RENTRÉE 2018 | RENTRÉE 2018 | RENTRÉE 2018 |

programmez.com

Programmez!

Le magazine des développeurs

221 SEPTEMBRE 2018

LE TOP DES TECHNOS ET DES LANGAGES

C++ 17

SQL Server pour Linux

GitLab

GraphQL

Nintendo Labo

Construire son NAS

© mikkelwilliam

M 04319 - 221 - F: 6,50 € - RD

LE SEUL MAGAZINE ÉCRIT PAR ET POUR LES DEVELOPPEURS

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

Microsoft **experiences18**

L'ÉVÉNEMENT de l'intelligence NUMÉRIQUE

Mardi 6 novembre
JOURNÉE BUSINESS

Mercredi 7 novembre
JOURNÉE TECHNIQUE

PALAIS DES CONGRÈS DE PARIS

ÉVÉNEMENT GRATUIT

Inscrivez-vous : aka.ms/experiences18

#experiences18

Microsoft

EDITO

Programmez ! c'est vous !

Depuis 20 ans, et quelques mois, la rédaction cherche à répondre à vos besoins, à vos attentes, en essayant de proposer une grande variété de langages et de technologies. Ce qui n'est pas toujours simple. Ce qui nous passionne, c'est la technologie et le code.

Dans ce numéro, nous introduisons une petite astuce pour repérer plus rapidement le niveau des articles techniques :
100 : niveau découverte
200 : niveau intermédiaire
300 : niveau expert / expérimenté

N'hésitez pas à nous faire un retour sur cette notation.

Les codes sources des articles sont disponibles sur :
- La page du numéro sur www.programmez.com
- Sur notre Github : https://github.com/francoistonic/

Pour suivre l'actualité développeur et des technologies, www.programmez.com vous donnera les dernières tendances et nouveautés. Et n'oubliez pas de vous inscrire à notre newsletter hebdomadaire.

Le magazine organise régulièrement des meetups et des conférences techniques. Nos prochains rendez-vous sont :
- 11 septembre : meetup #2 « Du CP/M aux OS actuels », à partie de 19h dans les locaux de Cellenza à Paris
- 13 décembre à 42 : DevCon #7 « Infrastructure as Code »

Vous trouverez toutes les informations nécessaires sur www.programmez.com

YES, WE CODE

Programmez! cherche régulièrement de nouveaux contributeurs techniques pour le magazine et le site web. N'hésitez pas à proposer vos thèmes, suggestions et articles. Contactez-moi directement : ftonic@programmez.com

Bonne rentrée avec Programmez!.

François Tonic
ftonic@programmez.com

Suite à un problème technique, la partie 2 de l'article Principes avancés de conception objet sera publiée dans le n°222

SOMMAIRE

Dans le prochain numéro !
Programmez! #222, dès le 28 septembre 2018

La programmation orientée modèle : késako ?
Angular 6 : quelles nouveautés pour le dév ?
OpenMapStreet : l'alternative à Google Maps
Monter un cluster en Lego

Apple reprend de 0 ou presque les cartes de Plans. Depuis sa disponibilité, le service de la Pomme a du mal à rivaliser avec Google Maps même si le service a certaines qualités (si, si).

La technologie LiFi, le réseau par la lumière, va-t-il enfin sortir de l'ombre ? Les initiatives commencent à se multiplier.

Le **Steam** chinois arrive bientôt en Europe : Tencent WeGame

Microsoft a lancé en juillet une nouvelle tablette pour concurrencer les iPad : Surface Go à partir de 399 $.

Le patron de **Reddit** a affirmé qu'il était difficile de bannir les propos haineux du site. Les communautés du site doivent pouvoir s'auto-réguler. Mais le site pourrait agir concrètement dans les prochains mois. A suivre.

Tesla arrive enfin à produire à la cadence initialement espérée : environ 5 000 unités.

Intel encore en retard sur le 10 nm

Décidément, Intel et la gravure 10 nm ce n'est pas une histoire d'amour. Les processeurs Cannon Lake 10 nm étaient prévus cette année mais le fondeur a annoncé au coeur de l'été une disponibilité fin 2019 ! Au départ, cette gravure devait être prête dès 2016… La cause de ce retard supplémentaire serait un rendement de production trop faible. Intel va-t-il sacrifier cette gamme au profit d'une architecture plus récente (Ice Lake ?).

Baromètre HIRED recherche d'emploi

La côte des développeurs Vue monte en flèche

Tous les mois, Programmez! publie en exclusivité le baromètre Hired des technologies les plus recherchées par les entreprises. Elles peuvent permettre aux développeurs de connaître les nouvelles tendances du recrutement pour se former ou se démarquer des autres candidats.

Comme l'a déjà montré le baromètre mensuel Hired de la recherche d'emploi des développeurs, les technologies les moins pratiquées sont souvent plus recherchées par les recruteurs. Si les précédents baromètres illustraient cette tendance, cette dernière a été poussée à son paroxysme en juin.

Ainsi, le langage Vue, mis en avant par moins de 1% des candidats ayant postulé sur la plateforme de recrutement au mois de juin, est pourtant celui qui a suscité le plus d'engouement chez les recruteurs avec une moyenne de 7 entretiens par développeur. Le mois dernier, ce nombre était de 4,9 alors que presque 3% des candidats mettaient Vue en avant, preuve d'un certain engouement du marché autour de cette technologie et d'un tarissement de l'offre. Avec respectivement une moyenne de 6,8, 6 et 5,3 entretiens proposés par candidat, Go, iOS et .NET complètent le haut du classement des technologies les plus demandées au détriment de React et Ruby qui sont évincées du top 4. Ce dernier, le plus plébiscité en mai dernier avec 9,3 entretiens demandés par candidat n'a engendré en juin que 4 prises de contact en moyenne.

Toujours en tête des technologies les plus représentées chez les candidats Java, Node et PHP font moins office de valeur sûres en juin, surtout ce dernier qui est bon dernier du classement avec 3,57 demandes d'entretiens par candidat. Les deux autres sont respectivement crédités de 4,3 et 4,2 demandes d'entretiens en moyenne, confirmant la baisse constatée depuis le mois d'avril. Les résultats des 6 derniers mois illustrent d'ailleurs cette tendance. Sur cette période, Java et PHP ferment le classement juste devant .Net et Android. Les demandes pour Go, React et DevOps sont toujours au beau fixe.

juin 2018			
Technologies demandées	Pourcentage de candidatures développeurs	Technologies demandées	Nombre moyen de demandes d'entretiens
Java	18,72%	Vue	7
Node	18.64%	Go	6.8
PHP	12.71%	iOS	6
React	11.44%	.NET	5.3
Python	11.02%	React	5.2
Angular	10.59%	DevOps	5
Android	5.08%	Java	4.3
.NET	4.24%	Angular	4.2
Go	2.12%	Python	4.2
Ruby	2.12%	Node	4.2
Vue	0.85%	Ruby	4
iOS	0.85%	Android	3.7
DevOps	0.85%	PHP	3.6

Janvier 2017 à Juin 2018			
Technologies demandées	Pourcentage de candidatures développeurs	Technologies demandées	Nombre moyen de demandes d'entretiens
Java	17.82%	Go	9.3
Node	14.99%	React	8.3
PHP	14.44%	DevOps	8.3
Python	11.85%	Ruby	7.3
React	10.68%	Python	7
Angular	10.52%	Vue	6.8
Android	5.10%	Node	6.7
.NET	4.24%	Angular	6.6
Ruby	3.38%	iOS	6.2
Go	2.75%	PHP	6.1
Vue	2.28%	Java	5.4
iOS	1.96%	.NET	5.4
DevOps	1.18%	Android	4.3

A propos de HIRED

HIRED.com est une plateforme technologique de recrutement qui attire et sélectionne les meilleurs profils techniques du marché afin de permettre aux entreprises de recruter des candidats qualifiés rapidement et efficacement. Chaque semaine, entre 80 et 100 nouveaux candidats de la communauté française HIRED en recherche active (développeurs, data scientists, designers, etc.) sont triés sur le volet grâce à des algorithmes et prêts à être contactés.

DevCon #7

13/Décembre/2018

INFRASTRUCTURE AS CODE

Pourquoi coder son infrastructure ?

- 2 KEYNOTES
- 4 SESSIONS TECHNIQUES
- 2 QUICKIES
- 1 PIZZA PARTY

Conférence organisée par

[Programmez!] Le magazine des développeurs

Où ?

42

INFORMATIONS & INSCRIPTION SUR WWW.PROGRAMMEZ.COM

SEPTEMBRE

ReBuild 2018

14 septembre à Nantes

La grande conférence sur les technologies Microsoft revient mi-septembre. De nombreux thèmes seront abordés : réalité virtuelle / augmentée, eSport, santé, robotique, IA, IoT, DevOps, et bien entendu du code ! Des workshops se dérouleront toute la journée.
Pour en savoir plus : https://www.eventbrite.fr/e/billets-rebuild-46877819821

Agile en Seine

25 septembre à Newcap event center, Paris.

Après une première édition en 2017, la conférence agile de référence revient à Paris avec 600 participants attendus ! 19 conférences, 2 keynotes, 3 ateliers seront proposés durant toute la journée. Agile en Seine est organisé par l'association loi 1901 French Kanban User Group qui a pour but de promouvoir les pratiques agiles et leur évolution.
Deux invités de prestige animeront les deux plénières de la journée :

- Virginie Guyot, ex-pilote de chasse, leader de la patrouille de France 2010 et femme en or 2010, proposera de prendre ensemble de la hauteur et nous parlera de leadership,
- Michel Hervé, président du groupe Hervé et lauréat du trophée « Leader responsable » partagera son expérience du management concertatif.

Profitez du code promo Programmez ! – 30 % (places limitées) : PROGRAMMEZ-AES18

CONFÉRENCES TECHNIQUES

Xebia organise 2 conférences techniques :

- **FrenchKit :** la première conférence française dédiée aux développeurs iOS et macOS. 20 et 21 septembre 2018 - frenchkit.fr/
- **XebiCon :** la conférence qui vous donnera les clés pour tirer le meilleur des dernières technologies : Data, Architecture, mobilité, DevOps, etc. 20 novembre 2018 - xebicon.fr

OCTOBRE

Les assises de la sécurité

Du 10 au 12 octobre à Monaco

La référence des événements sécurité revient pour la 18e année ! Durant 3 jours, les conférences, ateliers, tables rondes se succèdent sur tous les thèmes autour de la cybersécurité. De multiples sessions plénières animeront les différentes journées. Les assises ne parlent pas uniquement de sécurité mais aussi de l'IT en général.
Pour en savoir plus :
https://www.lesassisesdelasecurite.com

Meilleur dév de France

23 octobre à Espace Grand Arche, Paris – La Défense

L'événement MDF revient pour la 6e année ! Les développeurs de toute la France vont s'affronter durant toute la journée et relever les défis de programmation. Plus de 2 000 participants, une multitude de conférences, cette journée est placée sous le signe du code, du code et du code. Venez transpirer et faites chauffer les claviers. Pour en savoir plus :
https://www.meilleurdevdefrance.com

Forum PHP 2018

25 & 26 octobre à Marriott Rive Gauche, Paris

L'AFUP organise la nouvelle édition de la grand'messe PHP de l'année. Le Forum PHP est l'occasion de rencontrer les meilleurs experts français et mondiaux de PHP et des outils liés, de rencontrer la communauté et de découvrir les dizaines de conférences techniques et les nombreux retours terrains. Le forum s'est de la technique mais aussi la partie entreprise et projets IT. Site : https://event.afup.org

NOVEMBRE

Microsoft Experiences18

6 & 7 novembre au Palais des congrès, Paris

C'est l'événement annuel de toutes les communautés et des utilisateurs des technologies Microsoft. Comme chaque année, la première journée est placée sous le signe du business et comment la technologie peut aider l'entreprise, les utilisateurs. La seconde journée est placée sous le signe de la technique et des développeurs. On parlera code, outils, méthodes agiles, sécurité ! Des centaines de sessions et d'ateliers seront proposés sur les deux jours. Une occasion unique pour rencontrer d'autres développeurs et les experts Microsoft. Venez rencontrer la rédaction de Programmez ! Inscrivez-vous dès maintenant sur www.experiences18.microsoft.fr

SophIA

7, 8 et 9 novembre à Sophia Antipolis

La grande conférence autour de l'IA se déroulera à Sophia Antipolis. De nombreuses conférences seront données sur les différents axes de recherches et de réflexions autour de l'IA : dans la biologie et la génétique, dans l'économie, dans la société, dans l'éducation, etc. Il s'agira aussi bien de conférences techniques que des réflexions et des tables rondes.
Une conférence qui promet beaucoup ! Pour en savoir plus : http://sophia-summit.fr/sophia2018/fr

DEVFEST TOULOUSE 2018

8 novembre à Toulouse

Le DevFest Toulouse est un événement organisé par les communautés de développeur.se.s de Toulouse, et porté administrativement par le GDG Toulouse.
Pour rendre tout cela possible, une équipe de bénévoles s'active en coulisse.
Fort des deux dernières éditions, l'équipe remet le couvert pour une troisième année. Les nouveautés de l'édition 2018 : plus de talks, de speakers, dans un lieu plus grand : le centre des Congrès Pierre Baudis, 600 participants attendus … et un super thème : le rétro gaming/rétro computing ! Plus d'infos sur : https://devfesttoulouse.fr

MongoDB Europe' 18

8 novembre à Londres

La conférence européenne de MongoDB se tiendra à Londres. Une bonne occasion de rencontrer les équipes et d'aborder de nombreux thèmes : développement, cas d'usages, cloud, clustering, MongoDB par secteur d'activité.
Pour en savoir plus :
https://www.mongodb.com/europe18/agenda

Devops D-Day#4

15 novembre au Orange Vélodrome, Marseille

La journée DevOps D-Day revient pour la 4e année ! On y parle DevOps mais pas que ! Docker, les conteneurs, la transformations IT et le cloud sont les thèmes de la journée. Elle promet d'être chargée avec de nombreuses sessions et les différentes keynotes.
Site : http://2018.devops-dday.com

Maker Faire Paris

Du 23 au 25 novembre à la Cité des sciences, Paris

Le grand événement maker change de date ! Prévu initialement début novembre, la Maker Faire Paris se déroulera du 23 au 25 novembre. L'édition 2017 avait réuni plus de 800 makers et des milliers de visiteurs. Cette année encore, il s'agit de rencontrer toutes les communautés makers et DIY, assister aux conférences et les nombreuses démos !
Site : https://paris.makerfaire.com

WINDEV®

LE CHAMP PLANNING : RICHE ET PUISSANT

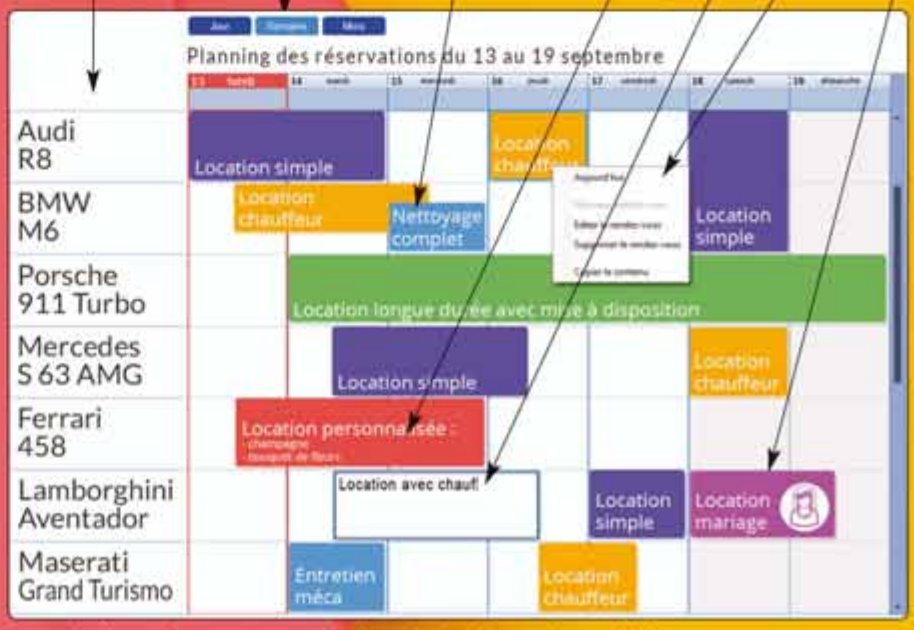
Un champ planning est très utile pour gérer l'affichage de ressources multiples et trouve sa place dans de très nombreuses applications ou sites.

Grâce à WINDEV 23, **il suffit de quelques heures** pour que vos applications bénéficient d'un **planning** évolué et totalement paramétrable.

Vous économisez des semaines.

UN PLANNING DANS VOS APPLICATIONS? FACILE !

Largeur paramétrable — *Changement de mode d'affichage* — *Gestion de la superposition* — *Look personnalisable* — *Menu FAA* — *Ajout: en saisie* — *Déplacement + changement de ressource*



+ Redimensionnement + Couleur de fond pour jours fériés

DECIDEUR

La programmation d'un champ planning sans WINDEV peut demander plusieurs semaines; avec WINDEV c'est quelques heures.

GERE AUTOMATIQUEMENT

Ajout de rendez-vous, modification de durée, d'heure, affectation à une autre ressource: toutes les FAA nécessaires sont disponibles

POINTS FORTS

Programmation archi-simple
Personnalisation totale

Logiciel professionnel. Document non contractuel

Tél : 04 67 032 032

WWW.PCSOFT.FR

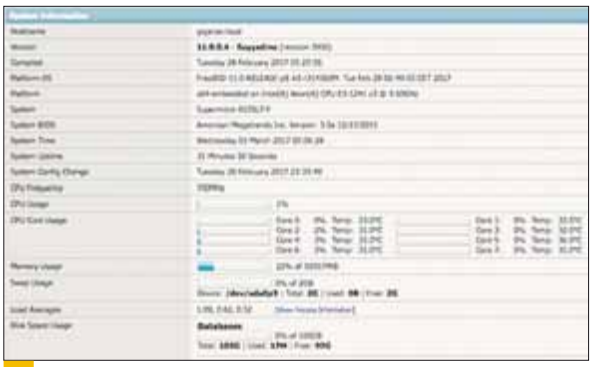
Le champ Planning est un des nombreux champs intégrés en standard dans WINDEV

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

Construire son NAS

niveau 200

Vous ne souhaitez pas investir dans un NAS du commerce prêt à l'emploi, c'est votre choix. Toutefois, vous aimeriez bien partager et centraliser vos photos, documents, etc., au sein de votre famille et des amis. C'est pratique et simple. Aujourd'hui, nous allons voir comment construire son propre NAS.



1

La décision de construire un NAS peut être un avantage économique et vous permettra d'avoir un périphérique performant. Toutefois il existe de nombreuses contraintes. Tout d'abord, il nécessite un peu de temps pour le monter et le configurer. Par ailleurs, vous devrez choisir les bons composants pour éviter un encombrement important de celui-ci ainsi qu'une consommation électrique plus élevée qu'un NAS du commerce.

C'est pourquoi l'article est avant tout à destination des bidouilleurs / makers, aux personnes qui souhaitent choisir leurs propres composants, jusqu'à la distribution servant de tour de contrôle.

MATÉRIELS

Le choix du matériel sera le point de départ pour construire son NAS car il est préférable de posséder un ordinateur dédié. La première idée viendra de regarder chez soi, si un ancien ordinateur pourrait répondre à cette attente en le transformant en disque de stockage ou de partages pour lui donner une seconde vie. Même si l'idée semble bonne, elle risque de se trouver assez vite limitée à cause du stockage parfois limité ou des temps de réponses plus longs, à cause de composants trop anciens. C'est pourquoi, nous privilégions des composants neufs pour garantir une durée de vie maximale et une utilisation intensive liée aux nombreuses personnes connectées en simultané.

Boîtier

Les boîtiers proposent une large gamme de produits et qui commence à partir de 2 baies. Pour obtenir un boîtier 2 baies, vous pouvez regarder les boitiers mini ITX. Les tarifs varient généralement entre 40 et 80 €.

Carte mère / processeur

La carte mère et le processeur sont 2 composants cruciaux. Vous aurez le choix entre :

- Une carte mère sans processeur, ce dernier sera à rajouter ;
- Une carte mère avec processeur.

Quand les deux sont intégrés, le tarif est généralement plus avantageux. Toutefois, selon l'utilisation, la solution intégrée ne sera pas forcément la meilleure. L'avantage de choisir le CPU est de pouvoir prendre un modèle qui correspond exactement aux besoins. Pour la carte mère, on regardera Asrock ou MSI. Prix : à partir de 80 – 100 €. Attention : respectez le format du boîtier. Dans notre cas il s'agit d'un ITX. Pour le processeur, un processeur Intel i3 est vendu aux alentours de 100 €, mais vous pouvez facilement doubler ou tripler le tarif selon le type de CPU.

Mémoire

La mémoire vive est cruciale pour les performances de la machine. Actuellement, les formats DDR3 et DDR4 sont les plus fréquents. Optez pour des marques reconnues : Corsair ou Kingston. Pour une barrette de 4 Go, comptez entre 30 et 45 €. Vérifiez les modèles acceptés par la carte mère.

Alimentation

Bien entendu, le bloc alimentation est important. N'oubliez pas que le NAS a vocation à fonctionner en 24/7 sans interruption. Il faut donc opter pour un bloc silencieux. Sur la puissance, il faut minimum 450 W. Là encore, préférez les marques reconnues telles que Corsair, Zalman. Entre 40 et 60 €.

Voilà, la base de votre NAS est fixée :

- Boîtier : 40 / 80 €.
- Carte mère et processeur intégré : 100 €.
- Mémoire 4 Go : 30 / 45 €.
- Alimentation 450 W : 40 / 60 €.

Soit un budget entre 210 et 285 €.

STOCKAGE

Le stockage ne fait pas partie des composants matériels car quel que soit le matériel venant du commerce ou pas, les disques durs peuvent être changés pour augmenter le stockage. L'offre est très large. Pour notre part, nous opterons pour des disques certifiés serveurs ou NAS tels que les WD RED et les Seagate. Ils sont généralement plus chers mais ils sont aussi plus robustes. Par exemple, un disque 6 To WD Red est vendu 190 €. A vous de choisir le nombre de disque souhaité. Si vous souhaitez en utiliser plusieurs, il sera possible de les mettre en RAID ou de les gérer indépendamment les uns et des autres.

DISTRIBUTIONS

Le choix est plutôt large. Il peut se faire sur les fonctionnalités et la communauté. Nous verrons les principales.

NAS4Free

Nas4Free est un fork du projet FreeNas. Il s'agit d'une distribution NAS (Network Attached Storage) open source. Elle s'appuie sur la distribution FreeBSD et par son ancienneté, elle est réputée pour sa stabilité, fiabilité et vitesse. Son environnement est très léger et peut-être installé sur une clef USB ou une carte Micro SD car celui-ci est chargé en RAM lors du démarrage. L'intérêt est d'offrir une machine ultra-puissante. Le système est compatible avec n'importe quelle plateforme matérielle. 1

Il prend en charge :

- Le partage des fichiers Windows, macOS, Unix/Linux ;

LEROY MERLIN PRÉSENTE

Maker Faire® Paris

DU 9 AU 11 NOVEMBRE 2018 - CITÉ DES SCIENCES ET DE L'INDUSTRIE

APPEL AUX MAKERS

NOUS RECHERCHONS

BRICOLEURS ★ MAKERS
DESIGNERS ★ CODEURS
GAMERS ★ ARTISTES
HACKERS ★ CREATEURS
INVENTEURS ★ YOUTUBERS
INGENIEURS ★ REVEURS

EN PARTENARIAT AVEC DESIGNSPARK PRÉSENTÉ PAR RS

INSCRIPTIONS JUSQU'AU 14 JUILLET 2018 SUR LE SITE

PARIS.MAKERFAIRE.COM

cité sciences et industrie

- Les formats ZFS, Raid, Raid-Z, Raid-Z2 ;
- Les protocoles réseaux actuelles, Wifi ;
- Les outils de monitoring.

Enfin, le projet est toujours actif et propose régulièrement des mises à jours.
Site : http://www.nas4free.org/

XPEnology

La distribution XPEnology est un fork de la version officielle Synology, embarquée dans les NAS de la marque. Elle peut s'installer sur une clef USB, une carte Micro SD ou un disque séparé.
Cette version est compatible avec n'importe quelle plateforme matérielle. Bien entendu, pour arriver à la même puissance offerte par un NAS Synology, comme la gestion de 10 disques de stockage, vous devrez calibrer le matériel. 2

Le système prend en charge :

- Le partage des fichiers Windows, macOS, Unix/Linux ;
- Les formats SATA2, Raid, AHCI, IDE Bios ;
- Les protocoles réseaux actuelles, Wifi ;
- Possède de nombreuses options pré-configurées.

De plus, vous bénéficiez des addons existants du fabriquant. Toutefois, vous ne devez pas oublier qu'elle est liée au fabricant, ce qui peut être problématique pour la stabilité et la pérennité.
Site : https://xpenology.org/

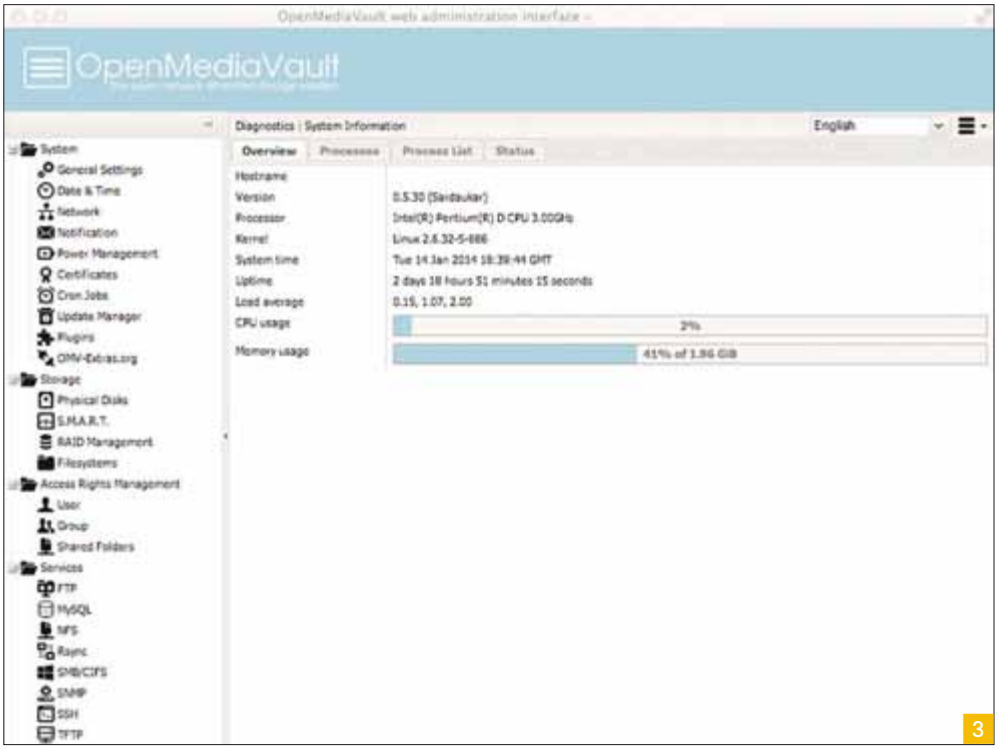
OpenMediaVault

OpenMediaVault est une distribution basée sur Debian, libre et gratuite. L'OS s'installe et se configure sur un disque. La distribution est reconnue pour sa stabilité. Son interface web se veut intuitive pour faciliter les différentes manipulations du NAS, tout en gardant une console pour les utilisateurs confirmés en lignes de commande. Cette solution offre la possibilité d'installer des paquets supplémentaires (*.DEB). Elle propose une liste de plugins (extensions) très fournie. Les plugins sont gérés par l'interface tout en gardant des bonnes performances. 3

Il prend en charge :

- Le partage des fichiers Windows, macOS, Unix/Linux ;
- Les formats ZFS, Raid, Raid-Z, Raid-Z2 ;
- Les protocoles réseaux actuelles, Wifi ;
- Les outils de monitoring.
- Nombreux plugins : Media serveur DDAP, BitTorrent

Le projet est très actif grâce à la communauté. Site : www.openmediavault.org



3

CONFIGURATION / PARAMÉTRAGE

La troisième étape passe par la configuration de votre NAS qui peut varier suivant la distribution que vous aurez choisie. L'ensemble de ces paramétrages peuvent être réalisés par l'interface web. Toutefois, les partages de dossiers dans votre réseau interne et externe sont les premières préoccupations de tous.

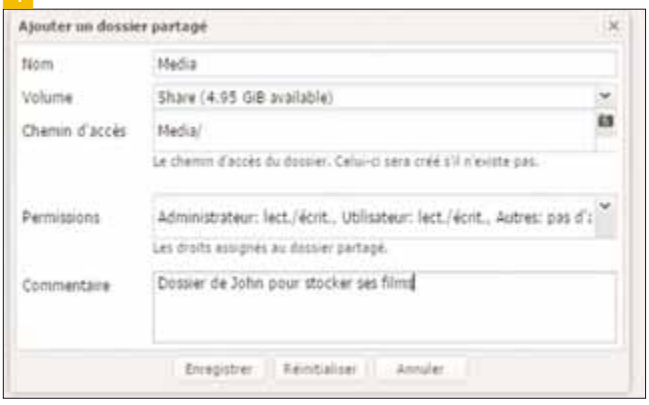
Partage

Dossiers

Il est important de créer un dossier de la taille que vous souhaitez par rapport à la capacité du disque. Pour cela, vous devrez renseigner les informations suivantes :

- Un volume sur lequel stocker les dossiers de partage ;
- Un chemin d'accès qu'il faudra rentrer dans l'explorateur de fichiers pour accéder au partage ;

4




2

- Des permissions par défaut, que l'on pourra affiner par la suite ;
- Un éventuel commentaire. 4

Les droits

Lors de la création du dossier partagé, vous définissez les privilèges et les ACLs (permissions) pour chaque utilisateur, c'est à dire, si celui-ci a les droits de lecture, lecture/écrire, ou aucun. 5 6

De nombreuses options existent comme la possibilité d'associer un utilisateur à un groupe, qui bénéficiera des mêmes options qu'un utilisateur. La mise en place de la configuration, s'effectue à partir de la section « services », du sous menu SMB/CIFS. 7

Vérification

La vérification passera par votre ordinateur ou un autre appareil. Vous verrez une nouvelle machine apparaître dans votre réseau. Bien entendu, si vous n'accédez pas à celle-ci, cela signifie que la configuration n'est pas complète ou mal paramétrée. Redémarrez votre NAS pour que les paramètres soient bien pris en compte. Si les problèmes persistent, revérifiez le matériel et la configuration.

Réseau / VPN

Côté NAS

Votre NAS se connectera directement au réseau local grâce à la distribution que vous aurez choisie et communiquera avec vos différents terminaux Toutefois, pour accéder à cet espace depuis l'extérieur, vous devez utiliser un VPN (Virtual Private Network).

Heureusement, il existe un plugin pour l'ensemble des distributions qui permet d'installer "OpenVPN" et de le configurer en quelques clics sans avoir à taper une seule ligne de script. Ce programme est disponible dans la bibliothèque de plugins et il vous suffit de choisir le plugin "openmediavault-openvpn". Une fois installé, vous trouverez un nouvel onglet OpenVPN dans la section "Services".

L'écran que vous obtenez comporte plusieurs options à configurer, qui sont :

- Port : par défaut le port est le 1194. Si vous posséder une box, vous devrez créer une règle de redirection de port. C'est à dire que si vous vous connectez sur votre box avec le port 1194, il faudra lui demander d'envoyer toutes les données vers le port 1194 de votre NAS.
- Vous choisissez le protocole de votre choix comme le protocole UDP.
- Une adresse : elle sera utile pour les machines qui se connecteront au VPN. Par défaut vous avez l'adresse 10.8.0.0/24, qui est modifiable pour retrouver une adresse du type 192.168.1.0/24.
- Gateway interface : cette option permet de définir quelle interface réseau va être utilisée pour le VPN. Par défaut, le champ est vide et vous devrez en attribuer une, par exemple : "eth0".
- Public Address : il s'agit de l'adresse publique de votre box, pour pouvoir s'y connecter depuis l'extérieur.

Pour connaître votre adresse publique, vous pouvez vous rendre sur n'importe quel site qui vous donnera l'information comme www.adresse-ip.com.

N'oubliez pas de sauvegarder les paramètres.

La dernière étape concerne l'ajout des certificats et des fichiers de configurations pour vos utilisateurs. Pour cela, allez sur 'certificates' et choisissez l'utilisateur et sauvegardez-le. Téléchargez les fichiers et envoyez-les à l'utilisateur. 8

Côté utilisateur

L'utilisateur qui souhaitera se connecter au dossier partagé du NAS doit au préalable installer 'OpenVPN' client sur sa machine et importer le fichier .ovpn préalablement récupéré. Il ne restera plus qu'à se connecter au dossier partagé.

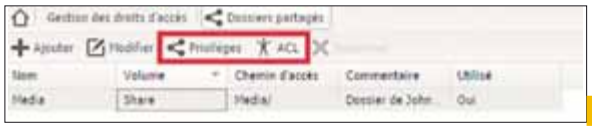
Nexcloud

Nexcloud est une application vous permettant d'obtenir une interface de navigation souple pour éviter de passer par une console pour les utilisateurs non avertis. Il est compatible avec toutes les distributions NAS.

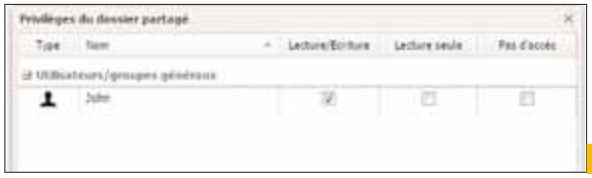
CONCLUSION

Monter un NAS n'est pas forcément très compliqué. La construction de celui-ci peut être une alternative intéressante par rapport aux appareils du marché. Cela va dépendre du nombre de baies. Il est parfois plus intéressant d'acheter un boîtier constructeur. Au niveau de la distribution et de la configuration, la construction d'un NAS n'est pas à la portée de tout le monde. Si vous souhaitez stocker/partager vos données en tout simplicité, privilégiez un NAS du commerce. Toutefois, avec un peu de temps et de patience, vous serez content de votre réalisation car chaque distribution se veut être simple et propose de nombreuses fonctions.

Vous pouvez aussi recycler un vieux PC et le transformer en NAS. L'investissement sera moindre, à condition, que la machine soit en bon état. •

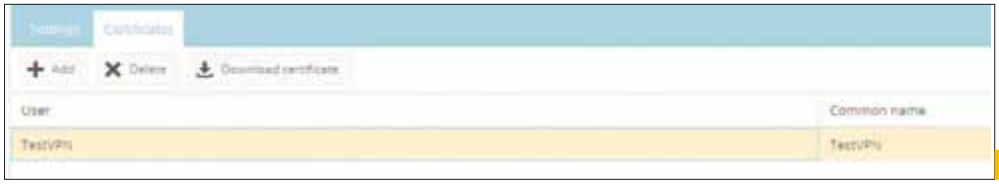


5



6


7



8

Denis Voituron
MVP, Speaker & Podcaster DevApps.be
.NET Senior Architect
@ http://oniryx.be
@DenisVoituron

niveau 100

SQL Server sur Linux, Microsoft s'ouvre au monde

Pourquoi Microsoft a besoin d'une version Linux ? Microsoft SQL Server est un système de gestion de base de données relationnel (SGBDR).

En 1989, Microsoft et Sybase codéveloppent une première version destinée aux plateformes Unix et OS/2. Jusqu'à la version 4.9, les logiciels Sybase et Microsoft SQL Server étaient pratiquement identiques. En 1993, à la suite de désaccords entre les deux sociétés concernant le partage des revenus, Sybase et Microsoft ont décidé d'arrêter de collaborer ensemble sur ce projet. Les produits ont ainsi évolué de façon différente avec des codes sources distincts. En 1994, Microsoft sort la version 6.0 puis 6.5 seul, sur la plateforme Windows NT. Ensuite, la version 7.0 est une réécriture majeure en C++ de l'ancien moteur Sybase qui, lui; était écrit en C.

Depuis lors, SQL Server fait partie des logiciels les plus importants de Microsoft. Les versions s'enchaînent et ajoutent toujours plus de fonctionnalités. Par rapport à ses concurrents que sont Oracle, MySQL ou PostgreSQL, SQL Server se distingue par le fait que c'est un SGBDR originellement multibase et multi-schéma. Il est possible de faire des requêtes nativement interbases.

Notons que Microsoft SQL Server reprend une grande partie du langage d'interrogation de données ANSI SQL, mais apporte beaucoup d'améliorations via des compléments du langages regroupés sous le terme T-SQL. Le Transact-SQL est une extension propriétaire de Sybase et Microsoft au langage SQL. De manière similaire au PL/SQL d'Oracle, Transact-SQL fournit le moyen d'étendre les fonctionnalités de base du SGBD.

Mais aujourd'hui, pourquoi SQL Server sur Linux est-il important, et pourquoi est-il nécessaire ?

Depuis plusieurs années maintenant, Microsoft mise beaucoup sur Azure, sa plateforme de Cloud. Avec ce changement, une approche strictement « Windows Only » n'a plus de sens. Si Microsoft obtient des revenus Azure à partir d'une version de SQL Server qui tourne sur Linux, alors c'est une victoire. Et on est bien obligé de constater que c'est une stratégie avantageuse. Depuis que Microsoft publie des logiciels multiplateformes, il augmente rapidement sa crédibilité dans le domaine concerné. Par exemple, nous avons pu constater cela pour Azure HDInsight, VSCode ou .NET Core.

D'autre part, Azure déploie beaucoup d'énergie pour intégrer les conteneurs dans le nuage. Les conteneurs sous Windows sont une chose, mais la communauté Docker est beaucoup plus présente dans le monde Linux. Et il se peut aussi qu'une version Linux de SQL Server facilite le jeu de Microsoft dans le monde des applications conteneurisées.

Peut-être plus important encore, une offre sur Linux facilite un certain nombre de partenariats avec d'importants fournisseurs de logiciels indépendants (ISVs) qui seraient soit impossibles, soit beaucoup plus difficiles avec une version qui ne fonctionnerait que sur Windows Server.

Nous pouvons résumer que Microsoft a besoin d'une version SQL Server sur Linux pour accroître son offre Azure, pour augmenter sa crédibilité dans les SGBDR et pour s'intégrer facilement dans le monde des solutions Docker et pour signer des partenariats plus importants encore avec les ISVs.

Il y a encore quelques questions, cependant. Y aura-t-il une version Open Source de SQL Server sous Linux ? Et y aura-t-il une version développeur de SQL Server qui fonctionne sur macOS (lui-même dérivé d'UNIX) ?

Installation en moins de 4 minutes

L'ADN de Microsoft est de fournir des solutions logicielles faciles à utiliser. Comme les autres produits de la multinationale, l'installation de SQL Server sous Windows ne déroge pas à cette règle : on exécute un programme d'installation (setup.exe), on répond à quelques questions et quelques minutes plus tard, une version du produit est utilisable.

A ma grande surprise lorsque j'ai voulu installer SQL Server sur Linux, Microsoft n'a pas perdu cette caractéristique fondamentale. En effet, il faut moins de 4 minutes pour installer et utiliser ce moteur de base de données sur un Linux existant.

La première étape est de vérifier quels sont les composants nécessaires pour installer ce produit. SQL Server 2017 est pris en charge sur Red Hat Enterprise Linux (RHEL 7.3 ou 7.4), SUSE Linux Enterprise Server (SLES 12 SP2) et Ubuntu 16.04. Il est également disponible sous une image Docker (1.8+) qui peut s'exécuter sur le moteur Docker sur Linux ou Docker pour Windows/Mac. Microsoft recommande d'une configuration système d'au moins 2 Go de mémoire, de 6 Go d'espace disque et d'un processeur x64 d'au moins 2 cœurs de 2 GHz.

Si, comme moi, vous n'avez pas de distribution Linux déjà installée, je vous conseille d'utiliser une VM Ubuntu préconfigurée dans Azure (1).

- Connectez-vous au **portail Azure** et ajoutez une nouvelle ressource.
- Recherchez une **VM Ubuntu Server 17.10**.
- Donnez-lui un **nom** (ex. *denis*) et encodez un **mot de passe** d'authentification.
- Choisissez une **taille de machine virtuelle**. Une B2S (~ 4 cents / heure) sera largement suffisante pour notre première installation.

(1) Vidéo d'installation de SQL Server sur Linux : https://youtu.be/z1_a3lYdzu4

- Après confirmation et quelques secondes plus tard, votre VM est créée et démarrée.

Depuis la page principale de la VM Azure, notez son **adresse IP** (ex. *40.114.137.190*) pour plus tard.

Cliquez sur l'onglet **Réseaux** (Networking) et ajoutez une **règle d'entrée au pare-feu** : ouvrir le **port 1433**, afin d'accepter le protocole utilisé par SQL Server lors de nos tests de connexions, à la fin de cet article. 1

Pour configurer **SQL Server sur Ubuntu**, exécutez les commandes suivantes dans un terminal SSH, pour installer le package mssql-serveur. Les procédures d'installation pour les autres distributions Linux sont similaires à celle-ci et sont reprises dans la documentation : https://docs.microsoft.com/sql/linux.

- Depuis Windows 10, connectez-vous sur le serveur Linux via la commande **Secure Shell** suivante (utilisez votre nom d'utilisateur et l'IP de votre serveur Ubuntu créé précédemment). Vous pouvez installer SSH depuis OpenSSH.com ou via Git-SCM.com. Notez que depuis la version 1803 de Windows 10, une version de OpenSSH est automatiquement présente sous Windows. Il vous suffit de l'activer via le panneau de configuration (Apps & features / Manage optional features).

```
C:\> ssh denis@40.114.137.190
```

- Sous session SSH, importez les clés GPG du référentiel public.

```
$ wget -qO- https://packages.microsoft.com/keys/microsoft.asc |
 sudo apt-key add -
```

- Enregistrez le référentiel des sources Microsoft SQL Server Ubuntu.

```
$ sudo add-apt-repository "$(wget -qO-
 https://packages.microsoft.com/config/ubuntu/16.04/mssql-server-2017.list)"
```

- Exécutez les trois commandes suivantes pour installer SQL Server et redémarrez le service si nécessaire.

```
$ sudo apt-get update
$ sudo apt-get install -y mssql-server
$ systemctl restart mssql-server.service
```

- À la fin de l'installation du package, exécutez le programme de configuration **mssql-conf** et suivez les instructions pour définir le mot de passe de l'administrateur système (SA) et pour choisir votre édition et votre licence.

1

```
$ sudo /opt/mssql/bin/mssql-conf setup
```

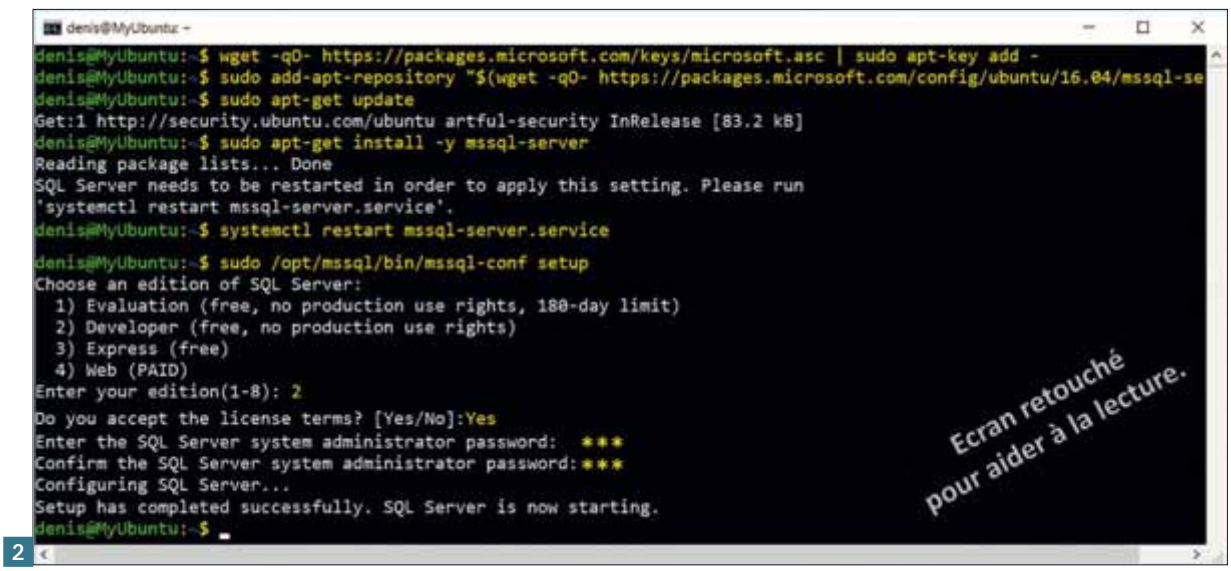
Les licences disponibles sont identiques à celles fournies sous Windows : une version d'évaluation de 180 jours, une version gratuite de développement, une version limitée fonctionnellement (Express) et des versions payantes (Web, Standard et Enterprise). 2

Quels sont les outils de connexion ?

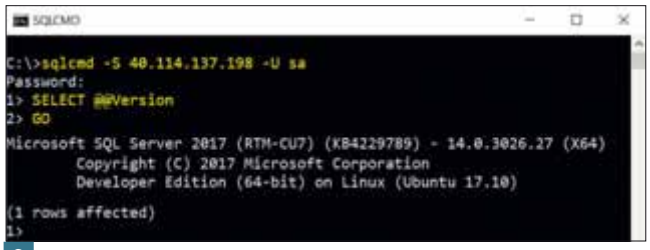
Une fois le serveur installé et configuré correctement, tous les outils et logiciels de connexion de SQL Server sous Windows peuvent être utilisés. Il n'y a pas de différence entre un SQL Server sous Windows ou sous Linux. Le protocole de connexion et les formats de fichiers sont exactement les mêmes. Avant de poursuivre l'exécution des instructions suivantes, vérifiez bien que vous avez ouvert le port 1433 de votre machine Linux (ce que nous avons déjà fait précédemment – étape 7 de la configuration Azure).

Pratiquement, pour se connecter à SQL Server, plusieurs logiciels simples et gratuits sont disponibles. Voici une sélection de mes outils préférés.

SqlCmd est un outil en ligne de commande très simple pour se connecter au serveur et pour exécuter une requête SQL rapidement. Cet utilitaire est présent avec la version Windows de SQL Server mais peut être téléchargé gratuitement depuis le centre de téléchargement de Microsoft. Après avoir renseigné le nom ou l'adresse IP du serveur (-S 40.114.137.198 dans mon cas) et le nom d'utilisateur (-U sa), le serveur distant me demande mon mot de

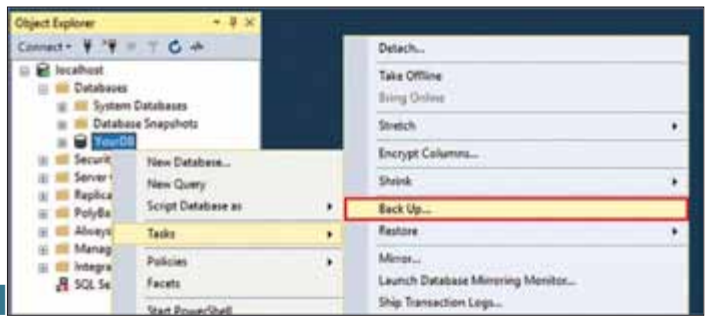


2



3

4



5

passe. Je peux ainsi écrire la requête SQL « SELECT @@Version » suivie de GO pour l'exécuter. Le serveur me retourne instantanément l'édition, la version et le système d'exploitation utilisé. 3

Cette première commande est souvent renseignée pour vérifier le fonctionnement de bout en bout de l'installation du serveur SQL. Depuis un poste client, vous avez pu ouvrir une connexion sur le serveur de base de données, vous avez pu exécuter une requête SQL et recevoir le résultat (dans notre cas, les détails de la version sur serveur).

SQL Operations Studio est un excellent éditeur de requêtes SQL développé par Microsoft. Il est principalement orienté « Développeurs » car il est très simple graphiquement et très puissant pour écrire et exécuter des requêtes SQL. Il se base sur le même noyau que l'application VSCode et propose la même interface d'utilisation. Cet outil est gratuit, open source et disponible pour Windows, macOS et Linux. Une fois installé et démarré, créez une nouvelle requête (File / New query) et appuyez sur le bouton [Connect] pour renseigner le serveur, le nom d'utilisateur et le mot de passe à utiliser.

Vous avez maintenant un éditeur de texte comprenant la colorisation syntaxique, les propositions Intellisense et des raccourcis clavier pour gérer et exécuter votre code. SQL Operations Studio est adapté aux développeurs qui doivent concevoir et maintenir du code SQL quotidiennement. De plus, il dispose d'une liste d'extensions à ajouter, pour améliorer la puissance de travail de cet outil (par exemple, l'excellent Redgate SQL Search).

Cet outil fera l'objet d'un prochain article reprenant les meilleurs outils SQL pour les développeurs. 4

SQL Server Management Studio (SSMS) est le plus ancien et le plus complet des logiciels de gestion de SQL Server. Il est présent depuis le début des distributions de Microsoft. Cet outil est fourni avec les versions payantes de SQL Server mais également en version Express gratuite. Il permet de se connecter et d'administrer les différents moteurs SQL Server (SSRS, SSIS, SSAS et le moteur SQL relationnel). SSMS contient un éditeur de scripts TSQL, avec la possibilité de regrouper l'ensemble de ceux-ci au sein d'une solution (comme sous Visual Studio).

Grâce à sa longévité, SQL Server Management Studio est le plus répandu et le plus complet. Il est probablement le logiciel préféré des administrateurs SQL Server car il possède une liste impressionnante d'écrans de configuration et de gestion. Bien que tous ces écrans génèrent des requêtes SQL, vous pouvez paramétrer finement le serveur en répondant simplement à quelques questions.

Comment migrer mes données, de Windows à Linux ?

La fonction de sauvegarde et de restauration de SQL Server est le moyen recommandé pour migrer une base de données de SQL Server sous Windows vers SQL Server sous Linux. Pour migrer une base de données, trois étapes sont nécessaires : [1] sauvegarder votre base de données Windows, [2] transférer le backup (.bak) sur le serveur Linux et [3] restaurer le backup dans une nouvelle base de données sous Linux.

Il y a plusieurs façons de créer un fichier de sauvegarde d'une base de données sous Windows. Les étapes suivantes utilisent **SQL Server Management Studio** (SSMS).

- Démarrez SQL Server Management Studio sur votre machine Windows.
- Dans la boîte de dialogue de connexion, entrez le nom de votre serveur SQL (localhost si vous êtes sur le serveur lui-même).
- Dans l'Explorateur d'objets, développez les bases de données.
- Cliquez avec le bouton droit de la souris sur votre base de données cible, sélectionnez Tasks, puis cliquez sur Back up. 5
- Dans la boîte de dialogue, vérifiez que le type de sauvegarde est Full et que la sauvegarde est sur Disk. Notez le nom et l'emplacement du fichier. Par exemple, une base de données nommée YourDB sur SQL Server 2016 a un chemin de sauvegarde par défaut de C:\Program Files\Microsoft SQL Server\...\Backup\YourDB.bak.
- Cliquez sur OK pour sauvegarder votre base de données.

Sans la présence de SSMS sur votre poste de travail, vous pouvez utiliser **SqlCmd** ou **SQL Operations Studio** pour vous connectez au SQL Server sous Windows et pour exécuter la requête T-SQL suivante.

```
BACKUP DATABASE YourDB TO DISK = 'C:\Backup\YourDB.bak'
```

Le fichier .bak ainsi obtenu contient l'ensemble des informations contenues dans votre base de données : données, procédures, sécurité, … Il est maintenant temps de transférer ce fichier vers le

serveur Linux, via scp (secure copy) et ssh (remote login), pour procéder ensuite à la restauration des données.

- Dans votre session Command Prompt, naviguez jusqu'au répertoire contenant votre fichier de sauvegarde.
- Utilisez la commande **scp** pour transférer le fichier vers la machine Linux cible. L'exemple suivant transfère YourDB.bak dans le répertoire *home* de *denis*, sur le serveur Linux qui a une adresse IP de 137.117.211.102.

```
$ scp YourDB.bak denis@137.117.211.102
```

- Dans la même session Command Prompt, connectez-vous à distance à votre machine Linux cible avec **ssh**. L'exemple suivant se connecte à la machine Linux en tant que *denis*. Vous pouvez maintenant exécuter des commandes sur le serveur Linux distant.

```
$ ssh denis@137.117.211.102
```

- Entrez dans le mode Super Utilisateur (sudo su)
  Créez un nouveau répertoire de sauvegarde (mkdir). Le paramètre -p ne fait rien si le répertoire existe déjà.
  Déplacez le fichier de sauvegarde dans ce répertoire (mv). Dans l'exemple suivant, le fichier de sauvegarde se trouve dans le répertoire personnel de *denis*. Modifiez la commande pour qu'elle corresponde à l'emplacement et au nom de fichier de votre fichier de sauvegarde.

```
$ sudo su
$ mkdir -p /var/opt/mssql/backup
$ mv /home/denis/YourDB.bak /var/opt/mssql/backup/
```

- Quittez le mode super utilisateur (exit) et la session ssh (exit). 6

La dernière étape consiste à restaurer le fichier .bak présent sur ce serveur Linux, dans une nouvelle base de données du Microsoft SQL Server Linux.

Pour restaurer la sauvegarde de la base de données, vous pouvez utiliser la commande T-SQL **RESTORE DATABASE**.

Sous **SqlCmd** ou dans **SQL Operations Studio**, connectez-vous à l'instance distante de SQL Server Linux, avec l'utilisateur SA. Entrez le mot de passe lorsque vous y êtes invité.

Entrez la commande RESTORE DATABASE suivante et exécutez-la.

```
RESTORE DATABASE YourDB
FROM DISK = 'var/opt/mssql/backup/YourDB.bak'
WITH MOVE 'YourDB' TO 'var/opt/mssql/data/YourDB.mdf'
MOVE 'YourDB_Log' TO 'var/opt/mssql/data/YourDB.ldf'
```

Vérifiez la restauration en listant toutes les bases de données sur le serveur : **SELECT Name FROM sys.Databases**.

Votre base de données restaurée devrait être répertoriée. 7

Quelles sont les différences avec SQL Server pour Windows ?

Globalement, toutes les fonctionnalités de SQL Server sont identiques entre la version Windows et la version Linux. Toutefois, certaines fonctionnalités et services ne sont pas encore disponibles sous Linux au moment de la publication de la version d'octobre 2017.

- **Moteur de base de données** : Transactional & Merge replication, Stretch DB, Polybase, Distributed query, Linked Servers autres que SQL Server, System extended stored procedures, Filetable,FileStream, CLR assemblies avec permissions externe ou unsafe, Buffer Pool Extension.
- **SQL Server Agent**
- **High Availability** : Database mirroring
- **Security** : Extensible Key Management, AD Authentication (Linked Servers & Availability Groups), 3rd party AD tools.

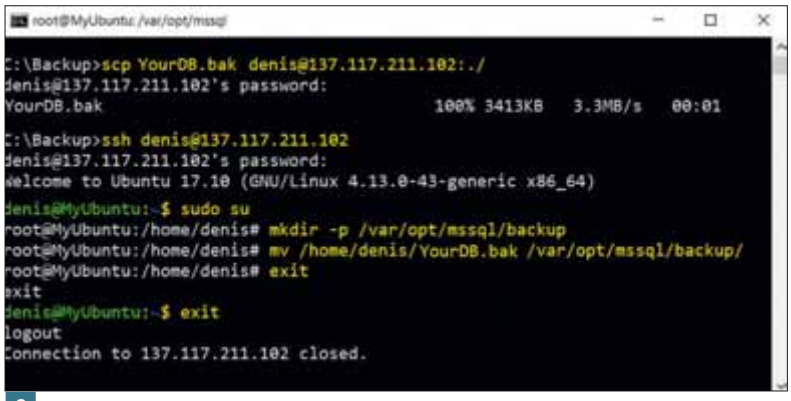
Services : SQL Server Browser, SQL Server R services, StreamInsight, Analysis Services, Reporting Services, Data Quality Services, Master Data Services, Distributed Transaction Coordinator (DTC)

Ces quelques fonctionnalités non supportées par la version Linux ne sont pas fréquemment utilisées dans nos applications. Dès lors, migrer une application pour utiliser SQL Server sous Linux revient à simplement adapter la chaîne de connexion qui pointe vers le serveur (voir ConnectionStrings.com). Il n'y a aucun changement à réaliser dans le code de votre application. Il n'est même pas nécessaire de la recompiler. C'est exactement la même application qui utilise une adresse IP différente : vers SQL Server Windows ou vers SQL Server Linux.

En allant plus loin, vous pouvez développer dans un environnement Windows, par exemple avec une version locale de SQL Server. Et utiliser une version Linux de SQL Server lorsque vous êtes dans votre environnement de production.

Au niveau des licences SQL Server, elles sont agnostiques et permettent le déploiement et l'utilisation sur les plateformes Windows ou Linux, au choix de l'utilisateur. Il n'y a pas de différence de prix pour acheter SQL Server pour Windows ou pour Linux.

Côté performances, selon Frédéric Brouard, il semble que la version Linux de SQL Server soit un peu plus rapide que sous Windows. Linux s'en sort mieux pour toutes les opérations dans les fichiers de données et en particulier pour les lectures ou le gain est



6

7

important (2,5 fois plus rapide que sous Windows). En revanche, Windows s'en sort généralement mieux pour toutes les opérations portant sur le journal de transactions. Mais il ne faudrait pas oublier certains détails importants : par défaut, Windows active beaucoup de services, il installe un antivirus maison et il dispose souvent d'une couche graphique. Ces quelques points pénalisent certainement les résultats des benchmarks.

Pourquoi utiliser la version linux ?

La réponse à cette question est probablement particulière à chacun de nous. Voici quelques raisons qui me tiennent personnellement à cœur.

C'est énorme ! Utilisant des logiciels Microsoft depuis des années, c'est tout bonnement énorme de voir le revirement de choix technologiques que cette entreprise a réalisés depuis l'arrivée de Satya Nadella. Ces seuls faits sont d'ordre sismique. Microsoft a, pour la première fois, publié l'un de ses produits serveur sur une plateforme autre que Windows Server. Vous vouliez la preuve que Microsoft est une entreprise très différente de ce qu'elle était il y a deux ou trois ans ? Voilà, c'est là.

Microsoft ne s'oriente pas vers l'open source pour ses produits serveurs. Même sur le plan pratique, il s'agit d'une interdiction ; légalement, les premiers développements et les morceaux de codes de nombreuses tierces parties dans ces produits serveurs prendraient une éternité à régulariser. Dès lors, ne considérez pas cela comme un prélude à ce que SQL Server devienne comme PostgreSQL ou MySQL. C'est plutôt Microsoft qui suit les traces de fournisseurs comme Oracle. Ce géant des bases de données n'a aucun problème à produire un produit serveur entièrement propriétaire pour Linux. La plus grande motivation de Microsoft est l'exposition et la part de marché. Les serveurs Linux restent beaucoup plus nombreux que les installations Windows Server, alors pourquoi ne pas tenter de capter une partie de ce marché ?

C'est une gifle chez Oracle. Un autre motif, directement déduit de ce qui précède, est que ce mouvement est un coup dur pour Oracle : tenter de prendre des parts de marché à cette entreprise, directement sur ses plateformes de prédilection. Oracle est la société qui récupère le plus de revenus sur le marché des bases de données commerciales, mais cela se traduit par des licences complexes et coûteuses. SQL Server possède le plus grand nombre d'instances sous licence. Dès maintenant, les clients Linux à la recherche d'une base de données de qualité commerciale soutenue par un fournisseur majeur n'auront pas à se contenter d'Oracle ou à envisager de configurer des instances sous Windows Server.

Il y a beaucoup de choses à aimer dans SQL Server. Pour ceux qui ne connaissent pas bien l'ensemble des fonctionnalités de SQL Server, il peut être difficile de comprendre l'attrait du produit pour les entreprises. Mais SQL Server 2014 et 2016 ont tous deux introduit des fonctionnalités qui plaisent à tous ceux qui essaient de construire des applications modernes : traitement en mémoire par table pinning, prise en charge de JSON, enregistrements cryptés, stockage et reprise après sinistre, intégration avec R pour l'analyse, etc. Avoir accès à tout cela est un bonus.

Microsoft a bien compris ce qu'est le Cloud. Au fur et à mesure que l'informatique d'entreprise se déplace dans le Cloud (même si certains resteront en interne), Linux restera une plateforme cible attrayante parce qu'il est à la fois économique et bien maitrisé en tant qu'environnement Cloud. Un monde où Microsoft n'a pas de présence sur d'autres plateformes que Windows est probablement un monde sans Microsoft.

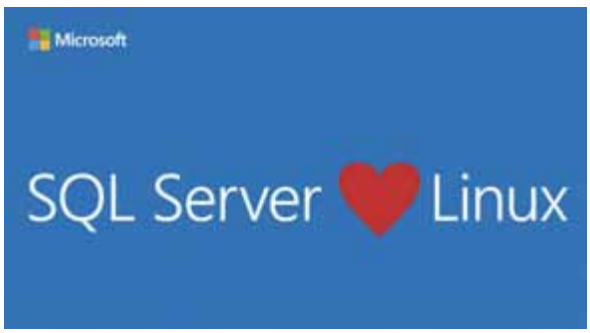
Ce n'est que le début. Bon nombre d'analystes pensent que d'autres applications serveurs Microsoft, comme Sharepoint Server et Exchange Server, pourraient faire le pas à moyen terme vers Linux. La difficulté est probablement de gérer le nombre de dépendances vers Windows qui ne sont pas facilement adaptables. SQL Server pourrait avoir été le premier candidat pour le déploiement vers Linux, en partie parce qu'il avait le plus petit nombre de ces dépendances. En effet, nous avons vu au début de cet article que SQL Server avait en fait ses racines dans le monde Unix, puisqu'il a commencé à migrer vers Windows.

Conclusion

Voici plus de 25 ans que Microsoft développe, maintient et fait évoluer le plus merveilleux des serveurs de bases de données. L'informatique se transforme constamment et le passage dans les nuages demande la mise en place de nouveaux choix technologiques. J'ai toujours été surpris de voir comment une entreprise de cette taille est capable de se remettre fondamentalement en question. Microsoft l'a déjà fait à de multiples reprises. Cela lui permet de rester à la pointe de nombreux domaines informatiques. Depuis quelques mois, elle nous propose une version Linux de son serveur de base de données. Elle conserve ses fondamentaux pour nous fournir des solutions puissantes mais très faciles d'utilisation. En moins de 5 minutes et via 7 commandes, nous disposons d'un serveur SQL complet, sécurisé et opérationnel. D'autres fournisseurs feraient bien de s'en inspirer ! Et tous les outils existants sous Windows ou sous Linux, même les plus anciens s'y connectent et l'utilisent. Je n'ai qu'un seul mot : chapeau !

Références

https://fr.wikipedia.org/wiki/Sybase
https://fr.wikipedia.org/wiki/Microsoft_SQL_Server
https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-release-notes
https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-migrate-restore-database
https://www.zdnet.com/article/why-microsoft-needs-sql-server-on-linux
https://www.zdnet.com/article/cloud-economics-pushed-microsoft-sql-server-onto-linux
https://blogs.microsoft.com/blog/2016/03/07/announcing-sql-server-on-linux
https://www.zdnet.com/article/microsoft-is-porting-sql-server-to-linux



François Tonic

Increvables mainframe et COBOL

Il y a des plateformes et des langages qui résistent à tout et longtemps. Depuis presque 60 ans, les mainframes sont le symbole de l'informatique d'entreprise : d'énormes machines installées dans des salles dédiées. IBM règne en maître sur le marché. Les concurrents ayant jeté l'éponge ou étant marginaux. Qui dit mainframe, dit COBOL. Le COBOL est le langage historique. Des centaines de milliards de lignes de codes COBOL tournent tous les jours dans le monde. Chaque année, des millions de lignes sont modifiées et rajoutées.
Ce legacy (= patrimoine) est tellement important dans certaines entreprises, qu'il est impensable de le migrer.

Le marché mainframe est cyclique. Il connaît des pics et des baisses selon les annonces matérielles. Avec le lancement l'an dernier de l'IBM Z14, ce segment du marché serveur a connu une hausse au dernier trimestre 2017 qui devrait se confirmer en 2018 avant de rebaisser à partir de 2019 – 2020. Le COBOL, langage historique du mainframe, est vieux de 59 ans. Les derniers chiffres sur son usage parlent d'eux-mêmes :

- **+70 %** des transactions se font en COBOL
- **95 %** des DAB ont du code COBOL
- **+ 220 milliards** de lignes de codes COBOL sont en production
- **3 000 milliards $** transitent par des systèmes COBOL
- **30 milliards** de transactions COBOL par jour

Vous l'aurez compris, le poids du langage, et par ricochet du mainframe, est considérable. Même si des migrations sont réalisées et que des entreprises "jettent" leurs gros systèmes, il faut le faire avec prudence et méthode. Réécrire ou migrer des applications COBOL non critiques / stratégiques est possible. De nombreux outils existent pour vous aider. Des entreprises n'osent pas y toucher pour x raisons : le coût, la durée d'un tel projet, l'incertitude d'une réécriture ou par manque de compétences et de connaissances des codes en production.

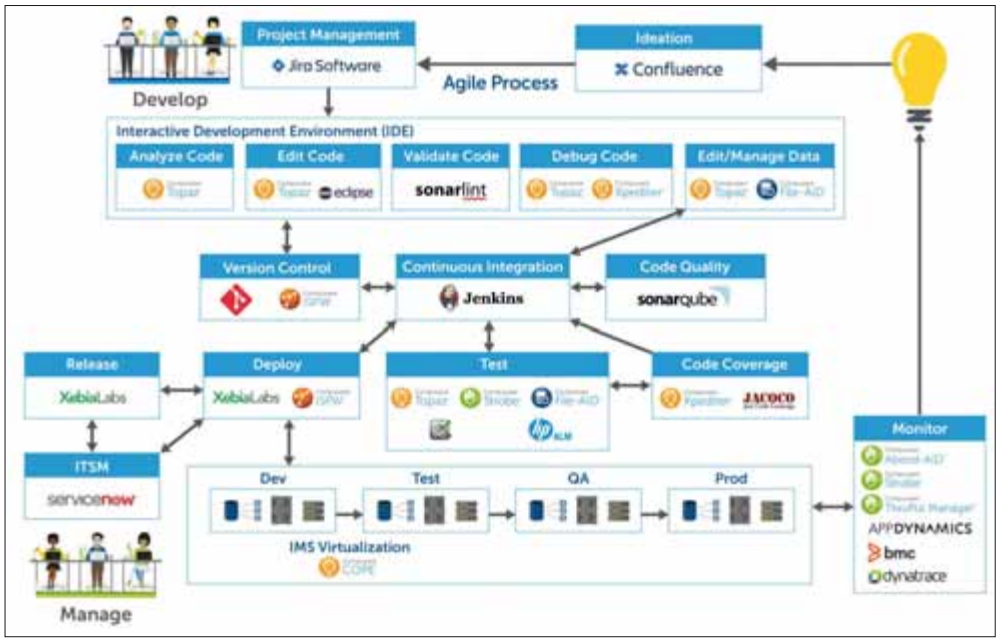
Distinguer mainframe et COBOL

Dans le marché mainframe, il faut distinguer deux réalités qui sont à la fois liées et non liées : la plateforme matérielle + système et les applications + COBOL. On pense toujours que COBOL = mainframe. En réalité, COBOL peut s'exécuter ailleurs : web, cloud, desktop, etc.
Aujourd'hui, même sans le savoir, vous pouvez attaquer un backend COBOL via des API, des couches middleware depuis votre frontend mobile, web, cloud, etc.
Le marché COBOL est un marché étonnant. Depuis 20 ans, on le donne pour mort, mais il est bien vivant. Il est globalement en baisse régulière, car finalement, il y a peu de nouveaux projets, de nouveaux mainframes, mais l'existant étant ce qu'il est, le récurrent est énorme pour le maintenir, l'entretenir, voire, le migrer, le réécrire. Certains parleront de rentes.
Aujourd'hui, le marché des outils COBOL se répartit principalement entre Micro Focus, Compuware, CA Technologies, LzLabs, Metrixware, COBOL-IT (racheté par Micro Focus en 2017)…
Les acteurs sont particulièrement actifs ces derniers mois. La transformation numérique touche les applications historiques, le besoin de modernisation permet aussi de dynamiser la demande et le besoin en outils. Des rachats consolident aussi une partie des acteurs. Micro Focus avait absorbé COBOL-IT en 2017. Ce dernier proposait des outils COBOL open source. La non-adhérence technique est une demande de plus en plus forte. Pour les acteurs historiques du marché, COBOL pèse très lourd dans l'activité : entre 50 et 60 %.

Un marché qui évolue

Mainframe et COBOL ne sont pas inertes. Ils bougent. L'an dernier, IBM présentait son dernier-né : le Z14. Pareillement, les éditeurs font évoluer leurs offres : DevOps, tests & qualités, sécurité, modernisation des applications, exposition d'API, connexions avec le Cloud. Les compilateurs se perfectionnent, les outils pour les développeurs intègrent toutes les fonctions modernes d'un IDE. On parle beaucoup de modernisation, de transformation numérique. L'idée n'est pas de jeter COBOL dans son entier, mais de faciliter l'accès aux données et au mainframe. Pour ce faire, on expose des services et des API. Le développeur ne doit pas se soucier de la nature de la source. Il ne verra que les API ou les modèles de données. Il s'agit aussi de sortir le COBOL du mainframe. Ce n'est pas une nouveauté mais Docker et le cloud facilitent cette tâche. Le mainframe à la demande n'est pas non plus un rêve, mais une réalité chez CA Technologies et Tmaxsoft. •

Compuware adapte les méthodes agiles aux Mainframes et nous explique pourquoi inclure le Mainframe à la chaîne d'outils DevOps.



Les grandes entreprises doivent sauvegarder de façon rigoureuse la qualité de leurs systèmes Mainframe qui sont critiques à leur métier alors que les exigences numériques les poussent à développer et à déployer du code mainframe plus rapidement et plus fréquemment. Une étude récente de Statista a révélé que 90% des développeurs de logiciels à l'international toutes technologies confondues ont commencé à implémenter DevOps. Dans cette dynamique, il est temps de mettre en place les métriques qui vont permettre de mesurer le succès de ces processus DevOps en termes de vélocité, de qualité et d'efficacité et de mettre en place une chaîne d'outils intégrée incluant le Mainframe qui va permettre d'améliorer ces KPI.

Les outils Mainframe traditionnels sont trop limités pour leur permettre d'atteindre cet objectif : ils ne sont pas adaptés aux méthodes de développement agile ni aux approches DevOps, leur interface utilisateur rebute la nouvelle génération de développeurs. Plus important et impactant encore, ces outils ne sont pas intégrés aux environnements d'automatisation des divers processus comme Jenkins, SonarSource ou XebiaLabs. Il est important qu'une chaîne d'outils DevOps puisse inclure le Mainframe au cycle de vie. Imaginez la valeur ajoutée que cela représenterait pour votre organisation si les équipes de développement, de QA/test et les opérations pouvaient travailler ensemble de manière unifiée avec le reste de votre entreprise y compris le Cloud. Ceci est tout à fait réalisable par la mise en place d'un plan de modernisation des environnements Mainframe soutenu dans le cadre duquel les équipes de développement se verront donner les moyens de travailler de manière collaborative selon les méthodes agiles et de mettre en place les meilleures pratiques DevOps dans le cadre de leur activité quotidienne d'amélioration du code Mainframe. Le premier point impliquera la mise en place de la gouvernance et de l'accompagnement. Le second point concernera les nouveaux outils à utiliser, citons l'automatisation des tests unitaires qui sont souvent réalisés manuellement sur le Mainframe. Ce qui est consommateur de temps, peu fiable, et source d'erreurs

Pourquoi automatiser les tests unitaires sur Mainframe et tester les API utilisées entre le Mainframe et les autres plateformes ?

Les tests unitaires sont des tests fondamentaux dans le processus de développement. Ils permettent d'identifier les anomalies dans le code le plus tôt possible, d'apporter des modifications de manière incrémentale plus rapidement pour améliorer la qualité du code tout en documentant plus précisément leur travail.

Les tests unitaires, historiquement effectués manuellement dans les environnements Mainframe, compromettent la capacité des développeurs Mainframe à adopter les bonnes pratiques agiles et DevOps. Afin de changer cela, Compuware a fait évoluer sa suite Topaz for Total Test et fait l'acquisition de XaTester de Xact Consulting A/S afin de permettre aux développeurs de créer rapidement des tests unitaires pour les programmes COBOL, PL/I, Assembler batch, mais également transactionnels CICS et IMS. Ces facilités sont particulièrement importantes pour la nouvelle génération de développeurs qui arrive sur le Mainframe et a souvent peu d'expérience de ces environnements. Les applications d'entreprise sont de manière générale déployées en environnement multiplateforme incluant notamment l'environnement distribué, le cloud et le Mainframe. Les services Web et mobiles destinés aux clients, aux employés, aux partenaires externes sont fournis via des applications qui dépendent presque toujours de la logique et des données des applications hébergées sur plusieurs plates-formes en « back-end » y compris le Mainframe. Lors des tests des nouvelles applications ou de certaines parties de ces applications, il est essentiel que les équipes puissent travailler, indifféremment, sur Mainframe, Cloud et les systèmes distribués. Compuware s'est également associée avec Parasoft, leader dans l'automatisation des tests de bout en bout. Le premier fruit de ce partenariat est l'intégration entre Parasoft SOAtest et Topaz for Total Test. Elle permet aux développeurs de tester rapidement et facilement les API entre les systèmes mainframe et non mainframe.

Pour conclure, les solutions peuvent être déjà utilisées par les équipes distribuées de type Jenkins, SonarQube et XL Release sont également intégrés avec les solutions Compuware et permettent de coordonner plus facilement leur travail sur Mainframe avec leurs activités dans les autres environnements •

François Tonic

Des plateformes qui évoluent

Comme nous l'avons dit en introduction de ce dossier, mainframe = COBOL, et inversement, n'est pas vrai, et ce, depuis de nombreuses années. Le mainframe a su s'ouvrir aux technologies du marché et COBOL est sorti de sa plateforme historique. Cette modernité va jusqu'à dématérialiser le mainframe, certaines entreprises veulent passer à autre chose, en conservant ou non l'histoire des applications. Mais en attendant, il faut continuer à supporter les applications COBOL, écrire de nouvelles lignes et surtout remplacer les développeurs COBOL qui partent en retraite.

Comme nous l'a précisé IBM, le mainframe n'est pas mort et l'écosystème bouge beaucoup et de nouveaux langages sont supportés : Java, R, Python, Swift ! Même si fondamentalement, le marché reste cyclique et lié aux annonces matérielles. Cependant, les éditeurs spécialisés demeurent très actifs et les rachats se multiplient.

Le réel problème de la compétence

Depuis 20 ans, nous en parlons : les entreprises ont du mal à trouver des compétences COBOL et particulièrement dans les nouvelles générations de développeurs. Or, pour maintenir les programmes existants, il faut des développeurs. Et il s'agit de ne pas perdre la connaissance acquise depuis 30 ou 40 ans. Car quand des développeurs partent en retraite ou change de société, la difficulté est de garder cette connaissance, la maîtrise du code et des environnements, les astuces pour que cela fonctionne même quand cela ne devrait pas, etc. L'entreprise doit absolument gérer cette transmission.

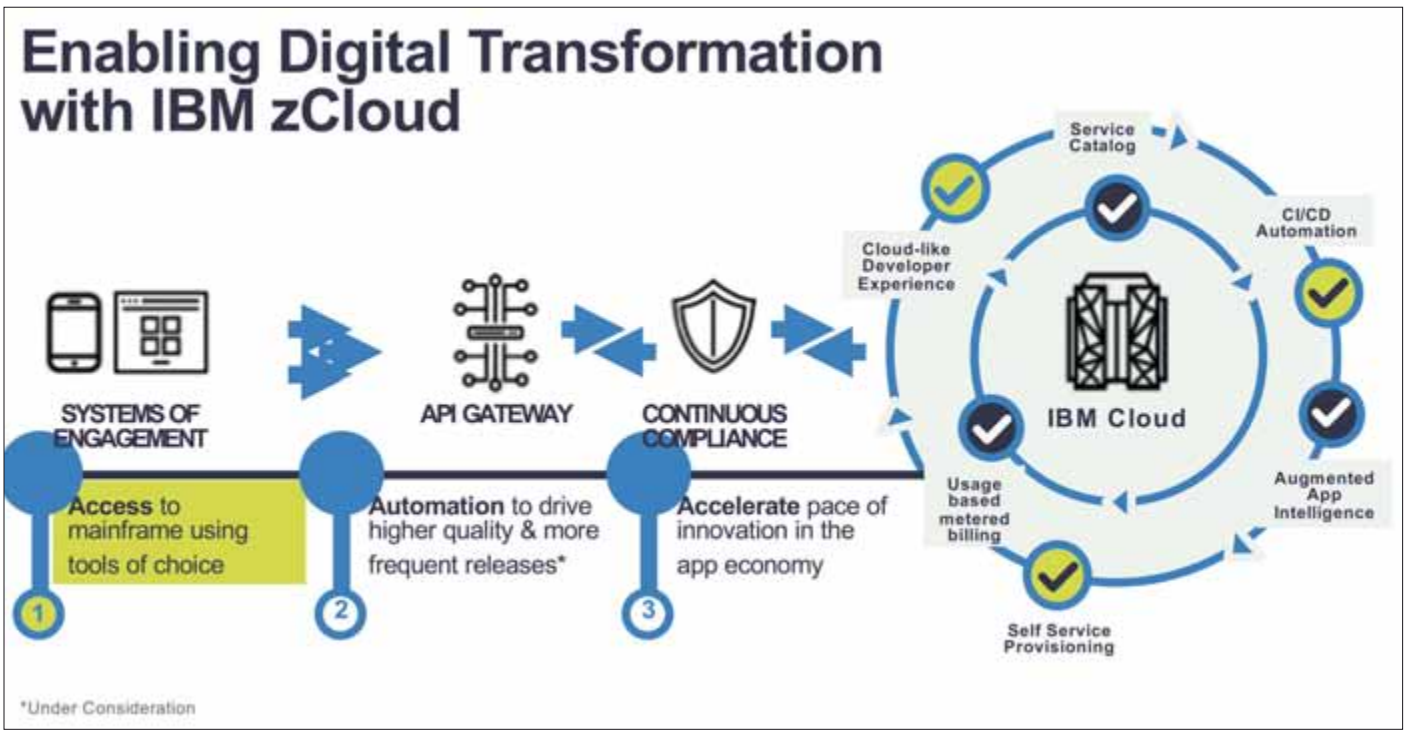
Pour les jeunes développeurs, le COBOL, et par extension le mainframe, est vu comme un dinosaure. Et parler de code historique, c'est tout sauf intéressant et excitant. Faites les tests !

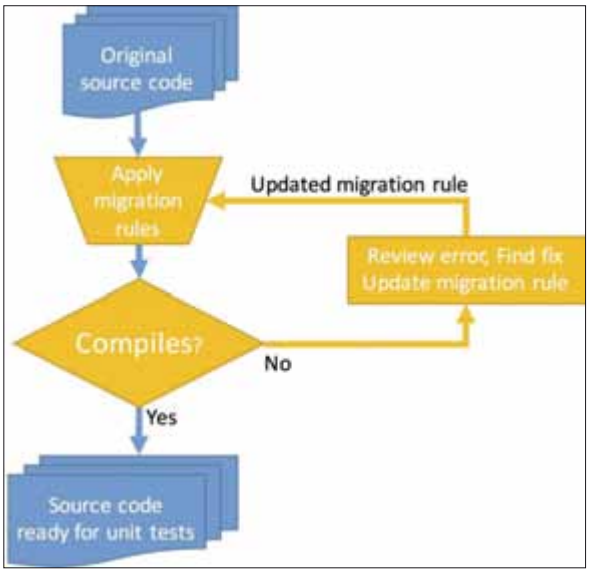
Entre « Tu vas maintenir nos bases de codes historiques COBOL » et « Tu vas bosser sur un projet Cloud et Spark », le choix sera vite fait. Et pourtant, de nombreux développeurs arrivant sur le marché, vont travailler sur les applications et projets historiques, COBOL, mais aussi sur du vieux Java.

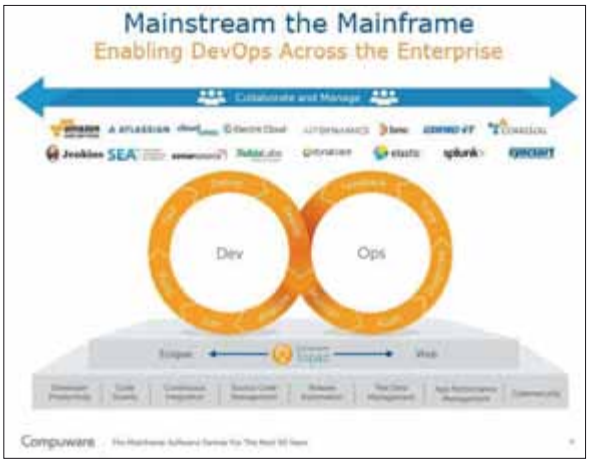
IBM met en avant, depuis plusieurs années, la zAcademy. Il s'agit d'une initiative pour former aux plateformes mainframe et COBOL. Elle existe depuis presque 10 ans. Des cursus complets sont proposés. Cette initiative fait suite à une précédente action d'IBM dans le domaine avec la zAcademic Initiative lancée en 2004. Aujourd'hui il est facile de trouver des formations COBOL & mainframe soit auprès des acteurs du marché, soit chez les centres de formation IT.

L'idée n'est pas de former des cobolistes, mais des développeurs ayant une compétence COBOL. Ce double profil est plus facile à « vendre ».

Sur la partie salaire, même si la compétence COBOL/mainframe est recherchée (mais ce n'est pas la compétence la plus rare, ni la plus demandée), il ne faut pas s'attendre à une surcôtation du profil. C'est même parfois l'inverse que l'on observe avec des salaires ne dépassant pas 30 000 €/an. Mais aujourd'hui, les salaires sont, souvent, alignés sur les fourchettes du marché, entre







35 et 50 000 €/an. Le freelance COBOL peut aussi trouver des missions, plus ou moins longues. Les tarifs journaliers varient énormément, nous avons vu des tarifs à -300 € / jour. Soyez particulièrement vigilant sur ce point, ainsi que les frais pris en charge ou non.

CA Technologies décrit trois profils du développeur COBOL :

- Le dév de +50 ans : il ne va pas changer d'environnement, il connaît parfaitement son environnement et son code. Souvent proche de la retraite
- Le dév de 35-40 ans qui bascule sur mainframe pour les besoins ponctuels ou réguliers pour écrire ou réécrire le back-end ou le frontal. Il connaît le COBOL
- Le jeune dév : il est plus jeune, avec ou sans expérience, il connaît (un peu) le mainframe et/ou le COBOL. Il a acquis une compétence sur ce domaine.

Bien entendu, nous restons dans des types d'entreprises et des secteurs précis : banque/assurance (notamment dans les mutuelles), grandes administrations publiques, grandes entreprises/industrie. Les intégrateurs sont régulièrement demandeurs de ces compétences.

Les outils

Une partie des développeurs COBOL ou utilisant des API mappant les services et données mainframe n'ont jamais vu d'écran vert (ah la chance). « On utilise les mêmes outils que pour développeurs en C, C++. » nous a expliqué IBM France. En effet, il est possible d'utiliser les IDE du marché comme Eclipse ou Visual Studio.

L'arrivée des systèmes Z14 d'IBM a été l'occasion d'étendre les API, les jeux d'instructions des plateformes. Les compilateurs COBOL savent les utiliser même s'il y a du travail à faire sur le code et surtout recompiler les codes.

IBM met en avant les possibilités de modernisation des workloads tournant sur les systèmes Z, notamment en virtualisant ou en passant par des conteneurs. L'ouverture vers les langages et les environnements les plus récents offre là aussi de nouvelles perspectives pour ces machines. La modernisation du socle technique et des workloads permet d'introduire dans les applications (en dehors de COBOL classique) les dernières versions de Java, d'être plus réactif sur les allocations et désallocations des ressources et des workloads, introduire de l'agilité dans les process.

Sur la partie purement COBOL, CA Technologies voit deux approches :

- Modernisation des apps actuelles : on réécrit une partie du COBOL dans un autre langage, typiquement Java. Tendance forte depuis 10 ans.
- Introduction des API et webisation des frontaux. Là encore, ce n'est pas une nouveauté. Il y a 20 ans, on parlait de web to host pour moderniser les frontaux.

Bref, aucune révolution, mais plutôt une autre manière d'exploiter le patrimoine et pérenniser les investissements réalisés depuis 30 ou 40 ans. Il est souvent plus facile de refondre les interfaces et les front-ends que de réécrire le COBOL. La réécriture, surtout pour les codes critiques, reste une opération longue et risquée. Car, il n'est pas toujours facile de retrouver le même niveau fonctionnel surtout si la documentation technique est lacunaire, ou obsolète, ou par manque de connaissances des codes existants.

Plusieurs COBOL

Le langage COBOL (Common Business Oriented Language) remonte à la fin des années 1950 avec COBOL 60. L'objectif était de créer un langage pour les entreprises, orienté métier/business. Le langage évolue rapidement avec les versions 61, 65, 68 et 74. La version suivante fut longue à être finalisée. Les retours sur les préversions furent mauvais. Finalement, c'est en 1983 que COBOL 84 fut finalisé, avec des ajouts en 89, 91, 93 et 94.

Plusieurs éditeurs engagèrent des travaux pour injecter de la programmation orientée objet dans le langage. Les travaux aboutirent à COBOL 2002. Puis les premières versions de COBOL fonctionnant sur les frameworks .Net apparaissent dans les années 2000. La dernière version standardisée est COBOL 2014.

Les outils et compilateurs COBOL ne supportent pas forcément les dernières versions du langage. Par exemple, GNUCobol supporte la version 2014. Et il existe aussi des variantes du langage selon les plateformes HP, Unisys, Futjitsu, etc. Au-delà du langage, les éditeurs et constructeurs ont régulièrement ajouté des extensions, notamment dans les compilateurs (IBM, Micro Focus, etc.). Ce qui ne facilite pas les migrations ou même de changer les outils de développement car certaines extensions peuvent être propriétaires.

Aujourd'hui, on parle beaucoup de migration, de refonte, de modernisation. Soyons clairs, migrer son patrimoine COBOL/Mainframe n'est pas simple. Comme nous l'avons dit, toucher au code critique ne doit pas se faire à la légère et le projet doit être cadré et suivi. Faites-vous la main sur du code non critique. Et il faut être conscient que ce genre de projet prendra parfois plusieurs années. Moderniser des apps COBOL est dans la tendance actuelle. L'idée est de rafraîchir les frontaux avec des technologies actuelles pour supprimer les écrans verts, par contre, on ne touche pas aux codes fonctionnels.

D'autre part, vous pouvez coder et maintenir du COBOL sans avoir de mainframe. La plupart des outils/IDE fonctionnent sur les plateformes desktops.

Le cloud peut aider à éteindre le mainframe

La migration intéresse aussi les acteurs du cloud public. Ainsi, AWS, et les partenaires, propose des couches de services pour faciliter cette opération. Une partie des opérations repose sur l'émulation mainframe dans les instances cloud pour faire du re-hosting/replatforming. Il existe plusieurs solutions comme Micro Focus Enterprise Server, OpenFrame, Oracle Tuxedo ART. Cette migration peut se réaliser pour les applications COBOL, le batch, le CICS. Il faut alors migrer les briques mainframe sur l'environnement cloud, mais attention, il faut procéder avec méthode et parfaitement évaluer la plateforme cible et voir les équivalents des stacks mainframes.

Au niveau des instances, pas de zOS, ou équivalent, mais du Windows ou du Linux. Cela signifie qu'il faut les piles techniques sur ces environnements : 1 ordonnanceur, impression, sécurité, gestion des codes, SGBD, les runtimes d'exécution. Pour les codes, une phase de recompilation sera à prévoir, une réécriture du code (souvent limitée) sera à prévoir. Mais bien entendu, il faudra tester, au préalable, les codes et vérifier leur bon fonctionnement.

Il y a du travail d'ingénierie et de rétro-ingénierie à réaliser et des processus d'automatisation à mettre en place pour faciliter la transition 2. Comme toujours, la connaissance de l'environnement de départ est cruciale. Vous devez avoir une cartographie précise et à jour de votre mainframe et des assets. Bien entendu, il existe des outils de découverte pour créer les modèles, mais rien ne vaut la connaissance initiale. La réussite, ou l'échec, du re-hosting/replatforming dépend beaucoup de ce pré-requis.

D'une manière moins radicale, vous pouvez mettre en place une architecture hybride : mainframe et le cloud. La connexion pourrait être assurée par les API et les passerelles.

Dans tous les cas, il faut prévoir de nombreux services cloud pour faire tourner vos assets.

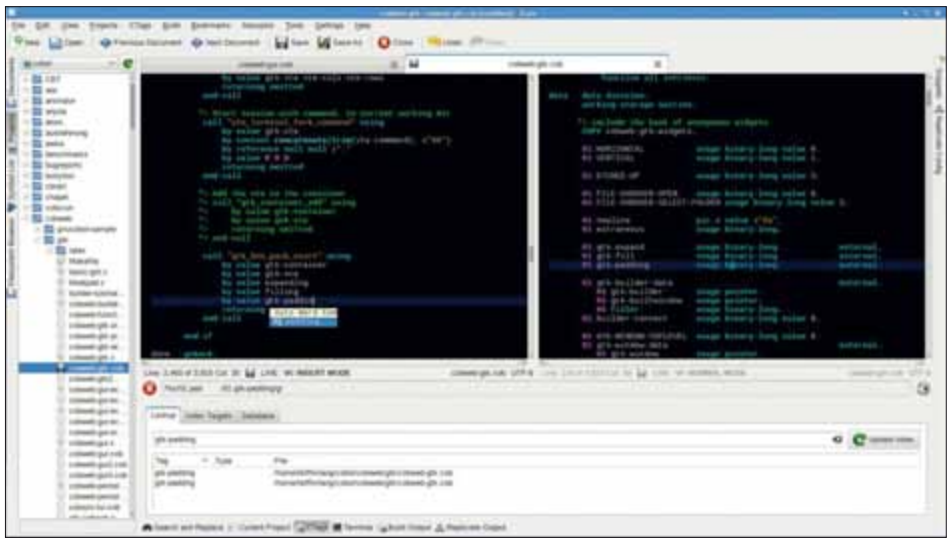
Fondamentalement, plusieurs pratiques sont possibles :

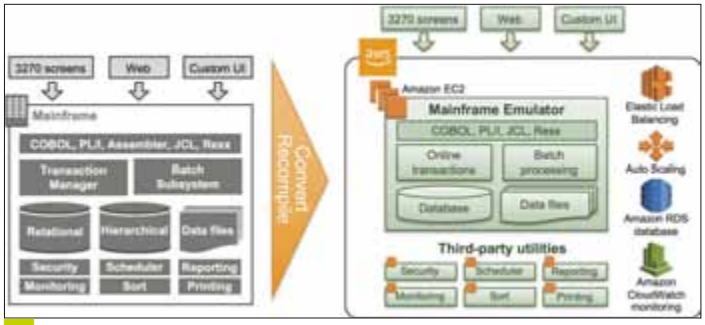
- Migration avec refactoring automatisé pour gagner du temps. Concerne principalement les assets COBOL vers une stack plus moderne ou vers un runtime COBOL
- Migration vers un re-hosting mainframe émulé : on fait du replatforming pour faire tourner les assets applicatifs avec un minimum de réécriture/d'adaptation. Les interfaces front sont les mêmes, avec les mêmes apps
- Approche hybride orientée donnée : on ne migre pas les workloads, mais on étend le mainframe avec des services agiles et de big data déployés sur des plateformes cloud comme AWS ou Azure. L'idée est d'étendre les capacités de traitements, utilisation d'outils plus modernes.
- Approche hybride orientée nouvelles expériences : l'idée forte est d'étendre les usages avec des assistants vocaux, des apps mobiles. Le mainframe reste le cœur, mais on étend les expériences utilisateurs même s'il ne voit jamais la plateforme legacy. Les services cloud apportent dans ce cas les éléments techniques nécessaires pour ces nouvelles approches.

Oui, le mainframe a aussi droit aux piles techniques les plus récentes

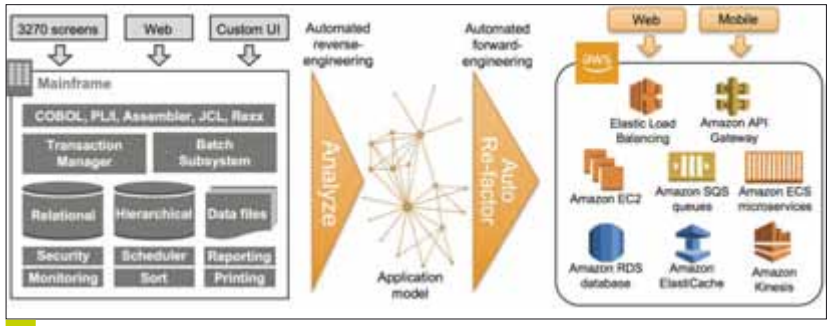
Si, si, je vous assure, le mainframe sait être moderne. Ce n'est pas seulement un écran vert et de grosses machines. IBM a introduit, depuis plusieurs années, une version Linux et le support de Java. Mais aujourd'hui, on peut faire tourner de nombreux langages. Et on peut y déployer des services cloud (IBM ZCloud) pour mettre en place des catalogues de services ou pourquoi ne pas y mettre des conteneurs.

Récemment, CA Technologies a dévoilé le projet ouvert Brightside. Il se compose d'un CLI (command line interface), d'une toolchain et de frameworks agiles, extensible via des plug-ins. Il s'agit de pouvoir proposer aux développeurs une autre expérience de développement sur/pour mainframe. Brightside se positionne en front et en middleware, la partie API gateway fait le lien avec le matériel. •





1



2

QUESTIONS – RÉPONSES À Stéphane Croce

COBOL-IT est connu dans le monde COBOL pour son offre open source et particulièrement son compilateur. La société avait été créée en 2008. Fin 2017, la société a été rachetée par le leader du marché COBOL, Micro Focus. Stéphane Croce, fondateur de COBOL-IT répond à nos questions.

Le mainframe est souvent vu comme un environnement très fermé et le COBOL un langage dépassé. Comment et pourquoi avez-vous décidé de miser sur une approche Open Source pour vos outils ?

Le mainframe est qualitativement faussement perçu de nos jours. IBM a depuis longtemps ouvert ses grands systèmes aux dernières technologies en faisant la promotion de Linux par exemple. Le véritable problème reste le coût du mainframe par rapport au volume décroissant des applications de plus en plus décommissionnées au profit d'applications packagées et exploitées sur des systèmes plus petits et moins chers. Le COBOL est le langage par excellence pour bâtir de solides programmes de gestion et il est très loin d'être dépassé ! Il reste le langage le plus utilisé au monde avec plus de 200 milliards de lignes en activité ! Malheureusement, nous avons entendu ces derniers mois une théorie d'inepties mélangeant allègrement l'obsolescence des ordinateurs, la vétusté des applications des grands systèmes, en pointant du doigt le langage COBOL, qui comme d'habitude, sert « d'usual suspect » pour masquer un manque d'anticipation dans l'urbanisation et la gestion des systèmes d'information. L'open source permet de réduire les coûts de la dette technique et de basculer ainsi une partie des budgets sur la modernisation des systèmes d'information par un principe vertueux de vases communicants. C'est grâce à cette nouvelle équation que l'on peut envisager une démarche où COBOL a toute sa place dans des architectures virtualisées dans le Cloud, logées dans des containers tout en s'appuyant sur un modèle « DevOps » permettant une transformation digitale des entreprises fondée sur l'équilibre subtil entre le Legacy et les nouvelles technologies.

On parle beaucoup de modernisation, de transformation des applications COBOL. Concrètement, de quoi parle-t-on ? Est-ce de la réécriture / migration des codes COBOL ou on modernise le COBOL avec d'autres technologies ?

On parle effectivement beaucoup, mais on fait peu ! Il y a aujourd'hui pléthore de solutions, allant de la réécriture automatique du code, la quadrature du cercle de la supposée transformation en Java (sic ! note de l'éditeur), et toutes sortes de systèmes de migrations ou de restructurations de code plus ou moins sérieuses. Avec plus de 200 milliards de lignes COBOL en production, cela reste un marché très juteux à conquérir. Cela concerne des applications qui ont été développées il y a une ou plusieurs décennies avec une perte partielle ou totale des connaissances. Cela laisse la place à toutes sortes de solutions, certaines assez solides, d'autres totalement farfelues, voire même très dangereuses pour la stabilité de l'entreprise qui s'y laisserait prendre. Ce que l'on oublie le plus souvent, c'est que l'on ne peut pas appliquer la même recette pour toutes les applications développées en COBOL. Certaines sont éligibles à une transformation, d'autres à un réhébergement sur une autre plate-forme, etc. Pour certaines, leur état est tellement délicat qu'il vaut mieux envisager la réécriture si toutefois on sait encore ce que fait l'application. Les « spécialistes » oublient l'essentiel : le caractère unique de chaque application et en fonction de sa typologie, l'orientation vers l'une ou l'autre des solutions applicables.

Comment les entreprises gèrent-elles les applications COBOL ? Réécriture ? Migration totale ? Surtout on ne touche à rien ?

Tout dépend de la démarche employée en amont : certaines sociétés cèdent aux sirènes de compagnies promettant une transformation automatique, mais qui va complexifier l'application au point qu'elle ne sera même plus maintenable. D'autres opteront pour une démarche de migration industrialisée permettant en amont de mettre en lumière les éventuelles difficultés qu'il faudra surmonter en y associant un coût. D'autre consultent beaucoup, mais décident qu'il est effectivement urgent de ne rien faire !

Comment fonctionne votre compilateur ? Quel est le projet type pour utiliser vos outils ?

Technologiquement parlant, notre compilateur transforme le code COBOL en code C et compile le résultat avec le compilateur C de la plate-forme ciblée. C'est une opération purement mécanique. Le développeur COBOL ne voit que du COBOL, et ce de la phase de développement jusqu'à celle de debug de son application. Cette technique permet une interopérabilité accrue avec les systèmes d'exploitation de type Linux / Unix, mais aussi avec une large gamme d'outils tiers comme les bases de données ou les moniteurs transactionnels, dispositifs de supervision, etc.

La structuration de notre compilateur nous autorise une organisation très agile, de type DevOps permettant d'implémenter en un temps record de nouvelles options facilitant la reprise d'applications COBOL avec des dialectes disparus et provenant de diverses plates-formes. Nos outils sont utilisés par nos partenaires certifiés dans des projets de migrations d'applications existantes en déroulant un Protocole très précis que nous avons défini.

Il permet d'identifier en amont toutes les difficultés potentielles liées à la construction de l'application à migrer – cela nous permet d'annoncer clairement la couleur au client en dressant une liste exhaustive de ce qu'il faudra implémenter dans notre compilateur pour assurer un portage sans changement de son code source original. C'est cette méthode qui a fait notre succès dans les plus grandes entreprises partout dans le monde.

On parle beaucoup de mainframe à la demande notamment sous forme de services cloud. Qu'en pensez-vous ? Quel avenir pour COBOL ?

C'est une histoire de contenant et de contenu. Le Cloud est à la mode aujourd'hui. Demain, nous aurons peut-être la technologie Edge Computing qui offrira encore plus d'avantages par rapport aux modèles actuels. La longévité du COBOL prouve que c'est un langage adaptable et qui a su traverser plusieurs décennies en remplissant parfaitement son rôle, il a su se marier en front end à des langages dits plus modernes. Aujourd'hui, nous réfléchissons à utiliser des algorithmes dits de « Deep Learning » pour mieux comprendre le fonctionnement de ce code mal documenté afin d'introduire des notions de maintenance prédictive, une sorte d'IA pour le legacy. Certaines études démontrent qu'il faudrait 5 millions de développeurs programmant pendant 65 ans pour redévelopper le patrimoine COBOL mondial ! Alors, pas de panique, que le COBOL soit exploité sur mainframe ou pas, il a encore de beaux jours devant lui ! •

IBM i + Eclipse = pari gagnant

En 2010 (derniers chiffres connus), le système d'exploitation **IBM i** (et ses prédécesseurs OS/400 et i5/OS) était la plate-forme IBM disposant du plus grand nombre de clients (**250 000 entreprises** environ clientes IBM i dans le monde). L'IBM i dans ses différentes itérations a été vendu à plus d'un million d'exemplaires depuis son introduction en juin 1988.

En recul dans sa commercialisation, l'IBM i reste dans beaucoup d'entreprises la plate-forme informatique de référence, celle sur laquelle sont hébergées les applications et les données vitales. Ses atouts ? Un serveur d'applications, une base de données et un système de stockage interne hors pair. Elle a su évoluer avec le temps pour devenir le couteau suisse de l'informatique en intégrant de multiples fonctions (JAVA, PHP …) et en s'ouvrant à d'autres systèmes (AIX, Linux).

Son défaut majeur ? Un environnement de programmation désuet – **SEU** – et un langage de programmation peu moderne – **RPG**.

Aujourd'hui les entreprises peinent à trouver des spécialistes du RPG. Le renouvellement des compétences RPG est en effet très limité, ce qui amplifie un phénomène de fond : la raréfaction des développeurs RPG. Cette carence s'accentue par une vague massive de départ à la retraite programmée sur les 10 prochaines années associée à une forte volonté des grands éditeurs et intégrateurs de gérer à leurs profits cette raréfaction. Au final, il s'agit bel et bien d'un risque énorme pour les entreprises de perdre leur capital et savoir-faire métier, sans pouvoir assurer la relève.

Le langage n'est pas le principal problème

Dans un contexte de crise économique, on pourrait penser qu'il est facile de trouver des candidats disponibles pour les postes de développeur RPG. Malheureusement, il n'en est rien. En effet, le langage RPG n'est plus attractif comparé aux langages modernes. Les applications cibles ne sont pas « sexy » notamment parce que la couche de présentation graphique n'est pas développée en RPG.

Autre phénomène : il n'est pas évident de se mettre en avant lors d'une soirée devant les copains quand on programme en RPG ! Il est même difficile d'expliquer que la majeure partie du business des entreprises est programmée dans ce langage.

Ceci étant, le langage RPG est procédural, facile à appréhender et ne constitue pas à lui tout seul le point de blocage quant à l'embauche de jeunes développeurs.

L'environnement de développement : la double peine

L'accès à l'environnement de développement RPG natif (SEU) se fait via des émulateurs 5250 en mode caractère. L'outil de travail numéro 1 du développeur étant l'éditeur de texte, les développeurs IBM i passent le plus clair de leur temps dans l'éditeur SEU datant des années 80 !

Lors de leur formation, les stagiaires considèrent l'accès à l'IBM i en 5250 comme une sorte de minitel antédiluvien. Certains d'entre eux abandonnent purement et simplement lorsqu'ils comprennent qu'ils vont passer leurs journées dans un tel environnement.

Cet outillage est clairement un frein pour le recrutement de jeunes talents et constitue un filtre de sélection naturelle retenant certes les plus motivés mais pas forcément les plus talentueux…

La comparaison entre l'interface 5250 et les interfaces graphiques modernes est tellement désavantageuse pour le 5250 qu'il ne faut pas espérer que les jeunes recrues deviennent des experts dans l'utilisation du SEU … même en Inde ! Leur efficacité dans un tel environnement restera donc très limitée pour la plupart d'entre eux.

La solution : Eclipse

Par quels outils remplacer l'outillage traditionnel sous SEU ? Il n'y a désormais plus beaucoup de questions à se poser : **Eclipse** est devenu un standard de fait.

Eclipse fournit en effet un cadre permettant de développer des outils de développement intégrés largement utilisés dans le monde Java (Eclipse, lui-même, est développé en Java). Faire profiter les développeurs RPG de cette infrastructure présente plusieurs avantages :

- environnement moderne, intégré, attractif pour les jeunes développeurs
- fonctionnalités ouvertes sous forme de plugins
- rapprochement des développeurs Java et RPG
- bénéficie d'une forte communauté, dynamique, active et internationale

Etant largement utilisé dans le monde Java, Eclipse dispose d'une multitude de plugins de gestion de sources, gestion des anomalies, gestion de projet, tests unitaires, etc. Tous ces plugins sont mis en œuvre aujourd'hui dans le cadre de méthodes projet agiles et constituent une opportunité d'évolution pour les méthodes de développement IBM i.

Outils : opter pour l'Open Source

Les plugins Eclipse sont généralement développés en EPL (Eclipse Public Licence), modèle de licence qui permet à tout un chacun de modifier, compiler et éventuellement commercialiser des plugins. Ceci favorise l'émergence de solutions professionnelles basées sur des plugins Open Source.

Il devient alors possible de bâtir un atelier de développement RPG sur mesure en assemblant les plugins qui conviennent aux modes de fonctionnement et aux outils de son entreprise. Il suffit seulement que **les plugins accèdent de manière native aux ressources Eclipse** et n'enferment pas l'utilisateur dans un cadre propriétaire.

Cet atelier devient un logiciel résultant de l'assemblage de plugins dans lequel on configure les fichiers de paramétrage communs à l'ensemble des développeurs et que l'on distribue sur les postes de travail. Une bonne pratique communément partagée est de faire une mise à jour annuelle.

Dans certains cas, le développement de plugins spécifiques et/ou la contribution à l'évolution de plugins existants doit être considéré.

A propos du ROI

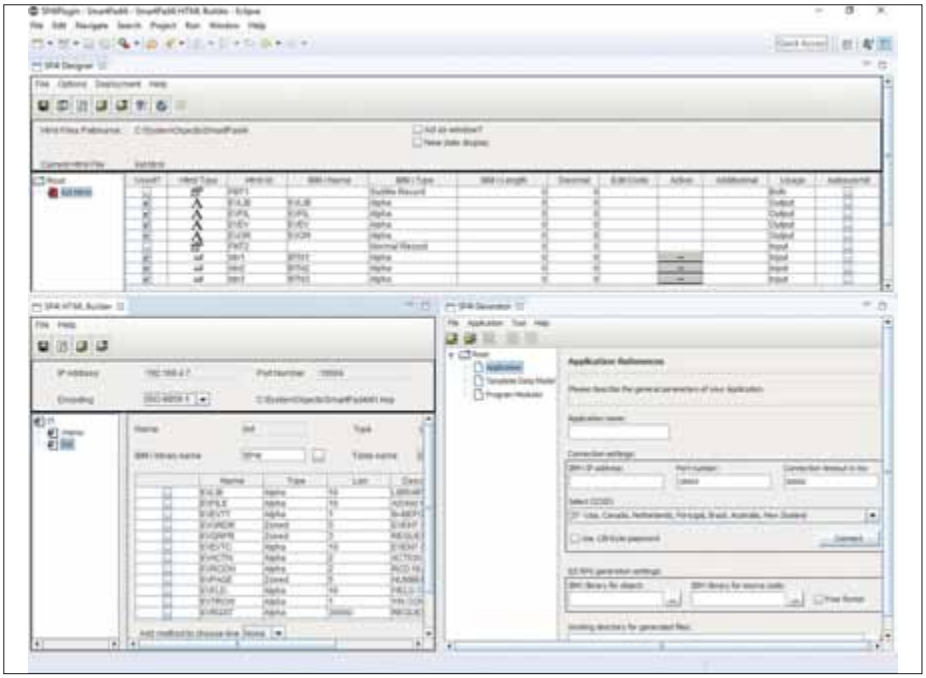
Si Eclipse est gratuit, tous les plugins ne le sont pas ! Outre l'achat des plug-ins, la mise en place d'un atelier Eclipse RPG va nécessiter des coûts d'adaptation de l'existant plus ou moins importants en fonction des choix fonctionnels décidés. Les coûts de formation ne sont pas non plus à négliger. Ceux-ci peuvent toutefois être minimisés si on ne remet pas en cause les processus existants ; dans le meilleur des cas (et pour une solution bien adaptée) une seule journée suffit !

Une recommandation : commencer petit en évitant dans un premier temps de remettre en cause toute l'infrastructure de gestion des sources (sauf si cette infrastructure est le problème).

Bien que les gains en transmission de la connaissance de l'entreprise, des applications et du savoir faire ne fassent pas souvent l'objet d'un chiffrage (on se demande d'ailleurs pourquoi ?), il est possible de considérer des gains de productivité « robotiques » sur les opérations de base du développement (navigation dans les sources, édition, auto-complétion, contrôle syntaxique local, etc…). Un benchmark a été réalisée par IBM et exhibe un gain de productivité de l'ordre de 30%.

Les gains de type « robotique » sont à relativiser car un développeur ne passe pas tout son temps à manipuler des sources. En revanche, la différence d'efficacité entre les éditeurs SEU et Eclipse est tout simplement indiscutable même pour des programmeurs RPG chevronnés.

Faut-il procéder à des calculs d'apothicaire ? Tout



dépend du prix de la solution évaluée. Si celle-ci est peu onéreuse, on peut faire l'économie d'une étude de ROI pour peu que l'on soit convaincu que la modernisation des outils de développement RPG est incontournable pour continuer à faire vivre son patrimoine applicatif IBM i.

J2EE / IBM i : opter pour l'agilité et la convergence

Mettre en place un atelier de développement RPG/400 sous Eclipse est une formidable opportunité pour faire évoluer les méthodes de travail et les mentalités : commencer à mettre en place de l'agilité côté AS/400. Afin de tendre vers des livraisons régulières, fiables et en phase avec les demandes du métier, il est nécessaire de ne pas sacrifier la qualité de la production :

- contrôle qualité du code : ce point est en général déjà adressé dans les organisations (CMMI oblige) mais de manière inégale.
- tests automatiques (unitaires et de non régression) : ce sujet est plus délicat et fait l'objet souvent de réserves alors qu'il est avant tout une question de volonté de la direction et de méthodologie.

L'enjeu, à terme, serait de pouvoir **unifier les usines RPG et Java,** grâce à des méthodes et des processus communs. Cette tendance apporterait une vraie plus-value en termes de productivité, de rendements et de coûts.

Ainsi, les développeurs formés au langage Java seraient davantage incités (motivés !) à monter en compétence RPG, si les outils sont les mêmes. A l'inverse, l'apprentissage du langage Java serait facilité pour un RPGiste. D'un point de vue managérial, **cette transversalité reste un facteur clé d'avenir** pour la continuité des patrimoines AS/400 et un **facteur clé de motivation pour les nouveaux développeurs !**

Fonctionnalités de Cobos/400 (SP4i)

- Applications hybrides

Les applications SP4i tournent aussi bien sur les appareils mobiles que dans les navigateurs Internet.

- Applications multiplateformes

L'application SP4iv2 disponible sur App Store et Google Play permet de lancer vos programmes et d'accéder à votre IBM i.

- Fonctions mobiles

Les applications SP4i peuvent accéder aux fonctions des terminaux mobiles comme le carnet d'adresses, la géo localisation et la caméra.

- Emulation 5250

Ce module permet de traduire les écrans de vos applications RPG déjà existantes en html, les rendant ainsi accessibles aux navigateurs et aux Smartphones.

Les Plugins Cobos/400

Les plugins SP4i disponibles sous Eclipse sont:

- HTMLBuilder

Ce plugin permet de générer des fichiers html de base qui seront utilisés par les deux autres plugins.

- Designer

À partir de votre fichier html créé ou généré par HTMLBuilder, Designer crée un programme source dans RPG. Vous devez ajouter votre logique métier dans ce programme et accéder à votre base de données si nécessaire.

- Generator

Ce plugin permet de créer automatiquement des applications standard. En utilisant un modèle de données pour décrire la ou les tables de base de données utilisées, ce module générera un module complet avec :

- Un fichier html créé à partir de vos modèles html,
- Un programme source IBM i dans ILE RPG ou ILE RPG Free pour afficher, insérer, mettre à jour et supprimer des enregistrements dans vos tables.
-

A PROPOS DE METRIXWARE

Depuis 1995, l'éditeur français et indépendant Metrixware propose des technologies innovantes et des solutions industrielles pour la modernisation des applications cœur de métier et ce qui sert à les produire, l'usine logicielle.

Metrixware édite notamment Cobos (https://cobos.metrixware.com), l'IDE COBOL de référence sous Eclipse pour les Mainframe et Unix. Cobos permet de gagner en productivité en modernisant le poste de travail des développeurs COBOL et PL1, sans impact sur les infrastructures centrales.

Sa filiale SystemObjects édite Cobos/400 (anciennement SmartPad4i), l'IDE RPG de référence sous Eclipse pour IBM i. Aussi avantageux sur le plan technique que financier, Cobos/400 permet de gagner en productivité, de systématiser le contrôle des "bonnes pratiques" RPG et de moderniser le poste de travail des développeurs, sans impact sur le coût des ressources IBM i.

Pour en savoir plus sur Cobos/400 : https://www.systemobjects.com/homesp4i.html

Apprendre à programmer

au lycée avec Texas Instruments

TI-Innovator™ Rover

Le robot programmable par la calculatrice graphique pour une multitude d'expériences dynamiques

TI-Innovator™ Hub

Le boitier de commande doté d'une carte professionnelle MSP 432 pour contrôler le Rover et réaliser des projets simples

Les calculatrices graphiques

Les calculatrices graphiques TI-83 Premium CE et TI-Nspire™ CX/CX CAS pour se familiariser avec la programmation

Livres

Les livres Eyrolles "Algorithmique et programmation" pour le lycée général et professionnel afin de développer des projets pour tous les niveaux

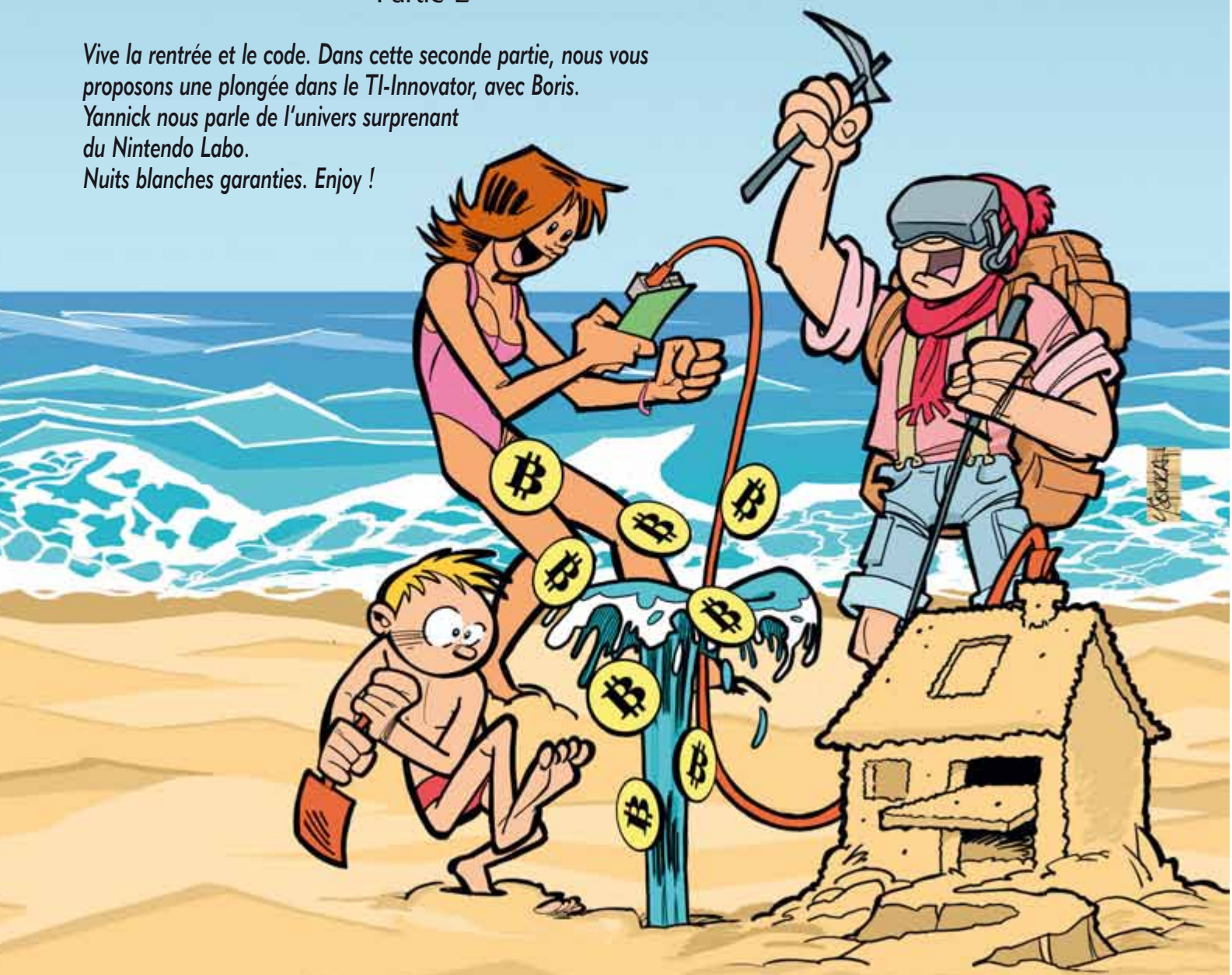
TEXAS INSTRUMENTS

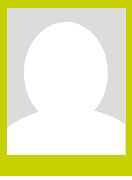
education.ti.com/france

Programmez en famille

Partie 2

Vive la rentrée et le code. Dans cette seconde partie, nous vous proposons une plongée dans le TI-Innovator, avec Boris. Yannick nous parle de l'univers surprenant du Nintendo Labo.
Nuits blanches garanties. Enjoy !





Boris Hanuš
Professeur agrégé de Mathématiques et d'ISN au lycée Condorcet de Limay

TI-Innovator et TI-Rover : les STEM arrivent en force au collège-lycée.

Depuis quelques temps, Texas Instruments connue en France pour ses calculatrices (TI83 Premium CE, TInspire, etc…) a mis sur le marché un petit hub appelé TI-Innovator qu'on branche sur sa calculatrice à l'aide d'un simple câble USB. 1 2

Le TI-Innovator dispose d'une led rouge, une led RVB, d'un capteur de niveau de lumière et d'un haut-parleur. De plus trois ports d'entrée (pour brancher par exemple un capteur de distance, de température, …), trois ports de sortie, un connecteur de platine d'essais avec 20 broches étiquetées et un port I²C. Les STEM (Science, Technology, Engineering, and Mathematics) sont de plus en plus à la mode à l'Education Nationale, et permettent aux élèves d'écrire des algorithmes qui interagissent avec des objets concrets. L'environnement de programmation de ce TI-Innovator hub est familier des élèves : Il utilise le TI-Basic (et bientôt Python…).
Par exemple pour programmer l'allumage de la led rouge on écrit : 3

```
Send("SET LIGHT ON")
```

Pour faire clignoter la LED 2 fois :

```
Send("SET LIGHT ON")
Wait 0.5
Send("SET LIGHT OFF")
Wait 0.2
Send("SET LIGHT ON")
Wait 0.5
Send("SET LIGHT OFF")
```

Ce qui est difficile pour les élèves du secondaire, c'est de conceptualiser la notion de boucles. Mais avec ce type de matériel, on l'introduit de façon naturelle. Ainsi lorsqu'on demande de faire cligner 15 fois de suite la LED, certains élèves commencent par recopier suffisamment de fois les lignes précédentes mais d'autres se manifestent : *« C'est relou Monsieur faut recopier tout ça 15 fois ? »* et puis très rapidement : *« Monsieur, c'est possible de lui dire de recommencer le programme précédent 15 fois ? »*… Voilà c'est fait ils vont bien voir l'utilité de la boucle FOR… Ce qui a donné naissance au programme suivant :

```
For(I,1,15)
Send("SET LIGHT ON")
Wait 0.5
Send("SET LIGHT OFF")
Wait 0.2
End
```

Les activités qui suivent s'enchainent facilement et on va du feu tricolore (afficher la LED RGB verte pendant 4 secondes, orange 1 seconde puis rouge 4 secondes) avec répétition à l'infini ce qui nous permet d'introduire une boucle TANT QUE…
Le TI-Innovator étant doté d'un haut parleur, on mêle mathématiques et musique et programmation avec les gammes Pythagoriciennes… En effet, elle utilisent le calcul de fractions pour obtenir les fréquences des notes de musique : pour passer du LA de référence (440Hz) au LA de l'octave supérieur on multiplie la fréquence par 2, et pour passer d'une note à la quinte suivante on multiplie par 3/2. Un premier travail permet d'obtenir le tableau suivant : 4

Pour émettre un son à 440 Hz pendant 5 secondes avec le TI-Innovator on entre l'instruction suivante : 5

```
Send("SET SOUND 440 TIME 5")
```

Pour jouer quelques notes il suffit d'écrire une ligne par note… Ou bien d'utiliser des listes et une boucle FOR…
Très naturellement, un groupe d'élèves faisant cette activité a demandé de l'aide pour connaitre les instructions afin de transformer son clavier de calculatrice en piano ! 6

1 *Fabriqué à partir d'une carte MSP-EXT432.*

2 3

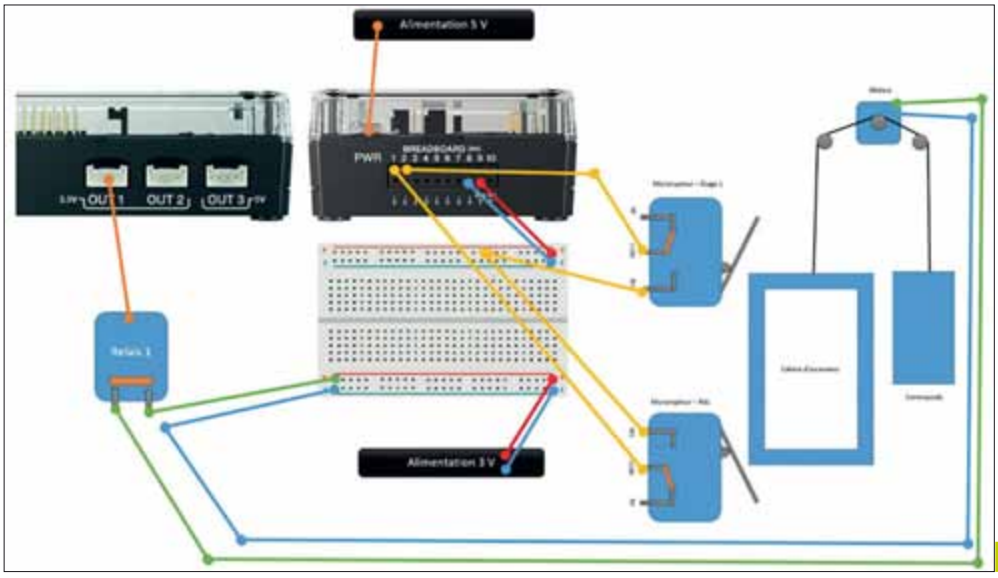
4

Note	do	ré	mi	fa	sol	la	si	Do
Fréquence en Hz	260,74	293,33	330	347,65	391,33	440	495	521,48

5


6

7



9

Mais on n'en a pas encore fini avec ce petit boitier !
Le TI-Innovator possède un capteur de luminosité : 7

Afin de lire et d'afficher l'état du capteur on écrit :

```
Send("READ BRIGHTNESS ")
Get(L)
Disp L
```

On peut donc simuler les éclairages publics en écrivant un programme qui affiche la LED en rouge lorsqu'il n'y a pas beaucoup de luminosité et l'éteint sinon.
On peut brancher aussi des capteurs sur les ports IN, par exemple un capteur de distance : 8


8

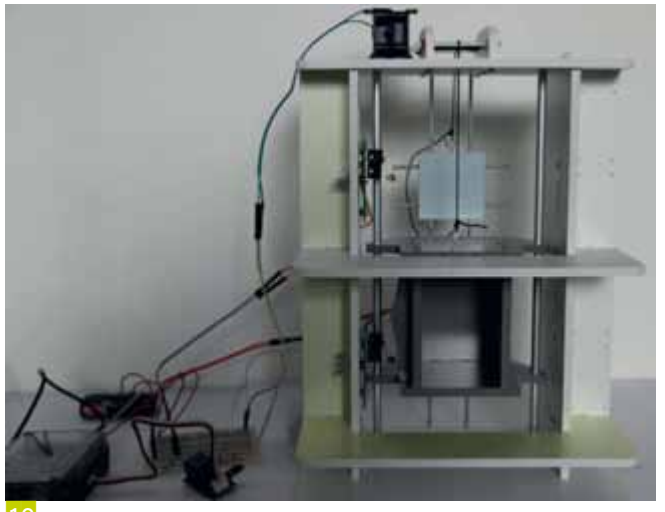
La gestion de ce capteur au boitier est très simple : Il faut tout d'abord déclarer ce capteur :

```
Send("CONNECT RANGER 1 TO IN 1")
```

Puis on lit son état et on affiche en écrivant :

```
Send("READ RANGER 1")
Get(D)
Disp D
```

Et on est prêt pour créer un programme qui simule le radar de recul d'un véhicule (avec usage des LED et du haut-parleur pour donner des informations à l'utilisateur).
Pour ceux qui veulent aller plus loin, on peut utiliser une platine et des maquettes (vendues par A4 technologies par exemple). Ici le montage d'une maquette de l'ascenseur : 9 10

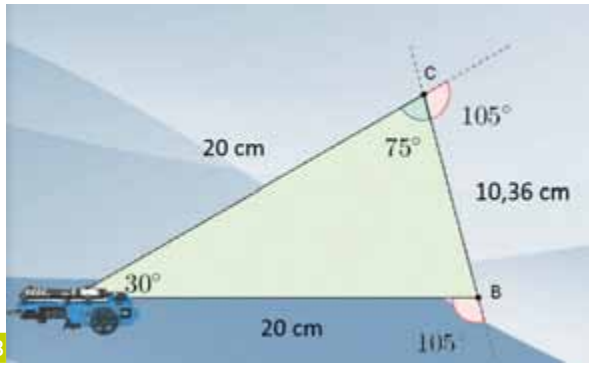

10

Passons maintenant au dernier outil sorti cette année par Texas Instruments : le TI-Rover : 11 12

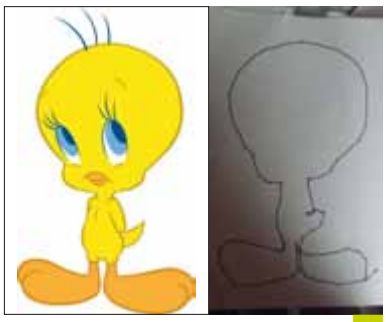
Tout d'abord on branche sa calculatrice TI-83 Premium CE (ou TInspire) au Hub. Puis on peut connecter le TI-Rover.
Il se pilote avec la TI83 Premium CE + le hub TI-Innovator (placé sous le TI-Rover) et le TI-Rover.
Le TI-Rover est composé de : deux roues motrices avec moteurs indépendants, un porte marqueur (pour dessiner au sol), un capteur de distance intégré à l'avant du Rover, un giroscope, un capteur couleur intégré sous le chassis, une led (RVB), le tout étant controlé par le Hub.
Les instructions pour faire fonctionner le TI-

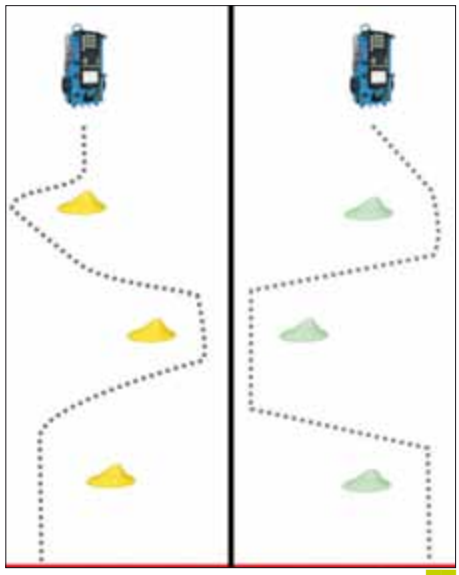

11


12



13


15


14


16

Rover sont très simples, et on peut utiliser plusieurs unités : le temps (de parcours), le mètre, le nombre de tour de roue… Il y a plein d'exercices de conversion à faire !

```
Send("CONNECT RV")
Send("RV FORWARD 5 M")
```

Est par exemple équivalent à :

```
Send("CONNECT RV")
Send("RV FORWARD 26.53 REVS")
```

REVS significant révolution.
Grâce à ce robot, on va pouvoir programmer sa trajectoire et lui adjoindre un marqueur pour dessiner sa trajectoire : 13

Calcul d'angles…

Le programme est simple et tient en quelques lignes :

```
Send("CONNECT RV")
Send("RV LEFT 30")
Send("RV FORWARD 0.2 M")
Send("RV RIGHT 105")
Send("RV FORWARD 0.1036 M")
Send("RV RIGHT 105")
Send("RV FORWARD 0.2 M")
```

Très vite les élèves ont organisé une course de voiture à partir de circuit identique : 14

L'optimisation de la trajectoire est une activité qui a eu beaucoup de succès.
On peut se servir aussi du TI-Rover comme d'un photocopieur en partant d'une photo qu'on insère dans la TI83 Premium CE, puis en utilisant l'algorithme de la main gauche pour détecter le contour de la forme, on fait alors parcourir au TI-Rover la forme détectée…
Cette activité un peu difficile car longue à construire donne par contre des résultats qui ont beaucoup enthousiasmé les élèves : 15

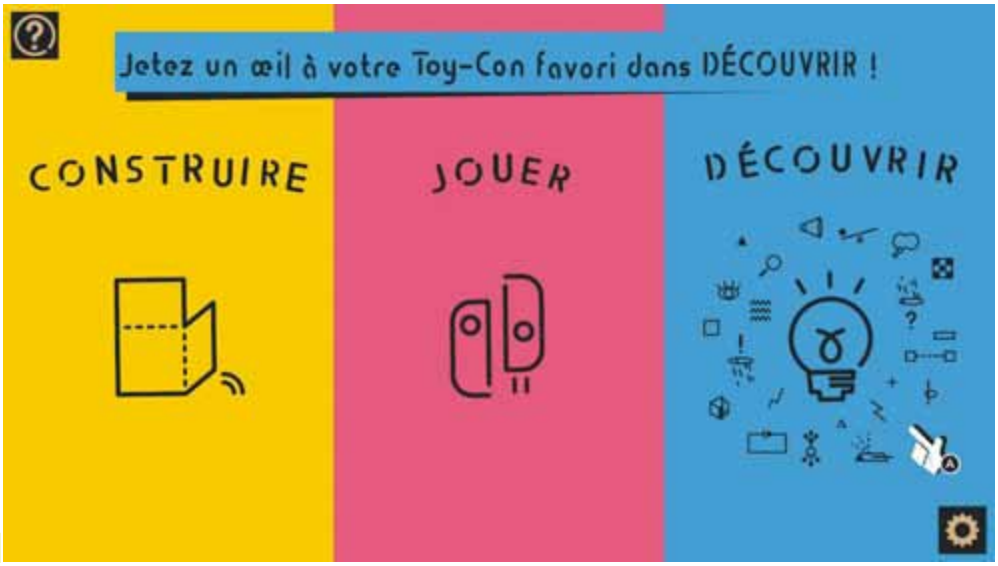
Avec une imprimante 3D, on peut ajouter des pinces pour faire des exercice de détection d'obstacle et de prise d'objet : 16

Conclusion

Par contre, malgré l'intérêt des élèves pour les STEM, on rencontre un certain nombre de difficultés.
Cette génération qui a l'habitude des correcteurs automatiques, des « gestions intelligentes » qui effectuent un certain nombre d'actions qui devancent le comportement de l'utilisateur et lui permet d'aller plus rapidement dans ses tâches, est un peu dépitée lorsqu'il s'agit de tout penser depuis le début, ne plus supposer d'aide « intelligente » de la machine. C'est un fossé qu'on n'avait pas il y a une décennie.
Il faut donc souvent commencer par des exercices simples pour espérer pouvoir aborder sereinement les activités plus complexes.
Texas Instruments a mis en ligne de nombreuses activités en particulier 10 minutes of Code pour permettre aux élèves de se familiariser avec le code en peu de temps : https://education.ti.com/fr/activities/ti-codes/innovator

Yannick Lejeune
Directeur Internet du Groupe IONIS (Epita, Epitech, Etna…) où il dirige un service de type agence web interne, Yannick est le créateur de l'Institut d'Innovation Informatique de l'Epita. Ancien Microsoft MVP et auteur d'ouvrages sur C#, les web services ou XML, il code encore de temps en temps, quand ses besoins de papa geek l'y amènent.

Premiers pas avec l'atelier Toy-Con de Nintendo Labo



1

2

Avec ses schémas de montages hautement précis, au sein desquels chaque morceau de carton a été finement étudié pour être bien placé et utile, le Nintendo Labo et ses Toy-Con, pourraient avoir l'air d'un univers dans lequel l'improvisation n'a pas sa place. Qu'il s'agisse du kit robot, du piano, de la maison, de la voiture télécommandée, etc., tout semble si bien pensé qu'il parait difficile de vouloir customiser la proposition faite par la firme de Mario. Et pourtant, la logique DIY de Nintendo a été pensée pour permettre à tout un chacun d'étendre le périmètre fonctionnel de cet ensemble de jouets ou, même, d'en créer de nouveaux : dans l'interface des Toy-Con se cache une « plaque d'égout » qui va nous permettre, telles des tortues ninja, d'aller voir le dessous des choses.

3 menus pour trois approches des Toy-Con

La cartouche Toy-Con propose 3 grandes entrées :

- le menu Construire qui permet de monter les Toy-Con avec une logique et une simplicité qui feraient rougir Ikea ;
- le menu Jouer qui permet de s'amuser avec ces mêmes Toy-Con ;
- le menu Découvrir qui en vulgarise le fonctionnement, propose quelques paramétrages et recèle l'entrée secrète de l'atelier. 1

Pour faire apparaître l'interface « secrète », vous devez avoir construit l'un des Toy-Con livrés dans un des deux kits… Une fois ceci fait, cliquez sur le Toy-Con en question, regardez la vidéo vous en expliquant le fonctionnement et vous devriez voir apparaître le fameux laboratoire secret. 2

Une ambition… à restreindre

Disons-le dès le départ, si les Toy-Con peuvent être un véritable plaisir pour adultes amateurs de jeux de construction intelligents dans leur phase de montage, leurs applications « ludiques » sont encore très orientées vers les plus jeunes. Après un certain âge, il est difficile d'imaginer passer des heures devant le jeu de destruction urbaine du robot géant, tout comme sur le piano, le guidon de moto ou la canne à pêche qui trouvent vite leurs limites.

Il en va de même avec cette première version de l'Atelier Toy-Con qui reste relativement limitée dans ce qu'elle permet de produire. La philosophie de développement très « root » amène vite à quelques limitations qui vont en faire un plaisir de hacker mais qui vont restreindre le plaisir ludique à la sortie. L'atelier est un outil de programmation visuel qui sait faire peu de choses. Pour faire une comparaison, si Scratch était le Turbo Pascal, alors l'Atelier Toy-Con serait une sorte d'assembleur avec une couche d'abstraction matérielle. Pour en faire sortir une application vraiment satisfaisante, c'est le hacker qui sommeille en vous qu'il va falloir mobiliser.

Programmation node-based et visuelle à l'ancienne

L'interface proposée par l'Atelier est un espace noir quadrillé de traits blancs servant de repères dans lequel on va pouvoir venir déposer différents composants visuels qu'on pourra relier entre eux. Il n'est pas nécessaire d'avoir une grande expérience du code pour pouvoir jouer avec l'interface. 3

Si vous cliquez sur le menu burger en haut à gauche, un petit tutoriel « Lire les thèmes » se propose de vous en expliquer les bases et après, c'est à vous de jouer.

Les composants se trouvent répartis en 3 grands ensembles :

- Les entrées ;
- Les « milieux » ;
- Les sorties.

Chaque composant, ou boîte, est représenté par un carré présentant quelques annotations et deux connecteurs : un bleu et un rouge. Situés de part et d'autre, ceux-ci servent de points de connexion aux entrées/sorties : il suffit de dessiner une ligne entre les deux pour les relier.

Par exemple, si vous prenez une entrée « toucher » et que vous la connectez à une entrée « allumer », à chaque fois que vous allez toucher la zone d'entrée, la zone de sortie va s'allumer. 4

Les entrées : les déclencheurs d'action 5

Dans le menu, les entrées s'affichent de la plus complexe à la plus simple.

- Toy-Con: c'est l'entrée de plus haut niveau. Il s'agit de pouvoir interagir avec les Toy-Con existants et de pouvoir appeler leurs commandes déjà implémentées. On peut donc utiliser le poing du robot ou le moulinet de la canne à pêche comme interface d'entrée. On peut faire appel à deux Toy-Con en même temps s'ils font partie du même kit.
- Si marqueur IR détecté : c'est sans doute l'une des fonctions les plus intéressantes et qui donnera le plus de possibilités au fur et à mesure de l'appropriation du kit par les développeurs. Le capteur IR, une caméra infrarouge, est situé en bas de la Joy-Con droite. Quand la console détecte un marqueur (souvent un morceau d'adhésif réfléchissant) ou un mouvement infrarouge, elle envoie un ordre de sortie.
- Console : des fonctions comme « si agité » ou « si face avant » (cf ci-dessous) mais adaptées à la console plutôt qu'aux manettes.
- Si face avant Joy-Con vers le haut : des fonctions pour obtenir la position des Joy-Con (6 positions différentes) avec réglage de niveau pour avoir des paliers plus ou moins flous.
- Si stick poussé : se déclenche en fonction de la position du stick directionnel d'un des Joy-Con.
- Si bouton appuyé : pour récupérer le ou les boutons appuyés et déclencher une sortie en fonction.
- Si agité : se déclenche lorsqu'on secoue le Joy-Con choisi.
- Si touché : se déclenche lorsqu'on presse le bouton visuel dessiné sur la console.

Les milieux : la logique de l'application 6

- Commentaire : un module qui sert juste à commenter votre écran de l'Atelier en y plaçant visuellement des blocs de texte. Le nombre de blocs étant semble-t-il limité à 5, il devient rapidement insuffisant quand les schémas deviennent très complexes.
- Pipette : la traduction est ici peu heureuse, la version américaine parle de bullseye, et on est effectivement plus dans le registre de la cible. Ce module permet d'activer la caméra IR de la joy con droite et de repérer un marqueur infrarouge de sorte qu'on déclenche une sortie lorsque celui-ci est détecté dans la bonne position. Point important, la fonction IR est basée sur un repère fixe lié à la manette, elle ne différenciera donc pas les capteurs visuellement mais plutôt par leur position par rapport à elle-même.
- Compteur : ce module modifie un compteur à chaque fois que l'entrée qui lui est connectée est exécutée. Il est connecté à 3 entrées, une qui fait monter, une qui fait descendre et une qui le remet à zéro. Attention, il s'agit bien d'un module incrémentable de type variable et pas d'un affichage.
- Minuteur : retarde de quelques secondes l'exécution de la sortie provoquée par l'entrée.
- NON : porte logique classique, tout ce qui n'est pas l'entrée va déclencher la sortie.
- ET : porte logique classique, lorsque ses deux entrées sont activées, envoie un signal à la sortie

Les sorties 7

- Emettre infrarouges : cette fonction allume la caméra infrarouge du Joy-Con désigné. Vous pouvez en avoir jusqu'à deux allumées en même temps, ce qui permet de faire du repérage.
- Allumer l'écran : allume la boîte de sortie sur l'écran. A noter qu'il n'est possible de l'allumer qu'en blanc.
- Produire son : joue une note de musique ou un son.
- Vibrer : fait vibrer le Toy-Con désigné.

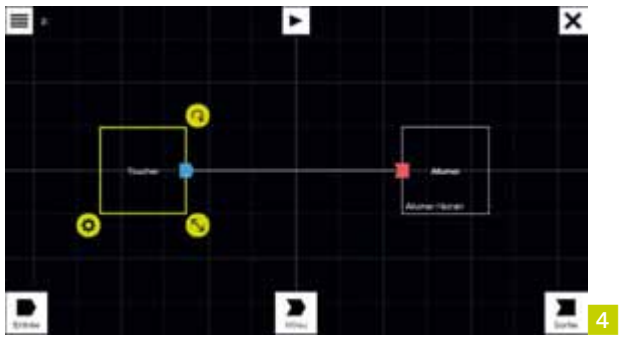
Démarrons notre premier Toy-Con.

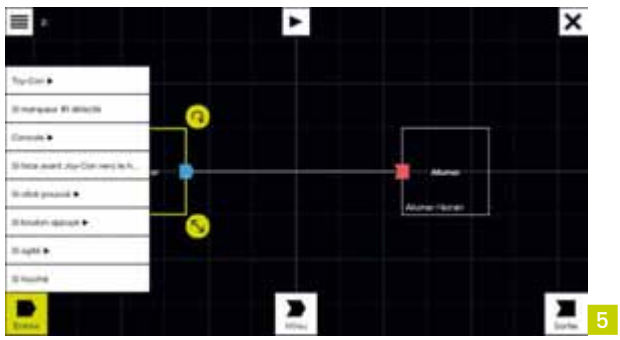
Je vous propose de réaliser un chi-fou-mi ou pierre-feuille-ciseau à jouer avec les deux manettes.

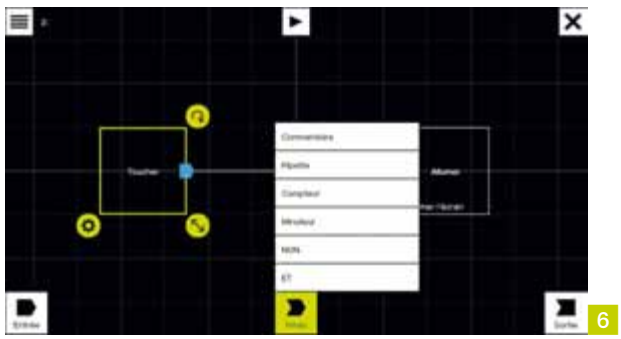
L'écran de jeu

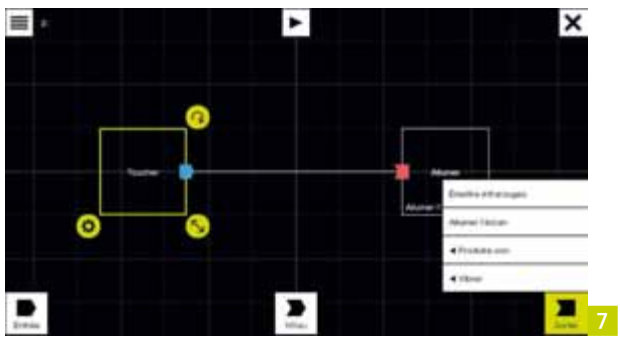
Les possibilités d'affichage de l'Atelier sont assez limitées. Le seul élément dont on dispose, c'est le module qui éclaire l'écran dans la surface carrée délimitée. En gros,

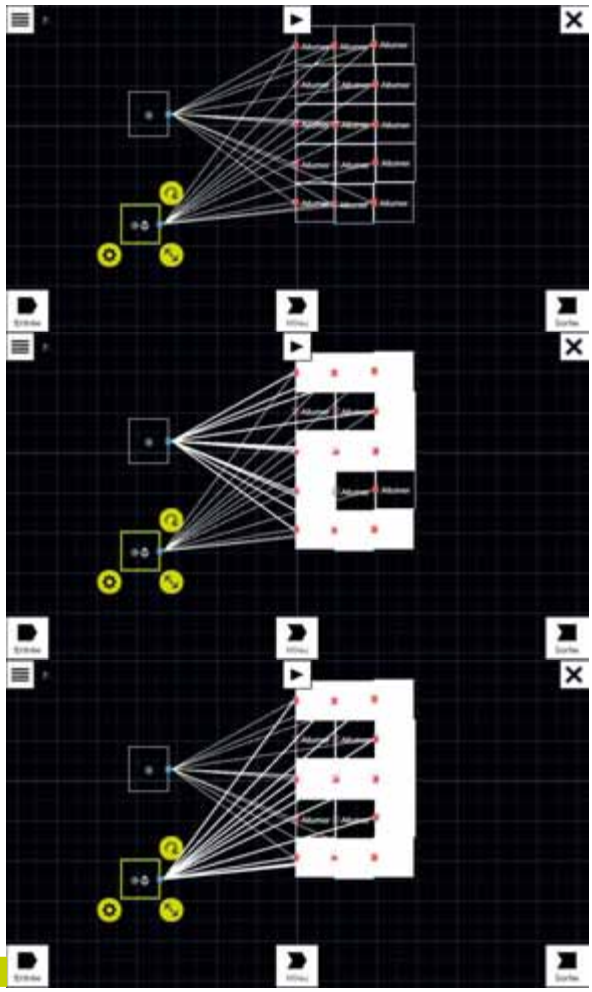
3


4


5


6


7


8

9

10

chaque module est une sorte de pixel virtuel blanc qui a deux états : allumé ou éteint. On peut s'en servir d'indicateur visuel comme nous allons le faire ci-dessous. Ou bien pour afficher des informations plus complexes, mais cela implique de concevoir un affichage digital à l'ancienne en utilisant chacun de ces blocs comme un pixel qu'on allumera ou pas en fonction des caractères à afficher. 8

L'autre solution, et c'est celle recommandée par Nintendo, consiste à utiliser le couvre-écran. 9

Pour le construire, rendez-vous sur la planche A de votre kit. 10

Une fois celui-ci monté, nous allons nous construire notre propre interface en carton à insérer dedans.

A gauche, ce que le joueur J1 a joué (pierre – feuille – ou ciseau)

A droite, ce que le joueur J2 a joué (pierre – feuille – ciseau)

Au centre le résultat de la partie (Joueur 1 vainqueur, Joueur 2 vainqueur, ou match nul)

Créez une interface de ce type dans un carton au format du centre du couvre-écran : ouvrez une fenêtre vide dans chaque zone qui doit s'allumer, nous allons positionner un bloc « allumer écran » sous chacune d'entre elles.

Mettez des légendes, vous devriez obtenir quelque chose comme cela (les zones en pointillé sont dans le carton) : 11

Si vous le faites avec des enfants, vous pouvez les laissez décorer l'interface du jeu à leur guise. Vous pouvez également pousser l'esthétique en ouvrant des fenêtres en forme de pierre, feuille ciseau au lieu des simples carrés des côtés, à vous de voir.

A la question « Pourrait-on avoir des indicateurs visuels de couleur, du genre une boîte qui s'allume en rouge, une boîte qui s'allume en vert ? », la réponse de Nintendo est simple : oui en insérant des papiers Cellophanes type papier de bonbon transparent sous le carton.

Quand je vous disais que c'était « roots »…

La logique de jeu.

La logique de pierre feuille ciseau est assez simple :

- la pierre casse le ciseau ,
- le ciseau coupe la feuille,
- la feuille enveloppe la pierre,

Quand on tire le même objet, c'est un match nul

J1 / J2	Pierre	Feuille	Ciseau
Pierre	NUL	J2	J1
Feuille	J1	NUL	J2
Ciseau	J2	J1	NUL

Tableau des positions de jeu.

Chaque joueur se voit équipé d'un Joy-Con. Pour l'exercice, on va choisir des modes de jeu différents pour chaque joueur.

Pour le Joy-Con 1 (celui de gauche) :

- Bouton flèche gauche : pierre
- Bouton flèche haute : feuille
- Bouton flèche droite : ciseau

Pour le Joy-Con 2 (celui de droite) :

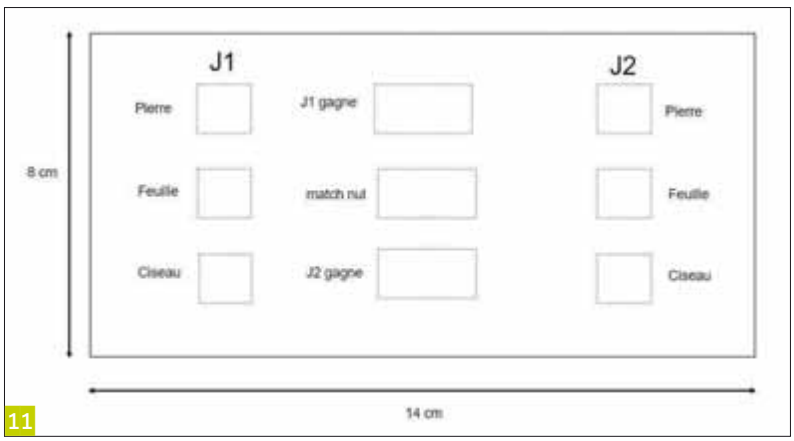
- Position sur la tranche : position neutre
- Position face en l'air : pierre
- Position redressée : feuille
- Position dos en l'air : ciseau

Commençons donc par créer des entrées pour chacune de ces positions.

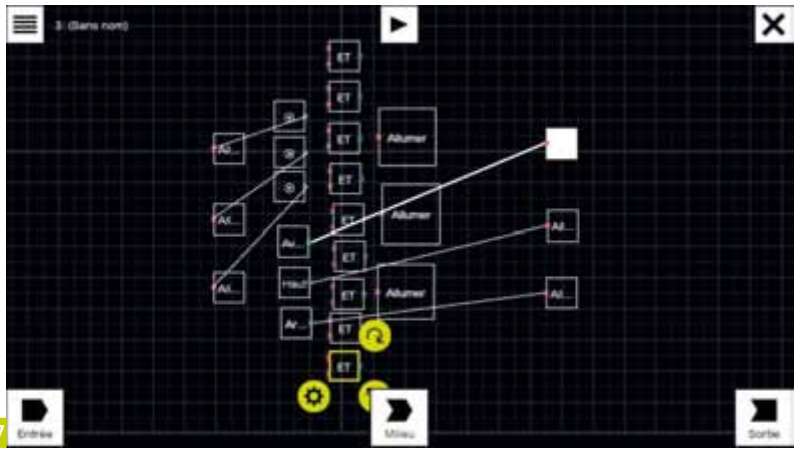
Commencez par dropper un module entrée « Si bouton appuyé » > Joy-Con (L)

Dans l'interface, décochez tous les boutons qui ne nous intéressent pas de sorte que seul le bouton gauche soit vert.

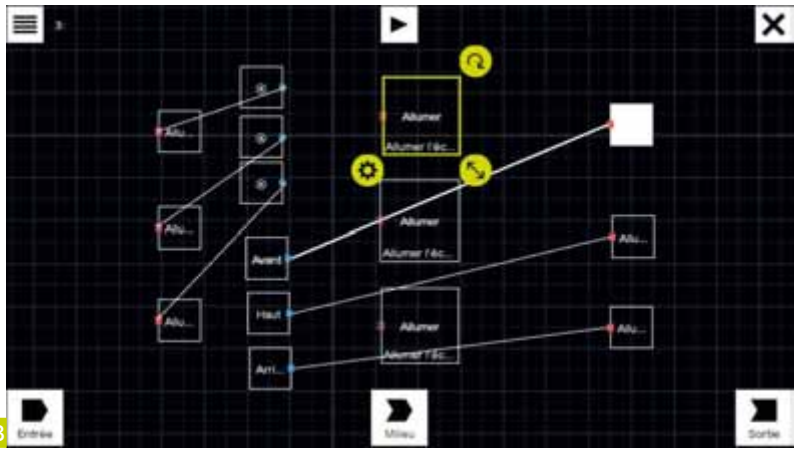
Vous noterez qu'à gauche se situe un numéro de manette. 1 manette est composée



11



17



18

de deux Joy-Con, on compte donc Joy-Con (L) et Joy-Con (R) comme faisant partie de la même. 12

Faites de même pour les 2 autres. 13

Passons maintenant au Joy-Con droit.

Déposez un module « Si face avant Joy-Con vers le haut » > Joy-Con (R) sur l'interface.

Créez les trois positions Avant (face vers le haut), Haut (Joy-Con redressé) et Arrière (dos vers le haut). 14 15 16

On va utiliser la position naturelle sur la tranche comme position neutre.

Analogue ou digital ?

En ouvrant les boîtes de configuration des modules, vous pourrez observer que chacun d'entre eux est paramétrable en fonction de vos besoins.

Il faut toutefois noter qu'elles sont nombreuses à disposer d'un switch analogue / digital. Le paramètre analogue permet de prendre en compte « l'intensité » d'une action. Si vous sélectionnez l'entrée « Si agité », le retour sera donc plus fort avec un mouvement brusque qu'avec un mouvement léger. Dans notre cas, choisissez donc Digital. Pour éviter d'avoir trop d'interférences entre les différentes positions de la main, réduisez la zone de détection à un range de 0.90 à 1.00.

Mise en place de la logique de jeu

Il s'agit de recréer les portes logiques du Tableau des positions de jeu.

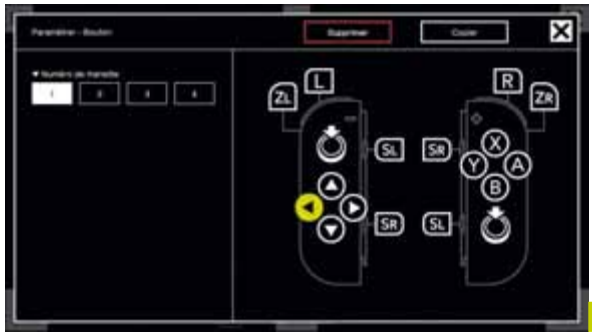
On va donc relier chaque boîte d'entrée à une zone de l'écran qui s'allumera à l'activation pour signaler le choix du joueur.

On crée donc 6 sorties « allumer l'écran », 3 pour chaque joueur, à relier à chaque position de jeu. 17

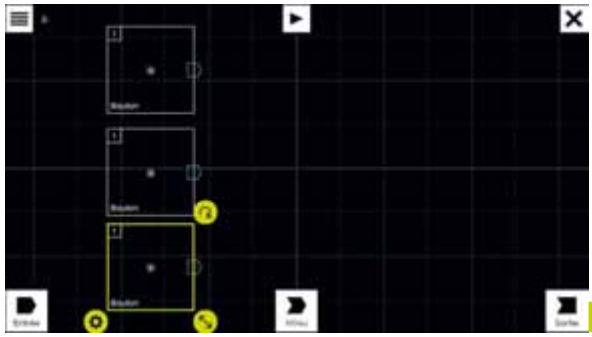
Vous pouvez déjà tester vos interfaces d'entrée en vérifiant que chaque pierre, feuille ou ciseau allume bien la bonne partie de l'écran. Passons maintenant aux 3 résultats de jeu. C'est un simple exercice de combinatoire. Commencez par déposer 3 zones à allumer correspondant à :

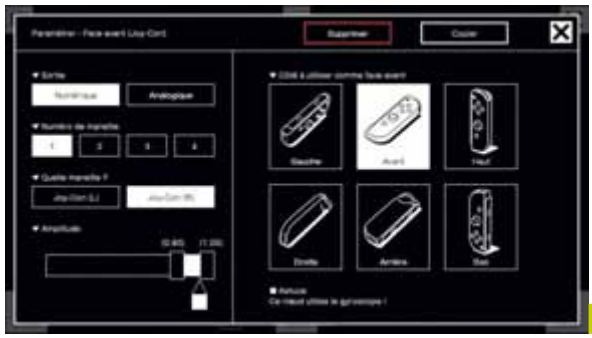
- Joueur 1 Gagne
- Match Nul
- Joueur 2 Gagne 18

Posons maintenant des portes logiques ET correspondant aux différentes combinaisons. 3 entrées chez le J1, 3 entrées chez le

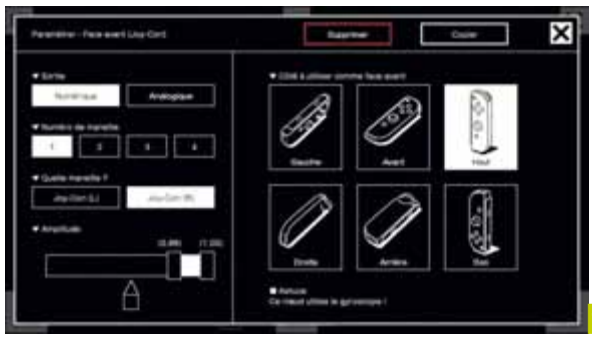


12

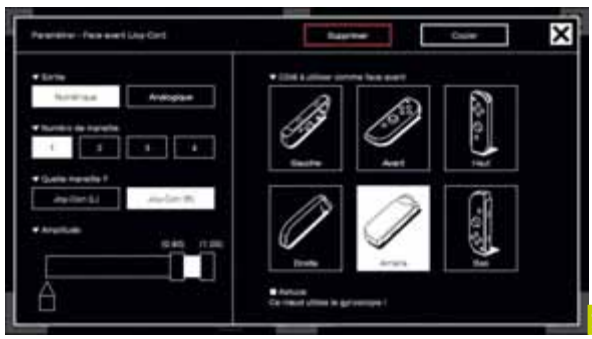

13



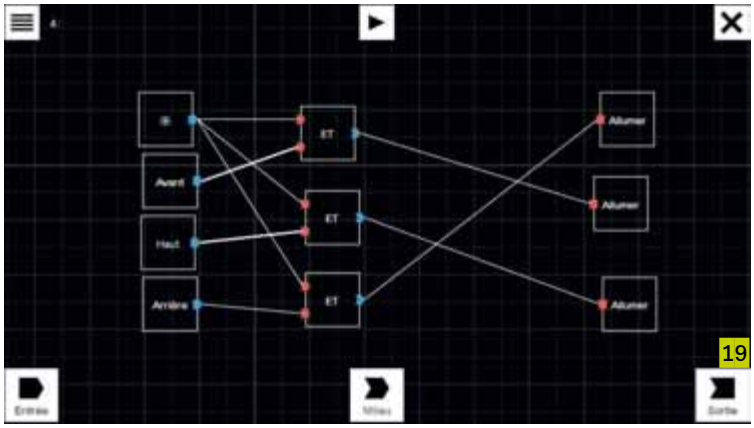
14



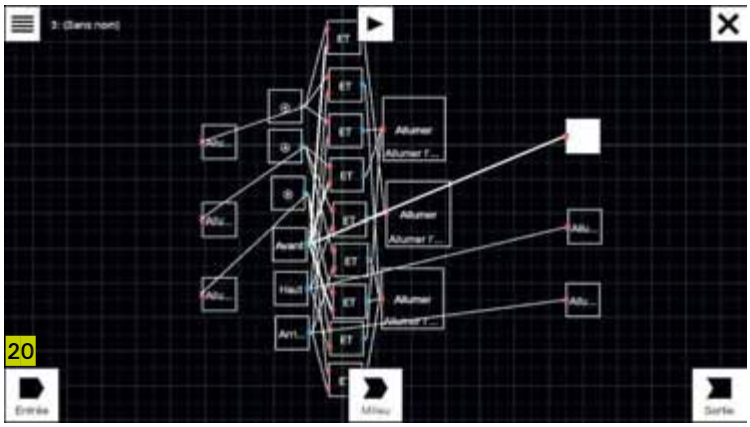
15



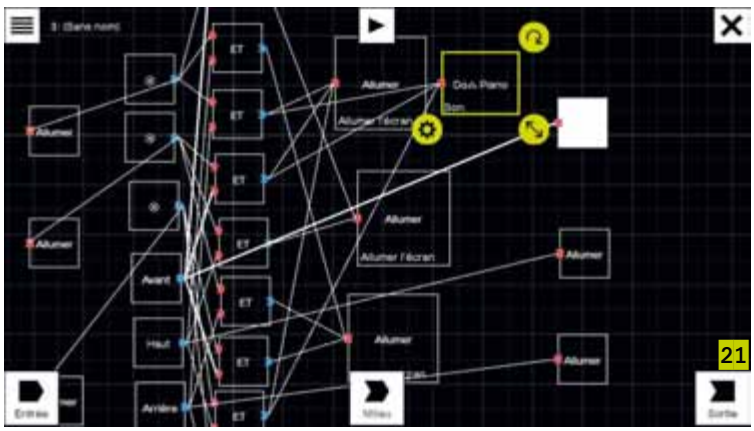
16



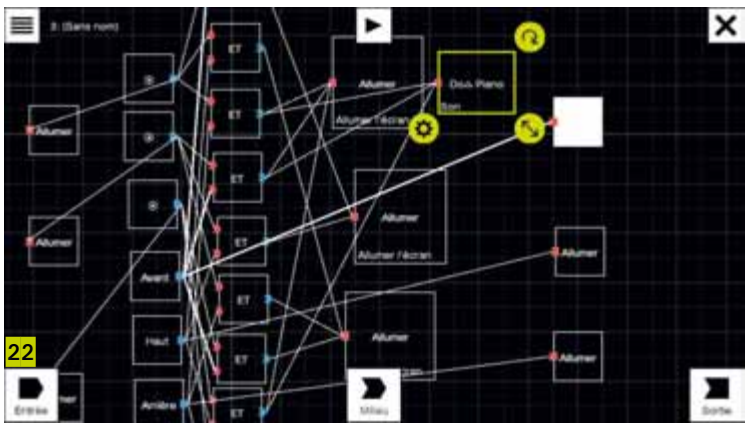
19



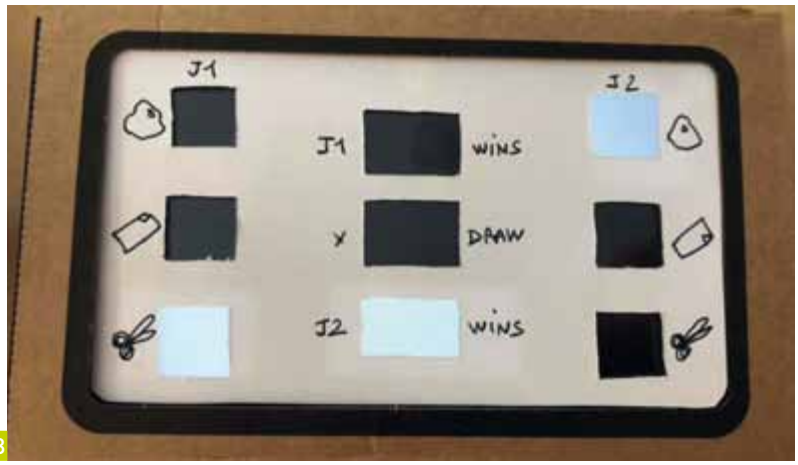
20



21



22



23

J2, ça fait 9 combinaisons possibles.

Pour chacune des entrées, il vous faut maintenant faire le maillage.

Prenons l'exemple pour la pierre du joueur 1 dans un écran simplifié :

- Pierre J1 (bouton gauche appuyé) ET Pierre J2 (Avant) -> Allume NUL
- Pierre J1 (bouton gauche appuyé) ET Feuille J2 (Haut) -> Allume J2
- Pierre J1 (bouton gauche appuyé) ET Ciseau J2 (Arrière) -> Allume J1 19

Comme vous pouvez le voir, il s'agit juste de relier les points pour reproduire les différents cas du tableau. Il ne reste plus qu'à faire la même chose dans l'écran de développement de notre application et à faire de même pour les deux autres entrées. Cela va donner un écran un peu plus complexe 20

Pour compléter, ajoutons à chaque statut final un son spécifique en dérivant les sorties des boites ET en parallèle des boîtes « allumer l'écran » des statuts finaux. 21 Faites ça pour les 3 et voilà le jeu prêt à être utilisé ! 22

Il ne vous reste qu'à ajuster les zones d'allumage de l'écran sous votre couvre-écran et c'est parti ! Appuyez sur Play en haut de l'écran ! A vous de jouer ! 23

Pour aller plus loin

On peut faire vibrer la manette du joueur qui gagne, c'est plutôt facile.

Si vous êtes motivé, vous pouvez faire un compteur de score qui s'incrémente en ajoutant le bloc adéquat aux sorties J1 gagnant et J2 gagnant. Et créer un affichage digital des résultats en créant une zone d'affichage avec des pixels virtuels à la manière des affichages numériques à l'ancienne de nos vieux réveils 8. Pour l'instant, les implémentations issues de l'Atelier sont assez limitées, vous trouverez sur le site Nintendo, une manière de créer une guitare virtuelle basée sur des zones à cliquer et des élastiques. Aux US, Ariana Grande a joué un morceau chez Jimmy Fallon uniquement accompagnée par des instruments basés sur des Toy-Con… Mais le plus intéressant reste sans doute encore à inventer en attendant une mise à jour évolutive de l'Atelier qui saurait lui ajouter quelques fonctions de plus haut niveau. •

Philippe Charrière
@k33g_org
Technical Account Manager pour GitLab
Associé chez Clever Cloud
Fondateur de Bots.Garden (éleveur de bots)

Partie 1.1

Prise en main de GitLab en douceur

Code, test, and deploy together

niveau 100

Dans cette première partie nous allons voir comment installer **GitLab** sur son propre poste et l'utiliser pour gérer un projet **git** de développement. Nous aborderons donc le principe de **merge request**, pour terminer avec les **GitLab Runners** afin de mettre en place une intégration continue sur notre projet, pour lancer des tests en automatique.

Tout d'abord, qu'est-ce que **GitLab** ? Cette question est importante, car beaucoup de monde pense que **GitLab** est une application **open source** pour "héberger et gérer des projets **git** de code source" comme le feraient **GitHub**, **BitBucket** et d'autres beaucoup plus anciens. Ceci est partiellement vrai. **GitLab** est une **application** qui vous permet de gérer le cycle complet de vos projets DEVOPS, c'est-à-dire que **GitLab** va vous permettre à partir d'une seule application de :

- Gérer vos projets de codes sources (au sens **git** du terme), comme le font **GitHub**, **BitBucket** (avec chacun leurs spécificités) avec des fonctionnalités de :
  - gestion des **issues**,
  - code **review**, **merge request** (l'équivalent des pull requests chez GitHub),
  - GitLab Pages (l'équivalent des GitHub Pages),
  - Wiki,
  - etc.
- Mais **GitLab** va aussi vous fournir les outils pour faire (selon les éditions de GitLab) :
  - votre intégration continue (**CI**),
  - vos déploiements (en continu aussi si vous le souhaitez) (**CD**),
  - de la qualimétrie de code,
  - de la vérification de sécurité,
  - de l'intégration avec K8S,
  - la mise en place d'un Docker registry,
  - de la haute disponibilité horizontale, hybride ou complètement distribuée,
  - etc.

Donc pas mal de choses.

Vous pouvez utiliser directement GitLab sur GitLab.com en créant un compte, ou installer une instance de GitLab sur votre ordinateur, serveur, VM sur le cloud …

Si vous prenez la solution SaaS **GitLab.com**, il vous sera proposé 4 plans (Free, Bronze, Silver, Gold). Le plan **Free** vous propose tout le nécessaire pour gérer vos projets : projets privés sans limites de nombre pour vous et votre équipe, CI/CD, GitLab Pages, Review Apps, …

Si vous choisissez la solution "**self-hosted**", elle existe aussi en 4 versions (Core, Starter, Premium et Ultimate), la version **Core** et la version communauté que vous pouvez installer librement où vous le souhaitez et sans aucune limitation.

Pour cet article j'ai choisi cette dernière option (et ce pour une raison très personnelle, je suis souvent dans le train, et pas toujours avec le WiFi, et j'ai souvent besoin d'utiliser GitLab, donc CQFD).

Commençons donc par l'installation de notre instance **GitLab**.

Installation(s)

Il y a de multiples façons d'installer **GitLab**, vous les trouverez ici https://about.gitlab.com/installation/. Je vais vous en proposer deux, une avec **Vagrant** et une avec **Docker**. Celle avec **Docker** a l'avantage de demander moins de ressources machine (normalement…), mais nécessite de bonnes connaissances en Docker lorsqu'il faut commencer à jouer avec les configurations réseaux par exemple. Celle avec **Vagrant** a l'avantage d'être simple et reproductible à la main dans une VM ou sur une machine "bare metal" et la mise en oeuvre de réseaux de VMs est plus simple à appréhender. Pour ceux qui ne connaissent pas **Vagrant**, c'est un outil qui permet d'automatiser par scripts la création de VM.

Je travaille sous macOS, mais en utilisant des outils standards, donc l'adaptation à vos OS est simple. Pour mes machines virtuelles, j'utilise **Virtual Box**, pour l'OS de mes VMs j'utilise **Ubuntu**.

Les prérequis seront donc les suivants :

- installer **Vagrant** https://www.vagrantup.com/, **Vagrant** est open source,
- installer **Virtual Box** https://www.virtualbox.org/ (open source aussi),
- et si vous voulez vraiment utiliser **Docker**, vous trouverez la version communauté par ici: https://www.docker.com/community-edition

Installation avec Vagrant

C'est extrêmement simple. Créez un dossier (appelons le 01-gitlab) dans lequel vous allez créer un fichier Vagrantfile avec le contenu suivant :

```
BOX_IMAGE = "bento/ubuntu-16.04"
GITLAB_NAME = "gitlab"
GITLAB_DOMAIN = "gitlab-ce.test"
GITLAB_IP = "172.16.245.121"

Vagrant.configure("2") do |config|
 config.vm.box = BOX_IMAGE

 ENV['LC_ALL']="en_US.UTF-8"

 config.vm.define "#{GITLAB_NAME}" do |node|

  node.vm.hostname = "#{GITLAB_NAME}"
  node.vm.network "public_network", bridge: "en0: WiFi (AirPort)"
  node.vm.network "private_network", ip: "#{GITLAB_IP}"

  node.vm.provider "virtualbox" do |vb|
   vb.memory = 4096
   vb.cpus = 2
   vb.name = "#{GITLAB_NAME}"
  end
```

```
  node.vm.provision :shell, inline: <<-SHELL
    # download and install GitLab packages
    wget —content-disposition https://packages.gitlab.com/gitlab/gitlab-ce/packages/ubuntu/xenial/gitlab-ce_11.0.3-ce.0_amd64.deb/download.deb
    EXTERNAL_URL="http://#{GITLAB_DOMAIN}" dpkg -i gitlab-ce_11.0.3-ce.0_amd64.deb
  SHELL

 end
end
```

Remarque: pour quelque chose de plus automatique et pour avoir la dernière version à chaque fois, vous pouvez utiliser ces 2 commandes à la place :

```
node.vm.provision :shell, inline: <<-SHELL
 # download and install GitLab packages
 curl -s https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh | sudo bash
 EXTERNAL_URL="http://#{GITLAB_DOMAIN}" apt-get install -y gitlab-ce
SHELL
```

Donc pour résumer, le script me permet de :

- sélectionner mon OS,
- fixer une adresse IP,
- fixer la taille de ma VM,
- et enfin d'installer **GitLab** en téléchargeant le package qui m'intéresse.

Si vous utilisez un autre OS ou souhaitez une autre version de GitLab vous pouvez faire un tour ici : https://packages.gitlab.com/gitlab/gitlab-ce.

Et bien sûr vous pouvez adapter le script à vos besoins.

Maintenant pour créer la VM, il suffit de taper les commandes suivantes :

```
cd 01-gitlab
varan up
```

Et d'attendre un petit peu (tout dépendra de la bande passante de votre fournisseur d'accès).

Quelques minutes plus tard…

Vous devriez obtenir ceci :

Thank you for installing GitLab!

GitLab should be available at http://gitlab-ce.test

Une fois votre VM opérationnelle, allez éditer votre fichier /,etc./hosts et ajouter cette ligne :

```
172.16.245.121 gitlab-ce.test
```

Maintenant vous pouvez accéder à votre instance **GitLab** avec votre navigateur avec l'url suivante : http://gitlab-ce.test

Quelques commandes Vagrant utiles

- stopper votre machine (toujours dans le répertoire 01-gitlab): vagrant halt
- re-démarrer votre machine: varan up
- accéder à votre machine en ssh: varan ssh
- détruire votre machine: varan halt; varan destroy -f

Mais avant de continuer, je vous explique comment faire la même chose avec **Docker** (sinon vous pouvez passer directement au paragraphe **Premier contact**).

Installation avec Docker

Bien sûr vous avez installé **Docker** sur votre machine. Dans un terminal lancez la commande suivante :

```
sudo docker run —detach \
  —hostname gl-ce-docker.test \
  —publish 8080:80 \
  —name gitlab-ce \
  —restart always \
  —volume /Users/k33g/gitlab_data/ce/config:/etc/gitlab \
  —volume /Users/k33g/gitlab_data/ce/log:/var/log/gitlab \
  —volume /Users/k33g/gitlab_data/ce/data:/var/opt/gitlab \
  gitlab/gitlab-ce:latest
```

Donc je vais créer un container avec le nom gitlab-ce et je pourrai accéder à mon instance sur le port 8080 et les données vont être stockées.

Pour plus de détails, vous pouvez consulter cette page https://docs.gitlab.com/omnibus/docker/.

Vous devez attendre un peu pour que votre container soit opérationnel et enfin vous pourrez atteindre votre instance **GitLab** avec cette url http://localhost:8080.

Cependant vous aimeriez pouvoir utiliser un nom de domaine pour accéder à votre instance **GitLab**, dans ce cas-là, la manipulation est un petit peu différente que celle utilisée avec **Vagrant**, car vous ne pouvez pas accéder directement à l'IP du container **Docker**.

Pour cela nous allons utiliser **Nginx** comme un reverse-proxy.

Dans un premier temps vous allez l'installer (dans mon cas sous macOS c'est simple : brew install nginx) puis éditez le fichier de configuration de **Nginx** /usr/local/etc/nginx/nginx.conf de cette manière :

```
worker_processes  1;

events {
 worker_connections 1024;
}

HTTP {
 server {
  listen 80;
  server_name gl-ce-docker.test;

  location / {
   proxy_pass http://localhost:8080;
```

```
 }
}
}
```

Ensuite il faut que gl-ce-docker.test resolve en 127.0.0.1, donc dans /,etc./hosts modifiez la ligne qui commence par: 127.0.0.1 en y ajoutant à la fin gl-ce-docker.test, puis démarrez (ou re-démarrez) le service **Nginx** avec la commande sudo brew services start nginx ou sudo brew services restart nginx et maintenant vous pouvez accéder à "votre **GitLab**" à partir de http://gl-ce-docker.test.
Mais revenons à notre version avec **Vagrant**…

Premier contact : création d'un utilisateur

Vous avez donc ouvert http://gitlab-ce.test et vous devez donc obtenir cette page : 1

Cette page vous permet de modifier le mot de passe de l'utilisateur root (qui est l'utilisateur administrateur). Donc, renseignez le nouveau mot de passe et cliquez sur **Change your password**.
Vous arrivez donc sur une 2e page vous proposant de vous connecter : 2

Mais cliquez plutôt sur l'onglet **Register** pour créer un 1er utilisateur (vous) : 3

Remarque : en tant qu'administrateur (root) vous pourriez désactiver la possibilité de vous enregistrer directement.

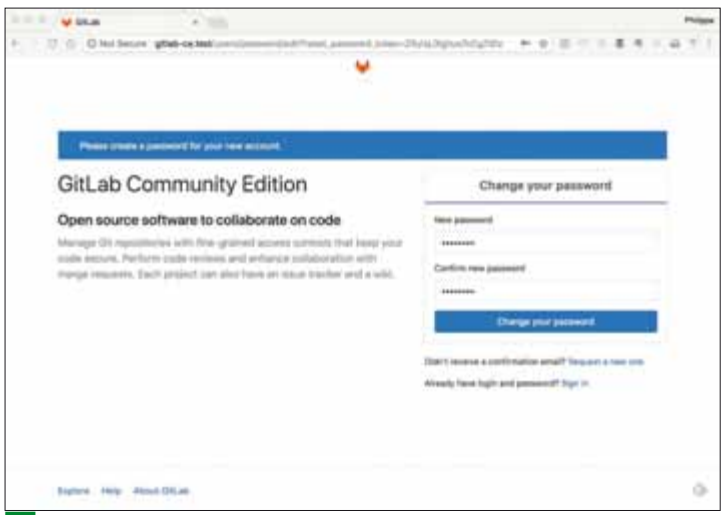
Cliquez sur **Register** et vous allez arriver sur la page d'accueil de votre instance **GitLab** : 4

Renseigner sa clé SSH

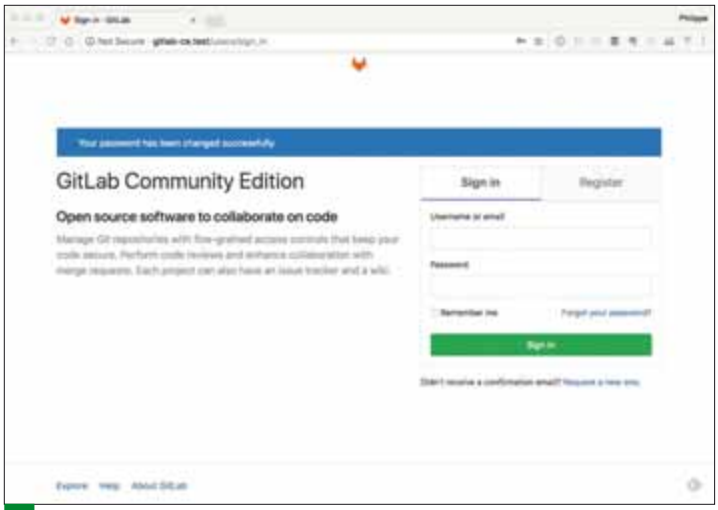
Maintenant pour que votre machine **git** puisse "discuter" (==utiliser des commandes git pour synchroniser votre projet local avec le projet sur le serveur) avec votre instance **GitLab** vous allez devoir renseigner votre clé SSH dans votre profil. Donc en haut à droite dans l'interface web de **GitLab**, déroulez la zone de liste de choix en cliquant sur votre avatar : 5

Sélectionnez **Settings** et vous arrivez sur la page d'édition de votre profil : 6

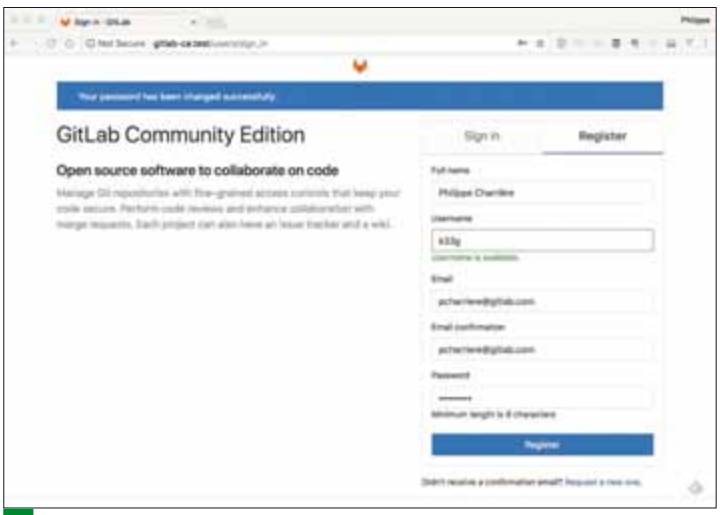
Dans la barre de navigation de gauche, cliquez sur le lien **SSH keys**, et vous allez arriver sur la page de saisie des clés SSH : 7



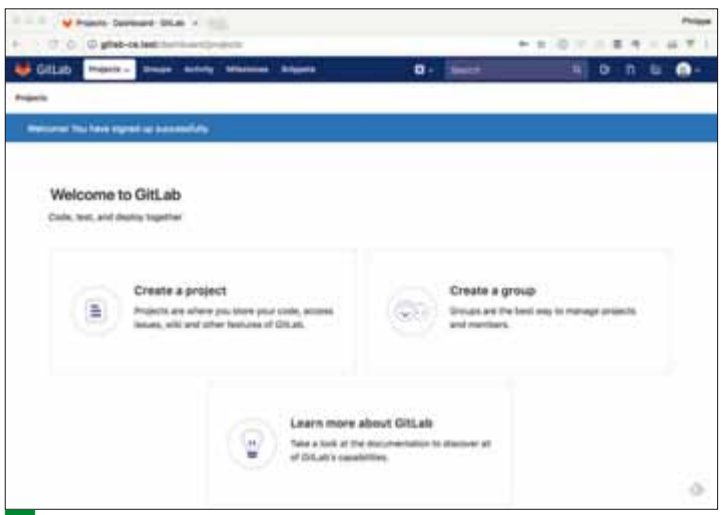
1 *Mot de passe administrateur*



2 *Sign in,*



3 *Création d'un utilisateur*



4

Ouvrez un terminal et tapez la commande cat ~/.ssh/id_rsa_gitlab.pub pour récupérer le contenu de votre clé publique.

Remarque : la mienne s'appelle id_rsa_gitlab.pub, la vôtre aura certainement un nom différent, généralement id_rsa.pub, sinon faites un ls ~/.ssh pour vérifier.
Pour créer une clé SSH vous trouverez les informations nécessaires ici https://docs.gitlab.com/ee/ssh/#generating-a-new-ssh-key-pair.

Copiez le contenu retourné par la commande cat ~/.ssh/id_rsa_gitlab.pub et collez-le dans la zone de texte **Key**, changez si vous le souhaitez le contenue du champ texte **Title** : 8

Cliquez sur le bouton **Add key**.
Voilà, vous êtes maintenant prêt à faire votre 1er projet.

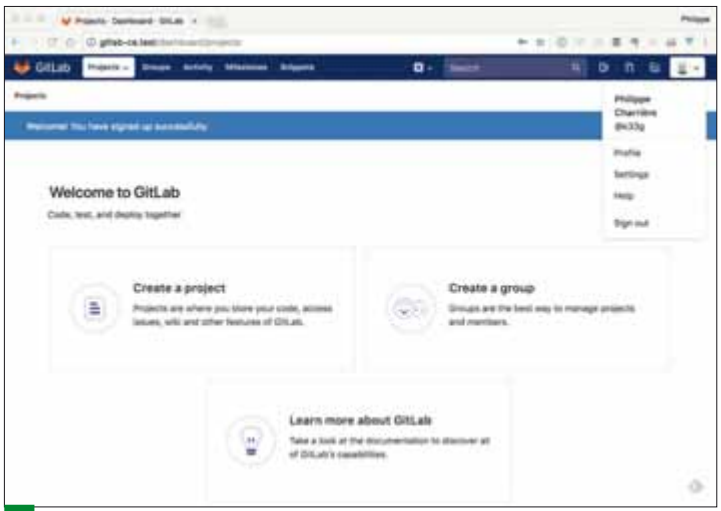
Premier projet

Revenez à la page d'accueil de votre **GitLab**: http://gitlab-ce.test/ 9

Vous avez la possibilité de créer un projet directement rattaché à votre **handle** (votre user de connexion sur **GitLab**), ou de créer des projets dans des groupes ou des sous-groupes. Les groupes vous permettent de vous organiser en équipe ou en projets globaux contenant plusieurs projets (le groupe est un moyen de classement).
Restons simples, nous allons créer un projet directement. Cliquez sur le bouton **Create a project** : 10

Renseignez les informations :
- le nom du projet: hello-world,
- une description du projet: My first project,
- la visibilité du projet: public.

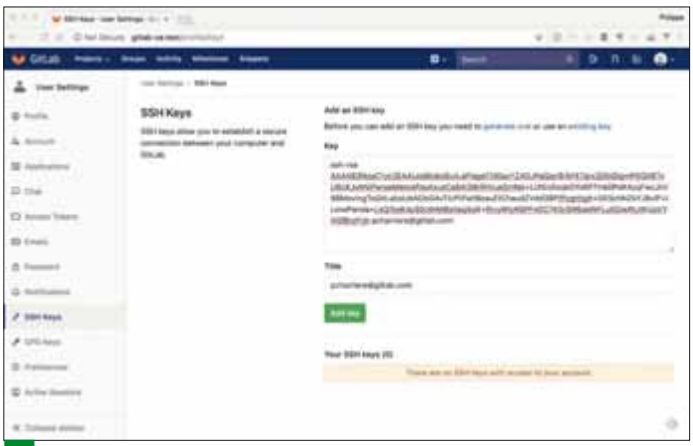
Puis cliquez sur le bouton **Create Project**, vous allez arriver sur cette page : 11



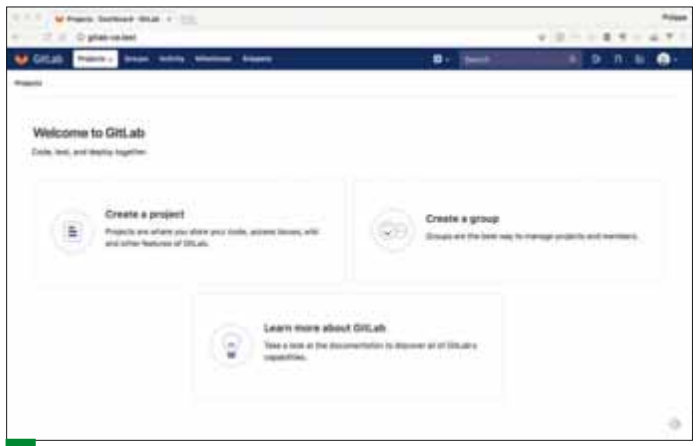
5 *Settings*

6 *Edit Profile*

7 *Add an SSH key*



8 *Add an SSH key*



9 *Welcome to GitLab*

Déroulez (scrollez) un peu, et vous allez trouver les instructions pour créer le projet en local sur votre poste et le "lier" à votre instance **GitLab** : 12

Ouvrez un terminal et tapez les commandes suivantes :

```
git clone git@gitlab-ce.test:k33g/hello-world.git
cd hello-world
touch README.md
```

Vous avez donc cloné le repository sur votre poste et créé un fichier README.md dans ce repository. Editez le fichier README.md avec votre éditeur préféré pour y ajouter une ligne de titre par exemple # Hello World (c'est du markdown).
Maintenant nous allons ajouter le fichier README.md à l'index git (c'est-à-dire : expliquer à git que l'on va suivre les modifications de README.md) et "commiter" les modifications et les pousser vers notre **GitLab** :

```
git add README.md
git commit -m "add README"
git push -u origin master
```

Vous pouvez vérifier sur votre **GitLab** que les modifications ont bien été prises en compte : 13

Si vous aviez un problème de clé SSH (Permission denied)

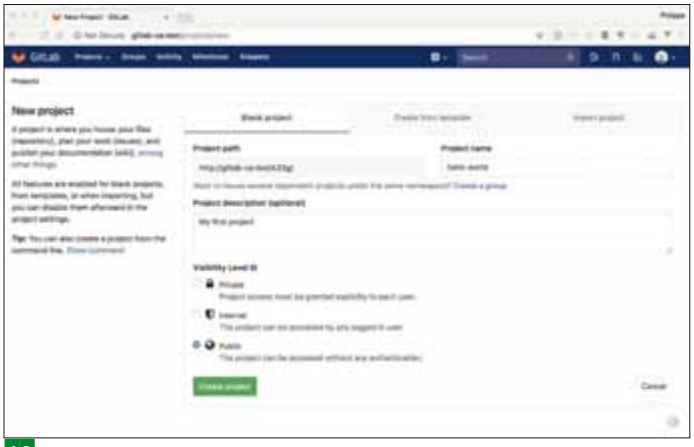
Il se peut que vous ne puissiez pas arriver à vous connecter avec votre clé SSH, et probablement cela signifie que vous utilisez plusieurs clés et que vous les "dédiez" à certains domaines. Dans ce cas-là, éditez le fichier ~/.ssh/config : Par exemple, ici, j'utilise des clés différentes pour **GitLab.com** et **GitHub.com** :

```
# GitLab account
Host gitlab.com
  HostName gitlab.com
  PreferredAuthentications publickey
  IdentityFile ~/.ssh/id_rsa_gitlab

# GitHub account
Host github.com
  HostName github.com
  PreferredAuthentications publickey
  IdentityFile ~/.ssh/id_rsa_github
```

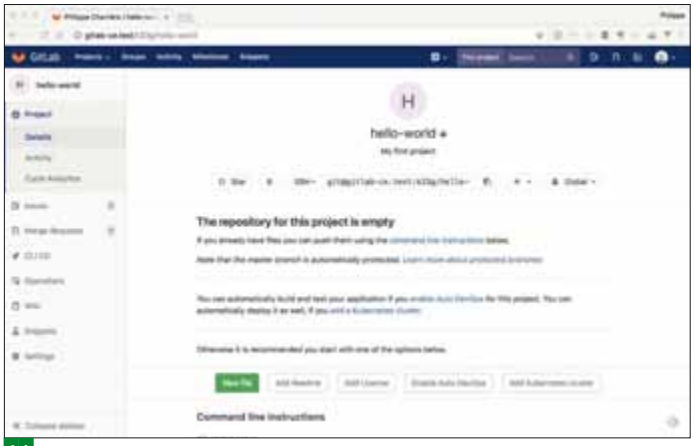
Donc j'ajoute ceci à mon fichier ~/.ssh/config pour pouvoir utiliser ma clé SSH :

```
# Local GitLab Account
Host gitlab-ce.test
  HostName gitlab-ce.test
  PreferredAuthentications publickey
  IdentityFile ~/.ssh/id_rsa_gitlab
```

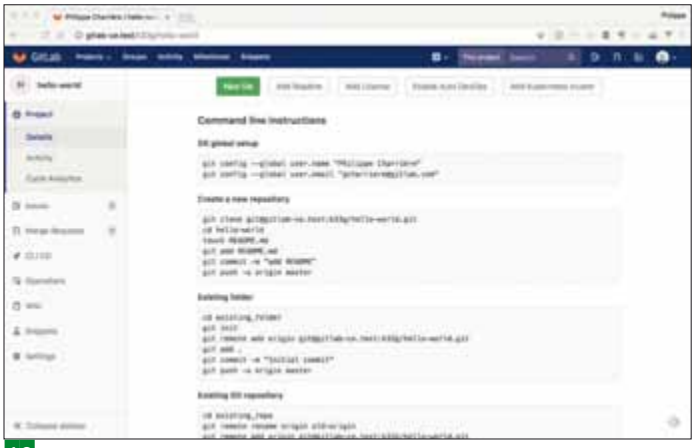
Je sauvegarde, et tout rentre dans l'ordre.



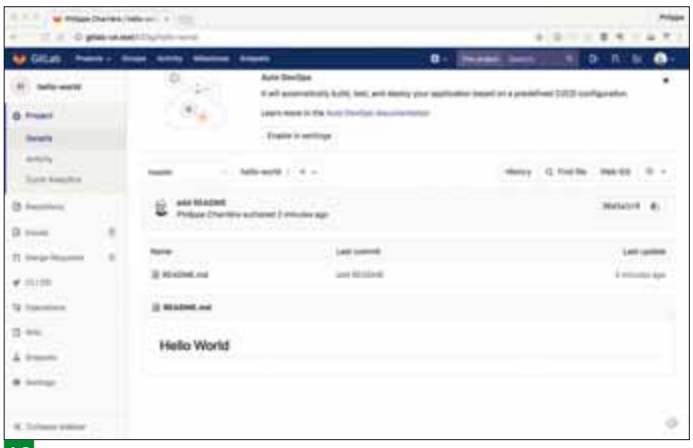
10 *New Project*



11 *The repository for this project is empty*



12 *Command line instructions*



13 *Hello World*

Votre première Merge Request

Définition

une "Merge Request" vous permet de proposer vos changements sur le code source au reste de l'équipe sur un même projet. Plus simplement, c'est une requête (request) pour fusionner (merge) votre branche (vos modifications) sur une autre branche du projet (généralement la branche principale ou "master").

Alors, dans notre cas, vous allez travailler avec vous-même, mais c'est une bonne pratique que d'utiliser des **MR** même pour ses projets personnels. Nous allons donc ajouter quelques fichiers à notre projet en utilisant un workflow de développement très simple.
Donc sur votre poste, dans le répertoire de votre projet, commencez par créer une nouvelle branche **git** en tapant la commande suivante :

```
git branch initialize
```

Si vous tapez git branch vous obtiendrez ceci :

```
 initialize
* master
```

Ce qui signifie que vous avez bien créé une nouvelle branche initialize, mais que vous êtes toujours positionné sur master la branche principale de votre projet. Donc tapez la commande git checkout initialize :

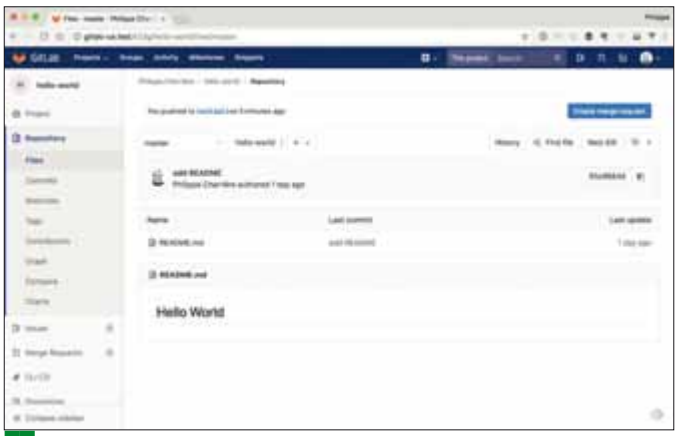
```
Switched to branch 'initialize'
```

Si, à nouveau vous tapez la commande git branch vous obtiendrez ceci :

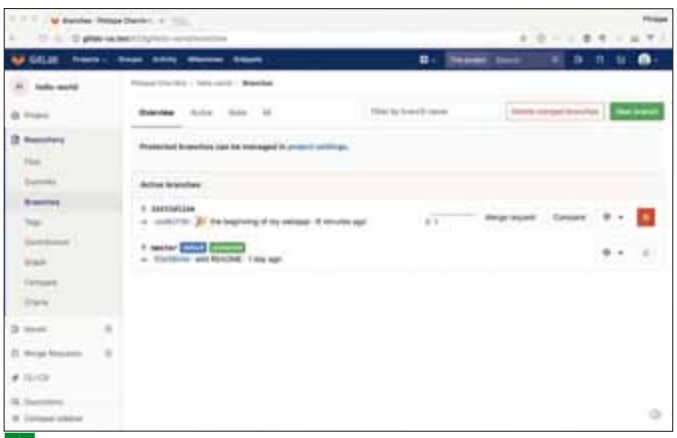
```
* initialize
  master
```

Nous sommes donc sur la bonne branche.

Remarque: pour créer une branche et se positionner sur celle-ci au moment de la création, vous pouvez utiliser la commande git checkout -b initialize



14 *Menu Repository*



15 *Branches*

Le début de notre application

Nous allons créer une webapp **node** (***vous pourrez ensuite adapter selon vos besoins et goûts personnels***). Ajoutez un fichier index.js avec ce contenu :

```
const HTTP = require('HTTP')
const port = process.env.PORT || 8080

const requestHandler = (request, response) => {
 response.writeHead(200, {"Content-Type": "text/html; charset=utf-8"});
 response.end(`
  <div style="text-align: center;">
   <h1 style="font-family: Sans-Serif; font-weight: 300; font-size: 100px;">
    Hello World
   </h1>
  </div>
 `)
}

const server = http.createServer(requestHandler)

server.listen(port, (err) => {
 if (err) {
  console.log('something bad happened', err)
  process.exit(1)
 }
 console.log(`server is listening on ${port}`)
})
```

Ensuite tapez les commandes suivantes pour commencer à suivre ne configuration votre projet :

```
git add index.js
git commit -m ":tada: the beginning of my webapp"
git push
```

Vous devriez obtenir le message suivant :

```
fatal: The current branch initialize has no upstream branch.
To push the current branch and set the remote as upstream, use

  git push —set-upstream origin initialize
```

Donc faites ce que vous dit **git** et tapez la commande suivante :

```
git push —set-upstream origin initialize
```

16 *New merge request*

Et vous obtiendrez quelque chose qui ressemble à ceci :

```
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 660 bytes | 660.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for initialize, visit:
remote:  http://gitlab-ce.test/k33g/hello-world/merge_requests/new?merge_request%
5Bsource_branch%5D=initialize
remote:
To http://gitlab-ce.test/k33g/hello-world.git
 * [new branch]     initialize -> initialize
Branch 'initialize' set up to track remote branch 'initialize' from 'origin'.
```

Si vous allez faire un tour sur http://gitlab-ce.test/k33g/hello-world et allez dans le menu **Repository** : 14

Et que vous choisissez le sous-menu **Branches**, vous pouvez vérifier que votre nouvelle branche a bien été "**poussée**" sur le serveur : 15

Notez le bouton **Merge Request** au niveau de branche initialize (mais ne cliquez pas dessus). Revenez sur la page d'accueil de votre projet. En haut à droite vous pouvez noter le bouton **Create merge request**, cliquez dessus et vous allez obtenir le formulaire suivant : 16

C'est à ce niveau que vous allez décrire ce que vous comptez proposer. Si vous déroulez l'écran, vous voyez que vous pouvez qualifier la merge request de différentes manières (comme lui affecter un responsable, une milestone, une date - nous pourrons traiter ceci dans un prochain article).
Vous pouvez voir aussi l'historique de commit liez à votre merge request ainsi qu'un bouton **Submit merge request**.
Cliquez sur **Submit merge request** pour soumettre votre merge request. Vous pouvez voir que votre **merge request** a bien été enregistrée, et vous pourriez même la merger tout de suite en cliquant sur le bouton **Merge**, mais ne le faites pas.

Complétons la merge request

Sur votre poste dans votre répertoire projet, ajoutez un fichier package.json (nous faisons une application node), avec le contenu suivant :

```
{
 "name": "hello-world",
 "main": "index.js",
 "scripts": {
  "test": "node tests.js",
  "start": "node index.js"
 }
}
```

Puis commitez votre fichier et poussez votre modification vers votre instance **GitLab** :

```
git add package.json
git commit -m ":wrench: add package.json"
git push
```

Et retournez "voir" votre **merge request** (http://gitlab-ce.test/k33g/hello-world/merge_requests/1), sélectionnez l'onglet "**Commits**" et vous pouvez voir que votre **merge request** continue à se mettre à jour avec l'historique des commits.
Vous pouvez donc maintenant merger votre branche initialize sur la branche master en cliquant sur le bouton **Merge** (cochez aussi **Remove source branch**) : 17

Si vous revenez sur la page d'accueil de votre projet, vous pouvez voir que la branche **master** contient bien vos modifications.

Synchronisez votre dossier projet

Sur votre poste, tapez les commandes suivantes :

```
git checkout master # vous vous repositionnez sur la branche master
git pull # vous récupérez les modifications du serveur.
```

Vous devriez obtenir quelque chose comme ceci :

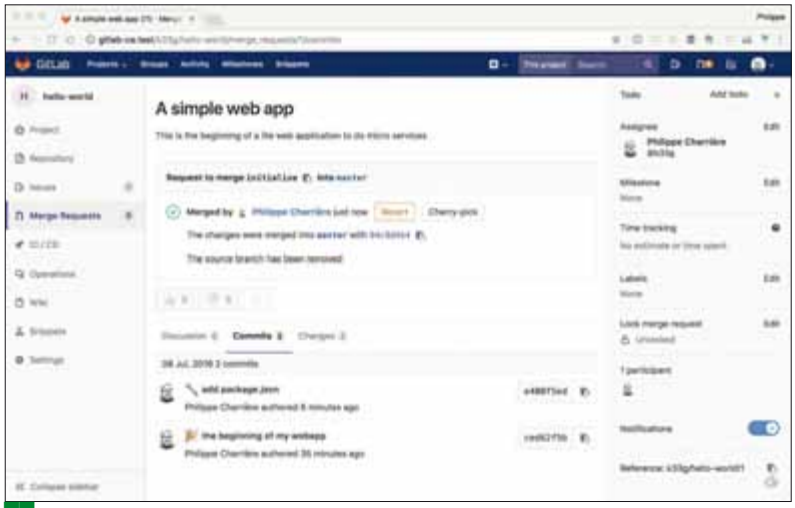
```
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (1/1), done.
From http://gitlab-ce.test/k33g/hello-world
  95e9664..94c9d4e  master     -> origin/master
Updating 95e9664..94c9d4e
Fast-forward
 index.js    | 23 ++++++++++++++++++++++
 package.json |  8 ++++++++
 2 files changed, 31 insertions(+)
 create mode 100644 index.js
 create mode 100644 package.json
```

Donc votre branche master est à jour, et vous pouvez supprimer votre branche locale initialize avec la commande :

```
git branch -d initialize
```

Voilà, à ce niveau vous avez tout ce qu'il faut pour faire vivre votre projet et le synchronyser avec votre instance **GitLab**. Il est maintenant temps d'utiliser les **GitLab Runners**.

Fin de la 1ere partie



17 *Merged*

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

Nos classiques

1 an
11 numéros
49€*

2 ans
22 numéros
79€*

Etudiant
1 an - 11 numéros
39€*

** Tarifs France métropolitaine*

Abonnement numérique

PDF ..... **35€**
1 an - 11 numéros
Souscription uniquement sur www.programmez.com
Option : accès aux archives 10€

© mikkelwilliam

Offres 2018

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

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

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

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

** Offre limitée à la France métropolitaine*

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

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

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

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

☐ Vidéo : Symfony 3
☐ Vidéo : Raspberry Pi

PROG 221
Offre limitée, valable jusqu'au 28 septembre 2018

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

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

E-mail : @

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

Offres 20e anniversaire !*

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

+

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

+

4 numéros vintages (selon les stocks)

+

1 carte maker Digispark ATtiny84 compatible Arduino

+

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

+

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

** Offre réservée à la France métropolitaine*

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

Christophe PICHAUD
Consultant sur les technologies Microsoft
christophepichaud@hotmail.com |
www.windowscpp.com

niveau 300

Au cœur d'un service Windows qui fait du NoSQL

Partie 2

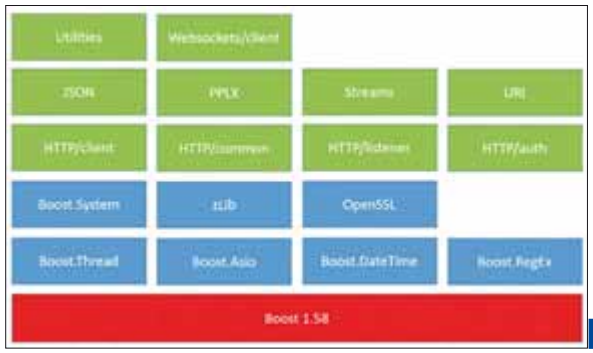
Maintenant que nous savons comment créer un service Windows, et que nous avons un thread pour vivre notre vie d'application principale, comment allons-nous faire pour mettre du code dans ce thread ? Et surtout on va y faire quoi ? Je vous propose de hoster un service REST Web API et d'exposer 2 méthodes get-data et set-data pour gérer une base NoSQL. C'est simple et ça peut servir…

Rappel de la fin de la partie I

Le code principal du service est ici :

```
UINT AutomateThread(LPVOID pParam)
{
    CString strLog;
    try
    {
        CoInitialize(NULL);

        // Add specific code here

        while (TRUE)
        {
            if (_Module.m_bStop)
            {
                break;
            }
            Sleep(200);

        } // Main loop

    }
    catch (...)
    {
        strLog.Format(_T("EXCEPTION: Arrêt du service"));
        //_Module.LogEvent(strLog);
    }

    CoUninitialize();
    return 0;
}
```

Nous sommes dans un thread qui sait gérer l'arrêt du service. Toute la plomberie est câblée. Maintenant, dans un service, on y met quoi ? On peut y mettre un point d'écoute comme un Pipe, une socket ou un handler http ? Je vous propose de mettre un service Rest Web API qui est autonome. Vous voulez dire une DLL que l'on charge sous IIS M'sieur ? Non, un serveur Rest complet qui gère le http dans sa totalité ! Ainsi on est autonome… Et comment on fait ça en C++ M'sieur ? On utilise le REST SDK de Microsoft qui permet de faire des applications portables REST JSON en C++.



1

Introduction au MS Rest SDK

Disponible sous GitHub à l'adresse https://github.com/Microsoft/cpprestsdk , ce SDK est basé du C++ ISO multiplateformes. Cela veut dire que vous pouvez builder sous Android, Windows, Linux et Mac.
Voici l'architecture de la solution (en rouge Boot, en bleu des modules et le code à compiler en vert): 1
Au travers ces noms de modules à compiler, on devine l'architecture de cette librairie. La librairie Boost permet d'être multiplateforme facilement et efficacement. Boost est reconnu dans la communauté C++ pour être à l'avant-garde de ce qui se fait de mieux. Voilà pour l'architecture. Pour ce qui est de l'utilisation, il faut maintenant compiler la librairie via Visual Studio pour x64 en Release et x64 en Debug. Une fois que c'est fait, on dispose d'un module debug de 12.5 MB et d'un module release de 6.3 MB. Donc le MS CPP REST SDK va nous permettre de hoster notre propre serveur Web avec des handlers http sur des ports et des verbes http. Génial !
On va hoster un service sur un port avec une URL donnée… Regardons ce qu'il se passe au lancement du serveur :

```
D:\Dev\GitHub_LMDB2\LMDBWindows\x64\Debug>myserver
08:39:42.324 - INFO - Init Master...
08:39:42.325 - INFO - IP : 192.168.175.241
08:39:42.408 - INFO - Worker node http://192.168.175.241:7001/MyServer/LMDB/
08:39:42.408 - INFO - Press ENTER to exit.
```

Dans notre cas, l'URL est /MyServer/LMDB et le port est 7001. On est capable de capter toutes les requêtes GET…
Reste à découvrir le code C++ qui permet de faire de si jolies choses… Si on regarde le code ci-dessus, on voit que l'adresse est

passée au constructeur de la classe. Ensuite on fait appel à la méthode Init(). Regardons ces deux bouts de code :

```
TheServer::TheServer(std::wstring url) : m_listener(url)
{
    std::function<void(http_request)> fnGet = &TheServer::handle_get;
    m_listener.support(methods::GET, fnGet);
    std::function<void(http_request)> fnPost = &TheServer::handle_post;
    m_listener.support(methods::POST, fnPost);
    std::function<void(http_request)> fnDel = &TheServer::handle_del;
    m_listener.support(methods::DEL, fnDel);
    std::function<void(http_request)> fnPut = &TheServer::handle_put;
    m_listener.support(methods::PUT, fnPut);

    this->_url = url;
}
```

On y aperçoit un membre m_listener à qui on passe une url et ensuite des pointeurs de fonctions sont passés à des méthodes http. Simple ! Le type m_listener est définit comme suit :

```
    http_listener m_listener;
```

Voyons maintenant la méthode Init() :

```
void TheServer::Init()
{
    this->_http = std::unique_ptr<TheServer>(new TheServer(this->_url));
    this->_http->open().wait();
}
```

Quel est donc ce membre _http ?

```
    std::unique_ptr<MyServer> _http;
```

C'est un smart pointeur sur la classe MyServer. Il y a une dernière méthode à expliquer :

```
    pplx::task<void> open() { return m_listener.open(); }
```

Dans le thread, voici les étapes d'initialisation à ajouter – le code d'exécution est là :

```
DWORD AutomateThread(LPVOID pParam)
{
    CString strLog;
    try
    {
        CoInitialize(NULL);

        std::wstring port = Constants::MasterNodePort;
        std::wstring ip = ServerHelper::GetIP();
        std::wstring url = ServerHelper::BuildURL(ip, port);
        http::uri uri = http::uri(url);
        std::wstring address = uri.to_string();
        std::wstring name = _T("Master");

        TCHAR sz[255];
        _stprintf_s(sz, _T("Node_%s_%s.log"), port.c_str(), name.c_str());
        g_Logger.Init(sz);
        g_Logger.WriteLog(_T("Init Node..."));

        //
        // Create the server node instance
        //

        _stprintf_s(sz, _T("IP : %s"), ip.c_str());
        g_Logger.WriteLog(sz);

        TheServer client(address);
        client._server = ip;
        client._port = port;
        client._name = name;
        client.Init();
        _stprintf_s(sz, _T("Worker node %s"), address.c_str());
        g_Logger.WriteLog(sz);

        while (TRUE)
        // …{
```

Voilà, on peut voir que hoster un serveur REST est d'une facilité terrible…
Maintenant le corps du serveur REST, c'est sa méthode GET :

```
void TheServer::handle_get(http_request message)
{
    g_Logger.WriteLog(_T("handle_get"));

    PrintRequest(message);

    // http://192.168.175.241:7001/MyServer/LMDB/?request=set-data&key=toto0&value
=toto1&name=cache2
    // http://192.168.175.241:7001/MyServer/LMDB/?request=get-data&key=toto2_k&
name=cache3

    std::wstring request = ServerHelper::FindParameter(message, _T("request"));

    if (request == Constants::VerbPing)
    {
        RequestVerbPing(message);
        return;
    }

    if (request == Constants::VerbGetData)
    {
        RequestVerbGetData(message);
        return;
    }

    if (request == Constants::VerbSetData)
    {
        RequestVerbSetData(message);
```

```
        return;
    }

    message.reply(status_codes::OK);
}
```

Comme vous pouvez le voir, c'est propre. Une ou deux helper fonctions et hop le tour est joué !

```
std::wstring ServerHelper::FindParameterInQuery(std::map<std::wstring, std::wstring>query,
 std::wstring var)
{
    auto it = query.find(var);
    std::wstring value;
    if (it != query.end())
    {
        value = it->second;
        std::wcout << _T("Extracted param ") << var << _T(": ") << value << std::endl;
    }
    return value;
}

std::wstring ServerHelper::FindParameter(http_request message, std::wstring var)
{
    auto query = uri::split_query(uri::decode(message.relative_uri().query()));
    std::wstring value = ServerHelper::FindParameterInQuery(query, var);
    return value;
}
```

Dans ma méthode GET, j'ai encapsulé les réponses que je fais à des verbes…

```
void TheServer::RequestVerbGetData(http_request message)
{
    USES_CONVERSION;
    CLMDBWrapper lmdb;

    g_Logger.WriteLog(Constants::VerbGetData.c_str());

    std::wstring key = ServerHelper::FindParameter(message, _T("key"));
    std::wstring dbNameW = ServerHelper::FindParameter(message, _T("name"));
    std::string dbName(dbNameW.begin(), dbNameW.end());

    if (lmdb.Init((LPSTR)dbName.c_str()) == false)
    {
        g_Logger.WriteLog(_T("LMDB Init not done !"));
        message.reply(status_codes::OK);
        return;
    }

    char szKey[255];
    LPSTR lpszValue;

    strcpy_s(szKey, W2A(key.c_str()));

    if (lmdb.GetData((LPSTR)szKey, &lpszValue) == true)
    {
        Data data;
        data.key = key;
        data.value = std::wstring((A2W(lpszValue)));

        free(lpszValue);

        TCHAR sz[255];
        _stprintf_s(sz, _T("Get Key:%s Value:%s"), data.key.c_str(), data.value.c_str());
        g_Logger.WriteLog(sz);

        std::wstring response = data.AsJSON().serialize();
        g_Logger.WriteLog(response.c_str());

        message.reply(status_codes::OK, data.AsJSON());
    }
    else
    {
        message.reply(status_codes::OK);
    }

    lmdb.Uninit((LPSTR)dbName.c_str());
}

void TheServer::RequestVerbSetData(http_request message)
{
    USES_CONVERSION;
    CLMDBWrapper lmdb;
    g_Logger.WriteLog(Constants::VerbSetData.c_str());

    std::wstring key = ServerHelper::FindParameter(message, _T("key"));
    std::wstring value = ServerHelper::FindParameter(message, _T("value"));
    std::wstring dbNameW = ServerHelper::FindParameter(message, _T("name"));
    std::string dbName(dbNameW.begin(), dbNameW.end());

    if (lmdb.Init((LPSTR)dbName.c_str()) == false)
    {
        g_Logger.WriteLog(_T("LMDB Init not done !"));
        message.reply(status_codes::OK);
        return;
    }

    LPSTR lpszKey = W2A(key.c_str());
    LPSTR lpszValue = W2A(value.c_str());
    lmdb.SetData(lpszKey, lpszValue);

    char sz[255];
    sprintf_s(sz, "Set Key:%s Value:%s", lpszKey, lpszValue);
    g_Logger.WriteLog(A2W(sz));

    Data data;
    data.key = A2W(lpszKey);
    data.value = A2W(lpszValue);

    std::wstring response = data.AsJSON().serialize();
    g_Logger.WriteLog(response.c_str());
```

```
    message.reply(status_codes::OK, data.AsJSON());

    lmdb.Uninit((LPSTR)dbName.c_str());
}
```

Vous le voyez, il ne faut pas beaucoup de code pour arriver au résultat voulu. Le code est plutôt simple à écrire. Vous remarquerez aussi que l'accès à la librairie LMDB est encapsulée dans la classe CLMDBWrapper. C'est important de construire des classes séparées. 2

Pour tester le service Web, il suffit de lancer un browser… et de lancer cette url : http://192.168.112.1:7001/MyServer/LMDB/?request=set-data&key=key_toto0&value=value_toto0&name=cacheA2 3

Voici la log du service : 4

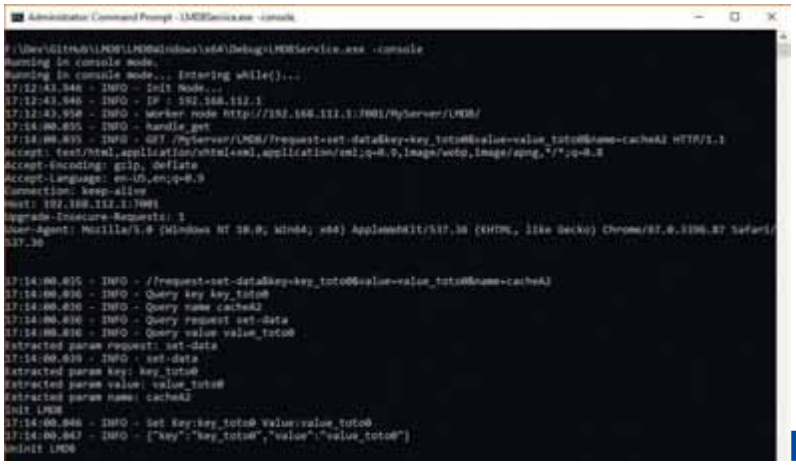
Pour obtenir la valeur de la clé key_toto0, utilisons l'url suivante : http://192.168.112.1:7001/MyServer/LMDB/?request=get-data&key=key_toto0&name=cacheA2

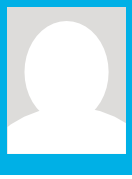
Conclusion

Le service Windows est une application Windows pas tout a fait comme les autres. Il faut s'exécuter dans un thread, savoir gérer un arrêt qui peut survenir n'importe quand. Un service Windows ne possède pas de code d'interface graphique, par contre c'est une boite noire ; c'est de la communication. Cela nécessite des compétences « server » car on doit logger des choses et que on l'utilise en aveugle : ça fait le job tout seul, sans intervention humaine. •

2


3


4



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

niveau 200

Réalisation d'un solveur de Sudoku s'appuyant sur les Dancing Links

Le Sudoku est un jeu de puzzle d'origine américaine popularisé en France au début des années 2000. La réalisation d'un programme permettant de résoudre automatiquement une grille de Sudoku constitue toujours un exercice intéressant pour les débutants en programmation. Dans cet article, nous vous proposons de découvrir comment développer un solveur de Sudoku en Java s'appuyant sur une technique ultra efficace répondant au doux nom de Dancing Links.

Le Sudoku est un jeu de type puzzle se présentant sous la forme d'une grille. Le but du jeu consiste à remplir cette grille avec une série de chiffres tous différents et ne se trouvant jamais plus d'une fois sur une même ligne, dans une même colonne ou dans un même bloc. Dans un Sudoku classique de taille 9, ces blocs auront donc des tailles de 3 par 3 et les valeurs pouvant être affectées aux cellules iront de 1 à 9. Pour des instances de Sudoku plus complexes, les blocs pourront être de 4 par 4 (Sudoku de taille 16 avec des valeurs allant de 1 à 16) ou encore de 5 par 5 (Sudoku de taille 25 avec des valeurs allant de 1 à 25).

Une première approche

La lecture des règles du Sudoku montre clairement que ce jeu est en fait un problème de satisfaction de contraintes. Une première approche simple de résolution consiste à appliquer un algorithme de BackTracking récursif dans lequel nous allons tenter de résoudre une grille en affectant à chaque cellule toutes les valeurs possibles avant de passer à la cellule suivante si l'affectation précédente respecte les 4 contraintes d'une grille valide Sudoku. Lorsqu'une affectation ne respecte pas ces contraintes, on revient en arrière et on tente d'affecter une nouvelle valeur. En procédant ainsi, on peut programmer facilement un algorithme de résolution du Sudoku.
Afin de mieux mettre en évidence la puissance de la méthode des Dancing Links en termes de performance, nous allons réaliser cette première implémentation à titre de comparaison. Nous commençons donc par modéliser une grille de Sudoku sous la forme d'un tableau d'entiers à 2 dimensions. Chaque cellule de ce tableau pouvant prendre une valeur allant de 0 à 9 avec le 0 indiquant que la cellule n'a pas encore été affectée avec une valeur légale.

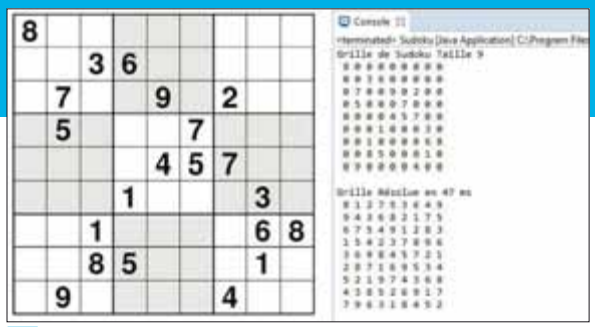
Vérification des contraintes

Ensuite, nous définissons les méthodes permettant de vérifier qu'une grille est valide ou non. La vérification de l'unicité des valeurs sur une ligne se fait en parcourant la ligne à laquelle appartient la cellule que l'on souhaite valoriser. Si la valeur en cours d'affectation est déjà présente dans la ligne, alors on renvoie true, ce qui indique que la valeur est déjà présente. La vérification de l'unicité des valeurs d'une colonne se fait de la même manière en parcourant l'ensemble des cellules de la colonne à laquelle appartient la cellule que l'on souhaite valoriser. Si la valeur en cours d'affectation est déjà présente dans la colonne, alors on renvoie true. Ceci nous donne l'implémentation suivante pour les méthodes isInRow et isInCol :

```
private boolean isInRow(int row, int number) {
 for (int i = 0; i < SIZE; i++)
  if (board[row][i] == number)
   return true;

 return false;
}

private boolean isInCol(int col, int number) {
 for (int i = 0; i < SIZE; i++)
  if (board[i][col] == number)
   return true;

 return false;
}
```

Il nous faut ensuite vérifier qu'au sein de la boîte de rattachement de la cellule que l'on souhaite valoriser, la valeur assignée n'est pas déjà présente. On effectue une petite opération mathématique à l'aide de l'opérateur modulo pour se placer au début de la boîte de rattachement de la cellule courante. On parcourt ensuite les différentes cellules de la boîte et on renvoie true si la valeur que l'on souhaite assignée est déjà présente. Enfin, on termine en définissant une méthode isOk se chargeant d'appeler ces 3 méthodes afin de retourner true uniquement si l'affectation d'une valeur à une cellule donnée respecte les contraintes du Sudoku :

```
private boolean isInBox(int row, int col, int number) {
 int r = row - row % BOX;
 int c = col - col % BOX;

 for (int i = r; i < r + BOX; i++)
  for (int j = c; j < c + BOX; j++)
   if (board[i][j] == number)
    return true;

 return false;
}

private boolean isOk(int row, int col, int number) {
 return !isInRow(row, number) &&
     !isInCol(col, number) &&
     !isInBox(row, col, number);
}
```



1 *Résolution d'une grille de taille 9*

Les lecteurs habitués à résoudre des grilles de Sudoku auront probablement remarqué que la quatrième contrainte liée à une grille de Sudoku n'a pas encore été vérifiée à ce stade. En effet, pour qu'une grille soit valide, il faut également que toutes les cellules de la grille soient valorisées. Cette contrainte sera implémentée directement dans l'algorithme de résolution comme nous allons le voir.

Algorithme de BackTracking

Les méthodes de validation d'une grille étant en place, nous allons pouvoir passer à l'implémentation de notre algorithme de BackTracking. Cette dernière repose sur une approche récursive au sein d'une méthode solve. Nous allons ainsi parcourir la grille à la recherche de la première cellule vide, c'est-à-dire valorisée par 0. Une fois cette cellule trouvée, nous essayons de lui affecter une valeur valide. Lorsqu'une valeur valide est trouvée, on rappelle à nouveau la méthode solve pour passer au niveau suivant de notre algorithme de BackTracking. Si, l'affectation n'est pas valide au niveau courant, on retourne false et on remet la cellule à 0 ce qui a pour effet de remonter d'un cran dans notre algorithme de BackTracking au sein duquel une nouvelle valeur est testée sur la cellule courante. Cette approche nous permet d'obtenir une solution à notre grille lorsque toutes les affectations ont été réalisées de manière valide. Ainsi, une fois sortie de la méthode solve, la propriété grid de notre classe Sudoku contiendra la grille sous sa forme résolue. Il est bon de noter que notre algorithme s'arrête dès qu'une solution est trouvée. Néanmoins, cela n'est pas un réel problème puisqu'une grille de Sudoku correcte n'est sensée avoir qu'une seule et unique solution. On obtient ainsi le code suivant pour la méthode solve qui contient le coeur de notre algorithme de résolution :

```
public boolean solve() {
 for (int row = 0; row < SIZE; row++) {
  for (int col = 0; col < SIZE; col++) {
   if (board[row][col] == EMPTY) {
    for (int number = 1; number <= SIZE; number++) {
     if (isOk(row, col, number)) {
      board[row][col] = number;

      if (solve()) {
       return true;
      } else {
       board[row][col] = EMPTY;
      }
     }
    }

    return false;
   }
  }
 }
```

```
 return true;
}
```

Résolution d'une grille

Avec ce code, nous pouvons instancier notre classe Sudoku avec une grille de taille 9 à résoudre puis lancer notre programme Java afin d'obtenir la résolution de la grille. La figure 1 permet de visualiser le travail réalisé par notre programme et on peut constater que la résolution s'exécute instantanément puisque le temps d'exécution est de l'ordre de 50 ms sur un ordinateur moyen.
Néanmoins, l'algorithme de résolution que nous avons implémenté montre rapidement ses limites pour des grilles plus complexes de taille 16 ou 25 par exemple. C'est là qu'entre en jeu la méthode Dancing Links et le fameux algorithme X sur lequel nous allons nous appuyer dans la suite de cet article.

Dancing Links et algorithme X

En l'an 2000, Donald Knuth, célèbre professeur d'informatique Américain, publie un article ayant pour titre "Dancing Links" au sein duquel il décrit l'utilisation de l'algorithme X pour résoudre des grilles de Sudoku. Cet article s'intéresse dans un premier temps au problème de la couverture exacte et à l'algorithme X permettant sa résolution de manière rapide et efficace. Dans un second temps, Donald Knuth montre comment transformer le problème de la résolution d'une grille de Sudoku en une instance du problème de la couverture exacte. Une fois, cette modélisation faite, il est aisé d'appliquer l'algorithme X et de résoudre des grilles de Sudoku très complexes. Le tout avec d'excellentes performances.

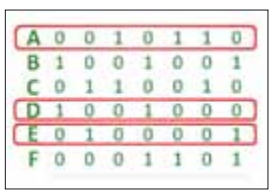
Problème de la couverture exacte

Avant toute chose, il paraît important de s'arrêter quelques instants sur le problème de la couverture exacte. Enoncé pour la première fois par l'informaticien Américain Richard Karp, le problème de la couverture exacte peut être énoncé de la manière suivante :

« Etant donnée une matrice binaire, c'est-à-dire composée uniquement de 0 et de 1, il faut trouver un ensemble de lignes contenant exactement un 1 dans chaque colonne. »

Considérons la matrice binaire présentée à la figure 2. Une solution au problème de la couverture exacte pour cette matrice pourra être le sous-ensemble de lignes A, D et E entourées en rouge.

Solution au problème de la couverture exacte 2



Algorithme X

Dans son article sur les Dancing Links, Donald Knuth décrit l'algorithme X qui permet de solutionner le problème de la couverture exacte. Il s'agit là encore d'un algorithme de recherche récursif utilisant la méthode du BackTracking. Néanmoins, son application s'avère particulièrement performante comme nous allons le constater par la suite. Appliqué à des matrices binaires, l'algorithme peut être formulé comme suit :

- Si la matrice A est vide, le problème est résolu. Retourner succès ;
- Autrement, choisir une colonne c dans la matrice A ;
- Choisir une ligne r dans la colonne c ;
- Inclure la ligne r dans la solution partielle ;

- Pour chaque indice j tel que A[r,j] = 1:
  - Supprimer la colonne d'indice j de la matrice A ;
  - Pour chaque indice i tel que A[i,j] = 1:
    - Supprimer la ligne d'indice i de la matrice A ;
- Répéter l'algorithme récursivement sur la matrice réduite A.

Les Dancing Links entrent dans la danse

La technique des Dancing Links décrite par Donald Knuth va nous permettre d'implémenter la formulation de l'algorithme X appliqué à des matrices binaires d'une manière remarquablement ingénieuse. Au coeur des Dancing Links, nous allons ainsi retrouver une structure de données quadruplement chaînée. La figure 3 présente ainsi la représentation graphique de la matrice binaire détaillée à la figure 2 précédemment sous forme de liste quadruplement chaînée. Implémenter une matrice binaire représentant une instance du problème de la couverture exacte via une liste quadruplement chaînée va s'avérer très avantageux à plusieurs titres. Tout d'abord, cette structure nous permet de gagner de la place en mémoire en tirant parti du fait que bien souvent les matrices représentant une instance du problème de la couverture exacte sont clairsemées. Ensuite, les manipulations réalisées à chaque appel récursif s'appliquent à la représentation initiale de la matrice. Toutes les manipulations sont faites in situ, il n'y a pas de recopie et de création de matrices intermédiaires. Enfin, les opérations de suppression et de réinsertion de colonnes sur les matrices de couverture vont se faire en temps constant. Ceci vient principalement du fait qu'au sein d'une liste circulaire doublement chaînée, la suppression d'un nœud x peut être réalisée facilement en jouant avec les pointeurs comme suit :

```
x.left.right = x.right;
x.right.left = x.left;
```

En considérant que x.right et x.left n'ont pas été modifiés entre temps, la réinsertion du noeud x à la même position au sein de la liste se fera alors comme suit :

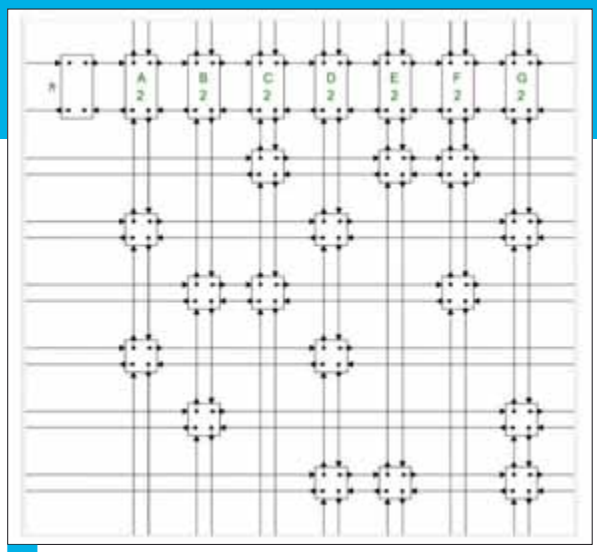
```
x.left.right = x;
x.right.left = x;
```

Il est bon de noter que le principe demeure le même pour les suppressions ou réinsertions dans le sens vertical.

Grille de Sudoku en tant que matrice de couverture

Ces fondamentaux théoriques mis en place, nous pouvons passer à la partie implémentation. C'est maintenant que nous allons voir comment résoudre une grille de Sudoku via la méthode des Dancing Links et l'algorithme X. La première étape va consister à modéliser une grille de Sudoku sous la forme d'une matrice de couverture. Cette modélisation va prendre en compte les 4 contraintes définies par le jeu de Sudoku pour qu'une grille résolue soit considérée comme valide :

- **Contrainte Cellule :** chaque cellule peut contenir seulement un entier compris entre 1 et SIZE qui correspond à la taille de notre grille.
- **Contrainte Ligne :** chaque ligne peut contenir seulement SIZE entiers uniques compris entre 1 et SIZE.
- **Contrainte Colonne :** chaque colonne peut contenir seulement SIZE entiers uniques compris entre 1 et SIZE.
- **Contrainte Boîte :** chaque boîte peut contenir seulement SIZE entiers uniques compris entre 1 et SIZE.



3 *Matrice Binaire sous forme de liste quadruplement chaînée*

Chaque ligne de la matrice de couverture que nous allons définir contiendra l'ensemble de ces contraintes. Ainsi, en considérant une grille de taille SIZE avec un nombre de contraintes par cellules égal à CONSTRAINTS et avec une plage de valeurs comprises entre 1 et MAX_VALUE pour une cellule, la matrice de couverture aura alors la taille suivante :

```
(SIZE * SIZE * MAX_VALUE) x (SIZE * SIZE * CONSTRAINTS)
```

Dans notre classe Sudoku, nous allons ainsi pouvoir implémenter ces méthodes spécifiques afin de transformer une instance de grille de Sudoku en matrice de couverture :

Voir code complet sur www.programmez.com

A ce stade, l'exécution de notre programme nous permet de constater que la création de la matrice de couverture pour une instance de grille Sudoku de taille 9 est correcte (figure 4).

Modélisation des éléments de la liste

A partir de cette matrice de couverture, nous allons maintenant pouvoir créer la liste quadruplement chaînée qui nous permettra d'implémenter l'algorithme X et l'approche Dancing Links de Donald Knuth. Dans cette optique, nous définissons un objet DancingNode afin de modéliser un noeud de notre liste. Cet objet aura 4 propriétés respectivement nommées left, right, top et bottom qui pointeront vers d'autres noeuds de la liste dans 4 directions. Enfin, une propriété column de type ColumnNode est utilisée pour relier le noeud courant à la colonne d'appartenance dans la matrice de couverture. Au niveau de ses méthodes, la classe DancingNode va implémenter les opérations de suppression et de réinsertion de noeuds au sein de la liste que nous avons détaillée précédemment, et ce, de manière horizontale mais également verticale. Quant à la classe ColumnNode, il s'agit d'un objet héritant de DancingNode qui va proposer les méthodes cover et uncover permet tant comme leurs noms l'indiquent de couvrir ou de découvrir une colonne dans la structure de données. Ceci nous donne le code suivant pour la classe DancingNode et la classe ColumnNode : **Voir code complet sur** www.programmez.com

Création de la liste quadruplement chaînée

Les objets DancingNode et ColumnNode créés, nous allons pouvoir définir la liste quadruplement chaînée représentant la matrice de couverture précédemment générée. Cela se fait au sein d'une mé-

4 *Conversion de la grille en matrice de couverture*

thode createDLXList prenant en entrée la matrice de couverture et renvoyant le noeud de type ColumnNode en sortie. En vous reportant à la figure 3, vous pourrez constater qu'il s'agit du noeud situé en haut à gauche dans la structure de données. La création est réalisée en deux étapes. Premièrement, on crée l'ensemble des colonnes. Ensuite, on va créer les noeuds en parcourant la matrice de couverture à la recherche des cellules valorisées à 1 afin de rajouter les noeuds nécessaires. Tout ceci nous donne le code suivant :

```
public class DLX {

 private ColumnNode header;
 private List<DancingNode> answer;
 public List<DancingNode> result;

 public DLX(int[][] cover) {
  header = createDLXList(cover);
 }

 private ColumnNode createDLXList(int[][] grid) {
  final int nbColumns = grid[0].length;
  ColumnNode headerNode = new ColumnNode("header");
  List<ColumnNode> columnNodes = new ArrayList<>();

  for (int i = 0; i < nbColumns; i++) {
   ColumnNode n = new ColumnNode(i + "");
   columnNodes.add(n);
   headerNode = (ColumnNode) headerNode.linkRight(n);
  }

  headerNode = headerNode.right.column;

  for (int[] aGrid : grid) {
   DancingNode prev = null;

   for (int j = 0; j < nbColumns; j++) {
    if (aGrid[j] == 1) {
     ColumnNode col = columnNodes.get(j);
     DancingNode newNode = new DancingNode(col);

     if (prev == null)
      prev = newNode;

     col.top.linkDown(newNode);
     prev = prev.linkRight(newNode);
     col.size++;
    }
   }
  }

  headerNode.size = nbColumns;

  return headerNode;
 }

 // ...
}
```

Implémentation de l'algorithme X

Notre instance de grille de Sudoku à résoudre représentée sous la forme d'une matrice de couverture au format liste quadruplement chaînée, nous allons pouvoir implémenter l'algorithme X décrit précédemment en pseudo-code. Ceci nous donne donc la méthode process suivante au sein de la classe DLX :

```
private void process(int k) {
 if (header.right == header) {
  // Fin de l'Algorithme X
  // on copie le résultat dans une liste result
  result = new LinkedList<>(answer);
 } else {
  // sélection de la colonne c
  ColumnNode c = selectColumnNodeHeuristic();
  c.cover();

  for (DancingNode r = c.bottom; r != c; r = r.bottom) {
   // on ajoute la ligne r dans la solution partielle
   answer.add(r);

   // on couvre les colonnes
   for (DancingNode j = r.right; j != r; j = j.right) {
    j.column.cover();
   }

   // appel récursif au niveau k+1
   process(k + 1);

   // on revient en arrière
   r = answer.remove(answer.size() - 1);
   c = r.column;

   // on découvre les colonnes
   for (DancingNode j = r.left; j != r; j = j.left) {
    j.column.uncover();
   }
```

```
  }

  c.uncover();
 }
}
```

Transformation du résultat

Le lancement de la résolution du problème de la couverture exacte se fait via un appel à la méthode process avec en paramètre 0. Une fois la résolution terminée, le sous-ensemble résolvant le problème modélisé est représenté au sein de la liste d'objets DancingNode result. Notre dernier travail consiste donc à transformer cette liste afin d'obtenir la grille de Sudoku résolue équivalente. Pour ce faire, on va parcourir la liste d'objets DancingNode correspondant à notre solution. Pour chaque noeud, on récupère la valeur associée ainsi que la position dans la grille de Sudoku. Il ne nous reste ensuite plus qu'à affecter cette valeur à la position obtenue dans notre grille. Tout ceci est implémenté au sein de la méthode convertDLXListToGrid suivante :

```
private int[][] convertDLXListToGrid(List<DancingNode> answer) {
 int[][] result = new int[SIZE][SIZE];

 for (DancingNode n : answer) {
  DancingNode rcNode = n;
  int min = Integer.parseInt(rcNode.column.name);

  for (DancingNode tmp = n.right; tmp != n; tmp = tmp.right) {
   int val = Integer.parseInt(tmp.column.name);

   if (val < min) {
    min = val;
    rcNode = tmp;
   }
  }

  // on récupère la ligne et la colonne
  int ans1 = Integer.parseInt(rcNode.column.name);
  int ans2 = Integer.parseInt(rcNode.right.column.name);
  int r = ans1 / SIZE;
  int c = ans1 % SIZE;
   // puis la valeur affectée
  int num = (ans2 % SIZE) + 1;
   // on affecte le tout sur notre grille résolue
  result[r][c] = num;
 }

 return result;
}
```

Les Dancing Links en action !

La dernière étape consiste à assembler les différentes parties de notre programme afin de résoudre une grille de Sudoku via l'algorithme X et la technique des Dancing Links. Pour rappel, la méthode solve que nous définissons au sein de la classe Sudoku décompose le travail de la manière suivante :

1 - Conversion de la grille en matrice de couverture via appel de convertInCoverMatrix.

2 - Création de l'objet DLX avec passage en entrée de la matrice de couverture. Au sein du constructeur de l'objet DLX, on appelle la méthode createDLXList permettant de représenter la matrice de couverture sous la forme d'une liste quadruplement chaînée.

3 - Appel de la méthode solve de l'instance d'objet DLX. Cette dernière se chargeant de lancer la résolution via l'algorithme X et l'utilisation des Dancing Links sur la liste quadruplement chaînée.

4 - Récupération puis conversion de la liste des DancingNode, solution du problème de la couverture exacte, en grille de Sudoku résolue via la méthode convertDLXListToGrid de la classe Sudoku.

On obtient ainsi le code suivant pour la méthode solve :

```
public class Sudoku {

 // ...
 private int[][] grid;
 private int[][] gridSolved;
 // ...

 public void solve() {
  int[][] cover = convertInCoverMatrix(grid);
  //printCoverMatrix(cover);
  DLX dlx = new DLX(cover);
  dlx.solve();
  gridSolved = convertDLXListToGrid(dlx.result);
   displaySolution();
 }

// ...
}
```

Exécution sur un Sudoku de taille 25

Tout naturellement, notre programme s'appuyant sur les Dancing Links et l'algorithme X ne fait qu'une bouchée du Sudoku de taille 9 résolu en début d'article par notre premier programme basé sur un BackTracking simpliste. Notre implémentation est en revanche attendue sur des grilles de Sudoku plus complexes. C'est ainsi que nous lui passons en entrée une grille de Sudoku difficile de taille 25. Alors que notre premier algorithme tourne encore au bout de 5 minutes, l'algorithme s'appuyant sur les Dancing Links résout sans difficulté cette grille en tout juste 334 ms !. On peut donc dire, sans prendre trop de risques, que les Dancing Links et l'algorithme X constituent une solution ultra efficace dans le cadre de la résolution du problème du Sudoku.

Conclusion

La réalisation d'un solveur de Sudoku basique est une tâche intéressante pour tout débutant en programmation. Néanmoins, la mise en place d'un algorithme de BackTracking simpliste montre très rapidement ses limites dès lors que les grilles à résoudre deviennent plus complexes et plus grandes. C'est alors qu'entre en action la technique Dancing Links et l'algorithme X. En proposant une transposition du problème du Sudoku en problème de la couverture exacte, cette approche permet d'obtenir d'excellents résultats comme nous avons pu le voir dans cet article alors même qu'on reste sur du BackTracking. Mieux encore, l'implémentation d'un tel solveur éveillera la curiosité des développeurs les plus aguerris, ce qui n'est jamais une mince affaire. •

Patrice Lamarche
Leader Technique
Chausson Matériaux

Mise en place d'un cycle DevOps pour applications mobiles avec Visual Studio App Center

niveau 100

Visual Studio App Center a déjà un historique important et a subi d'importantes évolutions pour arriver à être un produit complet et intégré à destination des applications mobiles. En effet, il est issu de l'union de HockeyApp racheté par Microsoft, de Xamarin Insights et Xamarin Test cloud intégrés suite au rachat de Xamarin, des services Azure App Service et Azure Mobile Engagement, et auparavant nommé Visual Studio Mobile Center…

Comme pour toute bonne solution Microsoft qui se respecte, Visual Studio App Center est un produit ouvert et cross-platform qui permet de gérer le cycle de vie des applications iOS et Android développés via Swift, Objective C, Reactive Native et bien sûr Xamarin.

Solution DevOps pour applications mobiles

Le cycle DevOps traditionnel peut se décomposer au travers des quatre phases suivantes : planification agile, le développement et les tests manuels et automatisés, la mise en place d'un pipeline de déploiement, et, enfin, la phase d'apprentissage. Phase dans laquelle on surveille le comportement de l'application d'un point de vue stabilité et où on analyse l'utilisation de l'application afin d'avoir du feedback afin d'améliorer les différentes fonctionnalités. 1

Mise en place

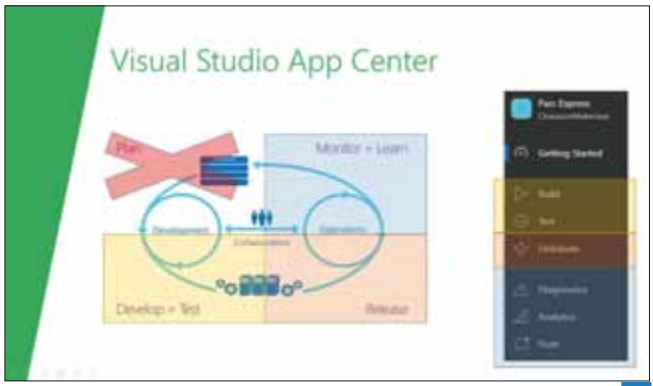
La mise en place et l'intégration de Visual Studio App Center ne peut pas être plus simple et plus rapide, il suffit d'ajouter les packages nuget Microsoft.AppCenter.Analytics et Microsoft.AppCenter.Crashes à votre solution, et d'ajouter une ligne de code que vous pouvez copier-coller pour avoir une base d'analytics et de monitoring en place. Cela est réalisable en 5 minutes chrono, et la mise en place est bien documentée quelle que soit la technologie de développement que vous utilisez.

Tarification

A l'exception des services qui nécessitent du compute tels que les builds et l'exécution de tests automatisés sur des devices réels, tous les services proposés par Visual Studio App Center sont gratuits.
Pour ce prix-là, Microsoft propose un stockage des données pendant une durée de 90 jours. A noter que ce stockage peut être étendu en optant pour un export vers Application Insights permettant des analyses plus fines, la mise en place d'alerte, etc. Autre limite, le nombre d'évènements trackés que nous verrons plus en détail dans la partie Analytics. Vous pouvez tracker 200 évènements différents. Les autres services gratuits n'ont pas d'autres limites notables : nombre illimité d'applications, nombre illimité d'utilisateurs, etc.

Planification Agile

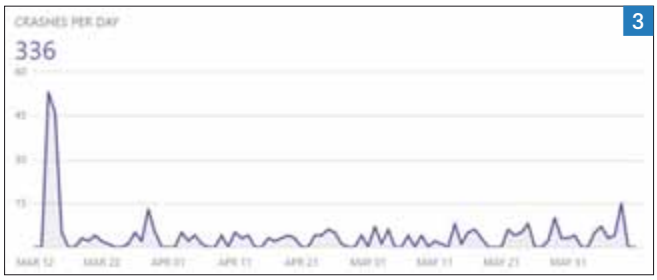
Visual Studio App Center propose des fonctionnalités pour les différentes phases du cycle DevOps à l'exception de la planification agile. La gestion de la planification de projets mobiles n'étant pas différente de la planification d'autres types de projets, vous pouvez utiliser votre outil habituel tel que VSTS/TFS, Jira, GitLab, YouTrack afin de gérer vos projets comme bon vous semble.
Visual Studio App Center n'a donc pas vocation à remplacer votre outil de planification habituel mais plutôt à s'intégrer à ceux-ci afin de créer automatiquement des bugs lorsque des crashes sont récupérés par App Center. Vous pouvez ainsi vous connecter à Visual Studio Team Services, GitHub, et Jira afin d'automatiquement créer un work item en fonction du nombre de crashs détectés. Ainsi si vous souhaitez éviter que votre bug tracker soit « pollué » avec des crashs qui n'ont été détectés que très rarement, vous pouvez définir un seuil à partir duquel le bug sera effectivement créé.



1

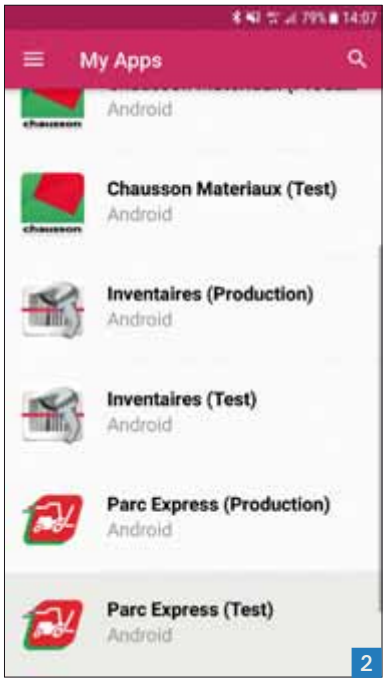
Compilation de vos applications mobiles

La mise en place de process de builds serveur pour des applications mobiles n'est pas aussi simple, par exemple, que pour des applications web. Si vous souhaitez mettre en place des agents de builds privés, vous devrez vous adapter aux spécificités des plateformes Android et iOS. En ce qui concerne iOS, vous devrez par exemple créer des certificats et des profils de provisionning et les installer/paramétrer sur votre ordinateur sous macOS utilisé en tant qu'agent de builds.
Si vous ne souhaitez pas faire l'effort d'installer des agents de Builds pour compiler vos applications Android et iOS, il est possible d'utiliser le service de Builds proposé par Visual Studio App Center. Celui-ci peut se connecter à des repositories hébergés sur Visual Studio Team Services, Github, ou encore Bitbucket. En plus du fait qu'il ne soit donc plus nécessaire de posséder un mac pour créer des packages iOS, App Center propose une fonction d'auto-provisionning qui permet de simplifier le déploiement de vos packages sur l'ensemble des devices souhaités, sans avoir à modifier les profils de provisionning ni enregistrer chaque device manuellement.



3

4



2

Déploiement de vos applications mobiles

Au-delà de la compilation, vous pouvez simplifier le déploiement de vos applications mobiles grâce à App Center. Il est possible d'automatiser la publication de vos applications publiques sur l'App Store d'Apple, et sur Google Play mais également de simplifier le déploiement de vos applications ad-hoc. 2

Après avoir uploadé vos packages sur App Center, les testeurs et utilisateurs peuvent récupérer les applications de deux manières : soit en accédant au portail App Center à vos testeurs, ou alors installer l'application HockeyApp. Disponible en version preview pour iOS et à présent disponible publiquement sur le Play Store pour Android, cette application est un store privé permettant aux utilisateurs d'installer eux-mêmes les versions des applications que vous souhaitez leur permettre de déployer.

Monitoring d'applications mobiles

Comme pour toute application, il est important d'être capable de suivre le bon fonctionnement des applications mobiles, et si les stores publics comme le Play Store ou l'App Store proposent des services permettant de détecter les crashes des applications, Visual Studio App Center peut être utilisé pour tout détecter et récupérer les crashs de vos applications privées et publiques. Pour ce faire, lors d'un crash App Center génère un fichier de reporting de crash en local sur le device, puis envoie ce rapport lors du prochain lancement de l'application s'il n'a pas pu être envoyé avant que l'application ne se termine.

Vous avez ensuite accès à la liste de tous les crashs directement depuis le portail web. Ceci avec le plus important : une stack trace bien détaillée s'il s'agit d'un bug applicatif présent au sein de votre code, ou qui le sera beaucoup moins si le bug est plus proche de l'OS et causé par les joyeusetés des OS mobiles.

En plus de la simple liste, des graphiques vous permettent d'évaluer l'impact de vos résolutions de bugs. 3

En plus du tracking de crashes, App Center vous permet de faire le suivi de vos propres erreurs. Si vous souhaitez tracer la levée de certaines exceptions dans votre code, vous pouvez ainsi, en une ligne de code, enregistrer les exceptions que vous souhaitez grâce à la méthode TrackError de la classe Crashes. Il est également possible d'enregistrer des informations complémentaires à celles de l'exception en lui passant un dictionnaire de string.

Analyser l'utilisation des applications mobiles

Visual Studio App Center vous permet de suivre l'utilisation/l'usage de vos applications mobiles via la consultation des statistiques d'utilisation comme le nombre d'utilisateurs et l'évolution du nombre de sessions, mais il permet surtout de tracker les évènements métiers ou techniques que vous souhaitez suivre.

Ainsi, si vous souhaitez savoir combien d'utilisateurs ont utilisé telle fonctionnalité, voir si une fonctionnalité est plus utilisée suite à une mise à jour ergonomique ou fonctionnelle, vous pouvez tracker vos propres évènement très simplement en une ligne de code. En effet, tout comme le tracking d'erreur, le tracking d'event se fait en une simple ligne de code, en utilisant la méthode TrackEvent de la classe Analytics. Cette méthode prend en argument le nom de l'évènement souhaité ainsi qu'un dictionnaire de string permettant d'indiquer le nom des valeurs que l'on souhaite stocker ainsi que leurs valeurs.

A noter la possibilité d'ajouter des informations complémentaires à vos évènements afin d'identifier l'utilisateur ou le device ou d'ajouter d'autres informations de contexte. 4

Intégration avec Team Foundation Server / Visual Studio Team Services

Vous avez la liberté d'utiliser les fonctionnalités de Visual Studio App Center que vous souhaitez, et de continuer à utiliser TFS ou VSTS pour d'autres besoins vous pouvez tout à fait le faire, sans restriction.

Il est ainsi possible d'utiliser par exemple TFS pour la planification agile, la gestion de sources sous git, la mise en place d'agents de builds privés pour la création de packages Android et iOS, la création de pipelines de releases plus ou moins complexes qui uploadent vos packages sur App Center pour publication sur le store privé, mais également pour publier les applications sur les stores publics.

Des tâches TFS/VSTS sont disponibles à cet effet soit directement au sein de ces services soit gratuitement depuis le marketplace.

Vous pourrez alors utiliser Visual Studio App Center pour le déploiement de vos applications en environnement de tests, ou pour vos applications privées, pour leur monitoring et le tracking d'utilisation en bénéficiant du meilleur des deux mondes.

•

Bacci Michael
michael.bacci.software@gmail.com - https://www.linkedin.com/in/michaelbacci
Senior Software Engineer R&D
Expert en HPC, C/C++, J2EE, Python. Freelance, Michael travaille dans des projets R&D où les enjeux de performance, scalabilité et algorithmique sont particulièrement sensibles.

niveau 100

C++17 : se préparer au changement

C++17 est fait! Le 6 Septembre 2017 le C++17 DIS (Draft International Standard) a formellement approuvé l'intégralité des révisions qui ont été apportées pendant le grand C++ ISO meeting de mars 2017 aux USA. Le comité ISO/IEC a publié le PDF 14882:2017 contenant les descriptions techniques finales le 4 décembre 2017. Dans cet article, on explorera ensemble non seulement les caractéristiques fondamentales du nouveau C++17 (ou C++1z) mais également comment vraiment exploiter ses points forts.

Pourquoi C++17 ?

Parce que l'hardware est en constante évolution et les techniques pour le programmer évoluent avec lui. Parce qu'un langage qui n'évolue pas est condamné à mourir et la communauté C++ défend son bijou. A noter que dans la communauté C++ il n'y a pas seulement des fous de programmation comme moi (et toi j'espère :) mais aussi des grands groupes industriels et financiers qui n'ont aucun intérêt à ce que le C++ disparaisse! Bien au contraire, C++ est un langage qui sait se défendre devant une concurrence de plus en plus compétitive. Voir par exemple Rust, Go, Scala et D.

En donnant des cours avancés de C++11 dans une école d'ingénieur, je me suis rendu compte que les élèves les moins avancés sont plutôt attirés par des langages comme Java ou Python : en effet, pour vraiment apprécier le C++, il faut avoir une connaissance approfondie du fonctionnement du CPU et de la mémoire. Mais si C++11 rend certains concepts comme les pointeurs plus faciles grâce aux références et move-semantic, le C++17 rend la programmation encore plus facile, comme par exemple les itérations à plusieurs variables et le template auto. Selon l'index TIOBE, qui mesure la popularité des langages les plus utilisés au monde, le C++ est en troisième position, après Java (en premier) et le C (qui est en deuxième) ; mon pari est qu'avec le nouveau C++17, le trend du C++ n'est pas prêt de baisser.

Roadmap

Regardons ensemble la roadmap du C++ avec ce schéma officiel du isocpp.org en Fig. 1

Compilateur supporté

Avant d'explorer les détails du C++17, il est important aussi d'aborder le sujet des compilateurs. A l'adresse http://en.cppreference.com/w/cpp/compiler_support vous trouverez des tableaux comme ceux-ci :

C++17 feature	Paper(s)	Version	GCC	Clang	MSVC
New auto rules for direct-list-initialization	N3922	c++17-lang	5.0	3.8	19.0*
static_assert with no message	N3928	c++17-lang	6	2.5	19.1*
typename in a template template parameter	N4051	c++17-lang	5.0	3.5	19.0*
Removing trigraphs	N4086	c++17-lang	5.1	3.5	16.0*
Nested namespace definition	N4230	c++17-lang	6	3.6	19.0*

Ces tableaux sont très utiles car ils donnent, pour chaque nouveauté du langage, la version minimale du compilateur capable d'interpréter la nouvelle fonctionnalité et aussi le lien vers l'historique des révisions, qui permet de comprendre pourquoi la *feature* a été introduite et quels avantages elle apporte.

Comment tester C++17 ?

La façon la plus simple et rapide est d'utiliser Docker.
Comme un coup de baguette magique, exécutez cette commande :

```
$docker run -it gcc::latest
```

Mettez à jour le système avec :

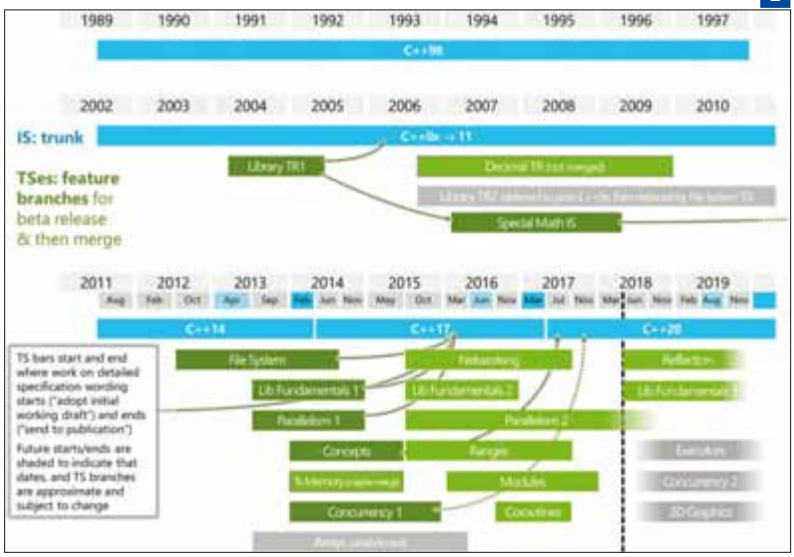
```
$apt-get update
$apt-get upgrade
```

Installez votre editor préféré, vim par exemple :

```
$apt-get install vim
```

Ecrivez votre test et compilez en utilisant l'option -std=c++1z en ligne de commande avec gcc. A noter que toutes les nouvelles fonctionnalités de C++17 ne sont pas forcément disponibles avec la dernière version 'stable' du gcc ou clang. Il vous faudra télécharger et compiler vous-même des versions 'unstable' ou bien 'experimental' dans des branches git maintenues par des geeks :)
Pour des test "simples", des sites comme wandbox.org et

1



godbolt.org vous donnent la possibilité de compiler on-line en spécifiant la version du compilateur et les options de compilation.

Nouveautés dans le langage

Explorons ensemble à l'aide de simples tableaux les principales nouveautés :

Namespace

Des namespace plus compacts, qui rendent aussi l'indentation plus légère :

C++11,C++14	C++17

namespace A {
namespace B {
namespace C {
/*code*/
}
}
}
	namespace A::B::C {
	/*code*/
	}

Fonction Lambda

Des améliorations avaient été demandées de la part de la communauté, qui en fait une grande utilisation depuis C++11.

C++11	C++14,C++17
for_each(begin(vector), end(vector), [](const decltype(vector)::& x) { cout << x; });	for_each(begin(vector), end(vector), [](const auto& x) { cout << x; });
auto my_x = std::unique_ptr<int>(10); auto inc_x = std::bind([](std::unique_ptr<int>& x) { ++(*x); }, std::move(my_x));	auto my_x = std::unique_ptr<int>(10); auto inc_x = [x(std::move(my_x))]() { ++(*x);};

En C++17 on a aussi la possibilité de passer *this comme copie

C++11, C++14	C++17
auto lambda = [this]{};	auto lambda_copy_ref = [this]{}; auto lambda_copy_val = [*this]{};

Constexpr-if

L'expression constexpr-if permet de simplifier les décisions **à compile-time.** La keyword constexpr peut être aussi associée avec l'expression if:

```
if constexpr(cond)
    statement1;
else
    statement2;
```

Un exemple de son efficacité :

C++11, C++14	C++17

template<typename T,
std::enable_if_t<std::is_pointer<T>::value>*
= nullptr>
auto get_real_value(T t) { return *t; }
 	
template<typename T,
std::enable_if_t<!std::is_pointer<T>::value>*
= nullptr>
auto get_real_value(T t) { return t; }
	template<typename T>
	auto get_real_value(T t) {
	if constexpr (std::is_pointer<T>) return *t;
	else return t;
	}

Voici une fonction lambda définie avec constexpr:

```
constexpr auto add = [] (int a, int b) { return a + b};
main () {
 static_assert(add(1,2) == 3); //compile-time
 auto x = add(1,2); //run-time
}
```

La constexpr rend aussi le code plus lisible en le combinant avec le #ifdef

```
#ifdef DEBUG
constexpr bool debug = true;
#else
constexpr bool debug = false;
#endif

/* code */
if constexpr (debug) {
  /* code */
}
```

Structured binding

La déclaration de variables multiples rend le code plus "léger". Dans d'autres langages de programmation, cette possibilité d'assignation multiple est aussi appelée *unpacking*.

C++17	C++17

std::map<std::string, int> map;
 	
//récupération de l'itérateur et du
//boolean du résultat
auto [it, ok] = map.emplace("key", 123);
assert(ok);
	struct Data {
	int a; float b;
	std::string c;
	};
	
	auto data = Data{1, 2.0f, "Reference"};
	//value-reference aux données publiques de data
	auto &[r_a,r_b,r_c] = data;
	++r_a;
	assert(data.a == r_a);
	
	//value-reference pour les valeurs temporaires
	de Data
	auto &&[c_a,c_b,c_c] = Data{1, 2.0f, "Copy"};

auto tuple = std::make_tuple(int,
float>(3, 3.14);
 	
//copie des valeurs de la tuple
auto [x_int, x_float] = tuple;
 	
//value-references des valeurs de la
tuple
auto &[r_int, r_float] = tuple;
++r_int;
assert(std::get<0>(tuple) == r_int);
	int vec[3] = {1,2,3};
	//copie de chaque élément
	auto [c_1, c_2, c_3] = vec;
	++c_1;
	assert(vec[0] != c_1);
	
	//value-reference de chaque élément
	auto &[r_1, r_2, r_3] = vec;
	++r_1;
	assert(vec[0] == r_1);

auto f = []() -> std::tuple<std::string,
int> { return { "hello", 123}; };
auto [v_str, v_int] = f();
	int a = 1, b = 2;
	//crée des lvalue-reference
	const auto& [r_a, r_b] = std::tie(a, b);

Son utilité dans les boucles *for* est imparable :

C++11, C++14	C++17

std::map<int,string> map = ...
for(auto& i: map) {
cout << i->first << ", " << i->second;
}
	std::map<int,string> map = ...
	for(auto& [key, value] : map) {
	cout << key << ", " << value;
	}

AUTO

La keyword auto a aussi des nouvelles règles pour l'initialisation :

C++17 syntaxe acceptée	C++17 syntaxe refusée

auto a = { 1 }; //std::initializer_list<int>
auto b { 1 }; //int b{1}
auto c = {1, 2 , 3}; //std::initializer_list<int>
	auto a {10, 20}; //trop d'éléments
	auto b {10, 20.0}; //types différents

En général, la règle des trois AAA Style: Almost Always Auto pour la déclaration des variables devient une évidence avec C++17:

Ancien C++ declaration style	Modern C++ AAA style

const char* a = "old C++ declaration";
my_object e = get_my_object();
my_datas a{ data1 };
my_object a{ 3, 6, 9 };
my_object* x = new my_object();
unique_ptr<widget> a = make_unique<widget>();
int a = 7;
float a = 7.;
unsigned long a = 7;
std::string a = "7";
chrono::nanoseconds a{ 7 };
	auto a = "modern C++ declaration";
	auto a = get_my_object();
	auto a = my_datas{ data1 };
	auto a = my_object{ 3, 6, 9 };
	auto a = new my_object();
	auto a = make_unique<widget>();
	auto a = 7;
	auto a = 7.f;
	auto a = 7ul;
	auto a = "7"s;
	auto a = 7ns;

A noter qu'il n'est pas toujours possible d'utiliser le AAA style, car auto fonctionne quand les types sont moveable ; voici des exemples qui ne marchent pas :

```
auto my_atomic = atomic<int>{};   // error, le type n'est pas moveable
auto my_lock = lock_guard<mutex>{ my_mutex };  // error, le type n'est pas moveable
```

```
struct NoMove {
 NoMove() = default;
 NoMove(NoMove&&) = delete;
};
NoMove no_mv = NoMove(); //ok
auto  no_mv = NoMove(); //error
```

If and switch

Déclaration de variable dans les clauses if et switch :

```
• if (init; condition) { … }
• switch (init; condition) { … }
```

Exemples :

```
• if (lock_guard<mutex> my_lock(my_mutex); my_vector.empty())
  {         my_vector.push_back(initial_value);
  }
• if (auto shr_ptr(weak_ptr.lock()); shr_ptr) {
   //shr_ptr->
  } else {
   //shr_ptr == nullptr
  }
• if (auto db = get_connection_to(host); db.is_connected()) { db.set_timeout(30s); }
• if (auto const [it, ok] = my_map.insert(key, value); !ok) { throw InsertionFailed(); }
• if (ErrorCode error; get_error(&error)) { cout << error; exit(-1); }
```

INLINE

La déclaration de variable inline a maintenant la même sémantique que la fonction déclarée inline:

C++17	C++17

struct data {
  static const int v;
};
inline int const data::v = 123;
	struct data {
	  inline static const int v = 123;
	};

RVO

Enfin la *Return Value Optimisation* est garantie, ce qui signifie que le compilateur est autorisé à supprimer les opérations de copie non nécessaires!! Dans les versions précédentes la RVO était seulement une option dans le processus d'optimisation du compilateur.

C++17 (RVO)	C++17 (Named RVO)

X get_X() {
 return X(); //pas de copie
}
X data = get_X(); //pas de copie
	X get_X() {
	 X tmp; //le type de return a un nom
	 return tmp; //copie locale
	}
	X data = get_X(); //pas de copie

Template & Smart Deduction: déduction automatique des arguments

Quand il n'y a pas de conflit possible, une simplification de la syntaxe est autorisée pour la déduction des arguments des template :

C++11,C++14	C++17

std::pair<int, double> p(2, 4.5);
std::tuple<int, int, double> t(4, 3, 2.5);
 	
template<class T> struct S { S(T){} };
S<float> s(3.14f);
 	
std::vector<int> v1(1,2);
std::vector<int> v2(v1.begin(),v1.end());
	std::pair p(2, 4.5);
	std::tuple t(4, 3, 2.5);
	
	template<class T> struct S { S(T){} };
	S s(3.14f);
	
	std::vector v1(1,2); //std::vector<int>
	std::vector v2(v1.begin(),v1.end());

Voici comment construire une smart déduction :

```
template <typename T>
struct SmartDeduction {
 T t;
};

SmartDeduction(const char *) -> SmartDeduction<std::string>;
SmartDeduction(int) -> SmartDeduction<double>;

int main() {
 SmartDeduction str("Cool"); //normalement "Cool" est interprété comme un const char *
 assert(typeid(str.t) == typeid(std::string));

 SmartDeduction x{3}; //int(3) est convertie en double
 assert(typeid(x.t) == typeid(double));
}
```

Nouveauté dans la STL Containers

Les containers plus utilisés de Boost - comme variant, optinal et any - sont enfin dans C++17.

std::variant<T,...>

Commençons avec un exemple facile pour voir ensuite le vrai avantage de ce nouveau type :

```
#include <variant>
#include <string>
#include <cassert>

int main() {
 std::variant<std::string, int, float> v;
 v = "cool";
 auto s = std::get<std::string>(v);
 assert("cool" == s);

 v = 123;
 auto i = std::get<1>(v);
 assert(123 == i);

 v = 3.14f;
 auto f = std::get<float>(v); //ou std::get<2>(v)
 assert(3.14f == f);
}
```

Il est facile de comprendre que std::variant est un type qui peut varier selon son assignation. Ajoutons maintenant une opération de *visit* selon chaque type déclaré :

```
struct visit_variant {
 bool operator()(std::string) { /* code pour string */ }
 bool operator()(int) { /* code pour int */ }
 bool operator()(float) { /* code pour float */ }
} constexpr my_visit {};

int main() {
 std::variant<std::string, int, float> v;
 /* … */
 //ici l'operator() - nommé 'call' - est déterminé à compile time!
 std::visit(my_visit, v);

 //ou bien en utilisant les fonctions lambda:
 std::visit([](auto&& val) {
```

```
    using T = std::remove_cv_t<std::remove_reference_t<decltype(val)>>;
    if constexpr (std::is_same_v<T, std::string>)
      /* code pour string */
    else if constexpr( std::is_same_v<T, int> )
      /* code pour int */
    else if constexpr( std::is_same_v<T, float> )
      /* code pour float */
  }, v);
}
```

Dans un cas réel de traitement d'image, le logiciel de déformation 2D/3D www.deformetrica.org sur lequel j'ai travaillé utilise les variantes pour traiter de façon homogène les types scalar, vecteur, matrice, vecteur de vecteur, vecteur de matrice, etc. De cette façon on considère que le type sur lequel on travaille est un concept purement mathématique et on ignore si c'est un scalar ou bien une liste de matrice, et on peut agir sur ces éléments grâce aux concepts de visit et aux overloading des opérateurs de façon indépendante du type de données. Une autre utilisation intéressante de *variant* est la possibilité de remplacer les enum, voici un exemple :

```
struct OsOperation {

 // remplacer une: enum OS { Linux, Windows, OsX }
 struct Linux {};
 struct Windows {};
 struct OsX {};
 using OS = std::variant<Linux, Windows, OsX>;

 // programmer pour chaque OS une opération spécifique de façon homogène
 struct sys_call {
  sys_call(OsOperation& osOperation) : op_(osOperation) {};
  void operator()(const Linux&){ /*...*/ }
  void operator()(const Windows&){ /*...*/ }
  void operator()(const OsX&) { /*...*/ }

 private:
  OsOperation& op_;
 };

 OsOperation(OS&& os) : os_(std::move(os)) {}

 void operator()() { std::visit(sys_call(*this), os_); }

 OS os_;
};
```

std::optional<T>

Définit un type qui n'a pas forcément de valeur attribuée. Voici un premier exemple :

```
struct Contact {
 std::string nom;
 std::string prenom;
 std::optional<std::string> email;
 std::optional<std::string> telephone;
};
```

Tout le monde a un nom et prénom, mais pas une adresse email ni un téléphone (ça existe). Ecrire une structure de données de cette façon a deux utilités :

1 - Au niveau conceptuel : un dev qui lit le code comprend que cette donnée en particulier est optionnelle ;
2 - Au niveau des ressources : le type T de std::optional<T> n'est pas instancié en mémoire jusqu'à ce qu'une valeur lui soit attribuée.

En les utilisant dans la signature d'une function/method, le code devient plus lisible :

Avant C++17	C++17

`//return false si erreur`
`bool try_to_do_something(Y y, X& x) {...}`
 	
`//return nullptr si erreur`
`X* try_to_do_something(Y y) {...}`
	`//la valeur peut ne pas être attribuée`
	`std::optional<X> try_to_do_something(Y y) {...}`
	
	`//et en utilisant la nouvelle syntaxe pour la if:`
	`if (auto X = try_to_do_something(y)) {`
	`  //use X`
	`}`

std::any<T>

Ce type de données n'est rien d'autre qu'un wrapper qui peut contenir n'importe quel type T. Regardons un exemple :

```
using namespace std::literals;
int main() {

 std::any x,y;
 x = "string"s;
 y = "const char*";

 std::cout << "x est un "  << std::any_cast<std::string>(x);

 if(y.type() == typeid(const char*))
  std::cout << "\ny est un const char*";
};
```

Any a aussi le grand avantage de rendre obsolète une vieille astuce dérivant du C : le void * utilisé pour faire des *cast* sur n'importe quel type de données.

std::string_view

Pour une question purement liée aux performances, ce nouveau type permet de ne pas allouer de la mémoire pour une string déjà existante.

Using <std::string>	Using <std::string_view>

`#include <iostream>`
`#include <string>`
 	
`void* operator new(std::size_t n)`
`{`
`  std::cout << "Alloc " << n << " bytes"`
`<< std::endl;`
`  return malloc(n);`
`}`
 	
`void use_string(const std::string& s) {`
`  std::cout << s;`
`}`
 	
`int main(void)`
`{`
`  use_string("test");`
`  return 0;`
`}`
	`#include <iostream>`
	`#include <string_view>`
	
	`void* operator new(std::size_t n)`
	`{`
	`  std::cout << "Alloc " << n << " bytes" <<`
	`std::endl;`
	`  return malloc(n);`
	`}`
	
	`void use_string_view(const std::string_view s) {`
	`  std::cout << s;`
	`}`
	
	`int main(void)`
	`{`
	`  use_string_view("test");`
	`  return 0;`
	`}`

`Output:`
`Alloc 29 bytes`
`test`
	`Output:`
	`test`

La string_view est intéressante au niveau des performances notamment pour les opérations de recherche et d'extraction de sous-string, car on peut y travailler en utilisant seulement les références d'adresses.

Nouveaux algorithmes dans la STL

Très attendus, les algorithmes du pattern *map-reduce*, sont disponibles en version parallèle ou, pour être plus précis, disponibles avec

une *policy d'exécution*. Les trois classes de *policy* disponibles sont :

Policy	Namespace	Description
sequenced_policy	std::execution::seq	Algorithm sequential Un seul thread Pas de partage mémoire
parallel_policy	std::execution::par	Algorithm parallel Plusieur threads Partage mémoire entre les threads
parallel_unsequences_policy	std::execution::par_unseq	Comme parallel_policy La possibilité de vectorisation

Soixante-neuf algorithmes de la standard library ont été modifiés pour accepter ce nouveau concept d'*execution policy*. Voici un exemple:

C++11, C++14	C++17

`#include <vector>`
`#include <numeric>`
`...`
`std::vector<int> v {1, 2, 3, 4, /* ... */};`
`int sum (std::accumulate(v.begin(), v.end(), 0));`
	`#include <vector>`
	`#include <numeric>`
	`#include <execution>`
	`...`
	`std::vector<int> v {1, 2, 3, 4, /* ... */};`
	`int sum (std::reduce(std::execution::par, v.begin(), v.end(), 0));`

Les deux codes produisent bien évidemment les mêmes résultats, mais avec un temps d'exécution bien différent !
Attention: la version qui utilise la std::reduce est <u>en théorie</u> plus performante, car pour activer la version parallèle de l'algorithme, en fonction du système d'exploitation, il y a un coût de création pour le threads, techniquement ça s'appelle *overhead*. Ce coût supplémentaire à payer pour activer le threads, devrait être amorti par la concurrence du job. Or, si le vecteur en question est trop "petit" (par exemple de l'ordre d'une centaine d'éléments, voire mille éléments), la version séquentielle pourrait être bien plus rapide que sa version parallèle. Il en va différemment pour la <u>seule</u> vectorisation! Car pour la vectorisation, on n'a pas forcément besoin de threads et le compilateur peut être capable de créer la version vectorisée plus rapide que sa version séquentielle. Je vous invite à étudier de près les nouveaux algorithmes de map-reduce suivants (car ils aident à résoudre des problématiques très fréquentes dans la programmation) : std::reduce, std::transform_reduce, std::transform_exclusive_scan, std::transform_inclusive_scan, std::inclusive_scan et std::exclusive_scan.

Note

La vectorisation d'un code est la capacité de la part de la CPU d'exploiter au maximum ses registres et ses capacités de calcul. L'acronyme SIMD (Single Instruction Multiple Data) signifie la capacité d'exécuter une instruction CPU (par exemple une addition, multiplication, etc.) sur plusieurs données à la fois, avec la contrainte que ces données soient enregistrées de manière séquentielle, comme un vecteur, d'où le nom Vectorisation.

Le filesystem

On a dû attendre la version C++17 pour avoir enfin une librairie standard pour le filesystem!
Un seul code pourra enfin gérer les différences entre le système Windows, Linux et macOS.
Examinons d'abord le concept de base de toute la librairie, le fs::path, avec l'exemple suivant :

C++17	Output

`namespace fs = std::filesystem;`
`auto path = fs::path("/");`
`path /= "usr";`
`path /= "include";`
`path /= "malloc";`
`path += ".h";`	

`cout << std::boolalpha` `<< "exists() = " << fs::exists(path) << "\n"`	true
`<< "root_name() = " << path.root_name() << "\n"`	root_name() = ""
`<< "root_path() = " << path.root_path() << "\n"`	root_path() = "/"
`<< "parent_path() = " << path.parent_path() << "\n"`	parent_path() = "/usr/include"
`<< "relative_path() = " << path.relative_path() << "\n"`	relative_path() = "usr/include/malloc.h"
`<< "stem() = " << path.stem() << "\n"`	stem() = "malloc"
`<< "filename() = " << path.filename() << "\n"`	filename() = "malloc.h"
`<< "extension() = " << path.extension() << "\n";`	extension() = ".h"
`for (auto& p : path) cout << "[" << p << "] ";`	["/"] ["usr"] ["include"] ["malloc.h"]

Dans un système Windows, si path = fs::path("C:\\Windows\\file.txt"); alors le root_name() = "C:" et le root_path() = "C:\".
Voici des fonctionnalités bien utiles pour vos programmes:

C++17	Output

`fs::path p = fs::path("/usr/include") / "..";`
`cout << "current_path() = " << current_path()`
`<< "\n"`
	current_path() = "/home/sebastien/c++"

`<< "canonical(p) = " << canonical(p) << "\n"`	canonical(p) = "/usr"
`<< "absolute(p) = " << absolute(p) << "\n"`	absolute(p) = "/usr/include/.."
`<< "system_complete(p) = " <<` `system_complete(p);`	system_complete(p) = "/usr/include/.."

Dans un système Windows, si p = fs::path("\Users\Sebastien\file.txt"); alors le system_complete = "C:\Users\Sebastien\file.txt";
Pour explorer le contenu d'un répertoire, rien de plus facile que d'utiliser un iterator :

```
for (const auto& p : fs::directory_iterator(fs::path("/usr/include")))
    std::cout << p << " is_dir = " << fs::is_directory(p) << "; is_file = " <<
fs::is_regular_file(p) << "\n";

"/usr/include/syslog.h" is_dir = false; is_file = true
"/usr/include/stdint.h" is_dir = false; is_file = true
"/usr/include/libdjvu" is_dir = true; is_file = false
...
```

Pour traverser récursivement un répertoire, il y a deux possibilités : utiliser directement l'itérator fs::recursive_directory_iterator() ou bien le faire soi-même et avoir du coup plus de contrôle, en utilisant fs::directory_iterator() dans une fonction récursive ou bien....itérative!

Note avancée : *il existe pour chaque algorithme récursif sa version itérative; l'inverse n'est toujours pas prouvé.*

Pour conclure cette section sur le filesystem, il est intéressant de citer :
- l'enum fs::perms qui définit les règles de droit UNIX 'rwx' pour l'*owner*, le *group* et *other* ;
- fs::file_write_time() qui peut en lecture et écriture, déterminer la dernière modification apportée à un objet fs::path.

Futur du C++

Le WG21, c'est-à-dire le comité standard ISOCPP travaille déjà sur la future version du C++20.
Parmi les fonctionnalités les plus innovantes, on y trouvera :
- Mémoire transactionnelle : c'est une technologie déjà exploitée pour certaines databases relationnelles (comme Oracle), où les opérations qu'on effectue ont la fameuse propriété ACID (Atomicity Consistency, Isolation et Durability)
- Coroutine: très exploitée dans des langages interprétés, une coroutine a la capacité de pouvoir arrêter et redémarrer son exécution à partir de son état.
- Concept: c'est un ensemble d'exigences, qui, appliqué à la définition d'un type de données, d'une classe ou d'une structure, en détermine ses restrictions sur les input et/ou output acceptés.
- Static Reflection : un nouveau support à compile-time pour supporter le principe de la Reflection, c'est à dire, la capacité d'inspecter un type de données inconnues.
- Networking : comme ça a été le cas pour le filesystem, le Networking est encore la grande pièce manquante pour rendre le C++ vraiment complet; C++20 pourrait enfin combler le besoin d'une librairie standard pour travailler sur le net.

Abdessamad Benali,
Ingénieur Concepteur Développeur SQLI Paris

niveau 100

GraphQL : une approche API pas en REST

De nos jours, REST (Representational State Transfer) est le standard le plus utilisé quand il s'agit de mettre en place une API (Application Programming Interface). Néanmoins une nouvelle approche a le vent en poupe : il s'agit de GraphQL, un nouveau format qui est venu pour remédier à certaines limites de REST.

REST, une valeur sûre

Si REST est la référence pour l'élaboration d'une API, c'est grâce à sa simplicité en termes de manipulation des données. Chaque entité exposée dans REST est une ressource qui peut être demandée avec des requêtes HTTP.
Si nous prenons l'exemple d'une API qui fournit des articles, avec REST vous pourriez effectuer les opérations suivantes :

```
GET /api/articles
```

Cette requête HTTP permet d'obtenir la liste des articles disponibles. Vous pourriez aussi récupérer un article en particulier, en ajoutant son identifiant par exemple :

```
GET /api/articles/166
```

D'autres opérations peuvent être effectuées, notamment pour créer, mettre à jour ou supprimer un article. Ces opérations standards sont souvent appelées CRUD (Create, Read, Update, Delete).
Chaque opération effectuée dans REST passe par un EndPoint (point d'entrée), qui n'est rien d'autre que l'URLS de la requête HTTP, données dans les exemples ci-dessus.
Et si par exemple, vous souhaitez récupérer les commentaires associés à un article, il faudra faire les mêmes opérations avec d'autres points d'entrées :

```
GET /api/comments
DELETE /api/comments/42
```

Autant dire, que pour un système d'information, dans lequel beaucoup de données transitent, le nombre des points d'entrées et des ressources à exploiter peuvent vite devenir problématique.
L'autre inconvénient de REST réside dans le fait d'exposer une ressource avec toutes les données qui lui sont associées. Ceci est sans doute inutile car toutes ces informations ne seront pas forcément exploitées.

GraphQL à la rescousse

GraphQL (QL pour Query Language) est un langage de requête pour les API. Il propose une approche totalement différente des API REST. Développé et utilisé en production par Facebook en 2012, il est ouvert au grand public en 2015. Depuis, d'autres sociétés, se sont lancées dans l'aventure.

Contrairement à REST, GraphQL permet de centraliser les requêtes dans un seul point d'entrée, les opérations à effectuer sont passées dans la requête elle-même.
Comme on peut le voir ci-dessous, GraphQL utilise uniquement deux opérations (query et mutation) pour consulter ou mettre à jour les données.

```
GET /api?query={ article(id:"166") { title, content } }
POST /api?mutation={ updateArticle(id: "43", title: "new title") { title, content } }
```

Dans ces deux exemples, l'API GraphQL permet de récupérer un article (operation query), avec uniquement les informations spécifiées dans la requête (title, content). La volumétrie d'informations qui transite est ainsi optimisée.
Si nous revenons à notre exemple des commentaires, GraphQL permet aussi de faire des requêtes imbriquées pour obtenir d'un seul coup l'article et les commentaires qui lui sont associés.
GraphQL est venu résoudre d'autre lacunes de REST, et notamment :

- Un typage fort des données, c'est à dire que vous pouvez savoir avec exactitude la nature des paramètres en entrée et des données en sortie (entier, texte…).
- Une documentation automatique qui reflète la structure des données qu'il est possible d'obtenir.

Mais tout n'est pas encore parfait, car contrairement à REST, l'un des principaux inconvénients de GraphQL est la gestion du cache. Comme les requêtes sont spécifiques, il est difficile de mutualiser le cache et de servir la même réponse.

En résumé

GraphQL n'a pas vocation à concurrencer REST, il suffit de voir l'exemple de GitHub qui l'utilise en supplément de son API REST. Il vient comme une alternative intéressante pour mieux cibler sa demande et alléger la volumétrie d'informations qui peut transiter entre un système client/serveur.
Pour montrer sa popularité grandissante, la plupart des technos et des frameworks du marché propose une implémentation de GraphQL. Mais REST a encore un bon avenir, car sa simplicité et sa mutualisation de cache en font une valeur sûre. •

Charles DE VANDIERE

INFINITE SQUARE

niveau 100

GraphQL : exposez vos données dynamiquement dans une API REST

Partie 1

Il est parfois nécessaire de développer une API permettant de retourner uniquement les champs demandés par l'appelant de façon complètement dynamique.

Si par exemple, si j'ai l'entité suivante représentant une personne modélisée de façon théorique :

```
{
 id
 firstName
 lastName
 age
}
```

L'API doit permettre à son client de lui demander de ne retourner qu'id et age, ou bien encore qu'id, firstName et lastName. Il serait aussi intéressant dans notre contexte de pouvoir demander à l'API les entités liées à celles requêtées. On pourrait ainsi demander à obtenir dans la même requête la mère de famille (une personne) et tous ses enfants (d'autres entités « personnes »).
La volonté derrière ce besoin est de pouvoir répondre à plusieurs problématiques :

- Réduire la volumétrie des requêtes en ne demandant que le strict nécessaire ;
- Réduire le nombre de requêtes (la requête étant sur mesure et pouvant demander les entités liées, cela évite au client de faire plusieurs requêtes pour obtenir ce qu'il veut) ;

Après plusieurs recherches, j'ai découvert GraphQL dont je vais vous présenter ma compréhension dans cet article.

GraphQL, qu'est-ce que c'est ?

GraphQL est un langage de requêtage développé par Facebook. Son objectif est de permettre au client de demander uniquement les champs qu'il souhaite lors d'un appel d'API. De plus, il fournit un système simple pour ajouter des paramètres à la requête.
GraphQL ne fait pas uniquement des requêtes GET, il fournit également un système de mutation pour créer ou modifier des entités en ne précisant là aussi que les champs que l'on souhaite. GraphQL n'est pas lié à HTTP, mais cela reste la méthode la plus classique pour l'exposer. Il n'est pas non plus lié à une base de données spécifique, il peut être branché sur n'importe quelle source de données.

Les Queries – demander des informations

Les queries permettent d'effectuer des lectures de données et sont exprimées dans un pseudo langage (nommé de façon très étrange GraphQL ;)) ayant une syntaxe et une grammaire qui lui sont propres. Dans celui-ci les champs demandés sont fournis dans un format propre sous la forme d'objets.

Voici un exemple d'une query permettant de demander une liste d'humains avec uniquement ces propriétés de renseignées : identifiant, nom et taille. Ce sont bien sûr des propriétés spécifiques à chaque modèle. Le lecteur attentif remarquera aussi qu'ici les sauts de lignes sont des caractères importants permettant de séparer les propriétés demandées.

```
{
 humans {
  id
  name
  height
 }
}
```

La réponse que pourrait nous faire GraphQL serait par exemple de ce type :

```
{
 "data": {
  "humans": [
   {
    "id": "1",
    "name": "Luke Skywalker",
    "height": 1.72
   },
   {
    "id": "2",
    "name": "Obiwan Kenobi",
    "height": 1.75
   }
  ]
 }
}
```

Il s'agit bien d'un objet sérialisé au format JSON contenant un nœud « data » dans lequel est envoyé le résultat de la requête. Celui-ci est présent sur une propriété correspondant au nom initialement demandé (ici « humans »).
Son contenu est ici sous la forme d'un tableau car cette query est de type liste. Chaque élément contient exactement et uniquement les champs demandés dans la requête initiale.

Les arguments

Pour pouvoir cibler de façon plus précise les données à retourner, il est nécessaire de spécifier des arguments et cela est bien sûr pos-

sible avec GraphQL. Les arguments se placent dans des parenthèses suivant le type de base demandé et peuvent être de tout type (entier, string, etc.), chacun séparé par une virgule. Attention cependant, de manière peut être inhabituelle pour un développeur .NET, il faut penser à indiquer le nom de l'argument avant de fournir sa valeur. Imaginons par exemple que nous souhaitons obtenir plus d'information sur le célèbre Luke Skywalker (toujours la tête dans le Cloud !!) à qui on aurait donné l'identifiant « 1 » de manière complètement aléatoire, nous pouvons faire la requête suivante en fournissant l'argument ID :

```
{
 human(id: 1) {
  name
  height
  planet
  friends {
   name
  }
 }
}
```

Voici le résultat que pourrait nous renvoyer notre API GraphQL :

```
{
 "data": {
  "human": [
   {
    "name": "Luke Skywalker",
    "height": 1.72,
    "planet": "Tatooine",
    "friends": [
     {
      "name": "R2-D2",
     },
     {
      "name": "Hann Solo",
     }
    ]
   }
  ]
 }
}
```

Plusieurs requêtes en même temps

Il est possible de faire plusieurs requêtes en un seul appel à l'API GraphQL. Cela permet encore une fois de gagner en performance et en latence en partant du principe qu'effectuer un appel ciblant exactement ce que l'on cherche est plus performant que de faire une multitude de petits appels mal calibrés. Il suffit de mettre à la suite de la première demande une autre requête dans notre GraphQL.

Si par exemple je souhaite obtenir en plus d'un jedi ayant l'identifiant « 1 », un sith ayant pour identifiant « 5 », il me suffit d'ajouter la requête correspondante à la suite. On utilise des alias (ici : "jedi" et "sith") pour différencier les différentes queries.

```
{
jedi: human (id: 1) {
        name
}
       sith: human (id: 5) {
      name
       }
}
```

Voici le résultat que pourrait nous renvoyer notre API GraphQL avec, dans le nœud data, les résultats de nos deux sous-requêtes :

```
{
  "data": {
    "jedi": {
      "name": "Luke Skywalker"
    },
    "sith": {
      "name": "Dark Vador"
    }
  }
}
```

Les mutations – modifier les données

Une mutation est l'opération GraphQL permettant de créer ou modifier une entité déjà existante (rien à voir avec les X Men donc).

Une mutation est modélisée en GraphQL en utilisant le mot-clef « mutation » suivi du type de paramètre fournissant les données d'entrées entre parenthèse. Le corps de la mutation est alors écrit entre accolades. On spécifie de même les propriétés retournées par la mutation. Prenons comme exemple le cas d'une mutation « createHuman » permettant la création d'un être humain qui serait modélisé de la manière suivante :

```
mutation ($human: HumanInput!) {
 createHuman(human: $human) {
  id
  name
 }
}
```

Dans cet exemple nous avons :

- $human est la variable envoyée en paramètre pour fournir les informations sur l'humain à créer ;
- HumanInput! est le type de la variable (l'apostrophe « ! » signifie que le paramètre human est obligatoire) ;

createHuman est le nom de la mutation ;

- { id name } sont les champs de l'objet human que l'API doit retourner après la création (cela fonctionne donc exactement comme les queries ici) ;

La variable à passer en paramètre (HumanInput) aura quant à elle cette définition :

```
{
 "human": {
  "name": "Boba Fett"
 }
}
```

Voici le résultat que pourrait nous renvoyer l'API GraphQL sur cette création. On remarque que le nom de la mutation est utilisé dans l'objet JSON pour stocker le résultat :

```
{
 "data": {
  "createHuman": {
   "id": 5,
   "name": "Boba Fett"
  }
 }
}
```

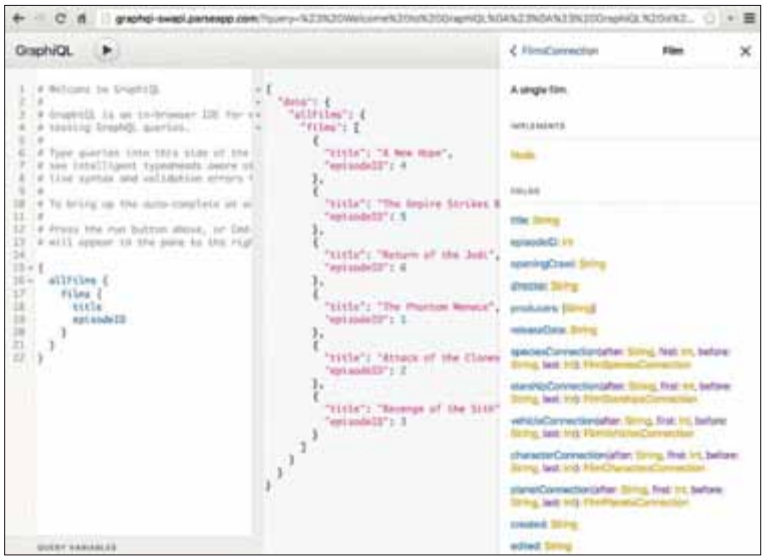
Gestion des erreurs de syntaxe

L'une des grandes forces de GraphQL est sa gestion des erreurs de syntaxe. Si vous faites une erreur de syntaxe dans votre query ou votre mutation, GraphQL va être capable de vous indiquer l'endroit exact de votre erreur, son type et une piste pour la corriger. C'est un confort de développement qui permet de gagner beaucoup de temps pour, par exemple, lire des articles de blogs Infinite Square.
Imaginons que nous écrivions une query GraphQL de manière complétement erronée :

```
{
 hero
}
```

Alors dans ce cas, le moteur GraphQL va nous retourner ce type de message précis indiquant que l'on a oublié de préciser les propriétés que l'on souhaitait avoir en retour de notre appel :

```
{
 "errors": [
  {
   "message": "Field \"hero\" of type \"Character\" must have a selection of subfields. Did you mean \"hero { ... }\"?",
   "locations": [
    {
     "line": 3,
     "column": 3
    }
   ]
  }
```




```
 ]
}
```

GraphiQL

Avant d'aller plus loin, il est important d'introduire un utilitaire des plus pratiques : GraphiQL.Il s'agit de l'outil ultime pour consommer une API GraphQL.
Il s'agit d'une application web (installée en local via les outils NPM et consorts) pour écrire ses queries et mutations de façon plus agréable. En effet, GraphiQL est en mesure de faire de l'introspection pour récupérer la documentation de votre API et propose de l'auto-complétion sur les champs.
Il est très simple à installer, donc ne vous en privez pas en suivant les instructions disponibles sur le GitHub de ce projet open-source : https://github.com/graphql-dotnet/graphql-dotnet. Basé sur .Net core il est possible de l'installer sous Windows, Mac ou Linux.

A venir, dans la partie 2, nous ferons une implémentation dans ASP.Net Core

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

Programmer avec TypeScript

Partie 3

Dans les deux premières parties de ce dossier nous avons vu plusieurs notions importantes à connaître pour développer en TypeScript. Pour conclure, nous allons maintenant aborder trois concepts avancés qui peuvent être un peu intimidants de prime abord, mais il est nécessaire de les connaître en 2018 lorsque l'on démarre un projet avec TypeScript ou JavaScript.

Modules

Le concept de module remonte déjà à plusieurs années. En 2009, la sortie du projet CommonJS a permis d'importer des fichiers JavaScript sans utiliser de balise script. Le but évident était de se dispenser d'avoir une page HTML. L'écriture d'applications serveur en JavaScript a été l'un des principaux cas d'utilisation de CommonJS. Node.js est l'un des premiers projets à avoir adopté le concept de module avec CommonJS.

Concrètement un module c'est quoi ? Pour faire simple, c'est un fichier JavaScript qui, une fois chargé, dispose de son propre scope. Le contenu du module est privé par défaut. Dans la première partie du dossier, lorsque l'on a abordé la notion de portée, j'ai expliqué qu'il était possible de rendre une classe publique. En réalité ce niveau de portée est lié à la notion de module et il faut exporter les classes, fonctions, variables pour qu'elles soient accessibles publiquement lorsque le module sera chargé.

CommonJS a été un précurseur sur ce sujet, et par la suite, d'autres formats de modules ont été publiés. Si l'on n'est pas familier avec cette notion c'est assez complexe de comprendre pourquoi il y a autant de choix pour résoudre une seule problématique. Avec TypeScript tout est beaucoup plus simple, le langage suit les dernières versions d'ECMAScript, et lors de la sortie d'ES2015, les modules ont été standardisés. Côté TypeScript, il suffit donc d'utiliser la syntaxe des modules ES2015. Ensuite, pour transpiler le code dans un format de module en particulier, il suffit de modifier la propriété « module » dans le fichier « tsconfig.json » :

```
"module": "commonjs"
```

TypeScript est donc capable de s'adapter aux différents types de modules. Voici la liste des principaux formats disponibles :

- CommonJS : format de module « legacy » qui est utilisé par Node.js. Le chargement des modules est synchrone ;
- AMD : format de module permettant un chargement asynchrone, principalement utilisé dans les navigateurs. RequireJS est une bibliothèque très populaire implémentant AMD ;
- UMD : format de module universel ;
- System : système de module dynamique capable de charger n'importe quel format ;
- E22015 : format de module standardisé par ECMA et implémenté dans tous les navigateurs récents. La norme est aussi disponible dans Node.js via l'utilisation de fichiers ayant pour extension « .mjs ».

Pour exposer publiquement du code contenu dans un module, il suffit d'utiliser le mot clé « export ». Il est ensuite possible d'utiliser le code en question via le mot clé « import ». Prenons l'exemple d'une fonction contenue dans un premier fichier « hello.ts » et qui est exportée :

```
export function hello(name: string) {
  console.log(`Hello ${ name }!`);
}
```

Il suffit d'importer la fonction depuis le module « hello » afin de l'utiliser :

```
import { hello } from "./hello";

hello("Sylvain");//Hello Sylvain!
```

Comme vous pouvez le remarquer, il n'est pas nécessaire de préciser l'extension du fichier. Si l'on change le format des modules, la compilation donne des résultats qui sont adaptés en conséquence (dans le code ci-dessous, la cible est ES5) :

```
//commonjs export
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
function hello(name) {
  console.log("Hello " + name + "!");
}
exports.hello = hello;
```

```
//commonjs import
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
var hello_1 = require("./hello");
hello_1.hello("Sylvain"); //Hello Sylvain!
```

```
//AMD export
define(["require", "exports"], function (require, exports) {
  "use strict";
  Object.defineProperty(exports, "__esModule", { value: true });
  function hello(name) {
    console.log("Hello " + name + "!");
  }
  exports.hello = hello;
});
```

```
//AMD import
define(["require", "exports", "./hello"], function (require, exports, hello_1) {
  "use strict";
  Object.defineProperty(exports, "__esModule", { value: true });
  hello_1.hello("Sylvain"); //Hello Sylvain!
});
```

Jusqu'ici la notion de module avec TypeScript est simple à mettre en œuvre. Toutefois, il est important de préciser que le compilateur TypeScript transpile le code pour respecter un format de module, mais il ne fournit pas le code permettant de les charger. Avec Node.js cela n'a pas d'impact, car CommonJS est disponible dans le runtime, mais pour les navigateurs c'est différent. Mis à part pour le format ES2015 (dans le cas d'une utilisation avec un navigateur récent), il est nécessaire de mettre en place le chargement des modules avec une bibliothèque respectant le format défini dans la configuration du compilateur TypeScript. Autre point qui peut avoir son importance côté navigateur : les performances. Un module étant un fichier il faut le charger, cela correspond donc à une requête HTTP pour le récupérer. Il est donc préférable d'utiliser un « bundler » afin de packager les modules pour éviter la multiplication des requêtes récupérant les fichiers JavaScript correspondants. Il existe de nombreux outils capables de gérer cette partie, on peut citer notamment Webpack, Rollup ou encore Parcel. Le sujet sort légèrement du cadre de ce dossier, mais il me semble important de vous partager cette information.

Dans l'exemple précédent le mot clé « export » est directement mis sur une fonction, mais il existe d'autres façons d'exporter. Les fonctions, classes ou variables, peuvent être encapsulées dans un objet qui ensuite sera exporté. Cette syntaxe revient au même que celle vue précédemment et la fonction sera importée de la même façon :

```
function hello(name: string) {
  console.log(`Hello ${ name }!`);
};

export { hello };
```

Une fonction peut aussi être exportée sans être encapsulée dans un objet :

```
function hello(name: string) {
  console.log(`Hello ${ name }!`);
};

export default hello;
```

Dans ce cas précis il est obligatoire de préciser que l'export est celui par défaut. Pourquoi ? Tout simplement parce que l'import de la fonction se fera sous la forme d'un alias qui correspondra à l'export par défaut :

```
import h from "./hello";//h is an alias here

h("Sylvain");//Hello Sylvain!
```

Bien évidemment il est possible de faire plusieurs exports, mais une seule fonction peut être définie en tant qu'export par défaut :

```
const hello = (person: Person) => {
  console.log(`Hello ${ person.name }!`);
};

export default hello;

export class Person {
  constructor(public readonly name: string) {
  }
};
```

Côté import, il suffit de mixer la syntaxe afin d'importer la fonction par défaut ainsi que tous les exports qui ont été encapsulés dans un objet :

```
import h, { Person } from "./hello";

h(new Person("Sylvain"));//Hello Sylvain!
```

Un des intérêts du mot clé « defaut » est qu'il permet d'exporter une fonction anonyme :

```
export default (person: Person) => {
  console.log(`Hello ${ person.name }!`);
};
```

Le développeur est ensuite libre de la nommer comme bon lui semble lors de l'importation :

```
import myImportName, { Person } from "./hello";

myImportName(new Person("Sylvain"));//Hello Sylvain!
```

L'ensemble d'un module peut être importé sous la forme d'un alias. Celui-ci fait alors office d'espace de noms afin d'accéder à l'ensemble des éléments qui ont été exportés dans le module :

```
import * as hello from "./hello";

hello.default(new hello.Person("Sylvain"));//Hello Sylvain!
```

Parfois un module n'a pas d'export, mais il exécute de la logique lorsqu'il est chargé. Dans ce cas précis on dit que le module peut être importé pour déclencher un « side effect » (effet de bord). Ce type d'import a une syntaxe particulière. Prenons un exemple :

```
function hello() {
  console.log("Hello Sylvain!");
}

hello();
```

Pour déclencher l'exécution de ce module, il suffit de l'importer directement via la syntaxe suivante :

```
import "./hello";// Hello Sylvain!
```

Il existe d'autres capacités qui sont propres au module, les exemples que nous venons de voir correspondent à la majeure partie des scénarios actuellement utilisés par les développeurs.

Asynchronisme

L'asynchronisme en JavaScript peut être géré par des fonctions de rappel (callback). Malheureusement l'utilisation importante de cette technique dans un projet peut complexifier la lecture du code et sa maintenabilité. Rapidement, le concept de Promise (promesse en français) est apparu pour gérer l'asynchronisme de manière plus élégante. Les Promises sont des tâches asynchrones qui peuvent avoir différents états :

- Pending : l'opération asynchrone n'a pas encore retourné de résultat.
- Fulfilled : l'opération asynchrone est terminée et un résultat est disponible.
- Rejected : l'opération asynchrone a échoué.
- Settled : l'opération asynchrone n'est pas en attente, mais elle est bloquée. Cet état particulier peut se produire lorsque l'on déclenche une succession de tâches liées entre elles.

Pour prendre en charge le retour d'une Promise, il faut utiliser la méthode « then ». Elle permet d'interagir avec le résultat issu d'un état « Fulfilled » ou « Rejected ». Détail important, « then » renvoie toujours une Promise afin de pouvoir chaîner les appels. Lors de l'instanciation d'une Promise, il est nécessaire de définir une fonction qui va être exécutée de manière asynchrone. Cette fonction en accepte deux autres en paramètres qui permettent de gérer l'état de la tâche asynchrone : « resolve » (pour l'état Fulfilled) et « reject » (pour l'état Rejected). TypeScript prend en charge les Promises à condition d'avoir ajouté « es2015 » dans la liste des bibliothèques à inclure lors de la compilation. Dans le fichier « tsconfig.json », cela correspond à la propriété « lib » :

```
"lib": ["es2015"]
```

Voici un exemple d'utilisation d'une Promise en Node.js qui permet de récupérer la configuration du compilateur TypeScript :

```
import { readFile } from "fs";

const promise: Promise<string> = new Promise<string>((resolve, reject) => {
  readFile("./tsconfig.json", "utf8", (error, data) => {
    if(error)
      reject(error);

    resolve(data);
  });
});
```

A présent, il suffit d'utiliser la méthode « then » pour interagir avec le résultat de la promesse :

```
promise.then((data) => console.log(data), (error) => console.log(error));
```

L'exemple présenté ci-dessus est relativement basique puisqu'il sert uniquement à prendre en charge la callback de la fonction « readFile ». Dans ce cas précis, il est possible avec Node.js de convertir une fonction ayant une callback en Promise :

```
import { readFile } from "fs";
import { promisify } from "util";

const readFilePromise = promisify(readFile);

readFilePromise("./tsconfig.json", "utf8")
  .then((value) => console.log(value), (err) => console.log(err));
```

Les Promises sont aujourd'hui disponibles nativement dans tous les navigateurs récents ainsi que dans Node.js. Il ne faut donc pas hésiter à les utiliser !

Maintenant que nous avons vu le concept de Promise, parlons des mots clés « async » et « await ». Initialement introduit dans C#, le concept a ensuite été porté dans le moteur JavaScript Chakra. Microsoft a proposé le concept d' « async/await » à ECMA pour en faire un standard. Concrètement, les promesses peuvent être résolues via le mot clé « await » sans avoir besoin d'utiliser la méthode « then ». « await » ne peut être utilisé que dans une fonction asynchrone. Ce type de fonction a pour retour une Promise afin d'être manipulable ensuite par un autre « await ».

En reprenant le code précèdent, voici ce que cela donne :

```
import { readFile } from "fs";
import { promisify } from "util";

const readFilePromise = promisify(readFile);

async function readTypeScriptConfiguration(): Promise<void> {
  try {
    const value = await readFilePromise("./tsconfig.json", "utf8");
    console.log(value)
  } catch(err) {
    console.log(err)
  }
}
```

La récupération de l'état « Fulfilled » se fait dans un « try », tandis que le « catch » permet de récupérer une erreur ayant déclenché l'état « Rejected ». La fonction « readTypeScriptConfiguration » peut être appelée par n'importe quelle autre fonction asynchrone. Si besoin, elle peut être déclenchée par une fonction asynchrone anonyme auto-exécutée :

```
(async () => {
  await readTypeScriptConfiguration();
})();
```

Cette capacité dans JavaScript/TypeScript est vraiment très intéressante, car elle permet d'écrire du code asynchrone de manière synchrone. Toutefois, il faut faire attention lors de son utilisation, car il est nécessaire d'encapsuler le code dans des « try/catch ». Cela peut entraîner des problèmes d'imbrications similaires à ceux des fonctions de rappel (le fameux « callback hell », que l'on retrouve parfois en Node.js). Autre point important, le code produit par le compilateur TypeScript peut être assez conséquent lorsque l'on cible des version d'ECMAScript inférieures à ES2017 (cette version définissant « async/await » dans JavaScript).

Décorateurs

Les décorateurs existent depuis la sortie de TypeScript 2.0. Ils sont la raison pour laquelle Microsoft et Google ont travaillé de concert sur le couple Angular/TypeScript. Le concept correspond à celui des annotations. Les décorateurs peuvent être appliqués sur une classe, une méthode, une propriété ou un paramètre. Concrètement un décorateur correspond à une fonction qui va permettre d'ajouter un comportement. Celui-ci sera exécuté lors de la manipulation de l'élément qui a été annoté avec le décorateur. C'est une fonctionnalité extrêmement puissante, mais elle est considérée comme expérimentale dans TypeScript. Elle est actuellement à l'étude au TC39 (https://github.com/tc39) pour en fait un standard dans une prochaine version d'ECMAScript.

Pour utiliser les décorateurs avec TypeScript, il faut activer la propriété « experimentalDecorators » dans le fichier « tsconfig.json » :

```
"experimentalDecorators": true
```

Pour annoter une propriété ou un paramètre c'est un peu plus complexe. La nature de JavaScript ne permet pas encore de récupérer des informations propres à ce type d'élément. La conséquence directe de cette limitation est qu'il n'est pas possible d'utiliser les décorateurs dans ce cas sans une bibliothèque tierce. TypeScript va donc transpiler le code pour qu'il soit utilisable avec le package « reflect-metadata ». Il est donc nécessaire de l'installer dans votre projet pour que les décorateurs fonctionnent dans ce cas d'utilisation. De plus, il est impératif d'activer la génération des métadonnées via la propriété « emitDecoratorMetadata » dans le fichier « tsconfig.json » :

```
"emitDecoratorMetadata": true
```

Un décorateur correspond à une fonction qui va être appelée lors de la manipulation de l'élément sur lequel il est attaché. Dans TypeScript il existe des interfaces permettant de nous aider à les écrire. Prenons l'exemple d'un décorateur de classe qui affiche « Hello! » à chaque fois qu'une instance de la classe est créée :

```
const hello: ClassDecorator = (target: Function) => {
  console.log("Hello!");
}

@hello
class Person {

}

const person = new Person();//Hello!
```

Un décorateur a parfois besoin d'un élément de contexte. Il est donc possible d'écrire des décorateurs ayant des paramètres. Il faut alors définir une première fonction qui va accepter des paramètres et renvoyer une nouvelle fonction correspondant à l'interface définissant le type du décorateur :

```
const hello = (name: string) => {
  return (target: Function) => {
    console.log(`Hello ${name}!`);
  }
}


@hello("Sylvain")
class Class1 {

}

@hello("Bill")
class Class2 {

}

const class1 = new Class1()//Hello Sylvain!
const class2 = new Class2()//Hello Bill!
```

Certains types de décorateurs permettent de créer des scénarios particulièrement puissants. Voici un exemple d'un décorateur qui, une fois appliqué sur une méthode, permet de tracer les paramètres qui vont être passés lors de son appel :

```
function logMethodInfo(target, key, descriptor) {
  const calledFunction = descriptor.value;
  descriptor.value = function () {
    const params = Array.from(arguments);
    const message = `Method ${key} called ${
      params.length > 0
        ? `with: ${params.join(';')}`
        : `without parameters`
    }`;
    console.log(message);
    return calledFunction.apply(this, params);
  };
  return descriptor;
}

class Person {
  @logMethodInfo
  setName(name: string): void {}

  @logMethodInfo
  setRoles(...roles: string[]): void { }
}
const person: Person = new Person();
person.setName("Bill"); //Method setName called with: Bill
person.setRoles("Founder", "Technology Advisor"); //Method setRoles called with:
Founder;Technology Advisor
person.setRoles(); //Method setRoles called without parameters
```

Les possibilités offertes par les décorateurs sont vraiment impressionnantes et Angular est un parfait exemple d'une mise en œuvre avancée !

Conclusion

Ces trois concepts sont relativement complexes à comprendre pour un débutant et il faut un peu de temps pour se les approprier. Il existe encore beaucoup de capacités dans TypeScript que nous n'avons pas pu aborder dans ce dossier. Résumer un langage de ce type en une douzaine de pages n'est pas possible bien évidemment, mais peut-être que ce dossier vous aura donné envie d'essayer ce magnifique langage. C'est un projet important chez Microsoft qui est porté par des ingénieurs talentueux, eux-mêmes épaulés par une communauté de développeurs passionnés. TypeScript est en pleine montée en puissant et de nombreux projets prometteurs l'utilisent pour pousser le monde du web encore plus loin !

Je profite de cette conclusion pour remercier Christophe Pichaud, Félix Billon, Charly Poly, Sébastien Bovo, Christopher Maneu et Stéphane Mestre pour leurs relectures sur les différentes parties de ce dossier.

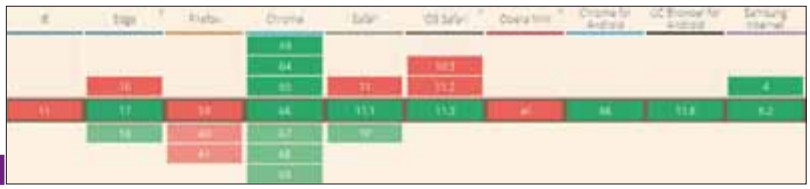
•

Guillaume ANDRÉ
CTO Wexperience & UX.Care
UX.Care est une solution en ligne dédiée à la mesure et à la compréhension de l'expérience utilisateur sur votre site web ou mobile.

Progressive Web App (PWA)
Le fichier Web App Manifest

Partie 2

Sans aucun doute la première porte d'entrée pour tout développeur web qui s'intéresse au domaine des Progressive Web Apps (PWAs), la fonctionnalité Web App Manifest permet à tout un chacun de transformer son site internet, son blog, son site e-commerce, ou sa single-page application (SPA) en Progressive Web App. C'est le fichier Web App Manifest qui va guider le navigateur en lui spécifiant les points d'entrées et les points clés de la PWA. L'une des fonctionnalités phares et par défaut liée à l'usage des Web App Manifest est la capacité de rendre un site "installable" sur l'écran d'accueil du smartphone de l'utilisateur et ce, au même niveau qu'une application native. On parle ici de la fonctionnalité A2HS pour Add 2 Home Screen aussi appelée Web App Install Banners côté Android.



9

Mettre à jour un Web App Manifest

Une fois qu'un utilisateur a installé sur son écran d'accueil une PWA, la question de la mise à jour du Manifest devient vite un enjeu important. D'après la spécification W3C :

Il n'existe à ce jour aucune possibilité de forcer une mise à jour du Web App Manifest de manière automatique.

L'utilisateur est alors obligé de ré-ajouter la PWA à son écran d'accueil pour profiter des éventuelles mises à jour... Pour pallier ce manque, une discussion officielle est ouverte dans le but de créer une nouvelle fonctionnalité qui permettrait de rendre explicite une mise à jour de la PWA vis-à-vis de l'utilisateur : https://github.com/w3c/manifest/issues/446

Il y a fort à parier qu'une solution sera implémentée avant la sortie de la première version de la spécification.

Support et compatibilité des Web App Manifest

Les plateformes d'information dites "platform status" des différents acteurs du web permettent de recueillir l'état d'avancement de l'implémentation des spécifications. Bien que les Web App Manifests soient encore au stade de Draft W3C, certains navigateurs sont pourtant très avancés dans leur implémentation (Mai 2018).

Chrome : *OPÉRATIONNEL* (https://www.chromestatus.com/feature/6488656873259008)

Edge : *OPÉRATIONNEL* (https://developer.microsoft.com/en-us/microsoft-edge/platform/status/webapplicationmanifest/)

Firefox : *EN DEVELOPPEMENT* (https://platform-status.mozilla.org/#app-manifest)

Safari : *EN DEVELOPPEMENT* (https://webkit.org/status/#specification-web-app-manifest)

Certaines écuries n'hésitent pas à pousser des versions intermédiaires ou basées sur des standards propriétaires afin de se mettre à niveau et éviter de perdre la course de l'adoption auprès des développeurs. Pour autant, le marché des Progressive Web Apps (PWAs) étant porteur, les différents acteurs semblent itérer assez rapidement sur ces fonctionnalités dans le but de respecter les standards. En effet, tout comme les dernières versions des principaux navigateurs, l'ensemble des protagonistes (qui pour la plupart participent à l'élaboration du Draft) sont conscients que l'avenir des PWAs passera par les standards et notamment par le W3C dans l'optique d'amener une approche de développement globale et cross-platform. 9

Can I Use expose le support du Web App Manifest par les navigateurs (Mai 2018)

Source : https://caniuse.com/#search=web%20app%20manifest

Faits marquants à propos des Web App Manifest

- Février 2018 l'équipe de développement Edge annonce travailler sur les Web App Manifest (Source : https://blogs.windows.com/msedgedev/2018/02/06/welcoming-progressive-web-apps-edge-windows-10/) :

Over the coming months, we're laser focused on polishing our initial implementation of the core technologies behind PWAs in EdgeHTML and the Universal Windows Platform—Service Worker, Push, Web App Manifest, and especially Fetch are foundational technologies which have a potentially dramatic impact to compatibility and reliability of existing sites and apps, so real-world testing with our Insider population is paramount.

- La version iOS 11.3, arrivée en Mars 2018, vient apporter un meilleur support des Web App Manifest au sein de iOS, mais aussi et principalement sur les Services Workers. Cependant, toujours pas de Web App Install Banners à l'horizon, fonctionnalité qui participerait à la démocratisation des PWAs sur la plateforme Apple.
- 30 Avril 2018 Microsoft annonce la mise à jour de EdgeHTML et

le support des Service Workers, Push, etc. avec une prise en charge des fichiers Web App Manifest. 10

Le Web App Manifest, une fonctionnalité qui manque de maturité sur certaines plateformes

Dans ce milieu des navigateurs où les itérations et le versioning sont extrêmement rapprochées, les Web App Manifests ne sont pas épargnés et font actuellement l'objet de développements intensifs de la part des géants du web.

Les Web App Manifest sont aujourd'hui en mesure de fonctionner, et ce sur la totalité des navigateurs dits "modernes", même si sur Safari, le chemin reste semé d'embûches et vous obligera à certaines conciliations.

Liste des éventuelles problématiques liées à iOS et l'utilisation du Web App Manifest (non-exhaustif) :

- Mauvaise gestion des splash screen ;
- Pas de gestion du theme-color ;
- Le display mode n'est pas opérationnel (fullscreen, minimal-ui) ce qui implique de gérer les boutons de navigation au sein de l'interface ;
- Impossible de forcer l'orientation de la PWA ;
- Pas de gestion de la transparence des icônes (comme sur Android par exemple) ;
- L'intégration au sein de l'environnement n'est pas finalisée, écran blanc lors du changement d'app.

Pour ce qui concerne les PWAs dans leur ensemble, il reste encore des problématiques de tailles à régler étant donné que cette technologie est dite "expérimentale" chez Apple.

- iOS limite l'espace de stockage du cache (mode offline) à 50 Mo ;
- Lorsque vous n'utilisez pas la PWA sur votre iPhone, iOS se permet de libérer l'espace de stockage en supprimant le cache de cette dernière ;
- iOS ne permet pas encore l'envoi de notification push (dont badge) ;
- iOS ne permet pas encore d'accéder aux informations de batterie, Bluetooth, Touch ID, ARKit, etc.

Le Web App Manifest pour une meilleure intégration dans les éco-systèmes

A terme, le fichier Web App Manifest permettra aussi de référencer votre PWA. En effet, ce fichier Manifest étant crawlable (fichier trouvable par un robot d'exploration type GoogleBot) les géants du web commencent déjà à se positionner sur son exploitation à grande échelle. Une PWA étant un dérivé des applications natives et des technologies web, les questions de visibilité dans les SERP (Search Engine Result Page) demeurent. Pour exemple, Microsoft a annoncé le 30 Avril 2018 que Bing était déjà en mesure d'indexer des PWAs sur base des fichiers Web App Manifest, et ce dans le but de les intégrer automatiquement dans le Microsoft Store disponible sur Windows 10.

Bing indexing service will automatically package your PWA in the .appx format required for installation on Windows 10 and assemble your app's store product page based on the metadata in its web app manifest. After claiming your PWA, you'll be able to further customize your store page and gain access to user analytics and other app management tools throug the Windows Dev Center dashboard.

Source : https://docs.microsoft.com/fr-fr/microsoft-edge/progressive-web-apps/microsoft-store#automatic-pwa-importing-with-bing

Etant à l'initiative du terme via Alex Russell et faisant largement la promotion des PWAs, Google s'est clairement positionné sur le fait que mixer les PWAs avec le format AMP (PWAMP) vient modifier la façon dont Google affiche les résultats de son moteur de recherche et de son store Google Play (UX, Rapidité, Services, etc.) et ce jusqu'à les mettre en avant en favorisant leur affichage.

Il y a fort à parier qu'Apple apportera rapidement une réponse. Toutefois il est important de préciser que même si Apple a été l'un des inventeurs du concept (par le biais de Steve Jobs lors de la WWD07 et des applications sur le "premier" iPhone), la firme est aujourd'hui un peu à la traîne concernant son implémentation et ses standards. L'AppStore étant un sujet épineux, il convient d'observer les réactions futures de la pomme à ce sujet. Au mois de Mars 2018 iOS 11.3 a été la première version d'iOS adressant des fonctionnalités avancées dites PWAs sans pour autant impacter l'AppStore.

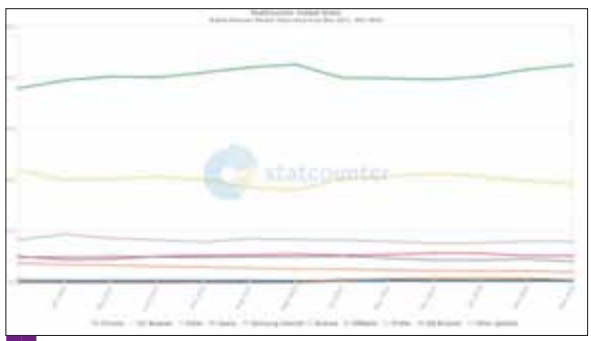
Le marché asiatique et indien est-il "Web App Manifest—Ready" ?

En Asie

Les GAFAM (Google, Apple, Amazon, Facebook, Microsoft) représentés par le couple Chrome + Safari restent largement en tête avec près de 60% du trafic sur le mobile en Mars 2018. 11

Cependant, les BATX (Baidu, Alibaba, Tencent et Xiaomi) proposent un nombre grandissant de navigateurs. Grâce à l'ère du smartphone, ils ont su doper leurs parts de marché en atteignant près de

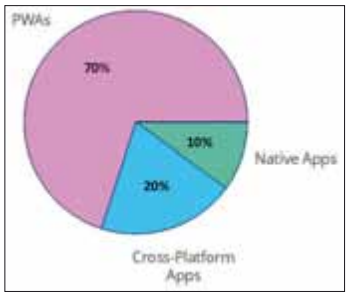
10



11

StatCounter expose les statistiques d'utilisation des navigateurs pour mobiles en Asie (Mars 2018)
Source : http://gs.statcounter.com/browser-market-share/mobile/asia#monthly-201703-201803

12 *Toxic Johann expose le support du Web App Manifest par les navigateurs (Mai 2018)*
Source : https://ispwaready.toxicjohann.com/#web%20app%20manifest



Les PWA vont s'imposer sur les mobiles

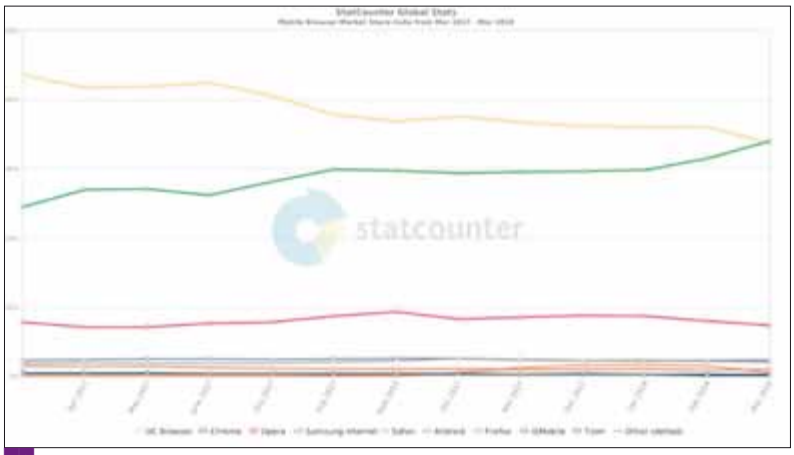
35% d'utilisation sur le mobile en Asie. Concernant l'implémentation des Web App Manifests sur ces navigateurs : 12
On constate que Baidu et Xiaomi proposent des solutions opérationnelles, mais celles-ci ne sont pas conformes aux standards. Lors de la génération de Web App Manifest, ce non-respect des standards va imposer la gestion de plusieurs cas spécifiques plus ou moins chronophages pour ces différentes plateformes.

En Inde

Au mois d'Avril 2018, le couple UC Browser + Chrome représentait près de 80% du trafic mobile et 90% en comptant le navigateur Opéra. 13
Il est à noter que UC Browser (40% du trafic mobile en Inde pour Avril 2018) a annoncé son support pour les PWAs :
The latest build is based on Chromium 57 with good supports for PWA and ES2015+'s new features
La documentation pour les développeurs d'UC Browser est accessible à cette adresse https://plus.ucweb.com/docs/ et fait référence à la MDN pour l'implémentation d'un Web App Manifest.

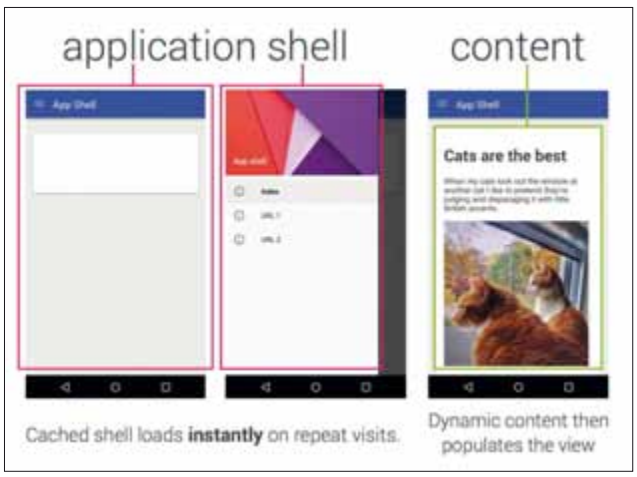
Pour aller plus loin avec les Web App Manifests

W3C https://www.w3.org/TR/appmanifest/ ;
MDN web docs https://developer.mozilla.org/en-US/docs/Web/Manifest ;
Les fondamentales par Google https://developers.google.com/web/fundamentals/web-app-manifest/.



13 *StatCounter expose les statistiques d'utilisation des navigateurs pour mobiles en Inde (Avril 2018)*

Support PWA dans Chrome. Source : goodbarber



Modèle application shell. Exemple sous Android.

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

Nos classiques

1 an
11 numéros
49€*

2 ans
22 numéros
79€*

Etudiant
1 an - 11 numéros
39€*

** Tarifs France métropolitaine*

Abonnement numérique

PDF ..... 35€
1 an - 11 numéros
Souscription uniquement sur www.programmez.com
Option : accès aux archives 10€

© mikkelwilliam

Offres 2018

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

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

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

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

** Offre limitée à la France métropolitaine*

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

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

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

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

☐ Vidéo : Symfony 3
☐ Vidéo : Raspberry Pi

PROG 221
Offre limitée, valable jusqu'au 28 septembre 2018

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

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

E-mail : @

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

*** Tarifs France métropolitaine**


Arnaud BREL
amaud.brel@sogeti.com


Laurent GRANGEAU
laurent.grangeau@gmail.com

Faites du minage !

Partie 2

Dans ce numéro, nous vous proposons de construire votre propre rig (plateforme) de minage afin de miner vos premières crypto-monnaies : de l'assemblage du hardware, à la configuration du software et un comparatif des différents logiciels de minage. Pour les bases voir Programmez!220.

1

Optimisation et maximisation du hashrate

Afin de gagner en performance de hashrate, il existe des astuces pour maximiser ce hashrate. La plus simple est d'overclocker ses GPU, par exemple avec le logiciel Afterburner de MSI. 1
Avec ce logiciel, plusieurs paramètres peuvent s'optimiser :

Core voltage	La tension d'entrée du GPU. A ne pas changer
Power limit	La puissance maximum que le GPU est autorisé à consommer. Le paramètre peut se situer entre 65% et 85% sans impacter les performances
Temp limit	Température maximum avant que le GPU ne s'éteigne ou ralentisse
Core clock (Mhz)	La vitesse principale du GPU. Peut généralement se mettre entre -75 et -100 sans affecter les performances
Memory clock (Mhz)	Le paramètre le plus important et qui influe le plus sur le minage. Certaines cartes supportent jusqu'à +800 Mhz
Fan speed	A laisser en automatique afin que les drivers adaptent la vitesse du ventilateur.

Si des cartes AMD sont présentes sur la plateforme, ajouter ces commandes avant de lancer le logiciel de minage aide à augmenter ses performances :

```
setx GPU_FORCE_64BIT_PTR 0
setx GPU_MAX_HEAP_SIZE 100
setx GPU_USE_SYNC_OBJECTS 1
setx GPU_MAX_ALLOC_PERCENT 100
setx GPU_SINGLE_ALLOC_PERCENT 100
```

Pool de minage

Un pool permet de répartir le travail. On ne mine pas en trouvant des blocs toutes les 30 secondes (il faut beaucoup beaucoup de puissance de calcul pour y arriver), mais des solutions qui contribuent à trouver un bloc. Le pool permet de séparer le travail en petites entités, et vous confie des calculs que votre matériel tente de résoudre. Si votre machine trouve une solution au calcul suffisamment vite, elle la partage avec le pool (share).
Notez qu'il est souvent peu contraignant d'être dans un pool. En effet, la majorité ne demandent aucune inscription, pour y participer il suffit d'utiliser leur lien serveur dans votre logiciel, et on s'identifie grâce à son wallet sur le site pour voir le résultat de son travail. La seule chose à bien vérifier, c'est comment le pool rémunère et combien de temps il faut pour commencer à avoir un aperçu de vos statistiques.
Il existe aujourd'hui une multitude de pools de minage, chacun avec leur puissance de calcul et plus ou moins de fees prélevées à chaque bloc. Afin de maximiser ses gains, il suffit de choisir le pool avec le plus de mineurs, qui augmentera statistiquement la chance de recevoir une récompense. Les 3 pools les plus intéressants sont :

Dwarfpool

Historiquement le plus gros pool, il paie selon un système HBPPS (le pool compte le nombre de blocs qu'il a trouvés durant la dernière heure et répartit les gains à tous les mineurs proportionnellement au nombre de partages effectués par chacun durant cette même durée), avec 1% de frais de paiement, et paient 1 fois par heure (mais seulement si votre montant plafond est atteint). Le minage est anonyme (pas de compte à créer). On peut régler le montant plafond à partir duquel les ether cumulés pour le travail effectué sont versés sur le wallet. Les paramètres de connexion sont les suivants :

```
URL_POOL : eth-eu.dwarfpool.com
PORT_POOL : 8086
```

Commande pour lancer ClayMore :

```
EthDcrMiner64.exe -epool eth-eu.dwarfpool.com:8086 -ewal WALLET_ADDRESS -eworker WORKER_NAME -epsw x"
```

Page de statistique :

http://dwarfpool.com/eth/address?wallet=WALLET_ADDRESS

Ethermine

Ethermine est actuellement le 2ème plus gros contributeur. Le mode de paiement se fait selon la méthode PPLNS (les gains obtenus pour le dernier bloc sont redistribués proportionnellement à la moyenne de vos partages durant les x derniers blocs trouvés). Il est possible de paramétrer la fréquence de paiement (Payment threshold in Ether). Attention par contre, il y a des commissions sur les paiements si votre seuil est inférieur à 1 Ether. Les frais de paiements sont de 1%. Le minage est anonyme (pas de compte à créer). Les paramètres de connexion sont les suivants :

```
URL_POOL : eu1.ethermine.org
PORT_POOL : 4444
```

Commande pour lancer ClayMore :

```
EthDcrMiner64.exe -epool eu1.ethermine.org:4444 -ewal WALLET_ADDRESS -eworker
WORKER_NAME -epsw x"
```

Page de statistique :
http://ethermine.org/miners/WALLET_ADDRESS

Nanopool

Ce pool paie 4 fois par jour, selon la méthode PPLNS (les gains obtenus pour le dernier bloc sont redistribués proportionnellement à la moyenne de vos partages durant les x derniers blocs trouvés avec N le nombre de partages durant les dernières 20mins). Il n'y a pas de commission fixe sur les paiements, juste des frais de paiement à hauteur de 1%, comme les autres. Le minage est anonyme (pas de compte à créer). Ils recommandent de ne pas miner chez eux si vous n'avez pas un hashrate d'au moins 5 Mh/s.
Les paramètres de connexion sont les suivants :

```
URL_POOL : eu1.nanopool.org
PORT_POOL : 9999
```

Commande pour lancer ClayMore :

```
EthDcrMiner64.exe -epool eu1.nanopool.org:9999 -ewal WALLET_ADDRESS -eworker
WORKER_NAME -epsw x"
```

Page de statistiques :
http://eth.nanopool.org/account/WALLET_ADDRESS

Acheter des Bitcoins / Altcoins

Dans cette partie nous allons vous expliquer comment acheter des bitcoins /altcoins.

1ère étape : choisir une plateforme (exchange) :

Pour réaliser votre premier achat de crypto-monnaie vous allez devoir vous inscrire sur une plateforme qui accepte la monnaie fiduciaire (EUR, USD, etc.), car la plupart des plateformes de trading acceptent uniquement les crypto-monnaies.
Les plateformes les plus connues qui acceptent la monnaie fiduciaire sont : Bitstamp, Kraken, Coinbase, Bitfinex …
Toutes les plateformes ne se valent pas : selon la monnaie que vous comptez utiliser certaines plateformes sont plus intéressantes que d'autres.
Il faut considérer :

- Les fees (frais de transactions) appliqués qui sont différents selon les plateformes.
- La sécurité, il faut privilégier les plateformes dites sécurisées, c'est-à-dire celles qui ont un nombre conséquent d'utilisateurs, qui n'ont jamais eu de failles de sécurité et qui sont en place depuis plusieurs années.

Si vous comptez acheter des crypto-monnaies avec des euros je vous conseille d'utiliser la plateforme Kraken.
Un piège à éviter : les plateformes de type ETORO sont à éviter, en effet au lieu de détenir les coins, vous spéculez juste sur leur valeur. Vous ne pourrez donc pas envoyer vos coins dans les plateformes qui acceptent uniquement les crypto-monnaies comme nous le verrons plus tard dans cet article. De plus les fees (frais de transactions) appliqués sur cette plateforme sont très élevés.
Une fois votre plateforme choisie, il faut « se faire authentifier » par la plateforme, c'est-à-dire donner une preuve d'identité (carte d'identité, passeport, …) et une preuve de domicile (rib, facture de téléphone, …) qui seront vérifiées. Lorsque cette étape est réalisée, vous allez pouvoir virer/déposer de l'argent sur la plateforme pour pouvoir commencer à acheter des crypto-monnaies comme nous allons le voir dans la suite cet article.

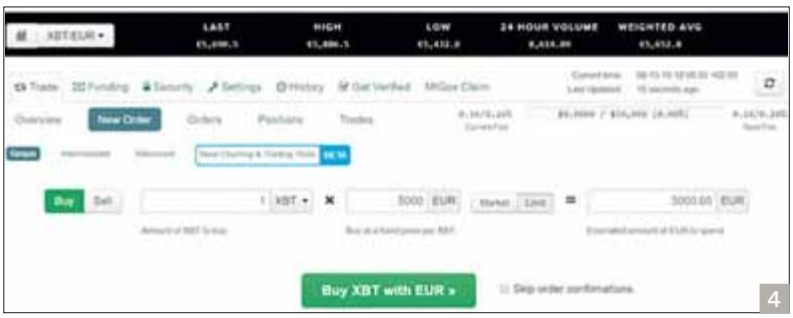
2ème étape : acheter une crypto-monnaie

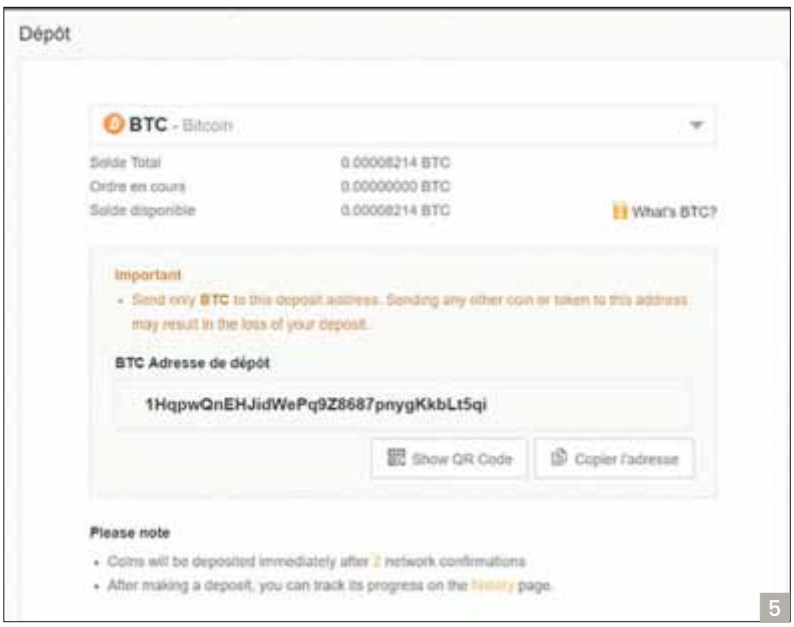
Tout d'abord, il faut savoir que chaque crypto-monnaie est associée à une paire.
Une paire est l'association de 2 monnaies, crypto ou fiduciaires, c'est-à-dire qu'on peut acheter la crypto-monnaie choisie seulement avec la ou les paires associées. Les paires les plus utilisées sont les monnaies fiduciaires, le bitcoin et l'Ethereum. Dans notre cas nous allons partir du principe que nous voulons acheter du bitcoin avec des Euros. Pour cette partie nous utiliserons l'interface de Kraken pour illustrer mes propos. 2
Pour acheter une crypto-monnaie il suffit de choisir la paire voulue (dans notre cas XBT/EUR) puis de cliquer sur « New order ».
Il existe deux façons de placer un ordre :
Soit on place un ordre en « market » c'est-à-dire qu'on achète au prix actuel : 3

2

3

Comme on peut le voir sur cette illustration, nous achetons au prix fixé par le marché à « l'instant t ». Donc on achète un bitcoin au prix de 5690,86.
Pour la deuxième possibilité, on place un ordre en « limit » c'est-à-dire qu'on choisit à quel prix on veut acheter notre bitcoin, cependant on n'a pas la garantie que notre prix sera atteint par le marché. De ce fait, il se peut que notre ordre ne passe jamais et donc l'ordre ne sera pas exécuté. 4
Sur cette illustration, on place un ordre en « limit » à 5000 € pour acheter un bitcoin, c'est-à-dire que nous achèterons un bitcoin si et seulement si le prix du bitcoin (actuellement à 5690 €) atteint 5000 €, sinon nous n'achèterons rien.





Pour vendre ses bitcoins (ou autres crypto-monnaies) c'est exactement la même démarche on peut vendre au prix du marché ou vendre à un prix fixé au préalable.
Malheureusement les plateformes qui acceptent les monnaies fiduciaires sont assez limitées en nombre de crypto-monnaies, ce sont souvent les crypto-monnaies les plus connues.
Pour avoir accès à plus de crypto-monnaies, il faut s'inscrire sur d'autres plateformes où l'on peut uniquement acheter la crypto-monnaie souhaitée avec d'autres crypto-monnaies (bitcoins, Ethereum le plus souvent).
Nous allons donc voir dans la troisième partie comment transférer des crypto-monnaies d'une plateforme à une autre.

3ème étape : transfert de crypto-monnaie d'une plateforme à une autre.

Pour illustrer mes propos je vais utiliser les interfaces de kraken et Binance.
Une fois votre crypto-monnaie achetée, vous pouvez la transférer d'un exchange à l'autre.
Pour ceci, il faut générer une adresse de réception de cette crypto-monnaie dans la plateforme où vous voulez l'envoyer.
Dans notre cas, si nous voulons envoyer le bitcoin acheté sur Kraken vers la plateforme Binance, nous devez générer une adresse de réception de bitcoin depuis Binance. 5
Sur cette illustration je demande à la plateforme Binance de me fournir une adresse de dépôt pour la crypto-monnaie bitcoin. De ce fait, Binance me donne une adresse où je vais pouvoir envoyer des bitcoins depuis Kraken.
Enfin je vais renseigner cette adresse à Kraken pour envoyer mes bitcoins sur Binance. 6
Sur cette illustration on voit bien qu'on a renseigné l'adresse où envoyer le bitcoin.
Il nous reste à renseigner le nombre de bitcoins que l'on veut envoyer.
Une fois l'envoi réalisé, il nous suffit d'attendre que les bitcoins arrivent sur la plateforme Binance (environ 20 minutes).
Attention des fees sont appliquées lors d'un envoi de bitcoin d'une plateforme à une autre pour rémunérer les mineurs qui sécurisent la blockchain.
Les fees varient d'une crypto-monnaie à une autre, c'est pourquoi il n'est pas forcément judicieux de transférer des bitcoins car les fees sont assez élevées.

Une fois vos bitcoins arrivés, vous pourrez acheter toutes les crypto-monnaies de la paire « bitcoin », c'est-à-dire acheter avec du bitcoin des crypto-monnaies indexées au bitcoin.

Toutes ces explications sont valables pour toutes les autres plateformes. Seule l'interface utilisateur diffère. •

Sébastien Warin
CEO/CTO Constellation
Creative Technologist
https://developer.myconstellation.lo
https://sebastien.warin.fr

niveau 200

Automatiser sa maison de A à Z

Partie 2

Après avoir découvert dans le numéro précédent comment connecter sa maison avec des technologies de domotique filaires ou sans fil, concevoir son architecture réseau et son système d'alarme et ajouter des objets connectés que nous avons interconnecté avec la plateforme Constellation, terminons notre dossier par la confection d'objets connectés « home-made » et surtout l'ajout d'une dose d'intelligence pour notre maison !

MAKER : CONCEVEZ VOS PROPRES OBJETS CONNECTÉS

Parfois on peine à connecter certains dispositifs car on ne trouve pas son bonheur dans le commerce : offre inexistante, mal adaptée à l'usage souhaité ou trop cher !
Cela nous amène à le réaliser soit même, petit tour d'horizon des possibilités !

Créer des capteurs Wifi

Pour connaître la luminosité d'une pièce on peut pour 5€ environ fabriquer un capteur de lux facilement.
Une prise 220V avec un transformateur 3,3v pour alimenter un ESP8266 connecté sur un capteur de luminosité TSL2561.
Cela se programme par un Arduino dans un pseudo C++ avec Arduino IDE et le SDK Constellation. Après avoir fait l'acquisition de la luminosité ambiante grâce à la librairie TSL2561 d'Adafruit il suffit de publier la donnée sous forme d'un StateObject dans Constellation par la ligne :

```
constellation.pushStateObject("Lux", stringFormat("{ 'Lux':%d, 'Broadband':%d, 'IR':%d }", lux, full, ir), "LightSensor.Lux", 300);
```

Autrement dit on publie un objet complexe contenant 3 propriétés avec une durée de vie de 5 minutes (300 secondes). C'est ce même StateObject que nous avons utilisé précédemment pour l'éclairage de l'entrée.
En suivant ce principe, on peut réaliser un tas d'objets en se basant sur des puces ESP/Arduino ou des Raspberry afin de capter/mesurer des données et les publier en tant que StateObject. On pourra ainsi les exploiter dans des algorithmes .NET, Python, Javascript ou autres.
Pour la consommation électrique, j'utilise également un ESP8266 pour se connecter en Modbus sur un compteur d'énergie type EM112 et ainsi publier dans Constellation un StateObject contenant le voltage actuel, intensité, puissance apparente, active et réactive, le facteur de puissance, fréquence, etc. 31

Un capteur infrarouge basé sur un ESP8266 est également déployé dans le salon. Cela permet de réagir à la télécommande IR comme par exemple piloter les lumières du salon avec les touches qui ne servent pas sur une télécommande de TV !

31 *Compteur Modbus avec un ESP8266*

32 *S-Opener*

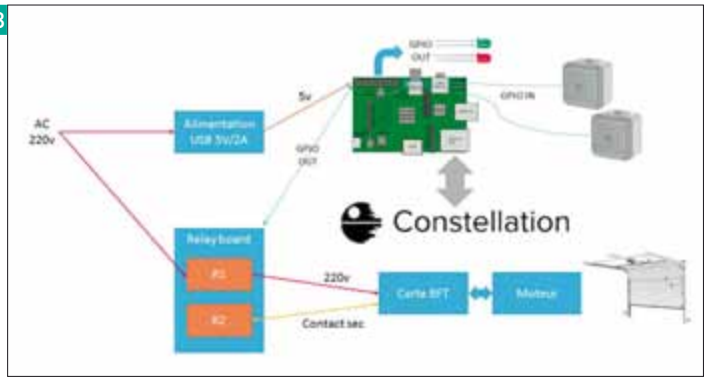
Connecter une porte de garage

Pour rendre ma porte de garage connectée j'ai tout simplement installé un Raspberry connecté dans Constellation qui, au travers d'un package Python expose des MessageCallback permettant d'activer un relais pour déclencher l'ouverture ou la fermeture de la porte. 32

Le package Python de la porte de garage est lui-même attaché aux StateObjects du système d'alarme Paradox. Ainsi lorsque l'alarme est armée, il empêche l'ouverture !

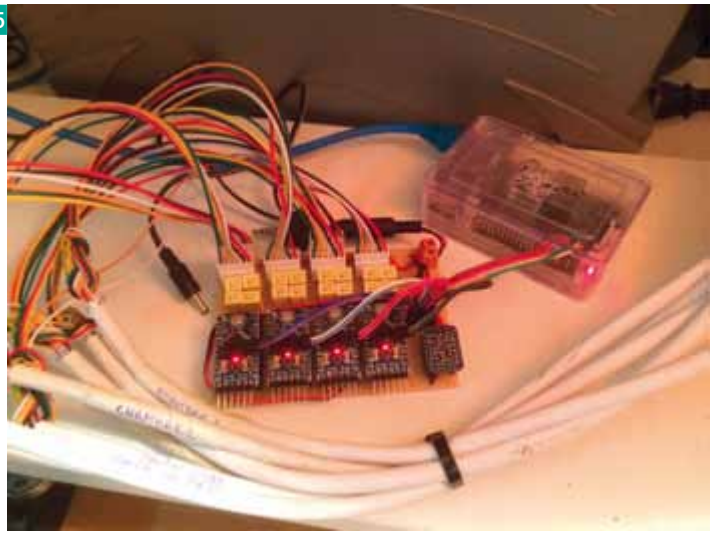
Une application mobile Android développée en Cordova exploite l'API Javascript de Constellation pour prévenir de notre arrivée (via le module GPS) et ainsi désarmer l'alarme et ouvrir la porte de garage automatiquement. 33

33 *S-Opener*



34 *Coffret dans l'abri de jardin*

35 *Pilotage des volets*



Dans cette architecture décentralisée, un package « IA » (le moteur de règle en définitive) de la maison allumera les lumières du garage lors de l'ouverture de la porte s'il fait nuit. En somme, l'algorithme est attaché au StateObject de la luminosité et du capteur de la porte de garage sur le système l'alarme pour déclencher le MessageCallback d'allumage d'un device Z-Wave qui pilote l'éclairage du garage. Même principe que celui de l'entrée expliqué précédemment. Au final, en rentrant chez moi, la porte de garage est déjà ouverte, l'alarme désarmée et la lumière allumée. Il me reste plus qu'à rentrer la maison !

Connecter un abri de jardin

Dans l'abri du jardin un switch PoE alimente deux cameras IP agrégées sur ZoneMinder et une borne Wifi wAP AC pour diffuser le réseau dans le jardin. Un Raspberry pilote deux relais pour l'éclairage intérieur et extérieur, détection de l'ouverture de la porte, mesure de la température et humidité via un DHT22 et compte l'énergie consommée par l'abri via une liaison Modbus (RS485) sur un compteur DZT.

Concrètement sur le Raspberry est préalablement installée une sentinelle connectée dans Constellation. A partir de la Console Constellation j'ai déployé 4 packages tous écrits en Python : un pour Modbus, un pour le pilotage des GPIO (éclairage et porte), un pour le capteur de température DHT22 et un dernier pour gérer l'écran OLED de présentation des informations de l'abri. 34

Domotiser les volets avec un ESP ou Raspberry en filaire

Lucas Dupuis vous avait présenté dans ce magazine à l'été 2015 sa solution de gestion de volets sur base d'un Raspberry.

L'état de chaque volet est connu dans Constellation via les StateObjects et des MessageCallbacks qui permettent d'activer des relais pour ouvrir ou fermer les volets de la maison. 35

Une solution économique à moins de 100€ lui permet, après configuration d'une passerelle HomeKit/Constellation, de piloter ses 10 volets depuis son iPhone.

Un dashboard central avec une tablette Android

Egalement présenté dans ce magazine à l'été 2015, S-Panel est un dashboard domotique mural home-made.

Il s'agit tout simple d'une tablette Android encadrée dans un cadre peint aux couleurs de la pièce. 36

Sur cette tablette est déployée une application Android/Cordova qui, en full-screen, présente une interface développée en Bootstrap/AngularJS pour afficher les différentes valeurs des StateObjects des principaux équipements de la maison.

Connecter la sonnette

Toujours à base d'un ESP8266, un appui sur la sonnette extérieure déclenche une entrée de l'ESP qui propage le message dans Constellation.

Ainsi n'importe quel package connecté dans Constellation peut réagir à cet évènement. On peut alors utiliser le package ZoneMinder pour récupérer la snapshot de la caméra sur la rue, le package PushBullet pour envoyer ce snapshot, le package Hue pour faire clignoter les lumières du salon si un utilisateur se trouve là (information qu'on peut récupérer avec les StateObjects de l'alarme ou ceux

du routeur Mikrotik en fonction de la borne Wifi où est connecté l'utilisateur). On peut également diffuser un message vocal via le package S-Sound ou encore afficher une notification avec le package de la TV ou du media-center Kodi.
Bref les possibilités sont immenses ! 37

S-Energy : suivi du compteur d'eau et de gaz

On peut également citer S-Energy, un Raspberry sur lequel est déployé un package Constellation écrit en Python qui s'occupe de faire des photos du compteur d'eau toutes les 30 secondes et qui après un petit algorithme de détection des caractères (OCR) publiera dans un StateObject la consommation actuelle d'eau. 38
Comme il s'agit d'un StateObject, n'importe quel système peut suivre cette valeur. Une page Web comme le dashboard S-Panel affichera la consommation d'eau en temps réel, un package comme « Graylog » s'occupera d'enregistrer toute consommation dans une base de données ElasticSearch ou encore un package C# de l' « IA » de la maison pourra utiliser S-Sound pour indiquer vocalement à l'utilisateur dans la salle de bain l'eau qu'il a consommé pendant sa douche et plus encore.

AJOUTER DE L'INTELLIGENCE

Maintenant que nous avons une maison connectée grâce à un système domotique filaire ou sans fil complété ou non avec des objets connectés, nous allons pouvoir ajouter de l'intelligence.
En effet jusqu'à présent nous avons juste une maison « connectée » c'est-à-dire où chaque dispositif peut être piloté depuis une interface informatique (une page Web, une application smartphone ou même à la voix), mais il n'y a rien d'intelligent.
Ceci dit avec une plateforme comme Constellation on a la tuyauterie nécessaire pour faire émerger cette IA. En effet nous pouvons ajouter des applicatifs (packages) quel que soit votre langage de développement dans le bus Constellation. Vous n'avez donc aucune limite autre que celle offerte par votre langage/plateforme et vos compétences de développeur.
Lorsque l'on développe une IA, qu'elle soit symbolique (typiquement un moteur de règle), ou connexionniste (un réseau de neurone) pour une « smart home », on peut appliquer le concept d' « agent intelligent », c'est-à-dire une entité qui peut être considérée comme percevant sont environnement et qui agit sur cet environnement via des effecteurs. Mathématiquement parlant, le comportement de l'agent est décrit par la fonction agent qui fait correspondre une action à une séquence de percepts (c'est-à-dire à l'historique des entrées perceptives de ce qu'il a perçu).
Avec Constellation un agent est en fin de compte un package (applicatif) qui peut être déployé sur une sentinelle connectée sur le bus. La consommation de StateObject dans son code est en définitive des percepts pour percevoir l'environnement (souvenez-vous du StateObject « Lux » et « MouvementEntree » dans notre exemple précèdent) et les MessageCallbacks des effecteurs pour enclencher des actions sur l'environnement (comme allumer une lumière dans notre exemple).
On peut donc grâce à Constellation implémenter facilement des fonctions agent plus ou moins complexes : reflexe simple pour réagir directement aux percepts (comme le fait d'allumer la lumière si il y a du mouvement), reflexe fondé sur des modèles (où il y a un maintien de l'état interne afin de suivre les aspects du monde non

36 *Dashboard mural S-Panel*

37 *Notification de la sonnette*

discernable dans le percept courant), sur des buts (pour atteindre des objectifs), sur l'utilité (pour maximiser la satisfaction espérée) et même des agents capables d'apprentissage pour améliorer leur performance.
Sans rentrer trop en profondeur dans les concepts théoriques de l'IA, voyons quelques exemples d'agent s'exécutant dans ma Constellation.

Les automatises simples

La plupart des agents sont à reflexe simple. Par exemple quand on rentre, les lumières s'allument automatiquement, la température du chauffage est réglée selon les préférences des personnes présentes (on connaît les personnes présentes dans le logement avec la liste des smartphones connectés sur les bornes Wifi de la maison), la musique est jouée (toujours selon les goûts des personnes présentes et de l'heure également), un message de bienvenue est diffusé, vocalement et visuellement sur le miroir, etc. A l'inverse le fait d'aller se coucher ou de partir automatisera la fermeture des lumières, volets, amplis, écrans/TV, chauffage, etc.
De manière globale les équipements de base tels que les volets ou les lumières sont tous automatiques, basés sur la présence, le mouvement, la luminosité et l'activité perçue (on est à table pour le repas, on regarde un film, il y a un feu de cheminée, etc.).

La sonnette et les enfants

La sonnette étant connectée dans Constellation, on prévient l'utilisateur quand quelqu'un sonne en utilisant différents effecteurs (notification smartphone, lumineuse, sonore, etc.). Là encore on peut déduire et notifier de la bonne manière l'utilisateur : il regarde

S-Energy 38

un film, on affiche la notification sur la TV, il est dans la cuisine, un appel lumineux, il prend sa douche, un message vocal et autrement un push sur le smartphone. Toutes ces informations sont connues avec les StateObjects. Bien entendu il y a le carillon qui sonnera dans la maison (du moins l'agent qui régit ces règles) : pour ne pas réveiller mes deux enfants lorsqu'ils dorment l'agent coupera ce carillon via un MessageCallback (effecteur) sur l'ESP8266 qui connecte la sonnette dans Constellation.

Là encore comment savoir que mes enfants dorment ? Avec quelques percepts comme les StateObjects du volet et le taux de CO^2 de leurs chambres respectives. A l'heure où j'écris ces lignes, à une heure un peu tardive, le volet de mon ainé est fermé à 100% et le taux de CO^2 est de 990ppm selon la sonde NetAtmo, on en déduit qu'il est dans sa chambre en train de dormir. Autrement dit, avec ce « reflexe », si un livreur ou de la famille sonne chez nous, on a la certitude qu'il ne pourra pas réveiller les enfants à cause de ce coup de sonnette !

Il pleut

Il pleut, on le sait avec le pluviomètre. L'anémomètre lui indiquera la direction et la force du vent. Selon ces deux indictions et si une fenêtre est restée ouverte la maison pourra nous prévenir selon le bon mode de communication en fonction de l'activité et de la présence dans la maison. En l'absence de réaction de notre part, elle fermera le volet correspondant.

La température de la chambre de Jules

On peut également ajouter un algorithme assez simple pour calculer la dérivée de la température dans chaque pièce et réveiller le chauffage central pour éviter les chambres trop froides. En effet la chambre de mon ainé étant mal isolée la température décroit plus rapidement que le reste de la maison. A chaque mesure de la température on calcule la dérivée de la pente pour prédire la température qu'il fera dans 30 minutes. En fonction de cela on réveille le thermostat Nest du salon en augmentant la température ambiante du salon d'un degré pendant le temps que la pente s'inverse.

Se souvenir d'une action et expiration des données

Un agent à simple reflexe peut par exemple ouvrir les volets lorsqu'il fait jour. Mais que faire si l'utilisateur le ferme derrière lui pour regarder un film ? Potentiellement rien, mais si l'agent redémarre il perdra son état interne et donc ouvrira de nouveau le volet. Un agent peut alors publier son état dans un StateObject de façon à le restaurer en redémarrant et ne pas perdre la mémoire. D'ailleurs son état interne peut être également considéré comme un percept pour un autre agent.

De plus on peut se poser la question de l'expiration des « contre-ordres ». Je ferme le volet en journée, ce qui irait à l'encontre du reflexe de l'agent, mais l'agent perçoit qu'un film vient de démarrer. Ne serait-il pas logique de rouvrir le volet à la fin du film ?

On peut aussi réfléchir à la « fraîcheur » des percepts. Que faire si le Raspberry ou l'ESP mesurant la luminosité est déconnecté ? La valeur du StateObject contenant une information à la base d'une multitude de règle est donc erronée. En pleine journée la maison pourrait penser qu'il fait nuit si le capteur s'est déconnecté pendant la nuit. Constellation apporte la notion de « Life time » sur les StateObjects pour indiquer la durée de vie d'une donnée. Ainsi dans vos algorithmes vous pouvez mettre en place des solutions « dégradées ». Par exemple si je n'ai pas une donnée valide sur la luminosité courante je me baserai sur le StateObject du package « DayInfo » me donnant très précisément l'heure de lever et de coucher du soleil.

Prédiction d'évènement

Les séquences de percepts, c'est-à-dire l'ensemble des évènements qui se sont produits peuvent alimenter une fonction d'apprentissage. Par exemple quand je vais me coucher, on sait que les enfants dorment (ils sont encore petits !), donc qu'il y a un taux de CO^2 plus important dans leurs chambres, volets et lumières sont fermés. On sait aussi (StateObject du package « WindowsControl ») que ma machine dans mon bureau est soit verrouillée ou soit éteinte.

L'agent perçoit ensuite une séquence d'évènements qui plus ou moins me font aller dans la salle de bain (allumage des lumières), consommer un peu d'eau (lavage de dents), voire une chasse d'eau, passer dans le couloir (détecteur de mouvement de l'alarme), arriver dans la chambre (roaming de mon smartphone sur la borne du dernier étage), allumer la lampe de chevet, puis après une augmentation du taux de CO^2 dans notre chambre, fermer l'une puis les deux lampes de chevet. Ils dorment !

La séquence n'est pas strictement identique (ordre, durée, etc.) mais en rentrant ça dans un algorithme d'apprentissage de séquence on est capable de classifier et de prédire, c'est-à-dire de savoir quand on dort et quand on part dormir voire même quand est-ce qu'on va dormir (statistiquement on se couche plus tard le weekend qu'en semaine pour ne prendre qu'un exemple de variable).

Conclusion

Comme vous pouvez le constater à travers ce dossier les perspectives sont immenses et la seule limite sera votre imagination.

Peu importe la ou les technologies employées il faut d'abord connecter son logement avec des plateformes ouvertes via des API ! Privilégier les technologies filaires : Paradox EVO, camera IP/PoE, domotique IPX/KNX. Plus fiable, plus sécurisé et plus sain mais si vous ne pouvez/voulez pas passer de câble, le Z-Wave couplé à différents objets connectés Wifi, ZigBee ou autre vous permettra le pilotage de tous vos équipements de la maison. Des lumières aux volets en passant par les équipements multimédia, thermostat, porte de garage, piscine ou autre !

Et si vous ne trouvez pas le bon objet connecté, n'hésitez pas, faites-le vous-même, les Raspberry ou les ESP8266/ESP32 sont de superbes plateformes avec un potentiel phénoménal une fois connectées dans Constellation ouvrant la porte de l'interconnexion avec tous les systèmes.

Quand votre maison sera connectée, fédérez tout cela dans un système global pour éviter le patchwork de technologie, développez votre propre interface centrale multimodales et implémentez dans le langage de votre choix n'importe quel algorithme et code en ayant à disposition une multitude de percepts que sont les StateObjects et effecteurs grâce aux MessageCallbacks. De quoi développer une IA avancée qui va au-delà des simples scénarios qu'on retrouve dans les box domotique afin de créer une véritable maison intelligente. N'hésitez pas à vous rendre sur le portail développeur Constellation ou sur mon blog pour plus de détails sur Constellation et les projets et solutions présentés dans ce dossier. •

Frédéric Sagez
Codeur du groupe NoExtra-Team
fsagez@gmail.com

Le graphisme sur ATARI ST

niveau 200

Nous allons nous intéresser au fonctionnement de l'affichage d'un Atari ST et comprendre comment sont gérées les parties graphiques entre la mémoire et la partie vidéo du ST.

Comment afficher des pixels sur un écran? Pour rappel 1 pixel est égal à 1 point et ce, en utilisant une couleur disponible sur la palette et suivant la résolution désirée. Si on veut utiliser le nombre maximum de couleurs sur notre écran, cela est possible mais uniquement en basse résolution pour le ST avec 16 couleurs au total que l'on pourra choisir sur un échantillon de palette de 512 couleurs pour un Atari STF ou 4096 couleurs pour un Atari STE. Petit rappel : l'Atari ST ne comporte que 3 résolutions d'écran possibles : Basse, Moyenne et Haute résolution spécifique au moniteur monochrome de type SM124.

La théorie selon Atari

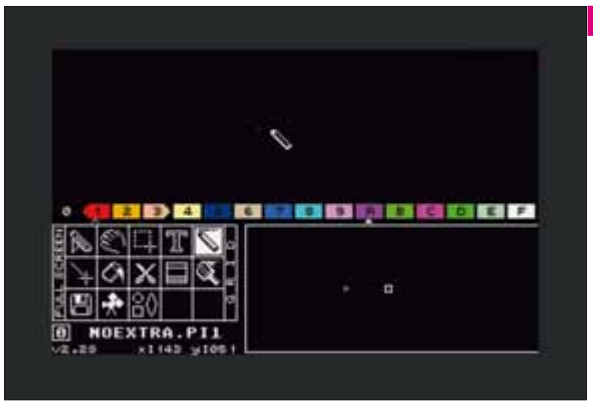
Ce qu'il faut retenir pour un écran en résolution ST-LOW (basse résolution), soit un affichage en 320 par 200 pixels de 4 bits composé de 4 plans (soit 16 couleurs utilisables) que cela représente un total de 320 x 200 x 4 bits = 256000 bits, soit 32000 octets au total de mémoire vidéo utilisée. Et cette zone de mémoire vidéo de 32 ko se trouve exactement à partir de la limite supérieure de la mémoire. Elle est initialisée en tant que telle dès la mise sous tension du ST. Donc si je calcule bien pour un Atari STF faisant 512 ko de mémoire d'origine, notre adresse mémoire au format hexadécimal devrait être de $80000 - $0FD00, soit l'adresse $78300. Eh bien non! Le calcul est faux !!!??? La bonne réponse est l'adresse $78000 car la taille finale de notre écran physique fait au total 32768 octets soit $08000. Alors comment faire pour ne pas se tromper, surtout si l'on veut utiliser aussi un autre écran de 32 ko comme écran physique ? On utilisera généralement les fonctions étendues appelées XBIOS du ST en passant par un appel du Trap #14 et en passant le numéro de la fonction à appeler en paramètre, XBIOS(2) pour obtenir l'adresse physique et XBIOS(3) pour obtenir l'adresse logique (un deuxième écran) quel que soit le nombre de RAM d'un ST.

```
move.w #0,-(a7)          ; set ST Low Resolution
move.l #$78000,-(a7)     ; set physical screen
move.l #$78000,-(a7)     ; set logical screen
move.w #5,-(a7)          ; function Setscreen()
trap   #14               ; XBIOS function called
lea.l  12(a7),a7         ; for Atari 520 STF
```

Pour rappel l'adresse physique de l'écran est la zone de mémoire qui est affichée sur le moniteur par le composant vidéo appelé SHIFTER. L'adresse logique est utilisée comme base de l'écran par toutes les fonctions de sorties du ST. Pour la suite de cet article, nous utiliserons la même adresse vidéo car pour notre démonstration, seul un écran est nécessaire à l'affichage vidéo.

Définition d'un écran graphique

Notre écran est composé d'une ligne de 320 pixels en horizontal et de 200 lignes en vertical : une ligne de 320 pixels x 4 bits est égal



1 *Tracé du point sur l'outil Néochrome Master*

à 1280 bits et si on divise par 1 byte cela donnera 160 octets. Une ligne fait 320 pixels en longueur et elle est découpée en 20 colonnes, donc pour afficher un pixel sur l'écran, nous disposons de 20 colonnes de 16 pixels. Un bloc de 16 pixels contient 4 mots de 16 bits pour chaque plan. Dans notre exemple suivant nous allons dessiner un point de couleur numéro 10 sur les 16 couleurs de la palette, il prendra les valeurs suivantes en mot – Word : $0000,$0100,$0000,$0100 (valeurs en hexadécimal).

Donc pour récapituler, si je veux écrire un point sur l'écran qui utilise un numéro de couleur :

- les couleurs de 0 et 1 utilisent un plan : j'utilise un seul mot,
- les couleurs de 0 à 3 utilisent deux plans : j'utilise les deux premiers mots,
- les couleurs de 0 à 7 utilisent trois plans : j'utilise les trois premiers mots,
- les couleurs de 0 à 15 utilisent quatre plans : j'utilise les quatre mots.

Et en tout on peut mettre 16 points de couleurs de 0 à 15 sur un bloc de 16 pixels contenant 4 plans. Bon vous n'avez pas tout compris, mais pour afficher un point sur l'écran il faut le positionner dans un bloc de 16 pixels, hum… pas de panique ! Voici un exemple pour vous rendre compte du premier niveau de complexité.

Un peu de pratique

Pour la démonstration j'utilise l'outil de dessin **Neochrome Master** pour dessiner un pixel, j'utilise la couleur numéro 10 (valeur $A en hexadécimale) sur la palette qui correspond à la couleur violette située dans le quatrième plan comme on peut le voir sur le dessin 1 en basse résolution. Le point se trouve à la position 136 pour l'axe des X et 51 pour l'axe des Y et comme on peut le voir, cela correspond au neuvième bloc de 16 pixels situé sur l'écran. Pour avoir la valeur en mot – soit en WORD – de ma ligne, j'exporte en hexadécimal via l'outil les valeurs de la ligne où se situe mon point pour connaître les valeurs exactes des quatre mots de mon bloc de 16 pixels. 2

Pour dessiner sur notre écran le point, nous allons utiliser l'outil **DEVPAC** et écrire une petite routine en assembleur permettant de

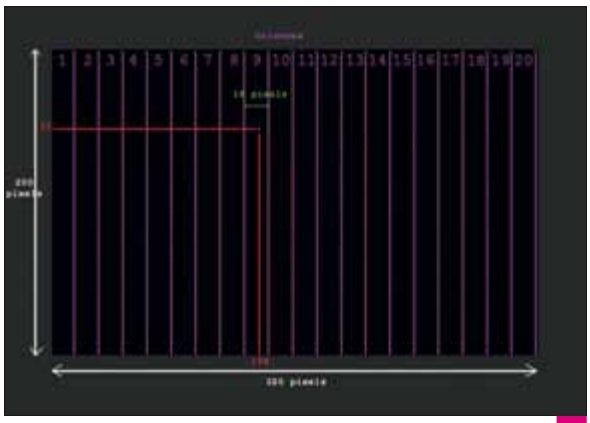
visualiser sur l'écran le résultat final. Dans un premier temps je vais déclarer un seul écran pour afficher mon résultat et je suis les préconisations du constructeur en utilisant l'adresse de la partie mémoire graphique d'un Atari 520 STF (soit 512Ko de mémoire) à l'adresse $78000 tout en utilisant une fonction étendue du BIOS. Je vais écrire une routine appelée « main » que je vais appeler en mode SUPERVISEUR pour accéder à tout l'environnement du ST. Dans la routine je vais sauvegarder la palette système de base utilisée par le Desktop, afficher un fond noir et initialiser la couleur numéro 10 de la palette de destination. Ensuite je remplis mon écran de lignes indiquant les 20 blocs de 16 pixels pour vérifier que je suis au bon endroit. 3

En passant par le registre d'adresse a0, je lui assigne l'adresse de mon écran physique et je place mes 4 mots sur les 4 différents plans. Pour connaître les coordonnées de la ligne c'est très simple : on multiplie le numéro de ligne par 160 octets et pour la colonne, c'est la 8ème colonne multipliée par 8 octets (comme on commence par zéro, on soustrait toujours le numéro de la colonne par un).

```
  SECTION TEXT

move.w #0,-(a7)             ; set low resolution
move.l #$78000,-(a7)        ; set physical screen
move.l #$78000,-(a7)        ; set logical screen
move.w #5,-(a7)             ; setscreen
trap   #14                  ; XBIOS function called
lea.l   12(a7),a7           ; for Atari 520 STF

pea main(pc)                ; exec routs main in super mode
move.w #$26,-(sp)
trap #14
addq.l #6,sp

clr.l -(sp)                 ; exit !
trap #1

main:
movem.l $ffff8240.w,d0-d7
movem.l d0-d7,sav_pal       ; save palette system

move.l #$00000777,$ffff8240.w ; put palette for color 0 and 1
move.l #$00E70FC7,$ffff8250.w ; put palette for color 10 and 11

lea.l   $78000,a0           ; Fill screen by chunck of 16 pixels
move.w  #(160*200)/4-1,d7   ; in 20 parts
move.l  #$00020000,(a0)+    ; of 2 first words
dbf    d7,*-6               ; loop above line instruction

*> calculate position on the screen
lea.l   $78000,a0           ; manage our screen
lea     160*51(a0),a0       ; position Y = 51
add.w#(9-1)*8,a0            ; position X = 136
                            ; 9 chuncks - 1 x 2 x 4 words
*> put 16 pixels with a pixel color number 10
* Rappel: dc.w  $0000,$0100,$0000,$0100
move.w #$0000,0(a0)         ; first bitplane
move.w #$0100,2(a0)         ; second bitplane
```

```
1 *
2 *  NEOchrome V2.28 cut buffer contents (left justified):
3 *  by Chaos, Inc. of the Delta Force (member of The Union)
4 *
5 *    pixels/scanline     = $013F (bytes/scanline: $00A0)
6 *  # scanlines (height) = $0002
7 *
8 *  Hardware color pallet  (color 0 to 15):
9 *
10 *      $0800,$0F00,$0FD0,$0FDC,$0FFC,$0005,$0E65,$083F
11 *      $00E7,$0FC7,$0E07,$01F2,$0E25,$0AD1,$0DED,$0FFF
12 *
13      dc.w  $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
14      dc.w  $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
15      dc.w  $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
16      dc.w  $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
17      dc.w  $0000,$0100,$0000,$0100,$0000,$0000,$0000,$0000
18      dc.w  $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
19      dc.w  $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
20      dc.w  $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
21      dc.w  $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
22      dc.w  $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000
23
                                        9ème colonne
```

Fichier de données hexadécimales de la ligne 2



Résultat de l'affichage de notre point en assembleur 3

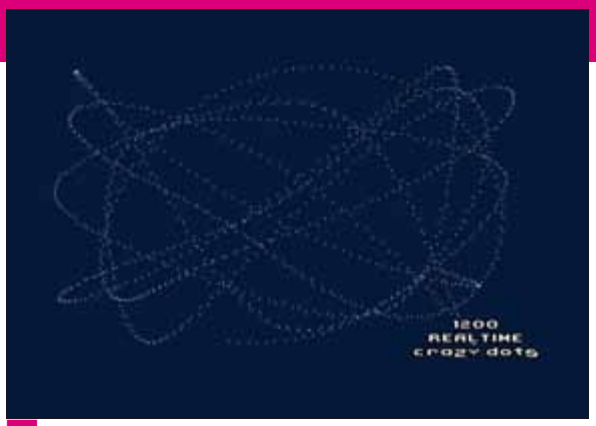
```
move.w #$0000,4(a0)         ; third bitplane
move.w #$0100,6(a0)         ; fourth bitplane

move.w  #7,-(sp)            ; wait for a key press
trap  #1
addq.l  #2,sp

movem.l sav_pal,d0-d7       ; restore palette system
movem.l d0-d7,$ffff8240.w
rts

sav_pal ds.l 8
```

Enfin j'attends qu'une touche du clavier soit appuyée pour rendre la main à l'utilisateur tout en m'assurant de remettre la palette d'origine. Au final j'écris 4 mots aux adresses suivantes : $7A060, $7A062, $7A064 et $7A066 de mon écran physique.

Maintenant que nous avons vu comment dans cet exemple on affiche un point sur l'écran, il serait intéressant de pouvoir placer un point où l'on veut sur l'écran avec simplement ses coordonnées.

Point, pixel et plan

Et pour cet exercice je vais vous montrer comment écrire un point à la position voulue. Mais dans notre cas, le point pourra être affiché uniquement sur l'un des 4 plans et ne pourra donc utiliser que 4 couleurs possibles. Pourquoi ? C'est juste pour une question d'optimisation sur l'utilisation de mes 4 Words car le premier plan utilise uniquement un Word comme on peut le voir sur les 16 pixels : $0080,$0000,$0000,$0000. Donc pour la couleur numéro 1 on pourra utiliser un seul mot à l'adresse écran $7A060 et on pourra mettre la même valeur pour les trois autres plans si on le souhaite et cela donnera ceci :



4 *Infinite live of the Blitter – NoExtra (2011)*

- Plan numéro 2 utilisera la couleur 2 à l'adresse écran $ 7A062,
- Plan numéro 3 utilisera la couleur 4 à l'adresse écran $ 7A064,
- Plan numéro 4 utilisera la couleur 8 à l'adresse écran $ 7A066.

Maintenant le plus difficile est à venir. Pour calculer mes coordonnées sur l'écran physique, on sait que pour placer notre point sur une ligne verticale il faut le multiplier par 160 octets (pour rappel : 320 pixels / 16 pixels x 4 mots x 2), donc on fera une simple multiplication par 160 avec l'instruction mulu. Pour la partie horizontale nous devons trouver où notre point doit se situer et cette fois-ci on enlève la notion de colonne, nous allons diviser la coordonnée par 16 pixels puis la multiplier par 8 octets. (Pour rappel : 4 mots x 2)
Ensuite pour afficher le point final à l'écran, nous utiliserons l'instruction bset pour activer le bit sur l'octet de poids fort du mot et pour cela nous récupérons le reste de notre division que l'on va soustraire à la valeur 15 (16 pixels – 1) tout en sachant que notre bit est compris entre 0 et 7 car l'instruction bset ne gère que des données sur 8 bits. Donc on obtiendra :

- Axe vertical : Offset Y = 51 x 160 = 8160, soit à partir de l'adresse vidéo $78000 + $1FE0 va donner l'adresse $79FE0
- Axe horizontal : Offset X = 136 / 16 x 8 = 64, on additionne $40 à $7A020 qui donnera la position finale à l'adresse $7A060 qui est l'offset de l'écran et on indiquera le bit à afficher sur cette adresse pour « allumer » le point à afficher. Voici le code source qui reprend la même structure que le premier programme, on réutilise seulement la procédure main pour implémenter le code suivant :

```
SECTION TEXT

move.w #0,-(a7)             ; set low resolution
move.l #$78000,-(a7)        ; set physical screen
move.l #$78000,-(a7)        ; set logical screen
move.w #5,-(a7)             ; setscreen
trap   #14                  ; XBIOS function called
lea.l  12(a7),a7            ; for Atari 520 STF

pea main(pc)                ; exec routs main in super mode
move.w #$26,-(sp)
trap #14
addq.l #6,sp

clr.l -(sp)
trap #1

main:
movem.l $ffff8240.w,d0-d7
movem.l d0-d7,sav_pal       ; save palette system

move.l #$00000777,$ffff8240.w; put palette for color 0 and 1

lea.l  $78000,a0            ; Fill screen
move.w #(160*200)/4-1,d7    ; in 20 parts
move.l #$00020000,(a0)+     ; of 8 words
dbf    d7,*-6

*> Display one pixel with the first bitplane
lea.l  $78000,a0            ; adress of the screen
move.w  #136,d0             ; position X is 136
move.w  #51,d1              ; position Y is 51
move.w  d0,d2               ; backup X in the register d2
andi.w  #$fff0,d0           ; which cluster of 16 pixels to display ?
sub.w   d0,d2               ; sub multiple of 16 with the X position
subi.w  #15,d2              ; seek bitnumber between 16 pixels
neg.w   d2                  ; the bitplane for the bset instruction
lsr.w   #1,d0               ; divided by 2 : offset X done !
mulu.w  #160,d1             ; multiplication by 160 : offset Y done !
add.w   d0,d1               ; position in the screen : $7A060
move.w  (a0,d1.l),d0        ; get bitplane word : $0080
bset    d2,d0               ; activate the bit for display
move.w  d0,(a0,d1.l)        ; display pixel on screen offset

move.w  #7,-(sp)            ; wait for a key press
trap  #1
addq.l  #2,sp

movem.l sav_pal,d0-d7       ; restore palette system
movem.l d0-d7,$ffff8240.w
rts

sav_pal   ds.l 8
```

Afficher un point ne prend pas trop de temps au niveau du processeur mais si on veut en afficher plus, on peut aller jusqu'à un maximum d'environ 400 points à l'écran par frame. Et comme vous avez pu le voir dans le code assembleur ci-dessus, tout reste encore à optimiser comme par exemple supprimer la multiplication et créer une table de multiplication sachant que l'on utilise 200 lignes au total.
Et puis utiliser un seul plan nous fait gagner en instructions avec l'utilisation d'un Word, donc peu coûteux en ressources machine ; il existe tellement de « tricks » que les codeurs n'ont pas fini de nous étonner, comme par exemple dans cette démo 4 où le nombre de points à afficher a été multiplié par 3. Et cet effet affiche toutes les dots en temps réel ! Dans un prochain article nous parlerons de l'utilisation de graphismes de tailles diverses tels que des sprites, éléments de zones graphiques qui font partie d'un jeu par exemple. Et bien sûr dans cet article je vous expliquerai comment on gère les déplacements de zone graphique un peu plus volumineux qu'un point sur un Atari ST.
Les sources en assembleur sont disponibles sur : https://github.com/NoExtra-Team/framework, plus précisément dans le répertoire framework / sources / PIXEL. •

Ça progresse

Une publication Nefer-IT, 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08 - Directeur de la publication & Rédacteur en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Ont collaboré à ce numéro : S. Saurel
Nos experts techniques : F. Sagez, S. Warin, A. Brel, D. A. Pham, S. Pontoreau, A. Benali, C. de Vandiere, M. Bacchi, P. Lamarche, C. Pichaud, P. Charrière, B. Hanus, Y. Lejeune, S. Croce, D. Voituron, C. Villeneuve
Couverture : © mikkelwilliam - **Maquette** : Pierre Sandré.
Publicité : François Tonic / Nefer-IT - Tél. : 09 86 73 61 08 - ftonic@programmez.com.
Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.
Marketing et promotion des ventes : Agence BOCONSEIL - Analyse Media Etude - Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD Téléphone : 09 67 32 09 34

Contacts : Rédacteur en chef : ftonic@programmez.com - Rédaction : redaction@programmez.com - Webmaster : webmaster@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908 - © NEFER-IT / Programmez, août 2018

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

Un code et des interfaces cross-plateformes :

WINDEV®

Application **WINDOWS**

Application **JAVA**

Application **LINUX**

CAPITALISEZ VOTRE EXISTANT :

EXEMPLE D'APPLICATION WINDEV RECOMPILÉE NATIVEMENT POUR CHAQUE PLATEFORME

Site avec serveur **LINUX** avec WEBDEV

Site en **PHP** avec WEBDEV

Application **UWP**

Application sur **Smartphone, Tablette et Terminal industriel** avec WINDEV Mobile

Avec WINDEV, WEBDEV et WINDEV Mobile 23, développez «une seule fois», et créez:

Des applications natives:
- Windows
- Linux
- Mac
- Java

+ Des sites pour :
- Windows
- Linux
- ou en PHP

+Des applications mobiles natives pour smartphones, tablettes et terminaux industriels:
- Android
- iOS
- UWP
- Windows CE.

Tout est natif.

WINDEV élu **«Langage le plus productif du marché»**

VERSION EXPRESS GRATUITE
Téléchargez-la !

Tél : 04 67 032 032

WWW.PCSOFT.FR