CLEAN CODE / SQL / DOMOTIQUE / VENOM / GIT / 2020 / 42

FLUTTER / REACT-NATIVE / HTML / ARCHITECTURE / TESTS / 2020 / GIT

PROGRAMMEZ!

Le magazine des développeurs

01/2020 N°236 22e année

/DÉVELOPPEUSES
le code est à vous !

© Octocat / GitHub

/DÉVELOPPEMENT FLUTTER
Mode sombre : pourquoi l'utiliser ?
Démarrer en REACT-NATIVE

/Saskia codeuse à 14 ans

/Gérer l'accessibilité en HTML
/Débuter en architecture hexagonale

Un numéro exceptionnel codé et buildé avec les Duchess France

M 04319 - 236 - F: 6,50 € - RD

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

Olymp

La formation nouvelle génération

Le logiciel indispensable aux centres de formation pour profiter des avantages de la formation présentielle, et de la formation en ligne.

Edité par

IdeaStudio

Le langage de programmation, fils caché des jeux ?

Je suis née dans les années 80, ancienne gameuse lors de mes études, puis plus du tout. Mais je m'y remets depuis peu. La gameuse s'est réveillée. Je ne joue pas à n'importe quoi, j'avoue je suis plus "retro-gaming", "vieux jeux », particulièrement sur Nintendo NES et tout ce qui va jusqu'à la Nintendo 64. En version portable et salon ! Je ne penses pas que l'on plonge dans le rétro-gaming par hasard.

J'entends déjà, « Mais dis donc Aurélie, pourquoi nous parles-tu de cela dans un édito, tu ne t'es pas trompée de magazine/de cible ? »
Non, non, rassurez-vous. Je sais où je vais dans cet édito (one point, parfois je galère à trouver un sens aux miens, François T., NDLR).

Ces vieux jeux ont eu un cycle de vie. Ils ont été joués maintes fois, par des personnes différentes, des contextes différents, ils se sont forgés une solide réputation (bonne ou mauvaise), et nous sommes parfois attirés par la nouveauté, par le tout dernier jeux qui vient de sortir sur la Switch ou bien sur XBox One X. Et quand on y pense, il s'agit de la même chose concernant les langages de programmation.

On utilise encore, pour des besoins/contextes/projets spécifiques, ou bien par connivence, familiarité, facilité, des "vieux" langages parce qu'ils ont fait leurs preuves, qu'ils ont une solide communauté et on est tout aussi bien attiré par la "nouveauté" avec des langages tels que Rust ou bien par le nouveau framework JavaScript à la mode (gare au phénomène de mode et à l'overdose dans ce dernier cas).

Tout cela pour vous dire, que la nouvelle année est là. Je vous souhaite, lectrices et lecteurs de Programmez!, une année 2020 riche en expériences, en développement (professionnel et personnel), en découvertes, en bugs(1), en debug et surtout en réussites.

Merci François de m'avoir laissé gérer un 2e numéro de ce magazine, j'espère qu'il vous plaira. (2)

Bonne année et bonne lecture ! :-)

Aurélie Vache
Duchess France

(1) Comme vous le savez, ce n'est pas un bug mais une fonctionnalité. Oui je sais c'est facile, mais nous pouvons le faire. NDLR

(2) Je ne pouvais pas builder Programmez! 236 et jouer à "Link's Awakening". Le choix a été difficile :-) NDLR

SOMMAIRE

MESSAGE PRIORITAIRE :
Programmez! n°237, prochain numéro date stellaire : 31.01.2020

Delphi revient sur Linux / Deno va-t-il défier Node.js ? / Focus sur Golang / DevOps : côté développeur

5G : l'appel d'offres aura (encore) du retard

L'un des gros dossiers en cours pour la mise en place de la technologie 5G concerne l'attribution des fréquences qui pourront être utilisées par les opérateurs sur leurs réseaux 5G. Cela fait plusieurs semaines que les opérateurs attendent de Bercy un calendrier précis pour le processus d'attribution des fréquences, qui prendra la forme d'une enchère entre les quatre opérateurs. Des dissensions entre l'ARCEP et le ministère des Finances seraient à l'origine de ces multiples retards, mais le gouvernement espère clore le processus d'ici Juin 2020.

Xerox et HP : quand ça veut pas…

Xerox lorgne sur son concurrent : nous vous expliquions le mois dernier que l'imprimeur avait proposé 33 milliards de $ pour racheter HP Inc. Ça n'était pas passé auprès du board de HP Inc., mais Xerox agite maintenant le spectre d'un passage en force, via une offre publique d'achat hostile. Les dirigeants de Xerox ont donc commencé à rencontrer les actionnaires de HP Inc. afin de les convaincre de céder leurs actions. Mais le board de HP campe toujours sur ces positions : le prix proposé n'est tout simplement pas suffisant.

La Chine ne veut plus du matériel américain

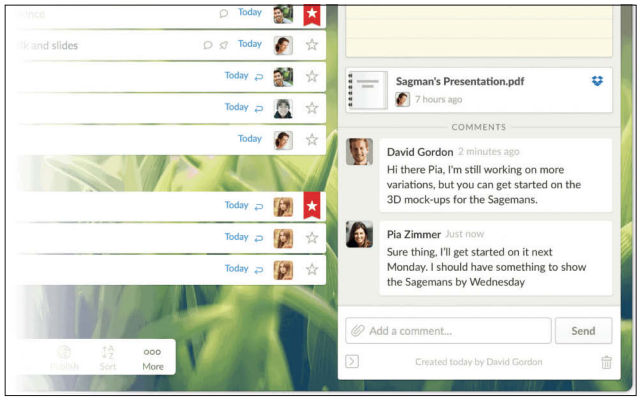
Une directive émise par le gouvernement chinois prévoit que les administrations du pays devront se passer de matériel et de logiciel étranger d'ici trois ans. Une nouvelle étape dans la guerre commerciale que se livrent les États-Unis et la Chine : les États-Unis ont notamment banni l'équipementier Huawei des infrastructures 5G, invoquant des craintes pour la sécurité nationale. La Chine relance donc de son côté avec ses propres mesures de rétorsion. Autant dire que l'ambiance entre les deux grandes puissances est visiblement au beau fixe.

Avast fait la lumière sur son goût pour les données de navigation

Firefox a supprimé quatre extensions de navigateurs publiées par Avast sur son magasin d'extension, au motif que celles-ci récupéraient l'historique de navigation des utilisateurs. Wladimir Palant, créateur d'Adblock Plus, avait dénoncé cette collecte publiquement sur son blog, ce qui a poussé le nouveau PDG de l'antivirus tchèque, Ondrej Vicek, à s'expliquer auprès de Forbes. Il a donc expliqué qu'Avast exploitait bien les données de navigations anonymisées de ses utilisateurs pour revendre des services d'analyse de tendance sur le marché au travers de sa filiale Jumpshot. Plutôt ennuyeux pour un acteur qui veut se réorienter sur le marché de la protection de la vie privée.

Wunderlist tire sa révérence

En mai 2020, Microsoft sonnera le glas de son logiciel Wunderlist. Ce logiciel était utilisé pour tenir à jour ses ToDo list sur Windows et mobiles, mais sera inutilisable à partir du 6 mai 2020. À cette date, Microsoft conseillera à ses utilisateurs de basculer vers son nouveau logiciel dédié : Microsoft To Do. L'éditeur a prévu le coup et promet qu'il sera possible de synchroniser les listes de Wunderlist sur Microsoft To Do. Ouf !



CHU de Rouen : l'ANSSI sur le pied de guerre

Le CHU de Rouen a été la cible d'un ransomware qui a contraint l'établissement à fonctionner plusieurs jours avec du papier et des crayons. Banale cyberattaque ? Pas vraiment : l'établissement a rapidement reçu l'assistance d'agents de l'Anssi, venus accompagner l'établissement face à cette attaque. Les premiers soupçons se tournent vers le ransomware CLOP utilisé par le groupe TA505. Un groupe bien connu des services : on leur attribue la paternité du malware bancaire Dridex, ainsi que plusieurs vagues de diffusion du rançongiciel Locky, qui avait fait de nombreuses victimes en 2016. L'ANSSI a laissé entendre que le groupe se serait lancé dans une vaste campagne de phishing durant le mois de novembre, et qu'il chercherait de nouvelles cibles à extorquer.

Vous vous souvenez de Magic Leap ?

Magic Leap, ça vous dit quelque chose ? La société promettait des lunettes de réalité augmentée à grand renfort de vidéo promotionnelle laissant entrevoir un futur en réalité virtuelle resplendissant, à tel point que certains doutaient franchement des promesses du constructeur. Le site the Information affirme avoir eu un aperçu des premières ventes de la société et celles-ci se révèlent particulièrement décevantes : 6 000 exemplaires vendus en 6 mois. Ennuyeux, surtout quand on apprend que la startup américaine tablait plutôt sur 100 000 ventes sur sa première année. Les licenciements se multiplient et la société cherche désespérément des fonds pour pouvoir proposer une nouvelle version de son produit. Un grand bond en avant, mais vers quoi ? Difficile à dire.

La fondation Raspberry Pi annonce 30 millions de Pi vendues dans le monde !

WALLIX

CYBERSECURITY SIMPLIFIED

PRENEZ DE LA HAUTEUR

EN GARDANT L'ESPRIT D'ÉQUIPE !

REJOIGNEZ UN LEADER EUROPÉEN DE LA CYBERSÉCURITÉ

A l'ère de la transformation numérique, les enjeux de cybersécurité touchent l'ensemble des entreprises. WALLIX a réussi à combiner trois exigences fondamentales : la gestion des identités, le contrôle des accès et la protection des données, grâce à des solutions simples à utiliser et faciles à déployer. Les solutions WALLIX accompagnent aujourd'hui plus de 1000 entreprises et organisations dans la gestion de leur cybersécurité, la sécurité des données et la conformité réglementaire.

PRENEZ DE LA HAUTEUR ! REJOIGNEZ UN LEADER !

La culture WALLIX, fortement imprégnée d'un esprit entrepreneurial, passe par des valeurs humaines comme l'audace, l'engagement, un fort esprit d'équipe, et le goût du défi... ET C'EST TOUT VOUS !

Plus d'informations sur www.wallix.com

2020

Janvier

14 : MongoDB Paris / Paris

Venez rencontrer MongoDB durant une journée dédiée à la base de données et à l'écosystème. La plus grande conférence MongoDB en France offre plein d'occasions d'apprendre et rencontrer les meilleurs experts. Un évènement à ne pas rater !
Site : https://www.mongodb.com/local/paris

22-25 : SnowCamp/Grenoble

SnowCamp est une grande conférence rassemblant dévs, ops et architectes. On y trouve une journée de formation, 2 jours de conférences et une journée d'échanges sur les pistes.
Site : http://snowcamp.io/fr/

28 : Duck Conf 2020 / Paris

La Duck Conf est la conférence technique OCTO consacrée a l'architecture de SI. Chaque année, nous construisons un programme à destination des architectes techniques, architectes de données et d'entreprise, tech lead et experts : DevOps, Big Data, Scalabilité, Cloud, Résilience, Rénovation de legacy… https://www.laduckconf.com/

31 : Touraine Tech/Tours

Nouvelle édition de Touraine Tech. Au programme : 18 sessions, des ateliers. On y parlera design, UI, mobile, big data, cloud, jeux, IoT, architectures !
Site : https://touraine.tech

Février

1 & 2 : FOSDEM 2020/Bruxelles

L'évènement incontournable des développeurs open source. C'est l'occasion de découvrir des projets et surtout d'autres développeurs. On échange, on discute, on oppose les arguments.
Site : https://fosdem.org/2019/

14 : DevFest Paris/Paris

La DevFest revient à Paris mi-février. L'événement promet de superbes sessions, des animations, et beaucoup de technos et de rencontres ! 36 sessions sont prévues !
Site : https://devfest.gdgparis.com

25-27 : BreizhCamp/Rennes

Le BreizhCamp, c'est 3 jours de conférence à Rennes. Créé en 2011 à l'initiative du BreizhJUG, ils sont en cours de planification de la 10ème édition. Le BreizhCamp propose de se faire rencontrer une communauté de développeurs et d'experts, avec un contenu à la carte sur plus de 100 thèmes présentés. Chaque participant est libre de suivre les sujets qui ont retenu son attention ou de préférer les ateliers pour mettre en pratique les connaissances acquises.
https://www.breizhcamp.org

28 : DevFest du bout du monde/Brest

C'est à la pointe de la Bretagne. Et on va y parler langages, codes et technologies. L'événement breton pour les développeurs ! site :
https://devfest.duboutdumonde.bzh

Avril

15-17 : Devoxx France/Paris

24 : Serverless Days / Paris

L'architecture serverless se répand rapidement dans les entreprises et les infrastructures. Une journée complète pour discuter avec les meilleurs experts du domaine : https://paris.serverlessdays.io

29 & 30 : MixIT/Lyon

De nombreux thèmes sont abordés : design, technologie, makers, éthique dans l'IT, style de vie, travail en équipe, etc. https://mixitconf.org

LES CONFÉRENCES DOT EN FRANCE

3 février : dotSwift / Paris

La conférence Swift revient à Paris. Avec les dernières évolutions du langage, le nouveau framework UI, il y aura de nombreuses choses à découvrir.

2 mars : dotPy/Paris

Une des plus importantes conférences en France sur le langage Python !
https://www.dotpy.io

30 mars : dotGo/Paris

Comme son nom l'indique, la conférence dot 100 % langage Go !

Mai

13, 14 & 15 : RivieraDev/Sophia Antipolis

+40 sessions techniques sont attendues pour cette édition 2020. https://rivieradev.fr

13 & 15 : Devoxx UK - UK (Londres)

Merci à Aurélie Vache pour la liste 2020, consultable sur son GitHub :
https://github.com/scraly/developers-conferences-agenda/blob/master/README.md

NOUVEAU

Retour vers le passé :

redécouvrez les ordinosaures et les technologies des années 1970 à 2000 !

Commandez directement sur **programmez.com**

6,66 € *(+frais de port*)* **36 pages**

*Revue trimestrielle. Editée par Nefer-IT. *Avec frais de port : 7,66 €*

Roadmap 2020 : que dit notre boule de cristal ?

Nouvelle année, nouvelles prédictions. Eh oui, que serait un début d'année sans prédictions, sans le top des supers tendances ? Tout de suite, ce serait moins fun. Honnêtement, nous ne voyons pas forcément de révolutions ni gros big bang. Plusieurs technos vont simplement continuer à s'imposer. Mais comme toujours, une top techno d'aujourd'hui peut rapidement disparaître. Un peu comme les chansons d'été ou les nouvelles danses.

La rédaction

BADASS

Python

Sans surprise, Python est la vedette de ces dernières années. Tous les index de popularités des langages le placent dans le top 3 ou 5. Le langage évolue lentement ce qui permet de ne pas bousculer la communauté. Le passage à la v3, contre la v2, n'a pas toujours été simple à gérer.
Python est un des langages préférés pour l'apprentissage de la programmation, en machine learning, en big data.

Niveau tendance	Type de dévs	Pourquoi suivre ?	Attention !
*****	Un peu de tout, sauf le dév mobile et le pur natif	C'est LE langage pour apprendre, pour le machine learning, l'IoT / DIY. Langage assez facile à apprendre. Communauté très dynamique	Plusieurs déclinaisons existent (Python - MicroPython), regarder les différences. Python reste fondamentalement de l'interpréter

Librairie / framework JavaScript

L'écosystème JavaScript est plus actif que jamais ! La multiplication des librairies et frameworks contribue à cette performance. Vue.JS, Vanilla, REACT, Ionic, Angular (même si c'est du TypeScript) renforcent JS sur le web et les apps mobiles.
A surveiller en 2020. Mais attention tout de même : la pérennité des composants et frameworks JS se posent à terme. Ce qui est populaire en 2020 ne le sera pas forcément en 2021 ou 2022 !

Laravel : la nouvelle star du PHP

Pas forcément le plus mis en avant et pourtant, Laravel est devenu une des références du monde PHP. Sa communauté est active. Il possède une flatteuse réputation. C'est le grand concurrent de Symfony. Si vous ne connaissez pas encore, n'hésitez pas !

Flutter : il a tout d'un grand

Google mise beaucoup sur Flutter et avec raison. Il propose un environnement pour les apps mobiles tout en supportant le multiplateforme. Il évolue rapidement et il possède déjà une belle communauté et une documentation technique fournie.
Nous sommes plus sceptiques sur les versions desktop et web, annoncées par Google au printemps dernier.

Niveau tendance	Type de dévs	Pourquoi suivre ?	Attention !
*****	Dév mobile avant tout	C'est la techno mobile du moment	Les autres déclinaisons ont-elles réellement un intérêt ?

BOLOSS

Le monde technologie est sans pitié. Hier, elles étaient au top, aujourd'hui, elles sont toujours là mais avec un certain désintérêt ou une tendance plate.

Java

Sommes-nous méprisants envers Java ? Surtout que l'univers Java est toujours aussi actif et présent mais après 25 ans d'existence, il faut reconnaître que Java n'est plus aujourd'hui le langage qui avait su s'imposer. Oui, il reste une valeur sûre du développement et du recrutement. Oui, Java est omniprésent : le patrimoine applicatif est tellement énorme. Mais cela ne signifie pas qu'il est une techno à suivre.
Java connaît une vie chaotique depuis le rachat de Sun par Oracle. Les évolutions majeures se faisaient tous les 2-3 ans. Désormais, tous les 6 mois, une nouvelle version sort. Mais cette cadence infernale pose une question de fond : à quoi ça sert ? Car les véritables nouveautés sont moins fréquentes. La multiplication des JDK n'aide pas à s'y retrouver.
Posons la question honnêtement : quel avenir pour Java et surtout a-t-il réellement un avenir ? Même si personne n'ose réellement le dire, il faudra bien, un jour, répondre à ces questions. Au-delà, Oralce n'aurait-il pas intéressé à lâcher définitivement Java comme il l'a fait pour JavaEE ?

Niveau tendance	Type de dévs	Pourquoi suivre ?	Attention !
***	Langage à tout faire sauf le mobile	Reste le langage d'entreprise et son patrimoine applicatif est considérable	L'intérêt d'une version tous les 6 mois

Ruby : peu mieux faire

Ah Ruby ! Que dire de ce langage qui était parti pour devenir LA star du développement ? Finalement, son intérêt retomba rapidement mais il continue à avoir une solide communauté et il évolue régulièrement. Restons lucide, il ne fait plus parti des tops langages mais n'hésitez pas à garder un œil sur lui.

Zend Framework : de la splendeur à la décadence

Comment passer de star PHP au looser en 3 ans ? Zend Framework était un des acteurs majeurs du monde PHP et des CMS. Depuis le rachat par Rogue Wave, c'est une véritable déchéance accélérée. En novembre 2018, plusieurs développeurs références avaient quittés Zend.
En avril 2019, Zend annonce Laminas : Zend Framework open source. Le but est que la communauté fasse vivre et évoluer le framework !
Y-a-t-il encore un intérêt de suivre Zend Framework ? Pour l'existant ?
Par nostalgie ?

Niveau tendance	Type de dévs	Pourquoi suivre ?	Attention !
*	Tout ce qui est PHP	Bonne question si vous avez une réponse, merci de nous prévenir	Clairement la mort du framework

Développeur fullstack

Là encore, soyons honnêtes : la notion même de fullstack était une absurdité. Malheureusement, des entreprises, des ESN cherchent encore et toujours le développeur maîtrisant tout et n'importe quoi. Si en plus, il comprend les plans de montages IKEA, ce serait un plus. Avec un salaire « on t'exploite ok mais tu es content tu as un job ». Faut pas déconner. Si 2020 marque la fin du dév fullstack en France ce serait déjà un progrès !

ZONE 51 (BIENVENUE DANS LA 4E DIMENSION)

De nombreuses tendances ou technologies excitent les communautés ou tout simplement les médias. Mais sur le terrain, elles soient soit absentes ou peu utilisées ou tout simplement elles vivent leurs vies car bien installées dans le paysage IT. Cette zone est parfois difficile à cerner. Certaines technos devraient être en badass ou en boloss mais nous avons choisi une position plus neutre.

C++

C++ est un langage inoxidable : souvent déclaré pour mort, il est bel et bien vivant. La norme 2020 promet encore de belles choses dans les prochains mois. Il demeure une référence pour le dév embarqué / IoT et bien entendu tout ce qui est natif !

Niveau tendance	Type de dévs	Pourquoi suivre ?	Attention !
**	Tout	C++ reste un langage majeur	A voir comment le langage va pouvoir évoluer avec les multiples évolutions prévues

Docker : paf, le rachat !

Il y a quelque de chose de pourri dans le royaume des conteneurs. Docker était devenu la nouvelle star de l'IT et de l'infrastructure. Mais avec un sérieux problème de design : le modèle économique. Et l'argent est devenu la question au cœur de sa survie. En automne dernier, Docker vend la partie entreprise à Mirantis, change (encore) de patron et se concentre sur le développeur. Pour continuer, l'éditeur a reçu 35 millions $ de financement.
Stop ou encore ? Réponse en 2020.

Niveau tendance	Type de dévs	Pourquoi suivre ?	Attention !
****	Nouvelles architectures, DevOps	1 mot : conteneur	Docker est désormais dans une zone de turbulence. Attention au décrochage rapide

Réalité virtuelle

Les casques et lunettes de réalités augmentées ont su trouver des usages concrets en entreprise, dans l'industrie et même dans le médical. Quid du grand public ? Il faut avouer que depuis 2 ans, la VR/VA n'arrive pas à décoller. Et pourtant on voit ici et là des usages réels et intéressants mais il nous manque toujours l'usage ou l'application qui va faire tout changer.
Les plateformes ne sont pas, ou peu, compatibles entre elles (sur les contenus), un contenu parfois limité, l'autonomie limitée des casques autonome. Pour le moment, nous ne voyons pas à court terme d'évolutions majeures.

.Net : des questions sur son évolution

Après .Net Core 3.0 attendu depuis longtemps, Microsoft regarde vers le futur de sa plateforme .net en général. Aujourd'hui, on dispose de .Net standard, .Net Core, .Net Framework. Les développeurs s'y perdent et ils n'évoluent pas forcément de la même manière.
La prochaine grosse étape sera .Net 5, prévu pour fin 2020. L'ambition est très grande : réunifier le monde .Net avec .Net Standard et .Net Core dans un unique environnement capable de cibler tous les marchés (Desktop, Web, Cloud, Mobile, jeux, IoT, IA). A voir comment cette nouvelle approche se mettra en place et les conséquence sur les codes actuels.
Microsoft adopte une politique de mise à jour / évolution agressive : 1 version majeur annuelle ! Bonne ou mauvais idée ?

Quantique

Ah le quantique. Tout le monde en parle. Mais finalement, peu de personnes savent réellement comment cela fonctionne et pourquoi c'est intéressant.
Mais soyons clairs : le quantique ne sera pas dans nos ordinateurs avant 10-15 ans et encore, avec quel intérêt ?

Niveau tendance	Type de dévs	Pourquoi suivre ?	Attention !
****	Calculs, analyses, sécurité, cryptographie	L'ordinateur quantique évolue rapidement et son impact sera important	Il existe plusieurs approches quantiques, notamment sur les processeurs

Le top des technos 2020

Technos	Catégorie	Tendance
Go	Langage	*****
Rust	Langage	*****
Kotlin & SWIFT	Langages	*****
TypeScript	Langage	*****
Craftmanship	Méthode	*****
Microservice	Architecture	*****
RISC-V	Microprocesseur	****
Langages fonctionnels	Langages	***
Kubernetes	Infrastructure	*****
PHP	Langage	***
Quarkus	Framework	***
DevOps	Méthode / philosophie	****
Dino	Serveur JS	?
PWA	Type d'app	**
API	Composant IT	****

Technos installées & matures 2020

Technos	Catégorie	Tendance
Cloud Computing	Plateforme	*****
IoT	Plateforme	****
Cobol	Langage	=
C#	Langage	=
Cordova	Plateforme dév	=
Tests	Méthodes / outils	=
Sécurité	Méthode / outils	*****
Scrum	Méthode	=

Saskia Blanc, 14 ans, rôle modèle

Il y a quelques années, à l'occasion d'une édition de Devoxx France, j'ai eu la chance d'assister à un talk « Le jaillissement de l'esprit » présenté par la famille Blanc (Sébastien, Saskia et Loïs) puis j'ai pu rencontrer Saskia Blanc lors d'un BOF (Bird of Feather) durant cet événement et de l'introniser « Duchess » ;-). Quelques années plus tard, nous invitons Saskia à faire la keynote de fermeture de la première édition du DevFest Pitchouns, à Toulouse, et je profite de cette occasion pour l'interviewer, car, pour moi, c'est un vrai rôle modèle ! :-)

Aurélie Vache

Bonjour Saskia, peux-tu te présenter ?
Bonjour ! Alors je m'appelle Saskia, j'ai 14 ans et je suis en 3e. Je suis passionnée par la programmation et plus tard, je veux devenir développeuse.

Depuis combien de temps es-tu tombée dans la marmite de la programmation ? Et comment es-tu tombée dedans ? As-tu un ou des langages de programmation de prédilection ou bien que tu n'aimes pas ?
Je suis tombée dedans grâce à mon père, quand j'étais en CE2, il y a 6 ans. J'adore Scratch, car c'est avec ça que j'ai découvert le code, mais j'aime aussi beaucoup Groovy et Python que je viens de découvrir ;). Pour l'instant, il n'y a pas de langage de programmation que je n'aime pas.

Tu as déjà été speaker lors de conférence technique et tu as même donné des keynotes ! Wow ! Peux-tu nous en dire plus et nous raconter cette expérience ?
En effet, j'ai donné des keynotes à Riviera Dev, Voxxed Days Luxembourg et à Devoxx Maroc ainsi qu'une conférence technique à Devoxx France et durant la DevFest Toulouse d'octobre ! Cette expérience a été l'une des plus incroyables de ma vie, car parler de ma passion et partager mes connaissances devant plein de monde a été vraiment extraordinaire !

Note : *Pour la petite histoire, elle a également fait la keynote de fin de la première édition du DevFest Pitchouns — une conférence pour les enfants par les enfants — le 2 octobre 2019 à Toulouse, puis le lendemain elle à co-présenter, au côté de son père et de son frère, le « jaillissement de l'esprit » - le talk a encore une fois fait un carton !*

J'ai vu que tu as créé ta chaîne YouTube « Le code pour ados ». Peux-tu nous en dire plus ?
J'ai créé la chaîne YouTube « Le Code pour Ados » pour donner envie aux enfants et adolescents de programmer et pour partager ma passion ! Je vais créer des vidéos qui expliquent clairement comment programmer avec Scratch, Groovy (peut-être même avec Python) et comment utiliser Makey Makey ! Par exemple, je prépare en ce moment une vidéo pour apprendre à faire un jeu avec Scratch !

En France on compte environ 10% de développeuses. Ce nombre est semblable en Europe. Qu'en penses-tu ?
Je pense que ce nombre est vraiment bas et que si une fille rêve de devenir

développeuse, elle ne doit pas hésiter une seule seconde et accomplir cela ! En toute modestie, j'espère pouvoir donner l'exemple !

Quels conseils voudrais-tu donner aux enfants ainsi qu'aux jeunes filles ?
Si vous êtes intéressé(e)s et que vous vous posez des questions sur la fabrication des jeux vidéo, des animations… N'hésitez pas et tentez de coder ! Notre génération a la chance d'avoir internet et YouTube, on peut trouver très facilement des tutos sur n'importe quel langage ! Alors, lancez-vous !

C'est presque fini, à présent tu peux nous dire tout ce que tu veux sur toi, c'est une question ouverte :-).
Je ne me limite pas qu'à la programmation, car j'ai d'autres passions, comme la guitare, la lecture (j'adore Harry Potter), les arts créatifs, le karaté…

URL de la chaîne YouTube : https://www.youtube.com/channel/UCx_wiwB_z2IgoOurhfipyFA

François Tonic

42 : objectif parité parfaite

Comment attirer des jeunes femmes dans les cursus informatique et notamment techniques ? C'est une question que se posent la plupart des écoles et des formations. 42 n'a pas échappé à ce problème et mène une politique volontariste depuis l'arrivée de sa nouvelle direction en 2018. Nous avons voulu en savoir plus avec Sophie Viger, directrice de 42.

Une femme à la tête d'une grande école informatique c'est plutôt rare. Sophie Viger a pris la tête de l'école dédiée au développement, 42. Actuellement, 85 % des grandes écoles sont dirigées par des hommes.

Toutes les écoles d'informatique ont du mal à recruter des jeunes femmes notamment dans le développement. Dans d'autres domaines, comme l'interface, le web, les femmes sont plus nombreuses. « *À 42, en année complète, nous arrivons à 21 %. C'est à la fois bien et triste. Mais, nous avons fait + 50 % en un an* » commente Sophie.

Un objectif ambitieux

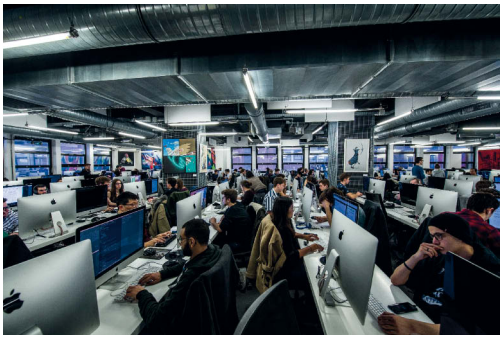
Comparé à d'autres écoles, c'est un niveau remarquable. Comment expliquer cette « explosion » ? « *Nous avons réalisé de gros efforts et mené de nombreuses actions envers les jeunes femmes et les femmes. Par exemple, à la dernière piscine, épreuve si redoutée, nous avions 33 % de femmes ! Notre objectif est clair : arriver à 30-35 % de femmes dans la prochaine promotion* » poursuit la directrice.

> ***« Aspect psychologique : 1 tiers ce n'est plus une minorité, le sentiment change » Sophie/42***

Que le développeur soit une femme ou un homme, cela importe peu. « *Il faut juger sur les compétences techniques. Est-ce un simple vœu ?* » s'interroge Sophie. Effectivement, un développeur, quel que soit son sexe, sa religion, son origine, devrait être jugé pour ces compétences, ces qualités. Plus simple à dire qu'à faire ?

« Pour nous avoir plus de femmes développeurs n'est pas une option, mais une nécessité, car nous voulons accueillir tous les talents d'où qu'ils viennent, parce que l'économie en a besoin et parce qu'in fine l'univers du numérique doit être à l'image de la société. Pour l'instant ce n'est pas le cas pour diverses raisons. Cela a un impact sur les femmes elles-mêmes qui voudraient aller vers ces métiers et c'est pour ça que nous mettons tout en œuvre pour y remédier. Ce n'est pas de la démagogie de dire qu'il n'y a pas (ou peu) de diversité. Si nous voulons des femmes parmi les grands patrons de demain, parmi l'élite qui dirigera, pour une société plus juste, représentative et efficace, il faut plus de femmes dans le numérique. » analyse Sophie.

> ***« De la diversité et pas uniquement de la mixité »***



Comment 42 a attiré plus de femmes ?

Le profil a-t-il changé à 42 ? Un des éléments favorables a été le levé de la limite d'âge il y a quelques mois. « *Avant, nous avions une limite d'âge à 30 ans. Aujourd'hui, nous avons des étudiants de 30 à 60 ans ! Sur les 30-40 ans, nous avons plus de femmes que d'hommes. Il s'agit souvent des reconversions professionnelles. Les adolescents ont plus touché à l'informatique que les adolescentes. La sensibilisation à l'informatique est souvent plus tardive. Au-delà, nous avons toujours des profils de jeunes femmes dont 42 est la première formation, qu'elles aient le bac ou non. Nous avons aussi des startupeuses sans diplômes, des femmes diplômées cherchant à acquérir de nouvelles compétences.* » explique la directrice de 42.

L'angoisse de la piscine ?

Que serait 42 sans la piscine ? Redoutable épreuve pour espérer obtenir son billet pour 42. Les premières piscines ont montré une sous-performance des femmes. « *Différents facteurs peuvent expliquer cette contre-performance : manque de confiance en soi, le manque de pratique du code, un environnement de jeunes ayant une habitude du code, le stress envers les maths…* » recadre Sophie. Et il y a encore le stéréotype : les garçons seraient meilleurs en technique et en math. La piscine rajoute un environnement, une expérience stressante. Ce n'est pas évident de dépasser cette appréhension.

Comme nous le confirme Sophie, le fait d'être minoritaire peut amener certaines jeunes femmes à la conclusion « *je ne suis pas à ma place sauf si j'en veux ou que je suis une battante* ».

La dernière piscine a montré une évolution par rapport aux années passées. « *Nous avons organisé des moments de paroles, pour booster les candidates, les rassurer, les accompagner, donner de la confiance. Nous avons proposé la même chose pour les candidats, mais peu sont venus. Nous avons aussi proposé des moments mixtes pour ces échanges.* » commente la directrice.

En entretien individuel, 42 constate parfois les mêmes difficultés, les mêmes interrogations chez les candidats. Un effet de groupe ? Difficile de le déterminer.

42 fait tout pour proposer un environnement où tous les profils peuvent cohabiter et travailler : profil non binaire, transgenre. Des étudiantes/étudiants ont fait leur coming-out dans l'école, y trouvant un espace pour s'exprimer.

Et les toilettes ?

Question bête : durant notre dernière conférence DevCon à 42, nous avions constaté que les toilettes du bas n'étaient plus mixtes. Nous avons tout naturellement voulu en savoir plus. « *Oui effectivement. Nous avons voulu préserver un espace pour les femmes. Nous avons posé la question aux étudiants. Il y a aussi des problèmes logistiques et pratiques. Les toilettes font aussi douches et ces dernières sont très utilisées durant la piscine, les étudiantes pouvaient se sentir gênées durant ces périodes, par exemple quand elles ont leurs règles. Mais les autres w.c. restent mixtes aux autres étages.* » précise Sophie.

Philippe BOULANGER
Manager des expertises C/C++ et Python
www.invivoo.comauteur

Portrait d'une developpeuse

J'ai rencontré Dorra pour la première fois lors de son entretien technique d'embauche début 2018 que j'ai dirigé. Elle sortait d'une mission conjuguant le C++ avec des mathématiques (de la recherche opérationnelle). Elle m'a alors fait part de son envie d'évoluer vers un nouveau langage, Python, qu'elle ne maîtrisait pas encore bien et était attiré par le machine learning. Elle travaille désormais sur des chatbots et contribuent à deux expertises, Python et machine learning, au sein d'Invivoo.

Quel parcours scolaire as-tu suivi ?
J'ai grandi en Tunisie où j'ai fait presque la totalité de mon parcours académique jusqu'aux classes préparatoires aux grandes écoles. Après les concours, j'ai intégré ***l'ENSTA ParisTech*** en 2010, une école d'ingénieur généraliste française. J'aime beaucoup les mathématiques et la logique infaillible qui régit ces principes, j'ai donc choisi des spécialités qui gravitent autour de ce domaine : statistiques, optimisation et, évidemment recherche opérationnelle. C'est ce dernier domaine qui m'a le plus attiré, c'est pourquoi j'ai choisi d'effectuer en parallèle un Master 2 au ***CNAM*** ; ce qui m'a permis d'approfondir mes connaissances.

Quand as-tu découvert l'informatique et la programmation ?
J'ai découvert l'informatique sur le tard. Quand je suis rentrée en école d'ingénieur beaucoup de disciplines m'intéressaient : les statistiques, les mathématiques, etc… Aujourd'hui, pratiquement tous les métiers qui utilisent, de près ou de loin, avec les mathématiques requièrent des compétences en informatique. C'est donc en commençant à programmer dans le cadre de mes études que j'ai découvert le développement. Etant au centre de nombreuses disciplines, connaître les grands principes et les bonnes pratiques de code m'ont permis de travailler dans plusieurs domaines et sur plusieurs problématiques.

As-tu rencontré des difficultés pour devenir développeuse ?
Je n'ai pas rencontré de difficulté particulière. Mon seul challenge fut de monter en compétence rapidement au début de chacune de mes expériences professionnelles. En effet, il a fallu s'adapter à des technologies et des métiers qui changeaient du tout au tout : apprendre de nouveaux langages par exemple.

Et dans quelles sociétés as-tu évolué ?
J'ai effectué mon stage de fin d'études chez ***Orange*** sur l'optimisation du déploiement du réseau optique « Fiber to the Home » en proposant et développant des algorithmes de mathématiques exactes. Ce qui m'a amené, notamment, à être co-auteur d'un article pour « International Conference on Applied Operational Research » (la conférence ICAOR 2014) sur les différents algorithmes implémentés.
Ma première expérience en CDI fut comme ingénieur en recherche opérationnelle à ***EURODECISION*** en mission à ***Air France***. J'ai travaillé sur les algorithmes de planification des emplois du temps des stewards et des hôtesses. Il y a de nombreuses contraintes à la fois d'ordre légal (pas plus de deux longs courriers consécutifs par exemple) mais aussi d'ordre personnel (je préférerais ne pas travailler tel ou tel weekend). Et compte-tenu du nombre de vols et du nombre de personnels, ce sont des problèmes de grandes tailles qui nécessitent des algorithmes complexes.
Après quelques années, j'ai eu envie de travailler sur un projet à échelle humaine et de pouvoir intervenir à tous les niveaux en ayant une vue d'ensemble : c'est exactement ce que j'ai trouvé à ***Invivoo*** dans l'équipe Machine Learning. Un projet sur lequel je peux intervenir sur toute la chaîne et où je participe activement à la définition des utilisations possibles.

Quel est ton quotidien chez Invivoo ?
J'ai intégré ***Invivoo*** en mars 2018 : c'est une société de conseil et de services qui développe aussi ses propres logiciels tel que la suite XComponent. J'ai intégré l'équipe R&D Machine Learning. Nous sommes une équipe de 3 personnes managées par Mouna Khetab. Nous travaillons à la conception d'une plateforme de création de chatbots. Mon travail principal consiste à coder les différents composants de cet outil. Mais étant dans une petite équipe, j'ai participé à l'écriture des spécifications du front de l'application web, à développer le back-end (principalement les API), mais aussi à découvrir les algorithmes de machine learning servant à tenir les conversations avec les utilisateurs.

Qu'est ce qui te plait dans les expertises d'Invivoo ?
Le principe des expertises est novateur car il permet aux consultants d'avoir de l'aide technique via des formations, des articles de blog, des séances de coaching ou de l'entraide via Teams, un outil de communication instantanée qui leur permet de poser des questions à la communauté. Souvent les consultants restent confinés sur leurs technologies de prédilection ou connexes, grâce aux expertises, chez ***Invivoo*** peuvent avoir accès à un parcours leur permettant de changer de métier.
Aujourd'hui, je participe activement aux expertises Machine Learning et Python. Ce qui est une vraie chance pour moi d'évoluer dans mon métier et mes compétences. J'ai été formé à faire passer les entretiens techniques d'embauche. J'écris régulièrement des articles pour le blog d'***Invivoo*** ou pour le magazine « ***Programmez !*** ». J'ai aussi participé à la préparation de meetups… Autant d'expériences très enrichissantes…

N'est-ce pas dur de concilier vie privée et vie professionnelle ?
En effet, ce n'est pas toujours évident. Dans nos métiers, nous avons des semaines où travailler le soir est presque obligatoire, et parfois, des semaines plus calmes. Respecter les deadlines de mise en production implique forcément de travailler plus sur certaines périodes. J'essaye tant que je peux de sanctuariser ma vie privée.

Quels sont tes objectifs futurs ?
A moyen terme, mon objectif est principal est d'améliorer ma connaissance de Python. A plus long terme, j'aimerais diriger une petite équipe et travailler sur des sujets qui me tiennent à cœur.
En dehors de ces indéniables compétences techniques, Dorra apporte chaque matin avec elle son sourire et sa bonne humeur qu'elle communique à ceux qui interagissent avec elle. Ne change rien, Dorra •

Hyperpanel OS : les applications temps réel accessibles à tous.

Micha Rivault
Expert en architecture logicielle. Plus de 20 ans d'expérience dans les télécoms, spécialiste dans l'intégration de systèmes communicants.
micha.rivault@hyperpanel.fr

Carolino Santonastasi
Ingénieur logiciel chez HyperPanel Lab. 35 ans d'expérience dans les systèmes embarqués temps réel.
carolino.santonastasi@hyperpanel.fr

La suite Hyperpanel OS est un outil logiciel spécialement destiné à l'informatique embarquée. C'est-à-dire celle enfouie dans tous les appareils de notre vie de tous les jours et dont la fonction première n'est pas « informatique » : téléviseurs, compteurs, appareils électroménagers, systèmes embarqués et IoT.

*Les textes issus de cet article sont extraits des travaux de recherche et de développement de **Jean-Yves Astier** – CTO d'HyperPanel Lab, que l'on remercie pour son aimable relecture.*

Cette suite comprend une gamme complète d'outils de développement dédiés à la production, aux tests et à l'intégration des logiciels embarqués. Cet environnement très intégré propose l'état de l'art en termes d'outils, et sur le plan fonctionnel des innovations déterminantes, à savoir une réduction drastique tant du cycle de production que du délai de mise sur le marché des produits finaux. L'une de ses principales innovations réside dans son architecture à deux conteneurs 1.

Cette architecture et surtout son implémentation lui confèrent une capacité temps réelle inégalée. Mais si jusqu'à ce jour seule une faible population d'érudits avait accès au développement d'applications temps réels, la suite Hyperpanel OS démocratise sa mise en œuvre et ouvre le champ des possibles à de nombreux nouveaux utilisateurs.

Hyperpanel OS est un système d'exploitation réparti constitué de deux conteneurs, un conteneur d'applications et un conteneur d'entrées/sorties. Le conteneur d'application dispose simultanément d'un RTOS complet et d'un moteur d'automates à nombre d'états fini autorisant la programmation évènementielle structurée en langage C. Le conteneur d'entrées/sorties dispose d'un Ultra Real Time OS qui est le couplage étroit d'un moteur d'automates à nombre d'états fini ultra rapide et d'un gestionnaire hiérarchique d'interruptions matérielles [encadré théorique]. La totalité des drivers, des protocoles, services et systèmes de fichiers se situent dans le conteneur d'entrées/sorties. Ils sont exécutés par l'Ultra Real Time OS ce qui leur confèrent des performances sans précédent, tout en permettant de diminuer dynamiquement la fréquence du ou des CPUs, économisant ainsi batteries ou piles. La programmation de l'application se fait, comme son nom l'indique, dans le conteneur applicatif. Elle est réalisée en langage C.

Grâce à la séparation en deux conteneurs, Hyperpanel OS est un OS dit « réparti » car les deux conteneurs dialoguant par messages, ils peuvent donc être situés dans des cartes électroniques distinctes, raccordées à un même réseau local. Nous abordons ici les outils simples et conviviaux qui, avec très peu de connaissances, nous permettent de réaliser notre première application temps réel.

Installation

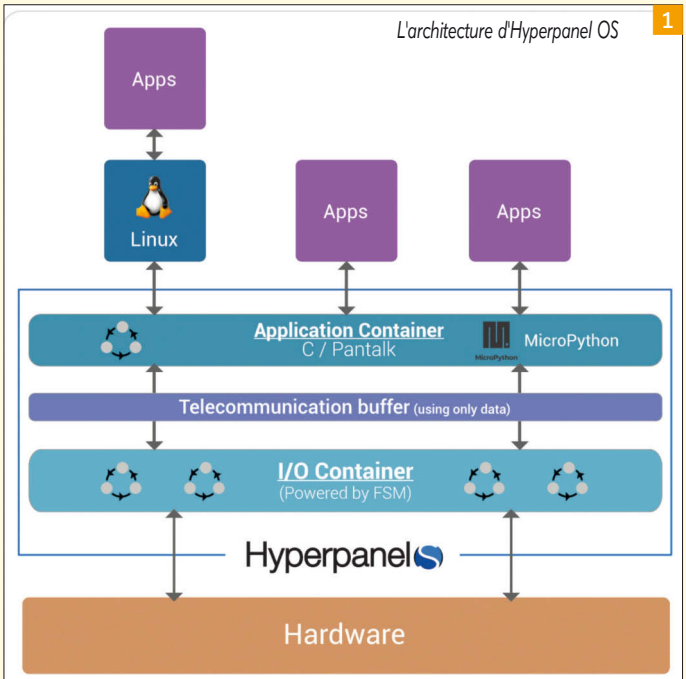
C'est souvent dès cette étape que l'on perd patience et que l'on abandonne si l'on n'est pas un geek de Linux ou bien de Windows. Avec la suite Hyperpanel OS, les longues recherches sur Google afin de comprendre les mystérieuses commandes d'installation de tel ou tel package de telle ou telle version n'existent plus. La première chose à faire est de se rendre sur le site www.hyperpanel.com 2. Vous y trouverez 4 choses : une présentation générale de l'Operating System, un Tutorials Center donnant accès à des exemples de programmations, un Toolkit en ligne permettant de développer en ligne vos applications, et enfin toutes les informations nécessaires pour se procurer un hardware de développement permettant de découvrir Hyperpanel OS.

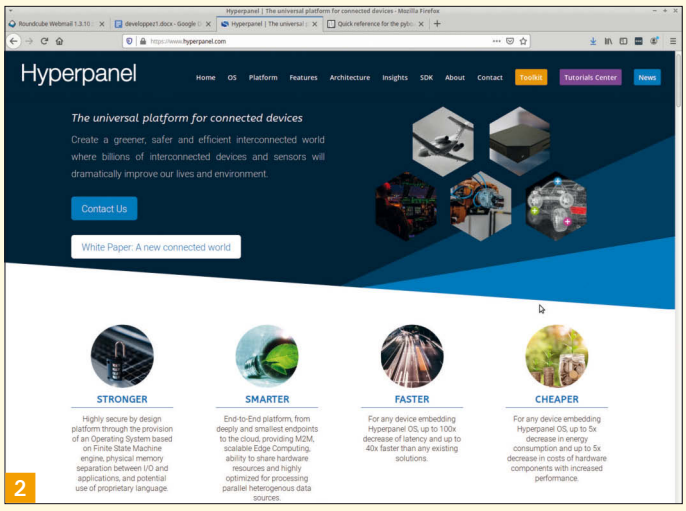
Un hardware de développement "ready to use"

Afin de pouvoir commencer à utiliser ce système exploitation, la société Hyperpanel Lab. propose un Kit hardware complet (Software Development Kit), disponible sur demande à partir de leur site. Il suffit pour cela de se

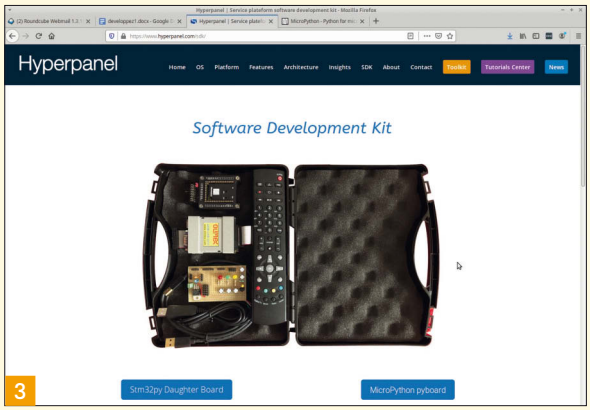
1

L'architecture d'Hyperpanel OS

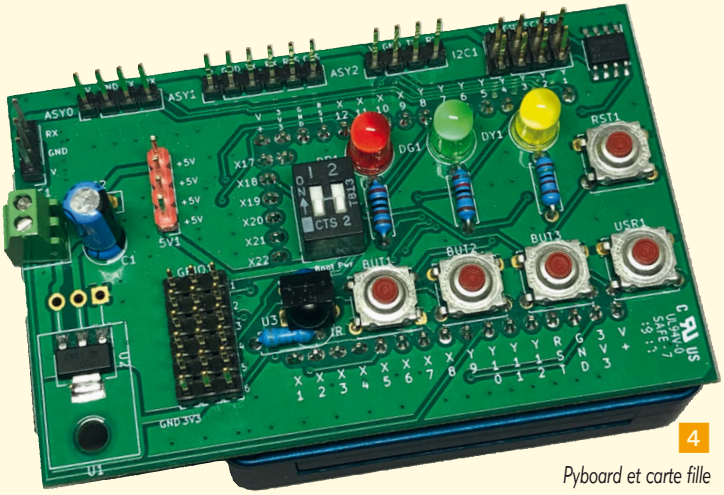


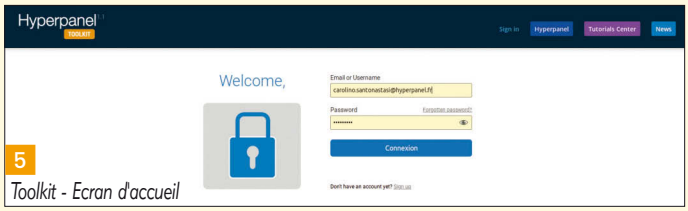


2



3


4
Pyboard et carte fille



5
Toolkit - Ecran d'accueil

DES ÉCOLES NOUS ONT DÉJÀ ADOPTÉES

Dans le cadre d'une relation de partenariat avec les écoles d'ingénieurs, trois groupes ont démarré un cycle d'apprentissage :

- A l'Epitech le centre de R&D – sous la direction de Clément Dubois, et de Theo Zapata, les étudiants : Killian Clette, et Naoufel Berrada.
- A l'ISEP, sous la responsabilité du Professeur Frédéric Amiel
- A Supcom, sous la responsabilité du Professeur Ridha Bouallegue et du chargé de projet Mohamed Ouldelhassen

rendre sur la page "SDK" 3 et de remplir le formulaire. Ce kit comprend principalement une carte Pyboard et sa carte fille conçue pour l'apprentissage 4. La carte Pyboard en basée sur un micro-contrôleur STM32F405 qui intègre un processeur ARM Cortex M4, de la mémoire FLASH et 192 Ko de mémoire RAM. Cette très faible quantité de RAM permet de réduire considérablement la consommation électrique. C'est dans cette carte que l'on va faire tourner le système d'exploitation Hyperpanel OS, ainsi que diverses applications d'initiation. La carte est pré-équipé d'un connecteur JTAG et le Kit est fourni avec une sonde Olimex ainsi que toute la connectique permettant le flashage de l'OS à partir d'un ordinateur. Pour rendre l'apprentissage concret, ludique et pédagogique, le SDK contient également une carte fille qui se branche directement sur la PyBoard, en lui ajoutant des boutons, des LEDs, un capteur infrarouge, une EEPROM, ainsi qu'une connectique variée (GPIOs, 3 ports séries, un bus I2C). Il est donc possible de concevoir très simplement et très rapidement des petites applications pouvant envoyer des messages sur un port série, recevoir des messages d'une télécommande infrarouge, utiliser un écran I2C pour afficher des informations, etc.

Un Toolkit de développement en ligne

L'inscription sur le site Hyperpanel OS vous donne également accès aux outils de développement en ligne. A l'aide de votre identifiant et de votre mot de passe fournis par HyperPanel Lab 5, vous allez pouvoir vous rendre sur le Toolkit de développement 6 et commencer à développer votre première application embarquée.

Ce Toolkit est principalement un éditeur de programmes C (le langage de programmation utilisé), une documentation en ligne, un forum d'assistance en ligne ("Insider"), des outils de compilation et de debug. Le principe est de tout faire en ligne, y compris l'édition de lien et la fabrication du binaire téléchargeable. En quelques clics, nous allons voir sur un exemple concret, comment éditer du code, compiler, linker et télécharger l'exécutable fabriqué en ligne.

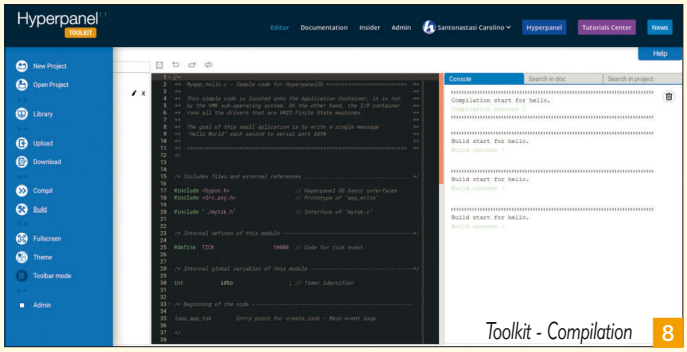
```
/*  Procedure loop_app_tsk ----------------------------------------------

    Purpose : This is our task main loop.
*/

int loop_app_tsk (void *param)
  {
    int             ev              ; // Our event
    int             cpt = 0         ; // Counter of messages
    char            mess[80]        ; // Message to be sent

/*********************************************************************
 * Step 1 : We start a timer that will send us an event every 1 second so    *
 * ------   that we can update the time.                                     *
 *********************************************************************/

    set_tto(CLOCK       ,               // Timer mode: clock
            1000        ,               // Duration in milliseconds
            TICK        ,               // Event code
            0           ,               // Event reserve field
            &idto         )           ; // Timer identifier


/*********************************************************************
 * Step 2 : Here is our main event loop. For each loop, we do as follows :   *
 * -------  - First we wait for an event.                                    *
 *          - Then if the event code is TICK we print "Hello World"          *
 *********************************************************************/

    wait_ev                         : // Beginning of loop label

    ev = wait_allevent()            ; // Unschedule until an event is
                                      // received

    if ( ev == TICK )                 // If the event is the tick event
      {                               //
        hsprintf(mess           ,     // We build in "mess" the message
           "%6d  Hello World\r\n",    // to be sent
            cpt ++                ) ; //

        asy_write(0             ,     // We send on ASY0
            (unsigned char*)mess ,    // the "mess" message
            strlen(mess)          ) ; // Count of bytes to be sent
      }

    goto wait_ev                    ; // Wait for the next event

    return 0                        ;
  }
```

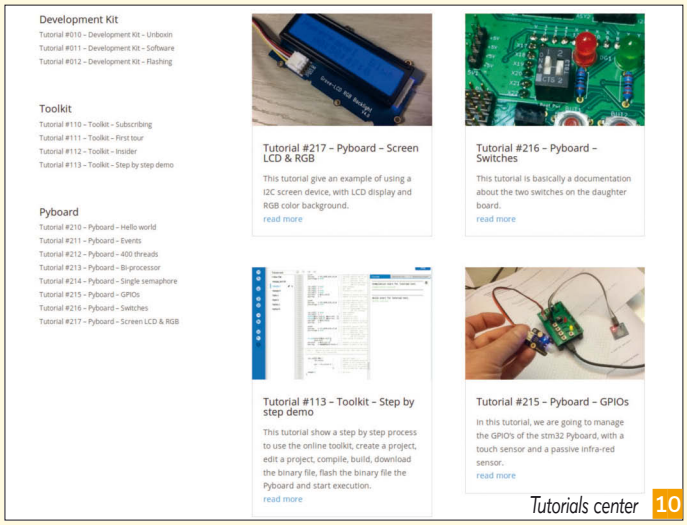
Exemple de source C 6

Développement en 3 étapes

Première étape : On écrit du code C. La première chose à faire est de créer un projet ("New project"), puis de choisir (bouton "Library") un exemple déjà existant ("hello"). Ce code peut être modifié en utilisant le Toolkit 8. La fenêtre de gauche de l'écran principal contient l'éditeur pour créer et modifier ses programmes C. La fenêtre de droite contient une fonction de recherche dans le projet courant, ainsi qu'une fonction de recherche dans la documentation. Le programme principal de notre application ressemble à ceci 7. Sa fonction est d'écrire toutes les secondes le message "Hello World" sur le port asynchrone "ASY0" de la carte fille. On voit ici que d'autres fichiers participent à ce code, on trouve en particulier un fichier de définitions d'interfaces appelé « mytsk.h » qui permet d'utiliser des sous-programmes C déjà existants, disponibles sur le site. Le Toolkit contient également une documentation en ligne complète de l'Operating System, les APIs des drivers et des procédures de librairies.

Deuxième étape : On compile le code 8. Le compilateur est hébergé





par le Toolkit en ligne. **Il n'est donc pas nécessaire d'installer un compilateur C sur sa machine**. La fenêtre de droite du Toolkit contient une console qui fournit toutes les informations sur la compilation (erreur de compilations, etc...).

Troisième étape : On réalise l'édition de liens afin d'obtenir un binaire pour la Pyboard (bouton "Build"). Comme pour la compilation, l'édition de liens s'effectue directement dans votre navigateur. Le binaire est créée en ligne et un pop-up permet de le télécharger. 9.

Flashage dans la cible

Le flashage est l'opération d'écriture du binaire (Operating System et application) dans la flash de la Pyboard. Il existe deux façons de flasher la cible : en passant par un câble USB relié à votre ordinateur, ou bien en utilisant l'interface JTAG Olimex livrée avec le SDK Hyperpanel. L'intérêt d'utiliser la connexion JTAG est essentiellement pour la mise au point (le debug) de vos applications. Le site Hyperpanel fournit un ensemble de tutoriaux ("Tutorials Center") et en particulier tous les tutoriaux concernant le flashage de la Pyboard par les différentes méthodes disponibles.

Démarrage de la Pyboard

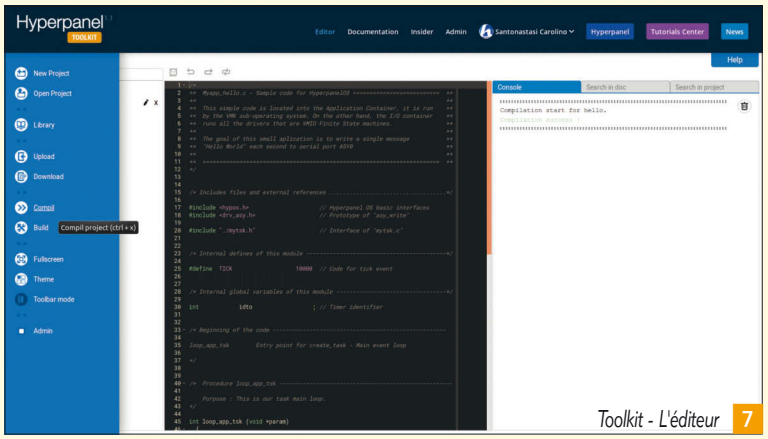
Une fois le logiciel flashé, le démarrage se fait tout seul à la mise sous tension de la cible. En connectant le port asynchrone "ASY0" de la carte fille sur une entrée USB de votre ordinateur, il suffit d'ouvrir un terminal (par exemple "minicom") et de voir les traces générées par notre application "hello". Tout ceci est également expliqué au travers des tutoriaux disponibles en ligne dans le Tutorials Center.

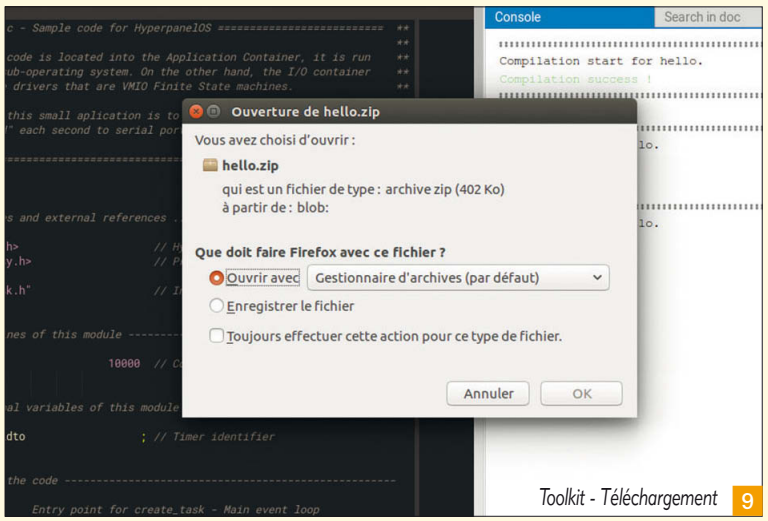
Pour aller plus loin

Le site Hyperpanel OS propose donc également un Tutorial Center libre d'accès, avec les explications complètes d'apprentissage de cet OS, mais également des exemples d'applications utilisant diverses fonctions de l'OS, des exemples d'applications multi processeurs, des gestions de sémaphores, des exemples d'entrées/sorties, et des cas concrets d'utilisation de périphériques I2C 10.

Les liens utiles

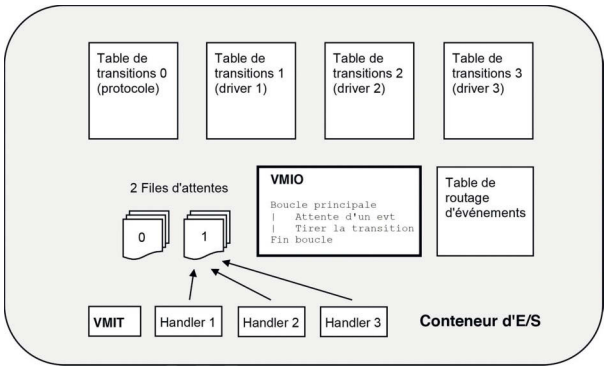
www.hyperpanel.com
https://tutorial.hyperpanel.com/tutorials-all/
www.micropython.org
www.olimex.com



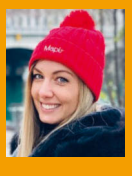


PRINCIPE D'UTILISATION D'UN MOTEUR D'AUTOMATES POUR LE CONTENEUR D'E/S

Un conteneur d'entrees/sorties reçoit des évènements de requête de la part de conteneurs d'applications d'une part et des évènements de notification de fin d'entrees/sorties physiques d'autre part, qui sont déposés par les routines de traitement (Handler) d'interruptions. Les événements de notification étant des conséquences de ceux de requête, ils doivent être traités en priorité afin que les mécanismes de contrôle de flots puissent être opérationnels. On utilise donc deux files d'attentes de priorités différentes, la plus prioritaire pour les évènements de fins d'entrees/sorties, la seconde pour les évènements de requêtes.



Fonctionnement de l'utilisation d'un moteur d'automates pour conteneur d'E/S



Marion FELIX
Développeuse web et entrepreneure française expatriée à Montréal. Aujourd'hui co-fondatrice et facilitatrice d'expatriation chez Maplr.co (www.maplr.co) Marion accompagne gratuitement les talents en TI dans leurs projets d'expatriation au Canada.

Pourquoi autant de développeuses/eurs partent vivre au Canada ?

Le Canada est devenu une destination de choix pour les développeuses/eurs du monde entier. Outre le capital sympathie dont bénéficient les Canadiens, le pays jouit d'une qualité de vie incomparable faisant rayonner son attractivité outre-Atlantique. Vous connaissez sûrement un(e) collègue ou connaissance, ou même une personne de votre famille qui a tout quitté pour démarrer un nouveau chapitre de sa vie et carrière là-bas. Voici quelques raisons mettant en lumière les attraits du Canada.

Une expérience de vie

En France en 2019, 76% des salarié(e)s du numérique désirent s'expatrier, selon une étude de BCG et Cadremploi. Si les universités françaises sont classées parmi les meilleures au Monde, l'Hexagone peine à conserver ses étudiants une fois diplômés. Les expert(e)s du numérique s'orientent davantage vers les États-Unis, l'Allemagne et le Canada, trio de tête devant l'Australie et le Royaume-Uni. La France, elle, n'arrive que 7e dans le classement des destinations les plus attractives.

Pour les professionnel(le)s du numérique, l'immersion dans une nouvelle culture est un véritable tremplin dans la vie professionnelle et personnelle.

Une qualité de vie incomparable

Chaque année, le U.S. News and World Report's Best Countries Report établit un classement des pays selon un grand nombre de catégories ; la qualité de vie y joue un rôle majeur, puisqu'elle est l'un des facteurs principaux de motivation des futurs expatrié(e)s. C'est aussi la grande force du Canada puisque le pays arrive numéro 1 en qualité de vie dans le monde entier.

Cela s'explique par un environnement naturel riche et accessible, plusieurs villes de choix où il fait bon vivre (Vancouver, Toronto, Ottawa, Montréal...), des valeurs fortes et humaines (les libertés individuelles sont très importantes au Canada, le bénévolat, l'entraide, l'attitude chaleureuse sont des valeurs très répandues). L'inclusivité est, par ailleurs, l'un des points forts du Canada ; ici, les lois comme la société condamnent toute forme de discrimination basée sur le genre, la race, la religion, l'apparence, etc.

La santé est également un point non négligeable de la qualité de vie : au Canada, 13 régimes d'assurance-maladie provinciaux et territoriaux assurent aux résidents canadiens un accès satisfaisant aux services médicaux et hospitaliers sans avoir à débourser d'argent. Enfin, selon l'Indicateur du vivre mieux de l'OCDE, le Canada obtient une note de 9.1 sur 10 quant à la sécurité et 81% des Canadien(ne)s affirment se sentir en sécurité lorsqu'ils/elles marchent seul(e)s la nuit - contre 69% en moyenne. Le taux d'homicide est aussi bas que 1,4% pour 100 000 habitants - contre 3,6% dans les pays de l'OCDE.

Un marché de l'emploi très actif

Depuis quelque temps, le Canada connaît une pénurie de main d'oeuvre en Technologies de l'Information (TI) ; Maplr en avait d'ailleurs fait un article complet (https://maplr.co/penurie-ti-au-canada-le-recrutement-international-est-la-solution) au début de l'année 2019 :

"le problème est que le système scolaire canadien ne permet pas de former suffisamment d'étudiants qualifiés pour combler ce manque. Entre 2004 et 2019, on estime ainsi que le nombre de futurs informaticiens dans des écoles canadiennes n'a pas bougé. Au Québec par exemple, 3000 nouveaux informaticiens sont diplômés chaque année… Tandis que dans le même laps de temps, plus de 10000 postes sont créés. Il est donc important d'encourager les Canadien(n)es à se diriger vers cette filière ; cependant, même en se fiant aux prévisions les plus optimistes, cela ne suffira pas." - (source www.maplr.co)

Les entreprises canadiennes en TI s'orientent aujourd'hui vers le recrutement international pour pouvoir fonctionner - 88 % des entreprises tech québécoises envisagent l'embauche via l'immigration comme une solution aux enjeux de recrutement actuels. Pour les Français et autres francophones, l'avantage est bel et bien là, puisque d'ici 2021, le Québec à lui seul aura besoin de 45000 travailleurs supplémentaires dans le domaine de l'informatique. En Ontario, on parle de 88000 créations de postes d'ici la même année.

Les conditions salariales sont elles aussi très attractives pour les travailleurs étrangers : ce sont plus de 50% des employeurs canadiens qui considèrent qu'un salaire compétitif est la clef pour attirer et garder les talents dans le secteur des TI et de l'informatique. C'est ainsi qu'en moyenne, à fonction égale, la rémunération est 30% plus importante au Canada qu'en France : les profils juniors gagnent au minimum 60000 CAD$ par an, les profils intermédiaires 75000 CAD$ par an et les profils séniors 90000 CAD$ au minimum par an.

En 2019, Hays a mené une étude (https://www.hays.ca/fr/guide-salarial-2019-canada/index.htm) quant aux mesures que les employeurs canadiens en TI vont prendre pour conserver leurs salarié(e)s : possibilité de travailler de la maison (56 %), élargissement des prestations de santé (54 %), flexi-

bilité des horaires de travail (49 %), plus de 10 jours de vacances pour les nouvelles embauches (45 %), cotisations à des REER (39 %), bonus liés à la performance individuelle (32 %)...

Dans le domaine des TI, le marché de l'emploi est à la fois actif et attractif au Canada : le chômage est très bas et de nombreux postes sont à pourvoir continuellement.

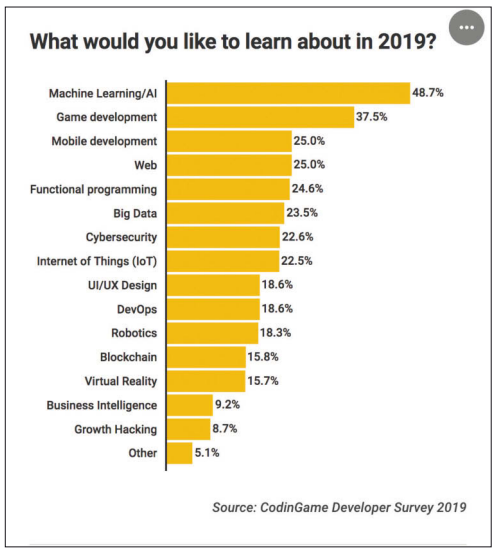
Un environnement de travail axé sur le bien-être des employé(e)s

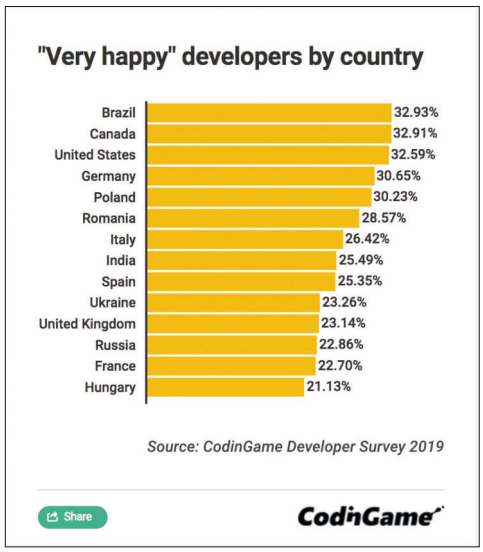
Au Canada, nombreuses sont les entreprises proposant le télétravail et la gestion flexible de l'emploi du temps pour leurs employé(e)s. Il est également coutume de terminer à 17h en moyenne, le pays favorisant grandement l'équilibre entre travail et vie personnelle, laissant alors aux personnes davantage de temps pour profiter de leur famille, de faire des activités, de voir leurs amis… La communication interne se construit à l'inverse de ce que nous pouvons connaître en France : au Canada, le management se veut horizontal, les échanges se font plus rapidement entre employés et direction et les possibilités d'évolution sont réelles. L'environnement professionnel est très peu stressant et la motivation personnelle, l'implication et l'engouement des employé(e)s envers leur entreprise sont aussi valorisés, voire davantage que leur performance. La culture du présentéisme n'existe pas et la confiance se tisse naturellement. En France, tout comme au Royaume-Uni ou en Russie, seulement 22% des développeurs/euses affirment être "très heureux", alors que ce taux de satisfaction avoisine les 33% au Canada.

Les talents en TI français ont bonne réputation

Les diplômé(e)s d'écoles et universités françaises sont fortement valorisé.e.s au Canada : cela s'explique par la bonne réputation des études dans l'Hexagone, de bonnes qualifications, des connaissances techniques pointues et un parcours académique en général exemplaire.

Au Canada, on ne s'intéresse pas à l'étiquette d'une école : chaque personne est considérée pour ce qu'elle vaut en termes d'implication dans les projets et au sein des équipes et à travers ses "soft skills", ses compétences qui rendent chacun de nous plus compétitif/ve et unique aux yeux des employeurs.





Des facilités d'immigration

Il est aujourd'hui facile d'immigrer au Canada, de nombreux choix s'offrent à celles et ceux souhaitant s'y installer pour travailler et la simplicité des démarches rassure même les plus frileux.ses à l'administration : Permis Vacances Travail (permettant de travailler dans n'importe quel domaine et valide jusqu'à deux ans), permis Jeune Professionnel pour les moins de 35 ans (permis rattaché à l'entreprise qui vous sponsorise)... Les concepteurs/trices et développeurs.euses informatiques font partie des travaillleurs/euses qualifié(e)s les plus recherché(e)s au Canada, notamment à travers les programmes d'immigration prioritaires et les permis de travail.

Un écosystème tech en plein essor

Montréal, plus grande ville du Québec s'offre, en plus de la place de leader mondial en intelligence artificielle et jeux vidéo, celle de capitale des technologies de l'information du Canada. Selon un récent rapport, le Machine Learning est de loin la compétence que les développeurs.euses planifient le plus d'acquérir en 2019 suivi de près par le développement de jeux vidéos. En phase de devenir "la silicon valley" du Canada, la ville jouit de nombreux événements techs, meetup et conférences. Un écosystème riche et animé qui motive d'autant plus les développeurs.euses français.es à démarrer une nouvelle vie outre-atlantique.

Conclusion

Attractif, inclusif et en besoin constant de nouveaux talents, le Canada s'impose comme une destination de choix pour les développeurs/euses français(e)s. En plus de proposer un cadre professionnel motivant, la qualité de vie qui règne dans le pays est une véritable valeur ajoutée incitant les travailleurs/euses à s'y installer. •

Aurélie Vache
Cloud Dev(Ops) chez Continental
Duchess France & DevFest Toulouse Leader
Toulouse Data Science core-team &
Marraine Elles bougent
@aurelievache

La reconversion, c'est maintenant !

Partie 1

Le temps où il fallait obligatoirement faire un Bac S, avec ou sans prepa, puis une école d'ingénieur pour devenir développeur, est révolu. Personnellement, j'ai su dès la 6ème que je voulais "travailler dans l'informatique", devenir "programmeuse". Je n'ai pas le précieux Bac +5, je ne suis pas "ingé", et cela ne m'a pas empêcher de faire un travail que j'aime, de devenir Développeuse / DevOps / Lead Dev.

Les témoignages que vous allez lire dans ce dossier proviennent de personnes en reconversion ou qui se sont reconverties. Elles ont chacune une histoire différente, des expériences différentes, et pourtant une chose les réunit : la passion, l'envie, la motivation, la curiosité. Trêve de bla bla, je vous laisse découvrir leurs témoignages !

Marina Rouillé :
reconvertie… 2 fois !

Je vais vous raconter ma vie, vous êtes prévenus. Je vais parler d'une reconversion et d'une reconversion.

La formation d'analyste-programmeur.

En 2000, après avoir arrêté la fac et travaillé quelques années en intérim dans des usines d'agro-alimentaire, je me décide à cesser d'attendre une vocation qui ne vient pas. Le secteur informatique embauche et il y a des formations courtes qui me conviennent notamment financièrement.

Je choisis l'Ecole Nantaise d'Informatique qui propose un contenu clair, collabore avec des entreprises pour adapter le contenu à leurs besoins réels et de laquelle j'ai de bons échos.

La formation commence en mars 2001 et c'est parti pour 6 mois détaillés en 3 modules :

- Visual basic
- Oracle forms et reports
- C++ et Java

On travaille en binôme, on reçoit un enseignement théorique aussitôt appliqué en exercices. Chaque module est ponctué d'un projet final qu'on réalise avec notre binôme. On s'autogère, on a les clés de notre salle, le code de l'alarme de l'école et on peut venir travailler quand on veut. L'ambiance est très collaborative et appliquée. Tout le monde est motivé et on travaille même plus en groupe qu'en binôme.

J'ai quand même des doutes. A chaque début de module j'ai l'impression que je n'y arriverai jamais. C'est Pierre, mon premier binôme qui m'a appris, effaré, à faire un copier/coller. Vous voyez le niveau.

Je m'accroche, je bosse pendant les cours et après. Je bosse le week-end. Tout le monde s'entraide et ça m'aide et me motive énormément. Ca prend forme, je sais coder !

Je ne me rappelle pas notre nombre exact mais on devait être environ une vingtaine dont 3 femmes. Nous sommes plusieurs à nous voir proposer des CDI avant même la fin de la formation.

Le travail de développeuse.

Je commence donc mi-septembre 2001 à travailler pour une ESN (mais on disait SSII) de la région parisienne.

Et pfff, les doutes de début de formation reprennent. J'y arriverai jamais. C'était une chose de réaliser des exercices sur quelque chose que je venais d'apprendre mais dans la vraie vie c'est bien plus compliqué. Déjà comprendre ce qu'on me demande. Bon et puis une fois qu'on a compris ce qu'on doit faire. Comment le faire techniquement ? Quand on est plus du tout guidé. Sacré challenge quand on a seulement une formation de 6 mois.

Mais petit à petit je progresse. En fait quand on travaille au sortir d'une formation courte (mais finalement aussi au sortir de l'université ou d'une école) c'est comme quand on a le permis. On a les bases mais tout à apprendre !

La SSII est… une SSII, mais l'ambiance entre collègues et spécialement dans mon open-space est très bonne. Mais je doute. Je me sens bien en-dessous du niveau de mes collègues. Je progresse mais je me dis que je n'arriverai jamais à les rattraper parce qu'eux aussi progressent. Et puis au-delà de la pure technique et même si mes collègues sont sympas je ne me sens pas à ma place. Il y a la programmation, ok, ça j'ai appris mais il y a tout ce qui est attenant. Quand pendant les pauses mes collègues parlent de RAM, de défragmentation, de CPU… je pige que dalle. Et quand je demande à ce qu'on m'explique, comment dire…. J'ai droit à une bonne dose de mépris et de condescendance. Et puis parler boulot pendant qu'on bosse, aux pauses cafés, pendant le déjeuner… Je me suis trompée ! Je ne suis pas du tout faite pour ce métier !!!

Ça n'est pas seulement un métier de passionnés.

Arrive un rayon de soleil : Rkeya ! Elle est embauchée quelques mois après moi. À l'époque je ne note pas ce genre de choses mais c'est la seconde femme qui arrive dans l'open-space. Je n'étais entourée que d'hommes blancs jusque là. Et qui n'avaient que l'informatique comme sujet de conversation. On devient amies, mon quotidien s'allège et je finis par me confier à elle. Je me suis trompée, je ne suis pas faite pour ce métier, même si je progresse, je vois

bien que je n'appartiens pas au milieu. Elle éclate de rire ! "Mais Marina, ça n'est pas comme ça partout, ici il n'y a que des geeks mais dans ma boite précédente, ça n'était pas du tout la même ambiance. On bossait dur mais on s'amusait énormément aussi.". Ah bon ??? Ouf !

Je persiste, je grandis, je prends confiance en moi, je quitte cette première boite et pendant 15 ans je ne douterai plus de ma place dans ce métier. Non seulement, nous ne sommes pas tous des rockstars ou des 10x, mais nous ne sommes pas tous des passionnés. C'est super d'être passionné par son travail mais on peut l'aimer énormément et ne pas avoir envie d'y passer son temps libre. Chacun son truc. Et chacun a sa place dans une équipe.

Ca peut faire peur quand on débarque dans ce métier, on peut avoir l'impression qu'on arrivera jamais à la cheville de gens qui ont une formation plus académique ou de ceux qui boivent, mangent, respirent informatique. Mais si, j'ai croisé des profils très différents et il n'y a pas de règle qui en ressorte. Vraiment, ça n'est pas pour prêcher pour ma chapelle mais j'ai vu des gens qui avaient une courte formation devenir plus brillants que des gens qui avaient fait des écoles d'ingénieur. La caractéristique que je crois reconnaître le plus souvent chez les développeurs, c'est la curiosité. Ce dont je veux témoigner c'est qu'il n'y a pas de condition préalable à exercer ce métier correctement, c'est-à-dire fournir le travail pour lequel on est payé.

La formation java.

Ah !!! J'y arrive, la deuxième reconversion.

Oui parce que moi je travaillais sur des produits Oracle. Les bases de données, bien sûr mais aussi des outils pour construire des IHM (Oracle forms) et faire des rapports (Oracle Reports).

Et bon je les appréciais ces outils mais j'avais envie d'apprendre de nouvelles choses. J'ai travaillé principalement pour des ESN et au bout de 6 mois sur un outil dans une ESN, on est déclaré expert. Parce que quand on est expert, on est facturé plus cher. Et quand on facturé cher sur une technologie, très difficile d'en changer.

J'ai donc continué à travailler sur les mêmes outils. Jusqu'à un déménagement à Toulouse. Et une recherche de travail infructueuse sur les technologies Oracle.

Dans les annonces, la technologie qui revient le plus c'est java. Je me sens prête à apprendre sur le tas mais les entreprises ne sont pas prêtes à me laisser essayer.

Donc je m'inscris dans une formation chez BGE, un organisme qui à la base fait de l'accompagnement d'entreprise et qui depuis quelques années propose aussi de former en informatique. C'est une formation de 3 mois, nous étions 15 dont 5 femmes. Trois modules cette fois aussi :

- HTML/CSS/javascript/jQuery/PHP
- Java SE
- Java EE

3 mois c'est beaucoup trop court ! Et les niveaux de compétences et de motivation de chacun était très disparates. Les formateurs ont essayé au mieux de s'adapter à cette différence de niveau mais ont échoué à former certains d'entre nous.

Nous n'avons accès à la salle que pendant les horaires de cours et chacun a son ordinateur, personnel ou fourni par l'organisme. J'ai vu nettement la différence sur l'ambiance de travail.

Pas très satisfaite de cette formation en soi mais l'objectif principal est rempli puisqu'elle m'a permis de trouver du travail à Toulouse.

La deuxième vie de développeuse.

J'ai appris finalement plus sur java pendant le stage de fin de formation et mon travail ensuite au CNRS. Au moins ça, ça ne change pas, on apprend réellement que quand on est confronté à de vraies situations de travail.

J'ai découvert avec bonheur la communauté "java",je ne sais pas comment l'appeler, ça va au-delà de java, la communauté de développeurs web si vous voulez.

Non seulement, c'est génial de découvrir de nouveaux outils mais là en plus il y a tellement de partage, je trouve ça vraiment extraordinaire. On a la chance d'avoir des pairs qui défrichent pour nous des outils, des méthodes et qui nous présentent ce qu'ils ont appris pour nous aider à les utiliser aussi.

Parce que quand on débarque dans l'écosystème java, ça n'est pas évident de s'y retrouver. Et j'ai eu les mêmes doutes qu'à l'occasion de ma première formation. Il faut dire que c'est frustrant quand on maîtrise d'autres outils de faire en quelques jours ce qu'on aurait fait en quelques heures avant. La tentation peut même être grande de revenir en arrière mais il faut tenir bon, on y arrive.

Les écueils sont toujours les mêmes qu'on developpe en POO ou pas mais le fait qu'il y ait une communauté aussi active ça rend le travail plus agréable.

Et puis changer de technologie, ça a reboosté ma motivation qui était un peu en berne à la longue. Et ça me permet de continuer à coder à 44 ans.

Mes conseils pour une reconversion réussie.

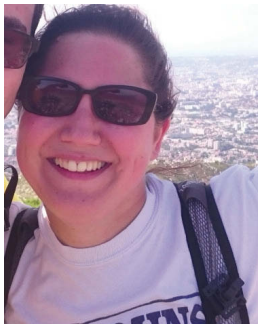
Il faut être motivé. J'ai presque envie de m'arrêter là. Une formation en informatique ça n'est absolument pas insurmontable même quand on est pas pro des maths, qu'on ne s'y connaît pas du tout en "matériel", même quand on ne sait pas faire un copier/coller !

Bon, au-delà de ça quand même, il faut bien se renseigner sur l'école qu'on choisit. Depuis quelques années, ça pousse comme des champignons ! Je sais que ça peut être compliqué au niveau du financement mais dans l'idéal il faudrait que ça soit au moins 6 mois. Si on vous dit que vous travaillerez en groupe, en binôme c'est un bon point, si vous avez accès aux locaux 24H/24 c'est un bon point aussi. Et surtout il faut essayer d'avoir des retours de gens qui ont fait la formation et voir si ce à quoi ça les a menés vous plairait aussi.

Vous ne pouvez pas vous contenter des heures et des exercices dispensés par les formateurs, il faut poursuivre le travail tout seul et au moins le temps de la formation, il ne faut pas s'attendre à faire des horaires de bureau. Au début de la vie professionnelle non plus d'ailleurs. Plus le temps passe, plus vous serez à l'aise et vous pourrez trouver un rythme qui laissera aussi la place à votre vie privée.

Servez-vous de la communauté, assistez à des meet-ups, des conférences (guettez les places à gagner grâce aux sponsors parce que les entrées sont parfois très chères).

Et puis bien sûr il y a des réseaux comme les Duchess ! C'est à la base une association pour valoriser le travail des femmes dans l'informatique mais c'est plus largement un communauté bienveillante qui promeut la diversité sous toutes ses formes. La diversité des parcours aussi donc. N'hésitez pas à chercher du soutien. •



Marie Laure Verjat
Simplon Carbone
(En reconversion)

Marie-Laure VERJAT, 32 ans, actuellement en reconversion professionnelle, en formation de Développeuse Web mobile à Carbonne, formation de Green It, dispensée par SIMPLON. Côté parcours, j'ai fait un bac littéraire, puis une licence de Marketing/Gestion en Savoie. Davantage motivée par le contact humain que le commerce, j'ai choisi de travailler un an comme auxiliaire de vie pour pouvoir passer les concours et devenir infirmière. J'ai fais 3 ans d'études dans un Institut de Formation en Soins Infirmiers, des stages, des stages et plein, plein, plein de choses, mais je n'ai pas validé ce diplôme. Cela m'a permis de travailler 3 ans comme Aide-Soignante et ça m'a appris bien des choses, travailler avec des contraintes horaires (JOUR/NUIT), collaborer, faire face à l'imprévu, prendre avec humour l'adversité et Vivre, Vivre, à fond! Les conditions de travail du métier ne me convenaient plus, j'ai donc décidé de mettre mon énergie au service d'un autre domaine et de me reconvertir professionnellement.

Pourquoi Simplon ?

Créative et curieuse, j'étais en veille sur les métiers de l'informatique, je souhaitais mieux connaître ce domaine quand Simplon et le SAS Hackeuse sont arrivés dans ma vie. Cette préqualification de 6 semaines m'a initiée au code, confrontée aux projets web, à la collaboration en équipe, à des perspectives, et ça m'a plu.

Pourquoi le numérique, l'informatique, car je suis créative et motivée par le fait, à partir de rien, de créer une page web et des programmes qui fonctionnent.

Parce que les recherches ne me font pas peur, que j'ai toujours aimé apprendre et que le fonctionnement en équipe est pour moi est un facteur de motivation car j'aime développer mes compétences.

Pour moi, être développeuse web, c'est créer des sites, chercher pour faire marcher des programmes, être autonome et collaboratif, capable de choisir les bons outils/langages pour accompagner les entreprises dans leur stratégie digitale, gérer le référencement des sites et toujours avoir envie d'apprendre. Apprendre à développer mes compétences à rester en veille au service de la stratégie mercatique de chaque entreprise, pour l'optimiser à travers mon travail.

Dans cette formation, nous apprenons grâce à une pédagogie active, à travers les projets, à travailler en équipe et à améliorer nos compétences sur les langages informatiques pour être adaptable et opérationnels directement arrivés en entreprise. Pour cela nous travaillons en équipe sur des projets pour intégrer les langages, nous menons énormément d'exercices différents pour développer nos connaissances et notre adaptabilité. La journée type commence par un stand-up meeting, puis une veille technologique présentée par chacun en alternance. Puis exercices, projets, pair-programming, rencontres avec des professionnels et avec des entreprises pour mieux connaître leurs exigences, et partage de code pour que chacun apprenne au maximum pendant la formation et des revues de codes régulières pour améliorer notre pratique. Nous apprenons entre autres le HTML, CSS, les frameworks CSS, Javascript, Jquery, Node JS, Ajax, PHP ainsi que des framework PHP. Nous sommes quinze apprenants, 7 femmes et 8 hommes. Nos différences de parcours et la collaboration nous apprennent énormément, nous avons tous hâte d'arriver en stage en entreprise. •

Cathy Laurent
Simplon Labège
(En reconversion)

Je m'appelle Cathy Laurent, j'ai 34 ans et je viens de Gindou, un petit village dans le Lot (46). Je suis en reconversion professionnelle après avoir travaillé 18 ans dans la vente. J'ai commencé à l'âge de 16 ans sur les marchés le weekend, pour pouvoir gagner de l'argent, tout en faisant mon BEP métier de la comptabilité. Après l'avoir obtenu, je suis partie sur Toulouse pour faire un BAC Informatique de gestion. N'ayant pas été acceptée ou en liste d'attente dans les lycées qui proposaient ce BAC, je l'ai passé avec le CNED, tout en travaillant. Avec du recul, je m'aperçois que je ne correspondais pas au critère des lycées demandés (je n'étais pas trop forte en math, et j'étais une fille). Malheureusement je n'ai pas eu mon BAC et la vie a fait que je me suis mise à travailler en tant que vendeuse en boulangerie et sandwicherie 35 heures par semaine (minimum). J'ai du mettre mon petit rêve de côté, pour pouvoir vivre. En 2018 après

quelques problèmes de santé, j'ai arrêté la vente et je me suis mis à réfléchir à ce que je voulais faire. J'ai fait activ' Projet avec Pôle Emploi qui m'a ramené vers le numérique. Ensuite j'ai postulé pour entrer dans la formation #SAS Hackeuses proposé par Simplon Occitanie. C'est une formation de découverte du numérique pour les femmes. La formation a durée 6 semaines, intense, riche en connaissances et en rencontres. J'ai appris le fonctionnement de Simplon qui est de travailler en autonomie et de savoir demander de l'aide si on ne trouve pas d'abord auprès de nos camarades de promotion et ensuite au formateur. J'ai pu confirmer mon envie de revenir dans le numérique et tout particulièrement dans le Développement Web, que je trouve passionnant et attractif. Simplon m'a permis de faire une Formation Développeur(se) web/web mobile qui dure 11 mois (8 mois de formation et 3 mois de stage), et au final je pourrais passer une équivalence pour obtenir un BAC+2 à la fin de la formation. J'ai choisi Simplon car leur pédagogie et leur façon de travailler me plaisent. Nous sommes un groupe de 15 dans la formation,(8 filles et 7 garçons) qui nous nous complétons par nos connaissances et nos savoirs. Nous venons tous de milieux et de carrières différentes mais nous avançons ensemble et nous essayons de ne laisser personne à la traîne. Un vrai travail d'équipe !

J'espère maintenant que les entreprises vont s'apercevoir qu'il n'y a pas besoin de beaucoup de diplômes pour être un bon développeur et qu'il faut donner sa chance à des personnes comme moi qui sommes en reconversion et qui nous donnons à 300% pour y arriver. •

Audrey BRIFFOD
SERIAL - Positive Thinking Company

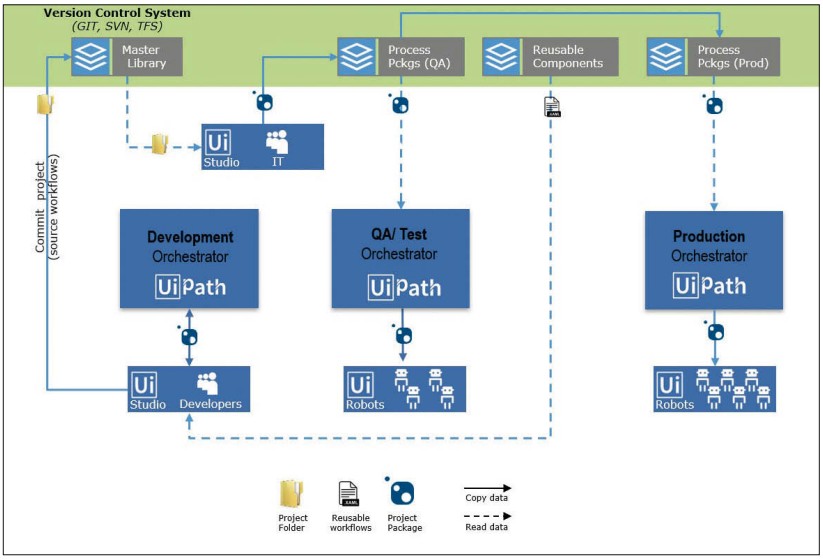
RPA : Pourquoi et Comment ?

Pour me présenter rapidement, je dirais que je suis une femme de 34 ans, maman solo d'un petit garçon de 3 ans et demi et consultante IT depuis plus de 10 ans. Cela fait 7 ans que je travaille pour SERIAL qui est une SSII basée à Genève (en Suisse) et entité du groupe Positive Thinking Company.

À l'origine, j'étais développeuse C# et SQL, mais après plusieurs années d'expérience, j'ai voulu donner un sens nouveau à ma carrière.

Ma principale motivation au travail, a toujours été de répondre au mieux aux besoins de mes clients et de leurs utilisateurs afin de les aider dans leur quotidien. Comprendre leur métier et leurs problématiques me semble indispensable. C'est donc naturellement que je me suis tournée vers la business analysis.

En recherchant des solutions, j'ai découvert le monde de l'automatisation et me suis prise de passion pour ce dernier. Je me suis d'abord formée sur le BPM (Business Process Management) puis sur la RPA (Robotic Process Automation).



Exemple d'une chaine d'outils RPA : UiPtah

Qu'est-ce que la Robotic Process Automation et quelle est son utilité ?

Je suis convaincue que cette technologie peut vraiment révolutionner la manière de travailler. En effet, si la RPA est en plein essor ce n'est pas juste par effet de mode. Grâce à elle, il est enfin possible d'augmenter sa productivité tout en améliorant la qualité de vie au travail de ses collaborateurs. De quoi satisfaire tout le monde...

J'ai bien conscience que les termes "processus", "automatisation" et "robotisation" peuvent effrayer les collaborateurs qui craignent de voir leur poste disparaître et de se faire remplacer par un robot. Mais en réalité ces robots sont là pour les aider dans leur travail quotidien et les décharger des tâches manuelles et répétitives qui se révèlent souvent fastidieuses et chronophages. Ainsi les collaborateurs auront plus de temps pour se consacrer aux tâches à plus forte valeur ajoutée et plus épanouissantes. Au lieu de disparaître, leur poste va évoluer…

Nous nous sommes tous retrouvés un jour, débordés ou agacés par une accumulation de tâches peu valorisantes mais néanmoins indispensables (administratif, saisie...)

Dans des pays comme la Suisse et la France où les salaires sont élevés, ces tâches représentent un coût important pour les employeurs. Sans compter le coût lié à une erreur ou un oubli éventuel. Ou encore le turn over d'équipes démotivées et sous pression.

C'est justement sur ces points que la RPA peut aider, en s'attaquant aux opérations manuelles, répétitives, récurrentes et faillibles.

Voici quelques-uns de ses nombreux avantages :

- Productivité : la RPA libère du temps et réduit les coûts,
- Disponibilité : les robots peuvent travailler 7j/7 et 24h/24,
- Qualité : la RPA élimine le risque d'erreurs et d'oublis,
- Traçabilité : toutes les actions effectuées par les robots sont loguées,
- Connectivité : les robots connectent les systèmes entre eux,
- Non intrusive : la RPA ne demande aucune modification des systèmes sous-jacents pour s'interfacer avec eux,
- Implémentation rapide,
- Evite le "Shadow IT" comme par exemple les fichiers Excel contenant des formules ou des macros sans fin. Ces "solutions" mises en place par le métier, représentent un réel risque car elles ne sont pas forcément à jour par rapport aux règles métier et difficilement maintenables.

Tout ceci permet non seulement, d'améliorer la productivité et la qualité mais aussi la satisfaction des clients, fournisseurs et collaborateurs.

Comment fonctionne la RPA ?

La technologie RPA imite un comportement humain en interagissant avec les différents systèmes du SI. Contrairement aux technologies traditionnelles, elle n'est pas limitée à l'utilisation d'API, de web services ou de scripts. Lorsque ces options ne sont pas disponibles ou trop longues à mettre en place, la RPA utilise les IHM ou la reconnaissance d'images et de caractères.

Les plateformes RPA sont généralement constituées de 3 éléments :

- les robots qui sont en réalité des agents logiciels qui exécutent les processus,
- l'outil permettant de dessiner les processus et d'implémenter les règles métier et exceptions,
- l'outil de management permettant d'administrer la ferme de robots, de gérer les processus et leurs versions, de centraliser les valeurs de variables, de consulter les logs, etc.

Au niveau des robots, on peut distinguer 2 types : les robots Attended et Unattended.

- Un robot attended est un assistant installé sur le poste d'un utilisateur. Il se lance suite à une action de la part de ce dernier.
- Un robot unattended dispose quant à lui de sa propre machine et fonctionne de manière autonome.

Le choix du type de robot dépend du type de processus à exécuter. D'un point de vue sécurité, les robots possèdent leurs propres identifiants et accès aux systèmes. Il est déjà étonnant de voir tout ce que l'on peut réaliser avec les robots mais ce n'est encore rien en comparaison du futur. Afin d'aller plus loin, la RPA intègre de plus en plus, de l'intelligence artificielle, du process mining, etc. Les possibilités ne font qu'augmenter avec le temps et tout devient réalisable…

Quels sont les acteurs du marché ?

Concernant les éditeurs RPA, actuellement il existe 3 leaders sur le marché (selon le Gartner Magic Quadrant 2019) : UiPath, Automation Anywhere et BluePrism. Chacun présente des points forts et des faiblesses. SERIAL est partenaire de UiPath et d'Automation Anywhere. Notre rôle de conseil implique que nous soyons capables d'orienter le client vers le produit le plus adapté à ses besoins et contraintes. Nos consultants RPA sont donc formés sur ces 3 produits.

Comment initier un projet RPA ?

Avant de lancer un projet RPA, il y a plusieurs questions à se poser. Tout d'abord, quels sont les objectifs visés ?

Certains veulent simplement améliorer la qualité de vie de leurs salariés, d'autres désirent rendre leurs processus plus sûrs, d'autres souhaitent éviter un pic d'activité et certains ont la seule volonté de réduire leurs coûts.

Cette étape va permettre d'identifier les sponsors du projet RPA : c'est le début de la création du Centre d'Excellence (COE). Le COE regroupe des personnes de départements et métiers différents qui prennent part au projet RPA. Des rôles leur sont attribués : développeur, BA, responsable infrastructure, etc.

La RPA est un pont entre le métier et l'IT : elle ne concerne pas seulement l'un ou l'autre mais touche tous les départements. C'est pour cette raison que le Centre d'Excellence est très important et ne doit pas être négligé.

Selon la volonté du client et des équipes, certains rôles peuvent être pourvus en interne, d'autres en externe. Une même personne, peut être en charge de plusieurs rôles.

Les personnes qui composent ce COE doivent être des personnes convaincues des avantages de la RPA et motivées à faire de ce projet, un succès.

Pour l'étape suivante, il faut pouvoir identifier les processus à automatiser puis les évaluer et les prioriser selon différents critères. Lors de cette identification, certains processus peuvent être découpés en sous-processus. Il peut être décidé de conserver une partie manuelle dans le processus.

Il est important de s'assurer qu'il s'agit de processus stables et matures et que les règles métier et exceptions soient identifiées en amont.

Cette étape ne peut être réalisée qu'en étroite collaboration avec le métier. L'immersion au sein des différents départements est souvent la solution la plus efficace.

Certains départements comme la Finance, la logistique ou l'IT sont de bons candidats à la RPA. C'est donc par eux qu'il est préférable de commencer.

Au départ, il faut viser des processus simples, avec un ROI visible et des acteurs motivés. Puis, la communication sur ces premiers quickwins permettra de convaincre les autres départements et de donner des idées pour alimenter le backlog.

Avant de lancer le projet, un POC d'une semaine ou deux est souvent nécessaire car il permet de :

- Conforter le client dans son choix de l'éditeur avant l'achat des licences,
- Prouver la faisabilité et la compatibilité de la technologie avec le SI du client,
- Donner des idées pour identifier des processus futurs.

En parallèle de tout cela, la communication et le change management sont indispensables à la réussite du projet. Les différents acteurs doivent être impliqués tout au long.

À terme, c'est l'institutionnalisation de la RPA qui est visée. On souhaite créer une "Digital Workforce" au sein de l'entreprise.

Quelle est ma valeur ajoutée dans tout cela ?

En tant que consultante RPA, ma mission est de guider et d'accompagner mes clients dans toutes ces étapes afin de les aider à atteindre leurs objectifs. En effet, mon rôle ne se résume pas seulement à faire du design de processus. Je suis impliquée dans toutes les phases citées plus tôt : du choix de l'éditeur au déploiement en production et à la maintenance, en passant par l'installation de la plateforme, l'identification et l'analyse des processus et la mise en place du Centre d'Excellence.

Les clients ont des niveaux de maturité très différents sur le sujet de la RPA : certains ont déjà sélectionné l'éditeur et identifié plus de 50 processus à automatiser. D'autres sont intéressés mais ne connaissent pas les produits et ne savent pas comment choisir et prioriser leurs processus.

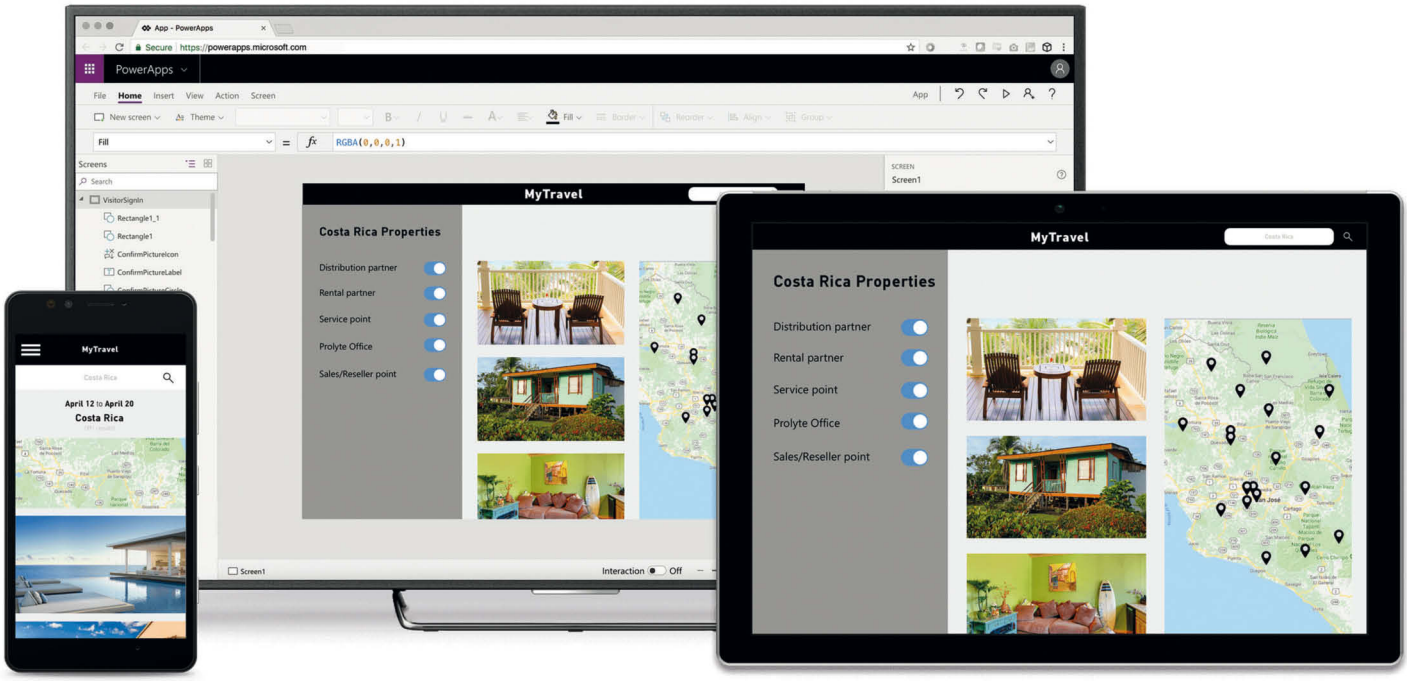
Nous avons donc appris à nous adapter à eux pour les accompagner et les conseiller dans les différentes étapes.

Depuis plus d'un an et demi, que je travaille sur des projets RPA, j'ai développé de bonnes compétences dans ce domaine. J'ai eu la chance de contribuer à la mise en place d'une méthodologie et d'une démarche d'accompagnement RPA.

J'ai aussi eu l'occasion d'animer plusieurs évènements RPA en partenariat avec UiPath, afin de faire connaître la technologie. Je participe également à des rendez-vous d'avant-vente en tant qu'experte.

Tout ceci est très intéressant pour moi car certaines de ces tâches étaient nouvelles et j'ai dû me former rapidement. Grâce à la RPA, je ne connais pas de routine dans mon travail, je suis proche du business et des utilisateurs. En bref, je suis passionnée par ce que je fais et travailler est un plaisir pour moi. •

Développement mobile : quelles tendances ?



Le mobile est un medium incontournable : +2h par jour à utiliser un mobile, les stores pèsent 101 milliards $ (2018), la France est 15e mondial dans le nombre d'installation (+1,8 milliard en 2018) et presque 1 milliard $ de revenus générés. Si vous visez un large public avec vos apps, vos efforts pour conquérir vos utilisateurs seront à la hauteur des ambitions. Mais avant de devenir la star des stores, il faut résoudre un (tout) petit détail : développer son app.

Pas toujours facile de s'y retrouver dans le développement mobile. On a vite tendance à opposer app hybride/Web et app native. Bref, le développement utilisant des couches Web et le développement natif (ou via des plateformes générant des apps natives). Mais comme souvent, la réalité est bien plus subtile et légèrement plus complexe. Aujourd'hui, il existe de multiples modèles de développement et d'outils/solutions/plateformes. Vous allez entendre parler de PWA, de Flutter, de multi-plate-forme, de modèle natif, d'app hybride/Web, de développement JavaScript ou via des frameworks complets, etc. Soyons honnêtes technologiquement parlant : on va souvent faire un 1er choix : modèle natif ou modèle hybride ou Web. C'est volontairement réducteur, voire, provocateur. Mais le fait de partir sur un développement natif, ou non, va induire beaucoup de choses. Si on part sur du natif, on va souvent faire du Kotlin (ou Java), du Swift ou prendre une solution multi-plate-forme capable de générer du natif.

« *IA, machine learning : les nouvelles tendances des apps mobiles* »

La veille techno : INCONTOURABLE

Ne sous-estimez pas l'importance de la veille techno sur le développement mobile. Aujourd'hui, la multiplication des SDK mobile, les évolutions des plateformes mobiles, les nouveaux environnements dév, les nouvelles tendances techniques ou d'UI obligent le dév mobile à être vigilant et se tenir au courant.

Le réflexe est toujours le même :

- Récupérer les versions gratuites
- Installer et tester avec un mini-POC
- Lire les docs, regarder les sessions techniques
- Communiquer avec la communauté

François Tonic

Dark Mode : tout sauf un gadget pour les apps mobiles

Quand Apple et Google ont dévoilé le mode sombre sur Android et iOS durant les conférences annuelles, les développeurs étaient excités. Pourquoi un mode d'affichage a suscité autant d'attentes et de joies ? Pour comprendre les enjeux, nous avons voulu en savoir plus avec Thomas Jaussoin de Lunabee.

niveau 100

« Comme chaque année, nous sommes allés à la Google I/O et à la WWDC. Quand ils ont annoncé le mode sombre, les développeurs se sont levés. Du coup, on s'est posé beaucoup de questions. Pourquoi une telle "folie" ? À partir de là, nous avons creusé la question et nous avons regardé ce que dit la science » commente Thomas.

Premier constat : en moyenne, nous regardons l'écran d'un smartphone/d'une tablette +3h par jour, en journée et le soir. *« On se mange de la lumière bleue et du rétroéclairage toute la journée. On s'éclate les yeux. »* précise Thomas. La lumière bleue est une lumière typique des écrans LED, mais aussi sur les OLED et les AMOLED. L'impact sur la santé est désormais connu et prouvé.

> *« L'implémentation n'est pas aussi simple »*

Le mode sombre comme résolution à la lumière bleue ?

Oui et non. *« Le monde sombre émet moins de lumière bleue, donc moins de fatigue oculaire, sur les OLED et les AMOLED, il y a aussi moins de pixels rétroéclairés. Nous devons gérer la santé des utilisateurs »* poursuit Thomas.

Il est donc conseillé d'activer le mode sombre le soir/la nuit. Certaines applications ou devices utilisent par défaut ce mode : les montres, Netflix. Une des raisons est une meilleure visibilité des couleurs sur un fond sombre que sur un fond clair/blanc. Mais contrairement à son appellation, il ne s'agit pas d'une teinte pure noire. Car un pur noir avec un pur blanc fournirait un ensemble brillant et donc moins lisible. En réalité, il s'agit plutôt d'un gris très foncé.

Android propose le dark theme et iOS, le dark mode. Grosso modo, ils fonctionnent de la même manière. La documentation, les bonnes pratiques sont disponibles : Material Design côté Android et les human interface guidelines chez iOS.

Comme le précise Thomas, par exemple, en thème clair (dit light), les couleurs/les teintes sont souvent saturées. En mode sombre, il ne faut pas utiliser de couleurs saturées, car le rendu visuel ne serait pas bon. Il faut donc faire attention aux codes couleurs utilisés et bien respecter les recommandations sur chaque plateforme.

Cela signifie aussi que vous devrez concevoir 2 assets graphiques, un pour chaque mode. Ce n'est pas vrai pour toutes les apps, mais pour un projet propre et une bonne séparation des couches, c'est mieux. Pour Thomas, il n'y a pas d'impact significatif sur le poids des binaires générés, sauf des cas très spécifiques. Sur les performances de son app, le mode sombre, là encore, pas d'impacts significatifs.

Par contre, le mode sombre est un avantage pour la batterie, donc la consommation. Un pixel noir (ou gris foncé) ne consomme pas, c'est l'un des intérêts de l'OLED.

Pour quelles apps ?

Potentiellement, toutes les apps sont concernées par ce mode. Certaines apps vont directement en tirer parti comme celles avec des graphiques couleurs. L'app bourse d'Apple est un bon exemple : fond noir, les chiffres, les textes et les courbes sont visuellement nettes. Et donc, l'œil voit mieux les différents éléments.

Sur Android, le support de Dark Theme se veut simple :

```
<style name="AppTheme » parent="Theme.AppCompat.DayNight">
```

Ou via MaterialComponents :

```
<style name="AppTheme » parent="Theme.MaterialComponents.DayNight">
```

L'utilisateur peut aussi changer le thème lui-même dans l'app si vous l'autorisez : light, dark et system default. Sur la partie iOS, le travail est un peu compliqué de par la structure du SDK et des langages, mais c'est relativement rapide à faire.

Un bon exemple d'implémentation : https://www.avanderlee.com/swift/dark-mode-support-ios/

Ressources

Android 10 minimum (API level 29) : https://developer.android.com/guide/topics/ui/look-and-feel/darktheme

iOS 13 minimum : https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/dark-mode/

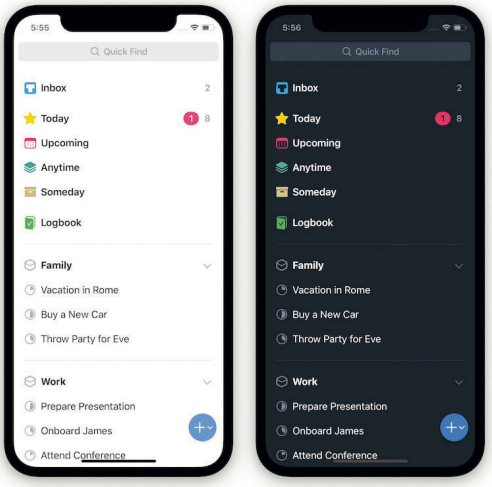
Exemple sur iOS : https://developer.apple.com/documentation/uikit/appearance_customization/adopting_ios_dark_mode

KÉZAKO LA LUMIÈRE BLEUE ?

La lumière bleue fait partie du spectre de la lumière visible. Sa longueur d'onde est comprise en 380 et 500 nanomètres. Le soleil en émet. Elle est émise par certains écrans comme les LED, les LCD. Chez les enfants, le cristallin de l'œil n'est pas encore un filtre naturel optimal. Des études montrent qu'elle influence, négativement le sommeil et peut perturber le rythme biologique et même endommager, sur la durée, l'œil.

FOCUS SUR LUNABEE

Lunabee Studio est initialement éditeur d'Apps mobiles - avec oneSafe notamment, un password manager avec plus de 2,5 millions d'utilisateurs dans le monde. oneSafe est leur plus gros succès, avec des mises en avant à de multiples reprises sur les App Stores, et le lancement d'une version 4 cette année qui s'est retrouvée dans le Top 25 des Apps payantes aux US ! Depuis 2014, Lunabee Studio a lancé son studio de création d'Apps.



John Thiriet
Global Mobile Tech Lead
chez **Edenred SA**
Microsoft MVP
@johnthiriet

Quelle technologie mobile multiplateforme choisir en 2020 ?

A cette question il y a deux approches. Quelle technologie apprendre et quelle technologie utiliser dès à présent pour son application mobile.

Rassurez-vous, ça ne sera ni le SDK Flutter, ni React Native, ni Xamarin, ni rien. La seule réponse valable à cette question est « Ça dépend ». Alors, installez-vous confortablement dans votre fauteuil, on va commencer.

Un peu de contexte

Travaillant dans la mobilité depuis 2011 d'abord sur Windows Phone suivi d'iOS et Android depuis 2014 avec Xamarin, j'ai parfois subi, mais toujours étudié et choisi les évolutions technologiques qui m'intéressaient. Je travaille maintenant en tant que Lead de plusieurs équipes mobiles et ce poste particulier m'a offert un point de vue différent sur les critères de choix d'une technologie mobile. Afin de lever toute ambiguïté, je précise que même si Xamarin est ma technologie de cœur depuis de nombreuses années, cet article se veut le plus objectif possible.

Avant toutes choses faisons un récapitulatif des technologies de développement mobile multiplateforme ayant une certaine forme de popularité en décembre 2019.

Cordova

Apache Cordova est l'une de première approches multiplateforme ayant vraiment gagné en popularité.

Développer en Cordova

Le développement en Cordova ne nécessite pas d'IDE particulier. Il est possible de travailler avec Visual Studio, Visual Studio Code ou n'importe quel autre éditeur.

On développe son application en utilisant les technologies web (*HTML*/*JavaScript*/*CSS*). C'est donc ni plus ni moins qu'un site web mais dans une coquille d'application native. Cordova permet le développement d'applications ciblant iOS, Android ou Windows et ce de manière très rapide grâce aux outils de développement web.

Il est par ailleurs devenu rare de développer une application directement en Cordova. On utilise plus généralement une surcouche graphique telle qu'Ionic.

Ionic permet d'avoir une présentation et un mode de fonctionnement s'approchant du natif et ainsi sortir de l'aspect très web du design. Il est de plus couplé à des frameworks applicatifs web tels que Angular, VueJS ou plus récemment ReactJS.

Fonctionnement

Le fonctionnement de Cordova est extrêmement simple et maintenant bien connu. Il crée une application native avec toutes les dépendances dont l'application a besoin, embarque le contenu *HTML*, *CSS*, et *Java Script* et lance le tout dans une webview.

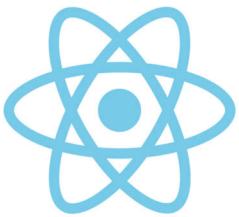
Accès à la plateforme native

Cordova fournit l'accès aux fonctions natives de la plateforme grâce à l'aide de plugins écrits en code natif et exposés à *JavaScript*. Il existe énormément de plugins à l'heure actuelle, il y a donc peu de chances pour qu'un développeur Cordova soit confronté au besoin d'écrire du code natif.

Avantages et inconvénients

Puisque l'application fonctionne intégralement dans une WebView, les performances sont inférieures à la concurrence et l'expérience utilisateur pourra être dégradée. En revanche cela permet de transformer un site web en application mobile grâce l'utilisation de plugins comme *cordova-plugin-remote-injection*. Également, l'utilisation de technologies web permet un cycle de développement rapide. Cordova convient bien aux application type formulaires, des applications de présentation simples, compagnons de sites web ou bien des applications internes. Il est assez facile de trouver des compétences sur cette technologie sur le marché. Il ne faut pas oublier que sans développeurs natifs dans l'équipe, il y a un risque de blocage si un des plugins utilisé ne donne pas satisfaction.

React Native



Nouveau venu de Facebook en 2015, il a beaucoup fait parler de lui car il permet d'utiliser peu ou prou la même approche que ReactJS mais pour développer une application native. ReactJS étant un des grands frameworks applicatifs web avec Angular ou VueJS, React Native est vite devenu extrêmement populaire auprès des développeurs web. React Native permet de développer des applications pour Android, iOS et Windows 10, mais également de partager une partie de son code *JavaScript* avec une application web en ReactJS si les choses sont bien faites.

Il est fréquemment couplé avec Expo, qui permet de se passer de Mac ou de déployer des mises à jour de l'application sans passer par les stores.

Développer en React Native

Un des éditeurs les plus utilisés avec React Native est Visual Studio Code.

Le langage de développement est *Java Script* mais l'interface est écrite en *JSX* avec un ensemble de contrôles spécifiques à React Native. On ne parle pas ici de *div* mais de *View*. Les styles sont écrits avec une abstraction de CSS. Le positionnement est basé sur *flex layout* et permet donc une adaptation aisée aux différents appareils mobiles. La fonctionnalité de rechargement à chaud est présente depuis le début et permet un cycle de développement très rapide inédit à cette échelle dans le monde mobile natif quand React Native l'a introduit.
En React Native, le code d'interface est écrit qu'une seule fois pour toutes les plateformes mais peut être personnalisé par plateforme. L'approche préférée est une approche par composants où chaque composant dispose, ou devrait disposer, de son fichier *JSX* et de sa logique propre.

Fonctionnement

React Native affiche des contrôles natifs, positionnés dans l'écran par ses propres soins. Il utilise un mécanisme de gestion d'états type MVU (Model View Update). Chaque modification de cet état entraîne un re-dessin de l'écran. Cette opération étant coûteuse, React Native essaie de voir ce qui a vraiment changé et ne re-dessine que la portion de l'écran nécessaire.

Accès à la plateforme native

L'accès à la plateforme native se fait à l'aide de modules écrits en code natif et exposés en *JavaScript*. La bibliothèque de modules est aujourd'hui importante mais le risque de devoir en faire un personnalisé est toujours très présent.

Avantages et inconvénients

Le gain de popularité rapide de ce framework a été entâché de quelques problèmes de jeunesse : des histoires de licences avec Facebook, des soucis pour avoir une navigation propre ou bien son abandon parfaitement documenté par Airbnb. Le temps passant pas mal de ces problèmes se sont estompé et, bien que le côté « hype » de la chose soit passé, le framework reste aujourd'hui très populaire.
React Native n'est pas ReactJS, malgré la forte ressemblance il y a donc une mise à niveau nécessaire pour le développeur web. Il est souvent utilisé dans des contextes où il faut un bon mélange entre vitesse de développement, bonnes performances, interface native et où les développeurs sont familiers avec ReactJS.
C'est un framework sur lequel il est relativement aisé de trouver des développeurs notamment des développeurs web souhaitant se mettre au mobile.

Flutter

Flutter est le framework « hype » du moment. Édité par Google, il propose une approche assez unique, fusion des approches précédentes. Il cible iOS, Android mais aussi le web avec le projet Hummingbird. C'est un tout jeune framework qui est très prometteur et fait beaucoup de charmants sourires aux développeurs React Native et Xamarin.

Développer en Flutter

Il est habituel de travailler soit avec Android Studio soit avec Visual Studio Code sur Flutter. Le plugin Visual Studio Code étant d'ailleurs édité par Google.
Le langage utilisé pour le développement d'applications Flutter est *Dart*, un langage maison qui commence à avoir pas mal de vécu et qui évolue beaucoup. *Dart* est un hybride entre du *JavaScript* et des concepts de programmation très modernes comme *await*, les flux asynchrones etc… Il est capable de fonctionner en interprété ou en compilé. Il permet ainsi des cycles de développement rapides grâce au rechargement à chaud et de très bonnes performances en compilé. Avec Flutter, le code d'interface n'est écrit qu'une seule fois également en *Dart*. L'approche par composants est privilégiée et les familiers de React ne seront vraiment pas dépaysés.

Fonctionnement

Les vues ne sont pas natives, tous les contrôles ont été réécrits en C++ et l'intégralité de l'interface graphique est rendue en OpenGL avec Skia, avec des thèmes Material ou Cupertino.
Flutter est constitué d'un runtime qui embarque toutes les fonctionnalités de bases telles que l'accès au système de fichiers, au réseau ou bien encore aux animations. Il fait également le pont avec les API natives. Lors du développement de l'application, le code *Dart* est interprété permettant le rechargement à chaud mais lors de la publication il est compilé en AOT (ahead of time) s'assurant de bonnes performances.

Accès à la plateforme native

L'accès à la plateforme native se fait grâce à des plugins écrits en code natif et exposés à la manière de React Native ou Cordova. Si les besoins de l'application ne sont pas couverts par des plugins, il faudra donc écrire du code natif et vue la jeunesse de la plateforme la probabilité que cela arrive est assez importante.

Avantages et inconvénients

Son approche si particulière fait sa force et sa faiblesse. Ainsi l'intégration de webviews n'est pas toujours aisée et l'utilisation de SDKs tiers nécessitant une UI native hasardeuse.
Les performances graphiques proposées en Flutter sont excellentes et il est souvent utilisé pour des applications graphiquement très riches ou avec beaucoup d'animations et ou la vitesse de publication prime sur la vision long terme.
L'utilisation de plugins écrits en code natif induit le besoin pour une équipe de développement Flutter de compétences Java/Kotlin ou Objective C/Swift. A la différence de tous les frameworks de cet article, Flutter ne supporte pas Windows.
Le fait que Google soit l'éditeur rassure, car on sait que le framework suivra les évolutions d'Android. Mais Google est également bien connu pour tuer sans pitié ses produits.
Avec à la relative jeunesse du SDK il me semble prématuré de baser une stratégie d'entreprise ou d'une application mobile longue durée sur Flutter pour le moment.
Flutter n'a pas encore eu son « Airbnb » comme pour tous les autres frameworks. Ce genre d'abandon est un signe que le framework commence à être éprouvé. C'est un bon marqueur indiquant que l'on passe de la « hype » à l'utilisation raisonnée de la technologie. Il est très difficile de trouver des développeurs Flutter à l'heure actuelle, mais pas nécessairement difficile d'en trouver qui veulent s'y mettre. En revanche si vous cherchez à vous lancer dans le développement mobile aujourd'hui ou à apprendre un nouveau framework je ne peux que vous recommander de vous y mettre.

Xamarin

Créée il y a un peu plus de 10 ans sous les noms MonoDroid et MonoTouch, Xamarin est une technologie éprouvée, stable qui a beaucoup évoluée. Il existe deux approches de développement, Xamarin Native et Xamarin Forms. Il me semble important de noter que même si Xamarin est une technologie Microsoft, elle est surtout open source et les contributions de la communauté sont les bienvenues. Xamarin fonctionne de manière officielle sur une grande variété de plateforme : Android, iOS, Tizen, MacOS, UWP, etc... Il existe des ports non officiels ou des tests fait pour d'autres plateformes telles que WPF, Linux ou le web avec Web Assembly et Blazor.

Développer avec Xamarin

Xamarin est basé sur Mono, le premier framework .NET open source disponible également sous Linux. C# y est le langage roi. On accès à toutes les fonctionnalités de .NET et l'interaction avec l'écosystème Microsoft et/ou Azure y est grandement facilité. Pour développer avec Xamarin, on utilise Visual Studio, Visual Studio for Mac ou Rider. Il est possible de travailler et débogguer une application iOS depuis Windows à condition d'avoir un Mac quelque part sur le réseau. Cela s'avère pratique dans certaines entreprises où les développeurs n'ont accès qu'à des machines Windows.

Fonctionnement

Xamarin est un framework qui expose directement les APIs natives aux développeurs dans un style adapté au développement .NET. Il est constitué d'un runtime et le code est directement compilé en byte code natif sur iOS ainsi que sur Android si vous faites le choix de l'AOT. Des mécanismes permettent de supprimer le code non utilisé à la compilation ce qui réduit de manière conséquente la taille des applications.

Accès à la plateforme native

En Xamarin, on accède à plateforme native directement en C#. Il existe un dépôt de frameworks natifs directement disponibles, vous pouvez également créer les vôtres en C# si besoin. L'ajout de bibliothèques natives n'existant pas dans ce dépôt est possible grâce à un processus appelé *binding*. Cette approche permet d'exposer l'intégralité d'un SDK natif et de l'utiliser directement sans avoir de faire de plugins. Cette possibilité d'intégration directe et complète de SDK tiers est un des points sur lesquels Xamarin se démarque. A noter que faire son propre *binding* est extrêmement rare puisque la quasi totalité des bibliothèques les plus populaires sont déjà disponibles directement.

Xamarin Native

Voyez Xamarin Native comme du développement natif mais où on remplace *Swift*, *Kotlin*, *Objective C* et *Java* par du *C#* ou, plus rarement du *F#*.
Ici, point question de code graphique partagé. On développe chaque interface en utilisant les mêmes outils que le développement natif : *Storyboards* et *Xib* sur iOS; *Activity* et *Fragment* sur Android.
Cela demande plus de code que certaines approches, mais c'est aussi la moins risquée pour un projet car, plus on est proche de la plateforme, plus il est plus facile de suivre les évolutions de cette dernière. La surcouche étant aussi minimale c'est une approche qui offre d'excellentes performances. Les SDKs tiers étant généralement prévus pour les applications natives ils sont facile d'utilisation. On reproche souvent à Xamarin Native de ne pas évoluer rapidement, mais son positionnement fait qu'il évolue en même temps que les plateformes cibles, ni plus, ni moins.

Xamarin Forms

Xamarin Forms est une surcouche à Xamarin Native qui propose un code UI partagé entre les différentes plateformes. Ce code peut être en *C#*/*F#* ou plus communément en *XAML*. C'est un langage bien connu des développeurs .NET qui le rend accessible facilement par ces derniers. C'est projet extrêmement dynamique puisqu'il est relativement indépendant des évolutions de la plateforme.
Tout comme en React Native, la disposition des éléments est gérée par le framework alors que les contrôles sont eux natifs. Il est d'ailleurs possible de redescendre au niveau natif à tout moment ou de mélanger à la fois Xamarin Forms et Xamarin Native dans le même projet. L'approche de développement la plus commune est ici MVVM. Des tentatives d'implémentation du pattern MVU sont en cours notamment avec les projets Comet et Fabulous. L'approche par composant est également possible mais pas imposée. Il est depuis peu doté d'une fonctionnalité de rechargement à chaud et d'un système de templating abstrayant la navigation et gérant de base les liens profonds, nommé Shell. Il y a aussi la possibilité récente de choisir un univers visuel commun aux applications iOS et Android avec *Visual* afin de ne plus avoir de différences entre les deux plateformes si tel est le besoin de l'application.

Avantages et inconvénients

Xamarin est une technologie très complète, plus ancienne que Flutter et React Native et qui manque de ce sentiment de nouveauté. C'est une technologie éprouvée, robuste et fiable qui réserve beaucoup moins de surprises qu'à ses débuts. Elle cible une très vaste variété de plateforme et le code .NET écrit est portable partout où .NET existe. Le fait d'avoir par défaut un rendu différent par plateforme fait qu'il est absolument nécessaire d'avoir un Mac quelque part pour travailler. Xamarin nécessite surtout qu'on le comprenne très bien pour en tirer le meilleur. La phase d'apprentissage n'est pas négligeable pour adapter le développeur .NET au milieu mobile ou le développeur mobile au monde .NET. C'est un framework sur lequel il est un peu difficile de trouver des développeurs disponibles en France, ce qui n'est pas nécessairement le cas d'autres pays.

Récapitulatif

Après ce tour d'horizon des particularités de chaque framework je vous propose de regarder ce tableau qui peut servir de base de décision. (voir tableau)
On le voit, chaque technologie a une approche différente. Ces choix induisent un subtil mélange d'avantages en productivité, en performances, en stabilité, en fiabilité, ou en facilité d'apprentissage.

Quel choix faire ?

Le choix qu'une personne fera d'une technologie de développement mobile dépendra de ce que cette personne considère comme une force ou une faiblesse dans chaque technologie. C'est donc, par définition, quelque chose de subjectif. Il faut également prendre en compte la typologie de l'application à développer. Afin de sortir de l'idée qu'il y aurait une technologie magique pour les gouverner toutes, je vous propose de suivre plusieurs personnes, techniques ou

non qui auront des réponses différentes à cette question. Je ne prétends pas couvrir tous les cas possibles mais je vais tenter d'adresser chaque grande catégorie de développeurs et de typologies de projets.

Les scénarios types

Scénario Cordova

George, développeur web depuis de nombreuses années, spécialisé en Angular, souhaite se réinventer et commencer le développement mobile. Il a une application mobile à développer sur iOS et Android pour un client et n'a ni le temps ni l'envie de se mettre au développement natif. Son expérience personnelle va naturellement le faire se tourner vers Cordova ou React Native. L'application qu'il doit développer est une simple application de 4 écrans qui n'a besoin d'aucun design particulier et d'aucune haute performance. Il va donc pragmatiquement faire le choix de Cordova. Cela lui permettra de se familiarisera au mobile et pourra peut-être commencer React Native après.

Scénario React Native

Yacine, développeur ReactJS a eu une idée géniale d'application mobile. Il monte une startup et doit vite pitcher pour obtenir des fonds. Son expérience en ReactJS lui permet de commencer plus rapidement React Native. Même s'il y a un peu de travail de mise à niveau le jeu en vaut la chandelle car il faut montrer un design natif et moderne pour convaincre. Il aura tout le temps de réévaluer son choix par la suite.

Scénario Flutter

Tingting, développeuse mobile Android a découvert Flutter lors des dernières conférences Google. Sa société fait actuellement exclusivement du développement Kotlin mais puisqu'elle est sur Mac et qu'elle a un iPhone elle se dit que Flutter est probablement un bon choix pour commencer iOS. Elle reste dans l'écosystème Android avec les outils qu'elle connaît comme Android Studio et peux faire ses premiers pas en auto-formation.

Scénario Xamarin Native

Dominique travaille dans une grosse entreprise et lance un projet s'étalant sur au minimum 3 ans avec des releases régulières. La cible est d'au minimum 2 millions d'utilisateurs dont on ne sait pas de quels téléphones ils sont équipés. Afin de mieux comprendre le comportement des utilisateurs et pouvoir faire des rapports précis, cette application intégrera divers SDKs natifs dont certains afficheront des questionnaires aux utilisateurs. Après avoir étudié les diverses options de technologies multiplateforme il s'avère que c'est Xamarin Native qui répond le mieux au besoin et ça tombe bien parce qu'il y a des développeurs .NET dans plusieurs services de la société.

Scénario Xamarin.Forms

Travaillant dans une société vendant des produits écolos sur internet, Makoto se rend compte qu'elle a besoin rapidement d'une application mobile. Le site de la société est fait en .NET Core et elle sait que sa cible est plutôt aisée et dispose de téléphones haut de gamme. Elle est donc en position idéale pour lancer le développement d'une application Xamarin Forms avec toutes les dernières technologies afin d'accélérer ses développement et rester dans l'écosystème de la société.

Récapitulatif

Ces quelques personnes sont vraiment les cas idéaux pour le choix d'une technologie donnée. Si vous êtes dans ce genre de situation le choix est assez évident. Malheureusement on est rarement dans une position si claire et dans ce cas là vous devrez peser le pour et le contre de chaque technologie. Confronter les points de vues et vous pourrez décider de ce qui est le mieux pour votre projet, votre entreprise et pour vous-même.

Pour finir : Le bon, la brute et le truand

Comme je le dis souvent, lorsqu'un projet multiplateforme échoue, tous se retournent contre le framework alors que si un projet natif échoue on blâmera les développeurs. Quels que soient les choix que vous ferez, rappelez-vous qu'il existe parfois une différence entre ce qui éveille votre intérêt, ce que vous avez envie de faire et ce qui est fondamentalement bon pour votre projet. De la même manière, il n'existe aucune technologie parfaite qui répondra à tous les besoins. Si après une étude impartiale, le meilleur choix pour votre projet n'est pas ce que vous souhaiteriez, soyez pragmatiques au travail et amusez-vous sur votre temps libre. Quelle que soit la technologie, un mauvais développeur fera une mauvaise application, et l'inverse est vrai également. Il existe de très belles applications dans toutes les technologies citées. Il est de notre responsabilité de professionnels du développement de savoir d'abord étudier les solutions qui s'offrent à nous et d'assumer nos erreurs dans les choix techniques ou dans l'implémentation.
Alors choisissez bien et surtout continuez d'apprendre ! •

	Cordova	React Native	Flutter	Xamarin	Xamarin Forms
Langage	HTML / Javascript	JavaScript / JSX / CSS	Dart	C# / F#	C# / F# / XAML
IDE		Visual Studio Code / Autres	Visual Studio Code / Android Studio	Visual Studio / Visual Studio for Mac / Rider	Visual Studio / Visual Studio For Mac / Rider
Approche	Hybride web	Hybride natif	Hybride natif	Hybride natif	Hybride natif
Pattern habituel		MVU	MVU	MVVM	MVVM
Accès à la plateforme	Plugins (code natif)	Modules (code natif)	Plugins (code natif)	Accès direct à la plateforme	Accès direct à la plateforme
Editeur	Apache	Facebook	Google	Microsoft	Microsoft
Contrôles	Web	Natif	Custom	Natif	Natif
Layout	Web	Custom	Custom	Natif	Custom
Rechargement à chaud	Oui (web)	Oui	Oui	Oui	Oui (stateless)
Code UI	Partagé	Partagé	Partagé	Séparé	Partagé

Camille Le Luët
Développeuse mobile,
Infinite Square
https://blogs.infinitesquare.com

INFINITE SQUARE

A la découverte de Flutter

niveau 100

En ce moment, quand on parle de développement d'applications mobiles, beaucoup de gens n'ont que le mot Flutter à la bouche.

Framework opensource de développement d'applications mobiles natives créé par Google en 2017, Flutter permet d'écrire des applications Android et iOS en un seul code partagé, à la manière de Xamarin.Forms. Flutter utilise des SDK d'Android et d'iOS afin d'avoir un rendu natif, que ce soit en utilisant le Material Design ou Cupertino. Il est ainsi possible d'avoir un rendu d'application utilisant le material design sur iOS, ainsi qu'une application cupertino sur Android. Comme pour tout développement d'application iOS, une machine macOS est nécessaire pour compiler l'application iOS.
L'objectif du dossier est d'avoir une première compréhension de Flutter, savoir comment l'installer sur Windows et iOS, et débuter une toute première app avec une prise en main basique.
La partie Linux sera abordé dans un futur article.

Comment ça marche ?

Le langage Dart

Flutter utilise le langage de programmation Dart, également inventé par Google. C'est un langage facile à prendre en main lorsqu'on connait C#, Java, etc. Concernant les éditeurs, on peut utiliser Android Studio en installant les plugins Flutter et Dart. Mais mon chouchou reste cependant Visual Studio Code qui supporte également Flutter après installation du plugin (qui installe également le plugin Dart). Contrairement à React Native, les applications Flutter, grâce à Dart, sont compilées AOT (Ahead of Time). Ceci permet de générer une application native pour Android et pour iOS. Ainsi le code est optimisé pour l'architecture de chaque plateforme. Une autre différence avec Xamarin et React Native est que Flutter dessine l'interface en utilisant Skia, au lieu d'être un wrapper au-dessus des composants UI natifs à Android et iOS. Un des principaux avantages de dessiner l'interface avec Skia est que l'interface utilisateur sera identique sur Android et sur iOS. Les applications Flutter se veulent pixel perfect, ainsi le design de l'application n'est pas dépendant de l'OS. Mais c'est aussi un inconvénient car le rendu des widgets ne sera pas natif à l'OS, ce ne sont pas les composants natifs.

La programmation réactive

La programmation réactive est basée sur des concepts comme des variations des valeurs dans le temps, des flots d'événements pour modéliser les mises à jour discrètes, des suivis de dépendances, et des propagations automatiques du changement
À tout instant T de l'exécution du programme, toutes les expressions du programme doivent rester correctes. Par conséquent, les valeurs des variables évoluent dans le temps au fur et à mesure de l'exécution du programme de manière automatique et discrète
En Flutter, la programmation est donc réactive, c'est à dire que l'interface réagit aux inputs de l'utilisateur en changeant des propriétés et/ou variables. Au changement de ces propriétés, l'interface est redessinée en fonction du nouvel état. C'est donc une modification des propriétés qui déclenche une modification de l'UI et non les fonctions qui la modifient.

Les widgets

En Flutter, l'interface se construit à l'aide de widgets. Chaque widget permet de décrire ce à quoi sa vue doit ressembler en fonction de sa configuration et de son état. Lorsque l'état d'un widget change, le framework calcule la différence avec l'état précédent afin de déterminer les modifications minimales nécessaires dans l'arbre de rendu pour passer de l'un à l'autre. Le catalogue des widgets Flutter se trouve sur https://flutter.dev/docs/development/ui/widgets.
Chaque widget est une sous-classe de StatelessWidget ou de StatefulWidget et doit implémenter la fonction build qui a pour but de décrire l'interface utilisateur du widget.
Les widgets peuvent être stateless ou stateful
Les widgets dits **stateless** sont des widgets qui ne gèrent pas d'états. Prenons par exemple le TextWidget, il ne possède pas d'état. En allant voir son code source on voit qu'il étend la classe StatelessWidget. Il est instancié via son constructeur qui construit ensuite le widget qui va être affiché à l'écran en fonction de ses propriétés, via la fonction build. Dans notre cas du TextWidget, il suffit donc de passer en paramètre son texte et son style par exemple pour le créer.
Un widget stateless n'est donc pas dynamique. Il ne dépend de rien en dehors de propriétés qui lui sont passées dans son constructeur. Ainsi, si une propriété du widget change, l'interface du widget ne sera pas redessinée.
Les widgets dits **stateful**, eux, sont dynamiques. C'est à dire, ils changent en fonction des inputs, du résultat d'une requête asynchrone. Par exemple, le widget Image, en regardant son code source on peut voir qu'il étend cette fois la classe StatefulWidget. Tout comme notre exemple précédent, le widget Image s'instencie en passant des paramètres à son constructeur. Mais cette fois, il a la méthode createState(). Cette méthode permet de créer l'état de notre widget.
Toujours avec notre exemple d'Image, nous avons donc une classe _ImageState, étendant State<Image>, qui définit l'état du widget Image. Dans notre cas d'un widget stateful, c'est cette classe qui implémente la méthode build construisant l'interface. Les modifications dynamiques de l'interface se font au travers des modifications des propriétés (en manipulant des Stream) ou bien au travers de méthodes appelant la méthode setState(). Par exemple :

```
setState(() {
    _imageIngo = imageInfo;
});
```

L'appel à cette méthode déclenche un nouvel appel à la méthode build du widget avec le nouvel état.

Un widget stateful est donc dynamique, il écoute les modifications et met à jour l'interface en fonction de ces changements.

Le hot reload

Dernier point, mais pas des moindres : Flutter inclut la fonctionnalité de hot reloading qui permet de rapidement tester des interfaces utilisateurs, des fonctionnalités, corriger des bugs lors du débogage. Le hot reloading fonctionne en injectant le code source modifié dans la machine virtuelle Dart en cours d'exécution (téléphone, émulateur ou simulateur). Un fois le nouveau code injecté, le framework reconstruit l'arborescence de widget.

Pour utiliser le hot reloading, il suffit de lancer l'application en mode debug sur votre device (physique ou virtuel), modifier un fichier puis d'enregistrer les modifications (ou de cliquer sur le bouton de reload dans la barre), and TADA ! la page est à jour avec les modifications.

Flutter semble être un framework très prometteur. Le langage Dart ainsi que le système de widgets sont faciles à prendre en main. Pour ma part, j'ai créé une simple petite application afin de faire mes premiers tests. Le développement se fait rapidement et assez simplement. Voyons voir comment installer Flutter et créer une première application afin de pouvoir commencer à jouer avec le code :

L'installation pour Windows

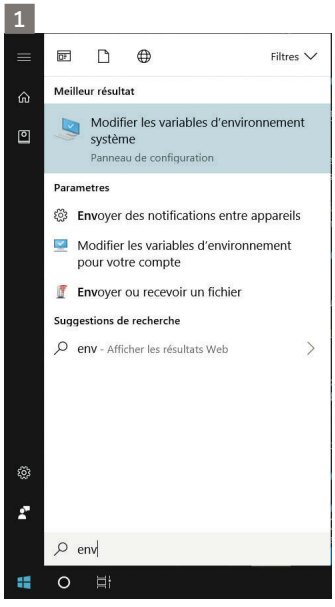
Configuration requise

- Système d'exploitation : Windows 7 SP1 ou plus récent (64-bit)
- Espace disque : 400 mo (sans inclure l'espace pour l'IDE/tools).
- Outils :
  - Windows PowerShell 5.0 ou plus récent (pré-installé sur Windows 10)
  - Git pour Windows 2.x, l'option d'utilisation de Git pour l'invite de commande Windows.

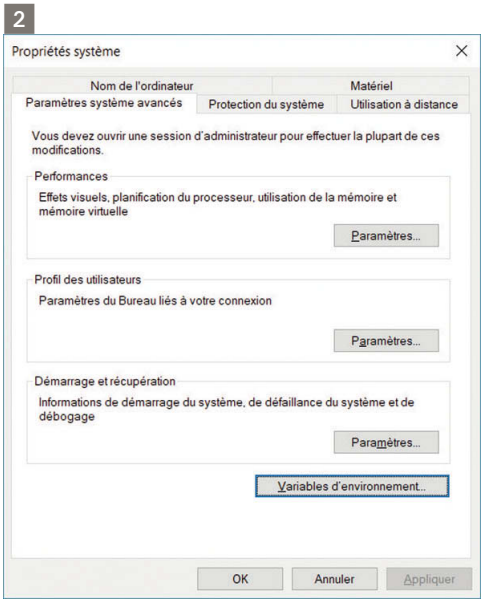
SDK Flutter

Si la confirmation de votre PC est conforme, téléchargez le SDK Flutter. Une fois le SDK téléchargé, il faut extraire l'archive dans le dossier choisi (par exemple : C:\src\flutter). Il ne faut pas installer Flutter dans un dossier comme C:\Program Files car cela requiert des privilèges élevés.

1



2



Maintenant, partons à la recherche du fichier flutter_console.bat dans notre dossier tout nouveau, tout beau, tout neuf, et lançons-le en double-cliquant dessus. Ce fichier a ouvert la console Flutter, on peut donc maintenant taper nos commandes Flutter ici.

Je ne sais pas vous, mais moi ça m'ennuie un peu de devoir aller chercher ce fichier avec la console flutter tout le temps pour taper mes commandes. Du coup nous allons voir comment faire pour pouvoir utiliser les commandes Flutter dans la console Windows. Pour cela, il faut ajouter Flutter aux variables d'environnement.

Allons donc chercher le panneau de modification des variables d'environnement (taper "env" dans le menu démarrer). 1 2

Cliquer sur "Variable d'environnement". Dans la partie des "Variables utilisateur", repérer la variable "**Path**" (ou la créer si inexistante) et ajouter le chemin vers le dossier bin de Flutter (C:\src\flutter\bin dans mon cas). Vérifions maintenant notre installation à l'aide de la commande **flutter doctor**. Cette commande vérifie votre environnement et affiche un rapport avec le statut de l'installation de Flutter. Voici donc mon output :

```
1  Doctor summary (to see all details, run flutter doctor -v):
2  [√] Flutter (Channel stable, v1.2.1, on Microsoft Windows [version 10.0.17134.648],
   locale fr-FR)
3  [✗] Android toolchain - develop for Android devices
4    ✗ Unable to locate Android SDK.
5      Install Android Studio from: https://developer.android.com/studio/index.html
6      On first launch it will assist you in installing the Android SDK components.
7      (or visit https://flutter.io/setup/#android-setup for detailed instructions).
8      If Android SDK has been installed to a custom location, set ANDROID_HOME to
   that location.
9      You may also want to add it to your PATH environment variable.
10 [!] Android Studio (not installed)
11 [√] VS Code (version 1.32.1)
12 [!] Connected device
       ! No devices available
```

Effectivement, venant de Xamarin et n'ayant jamais renseigné la variable ANDROID*HOME*, *Flutter est incapable de trouver l'emplacement de mon SDK Android. Pour réparer cela, nous pouvons simplement installer Android Studio, ou bien renseigner la variable ANDROID_HOME*.

Installation Android

Option 1 : Installation d'Android Studio

Si vous souhaitez l'utiliser plus tard, ou juste vous simplifier la tache, téléchargez Android Studio et suivez les étapes de l'installation. Une fois l'installation terminée, lancez l'IDE et allez dans *Préférences > Plugins*, puis dans *Browse Repositories*, recherchez le plugin Flutter. Il va ensuite vous proposer d'installer le plugin Dart, qui est nécessaire au fonctionnement de Flutter.

Option 2 : Renseigner la variable ANDROID_HOME

Pour rester fidèle à mon chouchou Visual Studio Code, j'ai choisi la cette deuxième option c'est donc cette partie qui sera la plus détaillée.

Après avoir repéré l'emplacement de notre SDK Android (ici dans mon cas : C:\Program Files (x86)\Android\android-sdk), retournons donc voir nos variables d'environnement (voir précédemment). Dans la partie des "variables système" cette fois, ajoutons (via le boutons Nouvelle...) notre variable ANDROID_HOME. Ajoutons

également cette valeur dans la variable PATH des "Variables utilisateur" (comme dans la section précédante). 3
Appelons une nouvelle fois le docteur avec flutter doctor, et voici notre configuration terminée :

```
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, v1.2.1, on Microsoft Windows [version 10.0.17134.648], locale fr-FR)
[√] Android toolchain - develop for Android devices (Android SDK version 28.0.3)
[!] Android Studio (not installed)
[√] VS Code (version 1.32.1)
[!] Connected device
    ! No devices available
```

Une nouvelle fois, vous avez la possibilité d'utiliser Android Studio, dans ce cas il suffit de l'installer, et ce warning disparaitra. Dans mon cas, j'ai choisi d'utiliser Visual Studio Code. Le dernier warning *Connected device* disparaitra dès que vous connecterez votre téléphone Android au PC.

Installation iOS

C'est le moment de la grande déception... Contrairement à Xamarin qui permet de développer des application iOS en se connectant à un mac distant, Flutter n'a rien de tel. Ainsi, si votre application cible iOS, vous pouvez abandonner votre cher et tendre PC Windows et basculer du côté obscur en développant depuis votre mac. Pas de panique si vous êtes dans ce cas, rendez-vous à la section suivante.

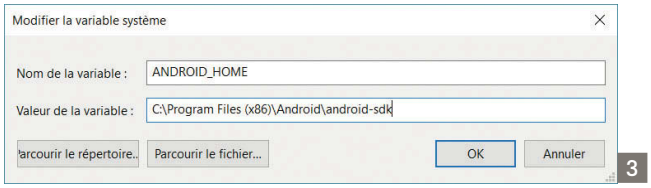
Mettre à jour Flutter

Pour mettre à jour le SDK Flutter ainsi que les packages dont dépend l'app, il suffit d'utiliser la commande flutter upgrade dans le dossier principal de votre application (qui contient pubspec.yaml). Cette commande récupère la dernière version du SDK disponible sur le canal Flutter puis met à jour tous les packages de l'application dans la version compatible la plus récente. Pour utiliser d'autres canaux que le canal Flutter il suffit d'utiliser la commande flutter channel <channel-name> juste avant flutter upgrade.

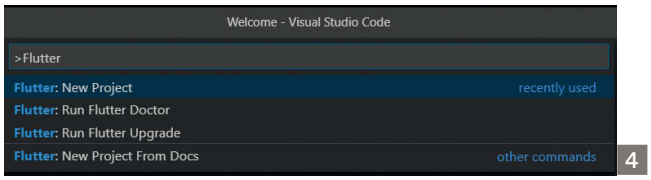
Première application

Si vous avez choisi d'utiliser Visual Studio Code, il faut (tout comme Android Studio) installer les extensions Flutter et Dart. Pour cela, allons dans la palette de commandes (Ctrl+Shift+P) puis choisissons *Extensions: Install Extensions*. Dans le menu qui vient de s'ouvrir, nous pouvons chercher "Flutter" (à l'aide de la barre de recherche) puis l'installer. Cette extension installe Dart en même temps. Vous pourrez ensuite utiliser la commande flutter doctor directement dans VS Code.
Maintenant que notre environnement est prêt, créons notre nouvelle application. Avec VS Code, cela se fait à l'aide de l'option *Flutter: New Project* accessible dans la palette de commandes (Ctrl+Shift+P). Ensuite, tapons "flutter" et la commande *Flutter: New Project* apparait 4
Choisissons un nom de projet et le dossier dans lequel le placer. L'application se créé puis le fichier main.dart s'ouvre. Nous pouvons maintenant démarrer le projet en utilisant la touche F5.
Pour ma part, une erreur de la sorte est apparue :

```
Error occurred during initialization of VM
Could not reserve enough space for 2097152KB object heap
```



3



4

```
* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to
 get more log output. Run with --scan to get full insights.
* Get more help at https://help.gradle.org
 Command: C:\git\Flutter\test_project\android\gradlew.bat app:properties
Please review your Gradle project setup in the android/ folder.
Exited (sigterm)
```

Si vous êtes familiers avec Android, vous reconnaitrez l'erreur lié au heap size. Pour corriger cela, il faut aller dans le fichier android/gradle.properties et modifier la valeur de org.gradle.jvmargs. La valeur à renseigner dépend de la mémoire de votre machine. Dans mon cas voici la modification que j'ai du faire

```
org.gradle.jvmargs=-Xmx1024m
```

TADA ! Après cette installation plus ou moins longue et plus ou moins fastidieuse, votre environnement est prêt et vous pouvez *enfin* commencer à jouer avec Flutter !

L'installation Flutter pour macOS

Comme expliqué dans la section précédente, pour développer des applications Flutter pour iOS, il est obligatoire de posséder un mac et de développer sur celui-ci.

Configuration requise

- Système d'exploitation : macOS (64-bit)
- Espace disque : 700 MB (sans inclure l'espace pour l'IDE/tools).
- Outils :

- bash	- git 2.x	- rm	- which
- curl	- mkdir	- unzip	

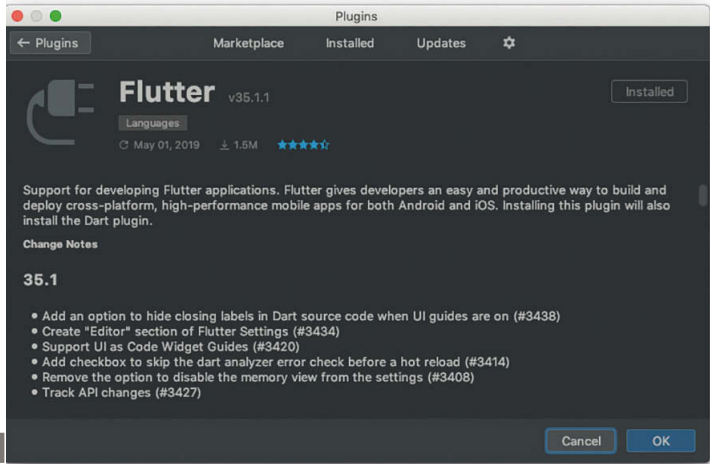
SDK Flutter

Si la confirmation de votre Mac est corforme, téléchargez le SDK Flutter. Une fois le SDK téléchargé, il faut extraire (si cela n'est pas automatique) l'archive dans le dossier choisi (par exemple : ~/development). Si l'extraction est automatique, copier le dossier dans le dossier cible. Ajoutons maintenant Flutter aux variables d'environnement. Pour cela, allons dans le terminal et plaçons-nous dans le dossier contenant le dossier flutter à l'aide de la commande cd ~/<votre-chemin> (cs ~/development dans mon cas).
Créons ou modifions le fichier $HOME/.bash_profile à l'aide de la commande vi $HOME/.bash_profile (j'ai choisi vi mais vous pouvez utiliser l'éditeur de votre choix). Ajoutons la ligne export PATH="$PATH:~/development/flutter/bin" (où ~/development est le dossier où vous avez placé flutter lors de l'étape précédente. Si

vous avez choisi un emplacement différent, remplacez cette partie var votre chemin). Ensuite, utilisons la commande source $HOME/.bash_profile pour mette à jour le terminal avec nos modifications. Nous pouvons maintenant utilser la commande flutter dans notre terminal. De la même manière que pour l'installation pour Windows, nous allons désormais vérifier notre installation à l'aide de la commande **flutter doctor**. Cette commande vérifie votre environnement et affiche un rapport avec le statut de l'installation de Flutter. Voici donc mon output :

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, v1.2.1, on Mac OS X 10.14.3 18D109, locale fr-FR)
[✗] Android toolchain - develop for Android devices
  ✗ Unable to locate Android SDK.
    Install Android Studio from: https://developer.android.com/studio/index.html
    On first launch it will assist you in installing the Android SDK components.
    (or visit https://flutter.io/setup/#android-setup for detailed instructions).
    If Android SDK has been installed to a custom location, set ANDROID_HOME to that location.
    You may also want to add it to your PATH environment variable.

[!] iOS toolchain - develop for iOS devices (Xcode 10.2.1)
  ✗ libimobiledevice and ideviceinstaller are not installed. To install with Brew, run:
    brew update
    brew install --HEAD usbmuxd
    brew link usbmuxd
    brew install --HEAD libimobiledevice
    brew install ideviceinstaller
  ✗ ios-deploy not installed. To install:
    brew install ios-deploy
  ✗ Brew can be used to install tools for iOS device development.
    Download brew at https://brew.sh/.
[!] Android Studio (version 3.4)
  ✗ Flutter plugin not installed; this adds Flutter specific functionality.
  ✗ Dart plugin not installed; this adds Dart specific functionality.
[✓] Connected device (1 available)

! Doctor found issues in 3 categories.
```

5



6

Flutter est bien installé mais il me manque le SDK Android, Android Studio, et les outils pour déployer sur des devices physiques iOS. Procédons maintenant à la configuration de la partie iOS.

Installation iOS

Flutter a besoin de Xcode version 9 ou plus récent. Si vous ne l'avez pas ou si vous avez une ancienne version, commencez par le télécharger et le mettre à jour.
Maintenant que nous avons une version de Xcode à jour, configurons les outils de ligne de commande pour utiliser cette version. Pour cela, utilisons la commande suivante :

```
sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
```

Le chemin peut être différent en fonction de la version que vous utilisez, si c'est le cas, modifiez-le en fonction de votre chemin vers Xcode. Vérifions maintenant que la licence Xcode a bien été signée, et sinon signons-la sinon, après avoir tout lu, *bien évidemment* :

```
sudo xcodebuild -license
```

Pour pouvoir déployer sur des devices iOS, il nous faut quelques outils supplémentaires. Commençons par installer Homebrew avec la commande suivante :

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Puis brew update pour s'assurer qu'il est bien à jour. "*Already up-to-date.*" Pile ce qu'on attendait.
Passons donc à l'installation des outils manquants avec les commandes indiquées par flutter doctor un peu plus tôt.

```
brew install --HEAD usbmuxd
brew link usbmuxd
brew install --HEAD libimobiledevice
brew install ideviceinstaller ios-deploy cocoapods
pod setup
```

Si une de ces commandes échoue, il est recommendé d'exécuter la commande brew doctor et de suivre les instructions.

Installation Android

Flutter dépend d'Android Studio pour récupérer les dépendances d'Android. Dans cette étape, nous allons donc commencer par installer Android Studio. Démarrons-le et suivons les étapes de l'installation. 5
Installons ensuite les plugins Flutter et Dart pour Android Studio dans le menu Configure > Plugins. 6
Maintenant que tout est installé, exécutons à nouveau la commande flutter doctor. Voici ce qu'il me dit :

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, v1.2.1, on Mac OS X 10.14.3 18D109, locale fr-FR)
[!] Android toolchain - develop for Android devices (Android SDK version 28.0.3)
  ✗ Android licenses not accepted.  To resolve this, run: flutter doctor --android-licenses
[!] iOS toolchain - develop for iOS devices
  ✗ Xcode installation is incomplete; a full installation is necessary for iOS development.
    Download at: https://developer.apple.com/xcode/download/
    Or install Xcode via the App Store.
    Once installed, run:
      sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
[✓] Android Studio (version 3.4)
[!] Connected device
  ! No devices available

! Doctor found issues in 3 categories.
```

Exécutons donc les commandes indiquées pour sélectionner la bonne application Xcode et accepter les licences Android. Une fois cela fait, appelons à nouveau le docteur

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, v1.2.1, on Mac OS X 10.14.3 18D109, locale fr-FR)
[✓] Android toolchain - develop for Android devices (Android SDK version 28.0.3)
[✓] iOS toolchain - develop for iOS devices (Xcode 10.2.1)
[✓] Android Studio (version 3.4)
[!] Connected device
  ! No devices available
```

Cette fois-ci, tout va bien :)
Nous sommes donc prêts à utiliser Flutter !

Première application

Quitte à avoir installé Android Studio et les extensions Flutter lors de l'étape précédente, nous allons l'utiliser pour créer notre première (ou pas) application Flutter.
Pour cela, démarrons Android Studio puis choisissons l'option "Start a new Flutter project". 7
Android Studio nous propose ensuite de choisir le type de projet que nous souhaitons créer. Choisissez donc le type de projet souhaité. Pour ma part, je choisis de créer une application donc je sélectionne "Flutter application".
Choisissons ensuite un nom de projet et le dossier dans lequel le placer. Si le SDK Flutter n'a pas été détecté, renseignez le chemin vers votre dossier Flutter (~/Documents/development/flutter dans mon cas).

Homepage et connexion

Pour commencer à prendre Flutter en main suite à l'installation, attaquons le développement en réalisant une première application contenant simplement un formulaire de connexion.

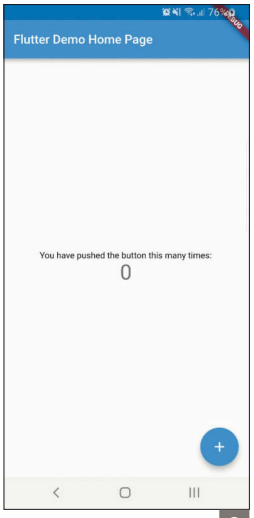
L'application

L'application de base générée par la commande Flutter : New project n'est pas l'habituel *hello world* mais une application contenant un texte "You have pushed the button this many times: N" et un bouton. Le clic du bouton incrémente le nombre affiché dans le texte. 8 . Le fichier main.dart créé automatiquement lors de l'étape précédente est composé de 3 classes :

- MyApp
- MyHomePage
- _MyHomePageState

MyApp

```
1  class MyApp extends StatelessWidget {
2
3    @override
4    Widget build(BuildContext context) {
5      return MaterialApp(
6        title: 'Flutter Demo',
7        theme: ThemeData(
8          primarySwatch: Colors.blue,
9        ),
10       home: MyHomePage(title: 'Flutter Demo Home Page'),
11     );
12   }
13 }
```

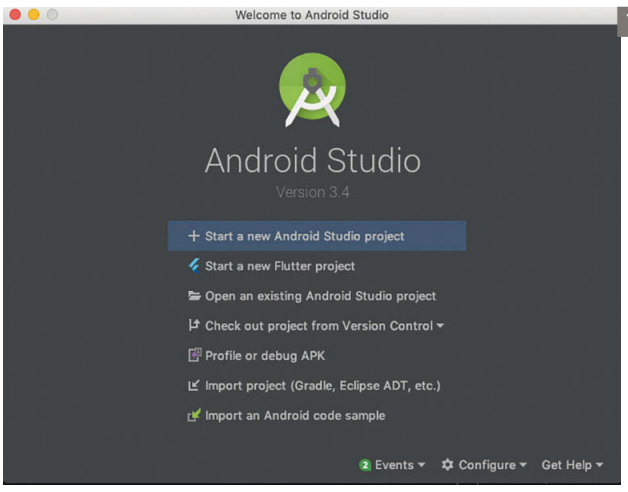


8

La class MyApp contient la définition de l'application. Ce widget *stateless* est le widget à la racine de l'application et, comme tout widget *stateless*, il contient la méthode build pour dessiner l'interface utilisateur représentant le widget. Ici, la méthode retourne simplement le widget MaterialApp afin d'avoir une application utilisant le *material design*. Nous verrons dans un article prochain comment personnaliser le thème de l'application. Ce qui nous intéresse principalement ici est la propriété home qui permet de définir quel Widget sera la page d'acueil de l'application. Ceci nous amène donc à la classe MyHomePage.

MyHomePage

```
1  class MyHomePage extends StatefulWidget {
2    MyHomePage({Key key, this.title}) : super(key: key);
3
4    // This class is the configuration for the state. It holds the values (in this
5    // case the title) provided by the parent (in this case the App widget) and
6    // used by the build method of the State. Fields in a Widget subclass are
7    // always marked "final".
8
9    final String title;
10
```



7

```
11  @override
12  _MyHomePageState createState() => _MyHomePageState();
13 }
```

La classe MyHomePage est le widget contenant la page d'accueil de l'application. Comme vous pouvez le voir, il s'agit d'un widget *stateful*. Si vous vous souvenez bien, un widget *stateful* est dynamique, il écoute les modifications des propriétés et met à jour l'interface en fonction de ces changements. À la différence du widget *stateless*, ce n'est pas le widget qui implémente la méthode de build mais son état (state). Il possède donc la méthode createState() qui permet d'indiquer au widget quel classe sera l'état du widget. Ici, l'état du widget est géré par la classe _MyHomePageState.

_MyHomePageState

```
1  class _MyHomePageState extends State<MyHomePage> {
2    int _counter = 0;
3
4    void _incrementCounter() {
5     setState(() {
6       _counter++;
7     });
8    }
9
10   @override
11   Widget build(BuildContext context) {
12    return Scaffold(
13     appBar: AppBar(
14      // Here we take the value from the MyHomePage object that was created by
15      // the App.build method, and use it to set our appbar title.
16      title: Text(widget.title),
17     ),
18     body: Center(
19      child: Column(
20       mainAxisAlignment: MainAxisAlignment.center,
21       children: <Widget>[
22        Text(
23         'You have pushed the button this many times:',
24        ),
25        Text(
26         '$_counter',
27         style: Theme.of(context).textTheme.display1,
28        ),
29       ],
30      ),
31     ),
32     floatingActionButton: FloatingActionButton(
33      onPressed: _incrementCounter,
34      tooltip: 'Increment',
35      child: Icon(Icons.add),
36     ),
37    );
38   }
39 }
```

La classe _MyHomePageState est l'état du widget MyHomePage. C'est donc ici que la méthode build est implémentée. Comme nous l'avons vu dans la classe MyApp, la méthode build dessine l'interface utilisateur. Sauf qu'ici, comme nous sommes dans un widget *stateful*, à la différence de la méthode MyApp.build qui n'est appelée qu'une seule fois, celle-ci sera appelé à chaque fois d'une propriété de l'état changera.

De base, l'état _MyHomePageState contient une propriété privée counter. Ce nombre est incrémenté à l'aide de la méthode _incrementCounter(). Comme vous pouvez le voir, cette méthode incrémente la variable counter à l'intérieur de l'appel à la méthode setState. C'est cette méthode qui indique à Flutter qu'une propriété de l'état a changé et qu'il faut donc redessiner l'interface. Sans l'appel à setState(), Flutter ne saurait pas qu'un changement a eu lieu et l'interface ne changerait pas.

Détaillons maitenant le contenu de la méthode build qui est plus complexe que dans la classe MyApp. Le principe de construction d'interface utilisateur avec Flutter est d'imbriquer des widgets dans des widgets. Ici le widget retourné par la méthode build est un Scaffold (littéralement *échafaudage*, (ou *échafaud*, mais c'est plus glauque)). Le scaffold est le widget qui implémente la structure de base d'un layout utilisant le *material design*. Il contient, ici, 3 widgets : appBar, body et floatingActionButton. 9

Le widget qui va nous intéresser est le body. Si vous souhaitez plus d'information sur l'appBar et le floatingActionButton, rendez-vous sur la documentation officielle ! Le widget du body est ici composé d'un widget Center, lui-même composé d'un widget Column, lui-même composé deux deux widgets Text. 10

Le widget Center contient un seul widget enfant (child) et le positionne au milieu du widget parent. En Flutter, tout est un widget, donc ici pour center un contenu, le widget Center est utilisé.

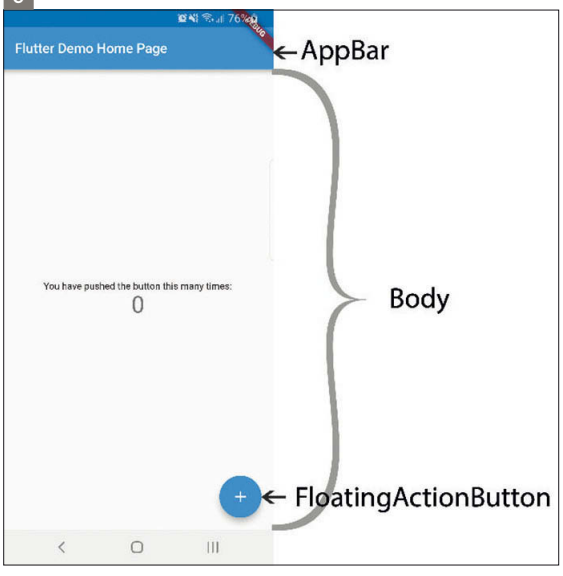
Ici, son widget enfant est un widget Column. Ce widget contient une liste de widgets enfants (children) et les places les uns à la suite des autres dans une même colonne (d'où le nom). Tous les widgets de la liste seront donc alignés verticalement.

Formulaire de connexion

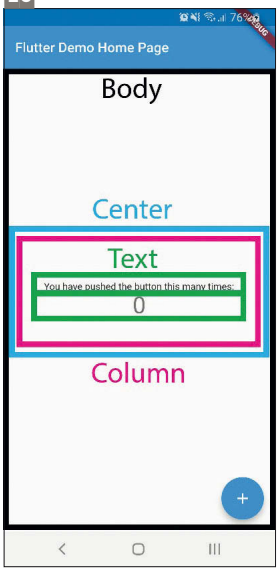
Maintenant que nous avons vu en détail la construction de la page, modifions-la pour obtenir un formulaire de connexion comme ceci : 11

Si vous avez bien compris, nos modifications vont donc se porter sur

9



10



la classe _MyHomePageStateet plus précisément dans la méthode build. Sur cette page de connexion, le floating action button n'a plus intéret, nous pouvons donc le supprimer ainsi que la propriété _counter et la méthode _incrementCounter auxquelles il est lié. Nous pouvons également supprimer tout le code qui compose le body puisque nous allons le réécrire intégralement. De quels widgets avons-nous besoin pour écrire ce formulaire ? Fouillons dans la liste des widgets et voyons lesquels nous seront utiles.

- Form pour définir le formulaire
- TextFormField pour les entrées texte
- RaisedButton pour valider le formulaire

Le widget Form a besoin d'une clé pour être identifié de manière unique. Ajoutons donc la propriété ci-dessous à notre classe _MyHomePageState.

```
// Create a global key that will uniquely identify the Form widget
final formKey = GlobalKey<FormState>();
```

Nous aurons également besoin de *widget de layout* tel que Column, et Padding pour rendre notre formulaire *légèrement* plus joli. Voici donc à quoi doit ressembler notre body après l'ajout de ces widgets.

```
body: new Form(
  key: formKey,
  child: Padding(
    padding: new EdgeInsets.all(20),
    child: new Column(
      children: <Widget>[
        new TextFormField(
        controller: emailController,
        keyboardType:
          TextInputType.emailAddress, // Use email input type for emails.
        decoration: new InputDecoration(
          hintText: 'me@example.com', labelText: 'Adresse email'),
        ),
        new TextFormField(
        obscureText: true, // Use secure text for passwords.
        decoration:
          new InputDecoration(hintText: '***', labelText: 'Mot de passe'),
        ),
        new Container(
          child: new RaisedButton(
            child: new Text('Connexion'),
            onPressed: logIn,
          ),
          margin: new EdgeInsets.only(top: 20.0),
        ),
      ],
    ),
  ),
),
```

Comme vous pouvez le constater, le widget TextFormField peut se composer de plusieurs propriétés :

- controller : pour récupérer le texte entré
- keyboardType : pour définir le type de clavier (utile pour les adresses email)
- decoration : pour personnaliser les inputs avec des hint et des label
- obscureText : pour masquer le texte (utile pour les mots de passe)

12

Notre formulaire pourrait être terminé à ce point, mais souvent dans les formulaires de connexion il est intéressant d'avoir des **validateurs** afin de vérifier que les champs sont remplis, que l'adresse mail est valide, ... Ici, j'ai bien envie de rajouter ces validateurs, vous aussi ? Let's go then !

La validation de l'adresse mail se fait à l'aide d'une regex, nous allons donc ajouter une propriété contenant la chaine de caractères de la regex.

```
final String emailPattern =
    r'^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([
```

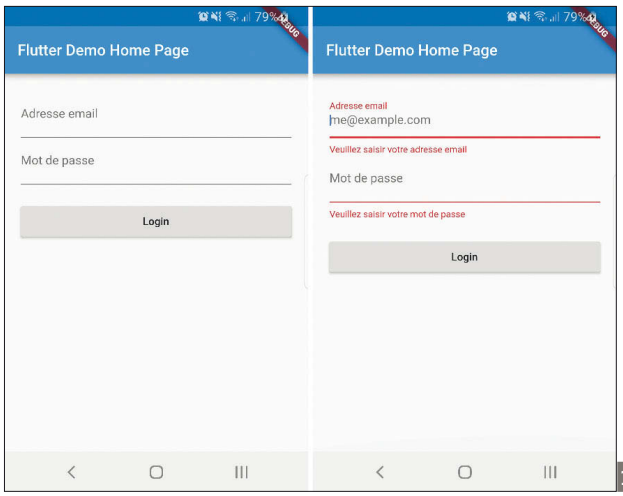
Ensuite, renseignons le champs validator des widgets TextFormField pour ajouter nos validateurs :

```
validator: (value) {
  if (value.isEmpty) {
  return 'Veuillez saisir votre adresse email';
  }

  RegExp regExp = new RegExp(emailPattern);
  if (!regExp.hasMatch(value)) {
  return "Veuillez saisir une adresse email valide.";
  }
},
```

Et voilà, nous avons réussi à créer un formulaire de connexion avec validateurs tout simplement. Vous pourrez retrouver l'intégralité du code source de la micro-application sur https://github.com/CamilleLL/FlutterLoginDemo

La navigation dans Flutter

Il est rare que les applications mobiles soient composées d'une seule page. Pour ajouter d'autres écrans il faut donc pouvoir naviguer entre ces écrans. Il existe deux façons de naviguer vers une page (ou route) en Flutter :



11

12

- en utilisant des routes nommées
- en créant des routes à la volée

Navigation par les routes nommées

Il est fréquent dans les applications mobiles d'avoir plusieurs façons d'arriver à une même page. Ainsi, le code pour naviguer vers cette page peut être redondant. C'est pour cela que nous allons commencer par parler de l'utilisation des routes nommées. Dans notre classe MyApp (main.dart), nous avions défini précédemment que nous allions créer une application de type MaterialApp.

La classe MaterialApp possède une propriété routes de type Map<String, WidgetBuilder>. C'est ici que nous allons définir les routes que nous allons utiliser dans l'application. Pour cela, il suffit d'ajouter des éléments dans la liste. La clé est la route que nous allons utiliser pour accéder à notre page et la valeur est notre widget contenant notre page cible. De cette manière, la façon de naviguer vers notre seconde page n'est définie qu'une seule fois, dans la classe MyApp.

Nous pouvons également définir la propriété initialRoute qui sera, comme son nom l'indique, la page initiale affichée par l'application.

```
return MaterialApp(
    title: 'Flutter Demo',
    theme: theme,
    routes: {
      '/': (context) => MyHomePage(title: "Connexion"),
      '/secondPage': (context) => MySecondPage(title: "Deuxième page"),
    },
    initialRoute: '/');
```

Une fois nos routes définies, il suffit d'appeler la méthode suivante pour naviguer vers notre seconde page

```
Navigator.pushNamed(context, '/secondPage');
```

Puis la méthode suivante pour revenir à la page précédante :

```
Navigator.pop(context)
```

Navigation en créant des routes à la volée

Au lieu d'utiliser les routes nommées avec la méthode Navigator .pushNamed, il est possible d'utiliser la méthode Navigator.push pour créer les routes à la volée. Comme expliqué précédemment cette méthode peut créer de la redondance de code si notre application possède plusieurs façons de naviguer vers une même page. Mais si votre application est simple, si vous n'avez pas plusieurs façons d'arriver à une même page, ou n'importe quelle autre raison que vous trouverez, vous pouvez utiliser cette seconde méthode pour naviguer. Cette méthode est tout aussi simple, car il suffit d'appeler la méthode suivante pour naviguer vers notre seconde page. Celle-ci prend deux paramètres : le context et la nouvelle route (de type MaterialPageRoute) vers notre seconde page.

```
Navigator.push(context,
            MaterialPageRoute<void>(
              builder:(BuildContext context) {
                return MySecondPage(title: "Deuxième page");
              }));
```

Utiliser la classeMaterialPageRoute permet de naviguer vers la nouvelle page en utilisant les animations de navigation propres à la plateforme cible. Puis pour revenir à la page précédente, nous utilisons toujours la méthode suivante :

```
Navigator.pop(context)
```

Voila, que ce soit à l'aide de la première méthode ou de la seconde, nous sommes maintenant capables de naviguer entre deux pages. Mais mais mais ce n'est pas suffisant ! Ah bon ? Eh oui, il est souvent nécessaire de faire passer des données d'une page à l'autre. Nous allons donc maintenant voir comment...

Passer des données d'une page à l'autre

Les fins observateurs que vous êtes avez sans doute remarqué que lorsque je navigue vers ma page MySecondPage, je passe un paramètre title. Effectivement, pour passer des données à ma nouvelle page, j'utilise le constructeur de celle-ci :

```
class MySecondPage extends StatelessWidget {
  MySecondPage({@required this.title});
...
}
```

Avec les routes nommées

En utilisant la méthode des routes, pour passer des paramètres à notre seconde page au moment de naviguer (en dehors de la route définie dans la classe MyApp donc), nous pouvons utiliser la méthode suivante :

```
Navigator.pushNamed( context, '/secondPage',
  arguments: <String, String>{
    'title':"Deuxième page",
  },
);
```

Et notre route ressemblerait à ceci :

```
routes: {
  ...
  '/secondPage': (context) => MySecondPage(),
},
```

Sans les routes nommées

En utilisant la méthode sans les routes, pour passer des paramètres à notre seconde page nous utilisons la méthode vue précédemment en les passant dans le constructeur de la seconde page. Effectivement, j'avais déjà passé mon paramètre title à ce moment-là :

```
Navigator.push(context,
            MaterialPageRoute<void>(
              builder:(BuildContext context) {
                return MySecondPage(title: "Deuxième page");
              }));
```

TADA ! Nous pouvons maintenant naviguer d'une page à l'autre puis de l'autre à l'une et ainsi de suite.

Vous avez désormais une première vue d'ensemble de Flutter, vous avez vu comment l'installer et comment créer un tout premier projet pour le prendre en main. Il ne vous reste plus qu'à laisser libre cours à votre imagination ! N'hésitez pas à aller faire un petit tour sur le blog Infinite Square si vous voulez aller plus loin et voir comment ajouter un thème, un splashscreen, des polices personnalisées, etc, à votre application. •

Jérémy Dollé
24 ans, chef de projet développeur chez The Coding Machine Lyon depuis 2 ans.

Comment construire une application React Native sur des bases solides ?

React Native s'est rapidement positionné parmi les technologies leaders du développement d'applications mobiles. Accompagnée d'une grande communauté, cette technologie est en perpétuelle évolution. Après plusieurs années d'expérience, nous avons appris à bâtir une architecture robuste, scalable et performante.

niveau 200

Les technologies mobiles

Les technologies mobiles prennent une place importante dans le quotidien de chacun. En effet, elles ne concernent pas que les applications pour smartphones, mais aussi les applications des téléviseurs connectés, des montres connectées, des tablettes… A cause de la multiplicité des terminaux utilisés, de nombreux paramètres sont à prendre en compte (OS, tailles des écrans, encoches…). C'est pour cela que les technologies mobiles se doivent d'évoluer rapidement afin de trouver un équilibre entre facilité de développement, performance et coût de production.

Plusieurs technologies sont disponibles pour le développement mobile : le Natif et les technologies dites « hybrides ».

Lors du développement Natif, il est nécessaire de développer l'application en deux langages distincts selon l'OS (iOS ou Android), qui peuvent avoir des syntaxes très différentes (c'est le cas de Kotlin et de Swift par exemple). Les applications développées en Natif sont au plus près de la couche hardware, elles peuvent donc utiliser presque 100 % des capacités du téléphone ce qui les rend très performantes. En revanche, la nécessité de développer dans deux langages différents et donc de maintenir deux codes à la fois représente un coût de production important.

C'est pour résoudre ce problème qu'interviennent les technologies "hybrides". Ces dernières permettent, à partir d'un seul code source, de déployer l'application sur plusieurs plateformes (Play Store et Apple Store). Ces technologies sont basées sur l'encapsulation dans un navigateur en plein écran. Il s'agit donc d'une application web responsive qui tourne sur la couche webview du téléphone. Cette solution est donc aussi facile à développer qu'une application web traditionnelle avec un code source unique ce qui réduit considérablement le coût de production. Néanmoins, l'utilisation de la couche webview impacte la performance de l'application, car elle connaît des fuites de mémoire et utilise une faible partie de la couche hardware du téléphone.

De nouvelles technologies voient le jour, comme React Native ou Flutter. Ces dernières utilisent le même principe de code source unique, mais sans utiliser la couche intermédiaire webview. Elles sont alors bien plus performantes et offrent un rendu et une expérience très proches du Natif.

React Native

React Native est une librairie née en 2015 à la suite d'un hackathon. Reprise et développée ensuite par Facebook, elle permet de concevoir des applications mobiles Android et iOS à l'aide d'un seul langage. Basée sur ReactJs, React Native a la force de produire des solutions mobiles simili natives en JavaScript, tout en conservant une logique de composants. Grâce à cela, React Native s'impose comme une technologie robuste, fiable et abordable par tout développeur web.

Présentation de l'architecture

Nous avons conçu un Boilerplate (architecture de base permettant de bien démarrer tous types d'applications mobiles) réunissant un ensemble de bonnes pratiques et une architecture testée et éprouvée sur des cas réels. Cette architecture est basée sur la séparation des préoccupations (separation of concerns) et utilise un minimum de dépendances.

Différencier Composants et Containers

La notion de composant est très répandue dans le monde du web. Elle permet de décomposer le contenu d'un écran en blocs afin d'avoir un code plus lisible mais également de pouvoir réutiliser certains composants. On découpe généralement les composants en deux catégories : les composants UI (user interface) et les composants "métiers" ou containers. Les composants UI sont des éléments qui ont pour seul but de recevoir des propriétés et de les afficher d'une certaine manière. Ils doivent posséder un minimum d'intelligence (exemple : boutons, menu, inputs…).

A contrario, les containers peuvent être assimilés aux écrans de l'application. Il s'agit bien de composants mais leur fonction est différente. Les containers se chargent de récupérer des données, de les formater si besoin et de les redistribuer aux composants UI qui constituent l'écran.

Naviguer entre les pages

La navigation au sein d'une application mobile est très différente de celle d'une application web. En effet, il n'y a pas d'URL rattachée aux différents écrans. Il faut alors bâtir et organiser un arbre d'écrans qui représentera les liens entre eux et permettra la navigation. Pour réaliser cette tâche complexe, nous nous reposons sur React Navigation. Il s'agit d'une librairie JavaScript, performante et robuste qui construit les liens entre les containers pour permettre une navigation la plus intuitive possible.

Le State Management

Au cours du développement d'une application mobile, il est important de centraliser les données et les actions à l'aide d'un gestion-

naire de state global. En effet, pour traiter les données, les composants peuvent gérer un "state" qui est seulement local. Ils peuvent également s'échanger des données via des propriétés (props). Cependant, les props sont en lecture seule et l'application se transforme rapidement en "usine à gaz" à causes des multitudes de propriétés envoyées à travers chacun des composants. En plus de permettre une séparation des préoccupations efficace, cela permet de simplifier grandement le code et de le rendre bien plus robuste. Au sein du Boilerplate on retrouvera Redux qui fait partie des gestionnaires de state global les plus reconnus. Il permet de gérer un objet (global state) qui correspond à l'état de l'application. Il est impossible d'accéder ou de modifier cet objet directement sans passer par une action définie dans le store. Pour finir, afin de lier le state à l'action, il faut implémenter des reducers, qui sont des fonctions qui vont agir sur le global state.

Se connecter à des APIs tierces

Comme souvent dans le web, nous développons une API (RESTful par exemple) et un client front (en Js). Le client front interroge notre API à travers des requêtes HTTPs, il reçoit une réponse et l'affiche d'une certaine manière. Une application mobile est tout simplement un front, au même titre qu'une application en VueJs ou en ReactJs. Elle a besoin de s'appuyer sur une API externe pour agir sur la base de données. Pour intégrer les données renvoyées par ces APIs et les connecter au state Redux, nous avons organisé l'architecture suivante :

- Un dossier Services qui contient les appels API effectués avec Axios.
- Un dossier Store qui contient le state global
- Un dossier Saga qui contient les méthodes associées à chacune des actions.

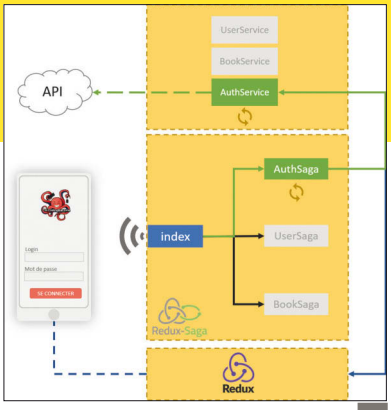
Dans le dossier Saga, un fichier index.js permet de capturer toutes les actions et les distribuer aux fonctions sagas adéquates. Elle va, par exemple, appeler le bon service, attendre le retour API et traiter la réponse en modifiant le state si besoin. Elle peut également provoquer une navigation. 1

Bien gérer son thème

Tout comme un projet web, et dans un souci de séparation des préoccupations, il est très important d'isoler le style des composants. Ainsi, elle est donc bien localisée et homogène au sein des applications. Le but est de créer le moins de fichiers possibles et d'obtenir des applications aux styles différents, toujours de la même façon et sans ajouter de style superflu.

Dans notre Boilerplate, la séparation est la suivante :

- Colors.js : ce fichier regroupe toutes les variables de couleur de l'application. Cela permet d'harmoniser et d'homogénéiser les écrans de l'application mais aussi de rendre leur modification plus rapide.
- Metrics.js : ce fichier regroupe les variables de taille standard pour l'application (par exemple les marges horizontales).
- Fonts.js : ce fichier contient des styles prédéfinis pour tout composant de type texte. (par exemple un style pour créer des « titre1 », des « titre2 » ou encore des paragraphes).
- Helpers.js : ce fichier est utile à tous les composants. Il regroupe plusieurs styles assez communs, notamment des styles pour gérer les alignements (par exemple créer un layout en flex qui centre verticalement et horizontalement ses enfants).



1

2

Cas concret

Nous allons à présent illustrer l'utilisation du Boilerplate pour implémenter un écran de connexion par session via une API tierce : regres.in. Dans cette partie de l'article, nous n'allons pas nous attarder sur le style de l'application mais sur la partie fonctionnelle.

Pour commencer, il faudra suivre la documentation du Boilerplate pour la création de votre nouvelle application via https://github.com/thecodingmachine/react-native-boilerplate.

Dans ce cas concret, nous allons concevoir un écran de connexion comportant un logo suivi de deux champs (un e-mail et un mot de passe) ainsi qu'un bouton qui nous permettra de soumettre le formulaire. 2

Connexion à l'api

La première chose à faire est de stocker l'URL de l'API tierce pour la connexion d'un utilisateur dans notre fichier de configuration comme suivant :

`App/Config/index.js`

```
export const Config = {
  LOGIN_API_URL: 'https://reqres.in/api',
}
```

Ensuite, nous allons créer le service permettant l'appel à l'API qui sera une instance Axios pointant vers l'URL stockée dans le fichier de configuration App/Config/index.js. Dans ce même service, nous allons exporter une fonction « login » qui va effectuer la requête et retourner le token si la connexion a abouti et null sinon. Ce service pourra donc être appelé dans notre application de la manière suivante :

AuthService.login('mon_login', 'mon_mot_de_passe')

`App/Services/AuthService.js`

```
import axios from 'axios'
import { Config } from 'App/Config'

const userApiClient = axios.create({
  baseURL: Config.LOGIN_API_URL,
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/json',
  },
  timeout: 3000,
})

function login(email, password) {
  return userApiClient
    .post('/login', { email, password })
    .then((response) => response.data.token)
    .catch((error) => null)
}

export const AuthService = {
  login,
}
```

Le State Management

Une fois notre service opérationnel, nous allons initialiser notre store Redux afin de stoker le résultat de l'appel API et ainsi pouvoir le réutiliser n'importe où dans notre application.
Pour ce faire, nous allons créer un dossier `App/Stores/Auth` qui comportera trois fichiers :
Un premier pour définir toutes les actions qui pourront être lancées depuis notre application. L'action « login » pour exécuter la phase de connexion. L'action « loginLoading » pour mettre l'application dans un état de chargement. L'action « loginSuccess » pour récupérer le token et le stocker si la connexion a réussi. L'action « loginFailure » pour stocker un message d'erreur dans le cas contraire.

`App/Stores/Auth/Actions.js`

```
import { createActions } from 'reduxsauce'

const { Types, Creators } = createActions({
  // Fetch login
  login: ['email', 'password'],
  // The operation has started and is loading
  loginLoading: null,
  // the token were successfully fetched
  loginSuccess: ['token'],
  // An error occurred
  loginFailure: ['error'],
})

export const AuthTypes = Types
export default Creators
```

Un deuxième fichier pour définir un state initial. Dans ce fichier, nous avons plusieurs clés disponibles (state.token, state.loginLoading, state.loginError) qui vont nous permettre de faire évoluer l'affichage de notre application. Par exemple, si « state.loginLoading » est à true, l'application est en état de chargement, nous allons donc afficher un composant de chargement. Si « state.loginError » contient la chaîne de caractères « empty inputs », nous l'afficherons pour identifier l'erreur qui s'est produite. Nous utiliserons ces états lorsque nous implémenterons le composant qui représentera l'écran de connexion.

`App/Stores/Auth/InitialState.js`

```
export const INITIAL_STATE = {
  token: null,
  loginLoading: false,
  loginError: null,
}
```

Pour finir, le fichier « Reducers.js » va faire office d'observateur et agir sur le state. Par exemple, si l'action « LOGIN_LOADING » (correspondant au « loginLoading » du fichier d'action) est lancée, alors la fonction « loginLoading » du fichier des reducers est exécutée. Le state est alors mis à jour de sorte que « state.loginLoading = true » et « state.loginError = null ». Ainsi, dans notre application, lors de la soumission du formulaire, nous allons indiquer à l'utilisateur qu'un chargement se produit et que son action a bien eu un effet.
`App/Stores/Auth/Reducers.js`

```
import { INITIAL_STATE } from './InitialState'
import { createReducer } from 'reduxsauce'
import { AuthTypes } from './Actions'

export const loginLoading = (state) => ({
  ...state,
  loginLoading: true,
  loginError: null,
})

export const loginSuccess = (state, { token }) => ({
  ...state,
  token: token,
  loginLoading: false,
  loginError: null,
})

export const loginFailure = (state, { error }) => ({
  ...state,
  token: null,
  loginLoading: false,
  loginError: error,
})

export const reducer = createReducer(INITIAL_STATE, {
  [AuthTypes.LOGIN_LOADING]: loginLoading,
  [AuthTypes.LOGIN_SUCCESS]: loginSuccess,
  [AuthTypes.LOGIN_FAILURE]: loginFailure,
})
```

Pour les actions plus complexes comme le login (qui effectue un appel API, met l'application en état de chargement et fait possiblement des redirections) nous n'allons pas nous servir d'un Reducer mais d'une Saga qui est une fonction génératrice nous permettant de mieux gérer la succession des actions à effectuer.

`App/Sagas/Auth/AuthSaga.js`

```
import { put, call } from 'redux-saga/effects'
import AuthActions from 'App/Stores/Auth/Actions'
import { AuthService } from 'App/Services/AuthService'

export function* login({ email, password }) {
  yield put(AuthActions.loginLoading()) // call the loginLoading action

  // Fetch user information from an API
  const response = yield call(AuthService.login, email, password)
  if (response) {
    yield put(AuthActions.loginSuccess(response)) // call the loginSuccess action
  } else {
    yield put(AuthActions.loginFailure('email or password incorrect.'))
    // call the loginFailure action
  }
}
```

Ici l'action « loginLoading » a pour but de mettre le state « loginLoading » à true. Nous pourrons ainsi nous en servir dans la vue pour indiquer à l'utilisateur qu'il se produit un chargement de données. Ensuite, nous faisons appel au service « AuthService ». Si nous avons une réponse positive, alors l'appel a bien fonctionné et nous exécutons l'action « loginSuccess(token) » qui a pour but de stoker le token dans le state. Sinon, nous exécutons l'action « loginFailure » qui a pour but de stoker dans le state un message d'erreur qui pourra ainsi être affiché à l'utilisateur. Pour que cette saga puisse être lancée, il faut l'ajouter au fichier suivant :

`App/Sagas/index.js`

```
import { takeLatest, all } from 'redux-saga/effects'
import { AuthTypes } from 'App/Stores/Auth/Actions'

import { login } from './Auth/AuthSaga'

export default function* root() {
  yield all([
    takeLatest(AuthTypes.LOGIN, login), //add this line
  ])
}
```

Ce fichier permet de créer un observateur sur l'action « LOGIN ». À chaque fois qu'elle est lancée depuis une vue de l'application, alors la saga « login » est exécutée aussi.
Il ne reste plus qu'à ajouter le reducer au store Redux de la manière suivante :

`App/Stores/index.js`

```
import { combineReducers } from 'redux'
import configureStore from './CreateStore'
import rootSaga from 'App/Sagas'
import { reducer as AuthReducer } from './Auth/Reducers'

export default () => {
  const rootReducer = combineReducers({
    auth: AuthReducer,
  })

  return configureStore(rootReducer, rootSaga)
}
```

Nous avons terminé la partie appel API et implémentation du store. Maintenant nous allons créer tous les composants dont nous avons besoin. Nous allons brancher l'action « LOGIN » sur le bouton de soumission du formulaire de connexion. Celle-ci va donc être captée par le fichier précédent et transmit à la saga. Dans la saga, le state « state.loginLoading » sera initialisé et nous pourrons alors afficher à l'utilisateur un composant de chargement, l'appel API sera ensuite exécuté grâce au Service « AuthService.login » et le token sera stocké dans le state et donc affiché à l'écran si la connexion a réussi, un message d'erreur sinon.

Intégration de l'écran

Notre écran sera symbolisé par le container `AuthScreen` composé de trois composants `BrandComponent`, `Input` et `Button`.
Premièrement, nous allons créer le composant pour l'affichage du logo afin de le rendre réutilisable. Pour se faire, il va posséder une propriété « height » qui permettra à ce dernier d'être affiché à des tailles différentes. Nous allons également lui donner une hauteur par défaut qui sera de 100.

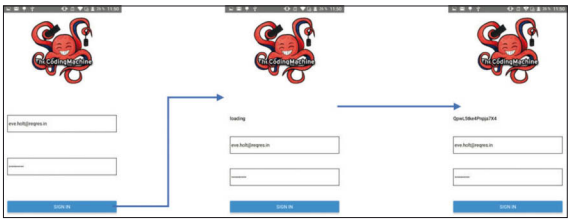
`App/Components/Brand.js`

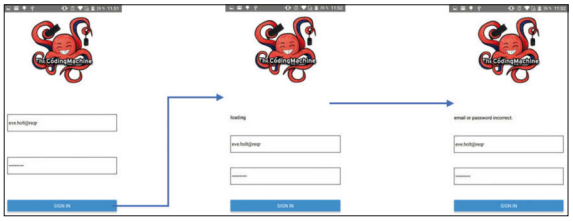
```
import React from 'react'
import { View, Image } from 'react-native'
import { Helpers, Images } from 'App/Theme'

const Brand = (props) => (
  <View style={[Helpers.fullWidth, { height: props.height || 100 }]}>
    <Image style={Helpers.fullSize} source={Images.logo} resizeMode={'contain'} />
  </View>
)

export default Brand
```

Les composant 'Input' et 'Button' sont des composants qui sont mis à disposition par la librairie React Native. Il est possible de les customiser mais ce n'est pas le sujet de cet article. Nous allons donc simplement les importer et les utiliser tels quels.
Après avoir créé notre composant 'Brand', nous allons implémenter notre écran de connexion comme suit : **Code complet sur programmez.com & github**

`App/Containers/Auth/AuthScreen.js`

Afin d'utiliser les variables et les actions présentes dans le store au sein de notre container, nous allons les affecter aux props de ce dernier grâce aux méthodes : « mapStateToProps », « mapDispatchToProps » et « connect ». Ainsi, l'action « login » et les states « token », « loginLoading » et « loginError » seront accessibles depuis l'objet « this.props ». Nous avons donc, au clic sur le bouton, appliquée l'action « login », qui va mettre la variable « loginLoading » à true (grâce à la fonction « login » du fichier « AuthSaga ») et nous permettre d'afficher un loader. Une fois l'appel API réussi, la variable « token » sera affectée et affichée à l'écran. Dans le cas où l'appel API provoque une erreur, cette dernière est affichée à l'écran.





Pour finir, il suffira de remplir le navigateur avec le nouvel écran. Pour faire simple, nous allons faire en sorte d'ouvrir notre application sur cet écran de connexion de la façon suivante :

`App/Navigators/AppNavigator.js`

```
import { createAppContainer, createStackNavigator } from 'react-navigation'
//...
import AuthScreen from 'App/Containers/Auth/AuthScreen' // add this line

const StackNavigator = createStackNavigator(
  {
    //...
    AuthScreen: AuthScreen, // add this line
  },
  {
    initialRouteName: 'SplashScreen',
  }
)

export default createAppContainer(StackNavigator)
```

Et modifier `App/Sagas/StartupSaga.js` pour rediriger vers la nouvelle page à l'ouverture de l'application.

```
//...
export function* startup() {
  // ...
  NavigationService.navigateAndReset('AuthScreen')
}
```

Nous avons ainsi un écran de connexion totalement opérationnel en quelques minutes !

Conclusion

React Native est un langage intuitif pour tout développeur web. Cette technologie reprend les fondamentaux du web grâce notamment au JavaScript. Il s'agit ici d'une architecture testée et éprouvée sur diverses applications (complexes ou simples). Il existe bien d'autres Boilerplates. Néanmoins, l'architecture présentée ici permet de créer de petites comme de grandes applications. Ce projet contient peu de dépendances toutes nécessaires et n'est rattaché à aucune autre technologie. Ceci permet d'être plus réactif en ce qui concerne les mises à jour. Ce Boilerplate est également accompagné d'une documentation dense qui permet d'aider et d'initier au maximum les bonnes pratiques chez ses utilisateurs.
Lien du projet : https://github.com/thecodingmachine/react-native-boilerplate

Lætitia Avrot *est experte PostgreSQL chez EnterpriseDB. Elle est co-fondatrice du mouvement Postgres Women, membre du Postgres Advocacy Group et membre du Postgres Funds Group. Elle a aussi été pendant un an membre du comité du Code de Conduite de la communauté PostgreSQL.Elle a écrit plusieurs patchs pour le projet PostgreSQL et donne régulièrement des conférences dans des événements communautaires.*

Partie 1

Le SQL avancé pour de vraies performances

Le SQL est souvent sous-estimé par les développeurs car malheureusement peu enseigné dans le supérieur, que ce soit en France ou à l'étranger... Dans cet article, nous allons découvrir trois fonctionnalités très avancées du SQL : les jointures latérales, les window fonctions et les CTE récursives. Les performances de ces 3 fonctionnalités seront comparées aux performances d'un code équivalent en python.

niveau 200

Le SQL est un langage

Avant toute chose, commençons par le commencement : le SQL est un langage de programmation. C'est un langage que l'on classe dans la catégorie des langages déclaratifs (comme le Prolog par exemple). De plus, il est Turing complete (voir [1]). Cela signifie que si l'on est suffisamment fou, il est possible de résoudre tous les problèmes de programmation en SQL.

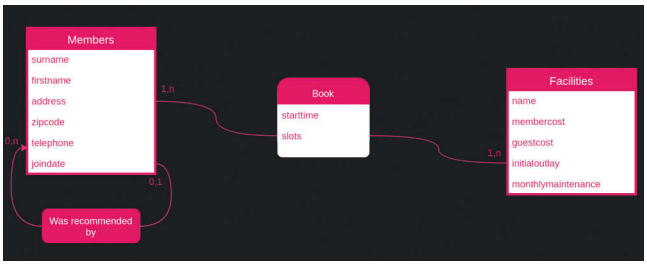
Depuis que j'enseigne en Master 2 à l'université Lyon 1, je pose régulièrement cette question à mes étudiants : "Qui aime écrire des requêtes SQL ?". Généralement, le nombre de mains levées est assez bas, de l'ordre de 5% de l'audience. Je leur demande alors "Qui aime déboguer les requêtes SQL écrites par d'autres ?". C'est alors assez drôle de voir toutes les mains se baisser. Cette situation est, selon moi, symptomatique d'un problème très important dans le SQL : comme il n'est pas considéré comme du code, il n'est pas écrit de manière lisible. Voici donc les règles d'or à appliquer au SQL pour faciliter son écriture et sa maintenabilité :

- Une seule chose par ligne (Dès qu'on passe à l'écriture de l'item suivant, on ajoute un retour chariot)
- Indentation (On indente correctement les différentes lignes pour faire apparaître les différentes clauses de la requêtes)
- Nommage des choses avec des noms appropriés et significatifs (pour les variables et les alias, que ce soit pour les colonnes ou pour les relations)
- Commentaires (Pensez-vous qu'il soit raisonnable de fournir une requête de 80 lignes sans un seul commentaire ?)
- Test (unitaires et de non régression)

Vous pouvez en revanche abandonner l'écriture des mots-clés en majuscules. Cette pratique était utile dans les années 80, mais depuis l'invention de la coloration syntaxique, elle est devenue totalement superflue. De plus, il a été prouvé que nous lisions moins bien les mots en majuscules car nous lisons plus souvent en minuscules.

Le modèle de données utilisé

Pour le reste de cet article, nous utiliserons le même modèle de données, ce qui permettra de ne pas avoir à le réexpliquer pour chaque nouvelle fonctionnalité évoquée. Comme je suis informaticienne, je ne vais pas réinventer la roue. De très nombreux bases de données pédagogiques existent, je vais utiliser celle de l'excellent site **pgexercises.com**. Si vous ne connaissez pas ce site, je vous encourage à aller y faire un tour. Si vous ne connaissez pas le SQL, commencez par les premiers exercices. Si vous pensez déjà tout connaître du SQL, regardez les derniers exercices. 1



1 *Le modèle de données de pgexercises.com*

Le modèle est très simple : 2 tables de données permettant d'enregistrer les réservations d'équipements dans un country club. Nous avons donc une table des membres du club et une table des équipements. Comme nous avons une relation many-to-many pour gérer les réservations, nous avons une table d'association des réservations avec des informations supplémentaires spécifiques à cette réservation (quand l'équipement est réservé et pour combien de temps).

Les jointures latérales

Les jointures latérales sont un outil très précieux pour effectuer une sorte de boucle sur chaque ligne du résultat. Elles répondent au besoin "pour chaque ligne, je voudrais…". Elles sont extrêmement efficaces mais ont l'inconvénient d'être très addictives. Quand on les maîtrise, on a tendance à vouloir en mettre partout, même lorsque ce n'est pas nécessaire. Or, comme le veut le principe KISS (Keep It Simple and Stupide = Garde le Simple et Stupide), ajouter une jointure latérale là où ce n'est pas nécessaire posera des problèmes en production, que ce soit en termes de performances ou en termes de maintenabilité.

Syntaxe

La syntaxe d'une jointure latérale est très simple : il suffit d'ajouter le mot clé 'lateral' derrière le 'join'. La condition de jointure se fera dans la sous-requête concernée par la jointure latérale, ce qui fait que par la suite, la clause 'on' de la jointure sera simplement 'on true'.

```
select ...
from table1
 join lateral (
    select ...
    from table2
    where table1.colonne1 = table2.colonne2)
 on true
where ...
```

[1] https://wiki.postgresql.org/wiki/Turing_Machine_(with_recursive)

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

[Programmez!] Le magazine des développeurs

Nos classiques

1 an 11 numéros	49€*
2 ans 22 numéros	79€*
Etudiant 1 an - 11 numéros	39€*

** Tarifs France métropolitaine*

Abonnement numérique

PDF 1 an - 11 numéros	35€
Option : accès aux archives	10€

Souscription uniquement sur
www.programmez.com

Offres 2020

1 an ..... 49€
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 ..... 79€
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 €)

eni

** 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 :** 49 €
11 numéros + 1 vidéo ENI au choix :
☐ **Abonnement 2 ans :** 79 €
22 numéros + 1 vidéo ENI au choix :

☐ Vidéo : Symfony 3
☐ Vidéo : Raspberry Pi

PROG 236
Valable jusqu'au 31 janvier 2020

☐ 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**

NOUVEAU ! Boutique Programmez!

Les anciens numéros disponibles

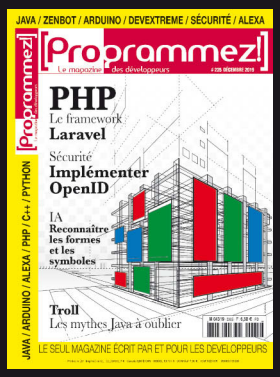
226

234

233

229

228



235

tarif unitaire 6,5 € (frais postaux inclus)

Technosaures n°1

7,66 €
(frais postaux inclus)

Histoire de la micro-informatique 1973-2007

12,99 €
(frais postaux inclus)

☐ 226 : ___ ex ☐ 233 : ___ ex
☐ 228 : ___ ex ☐ 234 : ___ ex
☐ 229 : ___ ex ☐ 235 : ___ ex

☐ Technosaures N°1 7,66 €
☐ Histoire de la Micro-informatique 12,99 €

soit ___ exemplaires x 6,50 € = _____ € _____ €

Commande à envoyer à :
Programmez!
57 rue de Gisors - 95300 Pontoise

soit au **TOTAL** = _____ €

☐ M. ☐ Mme ☐ Mlle Entreprise : _____ Fonction : _____

Prénom : _____ Nom : _____

Adresse : _____

Code postal : _____ Ville : _____

E-mail : _____ @ _____

Règlement par chèque à l'ordre de Programmez ! I Disponible sur www.programmez.com

Offres pouvant s'arrêter à tout moment, sans préavis PROG 236

Exemple

Prenons donc l'exemple suivant : notre country club fonctionne sur recommandation. Pour chaque membre du club, nous avons accès à l'id du membre du club qui l'a recommandé. Malheureusement, l'id n'est pas l'information que nous souhaitons obtenir. Ce numéro interne n'a aucune importance. Il n'a été implémenté que pour des raisons techniques (surtout pour des raisons de performance). Ce que nous aimerions trouver dans notre résultat, c'est la liste des colonnes suivantes :

- Le nom du membre du club
- Le prénom du membre du club
- Le nom de la personne qui l'a recommandé
- Le prénom de la personne qui l'a recommandé

On sera très tenté d'écrire la requête sous cette forme : 2

```
select members.nom,
       members.prenom,
       (
         select nom as nom du parrain,
                prenom as prenom du parrain
         from members as parrain
         where parrain.id = members.recommended by
       )
from members
```

2

Malheureusement, si on tente ce genre de requête, on obtiendra une erreur de syntaxe. Une sous-requête dans la clause from ne peut être que scalaire (une seule ligne, une seule colonne).
Voici comment écrire cette requête en SQL:

```
select
 mem.firstname,
 mem.surname,
 memref.firstname,
 memref.surname

from cd.members as mem
 /* La jointure latérale permet de récupérer le parrain pour chaque membre
   du club*/
 left join lateral
    (
     select
      firstname,
      surname
     from cd.members as mentors
     where
      mentors.memid = mem.recommendedby
    ) as memref
 on true
```

Voyons comment écrire cette requête en Python. L'algorithme est simple :

- Récupérer la liste des membres du club avec l'id de leur parrain
- Pour chaque personne récupérer les informations sur le parrain
- Afficher ces informations

```
#! /usr/bin/env python3

import sys
import psycopg2

CONNSTRING = "service=pgexercises"

def get_member_names(id):
 #Fetch the recommender's name and last name from his id
 sql = """
    select
     firstname,
     surname
    from cd.members
    where memid = %s
 """
 pgconn = psycopg2.connect(CONNSTRING)
 cursor = pgconn.cursor()
 cursor.execute(sql, (id,))

 # Id is a primary key -> only one result expected
 (firstname, surname) = cursor.fetchone()

 result = '{0}, {1}'.format(firstname, surname)

 return result


def list_members_and_recommenders():
 "Fetch the list of members"

 sql = """
    select
     mem.firstname,
     mem.surname,
     mem.recommendedby
    from cd.members as mem
 """

 pgconn = psycopg2.connect(CONNSTRING)
 cursor = pgconn.cursor()
 cursor.execute(sql)

 for (firstname, surname, recommendedby) in cursor.fetchall():

    if not recommendedby:
     print('{0}, {1}, , '.format(firstname, surname))
    else:
     mentor = get_member_names(recommendedby)
     print('{0}, {1}, {2}'.format(firstname, surname, mentor))

if __name__ == '__main__':
 list_members_and_recommenders()
```

Comparons maintenant les temps d'exécution de ces 2 programmes différents. Pour avoir le temps d'exécution d'une requête, on peut utiliser \timing on sous psql. Pour afficher le temps d'exécution d'un programme python, j'ai choisi d'utiliser la commande time.
Il faut garder à l'esprit que les temps fournis ne sont pas importants en tant que tel (sur un autre système, ces temps seraient certainement différents). Ce qui est important, c'est la comparaison de ces

temps. Il faut également garder à l'esprit que la base exemple que nous utilisons est très réduite en termes de volume (la table des membres du club ne comporte que 31 lignes et la totalité de la base occupe à peine 9 Mo sur le disque). Sur une vraie production, les écarts de performances seront donc pires.
La requête qui contient la jointure latérale s'exécute à peu près en 0.400 ms alors que mon programme python met à peu près 200 ms. Nous avons donc une requête SQL qui est de l'ordre de 500 fois plus rapide que le programme python qui fait la même chose. Une explication de ce phénomène se trouve dans les très nombreux aller-retours entre le programme python et la base de données. Les connexions ne sont pas gratuites et dans notre cas, pour chaque ligne, nous allons refaire une connexion pour obtenir les informations manquantes. Je tiens à souligner que notre cas est optimal et que pour un vrai programme en production, la différence de temps d'exécution serait bien plus importante car :

- la base de données n'est pas sur le serveur d'application (il faut donc rajouter de la latence réseau)
- la base de données de production fait rarement 9 Mo

Contre-exemple

Comme je vous le disais en début d'article, les jointures latérales sont très addictives et, si elles sont utilisées à mauvais escient, elles peuvent très facilement poser des problèmes de performance.
Reprenons donc l'exemple de notre liste de membres de country club. Imaginons que nous souhaitons récupérer les noms et prénoms de tous les membres du country club. Cette requête s'écrit très facilement ainsi :

```
select
 mem.firstname,
 mem.surname

from cd.members as mem
```

Mais nous pouvons aussi décider d'écrire cette requête avec une jointure latérale, parce qu'après tout, nous voulons tous les noms des membres et **pour chaque ligne**, nous souhaitons également obtenir tous les prénoms des membres.

```
select
 mem.firstname,
 mem2.surname

from cd.members as mem
 join lateral
   (
   select surname
   from cd.members as memBis
   where mem.memid = memBis.memid
   ) as mem2
 on true
```

Si on compare les temps d'exécution de ces deux requêtes, nous nous rendons compte que la première s'exécute dans un ordre de grandeur de 0.250ms alors que la deuxième version (qui fournit le même résultat) va s'exécuter en plus de 2.5 ms, soit plus de 10 fois plus lentement. Il est donc bien important de n'utiliser les jointures latérales que lorsqu'elles sont absolument nécessaires.

Méthodologie de test

Il est possible d'écrire une requête SQL de plusieurs manières différentes, toutes n'étant pas équivalentes en termes de plan d'exécution et donc de performance. Cependant, il est très facile d'introduire un bug en corrigeant une requête SQL et de ne plus obtenir le même résultat. Pour cette raison, il très important de mettre en place des tests unitaires sur les requêtes SQL.
Il existe plusieurs manières de faire. Dans notre cas, nous avons une base de données figée avec des données témoin censées représenter tous les cas d'usage possible de notre logiciel. Le plus simple est donc d'utiliser `except`. L'opérateur relationnel `except` permet de faire ressortir les tuples présents dans le premier ensemble mais pas dans le second. Notre test s'écrira donc ainsi :

```
(
 select
   mem.firstname,
   mem.surname

 from cd.members as mem
)
except
(
 select
   mem.firstname,
   mem2.surname

 from cd.members as mem
   join lateral
   (
   select surname
   from cd.members as memBis
   where mem.memid = memBis.memid
   ) as mem2
   on true
)
```

Les parenthèses sont facultatives, mais je trouve qu'elles améliorent la lecture. Si ma deuxième écriture de la requête est correcte, cette requête ne ramènera aucune ligne. Une autre manière de faire est d'enregistrer les résultats de chaque requête dans un fichier et de faire un `diff` entre les deux fichiers. La commande psql `\o` peut vous aider. N'hésitez pas à regarder aussi les options de formatage[2]. Là encore, le `diff` ne devra ramener aucune ligne.

Pour conclure sur les jointures latérales

Les jointures latérales sont un outil extrêmement puissant, mais il est important de ne bien les utiliser que lorsque c'est nécessaire, tout comme les super pouvoirs d'un super héro doivent être utilisés avec parcimonie. On ne peut que regretter que MySQL et MariaDB n'implémente pas encore cette fonctionnalité de la norme SQL:1999...
La suite dans le prochain numéro.

[2] https://www.postgresql.org/docs/current/app-psql.html

Pauline Iogna
Développeur sénior
Lunatech France
Duchess France Leader

Découplez vos applications avec l'architecture hexagonale

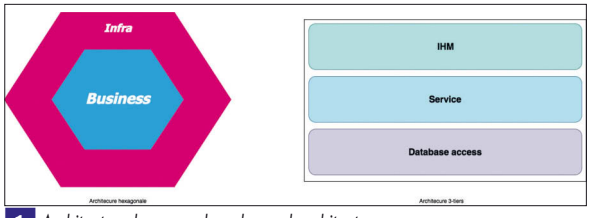
Partie 1

niveau 200

Qui n'a jamais rencontré des problèmes lors de montées de version, de migration de base de données, ou encore de changement de framework ou de librairie ? Qui ne s'est jamais retrouvé coincé dans un refactoring tant la complexité du code est importante ? Ou dans la difficulté de mettre à jour une règle de gestion ?

Un système d'information est composé de beaucoup de technologies différentes et parfois d'un grand nombre d'applications. Certaines de ces applications sont monolithiques, avec un grand nombre de couches, ce qui montre une volonté de séparer les choses. Mais quand vient le temps de migrer, c'est souvent douloureux, on se retrouve emmêlé au beau milieu du plat de spaghetti. On va la plupart du temps être confronté à des adhérences techniques. Par exemple avec un driver de base de données dont la montée de version peut impacter une grande partie du code. Dans le cas d'applications distribuées, même si il y a du réseau entre ces applications, elle peuvent être en forte connivence via le format des données. Changer le format d'un côté va avoir de forts impacts de l'autre côté. Faire évoluer client d'une open api dont le format a changé peut impacter toutes les couches d'un applications si elle ne sont pas découplées.

Mais alors comment se prémunir de ces problèmes ? Comment anticiper le futur sans faire de sur-conception ?



1 *Architecture hexagonale vs layered architecture*

2 *Séparer le business de l'infra*

Séparer le code business du code infra

Popularisée par Alistair Cockburn, l'architecture hexagonale offre une réponse possible aux problèmes évoqués.

L'*architecture hexagonale* est une façon de concevoir les application. Elle s'oppose (avec un certain nombre de points communs) à d'autres styles de conception d'applications comme l'*architecture 3-tiers*, ou l'*onion architecture*.

Sa mise en oeuvre se base principalement sur la séparation du code métier et du code technique en appliquant certains des principes DDD (Domain Driven Design), design pattern et principes SOLID (Single responsability principle, Open-closed principle, etc). 1

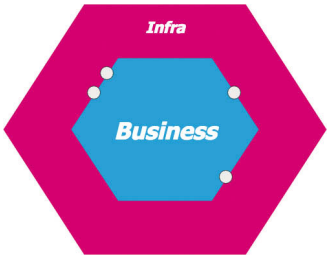
Business

Le coeur de l'application est le code business. C'est dans cette partie que doivent se trouver toutes les règles de gestion, c'est à dire le code fonctionnel de l'application.

Le coeur doit être découplé du code technique que l'on appelle *l'infra* en architecture hexagonale.

Le business est égoiste, il ne connait que son langage

A la lecture du code métier, on doit facilement comprendre l'intention des règles de gestion. Au contraire, le code métier ne doit donner aucune information sur la partie technique. Il est important de ne pas dupliquer les règles métier, donc les regrouper dans une couche business à part entière facilite l'application de ce principe.



3 *Utilisation de ports pour faire communiquer le business et l'infra*

Infra

La partie infra est la partie technique. C'est dans la partie infra que se trouve le code spécifique qui n'est pas métier.

On va trouver des adapteurs :

- Les DAO (Data Access Object) qui vont avoir des dépendances vers le ou les drivers de bases de données et ORM de type hibernate.
- Les clients de Web Service, toute la gestion de connections http doit se faire dans l'infra.
- La gestion des fichiers, l'accès au filesystem.

On va trouver du web :

- Le code MCV, les templates HTML, les controllers et un modèle dédié pour le front.

On va trouver aussi des configurations :

- Le code lié au framework d'injection de dépendances.
- Les schedulers, ordonnanceurs, job manageurs etc…
- Le code de monitoring de l'application, le code devops.

Hexagone 2

Le coeur : le business, autour : l'infra.

Comment ces deux parties peuvent-elles communiquer ? Via des ports. Les ports sont des contrats de méthodes, définis dans des interfaces pour les langages qui le supportent. L'infra et le *business* communiquent via ces ports.

Un port est une interface. 3

C'est l'utilisation des interfaces en tant que contrat de service qui crée du couplage faible. Une autre dénomination de l'architecture hexagonale est Ports and Adapters. Les ports permettent au business

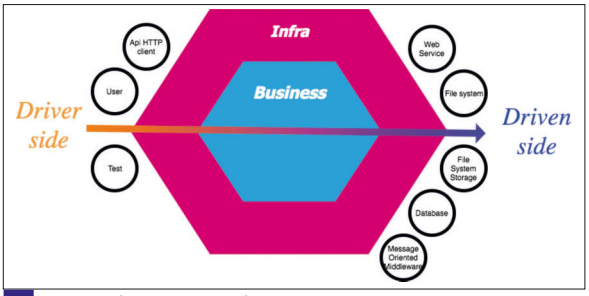
de communiquer. Les business services sont exposés à l'extérieur par des ports.
Le business est client des différentes actions nécessaires par des ports : par exemple un port vers un DAO ou un port vers un client de web service. 4
Dans ce service il a des dépendances vers un DAO, vers un autre service et vers un file gateway. Le service ignore les détails d'implémentations technique. Il utilise les ports avec un contrat métier. Il ne sait même pas quelle est la base de données utilisée dans l'application, ni quel est le système de stockage de fichier utilisé.

```
public class ProductServiceImpl implements PortProductCatalogService {

    @Inject
    private PortEventService eventService;

    @Inject
    private PortProductDao productDao;

    @Inject
    private PortFileGateway fileGateway;

    @Override
    public void registerNewProduct(Product product) {
        eventService.createNewEvent(CREATE_PRODUCT, product.getEan(), getCurrentUser().getName());
        productDao.save(product);
    }

    @Override
    public List<Product> allProducts() { return productDao.all(); }

    @Override
    public Optional<Product> findProductByEan(String ean) {
        eventService.createNewEvent(CONSULT_PRODUCT, ean, getCurrentUser().getName());
        return productDao.findByEan(ean);
    }
```

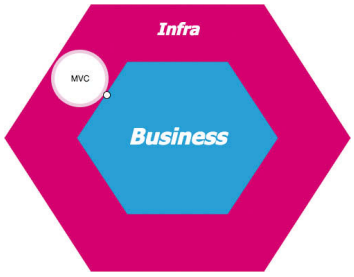
4 *Implémentation d'un service et ses dépendances (ports)*

Driver side vs Driven side

Contrairement à ce que l'on pourrait penser en premier lieu en regardant les schémas qui semblent symétriques, une application hexagonalisée n'est pas symétrique. Il y a un côté qui drive : user side (IHM, tests fonctionnels), et le côté drivé : server side (base de données, client de web service, production dans des files de messages). 5



5 *Driver side vs Driven side*



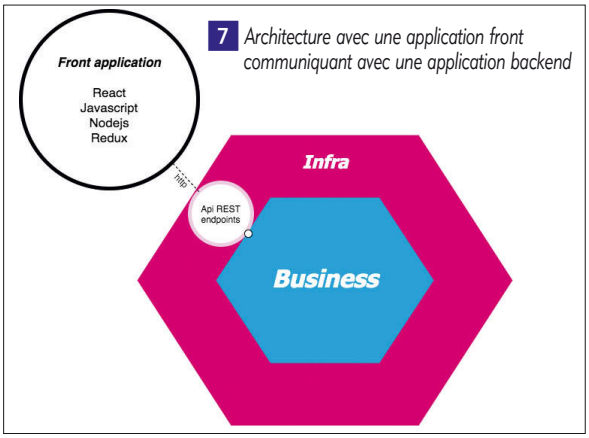
6 *L'implémentation MVC se trouve dans l'infra*

Exemple :
MVC et architecture hexagonale

Si je fais du développement MVC (Modele View Controller), est-il possible de faire de l'architecture hexagonale ?
Prenons le cas d'une application simple qui gère des produits. Une IHM permet de créer, modifier, supprimer ces produits. La partie front est développée en MVC.
Le code MVC qui est l'implémentation de de l'IHM se trouve dans l'infra, côté *user side*. Le MCV s'inscrit parfaitement dans une application conçue en architecture hexagonale. Habituellement on l'implémente à l'aide d'un framework comme playframework ou springMVC, c'est ce qui fait qu'en architecture hexagonale, le code MVC sera considéré comme technique. Cette partie du code aura un modèle dédié utilisé dans les vues et les controllers. Les controllers se chargent d'organiser la navigation d'une page à une autre mais il n'y a pas de règles de gestion dans la partie MVC. Ce sont les controllers qui doivent déléguer aux services tout traitement métier. 6

Et les frameworks javascript

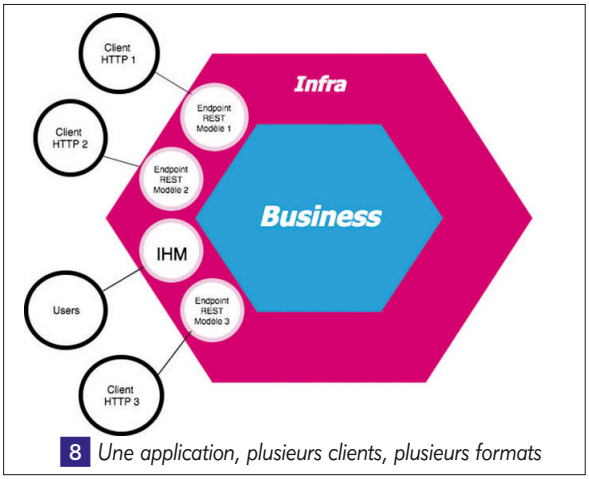
Prenons l'exemple d'une application composée d'une application front et d'une application java. L'application front a la responsabilité de la présentation des données. Dans ce cas l'application front ne devrait pas être conçue sur le modèle de l'architecture hexagonale. Il n'y a pas de métier dans l'application front mais uniquement de la mise en forme des données, de l'organisation de composants pour gérer les comportements dynamiques. L'application front est donc un client de l'hexagone, elle n'en fait pas partie. 7



7 *Architecture avec une application front communiquant avec une application backend*

Découplage

Ce qui fait l'efficacité de l'architecture hexagonale, c'est le très fort découplage entre les différentes parties (et pas couches) de l'application, la partie métier et aussi les différentes parties techniques avec la partie métier et entre elles, par opposition avec les *architectures 3-tiers* qui sont très utilisées dans les systèmes d'information. Dans ce système d'architecture *3-tiers* on parle de couches qui sont empilées les unes sur les autres. Le métier est bien séparé du reste, mais il y a une unique couche de présentation, et une unique couche d'accès au bases. En pratique, le modèle utilisé est unique. Une couche de présentation unique permet difficilement de répondre à un besoin de présentation spécifique en fonction des clients. 8



8 *Une application, plusieurs clients, plusieurs formats*

La suite dans le prochain numéro.

Marie-Alice Blete
Architecte Logiciel Java et Data Engineer installée à Lyon, vous pouvez la retrouver à bidouiller son installation domotique ou en salle d'escalade sur son temps libre.

Un serveur domotique maison petit budget, ou comment faire tweeter sa porte d'entrée

niveau 100

Vous avez un Raspberry Pi qui n'est pas (ou trop peu) utilisé chez vous ? Un petit budget ? Pourquoi ne pas en faire un serveur domotique maison ?

Pour moins de 200 €, vous trouverez ici de quoi commencer. La solution s'appuie sur le logiciel openHAB (https://www;openhab.org/) et sur les protocoles de communication Wifi, Bluetooth et RFXCom.

Attention cependant : RFXCom est un protocole radio ouvert, ce qui signifie que vos voisins peuvent y accéder s'ils le veulent. Par exemple, pour piloter un portail de garage, il vaut mieux se tourner vers les protocoles Zigbee ou Zwave. Pour une installation à petit budget, RFXCom est suffisant. Avant d'investir dans un protocole de communication donné, je vous conseille fortement de réfléchir aux objectifs de votre installation domotique. La suite de cet article se basera sur mon installation personnelle avec RFXCom, mais vous pouvez tout à fait envisager la même chose avec d'autres technologies.

LE MATERIEL

Il vous faudra :

- 1 Raspberry Pi 2 ou plus (+ alimentation, connexion internet, une carte SD ... bref, un Raspberry Pi fonctionnel). Si vous comptez utiliser le bluetooth et que vous avez un Raspberry Pi 2, il vous faudra en plus une puce bluetooth installée,
- Des objets connectés divers : une enceinte Bluetooth, une Chromecast, Alexa ...,
- Un RFXtrx433XL émetteur-récepteur RFXCom USB 433.92 Mhz (100 € environ),
- Des modules compatibles avec l'interface RFXCom. De mon côté :
  - Un thermomètre/hygromètre Oregon Scientific (30 €),
  - Un détecteur d'ouverture connecté DiO (15 €),
  - Une prise connectée DiO (30 €),
  - Des volets roulants Somfy RTS (cher), 1
- Une bonne motivation pour prendre en main openHAB. C'est un outil génial, mais qui nécessite un peu d'apprentissage…

1

PREMIERE ÉTAPE : INSTALLATION D'OPENHAB

OpenHAB est une plateforme domotique open-source. Sa particularité est d'être extrêmement flexible, de s'intégrer avec énormément de technologies, modules et systèmes avec une interface utilisateur agréable. La communauté est très vivante (c'est un point crucial pour ce type de projet !), mais plutôt anglophone. Je vous conseille vivement de lire la documentation disponible : https://www.openhab.org/docs/ .

Pour installer openHAB, vous avez plusieurs façon de faire. La première est d'utiliser l'image préconfigurée basée sur Raspbian : openHABian. Les détails d'installation sont ici : https://www.openhab.org/docs/installation/openhabian.html . Si vous n'êtes pas à l'aise avec Linux, ça peut être une bonne idée. Au contraire, si votre Raspberry Pi est déjà avec un OS fonctionnel, vous pouvez l'installer via un gestionnaire de package (apt, yum).

Voici ci-dessous les instructions pour installer openHAB sur un raspberry Pi avec Jessie :

```
#Installation
sudo apt-get update
sudo apt-get upgrade

sudo apt-get install openjdk-8-jre
wget -q0 - 'https://bintray.com/user/downloadSubjectPublicKey?username=openhab'
 | sudo apt-key add -
echo 'deb https://dl.bintray.com/openhab/apt-repo2 stable main' | sudo tee /etc/apt/
sources.list.d/openhab2.list
sudo apt-get update
sudo apt-get install openhab2
 -
### Pour que le service openHAB se lance au démarrage
 sudo /bin/systemctl daemon-reload
 sudo /bin/systemctl enable openhab2.service
```

```
## Donner accès au port USB à OpenHab
sudo usermod -aG dialout openhab
```

Vous devriez pouvoir accéder à l'interface Paper UI: http://[adresse de votre pi]:8080/paperui/index.html#/
Paper UI est une des possibilités pour réaliser la configuration : c'est celle que nous allons utiliser dans la suite, car la plus simple.

DEUXIÈME ÉTAPE : INSTALLATION DE BINDINGS, THINGS ET ITEMS

Le principe d'openHAB est la modularité : pour connecter un objet (**Thing**) il faut d'abord installer le module (**Binding**) correspondant. Une fois votre objet connecté, vous aurez la possibilité d'avoir des valeurs (**Item**).
Par exemple, pour connecter la Chromecast, il faut installer le **Chromecast Binding**, ce qui me permet ensuite d'ajouter le **Thing Chromecast**. Le Thing Chromecast me permet ensuite de rajouter un item de type switch : Mute. Cet item me permet de mettre la Chromecast en mode "mute". Vous me suivez toujours ?
Vous allez constater que la liste des possibilités est très longue, et qu'il est en plus possible de profiter des bindings développés par la communauté ou d'en créer un soi-même (Java). Pour plus de détails sur la configuration, je vous conseille de faire un tour sur : https://www.openhab.org/docs/tutorial/configuration.html
Pour revenir au RFXCOM, c'est un peu plus compliqué que pour la Chromecast : il faut en plus installer un "bridge" : le boîtier que vous avez est un Thing, mais il sert aussi de pont vers les autres objets que que vous avez, qui sont aussi des Thing. Il vous faudra dans l'interface de PaperUI :

- Installer le Binding RFXCOM (Configuration > Bindings > Icone +=) 2
- Installer le Thing RFXCOM USB Transceiver (Configuration > Things > Icône + > RFXCOM Binding > Add Manually) en spécifiant le port, et en activant les options correspondant au objets connectés que vous avez (AC, X10, Oregon Scientific et ARC pour ma part). Pour le reste, la configuration par défaut devrait fonctionner. 3 4
- Installer les Things qui devraient apparaître dans le menu Inbox : ils sont détectés par openHAB dès qu'ils envoient un signal.
- Ajouter les items correspondants. Par exemple, pour le contact de porte, il faut rajouter l'item "Contact" (Configuration > Things > [Nom donné à votre objet] > Channels > Contact > Icône +) 5

Je recommande fortement d'utiliser régulièrement le système de backup. Cela permet d'éviter que le travail d'une journée soit perdu à cause d'une fausse manip' difficile à réparer.
En cas de problème, vous pouvez aussi consulter les logs, soit dans **/var/log/openhab2**, soit via le client. Faites un tour sur :

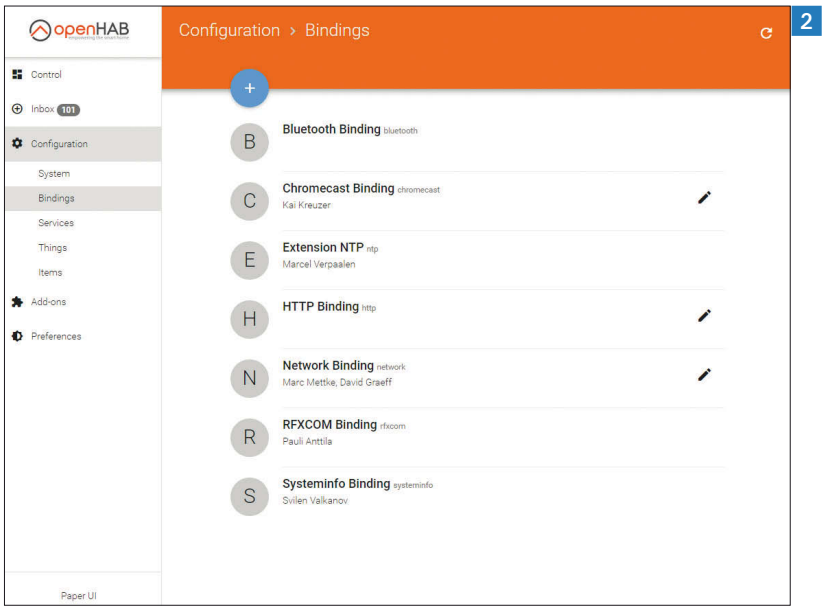
```
sudo openhab-cli console
```

(mot de passe par défaut : habopen)

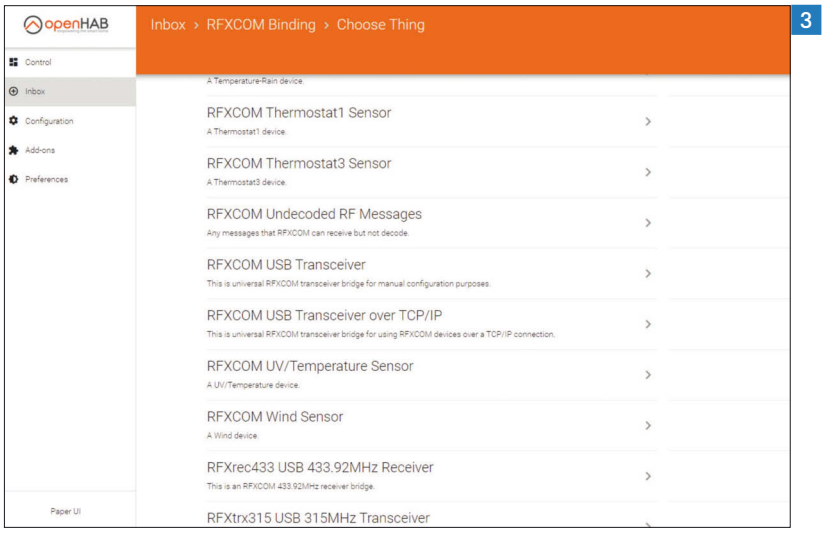
TROISIÈME ÉTAPE : CRÉATION DES RÈGLES

Il ne reste plus qu'à créer des règles. Ceci n'est pas possible dans la GUI, il faut passer par de la configuration par fichier. Par ailleurs, toutes les configurations réalisées dans Paper UI se retrouvent dans **/etc/openhab2/** . Les "rules" s'écrivent simplement dans un fichier **.rules** du répertoire rules.
L'exemple ci-dessous fonctionne avec l'add-on Twitter préalablement installé dans PaperUI.

```
rule "Porte Ouverte"
when
    Item Porte_Contact changed to OPEN
then
    sendTweet(String::format("%tT", new java.util.Date) + " La porte est ouverte")
end

rule "Porte Fermée"
when
    Item Porte_Contact changed to CLOSED
then
    sendTweet(String::format("%tT", new java.util.Date) + " La porte est fermée")
end
```



2



3

Maintenant, à vous de jouer ! Les possibilités sont tellement nombreuses que votre imagination sera la seule limite de votre installation...

Quelques exemples chez moi :

- Ma porte d'entrée tweete,
- Mes volets roulants sont compatibles RTS, et je peux les commander avec openHab. Plus besoin de télécommande.
- J'ai testé le text-to-speech sur ma Chromecast : elle disait "Bonjour [prenom] !" lorsque que quelqu'un entrait dans l'appartement par détection de nouvelle connexion Wi-Fi (mais j'ai arrêté, trop flippant),
- Avec une API REST d'une station météo externe, je compare les températures intérieures et extérieures pour gérer les volets.
- Avant qu'elle ne rende l'âme, je streamais de la musique sur une enceinte Bluetooth. Cela me permettait de pouvoir écouter de la musique via mon téléphone, sans que toute les sorties son de mon téléphone ne soient retransmises à l'enceinte.

EN BONUS : L'ACCÈS À DISTANCE ET L'APPLICATION MOBILE

Le petit plus sympa d'openHAB : le cloud, un accès à distance sécurisé. La façon la plus simple de faire est d'utiliser https://www.myopenhab.org/. Après avoir créé un compte et installé le service associé dans Paper UI, vous allez pouvoir accéder à votre installation domotique depuis l'extérieur de chez vous.

La cerise sur le gâteau ? L'application mobile ! Elle permet de se connecter à votre serveur en local ou à distance (si vous avez fait la configuration ci-dessus) via votre téléphone. L'application peut aussi servir de tableau de bord sur une tablette fixe.

Pour gérer l'interface, il vous faudra créer un sitemap. C'est le même principe que pour les "rules" : il 's'agit d'un fichier **.sitemap** dans le répertoire **/etc/openhab2/sitemaps** de votre pi.

La documentation https://www.openhab.org/docs/configuration/sitemaps.html est bien faite, et vous permettra de faire des choses sympas, bien qu'un peu limitées.

Ma version très simple me permet de gérer mes volets roulants (pratique quand on a qu'un interrupteur fixe), et de voir l'historique de température (pour déprimer pendant la canicule de cet été).

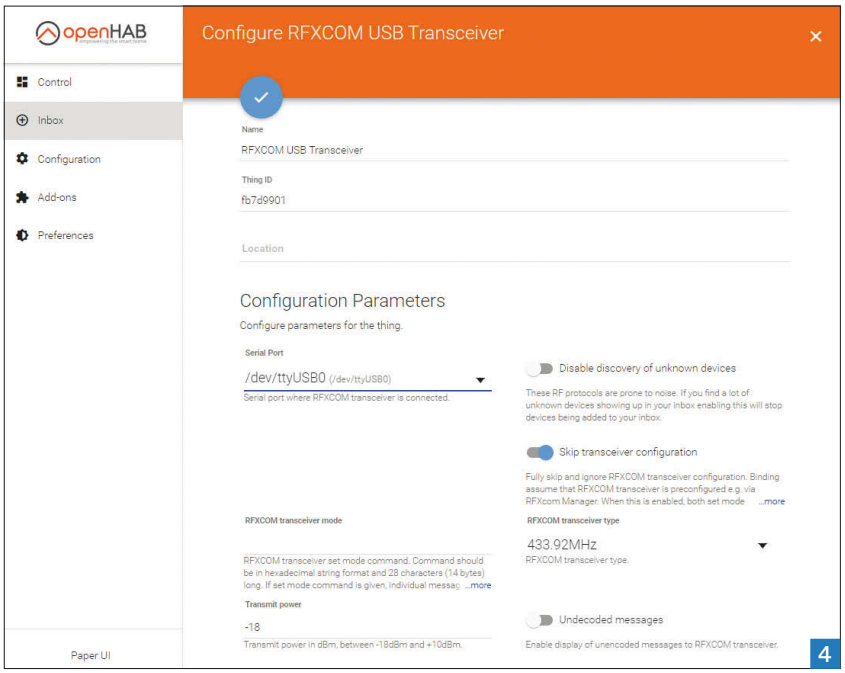
CONCLUSION

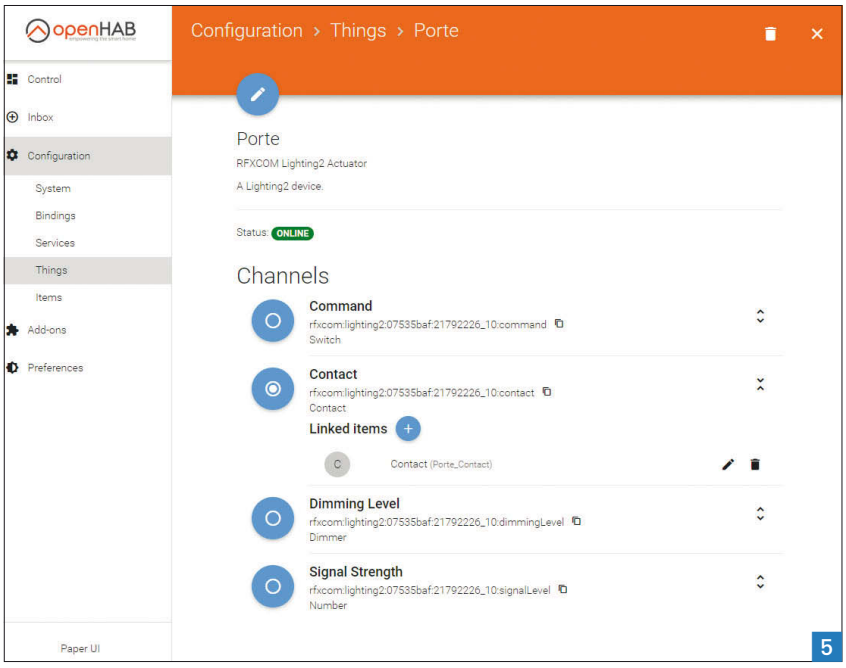
Cette solution avec openHAB vous demandera sûrement plus d'effort qu'une solution toute prête du commerce. En revanche, niveau budget et évolutivité, c'est imbattable !

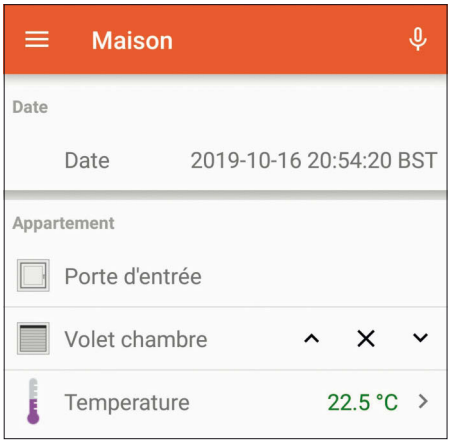
A titre de comparaison, le prix d'une box Somfy + détecteur d'ouverture + une prise connectée s'élève à plus de 350 € en magasin de bricolage, et les possibilités sont beaucoup plus limitées.

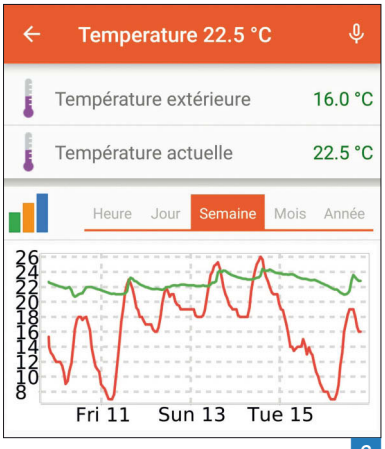
Il existe aussi d'autres plateformes open source domotique : j'ai aussi testé Domoticz. L'avantage de cette dernière solution est principalement la forte implication de la communauté francophone. OpenHAB est cependant vraiment plus flexible, et l'accès distant avec l'application est un plus.

Mon prochain projet sera de réguler mon chauffage avec openHAB. J'ai une vieille chaudière, et un thermostat pilotable par API REST ne ferait pas de mal. Avec un peu d'imagination (et pas mal de motivation), tout est possible... •









Claire Villard
Développeuse Java chez V3D, éditeur de logiciels lyonnais, et organisatrice du Lyon Java User Group.

niveau 200

Utilisation d'un cache distribué à travers l'exemple d'Hazelcast

Gerd Altmann de Pixabay

Nous connaissons tous la belle histoire d'un projet qui marche. Au début, tout commence avec une petite application, sa base de données, et son front-end. Les premières fonctionnalités s'étoffent et l'application est adoptée, elle monte en charge. C'est alors que les problèmes commencent à arriver.

L'un d'entre eux, très important à prendre en compte, est l'appel répété à une source de données, ou à une méthode coûteuse. Ces quelques secondes mises à l'échelle vont avoir un impact important sur la performance globale de l'application. Un cache distribué peut répondre à cette problématique, et son usage sera particulièrement adapté si une des collections de données est volumineuse ou si l'application fonctionne déjà en cluster.

Le cache va stocker le résultat de l'appel sous une forme directement utilisable par l'application et rapidement accessible. De plus, en étant distribué, cela permet qu'une donnée obtenue ou modifiée par un noeud du cluster soit disponible rapidement pour tous les autres, sans qu'il soit nécessaire que tous procèdent au traitement.

Si une collection est distribuée sur différents noeuds du cluster, il est également possible de s'appuyer sur cette répartition de la donnée pour répartir également la charge de traitement, sans qu'il soit nécessaire de mettre en place un système supplémentaire.

Un grand pouvoir implique de grandes responsabilités

Les caches, surtout distribués, doivent être manipulés avec précautions. Ils offrent de grands services, mais ajoutent une couche de complexité qui ne doit pas être négligée.

Ces outils apportent la disponibilité des données pour tous les noeuds du cluster. La tentation est alors grande d'y placer toutes ses données, puisqu'elles seront rapidement disponibles. Hélas, cela peut augmenter grandement la mémoire, le CPU, et la bande passante utilisés par l'ensemble du système, limitant le gain.

Il faudra donc veiller à l'utiliser à bon escient. Chaque accès à une donnée stockée dans le cache implique un coût de sérialisation et un coût réseau. Des données qui n'évoluent jamais, tels des dictionnaires de données figés, et relativement rapides à charger au démarrage ou au fil de l'eau, peuvent très bien être stockées dans une collection locale (une bonne vieille **HashMap** en Java), et non dans une collection distribuée.

A l'inverse, un catalogue de données très volumineux, demandant un chargement via des opérations coûteuses, et fréquemment appelé, aura toute sa place dans le cache. Il pourra être réparti entre les noeuds, s'il est trop volumineux pour un seul membre du cluster ou pour répartir la charge, ou au contraire être répliqué sur tous les noeuds pour limiter l'utilisation du réseau. Cela doit être étudié en prenant en compte les spécificités de chaque projet.

De nombreuses autres questions se posent, à propos du chargement des données et de leur durée de vie au sein du cache. Elles sont également importantes pour déterminer l'impact que la mise en place du cache aura sur l'application, en terme de performances et de complexité.
La mise à jour notamment, peut être un problème épineux. Une erreur dans sa gestion peut conduire à des données incohérentes, entre les noeuds du cluster, et/ou entre le cache et le silo de données persistant.
Une fois les données insérées dans le cache (au démarrage de l'application, ou à la demande), il faut gérer leur cycle de vie. Elles peuvent expirer, ou être mises à jour.
Lors d'une mise à jour, celle-ci peut être propagée aux autres noeuds de manière synchrone ou asynchrone, par copie (de la nouvelle donnée) ou par invalidation (de l'ancienne donnée).
Si les données proviennent d'une source persistante, la cohérence du cache et de la source doit également être assurée. Le cache peut être la référence et déclencher la mise à jour de la source, ou inversement.
Dans les deux cas, une mise à jour asynchrone et non transactionnelle implique que la cohérence est assurée "si tout va bien" (*eventually consistent*). Dès qu'une incohérence est détectée, il faut alors recharger les données depuis la référence. Cela n'étant pas toujours possible automatiquement, l'application peut alors nécessiter d'être complètement redémarrée ou le cache entièrement rechargé.
Si la cohérence doit absolument être assurée en permanence (*strongly consistent*), une réplication synchrone et un système de transactions devront être mis en place. Ces contraintes étant coûteuses en ressources et en temps de traitement, le besoin doit être évalué au regard de la criticité des données pour l'application, et de la tolérance possible.

Hazelcast

Une fois la mise en place d'un cache distribué décidée et les contraintes précisées, vient le choix de l'outil.
L'un d'eux est Hazelcast IMDG (pour *In-Memory Data Grid*, grille de données en mémoire) **(lien:** https://hazelcast.com/ **)**, développé par Hazelcast, Inc, société située à Palo Alto. Il est disponible sous forme d'un projet Open Source et d'une version propriétaire payante offrant des fonctionnalités de sécurité et de monitoring avancées. Les deux moutures sont disponibles au téléchargement ou en version *Software As A Service*.
Le choix d'Hazelcast pour cet article vient essentiellement de sa facilité d'intégration dans un environnement Java.

Note: *Tous les exemples de code sont inspirés de la documentation officielle d'Hazelcast 3.12.2, la dernière version à l'heure de la rédaction de cet article,* https://docs.hazelcast.org/docs/3.12.2/manual/html-single/index.html *et adaptés ou créés par l'auteur.*

```
Config cfg = new Config();
HazelcastInstance instance = Hazelcast.newHazelcastInstance(cfg);
```

Il suffit de lancer plusieurs fois ces 2 lignes de code, sur la même machine, pour obtenir un cluster fonctionnel.
On peut d'ores et déjà créer une Map, y ajouter des éléments, et les lire, indifféremment sur le même noeud ou sur un autre.

```
Map<Integer, String> mapCustomers = instance.getMap("customers");
mapCustomers.put(1, "Joe");
mapCustomers.put(2, "Ali");
mapCustomers.put(3, "Avi");

System.out.println("Customer 1: " + mapCustomers.get(1)); // Joe
System.out.println("Map Size:" + mapCustomers.size()); // 3
```

Cependant, démarrés ainsi, l'application et le noeud Hazelcast partagent la même JVM Java.
Cela offre quelques raccourcis pratiques, notamment pour la sérialisation, puisque tous les objets applicatifs sont connus du serveur Hazelcast.
Cependant, les threads et les espaces mémoires se retrouvant eux aussi partagés, en particulier la **Heap**, l'application et Hazelcast peuvent s'impacter mutuellement.
Bien sûr, cela oblige également à développer son application en utilisant un langage fonctionnant sur la JVM.
Faire le choix d'utiliser Hazelcast en tant qu'application autonome permettra de profiter des clients pour Java, .NET, C++, Python, Scala, NodeJS et Go, et de partager les données entre des applicatifs de technologies différentes. Hazelcast peut être également être utilisé avec le protocole Memcache, ou via son API REST (https://docs.hazelcast.org/docs/3.12.2/manual/html-single/index.html#hazelcast-clients) .
C'est pour cela qu'il est préférable d'utiliser Hazelcast en mode client-serveur.
Le noeud Hazelcast fonctionnera de manière autonome, dans sa propre JVM, et le client s'y connectera, qu'il soit situé sur le même serveur ou un autre. Cela permet également de mettre en commun le cluster Hazelcast pour différentes applications, par exemple différents services d'une architecture microservices.
Si le démarrage d'un noeud serveur ne change pas par rapport à l'exemple précédent, son accès est légèrement différent puisqu'on devra créer un client.

```
ClientConfig clientConfig = new ClientConfig();
HazelcastInstance client = HazelcastClient.newHazelcastClient( clientConfig );
Map<Integer, String> map = client.getMap( "customers" );

System.out.println("Customer 1: "+ map.get(1)); // Joe
System.out.println( "Map Size:" + map.size() ); // 3
```

Chaque accès à la Map provoque un appel à travers le réseau, le transfert de l'objet demandé au client, et sa désérialisation.
Cela pouvant être coûteux, pour les appels fréquents, on pourra mettre en place le *Near Cache*, ou "cache de proximité". Il s'agit d'un cache interne au client permettant de limiter les appels au cluster. On réservera cette solution à ce cas d'usage et quelques autres très spécifiques, les problématiques étant multipliées.

Les codes proposés sont basiques, et utilisent tous les paramètres par défaut d'Hazelcast, notamment pour la configuration des Maps et autres objets distribués.
Il est possible de configurer précisément chaque type d'objet, pour le cas général, ou en précisant des patterns de noms.
On peut par exemple créer une configuration générale pour toutes les Maps, avec 1 backup asynchrone, et une autre spécifique aux Maps dont le nom commence par "sync", qui elles auront un backup synchrone.

Configuration

Toute la configuration d'Hazelcast peut se faire via un fichier **hazelcast.xml** ou **hazelcast.yaml**, ou au moyen de la configuration Java. Le choix de l'une ou l'autre solution réside principalement dans le besoin de pouvoir faire évoluer la configuration indépendamment du code, ou au contraire du souhait de la figer pour éviter les erreurs. Les deux peuvent être combinés, le XML configurant par exemple l'utilisation du réseau (interface, ports, etc.) et étant complété par la configuration des objets écrite en Java.

```
<hazelcast>

 <!-- synchronized maps -->
 <map name="sync*">
   <backup-count>1</backup-count>
 </map>
 <!-- default configuration -->
 <map name="*">
   <backup-count>0</backup-count>
   <async-backup-count>1</async-backup-count>
 </map>

</hazelcast>
```

On prendra soin également de choisir le format de stockage le plus approprié pour chaque configuration. Une fois stockées dans une Map distribuée, les données peuvent y être conservées soit sous forme binaire (sérialisée, **BINARY**), soit sous forme d'objet (**OBJECT**). Si la forme binaire est plus performante pour les opérations de lecture et d'écriture d'objets, la forme objet permet de bénéficier d'une fonctionnalité indispensable lorsqu'on manipule des collections volumineuses : les prédicats et les processeurs.
Les prédicats permettent de sélectionner un ensemble de paires clés-valeurs correspondant à des critères.
Les processeurs, quant à eux, sont soumis aux noeuds du cluster, et réalisent de manière distribuée, en utilisant des threads issus des membres du cluster, des opérations sur les objets stockés.
Ces 2 mécanismes sont particulièrement utiles pour mettre à jour un grand nombre d'objets dans des maps en parallèle et en économisant le coût réseau et CPU lié à la lecture des données par le client applicatif.
Ces opérations nécessitant l'utilisation des données sous leur forme objet, il est donc nettement plus intéressant qu'elles soient stockées sous cette forme si ces opérations sont fréquentes.
Prenons une map **Customer**, chaque **Customer** ayant un identifiant, et un montant de commande dans un attribut nommé **totalInEuros**, contenant les données suivantes :

- Joe, avec un montant de 10€
- Ali, 100€
- et Avi, 150€

On peut, par l'utilisation d'un prédicat, retourner les clients ayant commandé pour plus de 50€, à savoir Ali et Avi.

```
ClientConfig clientConfig = new ClientConfig();
HazelcastInstance client = HazelcastClient.newHazelcastClient( clientConfig );

IMap<Integer, Customer> map = client.getMap( "customers" );

map.values(Predicates.greaterThan("totalInEuros", 50))
  .forEach(System.out::println);
```

Le code suivant permet, par l'utilisation d'un processeur, d'ajouter 10€ au montant de tous les clients :

```
map.executeOnEntries(new EntryProcessor<Integer, Customer>() {
  @Override
  public Object process(Map.Entry<Integer, Customer> entry) {
    Customer customer = entry.getValue();
    customer.add(10);
    // obligatoire pour sauver le résultat dans la map
    entry.setValue(customer);
    return customer;
  }

  @Override
  // traitement des réplicats
  public EntryBackupProcessor<Integer, Customer> getBackupProcessor() {
    return (EntryBackupProcessor<Integer, Customer>) entry -> {
      Customer customer = entry.getValue();
      customer.add(10);
      entry.setValue(customer); // obligatoire également
    };
  }
});
```

La map contient donc désormais :

- Joe, avec un montant de 20€
- Ali, 110€
- et Avi, 160€

Tous les objets contenus dans la Map ont été modifiés, sans qu'il ait été nécessaire de transférer toutes les données sur le client, de les mettre à jour, puis de les renvoyer sur le cluster.
Les processeurs assurent également une exécution atomique, très utile pour gérer les modification concurrentes.

Conclusion

Les caches distribués sont des outils incontournable pour les applications distribuées et manipulant un grand volume de données. Leur mise en place doit être faite avec précautions, afin d'éviter les écueils. Cependant, bien utilisés, ils permettent de diminuer fortement le temps d'exécution des traitements, d'éviter à l'application une gestion fastidieuse du parallélisme et de la répartition de la charge, et donc d'assurer la scalabilité. •

Emmanuelle ABOAF
Développeuse .NET chez **DCube** *et également sourde*
Depuis toujours, je suis concernée par l'accessibilité numérique

Comment gérer l'accessibilité en HTML ?

Selon l'article 47 de la loi du 11 février 2005, les entreprises publiques et les entreprises privées possédant un chiffre d'affaire d'au moins 250 millions d'euros doivent rendre accessible leurs sites internet, intranet et extranet.

niveau 100

Contrairement aux idées reçues, il n'est pas difficile, ni coûteux, d'implémenter l'accessibilité numérique. Tout au long de cet article, je vais vous montrer des astuces toutes simples pour pouvoir rendre accessible vos sites facilement.

Qu'est ce qu'est l'accessibilité numérique ?

Il s'agit de rendre l'information et le contenu du site accessible aux personnes handicapées quelque soit la nature de l'handicap et quelque soit la façon/le moyen/l'appareil via lequelle site est consulté. L'accessibilité concerne au moins 12 millions de personnes handicapées :

- Handicap visuel
  Les personnes aveugles et malvoyantes utilisent un lecteur d'écran pour naviguer sur internet ou une loupe agrandissante pour lire le contenu du site.
- Handicap auditif
  Les personnes sourdes et malentendantes ne peuvent pas comprendre ce qui est dit sur les vidéos d'où la nécessité de sous-titrer les vidéos.
- Handicap moteur (Ex : IMC, tétraplégie, Parkinson...)
  Par exemple, les personnes atteintes de Parkinson tremblent tellement qu'elles ne peuvent pas utiliser la souris et qu'elles utilisent uniquement le clavier.
- Handicap cognitif (Ex : dyslexie, photosensibilité, daltonisme...)
  Par exemple, un site comportant des textes jaunes sur un fond d'orange est complètement illisible pour les personnes atteintes de photosensibilité. Il faut donc assurer les contrastes suffisantes et les polices adaptées pour l'utilisateur.

En cherchant à faciliter la vie aux utilisateurs ayant une déficience invalidante - dyslexie, daltonisme, surdité, mauvaise vue, etc. - on en vient à faciliter la vie de **tous** les utilisateurs. Ainsi, **l'accessibilité est une affaire de tous**.

Tout d'abord, il est nécessaire de penser accessibilité **dès les balbutiements du projet**. Si vous décidez d'intégrer l'accessibilité à votre site alors qu'il est terminé, il va falloir revoir page par page pour s'assurer que chaque page est accessible. Du coup, en repassant chaque page, vous risquez de devoir changer pas mal de choses. Résultat, c'est une perte de temps et de coût. C'est pourquoi il est important de prendre en compte l'accessibilité dès le début.

Grâce à HTML5 complété de WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications), on peut faciliter l'accessibilité des pages HTML aux utilisateurs de lecteur d'écran. Pour tester l'accessibilité de votre site, vous n'avez besoin que deux choses :

- votre clavier
- un lecteur d'écran

N'utilisez en aucun cas votre souris pour naviguer sur votre site, ce serait trop facile. Pour naviguer sur votre site avec votre clavier, utilisez la touche TAB :

Si vous voulez utiliser un lecteur d'écran, il en existe trois :

- **NVDA** pour Windows
- **Voice Over** pour Mac
- **TallBack** pour Android

Étant sourde, j'utilise la visionneuse de paroles de NVDA pour comprendre ce que dit le lecteur d'écran. Et oui, même le lecteur d'écran est accessible aux personnes sourdes et malentendantes.

Définir la langue de votre site

Il est impératif de commencer à définir la langue de votre site dans la balise HTML avec l'attribut **lang**. Si vous ne le faites pas, le lecteur d'écran vocalisera le site en anglais, langue utilisée par défaut.

```
<html lang="fr">
```

Si, sur votre page HTML, vous écrivez un mot ou phrase dans une autre langue, précisez la nature de la langue dans le **span** pour que le lecteur d'écran vocalise le texte dans la bonne langue.

```
<span lang="en">Hello World !</span>
```

Définir les rôles de votre page

Ensuite, il faut définir les rôles de votre page. Il est important de respecter la structure du document. Dans cette structure, il doit toujours avoir les balises **header**, **nav**, **main** et **footer**. La **main** est unique et **nav** réservé aux zones de navigation.

De plus, il faut également attribuer les rôles à cette structure :

- L'attribut **role="banner"** est placé dans la balise **header**
- L'attribut **role="main"** dans **main**
- L'attribut **role="content-info"** dans **footer**
- L'attribut **role="search"** est utilisé dans une div contenant la zone de recherche
- L'attribut **role="navigation"** est placé dans la balise "nav" et peut être utilisé plusieurs fois.

Ces rôles peuvent paraître redondants face aux balises mais sont nécessaires. L'attribut *role* et ses valeurs permettent de nous donner des informations sur le but de l'élément en question. Grâce à ces rôles, les lecteurs d'écran détectent, au chargement de la page, immédiatement la navigation ou plusieurs, la bannière, le contenu principal, la zone de recherche et le pied de page.

Respecter l'ordre des titres

Pensez la hiérarchie des titres (h1 à h6) comme une table des matières. Pour faciliter la navigation des lecteurs d'écran, les

headings doivent être dans le bon ordre : h1 > h2 > h3 > h4 > h5 > h6 et il ne doit pas y avoir de trou entre, par exemple, h1 et h3. Cela permet aux utilisateurs de lecteurs d'écran d'assimiler la hiérarchie de l'information. N'utilisez pas le heading juste pour réduire ou grossir le texte. Si vous voulez réduire ou grossir le texte, il faudra le styler via les feuilles de styles en CSS.

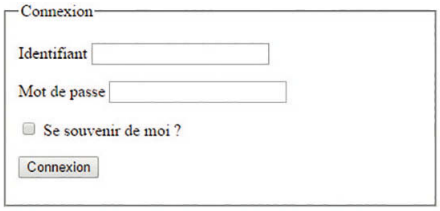
Rendre accessible un formulaire

Voyons maintenant comment nous pouvons rendre accessible un formulaire facilement.

Ce que nous codons :

```
<form>
<fieldset>
  <legend>Connexion</legend>
  <div>
   <label>Identifiant</label>
   <input type="text" id="txtIdentifiant" />
  </div>
  <div>
      <label>Mot de passe</label>
      <input type="password" id="txtMotDePasse" />
  </div>
  <div>
  <input type="checkbox" id="chkSeSouvenirDeMoi" />
      <label>Se souvenir de moi ?</label>
  </div>
  <input type="submit" id="btnEnvoyer" value="Connexion" />
</fieldset>
</form>
```

Ce que l'on voit :



Ce que le lecteur d'écran lit :

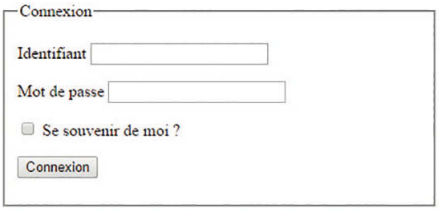
Connexion groupage
édition
vide
édition protégé
vide
case à cocher non coché
Connexion bouton

On voit déjà qu'il y a un problème lors de la restitution de l'information. Sans étiquette, le lecteur d'écran n'arrive pas à savoir ce qu'est le champ. Grâce à une astuce toute simple, je vais pouvoir rendre le formulaire accessible.

Ce que nous codons :

```
<form role="form">
<fieldset>
  <legend>Connexion</legend>
  <div>
   <label for="txtIdentifiant">Identifiant</label>
   <input type="text" id="txtIdentifiant" />
  </div>
  <div>
      <label for="txtMotDePasse">Mot de passe</label>
      <input type="password" id="txtMotDePasse" />
  </div>
  <div>
  <input type="checkbox" id="chkSeSouvenirDeMoi" />
      <label for="chkSeSouvenirDeMoi">Se souvenir de moi ?</label>
  </div>
  <input type="submit" id="btnEnvoyer" value="Connexion" title="Se connecter" />
</fieldset>
</form>
```

Ce que l'on voit :



Le visuel ne change absolument pas.

Ce que le lecteur d'écran lit :

Région formulaire
Connexion groupage
Identifiant édition
vide
Mot de passe édition protégé
vide
Se souvenir de moi ?
case à cocher non coché
Connexion
bouton Se connecter

Et voilà, grâce à l'attribut **for**, j'ai rendu mon formulaire accessible aux utilisateurs de lecteur d'écran. Les attributs id et *for* **doivent avoir le même nom** sinon cela ne fonctionne pas.
J'ai également mis l'attribut **role form** dans la balise form pour alerter que l'utilisateur entre dans la partie formulaire.
Attention, le *placeholder* ne remplace pas un label et n'est pas du tout vocalisé !

Rendre accessible des boutons

Voyons ensuite comment rendre accessible des boutons.

Ce que nous codons :

```
<h1>Choisir votre moyen de paiement</h1>
<button><img src="visa.png" /></button>
<button><img src="matercard.png" /></button>
<button><img src="paypal.png" /></button>
```

Ce que nous voyons :

Ce que le lecteur d'écran lit :

titre niveau 1
Choisir votre moyen de paiement
bouton
bouton
bouton

Comment savoir quel bouton appartient aux moyens de paiement VISA, MasterCard ou Paypal ? Grâce à une astuce tout simple, on peut le rendre accessible !

Ce que nous codons :

```
<h1>Choisir votre moyen de paiement</h1>
<button aria-label="Payez avec VISA"><img src="visa.png" alt="" /></button>
<button aria-label="Payez avec MasterCard"><img src="matercard.png" alt="" /></button>
<button aria-label="Payer avec PayPal"><img src="paypal.png" alt="" /></button>
```

Ce que nous voyons :

Choisir votre moyen de paiement

Ce que le lecteur d'écran lit :

titre niveau 1
Choisir votre moyen de paiement
bouton
Payer avec VISA
bouton
Payer avec MasterCard
bouton
Payer avec Paypal

Vous constatez ainsi deux choses :

- le visuel ne change pas
- rien qu'en ajoutant l'attribut **aria-label** (ou l'attribut **title** qui fonctionne de la même manière), vous rendez accessible les boutons.

Aussi, quand vous implémentez une icône, on vous demande de mettre **aria-hidden="true"** car l'icône n'est pas un élément pertinent à vocaliser. L'*aria-hidden* permet de ne pas être vocalisé par le lecteur d'écran. Par contre, veuillez toujours de mettre un *aria-label* pour accompagner l'icône afin de vocaliser le contenu du bouton comme l'exemple suivant :

```
<button aria-label="Fermer">
   <i aria-hidden="true">&times</i>
</button>
```

Rendre accessible les images

L'attribut **alt** est une information précieuse pour les lecteurs d'écrans. Mais attention, ne l'utilisez pas n'importe comment !

- Si l'image est purement décorative, laissez l'attribut alt vide.
- Si l'image contient une information importante, décrivez-la dans l'attribut *alt*.

Ne mettez pas le nom du fichier de l'image sinon le lecteur d'écran va vocaliser "*image_1-2019.jpg*" en "*image underscore un tiret deux mille dix-neuf point j p g*" et le nom du fichier n'est pas une information pertinente !

Par exemple, j'affiche cette image sur mon site et je voudrais la décrire.

```
<img src="img_1-2019.jpg" alt="un t-shirt avec un petit arc-en-ciel entourant le sigle a11y" />
```

Grâce à l'attribut **alt**, le lecteur d'écran va vocaliser "un t-shirt avec un petit arc-en-ciel entourant le sigle a11y".

Rendre accessible les liens

Si votre site comporte plusieurs liens ayant un même intitulé "Lire la suite", comment l'utilisateur du lecteur d'écran peut savoir quel lien "lire la suite" appartient à l'article qu'il veut lire ? Sans modifier l'intitulé, vous pouvez ajouter l'attribut *title* ou l'*aria-label*, les deux fonctionnant de la même manière.

```
<a href="#" target="_new" aria-label="Lire la suite de l'article comment gérer
 l'accessibilité en HTML - Nouvelle fenêtre">Lire la suite</a>

<a href="#" title="Lire la suite de l'article SQL et les fonctionnalités avancées">Lire la
suite</a>
```

Grâce à ces astuces, le lecteur d'écran vocalise :

Lire la suite de l'article comment gérer l'accessibilité en HTML - Nouvelle fenêtrelien
Lire la suite
lien Lire la suite de l'article SQL et les fonctionnalités avancées

Il est important de mentionner dans l'*aria-label* que le lien ouvre une nouvelle fenêtre. Cela donne une information supplémentaire pour l'utilisateur du lecteur d'écran qui saura quand il ouvre ce lien, ce sera sur une nouvelle fenêtre. Sans cette information, il ne saura pas et cherchera l'article sur la page où il est actuellement.

Rendre accessible une fenêtre modale

```
<div role="dialog" id="dialog1" aria-labelledby="dialog1Titre" aria-describedby
="dialog1Desc">
    <h2 id="dialog1Titre"> Bienvenue sur ma fenêtre modale </h2>
    <p id="dialog1Desc">
     Bonjour tout le monde ! Je suis contente de vous démontrer à quoi sert
     les attributs ARIA.
    </p>
</div>
```

Ainsi, en rendant accessible la fenêtre modale, le lecteur d'écran va vocaliser :

dialogue Bienvenue sur ma fenêtre modale Bonjour tout le monde ! Je suis contente de vous démontrer à quoi sert les attributs ARIA.

Avec les trois attributs, le lecteur d'écran va tout de suite vocaliser le titre et le contenu de la fenêtre.

- L'attribut **role="dialog"** permet de signaler que c'est une fenêtre de dialogue
- L'attribut **aria-labelledby** décrit le titre de ma fenêtre
- L'attribut **aria-describedby** décrit le contenu de ma fenêtre

Comment rendre accessible un message d'alerte ou une notification ?

Lorsque je clique sur mon bouton "Valider mon formulaire", je souhaite afficher un message d'alerte disant que mon formulaire n'est pas valide. Il s'affiche visuellement mais le lecteur d'écran ne restitue rien.
Du coup, quelle est la solution pour rendre accessible le message ?

```
<div id="notification" role="alert" aria-live="polite">
  <button onclick="close()" aria-label="Fermer la notification" ><i aria-hidden="true">
&times;</i></button>
  <span>Attention, le formulaire n'est pas valide !</span>
</div>
```

Tout d'abord, j'attribue le **role** "alert" pour signifier que j'ai un message d'alerte qui s'affiche. Mais avec seulement ça, le lecteur d'écran ne restitue toujours pas le contenu. Grâce à l'**aria-live**, on va restituer le contenu du message au déclenchement d'un événement et dans ce cas-précis au clic du bouton.
En ajoutant ces deux attributs, le lecteur va correctement me restituer le message dès que je clique sur le bouton :

alerte Attention, le formulaire n'est pas valide !
Fermer la notification bouton

L'aria-live possède trois valeurs :
- **off** (valeur par défaut) : l'aria-live est désactivé
- **polite** (valeur la plus utilisée) : cet attribut indique au lecteur d'écran qu'il doit attendre que l'utilisateur doit inactif avant de lui présenter une mise à jour
- **assertive** : au contraire de la valeur polite, assertive permet d'interrompre l'utilisateur tout de suite.

Comme vous le constatez, l'ARIA est facile à implémenter et est surtout utilisé pour rendre accessible les contenus simples et riches.
Voici quelques autres attributs ARIA existantes :
- aria-label : permet de titrer l'objet (équivalent du title).
- aria-labelledby : est utilisé pour indiquer les ID des éléments qui titrent l'objet.
- aria-describedby : est utilisé pour indiquer les ID des éléments qui décrivent l'objet.
- aria-required : élément obligatoire
- aria-checked: élément coché
- aria-invalid : élément invalide
- aria-live : permet de restituer le contenu à la suite d'un nouvel événement déclenché
- aria-expanded : élément déplié/replié
- aria-hidden : élément caché (visible visuellement mais caché au lecteur d'écran)

Il ne suffit pas seulement d'implémenter l'accessibilité dans le code source. Il faut aussi **visuellement** s'assurer qu'entre autres selon les critères du RGAA (référentiel général d'amélioration de l'accessibilité) :
- votre site respecte le contraste avec un ratio de 3:1 ;
- votre site garde le contenu sans l'altérer avec un zoom jusqu'à 200% ;
- que la prise de focus de chaque élément soit accessible depuis la touche TAB de votre clavier et visible ;
- que les images soient décrites et les vidéos sous-titrées ;
- que l'information ne doit pas être donnée uniquement par la couleur ;

Pour vous assurer que votre site est accessible, vous pouvez utiliser des outils existants :
- Accessible Colors (EN) (https://accessible-colors.com/) pour définir les contrastes de votre site
- Axe Deque (EN) (https://chrome.google.com/webstore/detail/axe-web-accessibility-tes/lhdoppojpmngadmnindnejefpokejbdd?hl=fr), une extension Chrome, pour effectuer des tests automatique ou assistés d'accessibilité
- Firefox Dev Tools (FR) (https://developer.mozilla.org/fr/docs/Outils/Inspecteur_accessibilite), comporte déjà dans son inspecteur l'onglet Accessibilité pour également effectuer des tests d'accessibilité
- Accessibility Insights (EN) (https://accessibilityinsights.io/), même utilisation que les deux précédents extensions mais pour Edge Insider
- Outil d'évaluation de l'accessibilité (FR) (https://github.com/Tanaguru/Accessibility-Evaluation-Reports) pour effectuer des audits d'accessibilité pour chaque critère RGAA 4.

Conclusion

En conclusion, en implémentant les balises d'accessibilité avec HTML et ARIA, vous ne changez pas le visuel de vos composants et de votre site. Grâce à cet article vous avez pu constater qu'avec quelques astuces toutes simples vous pouvez rendre accessible facilement et simplement votre site internet.
A noter, qu'il n'y a pas que les balises à utiliser, pensez également au **visuel** de votre site, nécessaire pour d'autres handicaps. Je vous invite à lire les critères de RGAA et à consulter les sources suivantes pour que votre site soit accessible.
Je vous recommande de consulter le site "L'accessibilité numérique, et si nous agissons ?" pour vous permettre de mieux comprendre les besoins de différents handicaps : https://www.atalan.fr/agissons/fr/
Pour en savoir plus sur ARIA :
https://developer.mozilla.org/fr/docs/Accessibilit%C3%A9/ARIA/Techniques_ARIA
Pour en savoir plus sur RGAA 4 :
https://numerique.gouv.fr/publications/rgaa-accessibilite/
Pour en savoir plus sur l'article 47 :
https://www.legifrance.gouv.fr/affichTexteArticle.do?cidTexte=JORFTEXT000000809647&idArticle=LEGIARTI000006682279&dateTexte=&categorieLien=cid

Pour en savoir plus sur l'utilisation des lecteur d'écran :
https://access42.net/Publication-des-resultats-de-l-etude-sur-l-usage-des-lecteurs-d-ecran.html

SEO

En ajoutant les étiquettes des champs (attributs **id** et **for**), la description des images (attribut **alt**), en définissant la langue du site et en respectant l'ordre des titres et l'utilisation des balises (title, header et footer) améliorent à la fois le référencement naturel et l'accessibilité d'un site internet.

Gaëlle Acas
Site Reliability Engineer chez Talend
co-organisatrice du CNCF (Cloud Native Computing Foundation) Meetup Nantes.
Cloud Addict, j'adore jouer avec les containers, surfer sur les skills Dev&Ops & jongler dans le monde du Serverless

Contour, un proxy pour Kubernetes

Partie 1

Aujourd'hui il existe de multiples solutions de proxy dans le monde de Kubernetes (nginx, traefik ...). Dans cet article nous allons explorer le projet Contour de VMware.

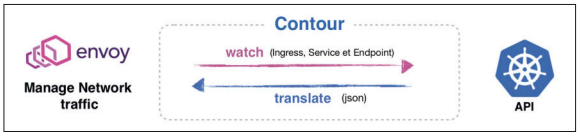
niveau 200

1 - Le projet

Le projet contour est né chez Heptio, une start-up à Seattle. A l'origine de ce projet, **Craig McLuckie** et **Joe Beda**, 2 des 3 créateurs de l'orchestrateur Kubernetes. Heptio a récemment été racheté par VMware, ce qui en fait un projet à suivre de près.

2 - Contour kézako ?

Contour est un proxy qui se base sur toute la puissance d'Envoy (un proxy open source situé au niveau de la couche d'application qui offre de nombreuses fonctionnalités avancées). Envoy reste très complexe à mettre en place et à configurer, Contour apporte justement cette simplicité en tant que surcouche. Il va communiquer les instructions à Envoy (en json) via l'API de Kubernetes.



C'est un projet qui bouge énormément. A l'heure où j'écris cet article la v1.0 vient tout juste de sortir. Contour est écrit exclusivement en Go et utilise gRPC pour communiquer avec l'API Kubernetes. Beaucoup de contributeurs sont issus du Projet Lyft Envoy, c'est un avantage considérable. En effet, quand il y a une dépendance avec un outil tiers, c'est important de pouvoir bénéficier du support direct de cet outil.

3 - Les ressources Ingress

Avant de parler de Contour, il vaut mieux comprendre le concept des Ingress. Un ingress vous permet de diriger le trafic au sein de votre cluster, c'est un load-balancer. Il se compose de 2 parties :

- Un contrôleur qui, comme son nom l'indique, va contrôler les requêtes via l'API Kubernetes (c'est le proxy)
- Une ressource qui va définir les règles de routage vers vos services.

Ingress, une ressource devenue obsolète :

Les ressources Ingress n'ont jamais évolué depuis la v1.1 de Kubernetes. Il suffit simplement de voir la déclaration d'un Ingress pour s'apercevoir que c'est toujours la version beta de l'API qui est appelée. Il y a beaucoup de discussions autour de l'évolution de ces ingress, d'autant plus qu'elles posent de plus en plus de limites.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
spec:
  backend:
    serviceName: testsvc
    servicePort: 80
```

Les limitations des Ingress

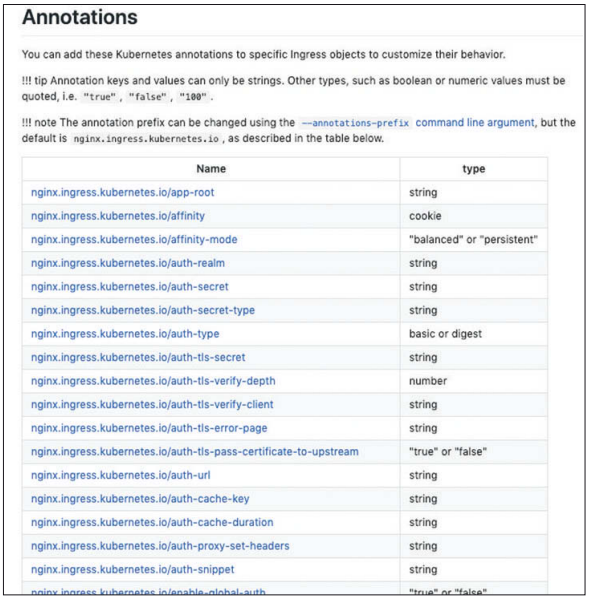
Un des problèmes majeurs de ces Ingress est qu'ils ne peuvent pas faire du cross-namespace.

Exemple, si on souhaite mettre en place du HTTPS, le certificat TLS doit se trouver dans le même namespace que l'Ingress. Quand on a plusieurs sous-domaine à gérer on ne peut donc pas profiter d'une organisation via les namespaces.

L'autre limitation vient de l'utilisation massive des annotations, trop d'annotations tuent les annotations !

Chaque contrôleur dispose de ses propres annotations. Rien qu'en suivant la doc pour NGINX, on peut voir qu'il y a plus d'une centaine d'annotations disponible pour adapter ses ingress à ses besoins. Cela devient vraiment confus et de plus cela introduit beaucoup de conflits.

Quand il faut faire du débug, il faut s'armer de patience !



Annotations

You can add these Kubernetes annotations to specific Ingress objects to customize their behavior.

!!! tip Annotation keys and values can only be strings. Other types, such as boolean or numeric values must be quoted, i.e. `"true"`, `"false"`, `"100"`.

!!! note The annotation prefix can be changed using the `--annotations-prefix` command line argument, but the default is `nginx.ingress.kubernetes.io`, as described in the table below.

Name	type
nginx.ingress.kubernetes.io/app-root	string
nginx.ingress.kubernetes.io/affinity	cookie
nginx.ingress.kubernetes.io/affinity-mode	"balanced" or "persistent"
nginx.ingress.kubernetes.io/auth-realm	string
nginx.ingress.kubernetes.io/auth-secret	string
nginx.ingress.kubernetes.io/auth-secret-type	string
nginx.ingress.kubernetes.io/auth-type	basic or digest
nginx.ingress.kubernetes.io/auth-tls-secret	string
nginx.ingress.kubernetes.io/auth-tls-verify-depth	number
nginx.ingress.kubernetes.io/auth-tls-verify-client	string
nginx.ingress.kubernetes.io/auth-tls-error-page	string
nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream	"true" or "false"
nginx.ingress.kubernetes.io/auth-url	string
nginx.ingress.kubernetes.io/auth-cache-key	string
nginx.ingress.kubernetes.io/auth-cache-duration	string
nginx.ingress.kubernetes.io/auth-proxy-set-headers	string
nginx.ingress.kubernetes.io/auth-snippet	string
nginx.ingress.kubernetes.io/enable-global-auth	"true" or "false"

Le futur des Ingress

Comme on peut le voir, les Ingress n'ont plus vraiment le vent en poupe. La Communauté Kubernetes envisage plusieurs implémentations basées sur Envoy pour inspirer la prochaine génération d'Ingress. D'ailleurs, depuis Kubernetes 1.10, il est maintenant plus facile d'étendre l'API grâce au CRD (Custom Resource Definition). Cela permet à n'importe qui de créer une nouvelle ressource et de la mettre à disposition de la communauté.

Contour a été l'un des premiers à exploiter ces CRD en créant la ressource HTTPProxy (anciennement appelée Ingress Route). HTTPProxy rend la gestion des routes beaucoup plus simple et sans limite dans vos clusters !

4 - Projet Contour, à la rescousse !

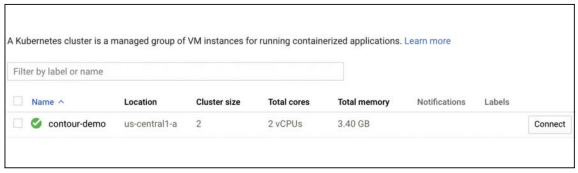
Comme dit précédemment, Contour utilise les CRD.
Voyons ensemble comment l'installer :
Il vous faudra dans un premier temps avoir un cluster Kubernetes, soit en local avec Docker4Mac ou k3s par exemple. Il existe aussi des clusters managés : GKE (Google Cloud), AKS (Azure) ou EKS (Amzone). Dans cet article j'utilise Google Kubernetes Engine, mais libre à vous de choisir.

A - Installation

1 - créer votre projet Google Cloud via la console
https://cloud.google.com/
2 - créer un cluster Kubernetes managé (GKE)
Via le SDK de google (que vous aurez préalablement installé - https://cloud.google.com/sdk/install) vous pouvez lancer la commande suivante :

```
$ gcloud beta container --project "[PROJECT_ID]" clusters create "contour-demo"
--zone "us-central1-a" --no-enable-basic-auth --cluster-version "1.14.6-gke.13"
--machine-type "g1-small" --enable-stackdriver-kubernetes
```

Une fois votre cluster créé, vous pouvez vous connecter depuis la console GCP en cliquant sur "Connect"



3 - Installation de Contour
L'installation de Contour se fait simplement via la CLI de kubernetes : kubectl apply :

Comme on peut le voir, Contour utilise bien les Custom Resource Definition afin d'étendre l'API de Kubernetes. Contour est installé dans un namespace projectcontour.

B - Pourquoi Contour résout des problématiques liées aux Ingress ?

Précédemment je vous ai parlé des limites des ingress. Dans un contexte multi-team cela peut même impacter une production. Voyons un cas concret :
En décembre, notre célèbre Père-Noël est en pleine effervescence ! Son site "Santa Ho Ho Ho" doit être en ligne le plus vite possible ! Il forme ses 2 équipes de lutins :

- la team "Lego"
- la team "Playmo"

Chaque équipe dispose de son Lutin DevOps :)
Santa Ho Ho Ho dispose du nom de domaine suivant : http://santa.local
Chaque team dispose de sa propre page :
http://santa.local/playmo pour la team Playmo
http://santa.local/lego pour la team Lego
Pour tous nos exemples nous allons utiliser une applications développée en NodeJS disponible sur Github : https://github.com/gaelleacas/contour-demo. Celle-ci est dockerisée et déployée en tant que service.

Ressource Ingress

Il faut d'abord mettre en place http://santa.local :
santa.local va pointer sur un service santa-v1

santa-ingress.yaml :

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: root-santa
 namespace: santa
 labels:

  app: root-santa
 annotations:
  kubernetes.io/ingress.class: "contour"
spec:
 rules:
 - host: santa.local
  http:
   paths:
   - path:
    backend:
     serviceName: santa-v1
     servicePort: 80
```

Exécutez la commande suivante : **kubectl apply -f root-santa.yaml**, après quelques secondes, l'URL http://santa.local répond :

Pour la team Lego on procède de la même manière :

lego-ingress.yaml :

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: lego
 namespace: lego
 labels:
  app: lego
 annotations:
  kubernetes.io/ingress.class: "contour"
spec:
 rules:
 - host: santa.local
  http:
   paths:
   - path: /lego
    backend:
     serviceName: lego-v1
     servicePort: 80
```

http://santa.local/lego répond :

La team playmo va appliquer cet ingress : **kubectl apply -f playmo-ingress.yaml**

playmo-ingress.yaml :

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: playmo
 namespace: playmo
 labels:
  app: playmo
 annotations:
  kubernetes.io/ingress.class: "contour"
spec:
 rules:
 - host: santa.local
  http:
   paths:
   - path: /playmo
    backend:
     serviceName: playmo-v1
     servicePort: 80
```

La team Playmo a maintenant son espace : **http://santa.local/playmo**

Jusqu'ici, la mise en production de chacune des teams se passe très bien. Chacune d'elles se concentre sur les features à venir, par exemple la mise en place d'un blog. La team Lego souhaite mettre son blog en production. Il sera accessible sur la route http://santa.local/blog. Elle doit donc ajouter sa route /blog à son ingress :

lego-ingress.yaml :

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: lego
 namespace: lego
 labels:
  app: lego
 annotations:
  kubernetes.io/ingress.class: "contour"
spec:
 rules:
 - host: santa.local
  http:
   paths:
   - path: /lego
    backend:
     serviceName: lego-v1
     servicePort: 80    # Route /blog
   - path: /blog
    backend:
     serviceName: lego-blog
     servicePort: 80
```

Le blog est accessible sur l'URL **http://santa.local/blog** :

De son côté, le lutin Ops de la team Playmobil met en production leur nouveau blog et ajoute donc sa route */blog*:

playmo-ingress.yaml :

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: playmo
 namespace: playmo
 labels:
  app: playmo
 annotations:
  kubernetes.io/ingress.class: "contour"
spec:
 rules:
 - host: santa.local
  http:
   paths:
   - path: /playmo
    backend:
     serviceName: playmo-v1
     servicePort: 80
   - path: /blog
    backend:
     serviceName: playmo-blog
     servicePort: 80
```

Ce blog est maintenant accessible : http://santa.local/blog

Comme on peut le constater, nous rencontrons l'un des principaux soucis des Ingress : pas de contrôle d'un namespace à un autre. Le blog Playmobil a écrasé le blog de la team Lego, et cela sans aucune alerte ni aucun warning. Nous sommes ici dans un contexte multi-team. Kubernetes apporte cette possibilité de séparer les domaines applicatifs via ses namespaces, cela devient donc important de pouvoir contrôler qui peut faire quoi au seins des clusters. Contour répond à ces problématiques via ses CRDs HTTPProxy.

Ressource HTTPProxy

Reprenons notre situation précédente mais cette fois-ci en utilisant HTTPProxy. La team de Santa va pouvoir mettre en place une délégation cross-namespace des routes. Pour se faire, c'est la ressource HTTPProxy root-santa qui sera maître : elle sera seule à déclarer un FQDN (Full Qualify Domain Name) et sera considérée "Root".

santa-ingress.yaml

```
apiVersion: projectcontour.io/v1
kind: HTTPProxy
```

```
metadata:
 name: root-santa
 namespace: santa
 labels:
  app: root-santa
spec:  # Resource "root" qui déclare le nom de domaine
 virtualhost:
  fqdn: santa.local
 includes:
 # délégation de la route /lego vers l'objet HTTPProxy
 # `lego` dans le namespace `lego`
 - name: lego
  namespace: lego
  conditions:
  - prefix: /lego
 # délégation de la route /blog vers l'objet HTTPProxy
 # `lego-blog` dans le namespace `lego`
 - name: lego-blog
  namespace: lego
  conditions:
  - prefix: /blog
 # délégation de la route /playmi vers l'objet HTTPProxy
 # `playmo` dans le namespace `playmo`
 - name: playmo
  namespace: playmo
  conditions:
  - prefix: /playmo
 # Route pour Santa Ho Ho Ho
 routes:
  - services:
    - name: santa-v1
      port: 80
```

Déclaration pour la team Lego :

```
apiVersion: projectcontour.io/v1
kind: HTTPProxy
metadata:
 name: lego
 namespace: lego
spec:
 routes:
 - services:
   - name: lego-v1
     port: 80
---
apiVersion: projectcontour.io/v1
kind: HTTPProxy
metadata:
 name: lego-blog
 namespace: lego
spec:
 routes:
 - services:
   - name: lego-blog
     port: 80
```

On retrouve bien la Home Page *Santa Ho Ho Ho* sur la route : http://santa.local, la team lego : http://santa.local/lego et le blog http://santa.local/blog
On remarque ici que la ressource "Lego" ne déclare pas de nom de domaine (ce qui n'était pas le cas dans les Ingress).
Voici la déclaration pour la team Playmo :

lego-ingress.yaml

```
apiVersion: projectcontour.io/v1
kind: HTTPProxy
metadata:
 name: playmo
 namespace: playmo
spec:
 routes:
  - services:
    - name: playmo-v1
      port: 80
  - conditions:
    - prefix: /blog
    services:
     - name: playmo-blog
       port: 80
```

Si la team Playmo ajoute une route */blog*, ce blog est maintenant accessible à : http://santa.loca/playmo/blog et n'endommage pas celui créé par la team Lego.
En cas de conflit, HTTPProxy donne un statut sur l'état des routes. Par exemple si la team Playmo déclare une nouvelle route :

playmo-faq-ingress.yaml

```
apiVersion: projectcontour.io/v1
kind: HTTPProxy
metadata:
 name: playmo-faq
 namespace: playmo
spec:
 routes:
  - conditions:
    - prefix: /faq
    services:
     - name: playmo-faq
       port: 80
```

la nouvelle route http://santa.local/playmo-faq ne sera pas accessible. Afin de comprendre pourquoi, il suffit d'exécuter la commande : kubectl get httpproxy --all-namespaces

```
(⎈ |contour-demo:santa)➜  gke git:(master) ✗ kubectl get httpproxy --all-namespaces
NAMESPACE   NAME         FQDN          TLS SECRET   STATUS     STATUS DESCRIPTION
lego        lego                                    valid      valid HTTPProxy
lego        lego-blog                               valid      valid HTTPProxy
playmo      playmo                                  valid      valid HTTPProxy
playmo      playmo-faq                              orphaned   this HTTPProxy is not part of a delegation
chain from a root HTTPProxy
santa       root-santa   santa.local                valid      valid HTTPProxy
```

On peut voir toutes les routes valides, sauf une, notre route playmo-faq qui est orpheline. En effet elle n'a pas été déclarée dans la ressource HTTPProxy root-santa et n'est donc pas prise en compte par Contour. Dans le cas où une team déclare un FQDN par erreur, cela met tout le routage en échec et plus aucune url n'est accessible. De plus le statut des routes devient "invalid". **Mais mieux vaut avoir un crash visible, qu'un dysfonctionnement silencieux**.
Il est cependant possible de restreindre la déclaration d'un HTTPProxy "Root" (déclarant un FQDN) à un ou plusieurs namespaces du cluster : par exemple on souhaite restreindre la déclaration des FQDN uniquement à la team santa, on ajoute cet argument `--root-namespaces=santa` au démarrage de Contour.
La suite dans le prochain numéro.

Michelle AVOMO
Développeuse fullstack et co-animatrice de la communauté craft chez **SOAT.**

niveau 100

Clean code : le nommage

D'expérience, parler de nommage comme introduction au clean code peut faire sourire. En effet, il est souvent dit (à tort) que s'intéresser à la question sur la qualité du code est irréaliste "pour la vraie vie". Aussi, quand une personne sensible à la qualité de son code dépasse ce préjugé et s'inscrit à une de nos masterclasses et que nous démarrons par la notion du nommage, nous lisons au mieux de l'interrogation sinon de la déception dans les regards.

Si je pouvais traduire l'expression qui nous parvient dans ces moments, ce serait : " en quoi le nommage fera de nous des "développeurs du futur" "?
Alors, pour commencer sur de bonnes bases, je ne crois pas que le nommage seul fasse de nous des "développeurs du futur", mais, il est indéniable que bien nommer ses classes, variables, méthodes, librairies etc. est une base solide vers des refactorings plus ambitieux. D'ailleurs, nous disons souvent que rendre son code lisible, c'est faciliter sa compréhension. Comprendre le code c'est pouvoir le maintenir donc pouvoir le faire évoluer sans en compromettre les fonctionnalités existantes.

L'importance du nommage

Comme souvent dans le développement logiciel, nous n'avons rien inventé, le livre **Code complete** de *Steve Mc Connel* aborde largement le sujet. Le livre **Clean Code** de *Robert C. Martin* y a pioché et a popularisé plusieurs notions clefs.
Si vous ne les avez pas lu/étudié, je vous les recommande vivement.
En attendant, pour revenir à l'importance du nommage, permettez-moi de vous soumettre à un petit jeu.
Soit le code ci-dessous :

```
public List getAll(List<Thing> list) {
  List result = new ArrayList();
  for (Thing x : list )
    if (x.getStatus() == 1)
      result.add(x);
  return list;
}
```

Sans aller plus loin, sauriez-vous expliquer le contexte fonctionnel de ce code ?
Maintenant, comparez avec le code ci-dessous :

```
public List getLivingCells(List<Cell> allCells ) {
  List livingCells = new ArrayList();
  for (Cell c : allCells)
    if (isAlive(c))
      livingCells.add(c);
  return livingCells;
}
```

Le code choisi est volontairement simple et pourtant, laquelle des deux versions exprime instantanément le contexte fonctionnel du code? Un jeu d'enfant n'est-ce pas ?
Allons plus loin.
Le manifeste agile promeut d'ailleurs ***" L'adaptation au changement plus que le suivi d'un plan "***, le métier l'a bien compris et c'est fort de cette assurance qu'il demande la nouvelle fonctionnalité suivante :

"Les cellules vivantes doivent avoir chacune 3 voisins maximum".

Pour un développeur, la version "mal nommée" nécessite quelques explications et souvent, ces explications sont pérennisées par … des … commentaires :(

```
public List getAll(List<Thing> list) { //list= ensemble des cellules
  List result = new ArrayList();// cellules vivantes à retourner
  for (Thing x : list )
    if (x.getStatus() == 1) // la cellule est en vie
      result.add(x);
  return list;
}
```

Autant vous le dire tout de suite, ce genre de commentaires de façon générale est une fausse bonne idée. Pourquoi ? Nous le voyons ci-dessous.

Ces commentaires qui nuisent au bon nommage

Bien que le but n'était pas de discuter des commentaires, il convient de voir comment leur simple présence peut nuire à un code se voulant propre.
Aussi, pour revenir l'exemple plus haut, la principale raison pour laquelle le commentaire associé est une fausse bonne idée est que souvent, les commentaires ne sont pas maintenus.
Par exemple, suite à la fonctionnalité ci-dessus, il est possible d'augmenter la condition existante comme ci-dessous :

```
if (x.getStatus() ==1 && nbVoisins <= 3) // la cellule est en vie
```

Le code a changé mais pas le commentaire. Ce commentaire est devenu obsolète.
Quand vous tombez sur ce type de mauvais commentaire, n'hésitez pas à renommer le bout de code et supprimer ledit commentaire.

```
Exemple : if (isAlive(x) && has3NeighborsAtMost())
```

L'exemple ci-dessus peut ne pas faire l'unanimité (il convient d'en discuter avec son équipe) mais, nous souhaitons que l'intention soit toujours auto-porté par le code.
Renommer (idéalement en utilisant son IDE) est la technique la plus pérenne et sans aucun risque pour y arriver (surtout quand on utilise son IDE!!).

Les commentaires "dans la vraie vie"

Dire que les commentaires sont des "code smells" et qu'il faut les

éviter suscite souvent des interrogations légitimes. Aussi, avant de continuer cet articles sur les bonnes pratiques de nommage, permettez-moi de revenir brièvement sur le cas des commentaires.
"Dans la vraie vie", un logiciel d'entreprise de plusieurs années d'existence (indépendamment du langage) contient plein de commentaires non fiables et obsolètes.
La plupart mériterait d'être supprimés. D'autres servent d'indices à un meilleur nommage. D'autres encore sont des informations clefs qui préviennent des risques de bugs, de perte de performance ou sont des rustines qui maintiennent un logiciel fonctionnel.
Parler d'éviter les commentaires dans le cadre du "clean code" revient souvent à éviter les "mauvais" commentaires.
Il y en a des bons fort heureusement mais il faut savoir les reconnaître.
Ci-dessous je vous propose quelques pistes pour différencier les bons et les mauvais commentaires.

Bons et mauvais commentaires	
Mauvais commentaires (non exhaustif)	Les commentaires explicatifs redondants. Les commentaires d'attribution Les commentaires trompeurs (à effacer ou corriger) Code commenté (code mort) Les commentaires de fin bloc Marqueurs de position
Bons commentaires	Commentaires Explicatifs (ex : RegEx, Algorithme) Commentaires de clarification et d'intention en cas d'échec de refactoring Commentaires expliquant les conséquences (ex: test très long à exécuter) Documentation d'API publique TODO ou FIXME : quand quelque chose reste à faire.

En définitive, un mauvais nommage n'est pas une fatalité. Nous avons la chance de travailler avec des environnements de développement permettant rapidement et sans aucun risque de renommer et exprimer l'intention de notre code au fil de l'eau. Profitons-en car, un code mal nommé est un "code smell". Le renommer est un réflexe de "boy scout" et peut éviter l'escalade vers plus de "code smells".

Refactorer par le renommage

Le simple fait de renommer ses classes et méthodes peut donner des résultats assez bluffants. J'ai tenté d'en faire la démonstration en 15 min à devoxx (https://frama.link/wcvEbgXF) plus tôt dans l'année.
Un des reproches qui m'a été fait suite à Devoxx France est de ne pas avoir donné de guide sur des règles et conventions de nommage.
La deuxième partie de cette article revient ainsi sur des conventions de nommages et quelques pratiques utiles.

Nommage : conventions et préconisations

Dans cette partie, nous allons revenir tour à tour sur :

- **les conventions de nommage** : ce sont des pistes à considérer lors du choix de nom de variable, de classes, énumérations etc.
- **les préconisations sur la taille optimale du nommage** : il s'agit de répondre au doute lié au fait d'abréger ou non versus exprimer exhaustivement l'intention de son code?
- **La règle du scope** : Cette règle s'applique aux variables et méthodes et il s'agit de revenir sur ce qui devrait influencer le choix de la taille idéale de leur nommage.

Les conventions de nommages générales

Les conventions ci-dessous ont pour but d'éclairer ou préciser les bonnes pratiques à considérer dans le cadre d'un projet standard. Elle peuvent servir de base mais nous encourageons toujours chacun à ouvrir le débat avec son équipe afin d'en réaliser une qui corresponde aux réalités de leurs langages / framework/paradigme de langage.

Quelques règles usuelles et bonnes pratiques		
Élément	**Convention**	**Exemple**
Classe	Noms / groupes nominaux Eviter autant que possible les mots parasites (manager, service...)	Recrutement, ConsultantRecruteur, ExperiencesCandidat
Variables de classe (field)	Noms / groupes nominaux	nom, adresse, preferencesDeMission
Variables locales (variable)	Noms / groupes nominaux Idéalement dérivé du nom de la classe	consultantsSurParis, clientActuelDuConsultant
Booléen	Prédicat d'état : verbe être + complément	estAccepte, estRefuse, estEnMission
Fonctions	Fonctions => Verbes à l'impératif Fonctions retournant un booléen => Prédicat	annulerEntretien(), enregistrer(), getNom()
Enumération / type fermé	Adjectif ou nom	EN_COURS, ACCEPTE
Événement	Verbe au participe passé	integrationTerminee, cooptationValidee, missionChoisie

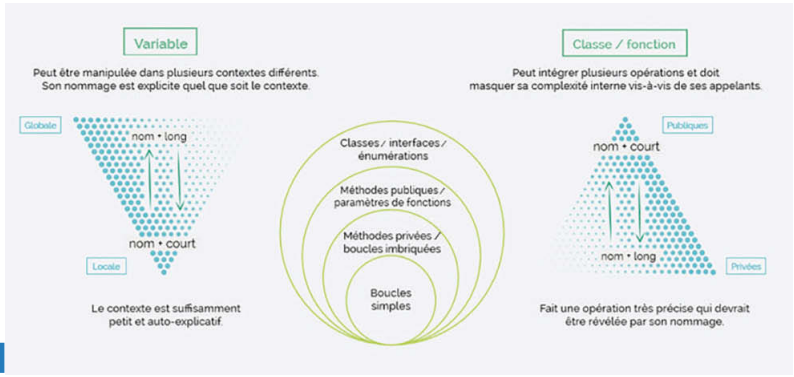
Lien entre la portée des variables (et méthodes) et leur nommage

Connue en anglais sous le terme : "scope length rule", cette règle propose de déterminer la taille « recommandable » d'un nom de variable ou de méthode suivant leur contexte et leur visibilité.
Ainsi donc :

- **Pour une variable** : Plus sa portée est grande, plus son nom doit être explicite (nom de variable plutôt long).
- **Pour une méthode/une classe** : C'est le contraire. Plus elle est exposée, plus son nom devrait être générique, donc court. 1

Taille optimale du nommage des variables et méthodes.

Malgré la règle du scope là dessus, le nommage n'est pas une science exacte; bien au contraire. Mais, si à la recherche d'un nom "parfait" de variable et ou de méthode, si vous tombez sur des noms trop courts ou trop long et que vous hésitez sur vers lequel aller, les quelques exemples ci-dessous peuvent servir de base de réflexion.



1

Les noms des variables et des méthodes devraient toujours être aussi courts que possible, aussi longs que nécessaire.	
Trop longs	nombreDeCvRecusDepuisLaDernierePromotion formationsProposeesAuxSoatiensEnMissionDePlusDeDixHuitMois
Trop courts	ndcv, cvs, f, fp, fps, enMission, dixHuit
Recommandables	nbCvRecusDepuisDernierePromo, formationsInternesPourMissionsLongues

Pourquoi éviter les noms trop longs ?

Dans le cadre d'une classe, méthode ou d'une interface, un nom trop long est souvent un indicateur de mauvais code (un "code smell"). Souvent, cela traduit le fait que cette méthode/classe/interface fait plusieurs choses (elle a plusieurs responsabilités) et cela va à l'encontre du principe de la responsabilité unique (Single Responsability Principle) des principes SOLID. Dans le cas d'une variable, c'est son contexte qui devrait permettre d'arriver à une taille optimale. Trop long, le contexte est sans doute trop large.

Pourquoi éviter des noms trop courts ?

Tout d'abord, parce que dans une classe ou dans une méthode, un nom de variable à un ou deux caractères rend sa discrimination difficile (exemple: o, i , ff etc.). À la recherche de leurs occurrences dans un fichier, les faux positifs sont multipliés et cela n'apporte aucune valeur au logiciel que le/la développeur/se élimine ces faux positifs à chaque recherche textuelle.

Ensuite parce que raccourcir au maximum un concept fonctionnel peut nous amener à privilégier un certain "confort d'écriture" au détriment du "confort de lecture". Privilégier le confort d'écriture se traduit par la présence d'abréviations subjectives, et/ou des noms imprononçables. Le confort de lecture revient à privilégier des noms faciles à lire et à prononcer permettant des conversations fluides et facilitant la compréhension du fonctionnel.

Choix de noms : les préconisations générales

Nous passons plus de temps à lire du code plutôt qu'en écrire. Dans tous les langages, nous pouvons décider de concevoir pour la lisibilité.

Nous avions édité quelques préconisations dans une refcard (https://frama.link/DpZUbAnk) dédiée au nommage, nous nous permettons de reprendre ci-dessous, quelques préconisations dans la recherche de l'expression de l'intention.

Toujours favoriser la communication

Faire une code review ou expliquer à voix haute ce qu'on a compris du code sont des exemples de situation où l'on a besoin de prononcer les noms des variables et des méthodes. Plus ils sont prononçables, plus la communication au sein de l'équipe est fluide.

Exemples à éviter	Ne pas obscurcir	Exemples recommandés
Nbr2Cdt, EtpRtmt	Choisir un nom imprononçable	nombreDeCandidats, etapeDuRecrutement
n2CVrc nbRec	Convention subjective (ex : abréviation)	nbCVRecus, totalCvRecus
p_strNom, _nom	Utiliser une notation désuète (ex : hongroise)	nom, nomDuCandidat

Éviter la désinformation

Il n'y a pas que les commentaires qui deviennent obsolètes. Les classes, modules, fonctions et variables peuvent aussi le devenir.

Exemples à éviter	Ne pas obscurcir	Exemples recommandés
getPLayer()	Nommer autrement que ce qui est fait dans le code	getWinner()
PrintScore()	Garder le même nom de classe/variable méthode alors que son sens a évolué	printScoreAndSendMail
Count()	Donner un nom générique, pas assez précis	countPoints(), countSets()

Éviter d'augmenter la complexité accidentelle

Exemples à éviter	Ne pas se répéter	Exemples recommandés
candidatList, candidatMap	répéter la structure de donnée dans le nom	candidats, candidatsParStack
p1,p2	numeroter les variables	premiereProposition, contrePropositio

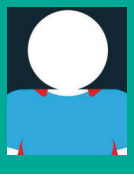
Conclusion

Nous écrivons du code d'abord pour les humains et non pour les machines. Prendre le temps de (re)nommer une variable, classe, méthode qui exprime l'intention c'est se rendre service à soi-même (vu que nous maintenons du code que nous écrivons à plus ou moins longue échéance) et aux autres après nous. Un bon nommage répond à la question du quoi, et pas à celle du comment. Dans cet article, nous avons voulu apporter notre pierre à l'édifice afin que chacun puisse à l'avenir contribuer à sa manière à un code plus propre. Et pour ceux et celles qui ne savent pas par où démarrer, nous espérons que les exemples ci-dessous tirées de notre refcard serviront de base de réflexion.

Bon code !

References :

Code complete - 2e édition - **Steve Mc Connel**
[refcard nommage] - https://frama.link/DpZUbAnk
Clean code nommage - Devoxx France - https://frama.link/wcvEbgXF



Ludivine Crépin
Docteur en Intelligence Artificielle et chercheuse publiée à l'international, est freelance depuis les années 2010. Passionnée d'IA, elle enseigne également et forme les pros sur ces technologies tout en continuant ses recherches théoriques et appliquées en IA grâce à l'école et au laboratoire qu'elle a créé, Up ESI.

Coder son premier apprentissage par renforcement simplement

Pour celui qui ne connait pas la méthode « éducative » de Pavlov, rappelez-vous de l'épisode de The Big Bang Theory où Sheldon essayait de changer le comportement de Penny en la récompensant par un chocolat. Voici donc le principe de l'apprentissage par renforcement (RL), beaucoup plus simple à comprendre et plus concret que les réseaux de neurones et le deep learning (DP).

niveau 100

Soyons un peu plus sérieux dans notre introduction. A l'heure d'aujourd'hui, l'industrie se focalise sur le machine learning (ML), une branche de l'intelligence artificielle (IA), la définissant comme une révolution, du même niveau qu'Internet. Pourtant les principes et les algorithmes du ML sont connus depuis plus d'une trentaine d'année… Le ML permet d'apprendre à reconnaitre des motifs, des patterns, aucun concept n'est réellement appris. Grâce à ces patterns, inconnus dans la plupart des cas de l'utilisateur et du concepteur, un programme peut reconnaître une image, prédire un comportement ou choisir une action à effectuer pour atteindre un objectif fixé. L'utilisation du ML en industrie a commencé par les arbres de décisions puis les réseaux de neurones qui se sont étendus au DL. Nous observons actuellement l'avènement du RL, notamment grâce à AlphaGo Zero qui a battu au jeu de go AlphaGo, un agent utilisant le DP.

Le RL représente un problème donné à une IA ou un agent autonome qui va chercher à le résoudre par expérimentations dans un environnement fermé donné. L'agent possède une stratégie et sera récompensé au cours des itérations par des bons points et des mauvais points, donnés par l'environnement. 1

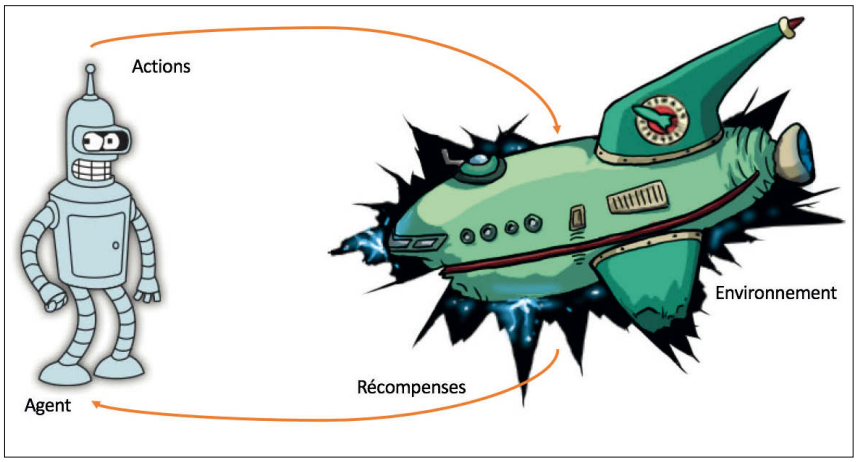
Ce type d'apprentissage est inspiré de la vie humaine. Souvenez-vous de vos notes à l'école, de vos bonus dans les jeux vidéo, des actions que vous répétez pour les améliorer à vos entraînements, etc. … L'objectif est le même ici : l'agent doit s'améliorer par lui-même en changeant sa stratégie à chaque itération en fonction des retours qu'il a eus.

Pour vulgariser, le RL possède deux approches :

- apprendre grâce aux états ou TD-Learning (*temporal distance*),
- apprendre grâce aux actions ou Q-Learning (*quality*).

Le TD-Learning sert à maximiser l'état supposé obtenu par une action alors que le Q-Learning sert à maximiser l'action par rapport à l'état courant. Ce dossier passera outre les détails mathématiques afin de donner un simple aperçu du RL pour que le lecteur puisse à son tour coder son propre agent apprenant et approfondir par la suite cet apprentissage s'il le souhaite.

Découvrons le RL par l'exemple et par le jeu. Le morpion est un jeu simple qui se joue entre deux joueurs, l'objectif étant d'aligner trois de ses pions sur une grille de 3x3. La faible complexité de ce jeu nous permet dans ce dossier de se focaliser uniquement sur le RL. Veuillez noter que les principes exposés ici sont largement applicables à des problèmes de plus grande envergure. Pour répondre à ce problème, nous utiliserons le Q-Learning : il paraît plus judicieux pour ce jeu de choisir la meilleure action apprise en fonction de l'état courant du plateau de jeu.



1 *Principe de l'apprentissage par renforcement*

L'environnement du morpion est tout simplement un objet manipulant la grille de jeu, une matrice de caractères, 'X' pour le premier joueur, 'O' pour le deuxième et ' ' pour une case vide. L'environnement doit pouvoir initialiser le plateau de jeu, savoir si une partie est finie avec un gagnant ou un nul, savoir quel joueur a gagné, organiser les tours de jeu et donner les récompenses à l'agent. Le code pour illustrer ce problème est réalisé en Python 3.x, sans utilisation de module tiers afin qu'il reste compréhensible aux novices en Python, quitte à perdre un peu en performance :

Code complet sur github

Pour qu'un agent puisse apprendre, il nous faut d'abord déterminer comment l'environnement va lui donner sa récompense.

Avec le jeu du morpion, cette question est assez simple : l'agent aura une récompense positive (1) s'il gagne une partie et une récompense négative (0) quand il perd une partie.

Quid de l'égalité ? Lorsqu'une partie est nulle, l'agent n'a pas perdu, il doit recevoir une récompense supérieure à 0, mais vu qu'il n'a pas gagné non plus, sa récompense doit être donc inférieure à 1. Par simplicité, en cas de partie nulle, l'agent recevra arbitrairement une récompense de 0.5. 2

Maintenant que l'environnement peut le récompenser et le faire

jouer, nous pouvons passer à l'apprentissage de notre agent.
On pourrait penser naïvement que l'agent va donner une valeur à chacune des cases jouables, et modifier leur valeur en fonction des coups joués. Or, l'agent va apprendre la grille entière, et cela par simplicité. Pour chaque grille possible, il va la retenir, lui attribuer une valeur relative à sa qualité, et modifier cette valeur avec les parties jouées. Ainsi l'agent sera capable, après un entraînement, de déterminer en fonction de ses expériences le coup à jouer pour maximiser sa victoire.
Pour ce faire, l'agent va posséder ce que l'on appelle sa stratégie, ou Q-table. Il s'agit pour notre agent d'un dictionnaire où la clé est la grille de jeu courante et la valeur de la clé la Q-value de cette grille. La Q-value va définir la qualité d'une grille donnée, plus elle sera grande, plus la grille sera propice à amener l'agent à la victoire. Cette valeur est initialisée au départ de manière arbitraire, dans notre cas 0. Pour simplifier, la Q-value pour un état donné se calcule avec la formule suivante :
Q_value = Q_value + a * (récompense + d * max(Q_value) - Q_value)
où a est le facteur d'apprentissage et d le facteur d'actualisation, tous deux un réel entre 0 et 1. A noter ces deux facteurs peuvent varier afin d'améliorer l'apprentissage de l'agent.

```python
def giveRewards(self):
    """ Give rewards for all player """
    end = self.playerWinner if self.winner() else ''
    if end == 'X':  # player1 win
        self.player1.receiveReward(1)
        self.player2.receiveReward(0)
    elif end == 'O':    # player2 win
        self.player1.receiveReward(0)
        self.player2.receiveReward(1)
    else : # nul game
        self.player1.receiveReward(0.5)
        self.player2.receiveReward(0.5)
```

2 *Comment l'environnement récompense un agent*

```python
def receiveReward(self, reward):
    """ Defines the policie, Q_table, with calculated q_value"""
    for board in self.oldBoards[::-1]:
        if str(board) not in self.policie.keys():
            self.policie[str(board)] = 0
        else:
            self.policie[str(board)] += self.alpha * (reward - self.policie[str(board)])
```

3 *Apprentissage Q-Learning par l'agent*

```python
if __name__ == '__main__':
    print('*** TRAININ PERIOD ***')
    # to log each action of agent
    logging.basicConfig(filename='game.log',level=logging.DEBUG)
    # change param to improve learning
    agent1 = Agent('X',0.2,0.9)
    agent2 = Agent('O',0.2,0.9)
    game = Environment(agent1, agent2)
    for i in range (10000):
        game.launchGame()
    # play with the agent
    print(" Let's go now !!! ")
    continuePlay = 'y'
    human = HumanPlayer()
    game = Environment(agent1, human, True)
    while continuePlay == 'y' :
        game.launchGame()
        continuePlay = input('Again?(y/n')
```

4 *Entraînement et déroulement d'une partie*

A chaque fin de partie, l'agent va donc parcourir tous ses coups (i.e. les grilles résultantes de ses coups appelées states dans le code), leur attribuer une Q_value et les stocker dans sa stratégie. 3

Ainsi l'agent va pouvoir déterminer grâce à son entrainement les grilles qui sont les plus susceptibles de le faire gagner et donc placer son pion stratégiquement. Pour choisir son prochain coup, l'agent va calculer tous les coups possibles (les cases vides). L'agent simule chacun de ces coups et sélectionne le meilleur en fonction de la Q_value de la grille qu'il va entraîner :

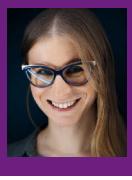
- Si un coup possible ne correspond à aucune grille de sa stratégie, l'agent lui donnera arbitrairement la valeur 0 ;
- Si tous les coups possibles ne sont pas connus par l'agent, l'agent choisit de manière aléatoire son prochain coup à partir des cases libres du plateau. Ce mécanisme permet à notre agent d'apprendre le plus de grilles possibles, créant une nouvelle grille à apprendre de manière aléatoire à chaque fois qu'il est face à l'inconnue. **Code complet sur github**

Pour apprendre, l'agent a maintenant tout ce dont il a besoin. Il suffit maintenant de l'entraîner. Comme pour AlphaGo Zero, notre agent va s'entraîner conte lui-même : il suffit de lancer un grand nombre de parties avec deux joueurs de type agents. Remarquez que nous n'avons jamais donné un seul indice à l'agent pour savoir jouer au morpion ou dicté explicitement les règles, l'environnement lui indique uniquement le résultat d'une partie.
Code complet sur github

Une fois l'entraînement fini, l'agent peut jouer contre un être humain ! 4
Pour réutiliser l'agent avec son apprentissage (disponible sur https://github.com/ludivineUp/programmezRL), il suffit de rajouter le stockage de sa stratégie et de la recharger à chaque partie, tout en sauvegardant sa nouvelle stratégie à chaque fin de partie afin d'améliorer son apprentissage. N'oubliez pas qu'un agent apprend par expérimentations, donc au plus il joue, au plus il affine son raisonnement.

Conclusion

Cet article expose un exemple simple du LR avec Q-Learning uniquement, laissant le lecteur découvrir le TD-Learning. Les principes développés ici sont utilisables pour tout autre problème où un programme informatique doit prendre une décision sans l'intervention d'un utilisateur.
Le LR permet à un agent, un programme, de trouver l'action la plus bénéfique pour résoudre un problème donné dans un environnement fermé. La problématique du RL réside principalement dans la définition des récompenses. Ce type d'apprentissage, comme les autres mécanismes du ML, requiert un grand nombre d'expérimentations avant d'atteindre un résultat satisfaisant mais se révèle des plus performants une fois un bon entraînement fini. Cet apprentissage peut être couplé à d'autres méthodes de l'IA telles que le DL, les CSP, les SMA, etc... Les combinaisons sont infinies.
Si vous voulez approfondir le ML et l'IA, qui n'est pas uniquement le ML ou le DL, vous trouverez plus de détails dans le livre de référence « L'intelligence Artificielle » de Russel et Norvig. •



Anastasia Lieva
twitter @lievAnastazia,
Fuzzy Humanist, Data Science Witch, Head of Data Science Comwatt

niveau 100

Extreme data science

Créateur et contributeur principal du framework de deep learning Keras, François Chollet a publié il y a quelques mois un tweet(1) dans lequel il soutient que, pour qu'une entreprise soit à l'aise avec le machine learning, un projet majeur ne suffit pas. Il est donc nécessaire d'acquérir une vraie culture du machine learning, c'est-à-dire construire les bons processus, avoir le bon esprit et le bon outillage à travers de petites victoires sur un spectre de différentes problématiques.

(1) https://bit.ly/2NZCzWG

1

Ce qui est intéressant dans son tweet est le fait que pour que vous soyez dans le machine learning comme un poisson dans l'eau, François Chollet ne mentionne pas l'utilisation du Big Data, du Edge ou du Quantum Computing. Il parle cependant des processus, de l'esprit et de l'outillage, les éléments indispensables pour hacker la culture Data-Science en entreprise.

Mais comment créer ces processus, cette culture et cet outillage ? Je me suis posé cette question, avec mes expériences professionnelles, mais aussi en recueillant des témoignages de mes collègues et de plusieurs participants de meetups et de conférences auxquels j'ai assisté. Dans ces témoignages, on retrouve d'un côté les frustrations des data-scientists qui sont bien loin d'être épanouis dans leur travail : soit leurs tâches sont assez éloignées de la data-science avancée, soit elles ne sont pas assez intéressantes ou challengeantes, soit le résultat de leur travail finit systématiquement au placard. De l'autre côté, il y a également de la déception chez les CTOs, CEOs, POs et développeurs, qui n'arrivent pas à collaborer efficacement avec les data-scientists et générer du profit de l'investissement dans ces projets.

Un feedback récurrent que j'ai pu avoir est celui de CTOs qui regrettent que l'investissement des data-scientists s'arrête à la création du modèle, en ne s'impliquant pas sur sa prise en production et sa maintenance, préférant passer à un nouveau challenge data-science.

J'ai commencé à faire mon étude sur plusieurs de ces cas d'usage, j'ai aussi analysé les échecs et les réussites, et j'ai présenté les résultats de mes analyses dans une série d'articles et de conférences intitulées "Make Data Science Great Again". Dans cette série j'ai proposé des pratiques, qui d'après mon expérience mais aussi d'après celles de mes collègues, se sont avérées efficaces pour la gestion de projets data-science. En y repensant avec du recul, l'ensemble de ces pratiques bienfaisantes m'a rappelé une autre approche… celle de l'eXtreme Programming.

L'eXtreme Programming de Kent Beck et d'aujourd'hui

L'extreme programming (XP) a été inventé par Kent Beck entre 1996 et 1999, et est précurseur en tant que méthode agile. Cette méthode répond à deux objectifs principaux :

Le premier est de responsabiliser toutes les parties prenantes qui participent au développement du logiciel, du client au développeur, tout en apportant de la transparence dans le cycle de développement.

Il faut savoir qu'à l'époque avant la naissance de ces méthodes, plusieurs profils travaillaient en silos : le client exprimait ses besoins avant que le projet ne commence, et revenait plusieurs mois plus tard pour valider la livraison finale. Les développeurs implémentaient les fonctionnalités, les testeurs, quant à eux , testaient ensuite ce code et renvoyaient une liste d'erreurs à corriger aux développeurs, et ainsi de suite. Tout ce processus s'avérait vraiment long et pénible, nécessitant plusieurs allers retours, qui auraient pu être évités ... avec l'XP.

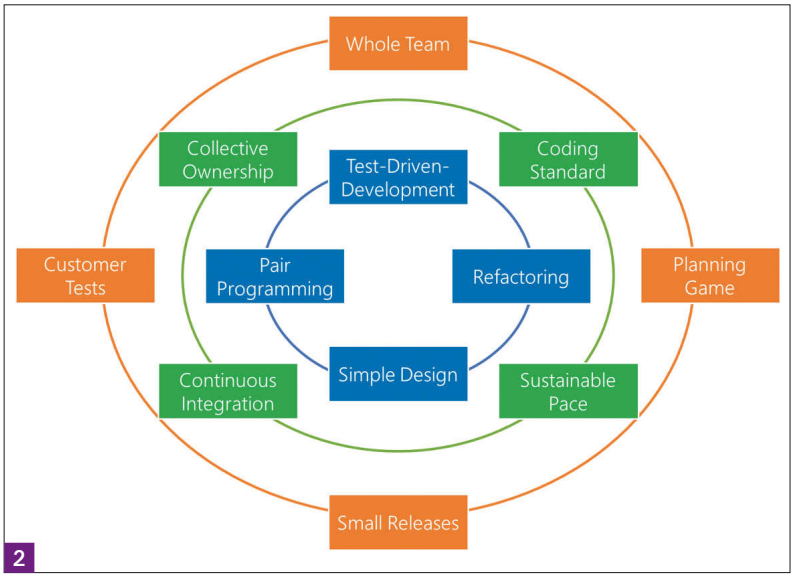
L'XP a suggéré de responsabiliser tous ces acteurs, en imposant des règles telles que la disponibilité obligatoire du client sur le site pour échanger avec l'équipe tout au long du développement, ou l'écriture de tests automatisés par les développeurs et l'approche du "Test Driven Development".

Grâce à plusieurs autres pratiques d'XP, tels que l'écriture obligatoire des user stories, l'appropriation collective du code, l'utilisation de métaphores, ou encore les livraisons fréquentes, le développement logiciel est désormais devenu transparent d'un bout à l'autre et tous ses acteurs y sont gagnants.

Le second objectif d'extreme programming est de créer des logiciels de qualité, ayant beaucoup moins de défauts que la norme de l'époque. Pour cela, on retrouve les bonnes pratiques de l'ingénierie dans le cœur de cette méthode, mais aussi une bonne organisation du projet, mêlée à une parfaite communication inter et intra équipes.

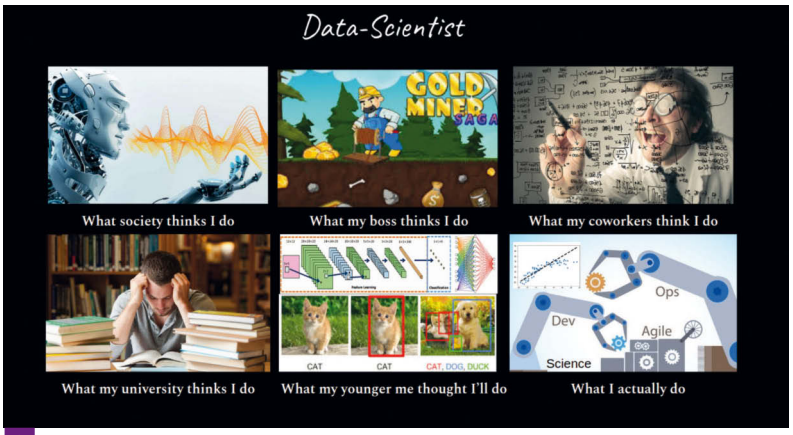
On peut représenter l'ensemble de ces pratiques avec trois cercles concentriques. Le cercle intérieur est au coeur de l'approche XP, il s'agit des bonnes pratiques d'ingénierie logicielle qui sont réalisables à l'échelle individuelle, comme l'écriture de tests unitaires. Le cercle intermédiaire regroupe les pratique à l'échelle d'une équipe, telle que pair programming. Enfin le cercle externe représente les pratiques liées à l'organisation et au management du projet. Avec l'évolution des approches agiles ces 20 dernières années, cette troisième catégorie de pratiques propres à XP peuvent être remplacées par les pratiques d'autres méthodes agiles telles que Scrum ou Kanban, sans nuir à l'effet positif de l'ensemble.

Cette courte introduction à l'XP étant terminée, si le sujet vous intéresse et que vous souhaitez en savoir plus, je vous recommande ces deux vidéos : *introduction à l'XP* de *Sylvain Fraisse*(1) et le talk de *Rachel Davis* sur *l'XP au 21eme siècle*(2), où elle présente un XP qui a évolué et qui s'est adapté aux cycles de développement modernes. 2



2

Source : Master of Project Academy https://masterofproject.com/blog/3543/extreme-programming



3

Data Science sur les épaules du géant XP

A ce stade vous devez certainement vous demander : tout ceci a l'air super intéressant et vraiment puissant, mais quel est le rapport avec la Data-Science, qui n'a ni les même caractéristiques ni le même cycle de production que l'ingénierie logicielle traditionnelle ? Et vous avez bien raison, car le développement de solutions de Data Science est plus complexe que le développement des logiciels traditionnels, et ceux qui ont développé au moins une fois une solution de machine learning end-to-end savent de quoi je parle (je vous conseille le début de la conférence de *Mathieu Zackaria*(3), où il décrit plus en détail les étapes qui augmentent la complexité des projets Data Science). Et donc, si l'XP a permis de faciliter le développement des logiciels traditionnels, tout en améliorant la qualité des livrables, nous avons d'autant plus de raisons d'expérimenter ses pratiques pour améliorer le développement des projets en data-science. Nous n'allons évidemment pas copier-coller toutes les pratiques d'XP, et l'extrême data science aura ses particularités, liées aux incertitudes du data-mining et du machine-learning, mais pour mener à bien les projets data-science, nous aurons beaucoup plus de facilités à nous reposer sur les épaules du géant XP.

Actuellement, dans la plupart des entreprises, les data-scientists créent des modèles de machine learning qui ne sont pas prêts pour la production, et c'est aux développeurs/data-engineers de les implémenter ainsi que de les mettre en production. La solution Data-Science fait donc plusieurs aller-retours entre les développeurs et les data-scientists, chacun travaillant en silo. Ajoutez à cela les allers-retours entre les data-scientists et les data-engineers, entre les data-engineers et les devops, ceux entre les data-scientists et les product-owners, et on se retrouve dans l'enfer de la période pré-XP.

Si l'on en revient au tweet de François Chollet, deux réponses dans le fil de la discussion nous montrent comment la data-science pourrait être perçue différemment dans l'entreprise. Dans une réponse(4), Leonardo Foderaro raconte que, souvent, les collègues et les managers ont peur de la complexité du machine learning, ce qui les empêche de découvrir leur potentiel, alors que Don Esteve pointe(5) les situations où les managers pensent que le machine learning pourrait magiquement résoudre tous les problèmes sans prendre en compte ses limites et ses coûts. Et ces deux tweets ne montrent que la partie émergée de l'iceberg.

La partie submergée de l'iceberg

Dans ce même "What I actually do", j'ai essayé de regrouper les perceptions les plus marquantes et influentes du métier de data-scientist. Sur le terrain, ces perceptions, qui peuvent sembler innocentes voir rigolotes, créent beaucoup de dégâts dans l'intégration de la data-science. 3

La société imagine que les développeurs IA créent des robots humanoïdes destinés à conquérir le monde et exterminer les humains, ou bien résoudre, comme par miracle, tous les problèmes de l'humanité. Et nos utilisateurs, nos managers et nos collègues, qui font partie intégrante de cette société, héritent de ce biais cognitif. Nous pouvons facilement différencier les comportements des chefs d'entreprise et de nos collègues.

Les chefs d'entreprise pensent que la data-science est la poule aux œufs d'or et que les data-scientists ont cette capacité surnaturelle

(1) https://bit.ly/2qJHYsM

(2) https://bit.ly/2NBil00

(3) https://bit.ly/2Cxp37t

(4) https://bit.ly/2NA3bhZ

(5) https://bit.ly/2X5gji0

à trouver dans les données des connaissances précieuses et monétisables sur le marché. Souvent c'est la première motivation de ces chefs d'entreprise pour intégrer la data-science, et ils ne font pas suffisamment d'efforts pour clarifier les besoins et travailler avec leurs équipes pour mieux définir les moyens d'y satisfaire.

Les collègues (développeurs, expert métiers, responsables fonctionnels) quant à eux prennent souvent les data-scientists pour des savants fous, et parler de data-science est forcément la porte vers le domaine sombre et complexe des mathématiques. Bien entendu, cela n'aide en rien l'intégration des solutions de data-science dans les produits.

En fin de compte, les data-scientists travaillent en isolation, et même si cela semble simple de percer cette bulle d'isolation, personne n'ose le faire.

En finir avec l'isolation des data-scientists dans la société

C'est à ce moment précis que **l'Extrême Data-Science nous donne le courage et surtout des méthodes concrètes** pour percer cette bulle, grâce à l'une des valeurs principales de l'XP : la communication.

Vulgarisation

La vulgarisation n'est pas une tâche évidente, d'autant plus si les data-scientists sont les seuls à s'en préoccuper. Il est difficile d'expliquer des principes complexes, surtout lorsque l'on ne connaît pas le niveau de connaissances de ses interlocuteurs. Les tensors, le clustering, les régressions linéaires - beaucoup de concepts semblent aux data-scientists extrêmement simples et inutiles à vulgariser, alors que la plupart des non-data-scientists les trouvent extrêmement abstraits et complexes.

Un exercice qui a fait ses preuves pour faciliter cette vulgarisation est la mise en place d'ateliers collaboratifs, impliquant des data-scientists et d'autres métiers, visant à créer des posters expliquant un concept ou un algorithme de machine learning. Ces posters pourront ensuite être consultés par les autres personnes de la société, favorisant le partage de connaissance, tout en réduisant l'isolation des data-scientists, avec un exercice ludique. 4

Cela rejoint une des pratiques de l'XP : utiliser des métaphores pour garder tous les acteurs informés sur les détails du développement du logiciel sans tomber dans l'utilisation des termes techniques, qui peuvent ne pas être compris par tout le monde.

Engagement

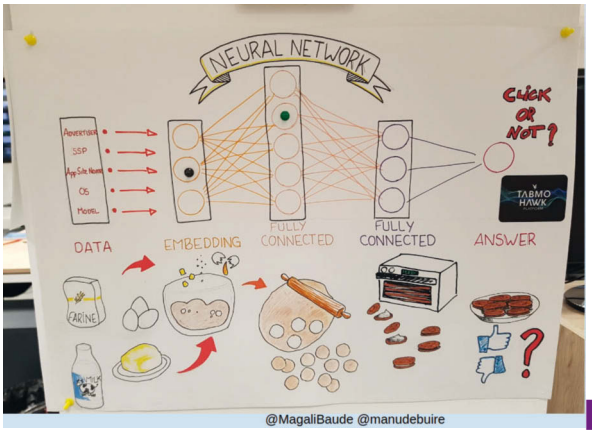
Une fois que les sujets complexes le sont un peu moins aux yeux de vos collaborateurs, il est essentiel de leur demander de s'engager. Un bon exercice à réaliser pour confirmer la réalisation de la première étape de vulgarisation est un jeu de projections des domaines de l'IA sur vos différents métiers. Après avoir expliqué (via la vulgarisation) les différents domaines de l'IA qui pourraient avoir du potentiel pour votre société, demandez à vos collègues (commerciaux, développeurs, supports, experts métier), de trouver des cas d'utilisation correspondant à ces domaines dans leurs métiers. 5

Proposez ensuite de catégoriser les idées recueillies avec des labels: nécessaire à faire, important, compliqué, facile, coup de cœur… À la fin, organisez un débat/debrief, qui sera l'occasion d'éclaircir pourquoi certaines idées sont meilleures que d'autres.

Collaboration permanente

La collaboration était déjà une composante fondamentale des deux premières actions, mais si elle est exercée occasionnellement, il ne faut pas s'attendre à des résultats remarquables. C'est pour cela que l'objectif de la troisième action est de mettre **un maximum d'efforts dans la collaboration au quotidien**. Voici quelques solutions, simples à mettre en place quel que soit le projet :

- **les réunions quotidiennes (stand-up daily)** auxquelles participent les développeurs et les data-scientists. Le but de ces réunions est de mettre en commun les avancées et les blocages de chacun, de s'organiser pour que l'équipe avance au même rythme, et surtout pour que personne ne reste bloqué plus d'une journée sans qu'un coéquipier ne l'accompagne dans la résolution de ce blocage. La situation que l'on souhaite à tout prix éviter, où les data-scientists travaillent avec un rythme décalé et où l'équipe n'est pas au courant de leurs besoins et de leurs résultats, peut être évitée grâce à ces stand-up daily meetings.
- **Programmation en binôme (pair programming)** entre le développeurs et le data-scientist est la solution quasi-magique qui permet de transformer le travail d'une équipe au niveau technique mais aussi au niveau humain. Les développeurs logiciels et les data-scientists abordent le code de manière différente : les développeurs pensent à la modularité du code et portent une attention toute particulière aux bonnes pratiques du développement, alors que les data-scientists pensent surtout à la solution d'un point de vue machine learning et gardent en tête toutes les opportunités et risques propres au machine learning. Développer ces solutions en binôme permet de prendre le meilleur de deux mondes, et donc de s'assurer que la solution data-science soit parfaite, tant au niveau de la programmation logiciel qu'au niveau métier du machine learning. D'autre part, le fait que la solution soit codée "à quatre mains" est une solution naturelle pour accroître **l'appropriation collective du code.** Cela évite donc les dérapages éventuels d'un profil envers d'autres, puisque le code est créé collectivement et sa bonne réalisation devient la responsabilité de tous.
- **Les ateliers avec les utilisateurs** ne sont peut-être pas coutume parmi les data-scientists, mais ils devraient l'être. Il est vrai



4

5

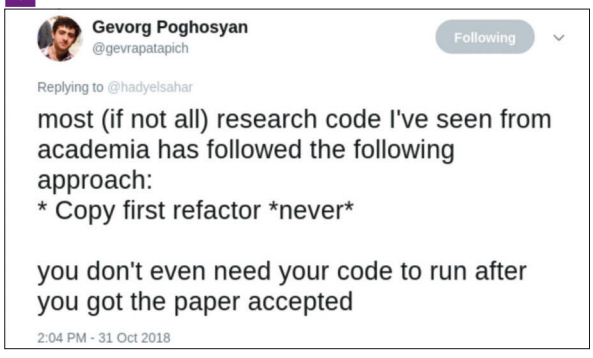
que l'on passe souvent par les chefs de produit ou les chefs de projet et les designers UX pour récupérer les retours et les besoin des utilisateurs, mais dans les projets data-science, qui sont souvent très techniques et cachent beaucoup d'incertitudes, mais aussi une multitude d'opportunités techniques, la communication directe entre le data-scientist et l'utilisateur est cruciale pour le succès des projets. Mieux connaître l'utilisateur permettra aux data-scientists d'anticiper et de résoudre plus rapidement les problèmes et les défis techniques, qui sont souvent négligés à cause du manque de liens utilisateur/data-scientist. Les ateliers de design thinking et de Game Storming sont des approches idéales pour permettre une collaboration efficace entre ces profils très différents.

Ces pratiques qui permettent une collaboration permanente sont rapides à mettre en place, mais demandent une organisation du travail qui facilite la communication intra et inter équipes.

On peut souligner **deux démarches importantes** qui assurent le succès de telles organisations :

- **Diviser le projet en petites itérations.** Cette démarche est avantageuse pour les data-scientists aussi bien que pour le client, car la première mise en production de la solution sera plus rapide. D'autre part, les data-scientists auront des repères beaucoup plus solides, car les petites itérations leur éviteront de partir dans une recherche sans fin de la solution parfaite. Ils devront au contraire réfléchir très tôt à comment livrer et tester leur solution en production, mais aussi comment la faire évoluer par la suite.
  Attention, car certains data-scientists, avec une expérience académique, bloquent sur l'idée de diviser le projet en petites itérations, pensant que cela limiterait leur créativité. Mais comme nous ne sommes plus dans un cadre académique, et l'objectif n'est plus de faire la recherche théorique mais d'avoir des résultats en production et de satisfaire l'utilisateur final, il est donc impératif pour ce type de profils de s'adapter au nouveau rythme d'industrie.
- Pour pouvoir diviser le projet en itérations, **il est indispensable de rédiger**, et ce dès le démarrage du projet, **les scénarios utilisateurs** (user stories). Nous pouvons adapter la technique d'implémentation, ou améliorer la compréhension du besoin d'une itération à l'autre, mais le scénario utilisateur sera la base de toute la planification et la répartition des tâches, l'estimation du travail et le choix de la méthodologie scientifique. Les user stories doivent décrire des fonctionnalités dont l'implémentation est peut être implémentée en une itération. Pour les data-scientists qui résistent à l'agilité, les user-stories deviendront un argument majeur, car ils mettront au centre le besoin concret des utilisateurs et la nécessité d'y répondre rapidement.

6



Bonnes pratiques d'ingénierie logiciel en data-science

Beaucoup d'entreprises souhaitent mettre en place des méthodes agiles, mais peu y arrivent.. Quel est donc leur secret ? Les sociétés qui réussissent leur transformation agile, en plus de réfléchir aux problématiques de gestion de projet, cherchent également à améliorer leurs pratiques d'ingénierie logicielle, autrement dit le **software craftmanship** ou encore l'artisanat du logiciel, qui est au cœur de l'XP. Ce tweet est un excellent témoignage du manque de bonnes pratiques parmi les data-scientists, qui, pour la plupart, viennent du monde de la recherche académique. 6

Le respect des standards du code, telles que l'écriture obligatoire des tests unitaires, l'utilisation de gestionnaire de version tel que Git, la production de code maintenable, couplé avec les revues de code et la programmation en binômes, permettrait d'améliorer substantiellement la qualité des solutions Data Science, et réussir la transformation agile.

Valeurs de l'Extrême Data Science

Les techniques, les méthodes et les approches évoluent d'une année à l'autre, mais ce qui reste intact et assure la bonne direction de ces évolutions, ce sont les valeurs de l'XP. L'Extrême Data-Science réinterprète ces valeurs comme nous avons pu le voir tout au long de cet article.

La communication : elle est au cœur de l'Extrême Data Science, elle permet de casser les barrières entre les différents métiers, et de mettre tout le monde sur la même longueur d'onde.

La simplicité : elle assure que la solution data-science sera maintenable et évolutive, et réduit *de facto* les risques d'over-engineering ou les expérimentations non-nécessaires.

Le Feedback : il est présent tout au long du développement de la solution data-science, c'est-à-dire lors de la programmation en binôme, lors des revues de codes, lors de stand-up daily et des rétrospectives, ainsi que lors des ateliers avec les utilisateurs.

Le respect : il permet d'avoir un feedback constructif et bienveillant, une confiance et une communication efficientes à tous les niveaux, entre utilisateurs, clients, développeurs et data-scientists.

Le courage : il est essentiel pour conduire le changement dans la culture Data-Science, y intégrer l'approche Extrême et respecter ses valeurs du début à la fin.

Si l'on souhaite livrer en production des solutions data-science de qualité, qui répondent aux besoins des utilisateurs, qui sont rentables pour le client, et agréables à réaliser pour les développeurs et data-scientists, l'Extrême Data-Science est LA MÉTHODE à mettre en place.

Cela devrait être un jeu d'enfant de convaincre les acteurs du projet à la mettre en place, puisque, comme nous l'avons vu dans l'article, chaque acteur a énormément à y gagner. En premier lieu, le client commence à en tirer profit très tôt, étant donné que la première version est mise en production rapidement, et il peut sans problème demander de nouvelles fonctionnalités qui seront intégrées facilement, car la base de code est évolutive grâce aux bonnes pratiques de l'ingénierie logiciel et data-science. Les data-scientists et les développeurs, quant à eux, sont sereins et épanouis dans leur travail, car le code ne représente plus une boîte noire pour les uns, et un spaghetti pour les autres. Alors n'attendez plus et hackez votre culture d'entreprise avec l'Extrême Data Science ! •

SQLI DIGITAL EXPERIENCE

Marion Rocher
Ingénieur Front-end
Développeuse ayant travaillé sur des projets d'envergures différentes, la plupart du temps front-end et aussi en fullstack. Sa technologie préférée ? Angular 8.

niveau 100

Git par la pratique

Lors de mon arrivée sur divers projets, je me suis souvent heurtée à des gestions de Git différentes les unes des autres. Pourtant, elles étaient toutes plus ou moins intéressantes. On a dû toutes une fois vivre cela : soit l'organisation du Git est optimisée, soit rien ne va et la puissance du versioning n'est pas utilisée. Dans ce dernier cas, l'envie d'améliorer l'utilisation de Git est confrontée aux retours négatifs des autres développeurs. Les sujets qui contrebalancent cette envie sont souvent importants, comme la gestion de la dette technique ou le manque de test unitaire. Or, la mauvaise utilisation de Git peut augmenter la difficulté de leur traitement de façon exponentielle.

Les trucs et astuces

Il existe quelques astuces dont la mise en place n'est pas chronophage pour les développeurs de toute équipe. Ce changement de pratique qui doit être accepté par tout le monde pour mener à une **utilisation optimisée de Git**.

La première astuce est la plus simple à mettre en place. Il s'agit de **définir une convention de nommage sur les commits**. Étant donné qu'ils seront gardés durant toute la vie du projet, il faut les rendre compréhensibles et concis. Je conseille d'utiliser la convention de nommage mis en place par Karma (https://karma-runner.github.io/4.0/dev/git-commit-msg.html) qui se résume à cette structure : **<type>(<scope>): <sujet>**. Le scope correspond soit au composant modifié soit, dans certains, cas au numéro de la tâche réalisée. Le sujet est une description succincte de la tâche réalisée. Le type, quant à lui, correspond à ce qui a été réalisé dans le commit :

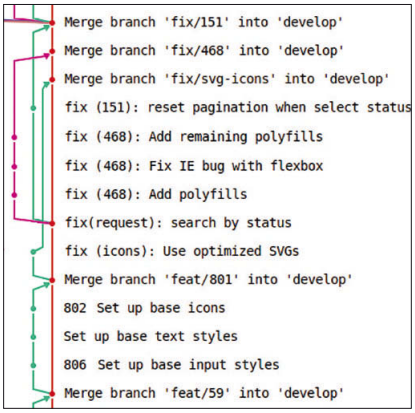
- **feat** : une fonctionnalité pour l'utilisateur ;
- **fix** : une rapide correction de bug ;
- **refactor** : une modification d'une partie du code pour optimisation ;
- **test** : ajout ou modification de test dans l'application ;
- **docs** : changement dans la documentation ;
- **style** : mise en forme de code, oubli de point-virgule ;
- **chore** : modification répétitive de code ;
- **build** : changements qui affectent le système de build ou des dépendances externes (npm, make…) ;
- **ci** : changements concernant les fichiers et scripts d'intégration ou de configuration (Travis, Ansible, BrowserStack…) ;
- **perf** : amélioration des performances.

Les buts de cette convention est de **simplifier la navigation dans l'historique Git** et de pouvoir en **générer facilement un changelog**. En effet, les commits ayant tous la même structure, il est plus aisé de retrouver le dernier **fix** ou le dernier **refactor** qui aurait pu engendrer des régressions. Cette astuce permet aussi, lors d'un **git blame**, de comprendre plus rapidement pourquoi la modification a été faite et dans quel but, grâce à la description du sujet traité. Si vous ne connaissez pas la commande git blame elle permet de savoir qui a fait la modification, quand et dans quel commit. 1

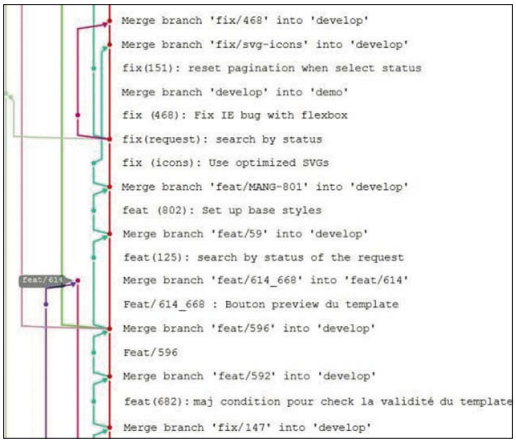
La deuxième astuce, pour récupérer les modifications d'une autre branche, consiste à **utiliser la fonction rebase au lieu de la fonction merge**. Dans la majorité des cas, les développeurs préfèrent utiliser le **merge** car la gestion des conflits est faite une seule fois et il y a moins de chance de perdre son code. Le souci lié à cela est que l'on perd le principe d'une branche qui ne contient que notre travail. En effet, en « mergeant » une autre branche, disons **develop** dans notre branche de travail, on incorpore les **commits** de **develop** dans celle-ci, alors que ce n'est pas réellement ce que l'on souhaite. En « rebasant » notre branche sur **develop**, on accède aux modifications faites sur celle-ci, sans pour autant, les incorporer à notre branche. 2 3

De plus, une fois le **merge** fait dans la branche souhaitée, on perd la linéarité de l'historique de **Git**. En regardant les deux figures ci-dessus, on se rend compte que l'image de la figure 2 est beaucoup plus lisible que celle de la figure 3. Sur la figure 2, en jetant un coup d'œil à la liste des commits sur l'arbre, on comprend facilement le travail qui a été réalisé.

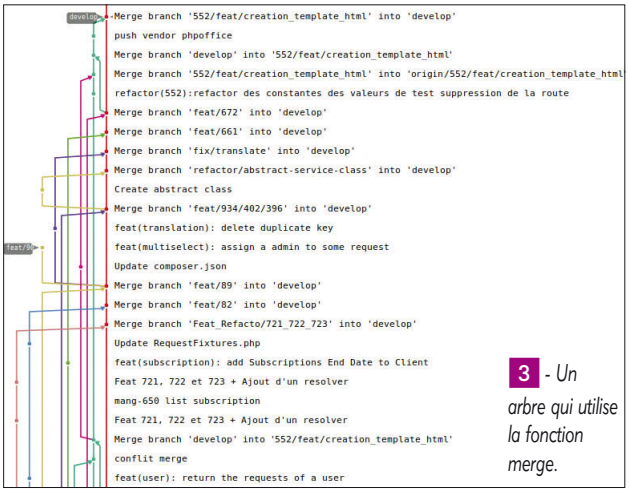
A l'inverse, pour comprendre le travail réalisé sur la figure 3, il faut étudier l'arbre pour trouver quelle branche est



1 - *On voit la différence entre des commits nommés comme Karma et ceux sans convention de nommage. La visibilité en est grandement impactée.*



2 - *Un arbre qui utilise la fonction rebase.*



3 - *Un arbre qui utilise la fonction merge.*

« mergée » dans laquelle, pourquoi telle branche a été fusionnée avec celle-ci, etc. Cette étude devient vite une importante perte de temps lors du développement de fonctionnalité.

Lors d'un bug en production, il est beaucoup plus simple de relivrer l'application si on a fait des rebases que si l'on a fait des merges. En effet, comme on peut le voir sur les images précédentes d'arbres **Git**, la liste des **commits** sera beaucoup plus simple à analyser. Ainsi, on trouvera assez facilement le commit qui a créé la régression ou le bug. On pourra donc facilement revenir en arrière. Si au contraire on utilise des merges, il faudra démêler l'entremêlement de branches pour savoir quel commit a entraîné la régression.

Les différentes raisons, évoquées précédemment, poussent à utiliser la fonctionnalité de **rebase** sur **Git**. Cette dernière permet de garder notre branche de travail dans son périmètre d'utilisation. Elle permet aussi de modifier la base de notre branche. Ainsi, on peut se mettre sur la branche d'un collègue pour récupérer des modifications en cours de développement. La fonctionnalité de rebase peut être utilisée à l'infini et ainsi permettre aux différents développeurs d'un projet de récupérer des modifications sur n'importe quelle branche du dépôt **Git**.

Les commandes indispensables

Afin de mettre en place ces différentes astuces, il faut connaître plusieurs commandes **Git** indispensables. Elles vous permettront de comprendre l'utilisation simplifiée de Git dans votre IDE. Souvent, les développeurs préfèrent optimiser leur utilisation de **Git** au strict minimum, alors que l'**on peut faire tellement plus avec deux ou trois commandes Git supplémentaires**. Je pense qu'avec les commandes **Git** ci-dessous, on peut gérer ses branches et ses commits pour avoir un résultat beaucoup plus satisfaisant, tout en conservant la cohérence de l'arbre. Voici donc les commandes que j'utilise le plus :

- **git pull origin {branchName} {options}**
- **git push origin {branchName} {options}**
- **git add {option}**
- **git status**
- **git commit –m {message}** ou **git commit –amend**

```
pick 84874d7 first change
pick ebaca4a second change
pick 729e180 third change

# Rebase 9383d56..729e180 onto 9383d56 (3 commands)
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
~
```

4 - *Éditeur nano qui s'ouvre lors du rebase interactif*

- **git rebase origin/{branchName} –i**
- **git log**

Les quatre premières commandes sont les plus connues et les plus utilisées sur **Git**. Je ne m'attarderai pas dessus. La cinquième est une commande aussi très utilisée, seulement ses options sont souvent ignorées. Pour ma part j'utilise beaucoup le **git commit –amend**. Cette commande **Git** permet d'ajouter vos dernières modifications au dernier commit ajouté à la branche courante. Attention, une fois cette commande faite, il ne faut surtout pas oublier d'avoir recours à **git push -force**, sinon vous aurez deux commits sur votre branche au lieu d'un. Je l'utilise régulièrement, en fin de journée, pour mettre à jour mon code sur le dépôt distant du projet. En faisant cela, on évite une situation que j'ai déjà vécue : la perte de tout mon travail si mon ordinateur ne redémarre pas le lendemain.

La commande **git rebase origin/{branch Name} -i**, présentée ci-dessus, ne devrait plus avoir de secret pour vous, à part peut-être l'option interactive. Cette option permet, avant de commencer le **rebase**, de pouvoir choisir les **commits** que vous voulez embarquer dans celui-ci. Celle-ci est indispensable lorsque vous changez la racine de votre branche. En réalisant cela, il est possible que vous embarquiez des commits qui ne sont pas les vôtres mais qui ne sont pas présents dans la nouvelle branche. Si l'option est activée, un éditeur (nano, vim, …) identique à celui de la figure 4 s'ouvrira, et vous pourrez alors choisir les commits à prendre en compte. Dans le cas on l'éditeur que vous utilisez ne vous convient pas il est toujours possible de le changer avec **git config --global core.editor <emacs/nano/vim...>**.

Dans mon utilisation quotidienne, j'active toujours cette option. Elle me permet de savoir exactement quel commit j'embarque dans mon rebase mais aussi de maîtriser de A à Z la réalisation de celui-ci. 4

La commande git log est pour moi la plus importante et à la fois la plus simple à utiliser. Elle permet de voir l'arbre de notre branche à un instant T, de savoir quel est le dernier commit poussé, s'il est sur le dépôt distant ou local… Grâce à cette commande, je suis en mesure de vérifier si j'ai réussi mon **rebase,** ou juste avant de le commencer, de vérifier sur quelle branche je suis basée. Elle me permet également lorsque j'arrive sur le poste d'autres développeurs de **savoir dans quel état est leur arbre Git**.

Le mot de la fin

En vous apportant ces conseils, je n'affirme pas détenir la vérité absolue. Je suis la première convaincue que je rencontrerai au fil de ma carrière de nouvelles utilisations de **Git** et que ma vision évoluera. J'ai parcouru un long chemin depuis les premiers projets où je n'appliquais pas moi-même, ou que très peu, les conseils que je vous transmets aujourd'hui. Cette vision a pu être acquise en participant à plusieurs projets avec des leaders techniques différents. Ils avaient chacun des manières de travailler différentes. C'est maintenant à moi de vous transmettre ce que j'ai pu apprendre, et d'essayer d'inciter toujours plus de monde à exploiter au maximum les capacités de **Git**.

Aurélie Vache
Cloud Dev(Ops) chez Continental
Duchess France & DevFest Toulouse Leader
Toulouse Data Science core-team &
Marraine Elles bougent
@aurelievache

Partie 1

Les tests d'intégration facile avec Venom !

Comme vous le savez, "le gras c'est la vie" mais lorsque l'on travaille dans le développement d'applications/sites web, et ben le diction "les tests c'est la vie" se vérifie également et nous allons voir dans cet article qu'il n'y a pas que les tests unitaires et que créer et automatiser des tests d'intégration cela peut être un jeu d'enfant ;-).

niveau 200

Des tests d'intégration, pour quoi faire ?

Vous avez participé au développement d'un site ou d'une application, vous avez, bien entendu, codé les tests unitaires et maintenant on vous "embête" avec les tests d'intégration, mais qu'est ce que c'est que ça encore ? Les tests d'intégration permettent de tester les fonctionnalités d'un système sur l'ensemble des composants construits via des scénarios utilisateurs.

Avec les tests unitaires on vérifie le bon fonctionnement d'une partie précise d'une application/d'un soft, d'une unité alors que les test d'intégration, comme son nom l'indique, permettent de tester l'ensemble des composants dans des scénarios fonctionnel :

- Est-ce que le site http://www.monsupersite.com répond (code de retour 200) ?
- Est-ce qu'il répond en moins d'1 seconde ?
- Est-ce que l'appel à l'URL http://www.monsupersite.com/monsuperjson.json contient un tableau composé de deux éléments composé de la clé "id" et de la clé "description"
- …

Dans la phase de tests, les tests d'intégration de situent entre les tests unitaires et les tests de validation.

L'objectif des tests d'intégration est de détecter les erreurs qui n'ont pas pu être détectées (car non testées) lors de la phase des tests unitaires.

Venom

Pour écrire et exécuter nos tests d'intégration, nous allons utiliser Venom. Venom est un outil créé et mis en open source par OVH : https://github.com/ovh/venom

L'outil, écrit en Go (et oui encore un ;-)), date de 2017, ne possède pas énormément de stars sur Github, a quelques bugs, mais suite à une présentation à Devoxx France cette année, nous l'avons trouvé très prometteur et avons décidé de l'utiliser pour nos tests d'intégration.

En pratique, Venom run des exécuteurs (script, requête HTTP, ssh, smtp …) et applique des assertions/des affirmations.

L'outil possède une liste d'exécuteurs existant : dbfixtures, exec (executeur par défaut), http, imap, kafka, ovhapi, readfile, redis, smtp, ssh, web et grpc.

Pour ma part, j'en utilises surtout deux : exec et http, qui couvrent déjà une grande majorité des cas à tester.

Mais si cette liste ne vous suffit pas, vous pouvez créer vos propre exécuteurs ;-).

L'outil possède une liste d'assertions/d'affirmations, voici une liste **non exhaustive** :

- ShouldEqual
- ShouldNotEqual
- ShouldAlmostEqual
- ShouldBeNil
- ShouldNotBeNil
- ShouldBeTrue
- ShouldBeFalse
- ShouldBeZeroValue
- ShouldBeGreaterThan
- ShouldBeLessThan
- ShouldContain
- ShouldNotContain
- ShouldContainKey
- ShouldNotContainKey
- ShouldHaveLength
- ShouldStartWith
- ShouldEndWith
- ShouldNotExist
- …

Liste complète : https://github.com/ovh/venom

Venom exécute des tests d'intégration qui doivent être listés dans un fichier au format YAML.

Dans ces scénarios de tests, Venom met à disposition des variables communes :

- {{.venom.testsuite}}
- {{.venom.testsuite.filename}}
- {{.venom.testcase}}
- {{.venom.teststep.number}}
- {{.venom.datetime}}
- {{.venom.timestamp}}

Une des fonctionnalités de Venom est que nous pouvons utiliser la sortie d'un cas de test en tant que paramètre ou entrée d'un cas de test suivant.

Par exemple, nous pouvons interroger un service web qui renvoie un JSON contenant deux éléments : un ID et un user name par exemple, puis ré-utiliser l'user ID en tant que paramètre de l'interrogation d'un deuxième web service. Nous verrons plus en détail et en exemple ce cas là dans la suite de l'article.

Installation

On m'a fait remarqué que je mettais très souvent une certaine expression dans (tous ?) mes articles, donc je ne vais pas déroger à la règle et l'utiliser encore une fois :

"Commençons par le commencement", commençons par installer venom sur notre machine. Pour ce faire, le pré-requis est d'avoir installé Golang sur votre machine.

Go, go, go !

La première étape consiste, si vous ne l'avez pas déjà fait, à installer Go sur votre machine. Pour cela vous pouvez suivre la procédure d'installation https://golang.org/doc/install sur le site officiel ou bien passer par GVM : https://github.com/moovweb/gvm.

GVM est un version manager pour Go, très pratique, qui permet de mettre à jour votre version de Go en spécifiant quelle version vous désirez.

Installation :

• Pour bash :

```
bash < <(curl -s -S -L https://raw.githubusercontent.com/moovweb/gvm/master/binscripts
/gvm-installer)
```

• Pour zsh :

```
zsh < <(curl -s -S -L https://raw.githubusercontent.com/moovweb/gvm/master/binscripts
/gvm-installer)
```

Utilisation :

```
$ gvm
Usage: gvm [command]

Description:
 GVM is the Go Version Manager

Commands:
...
```

La commande de GVM qui va nous intéresser tout particulièrement est la commande **gvm install**, elle s'utilise comme ceci :

```
$ gvm install [version] [options]
```

Installation de Go :

```
$ gvm install go1.12.9 -B
$ gvm use go1.12.9 --default
```

Dans votre .zshrc ou .bashrc veuillez à bien setter vos variables d'environnement **$GOROOT** et **$GOPATH**. Voici un exemple :

```
[[ -s "$HOME/.gvm/scripts/gvm" ]] && source "$HOME/.gvm/scripts/gvm"
export GOPATH=$HOME/go
export GOBIN=$GOPATH/bin
export                                                    PATH=${PATH}:$GOBIN
```

Ça y est, Go est installé, son gestionnaire de version également, il est temps de rentrer dans le vif du sujet.

Venom

Nous pouvons à présent installer l'outil Venom sur notre machine. Voici les commandes à exécuter :

```
$ go get github.com/ovh/venom
```

Le binaire de venom est maintenant dans le répertoire bin/ de votre **$GOPATH**, qui est lui même dans votre PATH, donc utilisable directement.

```
$ go install github.com/ovh/venom/cli/venom
```

Et, c'est tout … ;-). L'outil est installé et prêt à l'emploi !
Pour le tester, on va exécuter la commande help de l'outil :

```
$ venom -h
Venom - RUN Integration Tests

Usage:
 venom [command]

Available Commands:
 help      Help about any command
 run       Run Tests
 template  Export a TestSuite Template
 update    Update venom to the latest release version: venom update
 version   Display Version of venom: venom version
...
```

Appréhendons l'outil avec un exemple

Généralement lorsque l'on découvre un nouvel outil, pour essayer de comprendre comment cela fonctionne on va copier coller un morceau de code sur un article, un tuto, une page sur internet, sur stackoverflow ou bien sur github, avec Venom il y a une commande toute simple et sympatique, la commande *template*, qui vous permet d'avoir un template de base fonctionnel :

```
$ venom template
name: Title of TestSuite
testcases:
- name: TestCase with default value, exec cmd. Check if exit code != 1
 steps:
 - script: echo 'foo'
   type: exec
- name: Title of First TestCase
 steps:
 - assertions:
   - result.code ShouldEqual 0
   script: echo 'foo'
 - assertions:
   - result.systemout ShouldNotContainSubstring foo
   script: echo 'bar'
vars: {}
```

Cool, un template de base, on va donc commencer à découvrir Venom grâce à cette première suite de tests (testsuite) qui contient 2 cas de test (testcase).

```
$ venom template > sample.yml
```

Et pour exécuter notre suite de test il suffit d'appeler la commande *run* et de lui passer en paramètre le path vers votre fichier yml, si dans votre répertoire vous avez plusieurs fichiers yml contenant des suites de test. Ou bien il vous suffit d'exécuter uniquement la commande *venom run* (sans argument), si il n'y a qu'un testsuite dans votre folder :

```
$ venom run
SUCCESS sample.yml                    21.472181ms

Total:2 Duration:27.46326ms
OK:2
KO:0
Skipped:0
TestSuite:1
TestCase:2
```

La sortie du script répond SUCCESS donc nous pouvons en convenir que l'exécution des tests à bien réussi, cool ;-), mais décortiquons à présent cette suite de test.

```
name: Title of TestSuite (1)
testcases: (2)
- name: TestCase with default value, exec cmd. Check if exit code != 1 (3)
 steps:
 - script: echo 'foo' (4)
   type: exec
- name: Title of First TestCase
 steps:
 - assertions: (5)
   - result.code ShouldEqual 0
   script: echo 'foo'
 - assertions:
   - result.systemout ShouldNotContainSubstring foo
   script: echo 'bar'
vars: {}
```

(1) Une suite de test (testsuite) dois avoir un nom
(2) Elle est composée d'un ou plusieurs cas de test (testcase)
(3) Un cas de test est composé d'un name et d'une suite de steps
(4) Une step est composée d'un type (exec par défaut), d'un script (si type = exec) ainsi que d'assertions (de tests à effectuer)
(5) Les assertions (affirmations) ce sont les tests que nous allons faire pour valider que notre application/site/script fonctionne comme attendu

Concrètement, si l'on suit cet exemple, notre testsuite est composée de deux testcases qui ont pour but de :

- 1) exécuter echo 'foo'
- 2) exécuter echo 'foo' et valider si l'exit code est égal à 0 (ce qui signifie que tout le script s'est bien exécuté avec succès)
- 3) exécuter echo 'bar' et vérifier que la sortie de l'exécution du script (*result.systemout*) ne comporte pas la chaine de caractère "foo"

Bon, ok, rien de bien intéressant mais cela aide pour commencer à comprendre comment écrire des tests d'intégration ;-).

Ecrivons nos tests d'intégration

Nous allons à présent écrire une suite de tests d'intégration qui vont nous permettre de rentrer dans le vif du sujet, de voir plusieurs exécuteurs Venom et de nous rendre compte qu'il est bon de connaître quelques tips lorsque l'on fait du Venom pour éviter de perdre plusieurs heures sur un problème.

Commençons par créer un fichier yaml nommé "testsuite.yaml" :

```
name: IntegrityTest
version: "2"

testcases:
```

Le premier test que nous allons faire, dans cette suite de test, est de tester si le site que nous voulons tester (une API REST publique) :

- est accessible
- si sa réponse n'est pas vide
- si les éléments attendus sont bien dans sa réponse (qui est au format JSON)

```
name: IntegrityTest
version: "2"

vars:
url: https://dog.ceo/api

testcases:

- name: GetDogs
steps:
- type: http
  method: GET
  headers:
    Accept: application/json
    Content-Type: application/json
  url: "{{.url}}/breeds/list/all"
  retry: 1
  delay: 2
  assertions:
  - result.statuscode ShouldEqual 200
  - result.bodyjson ShouldContainKey message
  - result.bodyjson.message.bulldog.bulldog0 ShouldEqual "boston"
```

Dans ce petit exemple nous utilisons la notion de variable qui est à définir dans le bloc **vars** en haut du fichier yaml. Nous définissons une variable *url*.

Cette variable est appelable et réutilisable dans tout le fichier grâce au formalisme **{{.nomdemavar}}** . Une fois que nous avons fait l'appel à l'API, nous testons deux choses :

- que le résultat du code HTTP est égal à 200 (tout va bien)
- que le JSON que nous récupérons contient la clé message (c'est bel et bien le cas car le JSON commence comme ceci) : **{"message":{"affenpinscher":[],"african":[],"airedale":[],"akita":[],"appenzeller":[],"basenji":[],"beagle":[],"bluetick":[],"borzoi":[],"bulldog":["boston","english","french"...]...}}**

On ne change pas une équipe qui gagne, nous allons tester notre cas de test :

```
$ venom run testsuite.yaml
SUCCESS testsuite.yaml                    524.33801ms

Total:1 Duration:525.896185ms
OK:1
KO:0
Skipped:0
TestSuite:1
TestCase:1
```

Super, le test fonctionne ;-).

Concrètement, notre JSON (une fois formaté de manière lisible) se présente sous cette forme :

```
{
  "message": {
    "affenpinscher": [],
    "african": [],
    "airedale": [],
    "briard": [],
```

```
    "buhund": [
      "norwegian"
    ],
    "bulldog": [
      "boston",
      "english",
      "french",
      ...
    ],
    ...
  }
}
```

Histoire d'aller encore plus loin, nous allons maintenant explorer notre JSON afin de tester si dans notre JSON nous avons bien un attribut **bulldog** qui contient un array et que son premier élément soit égal à "bulldog".
Nous rajoutons donc une assertion qui test l'existence d'un élément "**boston**" en premier élément de l'array **bulldog** :

```
assertions:
- result.statuscode ShouldEqual 200
- result.bodyjson ShouldContainKey message
- result.bodyjson.message.bulldog.bulldog0 ShouldEqual "boston"
```

```
$ venom run testsuite.yaml
SUCCESS testsuite.yaml                    211.344661ms

Total:1 Duration:213.164227ms
OK:1
KO:0
Skipped:0
TestSuite:1
TestCase:1
```

Cool, notre cas de test fonctionne toujours.
A noter qu'une fonctionnalité sympa de Venom est que l'on peut réutiliser une valeur récupéré d'un résultat d'un cas de test, dans un autre cas de test. Ainsi si nous voulons récupérer la première sous race de bulldog récupéré ci-dessus, il suffira de l'appeler comme ceci :

```
{{.GetDogs.result.bodyjson.message.bulldog.bulldog0}}
```

Nous allons nous en servir pour faire un appel à l'API qui liste les images disponibles pour les **bulldogs** de type **boston** :

```
- name: GetBostonBulldog
steps:
- type: http
  method: GET
  url: "https://dog.ceo/api/breed/bulldog/{{.GetDogs.result.bodyjson.message.bulldog.bulldog0}}/images"
  retry: 1
  delay: 2
  assertions:
  - result.statuscode ShouldEqual 200
```

Une autre fonctionnalité sympa est l'écriture de cas de test qui exécute une CLI ou bien un script, exemple :

```
- name: GetDogName
steps:
- script: echo 'bulldog'
  assertions:
  - result.code ShouldEqual 0
```

Ce cas de test éxécute "echo 'bulldog" et test si le résultat du script est en succès ou pas (exit code égal à 0).
Si nous lançons un "venom run" sur nos trois cas de test, nous pouvons constater que la suite de test fonctionne toujours :

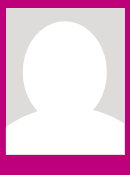
```
$ venom run testsuite.yaml
SUCCESS testsuite.yaml                    419.505045ms

Total:3 Duration:421.806366ms
OK:3
KO:0
Skipped:0
TestSuite:1
TestCase:3
```

Une information à savoir est que si on a le besoin d'exécuter un script sur plusieurs lignes, il y a une astuce à connaître pour que cela fonctionne. Le "multi line script" fonctionne avec ce formalisme pour l'exécuteur *script* :

```
- name: Says hi to readers
steps:
- script: |
    #!/bin/bash
    echo "salut les lecteurs" | base64
  assertions:
    - result.code ShouldEqual 0
    - result.systemout ShouldNotEqual ""
    - result.systemout ShouldEndWith "=="
    - result.systemerr ShouldEqual ""
```

Vous devez impérativement mettre un pipe "|".

La suite dans le prochain numéro.



Cécilia Bossard
Développeuse Java/Javascript chez Code Lutin

Yannick Martel
Développeur Java/Recruteur chez Code Lutin

Du Zéro-Déchet dans votre développement ?

niveau 100

Des études récentes montrent que 4% des gaz à effet de serre sont émis par notre consommation numérique, et elle s'accroît de 9% par an (source : The Shift Project - octobre 2018). Pour rester dans les chiffres, en 2008, la part d'Internet dans les émissions mondiales de gaz à effet de serre était de 2%, soit autant que les transports aériens mondiaux.

« Les opportunités du numérique étant précieuses, mieux les calibrer est essentiel pour en préserver l'utile. Être 'sobre' à l'échelle de notre société, c'est donc réinventer nos usages pour qu'ils soient compatibles avec les contraintes climatiques. »

Maxime Efoui-Hess, auteur du rapport, The Shift Project

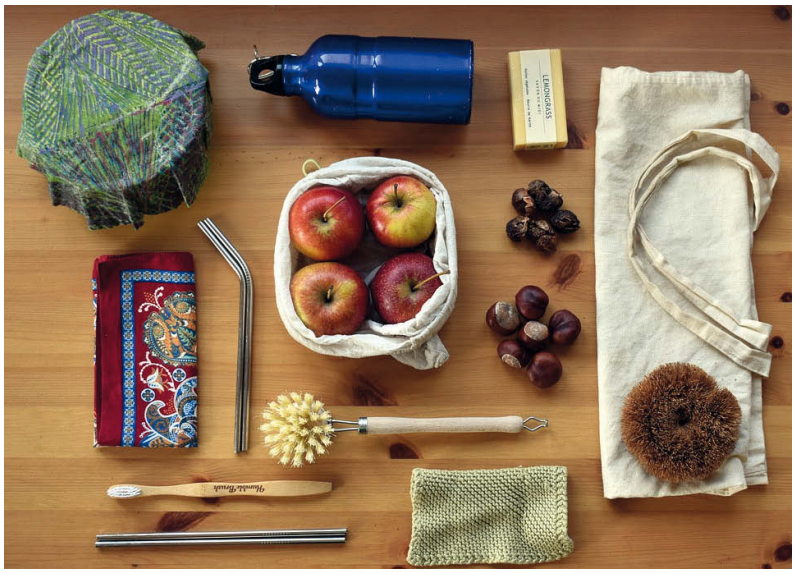


Image by RikaC from Pixabay

La solution toute simple consisterait à réduire le superflu, se contenter de ce qui est vraiment essentiel : une sobriété numérique. Dans les mots, c'est beau, mais dans les actes, c'est flou. Si on pense à notre quotidien à la maison avec cette idée de sobriété, on arrive alors facilement à la doctrine du "Zéro déchet", visant à réduire au possible nos déchets domestiques. Si les règles du Zéro Déchet fonctionnent à la maison, peut-on essayer de les transposer et les appliquer au développement logiciel ? Que pouvons-nous alors changer concrètement dans nos pratiques de développement pour améliorer notre empreinte carbone numérique ?

Zéro Déchet

Schématiquement, en commençant par la sobriété, il est possible de résumer la stratégie « zéro déchet » en trois points :

- supprimer le recours à l'incinération des déchets et structurer le système de collecte des déchets de manière à :
  - augmenter la quantité de matière différenciée récupérable (pour les particuliers cela signifie séparer les différents types de déchets et notamment les déchets organiques pour la fabrication de compost),
  - optimiser la qualité de la matière recyclable, ce qui aura pour conséquence de réduire simultanément la quantité de déchets produite ;
- encourager la réutilisation des matériaux recyclés par les industriels, la réparation des objets par les particuliers, et promouvoir les modes de vie qui réduisent la quantité de déchets produite (par exemple, utiliser l'eau du robinet, éventuellement filtrée à domicile, plutôt que l'eau en bouteille ou encore acheter en vrac et ainsi réduire les déchets d'emballage) ;
- soutenir la conception et la fabrication par l'industrie de produits entièrement recyclables, réutilisables et réparables.

(source : Wikipedia)

Béa Johnson (autrice du livre « Zéro-Déchet ») en a créé 5 règles (à appliquer dans l'ordre) :

- **Refuser** ce dont nous n'avons pas besoin (par exemple les échantillons, les sacs plastiques, les publicités dans la boîte aux lettres, etc.)
- **Réduire** ce dont nous avons besoin et ne pouvons pas refuser (par exemple éviter les objets en plusieurs exemplaires, surveiller la consommation énergétique, réduire le gaspillage alimentaire, etc.)
- **Réutiliser** ce que nous consommons et ne pouvons ni refuser, ni réduire (par exemple ne pas utiliser de produit jetable, réparer les pantalons troués des enfants, réinventer un nouvel usage pour un objet, etc.)
- **Recycler** ce que nous ne pouvons ni réutiliser, ni réduire, ni refuser (par exemple trier ses déchets)
- **Composter** le reste : il ne doit a priori rester que des déchets organiques compostables !

On constate que ces règles simples permettent déjà de réduire drastiquement la taille de nos poubelles. Mais qu'en est-il de nos "poubelles numériques" ?

Le matériel

Le matériel est de loin la plus grosse source de pollution numérique. Il faut 22 kg de produits chimiques pour fabriquer un ordinateur, auxquels il faut ajouter 240kg de combustible et 1,5t d'eau.
(Source : WWF).

On constate cependant que les règles du Zéro-Déchet peuvent facilement être appli-

quées à notre matériel technologique :

- **Refuser** : Combien de clés USB publicitaires avons-nous ? Combien d'objets connectés annoncés comme révolutionnaires sont oubliés dans nos tiroirs ou rendus obsolètes car le serveur hébergeant le service ferme ses portes ? Peut-être peut-on en effet commencer par refuser tous ces gadgets numériques qui nous encombrent et qui sont extrêmement difficiles à recycler en plus d'avoir une production écologiquement impactante.
- **Réduire** : A-t-on besoin d'avoir un PC, un PC portable, une tablette, un smartphone à la maison pour uniquement consulter ses mails et surfer sur internet ?
- **Réutiliser** : Avant de se débarrasser de nos objets vieillissants, cassés, peut-être est-il possible de les réparer ? Un ancien PC va peut-être pouvoir retrouver une nouvelle jeunesse une fois réinstallé, et pourra servir pour une utilisation simplifiée (ex : navigation sur le web, consultation des mails, traitement de texte, ...) ? Il pourrait même faire des heureux dans des écoles ou des associations :-)
- **Recycler** : Si vraiment notre objet n'est plus utilisable, il faut alors s'assurer qu'il sera bien recyclé. Et donc l'emmener à la déchetterie et non le jeter dans nos ordures ménagères.
- **Composter** : Bon, ok… là, c'est un peu plus compliqué… Mais logiquement, on devrait avoir réussi à recycler tous les composants ;-)

Avec le matériel, nous avons la face visible de l'iceberg. Mais le numérique est également à l'origine d'une pollution bien moins visible, celle dûe à son usage ! La consommation énergétique liée au numérique est une grande source de pollution : consommation des serveurs, des machines clientes, mais également de toute l'infrastructure réseau (routeurs, backbone, ...) qu'il faut maintenir, refroidir …Ici encore, il est possible d'appliquer les règles du Zéro-Déchet et viser à réduire la consommation énergétique de nos applications.

Développement 0 déchet

Le développement, c'est le cheminement de l'idée à une réalisation concrète et fonctionnelle, à défaut d'être tangible. Hors de l'aspect tangible, il peut être alors plus délicat d'envisager nos règles du Zéro-Déchet et leur application. Et pourtant, nous verrons

Photo by Blake Connally on Unsplash

que chacune des règles peut être appliquée à au moins une étape du développement de notre logiciel.

- **Refuser** :

Tout d'abord, la fonctionnalité qui consomme le moins d'énergie est celle qui n'existe pas. Cela commence donc par **refuser les fonctionnalités inutiles**, aussi appelées "gras" logiciel. Cela apparaît sûrement comme une évidence, mais il est important de le rappeler. Avant d'ajouter une nouvelle fonctionnalité, il est nécessaire de toujours se demander si elle est vraiment indispensable et si elle nécessite vraiment toutes les fioritures qui lui ont été ajoutées par des *Product Owners*.

Une chose sur laquelle les développeurs (ou du moins les architectes) auront plus facilement la main, sera le choix des technologies et des frameworks à utiliser. Utiliseriez-vous un bazooka pour écraser une mouche ? **Refusez l'utilisation des nouvelles technos "hypes"** qui n'apportent rien au projet (mais mises en place parce que c'est à la mode et que ça fait bien sur le CV) ! Et en plus, cela vous permettra d'avoir une meilleure maîtrise sur l'écosystème de votre application !

Refusez également le stockage de données inutiles : il est évident qu'on ne stocke pas d'informations personnelles inutiles à l'application, mais il faut également être attentif sur toutes les informations qui sont enregistrées et qui ne le nécessitent pas forcément (idem avec les logs).

Pour résumer, il faut toujours se poser la question « **Est-ce que l'utilisateur en a vraiment besoin ?** », et faire attention aux demandes techniques farfelues du client. Après tout, diriez-vous à votre maçon quel composant utiliser pour faire son béton ?

- **Réduire** :

"*On ne change pas nos ordinateurs et nos smartphones parce qu'ils ne fonctionnent plus mais parce qu'ils rament. Ils rament parce que les nouvelles versions de logiciel sont toujours plus gourmandes.*" *GreenIT.fr*

Cette citation résume plutôt bien l'état actuel de notre industrie ! Et il y a énormément de leviers à activer sur ce point pour essayer de réduire la voracité de nos applications.

L'action la plus simple à mettre en œuvre est de **réduire la taille des images et des CSS**. A-t-on vraiment besoin d'une image 4K pour l'afficher dans un carré de 15px ?

Il est également possible de **réduire la taille (et la quantité) des librairies embarquées** dans nos logiciels. Souvent, on embarque une librairie pour n'en utiliser qu'une seule fonction. Est-il possible de l'extraire ? Pensez à utiliser le *tree-shaking* pour vous "débarrasser" des librairies et dépendances inutilisées.

Et pour alléger un peu la charge du serveur, jetez un œil aux flux réseau et essayez de **réduire le nombre de requêtes vers le serveur**. Il n'est pas rare de constater des allers-retours inutiles. Et tant que vous êtes à regarder les trames réseau, essayez également de **réduire la taille des flux de données transférées**. Ne transmettez que le minimum d'information nécessaire au bon fonctionnement du front-end. Rarement l'affichage d'une liste d'utilisateurs nécessite toutes les informations d'un utilisateur par exemple.

Côté back-end, de la même manière, pensez à **réduire la quantité d'information remontée depuis la base** (pas de "select *").

Pour les Javaistes, il peut être également intéressant de jeter un œil à des stacks comme Quarkus ou Micronaut qui promettent une empreinte mémoire plus faible.

Et d'une façon plus globale, essayez de **réduire la complexité algorithmique** de vos applications (**Keep It Simple Stupid**).

- **Réutiliser** :

La réutilisation est un peu au cœur de notre métier et est souvent déjà mise en œuvre. Nous apprenons rapidement à ne pas réinventer la roue, et nous **réutilisons des com-**

posants, des librairies (idéalement sous licences libres !) qui ont déjà fait leurs preuves et sont pour la plupart déjà optimisées (on se souviendra toutefois de faire attention à la volumétrie de ce dont on a besoin dans une librairie).

L'utilisation du cache est également une bonne façon de réutiliser des anciennes requêtes. Pensez à en mettre en place aussi bien du côté front-end que du côté back-end (sans pour autant en abuser et n'en mettre que lorsque cela est utile, bien entendu).

• **Recycler** :

Le recyclage d'une application est rarement quelque chose auquel on pense. Néanmoins, une chose très importante doit être recyclée lorsque l'on change de système : les données ! Pour faciliter la migration des données de votre application à la nouvelle version, pensez à **proposer des exports dans des formats standards et utiliser des structures de données simples et compréhensibles.**

• **Composter** :

"Le compostage est le recyclage des déchets organiques pour produire naturellement un fertilisant, le compost."

Si on reprend cette définition du compostage, il est difficile d'imaginer en faire en développement logiciel. Et pourtant ! Donner des conférences, écrire des articles, binômer ou encore mob-programmer, toutes ces pratiques et ces échanges permettent aux plus expérimentés de **partager leurs connaissances,** leurs erreurs passées et aident les autres à monter en compétence, un peu comme s'ils leur apportaient de l'engrais.

Tout ceci représente des petites choses que nous pouvons faire seuls en tant que développeurs, dans notre quotidien. Ces actions peuvent sembler anodines, mais elles permettent aux applications d'avoir une durée de vie plus importante.

Surtout, elles permettent aux utilisateurs de pouvoir utiliser leur matériel plus longtemps, sans devoir le changer tous les six mois parce que la nouvelle version de l'application fait tout ramer, et donc de réduire la pollution matérielle.

Voici quelques exemples d'améliorations apportées par ces bonnes pratiques :

- la diminution du nombre de requêtes (de 165 à 23) et des feuilles de style CSS (de 36 à 2) nécessaires à l'affichage d'une page de connexion a permis de diminuer de 3s le temps de chargement de la page et de réduire de 30% la consommation énergétique.
- la suppression de requêtes effectuées en doublon a permis de réduire le temps de chargement d'une page de 16 à 3s, en réduisant la consommation énergétique de 80%.
- la mise en place d'un cache sur une application web a permis de réduire le temps de chargement de 12s à 5s.

Il est également possible de mesurer l'indice d'éco-conception d'une application via EcoIndex (http://www.ecoindex.fr/ - existe aussi en plugin firefox). Il devient donc aisé de vérifier en cours de développement si notre index s'améliore ou au contraire, si l'impact écologique est plus grand.

Eco-Conception

"L'éco-conception consiste à intégrer l'environnement dès la conception d'un produit ou service, et lors de toutes les étapes de son cycle de vie" (AFNOR, 2004)

Cette démarche plus poussée est plus particulièrement portée par la direction RSE et les directions métier qui conçoivent les produits et services vendus par l'entreprise.

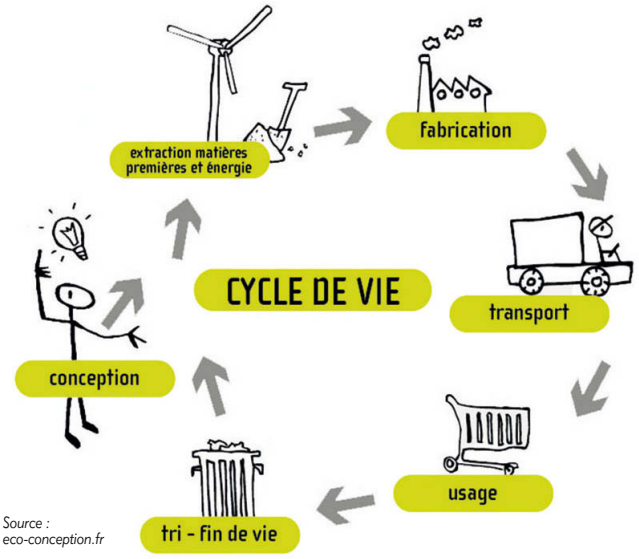
L'éco-conception intervient à tous les niveaux du cycle de vie d'un produit numérique : de sa conception à sa réalisation, au déploiement, à son administration, son utilisation, sa maintenance ainsi que sa fin de vie.

Les éléments dont nous avons parlé précédemment interviennent essentiellement dans les phases de conception et de réalisation. Beaucoup de choses peuvent être également améliorées dans les autres phases d'un projet afin d'en améliorer l'impact écologique.

N'hésitez pas à jeter un oeil aux formations certifiantes à l'éco-conception de services numériques pour approfondir le sujet.

Conclusion

L'éco-conception a évidemment un impact positif sur notre environnement, mais ces effets de bord ne s'arrêtent pas là. On constate



Source : eco-conception.fr

également une réduction des coûts (du fait de la réduction de la consommation énergétique et des besoins en matériel et infrastructure), une augmentation de la performance (un logiciel qui consomme moins de ressources sera plus performant), une anticipation des réglementations (l'impact du numérique commence à être pris en compte par le législateur). Et d'un point de vue plus marketing, cette démarche apporte également une différenciation du produit et permet aux entreprises de communiquer sur cette démarche écologique.

De nombreux labels commencent à apparaître (Green Code Label, Eco-Label, ...) mais aucun n'a pour le moment réellement émergé.

Quelques outils pratiques :

Le Club Green IT regroupe un ensemble d'outils tels qu'une checklist de 65 bonnes pratiques en licence Creative Commons, un lexique, une présentation de certifications.

Le Collectif conception numérique responsable référence une liste de 115 bonnes pratiques avec une checklist associée, une certification éco-conception, des systèmes d'évaluation, des services en ligne comme ecometer.org et ecoindex.fr, des listes de discussion.

(https://collectif.greenit.fr/outils.html)

CONCEPTION NUMERIQUE RESPONSABLE

Références

- GreenIT.fr
- Eco-conception web : les 115 bonnes pratiques
- Livre blanc éco-conception numérique (http://alliancegreenit.org/wp-content/uploads/Doc%20AGIT/LB-ecoconception-numerique.pdf)
- The Shift Project (https://theshiftproject.org/)
- Article du WWF : https://www.wwf.fr/vous-informer/actualites/etude-wegreenit-quel-impact-environnemental-du-numerique-dans-les-entreprises
- Recyclage du matériel informatique : https://www.eco-systemes.fr/recycler-du-materiel-informatique-ou-un-ordinateur

Laurent VACHE
@vachaldo
Les Productions de MOA

La place des femmes au cinéma

Il y a plusieurs années de cela, je vous écrivais quelques mots sur « La culture geek et les femmes » (cf. Programmez! n°196 de Mai 2016). Nous évoquions leur place dans les films ou les séries TV, plus souvent cataloguées à un « soi belle et tais toi ! », plutôt qu'à des rôles profonds et destinés à faire rêver les foules. Mais, en nuançant quand même ces propos de quelques belles surprises où elles reprenaient enfin une place (presque) à l'égal de l'homme.

Depuis ces trois ans et demi, les choses ont-elles changé ? Les femmes ont-elles réussies à conquérir le monde tel un super héros vêtu de sa cape ou de son bouclier qui arriverait à gagner le cœur des foules ? Mais surtout, comment pourraient-elles changer les choses ? Par quels moyens ?

Avant de répondre à ces questions, revenons en arrière pour mettre en exergue nombre de rôles au cinéma qui ont fait la part belle à ces femmes fortes au détriment des rôles de « plante verte » qui leur sont trop souvent réservés.

Une place à (re)prendre ?!

Si je vous donne le nom de Beatrix, certains vont immédiatement avoir l'image de Poudlard, d'un monde fait de magie et d'un certains Harry P. Et bien non ! Je vous parle de *Beatrix Kido*, interprétée par *Uma Thurman* dans les *Kill Bill* de *Quentin Tarantino*. Un film où l'héroïne, en plus d'être "badass", introduit au cinéma la possibilité d'une super-héroïne indépendante et qui a le pouvoir de se sauver elle-même. Une chose qui était peu répandu à cette époque. Les films d'action, de science-fiction ou d'arts martiaux ne faisant la part belle qu'aux hommes défendant la patrie et la jolie demoiselle en détresse pour faire des entrées et engranger le roi Dollar. Et pourtant… avec *Hunger Games*, qui fut en premier lieu une série de roman à succès, les films et les recettes de ces derniers ont permis de montrer aux producteurs que l'on peut avoir un rôle féminin comme personnage central de son film et réussir un carton au box-office. Vous me direz, faire entrer plus de 2 milliards $ de recettes, cela aide un peu à prendre conscience de cela. Habituellement, les blockbusters ne proposent pas forcément de rôles féminins forts, mais avec les succès de *Wonder Woman* ou des derniers *Star Wars*, les financeurs pourraient changer la donne.

On voit depuis le début des années 2000, que la scène est avec succès reprise par de plus en plus de rôles féminins. Et parfois, sur des genres cinématographiques exclusivement réservés aux hommes par le passé. Je nommerais les *Resident Evil* qui regroupent trois styles qui sont l'action, l'horreur et la science-fiction. Tous trois, auparavant, chasse gardée des hommes et où les femmes n'étaient là que pour être sauvées

Dans un style différent, des films comme *Joue-la comme Beckham* ou *Million Dollar Baby* utilisent un univers en apparence purement masculins pour pondre des films à la limite du féminisme et où les personnages principaux montrent leur rage et leur détermination à se battre pour exister en tant que personne.

Je pourrais vous faire une liste plus grande de ces films où le premier rôle féminin « casse la baraque » comme on dit. Mais que vaudrait une liste face à une comparaison entre ce qui fonctionne avec un personnage principal masculin et un féminin ? On ne saura jamais si Captain America ou Iron Man auraient aussi bien marché si les acteurs avaient été des femmes au lieu d'être des hommes.

Des sagas pour les changer tous ?

Si on prend comme référence l'une des plus grandes sagas de tous les temps, *Star Wars*, nous pouvons observer tout et son contraire dans les différents films. En premier lieu, *Carrie Fisher* dans le rôle de la *Princesse Léia*. Dans une époque et un style de film où les femmes n'étaient absolument pas mise à l'honneur autrement que par leur plastique, ce personnage a contribué à casser les codes. Dans des années 70 sombres pour les femmes et qui mettaient en exergue *Scarface*, *Taxi Driver*, *Le Parrain* et j'en passe, la *Princesse Léia* amenait une image de personnage féminin fort, qui a le sens de la répartie et qui se bat pour imposer ses idées. Elle dérogeait également au stéréotype de la demoiselle en détresse car en début de film, elle tient tête au puissant Dark Vador et, plus loin, son sauvetage n'est que pur hasard, *Luke Skywalker*, *Han Solo* et le reste de l'équipage du *Millenium Falcon* n'arrivant sur *l'Etoile Noire* que par accident. Et quand ils la trouvent, elle ne montre aucun stress, aucune panique et se permet même de critiquer la façon de faire des deux « héros » masculins que sont *Mark Hamill* et *Harrison Ford*.

Malheureusement, et les antépisodes vont dans le même sens, malgré un personnage aux apparences fortes, *Leia* est là en tant que soeur de *Luke* et/ou promise de

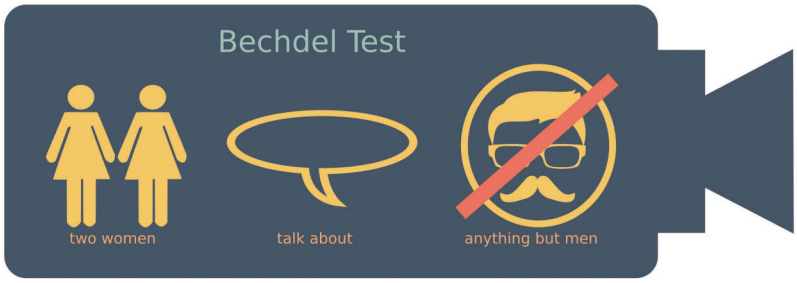
Han Solo. Tandis que le rôle de *Padme* est de devenir la femme de *Anakin*, et la mère de *Luke* et *Leia*. Elles passent de rôle fort à objet de convoitise masculin.

Enfin, le personnage de Rey dans la nouvelle trilogie annonce un tournant intéressant dans Star Wars. Le seul soucis réside dans le fait que, actuellement, l'impression qu'elle n'est juste un penchant féminin du Luke Skywalker de la première trilogie est fort... du moins, à mon humble avis. Le côté positif à tout cela, si l'on puisse dire, c'est que Rey est un personnage masculinisé, que ce soit par sa tenue ou sa démarche. Loin du stéréotype de science-fiction où l'on met en avant en priorité la plastique féminine au détriment de son intellect. On peut également se féliciter de voir dans le spin-off *Rogue One*, que le personnage principal reste une femme avec un petit bémol sur le fait qu'elle suit le mouvement au lieu de le créer.

Parlons maintenant de deux icônes féminines des années (70)80. *Sarah Connor* d'une part et *Ellen Ripley* d'autre part. Pour cette dernière, le film *Alien*, a permis de consacrer un personnage féminin fort au cinéma. *Sigourney Weaver* n'ayant rien à envier aux acteurs masculins de l'époque, portant elle-même, de bout en bout, un blockbuster qui a fait date dans le domaine de la science-fiction. Très vite, elle s'impose comme l'unique espoir pour défaire un monstre puissant au phallisme exacerbé. Une héroïne, dont on ne regarde pas forcément le physique tant l'histoire est prenante et haletante, et qui va ouvrir la voie en 1984, via Terminator, à la mise en avant d'une autre icône de la pop culture : *Sarah Connor*. Seule, ou presque, face au T-800 incarné par *Arnold Schwarzenegger*, elle incarne un héroïsme non pas axé sur les muscles et la puissance, mais sur le courage, l'intelligence et une capacité d'adaptation face aux situations complexes.

Enfin, plus récemment, dans *Mad Max : Fury Road*, *Furiosa* est un autre exemple de ce retour en force des femmes fortes et indépendantes dans le cinéma. Charlize Theron n'y est pas l'héroïne principale, mais elle y éclipse le héros éponyme du film grâce à son interprétation d'un personnage fort mais également par l'aura qu'elle dégage.

Toutes ces femmes, indéniablement convaincantes, pourraient permettre à de



plus nombreuses héroines d'émerger dans un panel cinématographique consacré au héros masculins musclé et sauveur de l'humanité. Même si ces quelques films trouvent un écho favorable auprès du public, ils ne sont que trop peu nombreux à mon sens.

Conclusion

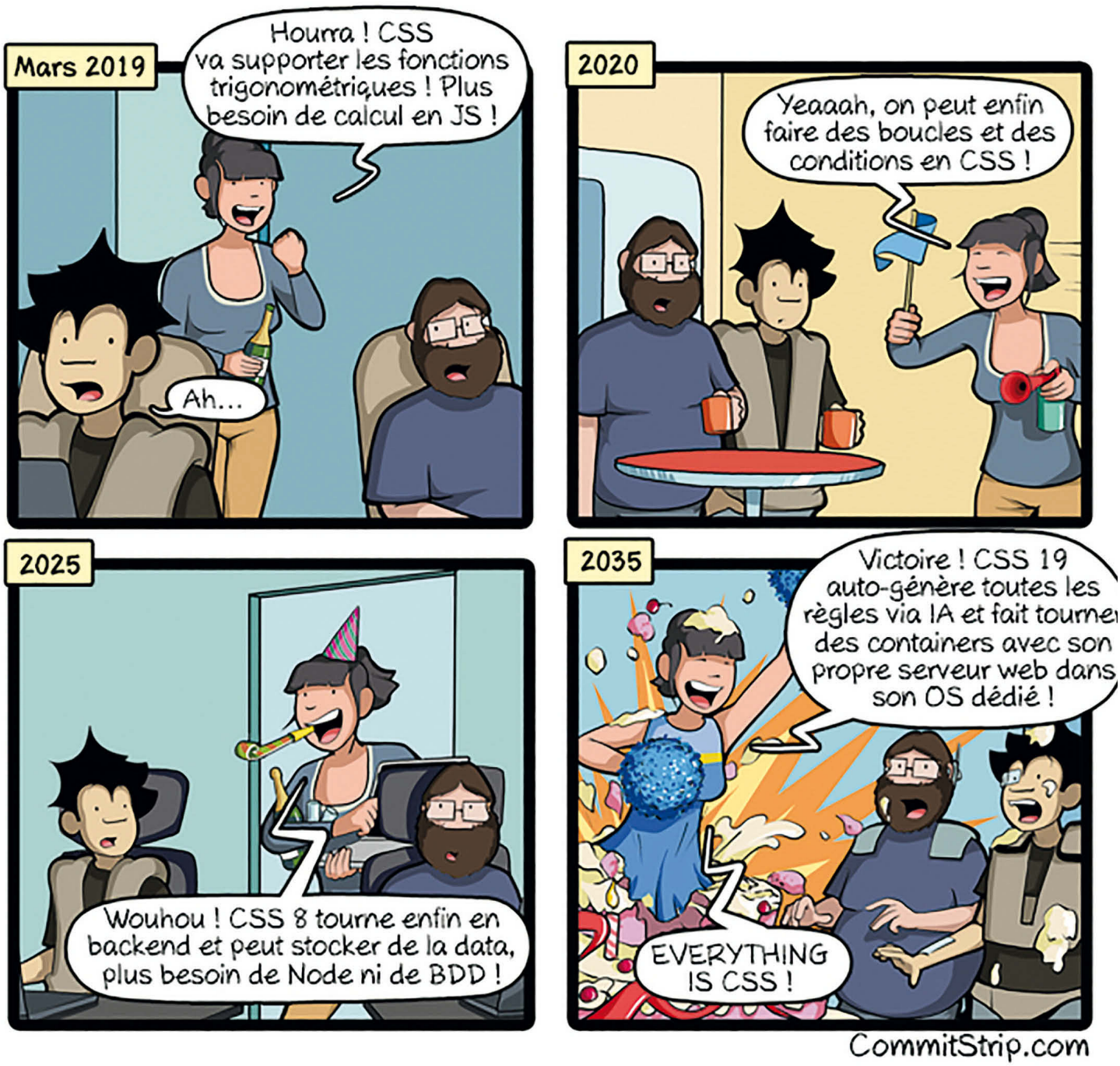
Après tout ceci, ce que l'on peut constater de plus flagrant est que le chemin est encore long pour que les femmes ne soient plus considérées comme des objets ou des faire-valoir dans les œuvres de fiction grand public. Le sexisme tend à s'estomper peu à peu, notamment grâce à des mouvements qui ont permis aux langues de se délier, comme par exemple le phénomène *#MeToo*. Mais ce n'est pas un hashtag et des accusations qui mettront fin à ces inégalités et au sexisme ambiant des réalisateurs majoritairement masculins, mais à l'évolution des habitudes de consommations des lecteurs, spectateurs et téléspectateurs.

De nombreux tests ont également émergé pour déterminer le degré de sexisme d'une oeuvre. Comme par exemple le test de Bechdel. Pour le résumer rapidement, ce test va tenir en trois questions : Y'a-t-il au moins deux personnages féminins identifiables dans l'oeuvre ? Parlent-elles l'une avec l'autre ? Parlent-elles d'autre chose que d'un personnage masculin ?

Ce test n'est en rien une preuve de dédouanement de sexisme dans un film, mais il aide à reconnaître les œuvres où se trouve un personnage féminin fort. On pourra noter comme exemple les films *Avengers* avec le personnage de la Veuve Noire. *Natalia Romanov* y est une femme forte, mais l'utilisation de l'image de *Scarlett Johansson* sur certaines affiches de films, fesses en direction des spectateurs, marque le long chemin à faire pour que les réflexes sexistes s'envolent.

A contrario, il ne faut pas risquer de gâcher tout ce combat légitime par des exagérations qui commencent à arriver d'une part infime de mouvements féministes extrêmes et où, le féminisme par les quotas ou par des combats sexistes inversés (donc en direction des hommes) pourraient décrédibiliser des demandes normales et justes des femmes actrices et artistes qui ont depuis trop longtemps été cantonné à des rôles subalternes à contrario du mâle alpha détenteur du pouvoir dans ce monde du divertissement. •

CSS est partout !



Une publication Nefer-IT, 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08 - Directeur de la publication : François Tonic
Rédacteurs en chef : Aurélie Vache & François Tonic
Ont collaboré à ce numéro : la rédaction de ZDnet
Nos experts techniques : A. Vache, P. Boulanger, M. Rivault, C. Santonastasi, M. Felix, A. Briffod, J. Thiriet, C. Le Luët, J. Dollé, L. Avrot, P. Logna, M-A Blete, C. Villard, E. Aboaf, G. Acas, M. Avomo, L. Crépin, A. Lieva, M. Rocher, C. Bossard, Y. Martel, L. Vache, Commitstrip
Couverture : Octocat / GitHub - **Maquette** : Pierre Sandré.
Publicité : François Tonic / Nefer-IT - Tél. : 09 86 73 61 08 - ftonic@programmez.com.
Imprimeur : Moderna (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, janvier 2020

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

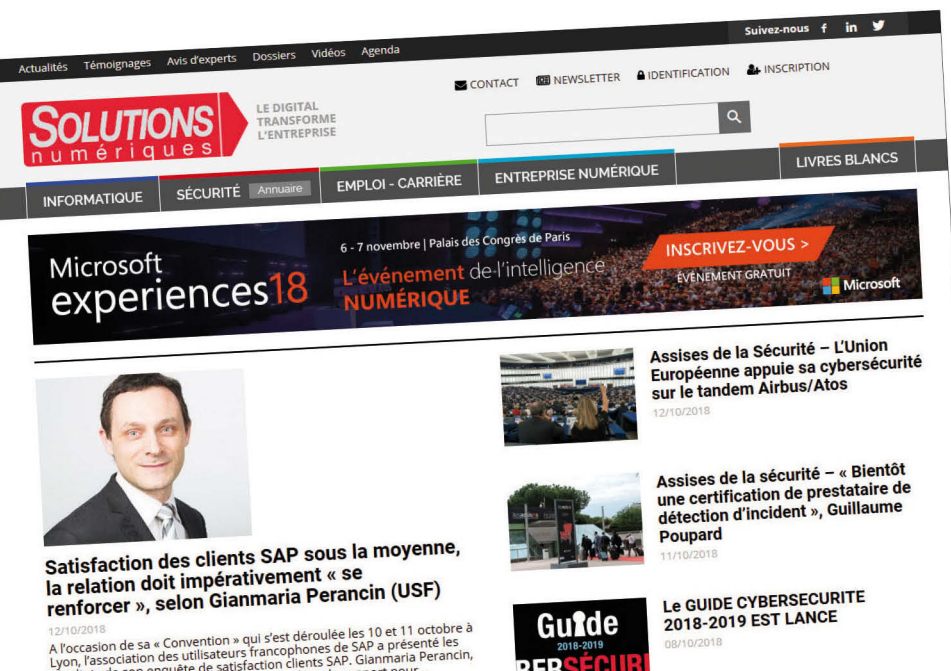
INFORMER
pour transformer l'entreprise

La dématérialisation, le Cloud,
les communications unifiées,
les nécessités de la cybersécurité
transforment le travail et toute l'entreprise,
les services publics.
Le magazine, le site, ses newsletters
vous informent sur cette actualité mouvante
et vous aident à décoder les tendances

Abonnez-vous

www.solutions-numeriques.com/abonnement/

4 sites thématiques, pour répondre à vos besoins d'information



- Vous êtes responsable informatique ou bien dirigeant ou cadre d'entreprise ? *2 sites répondent à votre profil*
- La cybersécurité vous concerne ? *Cliquez sur l'onglet. Vous trouverez les infos, l'annuaire, le lexique, etc*
- L'emploi, les salaires, les formations, les offres vous intéressent ? *Le site sur l'Emploi dans le numérique est à votre disposition*

www.solutions-numeriques.com

REWIND TO THE FUTURE

by TheCodingMachine

UNE EXPO EN MUSIQUE POUR REMONTER LE TEMPS

VOUS ALLEZ DÉCOUVRIR LES TECHNOLOGIES DE LA FIN DES ANNÉES 70 AU DÉBUT DES ANNÉES 2000, LE TOUT TEINTÉ DE LA CULTURE GEEK & POP.

DU 28 AU 30 JANVIER 2020
AU 56 RUE DE LONDRES - PARIS 8

#TCMVERSLEFUTUR

sur facebook, twitter et instagram

INSCRIPTION OBLIGATOIRE !