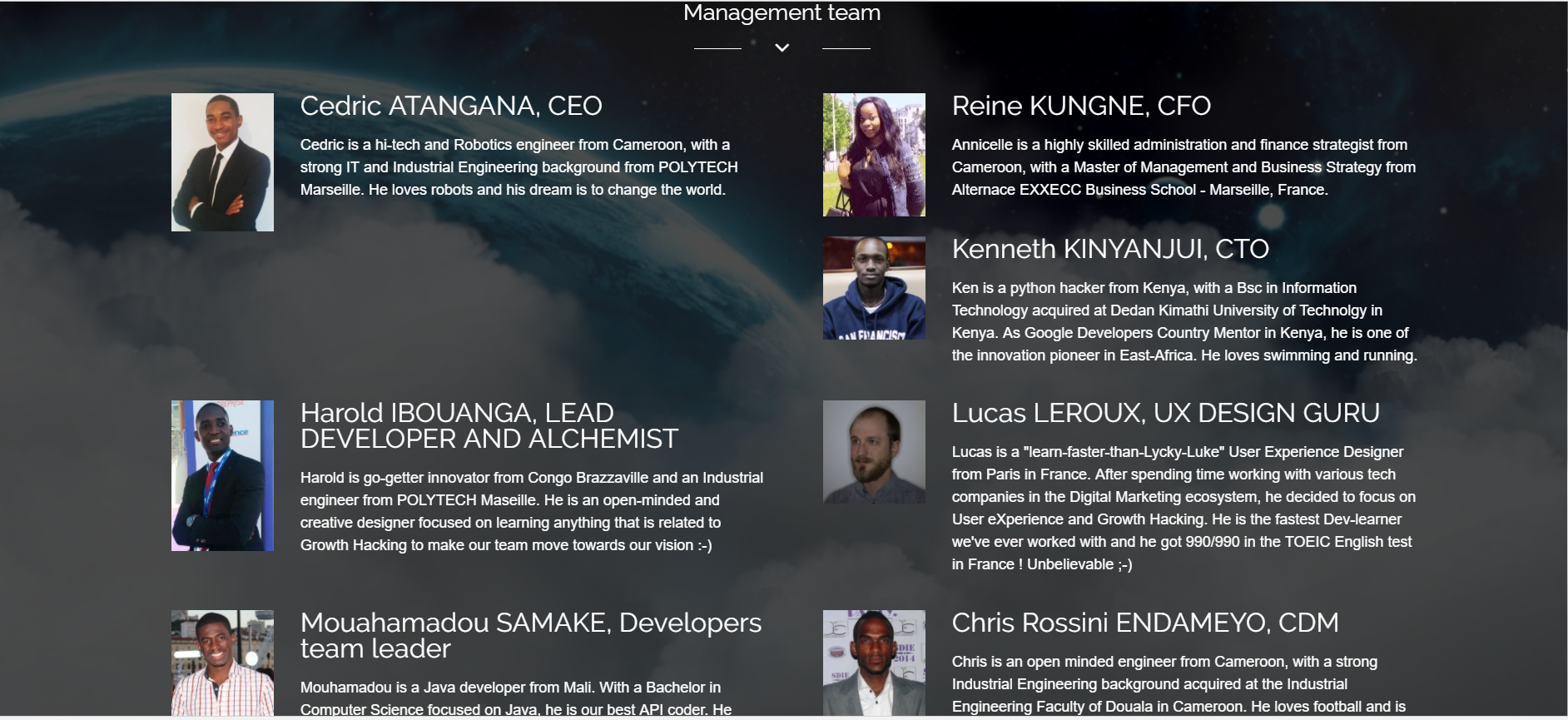
1. Présentation de l’entreprise :

L’entreprise INFINITY SPACE est une société par action simplifiée (SAS) au capital de 1.000€ qui a été fondée le 23 Mars 2015 par 7 jeunes diplômés issus de formations complémentaires : Ingénierie, administration, finance et Marketing.

De nos jour INFINITY SPACE est composée des jeunes expérimentés de 9 pays sur 4 continents: le Cameroun, le Kenya, le Togo, le Congo Brazzaville, le Mali, Madagascar, la France, la Chine et les États-Unis avec différents horizons tels que l'ingénierie, l'informatique, les finances, l'administration, la gestion, le marketing, Les médias numériques, Juridique, Recherche et développement, Développement des affaires, etc. 6 personnes en interne et les autres en externe qui travaillent en collaboration avec l’équipe de développement, parmi ces 6 personne deux des co-fondateurs, deux salariés en temps plein et un alternant y travaillent : son CEO **M. Cédric ATANGANA** et sa CFO (Chief Financial Officer), **Mlle Annicelle Reine KUNGNE**, + 3 stagiaires.



**Image N°2** : team.

**Source** : http://www.infinityspace.fr/.  
INFINITY SPACE est une FinTech d'innovation sociale né à Douala (Cameroun), développé à Nairobi (Kenya) et basé à Marseille (France).

INFINITY SPACE est une FinTech startup innovante basée à Marseille, plus précisément dans la pépinière de Marseille innovation.

Marseille Innovation est un important centre d’entreprises et d’innovation (CEEI) de la région PACA qui opère 3 pépinières et hôtels d’entreprises, 1 incubateur international sur la Méditerranée et 1 programme d’accélération de PME. Plus de 160 startups sont accompagnées au quotidien.

Créé en 1996(1) par Christian Rey, Marseille Innovation est le plus grand centre européen d’entreprises et d’innovation (CEEI) de la région PACA : **3 pépinières et hôtels d’entreprises** pour **héberger et accompagner des start-up** en phase de démarrage dans les domaines du numérique, du multimédia, de l’audiovisuel, des objets connectés et plus généralement des sciences de l’ingénieur. Depuis toujours Marseille innovation cultive une double ambition : **être à la fois accélérateur de croissance et attracteur de talents**. Ce sont plus de **100 start-up** qui nous font confiance aujourd’hui et plus de 500 depuis sa création.



**Image N°1** : Hôtel technologique de château gombert.

**Source :** google maps.

Une pépinière dédiée aux start-up du numérique et des sciences de l’ingénieur.

Sur le Technopôle Marseille Provence à Château-Gombert au cœur des écoles d'ingénieurs et des laboratoires de recherche, la pépinière héberge et accompagne des start-up issues des secteurs du numérique et des sciences de l'ingénieur. Elle offre un cadre privilégié, alliance réussie de l’innovation technologique et de la qualité de vie, un véritable lieu d’échange entre l’entreprise, la recherche, les écoles dans un cadre de vie fortement doté en services.

INFINITY SPACE développe [www.wecashup.com](http://www.wecashup.com), un système de paiement mobile universel qui permet aux personnes non-bancarisés dans le monde de payer en ligne avec leur téléphone portable comme moyen de paiement sans carte bancaire.

En effet, son marché couvre près de 2 milliards de personnes non-bancarisés dans le monde, localisés en Amérique latine (300 millions), Asie du Sud (500 millions), Europe de l’Est (200 millions) et en Afrique (800 millions) et qui ne peuvent pas faire d’achats sur internet parce qu’ils n’ont pas ni comptes bancaires classiques, ni de cartes bancaires.

Son focus stratégique pour le moment se tourne donc vers l’Afrique, son plus gros marché et continent présentant le plus gros challenge.

INFINITY SPACE s’est alors fixé pour mission principale de challenger l’impossible, faire des impossibles des possibles.

La startup INFINITY SPACE construit à cet effet un écosystème e-business pour aider les entreprises à accroître leurs activités à l’international et pour permettre aux personnes exclues du système bancaire traditionnel de faire des achats en ligne avec leur téléphone portable comme moyen de paiement sans carte bancaire ni compte bancaire.

Pour permettre à toutes ces personnes de faire les payements en ligne par leurs téléphone mobile, la jeune startup travaille avec les opérateurs télécom qui font mobile money.

Mobile money : Le Mobile Money est un service de portefeuille mobile, disponible dans de nombreux pays, qui permet à ses utilisateurs d’épargner, envoyer et recevoir de l’argent sur leur téléphone portable.

Le transfert **d’argent par téléphone mobile** est né au Kenya qui l’ont nommé **M-PESA**.

**M** : comme mobile et **PESA** : comme l’argent en swahilie.

Lancé en 2007 au Kenya par l’opérateur SAFARICOM (Filiale du groupe VODAFONE), dont le succès considérable a eu un retentissement mondial. Aujourd’hui, presque tout le monde utilise M-PESA au Kenya et notamment les personnes **sans compte bancaire**. On estime qu’environ 80 % des paiements par téléphone mobile réalisés dans le monde sont faits en Afrique de l’Est ! (Kenya, Tanzanie, Lesotho, Mozambique notamment).

Ceci dit, plus de 11 services de**Mobile Money** existent à ce jour en Afrique : Barthi Airtel, Etisalat, Millicom, MTN, Ooredoo, Orange, STC, Vodafone, Tigo-cash, Glo-Money etc. (2)

Les paiements effectués avec le Mobile Money sont sécurisés et simples et c’est donc une alternative populaire aux comptes bancaires. Le Mobile Money est disponible sur les Smartphones et aussi sur les téléphones numériques**.**

**La jeune startup a interconnecté tous ces opérateurs télécom pour en faire une API (Application programming Interface), afin de faciliter le payement à toutes ces personnes concernées, n’importe où, où ils se retrouvent chez n’importe quel opérateur, ils peuvent de même faire un split-payement (de payer une somme par groupe).**

Dans le cas de l'Afrique, les problèmes les plus importants auxquels ces PME sont confrontées dans la vente en ligne sont :

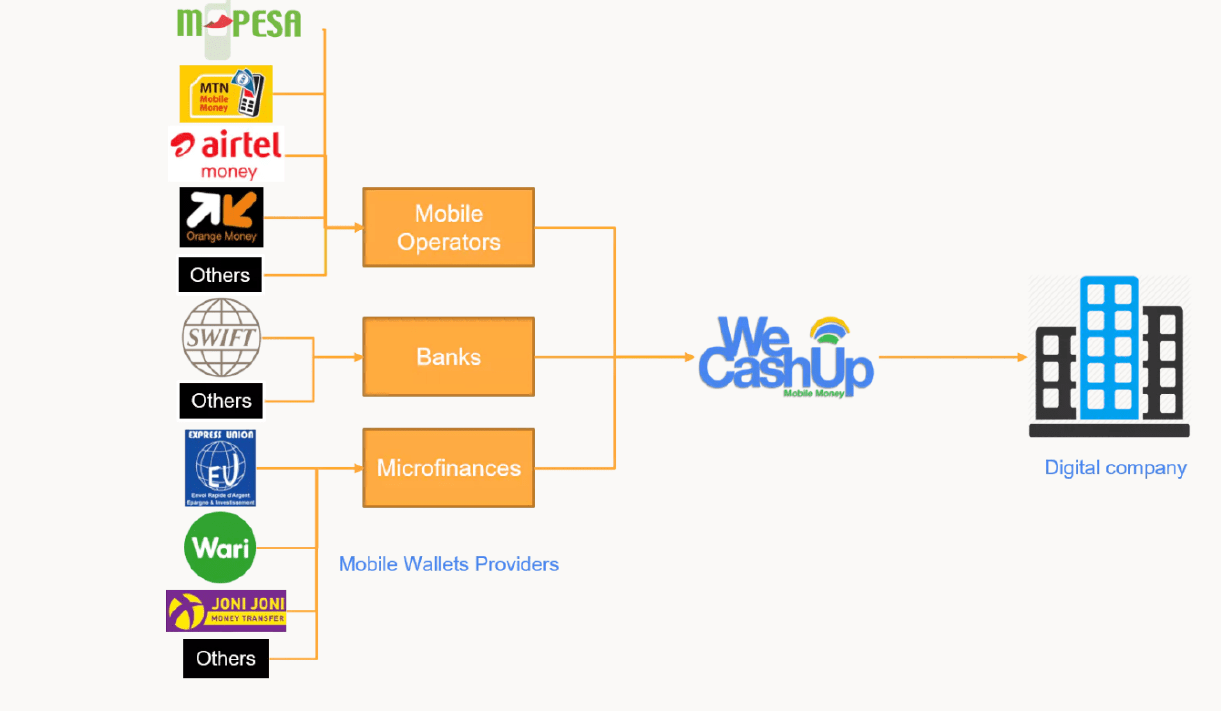
1. **- l’infrastructure d’adresse physique (système d’adressage de domicile) est quasi-inexistant et les livraisons pour le e-commerce sont difficiles à réaliser**
2. **- plus de 80% de la population africaine (800 millions de personnes) est très faiblement bancarisée et ne peut pas acheter en ligne.**

Donc, pour trouver une solution rapide à ces deux questions, INFINITY SPACE a regroupé les technologies adéquates pour aider les gens à acheter facilement et se faire livrer à temps. Cela va considérablement aider les PMEs Française et internationales à se concentrer sur le cœur de leur métier sans se soucier de la façon d'atteindre plus de clients, comment ils seront payés et comment ils vont livrer des marchandises à des clients qui sont très loin du lieu de leurs magasins physiques.

INFINITY SPACE propose deux activités principales :

1. ***WeShopUp*** : Une Marketplace d’achat participatif où les gens achètent en groupe et se font livrer individuellement (une plateforme de crowdpurchasing c’est-à-dire crowdfunding + e-commerce), développé au début de la création de l’entreprise, stoppé par les problèmes dont je vous ai expliqué au page précédente, plus 55% des utilisateurs de cette Platform se trouvaient en Afrique d’où le problème de le problème de la livraison intervient, ce qui a pousser la jeune startup de mettre en place une API pour résoudre ce dernier .
2. ***WeCashUp*** : Une plateforme de paiement mobile universelle qui permet aux populations non bancarisées de payer en ligne par Mobile Money, avec leur téléphone portable, et sans carte de crédit. La plateforme de paiement mobile est dédiée principalement aux pays sous ou non bancarisés. En effet, il existe beaucoup de système de paiement mobile suivant différents opérateurs téléphoniques. Souvent, dans un pays on peut avoir par exemple quatre opérateurs téléphoniques qui font du paiement mobile (exemple : Airtel Money, pour l’opérateur téléphonique Airtel / OrangeMoney pour l’opérateur téléphonique Orange / MobileMoney pour l’opérateur MTN / MoovMoney pour l’opérateur téléphonique Moov…) mais ils ne communiquent pas entre eux. Ainsi, une personne utilisant ce système doit posséder à elle seule les quatre cartes Sim de ces opérateurs. Ce qui n’est pas pratique d’autant plus que dans certains pays, on peut quelque fois atteindre douze opérateurs. Ainsi, WeCashUp est la solution qui offre de l’interopérabilité à ces systèmes. De ce fait, en détenant une seule carte Sim d’un opérateur quelconque, une personne peut transférer et recevoir de l’argent de tous les autres opérateurs et même faire des achats sur n’importe quel site d’un marchand qui a embarqué la solution WeCashUp.
3. Travaux réalisés :

A mon arrivé, l’INFINITY SPACE venait juste de lancer son projet phare, WeCashUp la plateforme de payement mobile, pour vous donner une brève définition de WeCashUp de façon technique WeCashUp est comme le contrôleur, son architecture ressemble à l’architecture MVC autrement dit l’intermédiaire entre le client et le provider, eh oui ! je vous montrerai un peu plus bas un schéma pour plus de détail. La technologie WeCashUp est conçu d’une façon à détecter le blanchichement d’argent ou encore détection de fraude, WCU est la combinaison de plusieurs technologie différente ce qui fait sa particularité, voici un schéma qui donne une image de WeCashUp dans son ensemble :



**Providers**

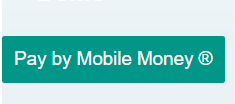
**Banks Marchants**

**Schéma N°1** : wecashup-Archtype.

**Banks** : IFINITY SPACE n’est pas une institution financière, donc pour pouvoir collecter ses fonds, elle travaille avec plusieurs Banks, dont une partie de ses transactions sont transférées directement dans son compte auprès de ces Banks, qui gagnent un pourcentage sur les transactions mensuelle. Les **Microfinances** sont à la fois une banque et un opérateur télécom, c’est-à-dire, elles peuvent faire leurs propres transactions et en même temps garder ses fonds, il est beaucoup plus facile pour l’INFINITY SPACE, de travailler eux, car elles seront à la fois leur client et en même temps leur banque.

**Customers** : sont tous les clients qui viendront sur le site du marchand pour un besoin quelconque, WeCashUp ne gère ces pas clients, elle s’occupe tout simplement à vérifier leurs informations dont les marchands a envoyé au serveur de WeCashUp.

**Marchant** : sont tous les sites d’e-commerce qui ont intégré la solution de WeCashUp, l’intégration de WeCashUp est aussi facile qu’à intégrer un seul bouton qui ressemble à ceci :



**Mobile Money :** est désormais la marque de d’IFINITY SPACE, c’est pourquoi au lieu de *paye by WeCashUp*, ils ont mis : **Pay by Mobile Money** car WeCashUp est le nom de la plateforme et Mobile Money est la marque. Les marchands mettront ce bouton sur leur page de payement, les clients qui n’ont pas de carte bancaire, peuvent alors effectuer leur payement via ce bouton. Une fois que le client clique sur **Pay by Mobile Money**, il lui sera demandé quelques informations, je ne vais pas rentrer en détails ici, mais WeCashUp à son tour se charge de vérifier ces informations (numéro de téléphone, montant payé, code secret …), auprès de Providers et informe le marchant de la crédibilité de ce client, et le marchant informe son client soi succès ou faild.

**Providers** : Sont tous les opérateurs télécom qui font le payement par mobile.

Bien vrai que, l’INFINITY SPACE reste une startup pour le moment, mais elle s’est doté d’une méthode de travail qui se présente comme suit :

II.1 Méthode de travail :

1. Outils utilisés :

Travailler en équipe devient de plus en plus facile avec l’utilisation de Git\* et de ses nombreuses plateformes créées par les utilisateurs.

Héberger vos projets un peu partout sur la toile (Mega, FTP perso, MediaFree, FileFactory …. Etc.) et de devoir mettre à jour les liens de téléchargement à chaque mise à jour ! Lorsqu’on se lance dans des projets de développement de grande envergure, il devient alors nécessaire de recourir à un gestionnaire de source. Comme WeCashUp est développée avec la technologie JAVA, donc il nous fallait un éditeur ou encore un IDE (Environnement de Développement Intégré) et aussi un outil de versionning. Pour éviter toute perte de données, ou pouvoir revenir sur la version précédente

La team WeCashUp utilise ***« Bitbucket\* »*** comme gestionnaire de source et le client Git nommé ***« Source Tree\* ».***

Pourquoi gestionnaire de source ***Bitbucket ?***

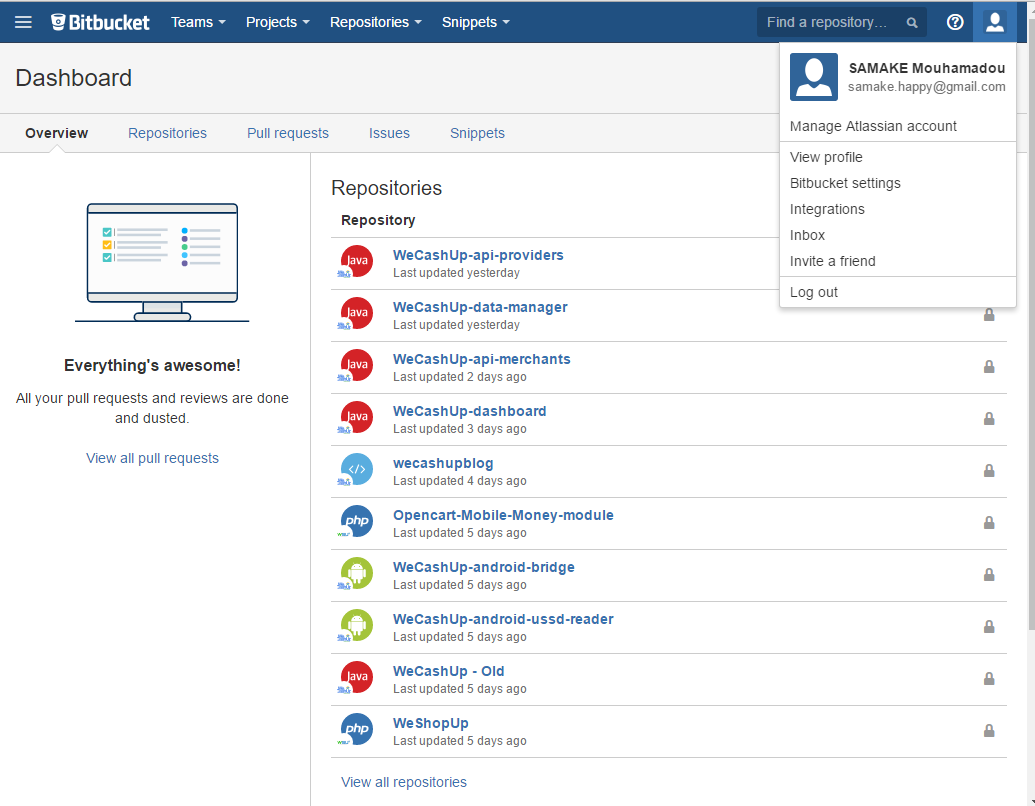
Voice en quelques lignes ce qui caractérise principalement ce gestionnaire de sources :

* Création/ gestion des dépôts privés et publics (dépôts = projet).
* Partage de projets avec d’autres personnes.
* Travail collaboratif avec d’autres développeurs.
* Accepter ou rejeter des « Pull Request » (soumissions d’ajouts de fonctionnalités, correctifs,) d’autre développeurs ayant bifurqués vos projets.
* Modification en ligne ou depuis le client Git « ***Source Tree*** » de votre code source (complet ou indexé).
* Navigation dans vos projets grâce à l’historique de commit.
* Gestion de versionning et des branches du projet.

Il est surtout primordial d’avoir l’assurance qu’un projet est toujours disponible depuis Internet et de pouvoir revenir sur des versions antérieures à n’importe quel moment !

Le principal avantage à tirer de cet outil (***Bitbucket)*** par rapport à son concurrent(GitHub) tient essentiellement dans la possibilité d’héberger autant de dépôts privés que vous voulez.

Accéder aux paramétrages du compte en cliquant en haut à droite sur votre avatar puis « Manage Atlassian Acount » afin d’accéder à ceci :



**Image N°3** : Bitbucket

**Source** : <https://bitbucket.org/product>

Le menu latéral de gauche vous permettra de paramétrer en profondeur votre compte.

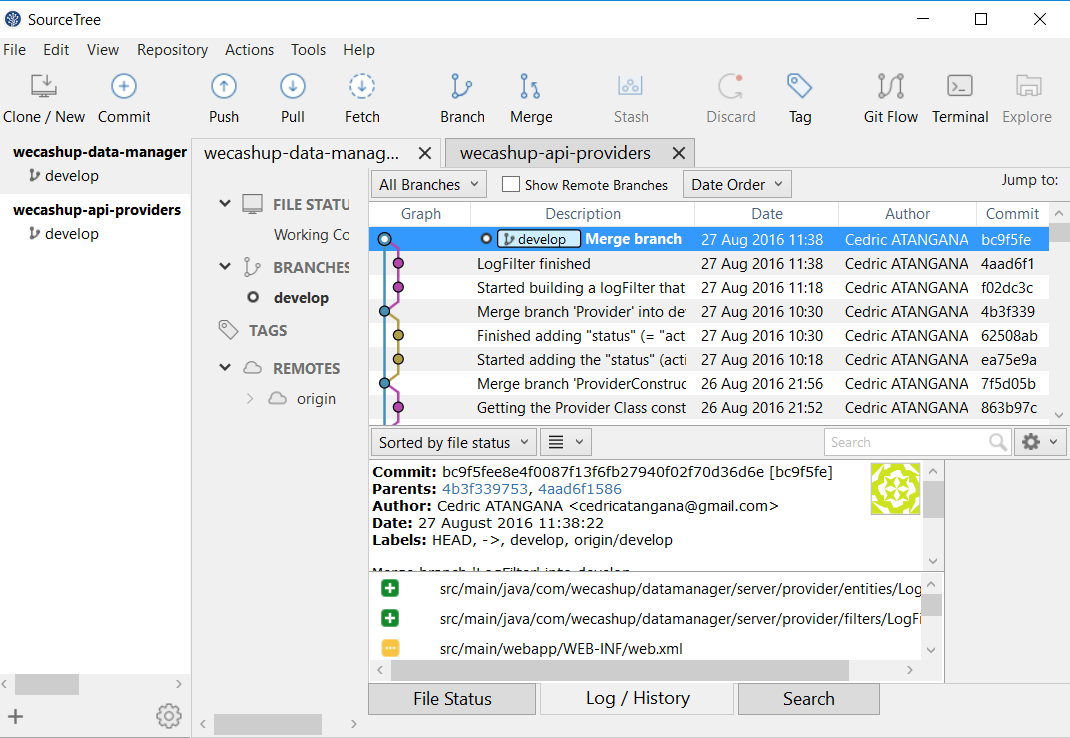
Le menu en haut à gauche donne accès principalement :

* Dashboard : vue d’ensemble (projets et publics), demande d’ajouts, signalement de bugs.
* Repositories : Créer un dépôt, importer un dépôt.
* Create : créer directement un dépôt.

Pouquoi ***Source Tree ?***

SourceTree : est un logiciel client développé par Attlassian, tout comme Bitbucket, permet aux utilisateurs d’accéder sur une interface utilisateur simple. SourceTree peut également être accéder à des services comme GitHub et BitBucket d’Atlassian qui prend en charge Mercurial et Git depuis un certain temps. SourceTree peut être aussi utilisé avec Stash, un Git Repository Management.

La plupart des utilisateurs accèdent à Git via ligne de commande et la plupart des environnements de développement offrent des plugins qui permettent aux utilisateurs d'interagir avec Git. Atlassian dit que 70% de ses clients sont des utilisateurs Windows et que probablement beaucoup de gens ne sont pas habitués à travailler à partir de la ligne de commande. C'est pour cela que SourceTree peut être très utile à ces utilisateurs, compte tenu du fait que ce logiciel prend en charge plus de commandes et offre plus de fonctionnalités que de nombreux plugins sur la plupart des IDE. En plus de cela il est gratuit ☻.



**Image N°4** : SourceTree

**Source** : <http://www.crazyws.fr/dev/sourcetree-un-client-git-et-mercurial-pour-windows-et-mac-N73D6.html>

La team WeCashUp travaille avec IDE Eclipse for java EE developpers, plus précisément Eclipse Néon 2, il est fluide et facile à utiliser, et vient avec plusieurs fonctionnalités tel que : J2EE, Maven, JSP ….

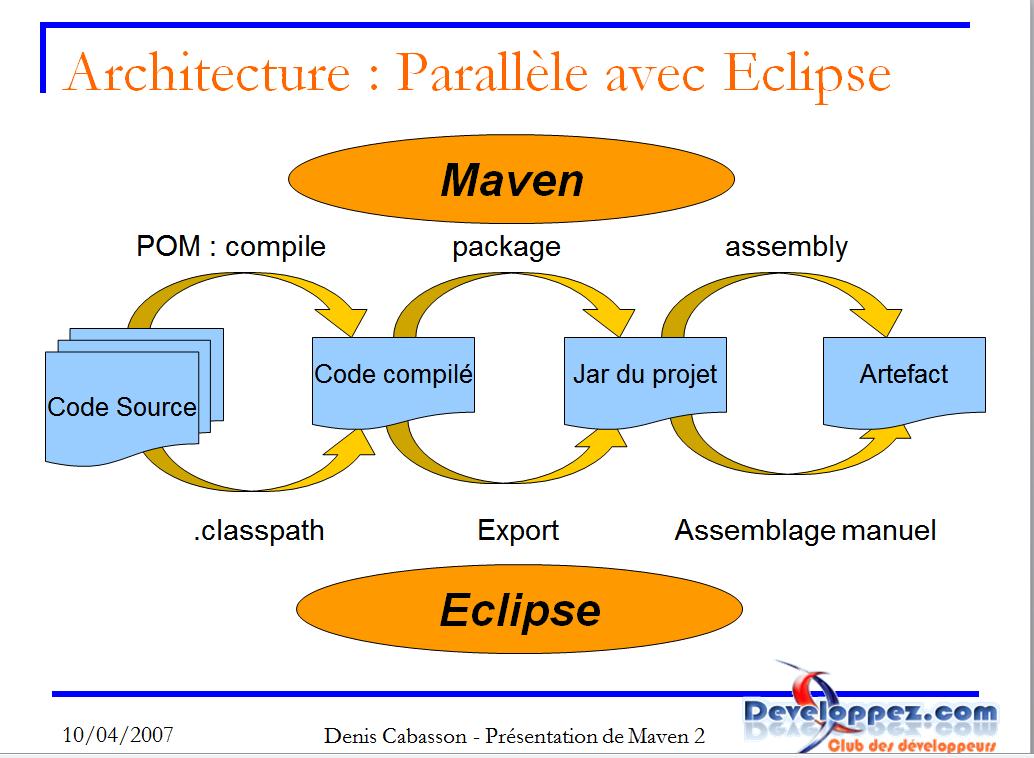
**J2EE** : Java Enterprise Edition, La plateforme Java Entreprise (Java EE) est un ensemble de spécifications coordonnées et pratiques qui permettent des solutions pour le développement, le déploiement, et de la gestion des applications multi-tiers centralisées sur un serveur. Construite sur la plateforme de Java 2 édition standard (Java SE), la plateforme Java EE ajoute les possibilités nécessaires pour fournir une plateforme complète, stable, sécurisée, et rapide de Java au niveau entreprise.   
La plateforme Entreprise fournit un ensemble de services permettant aux composants de dialoguer entre eux :

* HTTP et HTTPS
* Java Transaction API (JTA)
* Remote Method Invocation/Internet Inter-ORB Protocol (RMI/IIOP)
* Java Interface Definition Language (Java IDL)
* Java DataBase Connectivity (JDBC)
* Java Message Service (JMS)
* Java Naming and Directory Interface (JNDI)
* API JavaMail et JAF (JavaBeans Activation Framework)
* Java API for XML Processing (JAXP)
* Java EE Connector Architecture
* Gestionnaires de ressources
* Entreprise Java Beans (EJB)
* Java Server Pages (JSP)
* Servlet
* Java API for XML Web Services (JAX-WS, anciennement JAX-RPC)
* SOAP with Attachments API for Java (SAAJ)
* Java API for XML Registries (JAXR)

**JSP** : Java Server Pages, comme son nom l’indique, les es JavaServerPages (JSP) vous permettent d'insérer des petits bouts de code Java (scriptlets) directement dans du code HTML. Une page JSP est un document qui contient deux types de texte :

* Des données statiques (qui peuvent être exprimées en n'importe quel format texte tel que le HTML, le WML, et le XML) ;
* Des éléments de JSP, qui déterminent comment la page construit le contenu dynamique.

**Maven** (3) :  Maven est un outil de gestion de projet qui comprend un modèle objet pour définir un projet, un ensemble de standards, un cycle de vie, et un système de gestion des dépendances. Il embarque aussi la logique nécessaire à l'exécution d'actions pour des phases bien définies de ce cycle de vie, par le biais de plugins. Lorsque vous utilisez Maven, vous décrivez votre projet selon un modèle objet de projet clair, Maven peut alors lui appliquer la logique transverse d'un ensemble de plugins (partagés ou spécifiques). Voici une image qui m’a aidé à mieux cerner l’architecture de maven sous Eclipse :



**ImageN°5** : maven

J’ai utilisé les dépendances de Maven afin de me faciliter les imports de certaines bibliothèques, comme j’ai expliqué dans la page précédente, je n’ai pas eu besoin d’installer **maven** car Eclipse Néon vient avec, tout ce que j’ai eu à faire c’était de définir le nom de la bibliothèque dont j’ai besoin de fichier pom.xml de **maven**, comme tel :

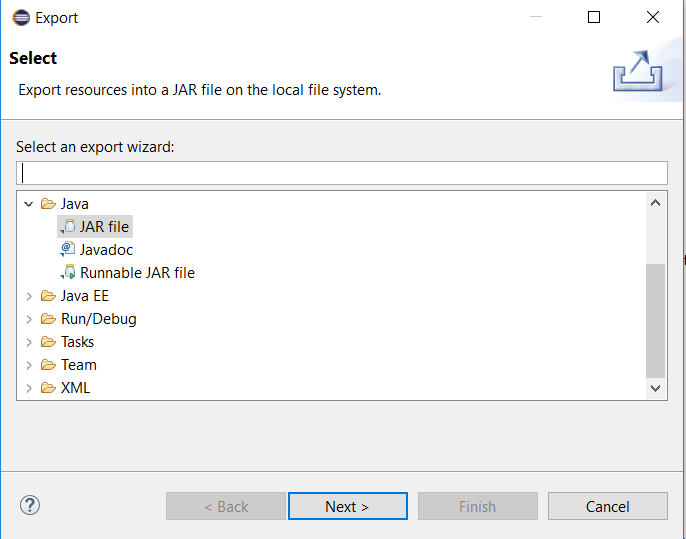


**Image N°6** : expleMaven.

Comme vous pouvez le remarquer, c’est dans la balise <dependency></dependency> qu’on définisse la bibliothèque dont on a besoin, le <groupId></groupId> prend le nom global de la bibliothèque soi en ligne, ou un de vos projet qui se trouve sur votre machine, et le <artifactId></ artifactId > prend le nom spécifique de la partie dont vous avez besoin dans la bibliothèque.

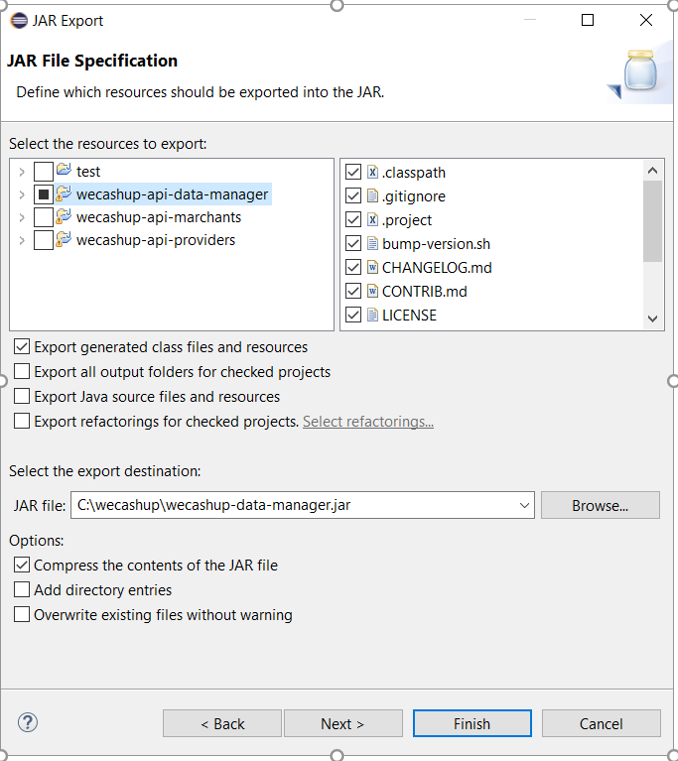
PS : Au niveau de l’import d’un de vos projets qui se trouve sur votre machine, vous aurez besoin de taper un code supplémentaire, avant cela, vous avez à exporter votre projet en fichier .jar, pour ce faire je procède comme suite :

Clic droit sur le projet à exporter, ensuit cliquer sur exporter, vous serez dirigé vers une boite de dialogue qui ressemble à ceci :



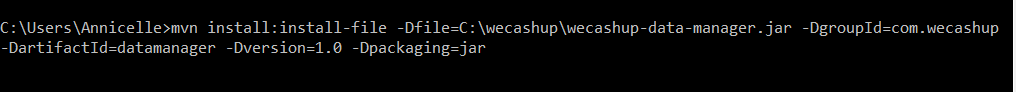
**Image N°7** : eclipseCapture.

Sélection JAR file et cliquez sur Next, une autre boite de dialogue s’ouvre :



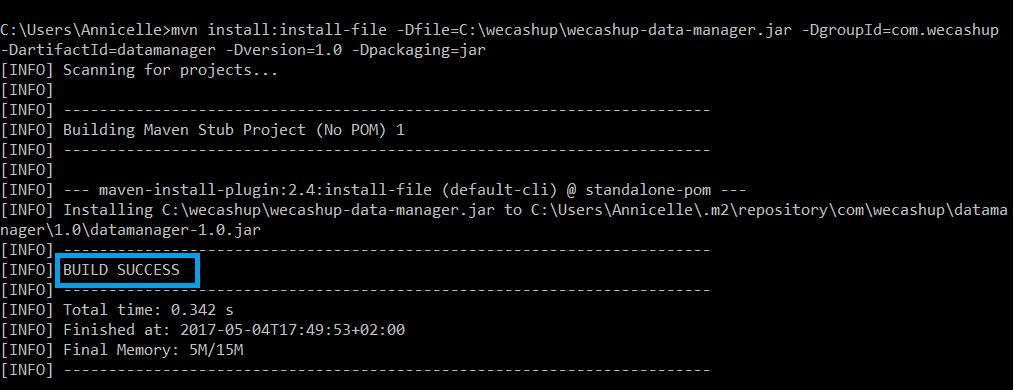
Cochez le nom de projet à exporter en haut à gauche et donnez le chemin où enregistrer, pour finir cliquez sur finish, ensuit ok jusqu’à la fin.

Une fois que le projet est exporté, afin de l’indiquer le chemin d’accès et le nom du fichier, taper cette ligne de commande dans la console :



* **Mvn install : install-file** : permet d’exécuter maven sur votre machine.
* **-Dfile=C\wecashup\wecashup-data-manager.jar :** est le chemin d’accès au fichier spécifique que vous venez d’exporté.
* **-DgroupId=……… -DartifactId =……. -Dversion=…. -Dpackaging=jar**: le nom du groupeId, artifacId et la version que vous avez défini le fichier pom.xml, laissez packaging comme tel, car il indique l’extension du fichier.

Rassurer vous que cette commande a bien été exécuté avec succès :



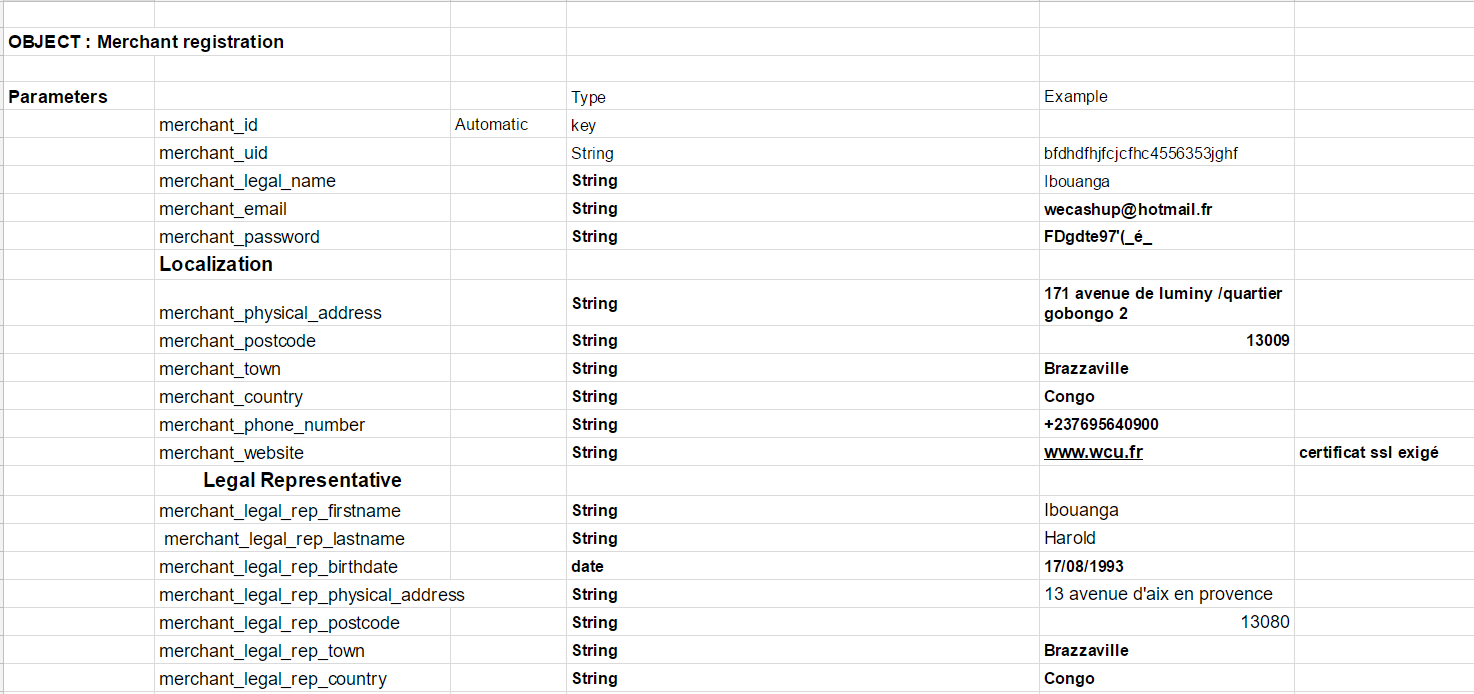
**Image N°8** : captureCmd.

1. Réalisation :

C’est WeCashUp dans son ensemble que je viens de vous présenter, passons maintenant à la réalisation. Dans un premier temps j’ai été amené à faire les inscriptions des Marchands, des clients et des providers, ensuite se rassurer de l’intégrité de données venant de chacune de ces entités, ensuite la rédaction de la documentation pour permettre au développeur des marchands d’intégrer plus facilement l’API WeCashUp, ensuite déploiement de la documentation de WeCashUp sur Google App engine, et la formation de quelques stagiaires sur la méthode de travail de l’INFINITY SPACE.

1. Inscription :

Avant de me lancer à créer des classes correspondantes à ces entités en JAVA, j’ai récolté toutes les informations que peut avoir ces entités dans un fichier xcl, afin de savoir quelle variable va contenir, quel type de donnée et quel est sa longueur ou sa capacité, qui ressemble à ceci :



**Image N° 9** : Merchant Object

Ces informations m’ont permis d’aller plus vite lors de la création des différentes classes, comme vous avez peut-être remarqué chaque paramètre à un nom, un type, et un exemple, qui compose bien évidement chaque classe, tout en respectant les règles de la modélisation et les relations entre ces classes (association, composition, agrégation …).

Intégrité de données : n’est autre que la cohérence des données lors de la transmission, de leur traitement, ou de la conservation, du côté utilisateur ou même venant du serveur de WeCashUp.

De manière générale, l'**intégrité des données** désigne l'état de **données** qui, lors de leur traitement, de leur conservation ou de leur transmission, ne subissent aucune altération ou destruction volontaire ou accidentelle, et conservent un format permettant leur utilisation (4).

Par exemple : un marchand menant ses activités au Mali, s’est inscrit avec la devise malienne, si par x raison un de ses client, envoie une transaction avec une autre devise, cette transaction ne doit pas passer, ou les données d’une carte Sim, dont cet opérateur télécom n’est pas opérationnel au Mali etc.

1. Rédaction de la documentation de l’API de WeCashUp :

**2.1. API**

**a) Qu'est-ce qu'une API ?**

Une API, interface de programmation applicative (Application Programming Interface) est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui spécifie comment des programmes consommateurs peuvent se servir des fonctionnalités du programme fournisseur. Cette description est la documentation de l’API.

**b) L’API WeCashUp :**

WeCashUp est une plateforme de paiement mobile dédiée principalement aux pays sous ou non bancarisés. En effet, il existe beaucoup de système de paiement mobile suivant différents opérateurs téléphoniques.

Souvent, dans un pays on peut avoir par exemple quatre opérateurs téléphoniques qui font du paiement mobile (exemple : Airtel Money, pour l’opérateur téléphonique Airtel / OrangeMoney pour l’opérateur téléphonique Orange / MobileMoney pour l’opérateur MTN / MoovMoney pour l’opérateur téléphonique Moov…) mais ils ne communiquent pas entre eux. Ainsi, une personne utilisant ce système doit posséder à elle seule les quatre cartes Sim de ces opérateurs. Ce qui n’est pas pratique d’autant plus que dans certains pays, on peut quelque fois atteindre douze opérateurs.

Ainsi, WeCashUp est la solution qui offre de l’interopérabilité à ces systèmes.

De ce fait, en détenant une seule carte Sim d’un opérateur quelconque, une personne peut transférer et recevoir de l’argent de tous les autres opérateurs et même faire des achats sur n’importe quel site d’un marchand qui a embarqué la solution WeCashUp.

* 1. **La documentation :**

1. **Pourquoi ?**

Une API est publique ; donc tout le monde doit pouvoir la comprendre. De ce fait, sa documentation doit être compréhensible et accessible à tout le monde via un navigateur web. Pour pouvoir l’implémenter un développeur doit connaître comment les entités sont décrites (leurs attributs) ; les fonctions et ce qu’elles renvoient…. Ce document sert de lien entre les concepteurs de l’API et les utilisateurs.

Ainsi d’une part, il s’agit d’un document qui en interne va permettre à toutes les personnes participant l’implémentation de l’API d’avoir les mêmes appellations de variables etc…, et d’autre part ce document servira de tutoriel à tout développeur voulant embarquer le système sur son site.

De ce fait, lorsque la documentation d’une API est bien faite (claire, concise et lisible), une entreprise peut se passer du support du produit qui générerait des frais supplémentaires à cette dernière.

1. **Que doit-elle contenir ?**

* **Les entités :**

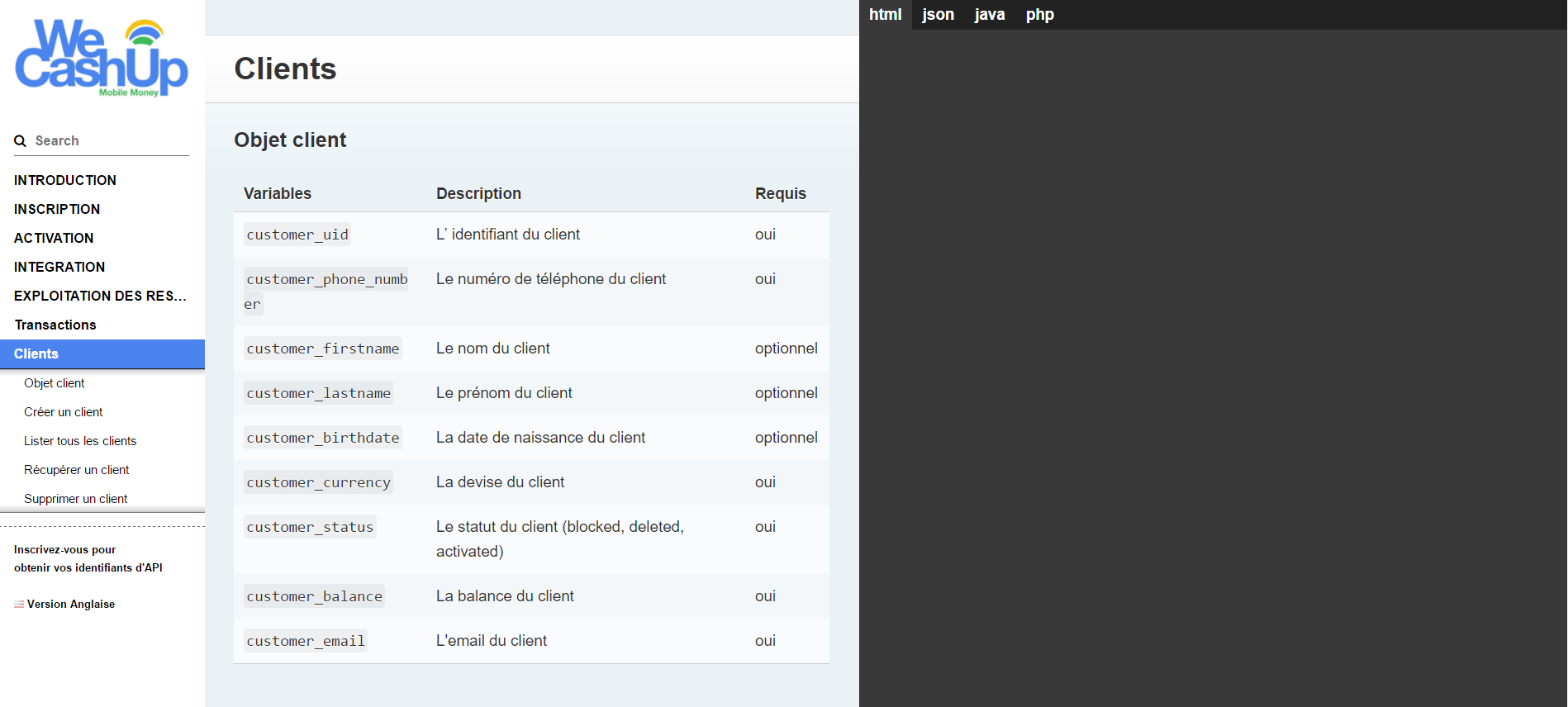
Dans le cadre d’une API web, une entité est un objet caractérisé par des attributs.

Formaliser toutes les entités dans la documentation d’une API permet à toute personne manipulant cette API de connaître tous les objets et les attributs sur lesquels l’on agit.

On peut manipuler les entités (sur le web) en utilisant des requêtes HTTP :

* + - POST à une URL donnée : pour créer une entité.
    - GET à une URL donnée : pour récupérer des entités.
    - PATCH à une URL donnée : pour mettre à jour des entités.
    - DELETE à une URL donnée : pour supprimer des entités existantes.

Exemple d’une entité client :



**Image N°10 :** Entité client.

**Source :** <https://www.wecashup.cloud/developers/docs/fr/?html#clients>

* **La description des fonctions :**

Pour pouvoir implémenter une API, un développeur doit non seulement connaître les entités et leurs attributs mais aussi toutes les fonctions (leur signature : ce qu’elles prennent en paramètre, ce qu’elles renvoient comme résultat…) auxquelles on fait appel pour pouvoir manipuler ces entités.

Notant que connaître les paramètres et les types de retour des fonctions est très important pour l’utilisateur de l’API. Ainsi, il pourra envoyer les bonnes requêtes avec les bons paramètres à la bonne adresse.

* **Les codes d’erreur :**

Les codes d’erreurs sont très importants lorsqu’on veut décrire une API. Ceux-ci permettent de communiquer dans un langage universel.

Ainsi, il est recommandé de les mettre dans sa documentation dans le but de permettre à l’utilisateur de pouvoir interpréter les erreurs afficher afin de localiser la source du problème. De cette manière, le problème peut être facilement identifié et résolu. Les codes d’erreurs peuvent être :

* + - 200 OK : la requête HTTP GET, PATCH ou DELETE a été traitée avec succès.
    - 201 Créé : la requête POST a été correctement traitée et a résulté en la création d'une nouvelle ressource.
    - 204 Pas de contenu : le serveur HTTP a correctement traité la requête mais il n'y a pas d'information à envoyer en retour.
    - 304 Non modifié : Document non modifié depuis la dernière requête.
    - 400 Mauvaise requête la requête a été mal formulée.
    - 401 Non autorisé la requête nécessite une identification de l'utilisateur.
    - 403 Interdit Le serveur HTTP a compris la requête, mais refuse de la traiter parce que l'auteur de la requête n'a pas assez de droits pour accéder à la ressource sollicitée.
    - 404 Non trouvé le serveur n'a rien trouvé qui corresponde à la ressource demandée.
    - 405 Méthode non autorisée la méthode utilisée par le client n'est pas supportée pour cette ressource.
    - 410 Parti la ressource n'est plus disponible et aucune adresse de redirection n’est connue.
    - 415 Format non supporté le serveur ne peut traiter la requête car son contenu est écrit dans un format non supporté.
    - 422 Entité incompréhensible l’entité fournie avec la requête est incompréhensible ou incomplète.
    - 429 Limite de nombre de requêtes atteinte le client a émis trop de requêtes dans un délai donné.
    - 500, 502, 503, 504 Erreurs serveur problème survenu sur le serveur de WeCashUp (cas exceptionnel).
* **Versioning :**

Le versioning est très important pour une API. Ainsi il est préférable de définir au préalable une politique de versioning qui permettra à l’utilisateur de garder un œil sur des potentielles modifications de l’API.

Une politique de versioning est par exemple la politique de dépréciation. Lorsqu’on veut modifier une méthode, au lieu de la supprimer du jour au lendemain, il faut la signaler d’abord comme « deprecated », proposer la nouvelle solution pour que les gens puissent avoir le temps de se familiariser avec et ensuite lors du changement, supprimer la méthode dépréciée de l’API.

Bien sûr tout ceci doit être prédéfini dans la documentation de l’API ; parce que si l’utilisateur de l’API ne sait pas comment tout cela fonctionne, cela risque d’être embêtant.

* 1. **Mise en place de la documentation de l’API de WeCashUp :**

1. **Ruby :**

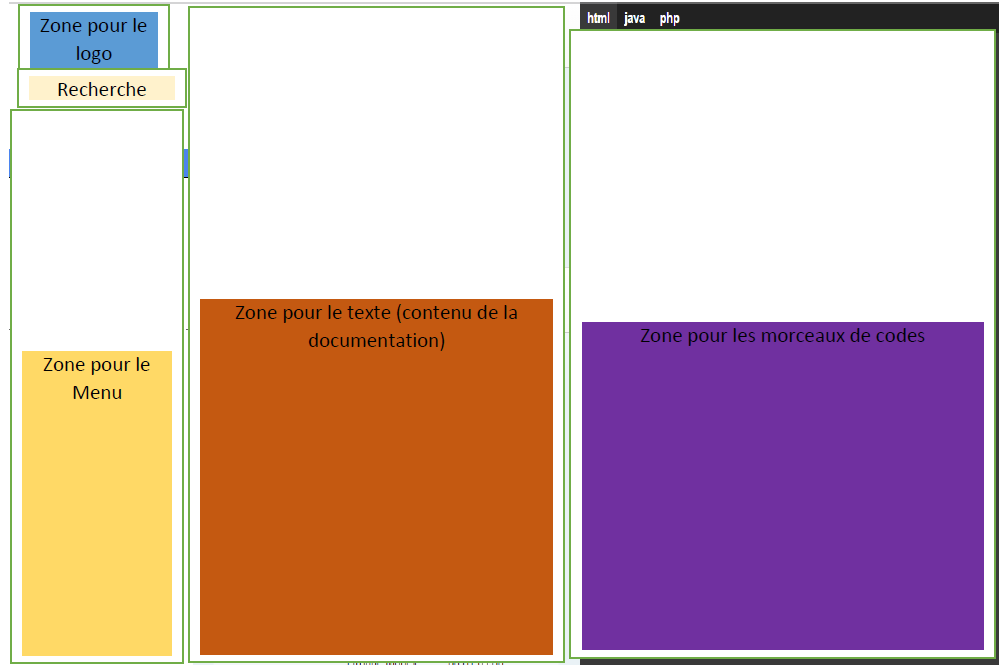
Ruby est un langage de programmation libre. Il est interprété, orienté objet et multiparadigme. Le langage a été standardisé au Japon en 2011 (JIS X 3017 :2011), et en 2012 par l'Organisation internationale de normalisation (ISO 30170 :2012).

1. **Slate :**

Slate est un outil de création de la documentation d’API, codé en langage Ruby. Il est complètement open-source.

Slate est un Template déjà prédéfinie. Son interface présente :

* + - Une zone pour un logo
    - Une zone de recherche
    - Une zone pour le sommaire (menu)
    - Une zone pour le texte
    - Une zone pour les exemples de codes dans différents langages
    - Une zone pour les recherches.



**Schéma N°** : Template Slate

Le document Slate contient un fichier .md où on peut rajouter notre documentation en fonction des différentes zones présentées ci-dessus.

Quelques notions élémentaires pour écrire dans le fichier .md :

* + - Un lien hypertexte : [le mot ou la phrase] (le lien)
    - Titre de niveau 1 (Grand titre) : # Mon grand titre
    - Titre de niveau 2 (titre moyen) : ## Mon moyen titre
    - Titre de niveau 3 (petit titre) : ###Mon petit titre (ce titre n’apparait pas dans le sommaire).

Notez que Les titres de niveau 1 et 2 vont s’ajouter automatiquement au Menu.

* + - Pour un paragraphe : Il suffit simplement d’écrire un texte (chaque paragraphe correspond à un retour à la ligne) Zone pour le logo Zone pour les morceaux de codes Zone pour le texte (contenu de la documentation) Zone pour le Menu Recherche
    - Pour rajouter du texte dans la zone de code il suffit de rajouter le symbole « > » au début de la phrase.
    - Pour rajouter du code dans la zone de code ; il suffit de rajouter les symboles suivants en fonction du langage en question :

```langage

Votre code ici

```

* + - Pour écrire en bas du Menu :

toc\_footers:

* + - Votre texte ici (par exemple : <a href='https://www.monsite.com/register'>Sign Up for a Developer Key</a>)
    - Pour rajouter les différents types de langages que vous allez utiliser dans la zone de code :

language\_tabs:

Html

Java

PHP

* + - Rajouter un tableau (c’est le code qui a généré l’entité client présenté ci-dessus) :

Attributs | Description|Requis

--------- | ------- | -------

<code>customer\_id </code>|L' identifiant du client| oui

<code>customer\_phone\_number </code>|Le numéro de téléphone du client| oui

<code>customer\_name</code>| Le nom du client| optionnel

<code>customer\_last\_name</code>| Le prénom du client| optionnel

<code>customer\_birthdate </code>| La date de naissance du client| optionnel

<code>customer\_currency</code>| La devise du client| oui

<code>customers\_status</code>| Le status du client (blocked, deleted, activated)| oui

1. **Installation de Ruby et exécution de Slate (via un invité de commande) :**

Avant tout, il faut vous assurer que Node js soit installé sur votre machine. Si non, il faudra l’installer.

En effet, Node js est une plateforme logicielle qui contient une bibliothèque de serveur HTTP ; ce qui permet de faire tourner un serveur web sans avoir à utiliser un logiciel externe comme Apache par exemple.

Ensuite :

* + - Télécharger le setup sur <http://railsinstaller.org/en>
    - Lancer le setup une fois le téléchargement terminé.
    - Lancer un invité de commande
    - Taper la commande **bundle** (Si la commande n’est pas reconnu, taper **gem install bundler**)
    - Ensuite retapez **bundle** pour être sûr que l’installation s’est bien passée
    - Puis déplacez-vous dans votre document slate avec la commande cd
    - Tapez **bundle install** pour installer toutes les dépendances
    - Puis exécutez **bundle exec middleman server**
    - Votre documentation est disponible à l’adresse suivante : <http://localhost:4567>
    - Si vous voulez l’exporter au format HTML, taper la commande **bundle exec middleman build –clean** : cette commande génèrera un document nommé « build » qui contient la doc au format HTML.