EPITECH INNOVATIVE PROJECTS

Cahier des charges

EIP E–Receipts Promotion 2017

BENZAHRA Marc BONIN Guillaume BOUTIN Paul CHANTOME Maxence

DROBIEUX Julien DURAND Benoît EVANNO Corentin MESSARA Redouane

Résumé

Le projet E-Receipt consiste à remplacer les tickets de caisse papier par une version numérique. Pour cela nous mettons en place une solution software et hardware. Le marché en France est très important car aucune solution alternative ne se dégage pour le moment.

Notre projet est soumit à des contraintes fonctionnelles liées à notre gestion de la technologie NFC et à l'anonymisation des données de nos utilisateurs. En effet, tous les smartphones ne permettent pas l'utilisation du NFC, pour combler cette problématique nous allons utiliser des QRCode. Cependant le système de QRCode a également des contraintes, comme la taille des données qu'il peut contenir.

Nous nous sommes fixés deux grandes exigences non fonctionnelles :

- Les différents temps de réponse qui doivent être les plus courts possibles pour le confort de nos utilisateurs.
- Nous devrons nous assurer que parmi tous les volumes de données échangées au même instant aucune ne se perde.

Les fonctionnalités seront réparties comme tel:

- La borne en magasin (Rasberry Pi) affiche le QRCode ou envoie les données via puce RFC à l'application mobile.
- L'application mobile permet de scanner le QRCode générer par la borne, de visualiser les tickets de caisse et de pouvoir les imprimer.
- Le serveur stock les tickets de caisse et génère des statistiques
- Le site web permet de visualiser les tickets de caisse et différentes statistiques et d'imprimer les tickets.

Nous mettons en place notre environnement de réalisation avec un environnement de production sur lequel nous allons pouvoir effectuer nos développements. Accompagné d'outils et de process pour nous conforter dans nos développements, enfin, nous aurons un environnement de déploiement pour tester et mettre nos solutions en ligne.

Nous allons utiliser un ensemble de tests pour nos fonctionnalités, des tests unitaires, de non régression, de configuration, de performance, de charge, de monté en chargé, de stress.

Pour nous organiser durant ces deux années nous avons opté pour la mise en place de la méthode Scrum. Cette méthode nous convient particulièrement puisqu'elle est divisée en sprint, avec un suivi régulier.

Information sur le document

Titre	[2017][ERECEIPTS][CDC3]
Date	27/06/2015
Auteur	EVANNO Corentin
Responsable	BOUTIN Paul
E-mail	Paul.boutin@epitech.eu
Sujet	Cahier des charges 3
Version du modèle	2.0

Tableau des révisions

Date de modification	Auteur	Parties concernées	Commentaire
27/06/2015	EVANNO Corentin	Toutes	Création du document
27/06/2015	EVANNO Corentin, BOUTIN Paul, DURAND Benoît, CHANTÔME Maxence, MESSARA Redouane, DROBIEUX Julien, BONIN Guillaume, BENZAHRA Marc	Toutes	Modifications de toutes les parties du document
28/06/2015	BENZAHRA Marc	Résumé	Modification du résumé

Table des matières

I. Rappel de l'EIP

a. Objectifs de l'EIP et Epitech

L'Epitech Innovative Project est l'ultime projet de notre scolarité à EPITECH.
Il nous permet d'appliquer tout ce que nous avons appris pendant ces cinq années.

Il est le plus important des projets d'EPITECH car il nous met en situation réelle de gestion de projet.

C'est l'étape qui nous fait passer du statut d'étudiant au statut de professionnel.
L'EIP se déroule sur trois ans, du choix du sujet à sa réalisation complète.

b. Principes de base du système futur

Le projet consiste à remplacer les tickets de caisse papiers par une version numérique. Pour cela, nous avons imaginé un système de tickets de caisse en ligne, à chaque passage en caisse l'utilisateur pourra demander au commerçant s'il est équipé du dispositif qui permet de transmettre le ticket via notre service. Une borne permettra alors à l'utilisateur de récupérer son ticket via son téléphone. Pour cela l'utilisateur devra scanner un QR code généré après le paiement et le ticket numérique sera directement mis en ligne sur son compte E-Receipts.

Grâce à l'application ou au site internet, l'utilisateur pourra retrouver sur le site web ou l'application dédiée tous ses tickets de caisses avec des fonctionnalités telles que le tri par domaine (nourriture, équipement, etc...). Une vision d'ensemble permettra de voir en un simple coup d'œil comment sont réparties les dépenses, des rappels pour les garanties pourront être configurés (exemple : fin de garantie pour lave-vaisselle 2 ans après son achat). Il sera également possible d'imprimer le ticket de caisse ou d'accéder à différents détails tels que la date d'achat, le lieu... .

En France, le domaine du ticket de caisse reste inchangé depuis plusieurs années. Les grandes enseignes les utilisent pour diffuser de la pub, mais aucune innovation n'a été faite dans ce domaine.

La plupart des tickets de caisse sont jetés directement après l'achat, oubliés dans une poche, perdus ou brûlés. C'est suite à ce constat que nous proposons une solution écologique et pratique en numérisant les tickets de caisse.

II. Contraintes

a. Contraintes fonctionnelles

Notre projet va poser plusieurs contraintes fonctionnelles listées ci-dessous :

- Contraintes liées au NFC

Actuellement, certains smartphones comme l'iPhone ne permettent pas d'utiliser le NFC pour des applications tierces. En effet, pour promouvoir notre projet au plus grand nombre d'utilisateurs, nous allons devoir utiliser un système de QRCode en plus du NFC. Ainsi, nous allons développer un module QRCode sur chaque application mobile et sur la borne pour l'affichage et la génération des QRCode.

- Contraintes liées au QRCode

L'utilisation d'un QRCode implique plusieurs contraintes, comme la présence d'un écran sur notre borne. De plus, les QRCodes possèdent des limitations de données. Pour contourner cette problématique, nous pouvons par exemple, faire scanner un ensemble de QRCode pour un seul ticket de caisse. Nous réfléchissons sérieusement à cette contrainte pour que la solution soit la moins contraignante pour les commerçants et les utilisateurs.

- Contraintes liées aux données privées des utilisateurs

Nous devons garantir l'anonymat de nos utilisateurs, ils ne seront pas identifiables sur les données que nous fournirons à nos clients. Un aspect sécurité s'impose sur nos serveurs et sur nos bases de données pour assurer la pérennité de celles-ci. Nous devrons donc utiliser des technologies de sécurité pour protéger nos données.

b. Exigences non fonctionnelles

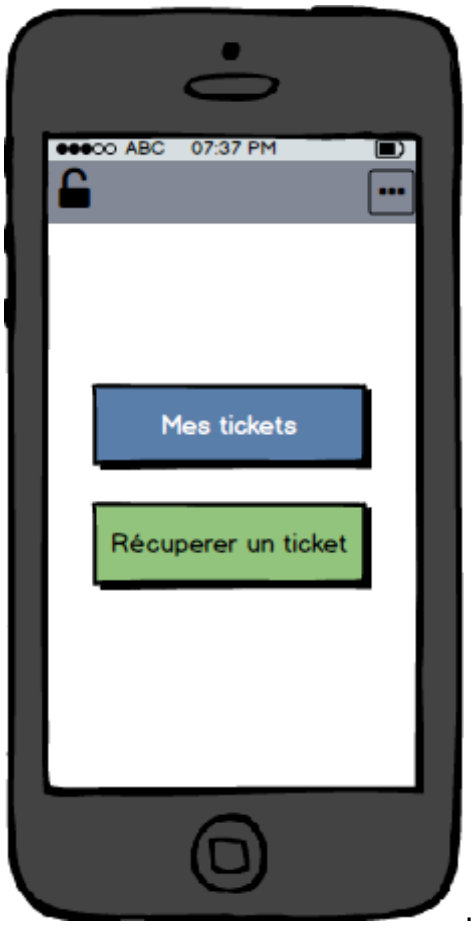
- Exigences liées aux temps de réponses

Le premier des temps de réponses dont nous devons nous préoccuper est celui de la transmission du ticket de caisse jusqu'au téléphone de l'utilisateur. En effet, si l'utilisateur doit attendre plus de cinq secondes à la caisse avant que son ticket soit transmis, notre projet n'a aucun intérêt pour lui, car il lui fera perdre du temps. Nous devrons donc minimiser le temps de réponse entre la caisse et la borne puis entre la borne RFID / QR Code et le téléphone pour que celui-ci soit inférieur à cinq secondes.
L'autre temps de réponse qui est important est celui du serveur. Notre projet doit pouvoir gérer en temps réel le téléversement des tickets de caisse sur le serveur pour que le client puisse consulter ces derniers justes après ses achats. Il faudra donc faire en sorte que le serveur et ses clients communiquent en temps réel, avec une base de données permettant une extraction des données en moins d'un centième de seconde par ticket.

- Exigences liées aux quantités de donnée échangées

A certains moments il pourrait y avoir plusieurs milliers de tickets de caisse générés et téléversés au même moment.
Nous devrons nous assurer que parmi tous les volumes de données échangées au même instant aucune ne se perde. Encore une fois, cela passe par une bonne conception du serveur et de la base de données.

III. Description des différentes parties du programme à réaliser

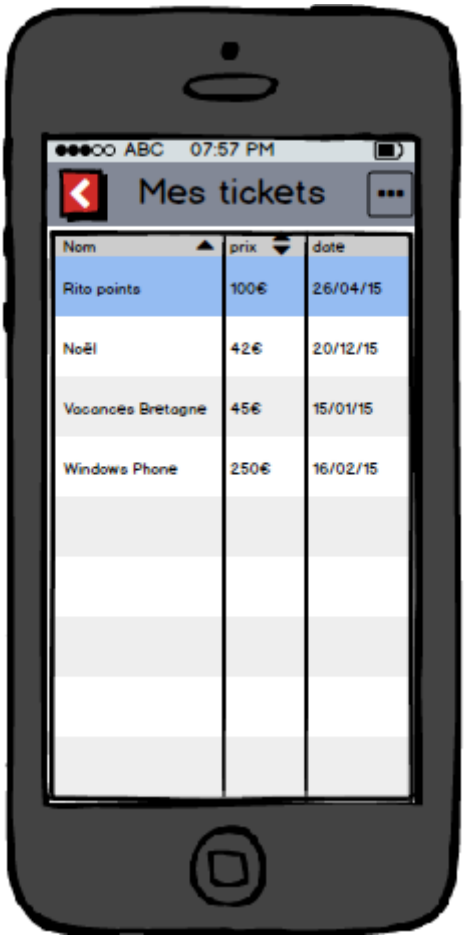


1- Interface pour récupérer les tickets d'E-Receipts

Au passage en caisse, le client aura la possibilité d'obtenir son ticket de caisse via E-Receipts, dans ce cas de figure, la caisse enregistreuse enverra les données du ticket de caisse au module E-Receipts. L'utilisateur démarrera alors l'application sur son smartphone et choisira l'option : "récupérer mon ticket de caisse", l'application lui demandera alors d'approcher son smartphone sur le module E-Receipts en caisse pour utiliser la technologie NFC ou alors de scanner le QRCode s'affichant sur cette dernière.

Une fois le ticket de caisse reçu sur le smartphone via une des deux solutions (QRCode ou NFC), le ticket de caisse est automatiquement sauvegardé sur le téléphone et lorsque que le téléphone de l'utilisateur sera connecté à internet, le ticket de caisse sera synchronisé en ligne sur son compte.

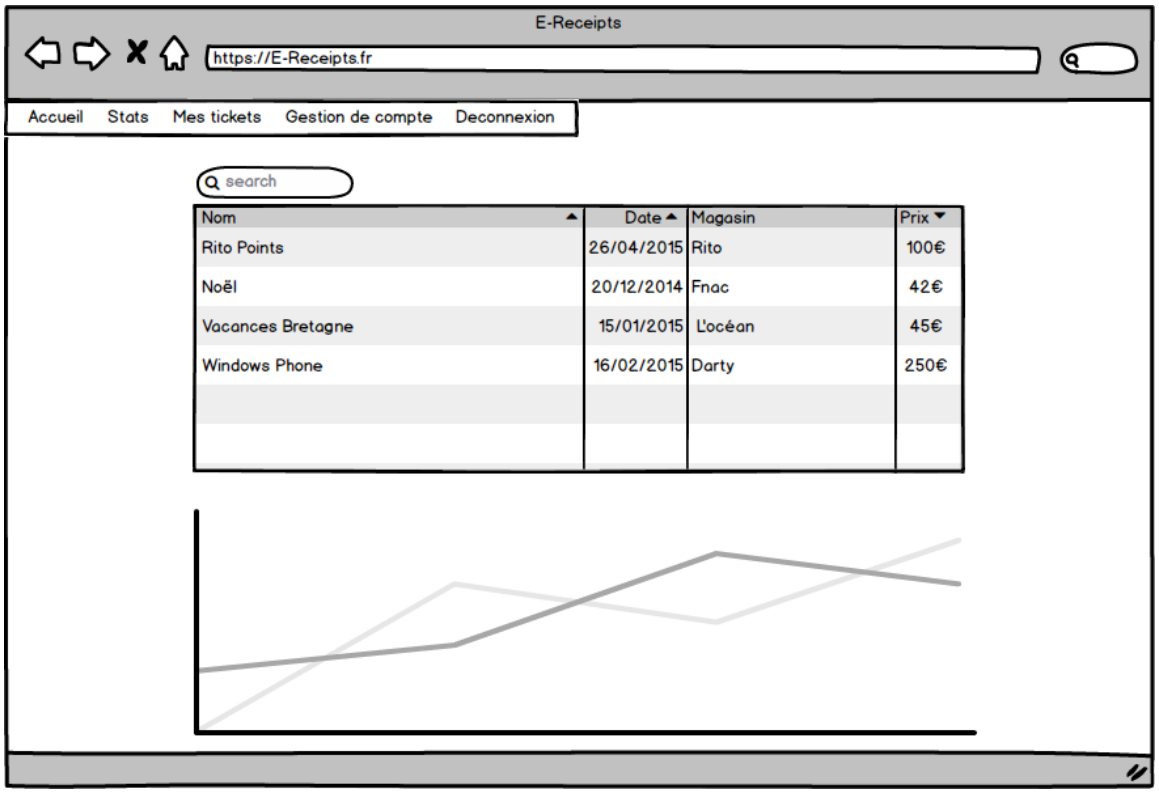
2- Interface pour scanner les QRCode



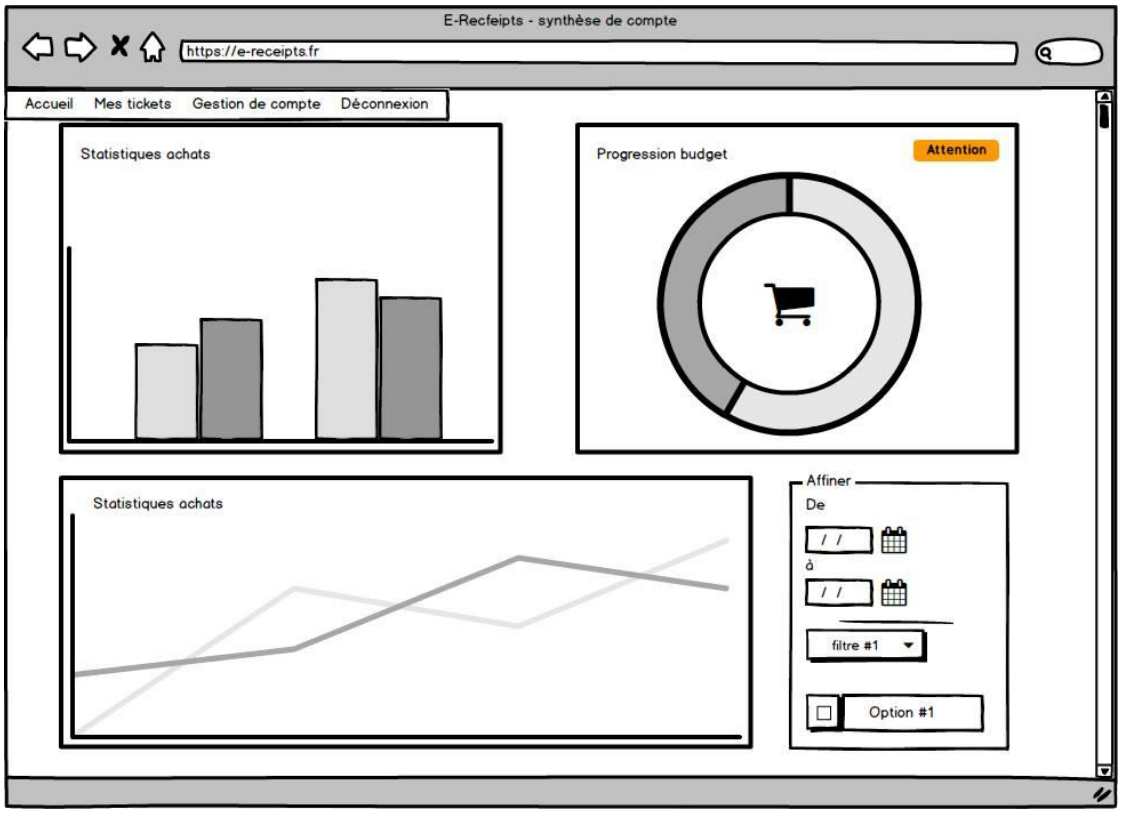
3- Interface pour visualiser ses tickets E-Receipts

Ultérieurement, l'utilisateur pourra démarrer l'application ou se connecter sur le site E-Receipts via ses identifiants et visualiser ses tickets de caisse via l'option appropriée. Il verra ainsi une liste de tickets accompagnés du prix et de la date d'achat. Une simple pression sur le ticket voulu affichera ensuite le ticket de caisse, il sera possible depuis la vue du ticket de caisse de l'envoyer par mail, de le télécharger en PDF ou en csv (format Excel), de l'imprimer ou même de le supprimer.
De plus, l'utilisateur aura accès à différentes statistiques. Il pourra obtenir un historique de ses dépenses mensuelles, quotidiennes et/ou journalières. Il sera en mesure de connaître la somme de ses dépenses dans un magasin précis ou celles effectuées lors d'une période qu'il a lui-même définie. Des listes de course en fonction des produits achetés de façon répétitive pourront être déterminées et le client connaîtra les évolutions des prix d'un produit. Dans l'idée, le tout se voudra assez flexible, c'est-à-dire que l'utilisateur pourra lui-même choisir quelle statistique il souhaite afficher, la période sur laquelle prendre les valeurs, il aura aussi accès à un

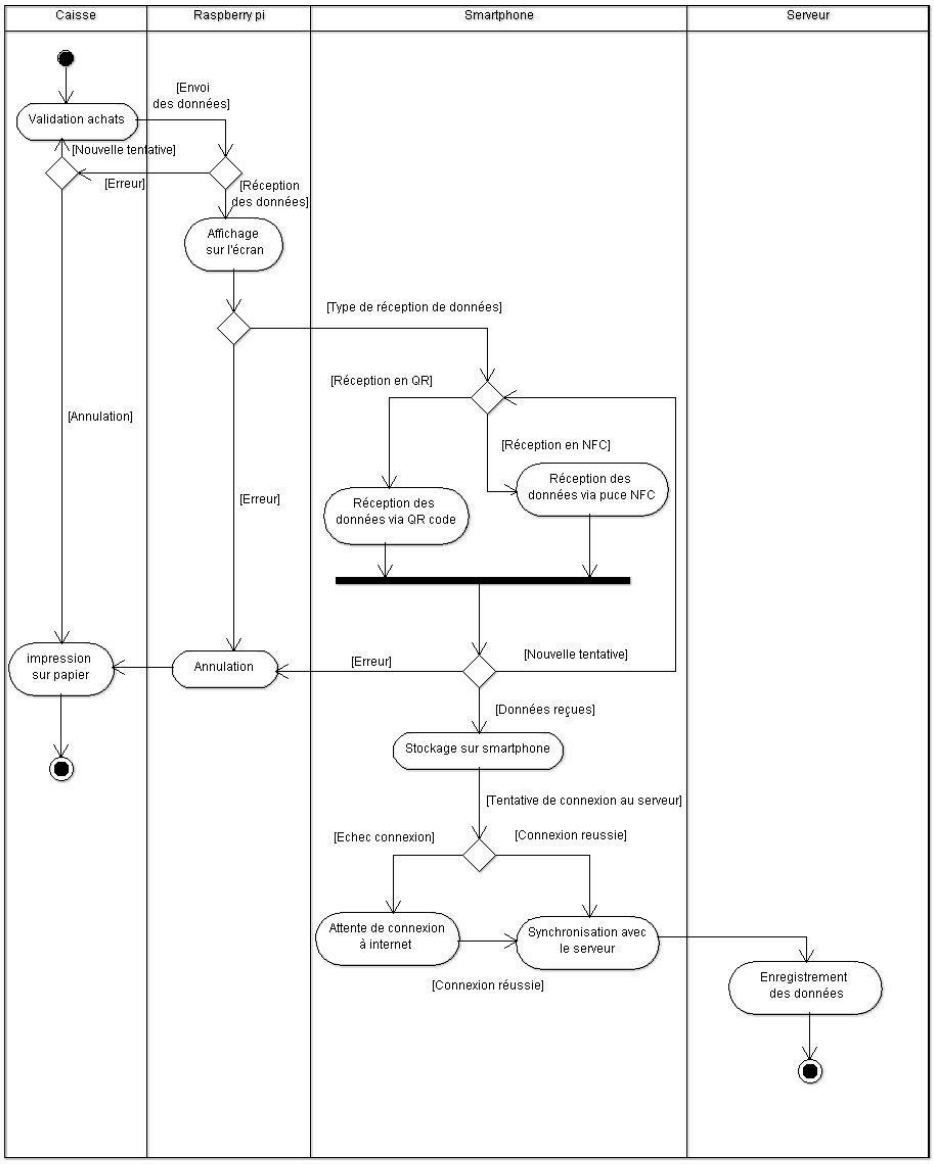
comparatif, et le tout sera affiché sous forme de dashboard et pourra être exporté dans différents formats.



4- Page web de la liste des tickets E-Receipts



5- Page web des statistiques d'E-Receipts



6- Diagramme d'interaction entre les composants

IV. Présentation de l'environnement de réalisation

a. Environnement de réalisation

Nous allons mettre en place notre environnement de travail en le découpant par organe comme suit :

- Environnement de production

Nous disposerons de serveurs pour tester nos développements et simuler des mises en production. Il nous faudra aussi des logiciels de caisse pour comprendre leurs fonctionnements afin de mettre en évidence l'ensemble des interactions qui pourraient être possibles entre une caisse et notre borne. Nous devrons disposer d'une ou plusieurs raspberry pi pour développer et tester les fonctionnalités de la borne. De smartphones et d'ordinateurs pour les développements et les tests.

- Environnement de développement

Pour développer sur les différentes plateformes et dans les différents langages nous utiliserons des environnements de développement (Android Studio, Visual Studio, XCode) ainsi que des éditeurs de texte (Sublime text, Emacs). Pour la base de donnée nous allons partir sur du NoSQL avec MongoDB. Afin d'uniformiser notre code et de faciliter la programmation nous allons suivre une norme interne pour crée une nomenclature à appliquer pour notre code base. Les applications Android et Windows seront développées sur Windows tandis que l'application IOS sera développée sur Mac OS X.

- Environnement de déploiement

Nous déploierons nos applications mobiles sur tous les stores (App store, Google play, Windows store). Nous aurons à respecter les exigences de ces stores comme des chartes de qualité de code. Pour l'hébergement de nos serveurs nous utiliserons probablement OVH avec une offre qui permet de réaliser des sauvegardes de donnée et une assurance de restauration des données (disques dur RAID). Nous utiliserons tous les utilitaires nécessaires au déploiement de notre code source et de nos outils coté serveur (Apache, Tomcat, Node.js, ...).

- Outils

Pour faciliter l'usage des environnements décrits ci-dessus, nous allons utiliser des outils pour organiser et optimiser notre workflow. Git pour le contrôle de version de notre code source, de notre documentation développeurs. Nous allons déployer un wiki pour avoir une documentation plus visuelle et agréable (Confluence ou Redmine) ainsi qu'un bug tracker pour ouvrir, assigné, résoudre des bugs ou des tickets pour de nouvelles fonctionnalités (Trello ou JIRA et Bugzilla). Enfin, des outils plus techniques pour nous conforter dans nos développements comme un gestionnaire, exécuteur de tests (Jenkins) et des scripts de logging sur nos serveurs pour le débogage et la traçabilité des performances (Pylot, Apache JMeter).

b. Composants existants

Notre dispositif nécessitera plusieurs éléments composants existants, tout d'abord du hardware :

- un Raspberry Pi 2 B
- Un écran LCD pour afficher le QRcode. Il sera relié au Raspberry Pi.
- Un dispositif NFC pour Raspberry pi
- Un ordinateur équipé d'un logiciel de caisse pour simuler des achats

Nous estimons la durée de vie de ces appareils d'au moins 5 ans en fonctionnement constant.
Néanmoins, en cas de panne les modules défectueux peuvent être changés individuellement et facilement.

Puis, nous utiliserons des bibliothèques de développement ainsi que des frameworks :

- AngularJS est un framework javascript pour le développement web et qui sera utilisé lors du développement de notre site. De plus, il respecte le modèle MVC.
- Symfony2 est un framework PHP pour le développement web. Il nous sera utile car ses modules permettent de déployer facilement de très nombreuses fonctionnalités. De plus, il offre une architecture MVC et est sécurisé.
- Twitter Bootstrap est un framework HTML/CSS qui nous permettra facilement d'avoir un site web responsive aussi bien sur desktop que sur mobile. Grâce a ses plugins nous allons ainsi pouvoir le lier à AngularJS.
- Twig qui est nativement utilisé avec Symfony2 est un template PHP qui nous permettra de facilement intégrer des données back-end dans notre front-end pour respecter au mieux le modèle MVC.
- google-gson est une bibliothèque qui sera utilisée pour le développement Android afin de manipuler le JSON.
- Volley est une bibliothèque qui permet de faciliter la gestion du réseau sur Android.
- Libnfc est une bibliothèque qui permet de manipuler le NFC, nous l'utiliserons pour la manipulation de la puce RFID de notre Raspberry Pi. La gestion du NFC sur android étant native, nous utiliserons NDEF sur Windows Phone.
- ZBAR est une bibliothèque qui permet de manipuler les QRCode et qui sera utilisé sur Raspberry PI, Android, Windows Phone et IOS.
- Json.NET est une bibliothèque qui nous permettra de manipuler JSON sur Windows Phone.
- JSONKit est une bibliothèque qui nous permettra de manipuler JSON sur IOS.

Chacune des bibliothèques sera encapsulée dans une API afin de faciliter notre développement.

c. Gestion de la sécurité

Concernant la sécurité, afin d'éviter qu'un ticket soit dupliqué sur le compte d'un individu malveillant, chaque ticket sera créé avec un identifiant unique (généré grâce à une librairie prévue à cet effet). Si notre serveur se rend compte que l'identifiant n'est pas lié avec le compte utilisateur, il s'occupera de le supprimer des données de l'application et du compte.

L'anonymisation des données des utilisateurs sera garantie par la mise en place d'une base de donnée dédiée aux données sensibles (cela concerne toutes les données relatives à l'utilisateur donc : nom, prénom, date de naissance, ...). Cette base de données contiendra des données chiffrées. Enfin, nous implémenterons ce que MongoDB nous offre en termes de protection et de sécurisation des requêtes.

Pour le site web, l'utilisation de Symfony va nous permettre d'avoir une partie back-end sécurisé. L'avantage de Symfony est que la sécurité est très poussée mais que nous pouvons la contrôler facilement avec une grande granularité.
Enfin pour les applications mobiles, le risque le plus grand est qu'une personne malveillante accède aux données importantes situées dans le cache ou dans un dossier non protégé. Nous allons donc encrypter nos données sensible et les stocker dans un dossier lui-même encrypté (la méthode de protection dépendra de l'OS du mobile).

d. Points sensibles

Nous risquons d'être confrontés à différents types de problèmes, aussi bien humain que matériel:

Problème de compatibilité: Il y a un risque de non compatibilité avec certaines caisses enregistreuses. Cependant, ces caisses sont souvent d'anciens modèles et ne sont quasiment plus utilisées.

Les caisses "Ipad": Une nouvelle génération de caisses enregistreuses sur Ipad et qui ne dispose pas de boîtier est en train de faire son apparition. Nous devrons donc, plus tard, créer un modèle d'application qui puisse communiquer avec.

Problème de motivation: C'est un risque à prendre en compte, c'est pourquoi nous avons décidé de créer entre nous de petits groupes qui auront chacun besoin du travail des autres pour avancer et ainsi, même en cas de démotivation d'un groupe, celui-ci se sentira coupable de ralentir les autres et essaiera de faire de son mieux.

Perte ou vol du matériel: Il est possible que nous perdions du matériel comme un Raspberry Pi ce qui provoquerait des retards sur le projet.

Perte d'un membre: Dans ce cas-ci, nous le remplacerons ou nous continuerons avec un membre de moins quitte à travailler plus.

Tension interne: Des tensions pourraient émerger dans le groupe et mettre en péril notre projet, dans ce cas, nous essayerons de rester professionnels pour que les tensions n'impactent pas le projet.

Retard dans le projet: En cas de retard dans le projet, nous devrons nous poser la question du pourquoi et nous le rattraperons en appliquant les solutions adaptées à la situation.

V. Description des tests

Nous mettrons en place plusieurs types de tests:

- Test unitaire :

Ces tests seront utilisés afin de tester des fonctionnalités précises du programme afin de vérifier que celles-ci aient bien le comportement attendu. Ils vont intervenir dès que nous aurons terminé d'en développer une. En effet, ils nous permettent de tester son bon fonctionnement, indépendamment de l'avancée des autres fonctionnalités. En cas d'échec au niveau du test, il faudra debugger la partie de code pour analyser d'où vient le problème et ensuite relancer les tests unitaires.

- Test de non-régression

Ces tests nous permettrons de vérifier que des modifications n'ont pas altéré le fonctionnement de l'application. Ce type de test interviendra chaque fois qu'une modification sera faite sur une partie d'un programme qui a déjà subi des tests unitaires. Ce test est nécessaire pour ne pas risquer d'altérer une fonctionnalité qui marchait avant une modification. En cas d'échec, il faudra relancer des tests unitaires sur la partie de code concernée pour trouver d'où vient le problème.

- Test de configuration

Ces tests devront être effectués pour s'assurer que nos applications mobiles fonctionnent sous les différents OS mobile mais aussi sous leurs différentes versions. Ils seront aussi utilisés pour s'assurer que le site web fonctionne de la même façon sous tous les navigateurs et sous leurs différentes versions. Ces tests interviendront à chaque fois qu'une grosse fonctionnalité du programme sera rajoutée.

- Test de performance

Ces test nous serviront à analyser les performances sur les fonctionnalités qui pourraient prendre le plus de temps à l'exécution comme les statistiques ou encore les transferts de données. Ces tests nous servirons à minimiser les temps d'exécution pour que ceux-ci soient les plus faibles possible. Nous effectuerons ces tests lorsque de grosses fonctionnalités nécessitant ce test seront achevées.

- Test de monté en charge

Durant ce test nous allons simuler un nombre d'utilisateurs sans cesse croissant. Ce test nous permettra de déterminer la charge maximale que peut supporter le serveur, nous pourrons ainsi être face à ce cas de figure et ajuster notre développement en fonction des résultats de ce test. Ce test sera effectué lorsque le serveur et les fonctionnalités principales auront été implémentés.

- Test de charge

Ce test consiste à simuler un nombre d'utilisateurs définis, afin de tester le programme avec une charge attendue d'utilisateur. Ce test nous sera utile pour déterminer si nous devons allouer plus de ressource au niveau du serveur, et mettra en évidence les points critiques de notre architecture réseau. Nous effectuerons ce test lorsque le serveur et les fonctionnalités de base auront été implémentés.

- Test de stress

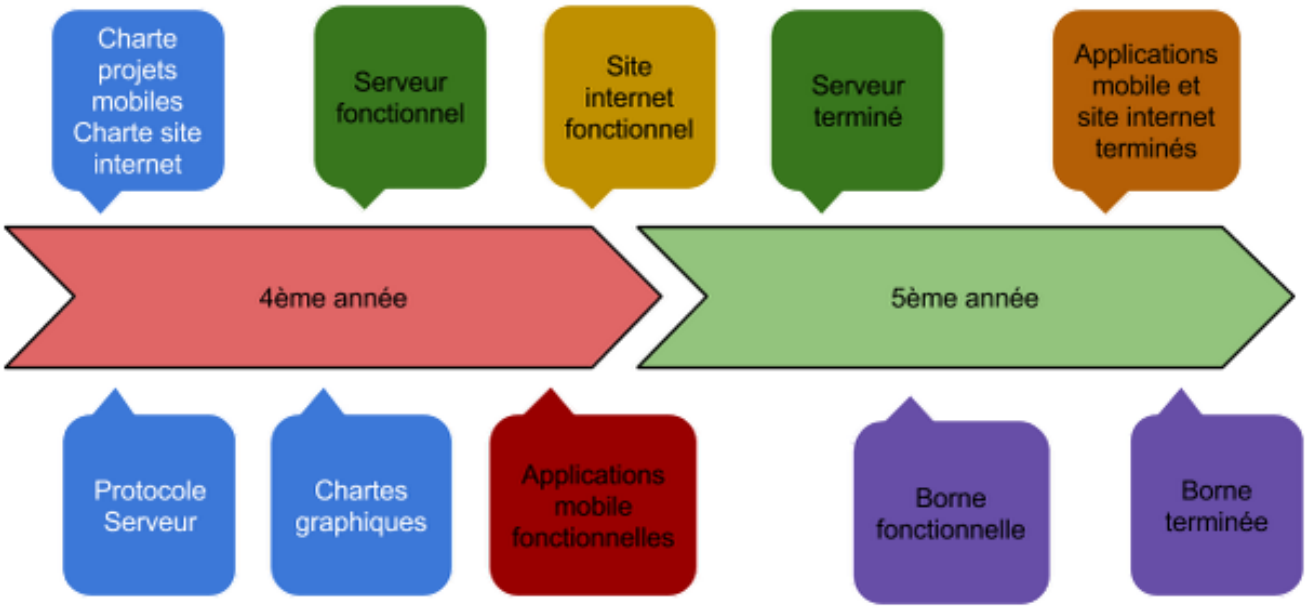
Ce test consiste à simuler une charge maximale d'utilisateur pendant plusieurs heures, le but de ce test et de voir comment notre programme va réagir à une charge qui est à la limite de ses capacités. Nous utiliserons les données collectées durant ce test pour optimiser certaines parties du code ou pour revoir notre architecture réseaux. Ce test sera effectué une fois que le serveur ainsi que toutes les fonctionnalités du programme auront été implémentées.

Les tests unitaires, de non régression et de configuration seront effectués à la main. Les tests de performance utiliseront Pylot. Enfin, les tests de charges, de montées en charge et de stress utiliseront Apache JMeter.
Les différents bugs rencontrés seront remontés et gérés sur Bugzilla.

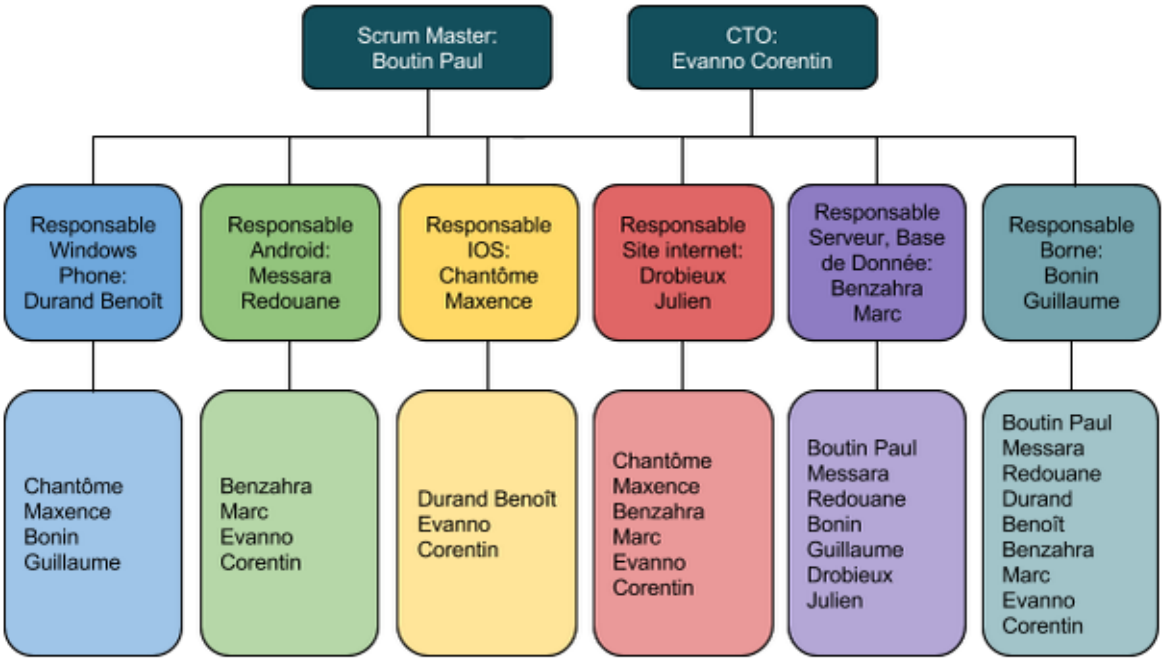
VI. Organisation du projet

Concernant la méthodologie d'organisation du projet, nous avons choisi d'utiliser la méthode Scrum.
Cette méthode nous convient particulièrement puisqu'elle est divisée en sprint, avec un suivi régulier. Cette méthodologie commencera par la construction du product backlog. Le product backlog consiste à établir la liste des fonctionnalités de notre produit. Ces fonctionnalités seront évaluées par toute l'équipe, et un certain nombre de point sera assigné à chaque fonctionnalité selon sa valeur dans le projet. Il est important de noter que ce backlog sera amené à évoluer, à être modifié, complété lors du développement et avec l'apparition de nouvelles idées ou problématiques.
Une fois le product backlog terminé, nous avons décidé de faire des sprints de deux semaines. Un sprint est une durée limitée dans le temps qui commencera par une planification des fonctionnalités à développer, et qui se finira par une démonstration de ce qui a été développé et par une rétrospective du même sprint. Ainsi, cela nous permettra de pouvoir améliorer nos méthodes de travail, de repérer les éléments positifs et les points à améliorer, et d'avoir une réelle vision sur ce qu'il reste à faire et ce qui a été fait. De plus, il sera simple de faire des tests unitaires et donc de repérer facilement les bugs ou autres problèmes inhérents au développement d'un programme.
Nous avons décidé de ponctuer ces sprints de courtes réunions tous les deux jours, qui nous permettront de voir ou chacun se situe et de noter les difficultés qui apparaissent pour proposer des solutions.
Il est à noter que chaque réunion sera dirigée par notre scrum master, Paul Boutin. Le scrum master attribuera les points des tâches qui ont été faites et validées lors des tests, et il devra suivre les équipes et être au courant des obstacles rencontrés lors du développement.
Les points nous permettront de connaître notre vélocité, ce qui nous permettra d'avoir une vision globale et chiffrée sur l'avancée du projet, et ainsi savoir si notre fréquence de sprint est adaptée ou non.

- Vue planning



- Organigramme



VII. Annexes

Lien :

- Android Studio : https://developer.android.com/sdk/index.html
- Visual Studio : https://www.visualstudio.com/
- XCode : https://developer.apple.com/xcode/
- Sublime text : http://www.sublimetext.com/
- Emacs : https://www.gnu.org/software/emacs/
- MongoDB : https://www.mongodb.org/
- Apache : http://www.apache.org/
- Apache TomCat : http://tomcat.apache.org/
- Node.js : https://nodejs.org/
- Git : https://git-scm.com/
- Jenkins : https://jenkins-ci.org/
- Bugzilla : https://www.bugzilla.org/
- Pylot : http://www.pylot.org/
- Apache JMeter: http://jmeter.apache.org/
- AngularJS : https://angularjs.org/
- Symfony2 : https://symfony.com/
- Twitter Bootstrap : http://getbootstrap.com/2.3.2/
- Twig : http://twig.sensiolabs.org/
- Google-gson : https://code.google.com/p/google-gson/
- Volley : https://developer.android.com/training/volley/index.html
- Libnfc : http://nfc-tools.org/index.php?title=Libnfc
- Zbar : http://zbar.sourceforge.net/
- Json.Net : http://www.newtonsoft.com/json
- JsonKit : https://github.com/johnezang/JSONKit
- NDEF : https://github.com/andijakl/ndef-nfc

Conclusion

Au-delà des contraintes techniques et technologiques, un projet de si longue haleine repose avant tout sur des problématiques humaines.

Nos plus gros challenges seront l'organisation et l'entente à longue distance sur une longue période.

Il ne faudra pas hésiter à prendre du recul et à s'écarter de ce cahier des charges si le besoin s'en fait ressentir,

Ce cahier des charges restera le fil rouge de notre projet mais nous voulons garder le maximum de flexibilité pour nous permettre de nous adapter à des conditions humaines qui seront en constante évolution.