Y, Z. Et ceux-ci ne vont pas évoluer automatiquement, exceptés des cas particuliers.

En février 2017, une faille WordPress avait permis de défigurer 1,5 million de pages web tournant sur la plateforme CMS ! Et pourtant, celle-ci avait été corrigée quelques semaines auparavant mais des webmasters, sysadmins et développeurs n'avaient pas appliqué la mise à jour. Et dès l'instant qu'une faille critique est officialisée et patchée, il faut réagir très rapidement pour éviter ces problèmes.

Le rapport State of Software Security 2017, édité par CA Technologies, indique que 28 % des entreprises évaluent régulièrement les composants utilisés en interne. Ce % peu élevé doit nous inquiéter. Admettons que ce chiffre soit sous-estimé et imaginons que nous soyons plus proche des 50 %. Cela signifie tout de même que la moitié des entreprises ne surveillerait pas les piles techniques. Et si cette surveillance est faite, le temps de réaction pour appliquer un patch, combler une faille laisse rêveur : 14 % des répondants de l'étude corrige dans les 30 jours.

Prenons l'injection SQL, un des grands classiques du hacking. En 2017, il y aurait toujours 25 % des applications concernées par celle faille. 25 % ! Et ce n'est pas faute de mettre en garde, de parler des failles, d'expliquer chaque année le OWASP. Mais non, l'injonction SQL est comme une sangsue, bien accrochée ! Heureusement les développements passent mieux les tests OWASP, mais nous sommes très loin des 100 %, à peine 40 %.

Ce manque de réactivité et surtout d'anticipation, concerne aussi bien les composants propriétaires qu'Open Source. Si votre Apache, Java, .Net Core, Drupal, etc., ne sont pas patchés, ne blamez pas l'éditeur ou les communautés.

Ce n'est pas pour rien que nous vous sensibilisons depuis 20 ans à la sécurité, aux méthodes de développements sécurisés, aux tests, aux bonnes pratiques, à l'agilité et DevOps. Aujourd'hui, le développeur dispose de tous les outils nécessaires. Mais, il faut aussi que l'entreprise, les utilisateurs comprennent que la sécurité est un sujet complexe qui exige du temps, de la veille constante et que le coût n'est pas neutre. Après, il ne faut pas faire des mises à jour à l'aveugle sans aucune précaution. Et ne faites pas des patchs directement en production…

C'est drôle sur CommitStrip et Les joies du code mais pas en vrai…

Je vous laisse, je vais déployer une obscure librairie non documentée trouvée sur le Darknet provenant d'un développeur au fin fond de la campagne chinoise ayant pour nom Unité 61398(*).

(*) si vous ne connaissez pas je vous laisse découvrir.

François Tonic
ftonic@programmez.com

SOMMAIRE

Dans le prochain numéro !
Programmez! #220, dès le 29 juin 2018

Migrer facilement d'AngularJS à Angular

Découvrez et utilisez GraphQL

Transformer son site web en Progressive Web App !

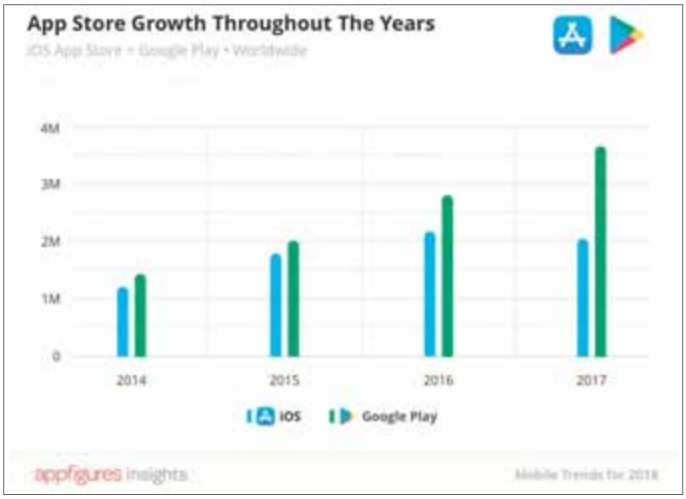
Une extension **Chrome** vole vos informations. Google avait déjà retiré plusieurs malwares de la boutique officielle mais visiblement il en reste encore.

Google propose un kit complet pour ajouter du Machine Learning dans des apps iOS et Android : MLKit. La solution est disponible sur Firebase. Site : https://developers.google.com/ml-kit/

Intel annonce ses premières puces 10nm, pour 2018 ou 2019.

Microsoft veut travailler avec Apple pour pouvoir afficher Messages dans la nouvelle application Your Phone.

2017 marque le net avantage d'Android sur iOS pour les nouvelles apps publiées dans les Store. Jusqu'à présent, l'écart n'était pas aussi net. C'est le double des apps qui ont été poussées sur Android (+1,5 millions). Cela se confirme sur le nombre total d'apps : 3,5 millions sur Play contre 2 sur App Store (source : appfigures).



Des employés de **Google** démissionnent à cause d'un projet IA, projet Maven, lié à l'armée américaine.

eX Core, un boîtier e-GPU trop beau pour être vrai, a été suspendu par Kickstarter. Le constructeur dit vouloir continuer et trouver des financements pour développer le produit.

Apple vs Samsung.

Le procès fleuve continue.

La folie des consoles rétro continue. La borne **Neo-Geo Mini** est annoncée mais sans date. La **NES Mini** devrait sortir fin juin. L'Atari VCS arrivera elle aussi bientôt pour un prix estimé à 200 €. Le prix est franchement cher quand on voit les clones de type Atari Flashback, vendus à partir de 45 €.

L'aérotrain, un transport sur coussin d'air glissant sur un rail en béton, renaît, un peu, de ces cendres avec le projet **SpaceTrain**. L'idée est venue d'une startup d'Orléans. L'ambition est de créer un mode de transport en tunnel interurbain. L'ambition est de construire et de tester les premiers prototype d'ici 2020 et éventuellement de réutiliser le rail de tests de l'aérotrain, toujours visible… La vitesse serait de 540 km/h, avec des pointes à 720. Avec l'effervescence autour de l'Hyperloop, intérêt qui reste à concrétiser dans le monde réel, le SpaceTrain peut-il convaincre ? Réponse dans les prochains mois.

ZTE EN PREND POUR 7 ANS

ZTE est un constructeur chinois et 4e marque de smartphones aux Etats-Unis. Le département du commerce américain a décidé d'interdire à ZTE pour 7 ans l'accès à certaines technologies essentielles pour le mobile : l'électronique et le logiciel. Fini Android et Qualcomm… Sa dépendance à certains fournisseurs technologiques oblige ZTE à revoir ses téléphones et surtout à trouver des alternatives et rapidement. On oublie souvent que les Etats peuvent influer sur les sociétés technologiques et de façon radicale. Mais la situation pourrait évoluer si le président américain décide de suspendre la sanction.

Angular 6

La dernière version d'Angular est sortie il y a quelques semaines. Une des nouveautés est le support complet d'Angular Elements. Elements permet de packager des composants Angular, sous forme de web components. La v6 embarque aussi un nouveau moteur de rendu : Ivy. Il promet une compilation plus rapide, des bundles de tailles réduites. Des soucis de full compatibilité existaient avant la version finale. Donc à vérifier. Sur la partie Angular CLI, on dispose maintenant de ng update pour faciliter la mise à jour des dépendances. Pour développer les composants Angular, on peut utiliser Component Dev Kit (CDK), qui n'est pas une nouveauté propre à la v6. Sur la partie Progressive Web Apps, Angular 6 apporte des corrections de bugs. On notera aussi l'apparition d'une nouvelle version de TypeScript (la 2.7), la mise à jour de RxJS. Pour la partie build, on pourra utiliser Bazel. L'outil s'intègre dans Angular Buildtools Convergence. Ce travail de fond sera complet d'ici fin 2018.
Cette version est dans la continuité des versions précédentes et ne nécessite pas de réécriture.

INDEX TIOBE

L'index TIOBE du mois montre finalement peu de changements pour les 6 premiers. Le top des recherches reste identique : Java, C, C++, Python, C#, VB.Net. Derrière, PHP connaît un petit regain d'activité tout comme Ruby. JavaScript baisse d'une place. L'index TIOBE prend en compte les recherches sur 25 moteurs de recherches

Mai 2018	Mai 2017	Changement	Langage	En %	Evolution en %
1	1		Java	16.380%	+1.74%
2	2		C	14.000%	+7.00%
3	3		C++	7.668%	+2.92%
4	4		Python	5.192%	+1.64%
5	**5**		**C#**	**4.402%**	**+0.95%**
6	6		VB .NET	4.124%	+0.73%
7	9	^	PHP	3.321%	+0.63%
8	7	v	JavaScript	2.923%	-0.15%
9	-	^^	SQL	1.987%	+1.99%
10	11	^	Ruby	1.182%	-1.25%

OPÉRATION POUR 1 EURO DE PLUS

Pour bénéficier de cette offre exceptionnelle, il suffit de commander WINDEV Mobile 23 (ou WINDEV 23, ou WEBDEV 23) chez PC SOFT au tarif catalogue avant le 29 Juin 2018. Pour 1 Euro de plus, vous recevrez alors le ou les magnifiques matériels que vous aurez choisis. Offre réservée aux sociétés, administrations, mairies, GIE et professions libérales, en France métropolitaine. L'offre s'applique sur le tarif catalogue uniquement. **Voir tous les détails sur : WWW.PCSOFT.FR** ou appelez-nous au 04.67.032.032

Le Logiciel et le matériel peuvent être acquis séparément. Tarif du Logiciel au prix catalogue de 1.650 Euros HT (1.980,00 TTC). Merci de vous connecter au site www.pcsoft.fr pour consulter la liste des prix des matériels. Tarifs modifiables sans préavis.

Aucun abonnement à souscrire.
Compatible tous opérateurs

WINDEV est un logiciel professionnel. Document non contractuel.

JUSQU'AU 29 JUIN

COMMANDEZ WINDEV MOBILE 23

OU WEBDEV 23 OU WINDEV 23

ET RECEVEZ LE NOUVEAU SAMSUNG Galaxy S9+

CHOISISSEZ :

- **SAMSUNG Galaxy Note8**

ou

- **SAMSUNG Galaxy S9** +carte SD 128Go

ou

- **SAMSUNG Galaxy S9+**

ou

- **Lot de 2 tablettes Samsung Galaxy Tab S2** 9,7"

ou

- **TV Wifi 140cm 4K SAMSUNG**

(Détails et **autres matériels** sur pcsoft.fr)

DAS S9+ : 0.362 W/kg. Le DAS (débit d'absorption spécifique) quantifie le niveau d'exposition maximal de l'utilisateur aux ondes électromagnétiques. La réglementation française impose que le DAS ne dépasse pas 2 W/kg pour une utilisation à l'oreille. Visuels non contractuels. Écrans simulés.

Atelier de Génie Logiciel Professionnel cross-plateformes

WWW.PCSOFT.FR

COMMUNAUTÉS

Annoncez vos meetups, et conférences sur Programmez ! :
ftonic@programmez.com

.Net Toulouse
18 juin : REX DevOps Apps mobiles avec Visual Studio App Center.
https://www.meetup.com/fr-FR/Meetup-NET-Toulouse/

GDG Toulouse
5 juin : GDG Banking, open banking & smart contracts.
http://www.gdgtoulouse.fr

ParisJUG
12 juin : Apache Maven et Java 9.
https://www.parisjug.org

Marseille JUG
7 juin : Microservices full-stack.
http://marsjug.org

MUG Strasbourg
5 juin : retour sur BUILD 2018.
https://www.meetup.com/fr-FR/MugStrasbourg/

React Nantes
Nouveaux meetups sur Nantes :
https://www.meetup.com/fr-FR/React-Nantes

JUIN

OW2con'18
La conférence annuelle de la communauté OW2 explore les valeurs éthiques et économiques de l'open source, les 7 et 8 juin 2018 à Paris.
Les points forts du programme incluent l'intelligence artificielle avec une rencontre dédiée au projet ACUMOS AI, par l'équipe Orange Labs, ainsi que des sujets d'actualité tels que la sécurité et la protection des données (incluant le RGPD), la blockchain, le test et la qualité. Les dernières avancées de la base de code OW2 seront également présentées. Enregistrement (gratuit) en ligne sur : https://www.ow2con.org/2018

AI Paris 2018
Les 11 et 12 juin 2018 ouvriront les portes du rendez-vous le plus attendu de la scène AI hexagonale : AI Paris 2018 ! Pour cette deuxième édition le congrès invitera l'ensemble des décideurs et utilisateurs métiers pour 48h d'immersion dans l'ère du business augmenté ! Pour en savoir plus : https://tinyurl.com/y9ko5w9x

EclipseCon 2018
13 & 14 juin, Toulouse
EclipseCon France est un des évènements majeurs organisés par la communauté open source Eclipse en Europe. Événement fédérateur, EclipseCon France est l'opportunité de rencontrer les contributeurs, les éditeurs, les intégrateurs ou les utilisateurs des technologies Eclipse et de créer des collaborations à la fois techniques et business. Les sujets traités sont variés : les objets connectés, Cloud & Devops, Data Science, Modeling, ainsi que Jakarta EE (ex-Java EE) et Eclipse MicroProfile, sujets présents en force pour la première fois. https://www.eclipsecon.org/france2018/

DevFest Lille
Le Devfest Lille 2018 se passera le 21 juin 2018 avec 400 personnes prévues. La conférence prendra place dans les locaux de IMT Lille-Douai (20 Rue Guglielmo Marconi, 59650 Villeneuve-d'Ascq). Le site est accessible via
https://devfest.gdglille.org/
Beaucoup de bonnes sessions : applications instantanées sur Android, EcmaScript, gRPC, Prometheus et Grafana, Kotlink, Angular, HTTP2, Android Things, Data stream.

Hack in Paris
Du 25 au 29 juin, Hack in Paris est la grande conférence sur la sécurité et le hacking en France. Un évènement à ne pas rater !
Les 3 premiers jours sont réservés aux formations. 18 formations seront données sur Corelan, hacking IPv6, rootkits sous Linux, tests de pénétration bas niveau matériel, attaque app mobile, etc. Les deux derniers jours sont réservés aux conférences, toutes particulièrement intéressantes : drones, exploit matériel, obfuscation toolkit, NFC, le développement logiciel non sécurisé.

Conférences techniques
Xebia organise 3 conférences techniques
- Paris Container Day, conférence pionnière en France dédiée à l'écosystème des conteneurs. Cette année, le thème sera : "Vivre avec l'Orchestration". 26 juin 2018 - paris-container-day.fr/
- FrenchKit, la première conférence française dédiée aux développeurs iOS et macOS. 20 et 21 septembre 2018 - frenchkit.fr/
- XebiCon, la conférence qui vous donnera les clés pour tirer le meilleur des dernières technologies : Data, Architecture, mobilité, DevOps, etc. 20 novembre 2018 - xebicon.fr

JUILLET – AOÛT

Apple II Festival France
La quatrième édition du petit frère français de Kansas Fest aura lieu du 1er au 5 août 2018 au Maska à Castéra-Verduzan dans le Gers. Au programme ici aussi des sessions sur le matériel et le logiciel, des ateliers et beaucoup de convivialité.
https://www.apple2festivalfrance.fr/

Donnez le meilleur à vos développeurs et à vos équipes !

Abonnez-les à [Programmez!] Le magazine des développeurs

© bowie15

Depuis 20 ans, le magazine Programmez! est un véritable guide mensuel de veille, de décryptage, de tendances, d'articles et de dossiers qui va à l'essentiel de l'actualité dans toutes les sphères de la programmation.

Nous proposons aux entreprises et aux écoles des tarifs spéciaux pour les équipes à partir de 5 personnes.

Contactez-nous dès aujourd'hui pour un devis personnalisé.

Contact : Olivier Pavie
06 12 36 29 36
opavie@programmez.com

Disponible en version papier et en version PDF. France et étranger.

François Tonic

Un nouveau jeu sort sur… Amstrad CPC.

Qui a dit que les jeux sur les plus belles machines des années 80 et 90 n'existaient plus ? Eric nous parle de son nouveau jeu qu'il vient de sortir sur… Amstrad CPC et sur disquette 3 pouces ! Un projet génial !

Peux-tu te présenter en quelques mots : ton parcours, tes dévs marquants ?

Je suis Eric SAFAR, j'ai commencé à programmer au début des années 80 sur Oric puis sur Atari ST. J'ai été engagé comme programmeur chez Legend Software fin 1988 où j'ai travaillé sur des titres comme :

- *Les Portes du Temps* (Atari ST) ;
- *1789 La révolution Française* (Atari ST) ;
- *RanXerox* (Atari ST) ;
- *Ninjanimal*, etc.

Puis fin 1992, je rencontre Remi Herbulot qui m'engage chez *Cryo Interactive* pour faire la conversion de *Megarace* sur Mega CD. Je vais rester 10 ans dans l'entreprise et bosser principalement sur la série de jeux d'aventures : *Atlantis*.

Cryo Interactive disparaît en 2003, je fonde alors *Atlantis Interactive* pour continuer la licence, je ferai deux titres supplémentaires.

- *Atlantis Evolution* (édité par Dreamcatcher);
- *Secrets of Atlantis* (édité par Nobilis).

Je fais du développement Homebrew sur Oric, Amstrad, Commodore, Atari et Amiga à partir de 2013.

Tu as sorti chez Safargames un nouveau jeu pour… Amstrad CPC. Pourquoi le CPC ?

Athanor est une trilogie ! Lorsque j'ai développé *Athanor 1 - L'éveil*, sur Oric (ma machine de cœur), j'ai découvert que la communauté Amstrad était assez active dans le Homebrew. J'ai donc décidé de faire la conversion sur Amstrad CPC. C'est de plus une machine attachante, pleine de ressources ! Pour *Athanor 2 - La légende des hommes-oiseaux*, l'Amstrad s'est imposé !

Sans dévoiler les secrets du jeu, il parle de quoi ?

Déformation professionnelle de mes "années" Cryo, je reste fasciné par la légende des Atlantes, on peut donc dire que le scénario s'en inspire. ;-)

Comment s'est passé le développement, l'audio et le graphisme ? Quel langage as-tu utilisé et l'outillage nécessaire ?

L'idée d'*Athanor 1* était de "rendre hommage" à la génération de jeux en graphismes fil de fer comme les machines des années 80 ont connu beaucoup de titres célèbres (*Mystère de Kikekankoi*, *Manoir du Dr Génius* et beaucoup d'autres). Pour *Athanor 2*, j'ai dû m'entourer de talents de graphistes et d'un musicien très impliqués dans la scène Amstrad car pour ce titre, tous les graphismes sont en Bitmap. C'est le mode 1 - 320x200 - 4 couleurs qui a été retenu. Pour le mode graphique sur CPC, nous aurons du 320x200 - 16 couleurs sur Atari et 32 couleurs sur Amiga !

Sur Amstrad CPC, j'ai voulu garder le couple mythique des langages utilisés pour ce style de jeux, à savoir, l'assembleur et le Basic. Un noyau en Basic pilote toutes les routines ASM qui gèrent le graphisme ou les scripts de scénarios notamment.

Combien de temps a demandé ce développement ? Et quels ont été les principaux problèmes ?

Dans les développements Homebrew, j'aurais tendance à dire que le problème principal est le temps ! Car très souvent c'est l'association de gens qui travaillent en dehors et qui prennent sur leur temps libre pour avancer sur le projet. Me concernant, 80% de mon temps libre passe dans cette passion dévorante, il faut avoir un entourage compréhensif ! Pour ce type de jeu, je dirais que 18 mois sont nécessaires depuis l'idée de scénario jusqu'à la production du packaging.

Athanor sera disponible à 100 exemplaires seulement. Pourquoi une disponibilité aussi limitée ? As-tu eu des problèmes à trouver les disquettes ?

Effectivement, pour la version Amstrad CPC, j'ai réservé 100 exemplaires.

La production d'un packaging de qualité est difficile et trouver un prestataire pour produire 400 boîtes n'est pas forcément aisé sans faire exploser les coûts !

Pour la BigBox d'*Athanor 2*, j'ai dû investir env. 3500€ et je ne parle pas des indices matériels qui se trouvent dans la boite ! Les autres exemplaires seront partagés entre la version Atari ST, Amiga et peut-être PC, Sega.

Quel sera ton prochain défi ?

Je suis un passionné de jeux d'aventures donc mon prochain titre sera logiquement une nouvelle aventure mais vers des horizons très différents que celle de la trilogie Athanor ! Ce jeu sera nativement développé pour Atari ST et Commodore Amiga donc plus ambitieux à tous points de vue, son nom de code est : Pennylane ! ;-)

Le dernier opus de la trilogie Athanor verra le jour seulement après ce titre intermédiaire.

Contact Safargames :
eric@safargames.fr
www.safargames.fr

tangente
l'aventure mathématique

Choisissez votre abonnement :
PLUS avec les HS « kiosque »
SUPERPLUS avec la Bibliothèque Tangente

Mieux comprendre le monde grâce aux mathématiques

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

La version numérique pour tous

Il est maintenant possible de consulter sans limitation de durée les numéros et hors séries de *Tangente* en ligne sur le site **tangente-mag.com** pour tous ceux qui prennent l'abonnement numérique ou une des formules d'abonnement papier. Un véritable site interactif et pas un simple PDF. L'abonnement permet aussi d'accéder gratuitement aux problèmes du « Monde » sur **www.affairedelogique.com**

• • • • • Tangente Éducation

Trimestriel qui, pour un coût symbolique, traite de thèmes pédagogiques variés, donne des outils aux enseignants **et permet l'accès à de nombreuses ressources Internet.**
Nouvelle version numérique interactive !

Les hors séries « kiosque » • • • •

4 fois par an, un hors série (format « kiosque »)
Ces numéros d'au moins 56 pages explorent un grand dossier de savoir ou de culture.
Derniers parus : Les angles - Les graphes - Les démonstrations - Les fonctions - Maths des assurances - Maths et médecine - La droite - Maths et architecture - Complexes Les ensembles - Maths et économie - Découpages et pavages - Espaces vectoriels
Disponibles :
– sur **www.infinimath.com/librairie**
– chez votre marchand de journaux
– **avec l'abonnement PLUS ou SOUTIEN.**

Les hors séries « Bibliothèque » • • • •

Pour nos lecteurs les plus curieux, les articles des hors séries de *Tangente* sont repris et complétés dans les livres de la Bibliothèque Tangente, magnifiques ouvrages d'environ 160 pages (prix unitaire 22,00 € à partir du 58), richement illustrés, disponibles
– sur la librairie du site **www.infinimath.com**
– chez votre libraire
– ou **en souscription avantageuse avec l'abonnement SUPERPLUS ou SOUTIEN.**

NOUVELLE FORMULE DE PAIEMENT !
Choisissez **l'abonnement permanent** par prélèvement.
Vous êtes débité **tous les 6 mois** de la somme indiquée dans la colonne "Permanent" en arrêtant quand vous voulez après deux ans.

Bulletin d'abonnement valable jusqu'au 31–08–18

À retourner à : Espace Tangente – service abonnement – 2 rue de la Prée – 27170 COMBON

NOM* ..... PRÉNOM* .....
ADRESSE* .....
CODE POSTAL* ..... VILLE* ..... PAYS* .....
MAIL* ..... PROFESSION .....

codif : Programmez

Abonnement Nos formules	Papier + Numérique			Numérique seul			Supplément papier hors métropole		
	1 an	2 ans	Permanent	1 an	2 ans	Permanent	1 an	2 ans	Permanent
ABO TG PLUS (6 n°s + 4 HS par an)	☐ 65 €	☐ 124 €	☐ 30 €	☐ 42 €	☐ 80 €	☐ 19 €	☐ 25 €	☐ 50 €	☐ 12,5 €
ABO TG SUPERPLUS (par an, 6 n°s + 4 livres Bibliothèque + 4 HS numérique)	☐ 102 €	☐ 198 €	☐ 48 €				☐ 30 €	☐ 60 €	☐ 15 €
ABO SOUTIEN : 6 n°s de *Tangente* + 4 HS « kiosque » + 4 « Bibliothèque » + 4 n°s de *Tangente Éducation*.	☐ 134 €	☐ 258 €	☐ 64 €				☐ 40 €	☐ 80 €	☐ 20 €

MONTANT TOTAL : ☐ €

MODE DE PAIEMENT
❒ Chèque (ordre Éditions POLE, uniquement payable en France) ❒ Bon de commande administratif
❒ Prélèvement : je recevrai mon mandat SEPA par mail et le renverrai signé
IBAN : ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐
BIC : ☐☐☐☐ ☐☐☐☐ ☐☐☐☐
❒ Carte bancaire numéro ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ Date d'expiration ☐☐/☐☐
Cryptogramme ☐☐☐ **DATE :**
SIGNATURE :

Baromètre HIRED recherche d'emploi

Tous les mois, Programmez! publie en exclusivité le baromètre Hired des technologies les plus recherchées par les entreprises. Elles peuvent permettre aux développeurs de connaître les nouvelles tendances du recrutement pour se former ou se démarquer des autres candidats. Les recruteurs IT plébiscitent les développeurs Go, React et Python. La demande pour les profils Vue et DevOps est en forte baisse sur les 6 derniers mois.

Pour le mois d'Avril 2018, le baromètre Hired de la recherche d'emploi met en évidence une certaine disparité entre les technologies attendues par les recruteurs IT et celles maîtrisées par les candidats. Le mois dernier, les langages qui ont le plus intéressé les entreprises étaient, dans l'ordre, Go, React et Python avec respectivement une moyenne de 10,9, 7,4 et 7,3 entretiens proposés par candidat. Pour autant, Go n'est mis en avant que dans 2,7% des candidatures tandis que Python et React le sont respectivement dans 12,6% et 9,6%.

Dans les valeurs sûres, on retrouve toujours le Java, le PHP et Node qui restent les technologies les plus représentées dans les candidatures. En avril 2018, elles ont respectivement permis aux candidats les ayant mises en avant d'être conviés à 5,1, 6 et 6,7 entretiens en moyenne. Au cours des 6 derniers mois la tendance est quasiment identique à deux grosses exceptions près : le DevOps et Vue. Alors que de novembre à avril, la moyenne d'entretiens proposés aux développeurs DevOps était de 8,9, elle a chuté à 3,7 le mois derniers. Pour Vue, ce chiffre est passé de 8,5 à 3,2 dans le même laps de temps. •

de Novembre 2017 à Avril 2018

Technologies demandées	Pourcentage de candidatures développeurs	Technologies demandées	Nombre moyen de demandes d'entretiens
Java	18.22%	Go	11.35
PHP	14.23%	React	10.65
Node	14.14%	Node	8.96
Python	12.33%	DevOps	8.87
React	10.61%	Vue	8.48
Angular	9.97%	Python	7.90
Android	5.35%	Angular	7.75
.NET	4.08%	PHP	6.84
Ruby	3.90%	Ruby	6.63
Go	2.81%	Java	6.00
iOS	2.45%	iOS	5.59
Vue	1.90%	.NET	5.40
DevOps	1.36%	Android	4.76

Avril 2018

Technologies demandées	Pourcentage de candidatures développeurs	Technologies demandées	Nombre moyen de demandes d'entretiens
Java	18,72%	Go	10.09
PHP	15.02%	React	7.38
Node	13.30%	Python	7.33
Python	12.56%	Node	6.69
React	9.61%	PHP	6.02
Angular	9.61%	Angular	5.77
Android	6.16%	Vue	5.50
.NET	5.42%	Java	5.09
Ruby	3.69%	.NET	5.09
Go	2.71%	Ruby	5.07
iOS	1.72%	DevOps	3.75
Vue	1.48%	Android	3.16
DevOps	0.99%	iOS	2.29

CYBERSÉCURITÉ : À SURVEILLER

Les métiers de la sécurité explosent ! La cybersécurité est un profil très recherché et les candidats ne sont pas assez nombreux pour répondre aux besoins actuels. La sécurité se retrouve à tous les niveaux : infrastructure, code, utilisateur, réseau. Ce sont plusieurs milliers de postes qui seront à pourvoir dans les prochaines années.

A propos de HIRED

HIRED.com est une plateforme technologique de recrutement qui attire et sélectionne les meilleurs profils techniques du marché afin de permettre aux entreprises de recruter des candidats qualifiés rapidement et efficacement. Chaque semaine, entre 50 et 100 nouveaux candidats de la communauté française HIRED en recherche active (développeurs, data scientists, designers, etc.) sont triés sur le volet grâce à des algorithmes et prêts à être contactés.

Recrutement printemps 2018 : +6 700 postes ouverts sur l'année !

Notre dernier grand dossier recrutement remonte à 2016 ! Il était temps de faire un point et de vous proposer un panorama des sociétés qui recrutent des développeurs et des profils techniques : 50 entreprises, 6700 postes.

La rédaction

LES TYPES D'ENTREPRISES

Sans surprise, 46 % des entreprises sont des SSII / ESN, des sociétés de services IT, de la petite structure à la plus grande (Cap Gemini). Puis nous trouvons : les TPE / PME, les éditeurs, les grandes entreprises, et, à peine visible, l'administration / service public. On oublie souvent que les TPE / PME sont un véritable vivier d'opportunités car elles cherchent aussi des développeurs. Ces postes sont parfois moins visibles.

On constate que la taille des entreprises (effective) est relativement homogène dans les réponses :

- *30 % sont à -50 salariés ;*
- *24 % entre 101 et 499 salariés ;*
- *22 % dépassent 1000 salariés.*

DES RECRUTEMENTS TRÈS DIVERS

Difficile de dresser un portrait du recrutement type. Si vous prenez Cap Gemini & Sogeti, c'est une recherche sur toute l'année de… 3 000 personnes ! D'autres entités ne cherchent que quelques profils. Les ESN (Entreprises de Services du Numérique) auraient souvent des campagnes de recrutement plus massives. Parmi les répondants, citons Cap Gemini, Devoteam, Smile, Squad, IT Link, Avanade, SQLi, qui cherchent minimum 200 personnes sur l'année.

Vous constaterez que l'on a, dans cette catégorie, des pures players (Open Source ou technos Microsoft), et des généralistes. Si vous êtes plus orienté open source, une ESN open source pourrait être une première piste. Que cela ne vous empêche pas de prospecter vers des sociétés généralistes. Parfois, les pures players cherchent aussi à élargir leurs compétences et ouvrir de nouveaux domaines techniques.

Les profils recherchés sont très divers, comme vous le constaterez dans le tableau général : Scrum Master, ingénieur, testeur, tech lead, cloud architect, développeur, chef de projet, technicien.

Même diversité sur la partie technique : 9 sociétés sur 50 recherchent des compétences Java, JavaScript (et les environnements liés notamment à Angular toujours demandés) suivies de près par les profils Cloud. Les compétences de développement mobile sont aussi très demandées. Python n'est pas le profil le plus recherché mais il est bel et bien présent.

Sur la partie salaire, difficile d'établir une grille complète. 23 réponses indiquent les salaires. Le salaire moyen brut annuel est d'environ 40,34 k €. Il s'agit d'une référence basse. Les écarts sont très importants selon les profils recherchés et les compétences. Les salaires débutent à 30-33 000 €. Selon le poste, on peut atteindre 60-70 000 €.
Un salaire médiant s'observe dans la tranche 40-45 000 €.

A vous de jouer !

Comment l'Epita prépare les étudiants ?

Les écoles d'informatique doivent préparer les étudiants au monde de l'entreprise et à passer les entretiens. Laurent Trebulle, directeur des relations avec les entreprises à l'Epita, répond à nos questions.

Comment une école comme l'Epita prépare ses étudiants à affronter les recrutements ? Ont-ils une idée de la réalité du marché ?

La préparation de nos étudiants au recrutement s'intègre dans un processus global de gestion du projet professionnel.

En janvier de chaque année les étudiants suivent une semaine dédiée à la construction de ce projet ; dans le rapport qu'ils rendent une partie est réservée aux outils de recherche de stage/emploi ; ils doivent avoir rédigé/actualisé leurs CV et également rédiger une lettre de motivation répondant à une annonce qu'ils ont sélectionnée.

Ces documents sont corrigés et annotés par un ancien RRH et les étudiants sont libres de prendre en compte tout ou partie de ses recommandations.

Au mois d'avril nous organisons une journée au cours de laquelle des RRH d'entreprises partenaires (cette année 25 entreprises), reçoivent les étudiants qui se sont inscrits (175 cette année). Les étudiants doivent se présenter et gérer cet entretien avec un recruteur, avec les avantages que ce système comporte : une écoute bienveillante et un retour constructif sur les points forts et les points faibles du candidat. Si certains présentent des difficultés face à l'exercice, ils sont invités à d'autres exercices.

Début mai nos étudiants ont une semaine spéciale dite « speed dating stage » au cours de laquelle les entreprises viennent se présenter et présenter leurs offres de stages pendant 30 minutes. Les étudiants rejoignent ensuite les entreprises qui les ont intéressés pour passer à la phase de candidature.

Toutes ces activités ont lieu durant la première année du cycle ingénieur. Elles sont destinées à permettre aux étudiants de trouver le stage qui correspondra le mieux à leur projet professionnel et d'être plus performants dans la conduite l'entretien en lui-même.

En dernière année du cycle ingénieur nous proposons de nouveau l'exercice des simulations d'entretien. Là les étudiants sont évidemment beaucoup moins nombreux à le suivre.

Enfin lors du forum stage emploi, les étudiants rencontreront de nombreuses entreprises (73 en 2017), quand des anciens élèves sont présents sur les stands, ils expliquent souvent le déroulement du processus de sélection.

Une à deux fois par an, des anciens élèves ayant été recrutés chez les GAFAM, reviennent à l'école pour partager leur expérience du processus de recrutement des géants du numérique, qui allient plusieurs difficultés : tests techniques de haut niveau, interviews en vidéo-conférence, etc.

Dans notre domaine de l'informatique et des nouvelles technologies, le marché du recrutement est extrêmement favorable aux jeunes. Il est clair qu'il y a beaucoup plus d'entreprises qui ont des besoins que de jeunes sortant de nos écoles, de ce fait les étudiants sont dans une situation rare et privilégiée : c'est à eux de choisir parmi les offres de stages ou d'emploi qui leur sont proposées. Toutefois, quand on cherche un emploi précis dans une entreprise donnée, on doit travailler sa recherche de poste et ses entretiens ! A l'EPITA, les 2/3 des étudiants signent leur 1er contrat dans l'entreprise dans laquelle ils ont fait leur stage de fin d'études. La recherche d'emploi est donc anticipée et l'étape cruciale est donc la recherche du stage idéal.

Aujourd'hui, les canaux de recrutements sont très diversifiés : réseaux sociaux, hackathon, « speed dating » de recrutements, salons, etc. comment observez-vous ces tendances car elles impactent les entreprises et les développeurs ?

Les canaux de recrutement sont effectivement plus diversifiés que par le passé et parmi les nouveaux médias, les réseaux sociaux tiennent la première place avec LinkedIn et Facebook.

Sur les groupes dédiés de l'école sur Facebook, des offres sont proposées dans un esprit de cooptation par des alumnis vers les étudiants.

Les jobboards gardent néanmoins leur attractivité car ils permettent à l'étudiant de faire une recherche multicritères au moment où il le veut. Des résultats parfois plus anciens peuvent aussi leur donner une piste vers une entreprise ou un contact sur un sujet correspondant précisément à ce qu'il recherche. JobTeaser est le site qui s'impose depuis plusieurs années comme la référence en France, d'autres existent pour le marché US par exemple.

De mon point de vue, les hackathons, les after-work ou les salons dédiés à une entreprise sont des démarches sympathiques qui peuvent fonctionner pour les entreprises à forte notoriété, mais attirent au final assez peu de nos jeunes.
D'autres expériences sont intéressantes comme les forums à thème de SeeKube ou la promotion de la marque-employeur avec WelcomeToTheJungle.

Aujourd'hui les conférences dans les écoles, les forums, la prise en charge de modules de cours, ou les rencontres directes avec des anciens élèves restent les vecteurs privilégiés. Ces évènements fonctionnent encore bien car ce sont les entreprises qui se mettent à la portée des étudiants, se déplacent pour les rencontrer et ceux-ci, en général, apprécient .
Enfin pour connaître un peu mieux l'ambiance dans les entreprises, l'arrivée d'un site comme Glassdoor va permettre au jeune de confronter l'image présentée par le recruteur à l'avis des salariés.

Quels conseils donner aux entreprises qui recrutent et aux développeurs ?
Investissez dans la relation école, c'est un processus long (comme celui de nos formations), mais qui installe l'image de l'entreprise auprès des étudiants. Ce n'est pas nécessairement immédiatement payant mais je suis persuadé que, dans la durée, c'est une carte incontournable. Ceci pour peu qu'il y ait une certaine pérennité des interlocuteurs. Il y a en effet quelques entreprises dans lesquelles j'ai l'impression de connaître plus de monde que mon nouvel interlocuteur fraîchement recruté pour promouvoir l'entreprise.
Mobilisez vos dirigeants pour des interventions dans les école ! D'une part, cela va les sortir un peu de leur business quotidien et les confronter à des jeunes qui seront leurs ressources de demain. D'autre part, les dirigeants sont – souvent – capables d'associer un discours technique à une mise en perspective, ce qui enrichit la culture de nos étudiants.

Quels profils sont les plus porteurs actuellement ? Et ceux qui le sont le moins ?
Ça recrute dans tous les domaines de l'IT en ce moment, avec sans doute un accent sur trois domaines en 2018 : le développement autour des infrastructures (passage au cloud et à l'infrastructure As A Service), la cybersécurité, et, bien sûr, l'Intelligence Artificielle. Cette dernière est le grand « buzzword » de l'année avec des annonces nombreuses d'ouverture de centres de recherche en France et de projets. •

Entreprise	Secteur d'activité	Type d'entreprise	Effectif	Recrutements prévus d'ici fin 2018	Les principaux profils recherchés (postes, compétences techniques)
8kmiles	Intégration de systèmes SSII / ESN	SSII / ESN	+1000	100	Cloud PMO - Cloud Architect - Cloud PreSales (Cloud = AWS or Azure or GCP).
ABAS FRANCE	Intégration de systèmes SSII / ESN	Editeur logiciel	10 à 49	10	Développeur, Consultant, Chef de projet.
AKERVA	Consulting	SSII / ESN	10 à 49	45	Auditeurs Sécurité / Pentester , Experts SOC, Experts en Sécurité IoT & Systèmes Industriels, Architectes Sécurité
Avanade	Intégration de systèmes SSII / ESN	SSII / ESN	+1000	200	Tout profil ayant des compétences sur les technologies Microsoft.
b.workshop	Intégration de systèmes SSII / ESN	SSII / ESN	10 à 49	3	Développeurs Juniors - Technologie Java / web services - dans le cadre de la mise en oeuvre de solution ERP en mode Saas.
CACD2 (Crédit Agricole)	Services	Grande entreprise	10 à 49	15	Développeurs.
Capgemini et sa filiale Sogeti	Intégration de systèmes SSII / ESN	SSII / ESN	+1000	env. 3000 d'ici fin 2018	- Profils recherchés : Consultants juniors et confirmés, consultants fonctionnels, consultants en stratégie et organisation, ingénieurs réseaux, administrateurs réseaux, ingénieur étude et développement concepteurs, architectes, chefs de projet, directeurs de projet, directeurs de programme, commerciaux... - Expertises recherchées : NTIC (type Java J2EE, .NET, FullStack, etc) et sur les technologies dites SMAC (Social, Mobile, Analytics, Cloud) : technologies liées à l'expérience et à la relation client ainsi qu'au digital, telles que Hybris (SAP), Salesforce, Exalead ou encore Eloqua (Oracle), et celles liées au Big Data (Hadoop, Pivotal…), pour répondre aux attentes des entreprises en matière de services digitaux et mobiles. En termes de compétences, tout ce qui est lié aux méthodes Agiles (Devops/Scrum…) nous intéresse.
CEGEDIM OUT-SOURCING	Services	Grande entreprise	101 - 499	25-30	Ingénieurs Infrastructure, Administrateurs Systèmes et Réseaux, Ingénieur Poste de travail, Développeurs.
Compart	Autre	Editeur logiciel	10 à 49	2	consultant informatique junior
Devoteam	Intégration de systèmes SSII / ESN	SSII / ESN	+1000	800	Ingénieur Big Data, Consultant Cloud, Consultant cyber sécurité, consultant Devops, Ingénieur ITSM, Ingénieurs Systèmes, Chef de projet Infrastructure, Chef de projets Réseaux, Architectes (Infrastructures, réseaux, application), développeur Java, Python, mobile.
Docteur Ordinateur	Services	TPE / PME	50-99	25	Technicien de maintenance en microinformatique au domicile des particuliers.
Econocom	Intégration de systèmes SSII / ESN	SSII / ESN	+1000	150	Concepteur-Développeur, Consultant, Designer, Développeur, Ingénieur Support Applicatif.
Exakis	Intégration de systèmes SSII / ESN	SSII / ESN	101 - 499	120	Consultant DataCenter & Cloud Azure / Architecte IAM / Consultant poste de travail / Consultant Sécurité SI / Consultant SharePoint /Chef de projet Infrastructure / Architecte et ingénieur PKI / Architecte Office 365 / Consultant .net / Architectes conseil dans les profils recherchés.
Expensya	Fournisseur technologique / informatique	Editeur logiciel	50-99	40	- Commercial B2B - Développeur FullStack - Développeur Front-End / Web Desig - Développeur Web - Développeur Mobile - Chargé de Marketing de contenu - Responsable SEM - Traffic Management -Stage en Marketing
Foliateam	Services	TPE / PME	101 - 499	25	Administrateur réseau et système, Expert virtualisation, Ingénieur avant vente Voix et réseaux, Ingénieur visioconférence et communications unifiées, Technicien support, Expert ToIP Mitel, Expert téléphonie Alcatel.
IARA	Construction / BTP	Editeur logiciel	-10	2	Développeur(euse) Unity 3D - C# - RA (Années d'expériences : 1 à 2 ans). Compétences : maîtrise du développement avec le moteur Unity et Visual Studio. Maîtrise de la programmation orientée objet C#, Connaissance des technologies de réalité augmentée, Notions en UX/UI).
IBM Interactive	Fournisseur technologique / informatique	Grande entreprise	+1000	30	Developpeur Front-end / Developpeur NodeJS / Developpeur Blockchain.
INEAT	Intégration de systèmes SSII / ESN	SSII / ESN	101 - 499	40	Développeurs Java, PHP, Angular, Webmethods, NodeJS, Android, Chef de projet technique.
Infinite Square	Intégration de systèmes SSII / ESN	TPE / PME	10 à 49	6	Nous recherchons des développeurs back et front passionnés souhaitant travailler sur une stack Microsoft & friends : .Net, C#, Xamarin, Azure mais aussi Angular et React.
ISTIA - Ecole d'ingénieurs de l'Université d'Angers	Education / formation	Administration / service public	+1000	Pas précisé	Qualité Logicielle, Test Logiciel, Sûreté de fonctionnement logiciel.
IT LINK	Intégration de systèmes SSII / ESN	SSII / ESN	500-999	250	Ingénieur développement logiciel, développement web, ingénieur système, ingénieur réseaux, ingénieur en sûreté de fonctionnement, ingénieur commercial.
JETPULP	Marketing	TPE / PME	50-99	15	Back end, Front End.
Kaliop	Intégration de systèmes SSII / ESN	SSII / ESN	101 - 499	40	Devops / admins.sys confirmé (H/F), Développeur front-end web HTML (H/F), Développeur / lead Fullstack NodeJS (H/F), Développeur / lead Drupal 8 (H/F), Développeur / lead Symfony 3+ (H/F), Développeur / Lead Ezpublish 5 / EZplatform H/F, Développeur / Lead Magento H/F, Intégrateur/trice sénior (Magento) H/F, Data / Bigdata Architects Sénior H/F, Développeur / Lead Mobile React Native H/F.
Kamp'n	Fournisseur technologique / informatique	Editeur logiciel	10 à 49	4	Profils avec expérience PHP Symfony et/ou AngularJS.
KYMONO	Communications	TPE / PME	10 à 49	Pas précisé	Vente et marketing.

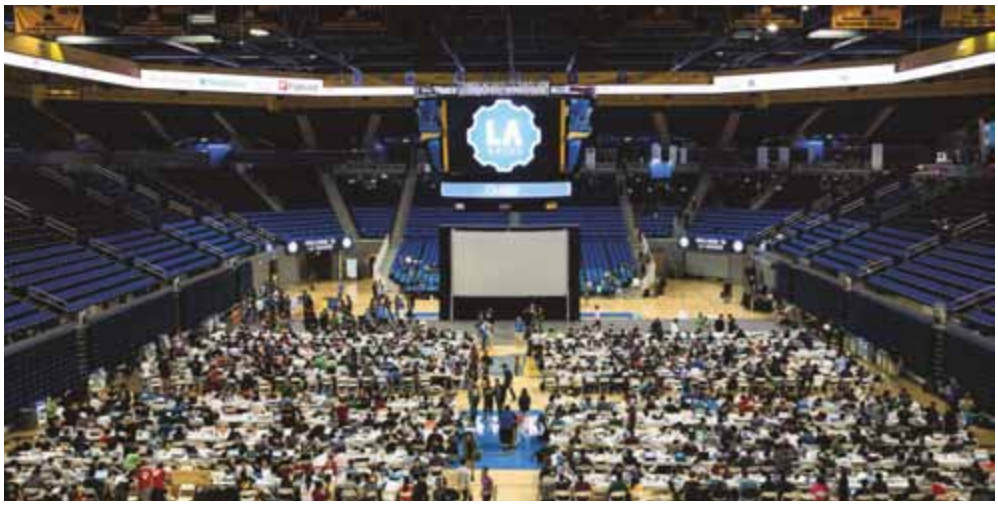
Salaires moyens proposés	Avantages	Adresse	Contact
à partir de 70 k€ (Profil sénior uniquement)	Télétravail (~30%) - Certifications - 2 summits/an minimum.	Switzerland	Nicolas CHABRIER (Regional Director - EMEA) - nicolas.c@8kmiles.com
Non précisé	Horaire flexible, programme de formation complet, autonomie.	26 Rue Louis Guérin 69100 Villeurbanne	Didier Delhaye, ddelhaye@abas-france.fr
50 k€	Partage des connaissances entre experts de la cybersécurité, missions variées, challengeantes & techniquement poussées, formation & certification pour nos consultants, Lab IoT, formation en e-learning pour l'ensemble de nos salariés.	Paris, Rennes & Toulouse	maud.bagourd@akerva.com
dépend du level	Package de rémunération, plan épargne entreprise abondé, budget annuel pour achat de matériel informatique, mutuelle prise en charge a 60%, prime vacances, abonnement téléphonique, carte AMEX	3 esplanade Du foncet 92130 Issy Les Moulineaux	antoine.quenet@avanade.com
30-35 k€ selon profil	Intéressement - tickets restaurant - Bonne humeur et esprit d'équipe dans une structure dynamique accompagnant la progression des collaborateurs.	37 rue de Neuilly 92110 Clichy	contact@bworkshop.fr
50 k€	Projets innovants, bonus, Télétravail possible Prise en charge des frais de transport Tickets Restaurant, salle de sport	9 rue Antoine Chantin 75014 Paris	olivia.tamvakis@cacd2.fr
Non précisé	Comité d'entreprise, restaurants d'entreprise, participation, système de formation continue très complet…	5/7 rue Frédéric Clavel 92287 Suresnes Cedex	https://www.capgemini.com/fr-fr/carrieres/ https://www.fr.sogeti.com/rejoignez-nous/nos-offres-demploi/
40/45 k€	13ème mois, Tickets Restaurant, CE.	15 Rue Paul Dautier 78140 Velizy	Mme Dominique Garros, Responsable Recrutement - dominique.garros@cegedim.com
Non précisé	NC	129 rue Servient 69326 Lyon Cedex 03	Benoit Arribart 0478636990
39 k€ pour un jeune diplômé, salaire moyen de 47 k€ depuis le début de l'année, toutes séniorités confondues	CE, Tickets Restaurant, mutuelle, PEE.	73 rue Anatole France 92300 Levallois-Perret	candidature@devoteam.com
Non précisé	Docteur Ordinateur recrute en 2018 des franchisés sur tout le territoire national.	62 rue Brançion, 75015 Paris	franchise@docteurordinateur.com
Non précisé	NC	11 Square Léon Blum 92800 Puteaux, France	Julien.VOYRON@econocom.com
Non précisé	CE, Tickets Restaurant, mutuelle, PEE, Plan de formation & de certifications	35 Boulevard des Capucines 75009 Paris	bertag@exakis.com (Sur Lyon, Aix en Provence, Toulouse, Grenoble, Clermont Ferrand) levente@exakis.com (Sur Paris, Nantes, Bidart)
45 k€ par an pour les développeurs et 40 k€ pour les commerciaux et les postes en marketing	Évoluer au sein d'une startup innovante et dynamique, travailler avec une équipe passionnée, forger ses compétences et en acquérir de nouvelles.	25 Rue de la Reynie 75001 Paris	jobs@expensya.com
35 k€	Tickets Restaurant, voiture de fonction (en fonction du profil)	4 passage Dartois Bidot 94100 Saint-Maur-des-Fossés	rh@foliateam.com
Non précisé	NC	46/48 avenue du Général Leclerc 92100 Boulogne-Billancourt	contact@horus-bim.com
Non précisé	NC	17 avenue de l'Europe 92270 Bois-Colombes	frtskrec@fr.ibm.com
Variable en fonction du profil	Tickets Restaurant, CE, coach sportif, RTT, cours de guitare.	2 allée de la Hayte du Temple 59160 Lomme	Céline KONA-BOUN, HR Business Partner 06 79 59 11 69
Non précisé	NC	33 rue du Faubourg Saint Antoine 75011 Paris	jobs@infinitesquare.com
35 k€ pour un débutant	NC	62 avenue Notre Dame du Lac 49000 Angers	NC
Selon profil	Cadre de travail agréable, 100% prise en charge transports collectifs.	67 avenue de Fontainebleau 94200 Le Kremlin Bicêtre	kjenck@itlink.fr
36 k€	Tickets Restaurant, PEE, Prime ancienneté.	12 av Tony Garnier 69007 Lyon	job@jetpulp.fr
Non précisé	NC	1401 Avenue du Mondial 98 34000 Montpellier	recrutement@kaliop.com
33 k€	Tickets Restaurant, Mutuelle, esprit startup !	17 rue fort notre dame 13001 Marseille	charles@kampn.com
Non précisé	NC	25, rue du petit Musc - 75004 Paris	www.kymono.co

Entreprise	Secteur d'activité	Type d'entreprise	Effectif	Recrutements prévus d'ici fin 2018	Les principaux profils recherchés (postes, compétences techniques)
Lelivrescolaire.fr	Education / formation	TPE / PME	10 à 49	5	Développeur Fullstack (JS, React, Node) ; UX Designer ; Product Manager.
LIMONETIK	Fournisseur technologique / informatique	TPE / PME	10 à 49	12	Développeurs .Net, Développeurs Full Stack, Expert Data, Chefs de projets techniques, Expert KYC.
Maltem	Consulting	SSII / ESN	+1000	100	Développeurs web, mobile et blockchain.
MONDIAL RELAY	Services	Grande entreprise	500-999	5	Chefs de projets, responsables de domaines.
Monext	Fournisseur technologique / informatique	TPE / PME	101 - 499	20	Des responsables d'applications : profils technico fonctionnels (capables de maîtriser le métier tout en ayant un background technique complet), des directeurs/trices de projets monétiques, des spécialistes réseaux et sécurité (administration et ingénierie, loadbalancing, firewalling...) ainsi que des personnes passionnées par le web !
NeoLynk	Intégration de systèmes SSII / ESN	SSII / ESN	50-99	30	Développeur Java, JS, PHP et Product Owner, Coach Agile, Scrum Master, Développeur Big Data.
Neos-SDI	Consulting	SSII / ESN	101 - 499	45	.NET / SharePoint / Dynamics CRM / MS BI / Office 365 / SCCM
NetXP	Consulting	SSII / ESN	50-99	20	Des profils juniors et/ou confirmés sur les postes suivants : Ingénieur d'affaires, Consultant et Architecte Cloud AWS ou Azure, Consultant Réseaux, SOC Manager, Consultant Sécurité des SI, RSSI, Consultant SMSI, Ingénieur de production, Ingénieur Systèmes Windows et Responsable Opérationnel de Comptes.
OBJECTIF LIBRE	Intégration de systèmes SSII / ESN	TPE / PME	10 à 49	5	Administrateur système et Consultants Cloud, 2 à 5 ans d'expérience, sur Paris et Toulouse.
OUTSCALE	Fournisseur technologique / informatique	TPE / PME	101 - 499	30	Ingénieur Développeurs Python, Ingénieurs Intégration Continue (culture devops), Chefs de projet Junior, Ingénieurs QA Automatisation, Administrateurs/Ingénieurs Système Linux (environnement essentiellement Linux)
Prevision.io	Fournisseur technologique / informatique	Editeur logiciel	10 à 49	6	Data Scientist, Intégrateur Front End, Software Engineer.
SCC France	Intégration de systèmes SSII / ESN	SSII / ESN	+1000	150	Ingénieurs Systèmes Microsoft, Ingénieurs Réseaux et Sécurité, Ingénieurs d'Affaires, Ingénieurs Cloud.
SEA TPI	Intégration de systèmes SSII / ESN	SSII / ESN	101 - 499	40	Pilote d'exploitation / Analyste / Ingénieur Système et BDD.
Sentryo	Fournisseur technologique / informatique	Editeur logiciel	10 à 49	5	Ingénieurs développement Linux embarqué, Golang, frontend Angular.
Smile	Intégration de systèmes SSII / ESN	SSII / ESN	+1000	500	Développeurs Java EE / Php / C++ / Python / Odoo / Talend ESB / Experts technique / Chefs de Projet & Directeurs de Projet Web / Consultants digitaux / Ingénieurs Système DevOps
SOFTBANK ROBOTICS	Fournisseur technologique / informatique	Grande entreprise	500-999	30	Développeurs senior, 2 à 5 ans d'expériences confirmés, notamment sur technologies C++, Python et/ou Android et experts fonctionnels et techniques dans des domaines variés (audio, image, traitement du signal, ...) afin d'accompagner les évolutions de nos solutions logicielles de robotiques humanoïdes pour les années à venir. également des postes ouverts en DataScience, DataAnalyst et gestionnaires de portefeuilles projets (AMO, PMO...).
SoftFluent	Consulting	SSII / ESN	50-99	30	Développeurs (Profils juniors et expérimentés), Team Leaders, Expertise sur les technologies Microsoft (C#, .NET).
Spectrum Groupe	Intégration de systèmes / SSII / ESN	SSII / ESN	50-99	5	Ingénieur informatique
SQLI	Intégration de systèmes SSII / ESN	SSII / ESN	+1000	500	Ingénieur concepteur développeur, leader technique, expert et architecte technique.
Squad	Intégration de systèmes SSII / ESN	SSII / ESN	101 - 499	250	Consultant Cybersécurité / Ingénieur systèmes Linux / Ingénieur réseaux et sécurité / Consultant DevOps / Consultant Cloud / Développeur Full Stack / Ingénieur, études et développement.net / Ingénieur systèmes et réseaux
Synapse Développement	Fournisseur technologique / informatique	TPE / PME	10 à 49	10	- Développeurs web - Développeurs back end - Chercheurs en Intelligence Artificelle - Ingénieurs linguistes - Commerciaux
TRADESHIFT	Fournisseur technologique / informatique	Editeur logiciel	500-999	5	Chef de projet, Architecte Technique, spécialiste intégration technique ERP.
Transatel	Fournisseur technologique / informatique	TPE / PME	101 - 499	30	Ingénieurs développeurs Java, Scrum Master, Chef de projet IT et Telecom, Architecte logiciel, Architecte Telecom, Ingénieur réseaux,...
WRAPTOR	Intégration de systèmes SSII / ESN	Editeur logiciel	10 à 49	1	Technicien Informatique Interopérabilité Java.
XEBIA	Intégration de systèmes SSII / ESN	SSII / ESN	101 - 499	60	Nous recherchons avant tout des personnes passionnées, agiles, férues d'innovation et de culture Devops. Profils techniques : Développeur(se)s Back (Java, Scala, Node.js). Développeurs Front/ Fullstack (Angular, Vue.js, React). Data Engineers, Data Scientists, Data Architects. Développeur(se)s Mobile (Java, Kotlin, Swift).

Salaires moyens proposés	Avantages	Adresse	Contact
Non précisé	Bureau magnifique, équipe en or, ambiance trop sympa :)	14 rue Rhin et Danube, 69009 Lyon	Jonathan Banon, cofondateur et CTO, jonathan.banon@lelivrescolaire.fr
40-60 k€	Mutuelle, Ambiance Startup, Paris centre, Participations/intéressement à termes, en plein développement international ...	8-10 Rue Saint Fiacre 75002 Paris	rh@limonetik.com
Non précisé	Projets grands groupes, stack technique innovante, Formations et Meet up.	8 place du marché 92200 Neuilly sur seine	Solene Simonet
selon expérience	Rémunération variable	5 avenue Antoine Pinay - 59510 Hem	contactdrh@mondialrelay.fr
La rém est fonction de l'expérience !	Une rémunération composée d'un fixe et d'un variable (proportionnel à l'atteinte d'objectifs... non capés à 100%), de participation, d'intéressement voire de bonification ! Un parcours d'intégration soigné avec notamment des formations proposées dès l'arrivée de nos nouveaux collaborateurs !	260 rue Claude Nicolas Le doux 13290 Aix	Rejoindremonext@monext.net
Selon expérience	Tribus technologique : JavaScript, PHP, JVM, Agile, Craftmanship - Formation « sur mesure » avec la possibilité de passer des certifications - Laboratoire d'innovation et de R&D - Stack technique innovante - Projets grands groupes - Atmosphère familiale - Great Place to Work depuis mars 2018	4 Rue Eugène Renault 94700 Maisons Alfort	jobs@neolynk.fr
Selon profil	Locaux Sympa, Paris Centre, Parc de la tête d'Or à Lyon, Dijon Toulouse, RTT, Ticket restaurant, Navigo, Mutuelle, Portable, primes, Télétravail	15-17 rue Auber - 75009 Paris	Aurélia Andreu-Menny
Selon profil	Télétravail - Portable professionnel avec abonnement - Formation & Certifications - Lab interne - Plan Epargne Entreprise - Management de proximité par des consultants seniors - Partage et capitalisation des connaissances - Un esprit convivial (ski, jeux, sports, afterwork) - Entreprise labelisée Great Place To Work 2018 et Happy Trainees 2018	62 bis Avenue André Morizet 92100 Boulogne-Billancourt	Brice MAROUQUE
40 à 60 k€	Tickets Restaurant, intéressement, RTTs, prévoyance, mutuelle contrat famille, team buildings et participation aux conférences Open Source mondiales.	359 rue Saint Martin 75003 Paris	jobs@objectif-libre.com
40 à 60 k€	Télétravail partiel/Formation/Accompagnement aux certifications/Environnement technique challengeant/Autonomie/Sens de l'initiative fortement valorisé/Prime de cooptation/Ambiance conviviale (jeux, goûter, challenges internes, cadeaux...)	1 rue royale - 92210 Saint-Cloud	recrutement@outscale.com
Non précisé	NC	Station F, Paris, France	florian.laroumagne@prevision.io
45 000 - 55 000€	NC	96 rue des Trois Fontanot 92744 Nanterre Cedex	Alexandre COTTEAUX Responsable Recrutement
2500 € (brut, mensuel)	Tickets restaurant / la valeriane / Complémentaire santè (+ selon les postes).	531 avenue serpolet 13600 La Ciotat	recrutement@seatpi.com
Non précisé	NC	66 boulevard Niels Bohr, INSAVALOR CEI-1, 69100 Villeurbanne	https://sentryo.workable.com
Non précisé	Formations "sur mesure" avec la possibilité de passer des certifications, Travailler dans un milieu à forte compétence fonctionnelle et technologique, Environnement évolutif et dynamique, Développer votre carrière vers de l'expertise, de la gestion de projet ... Atmosphère familiale, Support de la Direction Technique, constituée d'experts certifiés, Espaces collaboratifs (blogs, espaces, documentaires, wiki, …) et de capitalisation.	20 rue des Jardins 92600 Asnières Sur Seine	Géraldine Moreau - Luchaire
suivant expérience	Ouverture possible à l'international (nous sommes localisés sur la région Parisienne pour le centre d'ingénierie et au Japon et US pour les développements des ventes et des services associés).	43 rue du Colonel Pierre Avia 75015 Paris	rh@softbankrobotics.com
Rémunération fixée selon expérience	PC, smartphone, budget pour accessoires IT, accès Pluralsight, politique proactive de certifications, intéressement, participation, primes de cooptation, actionnariat salarié, Evènements réguliers techniques & corporate	5 Rue de la renaissance 92160 Antony	Marine GENETAY, Chargée de Recrutement mge@softfluent.com Tél 06 69 26 27 87
Selon profil	Tickets Restaurant - RTT - mutuelle.	8 rue de la renaissance - 92160 Antony	dora.delaporte@spectrumgroupe.fr
Selon profils : De 30 k€ pr un débutant en région à 80 k€ pr un expert à Paris	RTT Tickets Restaurant pris en charge à 60% par l'employeur 50% des transports Mutuelle familiale sans surcoût Primes de cooptation jusque 2000€ sans plafond Des locaux neufs et ultra design Nombreux événements créés spécialement pour nos collaborateurs	166 rue Jules Guesde 92300 Levallois Perret	Amélie Vieville (avieville@sqli.com)
En fonction du profil	Evolution de carrière (formation, certification, communauté) / recherche et développement / Evénements internes et externes réguliers : échange de savoir.	49 rue du faubourg Poissonnière, 75009 Paris	Roxane Desfarge : rdesfarge@squad.fr
Non précisé	Tickets Restaurant, RTT, mutuelle, sourires et bonne humeur !	7 Boulevard de la Gare 31000 Toulouse	patrick.seguela@synapse-fr.com
Non précisé	NC	54 AVENUE HOCHE 75008 Paris	cristina.zarici@tradeshift.com
Selon profils	NC	49-51 quai de Dion Bouton 92800 Puteaux	Virginie Desforges
SELON PROFIL	Tickets Restaurant + PEE	251 avenue des Paluds 13400 Aubagne	administratif@wraptor.fr
Selon expérience : entre 41 000 et 70 000 euros/ an	- Télétravail - Participation, PEE - MacBook Pro (ou autre équivalent si souhaité) - Remboursement illimité des livres techniques - Formation continue : journée mensuelle de partage de connaissance (appelées XKE) - Formation Softskills - Séminaire annuel - Mutuelle familiale sans surcoût, RTT CE,Tickets Restaurants... - Labellisée "Happy at work" :)	156 boulevard Haussmann 75008 Paris	laetitia janné - ljanne@xebia.fr

Les développeurs séniors sensibles à l'environnement de travail, les juniors au salaire

Sur un marché du recrutement de développeurs toujours très tendu, les entreprises peinent toujours à s'adresser correctement aux développeurs. En fonction de leurs profils, ces derniers n'ont d'ailleurs pas les mêmes attentes, ce qui rend la tâche d'autant plus ardue.



Avec l'arrivée des beaux jours il semble que le marché du recrutement de développeurs ait aussi profité de l'embellie. « *Nous avons eu un début d'année très difficile avec beaucoup de postes à pourvoir mais très peu de candidatures. Les choses commencent à s'améliorer depuis début mai* », témoigne Antoine Garnier-Castellane, directeur du bureau français de Hired, la plateforme de recrutement spécialisée dans les métiers de la technologie. Le dirigeant note toutefois que le marché reste très tendu avec toujours plus de postes à pourvoir par rapport au nombre de candidats.

Par ailleurs, Antoine Garnier-Castellane note une amélioration dans la stratégie adoptée par les entreprises pour leur recrutement de développeurs. « *Il y a toujours des sociétés qui essaient de recruter de manière traditionnelle avec des annonces très stéréotypées présentant des listes faramineuses de technologies à maîtriser. Ça ne marche pas ! Heureusement, d'autres tentent une autre approche beaucoup plus bénéfique* », explique-t-il. Certains RH commencent en effet à adapter leurs annonces. Ces dernières sont beaucoup plus personnalisées, plus qualitatives et s'adressent directement à un type de développeurs en mettant plus en avant les projets sur lesquels ils seront amenés à travailler. Il n'est plus questions de trouver un mouton à cinq pattes qui sera amené à tout gérer tout seul. « *Pour ce nouveau type d'annonces nous constatons des taux de réponse bien supérieurs de la part des développeurs* », assure Antoine Garnier-Castellane.

Les hackathons sont arrivés à maturité mais restent marginaux dans les processus RH

Il se montre toutefois plus critique à l'encontre des évènements comme les hackatons. « *Aujourd'hui, ces derniers, comme les serious games ou les concours restent plus un vecteur de communication que de RH. Et encore, avec un impact qui reste marginal* », estime Antoine Garnier-Castellane. Pour lui, ces initiatives ne ciblent qu'une partie de la population des développeurs qui ne se rendent pas tous à ce genre d'évènement. Il reconnaît toutefois qu'aujourd'hui, le procédé est arrivé à maturité. A voir la quantité et la variété des hackatons organisés (santé, expert-comptable, industrie, etc.), on ne peut que lui donner raison. Pour autant, le problème de communication existe toujours.

« *Nous avons toujours un problème de langage entre les entreprises et les développeurs, dans les deux sens du terme. Au delà des technologies vieillissantes sur lesquelles s'appuient encore certaines d'entre-elles, la question de la démarche à adopter pour recruter et des leviers de séduction reste toujours problématique* », avance le dirigeant qui rappelle qu'aujourd'hui, le salaire ne fait pas tout, surtout pour les profils de seniors. Avec ces derniers, il est en effet important de proposer un environnement technique attrayant et des projets stimulants. « *Si vous proposez 5 000 euros mensuels à un développeur ayant 10 ans d'expérience sans qu'il n'y ait derrière un projet intéressant, il y a peu de chances qu'il rejoigne votre entreprise. Non ! Ce qu'il faut c'est lui assurer un challenge technique et intellectuel* », explique Antoine Garnier-Castellane.

Les juniors sont moins sensibles à l'intérêt d'un projet

C'est toutefois moins vrai pour les profils de jeunes développeurs. « *Ceux qui sortent de l'école sont plus sensibles à la question du salaire qu'à celle de l'environnement de travail. A ce moment de leur carrière, c'est presque leur seule référence en matière de succès. Ils n'ont pas encore conscience de ce qu'est un projet intéressant* », explique le dirigeant. Les jeunes développeurs ont d'ailleurs tendance à se montrer plus difficiles que les seniors sur les questions de salaire. « *Quand ils sortent de l'école, ils sont sur-sollicités* », justifie le directeur de Hired. Les entreprises doivent donc adapter leur recherche à ces spécificités d'autant que certaines d'entre-elles, au-delà du problème de langage, se heurtent directement à un problème d'image.

« *C'est un fait, aujourd'hui, beaucoup des jeunes qui sortent des écoles refusent catégoriquement d'aller travailler dans certains secteurs, comme la banque et l'assurance qu'ils ne trouvent pas attrayants malgré tous les efforts menés pour y remédier* », conclut Antoine Garnier-Castellane. Il note toutefois que la situation s'améliore, bien que les grands comptes aient toujours un complexe vis-à-vis des startups. D'après lui, il est toujours difficile pour ces derniers de proposer des challenges qui intéressent réellement les développeurs.

•

Christophe GIGAX
MVP Visual Studio & Development Technologies

Tour d'horizon du développement Universal Windows Platform en C#

Avec l'arrivée de Windows 10 dans l'écosystème de Microsoft, un nouveau type de projet est apparu permettant aux développeurs C# de concevoir des applications Windows 10 fonctionnant à la fois sur tablette, PC, smartphone, Xbox, Raspberry, HoloLens : les Universal Windows Platform apps. La grande innovation ici c'est qu'avec un projet, donc un binaire final, l'application peut fonctionner sur chacune des plateformes citées. Le développeur n'aura donc besoin que de développer un code unique commun à toutes les plateformes. 1

1 *L'unicité de plateformes regroupées en plusieurs familles différentes*

Windows Desktop Extensions for the UWP	10.0.10586.0
Windows Desktop Extensions for the UWP	10.0.10240.0
Windows IoT Extensions for the UWP	10.0.10586.0
Windows IoT Extensions for the UWP	10.0.10240.0
Windows Mobile Extensions for the UWP	10.0.10586.0
Windows Mobile Extensions for the UWP	10.0.10240.0
Windows Team Extensions for the UWP	10.0.10586.0
Windows Team Extensions for the UWP	10.0.10240.0

2 *Extensions disponible pour votre projet*

Le nouveau SDK incorpore également un bon nombre de nouveautés permettant de gérer justement les différences entre les plateformes au niveau de la vue tout en gardant un seul code métier. De nouveaux contrôles et structures ont été rajoutés afin de diversifier un peu plus la palette d'outils XAML et d'améliorer les performances.

La notion de famille d'appareils est donc très importante ici. En effet, le développeur va pouvoir choisir quelle famille il souhaite cibler, permettant d'activer ou non un certain nombre d'API dans votre code.

Un système d'extension avancé

Afin de fonctionner sur les différentes plateformes, les projets UWP ont besoin d'ajouter les plateformes cibles dans les références du projet. Appelés des « Extensions », ces packages se trouvent dans le **Reference Manager** de votre projet. Faites clic droit sur votre projet, puis **Add Reference**. Sous l'onglet **Universal Windows** se trouve une section **Extensions**, c'est ici que vous trouverez les différentes extensions à rajouter. 2

Une fois les extensions rajoutées à votre projet, votre projet fonctionnera toujours sur toutes les plateformes, et une seule compilation sera nécessaire. Cependant, imaginons que nous voulions utiliser une API disponible sur Mobile mais pas sur un Raspberry Pi 2 (avec l'extension IoT et Mobile sur le même projet), par exemple le bouton **Volume** ou le bouton **Back**. Le développeur aura besoin de vérifier à l'exécution que l'API est bien présente sur la plateforme sur laquelle fonctionne l'application. Pour ce faire, nous allons utiliser l'**API Information**.

L'API Information

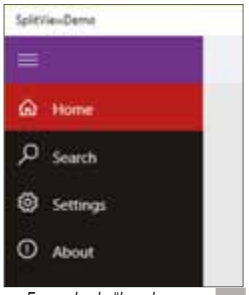
Cette API est apparue dans le SDK Windows 10 afin de vérifier à l'exécution qu'une classe ou encore une méthode existe bien sur la plateforme où fonctionne l'application. En effet, votre application est censée fonctionner à la fois sur mobile et sur un Raspberry. Cependant, il est clair qu'un Raspberry ne possède pas de touche de volume, ou de bouton pour ouvrir la caméra comme sur un smartphone. Il est donc important de bien vérifier que ces APIs sont présentes lorsque vous souhaitez les utiliser.

La classe statique **ApiInformation** se situe dans l'espace de nom **Windows.Foundation.Metadata** et contient plusieurs méthodes statiques telles que **IsApiContractPresent**, **IsEventPresent** ou encore **IsMethodPresent**. Ces méthodes vont vous aider à déterminer si l'API est présente ou non, et ainsi rendre plus robuste votre code.

Par exemple, pour vérifier que l'API concernant le Bluetooth est présente, on utilise le code suivant :

```
if (ApiInformation.IsApiContractPresent("Windows.Devices.DevicesLowLevelContract", 1))
{
 // Le Bluetooth est présent
}
```

Chacune des méthodes statiques prend en paramètre une chaîne de caractères correspondant à la classe ou à la méthode que l'on veut trouver. Ici, si la méthode **IsApiContractPresent** retourne vrai, cela veut dire que le Bluetooth est présent sur la plateforme courante où s'exécute l'application. Afin de trouver les APIs dont vous devez tester la présence, il n'y a pas de secret : il faut les trouver dans la documentation officielle de Microsoft. Le lien suivant regroupe la majorité des APIs utilisées dans les applications UWP : https://developer.microsoft.com/fr-fr/windows/develop.



Exemple de "hamburger menu" 3

Des contrôles XAML riches

Le SDK Windows 10 arrive avec son lot de contrôles XAML (l'éditeur étant directement intégré à Visual Studio), et notamment avec de nouveaux contrôles XAML très pratiques comme le **CalendarView**. Ce dernier permet d'ajouter un contrôle calendrier entièrement personnalisable (couleur de l'en-tête, couleur de fond, affichage des jours …) et vient ainsi compléter un peu plus les composants utilisables dans les applications Windows 10. Son utilisation est très simple :

```
<CalendarView
        DayItemFontStyle="Italic"
        CalendarItemForeground="Red"
      FirstDayOfWeek="Saturday"/>
```

Le calendrier ci-dessus a été personnalisé de la façon suivante :

- Les jours sont en italique ;
- La couleur des jours est rouge ;
- Le premier jour de la semaine est le samedi.

La liste complète des propriétés personnalisables du calendrier est ici : https://msdn.microsoft.com/library/windows/apps/windows.ui.xaml.controls.calendarview.aspx. Le **CalendarView** met à disposition un bon nombre d'évènements afin de récupérer les interactions utilisateur avec le calendrier, notamment l'évènement *SelectedDatesChanged* qui permet de notifier votre code que la date sélectionnée a changé.

Ensuite on retrouve le **CalendarDatePicker** qui peut servir afin de récupérer une date en particulier. Ce contrôle est de type « liste déroulante» et a été optimisé afin de récupérer une et une seule date.

```
<CalendarDatePicker
 IsGroupLabelVisible="True"
 IsTodayHighlighted="True"
 PlaceholderText="Sélection d'une date"/>
```

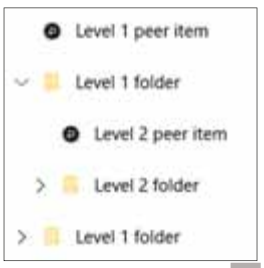
L'exemple ci-dessus montre un **CalendarDatePicker** avec les caractéristiques suivantes :

- Affichage du mois sur le premier jour du mois ;
- Affichage du jour d'aujourd'hui en surbrillance ;
- Placeholder spécifique.

Ici aussi, l'évènement *DateChanged* permet de savoir si la date a changé et de récupérer la nouvelle date indiquée par l'utilisateur.

Le contrôle **Maps** a également été enrichi de nouvelles fonctionnalités, comme l'imagerie 3D aérienne ou encore la vue au niveau des rues. L'autre information importante concernant le contrôle **Maps** est qu'il a enfin été unifié entre la version Mobile et Tablet/Desktop. Enfin, on ne retrouve plus qu'un espace de noms unique désormais permettant de faire fonctionner le contrôle sur toutes les plateformes. Sur Windows 8, une version existait pour Mobile et une autre pour le Desktop. Les nouveaux espaces de noms sont :

- *Windows.UI.Xaml.Controls.Maps* – espace de noms concernant le contrôle **Maps** ;
- *Windows.Services.Maps* – API pour des fonctionnalités avancées de géolocalisation et d'itinéraire. Afin d'utiliser ces services, il est important de posséder au préalable une clé depuis le Bing Maps Developer Center (https://www.bingmapsportal.com/).



4

Un nouveau contrôle fait son apparition, c'est l'**InkCanvas**. Il permet de facilement implémenter des fonctionnalités de dessin dans les UWP apps. De base, le contrôle ne supporte que le stylet, mais peut supporter la souris et/ou le *touch* via la propriété ***InkPresenter.InputDeviceTypes***. Le contrôle met également beaucoup d'évènements à disposition du développeur afin de gérer le cycle de vie du dessin de l'utilisateur via la propriété ***InkPresenter.StrokeInput***.

Enfin, le dernier contrôle important inclus dans le nouveau SDK Windows 10 est le **SplitView**. Ce dernier permet de mettre facilement en œuvre ce qu'on appelle un « hamburger menu ». 3

Le contrôle s'utilise de manière très simple. Il y a tout d'abord une partie ***Pane***, qui est la partie de gauche (ou de droite, c'est configurable), et qui permet de définir un menu avec divers boutons. Il est également possible de mettre ce que vous voulez, comme un calendrier, une carte ou bien d'autres choses. Ensuite, il y a une partie ***Content***, qui permet de définir le contenu de la page centrale. Le code ressemble alors à ceci :

```
<SplitView>
    <SplitView.Pane>
      <StackPanel>
        <Button Content="Menu 1"/>
        <Button Content="Menu 2"/>
        <Button Content="Menu 3"/>
      </StackPanel>
    </SplitView.Pane>
    <SplitView.Content>
      <StackPanel>
        <TextBlock Text="Mon contenu" />
      </StackPanel>
    </SplitView.Content>
  </SplitView>
```

Le développeur est ensuite libre d'effectuer la navigation entre les différentes pages en fonction du menu sélectionné.

Concernant la mise à jour d'Avril 2018, on retrouve principalement 3 nouveaux contrôles XAML : 4

- Le **TreeView** représentant une liste d'éléments hiérarchisés ;

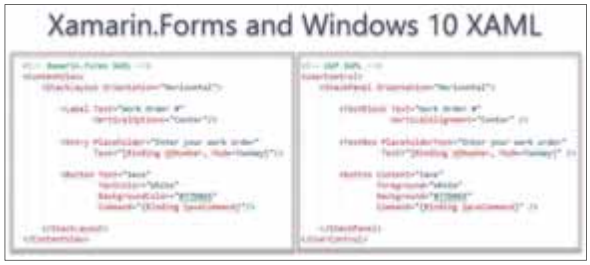
```
<TreeView>
  <TreeView.RootNodes>
    <TreeViewNode Content="Flavors" IsExpanded="True">
      <TreeViewNode.Children>
        <TreeViewNode Content="Vanilla"/>
        <TreeViewNode Content="Strawberry"/>
        <TreeViewNode Content="Chocolate"/>
      </TreeViewNode.Children>
    </TreeViewNode>
  </TreeView.RootNodes>
</TreeView>
```

- Le **Pull-to-refresh**, permettant à l'utilisateur de rafraîchir une liste d'éléments en « tirant » la liste vers le bas ;
- Les **ContentLinks** permettant de rajouter du contenu plus interactif dans les TextBox et les RichTextBox.

Le XAML Standard

Microsoft a lancé en fin d'année 2017 le **XAML Standard**. Comme son homologue .NET Standard, cette initiative vise à fournir une standardisation du langage XAML afin d'uniformiser l'écriture du XAML sur toutes les plateformes Microsoft. Diffusé en Open Source sur GitHub, l'équipe en charge de ce standard s'inspire grandement des propositions de la communauté afin d'établir ce standard : https://github.com/Microsoft/xaml-standard. D'ailleurs, une version préliminaire a déjà été fournie sous la forme d'un paquet NuGet à la plateforme Xamarin.Forms : https://docs.microsoft.com/fr-fr/xamarin/xamarin-forms/xaml/standard. On peut notamment y trouver une table des correspondances entre les versions Xamarin.Forms et XAML Standard des contrôles.

Xamarin.Forms	XAML Standard
Frame	Border
Picker	ComboBox
ActivityIndicator	ProgressRing
StackLayout	StackPanel
Label	TextBlock
Entry	TextBox
Switch	ToggleSwitch
ContentView	UserControl



Le Binding compilés

Depuis la sortie de la librairie WPF permettant de concevoir des applications modernes et riches, le binding est une technique très utilisée en XAML permettant de lier subtilement la vue avec des données. Grâce à cette technique, le pattern MVVM (Model-View-ViewModel) est facilement mis en œuvre, pour garantir ainsi un meilleur découpage entre la vue et le code métier de l'application.
Cependant cette technique a ses désavantages, comme les performances. Le binding est coûteux en temps d'exécution. Il est donc important de l'utiliser avec parcimonie. L'autre inconvénient que nous pouvons retenir est l'absence de vérification à la compilation. En effet, lors de la compilation, il est impossible de savoir si la propriété *liée* existe réellement ou non ; c'est uniquement à l'exécution que l'application va lever une exception si la propriété n'existe pas.
Le SDK Windows 10 apporte une nouveauté à ce niveau-là : c'est le binding compilé. A présent, le binding est résolu à la compilation et permet ainsi de détecter les erreurs plus tôt. L'écriture a également changé, et utilise maintenant la notation x:Bind pour mettre en œuvre le binding compilé.

```
<Button Content="{x:Bind MyProperty}"/>
```

Attention : la propriété liée (ici MyProperty) doit absolument se trouver dans le code-behind de la vue. Si vous utilisez le pattern MVVM, votre ViewModel devra obligatoirement exister en tant que propriété dans le code-behind pour retrouver le même fonctionnement qu'avant.

Les avantages de ce binding sont multiples :

- Meilleures performances (50% plus rapide) ;
- Résolution des types à la compilation ;
- Navigation à travers les propriétés du code-behind. Par exemple, le code suivant **Text="{x:Bind Employee.FirstName}"** ira chercher la propriété **FirstName** de la propriété **Employee** du code-behind.

Lorsqu'un **DataTemplate** est utilisé (pour lier une liste à une **ListView** par exemple), il est important d'indiquer le type avec **x:DataType** afin que le binding compilé puisse résoudre les types à la compilation. Le code suivant illustre cette nouvelle technique.

```
<DataTemplate x:Key="SimpleItemTemplate" x:DataType="data:SampleDataGroup">
    <StackPanel Orientation="Vertical" Height="50">
      <TextBlock Text="{x:Bind Title}"/>
      <TextBlock Text="{x:Bind Description}"/>
    </StackPanel>
  </DataTemplate>
```

Ici, on spécifie que le **DataTemplate** utilise des données de type **data:SampleDataGroup**, où se trouvent les propriétés **Title** et **Description**. Lors de la compilation, si ces propriétés ne sont pas présentes, une erreur surviendra.

La gestion des écrans

L'arrivée du SDK Windows 10 permet maintenant aux développeurs de développer la même application à la fois sur tablette, ordinateur et smartphone. De nos jours, il est relativement facile de partager du code métier entre différentes plateformes, mais est-il également facile de concevoir une même vue conçue pour fonctionner sur différentes tailles d'écrans ? Microsoft a dû intégrer plusieurs nouveaux mécanismes pour faciliter et améliorer le développement de nos applications de demain.

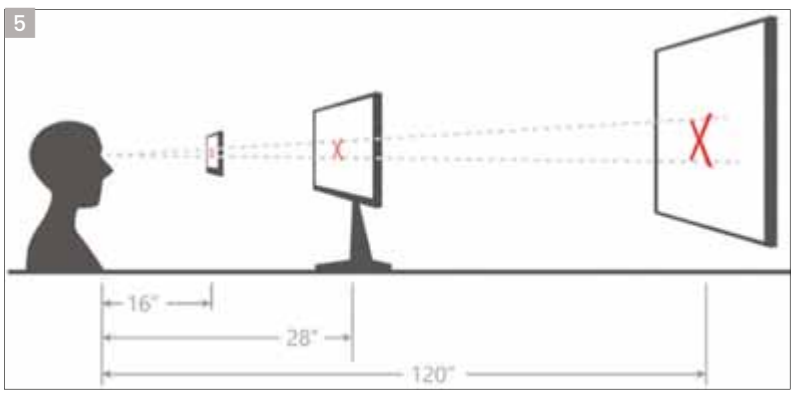
L'Effective Pixel

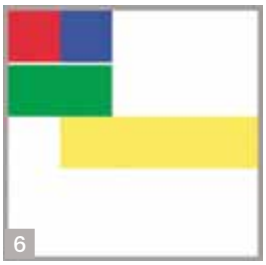
Lorsque l'on travaille avec une Universal Windows Platform app et que l'on conçoit une interface graphique, on ne travaille pas avec des pixels physiques, mais avec des **pixels effectifs**, permettant à la plateforme d'adapter automatiquement à l'échelle de l'écran les valeurs en pixel indiquées dans le fichier XAML. Cela est valable pour les marges, les polices et toutes autres propriétés acceptant des valeurs en pixel. 5

La raison d'un tel système au sein du SDK est simple. Un utilisateur utilisant l'application sur son smartphone ou sur sa télévision ne

Illustration de l'effective pixel

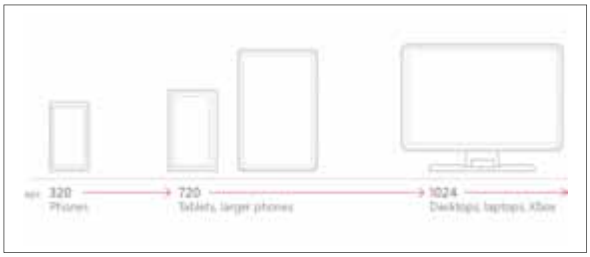
5





6

Résultat du code montré en exemple



sera tout d'abord pas à la même distance de l'écran. Ensuite, les résolutions n'étant pas les mêmes, si par exemple une police conserve le même nombre de pixels pour la police entre le smartphone et la télévision, le texte sera illisible sur le grand écran. Il est donc important d'avoir un algorithme qui recalcule la taille réelle des composants de la vue en fonction de la résolution.

Pour le développeur, cela ne change strictement rien durant le développement : l'algorithme de calcul des pixels physiques travaille en totale transparence sur l'application, et donc il n'a pas besoin de se soucier de la lisibilité de son contenu sur les différentes plateformes de son application. Pour résumer le principe de l'*effective pixel*, il est important de revenir sur ce que l'utilisateur perçoit. L'*effective pixel* permet à l'œil de l'utilisateur de croire que les éléments constituant les pages de l'application apparaissent tout le temps avec la même taille.

Le Relative Panel

Il y a des cas où le système de grille en XAML ne suffit plus pour construire sa vue. En effet, avec les composants à notre disposition aujourd'hui (Grid, Stackpanel, Canvas …) il est assez difficile de construire une vue complexe sans que notre fichier XAML soit très long et illisible.

Le SDK Windows 10 intègre un nouveau composant permettant de placer chaque élément l'un en fonction de l'autre, et ainsi de simplifier grandement la structure de notre vue : c'est le **RelativePanel**. Ce composant est de la famille des *Panels* et permet de disposer des éléments entre eux grâce à des *attached properties* (propriétés rattachées à un contrôle en particulier mais applicables à n'importe quel élément). Les principales propriétés sont les suivantes :

- **Above**, **Below**, **RightOf** et **LeftOf** permettent de positionner l'élément courant par rapport à un autre ;
- **AlignBottomWith**, **AlignTopWith**, **AligneVerticalCenterWith** et bien d'autres encore permettent de gérer l'alignement de l'élément courant avec un autre élément. Par exemple, la propriété **AligneBottomWith** va aligner le bas de l'élément courant avec le bas de la cible ;
- **AlignBottomWithPanel**, **AlignTopWithPanel** ou encore **AligneVerticalCenterWithPanel** permettent d'aligner l'élément courant avec le panel englobant de l'élément.

Prenons un exemple et analysons-le. Le code est le suivant :

```
<RelativePanel BorderBrush="Gray" BorderThickness="10">
  <Rectangle x:Name="RedRect" Fill="Red" MinHeight="100" MinWidth="100"/>
  <Rectangle x:Name="BlueRect" Fill="Blue" MinHeight="100" MinWidth="100"
        RelativePanel.RightOf="RedRect" />
  <Rectangle x:Name="GreenRect" Fill="Green" MinHeight="100"
        RelativePanel.Below="RedRect"
        RelativePanel.AlignLeftWith="RedRect"
        RelativePanel.AlignRightWith="BlueRect"/>
  <Rectangle Fill="Yellow" MinHeight="100"
        RelativePanel.Below="GreenRect"
        RelativePanel.AlignLeftWith="BlueRect"
        RelativePanel.AlignRightWithPanel="True"/>
</RelativePanel>
```

Le résultat est le suivant : 6

Les caractéristiques du code sont les suivantes :

- Le rectangle **Rouge** ne possède pas de caractéristique particulière, il est donc naturellement placé en haut à gauche ;
- Le rectangle **Bleu** possède la propriété **RelativePanel.RightOf** avec comme cible de rectangle **Rouge** : il est donc placé à sa droite ;
- Le rectangle **Vert** est, lui, placé en dessous du rectangle **Rouge** grâce à la propriété **RelativePanel.Below**. Ensuite, il possède les propriétés **AlignLeftWith** et **AlignRightWith** permettant de placer ses côtés au même niveau que les côtés des rectangles **Rouge** et **Bleu**. De ce fait, la largeur est automatiquement calculée par le **RelativePanel** ;
- Il en va de même pour le rectangle **Jaune** où on place son côté droit au même niveau que le côté droit du panel cette fois-ci. Dans ce cas, on utilise simplement un *booléen* (vrai ou faux) pour indiquer l'activation de l'alignement ou non.

Avec ceci, le développeur possède un premier outil afin de placer ses éléments dans sa vue, mais cela n'est pas suffisant. Il faut combiner ce mécanisme avec un autre pour adapter la vue en fonction de la taille de l'écran.

Le Visual State manager

Le **VisualStateManager** est un composant déjà connu avec la version de Windows 8 et 8.1 : il permet d'activer certains « états » (dans le code-behind) dans le composant permettant de modifier l'UI. Les modifications peuvent s'appliquer sur la visibilité d'un composant, sa taille, son placement, etc. Avec ce système, on pouvait déjà gérer les différentes tailles d'écrans mais cela se faisait côté C# via le code-behind. De ce fait, un designer ne pouvait pas intégralement concevoir une page XAML sans faire du code C#, ce qui peut être une contrainte s'il n'a pas les connaissances nécessaires.

Le SDK Windows 10 comble ce manque en intégrant les **AdaptiveTriggers** directement dans le **VisualStateManager**. Ces nouveaux composants permettent de définir des états dans le **VisualStateManager** qui vont être automatiquement lancés lorsque la condition de **l'AdaptiveTrigger** est remplie. Par exemple, il existe un **AdaptiveTrigger** permettant de gérer la taille de l'écran. L'idée serait alors de modifier l'UI lorsque l'écran dépasse un certain nombre de pixels, et ainsi adapter l'interface pour toutes les tailles d'écran. Prenons l'exemple ci-dessous :

```
<Grid x:Name="LayoutRoot">
    <VisualStateManager.VisualStateGroups>
      <VisualStateGroup>
        <VisualState x:Name="WideState">
          <VisualState.StateTriggers>
            <AdaptiveTrigger MinWindowWidth="600" />
          </VisualState.StateTriggers>

          <VisualState.Setters>
            <Setter Target="LayoutRoot.Background"
```

```
            Value="Green" />
          </VisualState.Setters>
        </VisualState>

        <VisualState x:Name="NarrowState">
          <VisualState.StateTriggers>
            <AdaptiveTrigger MinWindowWidth="0" />
          </VisualState.StateTriggers>

          <VisualState.Setters>
            <Setter Target="LayoutRoot.Background"
            Value="Red" />
          </VisualState.Setters>
        </VisualState>
      </VisualStateGroup>
    </VisualStateManager.VisualStateGroups>
  </Grid>
```

Les caractéristiques du code ci-dessus sont les suivantes :

- Le **VisualStateManager** définit 2 états de l'UI différents, un appelé **WideState** et l'autre **NarrowState** ;
- Pour chacun des états, on définit un **AdaptiveTrigger** permettant d'indiquer quand l'état doit être activé. Pour le **NarrowState**, l'état est activé lorsque l'écran fait au moins 0 pixel (sur smartphone il sera activé en premier) alors que pour **WideState**, il sera activé lorsque l'écran fera au moins 600 pixels ;
- En dessous de la section **VisualState.StateTriggers**, on définit ce que l'état doit faire lorsqu'il est activé. Dans chacun des états, on définit simplement une couleur de fond différente pour la grille avec le nom **LayoutRoot** (la grille englobante).

Avec ce système, il est ainsi facile de réorganiser sa vue avec un **RelativeLayout**. Le code suivant montre un exemple d'utilisation des **AdaptiveTrigger** avec un **RelativeLayout** :

```
<VisualState x:Name="WideState">
          <VisualState.StateTriggers>
            <AdaptiveTrigger MinWindowWidth="600" />
          </VisualState.StateTriggers>

          <VisualState.Setters>
            <Setter Target="FirstNameText.(RelativePanel.RightOf)"
                Value="FirstNameLabel" />
          </VisualState.Setters>
        </VisualState>

        <VisualState x:Name="NarrowState">
          <VisualState.StateTriggers>
            <AdaptiveTrigger MinWindowWidth="0" />
          </VisualState.StateTriggers>

          <VisualState.Setters>
            <Setter Target="FirstNameText.(RelativePanel.Below)"
                Value="FirstNameLabel" />
          </VisualState.Setters>
        </VisualState>
```

Le code ci-dessus modifie simplement la disposition du formulaire en fonction de la taille de l'écran. Lorsque l'écran dépasse 600 pixels, on place le champ de saisie à la droite du label car il y a assez de place pour le placer ainsi. Dans le cas contraire, on place le label en dessous : sur smartphone le formulaire devient ainsi plus lisible.

Il est également possible de définir ses propres **AdaptiveTriggers**, par exemple lorsque le développeur veut une vue particulière si l'application a accès à internet. Il suffit de créer une classe héritant de **StateTriggerBase** et d'utiliser la méthode **SetActive** (fournie par la classe mère) afin d'activer ou non l'état. L'exemple suivant montre comment gérer la connexion à Internet :

```
public class NetworkConnectionTrigger : StateTriggerBase
  {
    public NetworkConnectionTrigger()
    {
     NetworkInformation.NetworkStatusChanged += NetworkInformationOnNetworkStatusChanged;
    }

    public bool RequiresInternet { get; set; }

   private async void NetworkInformationOnNetworkStatusChanged(object sender)
    {
      await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
      {
        if (NetworkInformation.GetInternetConnectionProfile() != null)
          SetActive(this.RequiresInternet);
        else
          SetActive(!this.RequiresInternet);
      });
    }
  }
```

Le code ci-dessus active (ou non) l'état en fonction de la disponibilité de la connexion Internet. La classe **Dispatcher** est simplement utilisée ici pour revenir sur le thread principal de l'application (les évènements sont toujours exécutés dans un thread séparé) et activer l'état.

En combinant ces 2 systèmes, le développeur est maintenant capable de réagir à n'importe quelle taille d'écran et d'étendre ce système pour réagir finalement à n'importe quoi (connectivité Internet, Bluetooth, stockage, plateforme courante de l'application …). Il est possible de définir une infinité d'états et ainsi d'adapter les éléments de multiples façons possibles. Cela est d'autant plus vrai qu'avec Windows 10 l'application peut fonctionner sur des écrans allant de 5 pouces à plusieurs dizaines de pouces pour les télévisions, fonctionner en online comme en offline, posséder une carte SIM (ou non), fonctionner sur mobile ou sur desktop … Les **AdaptiveTrigger** permettent ainsi de gérer une infinité de cas possibles et travailler au mieux les vues de l'application.

Le Fluent Design

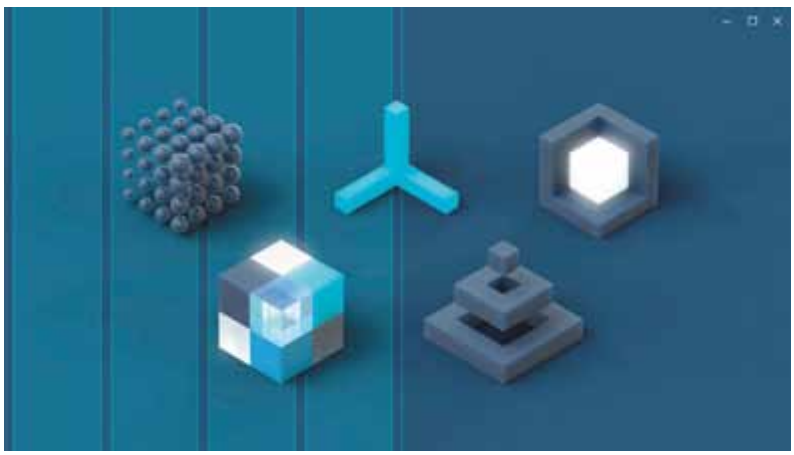
Le Fluent Design est un ensemble de concepts et de guidelines que Microsoft propose de plus en plus dans Windows 10 et dans le développement d'application UWP en XAML. Le développeur dispose ainsi de plusieurs outils et techniques afin de satisfaire au maximum au Fluent Design. Le premier concept à suivre est bien

7

Adaptivité de l'application selon les écrans

8

L'acrylique en tant que fond transparent d'une application



évidemment l'adaptivité de l'application à tout type d'écran. Les pages doivent automatiquement s'adapter à la résolution et la taille d'écran de l'appareil (smartphone, tablette, PC ...). Avec les **AdaptiveTrigger** et le **VisualStateManager**, le développeur dispose de tous les moyens pour y parvenir. 7

Qui dit multi-device, dit également multi-interaction. En effet, l'application doit rester fonctionnelle malgré les innombrables façon d'interagir avec elle : le *touch*, la souris, le clavier, le stylet, le *Surface Dial*, la manette Xbox … Les contrôles utilisés doivent également être adaptés à une utilisation sur n'importe quelle plateforme. C'est pour cela que le SDK Windows 10 intègre plus d'une trentaine de contrôles XAML afin de concevoir au mieux chaque interface graphique. Il est d'ailleurs possible de visualiser tous les contrôles XAML dans l'application **XAML Controls Gallery** disponible sur le Store.

Une autre facette du Fluent Design très intéressant est la partie **Style** et **Motion**. Ces 2 notions sont relativement récentes par rapport à ce qu'on a pu connaître avec le **Flat Design** sur Windows 8, et mettent l'accent sur la fluidité de l'interface au moyen de petit effets d'animation, de lumière, de transparence ou encore de couleur qui rendent réellement l'expérience utilisateur très immersive.

L'acrylique est un parfait exemple qui illustre les nouvelles pratiques UI/UX avec Fluent Design. Cet effet de texture transparent très particulier permet de rendre l'application plus agréable lors de son utilisation et au regard. 8

Pour le développeur, il est très simple de mettre en œuvre ce genre d'effet puisqu'il dispose de plusieurs ressources prédéfinies afin de l'appliquer à un fond d'un élément XAML. Il peut bien évidemment définir ses propres ressources acryliques selon la transparence et la lumière qu'il souhaite appliquer.

```
<Grid Background="{ThemeResource SystemControlAcrylicElementBrush}">
```

Les effets de mouvement fluides sont également très appréciés dans les interfaces modernes car elles permettent réellement de sublimer l'expérience utilisateur. Les effets de **Parallax** par exemple sont apparus il y a quelques années et s'intègrent maintenant dans la plupart des applications présentant une liste d'éléments avec des images. Le SDK Windows 10 propose un contrôle XAML ParallaxView permettant d'intégrer facilement ce genre de mécanisme.

```
<ParallaxView Source="{x:Bind ForegroundElement}" VerticalShift="50">
<Image x:Name="BackgroundImage" Source="Assets/turntable.png"
Stretch="UniformToFill"/>
</ParallaxView>
```

La documentation pour le Fluent Design se trouve ici : https://fluent.microsoft.com/. Très complète, elle est destinée aux designers car elle présente clairement les concepts avec des schémas, des exemples et les différentes guidelines à suivre. Elle est également destinée aux développeurs car elle présente des bouts de code XAML faits pour intégrer les concepts dans les applications.

L'intégration avec Cortana

Windows 10 intègre nativement le nouvel assistant de Microsoft et expose un certain nombre d'APIs utilisables dans les Universal Windows Platform app. Il est possible par exemple de demander à Cortana d'ouvrir son application à une page en particulier et d'initialiser certaines données de manière automatique, ou encore d'exécuter des calculs et de renvoyer une réponse précise. Pour ce faire, Cortana se base sur un fichier XML qui décrit les commandes reconnaissables et utilisables par Cortana pour interagir avec l'application : c'est le fichier VCD (Voice Command Definition).

```
<?xml version="1.0" encoding="utf-8" ?>
  <VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.2">
    <CommandSet xml:lang="fr-fr" Name="MyTripCommandSet_fr-fr">
      <AppName> Adventure Works </AppName>
      <Example>Montrer mon voyage à Londres </Example>

      <Command Name="showTripToDestination">
        <Example>Montrer mon voyage à Londres</Example>
        <ListenFor RequireAppName="BeforeOrAfterPhrase">
montrer [mon] voyage à {destination}
 </ListenFor>
```

```
<Feedback>Voici les détails du votre voyage à {destination}</Feedback>
<VoiceCommandService Target="MyService"/>
</Command>
    </CommandSet>
  </VoiceCommands>
</xml>
```

Le VCD ci-dessus comporte les caractéristiques suivantes :

- On définit un **CommandSet** nommé « *showTripToDestination* » ;
- La balise **Exemple** permet simplement de mettre une phrase de démonstration qui sera affichée à l'utilisateur lorsqu'il activera Cortana. Cette balise et obligatoire comme balise enfant de **Command** et **CommandSet** ;
- On définit ensuite la balise **ListenFor** qui va demander à Cortana de surveiller la phrase indiquée. Si elle est reconnue par Cortana, alors la commande est activée. La balise possède l'attribut **RequireAppName** pour dire si la commande à besoin du nom de l'application avant de spécifier la commande. Ici, on spécifie '*BeforeOrAfterPhrase*' pour indiquer que le nom de l'application doit être spécifié avant ou après la phrase indiquée dans la balise ;
- La notation [] indique que le mot est optionnel ;
- La notation {} permet de récupérer le mot spécifié par l'utilisateur dans le code plus tard ;
- La balise **Feedback** indique une chaîne de caractères qui sera affichée à l'utilisateur si la commande est activée ;
- Enfin, la commande *VoiceCommandService* indique quelle classe de service sera utilisée lors de l'activation de cette commande. Il est également possible de naviguer vers l'application en utilisant la balise **Navigate** à la place en spécifiant la page. La gestion de l'appel se fait alors dans le **OnActivated** de la page ;

Après avoir défini son VCD, il faut l'enregistrer dans l'application. Cet enregistrement se déroule dans **OnLaunched** de la classe *App* grâce aux deux lignes ci-dessous :

```
StorageFile vcdStorageFile = await Package.Current.InstalledLocation.GetFileAsync(@"MyVCD.xml");
await VoiceCommandDefinitionManager.InstallCommandDefinitionsFromStorageFileAsync(vcdStorageFile);
```

On va tout d'abord chercher le fichier dans le dossier d'installation de l'application, puis on enregistre le fichier.

Dans l'exemple de VCD que nous avons montré plus haut, nous utilisons une classe de service lorsque la commande est activée. Cette classe est simplement un **BackgroundTask** permettant d'exécuter du code de manière asynchrone. Ce code pourrait permettre de faire des calculs, appeler une API … Comme tout **BackgroundTask**, il est important d'enregistrer également la classe dans le manifeste de l'application. 9

Avec ceci, Cortana est alors capable d'exécuter du code contenu dans votre application, et ainsi renvoyer une réponse qui peut prendre la forme d'un texte, d'une liste d'éléments, d'une image … Le code ci-dessous présente comment renvoyer une réponse simple à l'utilisateur :

```
class MyService : IBackgroundTask
  {
    public async void Run(IBackgroundTaskInstance taskInstance)
    {
      Var userMessage = new VoiceCommandUserMessage();

      userMessage.DisplayMessage = "This is my response";
      userMessage.SpokenMessage = "My response";

      var response = VoiceCommandResponse.CreateResponse(userMessage);
      var voiceServiceConnection =
            VoiceCommandServiceConnection.FromAppServiceTriggerDetails(
              taskInstance.TriggerDetails as AppServiceTriggerDetails);


      await voiceServiceConnection.ReportSuccessAsync(response);
    }
  }
```

Les actions effectuées sont les suivantes :

- On instancie un objet de type **VoiceCommandUserMessage**, permettant de définir une réponse simple qui sera compréhensible par Cortana ;
- On définit ensuite les propriétés **DisplayMessage**, qui sera le message affiché par l'utilisateur, et **SpokenMessage** qui sera le message récité par Cortana ;
- On utilise ensuite la classe **VoiceCommandResponse** permettant de récupérer un objet pour interagir avec Cortana et indiquer la progression de la réponse (en cours, réussite ou échec).
- On récupère ensuite un objet **VoiceServiceConnection** pour envoyer la réponse à Cortana via la méthode **ReportSuccessAsync**.

L'API Cortana permet de faire bien plus de chose encore. La documentation complète se trouve ici : https://msdn.microsoft.com/fr-fr/windows/uwp/input-and-devices/interact-with-a-background-app-in-cortana.

Comment monétiser son application ?

Monétiser son application est un vaste sujet demandant à l'application une étroite collaboration avec le Store. En effet, le développeur au sein de son application va utiliser des services sur Store afin de récupérer plusieurs informations telles que :

- Les add-ons disponibles pour l'application ;
- Les achats de l'utilisateur sur l'application ;

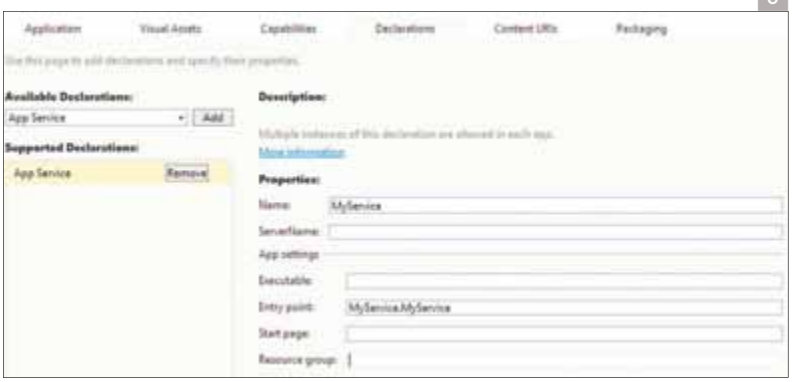
Il faut savoir qu'il existe 2 APIs Windows afin d'interagir avec ces services :

- **Windows.Services.Store** : disponible uniquement depuis la version 1607, cette API est l'API principale afin d'implémenter des add-ons et des achats *in-app* ;
- **Windows.ApplicationModel.Store** : API toujours disponible pour des raisons de rétro-comptabilité avec d'anciennes applications.

Le principe de monétisation avec UWP est simple. Le développeur commence par définir des achats *in-app* dans le Dev Center pour

Enregistrement de la classe en tant que BackgroundTask

9



son application. Il est possible de définir toutes sortes de paramètres pour un achat *in-app* : période d'essai, renouvelable, le prix et bien d'autres. Ensuite, il utilise les APIs afin de savoir si l'utilisateur à fait tel ou tel achat dans l'application lui permettant ainsi de débloquer certaines fonctionnalités de l'application. Chaque add-on défini dans le Dev Center est associé à une licence offrant ainsi plus d'information exploitable par le développeur. Au sein du code C#, le développeur va principalement manipuler un objet de type **StoreContext**. C'est cet objet qui va lui permettre d'interagir avec le Store afin de récupérer les infos d'achats de l'utilisateur.

```
Windows.Services.Store.StoreContext context = StoreContext.GetDefault();
```

Par exemple, afin d'activer un achat *in-app* pour un device, il suffit d'appeler la méthode suivante :

```
StorePurchaseResult result = await context.RequestPurchaseAsync(storeId);
```

Le *storeId* étant l'identifiant unique de l'application dans le Store. Le résultat, sous forme d'objet complexe, permet de réagir suivant la réponse du Store : il est possible, par exemple, que l'achat ait déjà été effectué par l'utilisateur.

```
switch (result.Status)
  {
    case StorePurchaseStatus.AlreadyPurchased:
      textBlock.Text = "Produit déjà acheté.";
      break;

    case StorePurchaseStatus.Succeeded:
      textBlock.Text = "Bravo : produit acheté !";
      break;

    case StorePurchaseStatus.NotPurchased:
      textBlock.Text = "Opération annulé ";
      break;
  }
```

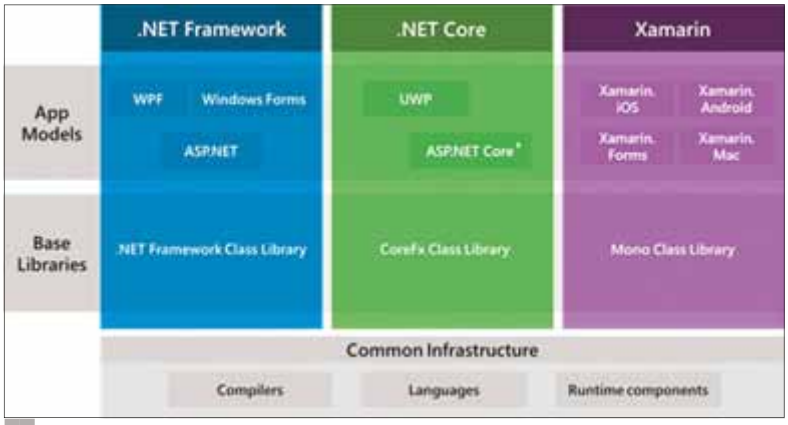
Le développeur est ensuite libre de réagir à tous ces résultats : déblocage de page / d'élément de l'application, ajout de crédit ou autre.

La publication sur le Store

Publier sur le Store est probablement la dernière étape lors de la conception d'une application. Votre app est prête ? Tous les tests sont au vert ? Vous avez votre logo ? Vous êtes fin prêt à publier votre app sur le Store et la diffuser au plus grand nombre.

Tout d'abord, il faut savoir qu'il existe plusieurs types de packages :

- **App Package (.appx)** : package d'application permettant d'installer l'application **sans passer par le Store**. Cela permet notamment d'installer l'application sur des environnements de test où à des utilisateurs en avance de phase. Ce genre de package n'est pas destiné à être publié sur le Dev Center ;
- **App bundle (.appxbundle)** : package contenant plusieurs binaires d'applications selon certaines architectures sélectionnées pendant la génération du paquet ;
- **App Package Upload File (.appxupload)** : package composé également de plusieurs binaires selon les architectures, mais comporte également des symboles afin d'évaluer les performances / crashes une fois l'app publiée sur le Store. C'est ce genre de package qu'il faut envoyer sur le Dev Center. Visual Studio génère par défaut ce type de package.



10 *Séparation des plateformes*

Dans Visual Studio, il suffit de faire clic droit sur le projet -> **Store -> Create App Packages.** Il suffit ensuite de suivre les directives, sélectionner les architectures désirées puis laisser l'assistant vérifier la *check-list* automatique pour l'application (si vous le désirez). N'oubliez pas de remplir le fichier *manifest* avec informations générales de votre projet et les logos !

Une fois fait, il suffit d'envoyer le fichier sur le Dev Center, en remplissant toutes les informations nécessaires à la diffusion de la nouvelle version de votre application, et le tour est joué !

Note

Pensez également à réserver votre nom d'application sur le Store avant.

La place d'UWP dans l'écosystème Microsoft ?

L'écosystème de Microsoft s'enrichit au fur et à mesure des années : .NET Core, Xamarin, Azure sont autant de technologies rattachées à l'entreprise qui ne cessent d'évoluer rapidement. Et UWP dans tout cela ? Quelle est sa place ? A qui et pour quel usage est destinée cette technologie ? Tout d'abord, il faut bien comprendre la séparation entre les plateformes. De plus, il est important de savoir que **UWP fonctionne grâce à .NET Core** ! Les usages sont certes souvent dissociés, mais les deux technologies ont des socles communs. Il ne faut pas l'oublier. 10

Le tableau récapitulatif ci-dessous tente de définir les caractéristiques de chacune des plateformes principales de développement Microsoft :

	.NET Framework	.NET Core	Xamarin	UWP
Usages	Application lourde Web	Web & Cloud	Mobile multiplateforme	Mobile Microsoft
Avantages	Rétro-compatibilité Framework éprouvé	Plus léger Optimisé Cloud	Mutualisation du code Génération native	Optimisé Windows Fonction native
Inconvénients	Gros historique Peu optimisé pour la scalabilité	Migration obligatoire	Développement plus long	Uniquement Windows
Domaine visé	Application legacy	Développeur Web / Cloud	Développeur mobile Android / iOS / Windows	Développeur Windows

Jonathan ANTOINE
Développeur @ Infinite Square
Blog : https://blogs.infinitesquare.com/users/jantoine

Quoi de neuf dans le prochain SDK ?

Microsoft communique maintenant de façon très ouverte sur les prochaines nouveautés du SDK ciblant UWP. Celui-ci est disponible depuis la sortie de la nouvelle mise à jour de Windows 10 en Avril/Mai 2018, disponible sous le numéro « 1803 ».

La liste des évolutions est relativement longue mais voici celles que je préfère :

- Applications console en UWP !
- Applications UWP multi-instances,
- Accès aux mêmes fichiers que l'utilisateur sans Picker !

Applications console en UWP

L'idée surprend au premier abord mais il s'agit bien de créer des applications Console en UWP déployables sur tous les périphériques supportant un Shell/une ligne de commande.

La cible initiale sera **donc dans un premier temps le Desktop et l'IOT** mais rien n'empêchera Microsoft de rajouter le Shell sur les XBox un jour ou l'autre.

D'un point de vue développement, on est sur du .net standard 2.0 et l**a volonté affichée est donc de pouvoir copier-coller le code existant d'une app console et que cela fonctionne tel quel**.

Pourquoi faire une app Console en UWP ? Notamment pour pouvoir bénéficier du déploiement via le Store : plus besoin d'ajouter votre application dans le PATH, elle le sera de base après installation. Il sera aussi possible de mettre des alias dans le manifest de l'application pour la rendre disponible sous un petit nom personnalisé.

Quelles sont les limites ?

- Il n'est pas possible d'utiliser de l'UI (logique en même temps),
- Certains contrats d'activations ne sont pas supportés (fichiers, protocoles, etc.) et c'est bien dommage :(
- Pas de possibilité de faire des tâches de fond.
- Uniquement disponible en C++ pour le moment.

Applications UWP multi-instances

Dans les versions précédentes du SDK, il était déjà possible de créer des applications multi-fenêtrées.

Ce que propose Microsoft est de créer plusieurs processus d'une même application. Là encore, cela ne sera disponible que sur les plateformes IOT et Desktop.

Bien sûr vous aurez le choix et il s'agira de **l'activer au sein de votre manifest avec l'option "SupportsMultipleInstances"**. Il sera **même possible de faire de la *multi-instances* redirection : choisir au lancement si l'on utilise une instance existante** (exemple : vous ouvrez le même document Word) **ou si l'on crée un nouveau processus** (vous ouvrez un nouveau document Word).

Chaque instance de votre application aura un identifiant que vous choisirez vous-même et il sera possible de retrouver une instance existante avec l'APIAppInstance.FindOrRegisterInstanceForKey(key). Ce choix devra être fait au tout début du lancement de votre application : dans la méthode Main ajoutée notamment pour les applications Console. Il y aura un template de disponible pour ce genre d'application car il faut penser à mettre certaines directives de pré-compilation pour gérer correctement ce Main *non classique*.

Il faudra aussi faire **attention à vos Background Task**. Si vous utilisez le modèle originel (Out of Process - une lib à part qui est gérée par l'OS) il n'y aura a priori aucun souci mais le mode InProcess introduit dans l'anniversary update ne sera pas supporté dans ces scenarii.

Accès aux mêmes fichiers que l'utilisateur sans Picker !

Et j'ai envie de dire "enfin" ! De base on était limité à l'accès à quelques emplacements bien restreints dans une application UWP et il fallait demander l'autorisation à l'utilisateur via des Pickers. D'un point de vue sécurité c'était assez génial mais d'un point de vue *liberté de développer* un peu moins, surtout pour des applications B2B.

Avec l'ajout d'une capacité au manifest

Dorénavant, en ajoutant la capacité "**broadFileSystemAccess**" à votre Manifest, **il deviendra possible d'accéder à tous les fichiers et dossiers accessibles par l'utilisateur.** La première utilisation d'une API du namespace "Windows.Storage" demandera quand même son autorisation à l'utilisateur une fois pour toutes. Pour information, il sera **possible d'activer cela automatiquement au niveau entreprise via une group policy**.

Sans ajouter de capacité au manifest

Aussi, **sans même ajouter de capacité à votre manifest, une application lancée depuis une ligne de commande aura accès au dossier depuis lequel elle est lancée**. Il a été décidé par Microsoft que si l'utilisateur s'amuse à aller dans un dossier en ligne de commande pour lancer une application, c'est qu'il veut vraiment donner accès à ce dossier à l'application.

Conclusion

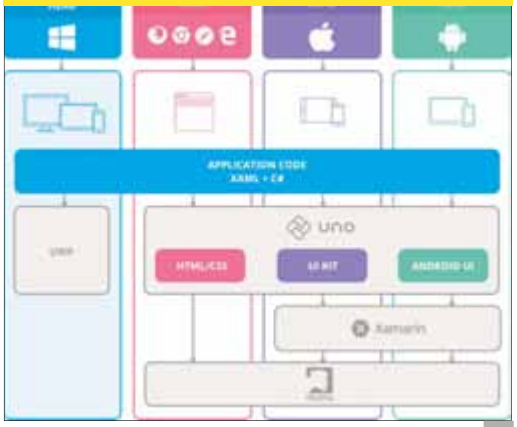
Pour conclure, il est clair que les nouveaux types de projet UWP app et le nouveau SDK ouvre encore un peu plus le champ des possibles pour les développeurs. Multiplateforme et responsive, ces nouvelles applications vont permettre d'accélérer les temps de développement et réduire les maintenances dues à du code dupliqué. L'interactivité avec Cortana apporte également son petit plus à l'application et permet d'accéder à des infos très rapidement sans forcément ouvrir l'application. Les développeurs Windows peuvent maintenant faire parler toute leur imagination autour d'un seul projet fonctionnant à la fois sur PC, tablette et smartphone. •

Jérôme GIACOMINI
Consultant Infinite Square
http://blogs.infinitesquare.com/
@jeromegiacomini

UWP sur iOS et Android : un rêve que j'avais imaginé dès l'annonce du rachat de Xamarin par Microsoft.

Durant la conférence BUILD 2018, ce n'est pas Microsoft mais un petit éditeur canadien qui a annoncé un outil pour UWP, Uno.

Schéma de l'architecture d'Uno :



Source : http://platform.uno 1

Uno permet d'utiliser le code UWP (XAML et C#) sur Android et iOS. Deux applications playground sont disponibles pour voir le rendu :
– Android :
https://play.google.com/store/apps/details?id=com.nventive.uno.ui.dem
– iOS :
https://itunes.apple.com/us/app/uno-gallery/id1380984680?ls=1&mt=8
En bonus, Uno peut aussi créer des application web en générant du code WebAssembly.
Cette fonctionnalité est pour l'instant expérimentale et vous pouvez voir une démo ici :
http://platform.uno/Playground/index.html.
Uno est basé sur Xamarin. C'est un concurrent à Xamarin Forms mais son approche est très différente : il ne réinvente pas un nouveau XAML, il utilise le XAML et les API d'UWP.
Ici le but est de développer son application directement sur UWP, avec toutes les facilités que les outils Microsoft apportent (XAML live reload, C# Edit & Continue …). Et, une fois le développement fait, on teste si les rendus sur téléphones iOS et Android conviennent bien.
Pour vous parler d'Uno, j'ai décidé de poser directement des questions aux développeurs de la plateforme.

Tout d'abord qui êtes-vous ?

Nous sommes nventive : une société montréalaise qui a été fondée à l'origine pour aider les professionnels à mieux développer sur les technologies *.NET*. Nous utilisons les technologies C# et XAML depuis notre fondation. On a d'abord commencé en WPF, ensuite WPF/E (devenu Silverlight), suivi de Windows Phone 7 puis *WinRT*. En parallèle, nous sommes peu à peu devenus une agence de développement d'applications pour la plateforme Windows, tout en conservant nos activités de R&D de développement d'outils et de frameworks.
Par la suite, nos clients nous ont demandé de développer des applications pour d'autres plateformes. Bien sûr, on a fait comme tout le monde : nous avons tenté du développement natif (Objective-C et Java), mais avoir plusieurs équipes qui utilisent des technologies différentes ne nous plaisait pas du tout. Petit à petit, nous avons porté nos outils maisons en utilisant Mono + Xamarin. Au début, uniquement pour la logique applicative tout en conservant une interface utilisateur spécifique par plateforme, mais nous n'étions pas satisfaits des outils existants et ça nous dérangeait d'avoir des équipes spécialisées par plateforme. Nous avons donc commencé à créer une petite librairie de contrôles qui pourraient être créés en C# et dont le code serait presque identique pour toutes les plateformes. À ce moment, notre but était d'abord et avant tout d'être plus efficace et diminuer l'équipe nécessaire pour faire un projet multiplateforme.
Comme nous avons des clients exigeants et que nos designers sont ambitieux, nous cherchions à obtenir quelque chose qui nous permettait un rendu à la fois flexible et identique sur toutes les plateformes. Il devenait de plus en plus évident que le XAML était le langage qui nous permettait d'atteindre ces objectifs, mais pour le faire il fallait non seulement « parser » le XAML, mais également générer du code à la compilation.
Nous avons donc conçu Uno comme une série d'outils « maison » pour adresser notre problématique en attendant que Microsoft nous arrive avec une plateforme vers laquelle nous pourrions migrer. Vous connaissez la suite de l'histoire : aujourd'hui nous ne l'attendons plus, nos outils sont devenus assez efficaces.

Qu'est-ce que c'est Uno ?

C'est une série d'outils assemblés dans le but de reproduire le plus fidèlement possible le comportement de l'interface graphique de Windows UWP XAML sur les autres plateformes tout en bénéficiant des particularités de chacune des plateformes. Ceci de sorte que le code conçu pour UWP fonctionne de manière presque identique sur les autres plateformes. Pour convertir une application UWP à utiliser Uno, il suffit de créer les projets iOS, Android et WebAssembly dans la solution et déplacer le code UWP vers un « shared project » pour qu'il soit recompilé sur chacune des plateformes. Ça se fait en quelques minutes.

Comment ça fonctionne ? Utilisez-vous Xamarin Native pour créer les contrôles natifs ?

Nous avons recréé la totalité des APIs de UWP de Windows (version 14393 – Creators Update) et implémenté une partie de ces APIs en fonction de nos besoins sur iOS, Android et WebAssembly. En ce qui concerne les contrôles UWP, tous les éléments visuels (Panels, Buttons, etc..) héritent des classes de primitives de chacune des plateformes afin de s'afficher, tout en adaptant leurs propriétés et les phases de *measure* et *arrange* pour se conformer au contrat implicite de UWP. 1
Ensuite il y a un générateur de code qui s'exécute à la compilation d'un projet qui utilise Uno et qui prend les fichiers Xaml, les *parse* et génère le code d'initialisation. Pour ça nous sommes basés sur un composant que nous avons rendu public plus tôt cette année, à savoir le « Uno.SourceGeneration » (https://github.com/nventive/Uno.SourceGeneration). Prenons le code XAML suivant :

```
1 <StackPanel>
2   <TextBlock Text="Hello world!" Foreground="{StaticResource
 MyColor}" />
3 </StackPanel>
```

Le code généré sera :

```
1 new StackPanel
2 {
3  Children =
4  {
5   new TextBlock
```

```
6  {
7      Text = "Hello world!",
8      Foreground = Application.Resources["MyColor"] as Brush,
9  }
10 }
11}
```

Bien sûr, il y a également un mode dynamique qui permet de parser du code pendant l'exécution à l'aide du XamlReader (https://docs.microsoft.com/en-us/uwp/api/Windows.UI.Xaml.Markup.XamlReader). Nous avons donc fait notre implémentation de ce composant qui permet de créer dynamiquement les contrôles pendant l'exécution. C'est d'ailleurs sur ce composant qu'est construit le Uno Playground.

Si je veux utiliser la suite Telerik pour UWP ou le UWP control toolkit sur Uno comment puis-je faire ?

Ça fait un moment que nous avons porté certains contrôles de la suite Telerik avec Uno. Un « fork » du repository de Telerik devrait être rendu public prochainement détaillant la solution.

D'une manière plus générale, la surface d'API de UWP est complètement définie, permettant à n'importe quelle librairie existante d'être compilée presque inchangée sur Uno. La surface d'API de UWP étant particulièrement grande, il se peut que ces librairies s'appuient sur des APIs non implémentées dans Uno. Dans ce cas, il est possible d'utiliser directement la plateforme sous-jacente.

Comment êtes-vous arrivé à rendre UWP utilisable sur iOS et Android ?

De manière itérative, au fur et à mesure que nous en avions besoin pour le développement de nos applications nous avons tenté d'implémenter les API de UWP et de les adapter à la plateforme.

Est-ce possible d'utiliser les rendus natifs de chacune des plateformes ?

Uno supporte la création des contrôles natifs directement dans le XAML au travers du *Control Templating*, donnant les rendus natifs. Cependant, cette approche réduit la possibilité d'avoir un rendu identique (pixel perfect) très souvent recherché par les designers.

Uno a été construit sur la possibilité d'avoir un mode hybride (xaml vs. natif). Nous avons utilisé cette approche à plusieurs occasions pour utiliser des contrôles disponibles publiquement sans avoir à les réimplémenter. Nous l'avons utilisé pour des contrôles natifs de lecture de vidéo, des carrousels, de la cartographie et pour intégrer « Lottie » dans certains projets.

Avez-vous des exemples d'application fonctionnant avec Uno ?

Pour le moment, nous avons l'application Gallery (qui contient le "Uno Playground") qui est publique, mais nous ne pouvons pas encore dévoiler le nom des applications de nos clients pour des raisons de confidentialité.

Avez-vous une roadmap publique ?

Elle n'est pas formalisée encore, car nous voulons jauger la réaction de la communauté pour orienter notre développement.

Combien d'applications avez-vous créé avec Uno ?

Plus de 150.

Combien de temps avez-vous mis à créer Uno ?

Plusieurs années.

Combien de personne travaillent sur Uno actuellement ?

Une dizaine.

Combien de personne l'utilisent en interne ?

Plus de 75.

Avez-vous été aidé par les équipes de Microsoft pour mettre en place ce Framework ?

Non.

Comment partage-t-on le code entre les différents OS ?

Le plus simple est de faire un « Shared Project » dans Visual Studio et de l'inclure dans toutes les plateformes. Si c'est un composant, il y a moyen d'utiliser les nouvelles fonctionnalités des projets « dotnet core » qui compilent en simultané vers différentes plateformes.

Nous utilisons déjà cette approche dans le project Uno.Core (https://github.com/nventive/Uno.Core/blob/master/src/Uno.Core/Uno.Core.csproj) qui cible à la fois les plateformes « UAP10.0 », « net46 » et « netstandard2.0 ».

Le code de Uno (https://www.nuget.org/packages/Uno.UI) a également ces plateformes comme cibles supplémentaires : « MonoAndroid6.0 », « MonoAndroid7.0 », « MonoAndroid7.1 », « MonoAndroid8.0 » et « Xamarin.iOS1.0 ».

Avez-vous prévu qu'on puisse partager son code via un projet .net standard ?

Non, nous préférons l'approche à base de partage de code source et de classes partielles. Partager les binaires donne tendance à compliquer inutilement le développement de code utilisant les particularités de la plateforme.

Si je veux utiliser un contrôle natif d'une plateforme qui n'existe pas en XAML ? Y a-t-il un moyen ?

Il suffit de l'instancier dans le Xaml. Il y a des moyens d'utiliser des alias de *namespace* pour orienter la génération de code de manière différente selon la plateforme. Il s'agit de XAML conditionnel par plateforme.

Dans le code qui a été publié, vous pouvez voir qu'il y a des préfixes de namespaces du type "<android:XXX>" ou "<ios:XXX>".

Allez-vous le rendre Open Source ? Avez-vous besoin d'aide ?

Oui nous sommes en train de rendre open source la totalité d'Uno.

Le travail nécessaire pour adapter les différents aspects de la plateforme est colossal. Il y a beaucoup de contrôles, de fonctionnalités qui ne sont pas encore supportées et nous sommes encore en train de chercher une manière qui permettrait à des tiers de fournir des implémentations spécifiques des APIs de UWP. Par exemple, si vous voulez utiliser les APIs pour faire du NFC, ce n'est pas implémenté car nous n'en avons pas eu besoin. Nous voulons que ce soit possible pour des tiers d'en fournir une implémentation.

Conclusion

Cette annonce a créé un mini débat dans la communauté Xamarin : faut-il vraiment un second framework graphique pour Xamarin ? Pour moi Uno semble beaucoup plus simple que Xamarin Forms, qui demande la création de nombreux renderer. Or avec Uno, si le visuel d'un contrôle ne convient pas, on peut changer son template comme sur UWP. On est aussi beaucoup plus proche du rendu : la notion de "Pixel Perfect" – avoir le même rendu au pixel près sur les 3 plateformes – est aussi très intéressante.

Les équipes de Uno ont l'air ultra motivées pour populariser leur outil, les développeurs ayant même pris le temps de répondre à mes questions. Merci encore à l'équipe d'Uno et tout spécialement à Carl qui a pris du temps pour répondre à mes questions. Personnellement j'ai hâte de tester de plus près cet outil sur une application plus conséquente. Je ne manquerai pas de vous partager mes découvertes. Happy Coding

Pour aller plus loin :

Le site officiel de Uno : https://github.com/nventive/Uno

La démo WASM (uniquement PC en alpha à l'heure où j'écris ces lignes) :
http://platform.uno/Playground/index.html

Le lien vers le Playground : https://github.com/nventive/Uno.Playground

François Tonic

Xamarin : les nouveautés et les évolutions de la plateforme

Pour en savoir plus sur Xamarin et son évolution depuis le rachat par Microsoft, nous avons posé quelques questions à Christopher Maneu, premier field engineer chez Microsoft France. Christopher est développeur et expert en développement mobile.

Que s'est-il passé pour Xamarin depuis son rachat ?

Pas mal de choses ! L'acquisition a été finalisée le 18 Mars 2016, il y a un peu plus de 2 ans. Il y a eu quelques changements immédiats. Tout d'abord, Xamarin a été inclus dans toutes les éditions de Visual Studio, y compris la version Community. On a tendance à l'oublier, mais Xamarin était jusque-là payant. Une version « d'essai » existait, mais à usage limité en nombre de lignes de code. La version Visual Studio Community permet maintenant de développer des applications, voire de les revendre, dans certaines limites (développeur indie, petite entreprise, … voir https://www.visualstudio.com/license-terms/mlt553321). Certaines parties propriétaires de Mono ainsi que Xamarin ont été publiées en open source. Voici pour les changements rapides.

D'autres changements sont plus longs. Xamarin Test Cloud – La solution des tests UI automatisés – a mis du temps à être intégrée à une équipe produit. C'est le cas désormais avec Visual Studio App Center. Xamarin Insights a dû trouver sa place entre Application Insights et HockeyApp.

Derrière les projets, ce sont aussi des humains. Il a fallu également intégrer les équipes Xamarin dans l'organisation Microsoft, ce qui peut prendre du temps. On peut noter que Nat Friedman est désormais *VP Developer Services* (ce qui est plus large que l'organisation Xamarin), et Miguel de Icaza est *Distinguished Engineer*, et continue de travailler activement sur Xamarin et .Net.

Microsoft est discret sur Xamarin et quand on regarde l'usage de Xamarin dans les développements mobiles, la plateforme est régulièrement derrière React, Cordova / PhoneGap. Comment aujourd'hui Xamarin est-il promotionné par Microsoft ?

Cette question me rappelle un autre débat lors de la sortie de Windows 8 : « Est-ce que Microsoft abandonne XAML/C# au profit de WinJS ? ». Après plusieurs années, il me paraît évident que XAML n'a jamais été abandonné ! L'actualité autour des technologies de développement Microsoft est très riche. Il est vrai qu'une grande partie de la communication est orientée autour de *l'intelligent cloud*, et des services Azure en particulier. Côté Xamarin, la communication du moment est plus orientée sur .Net Standard et .Net Core que sur Xamarin en lui-même. Nous avons également la chance d'avoir une communauté d'enthousiastes, de MVP (Most Valuable Professional) et de partenaires Microsoft qui participent à leur manière à la promotion et à l'adoption de Xamarin. On peut noter dans ce registre deux livres sortis depuis début 2018 – *Xamarin Unleashed* de Alec Tucker et *Xamarin.Forms essentials* de Gerald Versluis – et d'autres sont en cours d'écriture en ce moment même.

Pour React Native, tous les chiffres que j'ai pu voir (notamment le dernier rapport d'AppFigures) montrent que React Native est encore nettement derrière Xamarin. Il y a un certain écho sur ce projet, notamment sur les réseaux sociaux, mais qui ne se traduit pas aujourd'hui par une réalité des développements. Le monde des solutions de développement cross-plateformes est riche et en pleine évolution. React Native partage une certaine vision avec Xamarin : l'UI doit être rendue avec les composants natifs de la plateforme cible, ce qui, de mon opinion personnelle, est la bonne approche. Cordova/PhoneGap a en effet une part de marché importante dans les développements cross-plateformes. Cependant il ne cible pas les mêmes applications – et souvent pas les mêmes développeurs.

Un développeur qui voudrait démarrer avec Xamarin, comment faut-il faire ?

Tout dépend des connaissances du développeur. Connaît-il déjà C# et .Net ? A-t-il déjà fait du développement mobile natif (Java ou Kotlin sur Android, Objective-C ou Swift sur iOS) ?

Si vous êtes déjà développeur C#, je vous recommanderais de suivre les tutoriels et vidéos de la section *Getting Started* de la documentation officielle (https://docs.microsoft.com/xamarin). Dans un premier temps, je conseillerais de partir sur Xamarin.Forms – en particulier si vous avez déjà fait du WPF ou de l'UWP. Cela vous permet de vous concentrer sur l'outillage et une partie des contraintes du développement mobile, et de déférer la découverte des spécificités des plateformes mobiles.

Si vous êtes développeur mobile natif, vous partez avec un avantage !

Vous allez vous retrouver comme à la maison avec Xamarin Platform : activités, Layout XML pour Android, Contrôleurs et Interface Builder pour iOS. Votre plus gros travail sera donc d'apprendre le C# (https://docs.microsoft.com/dotnet/csharp/quick-starts/) et de vous familiariser avec l'environnement Visual Studio. Les guides *Getting started* pour Xamarin « Platform » (Android ou iOS) vous aideront à démarrer vos premiers projets.

Comment débuter dans le développement Xamarin peut également dépendre de ses moyens (ou des moyens de son entreprise). Si vous faites partie d'une entreprise avec un projet de développement, il peut être intéressant de s'orienter vers une formation avec un accompagnement. Xamarin University (https://university.xamarin.com) propose

une formation en ligne payante – avec des professeurs qui donnent cours en direct et avec qui on peut interagir. Microsoft France propose des formations inter-entreprise ou intra-entreprise payantes. Enfin, il existe également des partenaires Xamarin certifiés pour délivrer des formations, dont plusieurs en France.
Enfin, quelle que soit la technologie à apprendre, je ne peux que conseiller de se lancer dans le développement d'un petit projet. Aucun tutoriel ou formation en ligne ne remplace l'expérience de réaliser toutes les étapes du développement et du déploiement de A à Z.

Quelles sont les ressources disponibles pour les développeurs français/francophones ?
Il faut bien avouer que la majorité des ressources sont disponibles en Anglais, et non en français. Pour commencer avec Xamarin, plusieurs livres existent en français, écrits par les MVPs. Leurs blogs peuvent également être une bonne ressource d'information.
Le groupe Facebook *Xamarin Communauté des développeurs francophones* (https://www.facebook.com/groups/xamarindevfrance/) est probablement l'une des meilleures ressources en langue française. Beaucoup d'experts suivent attentivement les questions et y répondent. Il faut toutefois penser un peu à eux et faire vos devoirs avant de poser une question (comme rechercher si la réponse n'est pas dans la documentation ou sur Internet, et donner des informations précises et détaillées).

Quels sont aujourd'hui les développements types avec Xamarin ?
On constate beaucoup plus d'applications business ou internes aux entreprises que d'applications grand Public réalisées avec Xamarin, surtout en France. Je pense que cela est principalement dû à une disponibilité des développeurs C#, qui sont plus présents dans les entreprises que dans les startups. Cela est moins vrai outre-atlantique.
Dans les grands groupes, je constate une tendance – ou tout du moins un intérêt - pour les *centres de compétences* autour de Xamarin. Ces sociétés choisissent Xamarin pour l'ensemble de leurs développements mobiles, et développent des équipes spécialisées avec l'infrastructure nécessaire.

UWP, Xamarin : comment comprendre le positionnement et le rôle de chaque modèle ?
Le principe est relativement simple : UWP est le framework de développement de choix pour les plateformes Windows. Il est « universel » dans le sens où il s'adapte à l'ensemble des form factors de Windows : PC, Tablette, devices IoT, Hololens, Surface Hub, Xbox, … même si le développement mobile sous Windows est moins d'actualité.

Est-ce que XAML standard est abandonné ?
L'objectif de XAML standard est d'uniformiser le nommage d'éléments ou de propriétés à travers toutes les *versions* de XAML (WPF, Xamarin Forms, UWP, …). Cette uniformisation prend du temps, et pose des questions complexes de compatibilité. Je n'ai pas personnellement d'autres informations que celles qui sont sur le GitHub.

Vous avez beaucoup mis en avant Xamarin Forms, pourquoi ?
Xamarin *Platform* permet un grand partage de code entre les plateformes mobiles. Cependant, il nécessite de redévelopper l'interface utilisateur pour chacune des plateformes cibles. Cela nécessite non seulement des compétences spécifiques par plateforme, mais également un effort de développement important.
Xamarin Forms permet de développer votre interface graphique une seule fois, et avec un langage unique entre plateformes (XAML). C'est une forme d'aboutissement de la promesse d'un développement cross-plateformes.
Pour de nombreux scénarios d'applications mobiles, Xamarin Forms répond parfaitement aux exigences métiers, et permet de maximiser le partage de code. Ce partage peut avoir de nombreuses incidences à la fois sur la phase de développement initial, mais également sur la maintenance d'un applicatif. La version 3.0, qui vient d'être annoncée lors de la Build, inclut de nombreuses nouveautés pour améliorer encore la productivité des développeurs – telles que l'arrivée du *Visual State Manager* ou du support d'un format CSS pour le style de vos applications.
Toutes ces raisons – la maximisation du partage de code, les facilités de développement, les nouveautés – orientent naturellement la communication vers Xamarin Forms. Cependant, tous les projets de développement mobile ne sont pas faits pour Xamarin Forms. Si vous ne ciblez qu'une plateforme mobile, ou que vous avez un besoin de maîtrise fine de l'interface utilisateur, Xamarin Forms n'est peut-être pas l'option à privilégier.
Heureusement, ce choix n'est pas définitif. Il est tout à fait possible de mixer à la fois Xamarin *platform* et Xamarin Forms au sein de la même application – voire même d'héberger Xamarin Forms au sein d'une application native. L'effort pour passer de l'un à l'autre dépendra de l'architecture de votre application.

Pourquoi avoir migré la documentation Xamarin sur la documentation Microsoft ?
La documentation d'un produit est l'un des éléments de qualité, d'adoption et de satisfaction les plus importants. Jusqu'à présent, la documentation des produits Microsoft était hébergée sur de nombreuses plateformes différentes (MSDN, TechNet, etc.).
Depuis quelques mois, l'ensemble des équipes produits sont en train de migrer leur documentation produit sur une plateforme unique : docs.microsoft.com. Celle-ci a de nombreux avantages. Tout d'abord, l'ensemble des sources de cette documentation est open source, hébergée sur GitHub. La communauté est donc en mesure de soumettre des modifications à la documentation, telles qu'un exemple de code ou une correction. Enfin, il est également possible – sur certaines pages pour l'instant – de créer une remontée de bug ou une idée, elle aussi hébergée sur GitHub.
Tous ces outils doivent nous permettre de maintenir une qualité dans la documentation, et aux développeurs de la communauté d'aider à créer la documentation dont ils ont besoin.

Comment Xamarin s'intègre ou s'intègrera avec les Web Assembly ?
Pour WebAssembly, il faut différencier deux choses : exécuter du code .Net dans WebAssembly, et avoir un framework UI qui tourne en WebAssembly.
La première partie a déjà été réalisée avec


Jérôme GIACOMINI
Jonathan ANTOINE

Consultants Infinite Square
http://blogs.infinitesquare.com/
@jeromegiacomini
@jmix90

Quelles nouveautés chez Xamarin ?

Avec plus de 7 années d'existence, Xamarin n'a pas besoin de présentation. Créé en 2011 et racheté par Microsoft en 2016, l'outil ne cesse de s'améliorer. L'arrivée de .Net Standard 2.0 a permis à Xamarin, qui est compatible avec, d'avoir accès à une kyrielle de bibliothèques. De plus, de nombreuses annonces ont été faites sur l'évolution de Xamarin lors de la Build 2018. Revenons dans cet article sur les principales nouveautés de la plateforme.

l'équipe Mono. Ce n'est qu'une preview technique, mais c'est déjà là. Pour la seconde partie, il existe plusieurs solutions. A ce jour, Blazor (https://github.com/aspnet/Blazor) est le projet expérimental le plus avancé sur la question. Il est basé sur Razor – utilisé dans le framework ASP.net MVC, et donc plus proche du développement web que Mobile.

La communauté a proposé une alternative avec la librairie Ooui (https://github.com/praeclarum/Ooui). Ce projet permet d'utiliser certains contrôles Xamarin Forms dans un projet qui target WebAssembly.

Pour le moment, aucun projet officiel permet d'utiliser .Net avec Web Assembly. Est-ce que cela va changer dans le futur ? C'est une excellente question :). Je ne peux qu'encourager les lecteurs à *mettre des étoiles ("starrer") sur* les projets par lesquels ils sont intéressés ou à voter sur les issues Github pour montrer leur intérêt.

Une des questions qui revient à chaque mise à jour, c'est « qu'est-ce qui a été cassé » ?

Je vois parfaitement de quoi on parle ! C'est une question que j'ai abordée de nombreuses fois avec mes clients et avec l'équipe produit. Force est de constater que les moyens de le savoir ne sont pas nombreux, ou exhaustifs. En effet, des éléments dépréciés ou des régressions peuvent être liées aux SDK natifs, aux versions des outils (Visual Studio, XCode), ou bien à Xamarin lui-même. Chacun de ces composants ayant son propre cycle produit, prédire ce qui va casser, où et quand est un véritable art.

Le moyen le plus efficace que j'ai aujourd'hui à proposer – même si sa mise en place peut nécessiter des moyens – est de passer par de l'intégration continue et des serveurs de builds. En testant les nouvelles versions de SDK natifs et Xamarin sur un serveur de builds dédié, on peut plus facilement détecter les changements entre deux versions. Je connais plusieurs équipes – et même un développeur indépendant – qui ont la configuration suivante : trois Mac Mini, chacun avec un channel Xamarin différent (alpha, beta, stable). Chaque build est lancée en même temps sur les 3 Mac. C'est un processus lourd, mais aussi probablement celui permettant d'avoir l'information la plus pertinente au plus tôt. •

Essentials : Une bibliothèque pour les dominer toutes

Xamarin va centraliser dans une seule bibliothèque toutes les API multiplateformes essentielles pour les applications mobiles.

Par exemple : l'accès à la batterie, le presse papier, la connectivité, les informations sur le device... Le tout sera réuni en une seule et unique bibliothèque Essentials.

Le lien vers le nuget :
https://www.nuget.org/packages/Xamarin.Essentials
Le lien vers le Github : https://github.com/xamarin/Essentials

Xamarin Android : plus besoin de FindById

Avant, lorsque vous déclariez un bouton en XML, pour récupérer son instance en C# il fallait :

```
<Button android:id="@+id/mybutton"/>
Button mybutton= FindById<Button>(Resource.Id.myButton);
mybutton.Click += delegate { mybutton.Text="<3 Xamarin"};
```

Avec la nouvelle version de Xamarin, plus besoin de faire ça. La variable *myButton* est automatiquement créée. Le nom de la variable sera l'id utilisée dans le XML.

```
myButton.Click += delegate { myButton.Text="<3 Xamarin"};
```

Xamarin Android : Support de l'émulateur Hyper V

Les équipes de Xamarin n'en finissent plus d'optimiser l'émulateur Android.

Au début d'année, ils avaient rajouté le « Quick boot » qui permettait de démarrer l'émulateur en moins de 6 secondes.

Miguel de Icaza a annoncé que le nouvel émulateur Android sera compatible avec Hyper V.

Celui-ci nous a fait deux petits schémas pour nous faire comprendre le gain à passer par un emulateur Hyper V.

Ancien mode fonctionnement :

Nouveau mode de fonctionnement :

Xamarin iOS : les bindings c'est automatique !

Création automatique des bindings C# sur les bibliothèques swift grâce à l'outil Swift-o-Matic. Cet outil sera disponible début avril.

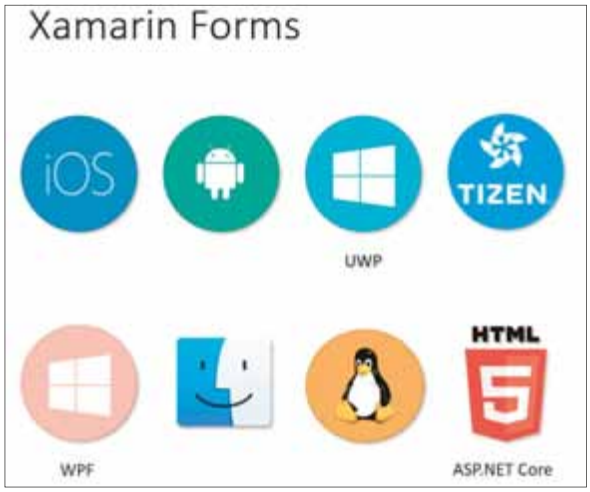
WeakAttribute

Désormais on peut utiliser un attribut sur les champs pour préciser que ceux-ci ne bloquent pas le garbage collect.

```
[Weak] _monChamp
```

Embeddinator-4000

L'outil permet de transformer du code .NET en bibliothèques natives.



Le support de tous les OS de Xamarin Forms

Désormais Forms supporte tous les OS desktop du marché avec l'ajout du support de Mac, Linux et WPF (Windows). Xamarin a aussi travaillé d'arrache-pied avec Samsung pour supporter Tizen.

Mais est-ce que tout est stable ?

iOS, Android, UWP sont supportés par la plateforme et supportent la dernière version de Xamarin Forms (3.0). Le support de WPF est en cours d'écriture; il n'est pas considéré comme stable actuellement. Le support de Tizen est stable, mais on peut noter qu'il n'est pas encore sorti en 3.0. Un projet Open source Ooui permet de aussi de créer un site web en Xamarin Forms. C'est clairement un POC.

Xamarin Forms 3.0

David Ortineau, PM de Xamarin Forms, nous a annoncé la sortie de la version stable de Xamarin Forms 3.0, tant attendue par les développeurs.

Les grosses nouveautés sont :

- Ajout du VisualStateManager cher aux développeurs XAML ;
- Ajout du FlexLayout ;
- Ajout du CSS ;
- Un gros travail d'optimisation des composants de XF a été fait sur cette version ;
- Native Forms : la possibilité d'intégrer des composants Xamarin Forms dans n'importe quel projet Natif ;

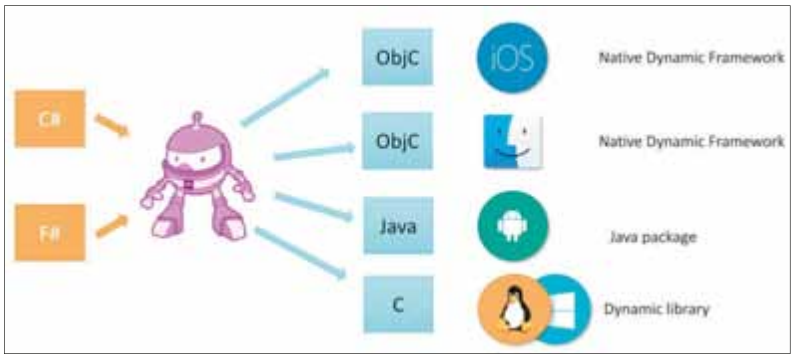
Xamarin Forms et la communauté

« The F100 collaboration » se rapporte aux 100 petites choses à améliorer dans Xamarin Forms. Les équipes de Xamarin.Forms ont travaillé main dans la main avec la communauté pour améliorer le framework. La version 3.0 de Xamarin Forms est la première à intégrer autant de fonctionnalités proposées par la communauté.

Les promesses de Xamarin Forms 3.1

David a aussi annoncé que Xaml Standard sera intégré a la version 3.1 de Xamarin Forms.

Le x:Bind sera lui aussi de la partie. Celui-ci peut déjà être testé dans les versions préliminaires de Xamarin Forms 3.1 qui sont déjà disponibles. David a annoncé que Xamarin Forms 3.1 sera disponible dans les 6 prochaines semaines. En somme, on voit que les équipes sont très impliquées pour faire évoluer le Xamarin. Xamarin Forms a beaucoup évolué et est maintenant un outil mûr pour les développements d'applications modernes. Happy coding with Xamarin •



Combiner Xamarin.Forms et Xamarin.Essentiels dans un projet multiplateforme (BUILD 2018) :

```
using Xamarin.Forms;
using Xamarin.Essentials;

namespace MyCompass
{
  public partial class MainPage : ContentPage
  {
    public MainPage()
    {
      InitializeComponent();

      // Register for reading changes
      Compass.ReadingChanged += Compass_ReadingChanged;
    }

    void Compass_ReadingChanged(CompassChangedEventArgs e)
    {
      LabelInfo.Text = $"Heading: {e.Reading.HeadingMagneticNorth}";
      ImageArrow.Rotation = e.Reading.HeadingMagneticNorth;
    }

    protected override void OnAppearing()
    {
      base.OnAppearing();
      Compass.Start(SensorSpeed.Ui);
    }

    protected override void OnDisappearing()
    {
      base.OnDisappearing();
      Compass.Stop();
    }
  }
}
```

Stéphane Cordonnier
Mobile Tech Lead @ SeLoger

SeLoger : le choix de Xamarin, pour les meilleures et pour les pires

Les applications mobiles du groupe SeLoger étaient historiquement développées en externe (SSII, agence web…) pour les mêmes raisons que beaucoup de sociétés : manque de temps, pas de compétences internes, etc.

Malgré les avantages que représente cette solution, elle comporte aussi de nombreux inconvénients (maîtrise des coûts, maîtrise technique, qualité,…). Début 2016, tous ceux-ci ont mené SeLoger à internaliser les compétences en créant un pôle de développement mobile.

Avant de prendre cette décision, il a fallu choisir une plateforme technique (natif, hybride, web,…). Après avoir étudié plusieurs solutions, le choix s'est porté sur Xamarin. Parmi les arguments ayant fait pencher la balance, on citera les nombreuses compétences .NET déjà existantes, et la capacité à pouvoir partager du code entre plateformes (iOS/Android).

Constitution d'une équipe de développement

Une fois la décision prise de créer un pôle de développement interne, le premier challenge a été de trouver les compétences humaines car même si désormais Xamarin est connu (au moins de nom) par de nombreux développeurs, début 2016 la situation était différente.

Qu'il s'agisse d'embauches ou de prestataires, les développeurs Xamarin sont une denrée rare, notamment lorsque l'on souhaite des profils immédiatement opérationnels. D'une manière générale, quand on parle de choses rares, leur prix est immédiatement en rapport avec celles-ci.

Les salaires souhaités par les développeurs et/ou les TJM pratiqués par certaines sociétés de services ont été autant d'obstacles à la constitution rapide de l'équipe. Il aura fallu compter 5 mois pour constituer le premier noyau de 4 développeurs nécessaire au développement des premières applications.

Développement des premières applications

Les premières applications, dont les développements avaient été démarrés en externe, ont pu être reprises en interne. Malheureusement du fait du peu de développeurs sur le marché ayant une réelle expérience sur la plateforme, la reprise a été douloureuse.

Les patterns de développement et les composants utilisés à l'époque n'étaient pas nécessairement mauvais (ex : MvvmCross et nombre de ses plugins) mais comme toute technologie, mise entre les mains de personnes peu expérimentées, cela peut s'avérer très problématique à court/moyen terme. Un développeur expérimenté vous dira qu'une des phases cruciales dans le cycle de développement concerne l'architecture originelle de l'application, comme le sont les fondations d'une maison. Partir sur de mauvaises fondations fait que tout l'édifice sera en danger à un moment donné.

C'est malheureusement ce qui est arrivé lors de la sortie sur les stores fin 2016, malgré un gros travail effectué pour tenter de consolider les bases sans devoir tout rebâtir. Crashs, bugs ou problèmes de performances ont été les points en souffrance mis en exergue par les utilisateurs sur les premiers commentaires publiés sur les stores.

Les enseignements tirés de cette première année d'utilisation de Xamarin ont été mitigés. Les raisons du choix de la technologie étaient bel et bien au rendez-vous avec une facilité de partage de code entre plateformes, permettant de sortir de manière quasi simultanée iOS et Android. Mais a contrario, un bon développeur .NET ne fait pas nécessairement un bon développeur mobile s'il ne connaît pas parfaitement la plateforme sur laquelle il travaille (iOS et/ou Android).

Stabilisation et amélioration de l'existant

Une fois les premières applications en ligne et au regard des retours négatifs, non dus à la technologie mais à l'utilisation qui en avait été faite, les mois suivants furent passés à tenter de stabiliser et d'améliorer l'existant. C'est à ce moment que surgirent les problèmes liés à la technologie cette fois.

Xamarin est une technologie vivante et en constante évolution. De nombreuses mises à jour sont publiées par Microsoft. Elles apportent des nouveautés, aussi bien sur le plan purement technique (ex : support des nouvelles fonctionnalités iOS et Android) que sur l'environnement de travail du développeur.

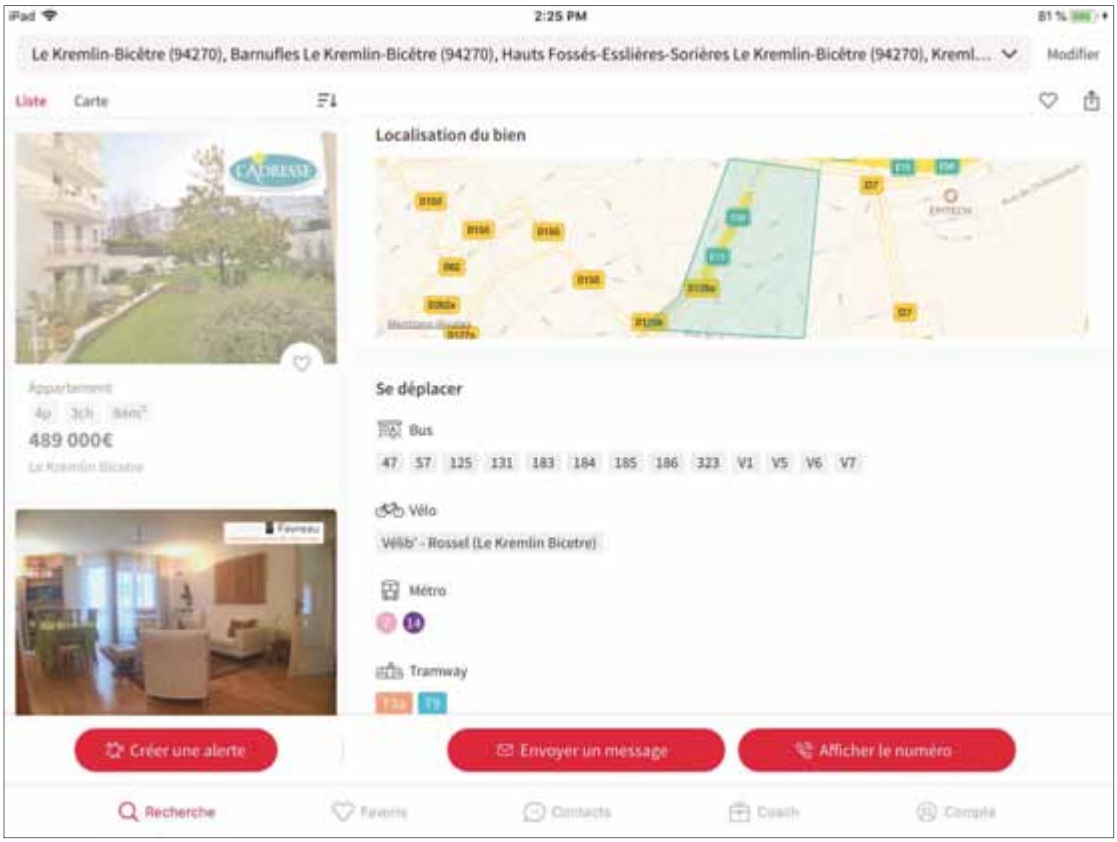
Mais ces évolutions régulières apportent également parfois leur lot de problèmes. Qu'il s'agisse de bugs dans l'environnement de développement qui impactent la productivité quotidienne du développeur, des régressions dans le fonctionnement des APIs de chaque plateforme, voire l'impossibilité d'utiliser certaines fonctionnalités. Les sujets sont divers et variés et peuvent amener à faire perdre tout le temps gagné sur le partage de code.

Un exemple d'inconvénient qui n'a jamais réussi à être totalement résolu, concerne l'utilisation de SDKs tiers. Leur utilisation ne pose généralement pas de soucis, mais l'utilisation de certaines options de compilation peuvent amener une application à ne plus fonctionner.

Vous devez donc vous résoudre à ne pas utiliser ces options qui font que dans notre exemple, le poids d'une application a été multiplié par 5 par rapport à l'époque où elle était écrite en Objective-C / Java, passant de 35 Mo à près de 170 Mo.

Après plusieurs mois ayant débouché sur de

multiples soucis tels que ceux évoqués ci-dessus, les applications ont toutefois été stabilisées et améliorées. Mais une application est un produit qui vit en permanence.



Différenciation des cycles de vie iOS / Android

Les technologies cross-platform telles que Xamarin communiquent énormément sur la force que représente le partage de code, et le gain de temps (et d'argent) que cela représente pour être à même de publier simultanément une version iOS et une version Android.

Lorsque vous réalisez des applications identiques en tous points (ex : des applications internes à une entreprise) et que vous ne souhaitez pas tirer parti de fonctionnalités avancées du système d'exploitation, alors cet argument est indéniablement un énorme avantage.

En revanche, lorsque vous souhaitez avoir des spécificités dans vos applications entre iOS et Android, qu'il s'agisse d'ergonomie, de fonctionnalités, d'A/B Test propre à une plateforme, etc., alors le partage de code perd un peu de son intérêt, voire amène de nouvelles problématiques.

C'est notamment ce qui s'est passé chez SeLoger mi-2017 où ce genre de besoins a commencé à émerger, lors de l'établissement de la roadmap produit pour les mois à venir et en se projetant à plus long terme sur la voie devant être prise pour les années à venir.

Retour aux technologies natives

Après plusieurs semaines de discussions autour des multiples difficultés évoquées précédemment (recrutement, bugs, productivité, performance…) et de la voie à suivre dans le futur, la décision a été prise en Août 2017 d'abandonner partiellement Xamarin.

Les applications ayant été développées sur Xamarin et pour lesquelles ce choix a été pertinent (applications à faible audience et avec très peu voire aucune différence entre plateformes) resteront telles quelles, seront maintenues et évolueront au fil des prochains mois. En revanche pour les applications à plus forte visibilité, devant tirer pleinement parti des spécificités de chaque plateforme et pour lesquelles les cycles de vie iOS et Android ont de grandes chances de diverger dans le futur, le choix a été fait de revenir sur les technologies et langages natif : Swift pour iOS / Kotlin pour Android.

Le choix de revenir en arrière n'est pas une chose anodine car après avoir utilisé Xamarin pendant près de 2 ans, cela a signifié devoir réécrire en intégralité l'application SeLoger. Et bien évidemment, cela ne se fait pas du jour au lendemain.

Cela a débuté début Septembre 2017 par la réécriture de l'application iOS, utilisée par 2/3 des utilisateurs, et pour laquelle il a fallu près de 5 mois de travail. Il a été profité au passage du changement de technologie, d'effectuer une refonte ergonomique et un enrichissement fonctionnel de celle-ci.

L'application Android pour sa part a été démarrée plus tard avec une sortie prévue juste avant l'été. Celle-ci bénéficiera au passage des premiers enseignements tirés des premiers mois de retours collectés auprès des utilisateurs iOS.

Au final Xamarin ou pas ?

Après plus de 2 ans d'utilisation de Xamarin, le bilan n'est pas mauvais ni particulièrement bon et il n'y a pas de réponse toute faite à cette question. En fait cela dépendra avant tout des besoins et des moyens à votre disposition.

Si votre but est avant tout de diminuer les coûts en misant à fond sur la réutilisation de code, que vous avez un écosystème .NET et êtes prêts à faire des concessions sur certains aspects (ergonomie uniformisée, difficulté à trouver des compétences, etc…) alors Xamarin est certainement une solution à considérer car elle possède de nombreux avantages.

En revanche si vous souhaitez avant tout tirer parti à 100% des plateformes, voulez dissocier vos cycles de vie iOS/Android et n'avez pas de temps à perdre en fouillant les forums (Stackoverflow, Microsoft, etc.) à la recherche de solutions aux problèmes que vous rencontrerez, alors Xamarin n'est peut-être pas la solution qui répondra à vos attentes. •

Cédric PRUDENT
Référent réalité virtuelle et augmentée
Expert Web/Javascript
Rédacteur et youtuber
Conserto : ESN conseils et services
ETR : site d'actualités VR/AR

La réalité augmentée : de quoi parle-t-on réellement ?

Commençons par présenter, ou du moins essayons, ce qu'est la réalité augmentée car elle peut s'avérer bien plus complexe qu'elle n'y paraît et se trouve être connexe à bien des technologies et disciplines différentes. D'ailleurs le terme même de 'réalité augmentée' a tendance à être remis en question car les puristes diront que ça n'est pas la réalité que l'on vient enrichir mais bel et bien notre perception.

OK tout ça c'est joli dans le texte mais comment fait-on ? Eh bien il s'agit en fait d'une technologie qui permet d'inclure en temps réel des informations qu'elles soient des textes, vidéo, photos ou objets en trois dimensions, en surimpression de notre vision de la réalité. Elle agit donc comme une d'interface entre la réalité et le virtuel. D'où le terme "Augmenté" car on vient enrichir ce que l'on voit (notre perception de la réalité) avec des informations, le tout de façon interactive. Aujourd'hui un simple smartphone peut nous permettre d'enjoliver cette réalité. Prenons un exemple que j'aime particulièrement : vous vous promenez tranquillement dans votre ville et vous tombez sur un gigantesque entrepôt que vous n'aviez encore jamais vu. Aucun panneau à l'horizon et vous avez beau en faire le tour aucune indication possible sur son usage… Naturellement vous prenez votre téléphone, consultez Google Maps, ou autre selon les convenances, et là il en ressort qu'il s'agit d'un club de fitness associatif dont l'inscription se fait en circuit fermé sur Facebook etc. Nous sommes bien d'accord qu'il s'agit d'une forme de réalité augmentée non ? Puisque sans les informations supplémentaires fournies par votre téléphone, vous n'auriez jamais pu deviner la fonction de ce bâtiment.

Imaginons maintenant que votre téléphone, une fois mis en mode caméra, puisse vous donner en temps réel toutes les informations d'un google street view et vous avez un cas d'usage typique de ce que peut apporter au quotidien la réalité augmentée.

Les similarités entre réalité virtuelle et augmentée.

De prime abord ces deux technologies peuvent sembler très éloignées. Et après tout comment vous donner tort ? L'une est conçue pour vous immerger entièrement dans un monde virtuel avec donc un casque occultant, et l'autre vise à apporter du contenu à votre réalité. Cependant si l'on se place un peu plus loin dans le futur, dans un monde où nos jeux vidéo deviennent tellement réels. Si dans le prochain jeux Grand Thief Auto, les développeurs remodélisaient entièrement la ville de New York en y incluant des données du trafic en temps réel. Alors un tel jeu ne pourrait-il pas être considéré comme si immersif qu'il pourrait être confondu avec la réalité ? Et a contrario dans le cadre d'une réalité que l'on viendrait enrichir à outrance d'éléments 3D, pourrions-nous encore parler de réel ? L'un des principaux usage de la réalité virtuelle dans le domaine professionnel est la formation initiale (apprentissage d'un nouveau poste) où typiquement on cherche à recréer l'environnement réel que l'on vient finalement enrichir avec des informations dédiée à l'apprentissage. Alors ici la réalité virtuelle a un avantage indéniable. Il vient du fait que l'utilisateur n'a pas besoin d'être dans les locaux là où la même version en réalité augmentée nécessiterait une présence physique devant les machines. Mais force est de constater que la démarche est quasiment identique.

Mais il y a également de fortes différences. Tout n'est pas aussi ambigu que ça et certaines différences peuvent fortement peser dans la balance quant au choix d'une technologie pour vos futurs projets :

- La réalité virtuelle nécessite énormément d'artifices afin de pouvoir bluffer complètement le cerveau humain là où, par nature, la réalité augmentée n'en a pas besoin.
- La réalité virtuelle peut être construite autour de lieux qui n'existent pas. Difficile

en réalité augmentée de présenter le nouveau projet d'urbanisme de la place de la Concorde à Paris.
- Les usages de l'AR (pour Augmented Reality) nécessitent forcément d'être présent physiquement devant ce que l'on veut augmenter.
- Le prix du matériel est très largement supérieur en AR (nous y reviendrons plus en détails).

Cette liste n'est évidemment pas exhaustive mais constitue les différences et les critères qui pour nous ont marqué un choix en matière de plateforme et de technologie.

Réalité ou Fiction ?

Je pense que tout le monde à encore bien en tête la vision des années 2054 que portait le réalisateur Steven Spielberg dans son film Minority Report. Il présentait la réalité augmentée, qu'elle soit au travail ou dans le quotidien, en parfaite symbiose avec la réalité. Pour les fans, rappelez-vous la scène où le personnage principal (Tom Cruise) se promène dans le centre commercial et que toutes les annonces marketing étaient en fait des hologrammes. Un panneau d'affichage qui l'appelle par son nom et l'invite à venir dans le magasin, une publicité sous forme d'hologramme qui lui demande s'il a été pleinement satisfait de son dernier achat, etc. Ce "nirvana publicitaire" est basé sur l'existence possible de scanners rétiniens. De plus, les écrans d'ordinateur et les écrans sont transformés en hologrammes. Avec la technologie de l'affichage rétinien virtuel, l'image est scannée directement dans la rétine de l'œil humain. Le spectateur voit ce qui semble être un affichage conventionnel flottant dans l'espace. Cette technologie sophistiquée est loin d'être le fruit de votre imagination et est actuellement en cours de développement au Human Interface Technology Laboratory de l'Université de Washington.

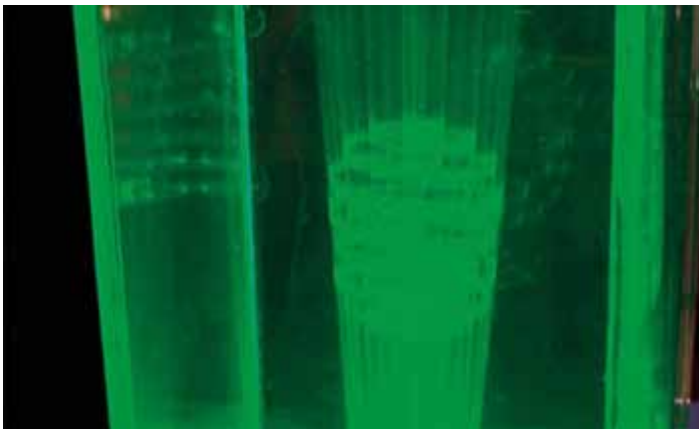
Par essence, la réalité augmentée change considérablement la façon dont nous voyons le monde qui nous entoure et un jour vous marcherez dans la rue avec des écrans d'AR qui ne seront pas plus gros qu'une simple paire de lunette et dont le rôle sera de vous faire apparaître des hologrammes dans votre champ de vision, le tout avec de l'audio synchronisé.

Revenons sur terre…

J'imagine que je ne vous surprends pas en vous annonçant que l'entreprise qui a remis tous les espoirs sur le devant de la scène se nomme Google avec le fameux projet "Google Glass". Il s'agit d'une paire de lunette équipée d'une caméra intégrée, d'un micro, d'un pavé tactile situé sur une des branches, d'un projecteur, d'une prise audio et de toutes les connexions possibles de nos jours telles que le Bluetooth Wi-Fi, etc. Basée sur le système d'exploitation Android, elle permet de se connecter à toutes les applications usuelles. Les lunettes n'étant finalement rien de plus qu'un écran. Commencé officiellement en 2012, ce projet a connu bien des déboires, notamment un effet déceptif auprès du grand public l'ayant acheté mais également pour de gros questionnements de sécurité, jusqu'à se faire stopper début 2015 et retirer de la vente. Depuis peu ces lunettes sont de nouveau en vente dans une nouvelle version destinée exclusivement aux entreprises et offre un confort amélioré.

Beaucoup d'entreprises sont également sur le secteur des lunettes de réalité augmentée, que l'on nommera '2D' en raison de leurs impossibilités à positionner un élément 3D dans le réel. On citera entre autres Epson et Optinvent (entreprise rennaise) avec pour chacune leurs subtilités et spécialités.

Mais comme je vous l'ai écrit plus haut, ce type de device est incapable de positionner un hologramme dans le monde réel afin de, par exemple, vous permettre d'en faire le tour ou même de vous en rapprocher. Et c'est là qu'un second mastodonte du numérique apparaît. Microsoft, qui, fort de ses retours d'expériences, sur le kinect notamment, et en collaboration avec la NASA, dévoile en 2015 le casque HoloLens. Ici on change de paradigme et on part du principe que l'on ne rentrera (du moins pour l'instant) pas tout dans une paire de lunette. C'est donc sous forme de casque qu'ils développent le produit. Il est autonome, c'est à dire qu'il n'y a nullement besoin d'un ordinateur pour le faire fonctionner, et est composé de trois processeurs : le premier est le CPU principal, le deuxième est un processeur graphique (GPU) et le troisième gère les hologrammes (baptisé HPU pour « Holographic Processing Unit »). Vient ensuite la variété de capteurs intégrés, dont une centrale inertielle, un capteur de lumière ambiante, et un jeu de quatre caméras servant à percevoir et cartographier l'environnement combiné à une caméra conçue pour évaluer les distances. On retrouve aussi une caméra de 2 mégapixels pour capturer des photos et des vidéos, en plus de quatre microphones intégrés pouvant notamment capter un son ambiophonique. Ce casque est actuellement encore le meilleur casque quant à son tracking ; les objets que vous spawnez (faire apparaître) dans votre champ de vision restent à leur place même lorsque vous bougez ce qui confère une sensation bluffante de réalisme. Le gros point noir réside dans son FOV (Field Of View) qui, pour vous donner une idée, revient à la taille d'une feuille A4 portée à bout de bras. Ce n'est pas énorme pour l'instant, mais cela devrait grandement s'améliorer dans un futur proche.



Plusieurs types de réalité augmentée ?

Nous l'avons vu la réalité augmentée ne s'arrête pas à votre smartphone. Il existe de nombreuses et diverses façons de venir modifier votre perception du monde. Les smartphones ou tablettes sont probablement, à l'heure actuelle, les objets les plus naturellement utilisés pour une raison d'accessibilité, ils sont omniprésents. Mais il y a de nombreux autres devices :

- La réalité augmentée par projection : le principe est simple. Dans cette catégorie vous allez pouvoir interagir avec des éléments projetés. La détection de l'endroit où l'utilisateur a touché la surface est faite en différenciant l'image de projection attendue (ou connue) et la projection modifiée par l'interférence de la main de l'utilisateur.

Un cas d'usage serait de projeter un clavier dans votre main afin de pouvoir taper un numéro de téléphone par exemple.
L'évolution de la projection est la technologie LaserPlasma qui permet de créer des images 3D dans les airs, ce qui devrait arriver dans un futur assez proche.

- La réalité augmentée à base de reconnaissance : cette méthode est surement l'une des plus utilisées car extrêmement rapide à mettre en œuvre. Des frameworks comme Vuforia permettent de détecter une image ou un marqueur qui permettra d'en déduire la distance, l'orientation et de placer un repère pour y faire "spawner" le modèle 3D de votre choix. Une autre utilisation concrète du système avec reconnaissance (Recognition) serait de remplacer les panneaux de signalisation étrangers par ceux de votre pays et le tout à la volée. Sachez que l'on peut également faire de la reconnaissance sur des objets du quotidien et que l'on y reviendra. Si j'ai cité Vuforia, il y a aussi son homologue open-source OpenCV qui fonctionne également très bien.
- Aide à la vision : bien que votre œil soit plus efficace que le plus puissant des appareils photo, il n'en reste pas moins limité. Nous ne pouvons voir les choses de près, nous ne voyons pas bien dans des conditions de faible luminosité, etc. Pour cela on se sert bien évidemment de caméras spécifiques et on transmet une image à l'utilisateur en superposant ce que la caméra voit avec sa propre vision. Une mise en situation concrète de ce type d'usage sera les parebrises AR que l'on ne manquera pas de voir sur les voitures d'ici à 5 ans sur les modèles haut de gamme. En effet, un tel dispositif pourra non seulement vous éclaircir la route, voire la dessiner, lors de grosses intempéries, mais également vous mettre en surbrillance le cerf situé sur le bord de la route que vous n'aviez pas encore vu.
- La réalité augmentée par superposition : son but est de fournir une vue alternative de l'objet concerné, soit en remplaçant entièrement la vue, soit une partie. Dans ce cas on fera obligatoirement appel à la reconnaissance d'objet, nerf de la guerre quand on parle d'AR. C'est typiquement cet usage qui tend à se démocratiser le plus avec des casques comme le HoloLens car ils permettent à des chirurgiens d'accéder à des informations tout en garantissant la stérilité des lieux, ou même de pouvoir superposer le résultat d'une IRM (Imagerie par Résonance Magnétique) exactement sur le patient afin d'avoir en visuel une vue de l'intérieur de son corps. L'éducation est également un secteur privilégié de ce type de réalité augmentée pour par exemple : la compréhension du corps humain en utilisant un hologramme sur un mannequin réel.

On repousse les limites

La réalité augmentée n'est pas qu'une lubie ou une passion permettant de créer des jeux et du divertissement. Il y a énormément de contenus qui permettraient de simplifier la vie dans le secteur de la médecine comme évoqué au-dessus mais aussi dans le domaine du sport ou même militaire… Au fur et à mesure que de nouvelles demandes d'information sont apparues, de nouvelles façons d'ajouter de l'information à la réalité ont été trouvées. Chaque jour qui passe voit un nombre croissant d'applications de réalité augmentée, et son potentiel semble infini. Je vais donc essayer de vous montrer comment l'AR modifie déjà nos vies dans certains domaines clés (non exhaustif) et à quoi s'attendre dans les années à venir.

Domaine Militaire :

Ce n'est pas nouveau, beaucoup de technologies sont portées soit par l'industrie de la pornographie soit par celle du militaire, quand ce ne sont pas les deux. Mais il faut bien admettre que cela possède de nombreux avantages car généralement les solutions qui atterrissent dans le domaine civil sont plutôt fiables. Aidée par des équipements tactiques spécifiques portés par le personnel militaire lors d'une opération, la réalité augmentée peut même permettre de révéler des choses cachées à l'œil nu comme des mouvements de troupes ennemies, des objets physiques, types et portées d'armes ennemies, points de vulnérabilité dans les défenses ennemies, suivi simultané de personnel ennemi multiple, etc. Toutes ces améliorations qui bientôt s'avéreront indispensables à l'infanterie de demain.

Domaine Médical :

Le fait que la réalité augmentée puisse fournir plus d'informations avec un minimum de distraction en fait un candidat idéal pour les sciences médicales. Combinée avec du matériel médical adapté, elle peut même aller jusqu'à superposer en temps réel les images de l'opération en cours directement sur le patient façon rayon X. D'autres utilisations incluent la visualisation des articulations en direct, le suivi des petits outils de chirurgie pendant une opération, etc. Là aussi nous sommes sur un domaine que la réalité augmentée, couplée également à la robotique, devrait complètement révolutionner dans les années à venir.

Domaine marketing :

Un incontournable. Impossible de ne pas en parler tellement il est important dans notre économie. Et dans ce milieu également, la réalité augmentée peut se retrouver à avoir un impact considérable sur les ventes sur internet. L'un des problèmes majeurs sur la vente à distance est, bien entendu, de ne pas pouvoir voir l'objet que l'on achète, son volume, son aspect. Mais elle peut apporter aussi beaucoup

dans les centres commerciaux avec l'apparition de cabine d'essayage virtuelle qui permettent aux client de se projeter dans une tenue particulière ou un accessoire sans avoir à le porter. La technique utilisée est une superposition, on prend une caméra que l'on pointe sur le client puis on vient poser l'article sur le corps de l'acheteur potentiel. Au dire de la société développant la première cabine virtuelle, le marché aurait été estimé à 600 millions d'euros.

Domaine touristique :

Le tourisme est également un client évident pour l'AR, que cela soit au niveau publicité ou au niveau informations supplémentaires du site touristique. Venir afficher aux visiteurs des informations ou de la reconnaissance de panneaux si vous êtes à l'étranger. L'une des applications, IOS ou Android, les plus connues s'appelle "Layar" qui se prononce layer et qui signifie couche ou strate en anglais et dont le principe est de permettre aux utilisateurs de créer leur propre contenu d'information sur des endroit qu'ils aiment ou ont aimés via la reconnaissance avec la caméra du smartphone.

Domaine du bâtiment :

Si vous avez vu "Prometheus" ou "Avatar" alors j'imagine que vous avez d'ores et déjà une petite idée de ce que l'AR peut faire pour nous en matière d'architecture. Mais pour les autres, on va refaire un point rapide. Elle peut vous aider à visualiser des structures avant qu'elles ne soient construites et simuler leur comportement dans diverses situations, météorologiques, catastrophes naturelles, etc. Et ce, que vous les vouliez au 1/50ème ou à l'échelle 1:1. Cela permet de voir concrètement les volumes pour un particulier et pour un contremaître de détecter les anomalies (exemple la fenêtre n'est pas au bon endroit à 1 mètre près) et de réagir en conséquence.

Domaine sportif et du divertissement :

Dans le cadre du sport, le Tennis, par exemple, a adopté et adapté avec brio l'AR afin d'aider les arbitres et les spectateurs à mieux voir l'action. Pour le divertissement on a également pu voir la réalité augmentée utilisée dans les manèges pour enfants qui pouvaient jouer avec des animaux virtuels allant du petit chiot aux dinosaures géants. Je passe rapidement sur l'art et l'éducation mais on peut se douter pouvoir y voir des choses incroyables et c'est l'exemple typique que les limites de cette nouvelle réalité ne seront que celles que l'on voudra bien y mettre.

Mais quels sont les choix de casques ?

Pour les casque de réalité augmentée, à l'heure actuelle seuls 4 gros constructeurs sont en compétition réelle pour la suprématie de la réalité augmentée. Microsoft a pris la tête avec son HoloLens dont on a parlé au-dessus mais il y a de plus en plus de concurrents comme par exemple Magic Leap qui sort un casque courant 2018 et qui, à ce jour, a réussi une levée de fonds de 2,3 milliards de dollars pour développer son produit depuis 2010. Des sommes colossales que la société justifie en vantant une véritable technologie de rupture. Moins médiatique, on retrouve DaQri beaucoup plus spécialisée sur le bâtiment avec notamment un système pour automatiser les exports et la visualisation avec des plateformes d'architecture comme Revit. Et le meta2 qui est le seul des quatre à devoir être relié à un ordinateur mais qui par extension bénéficie d'un boost de puissance de calcul permettant au casque de doubler le champ de vision par rapport à ses concurrents.
Ce sont tous des casques permettant de positionner un éléments 3D et de se déplacer par rapport à celui-ci.

Quid de l'avenir ?

Difficile de ne pas être optimiste sur ces technologies qui au-delà de pouvoir vous simplifier la vie en agrémentant votre quotidien d'informations choisies et maîtrisées peut également apporter un soutien énorme dans des secteurs comme dans le médical qu'il soit pour usage thérapeutique ou même pour les professionnels de santé eux-mêmes. Je ne doute pas que certains métiers comme par exemple les techniciens de maintenance bénéficieront de ces outils très prochainement si cela n'est pas encore le cas et nous ne voyons ici que la partie émergée de l'iceberg. •

Sebastien Bovo
Windows AppConsult Engineer chez Microsoft. En charge d'accompagner les partenaires et développeurs sur la plateforme Windows Mixed Reality Immersive et Holographique pour la publication sur le Store Microsoft.
sbovo@microsoft.com | twitter.com/sbovo

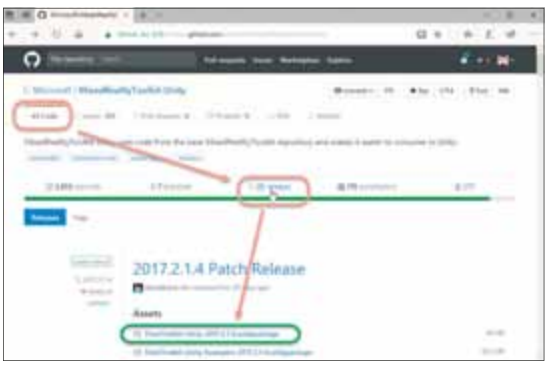
Aller plus loin sur Windows Mixed Reality

1

Mixed Reality Toolkit (MRTK)

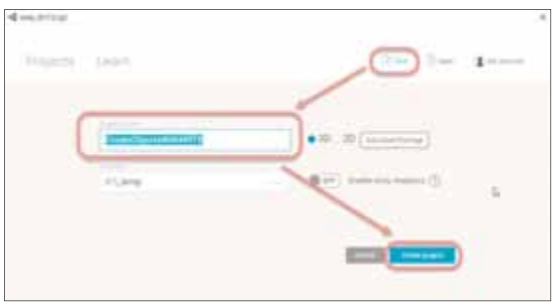
Ce projet Open Source a pour but d'accélérer vos développements Mixed Reality aussi bien Immersifs (VR) qu'Holographiques (HoloLens). Il n'est pas obligatoire mais vous apporte des scripts et des contrôles afin de faciliter la gestion des contrôleurs, du son spatialisé, des commandes vocales, de la manipulation d'objets ; la liste est très longue… 1

Alors comment utiliser ce toolkit ? Commençons par le télécharger à partir du projet Github https://aka.ms/MRTK. Choisir l'onglet **Code**, puis **Releases** ; il suffit ensuite de télécharger le dernier fichier **.unitypackage** de la release actuelle. Lors de l'écriture de cet article, il s'agit de la version "2017.2.1.4 Patch Release".

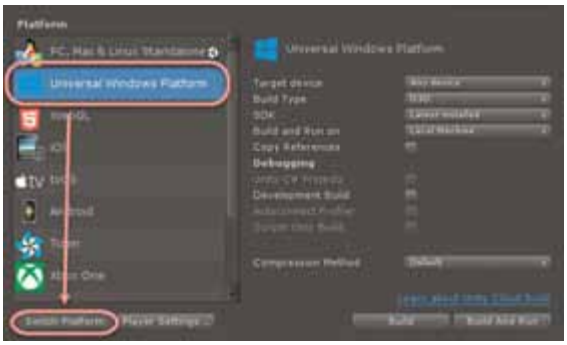


Partons ensuite sur la création d'un nouveau projet Unity :

- Lancer Unity3D ; créer un nouveau projet en cliquant sur **New,** le nommer "CreateObjectsWithMRTK" ; cliquer sur **Create project**.

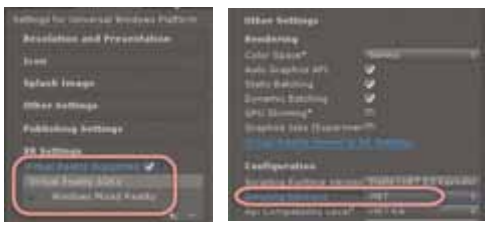


- Ciblez ensuite la plateforme UWP en utilisant le menu **File** puis **Build Settings…** ; choisir **Universal Windows Platform** puis cliquer sur **Switch Platform**.

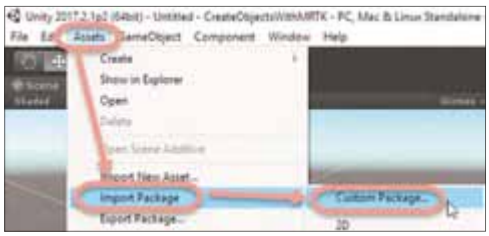


- Vérifier le support de la VR et du Scripting C# : toujours dans l'écran **Build Settings**, cliquer sur **Player Settings**.

- Le "fenêtre **Inspector**" affiche les options relatives à l'application. Cliquer sur **XR Settings** et cocher **Virtual Reality Supported**. Puis dans **Other Settings,** sélectionner **.NET** comme **Scripting Backend**.

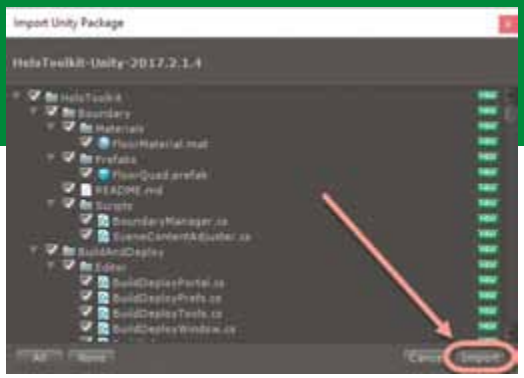


- Importer enfin le Mixed Reality Toolkit en utilisant le menu **Assets** / **Import Package** / **Custom Package…**



- Choisir le fichier **.unitypackage** précédemment téléchargé.

- Unity vous propose de sélectionner une partie ou tous les composants contenus dans ce package. Par défaut, tout est sélectionné ; cliquer sur **Import**.



• Sauvegarder le projet avec le menu **File** / **Save project**.

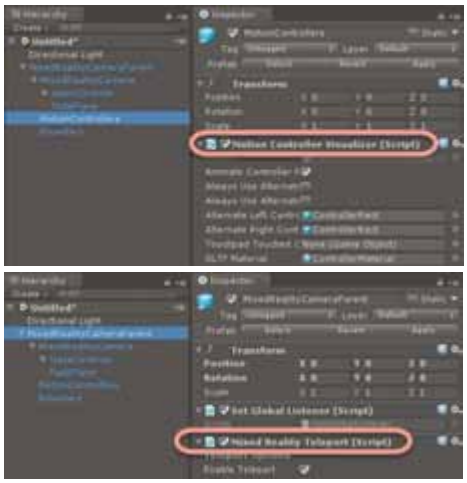
Nous avons donc maintenant un projet Windows Mixed Reality avec le MRTK mais vide !

Caméra, contrôleurs et téléportation

Chaque nouveau projet Unity contient par défaut une caméra nommée "Main Camera" et une source de lumière dite "Directional Light". Profitons du Toolkit pour utiliser en remplacement de la caméra un prefab (c'est-à-dire un modèle d'objet) avec des fonctionnalités plus avancées. Il s'agit de **MixedRealityCameraParent**. Ce prefab sert de caméra Immersive (VR) ou Holographique (HoloLens) en fonction du périphérique sur lequel l'application s'exécute.

• Dans la fenêtre **Project**, rechercher en utilisant les premières lettres "mixed". Le prefab **MixedRealityCameraParent** sera listé ; faire un glisser/déplacer de cet élément dans la fenêtre **Hierarchy**.

Le prefab **MixedRealityCameraParent**, en plus de positionner une caméra à l'origine (0, 0, 0), nous fournit un script qui se charge d'afficher et d'animer les contrôleurs Mixed Reality et un autre dédié à la téléportation.

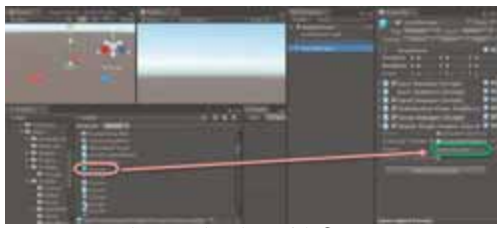


La gestion des évènements de connexion/déconnexion des contrôleurs ainsi que les actions et interactions effectuées avec ceux-ci est faite par un autre prefab **InputManager** :

• Dans la fenêtre **Project**, rechercher "inputmanager" ; faire un glisser/déplacer du prefab **InputManager** dans la fenêtre **Hierarchy**.

• Toujours dans la fenêtre **Project**, rechercher "cursor" ; une multitude de composants apparaîtront, repérer celui en forme de cube bleu nommé **Cursor** (Nous l'utiliserons juste après).

• Dans la fenêtre **Hierarchy**, sélectionner l'objet **InputManager** précédemment ajouté. Ces propriétés s'affichent dans la fenêtre **Inspector**. Localiser le script **Simple Single Pointer Selection** et en particulier la propriété **Cursor**. Effectuer maintenant l'opération la plus délicate de cet article : glisser/déplacer le prefab **Cursor** (de la fenêtre **Project**) vers l'emplacement de la propriété **Cursor** (dans la fenêtre **Inspector**).



La propriété se voit donc attribué le prefab **Cursor** qui servira, comme son nom l'indique, de visuel pour le curseur de visé (dit gaze).

Afin que les boutons des contrôleurs soient interceptés, nous avons une dernière configuration à effectuer ; ici aussi, le toolkit nous aide.

• Utiliser le menu nouvellement créé par l'ajout du toolkit : **Mixed Reality Toolkit** / **Configure** / **Apply Mixed Reality Project Settings**.

• Sélectionner seulement **Use Toolkit-specific InputManager axes**. Cette commande va modifier un fichier de paramétrage en indiquant les nouveaux boutons à prendre en compte (Pour votre information, ce fichier est /ProjectSettings/InputManager.asset dont le contenu est accessible par le menu Edit / Project Settings / Input).

Ajoutons ensuite un sol afin de pouvoir tester la téléportation :

• Créer un cube à partir du menu **GameObject** / **3D Object** / **Cube**

• Le nommer "Floor" et changer ces dimensions pour créer un objet représentant un sol : x=10, z=10.

Abonnez-vous à Programmez! Abonnez-vous à Programmez! Abonne

Nos classiques

1 an ..... 49€*
11 numéros

2 ans ..... 79€*
22 numéros

Etudiant ..... 39€*
1 an - 11 numéros

** Tarifs France métropolitaine*

[Programmez!] Le magazine des développeurs

Abonnement numérique

PDF ..... 35€
1 an - 11 numéros

Souscription uniquement sur
www.programmez.com

Option : accès aux archives 10€

Offres printemps 2018

1 an ..... 59€
11 numéros
+ 1 vidéo ENI au choix :

- Symfony 3*
  Développer des applications web robustes
  (valeur : 29,99 €)
- Raspberry Pi*
  Apprenez à réaliser et piloter une lumière d'ambiance
  (valeur : 29,99 €)

2 ans ..... 89€
22 numéros
+ 1 vidéo ENI au choix :

- Symfony 3*
  Développer des applications web robustes
  (valeur : 29,99 €)
- Raspberry Pi*
  Apprenez à réaliser et piloter une lumière d'ambiance
  (valeur : 29,99 €)

** Offre limitée à la France métropolitaine*

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

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

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

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

☐ Vidéo : Symfony 3
☐ Vidéo : Raspberry Pi

PROG 219
Offre limitée, valable jusqu'au 29 juin 2018

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

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

E-mail : @

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

*** Tarifs France métropolitaine**

Offres 20e anniversaire !*

1 an - 11 numéros	2 ans - 22 numéros
79,99 €	99,99 €
(au lieu de 147,99 €)	(au lieu de 177,99 €)

+

1 clé USB contenant
tous les numéros depuis le n°100

+

4 numéros vintages
(selon les stocks)

+

1 carte maker Digispark ATtiny84
compatible Arduino

+

1 lot de composants / capteurs
(selon arrivage du jour)

+

1 an de Pharaon Magazine soit 4 numéros :
pour découvrir l'Egypte des Pharaons !

* *Offre réservée à la France métropolitaine*

Sur notre site web uniquement : **https://www.programmez.com/catalog/20eme-anniversaire**

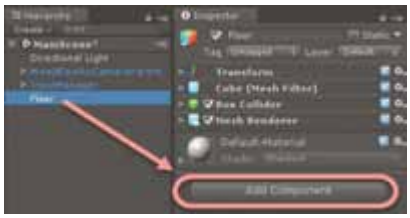
• Sauvegarder la scène avec le nom "MainScene" (menu **File** / **Save Scenes**).

Testons la téléportation en lançant l'application dans Unity avec le bouton **PLAY** .

Actions sur les objets

Allons un peu plus loin en faisant en sorte que le sol réagisse au bouton principal du contrôleur. Reprenons, par exemple, une partie du code de l'article de **programmez! 218** : à chaque appui du bouton, un cube est créé et tombe sur le sol.

• Dans la fenêtre **Hierarchy**, Cliquer sur l'objet **Floor**. Cliquer sur le bouton **Add Component** de la fenêtre **Inspector**.



• Dans la zone de recherche, taper "Script" et cliquer sur **New Script**

• Entrer le nom "Tap" et cliquer sur **Create and Add**

• Nous avons donc maintenant un nouveau script associé à l'objet **Floor**, pour le modifier double-cliquer sur le script **Tap** : Visual Studio s'ouvre pour l'édition.

• Implémenter l'interface **IInputHandler** amenée par le Mixed Reality Toolkit (espace de noms **HoloToolkit.Unity.InputModule**).

Cette interface permet de très simplement intercepter le clic, le tap ou le bouton principal du contrôleur.

```
public class Tap : MonoBehaviour, IInputHandler {
  void IInputHandler.OnInputDown(InputEventData eventData)
  { }

  void IInputHandler.OnInputUp(InputEventData eventData)
  { }
```

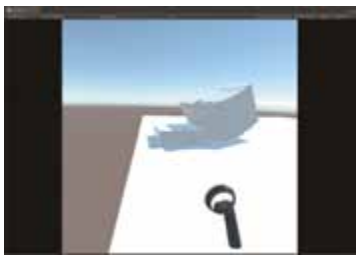
• Ajoutez les lignes de code suivantes dans la méthode **OnInputUp** :

```
// Créer un GameObjet simple
var o = GameObject.CreatePrimitive(PrimitiveType.Cube);

// Planche de 1m sur 1m (x et z), épaisseur 10cm (y)
o.transform.localScale = new Vector3(1f, 0.1f, 1f);

// Changer sa position : 2m de hauteur (y=2), 4m de distance (z=4)
o.transform.position = new Vector3(0f, 2f, 4f);

// Soumis à la gravité
o.AddComponent<Rigidbody>();
```

Le code final est le suivant :

```
using HoloToolkit.Unity.InputModule;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Tap : MonoBehaviour, IInputHandler
{
  void IInputHandler.OnInputDown(InputEventData eventData)
  { }

  void IInputHandler.OnInputUp(InputEventData eventData)
  {
    var o = GameObject.CreatePrimitive(PrimitiveType.Cube);
    o.transform.localScale = new Vector3(1f, 0.1f, 1f);
    o.transform.position = new Vector3(0f, 2f, 4f);
    o.AddComponent<Rigidbody>();
  }

  // Use this for initialization
  void Start()
  { }

  // Update is called once per frame
  void Update()
  { }
}
```

Vous pouvez de nouveau tester : la téléportation vous permet de vous déplacer et chaque appui sur le bouton principal crée une nouvelle planche qui tombe sur le sol !

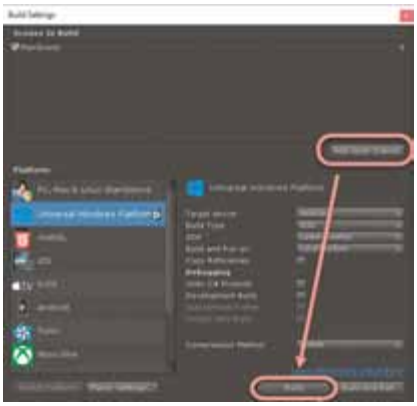


Le projet Visual Studio et le Store Microsoft

La compilation d'un projet Mixed Reality Unity ne produit pas un exécutable ou un installeur mais une solution Visual Studio de type "Application UWP" :

- Utiliser le menu **File** / **Build Settings**…

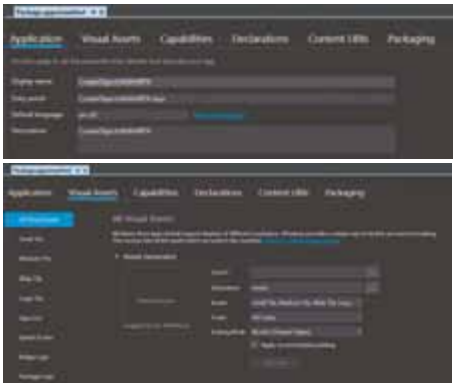
- Sélectionner les scènes à inclure dans la compilation avec le bouton **Add Open Scenes** et cliquer sur **Build**.



- Créer un nouveau dossier nommé, par exemple, "App" et cliquer sur **Select Folder**.

- Ouvrir la solution en double-cliquant sur le fichier **CreateObjectsWithMRTK.sln** contenu dans le sous-dossier **App**.

- Il est donc possible, à ce stade, de modifier et finaliser tous les réglages correspondant à l'identité, aux assets comme les tuiles, le splashscreen de l'application grâce au fichier **Package.appxmanifest**. Ceci comme toute application UWP.

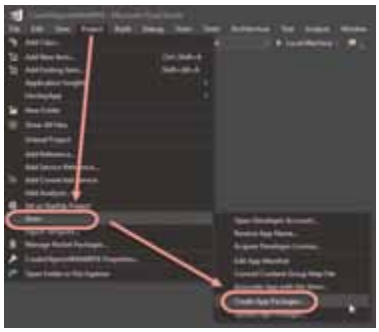


Le Store Microsoft en vue

La finalité est de pouvoir enfin générer le package et le soumettre sur le Store Microsoft. Pour ce faire, vous avez besoin d'avoir un **compte Dev Center Microsoft**. Si vous ne possédez pas encore ce "compte développeur", il vous suffit de vous rendre sur https://dev.windows.com pour le créer. Ce compte coute 14€ pour les particuliers et 75€ pour les entreprises. Vous devrez l'associer à un compte "Microsoft Live".

A partir du moment où votre compte Dev Center est actif, vous pouvez revenir sur la solution Visual Studio : nous allons créer le package et réserver un nom pour l'application dans le Store Microsoft.

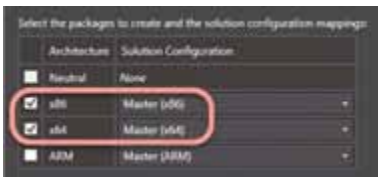
- Utiliser le menu **Project** / **Store** / **Create App Package**.



- Indiquer dans l'assistant que le package est à destination du Store Microsoft en sélectionnant **Yes**.

Se loguer avec le compte "Microsoft Live" associé au compte Dev Center. Utiliser un nom d'application que vous avez déjà réservé, ou cliquer sur **Reserve** pour en choisir un nouveau.

- Choisir **x86** et/ou **x64** en mode **MASTER** afin que les performances soient optimales et que l'application soit acceptée par le Store.



- Pour finir, cliquer sur le bouton **Create** afin d'obtenir le package **.appxbundle** que vous allez pouvoir uploader en vous connectant sur le Dev Center https://dev.windows.com/.

Conclusion

Nous avons vu ici comment mettre en œuvre le Mixed Reality Toolkit. La prochaine étape est de vous connecter sur le projet Github https://aka.ms/MRTK pour en parcourir la documentation mais aussi et surtout tester les multiples scènes d'exemples fournies. Vous trouverez le code complet de cet article sur https://aka.ms/mixedrealityappfrom0. Le monde virtuel est à portée de vos mains ! •


David PANZA
Développeur
@PANZATech


Jimmy TRAN
Développeur
@Société Générale


Ubald de BELLABRE
Développeur
@La combe du lion vert

Comment choisir sa bibliothèque ou son framework ?

" Hey, perdu dans cette jungle de bibliothèques et de frameworks ? Oui, vous qui ne savez pas quoi penser de tout ce bazar et du fait d'avoir 10 alternatives répondant à votre besoin. Vous qui ne savez pas quoi tirer des conférences, venez par-là, nous avons sans doute quelque chose pour vous ! "

L'intégration d'un framework ou d'une bibliothèque est un choix important et impactant pour une application. Vous vous êtes sûrement retrouvés dans des situations où vous deviez faire **un choix**. Par manque de méthodologie ou d'accompagnement, cette décision **peut être prise de manière peu constructive**.

Il nous est arrivé de regretter nos choix au cours de nos précédentes expériences, comme intégrer des bibliothèques internes non adaptées ou bien de fausses bonnes idées à nos applications ... De ces erreurs, nous avons essayé de tirer des conclusions et des "bonnes pratiques" que nous partagerons à travers cet article. Rappelons que dans le développement logiciel, l'expérience s'accumule au fil des succès et surtout des échecs. En effet, le **développement logiciel reste un processus d'apprentissage** et le code n'est qu'un effet de bord(1). Alors, si à vous aussi, on vous a déjà demandé pourquoi ce framework en particulier et que vous avez répondu ...

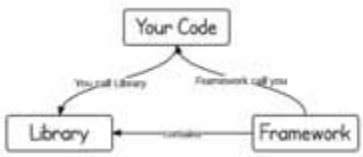
... cet article est fait pour vous !

Afin de bien commencer, il est donc primordial de connaître la différence entre bibliothèque et framework. Dans le cadre de notre démarche, nous expliquerons comment filtrer les choix qui ne correspondent pas à votre contexte. Puis, nous vous exposerons nos critères pour départager les différentes possibilités. Enfin, nous appliquerons cette démarche à travers une mise en situation.

Framework vs bibliothèque

Un projet comporte très souvent des fonctionnalités déjà développées par vos pairs et il est intéressant de les réutiliser au lieu d'essayer de réinventer la roue. Que vous adoptiez un framework ou une bibliothèque, les deux formats sont du code maintenu par d'autres développeurs. Il est potentiellement bien écrit et mieux pensé car il couvrira des cas limites auxquels vous ne penseriez pas. **Un bon framework ou une bonne bibliothèque devrait vous permettre de vous focaliser sur la vraie valeur attendue par votre métier**.

La différence entre les deux formats vient de leurs intentions :



Ledion Spaho, 2015(2)

En effet, **une bibliothèque** est un ensemble de fonctionnalités qui **se concentre sur un besoin bien précis** et ne fera que ça. Par exemple, il existe des bibliothèques de test (ex : Jest, Jasmine...), des bibliothèques de conversion (ex : Jackson, GSON...) etc. En général, le code d'une bibliothèque reste relativement petit et maîtrisable. Elles sont aussi pour la plupart **composables** avec d'autres bibliothèques car elles sont à responsabilité unique(3).

A l'inverse, comme expliqué dans le livre publié par Gang of four(4), un framework portera l'accent surtout sur la réutilisation de sa conception plutôt que sur la réutilisation d'une de ses fonctionnalités. Rappelons qu'**un framework**, par définition, est un "cadre de travail". Il se traduit par des règles, des conventions, des suppositions et des choix techniques. Cela **va vous contraindre à développer d'une certaine manière au profit d'une productivité accrue**, surtout dans le cas d'un POC(5) ou lorsque l'on démarre un nouveau projet.

Néanmoins, dans les deux cas, la réutilisation de code via une bibliothèque ou un framework oblige les développeurs à se documenter sur la manière dont on les utilise. C'est **une complexité qui sera transmise aux autres développeurs**. C'est pourquoi, il est important de bien analyser le besoin derrière chaque réutilisation, sous peine d'introduire une complexité accidentelle(6). Au début, votre choix peut vous sembler être une bonne idée, mais gardez en tête que vous êtes rarement seul(e) à travailler sur une application et que le gain doit se calculer sur la durée (du développement à la maintenance). Réutiliser du code **peut booster le démarrage d'un projet, mais pourrait ralentir ceux qui reprendront le code derrière vous**.

Sachez aussi qu'**introduire une dépendance augmente le coût de maintenance** de votre application. Une dépendance que vous ne mettez pas à niveau va potentiellement vous générer de la dette. Une dette qui augmentera en fonction de l'écart de version entre celle intégrée dans votre appli-

(1) The final words about software, Alberto Brandolini
(2) What's the difference between library and framework ? Ledion Spaho
(3) Single Responsibility Principle de SOLID. Les principes SOLID dans la vie de tous les jours, Raphaël Squelbut, Arolla
(4) Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm, 1994, Design Patterns: Elements of Reusable Object-Oriented Software
(5) Proof Of Concept
(6) 'There is no silver bullet' de Fred Brooks parlant de complexité accidentelle et essentielle

cation et la plus récente du marché.
Voici une illustration de ce qui nous est personnellement arrivé il y a quelques mois et qui nous arrive très souvent côté JS : 1

Prendre en compte son environnement

Avant chaque choix technique, il est important de bien prendre en compte son environnement de travail. En effet, votre choix sera bien sûr pondéré par les équipes qui travaillent avec vous mais aussi par le contexte de votre projet. **Il faut donc se poser les bonnes questions afin de filtrer les différentes solutions qui répondent à votre besoin**. Nous vous recommandons tout d'abord de vous intéresser aux contraintes fortes de votre contexte car elles filtrent beaucoup de possibilités.

- Si vous avez déjà des **contraintes liées à un langage de programmation ou à un framework**, votre choix devient tout de suite bien plus limité. Par exemple, il est évident que vous allez écarter toutes les bibliothèques écrites spécifiquement pour des projets Angular si votre projet est écrit en React.
- De plus, il y a souvent **des bibliothèques internes** qui répondent plus ou moins bien à votre besoin mais qui **vous seront parfois imposées pour un besoin de réutilisation et d'homogénéisation**. Parfois, vos choix seront donc restreints au profit d'une homogénéisation technologique sur l'ensemble de votre structure, même si cela ne représente pas la meilleure solution pour votre besoin client.

Voici **une carte mentale** que nous avons pensée et qui pourrait vous aider à déterminer quels critères de sélection méritent une attention particulière compte tenu de votre contexte : 2

Cette carte mentale vous aide à vous poser les bonnes questions afin d'**explorer votre environnement**, et à bien **visualiser l'impact de vos futurs choix techniques**. Elle vous permettra d'argumenter votre choix selon plusieurs critères de sélection en fonction de vos contraintes. Ces critères sont présentés dans la partie suivante.

Nos critères de sélection

Complétion du projet

Une bibliothèque supportée par une communauté active et régulièrement mise à jour vous offrira la possibilité de soumettre votre avis et de suggérer des corrections ou évolutions. Pour cela, il faut regarder :

- L'organisation autour du projet ;
- La roadmap du projet (ex : Trello avec les prochaines évolutions) ;
- La fréquence de releases ;
- Le nombre de forks ;
- Le nombre de branches actives sur le repository Github.

Il faut néanmoins garder en tête que des montées de versions seront potentiellement inévitables. Il y aura peut-être aussi des changements de versions majeurs qui nécessitent des efforts et engendrent même parfois des imprévus. En prenant une migration Spring par exemple, il peut y avoir de nombreux gains, mais il faut aussi faire attention aux changements d'API(7), aux méthodes dépréciées, aux changements de dépendances transitives, aux changements de packaging (ex : Spring 1.4 release notes(8)).

Un produit fini devient stable et représente un bon choix. Ainsi, le développement de Express est fini depuis des années, mais il reste le framework le plus utilisé pour les backends NodeJS d'après l'étude de NPM(9). Le cœur du projet n'évolue plus et les rares commits récents mettent à jour des dépendances ou suppriment des éléments dépréciés.

A contrario, **une bibliothèque abandonnée**, c'est-à-dire incomplète ou qui n'est plus maintenue par sa communauté, **représente un risque pour le projet**. Le concept de la bibliothèque était peut-être mauvais, et les développeurs l'ont abandonné. Elle peut aussi contenir des anomalies qui ne seront jamais résolues et qui risquent d'affecter ses fonctionnalités ou d'introduire des failles de sécurité. Il faut donc éviter d'importer les dépendances de ce type.

Enfin, **un produit en phase de développement** ou de bêta **n'est pas conseillé en production**, car il évolue trop, mais peut toutefois être **intéressant dans un contexte de veille technologique**.

Couplage

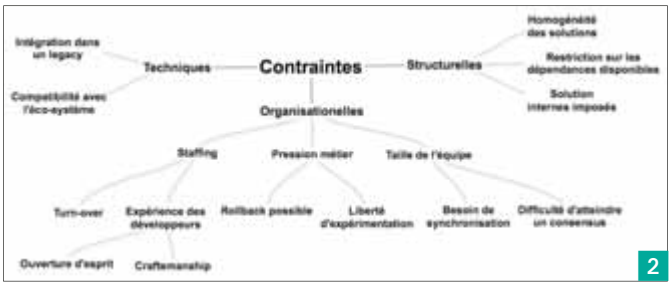
Il ne faut jamais sous-estimer **le couplage avec une bibliothèque et surtout avec un framework**. Un framework lié à un langage ou une version d'une bibliothèque peut être gênant, n'étant **pas agnostique**.

Dans le contexte d'une petite organisation, type startup, il est primordial de pouvoir revenir sur ses choix techniques, afin de s'adapter aux contraintes du marché. Plus le couplage est élevé, moins vous aurez la possibilité de changer lorsque le besoin s'en fera ressentir.

La question à se poser avant même d'adopter votre bibliothèque ou framework est la suivante : "**Combien me coûterait le changement ?**"

Une réduction des coûts peut être envisagée via l'utilisation du pattern Port/Adapter(10). En effet, si le découplage est bien réalisé, il est facile de remplacer une biblio-

JS Fatigue, dessin Lily Khamsyvoravong 1



2

(7) Application Programming Interface - (8) Notes de release Spring 1.4: https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-1.4-Release-Notes
(9) D'après le gestionnaire de dépendances front-end le plus populaire NPM https://www.npmjs.com/npm/the-state-of-javascript-frameworks-2017-part-3-back-end-frameworks
(10) http://alistair.cockburn.us/Hexagonal+architecture

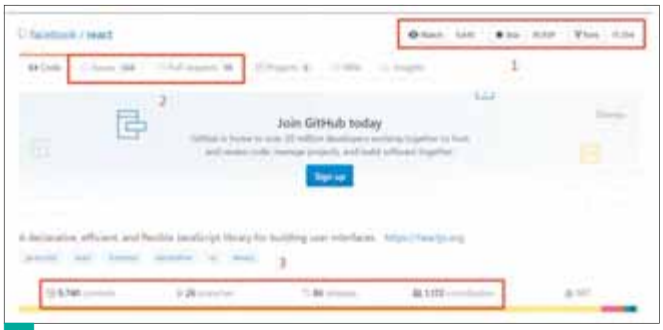
thèque par une autre.

De plus, quand on parle de **couplage**, on parle d'**une bibliothèque/framework**, mais la question se pose aussi pour les versions attenantes. Pour ceux qui ont eu la chance de connaître le passage d'AngularJS à Angular, la promesse de Google vis-à-vis de cette migration était très optimiste, soi-disant transparente, mais globalement loin de la réalité du terrain(11). Le passage a certes été facilité par leur version intermédiaire comme la 1.5 mais il restait douloureux en grande partie à cause du changement du paradigme, passant du MVC en component.

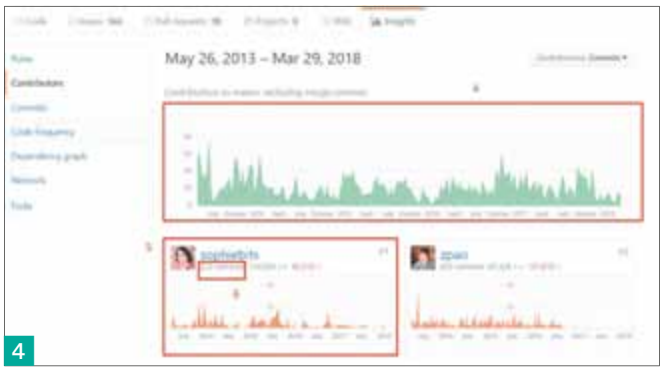
Attention aussi aux montées de version (voir section Complétion), car cela pourrait entraîner un changement cassant. Nous vous invitons donc à jeter un coup d'œil à un bon talk de Rich Hickey qui illustre ce fait : Spec-ulation Keynote(12).

Nous avons parlé du couplage de votre code métier à une bibliothèque/framework, mais il en va de même pour le framework en lui-même. Si l'on prend l'exemple de Spring Boot, **le couplage transitif** de ce framework **vous lie aussi** aux choix faits par celui-ci. Par exemple, le couplage de Spring Boot et le client ElasticSearch. Ces deux projets ont des cycles de release complètement différents et il n'est pas rare de devoir faire évoluer Spring Boot en lui-même pour pouvoir bénéficier de la dernière version d'ElasticSearch.

Ceux-ci ne sont que quel,ques exemples mais illustrent bien les difficultés d'évolutivité de votre stack technique vis-à-vis du marché. C'est pour cette raison que **nous vous conseillons de vous tourner vers des bibliothèques** qui ne sont pas supposées vous lier à un style de code impactant votre code métier. Et à l'inverse, un framework aura déjà fait le choix de votre stack et vous garantit une bonne intégration transitive, ce qui peut s'avérer très rassurant lorsqu'**on a peu de temps ou pas envie de monter sa propre stack**.



3



4

5

Popularité

La popularité d'une bibliothèque ou d'un framework n'est pas un critère rédhibitoire, mais reste un facteur rassurant dans la mesure où **vous aurez une communauté impliquée derrière le projet**. Elle pourra vous aider en cas de problème, de demande d'évolution ou pour répondre à vos éventuelles questions. Attention toutefois à ne pas tomber dans le hype ou l'effet de mode du moment et **focalisez-vous plutôt sur le besoin**.

Github 3

Si le projet est versionné sur Github, vous pourrez en constater la **popularité** (voir zone 1) **par le nombre de pairs qui suivent ses évolutions**, le **nombre d'étoiles** ou encore par **le nombre de forks** à des fins de contribution.

Nous parlions aussi précédemment de problèmes dans la bibliothèque et de demandes d'évolution, il s'agit là de voir **la réactivité des mainteneurs du projet** (voir zone 2). Vous pourrez suivre dans les indicateurs le rythme auquel les problèmes et les propositions d'évolutions sont traités.

Puis, vous trouverez des **informations sur l'activité du projet** (voir zone 3) : le nombre de commits depuis le début du projet, le nombre de versions livrées avec les notes de publication décrivant les modifications, et le nombre de contributeurs. Il est important de vérifier cette dernière information pour avoir plus de détails car ce chiffre ne reflète pas toujours le **nombre réel de contributeurs actifs**. 4

En zoomant sur la partie contributeurs, vous pouvez observer l'activité détaillée sur le projet (voir 4) et **déterminer si de fortes évolutions sont en cours**. Attention toutefois aux nombres de contributions apportés (voir 5) et surtout aux périmètres de celles-ci. Un commit sur de la documentation ou un déplacement de fichier, même si cela représente une évolution mineure, fait du développeur un contributeur actif. **Pensez à regarder les commits des contributeurs** (voir 6) **pour en déduire leur niveau d'implication**.

Présence sur le web

Il existe **de nombreuses communautés** où des développeurs posent des questions comme **sur Stack Overflow**. Avec un peu de chance, "Google est ton ami" et votre problème aura déjà été soulevé résolu.

De plus, vous pouvez **juger de la qualité de la communauté** en observant **la pertinence des réponses** aux différents sujets. 5

Pour jauger et comparer la popularité de plusieurs sujets, il existe des statistiques publiées par des développeurs(13), des communautés(14) ou bien des applications pour visualiser les recherches effectuées sur les différents moteurs de recherche comme Google Trends(15).

Retour d'expérience de vos pairs

La perception d'autres développeurs sur une bibliothèque ou un framework **pourrait vous aider à vous décider**. Certains publient des radars tels que Thoughtworks(16) pour partager les technologies qu'ils recommandent ou non. Les présentations et retours d'expériences pendant des conférences

(11) *Jeff Whelpley, 2014:* https://medium.com/@jeffwhelpley/screw-you-angular-62b3889fd678
(12) *Spec-ulation Keynote - Rich Hickey:* https://www.youtube.com/watch?v=oyLBGkS5lCk
(13) https://javascriptreport.com/the-ultimate-guide-to-javascript-frameworks/
(14) https://insights.stackoverflow.com/survey/2018
(15) https://trends.google.fr/trends/
(16) https://www.thoughtworks.com/radar

comme NgEurope ou bien pendant des meetups sont des moments riches qui vous permettront de peser leurs avantages et inconvénients.

Toutefois, **un projet utilisé, reconnu et maintenu par des boîtes influentes** (ex : Google, Netflix…) assure un produit certes performant et qui fonctionne, mais **ne garantit pas toujours la stabilité**. Comme exemple, nous pouvons encore une fois citer la transition d'AngularJS vers Angular, qui a été pour la plupart de la communauté une migration douloureuse.

Testabilité

Une application facilement testable permettra de vous faire un avis rapide sur la qualité de vos développements **et renforcera votre confiance** dans votre capacité à répondre au besoin exprimé. Idéalement, une dépendance externe ne devrait pas nécessiter beaucoup de code dans l'élaboration d'un test métier. En outre, il ne faut pas qu'il dégrade la maîtrise et l'observabilité de votre application (Robert V. Binder(17)).

La maîtrise représente votre capacité à contrôler le code et son comportement. Si l'intégration de votre bibliothèque la diminue, votre cycle de développement en souffrira car développer de nouvelles fonctionnalités par la suite sera de plus en plus fastidieux et donc coûteux. Cela peut venir de nouvelles contraintes imposées par la bibliothèque ou bien de sa mauvaise configuration dans vos tests.

Aussi, évitez **d'englober la bibliothèque dans tous vos tests sous peine de ralentir vos cycles de développement**. Si c'est le cas, c'est que vos tests n'ont pas été implémentés au bon niveau. Par exemple, **les tests** d'intégration **englobant** un framework tel que **Spring Boot** diminue fortement votre maîtrise du code car la gestion des composants n'est plus à votre main ! De plus, le framework décide même à votre place l'ajout ou non de fonctionnalités selon certains paramètres ou configurations. Votre code devient **plus difficile à prédire** et **vos tests deviennent plus lents et plus complexes**. Les tests d'intégration sont toutefois nécessaires afin de s'assurer du bon fonctionnement. Cependant leur nombre doit absolument être limité (cf. la pyramide de test(18)) ! Nous pouvons citer comme bonne pratique le principe FIRST(19).

L'autre axe de la testabilité est **l'observabilité, qui est la capacité à déterminer que votre code se comporte bien de la manière attendue**. Un test doit échouer pour les bonnes raisons lorsqu'on l'exécute.

Pour illustrer ce fait, nous pouvons penser **aux tests** de chaînes automatisés **englobant plusieurs applications à la fois**. Ce sont des tests métiers de très haut niveau intégrant de facto beaucoup de composantes techniques et fonctionnelles. Il devient difficile de garantir le bon comportement du système avec uniquement ce genre de tests car il **inclut beaucoup d'effets de bord. Cela va diminuer votre capacité à observer le comportement** et à en déduire son exactitude, et éventuellement à identifier l'origine de certains dysfonctionnements. Par exemple, pouvez-vous assurer, dans une architecture micro-service(20), le bon comportement de votre plateforme lorsque le client passe une nouvelle commande sur un site e-commerce ?

Documentation

La documentation est souvent le point d'entrée pour aborder une bibliothèque ou un framework. Elle doit expliciter **les concepts clés de manière claire** afin de comprendre rapidement quels problèmes la bibliothèque ou le framework peut vous aider à résoudre.

Une documentation **complète** devra expliciter tous les éléments disponibles dans l'API publique ainsi que tous les éléments permettant d'interagir avec elle. De plus, elle doit expliciter pourquoi ces objets sont nécessaires et comment les utiliser pour arriver à vos fins. Elle doit éviter au maximum les étapes implicites en référant d'autres articles qui détaillent déjà ce sujet.

Elle doit aussi être **concise** et doit éviter d'expliciter toute la partie privée de l'API. Un utilisateur final n'a pas besoin de connaître tous les rouages internes de ses fonctionnalités. Les informations non indispensables à la compréhension ne doivent pas apparaître pour éviter de perdre les utilisateurs. De la même manière qu'il est une bonne pratique d'appliquer le principe DRY(21) dans le développement logiciel, la documentation doit éviter au maximum la redondance d'informations. Souvent, un bon exemple est beaucoup plus parlant qu'un long texte d'explications ; une documentation illustrée ou ayant une partie interactive atteindra plus facilement cet objectif. Par exemple, MDN(22) inclut depuis récemment des exemples dans sa documentation. Aussi, le site Haskell(23) donne un accès direct à un REPL(24) pour tester des fonctionnalités ou le comportement du langage directement sur le site.

La documentation doit aussi être bien **organisée** pour permettre à l'utilisateur de comprendre le chemin à suivre en fonction de son besoin. Elle doit guider le novice en lui apprenant à utiliser la bibliothèque dans les plus simples use-cases. Par ailleurs, elle doit être efficace pour permettre aux confirmés de trouver rapidement le point de détail qui leur manque. Une interface bien organisée et suivant les principaux standards(25), comme le fait d'avoir des sections facilement identifiables ("Getting started", "Advanced tutorial"…), ou encore une fonctionnalité de recherche, aide beaucoup.

Par ailleurs, elle doit être **à jour**. Rien n'est plus frustrant qu'une documentation qui ne correspond pas à la version de la bibliothèque qu'on utilise. Le principe de living documentation aide énormément dans cette optique puisqu'elle liste toutes les fonctionnalités qui ont été automatiquement vérifiées au moment de la release(26). Les tests d'acceptance sous forme d'exemples (issus de la pratique BDD(27)) ou Swagger sont de très bons outils pour y parvenir. Côté frontend, l'approche "visual TDD"(28) permet de générer une living documentation très efficace pour des composants visuels et interactifs.

Enfin, la documentation doit être facilement **accessible**. De plus en plus de développeurs écrivent la documentation de leur bibliothèque grâce à des outils comme Github pages(29) ou sur des sites comme readthedocs(30). Devoir télécharger la

(17) http://robertvbinder.com/software-testability-part-1-what-is-it/
(18) https://martinfowler.com/articles/practical-test-pyramid.html
(19) https://github.com/ghsukumar/SFDC_Best_Practices/wiki/F.I.R.S.T-Principles-of-Unit-Testing
(20) https://samnewman.io/talks/principles-of-microservices/
(21) *Don't repeat yourself*
(22) https://developer.mozilla.org/fr/docs/Web/CSS/transform-function/rotate3d
(23) https://www.haskell.org/
(24) *Read–eval–print loop*
(25) https://guides.github.com/features/wikis/
(26) *Présentation living documentation, Cyril Martraire, bdxio 2015*
(27) *Behavior Driven Developpement* http://www.arolla.fr/blog/2012/06/bdd-c-est-quoi-donc/
(28) https://toucantoco.com/en/tech-blog/tech/visual-tdd
(29) https://pages.github.com/
(30) https://readthedocs.org/

documentation chez soi augmentera le risque de désynchronisation avec l'existant. A l'inverse, une documentation hors ligne(31) peut être avantageux car non dépendant de la stabilité d'un site. Le meilleur des deux mondes serait donc une documentation servie via une PWA(32).
Une documentation qui respecte correctement ces principes sera tout à fait acceptable. Il ne faut pas oublier que la plupart des outils open-source ont une documentation qui va **s'améliorer avec le temps** puisque c'est une partie où il est très facile de contribuer. D'autre part, des sites comme Stack Overflow ou des articles de blogs disponibles sur internet permettent souvent de compenser une documentation qui ne respecte pas bien tous les points cités.

Enseignabilité

La communication est clé dans le développement logiciel, que cela soit à travers le code ou encore oralement. En tant que développeur, il vous est sûrement arrivé d'expliquer un code ou un concept lors d'une revue ou d'une présentation. Dès lors, si on parle d'une bibliothèque ou d'**un framework que l'on maîtrise à peine**, il devient extrêmement **difficile de transmettre notre savoir**.
L'une des manières de faciliter la transmission du savoir, outre la documentation, est de comprendre les fondements de la bibliothèque, autrement dit ses concepts clés et ses patterns. En effet, les bibliothèques ou frameworks sont souvent basés sur des patterns existants. Par exemple, la "programmation réactive" est principalement fondée sur des patterns tels que le pattern Observer. Nous pouvons prendre Spring comme exemple aussi car il se base sur de l'AOP(33) qui est un design pattern Decorator.
Dès lors, il est beaucoup **plus facile de comprendre le fonctionnement d'une bibliothèque si on maîtrise au préalable ses patterns**.
Une autre manière est de **passer par des exemples**. La présence de **tutoriels** ou de tests clairs peut aussi vous aider à illustrer vos propos. Souvent, des exemples d'utilisation sont même accessibles dans le README d'un projet Github !
Enfin, si vous devez créer un modèle mental pour devoir expliquer le fonctionnement d'une bibliothèque, imaginez ce que devraient faire les développeurs qui vous succéderont ! Si tel est le cas, cela revient à introduire **un élément** tellement **complexe** qu'il **engendrera inévitablement une gêne, voire une dette**, pour votre projet. Nous pouvons citer comme exemple Hibernate, un ORM(34) disponible en Java. Malgré sa simplicité d'utilisation, il est extrêmement complexe d'en expliquer le fonctionnement !

Tooling

Certaines bibliothèques apportent tout un écosystème d'outils avec elles. En effet, **une bibliothèque qui se concentre uniquement sur un point précis cherchera à être plus flexible** et fonctionnera avec un système de plugins ou d'extensions. Il faut donc prendre en compte tous ces outils dans votre évaluation et non pas uniquement la bibliothèque que vous souhaitez introduire.
Les outils gravitant autour de votre bibliothèque peuvent ajouter de nouvelles fonctionnalités qui n'étaient pas prévues initialement. Par exemple, NgRx apporte le pattern flux dans l'écosystème Angular. **Cela permet de rendre l'outil plus flexible pour qu'il s'adapte à de nouveaux besoins.** Ce principe (Open-Close Principle issu de SOLID(35)) permet l'ajout ou la modification de fonctionnalités et est plus facilement open-sourçable ! D'autre part, une bibliothèque pensée pour fonctionner avec des plugins, arrivera plus facilement en phase stable. ExpressJS suit ce principe et fonctionne avec des middlewares qui augmentent ses capacités, il est aujourd'hui dans un état très stable et évolue relativement peu.
Certains outils fonctionnent extrêmement bien en coordination avec d'autres. Elasticsearch, à la base un moteur d'indexation, peut fonctionner en coordination avec Logstash et Kibana, qui permettent d'en faire un outil d'exploration des logs applicatifs très efficace. De même, Angular-material fait le lien entre Angular et Material design. Ces liens peuvent vous permettre **d'intégrer plus facilement un outil dans un écosystème** et donc potentiellement dans votre legacy.
Conjointement avec la bibliothèque, certains outils **vous rendent plus productif** ou améliorent grandement votre expérience de développement. Redux DevTools(36) vous permet de naviguer dans l'historique de votre store Redux, ce qui peut être très efficace pour des phases exploratoires ou d'analyse de bug.
Enfin, certains outils peuvent **résoudre certains problèmes** historiques. Par exemple, Angular Universal résout les problèmes liés au SEO(37) qui étaient soulevés par les premières SPA(38).
Attention, cependant, une bibliothèque avec beaucoup de dépendances implique potentiellement des complications dans les montées de versions (cf. section Couplage), surtout lorsqu'elles ne sont pas encore stables (cf. section Complétion). **Il faut donc rester vigilant quant aux dépendances que vous souhaitez introduire dans votre projet, et celles-ci doivent se justifier**.

Efficacité

Pour cet indicateur, nous allons nous concentrer sur **deux types d'efficacité** qui font sens dans notre contexte : l'efficacité **technique** et l'efficacité **développeur**.
L'**efficacité technique**, la plus naturelle, fait référence à **tous les aspects techniques** d'une bibliothèque dans l'accomplissement d'un besoin. Ce type d'efficacité peut être mesuré sous différentes formes :

- **Le temps de traitement** qui portera sur le temps nécessaire à notre code à produire un résultat(39) ;
- **La consommation en ressources** qui mesure l'allocation de ressources nécessaires au traitement. Parmi ces ressources, nous pouvons compter la consommation en mémoire, en CPU, en stockage ou encore la consommation réseau ;
- **La fiabilité** des résultats produits par rapport à ceux attendus.

De plus, **la mesure de l'efficacité technique** sera **différente selon le besoin**

(31) Un très bon agrégateur de documentations disponible hors ligne : https://devdocs.io/
(32) PWA : Progressive Web Application. https://developers.google.com/web/progressive-web-apps/
(33) Aspect Oriented Programming (ou Programmation Orienté Aspect) qui traite des préoccupations techniques autour d'un code métier. Souvent utilisé dans l'inversion de contrôle.
(34) Object Relational Mapping (Mapping Objet-Relationnel) permet de passer d'une modélisation relationnelle (Oracle par exemple) à une modélisation objet.
(35) Principe SOLID http://www.arolla.fr/blog/2017/02/principes-solid-vie-de-jours/
(36) https://github.com/gaearon/redux-devtools
(37) SEO: Search Engine Optimisation
(38) SPA: Single Page Application
(39) Exemple avec les moteurs de rendu : JS framework benchmark, Stefan Krause

auquel la bibliothèque ou le framework répond. Par exemple, la mesure de l'efficacité technique sera différente entre un framework de rendu type Angular et un broker de message comme Kafka.

L'**efficacité développeur** est l'autre critère à prendre en compte. Pour cela, observons **la manière dont le développeur utilise la bibliothèque et ses concepts avec plus ou moins de facilité**. Ce genre de performance n'est surtout pas à négliger car elle déterminera à coup sûr l'adoption d'une bibliothèque ou d'un framework plutôt qu'un autre. Un code difficilement utilisable, comme une application mobile mal conçue, ne perdure pas très longtemps.

Afin de **mesurer** l'efficacité développeur, nous pouvons retenir **le temps d'application** entre le moment où l'on se documente et la mise en pratique. Nous pouvons aussi parler de l'**ensemble des fonctionnalités** fournies par la bibliothèque et **réellement utilisées** dans notre code. Si vous remarquez qu'une bibliothèque est mal utilisée par rapport au besoin, c'est sans doute que son utilisation n'est pas assez intuitive. Et cela est d'autant plus vrai quand nous parlons d'un framework !

Qualité du code

Un code de qualité se doit, à l'instar d'un livre, de raconter une histoire et d'exprimer une intention de façon claire et compréhensive. Cela vaut à la fois pour le code de votre application mais aussi pour vos dépendances, car **ne pas comprendre le fonctionnement d'une dépendance revient à faire rentrer un inconnu chez soi** !

Par exemple, en cas de problème, il devient nécessaire de pouvoir débugger le code exécuté, en comprenant les dépendances. **Assurez-vous donc de la qualité globale de votre code et de celui de vos dépendances, de préférence, avant qu'un problème survienne**. Pour cela, nous sommes particulièrement soucieux de la structure, la lisibilité, le nommage des concepts et les tests inhérents au code.

Un code mal structuré peut ralentir la recherche d'information car naviguer à l'intérieur ne sera pas intuitif.

Un code pas clair aura pour effet systématique de **renvoyer le développeur à la communauté**, alors qu'il aurait pu trouver les réponses lui-même.

De plus, **la qualité du code** est aussi tirée **par le nommage des concepts**. Il est très important de lire la documentation afin de cerner les différents concepts. Ils devraient être facilement identifiables dans le code. Sans cela, vous aurez une compréhension partielle ou biaisée…

Enfin, dans une démarche purement TDD(40), **des tests décrivant le comportement** de la bibliothèque **aident à la compréhension,** mais aussi **à la documentation et à l'utilisation**. Personnellement, il nous arrive de parcourir les tests pour avoir un exemple d'implémentation d'une fonctionnalité de la bibliothèque. Nous sommes profondément convaincus que des tests bien écrits représentent une grande source de documentation et cela nous donne confiance dans la bibliothèque.

Mise en situation

Pour illustrer notre démarche, **imaginons que nous travaillons pour un nouveau projet au sein d'une grande entreprise ayant plusieurs équipes de développement**. Sur le marché actuel, plusieurs solutions s'offrent à nous concernant un moteur de rendu pour la partie front-end. Voici un exemple d'interview que vous pouvez proposer à votre équipe ou à vous-même (en cas de schizophrénie) afin de vous poser les bonnes questions :

- Est-ce que vous avez déjà une idée de la solution que vous souhaitez adopter ?

"Non, pas du tout ! Je t'avoue que je suis un peu perdu dans tous ces choix… On m'a parlé d'Angular et de VueJS"

- Est-ce que vous faites ce choix par rapport à ses atouts et à la valeur ajoutée dans le cadre du projet ?

"Par professionnalisme, ce choix sera fait en mettant l'intérêt du client au cœur de notre décision."

Contraintes fortes

- Avez-vous des contraintes liées à votre organisation (frameworks restreints par l'entreprise, bibliothèques internes imposées…) ?

"Il n'y a pas de frameworks internes Javascript, nous avons la liberté de choisir tant que nous pouvons expliquer notre décision."

- Avez-vous un legacy, un langage ou un framework particulier avec lequel vous devez être la compatible ?

"Nous avons un legacy mais pour ce nouveau projet, nous créons une application séparée. Nous n'avons donc pas de langage de programmation imposé. "

A ce stade, nous avons l'embarras du choix(41) : React, Angular, VueJS, Reframe, Aurelia, Elm… Devant toutes ces possibilités, il ne doit en rester qu'une ! Afin de filtrer tous ces choix, d'autres questions se posent sur les aspects structurels, culturels et organisationnels :

Structure de l'entreprise

- Avez-vous un département dans votre structure qui filtre les dépendances que vous pouvez utiliser ?

"Nous avons un département sécurité au niveau de l'IT. Nous n'avons pas accès à toutes les dépendances et avons des dépôts internes. Cependant, nous pouvons faire des demandes d'ajout en attestant de la maturité de la dépendance."

- Est-ce que vous avez des contraintes d'homogénéité technique ?

"Nous avons un pôle d'architectes qui veille à l'homogénéité technologique sur l'ensemble de l'entreprise. Cependant, ce choix en particulier reste local à notre projet."

Conclusion : étant donné la structure de l'entreprise, une solution stable (*indicateur de complétion*) est à envisager. Pas de couplage avec l'existant, pas de problématique d'homogénéité.

Culture de l'entreprise

- L'expérimentation est-elle possible à travers des SPIKE ou des POC(42) ?

"Oui, ce sont des pratiques qui font partie de la culture de l'entreprise, mais ils doivent respecter une limite de temps qui doit être acceptable pour le client."

- Est-ce que vous avez le droit à l'erreur (possibilité de rollback) ?

"Oui, on peut se tromper mais il faut s'en apercevoir rapidement ! Il sera difficile d'expliquer à nos clients, un mois plus tard, que nous nous sommes trompés."

- Est-ce que la qualité des applications est intégrée à la culture de l'entreprise ?

"Oui, nous avons besoin de respecter un certain niveau de qualité car nos clients ont vécu de mauvaises expériences et possèdent de nombreuses applications mal conçues et inmaintenables."

(40) Test Driven Development
(41) https://javascriptreport.com/the-ultimate-guide-to-javascript-frameworks/
(42) https://medium.com/studio-zero/spikes-pocs-prototypes-and-the-mvp-5cdffa1b7367

Est-ce que vos collègues sont expérimentés et ouverts d'esprit ?
"Oui, nous sommes ouverts d'esprit mais la plupart des développeurs sont juniors."

Conclusion : nous recommandons à ce stade une bibliothèque/framework facile à prendre en main (*indicateur Efficacité*) pour adresser les contraintes de temps. La notion de qualité ressort des réponses (*indicateur Qualité*) basées sur le retour des clients sur d'anciens produits. Et au vu de la maturité globale des développeurs, il serait judicieux de sélectionner une solution dont les concepts sont facilement transmissibles (*indicateur Enseignabilité*).

Organisation de l'entreprise

- Etes-vous nombreux à travailler sur ce projet ?

"Pas dans l'immédiat, nous ne serons que 5, mais d'autres équipes pourraient travailler sur notre application plus tard, à savoir plus de 50 développeurs."

- Ce choix aura-t-il un impact plutôt local ou deviendra-t-il un standard ?

"Seules les équipes travaillant sur ce sujet devraient être impactées."

- Avez-vous beaucoup de "turn-over" ?

"Oui, le marché actuel du recrutement de développeurs est en plein essor. Les développeurs vont et viennent fréquemment !"

- Avez-vous des contraintes de temps ?

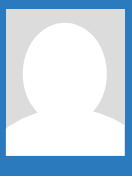
"Oui, le métier attend beaucoup de nous et veut voir des résultats très rapidement."
Avez-vous des contraintes de coût ?
"Non, pas dans l'immédiat."

Conclusion : Compte tenu du périmètre d'application de la solution, une bonne documentation sera nécessaire afin d'intégrer beaucoup de développeurs (*indicateur Documentation*). Enfin, concernant le turnover, il sera plus facile de recruter de nouveaux développeurs en misant sur une solution populaire (*indicateur Popularité*). Encore une fois, les contraintes de temps insistent sur la nécessité d'une solution rapide à mettre en œuvre (*indicateur Efficacité*). Nous pouvons donc statuer sur un choix. L'indicateur qui se démarque le plus est l'*Efficacité*. Notre recommandation portera donc sur un framework, qui impose des règles strictes au profit d'un temps de démarrage moins long et d'une prise en main plus facile, plutôt qu'une bibliothèque. Ensuite, l'indicateur de *Popularité* va privilégier une solution répandue avec une bonne communauté. Etant donné la structure, un framework stable est un prérequis. Enfin, au vu de la portée de la solution, une bonne documentation de base sera nécessaire afin de former le plus grand nombre de développeurs possible.
Dans ce contexte, notre recommandation pourrait porter sur *Angular* parce que c'est un framework populaire, simple d'utilisation et bien documenté. Il guidera la majorité des développeurs avec des choix prédéfinis en termes de stack. React et VueJS nécessitent de construire leur stack de départ et une bonne connaissance en développement d'application web est requise pour ne pas faire d'erreur de design. Elm ou Reframe sont d'excellentes alternatives centrées fonctionnelles, mais nécessitent une connaissance de leurs langages respectifs qui est aujourd'hui peu commune sur le marché.
Et vous qui lisez ces lignes, quelle serait votre recommandation ?

Le mot de la fin

Avant de conclure, nous voulons insister sur le rôle du **développeur logiciel**. Notre métier est avant tout de **résoudre des problématiques métiers**. Si vos choix entraînent plus de problèmes qu'ils n'en résolvent, c'est sûrement dû à un manque de pragmatisme vis-à-vis du métier. Rappelez-vous que dans la majorité des cas, les besoins métier et les besoins techniques sont décorrélés et changent indépendamment de l'autre. Distinguez bien ces deux aspects et posez-vous la question : **"Est ce que j'en ai vraiment besoin ?"**
Vos choix auront un impact sur les autres (les autres, c'est aussi vous dans le futur !). **Collaborez avec vos pairs** pour construire un argumentaire solide qui **justifie vos décisions et les factualisent** dans des ADR(43) par exemple. N'hésitez pas à challenger les solutions déjà en place pour vous **adapter à l'évolution du besoin de vos utilisateurs**.
Si nécessaire, **expérimentez** autant que possible afin d'en tirer des conclusions. Partagez-les ensuite avec votre équipe car cela forgera votre expérience et l'expertise globale.
Et si vos décisions n'ont pas été les bonnes, **capitalisez** dessus. Car comme on le dit souvent : "on apprend en marchant ! "
Enfin, ***"There is no silver Bullet !"*** *(44)*. Chaque solution comporte son lot d'inconvénients. Explicitez systématiquement les limites de votre solution autant que la valeur qu'elle apporte, sous peine d'avoir une approche commerciale plutôt que pragmatique. Par éthique et professionnalisme, défendez avant tout l'intérêt de votre client plutôt que votre désir d'essayer des technologies afin de remplir votre CV ! •

(43) Architecture Decision Record: Documenting architecture decision, Mickael Nygard
(44) 'There is no silver bullet' de Fred Brooks



Karim Boutouil
Apprenti Développeur chez ALD Internationnal
Etudiant à Paris 1 - Panthéon Sorbonne

Jean-Baptiste Da Costa
Apprenti Développeur chez ALD Internationnal
Etudiant à ITESCIA

Gabriel Hubert
Stagiaire chez ALD Internationnal
Etudiant à l'école 42 Paris

Projet Raspberry Pi DIY : the RasPinger

Le RasPinger est un système connecté utilisant la Raspberry pour déclencher un signal audio lorsqu'un membre d'une équipe de développeurs réalise un Push sur un gestionnaire de code dédié (GIT ou TFS).

Pour réaliser ce projet nous nous sommes d'abord penché sur son fonctionnement global et nous en avons déduit deux parties distinctes.

La première partie concerne le service web qui va permettre de récupérer les notifications envoyées par le gestionnaire de code lors d'un push sur celui-ci et de notifier les clients qui se seront inscrits sur le service pour qu'ils puissent jouer un son à ce moment.

La seconde partie concerne la configuration du client Raspberry sur laquelle il fallait implémenter un script qui s'inscrit sur le service web pour que celui-ci puisse le notifier lorsqu'un push aura été réalisé sur le gestionnaire de code. 1

Nous allons maintenant expliquer plus en détail le fonctionnement des deux parties du projet et vous indiquer la marche à suivre pour reproduire ce projet.

PARTIE SERVEUR (API)

Développement de l'API

Nous avons fait le choix de réaliser cette API en C# / ASP.NET MVC.

Nous allons tout d'abord créer le projet. Pour ce faire, rendez-vous dans Visual Studio, dans l'onglet Fichier => Nouveau => Projet.

Une nouvelle fenêtre s'ouvre : allez dans Visual C# => Web et cliquez sur ASP.NET Web Application (.NET Framework). Il est aussi possible de le faire en ASP.NET Core Web Application (avec peut être quelques remaniements). 2

Pour que l'ensemble des clients reçoivent la notification d'un Push, on a choisi d'utiliser SignalR. Pour cela on ajoute un dossier dans le projet (clic droit sur le nom du projet => Ajouter => Ajouter un dossier) on le nomme SignalR.

On y ajoute un fichier à l'intérieur :

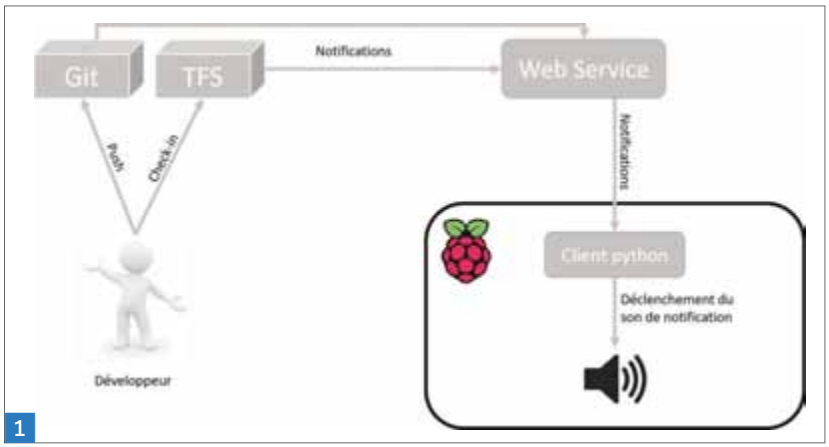
Ajouter => Ajouter un nouvel élément, dans la nouvelle fenêtre Visual C# => Web => SignalR => SignalR Hub Class (v2)

Nommez le comme vous voulez, nous l'avons nommé "HubPinger". 3

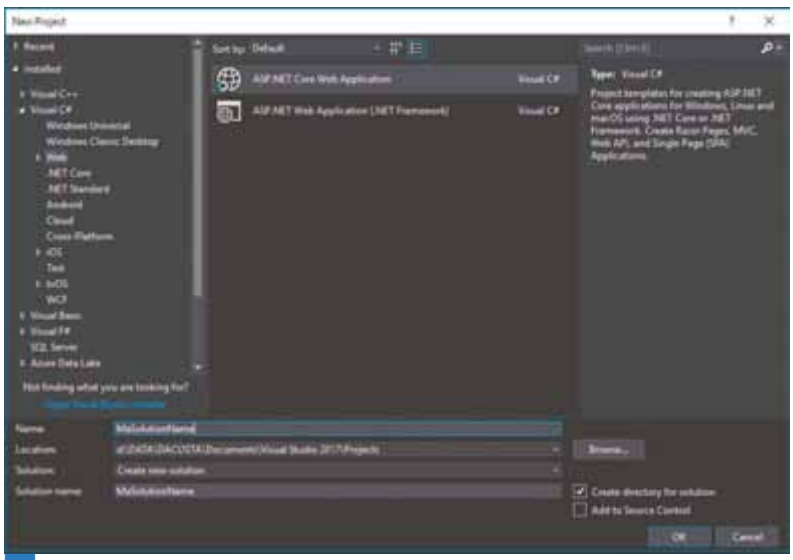
Comme vous pouvez le voir cette classe hérite de la classe Hub ce qui nous permet de créer le "SignalR Hub".

Au dessus de la classe on ajoute *[HubName("Push")]* (de cette manière, les clients pourront choisir sur quel hub "écouter"), puis on modifie la méthode Hello comme ci-dessous :

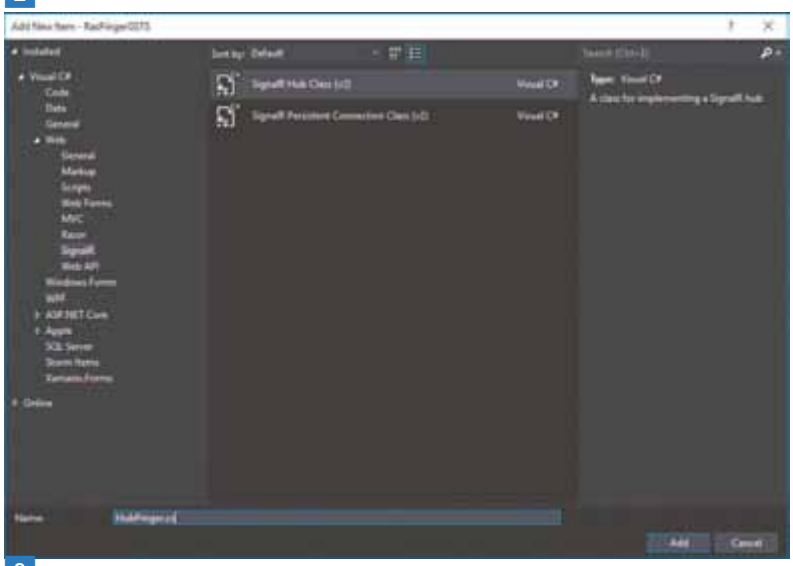
```
using System;
using System.Collections.Generic;
using System.Linq;
```



1



2



3

```
using System.Web;
using Microsoft.AspNet.SignalR;
using Microsoft.AspNet.SignalR.Hubs;

namespace RasPingerGSTS.SignalR
{
  [HubName("Push")]
  public class HubPinger : Hub
  {
    internal static void HelloServer()
    {
      IHubContext context =
GlobalHost.ConnectionManager.GetHubContext<HubPinger>();
      context.Clients.All.Response("SignalR Response");
    }
  }
}
```

Puis dans le fichier Startup (s'il n'a pas été créé dans votre Solution, ajoutez-le comme une classe avec le nom Startup.cs et modifiez le comme indiqué sur la photo ci-dessous).
Dans la méthode Configuration on ajoute la ligne suivante :

```
app.MapSignalR("/signalr", new HubConfiguration());
```

Note

"/signalr" étant la route à utiliser dans le Raspberry pour se connecter au hub (nous reviendrons sur cette partie par la suite dans cet article).

```
using Microsoft.AspNet.SignalR;
using Microsoft.Owin;
using Owin;

[assembly: OwinStartupAttribute(typeof(RasPingerGSTS.Startup))]
namespace RasPingerGSTS
{
  public partial class Startup
  {
    public void Configuration(IAppBuilder app)
    {
      //ConfigureAuth(app);

      //Enables SignalR
      app.MapSignalR("/signalr", new HubConfiguration());
    }
  }
}
```

Maintenant, nous allons voir les méthodes d'entrée de notre API, une fois qu'un push sera fait sur le gestionnaire de sources celui-ci effectuera une requête HTTP sur la méthode correspondante. Cette action initialisera une réponse du Hub vers tous les clients inscrits.
Dans le dossier "Controllers", clic droit sur Ajouter => Controller => Web API 2 Controller Empty.
On y configure une route qui permettra de récupérer une requête en méthode POST initialisées par le gestionnaire de sources.
Pour savoir de quel gestionnaire de sources provient la requête, on passera en paramètre le nom du gestionnaire de sources d'où provient la requête grâce à la variable FROM.
L'url entrée dans les WebHooks de GIT ou TFS ressemblera donc à : **http://{url}/api/post/gsts?FROM=GIT** (pour les événements provenant de GIT et TFS pour ceux provenant de TFS).
A l'intérieur, on appelle le hub: *HubPinger.HelloServeur()* à configurer comme ci-dessous.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using RasPingerGSTS.SignalR;

namespace RasPingerGSTS.Controllers
{
  public class GSTSController : ApiController
  {
    // POST: GSTS
    [Route("api/post/gsts")]
    public IHttpActionResult PostGSTSEvent([FromBody]dynamic
data, string FROM)
    {
      //Ici on définit les actions à effectuer en fonction du
gestionnaire de source dont provient la requête.
      //A noter qu'il est possible par ce biais d'effectuer
des actions différentes suivant la source de la requête.
      //L'Url passée au webhook devra donc ressembler à :
      //"http://{URL}/api/post/gsts?FROM=(GIT OU TFS suivant
le gestionnaire de sources)"
      if (FROM == "GIT")
      {
        HubPinger.HelloServer();
      }
      else if (FROM == "TFS")
      {
        HubPinger.HelloServer();
      }
      else if (FROM == null)
      {
        return NotFound();
      }
      else
      {
        return Unauthorized();
      }
      return Ok();
    }
  }
}
```

Maintenant nous allons tester notre Hub, dans le dossier SignalR, on ajoute un fichier JS: Visual C# => Web => JavaScript File, on le nomme "test.js".
On crée une variable qui stocke le HubName (on la nomme ici "HubPinger"), puis une fonction qui, une fois la réponse du serveur reçue, affichera la réponse sur la page web.

Enfin, on initialise la connection avec "$.connection.hub.start()".

```
var HubPinger = $.connection.Push;

// Créer une fonction que le Hub peut appeler.
HubPinger.client.Response = function (message) {

   $("#message").text(message);


};

// Initialiser la connexion.
$.connection.hub.start();
```

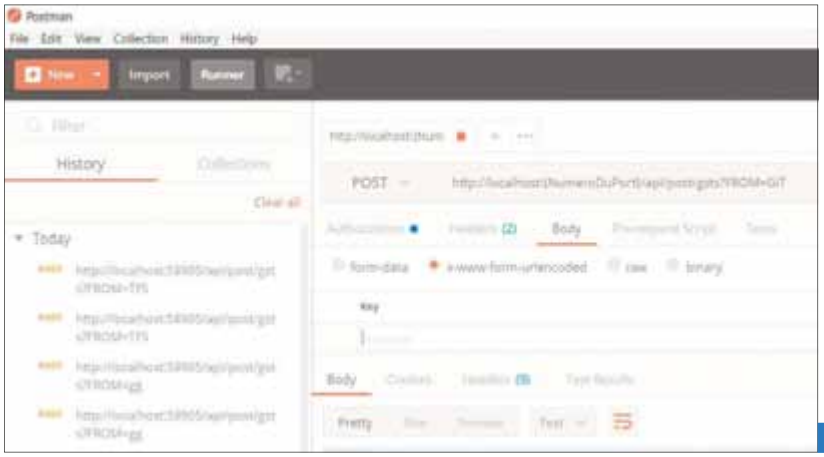
Dans le fichier qui se trouve dans Views => Home => Index.cshtml on ajoute les lignes suivantes dans l'ordre à la fin du fichier :

```
@section scripts{
   <script src="~/Scripts/jquery.signalR-2.1.2.js"></script>
   <script src="~/SignalR/hubs"></script>
   <script src="~/SignalR/test.js"></script>
}

<span id="message"></span>
```

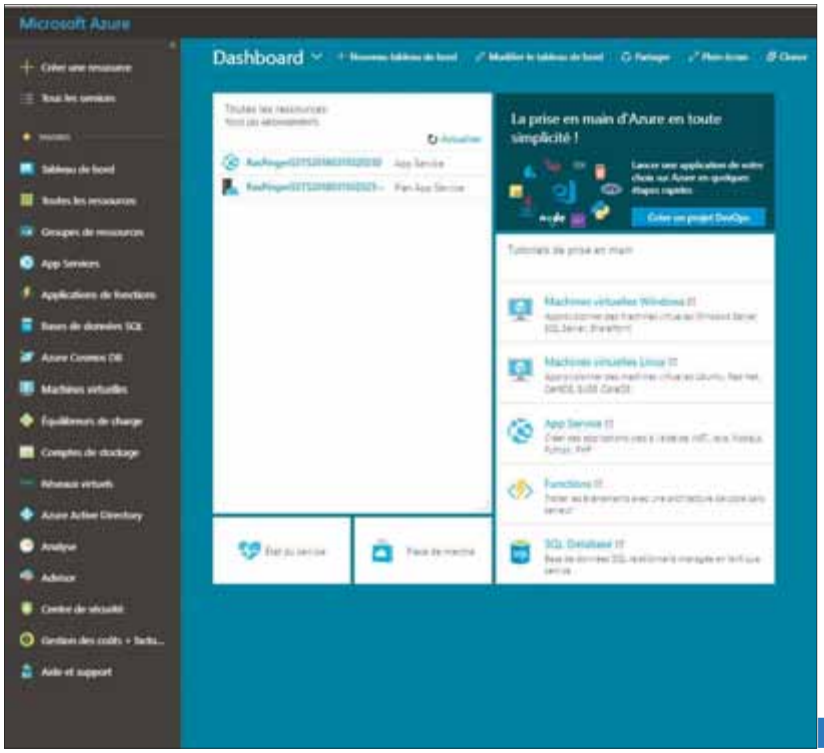
Puis on démarre notre serveur local (la flèche verte en haut de visual studio ou F5). Pour simuler un Push on utilise Postman :
On configure la requête en POST, on lui donne l'url qui se trouve dans le navigateur qui s'est lancé avec notre serveur et on ajoute la route vers la méthode correspondante (/api/post/gsts?FROM=GIT pour GitHub). L'URL devrait ressembler à :
http://localhost:{NumeroDuPort}/api/post/gsts?FROM=GIT 4
Envoyez la requête et vous devriez voir apparaître en bas de la page web le texte SignalR Response.
Bravo votre route et votre hub fonctionnent !

Création et mise en ligne du serveur

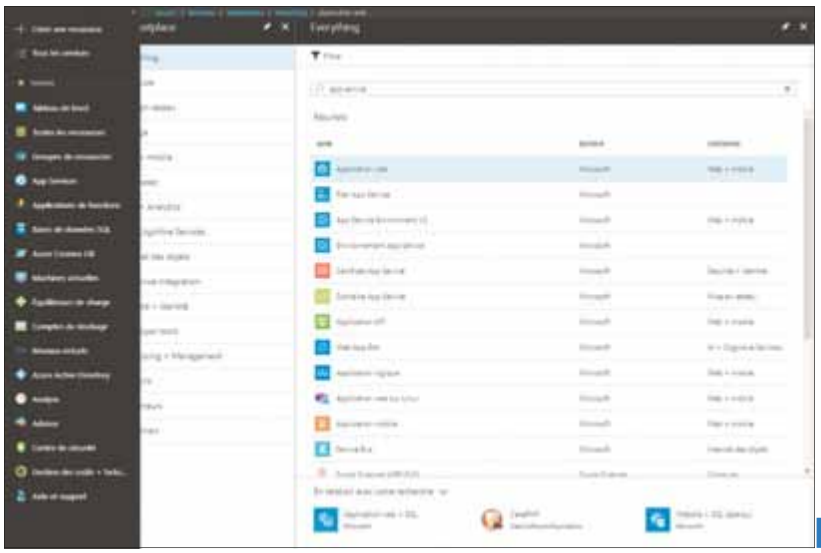
Pour héberger le service web, il vous faut un compte Microsoft Azure sur lequel vous allez devoir créer la ressource correspondante.
Sur la page d'accueil du site https://portal.azure.com, vous pourrez voir vos ressources actives et en créer de nouvelles. 5
Dans l'onglet "Créer une ressource" recherchez "Application web", cliquez sur le bouton "créer" et définissez les options de stockage et le groupe de ressources auquel votre Application appartiendra. 6
Une fois votre ressource créée retournez sur la page d'accueil et actualisez le Dashboard, vous devriez voir apparaître votre nouvelle ressource. 7
De retour sur Visual Studio, dans l'explorateur de solutions: clic droit sur votre application -> Publish. 8
Sur la page qui s'ouvre vous devrez vous connecter à votre compte Azure et sélectionner la ressource que nous avons créée un peu plus tôt. Pour finir, un clic sur Publish et votre ressource sera active et accessible via l'url disponible dans les détails de votre ressource sur le portail Azure. 9



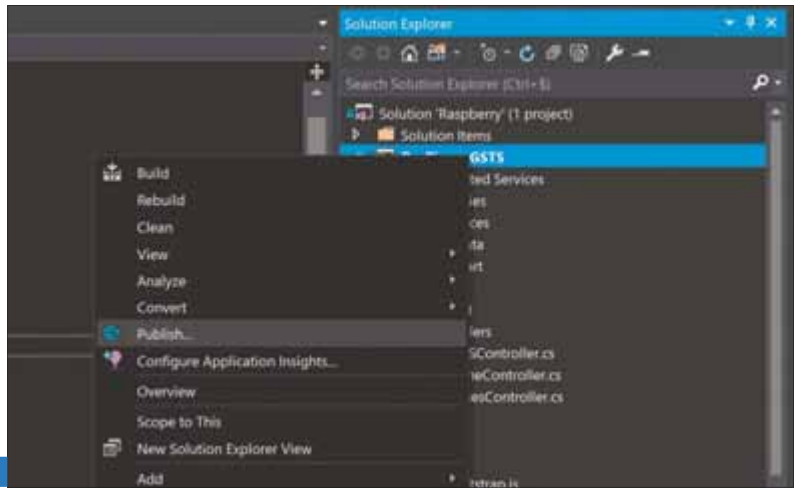
4



5

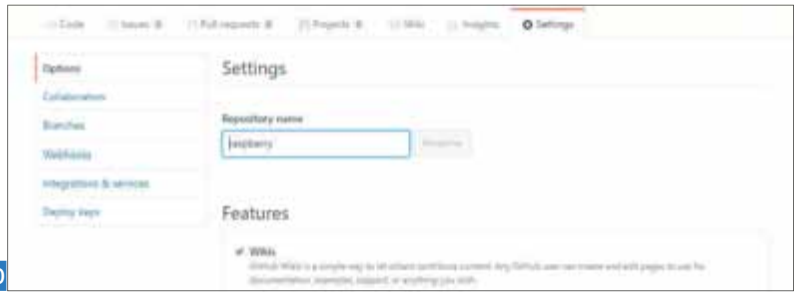

6

7

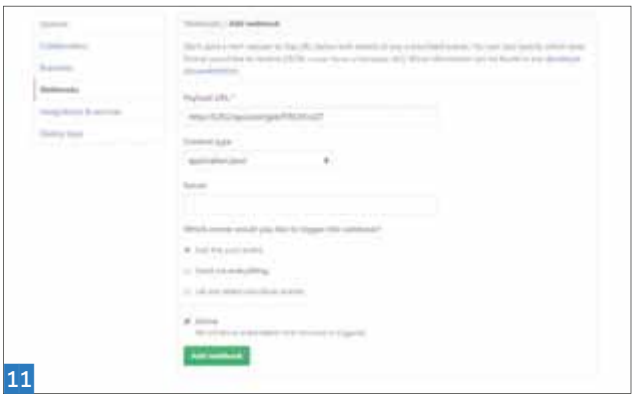


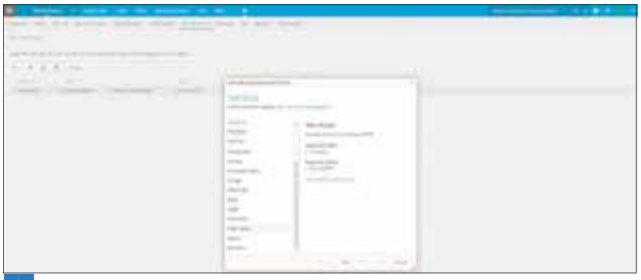
8

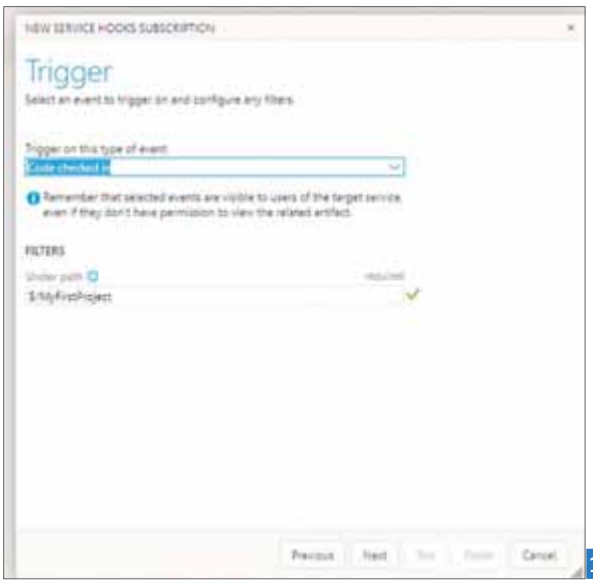

9



10


11


12



13

CONFIGURATION DE GIT ET TFS

Configuration de GitHub

A chaque fois qu'il y aura un push sur notre Git, GitHub devra envoyer une requête sur la méthode correspondante dans l'API, pour celà, rendez-vous dans les paramètres de votre projet sur GitHub. 10

Dans les onglets sur le côté sélectionnez Webhooks, puis add webhooks.

Dans Payload URL, entrez l'URL de votre API suivie de la variable FROM configurée pour GIT (http://{URL}/api/post/gsts?FROM=GIT).

Dans Content type, sélectionnez *application/json* et dans la partie *Which events would you like to trigger this webhook?* cochez *Just the push event* si vous voulez que le Raspberry déclenche un son uniquement lors d'un Push, ou sélectionnez d'autres events comme les *Pull requests* ou les *Forks* dans la partie *Let me select individual events*. Une fois que vous avez configuré votre déclencheur, cliquez sur le bouton "Add webhook").

C'est bon votre Git est configuré pour envoyer des requêtes sur votre API. 11

Configuration du TFS

Dans cette partie, nous allons voir comment configurer l'envoi de notification sur notre service web depuis TFS.

La première étape consiste à aller dans l'onglet *Service hooks* dans TFS, puis de cliquer sur le bouton *ajouter*.

La fenêtre ci-dessous devrait apparaître, choisir la catégorie *Web Hooks*. 12

Ensuite, choisir "code checked in" dans le menu déroulant. De cette manière à chaque check-in de code dans le dépôt TFS, une

notification sera envoyée à l'url que l'on précisera par la suite. 13
Entrez l'URL de votre API suivie de la route vers la méthode TFS (http://{URL}/api/post/gsts?FROM=TFS). Une fois l'url renseignée, appuyer sur le bouton *finish*. 14
Vous devrez voir apparaître votre configuration de notification comme le montre l'image ci-contre. 15

PARTIE CLIENT (RASPBERRY)

Pour la partie client, nous utiliserons un Raspberry PI sur lequel on installera une distribution Linux, Raspbian.

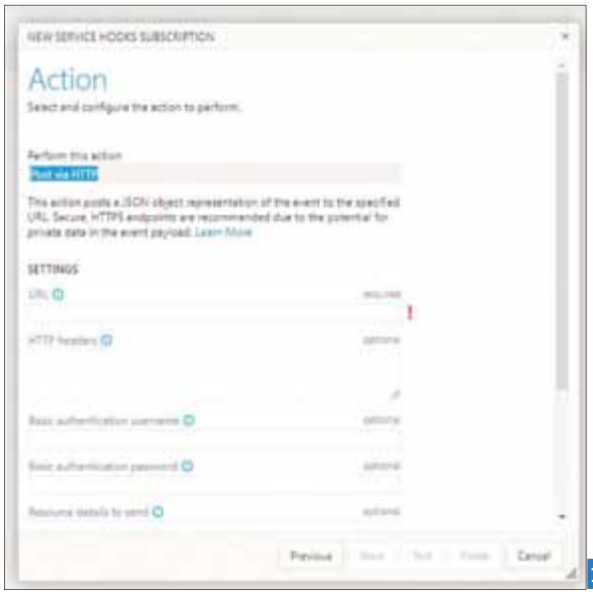
Configuration de Linux

Pour configurer votre Raspberry PI il faut tout d'abord que vous vous munissiez d'une carte micro SD (préférez une carte de 8GB minimum pour être confortable).Sur votre ordinateur, téléchargez le logiciel *Etcher* qui permet de flasher des disques externes et ainsi d'installer des images système sur ceux-ci à partir d'un fichier au format .img.
Une fois que vous avez installé Etcher sur votre poste, rendez vous sur le site www.raspberry.org/downloads/raspbian/ pour télécharger la version de RaspBian que vous voulez installer sur votre Raspberry (je vous conseille la version "RASPBIAN STRETCH WITH DESKTOP", plus complète que la version "LITE").
Après avoir téléchargé le fichier .zip contenant l'image de Raspbian, extrayez l'image de celui-ci et ouvrez Etcher.
Dans Etcher il vous suffit de sélectionner l'image précédemment téléchargée, puis de sélectionner la carte Micro SD sur laquelle vous voulez l'installer et cliquez sur "Flash!".
Une fois le flash terminé vous pouvez éjecter la carte normalement puis vous pourrez l'insérer dans votre Raspberry avant de brancher celui-ci à une alimentation micro USB (cette action devrait faire automatiquement booter le Raspberry sur la carte SD sans autre manipulations de votre part).
Il ne vous reste plus qu'à brancher votre clavier et votre souris USB au Raspberry pour pouvoir commencer à l'utiliser.

Configuration du wifi (optionnel)

Il est possible de connecter le Rapsberry à internet via une connection Ethernet.
Si vous souhaitez vous connecter en wifi, sur la plupart des Raspberry, le module wifi doit être ajouté grâce à un dongle USB. Pour configurer le wifi, il suffit d'aller dans le fichier qui se trouve au chemin suivant */etc/network/interfaces* et de rajouter les lignes de commandes suivantes:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0

iface wlan0 inet dhcp
wpa-ssid "nom du réseau"
wpa-psk "mot de passe du réseau"
```



14


15

Note

Votre carte SD doit être formatée au format "FAT32" pour pouvoir être lue correctement par votre RaspBerry.

Note

Si vous utilisez un clavier AZERTY, vous devrez changer la langue utilisée pour le clavier par RaspBian, vous pouvez changer ce paramètre dans l'onglet préférences -> mouse and keyboard settings -> keyboard -> Keyboard Layout puis sélectionner Français dans le menu déroulant de langues.

Ensuite, après redémarrage du Raspberry, le wifi sera automatiquement connecté.
Tester la connection en exécutant, par exemple, la commande suivante dans le terminal : *ping google.fr*

Installation des librairies

Le client sera développé en Python. Pour cela, il faut installer Python via la ligne de commande suivante : *sudo apt-get install python2.7*
Une fois que Python est bien installé, vérifiez en tapant *python* dans l'interface de commande shell. Si un interpréteur Python s'ouvre, c'est qu'il est bien installé. Nous utiliserons le Raspberry comme écouteur d'évènement. Pour cela, nous aurons besoin de signalR, le tuto complet est disponible ici : http://www.instructables.com/id/The-Raspberry-Pi-Arduino-SignalR-Home-Automation-H/
Après avoir téléchargé le zip à l'adresse ci-dessus. Décompressez le zip dans le même répertoire que votre projet et tapez les commandes suivantes dans l'ordre :

```
pip install gevent
pip install websocket-client
pip install sseclient
pip install requests
```

Ensuite testez dans le terminal, en ouvrant une instance de Python (en tapant tout simplement *python* dans le terminal), que les installations des librairies ont bien été effectuées en tapant les instructions ci-dessous.

```
import gevent
import websocket
import sseclient
import requests
```

Retournez ensuite dans le dossier que vous avez précédemment dézippé et exécutez la commande suivante :

```
sudo python setup.py install
```

Si une erreur apparaît, modifiez le fichier en mettant en commentaire (en faisant commencer les lignes par #).

```
#with io.open(path.join(here, 'DESCRIPTION.rst'), encoding='utf-8') as f:
#    long_description = f.read()
```

Puis en bas du fichier, remplacez la ligne :

```
long_description=long_description
```

par :

```
long_description='DESCRIPTION.rst FILE is missing! This was comment was added manually to replace the missing file!'
```

Partie code

Ouvrez votre éditeur de texte et tapez le code ci-dessous.
/!\ Attention : Le langage Python sépare les blocs d'instructions grâce aux indentations, il est donc fortement déconseillé de faire un copier-coller de code, sous peine d'injecter des caractères invisibles (ça vous épargnera des secondes/minutes/heures/jours de blocage :)) Pour exécuter ce code, taper dans le terminal la commande suivante dans le répertoire où se situe le code : *python client.py*
Laisser le terminal ouvert, faite un check-in ou commit de test et la magie opèrera !

----------client.py----------

```
import os
from requests import Session
signalr import Connection

with Session() as session:
  #create a connection
  connection = Connection("insérer l'url", session)

  #récupération du hub sur lequel l'évènement push est écouté
  pushhub = connection.register_hub('Push')

  #méthode qui va être appelé lorsque qu'un évènement survient
  def print_received_message(data):
    print('received: ', data)
    os.system('omxplayer -o local jingle.mp3 &')

  def print_error(error):
    print('error: ', error)

  #lorsque la réponse du serveur web arrive, on exécute la fonction print_received_message
  pushhub.client.on('Response', print_received_message)

  #on établit la connexion
  connection.start()
  #boucle infinie qui attend les événements provenant du serveur
  while True:
    connection.wait()
```

L'instruction suivante permet d'enregistrer une connection avec le serveur. Cette connection sera établie par la suite

```
connection = Connection("insérer l'url", session)
```

Ensuite cette instruction va enregistrer le hub nommé 'Push'

```
pushhub = connection.register_hub('Push')
```

La méthode suivante sera appelée lorsque le hub nous enverra une réponse. On affichera le contenu de la réponse, c'est ici que nous déclencherons l'appel système pour lire le fichier audio. La commande omxplayer est installée sur les distributions Raspbian.
L'option *-o local* permet de forcer la sortie son sur la prise jack. Autrement, le son sortira via l'HDMI. L'esperluète à la fin de la commande permet de lancer cette commande en tâche de fond.

```
def print_received_message(data):
    print('received: ', data)
    os.system('omxplayer -o local jingle.mp3 &')
```

L'instruction ci-dessous va écouter les messages du hub et fera l'appel à la méthode juste au dessus lorsqu'une réponse est reçue.

```
pushhub.client.on('Response', print_received_message)
```

La connexion est établie par l'instruction qui suit.

```
connection.start()
```

Et enfin, on lance la boucle dans laquelle, tant que l'on a pas de réponse du serveur, on attend.

```
while True:
    connection.wait()
```

Conclusion

Ce projet nous a permis de monter en compétence sur différentes technologies. Cet article est un condensé de ce qu'on a pu faire en une semaine. On a découvert des choses qui nous serons utiles et que l'on réutilisera par la suite.
La démarche de faire cet article a été très enrichissante ; expliquer notre travail nous a permis d'améliorer notre compréhension de ce que l'on apprend, ainsi que notre pédagogie.
Si on devait améliorer notre projet, on ferait en sorte de ne pas avoir de boucle infinie côté Raspberry qui ré-établit la connexion toutes les 30 secondes. Seconde amélioration possible, rendre paramétrable le Raspberry, comme par exemple mettre en place un annuaire qui présenterait les différents dépôts où l'on veut s'abonner, définir quel son sera déclenché suivant les dépôts et/ou suivant les actions sur les dépôt (commit/check-in, merge, création de branche, lancement d'un build, succès d'un build, erreur d'un build, etc.) •

Loïc Devulder,
Senior QA Engineer chez
SUSE

openQA, après la théorie, la pratique !

Partie 3

Après avoir vu comment fonctionne l'outil openQA le mois dernier, nous allons maintenant voir comment le mettre en oeuvre avec un cas concret.

Bon ça y est, on va enfin pouvoir jouer ! Euh non, travailler !

Avant de ~~jouer~~ travailler, il nous faut tout d'abord un OS sur lequel installer notre outil **ScummVM**. Il est tout à fait possible de récupérer une image déjà générée par https://openqa.opensuse.org, mais comme c'est finalement très facile à faire autant voir cette partie-là également. La première question qui peut venir à l'esprit de certains lecteurs c'est pourquoi Xfce et pas GNOME ou KDE ? La réponse est simple : c'est parce que je maîtrise beaucoup mieux cet environnement que les deux autres ! Et comme on peut le choisir lors de l'installation d'openSUSE je l'ai tout naturellement utilisé. Mais rien ne vous empêche de tester avec GNOME/KDE et faire les adaptations nécessaires dans les tests si besoin (mais je vous conseille de le faire après avoir vu ce que ça donne).
Je ne rentrerai pas dans le détail pour expliquer comment fonctionne le test d'installation car cela prendrait trop de temps, mais libre à vous d'aller voir après afin de vous approprier encore plus openQA.
Par défaut tous les tests liés à un couple OS/architecture sont exécutés, donc dans notre cas la création de l'image de l'OS sera obligatoirement effectuée avant car les tests ***test_scummvm_v**** ont comme paramètre **START_AFTER_TEST=create_hdd_xfce**. OpenQA ordonnancera donc l'exécution des tests en fonction de ce paramètre. Mais, me direz-vous, comment dire à openQA d'exécuter ces tests ? Il suffit tout simplement de se connecter sur le serveur openQA (souvent en root sur un serveur de test) et d'exécuter la commande suivante :

```
# openqa-client isos post DISTRI=opensuse VERSION=42.3 FLAVOR=DVD ARCH=x86_
64 ISO=openSUSE-Leap-%VERSION%-%FLAVOR%-%ARCH%.iso _NO_OBSOLETE=1
```

Vous pouvez facilement retrouver des paramètres déjà définis plus haut, comme DISTRI, FLAVOR, ou encore ARCH. Le paramètre **_NO_OBSOLETE** est facultatif et sert principalement dans un environnement de production, cela permet de ne pas redémarrer tous les tests en cours lorsque l'on souhaite en redémarrer certains. Sinon les copains ayant des tests en cours d'exécution en même temps que vous risquent de ne pas trop apprécier...
Une fois l'image de l'OS créée, les autres tests démarreront automatiquement. A ce sujet, intéressons-nous un peu plus aux tests ***test_scummvm-v****. 2

Note

Pour ceux qui veulent se simplifier la vie, un fichier est disponible pour patcher rapidement votre instance openQA sur GitHub : https://github.com/ldevulder/openqa-scummvm/blob/master/0001-ScummVM-BASS-test.patch.

Note

L'aide sur l'API standard d'openQA se trouve sur http://open.qa/api/testapi/.

Chargement des tests ScummVM

Il faut tout d'abord dire à openQA que l'on souhaite exécuter ces tests. Pour cela openQA charge de façon arbitraire le script *main.pm* se trouvant dans **/var/lib/openqa/share/tests/ <DISTRI>/products/<DISTRI>/**, donc dans notre cas **/var/lib/openqa/share/tests/opensuse/products/opensuse/main.pm**.

Comme vous pourrez le voir, ce script contient déjà beaucoup de lignes et nous allons encore en ajouter ! Il faut donc mettre les lignes suivantes (voir fichier *patch* pour l'emplacement exact) :

```
elsif (get_var('SCUMMVM')) {
  # ScummVM test - for fun!
  boot_hdd_image;
  loadtest "console/consoletest_setup";
  loadtest "scummvm/disable_pkgkit" if !get_var('ALT_SCUMMVM_INSTALL');
  loadtest "scummvm/install_game";
  if (get_var('ALT_SCUMMVM_INSTALL')) {
    loadtest "scummvm/start_scummvm_v2";
  }
  else {
    loadtest "scummvm/install_scummvm";
    loadtest "scummvm/start_scummvm";
  }
  loadtest "scummvm/add_game";
  if (get_var('SCUMMVM_FULLSCREEN')) {
    loadtest "scummvm/test_game-fullscreen";
  }
  else {
    loadtest "scummvm/test_game";
  }
  loadtest "x11/shutdown";
}
```

Vous pouvez voir des appels à *get_var*, cette fonction sert à récupérer la valeur des variables passées en paramètre, ces mêmes variables que l'on a définies précédemment. Par exemple, dans notre cas, la variable **SCUMMVM** permet de dire que l'on souhaite exécuter les tests **ScummVM**. Je parlerai des autres variables spécifiques plus tard dans l'article.
La fonction *loadtest* permet de charger d'autres scripts ou d'autres tests, cela permet principalement de découper les tests en plusieurs scripts réutilisables et facilement maintenables. Ici nous chargeons

2

le script *consoletest_setup*, qui permet principalement de vérifier que certains paramètres de base sont corrects et d'installer certains composants nécessaires (comme l'outil wget par exemple). Viennent ensuite *disable_pkgkit* (exécuté uniquement si **ALT_SCUMMVM_INSTALL** est positionné), *install_game*, *start_scummvm_v2* (idem *disable_pkgkit*), *install_scummvm* (non exécuté si **ALT_SCUMMVM_INSTALL** est positionné), *start_scummvm* (idem *install_scummvm*), *add_game*, *test_game*.(ou *test_scumm-fullscreen* si **SCUMMVM_FULLSCREEN** est positionné) et *shutdown*. Le script *shutdown*, comme son nom l'indique, sert à arrêter proprement le serveur de test.
Je vais maintenant (enfin !) détailler le test de **ScummVM** et du jeu **BASS**.

test_scummvm_v1

Le premier script, *install_game*, installe le jeux **BASS** sur la VM :

```
my $timeout    = 300;   # Max waiting time
my $download_dir = '~/Downloads';
my $install_dir  = '~/Games';
my $game_url    = 'http://scummvm.org/frs/extras/Beneath%20a%20Steel%20Sky
/bass-cd-1.2.zip';
my $game_file   = "$download_dir/bass.zip";

# Select x11 console (just to be sure)
select_console 'x11';

# Execute xterm as normal user
x11_start_program 'xterm';

# Create the needed directories
assert_script_run "mkdir -p $download_dir $install_dir";

# Get the game
assert_script_run "wget -c $game_url -O $game_file", timeout => $timeout;

# Install the game
assert_script_run "unzip $game_file -d $install_dir", timeout => $timeout;

# Close the xterm console
wait_screen_change { send_key 'alt-f4'; };
```

La première partie est assez simple, elle définit les variables du script. La fonction *select_console* sert, comme son nom l'indique, à sélectionner le type de console (root, user, x11, etc.). Ici on veut sélectionner l'interface graphique Xfce. *x11_start_program* permet d'exécuter un programme, ici xterm. Une fois xterm démarré, les commandes suivantes sont exécutées dans ce terminal. *assert_script_run* exécute une commande (shell ou appel d'un script/binaire) et arrête le test si la commande sort en erreur. Ici on crée les répertoires nécessaires à l'installation du jeu. On récupère ensuite le jeu sur le site de **ScummVM** et on le décompresse. Enfin, *wait_still_screen* attend que l'écran arrête d'être modifié par une commande, ici *send_key*, qui envoie la séquence de touche *ALT-F4* pour clore le termninal xterm.
Le second test, *install_scummvm*, installe la machine virtuelle **ScummVM** :

```
# Execute xterm as root user
x11_start_program 'xterm';
become_root;

# Install ScummVM using the zypper_call function
zypper_call 'in scummvm';

# Close the xterm console
wait_screen_change { send_key 'alt-f4'; };
```

Ce test est assez simple aussi, il exécute un terminal xterm, se connecte en mode administrateur via la fonction *become_root* et installe ensuite le package *scummvm* via la fonction *zypper_call*. Cette dernière fonction appelle simplement la commande *zypper* (commande d'installation de package spécifique à SUSE) en mode automatique avec quelques tests supplémentaires. On ferme ensuite le terminal xterm. Certains pourront demander : pourquoi ne pas effectuer ces actions dans le même script qu'avant ? Je répondrai : oui c'est possible et de 2 façons. Soit via un terminal xterm en effectuant tout dans le même test, soit en attendant de voir le détail du test ***test_scummvm_v2***.
Le troisième test, *start_scummvm*, démarre simplement la machine virtuelle **ScummVM** :

```
# Execute ScummVM as normal user
x11_start_program 'scummvm';
save_screenshot;
```

C'est sans doute le test le plus simple, car il exécute la commande *scummvm* et il appelle ensuite la fonction save_screenshot qui, comme vous l'aurez sans doute deviné, effectue une copie d'écran qui sera visible en analysant le rapport de test. C'est assez utile, surtout en cas d'erreur pour l'analyse.
Le quatrième test, *add_game*, ajoute le jeu **BASS** dans **ScummVM** :

```
# To be sure that ScummVM is started
assert_screen 'scummvm-main-menu';

# Add the game
assert_and_click 'scummvm-select-add-game';
assert_screen 'scummvm-add-game';
assert_and_dclick 'scummvm-select-game-directory';
assert_and_dclick 'scummvm-select-bass-directory';
assert_and_click 'scummvm-select-choose';
assert_and_click 'scummvm-select-graphics';
assert_and_click 'scummvm-override-graphics-settings';
assert_and_click 'scummvm-select-graphic-mode';
assert_and_click 'scummvm-select-enhanced-mode';
assert_and_click 'scummvm-select-fullscreen-mode' if get_var 'SCUMMVM_FULLSCREEN';
assert_and_click 'scummvm-select-ok';
assert_screen 'scummvm-bass-game';
save_screenshot;
```

Ce test, bien qu'assez simple, commence à montrer la partie la plus compliquée : on doit bouger et cliquer avec la souris à certains endroits. Pour cela deux fonctions sont très utiles : *assert_and_click* et *assert_and_dclick*. La première fonction permet de définir une zone graphique où l'on souhaite cliquer à l'aide d'un *needle*. La seconde fonction fait exactement la même chose si ce n'est que

l'on effectue un double clic (*_dclick*) au lieu d'un simple clic (*_click*). La fonction *assert_screen* quant à elle attend simplement que le *needle* demandé soit affiché à l'écran. Cela permet généralement d'attendre le chargement d'un programme ou l'ouverture d'une fenêtre. Basiquement dans ce test on exécute **ScummVM**, on sélectionne le menu "**Add Game...**" et on ajoute/configure le jeu. On retrouve également la variable **SCUMMVM_FULLSCREEN** dont je parlerai d'ici peu, mais je pense que tout le monde aura déjà deviné à quoi elle sert ! 3

Le cinquième et dernier test, *test_game*, sert à tester le jeu **BASS** : Code complet sur www.programmez.com.

Comme dans le test précédent, il faut que l'on bouge et que l'on clique avec la souris. Mais ici nous ne pouvons pas forcément utiliser la fonction *assert_and_click*, car certaines parties nécessitent de bouger la souris à un endroit particulier afin, par exemple, de faire apparaître l'inventaire. J'utilise donc les fonctions *mouse_set* et *mouse_click* qui permettent respectivement de déplacer la souris aux coordonnées souhaitées et ensuite de cliquer avec le bouton spécifié, par défaut le gauche. L'introduction du jeu étant assez longue, on peut gagner du temps en positionnant la variable **BASS_SKIP_INTRO**, cela dépendra de ce que l'on veut tester.

Pour résumer, j'effectue quelques actions afin de pouvoir accéder au deuxième écran du jeu, puis je vais ensuite jusqu`au troisième écran. Rien de plus pour le moment, mais cela permet de montrer qu'openQA s'adapte assez facilement à n'importe quel type d'application. De plus, si rien de plus adapté n'existe, openQA pourra le faire avec plus ou moins de facilité. Ce qui est intéressant c'est le côté modulaire du produit, rien de vous interdit de créer votre propre bibliothèque de fonctions afin qu'openQA s'adapte à vos besoin. 4

test_scummvm_v2

Bon en quoi consiste cette variante du test *test_scummvm_v1*, car vous l'aurez aisément compris en lisant le code de *main.pm* (et comme je l'ai suggéré plus haut), il s'agit simplement d'une autre façon d'installer et d'exécuter **ScummVM** :

```
# With assert_gui_app, we can directly install and execute ScummVM!
assert_gui_app 'scummvm', install => 1, remain => 1;

# Wait for ScummVM to be started
assert_screen 'scummvm-main-menu';
save_screenshot;
```

En fait, *start_scummvm_v2* combine les tests *install_scummvm* et *start_scummvm* en un seul test en utilisant la fonction *assert_gui_app* qui exécute une application x11 tout en l'installant automatiquement si nécessaire. Cela permet de gagner des lignes de code. Cela peut être plus intéressant parfois. C'est un bon exemple d'une fonction ajoutée par rapport à un besoin des développeurs QA.

Mais, et mon option de tout à l'heure alors ?

Revenons à notre option oubliée, **SCUMMVM_FULLSCREEN**. Comme vous avez pu le voir soit en exécutant vous-même les tests, soit avec les copies d'écran, le jeu **BASS** (enfin plutôt **ScummVM**) s'exécute en mode fenêtre. C'est bien mais un jeu en mode plein écran c'est mieux ! Fort heureusement, **ScummVM** dispose d'une option qui permet de spécifier si l'on souhaite démarrer le jeu dans ce mode. Cette option est soit définie globalement soit jeu par jeu. Dans le script *add_game* vous pouvez voir que cette option sera sélectionnée lors de l'ajout du jeu si la variable **SCUMMVM_FULLSCREEN** est positionnée. Mais un souci va arriver assez vite : les coordonnées x,y de la souris dans le script *test_game* ne seront plus forcément les mêmes dès lors que l'on sera en mode plein écran… Que faire ? Deux solutions sont possibles : soit écrire un script *test_game_fullscreen* avec les bonnes coordonnées, soit adapter *test_game* avec une fonction style *wrapper_mouse_set* qui pourra utiliser un système d'offset pour passer facilement d'un mode à l'autre sans changer les coordonnées. Je n'ai pas écrit cette fonction, mais je vous laisse le soin de travailler un peu et d'essayer de faire ça ! Maintenant que vous avez une base de test fonctionnelle il n'y a plus qu'à !

Pour aller encore un peu plus loin...

Pour faire encore mieux il est possible d'adapter les scripts à d'autres jeux. En créant par exemple des fonctions basées sur les scripts. Fonctions qui pourront être réutilisées pour les jeux LucasArts ou Sierra par exemple. Le script *add_game* pourrait ainsi facilement se transformer en fonction du même nom dont les tags des *needles* seraient variabilisés en fonction du jeu à tester.

Mais toutes les bonnes choses ont une fin...

Voilà, on arrive à la fin de cet article et de cette mini-série sur openQA. J'espère que cette introduction ludique vous a permis d'appréhender l'outil et de voir ses possibilités. Ce test de **ScummVM**/**BASS** a permis de montrer que l'on peut assez facilement et rapidement tester quelque chose de "nativement non-automatisable" avec openQA. Mais attention, cela ne veut pas dire qu'il peut tout faire et tout tester ! Comme je le notais dans l'article précédent, il faut essayer de bien choisir son outil en fonction de ses besoins. Et si openQA est cet outil, alors amusez-vous bien avec lui ! •

3


4

Julien Demangeon
Consultant / Développeur Full-Stack
Marmelab - marmelab.com

Crystal, le langage qu'il vous faut ? Partie 2

Le langage Crystal a été publié pour la première fois en 2014 et est encore activement développé. Il fait aujourd'hui partie des langages compilés à considérer pour démarrer un nouveau projet. Pourquoi ce langage plutôt qu'un autre ? À quel type de projet est-il le plus adapté ? Nous allons tenter de répondre à ces questions.

Concurrence : Fibers et Channels

En dehors de sa syntaxe et de son typage évolué, Crystal fournit également un système de tâches concurrentes et synchronisées à travers les Fibers et les Channels.

Les Fibers peuvent être comparées à des "threads" légers gérés par la couche logicielle (on appelle cela des "Green Threads"). Elles ont le même rôle que les fameuses goroutines du langage Golang.

Les Channels quant à eux, permettent de synchroniser des données entre plusieurs Fibers grâce à un système de "canaux" de messages. Un système du même nom est également utilisé en Go.

Pour illustrer au mieux cette fameuse fonctionnalité, voici un petit exemple avec son équivalent dans les deux langages.

```
// GOLANG

messages := make(chan string)
defer close(messages)

go func() {
  time.Sleep(time.Second * 5)
  messages <- "ping"
}()

msg := <-messages // Attend 5s ici
fmt.Println(msg) // => ping
# CRYSTAL

messages = Channel(String).new

spawn do
 sleep 5.seconds
 messages.send("ping")
end

msg = messages.receive # Attend 5s ici
puts msg # => ping
```

Dans ces exemples, un Channel est utilisé afin de fournir un canal d'échange (ici de type String) entre le thread principal et la routine asynchrone.

Pour déclarer cette routine, le mot clé spawn est utilisé en Crystal, là où le préfixe go suivi d'une fonction anonyme est utilisé en Golang.

Au moment de l'assignation à la variable "msg", le thread principal est bloqué en attente d'un message en provenance du channel "messages".

Comme on peut le voir, les Channels sont très utiles car ils permettent de faire transiter des informations de manière asynchrone entre plusieurs contextes d'exécution.

Petite ombre au tableau cependant, Crystal ne permet pas (pour l'instant) de distribuer des opérations sur plusieurs cœurs, comme le fait Golang avec ses goroutines.

Tests unitaires

Encore une fois, Crystal s'est inspiré de Ruby, à travers la librairie RSpec, afin de développer son module de tests unitaires intégré.

Ce module, nommé "Spec", permet de créer des suites de tests qui utilisent une syntaxe proche du domaine. En voici un exemple tiré de la documentation.

```
require "spec"

describe Array do
 describe "#size" do
  it "is the number of elems in the array" do
   [1, 2, 3].size.should eq 3
  end
 end

 describe "#empty?" do
  it "is true if the array is empty" do
   ([] of Int32).empty?.should be_true
  end

  it "is false if the array is not empty" do
   [1].empty?.should be_false
  end
 end
end
```

Cette syntaxe de tests (que l'on retrouve dans de nombreuses librairies d'autres langages) permet de décrire les tests de manière naturelle et donc d'en faciliter la lecture.

Par ailleurs, le module "Spec" possède des commandes d'assertion assez simples et limitées en nombre, mais qui permettent (presque) de tout faire.

```
actual.should eq(expected) # actual == expected ?
actual.should be(expected) # actual is expected ?

actual.should be_truthy # actual value == true ?
actual.should be_falsy # actual value == false ?
```

```
actual.should be < expected # actual < expected ?
etc...
```

Par convention, la commande de test exécute tous les fichiers situés dans un dossier "spec/" à la racine du projet. Cependant, il est tout à fait possible de modifier ce comportement (pour par exemple, appliquer une architecture "Domain Design Driven" à notre projet).
Voici quelques cas d'usage particuliers de la commande de tests unitaires.

```
# Lance tous les "*_spec.cr" dans "spec/"
crystal spec

# Lance tous les "*_spec.cr" dans "spec/my/test"
crystal spec spec/my/test/

# Lance tous les tests de "file_spec.cr"
crystal spec spec/my/test/file_spec.cr

# Lance les tests à la ligne 14 de "file_spec.cr"
crystal spec spec/my/test/file_spec.cr:14
```

Fonctions intégrées

De nombreuses fonctions d'usage courant ont été incluses au sein du langage. Elles permettent de gérer de larges besoins tels que la manipulation de formats (JSON, CSV, Gzip, etc), l'authentification (Oauth), les sockets (UDP, TCP…), etc.
Il est par exemple très facile de mettre en place un serveur HTTP basique en Crystal grâce au module "http".

```
require "http/server"

server = HTTP::Server.new(8080) do |context|
  context.response.content_type = "text/plain"
  context.response.print "Hello from #{context.request.path}!"
end

puts "Listening on http://127.0.0.1:8080"
server.listen
```

Gestion de dépendances

Bien que parfaitement outillé, tout n'est pas intégré au langage pour autant. C'est là qu'intervient le système de gestion de dépendances de Crystal.

Comme la plupart des langages actuels, Crystal permet de déclarer des dépendances externes à un projet à l'aide d'un simple fichier de configuration (ici nommé shard.yml).

```
name: MyAmbitiousProject
version: 0.1.0

dependencies:
  openssl:
    github: datanoise/openssl.cr
    branch: master

development_dependencies:
  minitest:
    git: https://github.com/ysbaddaden/minitest.cr.git
    version: ~> 0.3.1

license: MIT
```

Comme vous pouvez le voir, le fichier contient également de nombreuses métadonnées sur le projet courant (nom, version, licence, etc.) afin d'en faciliter la distribution.
Pour installer les dépendances indiquées dans ce fichier, l'usage d'une simple commande crystal deps dans le terminal suffit.
Un portail dédié (https://crystalshards.xyz/) recense la plupart des librairies du langage. N'hésitez pas à y jeter un oeil, vous y trouverez certainement ce dont vous avez besoin pour vos futurs projets.

Pour quels usages ?

Bien que compilé, Crystal n'offre cependant pas autant de performances que des langages tels que Golang ou Rust qui possèdent une gestion accrue des architectures multicœurs et / ou de la mémoire. Si vous avez déjà développé en Ruby et que vos besoins sont assez simples (scripts, petits serveurs, etc.), Crystal est fait pour vous. Dans le cas contraire, il faudra certainement prendre votre mal en patience quelques temps encore, que le langage gagne en maturité (et en communauté).
Bien évidemment, cet article ne donne qu'un aperçu des possibilités offertes par ce langage. Pour en savoir plus, n'hésitez pas à consulter le site officiel (https://crystal-lang.org/) qui est très fourni.
Vous pouvez aussi à tout moment contribuer à l'évolution du langage dont le dépôt de code est à plus de 10 000 stars sur Github (https://github.com/crystal-lang/crystal) à l'heure où j'écris ces lignes. •

Sylvain Pontoreau
Premier Field Engineer
sylvain.pontoreau@microsoft.com
https://sylvain.pontoreau.com

Programmer avec TypeScript

Partie 2

Dans la première partie de ce dossier consacré à TypeScript, nous nous sommes intéressés aux capacités basiques du langage et plus particulièrement à celle liées à la programmation orientée objet. Les quelques concepts que nous avons vus sont pour la plupart disponibles dans JavaScript depuis ES6. Cependant, TypeScript dispose aussi de capacités qui lui sont propres et lui permettent d'aller plus loin dans certains types d'implémentations. Dans cette deuxième partie nous allons donc nous pencher sur certaines particularités importantes et propres au langage.

Abstraction

Dans la première partie du dossier nous avons abordé la notion d'héritage. C'est un point important de la POO. Avec TypeScript il est possible d'aller plus loin en déclarant des classes abstraites. Cette notion n'existe pas de base en JavaScript (même s'il est tout à fait possible de la reproduire). De son côté TypeScript supporte pleinement l'abstraction et suit des règles similaires à la POO traditionnelle. Premièrement, une classe abstraite ne peux pas être instanciée :

```
abstract class Person {
  name: string;

  constructor(name: string) {
    this.name = name;
  }
}

const person = new Person();//Error
```

Deuxièmement, les classes abstraites peuvent contenir des méthodes abstraites. C'est un type de méthode particulier qui ne dispose pas de corps. Comme pour les classes, il faut utiliser le mot clé « abstract » pour définir une méthode abstraite :

```
abstract class Person {
  name: string;

  constructor(name: string) {
    this.name = name;
  }

  abstract greeting(): string;
}
```

Une fois déclarée abstraite, une méthode doit obligatoirement être définie lorsque la classe est héritée :

```
class Employee extends Person {
  role: string;
  constructor(name: string, role: string) {
    super(name);
    this.role = role;
  }

  greeting(): string {
    return `Hello ${this.name}! You're a Microsoft ${this.role}.`;
  }
}

const employee: Person = new Employee("Sylvain", "PFE");
console.log(employee.greeting());//Hello Sylvain! You're a Microsoft PFE.
```

L'abstraction est une mécanique très utile pour définir des implémentations partielles et/ou généralistes. Le concept peut aussi être utilisé comme intermédiaire entre les différentes couches d'un logiciel. Il permet alors de faciliter la maintenance, car les classes utilisant l'abstraction n'adhèrent pas directement aux implémentations. C'est un des principes que l'on retrouve dans SOLID et TypeScript permet de le respecter (si ces principes ne vous sont pas familiers, je vous invite à aller jeter un coup d'œil sur Wikipédia pour en avoir une brève description).

Le fait que TypeScript dispose de capacités d'abstraction est une excellente chose pour quiconque veut implémenter des cas avancés en programmation orientée objet. Cependant, j'insiste très fortement sur le fait qu'une fois le code transpilé cette notion n'existe plus. C'est donc une règle applicable uniquement dans du code TypeScript. Une fois le code transpilé, une classe abstraite est instanciable en JavaScript, n'oubliez jamais cela !

Interface

Les interfaces sont une autre manière d'abstraire du code en POO. TypeScript permet de les utiliser, mais comme pour les classes abstraites la notion d'interface n'existe plus après compilation. Le résultat après transpilation d'une interface est un fichier vide ! Avant même de vous décrire à quoi elles servent, vous êtes à nouveau prévenu que ce concept est inexistant en JavaScript ! Malgré ce détail, les interfaces sont très importantes dans TypeScript et elles peuvent être extrêmement utiles dans de nombreux cas de figure. Commençons déjà par un cas d'utilisation classique. Une interface permet de décrire un comportement qui sera implémenté par une classe. Au vu de la nature d'une interface il est impératif que tous ses membres soient déclarés publics. Une classe peut implémenter une interface en utilisant le mot clé « implements ». Ici on reste à nouveau sur une utilisation proche de ce que l'on connaît en C# ou Java.

```
interface Logger {
   debug(message: string): void;
}

class ConsoleLogger implements Logger {
   debug(message: string): void {
      console.debug(message);
   }
}
```

Bien évidemment, l'héritage d'interface est possible avec TypeScript. On peut définir une interface qui hérite d'autres afin de combiner différents comportements :

```
interface Person {
   name: string;
}

interface Entity {
   id: string;
}

interface Employee extends Person, Entity {
   pin: number;
}
```

Dans les langages de POO traditionnels, la notion d'interface s'arrête souvent là (bien que l'on voit depuis peu une évolution sur le sujet). En TypeScript ce n'est pas le cas. Contrairement à ce que l'on peut penser de premier abord, les interfaces sont en réalité bien différentes dans ce langage. Lorsque l'on utilise du typage dynamique, il est parfois nécessaire de déterminer précisément le type d'un objet. Dans ce cas de figure il est fréquent d'utiliser une technique que l'on appelle le « Duck-Typing ». Pour faire simple, si un objet dispose d'un comportement particulier, alors il peut être traité comme étant du type correspondant. Voici un exemple en JavaScript qui permet d'illustrer le propos :

```
const duck = {
   fly: () => console.log("I can fly!"),
   swim: () => console.log("I can swim!"),
   walk: () => console.log("I can walk!"),
   feather: true
};

 const dog = {
   swim: () => console.log("I can swim!"),
   walk: () => console.log("I can walk!"),
   fur: true
};

 const isDuck = function(obj) {
   return obj.hasOwnProperty("feather")
         && obj.fly
         && obj.swim
         && obj.walk;
}

if(isDuck(duck)) {
   (duck).fly();
}

if(!isDuck(dog)) {
   console.log("Not a Duck...")
}
```

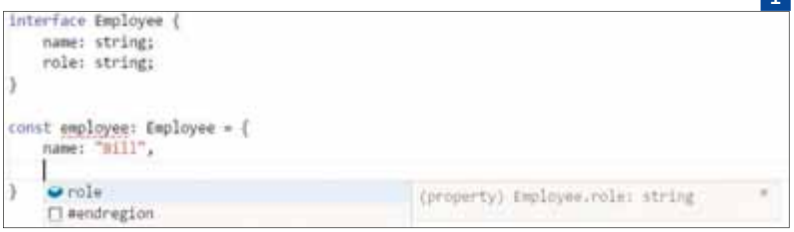
Cette technique est souvent utilisée en JavaScript. Elle a cependant un inconvénient assez flagrant, pour chaque type d'objet il faut écrire du code qui va être exécuté au runtime pour vérifier la structure de l'objet. S'il y a des cas où en TypeScript il est nécessaire d'utiliser le « Duck-Typing » (On peut citer comme exemple la désérialisation d'un JSON qui renvoie un objet anonyme), pour les types pouvant être déterminés avant la compilation, il est préférable de faire autrement. Le premier réflexe d'un débutant sur le langage serait d'écrire une classe. Cela a du sens si cette classe dispose de méthodes contenant de la logique, mais dans le cas d'un objet littéral il est bien plus élégant d'utiliser une interface. Cela permet au compilateur d'interpréter le type de l'objet et de ne pas avoir à transpiler du code. L'interface fait donc office de « contrat » qu'un objet littéral doit respecter. Voici un exemple concret :

```
interface Employee {
   name: string;
   role: string;
}

function greeting(employee: Employee): string {
   return `Hello ${employee.name}! You're a Microsoft ${employee.role}.`;
}

const employee = {
   name: "Sylvain",
   role: "PFE"
};
console.log(greeting(employee));//Hello Sylvain! You're a Microsoft PFE.
```

Dans cet exemple, l'objet littéral n'est pas instancié avec un type. En revanche, il est contractuellement conforme à l'interface « Employee », donc il peut être accepté en tant que paramètre de la fonction « greetings ». Bien évidemment il est possible de préciser le type sur la variable « employee », cela permet d'avoir l'auto-complétion : 1

Une particularité des interfaces (qui n'est pas toujours très claire lorsque l'on débute en TypeScript) est qu'elles peuvent avoir des

1



membres déclarés optionnels (syntaxiquement représenté par un « ? »). Il faudra alors vérifier si le membre a été défini ou non lors de l'instanciation :

```
interface Employee {
  name: string;
  age?: number;
}

function greeting(employee: Employee): string {
  return `Hello ${employee.name}! ${
    employee.age
    ? `You're ${employee.age} years old.`
    : "There is no age limit to understand this example :)"
  }`;
}

console.log(greeting({ name: "Satya" }));
//Hello Satya! There is no age limit to understand this example :)
console.log(greeting({ name: "Bill", age: 62 }));
//Hello Bill! You're 62 years old.
```

Autres détails, un membre d'une interface peut être déclaré en lecture seule :

```
interface Person {
  readonly name: string;
}

const person: Person = {
  name: "Bill"
};
person.name = "Satya"//Error
```

L'utilisation des interfaces pourrait s'arrêter là, mais il reste un dernier cas particulièrement intéressant lié à la nature de JavaScript. Le langage est multiparadigme, il est possible d'écrire du code impératif, objet ou encore fonctionnel. TypeScript étant un surensemble de JavaScript, il est donc lui aussi multiparadigme. Une des conséquences directes est que l'on peut affecter l'instance d'une fonction dans une variable, celle-ci étant considérée comme un objet en JavaScript :

```
const greeting = function(name: string): string {
  return `Hello ${ name }!`;
}

console.log(greeting instanceof Object);//true
```

TypeScript permet de sublimer cette caractéristique de JavaScript en permettant de typer une fonction grâce à une interface. Voici l'exemple pour la fonction « greeting » que nous venons de voir précédemment :

```
interface Greeting {
  (name: string): string;
}
```

Ce genre d'interface permet, par exemple, d'appliquer un type à une fonction de rappel (« callback » en anglais), ou encore d'être utilisé par une autre fonction dans une portion de code écrit en programmation fonctionnelle. Prenons un exemple avec une interface définissant une personne et une autre permettant de typer une fonction afin d'appliquer un filtre sur une instance implémentant la précédente :

```
interface Person {
  name: string;
  isCeo: boolean;
}

interface Filter {
  (person: Person): boolean
}
```

Cette interface peut être utilisée par une fonction afin de typer un de ses paramètres. Continuons l'exemple en définissant une fonction de recherche utilisant le type « Filter » :

```
function search(persons: Person[], filter: Filter): Person[] {
  const filteredPersons: Person[] = [];
  persons.forEach(function(person) {
    if(filter(person)) {
      filteredPersons.push(person);
    }
  });
  return filteredPersons;
}
```

Il est à présent possible d'utiliser la fonction « search » avec différentes implémentations de l'interface « Filter » pour appliquer des conditions lors de l'exécution d'une recherche :

```
const isNotCeo = function(person: Person): boolean {
  return !person.isCeo;
}

const startWithC = function(person: Person): boolean {
  return person.name.charAt(0).toUpperCase() === "C";
}

const employees: Person[] = [
  { name: "Satya", isCeo: true },
  { name: "Sylvain", isCeo: false },
  { name: "Sébastien", isCeo: false },
  { name: "Christopher", isCeo: false }
]

const notCeoEmployees = search(employees, isNotCeo);
console.log(notCeoEmployees);
//[ { name: 'Sylvain', isCeo: false },
//  { name: 'Sébastien', isCeo: false },
//  { name: 'Christopher', isCeo: false } ]
```

```
const nameStartWithC = search(employees, startWithC);
console.log(nameStartWithC);
//[ { name: 'Christopher', isCeo: false } ]
```

Etant donné qu'ES6 a normalisé les fonctions fléchées (« Arrow Functions », aussi connues sous le nom de Lambda), il est possible d'utiliser une syntaxe encore plus compacte :

```
const notCeoEmployees = search(employees, (person) => !person.isCeo);
console.log(notCeoEmployees);
//[ { name: 'Sylvain', isCeo: false },
//  { name: 'Sébastien', isCeo: false },
//  { name: 'Christopher', isCeo: false } ]
```

Comme vous pouvez le constater, les interfaces en TypeScript sont vraiment très utiles et offrent pas mal de possibilités, en plus de bien structurer votre code.

Généricité

La notion de généricité n'est certainement pas une nouveauté pour les développeurs Java ou C#. Elle existe depuis longtemps et permet d'écrire du code qui fonctionnera sur plusieurs types au lieu d'un seul. C'est donc un concept très pratique pour écrire du code réutilisable. Les génériques n'existent pas à proprement parler en JavaScript, même si l'on peut considérer que le typage dynamique permet de réaliser ce genre d'implémentation. Néanmoins, il peut être difficile de traiter certains cas génériques sans disposer de cette notion dans un langage. Du côté de TypeScript, pas de problème concernant la généricité puisque le langage propose ce concept à un niveau « proche » de celui de C#. Je mets proche entre guillemets, car il existe une limitation que nous verrons au cours de ce paragraphe.

En TypeScript, pour déclarer un générique on peut utiliser par convention la lettre « T ». Bien évidemment cela reste une convention et il est donc possible d'utiliser une autre valeur, sachant qu'il est aussi possible de déclarer plusieurs génériques. Lorsque l'on n'est pas familier du concept de généricité, il n'est pas évident d'implémenter un cas d'utilisation réelle. Il existe pourtant un type utilisant les génériques en TypeScript et qui permet de comprendre le principe très rapidement : « Array ».

```
const persons: Array<Person> = [];
persons.push({ name: "Bill" });
```

Lors de l'instanciation d'un tableau, il est nécessaire de préciser le type qui va être manipulé par celui-ci. « Array » définit le type générique au niveau de l'interface, ce qui permet de l'appliquer à l'ensemble de ses membres. Voici un exemple de déclaration :

```
interface Entity<T> {
  id: T;
}
```

Lors de la création d'une variable avec cette interface, il faudra alors préciser le type concret qui va être utilisé. Bien évidemment l'instance doit s'y conformer, il devient donc impossible de manipuler les membres ayant une signature utilisant un générique avec un autre type que celui qui a été utilisé pour déclarer la variable :

```
const entity: Entity<number> = {
  id: 1
};
entity.id = "1";//Erreur
```

Il est aussi possible d'utiliser la généricité sur une classe :

```
class Person<T> {
  id: T;

  constructor(id: T) {
    this.id = id;
  }
}
```

Et d'implémenter une interface générique tout en conservant la généricité au niveau de la classe :

```
class Person<T> implements Entity<T> {
  id: T;

  constructor(id: T) {
    this.id = id;
  }
}
```

Ou tout simplement de préciser le type concret qui va être utilisé lors de l'implémentation :

```
class Person<T> implements Entity<number> {
  id: number;

  constructor(id: number) {
    this.id = id;
  }
}
```

La déclaration globale d'un générique n'est pas le seul cas de figure. Il est en effet possible de déclarer un générique sur une méthode sans nécessairement avoir besoin de le faire sur l'interface ou la classe. Il faudra alors préciser le type concret lors de l'utilisation de la méthode :

```
class Values {
  add<T>(value: T): void {
    //Do something here
  }
}

const values = new Values();
values.add<number>(1);
values.add<string>("A value");
values.add(true);// Type inference works!
values.add<string>(1);//Error
```

De base les génériques sont déjà très pratiques pour structurer et réutiliser du code, mais il est possible d'aller encore plus loin.
Premièrement, on peut déclarer une contrainte sur un générique. Elle permet d'appliquer une règle sur le générique afin qu'il respecte un certain niveau d'implémentation. Dès lors, seuls les types concrets correspondant à la contrainte pourront être utilisés. Le principal intérêt des contraintes est de permettre la manipulation d'un objet en tant qu'instance disposant des capacités définies par la contrainte. Il est important de préciser que l'on peut en appliquer plusieurs sur un générique. Le mot clé « extends » permet de déclarer une contrainte, voici un exemple très simple :

```
interface Person {
  name: string;
}

function greeting<T extends Person>(person: T): string {
  return `Hello ${ person.name } !`;
}
```

Deuxièmement, il est possible de faire une référence au constructeur d'un générique afin de pouvoir en créer une instance, cependant cette référence doit être un paramètre et ne peut pas être une contrainte. Pour un développeur venant du monde C# ou Java cela peut être perturbant, car dans ces langages il est possible d'appliquer une contrainte permettant l'instanciation. C'est là qu'est la limitation avec le langage TypeScript. Etant donné que l'équipe d'Anders se refuse à transpiler du code permettant d'émuler des capacités venant d'un autre langage (le but étant de coller au plus proche de la norme ECMAScript), la seule façon de créer une instance d'un type générique est donc de passer la référence au constructeur via un paramètre. Un bon exemple d'utilisation de cette capacité est l'implémentation d'une « Factory » générique :

```
interface Person {
  name: string;
}

class Employee implements Person {
  company: string = "Microsoft";
  name: string = "";
}

class Factory<T> {
  private readonly ctor: new () => T
  constructor(ctor: new () => T) {
    this.ctor = ctor;
  }

  getInstance(): T {
    return new this.ctor();
  }
}

const personFactory = new Factory<Person>(Employee);

const person = personFactory.getInstance();
person.name = "Sylvain";
console.log(person);//Employee { company: 'Microsoft', name: 'Sylvain' }
```

Malgré cette limitation, la référence au constructeur d'un générique permet d'implémenter tout un tas de scénarios avancés.

Conclusion

Dans cette partie nous avons pu voir certaines capacités qui sont propres à TypeScript. Depuis le début de ce dossier j'ai pris le parti de présenter le langage en faisant des rapprochements directs avec la POO traditionnelle. Cependant, comme vous avez pu le constater avec les interfaces, sa nature réelle est bien plus proche de JavaScript. Le langage reste et restera un sur-ensemble de JavaScript. C'est pourquoi lors de la dernière partie nous nous attarderons sur certains points hérités directement d'ECMAScript qu'il est nécessaire de connaître pour créer un vrai projet. Actuellement, les exemples de code ne fonctionnent que dans un seul fichier, il est donc grand temps de parler des modules ! Nous verrons aussi plusieurs concepts avancés qu'il est possible de mettre en œuvre avec TypeScript.

•

N'oubliez pas !

Retrouvez les codes sources des articles sur programmez.com et sur GitHub :
https://github.com/francoistonic

Fabien Pigère
freelance html5 en remote, spécialisé dans l'optimisation et le multimédia.

Effets spéciaux sur la balise vidéo

HTML5 a démocratisé la vidéo, dorénavant, plus de plug-in obligatoire. La vidéo peut se jouer partout, y compris sur les portables. Une simple vidéo en MP4 suffit pour être jouée sur n'importe quel support de diffusion.
Nous allons apprendre à exploiter cette balise, afin d'en montrer la puissance couplée à d'autres balises HTML5 récentes.

Voyons une simple page contenant une vidéo :

```
<!DOCTYPE html>
<html>
<body>

<video id="myVideo"  width="320" height="240" controls>
 <source src="movie.mp4" type="video/mp4">
</video>

</body>
</html>
```

Cette page affiche une vidéo en 320/240 avec les contrôles (son, position, play, pause ...), ça se passe de commentaire !
Voyons les méthodes intéressantes associées à cette balise :
Play() qui joue la vidéo, et Pause() la met en pause.
Au niveau des évènements, onloadedmetadata est appelé quand les données importantes ont été chargées, prêtes à être jouées.
La balise contient de multiples autres évènements et propriétés, mais seules celleslà nous serviront. Comme on le voit dans l'API on peut jouer des vidéos et...rien d'autre. Mais nous sommes exigeants, et voudrions écrire dessus, mettre un logo transparent à notre nom, voire complètement transformer la vidéo, soyons fou !

La balise Canvas à la rescousse !

La balise Canvas est elle aussi soutenue par tous les navigateurs majeurs depuis longtemps. Pour rappel, elle permet de dessiner en 2D tout ce que nous voulons : texte, rectangle, courbes de Bézier. Elle est parfaite pour nous. Maintenant, il nous reste à joindre la balise canvas à la balise video.
On commence par rajouter la balise canvas a notre HTML :

```
<canvas id="myCanvas"   width="320" height="240"></canvas>
```

Puis dans un script, on opère la magie :

```
"use strict";

document.addEventListener('DOMContentLoaded', init, false);

var video, canvas, context;

function videoReady() {
   var width = this.videoWidth;
   var height = this.videoHeight;
   setTimeout(draw, 0, video, context, width, height);
}

function init() {
   video = document.getElementById('myVideo');
   canvas = document.getElementById('myCanvas');
   context = canvas.getContext('2d');
   video.addEventListener("loadedmetadata", videoReady, false);
}

function draw(v, c, w, h) {
   setTimeout(draw, 0, v, c, w, h);
   if (v.paused || v.ended) return false;
   c.drawImage(v, 0, 0, w, h);
}
```

Quelques commentaires :
1) On doit attendre le chargement des metadata avant d'appliquer notre traitement,
2) On utilise un settimeout au lieu de l'évènement timeupdate. En effet, dans les faits, cet event n'est pas déclenché assez régulièrement. On aurait pu prendre un requestAnimationFrame pour obtenir le même effet.

A partir de là, les possibilités sont multiples, voyons-en une simple pour commencer :

```
function draw(v, c, w, h) {
   setTimeout(draw, 0, v, c, w, h);
   if (v.paused || v.ended) return false;
   c.drawImage(v, 0, 0, w, h);
   var img = document.getElementById("btn");
   c.drawImage(img, 10, 10);
}
```

Avec « btn », une image à votre convenance dans votre HTML.
Nous avons rajouté un bitmap à notre vidéo, notre marque est enfin incrustée ! Poussons plus loin :
Code complet sur www.programmez.com

Que voyons-nous ? La vidéo est alors en noir&blanc ! Et elle garde notre logo.

Pour résumer :

La balise vidéo offre un moyen simple de jouer de la vidéo.
Il est possible d'obtenir l'image courante, de la modifier grâce à Canvas, et de lui appliquer tous les post-traitements imaginables !
En cas de ralentissement, je vous conseillerais de regarder du côté des WebWorker, ils permettent le traitement multithread simple en JavaScript, de façon élégante. •

Sylvain SAUREL
Développeur Java / Android
https://www.ssaurel.com
sylvain.saurel@gmail.com

Boostez vos développements Android avec ces 10 fonctionnalités de Kotlin

Dans cet article, nous vous proposons de découvrir 10 fonctionnalités de Kotlin qui vont vous convaincre de l'adopter sur votre prochain projet de développement d'application Android.

1 Import statique des layouts

Quand on parle de code boilerplate fastidieux à écrire sous Android, comment ne pas penser aux appels récurrents à la méthode *findViewById* afin d'obtenir les références aux vues d'une activité ou d'un fragment. L'écriture de ces appels nuit considérablement à la productivité du développeur Android.

Bien entendu, des solutions existent d'ores et déjà sous la forme de bibliothèques de code à intégrer. La plus célèbre étant probablement ButterKnife qui permet de s'abstraire d'une partie de ce travail en marquant les vues à récupérer avec des annotations spécifiques. Kotlin va cependant plus loin en permettant d'importer toutes les références des vues depuis le layout utilisé par l'activité courante avec une seule ligne d'import !

Considérons le layout XML suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context="com.ssaurel.kotlincounter.MainActivity">

 <TextView
  android:id="@+id/value"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textSize="100sp"
  android:textStyle="bold"
  android:text="0"
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintRight_toRightOf="parent"
  app:layout_constraintTop_toTopOf="@id/plusBtn"
  app:layout_constraintBottom_toBottomOf="@id/minusBtn"
 />

 <Button
  android:id="@+id/plusBtn"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textSize="50sp"
  android:text="+"
  app:layout_constraintTop_toTopOf="parent"
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintRight_toRightOf="parent"
  android:layout_marginTop="50dp"
 />

 <Button
  android:id="@+id/minusBtn"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textSize="50sp"
  android:text="-"
  app:layout_constraintBottom_toBottomOf="parent"
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintRight_toRightOf="parent"
  android:layout_marginBottom="50dp"
 />
</android.support.constraint.ConstraintLayout>
```

Le code de l'activité utilisant ce layout et travaillant sur les vues qu'elle contient pourra avoir l'allure suivante :

package com.ssaurel.kotlincounter

```
import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

 var id = 0

 override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  setContentView(R.layout.activity_main)

  value.setText("" + id)

  plusBtn.setOnClickListener {
   value.setText("" + ++id)
  }

  minusBtn.setOnClickListener {
   value.setText("" + --id)
```

```
  }
 }
}
```

On voit clairement qu'il n'est plus nécessaire d'effectuer des appels à la méthode *findViewById* ni de marquer les propriétés qui correspondent à des vues avec des annotations spécifiques. Les vues contenues dans le fichier layout utilisé par l'activité sont accessibles directement. La magie, si magie il y a, se situe tout simplement dans la ligne d'import suivante :

```
import kotlinx.android.synthetic.main.activity_main.*
```

Le plugin Kotlin Android Extensions se charge de la génération du code boilerplate et ce simple import va faciliter la vie de tous les développeurs Android. Il faudra cependant faire attention à bien importer le layout utilisé par l'activité ou le fragment. Quant au fragment, les vues ne seront correctement référencées, et donc accessibles, qu'une fois la méthode *onCreateView* atteinte.

2 Ecriture simplifiée des classes POJO

L'écriture des classes POJO (Plain Old Java Object) dans des projets Java ou Android est déjà simplifiée puisque les IDE modernes permettent de générer la majeure partie de leur contenu. Néanmoins, une fois le code généré, ce sera aux développeurs d'en assurer la maintenance. Cela constitue une réelle perte de temps. Kotlin propose, là encore, une approche différente. En effet, si le code d'une classe POJO est aussi généré, il demeure invisible pour le développeur qui n'a alors plus à se soucier de la maintenance du code généré au cours de la vie d'un projet.
Considérons le POJO User suivant en Java :

```
public class User {

 private String firstName;
 private String lastName;

 public String getFirstName() {
  return firstName;
 }

 public void setFirstName(String firstName) {
  this.firstName = firstName;
 }

 public String getLastName() {
  return lastName;
 }

 public void setLastName(String lastName) {
  this.lastName = lastName;
 }
}
```

Avec Kotlin, vous n'avez plus à écrire ou à générer de getters / setters au sein d'une classe. En outre, par défaut, tout est public avec Kotlin ce qui implique qu'il n'est plus nécessaire de marquer une classe comme telle si elle l'est. La classe User équivalente en Kotlin aura l'allure suivante :

```
class User {
 var firstName: String? = null
 var lastName: String? = null
}
```

On constate clairement le gain en lignes de code apporté par Kotlin au niveau des classes POJO. Cet exemple est également intéressant afin de souligner qu'en Kotlin, deux options sont proposées aux développeurs lors de la déclaration d'une variable :

- Marquer la variable avec le mot-clé *var* s'il s'agit d'une variable qui peut changer de valeur ;
- Marquer la variable avec le mot-clé *val* s'il s'agit d'une variable immuable.

Nous pouvons ainsi ajouter une variable immuable permettant de retourner le nom complet d'un utilisateur comme suit :

```
class User {
 var firstName: String? = null
 var lastName: String? = null
 val fullName: String?
  get() firstName + " " + lastName
}
```

La variable *fullName* sera ainsi en lecture seule et l'on définit ce qu'elle retournera au sein de son getter. Enfin, il est bon de noter qu'en Kotlin toutes les propriétés d'une classe doivent être assignées lors de leur déclaration ou au sein du constructeur.

3 Construction d'objets boostée

Au niveau de la définition du constructeur d'une classe, Kotlin propose également une syntaxe plus concise que Java en introduisant le concept de constructeur primaire et secondaire. Une classe Kotlin a un constructeur primaire et peut avoir un ou plusieurs constructeurs secondaires. En reprenant l'exemple de notre classe User, son constructeur primaire peut être défini de la sorte :

```
class User constructor(firstName: String, lastName: String) {
 // …
}
```

Dans le cas où le constructeur primaire ne nécessite pas d'annotations ou de modificateurs liés à la visibilité, le mot-clé *constructor* peut être omis :

```
class Person (firstName: String) {
 // …
}
```

Il est bon de préciser également qu'un constructeur primaire ne contient pas de code. Si le développeur doit définir du code d'initialisation spécifique, cela peut être fait au sein d'un bloc *init* comme suit :

```
class Person (firstName: String) {
 init {
  // Initialisation …
 }
}
```

En sus, nous allons pouvoir définir le type des propriétés d'une classe directement au sein du constructeur primaire à l'aide des mots-clés *var* et *val* :

```
class User (var firstName: String, var lastName: String) {
 // …
}
```

Comme expliqué précédemment, les classes en Kotlin peuvent avoir des constructeurs secondaires. Ces derniers devant obligatoirement s'appuyer sur le constructeur primaire à l'aide du mot-clé *this* bien connu des développeurs Java :

```
class User (val firstName: String, val lastName: String) {
 constructor (firstName: String) : this(firstName, "") {
  // …
 }
}
```

Pour terminer sur la partie construction d'un objet en Kotlin, signalons que le langage ne possède pas de mot-clé *new* comme en Java. L'instanciation d'un objet User se fera donc de la sorte :

```
val user = User("Sylvain", "Saurel")
```

4 Héritage simplifié

En Kotlin, toutes les classes héritent de *Any*, qui correspond à la classe *Object* de Java qui est le parent de toutes les classes du JDK. Par défaut, une classe Kotlin est considérée comme fermée, équivalent du marquage à *final* d'une classe Java. Ainsi, pour permettre l'héritage d'une classe, il faut la marquer à l'aide du mot-clé *open* ou *abstract* :

```
open class User(val firstName, val lastName)
class Admin(val firstName, val lastName) : User(firstName, lastName)
```

Sur le même principe de l'obligation de recourir au constructeur de la classe parente à l'aide de l'appel à *super()* en Java, il est nécessaire en Kotlin d'appeler le constructeur de la classe parente.

5 Support en standard des Lambda Expressions

Les différentes fonctionnalités de Java 7 ainsi qu'un sous-ensemble des nouveautés de Java 8 sont supportées par le SDK Android et Android Studio à partir de la version 3.0. Il est donc possible d'utiliser les Lambda Expressions de Java 8 au sein d'un projet Android réalisé en Java. Néanmoins, cela impose de définir une version minimum du SDK à 24. Ceci prive alors le développeur d'une part importante du parc des smartphones et tablettes actuellement déployés. Kotlin propose un support standard des Lambda Expressions affranchissant le développeur de cette problématique de version minimum du SDK. Les Lambda Expressions de Kotlin peuvent être vues comme les classes anonymes en Java avec une syntaxe plus concise. Considérons l'exemple de l'implémentation d'un objet *OnClickListener* sur une vue en Java :

```
view.setOnClickListener(new OnClickListener() {
 @Override
 public void onClick(View v) {
  Toast.makeText(v.getContext(), "Clic sur la vue", Toast.LENGTH_SHORT).show();
 }
};
```

En Kotlin, l'emploi d'une Lambda Expression va nous permettre de raccourcir le code de la sorte :

```
view.setOnClickListener({ view -> toast("Clic sur la vue") })
```

Une seule ligne de code suffit donc ici. C'est également l'occasion de souligner qu'une Lambda Expression est délimitée par des accolades. Le ou les paramètres sont déclarés en premier et le corps de la Lambda Expression est défini après la flèche. Dans le cas d'un objet *OnClickListener*, il n'est en outre pas nécessaire de définir le type du paramètre puisqu'il peut être inféré. Enfin, on recourt à la fonction *toast* proposée en standard par Kotlin pour raccourcir la syntaxe pour appeler un *Toast* sous Android.
Kotlin étant un langage visant à gagner en productivité, il permet naturellement de simplifier encore cette syntaxe. Dans le cas où le paramètre *view* n'est pas utile et où seul un appel à une fonction est réalisé, on peut alors avoir le code suivant :

```
view.setOnClickListener() { toast("Clic sur la Vue") }
```

6 Fonctions d'extension

Sur le même principe que ce que propose le langage C#, Kotlin donne la possibilité d'étendre des classes existantes avec de nouvelles fonctionnalités via le concept de fonctions d'extension. Au sein d'un projet, il peut arriver que nous ayons besoin d'une fonction utilitaire pour calculer le hash MD5 d'une chaîne de caractères. Plutôt que de réaliser des appels à une méthode statique d'une classe utilitaire, Kotlin nous permet d'étendre les fonctionnalités de la classe standard *String* en lui ajoutant une fonction *md5()* :

```
fun String.md5(): ByteArray {
 val digester = MessageDigest.getInstance("MD5")
 digester.update(this.toByteArray(Charset.defaultCharset()))
 return digester.digest()
}
```

Pour calculer le hash MD5 d'un objet String, il suffira par la suite de réaliser l'appel suivant :

```
val hashMD5 = "Sylvain Saurel".md5()
```

7 Dites adieu aux NullPointerException

Une des sources d'erreurs les plus fréquentes en Java provient d'une mauvaise gestion de la valeur *null* pour des objets conduisant aux fameuses *NullPointerException*. Partant de ce postulat, les concepteurs du langage Kotlin ont intégré une fonctionnalité de null-safety au coeur du langage afin que le développeur n'ait plus à s'en soucier. Par défaut, toutes les variables et propriétés en Kotlin sont considérées comme non nulles puisqu'elles ne peuvent être affectées avec la valeur *null*. Le seul moyen de pouvoir leur affecter cette valeur est de les marquer explicitement comme pouvant être nulles. Pour ce faire, on devra ajouter le point d'interrogation "?" après la définition du type d'une variable :

```
var number: Int? = null
```

Cependant, même avec une variable marquée comme pouvant être nulle, le compilateur Kotlin continue de prémunir le développeur au maximum de potentiels *NullPointerException* puisqu'il va effectuer des vérifications empêchant notamment la compilation du code suivant :

```
var number: Int? = null
number.toString()
```

Pour que ce code puisse être compilé, il faudra ajouter une vérification sur la variable *number*. Un if pourrait être employé comme en

Java, mais Kotlin propose l'opérateur "?." pour cela :

```
var number: Int? = null
number?.toString()
```

Avec Kotlin, le développeur va pouvoir (enfin) dire adieu aux *NullPointerException*. Néanmoins, il est bon de souligner que ces dernières peuvent encore se produire dans les cas bien spécifiques suivants :

- Un lancement explicite de l'exception *NullPointerException ;*
- L'utilisation de l'opérateur "!!" ;
- Utilisation de code Java externe via des bibliothèques de code par exemple ;
- Si une propriété en *lateinit* est accédée au sein du constructeur avant qu'elle soit complètement initialisée, une exception *UninitializedPropertyAccessException* sera alors lancée

Concernant l'opérateur "!!", il est bon de préciser qu'il va indiquer à Kotlin que le développeur souhaite une gestion similaire à celle de Java pour les valeurs nulles. Ainsi, l'appel *number!!.toString()* va bien compiler et de fait lancer une *NullPointerException* comme en Java au runtime. L'emploi de cet opérateur est donc réservé à des cas bien spécifiques.

8 Tirez parti de la fonction with()

Présente au sein de la bibliothèque de Kotlin, la fonction *with()* permet de simplifier certaines expressions en réduisant le code à écrire lorsque l'on doit travailler sur plusieurs propriétés d'un objet. Prenons l'exemple d'une vue de type *TextView* :

```
with(myTextView) {
  text = "Sylvain Saurel"
  visibility = View.VISIBLE
}
```

D'un point de vue technique, le bloc de code est une Lambda Expression pour la fonction d'extension qui sera associée à l'objet spécifié en paramètre de la fonction *with()*.

9 Un code plus puissant avec la surcharge d'opérateurs

A l'image du C++, Kotlin permet la surcharge d'un certain nombre d'opérateurs standards. Afin de surcharger un opérateur, une fonction membre d'une classe ou une fonction d'extension doit être définie avec le nom associé à l'opérateur surchargé ainsi que le mot-clé *operator*. Prenons l'exemple de l'opérateur de multiplication que nous souhaiterions voir surchargé pour l'objet *String* :

```
operator fun String.times(b: Int): String {
  val buffer = StringBuilder()
  for (i in 1..b) {
    builder.append(this)
  }
  return builder.toString()
}
```

On pourra ensuite utiliser l'opérateur de multiplication * sur une chaîne de caractères comme suit :

```
val myString = "Hello" * 4
Cela produira en sortie :
HelloHelloHelloHello
```

Cette possibilité offerte par le langage Kotlin se révèle très puissante mais nécessite certaines précautions. En effet, puisqu'elle peut être combinée avec les fonctions d'extension, cela signifie que le comportement par défaut des opérateurs de n'importe quel objet peut être changé au sein d'un projet Android. Il faudra donc prendre garder à utiliser cette fonctionnalité à bon escient.

10 Factorisation de code avec les propriétés déléguées

Les propriétés de certaines classes peuvent partager un certain nombre de comportements. Cela peut concerner les propriétés initialisées tardivement lors de leur premier accès, les propriétés implémentant l'interface *Observable* ou encore les propriétés stockées au sein de maps plutôt que dans des champs séparés. Afin de rendre ces cas d'utilisation plus simples, Kotlin propose les propriétés déléguées. Considérons la classe simple suivante :

```
class MyClass {
  var p: String by Delegate()
}
```

Cette construction signifie que la gestion des getter / setter de la propriété *p* va être déléguée au sein de la classe *Delegate*. Cette dernière pouvant être implémentée de la façon suivante :

```
class Delegate {
  operator fun getValue(thisRef: Any?, property: KProperty<*>): String {
    return "$thisRef, vient de me déléguer '${property.name}' !"
  }

  operator fun setValue(thisRef: Any?, property: KProperty<*>, value: String) {
    println("$value a été assigné à '${property.name} dans $thisRef.'")
  }
}
```

Ici, on se contente d'afficher ou de renvoyer un message quand la propriété p sera lue ou écrite. Les propriétés déléguées sont utilisables pour les variables muables mais également les variables immuables. En standard, Kotlin propose les objets délégués spécialisés *Lazy*, *Observable* et *Vetoable* répondant aux situations les plus couramment rencontrées par les développeurs. Une délégation avec *Vetoable* permettant par exemple de bloquer l'assignation de certaines valeurs à une variable suivant des conditions définies par le développeur. Il est donc important pour le développeur de se documenter sur ces propriétés déléguées pour en tirer le meilleur parti.

Conclusion

Loin d'être exhaustive, la liste proposée dans cet article aura permis de mettre en exergue 10 fonctionnalités de Kotlin qui vont, à n'en pas douter, booster la productivité des développeurs d'applications Android au quotidien. La parfaite intégration, de facto, de Kotlin au sein de l'éditeur Android Studio facilitera la transition des développeurs vers Kotlin lors de la création de nouveaux projets Android. Avec sa syntaxe concise, intuitive et moderne, Kotlin est clairement promis à un brillant avenir et devrait prendre de plus en plus d'importance dans le monde Android. Il est donc grand temps de s'y mettre ! •

François Tonic

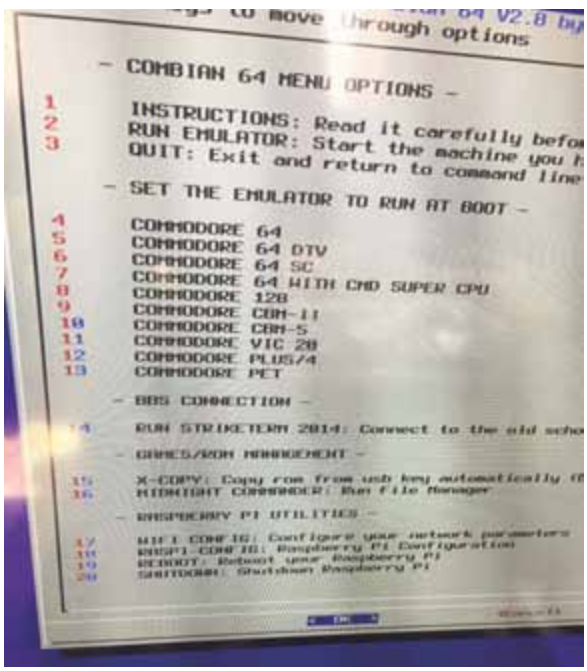
Un Commodore 64 sauce framboise

Vous avez un vieux C64 en panne ou qui prend la poussière dans un coin ? Pourquoi ne pas lui donner une nouvelle vie ? Avec quelques heures de travail, quelques matériels et une Raspberry Pi, nous allons voir comment.

En tout premier lieu, ce montage nécessite une carte de type Keyrah v2. Cette carte permet d'interfacer les claviers Commodore en USB avec une Raspberry Pi. Cette carte est facilement trouvable sur Internet.

Liste du matériel nécessaire

- 1 Commodore 64 (minimum : clavier complet avec sa nappe et le boîtier)
- 1 Keyrah v2
- 1 Raspberry Pi 3
- 1 carte SD
- 1 câble USB interne avec headers
- 1 câble court USB A mâle-mâle
- 1 câble court HDMI
- 1 câble court Ethernet
- 1 connecteur Ethernet (ex. : Neutrik NE8FDP RJ45 +1 câble court)
- 1 connecteur USB (ex. : Neutrik NAUSB)
- 1 connecteur HDMI (ex. : Neutrik NAHDMI)
- 1 tube résine / colle epoxy



Pour faciliter l'installation, vous pouvez imprimer les supports compatibles, par exemple : https://www.thingiverse.com/thing:2852922

Attention : ces supports ne sont pas parfaitement adaptés, vous devrez faire des modifications. Des kits complets sont disponibles sur ebay ou d'autres sites web, mais les prix sont parfois abusifs.

Côté outils, il vous faut : des tournevis, des pinces, un fer à souder, des petites rapes, une surface de travail assez grande et dégagée et plusieurs heures !

Préparation du C64

Il nous faut un C64. En réalité, nous n'avons besoin que du boîtier et du clavier. Ce dernier doit être en état de fonctionnement, avec sa nappe de connexion. Tous les deux se trouvent assez facilement : environ 20-25 euros / chaque élément.

Si vous avez un C64 complet, il vous faut démonter le capot puis retirer le clavier pour extraire l'ensemble de la carte mère. Prenez soin de ne pas forcer sur les fixations ni sur les nappes.

Si besoin, nettoyez le boîtier.

On teste la partie carte

Avant de monter quoi que ce soit, testons la Raspberry Pi et la Keyrah v2. Keyrah supporte Vic-20, C16, C64 / C64C, C128 / 128D, C116, Plus/4, Amiga 600 et 1200. Chaque modèle est gravé sur le PCB pour ne pas se tromper.

Pour la connexion USB entre la carte et la Pi 3, vous pouvez passer par le port USB ou via le connecteur du PCB. Attention : pour passer en interne, vous devez souder des headers et posséder un câble USB interne. Pour le test, le plus simple est de passer par le port USB.

Avant tout chose, téléchargez une distribution C64. Nous avons opté pour la Combian. Pas forcément la plus simple à configurer, mais sans doute une des meilleures distributions C64. Site : https://cmaiolino.wordpress.com

Vous devez demander un accès à son développeur puis télécharger l'image système pour la graver sur une carte micro-SD. Attention : optez pour une carte 16 Go minimum et avec un indice de performance élevé. Insérez ensuite la SD dans le port SD de la Pi 3.

On connecte la nappe du clavier sur les headers C64/VIC20 de la Keyrah. Surtout ne pas oublier de configurer les jumpers situés à droite des connecteurs blancs A1200 / A600. Ils permettent de configurer le clavier 8-bit des Commodore (donc le C64) et des Amiga. Vous avez le choix entre la configuration allemande et américaine. Nous optons pour la configuration allemande : on met donc le jumper sur les 2 pins du haut (à partir de la gauche). Ne pas dépla-

cer les jumpers de droite. On connecte l'USB entre la Keyrah et la Pi 3, sans oublier un écran HDMI et éventuellement un clavier USB par sécurité.
On branche l'alimentation de la Pi. Si tout va bien, l'écran affiche le Basic du C64.

Le montage

Passons aux choses sérieuses : le montage. Celui-ci n'est pas difficile. Il faut penser à agencer les câbles et à les fixer. La position de la Pi n'est pas l'élément important, car nous avons décidé d'ajouter des connecteurs HDMI / Micro-USB / Ethernet / USB.

1 On fixe les différents connecteurs sur la coque. Pour aller vite, nous utilisons la pâte epoxy qui sèche en 24h. Ce n'est pas le plus joli, mais c'est le plus rapide. Au préalable on retire les caches de connecteurs. On dispose maintenant des connecteurs HDMI, Ethernet, micro-USB et USB sur la face arrière de notre C64
2 On connecte l'ensemble des câbles : la nappe du clavier, la LED du C64, USB, HDMI, Ethernet, etc.
3 La fixation de la Pi se fait sur la coque. Il faudrait percer au moins deux trous pour les vis.

Nous n'avons finalement pas utilisé les supports et caches imprimés en 3D. Les ajustements sont trop approximatifs pour être exploitables tels quels. Ce sera pour une future v2.

On allume

Etape cruciale : on branche l'alimentation de notre C64. La LED du boîtier doit s'allumer et après quelques dizaines de secondes, l'écran affiche le BASIC. On teste quelques touches du clavier.
Si tout est OK. On éteint et on ajuste les deux parties du boîtier et on visse.
Idéalement, il faudrait mettre en place un bouton d'alimentation. Ce montage nécessite quelques soudures, un interrupteur. Là encore, on verra cela dans une future v2.

Combian64

La distribution Combian64 n'est pas forcément la plus facile à utiliser, mais c'est sans doute la mieux adaptée à notre usage. Vous pouvez utiliser une distribution de rétro-gaming si vous voulez. Mais attention : n'oubliez pas qu'il faut obligatoirement le support de la Keyrah.
Tout d'abord, nous allons récupérer des jeux et logiciels C64. Vous avez l'embarras du choix sur Internet. Formatez une clé USB en FAT32 puis copiez l'ensemble des logiciels à la racine.
On presse sur la touche F8 pour accéder au menu principal de l'émulateur VICE (utilisé par Combian64). On sélectionne le dernier item « Quit » pour lancer Combian64 et accéder à l'ensemble des options de la distribution.
Par sécurité, on peut prendre dans la partie Set the emulator to run at boot : Commodore 64 (option 4).
Allez jusqu'à l'option 18 : RASPI-CONFIG -> Entrée -> Option 7 Advanced Options -> A1 Expand Filesystem.
L'option A1 permet d'étendre la partition et d'utiliser l'ensemble du stockage disponible sur la carte SD.
Retournez au menu principal.
Il faut copier les programmes de notre clé USB sur la carte SD pour pouvoir utiliser facilement les logiciels. Pour ce faire, nous allons utiliser l'option 15 : X-Copy. L'opération sera plus ou moins longue selon le nombre de programmes présents sur la clé. Préférez les extensions .D64 pour les binaires C64.

Place aux jeux !

Encore un petit effort ! La fonction X-COPY est terminée. Nous pouvons maintenant quitter les options Combian64 : option 2. Vous retournez sur le BASIC et enfoncez F8 pour retrouver le menu VICE.
La gestion des fichiers n'est pas toujours simple avec Combian64. Pour faire nos premiers tests, nous allons attacher au Drive une image programme :

- Drive -> Attach disk image to drive 8
- Dans le répertoire Game, on sélectionne le programme que l'on souhaite. Entrée. Puis F8 pour retourner au Basic.

On tape la commande LOAD ''*'',8,1 puis Entrée. Le programme se charge automatiquement. Si vous voyez READY sur l'écran, tapez la commande RUN puis Entrée. Le programme se lance.
La mémoire est limitée comme sur un vrai C64. En cas de manque de RAM, rebootez. Vous pouvez faire un reset (logiciel ou matériel) directement dans le menu VICE.
Vous pouvez connecter un joystick sur le port manette de la Keyrah.
Attention : la configuration des joysticks est parfois difficile et aléatoire. N'oubliez pas de configurer les keysets pour chaque manette. Et surtout : ne pas oublier de sauvegarder les paramètres en cours. Tout se déroule depuis les menus VICE. Malgré une bonne configuration (Joystick avec mapping sur le keyset défini), nous avons eu des inversions de positions de la manette. Pas pratique quand on joue.

On termine le montage

Si tout fonctionne bien, vous pouvez maintenant tout refermer et visser les deux parties du boîtier. Assurez-vous que le clavier est bien fixé. Enjoy ! •

Denis Duplan
Sociologue et développeur à ses heures.
Blog : http://www.stashofcode.fr

Une brève archéologie des *diskmags* sur Amiga

L'histoire de la micro-informatique est avant tout celle des micro-informaticiens. A ce titre, elle peut être notamment documentée à l'aide des témoignages publiés par les fanas des technologies qui se sont succédés. Avant l'avènement du Net, c'est notamment par le moyen de fanzines sur disquettes que de tels témoignages ont pu circuler. Une brève archéologie des diskmags produits par la communauté Amiga à la fin des années 80 démontre la richesse du matériau disponible.

Quelques splash screens de diskmags.

Un des aspects de la scène Amiga

L'Amiga fut la dernière gamme de machines grand public de Commodore. Sa commercialisation fut un échec aux Etats-Unis mais un succès en Europe. Sorti en 1987, soit deux ans après l'Amiga 1000, l'Amiga 500, grand rival de l'Atari 520 STF, suscita sur le vieux continent l'émergence d'une communauté internationale d'afficionados. Parmi ces derniers, d'aucuns constituaient une bande à part, relevant de ce qu'il convenait d'appeler – et qu'il convient toujours d'appeler, car fort heureusement, elle demeure –, *la scène*. 1

La scène était avant tout structurée en groupes conduisant des activités allant du licite à l'illicite, voire au carrément mafieux. En cette fin des années 80, s'y côtoyaient ainsi des groupes de *demomakers* composés de *coders* associés à des graphistes et à des musiciens pour produire des démonstrations de savoir-faire à contempler – les démos, intros et autre cracktros –, et des groupes commercialisant le fruit de leurs piratages, composés de *leaders*, *suppliers*, *crackers* et autres *swappers*. Ajoutons à cela des *sysops* administrant des *bulletins board systems* (BBS), c'est-à-dire des serveurs de fichiers tournant sur Amiga rendus accessibles par modems, et l'inventaire des acteurs de la scène sera à peu près complet. Ah si ! J'allais oublier l'essentiel, à savoir l'utilisateur de base, souvent représenté de manière caricaturale en la personne du *lamer*. On y reviendra.

De tout cela, il ne serait resté que d'heureux souvenirs dans la mémoire des jeunes – et moins jeunes – d'antan si jamais certains n'avaient pas alors entrepris, sans le savoir, de se faire les chroniqueurs de leur époque en rapportant sur le moment ce qu'ils considéraient en être l'actualité. C'est le fanzine sur disquette, le *diskmag*, qui devait alors leur servir de média.

Des *diskmags*, il y en eut à la pelle : *Chit Chat*, *Cracker Journal*, *D.I.S.C.*, *Deadlock*, *Fourth Dimension*, *Grapevine*, *Hack-Mag*, *Maggy*, *McDisk*, *Outer Limits*, *P.E.N.I.S.*, *R.A.W.*, *Rage!*, *Scene Lyrics*, *Scene Talk*, *Stolen Data*, *Suicid*, *Top Secret*, *Trade Center*, *Upstream*, *Visual Intensity*, *Zine*, pour ne citer qu'eux. Les titres semblent suggérer ce qu'ils recèlent, des informations croustillantes sur ce qui pouvait se passer dans l'entre-soi de la scène. Toutefois, la consultation des menus listant les articles révèle combien il serait superficiel de se croire tombé sur des versions microcosmiques de *Gala*.

Une interface souvent léchée

Prenons *Grapevine* #8, édité par le groupe LSD. Liste des articles, navigation en cliquant sur le titre dans cette liste ou en saisissant un numéro de page, barre de menu pour accéder aux principales sections, texte scrollant dans la trame au pixel et musique de bonne qualité qu'on peut couper et rétablir : s'il n'en fallait guère pour que le lecteur soit satisfait – ce n'est que de nos jours qu'on déplorerait l'absence d'un moteur de recherche et d'un moyen d'exporter les articles. 2

L'Amiga 500 de Commodore, sorti en 1985.

1

2 *Grapevine #8, publié en 1992.*

Encore fallait-il que cela soit au rendez-vous. De fait, certains *diskmags* pouvaient faire plus frustre. Par exemple, *Stolen Data #7* souffre de la comparaison. Ici, les articles sont en petite quantité, ce qui doit peut-être tant à la ligne éditoriale qu'à l'interface, laquelle ne permet d'accéder aux articles qu'en cliquant sur les touches de fonction en nombre limité. A la décharge d'Anarchy, le groupe éditant *Stolen Data*, il y a deux années d'écart entre les numéros des *diskmags* cités. Une éternité offrant l'opportunité de beaucoup s'améliorer dans le monde de la micro-informatique en général, dans celui de la scène en particulier… 3

Vers la fin, l'interface de certains *diskmags* était particulièrement léchée, comme en témoigne celle de *R.A.W. #9* qui exploitait les nouvelles capacités graphiques de l'Amiga 1200 doté du chipset AGA pour afficher en 256 couleurs. 4

Le *diskmag* lui-même était souvent précédé d'une petite démonstration et/ou d'une série d'images produites pour l'occasion – des *splash screens*, dira-t-on. Une manière agréable d'accueillir le lecteur, voire de le faire patienter le temps que le programme principal soit décompacté en arrière-plan – avec 880 Ko de disponible sur la disquette au plus, un *diskmag* était généralement compacté, ce qui nécessite d'ailleurs de les faire tourner dans un émulateur pour accéder à ses articles de nos jours. 5

Une mine d'informations sur la scène

Que trouve-t-on donc au menu de *Grapevine #8* ? Le contenu listé dans la table des matières se révèle particulièrement éclectique.

Tout d'abord, le classique éditorial, qui en l'espèce mentionne l'existence d'un système de mots de passe donnant accès à un article secret – une marque de fabrique de la rédaction *Grapevine*, il semble. En l'occurrence, c'est « recoded » suivi d'Entrée qu'il faut saisir après avoir pressé ESC, ce qui donne accès à une liste des messages de membres de LSD à d'autres membres de la scène.

Par exemple, Watchman / LSD – dans la scène, on s'identifie par un couple [pseudonyme] / [groupe] – félicite Monty Python / LSD pour la qualité de son code : « *How come you can code is so fucking good. Are you an alien or something?* » Plus intelligible que le message de Torch / LSD à Marc Panther / Soudwave : « *Oh, go on. Tell me. I'll be your best friend for life. Honest!* »

Rien de bien secret ici. Ce sont des *greetings*, un mode de communication par l'intermédiaire de messages publics dont l'objet est de contribuer à la réputation du destinataire tout en contribuant à celle de l'auteur par l'étalage de son réseau de relations. Pas de production de la scène qui ne contienne de *greetings* ; c'eut été comme publier un article scientifique sans mentionner une bibliographie – parfois pas toujours utilisée ! Du *name dropping* dira-t-on, le contenu des messages relevant souvent de la *private joke* et étant donc totalement opaque pour un tiers, au point que souvent les *greetings* se limitaient à une liste de [pseudonyme] / [groupe] ? Sans doute, mais pas seulement, car si un thème revient dans les propos tenus par ceux qui décrivent la scène dans les *diskmags*, c'est avant tout celui de *friendship*.

Tournons les pages, ou faisons-les défiler pour être plus exact. Après une aide, voici une partie adressée aux contributeurs. Il apparaît que tous les articles sont les bienvenus : « *hints & tips, opinions, reviews, jokes, silly or serious stories, cartoon strips and just about anything else you can think of!* » C'est sans doute moins par souci d'ouverture que de remplissage que cette ligne éditoriale est affichée : il faut trouver du contenu pour alimenter les numéros qui, vers la fin de l'Amiga, comprendront parfois jusqu'à deux disquettes de 880 Ko de données compactées – dont des images, toujours volumineuses, il est vrai.

Des instructions sont prodiguées à l'attention des potentiels contributeurs. L'époque n'est pas encore au HTML, et chaque *diskmag* s'appuie sur un système spécifique pour gérer une présentation du texte qui restera toujours sommaire, mais efficace. En ce qui concerne *Grapevine*, il est possible de spécifier la couleur du texte, de diviser de moitié son épaisseur, de doubler sa hauteur, et de le faire clignoter en le précédant de séquences de caractères particulières. Ainsi, `{2{9{8{7Hello` permet d'afficher « Hello » avec des caractères d'un pixel d'épaisseur, en rouge, double hauteur, clignotant.

La partie suivante est consacrée à l'abonnement. Moyennant 8 livres sterling à faire parvenir par courrier postal (« *cash is preferred* »), il est possible de s'assurer de l'expédition des six numéros annuels le jour de leur sortie. L'abonnement rapportait-il ? Rien ne permet de le savoir, mais il y a lieu d'en douter. Peut-être cela rapportait-il à d'autres, tout particulièrement les *PD distri-*

3 *Stolen Data #7, publié en 1990.*

4 *La belle interface de R.A.W. #9, pour Amiga doté du chipset AGA uniquement.*

5 *Un splash screen de Grapevine #8 par Watchman / LSD.*

6 *Avertissement aux PD distributors au chargement de Grapevine #8.*

7 *Annonce du groupe Dianetix au format image dans Grapevine #8.*

butors, c'est-à-dire ceux qui avaient monté de petites affaires pour distribuer moyennant finance des logiciels supposés être dans le domaine public. *Grapevine* #8 s'ouvre par un écran enjoignant ces distributeurs – souvent perçus comme des parasites par une scène dont ils monnayaient les productions – de ne pas faire payer plus de 2 livres sterling le numéro. La relation aux *PD distributors* est d'ailleurs traitée sur cinq pages par Phreak / Jesters dans « *Rip Me Off, Please!* », plus loin dans *Grapevine* #8 – qui lui vaut un cinglant commentaire de Torch / LSD : « *Most of the people responsible for the demos etc in such libraries are 'scene' members. In other words, software pirates. They seem to think its perfectly okay for them to copy.* » 6

Les annonces n'étaient même pas payantes dans *Grapevine*, mais ce n'était pas le cas pour tous les *diskmags*. Par exemple, les instructions aux annonceurs potentiels prodiguées dans *Zine* #10 stipulent qu'il faut payer 10 francs suisse (la somme est bientôt déclinée dans toutes les devises) pour trois annonces de 18 lignes, 34 caractères par ligne. *Grapevine* #8 contient bon nombre d'annonces, sous la forme de textes jouant sur les caractères pour produire des graphismes ou sous la forme d'images plus ou moins travaillées. 7

Qui annonce dans les *diskmags* ? Des groupes qui cherchent à s'étoffer en recrutant un musicien, un graphiste ou un *coder*. D'autres qui cherchent à attirer du monde sur leur *BBS* – dont il est possible qu'ils commercialisent l'accès pour télécharger des logiciels piratés à ceux que contribuer rebute – un ratio *disabled* sur un gros BBS allemand, que rêver de plus ? Des particuliers qui veulent *swapper*, comprendre échanger par courrier postal des disquettes contenant des logiciels ici encore souvent piratés, mais pas que… Et aussi des groupes qui donnent exclusivement dans l'illégal, commercialisant des abonnements pour recevoir par courrier postal les derniers jeux dès leur sortie, des machines pour simuler les cartouches de jeu Super Nintendo, etc.

Mais le contenu, le contenu ! Le voici, classé en plusieurs rubriques : *News & Events*, *Group Information*, *Interviews*, *Computer Information*, *Amiga Scene Information*, *Humour*, *Quizz*, *Miscelaneous*. Et si cela ne devait pas suffire, qu'on sache que dans le dernier *Grapevine* #21 qui ne compte pas moins de 200 articles, on ira jusqu'à trouver : *General Computing*, *Computer Gaming*, *Hacking And Telecommunications*, *Coder's Corner*, *Amazing But True*, *Bizarre*, *Drugs* (traitant le sujet au fond, comme par exemple « *Should Cannabis Be Legalized?* »), *Fiction*, *Movies*, *Television And Other Media*, *Music*, *Fantasy Role-Playing Gaming*, *Poetry*, *Vegetarianism And Animals*. Les rubriques sont parfois ad hoc, au gré des articles qui parviennent aux rédactions.

Parmi les rubriques incontournables, il faut relever *News & Events*, où la rédaction de *Grapevine* #8 a regroupé les dernières informations sur la vie de la scène. On y trouve notamment des articles annonçant de futures *parties*, inévitablement centrées sur des concours de démos où les groupes du monde entier sont invités à étaler les talents de leurs *coders*, graphistes et musiciens durant quelques jours. Des comptes rendus de *parties* qui se sont tenues aussi. Tiens, en voici une qui s'est tenue à Nesles-la-Vallée à l'initiative du groupe IRIS. Elle aurait réuni plus de 400 personnes – l'entrée n'était pourtant pas donnée : 60 francs, ou 40 francs pour qui venait accompagné de… son ordinateur. Eh, l'Amiga 500 n'était pas un portable ! Allez-vous donc vous balader avec ! A noter que Shadow / LSD, qui tient la plume, rapporte entre autres un fait intéressant, à savoir que la *party* était sponsorisée par l'éditeur de jeux Titus et par Commodore France. « *As all you know, only legal stuff, tool and programs were accepted* », il précise, ce qui ne surprend pas quand on se rappelle les liens entre Titus et l'Agence de Protection des Programmes (APP).

Group Information réunit des présentations de groupes. S'agissant d'articles rédigés par des membres des groupes en question, la tonalité est évidemment hagiographique, mais cela n'en retire pas l'intérêt. Bien au contraire, il est très instructif de constater ce que les auteurs considèrent comme essentiel à dire, donc ce qui fait un groupe à leurs yeux. Ainsi, le groupe End Of Century 1999 montre qu'un groupe, c'est une identité collective à défendre : « *We decided to put this in because certain people claim to be in EOC 1999 and yet they were "not", so here is a full member list* ». Et l'appel aux coders qui suit l'annonce des productions en cours vient rappeler qu'un groupe n'existe que tant qu'il produit, si bien qu'il lui est indispensable de recruter. Trouver les talents n'apparaît d'ailleurs pas chose aisée. Ainsi, Loco / Paragon présente son groupe comme le résultat de la fusion de Bohemians et Artic Force pour cause de pénurie : « *The reason why we formed a totally new group, was that Artic Force was in a terrible need of graphicians.* » Toutefois, il ne faut pas sous-estimer la nécessité pour un individu de rejoindre un groupe pour exister dans la scène – ceux qui peuvent être graphiste, musicien et coder à la fois sont rarissimes. Andy / Doom – au passage, « *We desperately need a musician!* » – explique « *TKI joined us, as he was in need of a group to join, and so did T2 as well* » et encore que « *Deathlok/Eclipse (formelly Killers) left Killers in search of a new group to join.* » Au passage, il faut constater que la scène est labile : les groupes vont et viennent dans la scène, de même que les membres vont et

viennent dans les groupes.

Ce n'est là qu'une infime partie des renseignements qu'on peut tirer de la lecture des articles de la rubrique. Mais comme si cela ne suffisait pas, la rubrique *Interviews* contribue encore plus à nous édifier sur la sociologie de la scène à travers le récit que ceux qui la forment font non seulement de leurs groupes, mais aussi d'eux-mêmes. Il s'avère que les âges sont les plus divers, même s'il faut supposer que la moyenne doit friser la vingtaine : Clauz / Extreme a 17 ans, Pride / Brainstorm en a 20, Numitor / Pulsar en a 26, et Franz / Italian Bad Boys en a 40 ! Les uns et les autres sont notamment interrogés sur leur profil, leur hobbies, leur vision de la scène nationale et internationale – sur leur pays, les propos des uns et des autres sont parfois cruels : « *(lame?)* » interroge le français Clauz / Extreme ; « *a bit lame* » déclare le danois Krueger / Kefrens –, leur classement personnel de ces groupes et membres les plus méritants, leur groupe et ses projets.

Dans *Computer Information*, les articles abordent les sujets les plus variés en lien avec la technique. Dans « *Coder's Corner* », Orcrist / LSD explique – bien succinctement, mais tout de même – comment programmer de la 3D en temps réel en s'appuyant notamment sur le blitter, un des coprocesseurs graphiques de l'Amiga. Plus loin, Ken D. passe en revue l'Amiga 500+, une nouvelle version de l'Amiga 500 doté de coprocesseurs légèrement améliorés, mais surtout d'un nouveau système d'exploitation en ROM : le Kickstart 2.0 remplaçant le Kickstart 1.3. Dans l'article suivant, Darren Ewaniuk apporte un complément utile en expliquant comment installer un interrupteur pour basculer entre les deux ROM, et ainsi résoudre les problèmes de comptabilité ascendante. Dans un tout autre registre, celui beaucoup plus obscur du *hacking*, Shadowrex rapporte les consignes prodiguées par AT&T à ses clients pour lutter contre le piratage de *calling cards* – avec le détournement de PABX d'entreprise, le procédé en vogue pour appeler à moindre frais des BBS à l'étranger, comme le prouvera le démantèlement de tout un réseau fin 1994. Enfin, pour rester dans l'actualité du moment, rajoutons qu'un anonyme dresse sur par moins de neuf pages un inventaire des différents *game doctors* sur le marché – « *"Game Doctor" is the name giving to devices that can read the ROM on the video game cartridges and store it on floppy disk* », pour ceux qui l'ignoreraient encore.

La rubrique *Amiga Scene Information* relève presque du divers tant le contenu est éclectique. Tous les articles sont toutefois centrés sur la scène, leur objet étant d'en commenter les pratiques ou de les exposer de manière pédagogique. En tant que tels, leur lecture permet de se faire une idée des problématiques du quotidien qu'affrontaient les possesseurs d'Amiga impliqués d'une manière ou d'une autre dans la scène.

Et d'abord, comment alimenter le lecteur de disquette vorace de la bête ? Dans un passionnant « *A Swapper Guide* » de pas moins de sept pages, Phreak / Jesters explique notamment comment pratiquer le *stamp faking* pour limiter les frais de port et échanger ainsi plus facilement des disquettes par courrier postal. Ce n'est pas en recouvrant le timbre de résine qu'on y parvient, il explique, avant de dérouler sa méthode en cinq étapes consistant à imprimer de faux timbres, qu'il juge imparable – « *Fools them everytime* ». 8

Mais enfin, « *Piracy - Is It Really A Crime?* » Sur deux pages, mUB / LSD disserte de la question pour en venir à dénoncer les éditeurs de jeux qui font de l'argent sur le dos des développeurs et qui mentent en prétendant que si leurs logiciels étaient moins piratés, ils seraient moins onéreux. Il conclut par un jugement de Salomon devant la logique duquel ne reculerait sans doute pas un économiste marxiste : « *Keep the piracy balance level continue on your quest to pirate software (though it is immoral etc.) but don't be too greedy (let the developper live but strangle the capitalist)* ».

D'autres problématiques sont abordées, qui concernent moins le *lamer*. Le *ripping*, c'est-à-dire le vol de code, de graphisme, de musique et même d'articles pour être réutilisés dans d'autres productions par des individus peu scrupuleux qui peuvent aller jusqu'à se les approprier est une source de préoccupation récurrente. Fort heureusement, un *diskmag* offre une tribune pour faire justice. Aussi Steinar / Addonic en profite-t-il pour se lamenter sur les trois pages de « *Ripping Is Not The Only Sin* » que Quartz a sorti un *music disk* – une compilation de musiques – où le groupe a repris

8 *Comment tromper la royale Poste selon Phreak / Jesters dans Grapevine #8.*

sans lui demander sa permission une des compositions en écorchant son titre, en oubliant de lui attribuer, et en poussant l'audace jusqu'à modifier un instrument – « *This is the jewel in the crown !* » On comprend qu'être accusé de *ripping*, ce n'était pas rien dans un milieu où la hiérarchie symbolique se fondait sur le mérite personnel. Car comme Conquest / Anarchy l'explique dans « *Lamers…Get Lost!* » publié dans *Stolen Data* #10, « *There are two kinds of scene: The 'good' one also known as the elite and the bad one which consists of guys called 'lamers' by the elite one! The first one works very hard to set new standards on Amiga… they only produce quality stuff and want to see back stuff of the same quality while the lamers rip it all to get a few credits and to boast around!* » On reviendra plus loin sur la figure du *lamer*, le *ripping* n'étant pas le seul crime de lèse-majesté qu'il puisse commettre.

Noter que c'est dans « *Rogues Gallery* » de la rubrique *Amiga Scene Information* que résident de rares photos de l'époque, numérisées avec plus ou moins de bonheur. L'occasion de mettre des visages – qui pour certains sont bien ceux d'adolescents de leur époque ! – sur Ford / LSD, Ken D., ainsi que Mutant Mango et Shame, tous deux membres de Quartz.

Pour le reste, un contenu riche de son éclectisme

Les rubriques *Humor* et *Quiz* qui suivent sont de moindre intérêt pour se replonger dans l'époque. De ce point de vue, *Miscellaneous* est plus édifiante, permettant d'accéder non plus seulement à des aspects de

la scène, mais plus généralement à ceux de la société d'alors.

En décrivant la manière de bidouiller un « *extremely effective "bugging" transmitter with a maximum power output of 2 watts! Tunable 88 to 100Mhz, of course…* », Maxima / LSD nous rappelle l'univers disparu de la CB. James 3 / Pol?rone produit un guide très complet des épisodes de *Blake 7*, présentée comme une série de science-fiction de la BBC très populaire. MFD / The Silents débat d'un sujet crucial « *from my own observations made over the last fifteen years of regularly watching our national sport* » : quelles sont les équipes de football les plus fortes et les plus faibles du moment ?

L'implication des auteurs dans la rédaction d'articles de *Miscellaneous* varie considérablement. Visiblement, il n'est pas attendu d'eux qu'ils produisent leur propre réflexion, mais qu'ils rapportent quelque chose d'intéressant, quitte à ce que le travail se limite à reprendre un écrit publié ailleurs. Le lecteur passe donc incontinent d'une simple chronique de faits divers repompée – dans « *Fighting Talk* », un anonyme reprend de *GREEN* une liste d'« exploits » commis par des activistes environnementaux – à un récit très personnel – dans « *Going To A Cremation* », mUb / LSD raconte sa première expérience d'une crémation qu'il dépeint sous un jour ridicule. Ce sont les articles de ce dernier registre qui sont les plus riches. Citons notamment « *Attitudes Toward Disabled Peoples* » où Fish / LSD rapporte sur pas moins de 14 pages son expérience de personne handicapée, vissée sur un fauteuil roulant depuis trois ans du fait d'une sclérose en plaques. « *I had friends while I was walking, but as soon I went into the chair, my so-called friends didn't want to know me any more* », il explique. Mais plus loin, pour l'honneur de la scène : « *I was a bit scared on how I would be accepted at first, but most of the Amiga freaks I know have treated me as they would any other able-bodied person, except they don't always think I may need help with my chair into the car or stuff like that.* » Poignant.

Et de là, le lecteur passe à tout autre chose, car dans « *Home Distilling Or How To Make Rocket Fuel* », Pazza / LSD explique par le menu comment, en cherchant à s'enivrer avec son pote Monty Python / sans-doute-du-même-groupe un soir de dèche, ils ont bricolé un alambic à l'aide d'une cocotte-minute pour distiller du vin rouge dans l'espoir d'en tirer un ersatz de whisky… Dans *Miscellaneous*, tout est décidemment à l'avenant.

Qui a du temps à tuer peut se faire meurtrier en lisant tout ce qui lui tombe sous la main. C'est l'opportunité de découvertes. Rétrospectivement, on trouvera à l'éclectisme des articles des *diskmags* non seulement le charme, mais l'utilité, d'avoir attiré l'attention de lecteurs sur des sujets auxquels ils ne se seraient jamais intéressés par défaut – qui plus est en anglais. De quoi donner matière à réflexion quand désormais l'accès à l'écrit se fait via un moteur de recherche qui ne retourne jamais que les articles traitant précisément du sujet sur lequel on les interroge…

Le lamer, bouc-émissaire de la scène

Chose promise, chose due. Il a été évoqué plusieurs fois, et il faut maintenant en dire plus à son sujet. S'il est un acteur de la scène qui n'est pas à la fête dans les *diskmags*, c'est bien le *lamer*.

A partir du numéro 6, *Stolen Data* publie la « *Lame letter of the issue* » qui permet de mieux cerner ce qui est reproché à cet acteur de la scène frappé d'ostracisme. Mole / Anarchy commente un courrier de Phoenix / Timecode, qui lui adresse deux disquettes pour commencer à échanger avec lui : une liste de ce qu'il détient et une version certainement piratée de *Wings of Fury*. Que n'a-t-il pas fait ! « *For the first and the last time I AM NOT A FUCKING CHARITY!* », écrit un Mole qui estime visiblement ne pas avoir de temps à perdre avec quelqu'un qui prétend accéder à lui – même si c'est visiblement aussi humblement que poliment. Visiblement très irrité, Mole / Anarchy voue Timecode aux gémonies en promettant de se moquer du groupe systématiquement dans les numéros à venir du *diskmag*.

Le *lamer*, c'est celui qui voudrait en être mais dont la maladresse – ou, circonstance tout à fait aggravante, la cuistrerie – démontre qu'il ne le peut pas. C'est un repoussoir absolu, et il n'y a visiblement rien de pire que d'être accusé d'être un *lamer*, surtout dans la tribune publique que constitue un *diskmag*. Tout impétrant est un *lamer* en puissance, aussi est-il attendu de lui qu'il fasse profil bas, qu'il n'ose pas déranger ceux au niveau desquels il ne s'est pas hissé et que surtout il ne s'auréole que du mérite que les autres lui reconnaissent à la lumière des fruits de son labeur, et pas d'une once de plus.

A moins que ce ne soit l'inverse, le terme de *lamer* a donné celui de *lame* pour désigner une activité qui ne présente pas d'intérêt. Dans *Stolen Data* #8, Chromag / Cult revient dans « *Cool Or Lame Swapping!* » avec un peu d'amertume sur sa pratique intensive du *swapping*, qu'il juge rétrospectivement *lame* : « *What I want to say is that hanging in front of XCopy and just copying, copying, copying… is not the way to get friends.* » Comme quoi, on n'est jamais à l'abri de soi-même.

Il ne faut pas pour autant rester sur l'image d'une scène dont la passion des membres serait de désigner le bouc-émissaire. Pour finir sur une note plus consensuelle, on citera donc Franz / Italian Bad Boys, dans l'interview précédemment citée : « *I think the best thing the Amiga given me was that thanks to this machine I've a lot of friends around all the world, guys that I never saw and probably never will I see, and we are really friends, as dudes. This is a marvellous thing!* »

Pour finir… et pour toujours

La sortie des *diskmags* – des mensuels au plus – ne respectait pas toujours une périodicité rigoureuse pour des raisons diverses et variées, dont certaines paraîtront certainement familières à qui a un jour travaillé dans la rédaction d'un magazine des plus classiques. Ainsi, dans sa contribution à l'éditorial de *Grapevine* #21, devenu bimensuel à partir du #9 par manque de temps, Oedipus / LSD s'excuse du retard en expliquant que le disque dur de Plazza / LSD a crashé, entraînant la perte de nombreux articles. Il explique aussi que l'organisation de l'équipe a changé, le dit Plazza ayant renoncé à son rôle de rédacteur en chef pour passer la main à Oedipus.

La durée de vie des *diskmags* doit s'apprécier à l'aune de celle de la scène Amiga qui n'aura somme toute duré que quelques années. L'Amiga 500 apparaît en 1987, et Commodore met la clef sous la porte en 1994. Deux ans auparavant, le constructeur avait tenté de lutter contre la déferlante

du PC en renouvelant sa gamme, notamment avec l'Amiga 1200, mais la tentative avait échoué. La scène Amiga ne survit pas, du moins sous la forme qu'on pouvait lui connaître. Quelques années, c'est bref, mais il suffit de comparer la *Megademo* de Red Sector Inc. de 1989 et *How 2 Skin A Cat* de Melon Dezign en 1993 pour comprendre qu'en très peu de temps, la scène a pu révolutionner ses codes.

La consultation d'une liste de *diskmags* laisse entendre que ce n'est que vers la toute fin des années 80, et même plutôt le début des années 90, que le genre s'est développé sur Amiga. Les derniers numéros de nombre d'entre eux remontent à la disparition de Commodore. *Grapevine* #21 sort en 1995, *Top Secret* #12 en 1994, *R.A.W.* #9 en 1995, etc. La disparition de l'Amiga a nécessairement contribué à tuer les *diskmags*, mais il ne faut pas non plus sous-estimer d'autres facteurs. De fait, les *diskmags* cités ont fait preuve d'une exceptionnelle longévité, car rares sont ceux qui ont dépassé les vingt numéros. Pourquoi ?

Ici encore, la lecture des articles renseigne sur ce qui a pu survenir. Dans *Grapevine* #21, Plazza / LSD fait ses adieux sur trois pages. « *Now I have a job, a wife and a house (I moved from a flat). I have other hobbies as well as my Amiga I wish to pursue, and a busy social life.* », il explique. La vie c'est la vie, et elle ne se déroule pas que sur écran. Ou encore, dans *Stolen Data* #9, RokDaZone étale sur quatre pages son mécontentement de voir les rédactions de *diskmags* si peu récompensées de leur labeur. « *So, why are editors still regarded at as second class dudes?* », il s'insurge après avoir rappelé qu'un *diskmag* sans contenu ne saurait présenter d'intérêt. Il n'annonce pas renoncer pour autant, mais il l'a décidemment bien mauvaise. Sans doute, le manque de reconnaissance a-t-il pu en décourager d'autres. Dans *Chit Chat* #14, r@T@p signe un éditorial comminatoire dans lequel il reproche aux lecteurs de ne pas assez contribuer : « *If the support fort issue 15+ goes on this particular one we will definitely STOP making Chit Chat !!* » Menace mise à exécution d'ailleurs, *Chit Chat* s'étant arrêté au numéro suivant…

Peut-être certains *diskmags* se sont-ils poursuivis un temps sur le Net, à l'exemple de *R.A.W.* dont le numéro 9 annonçait avec fierté le succès de *R.A.W. on-line*, avec plus de 300 hits par jour. Le site était accessible via une URL qui aujourd'hui a pris le cachet du vintage parce que renvoyant à un répertoire d'utilisateur, http://www.xs4all.com/~blahh, mais il a bien évidemment depuis longtemps disparu en laissant moins de trace que la version disquette – de quoi s'inquiéter plus généralement pour la préservation de tout contenu publié en ligne.

Un quart de siècle après, le fruit du travail laborieux des rédactions et des rédacteurs a fort heureusement été conservé, et la seule lecture superficielle d'un numéro d'un *diskmag* entreprise plus tôt veut démontrer la richesse du matériau que ce travail constitue pour reconstituer une période révolue. Qui s'intéresse aux *diskmags* en trouvera facilement des palanquées sur plus d'un site Web à l'aide d'une recherche à base des mots clés « Amiga » et « diskmag », tout simplement. On murmure que le site http://grandis.nu/ donne accès à un site FTP dont le répertoire Commodore_Amiga contiendrait un répertoire TOSEC (*The Old School Emulation Center*, pour les intimes), lequel contiendrait un gigantesque répertoire Commodore Amiga - Diskmags, l'équivalent de la bibliothèque d'Alexandrie en ce qui concerne les feuilles de choux numériques dont il a été question ici. Mais chut ! Je ne vous ai rien dit…

Dans une belle critique du film *Les invasions barbares*, Gérard Allard revient sur ce fascinant passage où la caméra s'attarde sur quelques-uns des ouvrages trônant dans la bibliothèque d'un intellectuel disparu. Rebondissant sur l'évocation du journal de Samuel Pepys, l'auteur écrit : « *cet homme, qui n'était pas grand-chose, a raconté son époque sans le savoir, en se racontant tout bêtement.* » Cette phrase si joliment tournée, je veux l'emprunter pour dire ce que les auteurs des textes, parfois très personnels, que j'ai évoqués peuvent apporter de nos jours. Si contribuer à des fanzines sur disquettes a pu paraître superficiel à l'époque, cela n'apparaît pas l'avoir été rétrospectivement. Mais quelle surprise ? Il n'y a jamais rien d'inutile dans ce qu'on fait dès lors que c'est créatif, quitte à ne pas l'être sur l'instant.

Pour en savoir plus…

Sur l'histoire de Commodore qui est aussi une grande partie de l'histoire de la micro-informatique, il est indispensable de lire le fascinant ouvrage *Commodore: A Company on the Edge* de Brian Bagnall publié par Variant Press en 2010.

Sur l'histoire de l'Amiga en particulier, le second opus du documentaire *From bedrooms to millions* sous-titré *The Amiga Years*, sorti en 2016, est tout aussi incontournable :

http://www.frombedroomstobillions.com/amiga

A noter, *Got Papers?*, une initiative de préservation du patrimoine de la scène qu'il faut applaudir à tout rompre. L'idée est de collecter les courriers papier qui pouvaient être échangés entre membres de la scène de tout niveau :

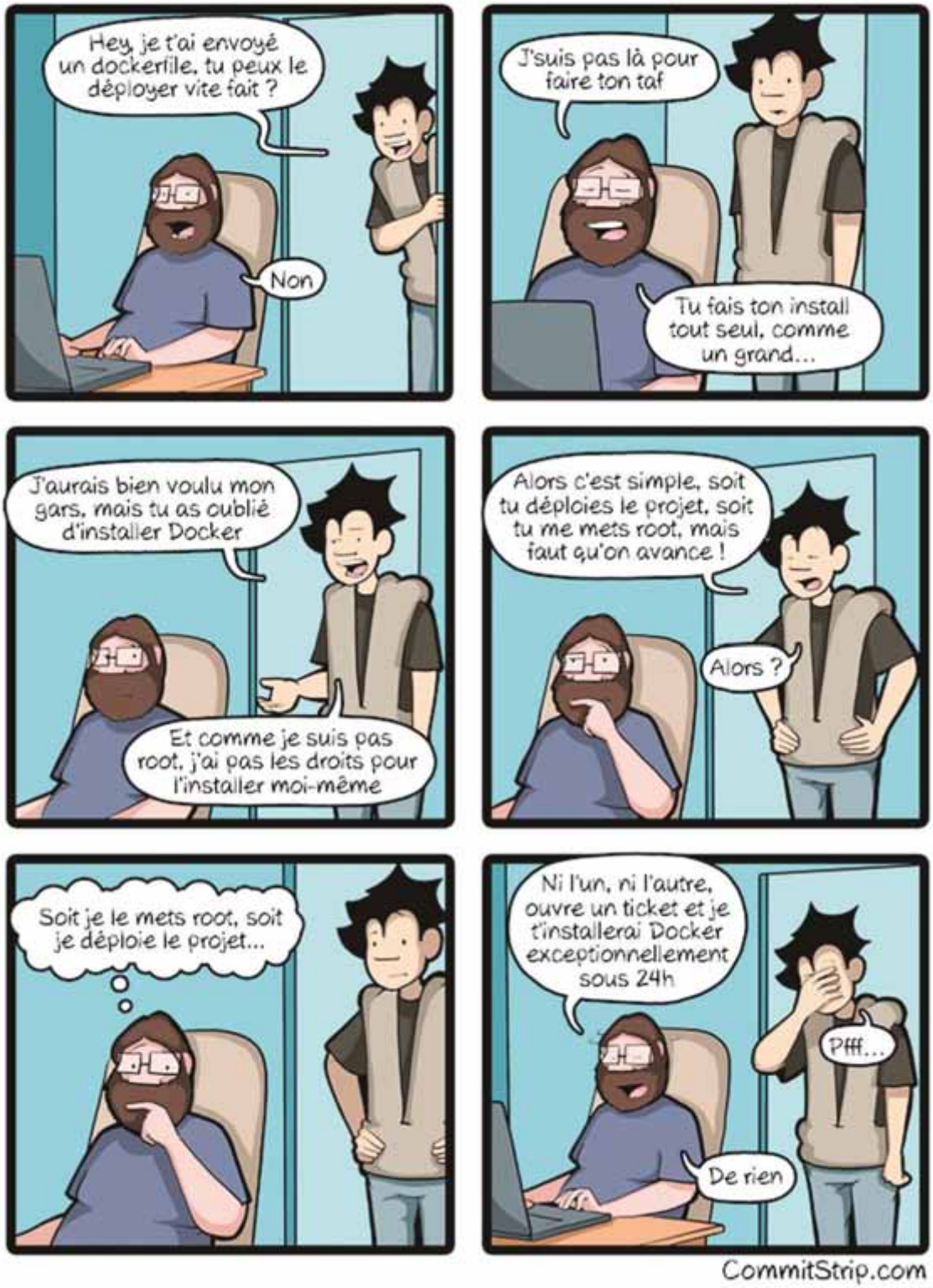
http://gotpapers.untergrund.net/

Enfin, pour ceux qui apprécient le bon français et qui ont apprécié le film, la critique du film *Les invasions barbares* de Gérard Allard citée en conclusion est disponible sur :

http://www.revueargument.ca/article/2003-10-01/257-variations-sur-des-themes-invasifs.html

•

Quand le sysadmin file un coup de main



Une publication Nefer-IT, 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08 - Directeur de la publication & Rédacteur en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Ont collaboré à ce numéro : S. Saurel
Nos experts techniques : D. Duplan, S. Pontoreau, F. Pigere, J. Demangeon, L. Devulder, K. Boutouil, J-B Da Costa, G. Hubert, D. Panza, J. Tran, U. de Bellabre, c. Prudent, S. Bovo, J. Giacomini, J. Antoine, S. Cordonnier, C. Gigax.
Couverture : © StockPhotoAstur - **Maquette** : Pierre Sandré.
Publicité : François Tonic / Nefer-IT - Tél. : 09 86 73 61 08 - 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
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908 - © NEFER-IT / Programmez, mai 2018

Ce numéro comporte un encart jeté PC Soft pour les abonnés.

Abonnement :
Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex. - Tél. : 01 55 56 70 55 - *abonnements.programmez @groupe-gli.com*
Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30.

Tarifs
Abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 € - Autres pays : nous consulter.

PDF
35 € (monde entier) souscription sur www.programmez.com

OPÉRATION POUR 1 EURO DE PLUS

Pour bénéficier de cette offre exceptionnelle, il suffit de commander WINDEV Mobile 23 (ou WINDEV 23, ou WEBDEV 23) chez PC SOFT au tarif catalogue avant le 29 Juin 2018. Pour 1 Euro de plus, vous recevrez alors le ou les magnifiques matériels que vous aurez choisis. Offre réservée aux sociétés, administrations, mairies, GIE et professions libérales, en France métropolitaine. L'offre s'applique sur le tarif catalogue uniquement. **Voir tous les détails sur : WWW.PCSOFT.FR ou appelez-nous au 04.67.032.032** Le Logiciel et le matériel peuvent être acquis séparément. Tarif du Logiciel au prix catalogue de 1.650 Euros HT (1.980,00 TTC). Merci de vous connecter au site www.pcsoft.fr pour consulter la liste des prix des matériels. Tarifs modifiables sans préavis.

WINDEV est un logiciel professionnel. Document non contractuel.

JUSQU'AU 29 JUIN

COMMANDEZ WINDEV MOBILE 23

OU WEBDEV 23 OU WINDEV 23

ET RECEVEZ LE NOUVEL iPhone X

Aucun abonnement à souscrire.
Compatible tous opérateurs

CHOISISSEZ :

- **iPhone X** 64GB
  ou
- **iPhone 8** 256GB
  ou
- **iPhone 8 Plus** 64GB
  ou
- **MacBook Air** 13,3" 128GB
  ou
- **lot de 2 iPad New** 9,7" 128GB

(Détails et **autres matériels** sur www.pcsoft.fr)

Choix de la couleur sur le site

Atelier de Génie Logiciel Professionnel cross-plateformes

Apple® iPhone® iPad® iPad Air® iPad Mini™ sont des marques déposées de la société Apple. Apple n'est pas un organisateur ou un sponsor de cette opération.

WWW.PCSOFT.FR