École Polytechnique de Montréal
Département de génie informatique et génie logiciel
INF8405—Informatique mobile

PROJET : SHOPPING SOLVER

Jiajing Liang	**1687588**
Zhang Chunxia	**1751567**
Olivier Rwahama	**1754583**

Rapport de laboratoire présenté à
Mauricio Mendoza Medellin

27 Avril 2015

Table de matière

I. Introduction

Notre projet consiste à réaliser une application permettant de faire les paiements pour les achats et de donner des recommandations des promotions selon les listes d'achats historiques.

Le système entier va se composer de trois parties. Une application installée sur les mobiles Android destinées aux utilisateurs généraux, qui permet aux utilisateurs de scanner les codes barre des produits dans les boutiques, de faire les paiements à la fin, de consulter les informations concernant le compte notamment le solde restant, les transactions historiques ainsi les produits achetés les plus fréquemment; une deuxième partie implanté sur les terminal des supermarchés, et la troisième partie à coté d'un serveur pour gérer les transactions des paiements concernant les comptes d'utilisateur et pousser les informations des promotions aux terminaux mobiles des utilisateurs.

Le principal scénario de notre projet est la description suivante: Un utilisateur de notre application qui fait des achat dans un supermarché peut scanner les codes barre de tous les produit dans son chariot en choisissant leurs produits envisagés. Ça peut lui donner une liste des produits comme la facture qu'il va payer quand il passe au caisse. A ce moment la, le terminal du supermarché échange les informations de la transaction avec le mobile afin d'effectuer le paiement et de gérer un reçu. Jusqu'à là, on finit tous les processus d'achat indépendamment. Dans notre projet, on va se concentrer sur la partie de l'application mobile et un serveur simplement simulé afin de faire un scénario simple. La partie des terminaux de supermarché ne sera pas notre tâche principal pour notre projet. Du coup, on simule l'échange de données pour le paiement entre l'application Android et un terminal dans les supermarchés en utilisant juste une autre code barre ou code QR pour donner les données de la boutique à l'application mobile.

En faisant les achats, l'application peut analyse le statistique d'habitude d'achats qui trouvera les produits achetés le plus fréquemment d'une personne selon des listes d'achat historique afin de permettre lui pousser les promotions qui proposent ces produits avec un prix moins élevé. C'est-à-dire, on peut lui donner des recommandations lesquels supermarchés des promotions ou proposent un prix plus bas avec leurs localisations sur une carte ainsi l'acheminement de la localisation courante de l'utilisateur jusqu'à les boutiques.

II. La méthodologie

A. La répartition des travaux

La mise en marche de répartition de tache dans notre équipe a été fait selon la disponibilité et la connaissance de chacun(e) membre du groupe. La spécification, la conception générale et détaille sont effectuées par l'ensemble de l'équipe. L'interface de l'application Android est dessiner par Olivier Rwahama; le codage de côté d'application mobile a été fait par Chunxia Zhang, le codage de côté serveur a été fait par Jiajing Liang. Enfin l'intégration et le rapport sont été élaborés par l'ensemble de l'équipe.

B. Les phases de réalisation

Voici dessus les différentes phases pour la réalisation de application:

- Analyse des besoins et faisabilité
  On commence le développement d'abords par l'expression, le recueil et la formalisation des requis fonctionnels et non fonctionnels de l'application ainsi de l'ensemble des contraintes.
- Conception générale
  Cette phase consiste à élaborer la spécification de l'architecture générale du logiciel.
- Conception détaillée
  Ensuite on passe à l'étape de la conception détaillée qui a pour le but de définit précisément chaque sous-ensemble du logiciel.
- Codage
  Ceci est l'étape de l'implémentation, l'écriture et la mise au point des fonctionnalités définies lors de phases de conception
- Tests unitaires
  Cette phase consiste à vérifier individuellement que chaque sous-ensemble du logiciel est implémentée conformément aux spécifications. Dans le cadre de notre projet, cette phase est réalisé en parallèle avec la

phase de conception. C'est-à-dire, on a fait les tests fonctionnels au fur et à mesure une fois qu'on a implémenté certaine fonctionnalité.

- Intégration
  L'objectif de cette phase est de s'assurer de l'interfaçage des différents éléments (modules) du logiciel. Dans ce projet, on a décidé de répartir des tâches de développement au différents membres de l'équipe, et chacun développe les certaines fonctionnalités du logiciel isolément, il est important de faire un point et faire les tests d'intégration ensemble régulièrement. Ceci peut garantir la qualité du développement.
- Qualification
  A cette étape, on essaie de vérifier de la conformité du logiciel aux spécifications initiales.

C. La structure de l'application

Voici dessus un diagramme de déploiement (Figure 1) qui montre l'architecture globale de notre application. En gros, notre système est décomposé de quatre composants: une application Android, un service Web déployé sur un sur le Glassifish serveur, un MySQL serveur de base de données et le Google GCM connection serveur. L'application Android communique avec le service Web en utilisant le protocole SOAP avec XML comme le format de données d'échange, afin de chercher et de mettre à jours les informations concernant le compte d'utilisateur, et les transactions des paiements. A côté du service Web, à la réception des requêtes venant du client Android, il collecte les données dans la base de données MySQL en établissant les connections à l'aide de JDBC connection pool, effectue les opérations relatives et donne une réponse au client. Quand il y a des produits promotionnels, il faut pousser les notifications concernant la promotion au client. Ce fonctionnement est implémenté de façon que notre serveur d'application lance une requête HTTP avec les données sous format Json au Google GCM Connection Server en indiquant les IDs des périphériques Android pour que le serveur de Google puisse acheminer les notification aux mobiles via le protocole HTTP avec les données Json.

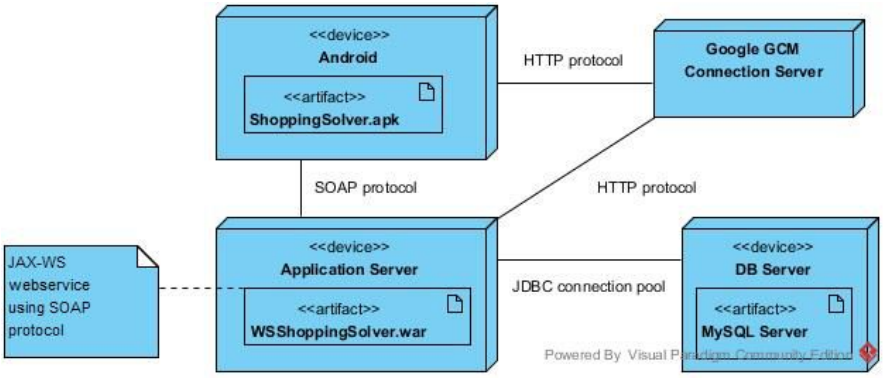


Figure 1: Diagramme de déploiement de architecture global

Afin de nous aider à comprendre mieux l'architecture de notre système de façon plus détaillée, voici dessus un diagramme de déploiement qui détaille plus sur l'architecture de notre application Android et de notre service Web. Comme illustré dans la Figure 2, notre application Android et service Web est complètement implémentés avec une architecture multi-couche. De côté mobile, l'application est décomposée de la couche de présentation, la couche de service (métier), la couche d'accès aux données ainsi les composants 'Cross Cutting'. A l'issue de ce concept de multi-couche, notre application est décomposée en 5 package: un package « ui » dans lequel on trouve tous les classes qui modifier l'interface d'utilisateur ainsi les activités; un package « service » dans lequel on trouve les service qui offre les opérations d'arrière-plan comprenant chercher les données dans la base SQLite via les objets d'accès aux données et communiquer avec le côté serveur afin de gérer le compte et les transaction; un package « database » dans lequel on a les classes qui proposent l'accès aux données dans la base SQLite; un package « model » dans lequel se trouve tous les classes qui modélise les donnés concernant le compte du client et les transactions; un package « util » dans lequel les classes fournissent les méthodes statique de fonctions indépendantes pour opérer les objets de modèle, faire la conversion de différents formats de données, etc.

Au côté serveur, l'architecture est conçu du même principe. On a en gros 4 package: un package « admin_ui » dans lequel on trouve les classes pour la conception d'une application Web simple afin de proposer la possibilité d'administrer les données concernant les produits, les boutiques, les comptes client, les transactions, etc; un package « webservice » dans lequel on a les classes de service Web qui nous permettent de répondre le requêtes provenant des client Android; un package « dao » dans lequel on trouve les classes d'accès aux données; un package « model » dans lequel on a tous les classe d'entité qui modélisent les comptes client, les produits, les boutiques, les transactions, etc.

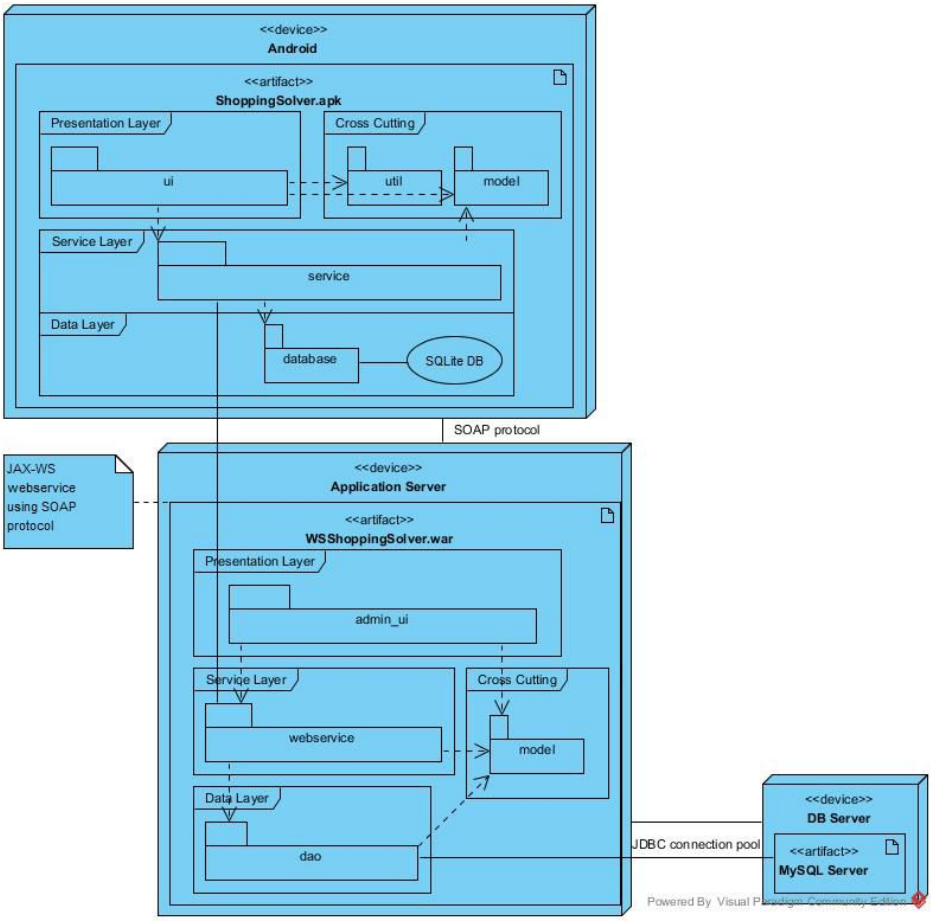


Figure 2: Diagramme de déploiement avec les paquetages détaillés

Avant d'entrer en détail d'architecture de notre application Android, voici dessous la Figure 3 qui illustre la modélisation des données afin de répondre les besoins fonctionnels de notre application. Pour chaque *client*, il dispose une liste de *registeddevice* qui sont les mobiles sur lesquels il a installé notre application et login avec son compte. On a un ensemble de *product* qui représente tous les produits sur le marché. Pour chaque produit, il appartient d'un *productcategory*, et il a de même code barre peu importe il est dans quel supermarché. On a un ensemble de *shopbranch* qui représente tous les boutiques dans notre système. Chaque boutique appartient d'un *shopbrand* qui est le groupe (ou la marque) de la boutique (comme Walmart, Metro, etc). *product_price_in_shop* modélise le prix avec les détails de taxe d'un produit dans une boutique puisque un produit peut se vendre à différent prix avec taxe provinciale différent dans différentes boutiques. *transact* modélise une transaction qui se passe quand un client effectue un paiement en détaillant le client associé et le boutique où la transaction se passe. *product_transact_record* modélise les produits achetés dans une transaction avec le détail sur le prix, la quantité ainsi les taxes appliquées. Enfin, *client_favorite_products* représente les produits qui sont achetés le plus par un client.

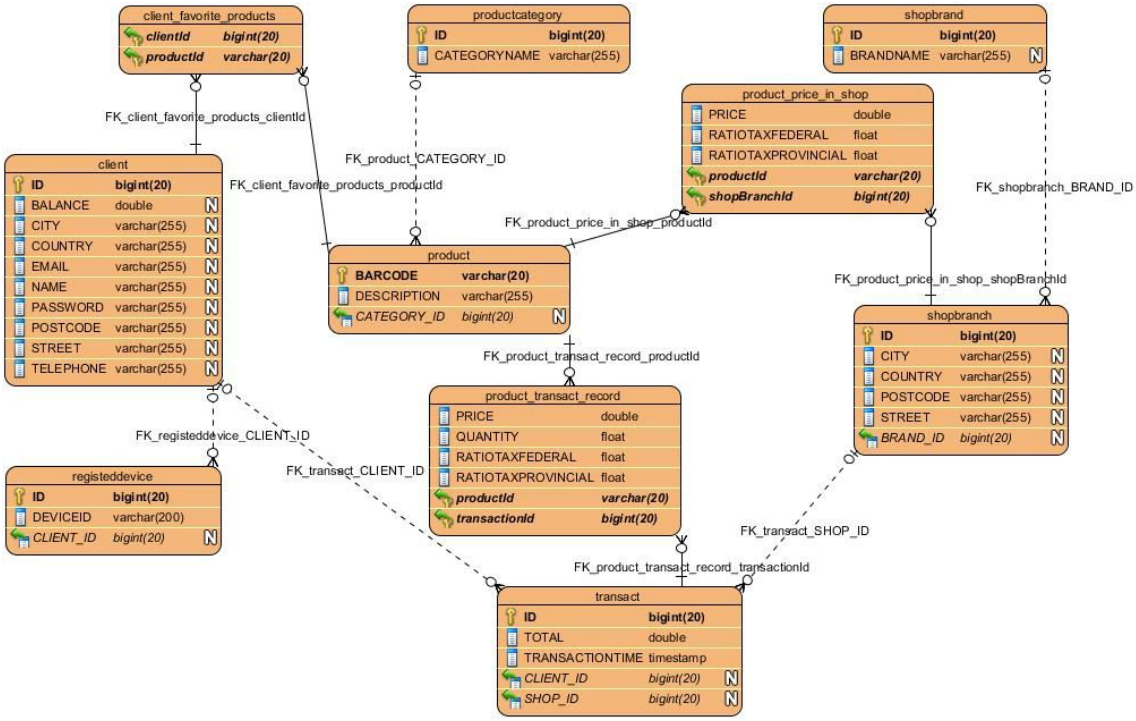


Figure 3: Diagramme de entité-assoiciation pour la modélisation du problème

Après une description générale sur l'architecture globale de notre système, d'application ainsi sur la modélisation de données, on va entrer dans le détail de l'architecture de l'application client d'Android à l'aide de la Figure 4 dessous qui illustre les classes principales dans chaque packages. Comme on a mentionné précédemment, l'architecture de l'application mobile est composée de 5 packages dont une explication sur les classes regroupés par package est donné au dessous:

- Package « model »: ici on détaille juste les 3 classes les plus importantes pour la modélisation du problème
  - Classe « Client »: représenter le compte d'un client qui utilise l'application
  - Classe « ShoppingRecord »: représenter un produit acheté pour un paiement avec les informations sur la description du produit, le catégorie, le prix, la quantité et les taxes appliquées du produit.
  - Classe « Transaction »: représenter une transaction avec les détails sur la boutique dans laquelle on fait l'achat, la somme à payer, les taxes totales, ainsi une liste de produits (ShoppingRecord) achetés.
- Package « util »: ici on a 2 classes les plus importantes pour la mise en format des données et la maintenance d'application
  - Classe « HandleXML »: la classe pour convertir et extraire les données d'XML puisqu'on utilise le protocole SOAP
  - Classe « ShoppingSolverApplication »: une classe héritant la classe « Application » qui manipuler les données à travers l'application ainsi les différentes activités
- Package « database »: ici on trouve les classes pour l'accès aux données de la base SQLite (données locales)
  - Classe « ClientDataSource »: la classe pour l'accès particulièrement les données concernant le compte d'un client
  - Classe « HabbitDataSource »: la classe pour l'accès particulièrement les données concernant les statistiques sur l'habitude d'achat du client
  - Classe « DBHelper »: la classe pour la maintenance les tables SQLite du point de vue globale (la création, la suppression)
- Package « service »: ici on a une service d'arrière-plan pour chercher les données dans la base locale ou à travers le service Web correspondantes aux l'interaction de l'utilisateur sur l'interface graphique.
  - Classe « ShoppingSolverIntentService »: cette classe cherche les données dans la base SQLite ou envoie les requêtes SOAP à la service Web pour demander les données au serveur, afin de répondre la demande de l'utilisateur. Puis, elle regroupe et traite les données afin de les renvoie aux différents « Receiver » pour la mise en page sur l'interface graphique.

- Package « ui »: ici on a les activités, les fragments pour interagir avec l'utilisateur.
    - Classe « NewCountActivity »: l'activité qui lance quand on veut créer un nouveau compte. Elle demande au « ShoppingSolverIntentService » pour envoyer les données au service Web pour la création de compte.
    - Classe « CreateCountResultReceiver »: la classe privée de « NewCountActivity » pour traiter les données de la réponse SOAP pour la création de compte.
    - Classe « LoginActivity »: l'activité pour le login d'un compte. Elle demande au « ShoppingSolverIntentService » d'envoyer les données au service Web pour la vérification de l'identifiant du compte et le mot de passe.
    - Classe « LoginResultReceiver »: la classe privée du « LoginActivity » qui traite la réponse SOAP pour le login. Si la vérification a réussie, on lance le « MainActivity », sinon l'on reste sur le « LoginActivity » et informer l'utilisateur l'erreur d'identification.
    - Classe « MainActivity »: l'activité principale de notre application, qui fournit la fonctionnalité de faire un paiement, voir l'état courant du compte, voir les statistiques sur les habitudes d'achat, et voir les historiques des transaction. On utilise les « Fragment » attachés à l'activité afin de gérer les interactions avec l'utilisateur pour chaque de ces fonctionnalités. Quand l'utilisateur choisit une fonctionnalité sur la liste de navigation, le « Fragment » correspondant remplace le « Fragment » courant.
    - Classe « PaymentFragment »: le « Fragment » pour faire un achat et le payer. Il envoie une requête SOAP pour faire une paiement via « ShoppingSolverIntentService ».
    - Classe « ItemDialogFragment »: le « Fragment » pour consulter la description d'un produit et éditer la quantité d'un produit.
    - Classe « CustomerBaseAdapter »: l'adapteur pour modifier les données concernant l'achat courant quand on éditer la liste d'achat.
    - Classe « PayResultReceiver »: le classe privée de « PaymentFragment » pour traiter particulièrement la réponse SOAP après une transaction est envoyée sur le service Web, et gérer une facture électronique.
    - Classe « CountFragment »: le « Fragment » pour la consultation de l'état du compte. Il envoie une requête SOAP pour chercher ces information via « ShoppingSolverIntentService ».
    - Classe « TransactionFragment »: le « Fragment » pour voir les historiques des transactions. Il envoie une requête SOAP pour ces données via « ShoppingSolverIntentService ».
    - Classe « HabitFragment »: le « Fragment » pour consulter les habitudes d'achat. Il chercher les données dans la base SQLite via « ShoppingSolverIntentService ».
    - Classe « SSResultReceiver »: la classe privée de « MainActivity » qui traite la réponse SOAP quand on veut consulter l'état du compte, les transactions historiques, etc.
    - Interface « StartCommunication »: l'interface de « MainActivity » pour partager les données entre les « Fragment »
    - Classe « MapsShopsActivity »: l'activité qui lance quand on clique sur une notification de promotion arrivée sur le mobile. Il affiche la localisation de la boutique où il y a une promotion sur un produit que le client intéresse, par rapport la localisation courant du client.

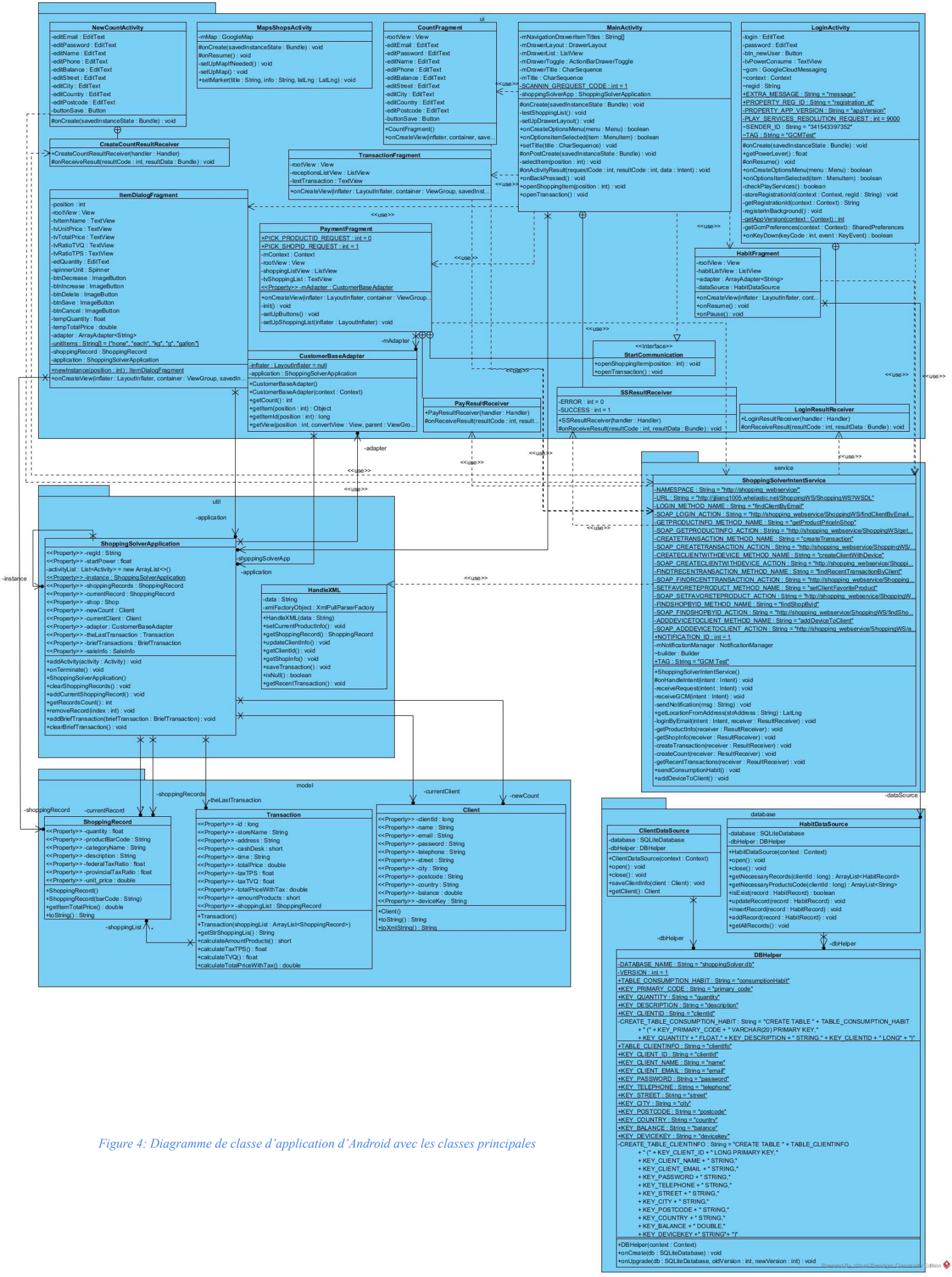


Figure 4: Diagramme de classe d'application d'Android avec les classes principales

D. Les séquences d'exécution

Il y a en général trois séquences d'exécution les plus importantes dans notre application: la séquence de login, la séquence pour faire un paiement, et la séquence pour consulter la localisation d'une boutique où il y a un produit promotionnel intéressé par le client.

Voici dessous la diagramme de séquence sur la Figure 5 qui illustrer les interactions entre les classes lorsqu'on login. L'utilisateur saisit l'identifiant et le mot de passe du compte, puis clique sur le bouton de connexion. Le « LoginActivity » demande le « ShoppingSolverIntentService » d'envoyer ces données d'identification au service Web à travers une requête SOAP. A la réception de la réponse SOAP, le « ShoppingSolverIntentService » extrait le résultat à partir des données XML et le retourne au « LoginActivity » afin de lui laisser décider si l'on peut amener à l'activité principale d'après le résultat d'identification.

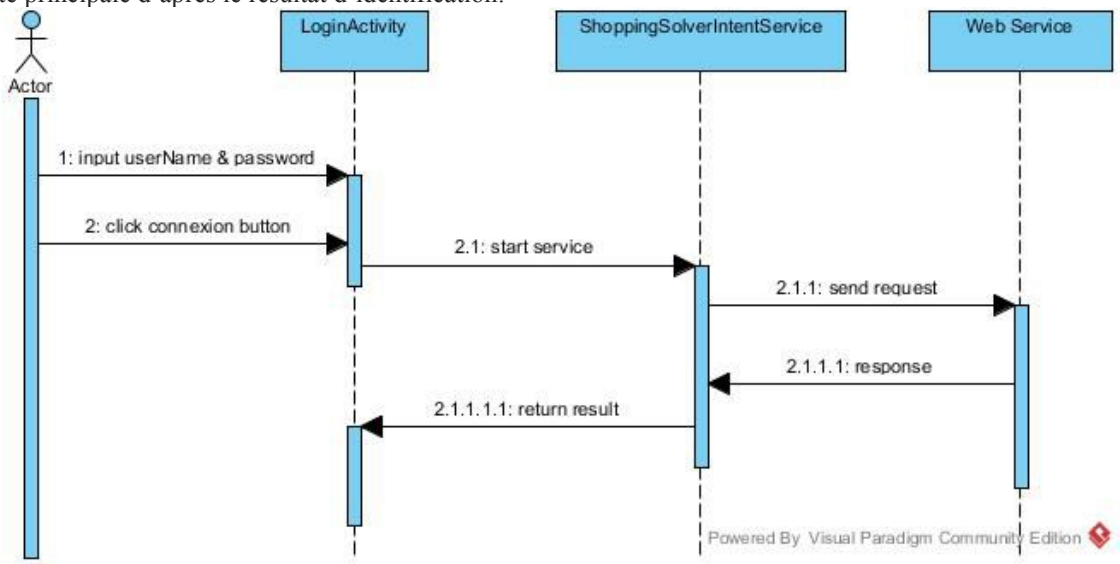


Figure 5: Diagramme de séquence pour le login

La deuxième séquence importante est celle pour faire une paiement. Voici dessous la Figure 6 qui montre cette séquence d'exécution. Sur le « PaymentFragment », l'utilisateur clique sur le bouton pour scanner le code barre d'un produit. Le fragment lance l'activité « CaptureActivity » pour scanner d'un code. Dès que l'information dans le code barre est extrait, on passe la main au « MainActivity » qui demandera au « ShoppingSolverIntentService » d'envoyer une requête SOAP au service Web pour chercher les information concernant le produit scanné. A la réception de la réponse SOAP, le « ShoppingSolverIntentService » traiter les données de retour sous format XML et extrait les informations qu'on a besoin. Ensuite, il envoie ces informations au « MainActivity ». Le « MainActivity » passe ensuite ces données au « ShoppingSolverApplication » pour mettre à jours les données concernant la liste d'achat courant et lui laisser mettre à jour l'interface graphique sur le « PaymentFragment ». Ce processus d'ajout des produit à la liste d'achat se répète autant que l'utilisateur veut jusqu'à il veut terminer et faire le paiement. Quand l'utilisateur clique sur le bouton pour le paiement sur le « PaymentFragment », il demande au « ShoppingSolverIntentService » d'envoyer une requête SOAP au service Web pour ajouter les données d'une transaction. Le service Web va retourne les informations pour la facture de ce transaction. A la réception de ces informations, le « ShoppingSolverIntentService » traiter ces données XML et les envoie au « PaymentFragment » pour l'affichage de la facture.

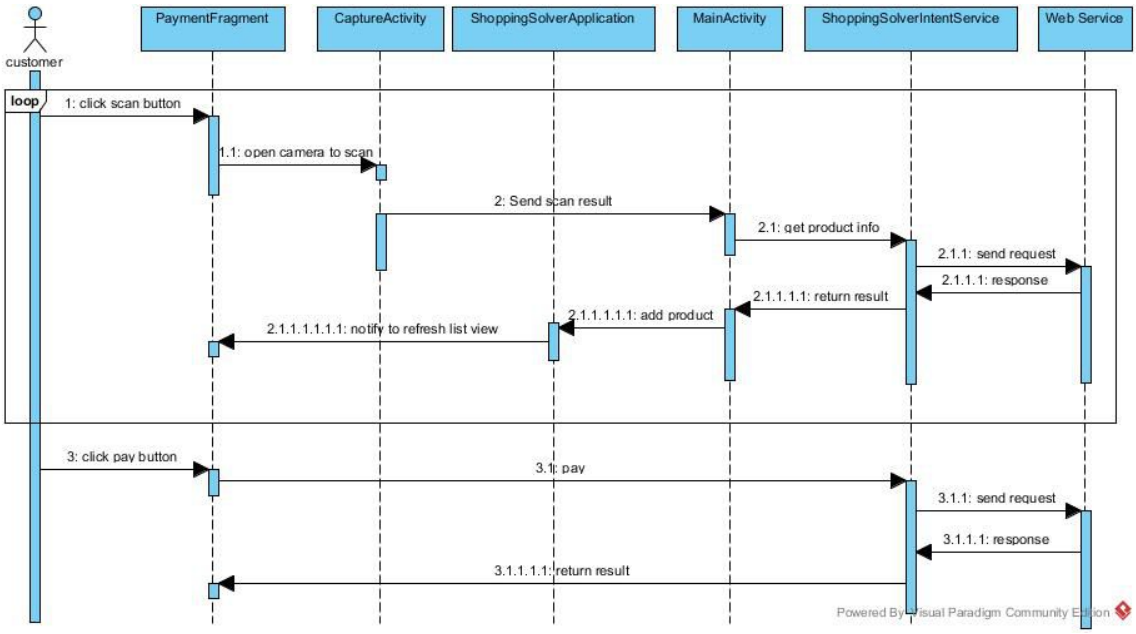


Figure 6: Diagramme de séquence pour effectuer un achat

Le troisième séquence importante est celle pour l'affichage de la localisation d'une boutique ou il y a une produit promotionnel intéressé par le client à la réception d'une notification. Voici dessous la Figure 7 qui montre ce diagramme de séquence. Quand il y une promotion sur un produit, on trouve tous les mobiles des client qui sont intéressé par ce produit et envoie les messages au Google GCM Service en indiquant les mobiles à notifier. Le Google GCM Service va acheminer les messages de notification aux mobiles concernant. A la réception du message , le « GcmBroadcastReceiver » demande au « ShoppingSolverIntentService » de chercher les données de localisation via le Google Map Service. A la réception de localisation de retour, le « ShoppingSolverIntentService » met en place la notification avec une mise en page organisée. Quand l'utilisateur voit la notification et le clique, le « MapsShopsActivity » se lance et cherche l'itinéraire à partir de la localisation courant de l'utilisateur jusqu'à la localisation de la boutique à l'aide de Google Map Service. Finalement, le « MapsShopsActivity » affiche l'itinéraire au utilisateur.

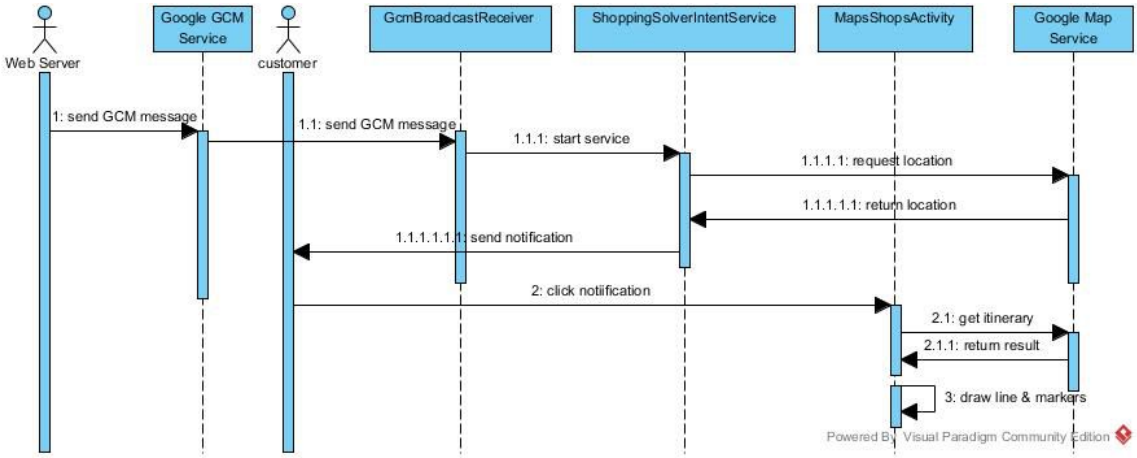


Figure 7: Diagramme de séquence à la réception du message de notification d'une promotion

E. Les difficultés rencontrées et les décisions prises

Le plus grand défi au cours du développement de notre projet est la définition de l'architecture globale de notre projet. Comme on a besoin les différents composants (l'application Android, le Service Web, la base de données, les services de Google), il est important de faire beaucoup de cherches sur les différents techniques pour chaque de ces composant rapport leurs points forts et faibles afin de trouver un plan global pour mettre en place les différents morceau de notre système et faire une décision sur le choix de protocole de communication. Ce plan doit être établi en

faisant la négociation entre les différents technologies disponible. Pour faire les choix correcte, on a faire beaucoup de recherche documentation avant passer à l'étape de conception logicielle. En analysant les différentes technologies possibles qui pourraient répondre nos besoins fonctionnels, on a finalement trouvé une solution d'architecture décrit dans II.C. Dès que une décision est pris, la conception des logiciels devient plus évidant.

III. Les tests d'exécution

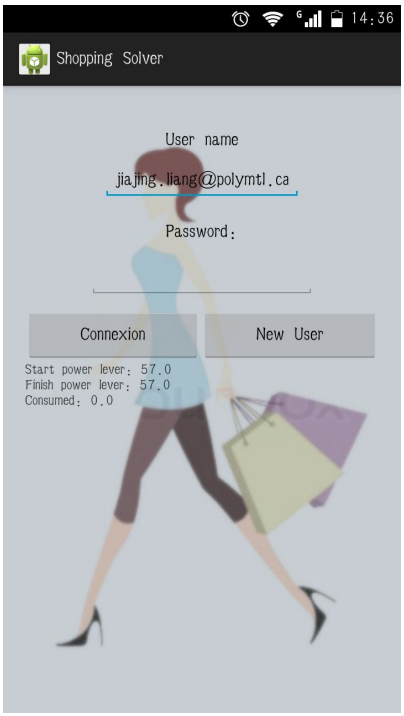
Voici dessous les captures d'écran qui montrent les résultats d'exécution de notre logiciel. La figure *8(a)* montre le lancement de l'application. Si l'on clique sur le bouton 'New User', on sera amène sur l'écran comme la figure *8(b)* pour saisir les informations afin de créer une nouveau compte. Quand le bouton 'Save your information' est cliqué, un compte sera créé, on login automatiquement et on arrivera sur l'écran comme la figure *8(c)*. Sur l'écran de la figure *8(a)*, si l'on entre le mot de passe correct et clique sur le bouton 'Connexion', on arrivera sur l'écran de la figure *8(c)*. Sinon, on s'est bloqué sur l'écran de *8(a)*.

La figure *8(c)* est l'interface pour la construction d'un liste d'achat. Si l'on clique sur le bouton avec l'icône d'un code barre, on sera amène sur l'écran comme la figure *8(d)* pour scanner le code barre d'un produit. Dès que l'information sur le code barre est capturé, on revient sur l'écran pour la construction d'une liste avec le produit ajouté sur la liste comme dans la figure *8(f)*. Si l'on clique sur un élément dans la liste, un écran comme la figure 8(e) s'affiche pour modifier la quantité d'un produit et consulter ses détails (prix, taxes, etc). Quand on clique sur le bouton 'Save', la modification de la quantité se sauvegarde et on revient sur l'écran de la liste d'achat. Si l'on clique sur l'icône de poubelle dans la figure *8(e)* ou *8(f)*, le produit correspondant sera supprimé dans la liste. Dès qu'on finit l'achat et clique sur le bouton 'Buy Now'(figure *8(f)*), une facture comme la figure *8(g)* s'affichera et on a finit un achat.

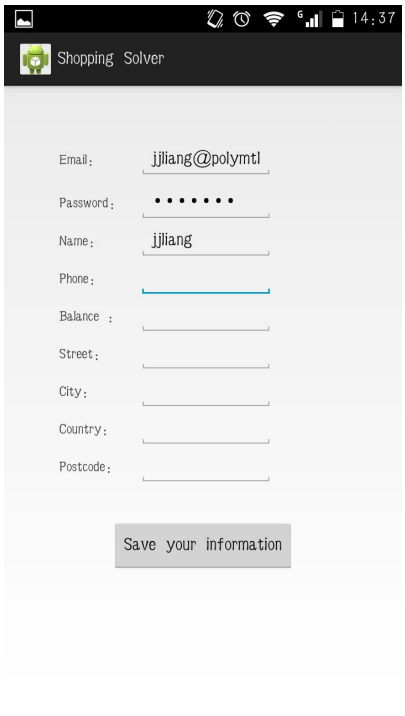
Si l'on clique sur l'icône de navigation (en haut à gauche de l'interface avec 3 barres horizontales), une liste de navigation comme la figure *8(h)* se sort de gauche. Ici on propose 3 autres fonctionnalités: consulter les informations du compte, consulter l'habitude de l'achat et consulter les historiques des transaction. Si l'on choisit 'Count Detail' dans la liste de navigation, un écran comme la figure *8(i)* montrera les détails du compte. Si l'on choisit 'Consumption Habit', un écran comme la figure *8(j)* montrera les produits achetés le plus fréquemment. Quand on clique sur 'Your Receptions' dans la liste de navigation, on verra les 10 dernières transactions faites comme la figure *8(k)*.

On a fait un petit site Web (admin_ui décrit dans le diagramme de déploiement de la figure 2) pour administrer les prix des produits dans les boutiques. Si l'on baisse le prix d'un produit dans un boutique (figure 9 au dessous), une notification arrivera sur le mobile d'un client qui l'achète souvent pour le signaler qu'il y a une promotion comme montré dans la figure *8(l)*. Si l'on clique sur la notification, on arrivera sur l'écran de la figure *8(m)* qui affiche la localisation de la boutique par rapport la localisation courante de l'utilisateur avec une itinéraire calculée sur la carte. Si l'on clique sur l'icône de la boutique, un détail (l'adresse de la boutique, le produit en promotion et le prix) s'affichera comme dans la figure *8(n)*. Quand l'on sortir l'application en cliquant sur le bouton de retour, on revient d'abord sur l'écran de login et la consommation d'énergie de toutes ces opérations affichera en bas. En cliquant encore une fois le bouton de retour, on quitte l'application.

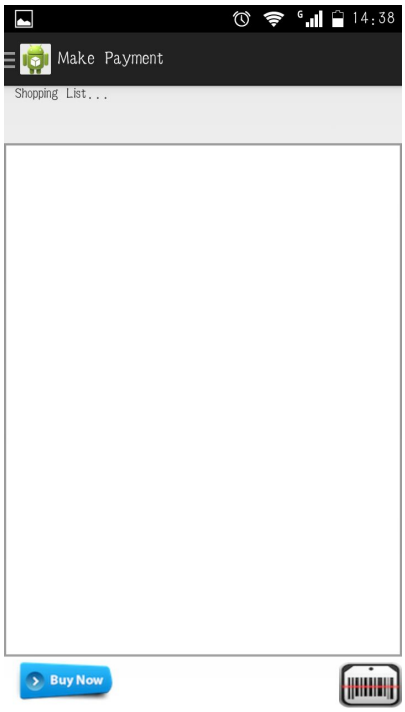
Dans le cas d'une interruption de réseau, on envoie aucune requête SOAP. Du coup, aucun transaction se passera pour un achat jusqu'à la rétablissement du réseau.


14:36
Shopping Solver
User name
jiajing.liang@polymtl.ca
Password:
Connexion
New User
Start power lever: 57.0
Finish power lever: 57.0
Consumed: 0.0

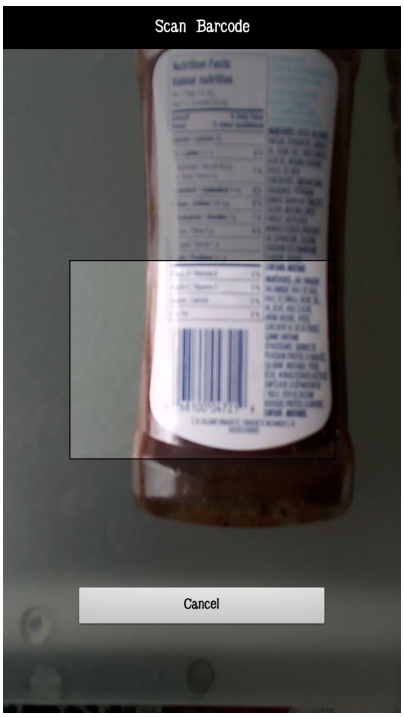
(a)


14:37
Shopping Solver
Email: jjliang@polymtl
Password: •••••••
Name: jjliang
Phone:
Balance :
Street:
City:
Country:
Postcode:
Save your information

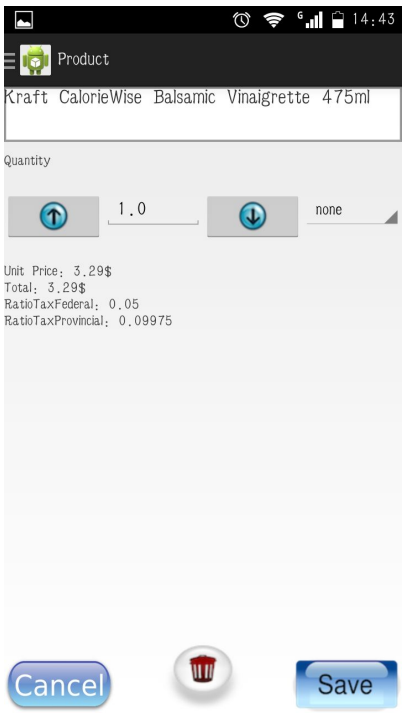
(b)


14:38
Make Payment
Shopping List...
Buy Now

(c)


Scan Barcode
Cancel

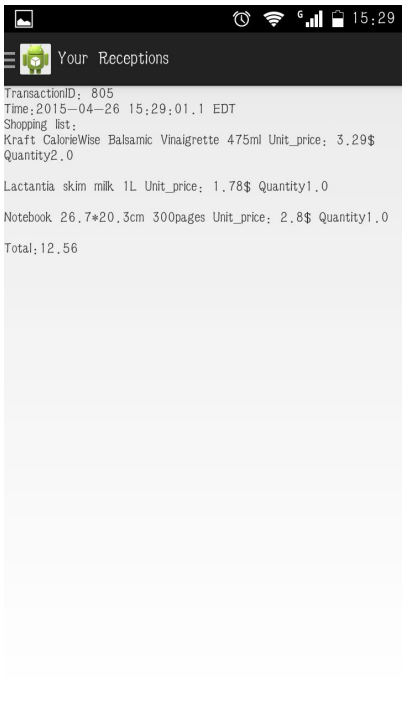
(d)


14:43
Product
Kraft CalorieWise Balsamic Vinaigrette 475ml
Quantity
1.0
none
Unit Price: 3.29$
Total: 3.29$
RatioTaxFederal: 0.05
RatioTaxProvincial: 0.09975
Cancel
Save

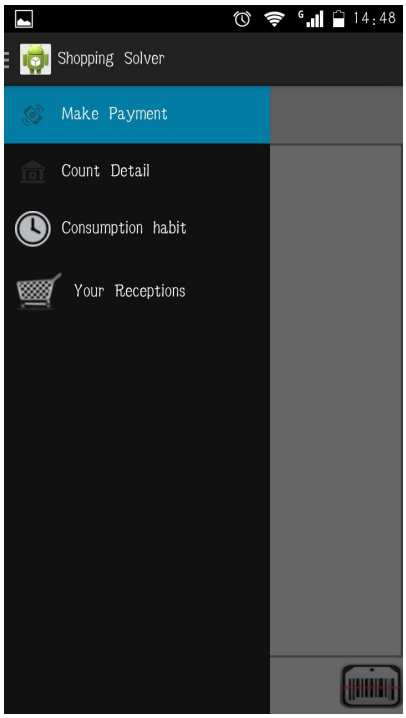
(e)


14:44
Product
Shopping List...
Kraft CalorieWise Balsamic Vinaigrette 475ml
Unit price: 3.29$ quantity: 2.0
Lactantia skim milk 1L
Unit price: 2.19$ quantity: 1.0
Notebook 26.7*20.3cm 300pages
Unit price: 2.6$ quantity: 1.0
Buy Now

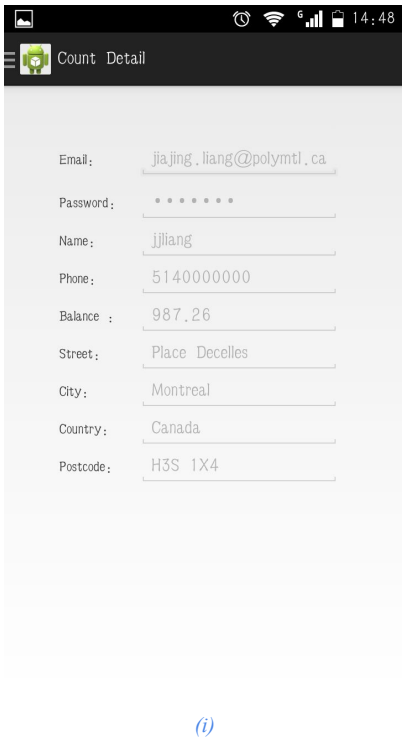
(f)


15:29
Your Receptions
TransactionID: 805
Time:2015-04-26 15:29:01.1 EDT
Shopping list:
Kraft CalorieWise Balsamic Vinaigrette 475ml Unit_price: 3.29$ Quantity2.0
Lactantia skim milk 1L Unit_price: 1.78$ Quantity1.0
Notebook 26.7*20.3cm 300pages Unit_price: 2.8$ Quantity1.0
Total:12.56

(g)


14:48
Shopping Solver
Make Payment
Count Detail
Consumption habit
Your Receptions

(h)


14:48
Count Detail
Email: jiajing.liang@polymtl.ca
Password: ••••••••
Name: jjliang
Phone: 5140000000
Balance : 987.26
Street: Place Decelles
City: Montreal
Country: Canada
Postcode: H3S 1X4

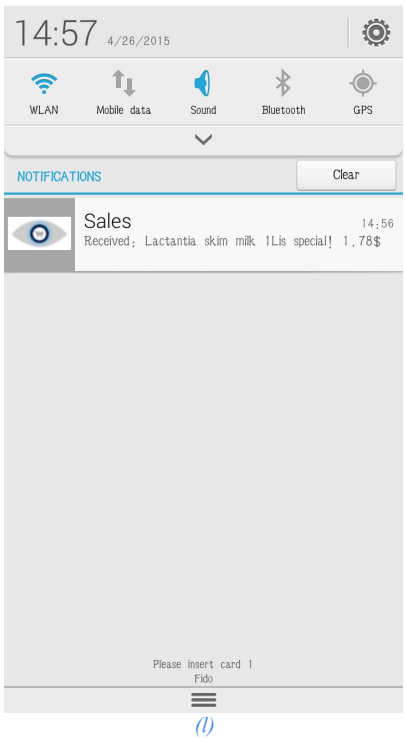
(i)


14:48
Consumption Habit
Your consumption habit...
Kraft CalorieWise Balsamic Vinaigrette 475ml,
Barcode: 068100047219
Quantity: 2.0
Lactantia skim milk 1L,
Barcode: 068200010311
Quantity: 1.0
Notebook 26.7*20.3cm 300pages,
Barcode: 612668062353
Quantity: 1.0

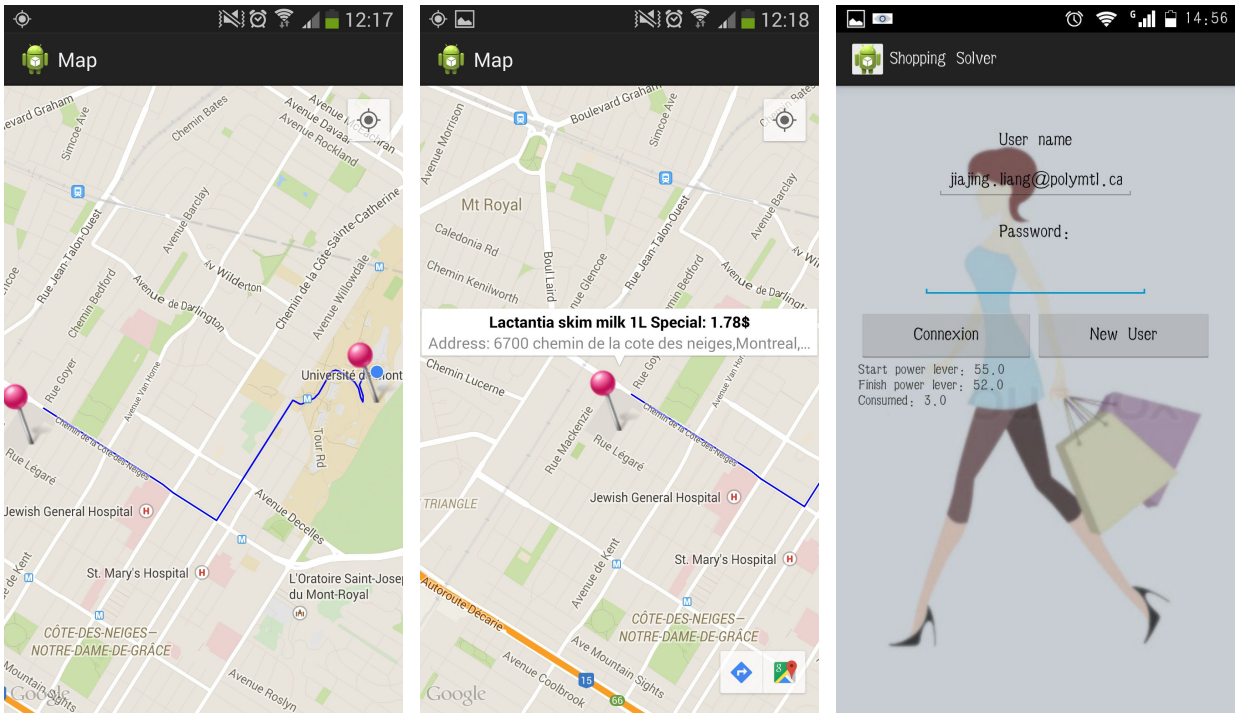
(j)


15:25
Your Receptions
Transaction Id: 706
Total: 3.45
Time: Fri Apr 24 12:10:00 EDT 2015
Shop name: Walmart
Address: 6700 chemin de la cote des neiges
Montreal H3S 2B2 Canada
Transaction Id: 703
Total: 3.78
Time: Tue Apr 21 01:14:00 EDT 2015
Shop name: Walmart
Address: 6700 chemin de la cote des neiges
Montreal H3S 2B2 Canada
Transaction Id: 702
Total: 3.78
Time: Mon Apr 20 23:55:00 EDT 2015
Shop name: Walmart
Address: 6700 chemin de la cote des neiges
Montreal H3S 2B2 Canada
Transaction Id: 701
Total: 3.78
Time: Mon Apr 20 23:53:00 EDT 2015
Shop name: Walmart
Address: 6700 chemin de la cote des neiges
Montreal H3S 2B2 Canada
Transaction Id: 551
Total: 31.26
Time: Sat Apr 18 17:26:00 EDT 2015

(k)


14:57 4/26/2015
WLAN
Mobile data
Sound
Bluetooth
GPS
NOTIFICATIONS
Clear
Sales
14:56
Received: Lactantia skim milk 1Lis special! 1.78$
Please insert card 1
Fido

(l)



(m) *(n)* *(o)*

Figure 8 : Les captures d'écran de l'application.

068200010311	Lactantia skim milk 1L	MILK	2.19	0.0	0.0	1.78 Change Price

Figure 9: La capture d'interface de site Web pour change le prix d'un produit d'une boutique

IV. La conclusion et l'amélioration possible

A. La résumé de l'expérience

L'expérience acquise au moment de l'implémentation du projet nous donne l'accès de travail dans un environnement Android et de savoir vérifier que l'application implémenté est compatible aux autre plates-formes de différent appareil mobile sans oublier de concevoir l'architecture d'une application mobile ; de développer une application mobile ; d'organiser l'information dans un contexte de mobilité enfin d'assurer la qualité de l'application. D'ailleurs, cette expérience du projet nous offre l'opportunité d'établir la communication entre l'application Android avec le serveur en tant que un client avec le protocole SOAP et de traiter les requêtes HTTP qui vient de Google GCM Service pour la mise en place des notification de promotion. Ceci est une expérience de développement assez enrichissante.

B. L'amélioration possible

Comme dans le projet, on travail dans le domaine de paiement électronique qui est un sujet ayant des fortes contraintes sur la sécurité des données des transaction. Du coup, une amélioration possible pour le future travail est d'encoder les données concernant les informations privées d'un client et les données des transactions à côté de client Android avant d'envoyer les requêtes ou les décoder après la réception d'une réponse du service Web. Il faut faire les encodages et les décodages de même façon à côté de service Web.

C. Les critiques et des suggestions pour l'amélioration du contenu

Notre équipe définit le contenu du projet dans le domaine de paiement électronique puisque c'est un sujet très à la mode et souvent parlé dans le domaine TI. On pense que notre contenu est assez riche et intéressant. Alors les améliorations possible pour le contenu est d'ajouter plus de fonctionnalité à notre application en intégrant les technologies de paiement plus récentes.