OceanStream et la MIAGE Sorbonne s'associent pour créer

L'OceanBox

Un projet AGILE et DevOps écoconçu

PARTIE ANALYSE

Abdel Benamara | Julien Doujet | David Ekchajzer | Mathieu Ridet

20/01/2020

Table des matières

Introduction

On dit souvent que les océans sont les poumons de la terre. En effet ils produisent à eux seuls la moitié de l'oxygène que nous respirons. Les écosystèmes marins sont aujourd'hui au cœur des problématiques environnementales dues au réchauffement climatique et aux pollutions humaines comme les déchets plastiques. Animées par les recherches scientifiques, notamment celles de l'ONG TARA, de nombreuses associations essaient de faire prendre conscience de l'importance pour la planète de préserver ces écosystèmes et de sa biodiversité. Long de 2300km la barrière de corail en Australie est un des enjeux cruciaux de cette préservation. Le réchauffement climatique ainsi que les activités humaines locales polluantes (agriculture, tourisme, surpêche…) ont rompu l'équilibre de cette écosystème si fragile. Bien que nous somme pour la plupart au courant de la situation dramatique de la grande barrière, les citoyens ne se rendent pas forcement compte de son importance et de la vitesse de sa dégradation.

OceanStream est une association loi 1901 fondée en 2018 part des amoureux des milieux marins. Leur objectif est d'éduquer la population à la complexité de l'équilibre des Océans, mers et cours d'eau. Ils souhaitent que les populations qui ne sont pas en contact avec ces milieux, puissent se rendre compte de leur beauté et de leur fragilité. Pour ce faire, ils conduisent des projets de médiation scientifique, culturelles et artistiques à destination de différents types de populations. Un de leurs projets majeurs vise à utiliser la vidéo pour que la mer devienne un environnement familier en diffusant, par exemple, des images de la grande barrière de corail. OceanStream souhaite s'inscrire dans le mouvement de la slow TV qui a pour objectif de montrer des plans fixes et lents pour s'adapter à la vitesse lente de la nature.

Certains des membres d'OceanStream sont des anciens étudiants de Paris 1. Ils ont pu s'appuyer sur le soutien de La Sorbonne pour développer leurs projets. C'est donc tout naturellement que le président de l'association : Adrien Landa s'est tourné vers la MIAGE Sorbonne pour développer un POC de boitier électronique visant à diffuser le flux vidéo généré par leurs caméras.

Nous avons tout de suite adhéré au projet. Celui-ci nous permettant de travailler en collaboration avec une association sur un projet innovant. C'est une opportunité unique qui ne se représentera pas forcement dans notre vie professionnelle, et plus spécialement sur des questions d'écologie désespérément d'actualité. Conscient de l'impact très important et grandissant du numérique sur l'environnement, ce projet était pour nous l'occasion d'apprendre à développer des projets informatiques en y intégrant l'écoconception. Les compétences que nous acquerrons au cours de ce projet pourront potentiellement nous permettre de faire évoluer les pratiques dans les entreprises où nous travaillerons plus tard. Notre deuxième motivation réside dans les problématiques techniques auxquelles nous confronte le projet. Nous pourrons utiliser et développer des compétences en informatique hardware, programmation système, réseaux, administration serveur, etc…

Préanalyse

Il existe aujourd'hui une très grande communauté qui s'articule autour du streaming de flux provenant de webcam à différents endroits dans le monde. Les spectateurs accèdent à des flux depuis des sites spécialisés, comme EarthCam par exemple, et peuvent échanger via une section « commentaire ». Certains de ces sites proposent des vidéos provenant de milieux marins.

Cette implémentation pose plusieurs limites. Premièrement le récepteur de la vidéo doit connaître ces sites et savoir chercher parmi des millions de flux webcam. Il doit donc faire la démarche d'aller vers cette vidéo. Cette démarche est réalisée par des individus déjà très mobilisés et informés sur la question d'écologie en milieu marin. De plus, ces sites sont réservés à une communauté très spécifique qui ont ce hobby et n'est donc pas adapté à la médiation scientifique qui doit viser la population la plus diversifiée possible.

Créer un objet, une OceanBox, est un moyen de rendre ce service plus concret et plus quotidien. Cela permettrait aussi plus facilement de monétiser le flux vidéo fourni par l'association qui se servira de l'argent pour leurs activités de dépollution marine. La box peut même devenir un objet de décoration. Si le cahier des charges est respecté, la solution sera Plug and Play et facilitera l'utilisation pour toutes les personnes peu importe leurs connaissances en informatique.

On peut également considérer les aquariums comme un existant. En effet, ils permettent d'avoir un contact avec un milieu aquatique au quotidien mais ils nécessitent un entretien, et ne sont pas adaptés aux espaces publiques. De plus ils donnent l'image d'une nature contrôlée par l'homme alors qu'OceanStream souhaite montrer la réalité des espaces marins.

L'OceanBox pourra être implémentée facilement dans les espaces publiques pour remplacer totalement ou partiellement la publicité. Enfin, elle permettra de rapprocher les récepteurs de lieux très exotiques comme la grande barrière de corail.

Nous avons tout de même trouvé des limites à la création de l'OceanBox qu'il convient d'expliciter. Son utilisation va utiliser de l'énergie pour filmer, transmettre et diffuser le flux sans compter l'énergie et les ressources nécessaires à la production du boitier en lui-même. C'est pourquoi nous avons décidé d'essayer de réduire au maximum l'impact environnemental de l'OceanBox. Par rapport à nos choix qui, en plus des justifications techniques et financières, devront être justifiés écologiquement. C'est ainsi que nous souhaitons ce produit : Eco-conçu. Le Maître d'ouvrage et président d'OceanStream, Adrien Landa, a validé la proposition car il partage avec nous l'envie de réduire l'impact des projets numériques sur la planète et spécialement ceux de son association.

Après avoir parlé avec Adrien Landa ainsi qu'à des clients potentiels particuliers ou entrepreneurs de notre entourage, nous avons identifié plusieurs personas qui nous permettront d'adapter notre projet aux différents profils d'utilisateurs amenés à utiliser le produit :

Le client/receveur particulier

Il récupère le boîtier contre caution et s'abonne au flux vidéo pour soutenir les actions de l'association. Il a simplement à le brancher à sa télévision (même si elle n'est pas connectée) pour recevoir et retransmettre le flux vidéo proposé par OceanStream en continue. Il ne veut pas se soucier de devoir l'éteindre et la rallumer régulièrement mais souhaite économiser le plus d'énergie possible. Le boitier reconnaît quand quelqu'un est présent grâce au capteur de mouvement et s'allume.

Le client professionnel

Le client professionnel souhaite soutenir les actions de l'association et le faire savoir. Il utilise le boîtier pour streamer sur des écrans dans les lieux qu'il détient (salle d'attentes, gares, magasins, salons, bureaux, évènements…). L'installation et la désinstallation de la box doivent être facile s'il souhaite la changer régulièrement d'emplacement et/ou en installer plusieurs.

Le receveur en lieu public

Le receveur dans un espace public a accès aux vidéos ponctuellement ou quotidiennement si la box diffuse sur son lieu de travail ou sur un chemin emprunté régulièrement comme une gare ou une station de métro. Il peut découvrir les actions d'OceanStream par ce biais, soutenir leurs actions et éventuellement acheter sa propre box.

L'équipe de développement

L'équipe de développement souhaite que les mises à jour des box soient faciles et ne nécessitent pas de rapatrier toutes les box pour les mettre manuellement à jour. C'est pourquoi elle souhaite qu'une chaine DevOps soit mise en place. Elle est constituée de bénévoles. Elle souhaite que les flux soient automatisés pour ne pas avoir à gérer quotidiennement l'administration des box.

Il est important pour nous de définir les persona car nous avons opté pour une méthode de développement projet proche des utilisateurs.

Méthodologie Projet

Un projet AGILE

Nous avons choisi la méthodologie de développement projet AGILE. Elle repose sur quatre principes fondamentaux énoncés par les dix-sept experts du développement de projets informatiques rédacteurs du « manifeste agile » :

« Nous découvrons de meilleures approches du développement logiciel en le pratiquant et en aidant les autres à le pratiquer. Ce travail nous a amené à accorder de l'importance :

- Aux individus et leurs interactions plutôt qu'aux processus et aux outils ;
- À un logiciel fonctionnel plutôt qu'à une documentation exhaustive ;
- À la collaboration avec les clients plutôt qu'à la négociation contractuelle ;

- À l'adaptation au changement plutôt qu'à l'exécution d'un plan. »

Les fonctionnalités que nous allons développer devront être compréhensibles par les utilisateurs pour qu'ils puissent nous faire un retour éclairé. Le découpage des fonctionnalités sera donc fait en priorité en fonction des utilisateurs et non en fonction des implémentations techniques. OceanStream et ses membres seront pour nous d'abord des collaborateurs avant d'être des clients ce qui, dans la situation contractuelle où nous sommes, est très facile à concevoir. Enfin, la méthode AGILE nous donnera une souplesse dans le développement. En effet, certaines fonctionnalités (User-Stories), ou implémentations de ces fonctionnalités, pourront être ajoutées, enlevées ou modifiées en cours de projet en fonction des retours utilisateurs ou de problématiques techniques. Cela est un réel avantage puisque notre faible expérience pourrait nous mener vers de mauvais choix. Ainsi, nous pourrons capitaliser plus facilement sur nos échecs. Enfin, cela nous permettra de continuer à développer des fonctionnalités tant qu'il nous reste du temps.

Nous allons adapter ces principes à notre projet de petite taille. Nous allons utiliser l'implémentation Scrum en utilisant la plupart de ses principes en nous accordant une certaine liberté pour ne pas nous enfermer dans une méthodologie trop lourde par rapport à l'équipe réduite et peu diversifiée qu'est la nôtre. Nous allons découper notre projet en plusieurs sprints correspondant à plusieurs fonctionnalités.

Notre workflow pour un sprint est le suivant :

- Ecriture des User-Stories : En collaboration avec la MOA/le PO (Maîtrise d'ouvrage/Product Owner), nous décrivons une fonctionnalité du point de vue utilisateur. Un résumé de l'US est fait sous la forme « je fais…grâce à…pour… ». Les critères de validation sont également énoncés du point de vue utilisateur. Cela nous aidera dans notre développement de tests.

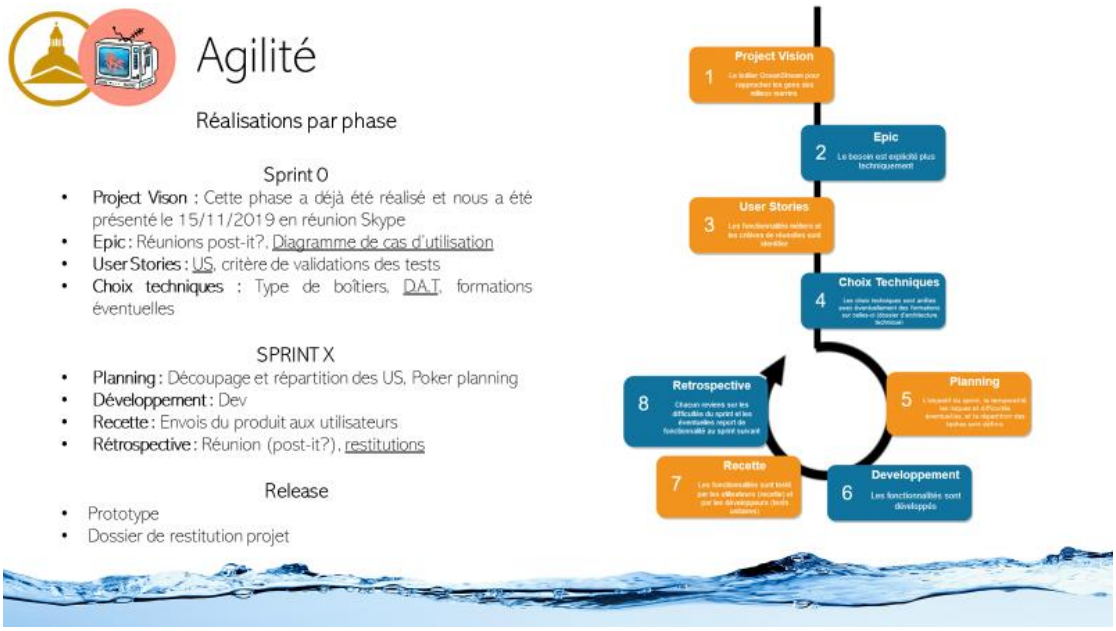
- Sprint Backlog : Nous choisissons les US à développer lors de ce sprint en prenant en compte les dépendances entre les US et la volonté du PO.

- Poker Planning : Pour nous répartir les US, nous organiserons un poker planning à chaque sprint. Cette réunion consiste à noter la difficulté technique d'implémentation d'une US part des notes suivant la suite de Fibonacci. Ainsi, nous nous mettons d'accord sur les difficultés que nous pourrions rencontrées et nous nous répartissons équitablement le développement des US. Enfin, nous espérons que le PO pourra être présent à ces réunions pour qu'il se rende compte de la difficulté de développement des différentes fonctionnalités même s'il n'a pas de profil technique. Il comprendra ainsi mieux les délais et les potentiels échecs.

- Phase de développement : Nous développons les fonctionnalités en suivant les instructions de l'US avec ses tests associés. Si besoin, nous nous formons à des compétences nécessaires au développement de l'US.

- Test Utilisateur : Nous aurons une box chez un utilisateur pour qu'il puisse nous faire des retours sur les nouvelles fonctionnalités.

- Rétrospectives : Nous revenons avec toute l'équipe sur les résultats du sprint, les difficultés rencontrées et les améliorations à apporter au sprint suivant.



Dans la méthode Scrum, tout le monde est au même niveau d'égalité. Ceci est une bonne chose car tout le monde peut intervenir et proposer des améliorations. Dans notre cas, nous avions prévu dès le début de ne pas instaurer de hiérarchie dans notre groupe. Ainsi, nous aurons plus de fluidité. Cependant tout le monde doit être responsable pour que cela fonctionne. C'est pourquoi Scrum définit des rôles avec des prérogatives précises que nous nous sommes appropriés :

- Product Owner : Dans notre cas il s'agit également de la maîtrise d'ouvrage. Il s'agit de notre contact à OceanStream : Adrien Landa. Il a pour rôle de représenter les commanditaires du projet. Il sera chargé, en collaboration avec nous, de définir les User-stories. Il gérera également la partie budgétaire.
- Utilisateur : Il est l'élément central de notre projet. Il doit tester les fonctionnalités à la fin de chaque sprint et faire ses retours au PO.
- Scrum master : Il a pour rôle d'organiser les Sprint, de rappeler les risques et est garant de l'harmonie du groupe. Il aura la charge d'administrer le tableau Trello que nous avons mis en place pour représenter et répartir les User-Stories.
- Maitrise d'œuvre : Il est le contact privilégié du PO/MOA. Il a pour rôle de justifier les choix techniques et d'organiser les modélisations du système.

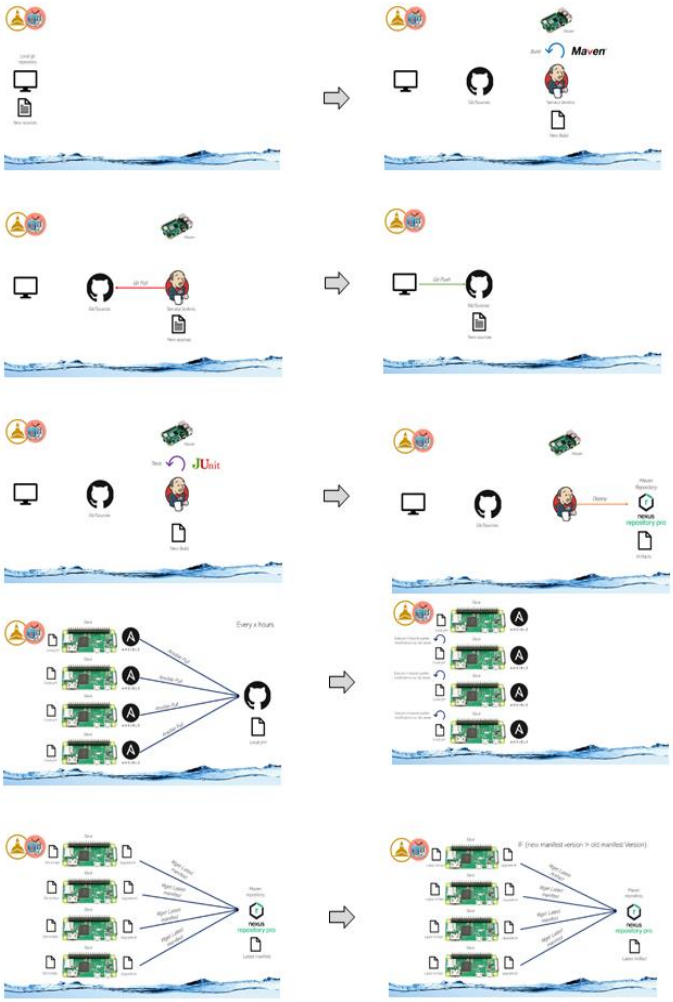
Un projet DevOps

Le mot DevOps vient de la contraction entre Développement et Opérations. C'est un mouvement visant à rapprocher, voire fusionner, le développement avec la production. Cela passe par la fusion partielle ou totale des équipes, des environnements, des workflows et des objectifs de développement et des opérations de production. Cette méthode permet une mise en production très rapide et se marie très bien avec la méthode AGILE. Elle part du principe qu'une application peut être testée en intégration pendant des mois mais rien ne vaudra un test grandeur nature avec un déploiement et des correctifs rapides à mettre en œuvre à posteriori. Pour ce faire, une chaine DevOps industrialisée doit être mise en place depuis le développeur jusqu'aux utilisateurs sans ou avec peu d'interventions humaines. Ainsi, un correctif mineur peut être effectué en quelques minutes/heures.

Cette méthode est intéressante car elle nous permettra, une fois la chaine DevOps mise en place, de développer des micro-fonctionnalités et de la mettre en production facilement.

Le Chief DevOps Officer aura pour rôle d'administrer cette chaine DevOps ainsi que la partie production du projet (numéros de release, retours de bugs, serveurs de production…).

Nous allons utiliser des outils gratuits et Open Sources à l'état de l'art pour notre chaine DevOps. GitHub pour gérer les répertoires VCS, Jenkins en tant que gestionnaire de Build et Ansible pour le déploiement. Une modélisation est le plus approprieriez pour expliquer notre chaine DevOps :



Green IT

Puisque l'écologie est au cœur de notre projet nous avons souhaité qu'elle soit également un élément central de notre méthodologie projet. Nous avons donc décidé de suivre une partie des nouveaux principes apportés par le Green IT ou informatique durable. L'informatique à un impact important puisqu'elle représente 7% de la consommation d'électricité mondiale.

L'écoconception porté par la Green IT est donc un nécessité. Les choix fonctionnelle et techniques doivent prendre en compte des indicateurs environnementaux au même titre voir plus important que les indicateurs financiers et techniques.

Dans notre projet, cela se matérialise par les justifications de nos choix hardware (type de carte utilisé) et software (architecture des flux).

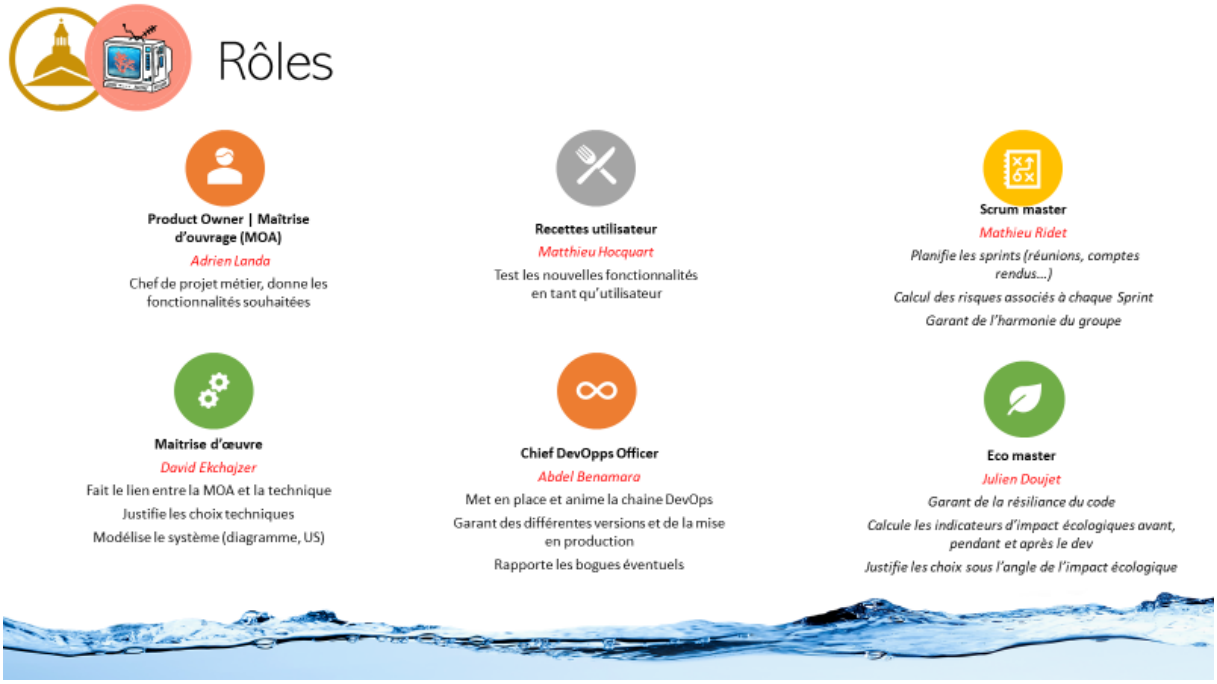
Un des éléments du cahier des charges consiste en un détecteur de mouvement. Cela est un exemple d'écoconception car il permettra d'économiser l'énergie utilisée par l'écran quand aucun utilisateur n'est là pour recevoir le flux vidéo.

Une personne du groupe prendra le rôle d'éco-master. Son rôle sera de nous aider à faire des choix éclairés concernant l'impact environnementale à partir d'articles ou de données récupéré et compilé.

Répartition des taches

Même si chacun aura un rôle défini, les prérogatives pourront être partagées mais devront être déléguées par la personne responsable du pôle concernée pour que chacun se sente responsable d'une partie du projet.

- Adrien Landa en tant que commanditaire du projet sera PO/MOA.
- Mathieu Hocquart, membre d'OceanStream, sera notre utilisateur. Il aura une carte connectée à notre chaine DevOps chez lui. Il a l'avantage d'être freelance en informatique et sera capable d'effectuer des manipulations si nous lui demandons. Par ailleurs ses retours auront l'avantage de pouvoir être techniques.
- Mathieu Ridet sera Scrum Master.
- David Ekchajzer sera MOE.
- Abdel Benamara sera Chief DevOps Officer.
- Julien Doujet sera Eco master. Ayant une formation de biologiste, il pourra plus facilement lire et comprendre les données et articles environnementaux.
- Nous nous partagerons tous les quatre la partie développement.



Rendu attendu

Comme vu avec Adrien Landa, nous souhaitons lui rendre un projet qui pourra ensuite être facilement repris par une autre équipe. Nous avons donc en tête que notre rendu doit être assez précis avec de la documentation pour assurer la réversibilité.

Le premier des éléments rendu est l'architecture techniques de nos box dans le dossier d'architecture technique (DAT). Il y sera détaillé les choix Hardware, les branchements, les choix d'architecture des flux vidéo, les fichiers exécutables, les dépendances…

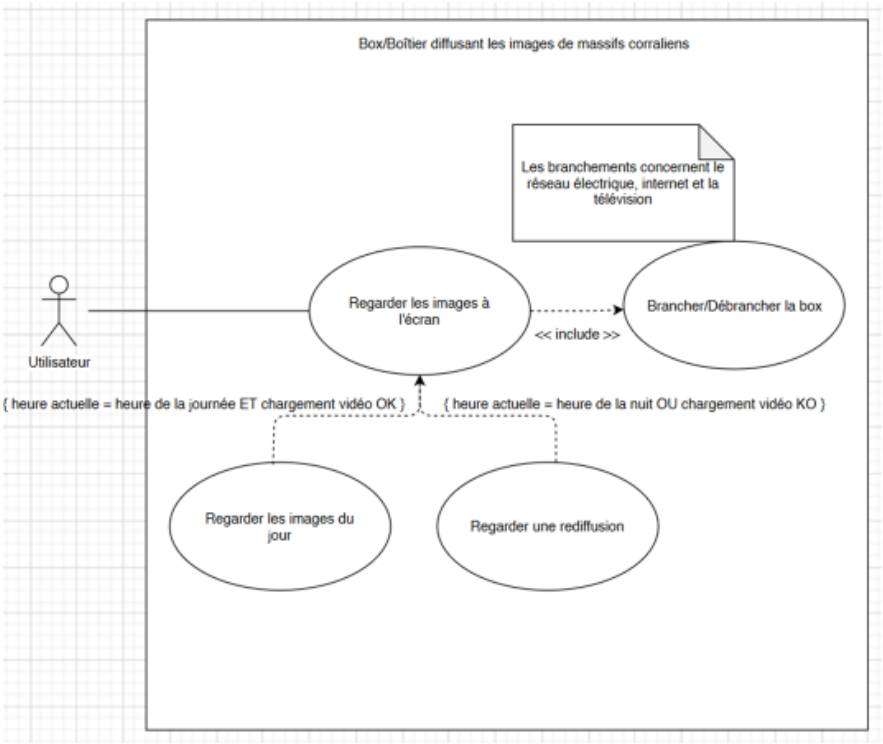
Ensuite, une preuve de concept (POC) sera également rendue pour montrer les fonctionnalités développées. Ce POC sera constitué d'une Master (master DevOps) et de deux Slaves (OceanBox).

Nous aimerions travailler avec des designers pour désigner le boîtier imprimable en 3D à partir de plastique recyclé provenant des océans. Adrian Landa a proposé de contacter l'entreprise « Precious Plastic » qui pourrait être un fournisseur de matière plastique recyclée ainsi que l'imprimeur.

Enfin, nous aimerions qu'une chaine DevOps complète puisse être mis en place pour que la prochaine équipe de développement puisse continuer à implémenter de nouvelles fonctionnalités sans avoir à « rapatrier » les box. Cela facilitera grandement leurs phases développement.

Enfin tous les documents et présentations pptx seront rendus comme livrables pour documenter le plus précisément notre projet.

Système



Le système se veut extrêmement simple du point de vue de l'utilisateur. En effet, nous souhaitons qu'il ait simplement à brancher la box à interner (via câble Ethernet qui consomme moins que le WiFi et a un meilleur rendement), au réseau électrique et a un écran en HDMI (norme la plus populaire actuellement dans les pays occidentaux).

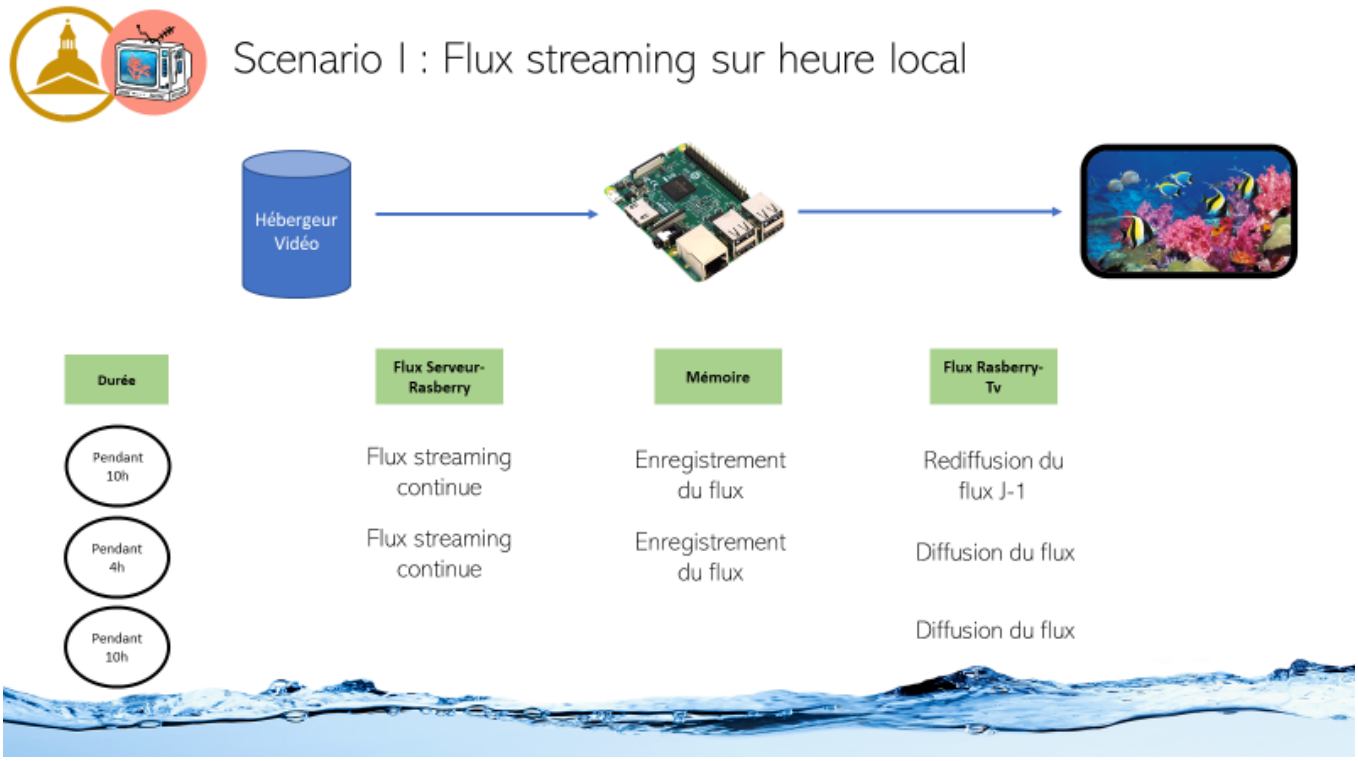
Il peut ensuite regarder les images de la vidéo de la webcam d'OceanStream à l'écran quand il est présent dans la pièce.

Il est important de noter qu'il y a une problématique importante de coordination temporelle. En effet, les webcam d'OceanStream filment pendant 14h (quand la lumière du jour est disponible). Nous devons donc implémenter une rediffusion pendant 10h.

De plus, les utilisateurs souhaitent voir les images filmées à l'heure x locale de la webcam à l'heure x de leur fuseau horaire. Un utilisateur qui se réveille à 10h voudra voir les images du lever du soleil et non du coucher. En d'autres termes, on doit supprimer les effets du décalage horaire pour les utilisateurs.

La solution proposée dépendra du type de flux choisi pour faire parvenir la vidéo. Nous avons défini 2 scénarios. Un avec un flux FTP et un avec un Flux Streaming. Nous faisons l'hypothèse la plus probable d'une caméra dans la grande barrière de corail (Australie +10h) et que l'architecture du flux est organisée sur 24h. Enfin la vidéo doit être diffusée en qualité haute entre 1080 et 2100 pixels. Les vidéos seront filmés en 4k.

Flux Streaming



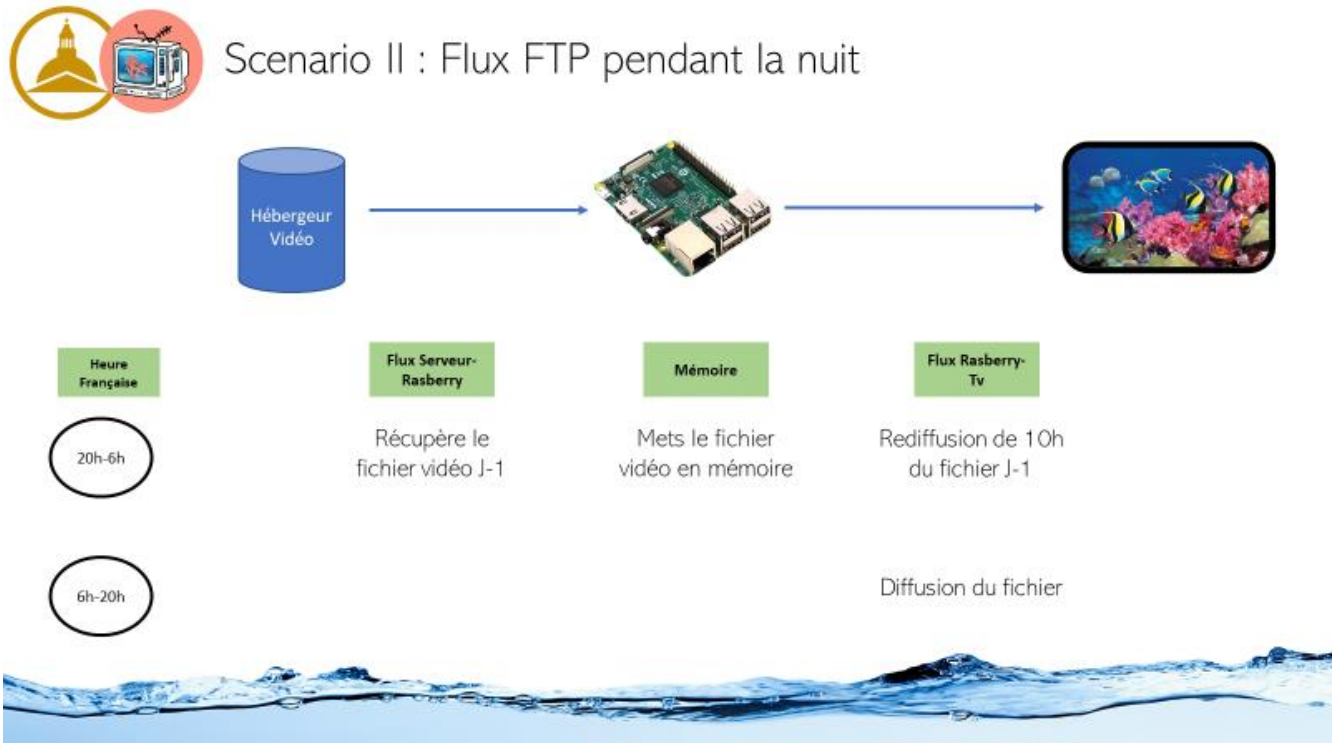
Pendant 10h, nous utilisons dans ce scénario un flux streaming pour récupérer la vidéo en continue et nous l'enregistrons. Puis, la vidéo enregistré le jour d'avant est rediffusée. Pendant 4h nous continuons de récupérer le flux streaming et diffusons le début du flux récupéré en streaming. Enfin, pendant 10h nous ne récupérons plus de vidéo et nous diffusons la fin de la vidéo récupérée en streaming.

Beaucoup d'éléments ont fait que nous n'avons pas choisi cette solution. Déjà d'un point de vue écologique. En effet, les flux streaming occupent en continue les infrastructures réseaux. Il est donc responsable de l'utilisation de beaucoup d'énergie en continue. De plus le Box devra également consommer plus pendant 14h pour récupérer le flux en continue. Il faudrait stocker 2 vidéos en 4k ce qui représente des cartes ou des disques couteux en ressources rares et polluant à la fabrication, à l'utilisation et au recyclage (quand il est fait).

Ensuite techniquement, cette implémentation pose plusieurs limites. Déjà, elle nécessite une bonne connexion internet pour récupérer sur 14h une vidéo en 4k. Des coupures réseaux, électriques ou simplement des drops de débit entraineront une qualité moindre sur certaine partie de la vidéo voir des parties de vidéo manquante qu'il faudrait gérer. Le fait d'avoir une qualité différente entre chaque box et différente dans le temps n'est pas acceptable pour OceanStream.

Enfin, d'un point de vue budgétaire, le(s) serveur(s) faisant transiter le flux à l'ensemble des box devront être puissant et nombreux pour un coup énergétique et donc financier important.

Flux FTP



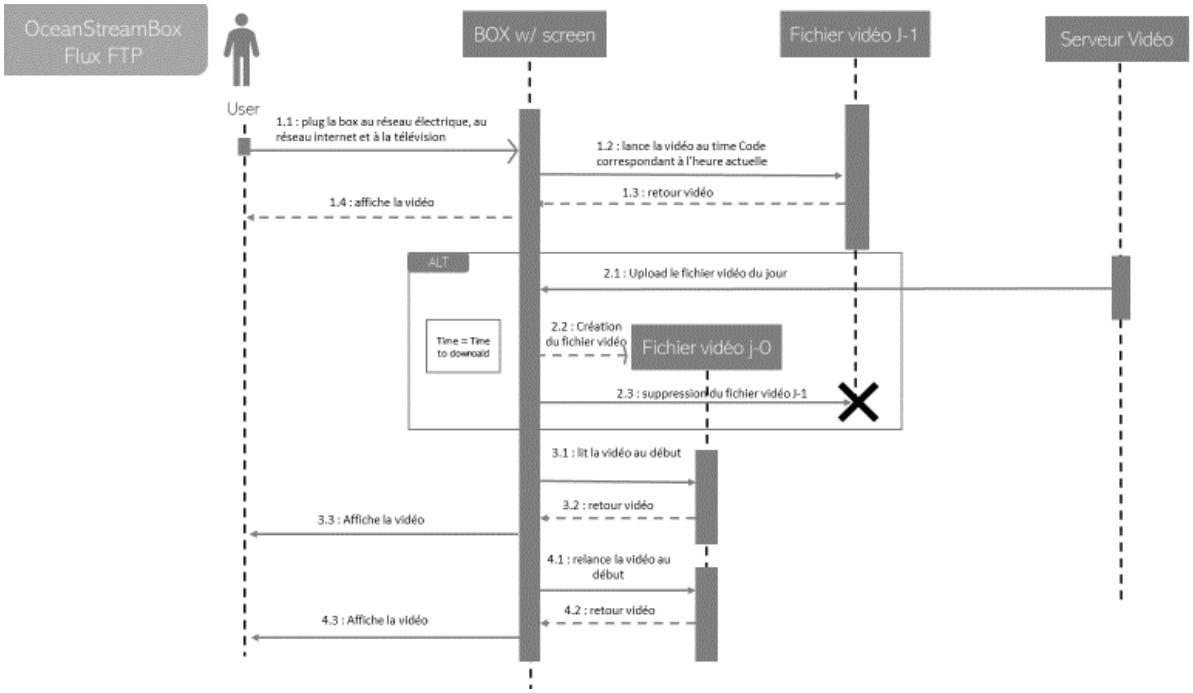
Dans ce scénario, nous récupérons le flux vidéo sous forme de fichier entre 20h et 6h (heure française) pendant que nous rediffusons la vidéo de la veille. Entre 6h et 20h, nous diffusons le fichier vidéo récupéré la nuit via FTP.

Le fait de récupérer un fichier vidéo nous permet de le compresser pour qu'il prenne moins de place sur les box. Moins de matériaux électroniques seront nécessaires. L'image étant fixe avec peu de mouvement, le fichier se prête très bien à la compression. Des tests devront être fait pour choisir le moteur et le format de compression. La récupération du fichier sera moins longue qu'avec un flux streaming et occupera moins les infrastructures réseau. De plus, récupérer le fichier dans la nuit est un choix judicieux du point de vue de l'écoconception car c'est le moment où l'énergie électrique ainsi que les infrastructures réseaux sont les moins utilisées. Nous pouvons même choisir l'horaire précise où elles sont le moins utilisées pour lancer le téléchargement FTP.

La qualité du fichier vidéo sera constante contrairement au streaming. Le téléchargement pourra être relancé en cas de coupure réseau ou électrique.

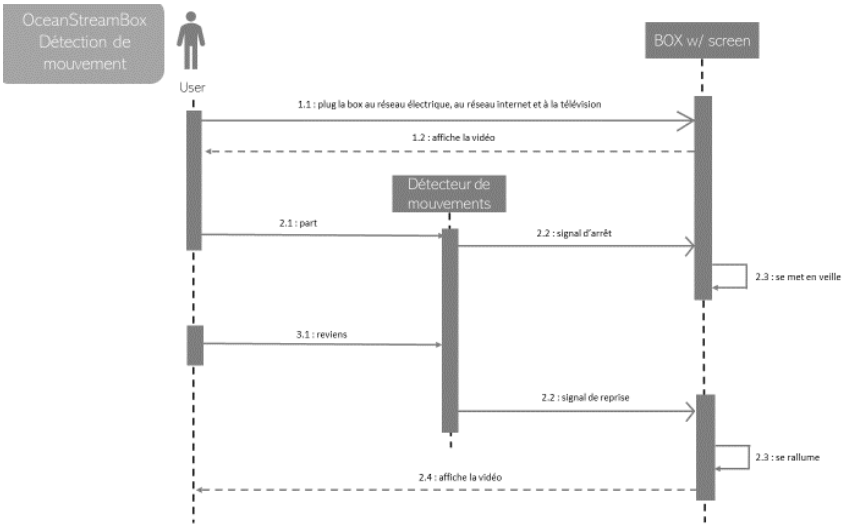
C'est pour toutes ces raisons que nous avons choisi d'utiliser un flux FTP.

Le diagramme de séquence qui décrit la récupération du fichier via un flux FTP est donc le suivant.



Détecteur de mouvement

Comme énoncé plus haut, par soucis d'économie d'énergie, nous souhaitons implémenter un détecteur de mouvement. Il sera responsable du signal de mise en veille et de la sortie de veille. Le diagramme de séquence suivant décrit son utilisation.



Choix Hardware

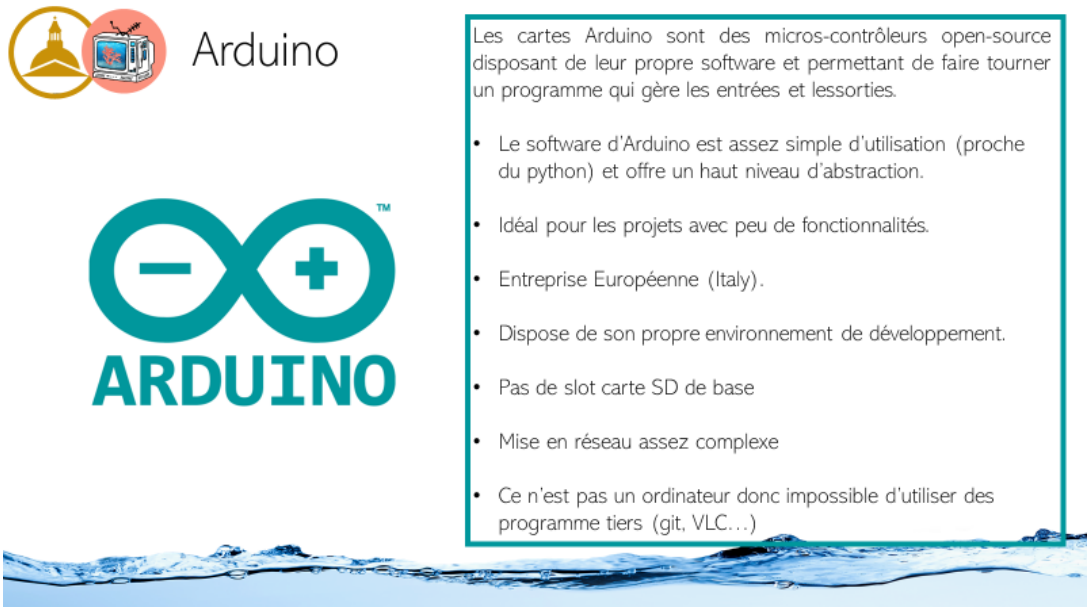
Pour choisir la technologie de carte mise dans notre box nous nous sommes fixé les critères suivants :

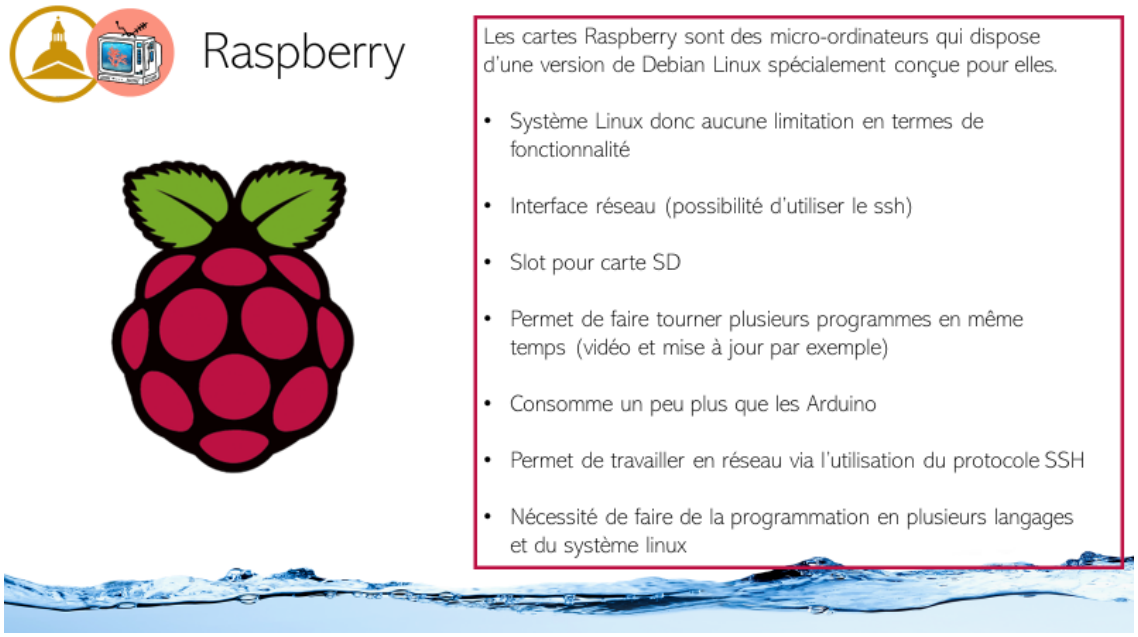
- La carte doit consommer peu d'électricité notamment via un système d'exploitation optimisé pour sa configuration hardware.
- La carte doit avoir une bonne capacité réseau pour récupérer assez rapidement le fichier vidéo quotidiennement. Elle doit disposer d'un port Ethernet pour que notre boîtier soit Plug and Play. Par ailleurs la technologie Ethernet offre un rendement débit/consommation d'énergie bien meilleur que le WiFi.
- La carte doit utiliser le moins de ressources possibles pour sa construction (métaux rares, matières premières…) pour réduire l'impact environnementale de sa construction ainsi que son recyclage.
- La carte doit pouvoir supporter un système d'exploitation supporté par git (noyaux Unix ou Microsoft) pour pouvoir mettre en place notre chaîne DevOps

Marque

Deux marques ont été confrontés. Il s'agit de Arduino et Raspberry leaders sur le marché des cartes programmables non industrielles.

Nous avons présélectionné ces deux marques car il existe beaucoup de documentation et de forums sur leurs utilisations. Il existe également de nombreux modules complémentaires que nous pourrons utiliser.





Les Raspberry PI semblent plus convenir à nos besoins du fait de la multiplicité des fonctionnalités de l'OceanBox, de notre volonté de mettre toutes les cartes en réseaux, de notre connaissance en programmation système et de notre envie de mettre en production cette box.

Choix de la gamme[1]

Raspberry propose plusieurs gammes de cartes avec chacune leurs spécificités.

En termes de prix, la PI zéro se démarque du lot. En effet à 5€, elle représente un prix de revient pour OceanStream assez intéressant. Les B/B+ et A/A+ sont nettement plus chères car plus puissantes et proposant plus de ports. La PI ZeroW offre une interface Wifi par rapport à la PI Zero pour 5€ supplémentaires.

[1] Nous utilisons le tableau communautaire de ThePiHub : https://thepihut.com/blogs/raspberry-pi-roundup/raspberry-pi-comparison-table pour comparer les gammes. Vous pouvez retrouver ce tableau en annexe

Comparaison en terme de consommation électrique


B/B+


A/A+


Zero


ZeroW

450mA-1.2A | 200-300mA | 100-350mA | 100-350mA

Encore une fois, c'est la PI Zero et ZeroW qui se démarquent avec une consommation en moyenne plus faible que les autres cartes.

Comparaison en terme d'impact environnementale du Hardware


B/B+

C'est la carte qui utilise le plus de métaux notamment à cause des nombreux ports


A/A+

Un peu moins consommatrice en ressource


Zero


ZeroW

Etants les plus petites carte Raspberry sur le marché, elle utilisent asse peu de métaux et de ressources dans leur fabrication

Enfin, en termes d'utilisation des ressources à la construction, la PI Zero et ZeroW se démarquent encore dues à leurs petites tailles et leur faible nombre de ports.

PI A

- Moins cher de 10€ que le modèle B+
- Moins consommatrice en électricité
- Plus petite
- Toujours une quantité de port exaltable (USB, SATA, pins, HDMI)

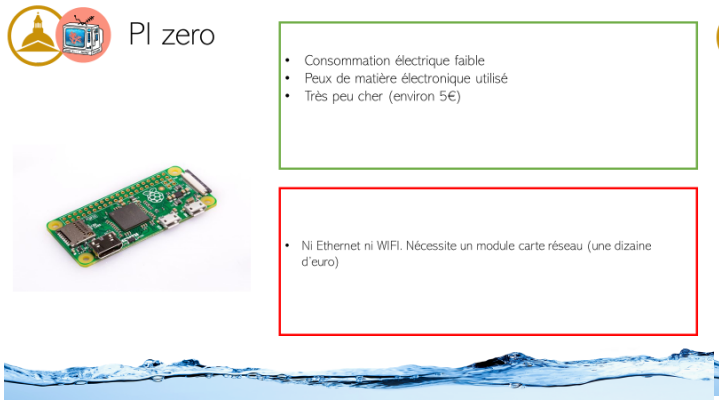
- Pas de port Ethernet. Nécessite une module Ethernet ce qui rends la carte plus cher et plus gros que le modèle B+

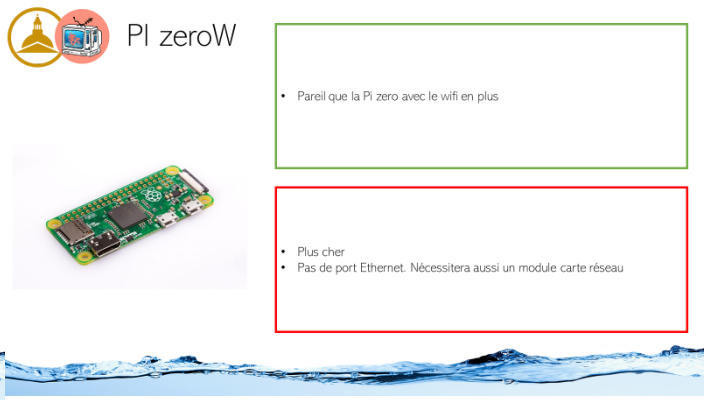
PI B+

- Carte la plus puissante
- Choix de la quantité de RAM (1, 2, 3 ou 4GO)
- Nombreux ports (USB 2.0, USB 3.0, Ethernet, HDMI, SATA, Pins…)
- Port Ethernet nécessaire à notre projet

- Carte la plus consommatrice en électricité
- Beaucoup de ports inutiles dans notre cas
- Puissance très importante par rapport à nos besoins
- Prix de 40€ assez conséquent

Nous ne choisirons donc pas la PI A/A+ et B/B+ qui sont trop puissantes et donc trop consommatrices en ressources et en énergie par rapport à nos besoins.





Reste donc à choisir entre la PI Zero et sa sœur la PI ZeroW. Il se trouve que les deux cartes ne disposent pas de port Ethernet. Il faudra donc leur ajouter un module Ethernet (prix d'environ 10€). Il n'est donc pas nécessaire d'avoir une interface Wifi. Nous choisissons donc la Raspberry avec la plus petite configuration : la PI Zéro

Après s'être réunis avec Adrien, la PI Zéro W a été choisi. En effet, dans un esprit AGILE, Adrian a souhaité qu'une interface WiFi soit disponible au niveau hardware dans l'éventualité qu'une connexion WiFi de la Box soit proposés dans des Sprints ultérieurs.

OceanStream et la MIAGE Sorbonne s'associent pour créer

L'OceanBox

Un projet AGILE et DevOps écoconçu

PARTIE ARCHITECTURE

Abdel Benamara | Julien Doujet | David Ekchajzer | Mathieu Ridet

08/05/2020

Table des matières

Architecture de l'application et de l'infrastructure

Devops

Notre infrastructure DevOps se base sur des technologies à l'état de l'art pour le déploiement java. GitHub est utilisé comme repository des fichiers sources. Nous utilisons Jenkins comme chef d'orchestre de notre chaine et nexus comme repository des artefacts (binary). En bout de chaîne, nous utilisons la fonction pull d'Ansible qui permet d'inverser l'architecture classique DevOps. En effet Ansible est communément utilisé en mode push (le serveur Devops envoie les instructions de modifications aux machines cibles). Comme nous ne pouvons pas connaître par avance les adresses IP de nos OceanBox, nous ne pouvons pas utiliser cette architecture. C'est pourquoi nous utilisons le mode pull où les machines cibles demandent à un intervalle de temps régulier les mises à jour à effectuer.

Pour le moment deux chaînes DevOps coexistent :

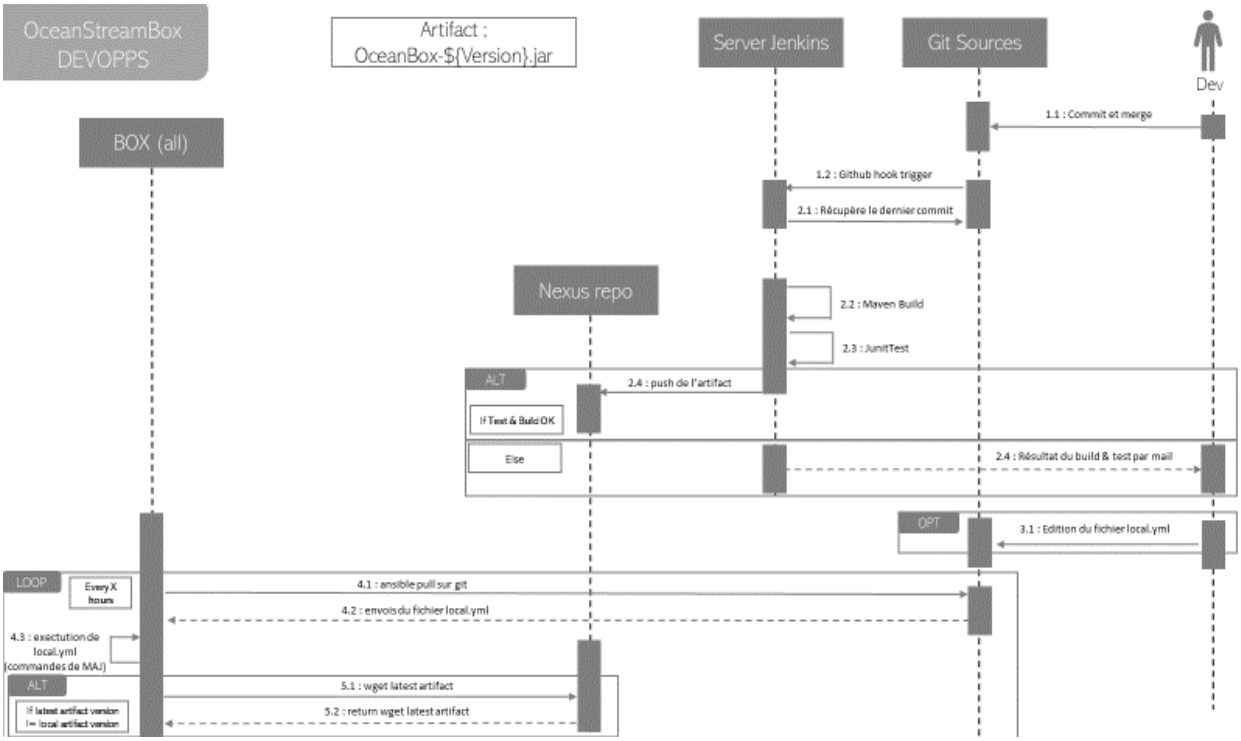
- La chaîne de Dev branchée aux OceanBox de développement et de recette.
- La chaîne de Production branchée aux OceanBox de production.

Ainsi la recette ainsi que le déploiement sont facilitées même à distance.

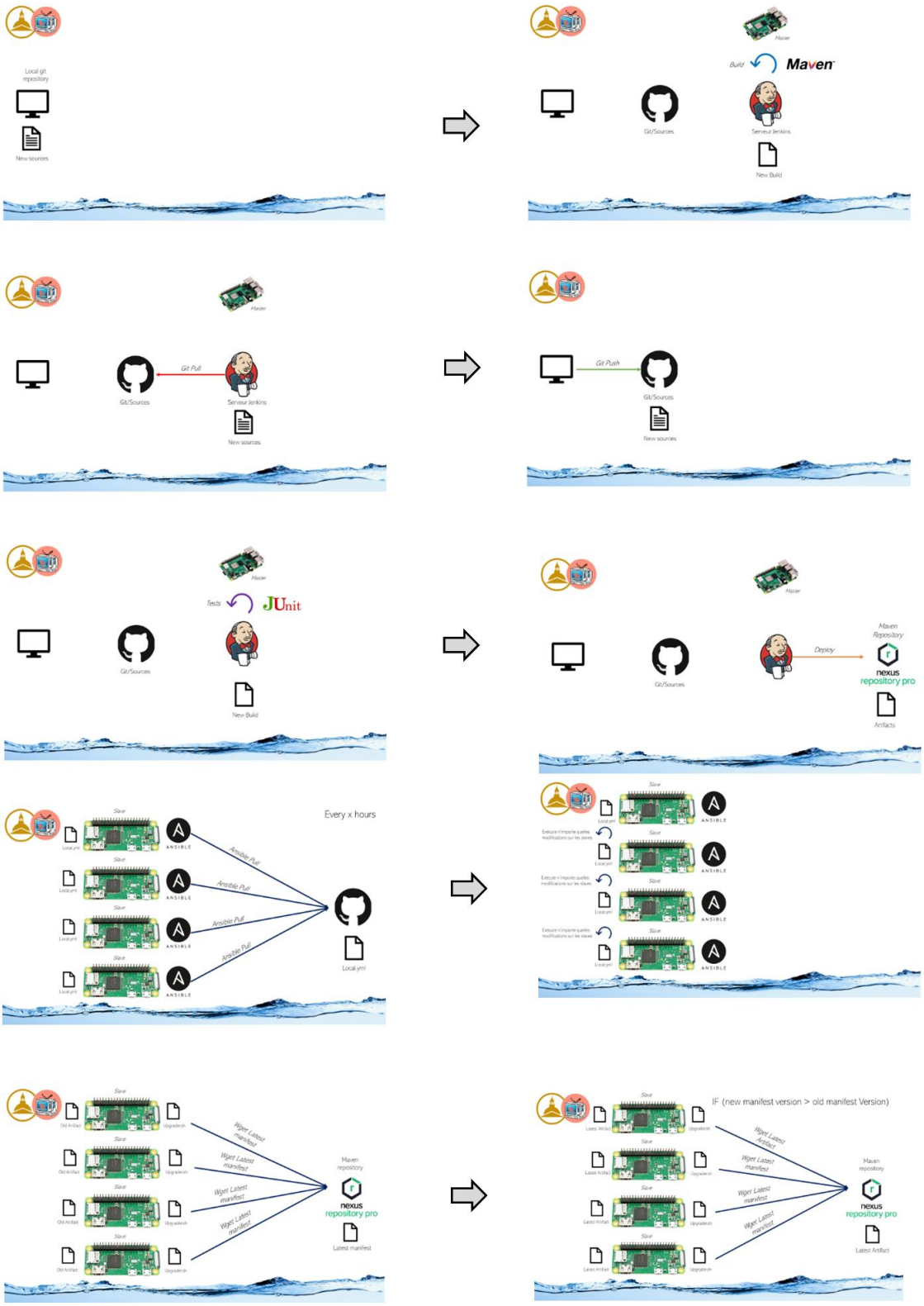
Vous trouverez ci-dessous deux schémas de notre chaîne DevOps sous la forme d'un diagramme de séquence ainsi que sous la forme d'un powerpoint (ppt) plus adapté à la présentation au client.

Le serveur Jenkins ainsi que le repo nexus sont hébergés sur un serveur Raspbian installé sur Raspberry 4.

Via un diagramme de séquence



Via un schéma ppt



Téléchargement FTP

Les vidéos de l'OceanBox sont récupérées de manière journalière sur un serveur FTP. Ce serveur est installé sur le cloud. Il s'agit d'un serveur NGINX, le serveur ftp est géré par VSFTPD la gestion des ports est assurée par ufw. Les échanges ftp sont sécurisés de bout en bout par le protocole SSL.

Les téléchargements sont effectués de nuit pour rationaliser l'utilisation des infrastructures réseaux qui sont peu utiliser de nuit. Cela permet également d'utilisées l'énergie électrique quand la demande est la plus basse pour ne pas surcharger le réseau électrique.

Le téléchargement est un événement trigger à une heure fixe calculée à chaque allumage de la box. Le cycle de vidéo d'une journée ayant une durée de 12h ou moins, les vidéos sont potentiellement jouées plusieurs fois dans la journée. L'heure de téléchargement correspond à la dernière lecture du cycle de vidéo avant l'heure de réveil de l'utilisateur. Ainsi le nouveau cycle de vidéo est lancé dès le réveil de l'utilisateur.

La capacité de la carte SD ne permettant pas de stocker simultanément les vidéo J et J+1 nous avons dû opter pour du stockage par paquet. Concrètement le cycle de vidéo d'une journée est divisé en une multitude de paquets qui sont joués les uns après les autres. Pour le téléchargement, nous récupérons les paquets un par un depuis le serveur ftp. Pour chaque paquet téléchargé son paquet équivalent de la veille est supprimé. Nous devons nous assurer qu'un paquet a été joué pour être supprimé (pour ne pas avoir à lire un paquet ayant été supprimé) c'est pourquoi le thread de téléchargement est synchronisé avec le « main » thread (thread de lecture).

Schémas ppt d'un téléchargement pour un cycle de vidéo de 12h sans et avec la veille

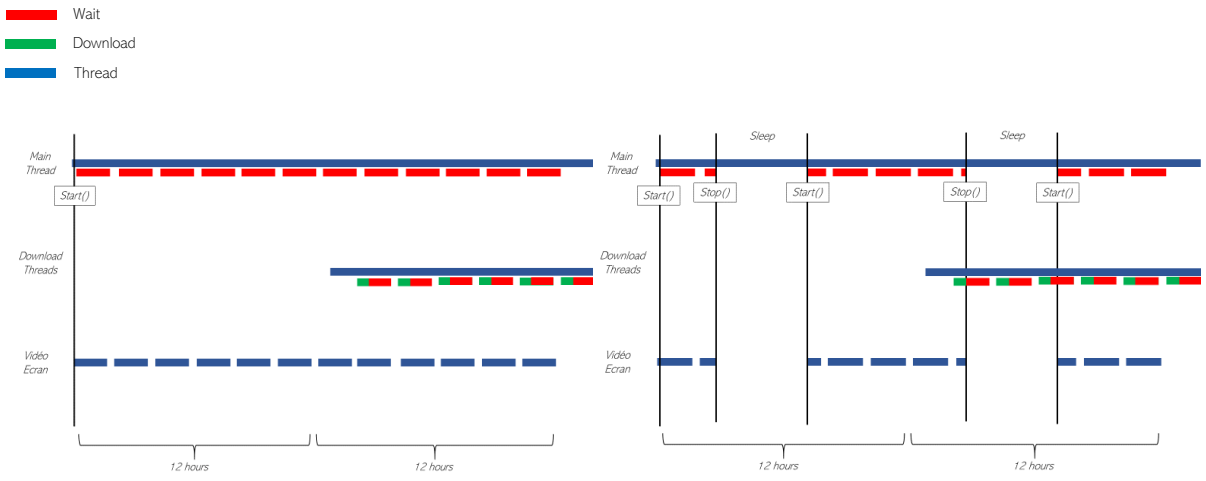


Diagramme de séquence

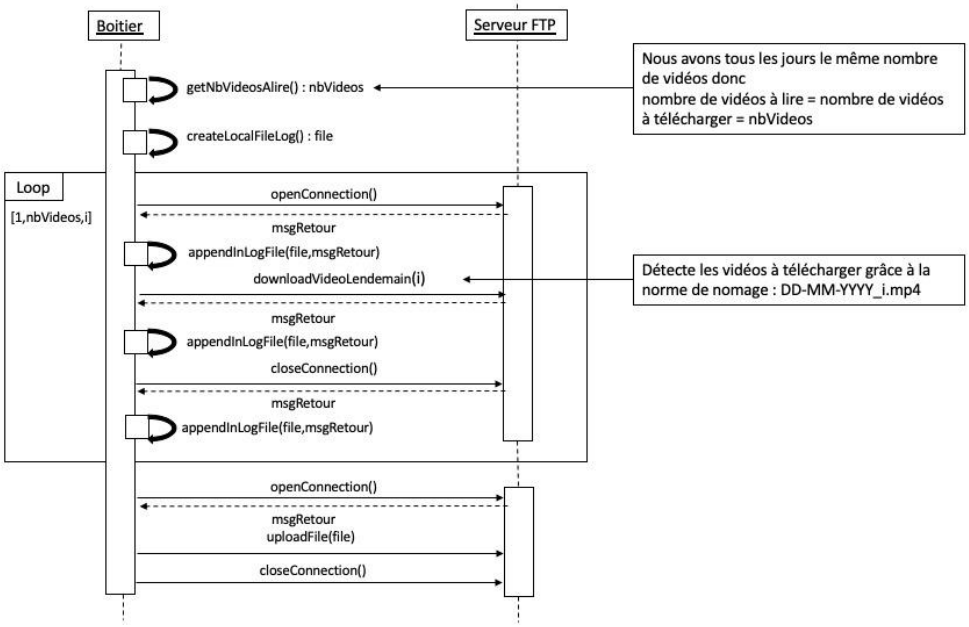
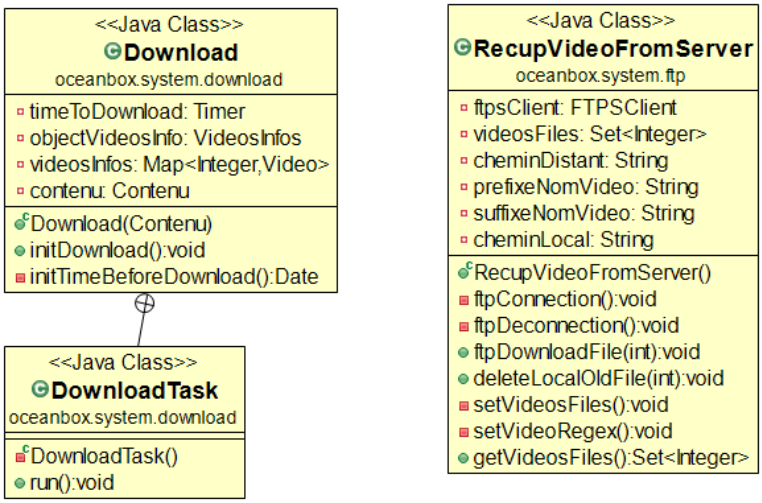


Schéma UML des classes de téléchargement



Les logs sont remontés par le biais de l'architecture ftp de manière journalière.

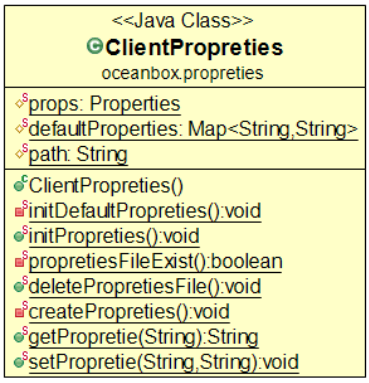
Paramètres et BDD

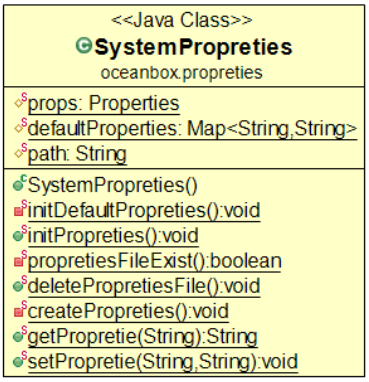
Les OceanBox sont paramétrées à deux niveaux :

- Un paramétrage système qui nous permet de définir des constantes identiques à toutes les OceanBox comme les connexions au serveur ftp et à la base de données ou encore les chemins d'accès aux fichiers de logs et aux vidéos. Ainsi, les modifications mineures sont facilitées car nous n'avons pas besoin de changer les paramètres mais seulement le code source.

- *Un paramétrage client* qui permet de personnaliser le comportement de l'OceanBox en fonction de la volonté des clients. Le client peut ainsi choisir son heure de réveil à laquelle les vidéos de la journée suivante débuteront. Il peut décider de garder ou de retirer la veille ainsi que paramétrer le temps avant le déclanchement de la veille.

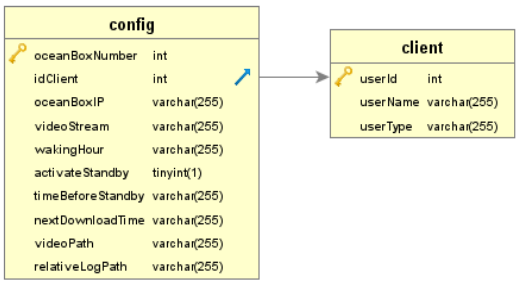
Schémas UML du paramétrage

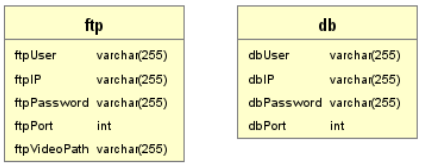




Ces paramètres sont récupérés depuis la base de données à chaque allumage de l'OceanBox ainsi qu'avant chaque téléchargement ftp des vidéos. Le moteur de base de données utilisé est MariaDB (MySQL). Une future interface web de paramétrage devra être créée par OceanStream pour alimenter cette base de données.

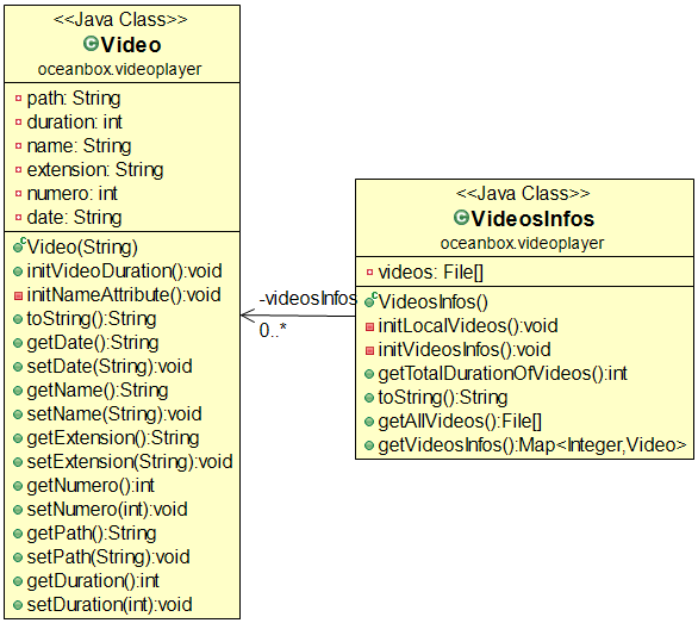
Schémas de la base de données





Initialisation des vidéos

Chaque contenu est divisé en plusieurs vidéos (paquets) qui se suivent pour pouvoir réduire le risque que le téléchargement soit interrompu et de permettre de ne pas surcharger la bande passante.



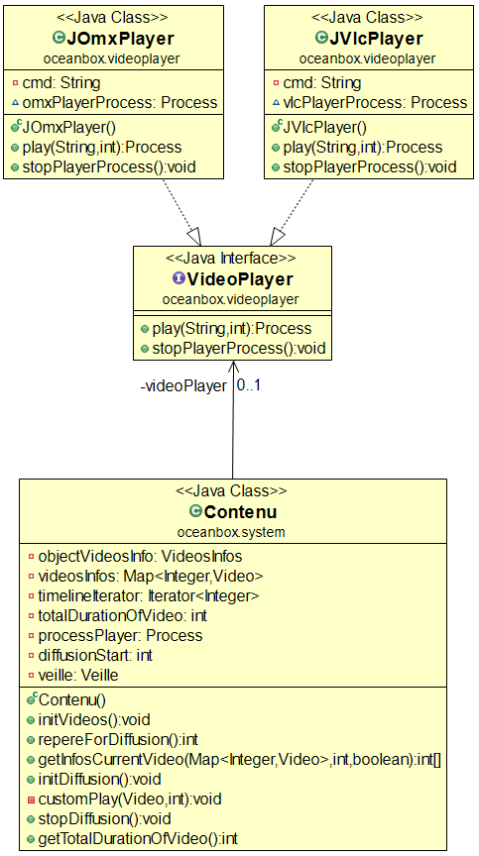
Au lancement de l'OceanBox ou à la fin d'un téléchargement d'un nouveau contenu, le système a besoin de connaître les vidéos qui le compose. Pour ce faire, nous appelons la classe VideoInfos qui initialise un objet vidéo pour chacune des vidéos en mémoire.

Chaque objet vidéo récupère les métadonnées des vidéos qu'il garde en attribut. Deux manières sont utilisées pour récupérer ces données :

- Le nom des vidéos étant normalisés « jour-mois-année_numéro.extension », nous découpons le nom de la vidéo à l'aide d'expression régulière pour récupérer les informations contenues dans le nom.
- Pour la durée de la vidéo, aucune classe utilitaire fonctionnant sur une architecture ARM n'existe pour la récupérer. Nous avons donc créé une classe wrapper utilisant le programme MediaInfo disponible pour Raspbian, mac, linux et Windows.

Ces objets ainsi créés sont nécessaires au fonctionnement de la lecture et du téléchargement.

Contenu & VideoPlayer



Afin de tester l'application depuis nos IDE, tel qu'Eclipse, avant de l'exécuter sur nos raspberry, nous avons décidé d'utiliser le pattern Strategy. Celui-ci permet de déclarer des classes qui vont effectuer les mêmes actions avec des méthodes différentes. Plus concrètement cela signifie dans notre cas que nous allons pouvoir lire les vidéos avec le lecteur VLC sur nos ordinateurs munis des systèmes d'exploitation Windows, Mac OS et Linux, et que nous pourrons travailler avec OmxPlayer à la place de VLC sur le système Raspbian. Pour ce faire, nous avons une interface VideoPlayer et deux classes qui l'implémentent, JVlcPlayer et JOmxPlayer. Il nous suffit d'instancier l'une ou l'autre dans notre classe Contenu qui gère la lecture des vidéos : les machines de dev et de prod implémentent JOmxPlayer, le code de développement sur pc implémente JVlcPlayer.

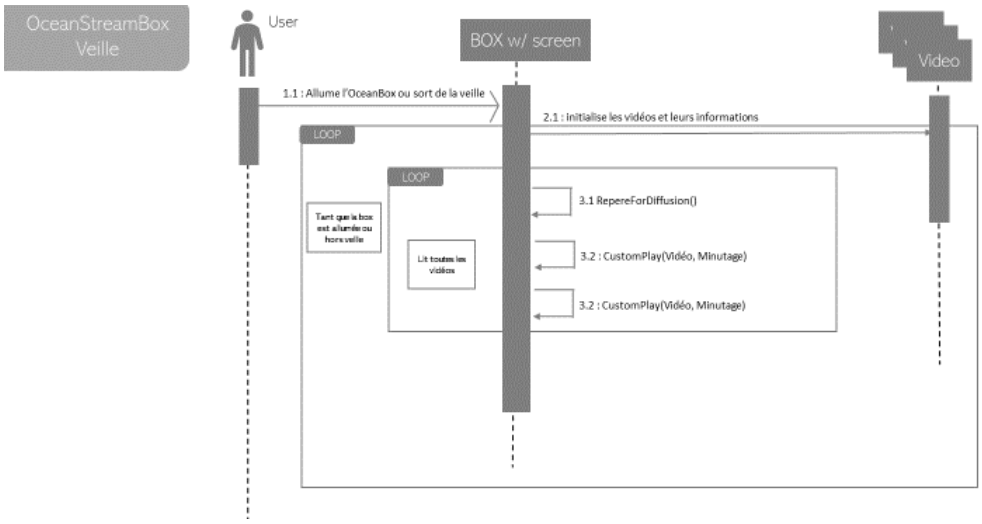
La lecture des vidéos est synchronisée sur l'heure de réveil renseignée par l'utilisateur. Ainsi chaque nouveau contenu téléchargé est joué pour la première fois à partir de l'heure de réveil.

Pour garantir la synchronisation, comment déterminer quelle vidéo lancer et à quel minutage quand le boîtier s'allume ou sort de veille ? La méthode RepereForDiffusion de la classe Contenu tient ce rôle. Elle récupère l'horaire défini par l'utilisateur ainsi que la durée totale du contenu quotidien. Ces deux informations sont utilisées pour déterminer la vidéo à lancer ainsi que son minutage. On récupère dans un premier temps la différence entre l'heure actuelle et l'heure de réveil modulo le temps du contenu. Ce modulo correspond à la durée à laquelle lancer le contenu. Il nous reste qu'à parcourir les vidéos pour déterminer quelle vidéo lancer à quel minutage.

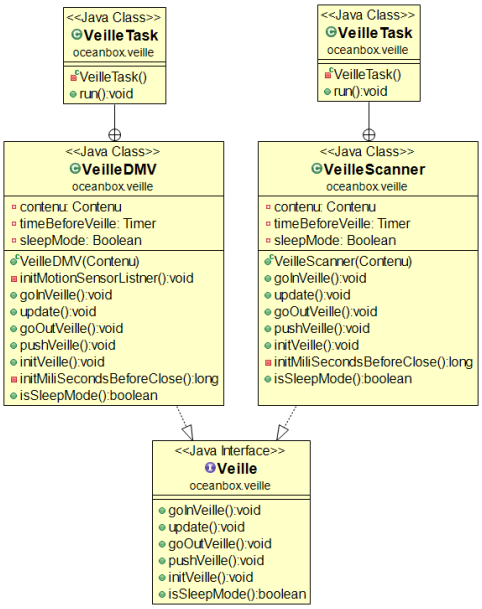
$$(NOW - HeureDeRevei)\%DureeContenu$$

La multitude de vidéos qui constitut le contenu du jour doivent être lues les unes à la suite des autres. Pour ce faire, celles-ci respectent toutes la même convention de nommage, débutant par la date au format « jour-mois-année » et se terminant par le numéro de l'ordre de lecture sous la forme « _nombre ». Ainsi, la méthode CustomPlay de la classe Contenu peut facilement savoir quelle doit être la prochaine vidéo à lire. Notre fonction est récursive. Elle s'appelle elle-même une fois la lecture d'un vidéo terminée. Dans notre cas, notre récursivité est infinie. En effet, si l'on ne touche à rien, les vidéos vont se lire les unes à la suite des autres indéfiniment.

Diagramme de séquence



Veille

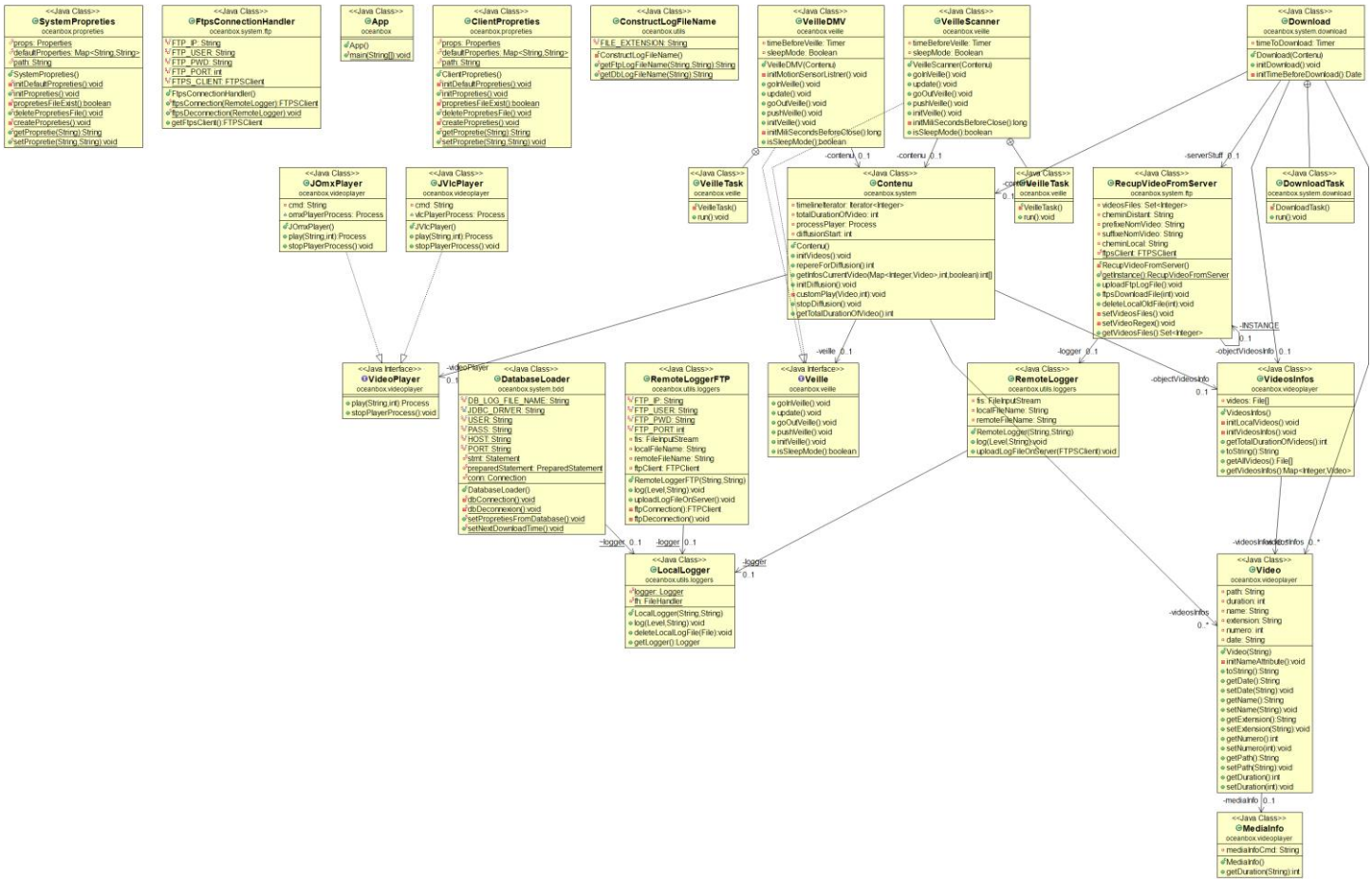


Après un certain temps d'inactivité nous souhaitons que l'OceanBox puisse passer en veille. La veille est déclenchée après un certain temps d'inactivité appelé *timeBeforeStandby* décidé par l'utilisateur. Pour ce faire, un évènement est déclaré qui met en veille l'OceanBox après la durée *timeBeforeStandby* paramétrée par l'utilisateur. Pour repousser la veille, quand un évènement est trigger à l'entrée écoutée, l'évènement de veille est annulé et un nouveau est créé avec la durée *timeBeforeStandby*.

A la mise en veille, les vidéos sont arrêtées et le lecteur vidéo est KILL. A la sortie de la veille, les vidéos sont relancées. Le minutage du lancement est déterminé par la méthode RepereForDiffusion développé plus tôt.

De la même manière que pour notre lecteur vidéo nous avons utilisé le pattern Strategy pour implémenter la veille de deux manières différentes. Une veille est simplement à l'écoute de l'entrée standard (la console) il s'agit de la VeilleScanner utilisant la classe Scanner pour écouter n'importe quelle entrée sur la console. L'autre veille est à l'écoute des GPIO de la Raspberry auxquelles sont connectés un détecteur de mouvement.

UML de l'application JAVA



Questionnaire

N°	Question	Réponse
1	Votre architecture vous permettrait-t-elle de supporter facilement une autre langue utilisateur (français et anglais par exemple) ?	*Ne s'applique pas*
2	Evaluez la difficulté du passage de votre application web à du mobile.	*Ne s'applique pas*.
3	La modularité est un critère important de qualité des architectures (Un composant ou classe n'a qu'un seul objectif). Les composants de votre architecture respectent-ils ce principe de modularité ?	*Oui*
4	Des éléments de votre architecture seraient-ils réutilisables dans un autre projet ?	*Oui*
5	Votre architecture permettrait-elle de changer de SGBD facilement ?	Oui
6	Si vous deviez changer totalement la réalisation de l'interface graphique, y-aurait-il des composants ou classes inchangé(e)s ?	Oui
7	Quels seraient les effets d'un éventuel changement dans le schéma de la base de données (ajout d'une nouvelle table ou nouvel attribut, changement du nom d'une table ou d'un attribut, changement sur un type d'attribut, sur une contrainte...) dans l'application ?	L'ajout de table est envisagé pour supporter un site web (hors cadre de notre projet) permettant aux utilisateurs de pouvoir paramétrer leur box avec une interface. L'application web pourrait même avoir des informations sur leur box (leur consommation par exemple). Il n'y a donc pas de contrainte sur la modification du modèle de données.

OceanStream et la MIAGE Sorbonne s'associent pour créer

L'OceanBox

Un projet AGILE et DevOps écoconçu

PARTIE RETOUR

Abdel Benamara | Julien Doujet | David Ekchajzer | Mathieu Ridet

08/05/2020

Table des matières

Limites techniques et axes d'amélioration

Première implémentation

Notre premier choix d'architecture s'est porté vers l'outil graphique de java : JavaFX. Nous l'avons choisi car il nous a été enseigné cette année et nous paraissait approprié. Le programme de l'OceanBox développé avec JavaFX lit les vidéos grâce à sa bibliothèque MediaPlayer. Pour cela, il génère au sein de son conteneur principal, aussi appelé Stage, un composant graphique appelé MediaView qui peut afficher les éléments des différentes bibliothèques Media de JavaFX (vidéo, musiques, GIF…). C'est la partie graphique de l'application qui a été une source de blocage. En effet, celle-ci devait-être exécutée sur un système d'exploitation recommandé pour les raspberry : Raspberry Pi OS ou anciennement Raspbian. Malheureusement, une incompatibilité a rendu l'utilisation de cette technologie impossible.

N'ayant pas accès à ces micro-ordinateurs ou des machines virtuelles suffisamment proches durant la phase de développement, la grande majorité des tests a été effectué grâce à la JVM (Java Virtual Machine) d'Eclipse sur nos machines. Cependant, quand nous avons testé notre programme sur une Raspberry pour la première fois, une erreur JavaFX nous a été retournée. La première erreur à laquelle nous avons été confrontés réside dans la difficulté d'installer la version 8 de Java avec les bibliothèques JavaFX car la plupart des JDK (*Java Development Kit*) ne les contiennent pas compte tenu de leur faible taux d'utilisation. Une fois ce problème résolu, il était toujours impossible de lancer l'archive Java de l'application en raison d'une incompatibilité qui nous était inconnue jusque-là. A la suite de recherches sur des forums traitant du langage Java et du système Raspbian, nous avons finalement appris que les composants graphiques générés par les bibliothèques de JavaFX ne sont pas supportés par les processeurs ARM utilisés par les raspberry – à cause notamment de frictions entre Oracle (éditeur de JAVA) et les constructeurs de processeurs ARM - . Cela signifiait donc qu'il nous était inenvisageable de continuer à développer l'application à l'aide de ces bibliothèques. Il a fallu retirer tout élément lié à la librairie JavaFX (80% de notre application) pour s'assurer que notre logiciel soit exécutable sur des processeurs ARM. Cela impliquait de soustraire toutes les fonctionnalités supplémentaires et non indispensables que nous avions ajoutées au fur et à mesure du développement pour donner suite aux demandes de notre client (bandeau d'information, horloge graphique…).

En outre, nous utilisions une méthode implémentée par JavaFX au sein de sa Class Media qui nous permettait de récupérer la durée de chacune des vidéos. Cette information est indispensable pour l'application car elle nous offre la possibilité de calculer le temps exact auquel la vidéo doit débuter à l'écran quand le programme est exécuté pour la première fois ou quand il sort de son mode veille. Par ailleurs, elle permet de savoir combien de temps de lecture il reste afin de synchroniser correctement le téléchargement et le passage au jour suivant en termes de contenu vidéo. Mais comme dit précédemment, nous avons dû nous séparer de cet outil, et même si récupérer la durée d'un fichier audio ou d'une vidéo peut sembler basique, et donc simple, ce n'est pas le cas en réalité. Pour récupérer cette donnée parmi d'autres il faut avoir recours à un programme ou des bibliothèques à part entière. Après avoir lu quelques articles sur internet, notre choix s'est porté sur Xuggler, une Class Java permettant la gestion de fichier et notamment la lecture des meta-data comme la durée dans le cadre de fichiers multimédias. Ce dernier a été importé en l'ajoutant aux dépendances Maven. Cette implémentation répondait entièrement à nos attentes jusqu'au moment des tests sur raspberry. Là encore, une incompatibilité cette fois avec

le système d'exploitation Raspbian à laquelle nous n'avions pas pensé et nous avons fini par la découvrir lors de nos tests. Heureusement, nous avons su rebondir.

Tant pour la partie graphique que pour la récupération des temps de vidéos, nous avons trouvé les implémentations adéquates grâce aux class Wrapper. Nous les avons découvert plus tôt dans l'année lors de l'installation de notre Chaine DevOps. En effet, en installant Nexus (repository d'Artifact) sur notre Raspberry nous avons dû utiliser une class wrapper qui permettrait de contrôler depuis java un programme externe en « l'enfermant » (le « wrappant ») dans une classe. Pour la lecture des vidéos nous avons "wrappé" OmxPlayer et VLC. Pour la récupération du temps de vidéo, nous avons utilisé le programme MediaInfo qui possède une version spécialement conçue pour les architectures ARM raspberry. Les class wrapper sont utilisées actuellement dans notre code.

Finalement, nous avons une application Java qui exécute un processus OmxPlayer pour lire le contenu à l'écran et faisant différents calculs grâce aux durées des vidéos récupérées en exécutant le programme MediaInfo depuis notre programme java. Notre application est fonctionnelle et répond aux besoins primaires de notre client. Cependant, ce dernier nous avait demandé des ajouts et améliorations quand nous travaillions avec JavaFX pour communiquer des informations supplémentaires au client. Par exemple telles que la température de l'eau, sa salinité ou encore la profondeur d'enregistrement du clip. Mais pour afficher ces éléments sous la forme d'un bandeau déroulant d'information, comme c'était le cas dans notre première implémentation, nous devrions utiliser une autre technologie compatible. De ce fait, le première limite de notre implémentation actuelle que l'on peut relever est celle d'affichage graphique avec des outils Java.

Axes d'améliorations

La limite expliquée plus haut se révèle être en réalité un axe d'amélioration possible car il faut trouver le moyen de faire apparaître les informations supplémentaires comme initialement implémentées. Pour cela, il faudra utiliser des composants graphiques compatibles avec Raspberry Pi OS et les architectures hardware ARM pour ne pas avoir de mauvaises surprises, comme celle que nous avons eu avec les éléments provenant de Java.

En outre, durant le développement nous nous sommes principalement concentrés sur le fait de rendre rapidement quelque chose de fonctionnel et nous avons donc peu développé certains aspects du projet pour gagner du temps.

Parmi eux, la sécurité :

- Tout d'abord la sécurité des données. En effet les identifiants de connexion à la base de données et au serveur FTP sont lisibles en clair dans les fichiers propreties générées par notre programme. Bien que le SSL soit implémenté, les identifiants peuvent être récupérés aisément par quelqu'un ayant une OceanBox en main même si ces identifiants n'ont qu'un droit de lecture. De ce fait, il faudrait mettre en place un système d'autorisation de connexion aux services précédents en fonction de l'adresse IP émettrice de la requête d'authentification mais cela poserait le problème de devoir connaître à l'avance toutes les ip des OceanBox. On pourrait également envisager de crypter les fichiers de propreties ou de crypter les données qui y sont inscrites pour rendre la tâche de la lecture plus ardue à un individu qui ne possèderait pas la clé de décryptage. Cependant les données en base

ne sont pas très sensibles et nous recommandons de ne pas y mettre de données personnelles.

- Ensuite la sécurisation face aux vols de vidéos. En effet, il est aujourd'hui possible pour quelqu'un ayant l'OceanBox entre les mains de récupérer sur un disque dur les vidéos qui appartiennent à OceanStream. La sécurisation des contenus face aux vols est extrêmement difficile voire impossible car il existera toujours une solution (filmer son écran par exemple). Nous n'avons pas développé davantage cette partie pour se consacrer à notre mission principale et pouvoir l'aboutir.

La lecture des vidéos par paquets rend le passage d'un paquet vidéo à l'autre un peu saccadé. Il faudrait optimiser cette partie car elle peut rendre l'utilisation de l'OceanBox moins fluide et donc moins agréable pour les utilisateurs.

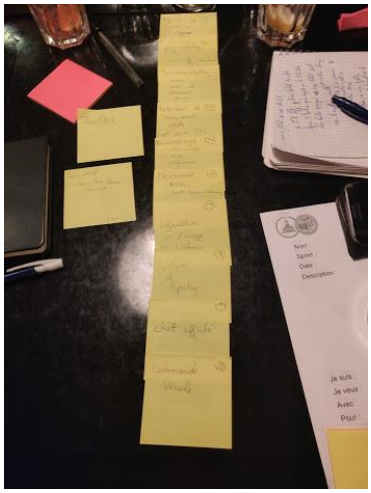
D'autres paramètres pourraient être proposés aux utilisateurs notamment la lecture de différents flux vidéo.

Un tableau de bord pour OceanStream pourrait être proposé pour suivre l'utilisation de leurs boîtes.

Méthodologie

Avantage de notre méthodologie

Nous avons débuté notre projet avec une méthodologie voulu innovante. Nous avons pu mettre en pratique une partie de cette méthodologie.



- La méthode AGILE que nous avons utilisée nous a permis de montrer nos avancés au fur et à mesure à notre Sponsor OceanStream. Nous avons ainsi évité plusieurs fois de nous orienter vers un rendu ne correspondant pas aux attentes. En effet, nos points mensuels permettaient de recadrer le projet pour que celui-ci se rapproche le plus possible des demandes du sponsor. De plus, la méthode AGILE a permis au sponsor de prendre davantage part au projet et de comprendre les limites auxquelles nous étions confrontés et d'accepter plus facilement nos retards. Ceux-ci étant la conséquence des différentes

difficultés rencontrées. Le découpage du projet en User Stories a également été un moyen adapté à notre projet. En effet, nous savions que toutes les demandes du sponsor ne pourraient pas être réalisées compte tenu à la grandeur de l'objectif. Découper le projet en US nous a permis de prioriser les éléments du projet et de discuter pour estimer la difficulté de chacune d'entre elles afin que nous puissions tous anticiper la durée des différents éléments à développer. Ainsi, le produit fini est supposé être le plus complet au regard des moyens alloués (principalement en termes de délais).

- L'éco-conception nous a permis de réfléchir aux choix techniques avec un regard différent de ce qui nous est enseigné en cours et en entreprises. En effet, le budget, le délai et la qualité (triptyques classiques de la gestion de projet) sont supplantés par l'impact environnemental du Système d'Information (SI) développé. Les choix Hardware comme les choix de technologie de téléchargement ont été choisis pour leurs faibles impacts sur l'environnement en comparaison à d'autres solutions alternatives. Souvent, ces choix ne coïncidaient pas avec l'état de l'art comme pour le téléchargement FTP qui aurait été remplacé par un flux streaming si nous n'avions pas souhaité éco-concevoir notre OceanBox. Ainsi, effectuer ce projet nous a permis de nous poser des questions sur la direction prise par l'innovation informatique. L'évolution du secteur tend à s'appuyer sur la hausse des performances des composants hardwares et de l'augmentation des débits internet à travers le monde malgré leurs impacts environnementaux et sociaux. Le regard apporté par l'éco-conception croise souvent celui de l'optimisation algorithmique. En effet un SI qui consomme moins de ressources pour effectuer une tâche réduira de fait sa consommation énergétique et allongera en moyenne sa durée de vie. Les cours d'algorithmique de M. Diaz nous ont donc été très utiles pour la phase de développement.

- La méthode DevOps nous a pris beaucoup de temps à appréhender puis à appliquer. En effet, il existe de nombreux outils complémentaires ayant chacun leurs spécificités que nous devions comprendre. Nous avons pu nous aidé de plusieurs documentations. Malheureusement, notre architecture DevOps est assez particulière et aucun article ne documentait une chaine complète iso à la nôtre. En effet, les architectures DevOps sont usuellement utilisées en mode push. Après le build du programme, celui-ci est push sur la ou les machines cibles souvent des serveurs hébergeant un webservice. Le client peut requêter le même serveur avant et après le push, le déploiement n'a eu aucune incidence sur lui. Dans notre cas, les serveurs cibles sont nos OceanBox potentiellement nombreuses, éteintes et dont les IP sont inconnus et possiblement variables, car le client en fait ce qu'il souhaite (changer de réseau, débrancher…). Il nous est donc difficile de déployer depuis un serveur DevOps centralisé. Un article RedHat nous as permis de découvrir une architecture DevOps en mode Pull. Cette fois ce sont les machines cibles qui demandent un déploiement par une requête pull. Nous avons donc utilisé notre propre script d'update exécuter tous les jours pour déployer les nouvelles versions.

Limites de notre méthodologie

Mais notre méthodologie était assez conséquente et pouvant être "lourde" à suivre et nous avons dû l'alléger parfois avec regret au moment de l'appliquer :

- La méthode AGILE nécessite de planifier un grand nombre de réunion à intervalle très régulier. Cet investissement temporel nous paraissait être du temps perdu face aux délais imposés. Nous avons donc condensé toutes les cérémonies AGILE dans une réunion mensuelle. De plus, le confinement nous a empêché de suivre parfaitement la priorisation des US ainsi que d'effectuer les tests réguliers puisque nous n'avions pas accès aux composants hardware. Nous nous sommes rendu compte que pour utiliser parfaitement la méthode AGILE il faut que tous les acteurs aient beaucoup de temps ce qui n'était pas notre cas.

- L'éco-conception a dirigé nos choix. Cependant, nous aurions aimé nous appuyez sur des recherches statistiques plus poussé et ensuite sur des études plus concrètes de la consommation énergétique et matériel de notre SI plus particulièrement. Cependant nous avons butté sur plusieurs problèmes notamment le manque de temps (que nous avons préféré investir dans l'élaboration d'un produit fonctionnel), notre manque de connaissance et enfin le manque d'études statistiques sur les impacts comparés de technologies et d'architectures. Les seules études que nous avons trouvées présentaient les impacts globaux de l'informatique et quand elle était plus spécifique (notamment l'étude de l'ADEME sur l'impact d'un Mo en flux streaming) elles présentaient des moyennes à l'échelle mondial qui ne sont pas représentatives en France. En effet, l'électricité utilisée en France qui utilise principalement des Centrales nucléaire est très peu carbonée et donc les SI qui utilisent cette énergie le sont également moins. Nous nous sommes donc rendu compte qu'il aurait fallu avoir des outils capables de calculer les impacts réels de notre SI. Malheureusement ces outils n'existent pas aujourd'hui. Nos choix ont été fait au regard de comparaisons globales de technologies présélectionnées, elles sont donc critiquables et ont vocation à l'être pour améliorer l'impact de notre SI. Enfin, le projet en lui-même est critiquable car il augmente l'utilisation de composants Hardware gourmands en métaux rares. Même s'il vise à alerter sur l'effondrement des écosystèmes marin, il n'est pas envisageable selon nous de mettre une OceanBox dans l'ensemble des foyers. Ainsi nous avons conseillé à notre sponsor de proposer l'OceanBox aux lieux publics pour toucher le maximum de personne avec une seule Box. Même si le choix du JAVA est mauvais d'un point de vue de l'éco-conception car il consomme plus de ressources pour fonctionner qu'un langage plus bas niveau comme le C par exemple. Cependant il présente l'avantage de pouvoir s'exécuter sur tous les ordinateurs disposant d'une JVM. Il serait donc envisageable de porter des versions pour les PC, les télévisions connectées ou les panneaux de publicité numériques pour utiliser des composants Hardware existants. Une mise à niveau de notre chaîne DevOps permettrait même de déployer facilement sur l'ensemble de ces devices les nouvelles versions.

Impacts de la crise du Covid-19 sur notre projet

Nous sommes fiers de ce que nous avons produit durant ce projet, et ce d'autant plus compte tenu des événements exceptionnels survenus cette année qui nous ont poussé à tous être confinés chacun de notre côté. Ainsi, comme beaucoup de travaux en entreprise, nous avons dû continuer notre développement à distance. Pour y parvenir, la plateforme de visioconférence Zoom nous a été très utile. Nous avons pu nous appeler régulièrement et partager nos écrans durant les appels pour expliquer ce que nous avions fait ou encore pour demander de l'aide en montrant où nous rencontrions des difficultés. De ce fait, la progression de la partie software s'est globalement bien déroulée car tout était faisable depuis chez nous, depuis nos ordinateurs. En revanche, en ce qui concerne le côté hardware nous avons été grandement ralenti. En effet, seul l'un d'entre nous possède ses Raspberry personnelles mais malheureusement il était confiné à la campagne et il n'avait pas emporté ses Raspberry avec lui. Ce ne sont pas des éléments indispensables que l'on pense à prendre partout avec soi, et cela encore moins juste après une annonce de crise sanitaire. Nous n'avons eu aucun moyen de tester notre programme en conditions réelles avant le déconfinement. Nous nous sommes installés des machines virtuelles sur nos postes mais malheureusement elles présentaient des performances extrêmement faibles et des problèmes d'incompatibilités qui nous ont menés à cesser rapidement nos tests sur celles-ci. Nous avons fait part de ces difficultés à notre client et nous avons conclu qu'il fallait qu'il commande des Raspberry et qu'il nous les fasse livrer. Cependant, un grand nombre de bureaux de poste étant fermés durant le confinement les courriers ainsi que les colis mettaient un temps considérable à parvenir à leur destinataire. De ce fait, nous avons pu tester notre application sur une véritable Raspberry que très tardivement après le déconfinement partiel du 11 mai dernier. Le premier essai fut donc un échec (comme expliqué plus tôt) et nous mena à devoir réagir très vite pour réussir à concevoir un produit fini et fonctionnel selon les principaux besoins de notre client.

Retours personnels

Abdel Benamara

Ce projet fut plein d'expériences enrichissantes qui m'ont permis de cerner plus précisément mes points forts et mes axes d'améliorations. En effet, très tôt dans la répartition des rôles de chacun, j'ai été très intéressé par le côté Devops que nous voulions mettre en place. Cependant, une fois la casquette de « Chief DevOps Officer » sur la tête, je me senti perdu et je ne me trouvais pas à la hauteur de la tâche. Heureusement, mes collègues, et amis, se sont montrés compréhensifs et ont repris ce côté du projet à ma place, bien que je faisais tout ce était dans mes cordes pour me documenter et suivre l'avancée de la mise en place de la chaîne Devops car je trouve le résultat de ce type de réalisation très utile et intéressant pour tous. Ce fut alors un premier sentiment d'échec dans ce projet mais duquel j'ai retiré bon nombre d'éléments bénéfiques pour avancer car j'ai aussi appris auprès de mon groupe et je leur étais reconnaissant de m'avoir aidé et d'avoir pris les choses en mains à un moment où je me sentais dépassé.

Peu après la finalisation de la chaîne devops, il fallait que je me rattrape et que je me montre utile à mon tour. Je me suis donc penché sur un grand morceau du projet : le

développement de l'application de l'OceanBox. Bien que cette partie ait été traitée par toute l'équipe, il fallait bien que quelqu'un commence. J'ai donc pris les devants assez tôt dans le but de tester un programme le plus tôt possible, tant sur un environnement de développement qu'en conditions réelles sur une raspberry grâce à la chaîne devops tout juste finalisée. Ayant abordé dans le même temps les notions d'interfaces dans le cadre du cours d'Interface Homme-Machine, je me suis lancé dans l'élaboration d'un premier programme utilisant les outils de la bibliothèque JavaFX de Java. Le choix de cette technologie sembla plutôt logique et cohérent aux précédentes discussions que j'avais eu avec mes camarades car nous souhaitions developper une application en Java compte tenu de nos diverses expériences antérieures en programmation en Java. De plus, le cours d'IHM nous avait introduit les outils de JavaFX, d'où son utilisation pour la première implémentation de l'application des futures Box d'OceanStream. Une fois lancé j'avais rapidement une version primaire qui répondait aux principaux besoins exprimés dans les premières User-stories : la vidéo était lue en boucle et l'utilisateur avait la possibilité de définir une heure de réveil, autrement dit pouvait choisir à partir de quand la diffusion du nouveau contenu devait débuter. Puis, étant en avance sur le développement du software et suite aux démonstrations régulières sur Zoom qui étaient faites à Adrien Landa, le président de l'association et notre MOA dans ce projet, ce dernier nous demanda petit à petit d'implémenter des ajouts facultatifs mais qui seraient pratiques pour le client. JavaFX proposant un catalogue notoire de composants graphiques et des méthodes très utiles, je me suis énormément documenté sur ce qui était faisable et sur les limites. J'ai rencontré bon nombre de petit soucis de développement car le fonctionnement de certaines méthodes ou le comportement de quelques éléments graphiques n'est parfois pas très intuitif, mais j'avançais et j'avais un sentiment très agréable de servir le projet, d'aider mon groupe et surtout de me rattraper par rapport à la première étape que j'avais perçu comme un échec personnel, bien qu'elle ait été une formidable réussite en soit.

Nous devions donc rapidement passer aux tests sur raspberry car nous avions bien avancé et nous étions au début du mois de mars. Cependant, c'était sans compter sur la crise sanitaire sans précédent due au Coronavirus qui a radicalement changé notre façon de vivre en étant confiné et en imposant d'étudier et de travailler à distance. Après quelques semaines chez soi, nous avions bien pris l'habitude de collaborer à distance et le développement de fonctionnalités supplémentaires dans le programme avait continué car nous ne voulions pas mettre en pause le projet même si les tests sur raspberry ont été repoussés compte tenu des difficultés à se faire livrer les petites cartes programmables. C'est alors que j'ai connu mon second échec de cette année. En effet, peu avant le déconfinement, nous avons pu tester notre application en conditions réelles et malheureusement nous nous sommes rendus compte qu'elle ne s'exécutait pas alors que tout avait été fait pour que l'expérience se passe correctement. Le problème s'est révélé être une incompatibilité entre composants graphiques de JavaFX et le système d'exploitation Raspberry Pi OS. Cela signifiait que nous devions entièrement retiré tout ce qui venait de la bibliothèque JavaFX, et par conséquent modifier presque l'entièreté du code source. J'ai eu le sentiment que le retard que l'on venait de prendre dans l'avancée du projet était de ma faute, je m'en suis voulu de ne pas avoir su plus tôt que ce problème technique était connu et déjà référencé sur certains forums dédiés à la programmation sur raspberry. La désillusion de passer de celui qui fait prendre de l'avance sur le développement du programme à celui qui a finalement pas mal retardé les choses a été quelque chose de vraiment difficile à accepter et à réaliser.

Mais là encore, je ne me suis pas attardé sur ce qui avait échoué mais je me suis plutôt concentré sur les raisons qui ont menés à ce problème. Ainsi, j'ai pu apprendre de mes erreurs et m'investir pleinement à la résolution de notre problème d'incompatibilité technique entre le

programme et le système sur lequel il doit être exécuté. Cette fois, vérifier que les éléments que nous allions choisir de garder ou d'ajouter allaient pouvoir fonctionner correctement sur le support final fut la priorité. Nous avons donc pu produire une nouvelle implémentation plus simple et sobre que la précédente car nous avons dû cibler l'essentiel et mettre l'accent sur le fait de fournir une application fonctionnelle et viable rapidement pour respecter les délais que nous nous étions fixés avec Adrien Landa. Par ailleurs, le système d'exploitation des raspberry ne supportant pas les composants graphiques de Java, nous aurions dû nous former à de nouvelles technologies pour les incorporer à notre programme et malheureusement nous n'avions plus le temps nécessaire. De plus, ce nouveau départ en cours de route nous a permis de consacrer plus de temps et d'énergie à des fonctionnalités essentielles comme celle du téléchargement. Celle-ci a été le centre de nombreuses discussions et réflexions pour pouvoir aboutir à une implémentation qui n'est pas visible par l'utilisateur et qui est respectueuse de l'environnement : concrètement, on télécharge les paquets les uns après les autres, en veillant à ne jamais télécharger le même paquet que celui qui est lu ou un paquet postérieur afin de ne pas avoir d'incohérence de lecture, et tout cela est fait de nuit pour utiliser internet quand le réseau est le plus léger possible.

Pour conclure, je retiens principalement de ces derniers mois qu'il est important de bien prendre le temps d'analyser ce que l'on va devoir faire et surtout être honnête envers soi-même. Notamment, ceci permet de ne pas se fourvoyer et faire perdre un précieux temps à tout le groupe dans la correction de ses erreurs ou encore la demande d'aide si importante qu'elle en ralentit les tâches des autres. De plus, je suis fier de moi pour toutes les fois où j'ai rencontré des difficultés et que j'ai su persévérer pour trouver des solutions seul ou en sollicitant mes collègues suffisamment tôt pour ne pas trop impacter l'avancée du projet. Finalement, j'ai eu la chance de faire partie d'une équipe remarquable qui s'est toujours montrée positive et motivante pour avancer et rester focalisée sur le fait de rendre un produit fini comme dans l'objectif initial que l'on s'était fixé. Bien sûr, le rendu ne correspond pas exactement à ce que j'imaginais mais les fonctionnalités primordiales sont bien présentes et fonctionnelles. En outre, la réutilisation et la reprise du projet sont facilitées grâce à la chaîne devops et la documentation que nous avons rédigée. Une fois de plus, tout cela a été possible grâce à l'entraide et aux partages de connaissances de toute l'équipe et je leur en suis donc très reconnaissant.

Julien Doujet

Pour mon retour réflexif je souhaite porter ma réflexion sur un élément qui a probablement joué un rôle déterminant sur le développement et la réussite de notre projet. Après ces plusieurs mois passés je me questionne sur le point suivant qui a pu contribuer ou non dès le début du projet à notre progression :

Est-ce que définir un plan de développement, une organisation initiale d'un projet détermine nécessairement la réussite de celui-ci ?

Au cours de mon cursus universitaire j'ai travaillé sur différents projets traitant de problématiques diversifiées mais jamais d'une aussi grande envergure avec un tel challenge regroupant autant d'acteurs autour de celui-ci. En effet, j'avais l'habitude d'être au maximum sur des projets en binôme avec de « petits » développements d'un mois ou deux maximums. De plus, sur des projets universitaires beaucoup plus courts l'organisation autour du développement du programme est rapidement mise en place et ne semble pas avoir autant d'impact sur le bon déroulement du

projet. En revanche le développement de l'OceanBox s'étendait sur plusieurs mois et divers éléments étaient souhaités et attendus nécessitant de planifier et de s'organiser davantage, ce que nous avons fait dès le début. Nous avons collaboré avec une association ici Ocean Stream ce qui pour moi est aussi une première car jusqu'à présent j'étais confronté à des projets avec un aspect uniquement universitaire (mis à part mon alternance) où la partie organisation et communication avec d'autres acteurs autres que les étudiants n'est habituellement pas présente.

En revanche j'ai constaté avec mes collègues que face aux problèmes de compatibilité (bibliothèques JavaFX et Raspberry), plus les problèmes liés à la crise du COVID 19, l'organisation et le cheminement des étapes de développement que nous aurions dû suivre se sont vues fortement modifiées. En effet, avant ces imprévus l'accomplissement des User Stories se faisait dans l'ordre chronologique initialement prévu avec Adrien et le développement du projet se déroulait comme convenu. Cependant, à partir notamment du début du confinement à cause du COVID 19 nous avons été perturbé dans l'évolution du projet et l'ordre des US a été modifié comme évoqué précédemment. Je pense qu'à cet instant notre principale force a été notre réactivité et flexibilité face à cette situation.

Je pense que notre bonne organisation initiale et le fait d'avoir défini un plan d'action solide initialement a contribué énormément à la réussite du projet mais ce projet nous a montré que savoir s'adapter et être réactif face au changement sont des points tout aussi importants pour la réussite d'un projet. La communication, les échanges entre nous à distance ont aussi contribué à l'évolution du projet.

Pour ma part je me suis concentré sur la partie correspondant à la connexion et à la construction de la structure de la base de données avec le paramétrage des properties du client. J'ai pu aussi acquérir des notions sur la partie serveur FTP que j'avais installé sur un ordinateur initialement. Ce projet m'a également permis de me rendre compte de l'utilité de Git pour l'organisation d'un développement durant un projet. Cependant pour la partie plus hardware j'ai développé récemment une classe java pour mettre en veille, éteindre ou allumer la télévision à partir de la Raspberry et il est plus difficile de la tester à cause du confinement malgré ce que nous avions planifié initialement. L'US concernant le Plug & Play était un des points essentiel et prioritaire du projet mais heureusement le groupe est réactif et je les remercie pour leur aide, leur communication et en particulier David pour faire les tests à distance avec le matériel nécessaire malgré les circonstances.

Le fait justement de devoir modifier l'organisation fixée à l'origine est une situation que je n'ai jamais vraiment vécu avant ce projet ou du moins pas de manière aussi importante (ex : pour des développements j'ai dû parfois revenir sur mes décisions au niveau de la structure de mon code, de mes classes à cause d'erreurs de conception ou pour améliorer mon rendu). Peut-être aurait-il fallu que nous pensions à l'avance à un second plan de développement malgré la numérotation des US définis (ex : penser à un autre ordre des priorités des US en cas de problème). A prendre en compte aussi que nous aurions aimé avoir plus de temps à consacrer au sujet ce qui aurait pu nous permettre de trouver des solutions à nos problèmes plus rapidement.

Je pense par rapport à mes expériences précédentes que s'organiser et fixer un plan de développement initial peut suffire pour réussir un projet si aucun imprévu ne se présente mais ce projet m'a appris au-delà de la partie développement qui m'a beaucoup apporté, à penser autrement l'organisation d'un projet et essayer d'anticiper davantage différents cas de figure même si nous étions efficaces. Ce projet nous a montré qu'un plan d'actions initial n'aurait pas suffi et que la communication à distance entre les différents membres (mails, Slack, réunions Zoom, …) en plus de notre adaptabilité (replanifier nos missions et repenser l'ordre des US par rapport à ce qui est possible en fonction des circonstances) sont des atouts à conserver pour de futurs projets.

David Ekchajzer

Ce projet a été pour moi l'occasion de mettre en pratique l'envie que j'avais depuis quelque temps d'adapter les méthodes AGILE et DEVOPS à l'éco-conception. J'espérais voir les avantages et inconvénients de ce rapprochement à travers un exercice concret comme celui proposé par le projet commun. Les avantages existent bel et bien selon moi si l'ensemble des acteurs portent la volonté d'éco-concevoir leur produit comme cela a été le cas dans notre projet. En effet, les cérémonies et les User Stories permettent de pousser l'équipe à se poser les bonnes questions quant à l'impact d'un choix (souvent technique). Cependant je regrette le temps trop court que nous avons pu accorder, même s'il a été important, au regard de l'ensemble des enseignements à suivre cette année. En effet, je pense que pour appliquer dans les meilleures conditions cette méthodologie il est nécessaire d'avoir beaucoup de temps pour se poser les bonnes questions et répondre de manière quantitative aux problématiques proposées. Cela nécessite des semaines de recherche que nous n'avions pas.

Ce manque de temps est aussi dû au volet technique de notre projet également très ambitieux. En effet, notre projet s'appuie sur d'innombrables technologies dont nous n'avions pas connaissance avant ce projet. Il a fallu les comprendre, les apprendre et les appliquer. Cela a été très enrichissant car j'ai pu me former à de nombreuses technologies qui se sont ajoutées à ma culture informatique. Je pense notamment aux outils DevOps. J'ai dû imaginer la chaîne DevOps puis m'approprier les outils que j'ai dû installer sur un serveur Raspbian puis paramétrer avec Mathieu comme Jenkins ou Nexus.

J'ai également appris beaucoup sur les infrastructures (serveurs). En effet, ayant plusieurs Raspberry chez moi, je me suis occupé de paramétrer sur celles-ci notre serveur DevOps puis notre serveur FTP et enfin notre base de données, de manière sécurisée. Je me sens aujourd'hui à l'aise avec l'utilisation de Linux sur lequel je tends à migrer. Ce travail d'installation et de paramétrage d'infrastructures a également été l'occasion de mettre en pratique mes cours de réseau qui m'ont été d'une très grande utilité. Ayant appliqué les enseignements appris en cours je me sens aujourd'hui à l'aise avec les problématiques réseaux.

J'ai également apprécié travailler avec Abdel sur l'architecture de notre application java qui nous a permis d'appliquer notre cours d'architecture du second semestre. Le binôme que nous avons formé m'a paru très efficace et complémentaire en qualités et compétences. Le travail d'abstraction nécessaire à l'élaboration d'une architecture robuste et évolutive nous a pris du temps. Cependant je comprends aujourd'hui l'intérêt de ce temps car il est beaucoup plus simple de faire évoluer l'application.

Enfin, j'ai également pu mettre en pratique mes compétences en hardware (acquises lors d'un stage de seconde chez Withings entreprise de l'IOT de santé) pour intégrer un capteur de mouvement à notre OceanBox.

Finalement, bien que ce projet m'ait demandé un temps considérable, il a été l'occasion d'en apprendre beaucoup sur des technologies et des méthodes de développement projet nouvelles. Ainsi, je crois que ce temps a été un très bon investissement. C'est d'ailleurs en partie grâce à ce projet que j'ai décidé d'effectuer ma prochaine alternance dans une entreprise œuvrant pour l'informatique durable.

Mathieu Ridet

L'objectif de ce projet fût le développement de boîtiers qui, branchés à un écran, affichent un aquarium digital sur celui-ci à l'aide de vidéos filmés la veille. Dès le début de la coopération nous avons ressenti l'importance de ce projet pour Adrien - le responsable du côté de l'association – ce qui nous a motivé à fournir un travail efficace et de qualité tout au long de l'année.

Pour développer ce projet, nous avons utilisés de nombreuses technologies différentes qu'il nous a fallu prendre en main pour certaines même si nous avons choisi pour langage de programmation principal le langage Java. Ainsi, nous avons pu mettre à profit nos connaissances de ce langage *et ses différents Framework comme Lombok*.

Lors de ce projet j'ai, notamment, pu participé activement à la mise en place de notre chaîne DevOps ainsi qu'au développement de la classe permettant le téléchargement des paquets vidéo depuis notre serveur FTP distant.

Aussi, je me suis occupé des classes de log[1] de notre projet et à leur dépôt (ou pas) sur le serveur.

Pour cela, j'ai débuté par une approche conceptionnelle afin de cerner le problème ; ce qui m'a permis de déterminer, dès le départ, la meilleure solution technique possible : l'API de logging de Java.

J'ai eu l'idée de réaliser nos classes de log de cette manière grâce à ma bonne connaissance de ce langage et à mon activité professionnelle qui me permet de traiter ce genre de sujets.

Le monde informatique étant vaste et proposant une multitude de solutions à un problème, de nombreuses API de logging comme SL4J ou Log4J auraient pû me permettre de réaliser ces classes différemment. Néanmoins, nous avons choisi de ne pas utiliser ces différentes API car elles sont beaucoup plus complexes à mettre en place et à utiliser.

Les classes de logging développées me paraissent suffisamment efficaces et complètes pour notre projet et ne présentent, selon moi, aucune limite apparente par rapport au besoin actuel de l'association et du projet développé.

Le fonctionnement de l'échange de fichiers (texte ou vidéo) entre mon programme local et le serveur FTP m'a demandé un certain temps de compréhension et m'a posé quelques soucis tout au long de mon développement. Ce fût également le cas concernant l'adaptation de mes programmes à ceux de mes collègues même si cela m'a permis de développer mon "esprit Agile"[2].

Travaillant en mode Agile en entreprise, je suis confronté quotidiennement aux soucis d'adaptation et de flexibilité qui, avec du travail et au fil du temps, deviennent graduellement une force.

Malheureusement j'ai connu une période d'absence due à un problème personnel; ce qui m'a empêché d'apprendre autant de choses que possible sur ce projet.

Néanmoins, j'ai tout de même appris énormément de choses et ce à plusieurs niveaux.

---

[1] Log (en français Journal) : Fichier (local et/ou distant) dont le principe consiste à stocker un historique des événements.

[2] Agile : Approche de gestion de projet IT selon laquelle des changements techniques et/ou fonctionnelles sont fréquents tout au long du projet pour s'adapter aux besoins métiers et fournir le meilleur résultat possible.

Les multiples réunions avec l'équipe et/ou avec Adrien – le responsable du projet du côté de l'association – et la compréhension des besoins métiers m'ont permis de m'améliorer au niveau fonctionnel et de la communication.

Etant considéré comme le Scrum Master de ce projet, j'ai pu, grâce à des outils comme Trello ou à de nombreux échanges, m'améliorer au niveau de la communication ; point crucial pour la réussite d'un projet.

Au niveau technique, je me suis amélioré en programmation Java. Je possède, désormais, une large connaissance des outils de logging de ce langage et également la mise en place d'une communication bi-directionnelle avec un serveur FTP distant. Finalement, grâce à la réalisation de la chaîne DevOps en début de projet avec mon collègue David, j'ai pu développer de nombreuses connaissances concernant l'outil de versionning Git, l'outil d'intégration continue Jenkins ou encore la plate-forme logicielle pour la configuration et la gestion des ordinateurs Ansible.

Pour ma part, ce projet fût très intéressant et enrichissant car il fût l'occasion parfaite de mettre en pratique mes connaissances techniques et fonctionnelles dans un cadre semi professionnel.

L'association entre l'université Paris 1 et des associations comme Ocean Stream est une bénédiction pour les étudiants car elle nous permet d'apprendre des aspects intéressants pour notre avenir tout en fournissant une aide non négligeable. La forte implication d'Adrien ainsi que celle de l'équipe ont donné à ce projet un intérêt encore plus important et nous a permis de travailler efficacement au développement de la solution recherchée.

Adrien Landa (PO)

Pour ma part, c'est la 5ème équipe d'étudiants en formation MIAGE de Paris 1 avec laquelle je travaille sur le développement d'un projet. Chaque fois, c'est une expérience différente qui enrichi ma connaissance et culture en informatique et qui questionne et fait évoluer mes méthodes de gestion d'équipe. Sans chercher à effectuer la moindre comparaison, je tiens à souligner qu'au vue de ma perspective, Abdel, David, Julien et Mathieu ont su créer une équipe soudée et performante et ce malgré le contexte difficile lié à la pandémie de COVID19. Sur les quatre équipes avec qui j'ai travaillé au développement d'un projet pour le compte de l'association Ocean Stream dont je suis le président, la leur est celle qui a montré le plus d'autonomie, de résilience, et de compétences dans l'exécution du projet.

En ce sens, ça a été pour moi une nouvelle façon d'interagir avec une équipe. Mon rôle n'a pas été d'encadrer et de superviser la réalisation du projet, ce à quoi j'étais habitué, mais plutôt de parler au nom du projet en exprimant les besoins de celui-ci. Ce rôle de MOA (maître d'ouvrage ou Product Owner) que l'équipe m'a proposé, m'a permis d'exprimer mes attentes par rapport au produit développé sans devenir intrusif dans la réalisation de celui-ci et en laissant à l'équipe toute liberté sur les choix technologiques et organisationnels. Le risque de cet arrangement était à mes yeux une potentielle divergence entre les réalisations de l'équipe et mes attentes, et un manque de dynamisme et de rythme sur l'avancement du projet. A aucun moment ce risque ne s'est révélé vrai. L'équipe a su être à l'écoute des besoins que j'ai exprimé par rapport au projet et y répondre d'une façon satisfaisante. De plus, les membres ont su mettre en place une dynamique de travail performante qui répondait aux contraintes de temps imposées, sans mon intervention.

Selon moi, le succès de cette équipe réside dans sa capacité d'organisation rapide, sa relative homogénéité sociale et genrée, son enthousiasme pour le projet en lien avec des intérêts personnels, une capacité d'écoute et une sensibilité importante de ses membres. Très rapidement après avoir choisi le projet d'OceanBox (boîtier de diffusion transformant un téléviseur en « aquarium digital ») proposé par l'association Ocean Stream, ils ont souhaité prendre rendez-vous pour répartir les rôles de chacun, définir une méthodologie de travail et définir les fonctionnalités à développer qui formeraient la feuille de route du projet. Je cite aussi, l'importance de l'homogénéité relative du groupe comme facteur de réussite, car d'expérience j'ai constaté que les équipes mixtes (hommes et femmes) et/ou avec des membres très différents en termes de milieu socio-économique et culturel ont beaucoup plus de mal à se structurer, à communiquer régulièrement et donc à créer la structure sociale nécessaire pour travailler efficacement. Dans leur cas, leurs différences en termes de bagage éducatif et d'expérience de vie ont été complémentaires et au service d'une dynamique de travail enrichie. De plus, parmi les membres de l'équipe, deux au moins ont une relation importante au vivant. L'un est plongeur sous-marin tandis qu'un autre a une formation initiale de biologiste. Et je soupçonne les deux autres d'avoir une sensibilité personnelle pour l'écologie liée à des expériences vécues ou à une curiosité intellectuelle.

L'une de mes défaillances sur ce projet a été mon incapacité à créer un ou des moments signifiants qui lient l'équipe à l'environnement marin qui est le bénéficiaire final du projet. Je souhaitais particulièrement permettre aux membres de l'équipe de venir sur le littoral de la Côte d'Opale pour leur permettre de visiter Nausicaa où l'équipe permanente d'Ocean Stream travaille sur le projet Le Récif qui intègre le développement des OceanBox. De plus, ce déplacement aurait été l'occasion pour l'équipe des MIAGE, de rencontrer physiquement d'autres membres d'Ocean Stream pour nouer des liens plus complexes qui peuvent fluidifier les échanges de travail et initier un engagement à plus long-terme au sein de l'association et de ses projets. Cette rencontre prévue fin avril n'a pas eu lieu à cause de la crise sanitaire liée au covid. Cependant, il n'est pas exclu qu'elle puisse avoir lieu ultérieurement, bien que dans ce cas ses impacts positifs sur la dynamique d'équipe seront certainement moindres suite à la fin de l'engagement universitaire dans lequel s'inscrivait le travail de l'équipe. Car malgré la documentation produite sur le projet pour faciliter la prise en main de celui-ci par une nouvelle personne, l'un des enjeux pour moi était qu'un des membres de l'équipe MIAGE souhaite continuer de s'investir dans le projet Le Récif. Notamment, en intégrant le projet en tant qu'associé pour bénéficier des perspectives entrepreneuriales de celui-ci.

Autrement, j'ai aussi endossé le rôle de fournisseur/sponsor en gérant les commandes de matériel et en assurant la logistique nécessaire à leur acheminement. La crise sanitaire a quelque peu ralenti et complexifié cette activité. Cependant, elle m'a permis avec mon associé, de développer une méthodologie et des outils de suivi du matériel et de gestion du budget. De plus, Abdel, David, Julien et Mathieu m'ont notamment permis d'identifier de nouveaux fournisseurs pour réaliser les OceanBox et d'initier une structuration du processus de production de celles-ci. J'ai même été initié par l'équipe à l'utilisation de nouveaux outils informatiques pour configurer les OceanBox et pour accéder au serveur FTP utilisé pour héberger les vidéos de récifs diffusées par les boîtiers.

J'aurais aussi aimé approfondir cette démarche d'apprentissage informatique notamment en m'initiant au codage et en prenant d'avantage de temps pour comprendre les technologies employées. Initialement j'ai démarré un cours d'introduction en ligne qui s'est soldé par un abandon suite à un manque de rigueur et d'investissement temporel de ma part. De plus, j'ai eu tendance à déléguer les questions techniques à Matthieu Hocquart qui assiste l'association sur toutes les questions liées à l'informatique. Bien que je pense cette mise en relation pertinente, elle a aussi réduit

mon besoin de me former à ces questions techniques et j'ai limité ma monté en compétences informatiques qui me permettrait de mieux prendre en main le produit développé.

Autrement, l'utilisation du serveur FTP pour mettre en ligne les vidéos à diffuser par les OceanBox s'est avérée simple mais la mise en ligne compliquée. Je soupçonne la taille importante des fichiers d'avoir été responsable d'upload incomplets. Il va être nécessaire de vérifier que c'est la raison de mes difficultés à mettre du contenu en ligne et il reste à trouver la taille de fichier optimale pour répondre aux enjeux de mise en ligne et de temps de visionnage suffisant par paquet.

En conclusion, j'aimerais remercier Abdel, David, Julien et Mathieu pour leur travail sur le projet des OceanBox. Grâce à eux, j'ai été initié à de nouvelles méthodologies de travail pour organiser et gérer un projet informatique en plus d'endosser de nouveaux rôles et de participer au développement d'un nouvel outil important pour le projet Le Récif porté par Ocean Stream. Il me reste désormais à utiliser cette première version de l'OceanBox pour prospecter les premiers clients du projet Le Récif et obtenir des retours utilisateurs qui vont servir à diriger l'optimisation des performances des boîtiers.

OceanStream et la MIAGE Sorbonne s'associent pour créer

L'OceanBox

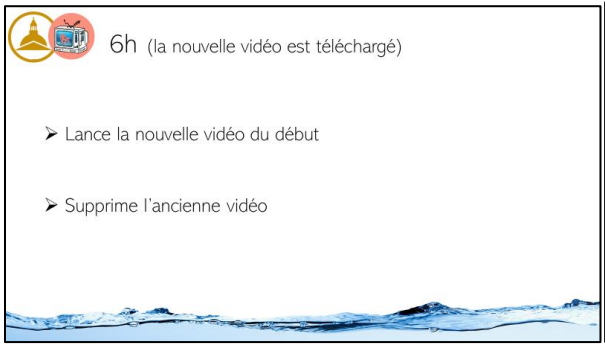
Un projet AGILE et DevOps écoconçu

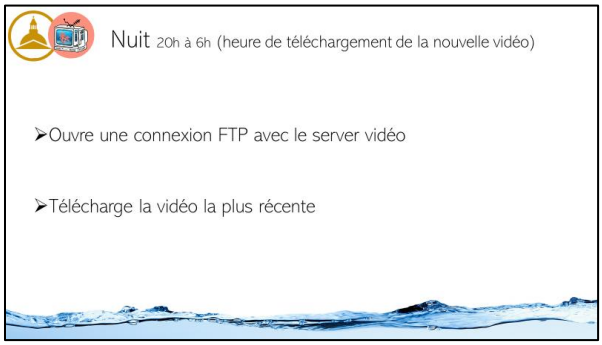
ANNEXES

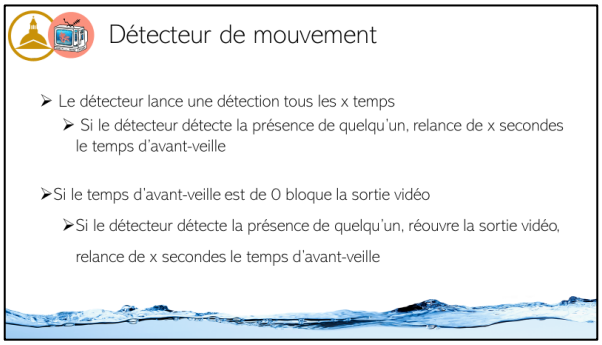
Abdel Benamara | Julien Doujet | David Ekchajzer | Mathieu Ridet

08/05/2020

Etapes d'une box







OceanBox Tuto

Exécuter le programme sur un PC

Abdel Benamara | Julien Doujet | David Ekchajzer | Mathieu Ridet

17/06/2020

I- Environnement de travail

1) Au minimum java Oracle 8 ou l'OpenJDK 8
2) VLC
3) MediaInfo CLI 20 (*https://mediaarea.net/fr/MediaInfo*)
4)

II- Paramétrage de l'application

1) Récupérer la dernière version des sources depuis le GIT :
   - future_feature : utilisé pour des modifications en bac à sable
   - current_feature : correspond aux développement en cours
   - dev : correspond au code en cours de test
   - Master : correspond au code en production

2) Créer un fichier « *video* » avec les vidéos à jouer sur votre ordinateur puis paramétrer le chemin d'accès aux vidéos.

- Dans le code source dans src/oceanbox/app.java :

```
SystemPropreties.setPropretie("videoPath","AbsolutPathToVideo");
```

- Ou dans les fichiers paramètres SystemProperties.properties :

```
videoPath=AbsolutPathToVideo
```

3) Paramétrer plus précisément le system :

- Dans SystemProperties.properties :

  - IdUnique de l'OceanBox : oceanBoxNumber=0
  - Mot de passe du serveur ftp : ftpPassword=Stream2020
  - Chemin d'accès aux fichier de log : FtpLogPath=ftpLogFile.log
  - Ip de la base de données : dbIP=37.187.107.122
  - User du serveur ftp : ftpUser=ocean_ftp
  - Port de la base de données : dbPort=3306
  - Chemin d'accès dans le serveur ftp des vidéos : ftpVideoPath=/default_video/
  - User de la base de données : dbUser=ocean_bdd
  - Ip de l'oceanBox (inutile pour le moment) : oceanBoxIP=127.0.1.1
  - Ip du serveur ftp : ftpIP=37.187.107.122
  - Port du serveur ftp : ftpPort=21
  - Chemin d'accès au fichier de log : DbLogPath=dbLogFile.txt
  - Mot de passe de la base donnée : dbPassword=OceanBox2020

- Dans ClientProperties.properties :

  - Temps sans interaction avant mise en veille : timeBeforeStandby=00\:00\:20
  - Type de flux vidéo : videoStream=default
  - Prochaine date de téléchargement (est défini par l'application si le téléchargement est activé) : nextDownloadTime=Tue Jun 16 08\:28\:45 CEST 2020
  - L'heure de réveil qui correspond à l'heure à laquelle la nouvelle journée de vidéo commence : wakingHour=08\:30\:00
  - Activer ou non la veille : activateStandby=false
  - Type d'utilisateur (aucune influence sur le programme) : userType=Test
  - Nom de l'utilisateur (aucune influence sur le programme) : userName=User Demo
  - ID de l'utilisateur (aucune influence sur le programme) : userId=0

4) Définir le chemin d'accès à MediaInfo

Dans les SystemProperties, spécifier le chemin d'accès à la commande MediaInfo sur votre poste.

MediaInfoCMD=/usr/bin/mediainfo

III- Paramétrage du lecteur vidéo

1) Utiliser un lecteur existant

Aujourd'hui deux lecteurs java existent un utilisant OMXplayer (*JOmxPlayer*) et un VLC (*JVlcPlayer*) - privilégié dans le cadre d'une exécution sur pc/mac. Pour passer de l'un a l'autre il faut modifier l'implémentation du lecteur vidéo dans le constructeur de contenue :

```
videoPlayer = new JVlcPlayer();    OU   videoPlayer = new JOmxPlayer();
```

Attention : l'implémentation VLC nécessite de préciser le chemin d'accès absolu à la commande vlc. Ce chemin peut changer d'un pc à l'autre dans les SystemProperties :

```
VlcCMD=/usr/bin/vlc
```

2) Ajouter un lecteur vidéo

Si vous souhaitez ajouter un autre player vidéo il vous suffit de créer une nouvelle implémentation de l'interface *videoPlayer.java* et de préciser cette implémentation dans le constructeur de *contenu*.

IV- Paramétrage de la veille

La veille est normalement à l'écoute d'un détecteur de mouvement souder aux GPIO des Raspberry. Dans le cas d'un pc, la veille est difficilement gérable par détecteur de mouvement. Nous avons donc une seconde implémentation de la veille par scanner dans la console.

1) Choisir l'implémentation VeilleScanner.java

Dans le constructeur de contenu :

```
veille = new VeilleScanner(this);     OU     veille = new VeilleDMV(this);
```

2) Ajouter une implémentation de la veille

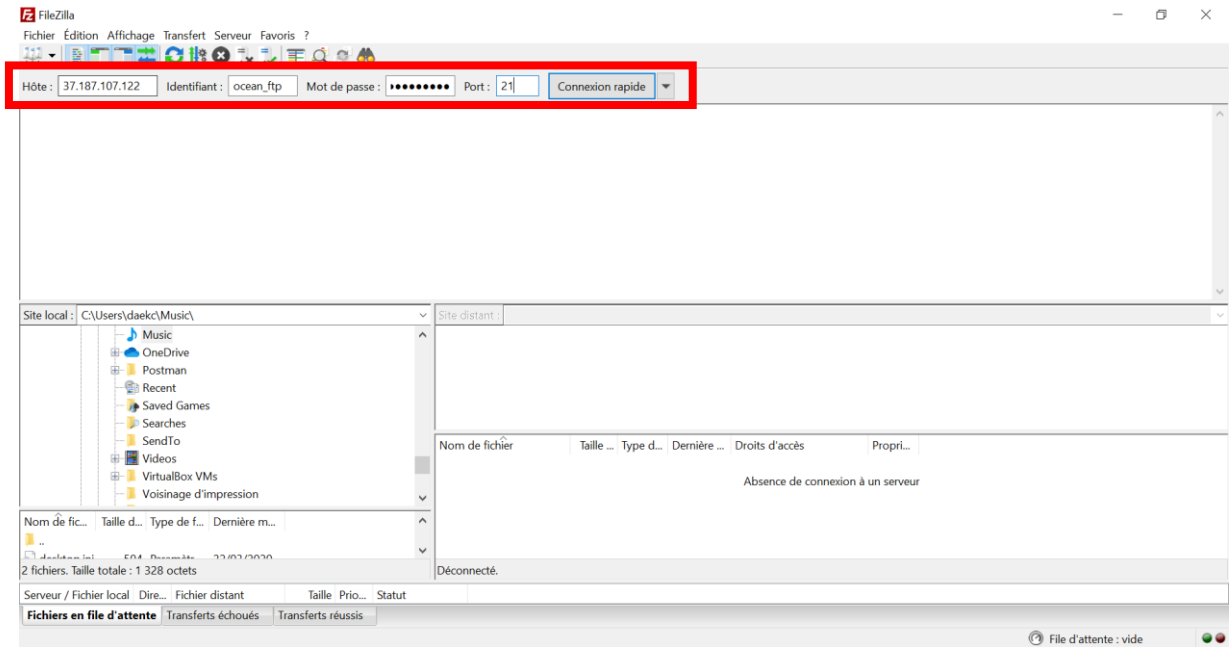
Vous pouvez ajouter une nouvelle implémentation de la veille en implémenter l'interface Veille.java

Documentation FilleZilla

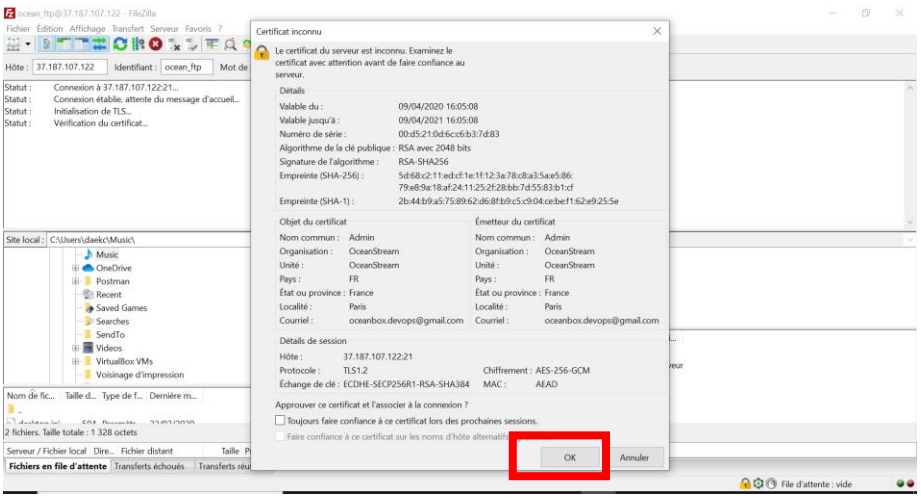
1- En haut gauche entrez les informations de connexion suivantes

- Hôte : 37.187.107.122
- Identifiant : ocean_ftp
- Mot de passe : Stream2020
- Port : 21

Puis cliquez sur Connexion rapide

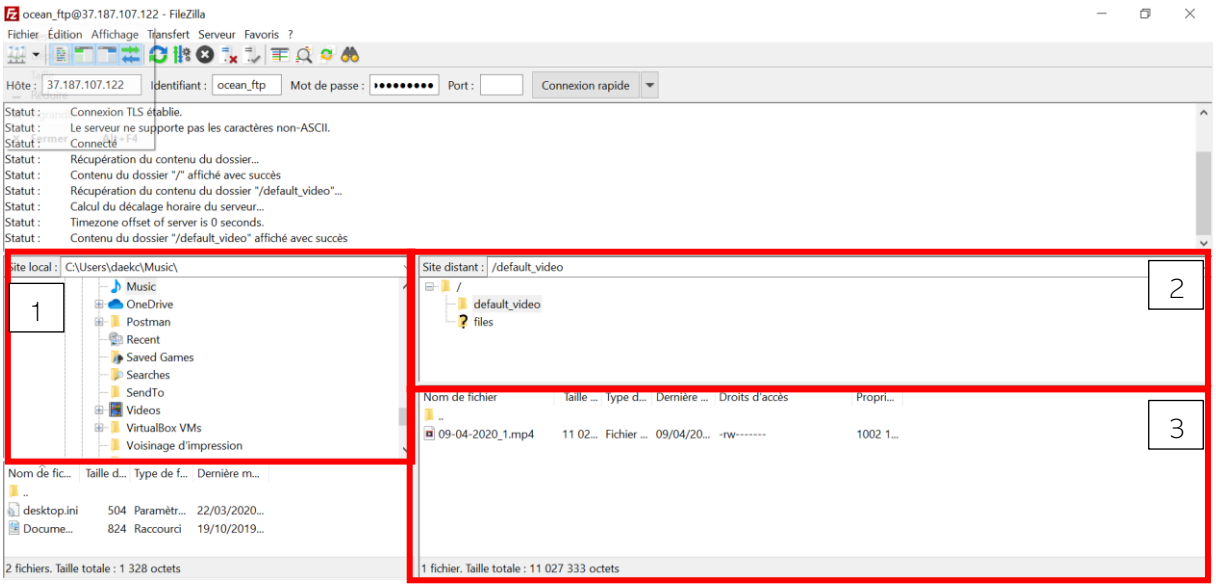


2- Accepter le certificat en cliquant sur OK



3- Vous pouvez maintenant intéragir avec les dossiers distants par glissé-déposé.

1) Vos fichier locaux
2) Arboresance des fichiers distants (mettre les videos dans default_video)
3) Fichiers dans le repertoir selectionné



OceanBox Tuto

Chaine DevOps

Abdel Benamara | Julien Doujet | David Ekchajzer | Mathieu Ridet

17/06/2020

I- Ajouter une OceanBox à la chaine DevOps

- Dans le répertoire updater, importer la dernière version de l'updater (updater.sh) pour la dev ou pour la prod
- *chmod +x updater.sh*
- Pour initialiser la chaine il faut récupérer le manifeste correspondant à la version sur l'OceanBox. Pour récupérer le manifeste généré par le build maven de la dernière version :
  - ✓ *Dev : wget -O MetaLatestOceanBox.xml "http://176.158.51.172:8081/nexus/service/local/artifact/maven/resolve?g=oceanbox&a=OceanBox&v=LATEST&r=DevOceanBoxReleases&e=jar"*
  - ✓ *Prod : wget -O MetaLatestOceanBox.xml "http://176.158.51.172:8081/nexus/service/local/artifact/maven/resolve?g=oceanbox&a=OceanBox&v=LATEST&r=OceanBoxReleases&e=jar"*
- Ajouter l'updater à la cronTab :
  - ✓ crontab -e
  - ✓ *add /home/pi/OceanBox/updater/updater.sh*

Documentation installation sur Raspberry

OceanBox Tuto

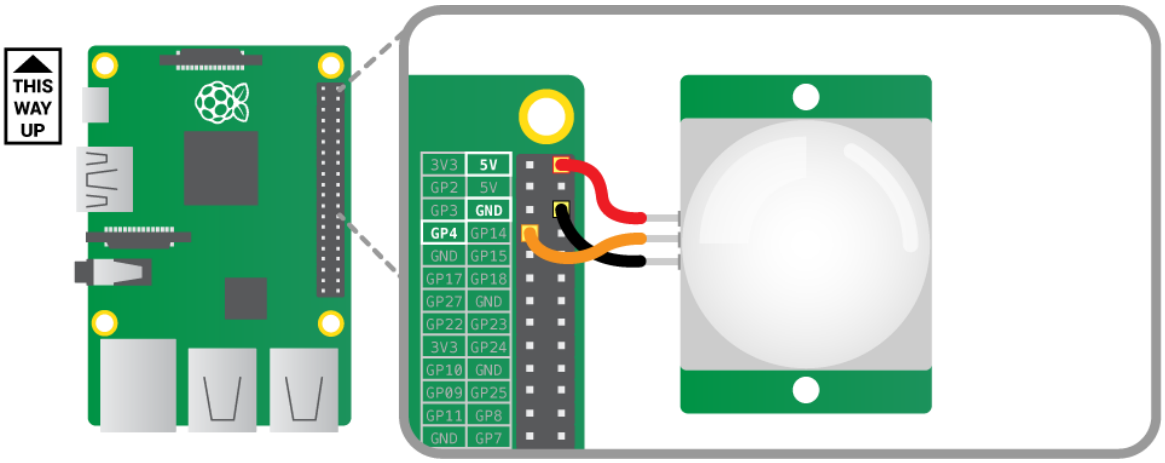
Installer le programme sur une Raspberry

Abdel Benamara | Julien Doujet | David Ekchajzer | Mathieu Ridet

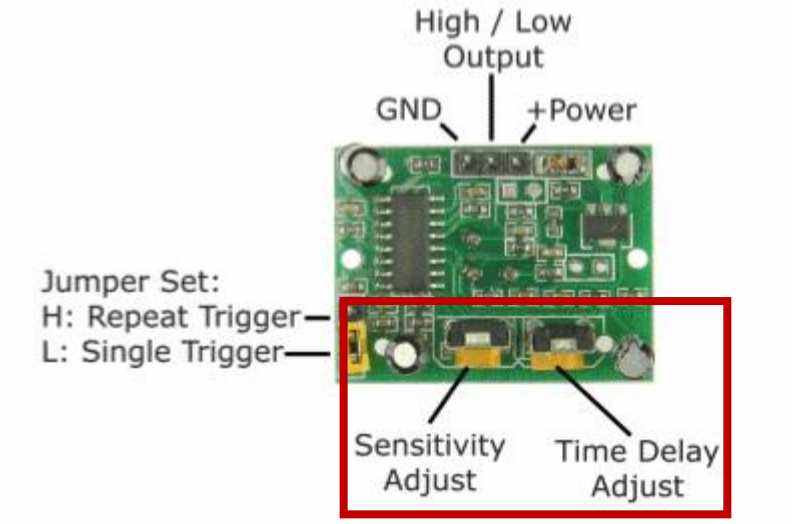
17/06/2020

Brancher le détecteur de mouvement

1) Brancher les câbles Dupont femelles – femelles comme dans le schéma suivant



2) Tourner les deux potentiomètres au maximum à gauche



3) Choisie l'une des méthodes suivantes

Méthode simplifier

1) Récupérer l'ISO RasbianOceanBox.iso
2) A l'aide de Balena etcher (ou tout autre programme permettant de booter une carte SD) installer l'ISO sur une carte micro-SD.
3) Installer la carte SD dans votre Raspberry et lancez là

Autres méthodes

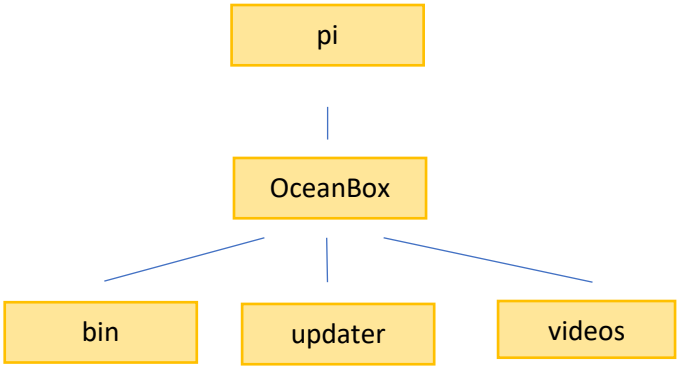
I- Prérequis

1) Installer BallenaEtcher ou tout autre programmer permettant dans rendre une carde SD bootable.
2) Installer RaspBerry PI OS (32-bit) with desktop.
3) A l'aide de BallenaEcher booter une carte micro-SD avec la version de RaspBerry PI OS téléchargé.
4)
   - *Vous avez un clavier, une souris et un écran :* Après avoir branché les connectiques et les périphériques, allumé la Raspberry (en branchant l'alimentation), connectez-vous au wifi (ou brancher un câble Ethernet) et autorisé l'accès SSH (*https://www.raspberrypi-france.fr/guide/connecter-ssh-raspbian/*).
   - *Sinon :* https://raspberry-pi.fr/raspberry-pi-sans-ecran-sans-clavier/.
5) Connectez-vous à la Raspberry en SSH (ou ouvrir une console sur la Raspberry).
6) Installer les programmes requis :

- *sudo apt-get update*
- *sudo apt-get install git*
- *sudo apt-get install ansible*
- *sudo apt-get install openjdk-8-jre*

- *sudo apt-get install xml-twig-tools*
- *sudo apt-get install mediainfo*
- *sudo apt-get install wiringpi*

7) Dans /home/pi/ créer l'arborescence suivante :



- *mkdir OceanBox*
- *cd OceanBox*
- *mkdir bin*
- *mkdir video*
- *mkdir updater*

8) Autostart

- copier la dernière version de l'autorun (Dans le drive : script/autorun/OceanBox-AutoStart.sh) dans bin

- *chmod +x OceanBox-AutoStart.sh*

- *mkdir /home/pi/.config/autostart*

- *nano /home/pi/.config/autostart/clock.desktop*

```
[Desktop Entry]
Type=Application
Name=OceanBox
Exec=/home/pi/OceanBox/bin/OceanBox-AutoStart.sh
```

II- Installer le software

1) Depuis le répertoire nexus (de la chaine DevOps)

Pour récupérer la dernière version du software aller dans bin puis :

- ✓ *Pour la dev : wget -P "http://176.158.51.172:8081/nexus/service/local/artifact/maven/redirect?g=oceanbox&a=OceanBox&v=LATEST&r=DevOceanBoxReleases&e=jar" --content-disposition*
- ✓ *Pour la prod : wget -P "http://176.158.51.172:8081/nexus/service/local/artifact/maven/redirect?g=oceanbox&a=OceanBox&v=LATEST&r=OceanBoxReleases&e=jar" --content-disposition*

2) Ou manuellement

Builder le projet à l'aide de l'outil d'export d'Eclipse ou avec un maven build. Mettez le jar (avec les dépendances) dans le fichier bin.

III- Ajouter l'OceanBox à la chaine DevOps

Suivez les instructions pour ajouter la Raspberry à la chaine Devops dans la documentation DevOps.

Post-requis pour TOUTES les méthodes

1) Transférer des vidéos dans OceanBox/video en respectant la convention de nommage « dd-mm-aaaa_idVideo.mp4 »
2) Lancer le programme une fois pour vérifier son bon fonctionnement. Les fichiers de propriétés ont été générer.

3) Créer une nouvelle OceanBox dans la base de données et remplacer la propriété `oceanBoxNumber` par l'ID correspondant dans la BDD.

IV- Liens utiles

Transférer fichier en SFTP : https://pobot.org/Envoyer-des-fichiers-sur-la.html

Gestion - Achats

Type	Dépense	Date	Fournisseur	quantité	Prix Unitaire TTC	TVA	Prix Total	Commentaire	Lien
Infrastructure	Starter Kit Raspberry Pi4 - 2GB	02-mars	Kubbi	1	69,90 €	20,00%	69,90 €	Master de la chaine DevOps	
Infrastructure	Carte SD 128Go	02-mars	Kubbi	2	24,95 €	20,00%	49,90 €		
Remboursement Infrastructure	Carte SD 128Go	Prévisionnel	Kubbi	1	-24,95 €	20,00%	-24,95 €		
Remboursement Infrastructure	Carte SD 128Go	Prévisionnel	David Ekchajzer	1	-24,95 €	20,00%	-24,95 €		
OceanBox Dev/Recette	Pi zero WH	Prévisionnel	Kubbi	2	14,60 €	20,00%	29,20 €	Slave avec connecteur wifi | Prix estimé	
OceanBox Dev/Recette	Adaptateur ethernet	Prévisionnel	Amazon	2	13,95 €	20,00%	27,90 €	Prix estimé	https://www.amazon.fr/Adaptateur-Plugable-*6*3%49%*3%49phones-intelligents-
OceanBox Dev/Recette	Alimentation	Prévisionnel	Kubbi	2	7,90 €	20,00%	15,80 €	15.3W USB-C	https://www.kubii.fr/14-chargeurs-alimentations-raspberry/1754-alimentation-micro-usb-5v-25a-pour-raspberry-pi-3-kubii-3272496006218.html
OceanBox Dev/Recette	Capteur de mouvement	Prévisionnel	Ebay ?	2	2,00 €	20,00%	4,00 €	Prix en fonction de la qualité souhaité	
OceanBox Dev/Recette	Carte SD	Prévisionnel	Amazon ?	2	82,98 €	20,00%	165,96 €	Il en existe des moins chers mais surment pas assez rapide pour lire de la 2k	Mc512Gaeu-Carte-

Gestion – Feuille d'heure

Mois	Gestion de projet	Infratstrutrure	Developpment	Formation	Documentation	Total	Prix/jour*	Prix Total
Novembre	5h	0h	0h	5h	0h	10h	230 €	329 €
Décembre	5h	0h	0h	5h	0h	10h	230 €	329 €
Janvier	10h	10h	0h	10h	8h	30h	230 €	986 €
Février	5h	15h	0h	10h	0h	30h	230 €	986 €
Mars	5h	15h	20h	10h	0h	50h	230 €	1 643 €
Avril	3h	8h	35h	2h	0h	48h	230 €	1 577 €
Mai	1h	10h	35h	5h	0h	51h	230 €	1 676 €
Juin	3h	2h	20h	3h	10h	28h	230 €	920 €
Total	37h	60h	110h	50h	18h	257h	230 €	8 444 €

**forfait jour devloppeur java junior constaté à Paris*

Users Stories (US)

Nom : DevOps

Sprint : 0

Date : 14/02/2020

Description :

Les développeurs souhaitent qu'une chaine DevOps industrialisé soit mis en place. En d'autres termes, à chaque nouvelle fonctionnalité, un push sur la branche master entraine le build des dernières sources puis le déploiement de ce build sur l'ensemble du parc d'OceanBox. Ainsi, peut importe ou ce trouve les OceanBox, elles pourront toujours avoir les dernières mises à jour sans aucune lourdeur dans le déploiement.

Je suis développeur je veux déployer automatiquement les dernières fonctionnalités avec Git/Jenkins/Ansible

Méthode de test :

Nom : US2 - Affichage vidéo

Sprint : 1

Date : X/X/X

Description :

Quand le programme est lancé, il affiche à l'écran la vidéo de fonds marin stockée pendant un temps donné.

Une fois finit la vidéo est relancée pendant un temps donné.

La vidéo doit être décompressée avant d'être lancée.

Je suis : locataire d'une OceanBox

Je veux : regarder la vidéo stockée sur mon OceanBox

Sur : ma télévision

Critères de validation des tests :

La vidéo doit s'afficher à l'écran et rester active le nombre de minutes paramétré. La vidéo doit être relancée à chaque fin de vidéo.

Nom : Plug&Play

Sprint : 1

Date :

Description :

Quand un utilisateur branche l'OceanBox en HDMI, en alimentation et en câble Ethernet le système est fonctionnel.
La vidéo s'affiche et est relancée comme explicité à l'US2.

Si le câble Ethernet n'est pas branché, l'utilisateur peut voir la vidéo la plus récente disponible.
S'il s'agit de la première connexion de l'OceanBox, l'utilisateur peut voir une vidéo par défaut.

Je suis : un locataire de l'OceanBox

Je veux : installer mon OceanBox

En : en branchant simplement l'OceanBox aux différentes connectiques.

Critères de validation des tests :

Le système se lance à l'allumage de l'OceanBox.

Une vidéo est toujours diffusée si l'alimentation et le câble vidéo sont branchés.

Nom : US4 - Synchronisation fuseau horaire

Sprint : 2

Date : X/X/X

Description :

Pour annuler l'effet du décalage horaire entre le lieu de captation et le lieu de diffusion et ainsi augmenter l'immersion la vidéo doit être lancée en horaire décalé.

La vidéo est lancée à 6h heure locale et est diffusée pendant 14h. Puis la vidéo est relancée à 20h pendant 10h jusqu'à 6h.

Je suis : un spectateur

Je veux : que la lumière dans la vidéo soit en cohérence avec mon heure locale

Via : mon fuseau horaire

Critères de validation des tests :

Le cycle de vie d'une vidéo est respecté est dépends bien du fuseau horaire.

Nom : US5 - Récupération journalière de la vidéo

Sprint : 2

Date : X/X/X

Description :

Chaque jour une nouvelle vidéo est diffusée sur l'OceanBox. Cette vidéo est toujours la dernière tournée dans l'idéal celle de la veille.

La vidéo doit être récupérée entre 2h et 4h du matin pour éviter la saturation du réseau local et optimiser la consommation des infrastructures réseau en accord avec l'écoconception.

Je suis : un spectateur

Je veux : voir des vidéos différentes tous les jours et des vidéos d'actualité (idéalement la veille)

En : Utilisant le réseau en horaire creuse

Critères de validation des tests :

Une vidéo est récupérée chaque jour à heure fixe depuis le dépôt vidéo.

Cette vidéo est la plus récente.

Nom : US6 - Gestion de la veille par détecteur de mouvement

Sprint : 3

Date : X/X/X

Description :

Tant qu'un utilisateur est détecté à intervalle régulier l'OceanBox diffuse le flux.

Quand aucun utilisateur est détecté pédant 20mins l'OceanBox se met en veille jusqu'à ce qu'un utilisateur soit détecté. A ce moment l'OceanBox se remet en marche et diffuse le flux.

Je suis : un locataire de l'OceanBox

Je veux : que l'OceanBox soit autonome dans sa gestion de la consommation électrique. *Je ne veux pas m'occuper de l'allumer et de l'éteindre. Je souhaite cependant qu'elle optimise sa consommation d'électricité.*

Avec : un détecteur de mouvement

Critère de validation des tests :

Un signal de mouvement ajoute 20mins au temps de diffusion. Quand le temps de diffusion est de 0 la box envoie un signal de mise en veille. Quand un signal de mouvement a été détecté la box se rallume pour 20mins.

Nom : US7 - Gestion de la télévision

Sprint : 3

Date : X/X/X

Description :

Si aucune source de la télévision est utilisée l'OceanBox change la source de la télévision vers son port de diffusion et diffuse son flux vidéo.

(Si une source devient active, l'OceanBox change la source de la télévision vers le port actif)

La mise en veille de l'OceanBox entraine l'extinction de la télévison par l'OceanBox. La sortie de veille entraine l'allumage de la télévision par l'OceanBox.

Je suis : un utilisateur de la télévision

Je veux : que l'OceanBox me donne l'impression d'un écran de veille de la télévision et éviter les conflits avec les différentes sources actives. C'est pourquoi je veux que l'OceanBox soit la source la moins prioritaire.

Je veux également : que l'extinction de l'OceanBox entraine l'extinction de la télévision par soucis d'économie d'énergie.

Critère de validation des tests :

L'OceanBox est la source la moins prioritaire.

Une mise en veille entraine une extinction de la télévision.

Une sortie de veille entraine l'allumage de la télévision.

Nom : Sp4 - US8 - Paramétrage

Sprint : 4

Date : X/X/X

Description :

Les paramètres suivants indiqués dans le panneau de configuration des clients doivent être pris en compte par l'OceanBox :

- L'heure du levé indique le paramétrage de l'heure de lancement de la nouvelle vidéo
- Les périodes de désactivation indiquent les périodes où la Raspberry est mise en veille sans l'intervention du détecteur de mouvements.
- Boolean indiquant l'activation/désactivation de l'OceanBox (pour départ en vacances par exemple)
- Le flux vidéo indiquant le flux de quelle caméra l'OceanBox doit diffuser (cas de plusieurs flux disponibles)
- Boolean indiquant l'activation/désactivation du bandeau
- Le temps avant la veille de l'OceanBox si aucun mouvement n'est détecté.

Je suis : un locataire de l'OceanBox

Je veux : paramétrer ma Box

Nom : Sp4 - US9 Désactivation temporaire de la Box

Sprint : 4

Date : X/X/X

Description :

Un signal (à définir) entraine la désactivation de l'OceanBox pour un temps à définir.

Je suis : un spectateur

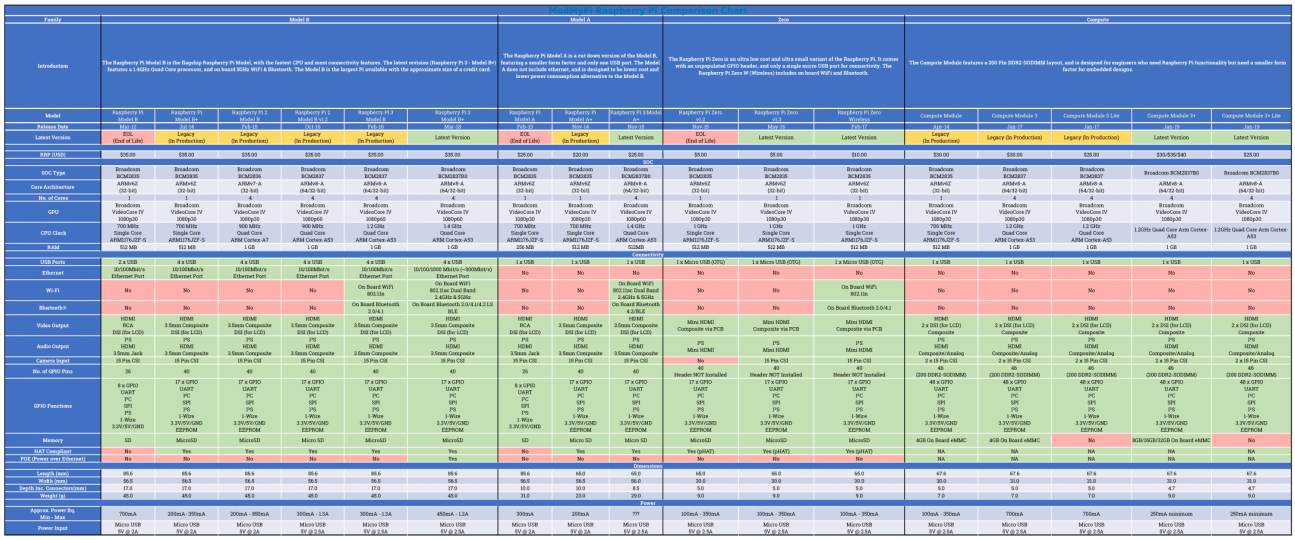
Je veux : désactiver l'OceanBox pendant un temps à définir

Avec : à définir

Critères de validation des tests :

L'OceanBox se met bien en veille le temps donné après le signal (à définir)

Figures



Family	Model B						Model A			Zero			Compute				
Introduction	The Raspberry Pi Model B is the flagship Raspberry Pi Model, with the fastest CPU and most connectivity features. The latest revision (Raspberry Pi 3 - Model B+) features a 1.4GHz Quad Core processor, and on board 802.11 WiFi & Bluetooth. The Model B is the largest Pi available with the approximate size of a credit card.						The Raspberry Pi Model A is a cut down version of the Model B, featuring a smaller form factor and only one USB port. The Model A does not include ethernet, and is designed to be lower cost and lower power consumption alternative to the Model B.			The Raspberry Pi Zero is an ultra low cost and ultra small variant of the Raspberry Pi. It comes with an unpopulated GPIO header, and only a single micro USB port for connectivity. The Raspberry Pi Zero W (Wireless) includes on board WiFi and Bluetooth.			The Compute Module features a 200 Pin DDR2-SODIMM layout, and is designed for engineers who need Raspberry Pi functionality but need a smaller form factor for embedded designs.				
Model	Raspberry Pi Model B	Raspberry Pi Model B+	Raspberry Pi 2 Model B	Raspberry Pi 2 Model B v1.2	Raspberry Pi 3 Model B	Raspberry Pi 3 Model B+	Raspberry Pi Model A	Raspberry Pi Model A+	Raspberry Pi 3 Model A+	Raspberry Pi Zero v1.3	Raspberry Pi Zero v1.3	Raspberry Pi Zero Wireless	Compute Module	Compute Module 3	Compute Module 3 Lite	Compute Module 3+	Compute Module 3+ Lite
Release Date	Mar-11	Jul-14	Feb-15	Oct-16	Feb-16	Mar-18	Feb-13	Nov-14	Nov-18	Nov-15	May-16	Feb-17	Apr-14	Jan-17	Jan-17	Jan-19	Jan-19
Latest Version	EOL (End of Life)	Legacy (In Production)	Legacy (In Production)	Legacy (In Production)	Legacy (In Production)	Latest Version	EOL (End of Life)	Legacy (In Production)	Latest Version	EOL (End of Life)	Latest Version	Latest Version	Legacy (In Production)	Legacy (In Production)	Legacy (In Production)	Latest Version	Latest Version
RRP (USD)	$35.00	$35.00	$35.00	$35.00	$35.00	$35.00	$25.00	$20.00	$25.00	$5.00	$5.00	$10.00	$30.00	$30.00	$25.00	$35/$35/$40	$25.00
SOC																	
SOC Type	Broadcom BCM2835 ARMv6Z (32-bit)	Broadcom BCM2835 ARMv6Z (32-bit)	Broadcom BCM2836 ARMv7-A (32-bit)	Broadcom BCM2837 ARMv8-A (64/32-bit)	Broadcom BCM2837 ARMv8-A (64/32-bit)	Broadcom BCM2837B0 ARMv8-A (64/32-bit)	Broadcom BCM2835 ARMv6Z (32-bit)	Broadcom BCM2835 ARMv6Z (32-bit)	Broadcom BCM2837B0 ARMv8-A (64/32-bit)	Broadcom BCM2835 ARMv6Z (32-bit)	Broadcom BCM2835 ARMv6Z (32-bit)	Broadcom BCM2835 ARMv6Z (32-bit)	Broadcom BCM2835 ARMv6Z (32-bit)	Broadcom BCM2837 ARMv8-A (64/32-bit)	Broadcom BCM2837 ARMv8-A (64/32-bit)	Broadcom BCM2837B0 ARMv8-A (64/32-bit)	Broadcom BCM2837B0 ARMv8-A (64/32-bit)
No. of Cores	1	1	4	4	4	4	1	1	4	1	1	1	1	4	4	4	4
GPU	Broadcom VideoCore IV 1080p30 700 MHz Single Core	Broadcom VideoCore IV 1080p30 700 MHz Single Core	Broadcom VideoCore IV 1080p30 600 MHz Quad Core	Broadcom VideoCore IV 1080p60 900 MHz Quad Core	Broadcom VideoCore IV 1080p60 1.0 GHz Quad Core	Broadcom VideoCore IV 1080p60 1.4 GHz Quad Core	Broadcom VideoCore IV 1080p30 700 MHz Single Core	Broadcom VideoCore IV 1080p30 700 MHz Single Core	Broadcom VideoCore IV 1080p60 1.4 GHz Quad Core	Broadcom VideoCore IV 1080p30 1 GHz Single Core	Broadcom VideoCore IV 1080p30 1 GHz Single Core	Broadcom VideoCore IV 1080p30 1 GHz Single Core	Broadcom VideoCore IV 1080p30 700 MHz Single Core	Broadcom VideoCore IV 1080p30 1.2 GHz Quad Core	Broadcom VideoCore IV 1080p30 1.2 GHz Quad Core	Broadcom VideoCore IV 1080p30 1.2GHz Quad Core Arm Cortex-A53	Broadcom VideoCore IV 1080p30 1.2GHz Quad Core Arm Cortex-A53
CPU Clock	ARM1176JZF-S	ARM1176JZF-S	ARM Cortex-A7	ARM Cortex-A53	ARM Cortex-A53	ARM Cortex-A53	ARM1176JZF-S	ARM1176JZF-S	ARM Cortex-A53	ARM1176JZF-S	ARM1176JZF-S	ARM1176JZF-S	ARM1176JZF-S	ARM Cortex-A53	ARM Cortex-A53		
RAM	512 MB	512 MB	1 GB	1 GB	1 GB	1 GB	256 MB	512 MB	512MB	512 MB	512 MB	512 MB	512 MB	1 GB	1 GB	1 GB	1 GB
Connectivity																	
USB Ports	2 x USB	4 x USB	4 x USB	4 x USB	4 x USB	4 x USB	1 x USB	1 x USB	1 x USB	1 x Micro USB (OTG)	1 x Micro USB (OTG)	1 x Micro USB (OTG)	1 x USB	1 x USB	1 x USB	1 x USB	1 x USB
Ethernet	10/100Mbit/s Ethernet Port	10/100Mbit/s Ethernet Port	10/100Mbit/s Ethernet Port	10/100Mbit/s Ethernet Port	10/100Mbit/s Ethernet Port	10/100/1000 Mbit/s (~300Mbit/s) Ethernet Port	No	No	No	No	No	No	No	No	No	No	No
Wi-Fi	No	No	No	No	On Board WiFi 802.11n	On Board WiFi 802.11ac Dual Band 2.4GHz & 5GHz	No	No	On Board WiFi 802.11n	No	No	On Board WiFi 802.11n	No	No	No	No	No
Bluetooth	No	No	No	No	On Board Bluetooth 2.0/4.1	On Board Bluetooth 2.0/4.1/4.2 LS BLE	No	No	On Board Bluetooth 4.0/BLE	No	No	On Board Bluetooth 2.0/4.1	No	No	No	No	No
Video Output	HDMI RCA DSI (for LCD)	HDMI 3.5mm Composite DSI (for LCD)	HDMI 3.5mm Composite DSI (for LCD)	HDMI 3.5mm Composite DSI (for LCD)	HDMI 3.5mm Composite DSI (for LCD)	HDMI 3.5mm Composite DSI (for LCD)	HDMI RCA DSI (for LCD)	HDMI 3.5mm Composite DSI (for LCD)	HDMI 3.5mm Composite DSI (for LCD)	Mini HDMI Composite via PCB	Mini HDMI Composite via PCB	Mini HDMI Composite via PCB	HDMI 2 x DSI (for LCD) Composite	HDMI 2 x DSI (for LCD) Composite	HDMI 2 x DSI (for LCD) Composite	HDMI 2 x DSI (for LCD) Composite	HDMI 2 x DSI (for LCD) Composite
Audio Output	I²S HDMI 3.5mm Jack	I²S HDMI 3.5mm Composite	I²S HDMI 3.5mm Composite	I²S HDMI 3.5mm Composite	I²S HDMI 3.5mm Composite	I²S HDMI 3.5mm Composite	I²S HDMI 3.5mm Jack	I²S HDMI 3.5mm Composite	I²S HDMI 3.5mm Composite	I²S Mini HDMI	I²S Mini HDMI	I²S Mini HDMI	I²S HDMI Composite/Analog	I²S HDMI Composite/Analog	I²S HDMI Composite/Analog	I²S HDMI Composite/Analog	I²S HDMI Composite/Analog
Camera Input	15 Pin CSI	15 Pin CSI	15 Pin CSI	15 Pin CSI	15 Pin CSI	15 Pin CSI	15 Pin CSI	15 Pin CSI	15 Pin CSI	No	15 Pin CSI	15 Pin CSI	2 x 15 Pin CSI	2 x 15 Pin CSI	2 x 15 Pin CSI	2 x 15 Pin CSI	2 x 15 Pin CSI
No. of GPIO Pins	26	40	40	40	40	40	26	40	40	40 Header NOT Installed	40 Header NOT Installed	40 Header NOT Installed	46 (200 DDR2-SODIMM)	46 (200 DDR2-SODIMM)	46 (200 DDR2-SODIMM)	46 (200 DDR2-SODIMM)	46 (200 DDR2-SODIMM)
GPIO Functions	8 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	8 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	17 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	46 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	46 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	46 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	46 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM	46 x GPIO UART I²C SPI I²S 1-Wire 3.3V/5V/GND EEPROM
Memory	SD	MicroSD	MicroSD	Micro SD	MicroSD	MicroSD	SD	Micro SD	Micro SD	MicroSD	MicroSD	MicroSD	4GB On Board eMMC	4GB On Board eMMC	No	8GB/16GB/32GB On Board eMMC	No
HAT Compliant	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes (pHAT)	Yes (pHAT)	Yes (pHAT)	NA	NA	NA	NA	NA
POE (Power over Ethernet)	No	No	No	No	No	Yes	No	No	No	No	No	No	NA	NA	NA	NA	NA
Dimensions																	
Length (mm)	85.6	85.6	85.6	85.6	85.6	85.6	85.6	65.0	65.0	65.0	65.0	65.0	67.6	67.6	67.6	67.6	67.6
Width (mm)	56.5	56.5	56.5	56.5	56.5	56.5	56.5	56.5	56.5	30.0	30.0	30.0	30.0	31.0	31.0	31.0	31.0
Depth (inc. Connectors)(mm)	17.0	17.0	17.0	17.0	17.0	17.0	10.0	10.0	8.5	5.0	5.0	5.0	5.0	5.0	5.0	4.7	4.7
Weight (g)	45.0	45.0	48.0	45.0	48.0	45.0	31.0	23.0	29.0	9.0	9.0	9.0	7.0	7.0	7.0	9.0	9.0
Power																	
Approx. Power Eq. Min - Max	700mA	200mA - 350mA	200mA - 850mA	300mA - 1.3A	300mA - 1.3A	450mA - 1.2A	300mA	200mA	???	100mA - 350mA	100mA - 350mA	100mA - 350mA	100mA - 350mA	700mA	700mA	250mA minimum	250mA minimum
Power Input	Micro USB 5V @ 2A	Micro USB 5V @ 2A	Micro USB 5V @ 2A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2A	Micro USB 5V @ 2A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A	Micro USB 5V @ 2.5A

Tableau 1 Raspberry by comparison - https://thepihut.com/blogs/raspberry-pi-roundup/raspberry-pi-comparison-table

Bibliographie

EcoConception

Page Wikipédia dédiée à l'écoconception - https://fr.wikipedia.org/wiki/%C3%89coconception

Restitution du travail de Greenpeace sur la pollution numérique – 10 janvier 2017 - https://www.greenpeace.fr/la-pollution-numerique/

Site de la communauté française de la green IT - https://www.greenit.fr/

Méthode AGILE

Page Wikipédia dédiée à la méthode AGILE - https://fr.wikipedia.org/wiki/M%C3%A9thode_agile

Points problématiques de la méthode Scrum - https://blog.myagilepartner.fr/index.php/2019/12/20/le-sprint-objectif-un-scrum-implementation-point-of-failure/

How to implement Scrum - https://www.knowledgehut.com/blog/agile/implementing-scrum-in-organizations

Page Wikipédia dédiée à la méthode Scrum - https://fr.wikipedia.org/wiki/Scrum_(d%C3%A9veloppement)

DevOps

Page Wikipédia dédié au DevOps - https://fr.wikipedia.org/wiki/Devops

Page Git de conseil en DevOps - https://gist.github.com/jpswade/4135841363e72ece8086146bd7bb5d91

Flux

Blog Raspberry concernant la création d'un flux streaming - http://www.magdiblog.fr/divers/raspberry-pi-camera-5-facons-de-faire-du-streaming/

Documentation officielle Raspberry concernant les flux ftp - https://www.raspberrypi.org/documentation/remote-access/ftp.md

notre-plantet.info - Le Streaming est-il un gouffre énergétique ? –https://www.notre-planete.info/actualites/247-streaming-emissions-CO2

Raspberry et Arduino

The PIHUB – Site de vente de produits autour de la Raspberry - https://thepihut.com/

Comparaison Arduino vs Raspberry - https://www.electronicshub.org/raspberry-pi-vs-arduino/

Choix du hardware

https://makezine.com/2015/12/04/admittedly-simplistic-guide-raspberry-pi-vs-arduino/

https://thepihut.com/blogs/raspberry-pi-roundup/raspberry-pi-comparison-table

https://lunarfrog.com/blog/devops-desktop-software-development

https://medium.com/splunkuserdeveloperadministrator/using-ansible-pull-in-ansible-projects-ac04466643e8

https://www.stavros.io/posts/automated-large-scale-deployments-ansibles-pull-mo/