Université de Sherbrooke
Faculté de génie
Département de génie électrique et de génie informatique

Rapport de projet
Rapport final

Conception d'un système embarqué et réseauté
GIF500

Présenté à
M. Domingo Palao Muñoz

Présenté par l'équipe 2 :

Mathieu Dostie	DOSM2902
Émile Fugulin	FUGE2701
Philippe Girard	GIRP2705
Damien Hulmann	HULD1501
Julien Larochelle	LARJ2526
Samir Lechekhab	LECS2813
Donavan Martin	MARD1206

Sherbrooke – 4 décembre 2017

Table des matières

1 Présentation du prototype **1**

2 Cahier des charges **2**

3 Branchement des différents éléments **6**
3.1 Général 6
3.2 Pont 6
3.3 Terminal de paiement 6
3.4 Terminal de recharge 6

4 Procédure de test du système du prototype **8**
4.1 Code C 8
4.2 Code Javascript 8
4.3 Assurance qualité 8

5 Explications techniques des fonctions et composants **11**
5.1 Pont 11
5.2 Terminal de recharge 11
5.3 Terminal de paiement 11
5.4 Serveur et base de données 11
5.4.1 API 11
5.4.2 Site Web 11
5.4.3 Postgres avec tables 14

6 Diagrammes UML **15**

7 Structure du code du projet **16**
7.1 Code embarqué 16
7.2 Code serveur 16

8 Réflexion sur les corrections à apporter à la 2e itération **18**

9 Gestion de projet **19**

10 Gestion des risques **20**

11 Schémas électriques **23**

1 Présentation du prototype

La première itération du prototype suit les requis établis en début de session. En effet, il comprend trois grandes parties : le terminal de recharge, le terminal de paiement et le pont/serveur.

Premièrement, à partir du terminal de recharge l'utilisateur peut savoir l'état de son compte et faire un dépôt d'argent. Le prototype accepte seulement les pièces de monnaie. Pour être en mesure d'effectuer une recharge, l'utilisateur doit présenter sa carte NFC. Le prototype est transportable et fonctionnel sur batterie, ce qui est utile pour l'avènement d'événements spontanés, comme les 5@8.

Deuxièmement, le terminal de paiement est le lien entre les consommateurs et les marchands. En effet, le marchand entre les articles achetés par le client pour obtenir le prix total des achats. Ensuite, le client doit présenter sa carte NFC pour effectuer le paiement. Une DEL et un écran transmettent les informations reliées aux transactions. Ce terminal fonctionne aussi sur batterie et il est facile d'installation, étant donnée sa forme compacte.

Troisièmement, le pont permet de transmettre les requêtes des terminaux au serveur pour effectuer les bonnes requêtes et obtenir les informations nécessaires. Un site internet permet aux utilisateurs de voir leur état de compte en temps réel. Pour les marchands, ils ont la possibilité de modifier leur liste de produits offerts et de voir leurs ventes. Pour les clients, ils ont la possibilité de voir leurs achats et leur solde de compte. Ce site web est disponible à tout moment.

2 Cahier des charges

Le cahier des charges contient une échelle de priorité dont voici les différents niveaux :

1. Fonctionnalités requises à la réussite du projet ;
2. Fonctionnalités primaires ;
3. Fonctionnalités secondaires ;
4. Fonctionnalités facultatives (celles qui ne seront probablement pas implémentées).

Tableau 2-1 – Fonctionnalités générales

Code	Fonctionnalité	Objectif	Description	Contraintes	Priorité	Implanté
1.1	Avoir des nœuds mobiles	Avoir des stations mobiles	Les stations de paiement doivent être autonomes	Utilisation d'une batterie et communication Zigbe	1	✓
1.2	Avoir un nœud fixe	Avoir un pont entre le serveur et les nœuds mobiles	La station fixe doit acheminer les messages entre les nœuds mobiles et le serveur	Communique en TCP/IP et Zigbee	1	✓
1.3	Communication Zigbee	Les nœuds mobiles et fixes doivent communiquer de façon sans-fil	Doit fonctionner dans un rayon moins de 40 mètres	La distance séparant 2 nœuds	1	✓

Tableau 2-2 – Fonctionnalités du terminal de paiement

Code	Fonctionnalité	Objectif	Description	Contraintes	Priorité	Implanté
2.1	Effectuer un paiement	Payer un article	Utiliser l'argent du compte client pour payer un produit	1 paiement par 2-3 secondes. Argent canadien.	1	✓
2.2	Fonctionner sur batterie	Le terminal de paiement doit être portatif	Le terminal peut être facilement déplacé	Durée : environ 12h	1	✓
2.3	Entrer le numéron de l'article acheté et la quantité	Définir un prix pour chaque article dans la BD	L'utilisateur doit être en mesure d'écrire le la quantité achetée de chaque article	Entre 1 et 9 produits par marchand	2	✓
2.4	Afficher l'état du paiement	Connaître l'état du paiement	L'utilisateur doit être assuré que le paiement à fonctionner	Au moins les informations sur autoriser ou non	2	✓
2.5	Lire le numéro d'un puce NFC	Identifier le client	Associer le numéro de carte à un compte dans la base de données	Doit être capable de lire la carte à 1 cm	3	✓
2.6	Afficher le prix total de la transaction	Connaître le prix total du panier	L'utilisateur doit être en mesure de savoir le prix payé pour son achat	Montrer jusqu'au 5 sous. Max 999$	3	✓
2.7	Détecter la position du terminal	Connaître l'emplacement du terminal	Être en mesure de connaître l'emplacement du terminal à tout moment	6-7 mètres près	4	

Tableau 2-3 – Fonctionnalités du poste de recharge

Code	Fonctionnalité	Objectif	Description	Contraintes	Priorité	Implanté
3.1	Effectuer une recharge	Déposé de l'argent dans le compte d'un usager	Les usagers doivent être en mesure	Doit être capable de prendre les pièces (2, 1, 25¢, 10¢, 5¢)	1	✓
3.2	Afficher l'état du compte	Connaître le balance du compte d'un usager	L'usager doit être capable de savoir comment il a d'argent dans son compte	Jusqu'au 5 ¢	2	✓
3.3	Compteur d'argent	Être capable de connaître la valeur de l'argent déposé	Doit compter la valeur de l'argent déposé par l'usager	Compter les pièces	2	✓
3.4	Fonctionner sur batterie	Le terminal de paiement doit être portatif	Le terminal peut être facilement déplacé	Durée : environ 4 mois	2	✓
3.5	Lire le numéro d'un puce NFC	Identifier le client	Associer le numéro de carte à un compte dans la base de données	Doit être capable de lire la carte à 1 cm	3	✓

Tableau 2-4 – Fonctionnalités du serveur

Code	Fonctionnalité	Objectif	Description	Contraintes	Priorité	Implanté
4.1	Autorisation des paiements	Associer une transaction au bon compte	Doit être capable de savoir si le compte a assez d'argent	État autorisé ou non	1	✓
4.2	Autorisation des recharges	Associer un dépôt d'argent au bon compte	Doit être capable d'ajouter le montant déposé au bon compte	État autorisé ou non	1	✓
4.3	Persistance des transactions et des états de compte	Être en mesure d'avoir un état du compte à tout moment	Le système doit enregistrer toutes les transactions reliées à un compte	À vie – Procédure de backup possible	1	✓
4.4	Interface Web pour les usagers	Faire le suivi de ses achats	L'utilisateur doit être en mesure de voir toutes les achats qu'il a fait	Affichage du prix, de la quantité et de l'endroit	2	✓
4.5	Interface Web pour les vendeurs	Faire le suivi de ses ventes	Le vendeur doit être en mesure de voir toutes les ventes qu'il a fait	Affichage du prix, de la quantité et de l'endroit (ou client)	3	✓
4.6	Protéger le réseau contre la falsification	Sécuriser les transactions entre les différents noeuds	Utiliser une connexion sécurisée pour éviter que quelqu'un vole des données ou de l'argent	Encryption des communications	4	

3 Branchement des différents éléments

3.1 Général

Afin de faire fonctionner le système, tous les éléments sur le schéma ci-haut doivent être branchés. Il faut suivre l'ordre suivant afin que tout fonctionne correctement :

1. Alimenter le routeur fourni et attendre qu'il soit démarré
2. Brancher deux câbles ethernet sur des ports libres (gris) du routeur
3. Brancher un câble ethernet dans le raspberry pi et l'autre dans le pont
4. Alimenter le raspberry pi et le pont, attendre un moment qu'ils démarrent
5. Alimenter les deux terminaux, ils se connecteront au réseau automatiquement

Note : La base de données sur le raspberry pi démarre automatiquement, mais pas l'application « serveur », il faudra suivre la méthode suivante :

1. Connecter en SSH avec pi :raspberry
2. Naviguer vers /ByPass/server
3. Exécuter la commande pm2 start server.js

3.2 Pont

Le branchement du pont est relativement simple. Le LPC est connecté au port RJ45 afin de communiquer avec le serveur. Il est aussi connecté au module Xbee Contrôleur afin de recevoir les requêtes des terminaux et transmettre les réponses du serveur.

3.3 Terminal de paiement

L'alimentation se fait à grâce à une batterie Li-ion sa tension nominal étant beaucoup trop élevée par rapport à notre circuit nous utilisons un régulateur de tension à découpage de type buck (MP1564) pour abaisser la tension à 5VDC, ce type de régulateur a un excellent rendement ce qui est une caractéristique recherchée dans un application embarquée, le XBee et le détecteur RFID MFRC522 n'acceptent cependant pas le 5VDC, nous utilisons donc le régulateur de tensions 3.3V du LCP1768 pour son alimentation.

Le LCD est alimenté en 5VDC et est commandé par un interface de 4 bits en parallèles pour les données et de 2 pins pour l'activation du module et une pour lui indiquer s'il s'agit d'une commande ou d'un caractère.

Le détecteur RFID est interfacé en SPI (mosi, miso, sck et le ss) de plus une pin (IRQ) nous indique quand une carte est détectée et une pin pour le reset. Une fois la carte détectée le module nous retourne son identifiant via SPI.

Finalement le XBee est interfacé en UART et possède une pin de plus pour le reset.

3.4 Terminal de recharge

Tout comme le terminal de paiement l'alimentation se fait grâce à une batterie Li-ion, sur ce montage nous avons ajouté un régulateur de tension 12V pour l'alimentation du monnayeur,

étant donné que nous n'avions à notre disposition qu'un régulateur de tension linéaire de type LM317, l'inconvénient avec ce type de régulateur est que beaucoup de puissance est dissipé dans ce dernier. C'est pour cela que nous avons rajouté le transistor mosfet 2N7000 pour alimenter ou non le monnayeur, en appliquant un tension sur la gate du 2N7000 nous connectons l'alimentation négative du monnayeur au GND et ceci nous permet d'économiser beaucoup d'énergie. Le monnayeur quant à lui est un module qui se configure à l'aide d'une interface utilisateur sommaire, une fois configuré et lorsqu'une pièce est détectée, le monnayeur nous délivre N flancs descendants (N correspondant à la valeur de la pièce que nous avons configuré).

4 Procédure de test du système du prototype

4.1 Code C

Conséquemment aux ressources limitées du microcontrôleur, les tests unitaires et fonctionnels sont effectués lors d'une recompilation avec un « flag » de test d'activité. Lorsque ce « flag » est activé, une macro active des parties de code qui impriment des informations de déverminage à l'écran.

Il est facile avec les informations de déverminage de détecter les valeurs erronées. Cette technique permet de cerner rapidement les sections de code problématiques lorsqu'un bug est détecté.

4.2 Code Javascript

Du côté du serveur javascript un « framework » de test automatisé a été utilisé. L'équipe a utilisé une technique de développement TDD conforme aux règles de l'art de l'industrie. Le développement se faisait de la façon suivante :

1. Écriture d'un test d'intégration d'une fonctionnalité
2. Écriture de la fonctionnalité
3. Amélioration cyclique en continu

L'avantage de ce type de développement est qu'il permet à chaque développeur d'être agnostique du travail des autres. En effet, chaque développeur peut exécuter la commande « npm test » pour que la suite de tests s'effectue. Si tous les tests passent, cela indique qu'ils n'ont pas brisé une fonctionnalité et qu'ils peuvent envoyer leur code sur github.

La configuration du « framework » de tests et des environnements de tests a été très long. Cependant, maintenant que c'est configuré, la productivité a nettement augmenté. L'équipe a donc été gagnante d'utiliser le TDD pour développer le projet.

4.3 Assurance qualité

Tableau 4-5 – Plan de test du prototype

Test	**Procédure**	**Résultat attendu**
Initialisation	Envoie de commande et de caractère avant l'initialisation	Rien ne s'affiche
Affichage d'une chaîne de caractère	"Envoie de la trame ""Borne de Paiement"" grâce à la fonction d'envoie"	La trame est bien affichée
Mise en veille	Mettre la pin correspondant à la led du rétro-éclairage à l'état bas	Affichage est en veille et consomme moins de courant

Sortie 5V	Alimenter l'alimentation de 9V à 15V et contrôle de l'oscillation résiduelle à l'oscilloscope à vide et avec une charge de 500mA (résistive)	La sortie est stable à 5V
Prise en charge du mode veille	Utiliser la fonction de mise en veille implémentée par MBED	Le MBED est en mode veille
Acquisition d'un caractère	Appuyer sur une touche et décoder le caractère reçu par les GPIO	Le caractère est le bon
Tester la communication client vers pont	Envoyer une trame simple du zigbee client (type routeur) vers le pont (type coordinateur)	Réception de la trame sur le coordinateur
Tester la communication pont vers client	Envoyer une trame de réponse du pont vers client lors d'une requête	Réception de la trame sur le client
Reconnaissance des pièces	Insertion des pièces devant être reconnues et décodage du signal reçu	Le montant reçu est celui inséré
Acquisition de la position	Décoder la trame reçue par la communication série	Latitude et longitude
Mise en veille	Mettre la pin de commande du transistor en haute impédance	GPS est en veille et consomme moins de courant
Reconnaissance de badge	Passer un badge à 1cm du lecteur	ID de carte lues par le SPI
Mise en veille	Mettre la pin NRSTPD à '0'	NFC en veille et ne consomme pas de courant
GET /api/accounts/ :id	GET /api/accounts/ :id	Le serveur renvoie un JSON contenant les données de l'utilisateur ou un JSON expliquant l'erreur rencontré lors de la création d'utilisateur.
GET /api/accounts	GET /api/accounts	Le serveur revoie la liste de tous les utilisateurs.
GET /api/items/ :id		Le serveur renvoie un JSON contenant les données de l'item ou un JSON expliquant l'erreur rencontré.

GET /api/items	GET /api/items	Renvoie la list de tout les items.
GET /api/accounts/transactions/ :id		
Tester la communication zigbee vers serveur	Envoyer une requête HTTP Post du pont vers le serveur	Réception de la trame sur le serveur
Tester la communication client vers serveur	Envoie d'une trame par zigbee du client	Réception de la trame sur le serveur
Tester la communication serveur vers client	Réponse du serveur lors d'un réception de trame du client	Réception de la trame sur le client
Roulement de la MEF	Faire rouler le squelette du code du terminal de paiement	Les bons états s'exécutent (confirmé par des printf)
Roulement de la MEF	Faire rouler le squelette du code du terminal de recharge	Les bons états s'exécutent (confirmé par des printf)
Lecture de la carte RFID	Présenté une carte valide devant le lecture RFID	Le lecture détecte la carte et il associe le bon compte à la carte
Effectué une recharge	Effectué une recharge dans un compte utilisateur. Avec lecture d'une carte et dépôt d'argent	Le solde de l'utilisateur est mis à jour suite au dépôt d'argent

5 Explications techniques des fonctions et composants

5.1 Pont

5.2 Terminal de recharge

5.3 Terminal de paiement

5.4 Serveur et base de données

Un serveur « node » assisté du langage javascript a été utilisé pour construire les fonctionnalités « backend » de notre projet. Le serveur est constitué de deux parties principales : un service d'api et un site web.

5.4.1 API

Le service d'api est du format REST. Il permet d'accéder, d'ajouter, de modifier ou de retirer des ressources de la base de données. Chacune de ces opérations est sans état. L'avantage d'utiliser un api REST est que toutes les plateformes intégrant une fonctionnalité de requête HTTP et de sérialisation json peuvent facilement accéder à nos services de manière standardisée et agnostique aux détails d'implémentation de notre serveur. Voici une liste des services de notre api :

- Create Account
- Login Account
- Get Account
- List All Accounts
- Create Item
- Delete Item
- List Items
- Create Transaction
- List Transactions

L'équipe a d'abord essayé d'utiliser les services ci-dessus avec le pont zigbee, cependant le microcontrôleur du pont n'avait pas assez de mémoire pour sérialiser en JSON les informations reçues par ses nœuds, tel que mentionné précédemment. Cependant, les nœuds pouvaient le faire et le pont a donc été utilisé comme un relayeur des paquets JSON au serveur web. Toutefois, puisque le pont ne connaît pas le contenu des trames qu'il relaie, il ne peut envoyer ses requêtes qu'à un seul « endpoint » de l'api REST.

Un « endpoint » spécial nommé « zigbee/bridge » a été créé pour désérialiser le JSON relayé par le pont zigbee et exécuter les actions nécessaires. Ce service comporte 4 fonctionnalités :

- Créer une transaction (débiter et créditer les bons comptes)
- Recevoir le montant restant sur un compte client
- Ajouter de l'argent à un compte
- Afficher le total pour un achat

5.4.2 Site Web

Le site web est une interface qui permet aux marchands de voir leurs transactions effectuées, voir leurs produits, ajouter leurs produits et supprimer leurs produits. Les clients peuvent également accéder au site web pour voir leurs achats.

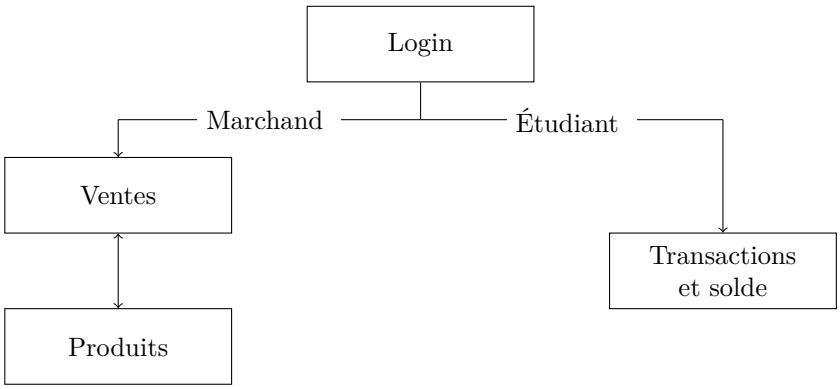

FIGURE 5-1 – Schéma des interactions avec le site web

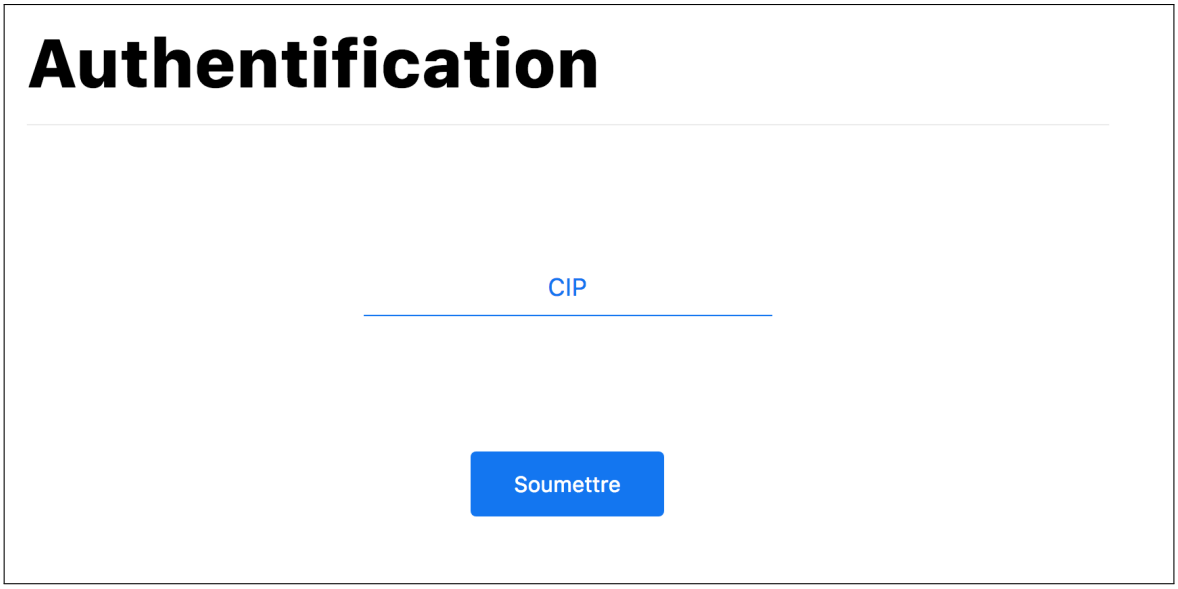

FIGURE 5-2 – Fenêtre d'authentification du site web (login)

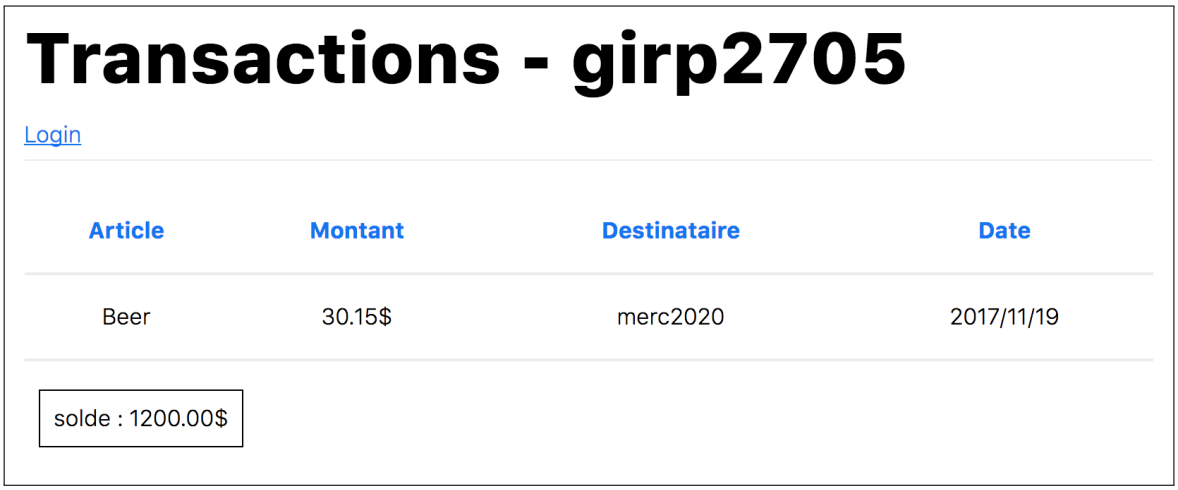

FIGURE 5-3 – Transactions et solde d'un étudiant

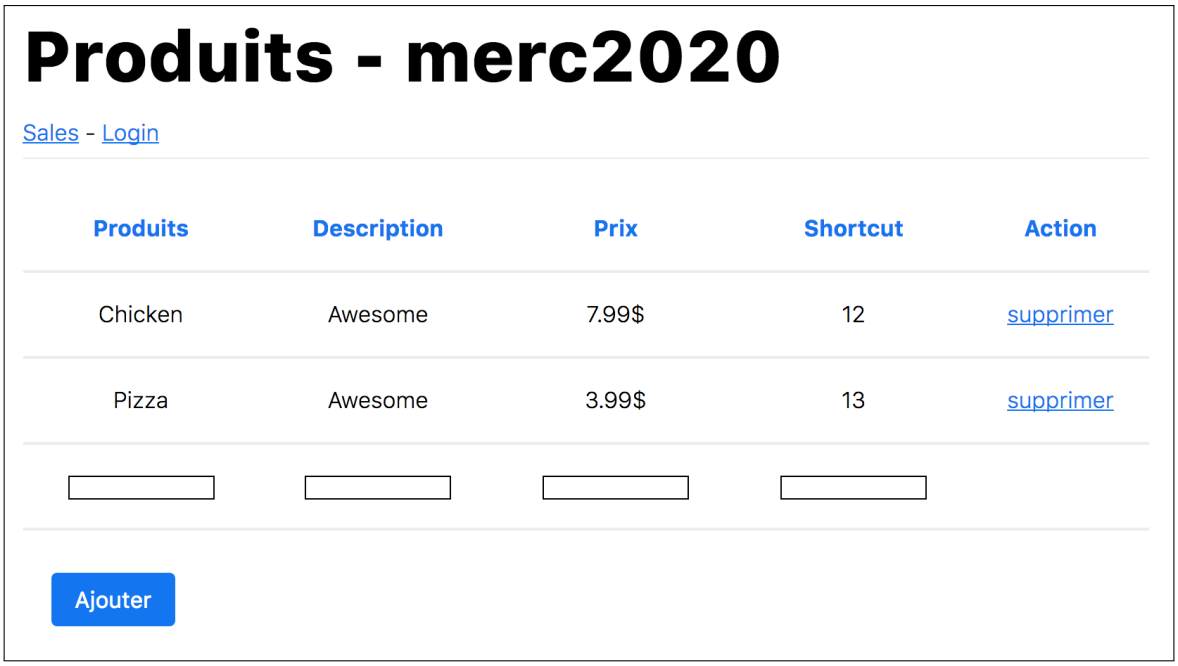

FIGURE 5-4 – Liste des produits d'un marchand

FIGURE 5-5 – Ventes d'un marchand

Le site web a été créé sur le même serveur web que celui de l'api REST. Il utilise un modèle MVC. Il possède ses propres contrôleurs, routes et vues (*views*), mais il partages ses modèles avec le service REST. Le « rendering » de l'affichage se fait du côté serveur et des requêtes ajax sont utilisées pour rafraîchir les données automatiquement sans rafraîchissement de la page.

5.4.3 Postgres avec tables

Une base de données du format *postgres* a été utilisée pour le projet, laquelle comporte 4 tables :

— Accounts

— Items

— Transactions

— LineItems

La table « Accounts » comporte les comptes des clients et des marchands, leurs informations à propos de leur CIP, leurs cartes RFID, leur PIN et leur solde. La table « Items », fait la relation entre des items et un marchand qui les possède. La table « LineItems » fait la relation entre des items, leur quantité et une transaction. La table « Transactions » fait la relation entre un client et un marchand pour une transaction.

Le plugin « sequelize » a été utilisé pour faire nos modèles de la base de données en javascript et pour faire nos migrations de base de données ainsi que des « seeds » pour la base de données.

6 Diagrammes UML

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7 Structure du code du projet

7.1 Code embarqué

Le code de chaque composante embarquée est structuré de la même façon. À la racine du fichier, on observe les dossiers *lib* (qui contient les bibliothèques utilisées) et *src* (qui contient le code du projet). Les bibliothèques utilisées sont parfois maintenues par l'équipe, notamment pour le RFID, le clavier, le JSON et le Xbee (pour complémenter la librairie officielle). Cela permet de mettre à jour le code commun de toutes les composantes facilement lorsqu'il change.

Au niveau du code source du projet, la sous-division dépend de la fonction de chaque composante, mais chaque fonction est encapsulée dans son propre fichier avec les fichiers sources et les en-têtes nécessaires.

Terminal de paiement Contient des modules pour les entrées claviers (Input), pour la communication avec le serveur par Xbee (Server) et pour gérer les LED (`LED`)

Terminal de recharge Contient le module pour la communication avec le serveur par Xbee (`Server`)

Pont Contient le module pour la communication Xbee avec les terminaux (`Xbee`) et le module pour la communication Ethernet avec le serveur (`Server`)

Dans tous les cas, les projets contiennent un fichier de configuration (`config`) qui permet de configurer la composante facilement. Ils contiennent également un fichier debug pour activer le mode debug (`debug` ou `BypassDebug`).

7.2 Code serveur

Au niveau du code source pour le serveur, une organisation MVC est utilisée avec le framework Express de node.js.

Voici les fichiers de configuration importants :

Package.json Liste des dépendances du projet et leurs versions

Server.js Configuration du serveur

Routes.js Routes du serveur API

Ejs_routes.js Routes de l'interface web

Docker-compose.yml Configuration de docker pour démarrer la base de données

Voici les dossiers important pour le développement :

Models Comporte tous les modèles en relation avec la base de données

Views Comporte les vues pour le site web

Controllers Contient les contrôleurs qui font la relation entre les vues et le modèle et implémente les API REST.

Voici les dossiers importants pour l'aide au développement de l'application :

Test Collection de tests automatisés à exécuter à avant d'envoyer du code sur github pour vérifier que les nouvelles fonctionnalités n'ont rien brisés. On devrait écrire un test pour chaque fonctionnalité au fur et à mesure qu'ils sont développés

Migrations Les migrations de la base de données

Helpers Des fonctions pour aider les contrôleurs et les modèles dans leurs tâches

8 Réflexion sur les corrections à apporter à la 2e itération

La première amélioration importante à réaliser pour la deuxième itération serait au niveau de la sécurité. En effet, pour le moment il n'y a aucune sécurité et c'est loin d'être idéal pour un système de paiement. Il faudrait commencer par chiffrer les communications Xbee pour éviter que des acteurs malveillants puissent espionner le réseau. Ensuite, il faudrait authentifier les nœuds sur le réseau afin de s'assurer qu'ils sont autorisés à effectuer des actions sur les comptes. Finalement, il faudrait aussi avoir un mécanisme d'authentification sur le site internet pour s'assurer que seul le propriétaire du compte puisse accéder à ses informations.

La deuxième amélioration qu'il serait important de réaliser est au niveau du code qui tourne sur les microcontrôleurs. Il faudrait essayer d'effectuer une gestion d'erreurs plus complète et avancée. Pour le moment, l'utilisateur a seulement une idée minimale des problèmes qui surviennent dans le système et le programme ne vérifie pas si les numéros de marchand ou de produits existent avant le paiement. Également, il faudrait que le système réessaye lorsque des erreurs de communication surviennent au lieu d'annuler la transaction. Finalement, il serait bien de faire un code un peu plus asynchrone sur les terminaux, qui sont très linéaires pour le moment, notamment pour permettre d'annuler ou modifier une transaction à n'importe moment dans le processus.

La troisième amélioration qui serait intéressante serait au niveau du matériel. Il faudrait d'abord incorporer le GPS tel qu'il était prévu dans le cahier des charges initial afin d'avoir une position des différents terminaux en tout temps. Ensuite, il faudrait tester de façon précise la durée de vie des terminaux sur batterie afin de pouvoir informer les clients et potentiellement les changer si nécessaire. Finalement, il faudrait essayer de réduire au minimum la consommation électrique en mettant en veille les différents appareils lorsque possible.

9 Gestion de projet

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

10 Gestion des risques

Tableau 10-6 – Grille de cotation de 1 à 10

Note F	**Fréquence**	**Note G**	**Gravité**	**Note D**	**Probabilité de non-détection**
10	Permanent	10	Mort d'homme	10	Aucune probabilité de détection
5	Fréquent	5	Conséquences financières et/ou matérielles	5	Possibilité de détection
1	Invraisemblable	1	Pas grave	1	Détection infaillible

Tableau 10-7 – Risques de gestion

Mode	**Effet**	**Cause**	**F**	**G**	**D**	**C**	**Action**	**F**	**G**	**D**	**C**
Airtable hors service	Aucun accès à la gestion de projet	Hors de notre contrôle	1	4	1	4	Suivre les instructions du site web	1	1	1	1
Mauvaise mise à jour des tâches	Status erroné du projet	Indiscipline des membres de l'équipe	3	2	3	18	Faire un suivi hebdo-madaire	2	2	2	8
Limite de cartes sur Airtable	Aucune possibilité d'en ajouter de nouvelles	Utilisation de la version gratuite	1	1	1	1	Créer un 2e Airtable	1	1	1	1
Éléments de conception manquants	La définition du projet n'est pas complète	Oubli de l'équipe	1	1	2	2	Concevoir les éléments manquants	1	1	1	1

Tableau 10-8 – Risques par rapport au *hardware*

Mode	**Effet**	**Cause**	**F**	**G**	**D**	**C**	**Action**	**F**	**G**	**D**	**C**
Mauvais branchements	Le non-fonctionnement du système	Les pièces ont été mal branchées	2	4	2	16	Vérifier les points de tests	2	2	2	8
Défectuosités	La pièce ne fonctionne pas comme prévu	La pièce est brisée	1	2	2	4	Acheter des nouvelles pièces	1	3	1	3
Manque de broches	Incapable de brancher toutes les pièces	Utilisation de toutes les broches	2	2	2	8	Mettre un multiplexeur	1	2	2	4
Manque d'événements	Les ZigBee ne reçoivent pas tous les paquets	Mauvaise communication entre les ZigBee	1	2	3	6	Gérer la réception des paquets	1	2	2	4
Température trop élevée	Incendie dans l'un des modules	Gestion des courants avec la batterie	3	2	3	18	Ventiler les modules	1	2	2	4
Court-circuit	Défectuosité des composantes électroniques	Mauvais branchements	3	2	2	12	Confectionner des branchements à sens unique	1	1	2	2
Chute fatale	Défectuosité des composantes électroniques	Choc brisant les composants	2	2	2	8	Vérifier les points de tests et changer les composantes défectueuses	1	1	2	2
Obstruction	L'utilisateur ne peut pas recharger son compte	Vandalisme. Par exemple : une gomme dans la machine	3	3	3	27	Réparation de la machine	2	3	2	12

Tableau 10-9 – Risques par rapport au *software*

Mode	Effet	Cause	F	G	D	C	Action	F	G	D	C
Mauvaise communication ZigBee	Les ZigBee fonctionnent mal	Problème dans le protocole de communication	2	3	3	18	Corriger les problèmes de communications (protocole)	1	3	2	6
Segmentation de mémoire	Le programme ne fonctionne pas bien	Trop grande utilisation de la mémoire Hype	2	2	3	12	Vérifier la gestion/déclaration des variables, utiliser une mémoire pool	1	2	3	6
Maintenance du code	Le code n'est pas lisible	Programmeur non soucieux du détail	2	1	1	2	Ajout de commentaires, clarification du code	1	1	1	1
Crash du programme	Le programme arrête de fonctionner	Il y a un problème majeur dans le code	1	5	2	10	Trouver les défectuosités du programme, mieux gérer les erreurs	1	2	2	4
Limite de performance	Le programme n'est plus en temps réel	Le CPU n'est pas assez puissant pour la tâche demandée	2	5	2	20	Développer sur un Mbed plus puissant	1	4	3	12
Json trop lourd pour le LPC1768	On ne peut pas charger le programme sur la carte	Le LPC1768 ne dispose pas d'assez de ressources	5	3	1	15	Ne pas utiliser le json	1	2	3	6
Erreur traitement du serveur	Gestion des erreurs à risque	Mauvaise gestion des erreurs	1	2	2	4	Améliorer la gestion des erreurs	1	1	2	2
Mauvaise lecture des pièces de monnaie	Mauvaise détection des pièces	Erreur de lecture avec l'ADC	1	2	2	4	Améliorer la lecture des pièces	1	1	2	2

11 Schémas électriques

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.