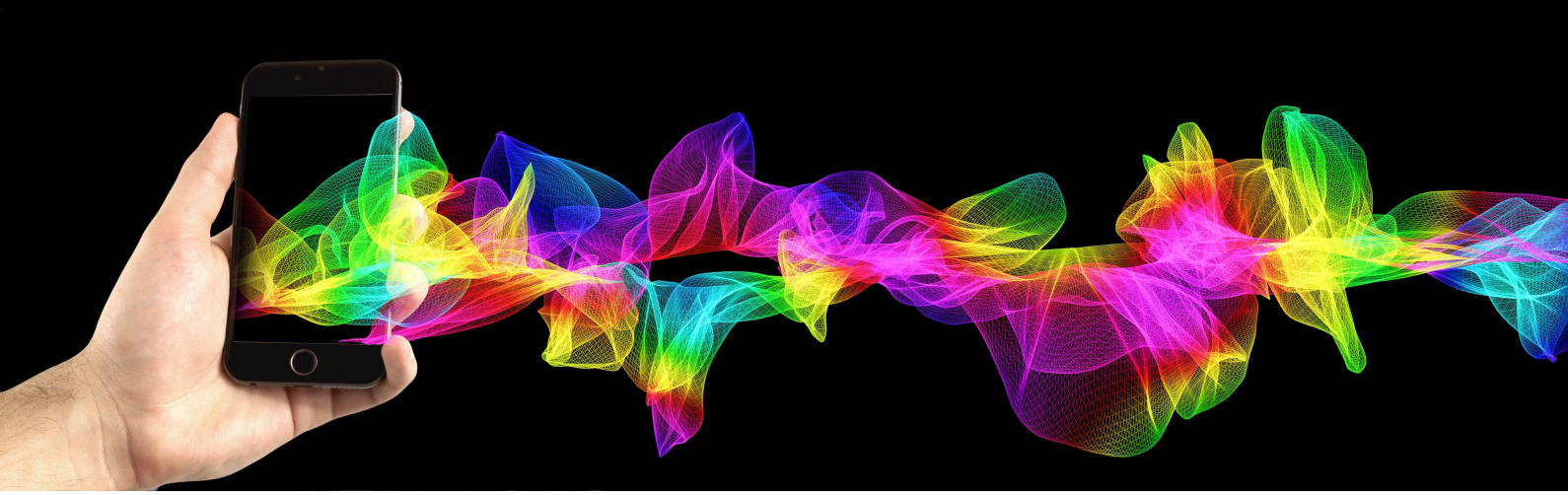

UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Développement d'applications mobile sous iOS et Android

Rapport de stage ST40 - A2019

Koffi Moïse Agbenya

Département Informatique

LUXEMBOURG ONLINE S.A.

14 Avenue du X Septembre
L-2550 Luxembourg
www.internet.lu

Tuteur en entreprise
RETTER Paul

Suiveur UTBM
Oumaya Baala Canalda

utbm
université de technologie
Belfort-Montbéliard

Ce document décrit le projet Rapport de stage ST40 - A2019.

TEX et LATEX sont des marques de la Société Américaine de Mathématiques.

`tex-upmethodology` est la propriété de Stéphane Galland, *Arakhnê*.ORG, France.

Ce document a été réalisé avec LATEX et `tex-upmethodology`.

Référence : -

Synoptique	
Projet	Rapport de stage ST40 - A2019
Document	Développement d'applications mobile sous iOS et Android
Référence	-
Version	1.0
Dernière modification	12/02/2020

Auteurs		
Noms	*Commentaires*	*Emails*
Koffi Moïse Agbenya	Étudiant en branche INFO	koffi.agbenya@utbm.fr

Validateurs		
Noms	*Commentaires*	*Emails*
Oumaya Baala Canalda	Suiveur UTBM	oumaya.baala@utbm.fr
Paul Retter	Tuteur en entreprise	emploi@online.lu

Pour information		
Noms	*Commentaires*	*Emails*
Oumaya Baala Canalda	Suiveur UTBM	oumaya.baala@utbm.fr
Paul Retter	Tuteur en entreprise	emploi@online.lu
Celine Beri	Responsable administratif (RH)	celine.beri@silis.lu

Remerciements

Tout d'abord, je tiens à remercier mon maître de stage **Paul RETTER**, administrateur délégué de la société Luxembourg Online, de m'avoir accueilli dans la société, de m'avoir encadré et de m'avoir permis de travailler sur des sujets intéressants et formateurs. Je le remercie aussi pour ses conseils ainsi que la confiance qu'il m'a accordé dans le choix de mes idées pendant l'implémentation des solutions tout au long de mon stage.

Je tiens à remercier **Ghislain ANCIAUX**, ingénieur informatique, avec qui j'ai travaillé sur les applications, de m'avoir rendu la tâche facile en étant rapide dans ses cycles de développement et surtout de m'avoir pris sous ses ailes.

Je tiens à remercier **Louis RETTER**, ingénieur informatique, pour nos différentes discussions lors de la conception des architectures logicelles, ce qui m'a été d'une grande aide.

Ensuite je tiens à remercier **Céline BERI**, Gestionnaire ressources humaines, pour m'avoir aidé dans toutes mes tâches administratives.

Je tiens à remercier toutes les personnes avec lesquelles j'ai passé mes 6 mois de stage pour les moments conviviaux que nous avons pu partager ensemble.

Jet tiens à remercier **Oumaya BAALA CANALDA**, mon suiveur UTBM, de s'être soucier de mon statge.

Et enfin je remercie **Christopher STANIC-PAGANO**, étudiant à l'UTBM pour m'avoir présenter l'entreprise et fourni mon CV à mon maître de stage.

Table des matières

Table des figures

Introduction

Selon une étude de Statista sur l'utilisation du téléphone mobile dans le monde, le nombre d'utilisateur de téléphone mobile dans le monde devrait dépasser la barre des cinq milliards en 2019 et d'ici 2020, le nombre d'utilisateurs de smartphone devrait atteindre 2,87 milliards d'individus. Selon une autre étude réalisée par « Internetworldstats », au 30 juin 2019, le nombre d'utilisateurs d'internet dans le monde a atteint les 4.536.248.808 soit 58% de la population mondiale dont environ 3,986 milliards[1] sont des connexions réalisées à partir des téléphones mobiles.

De ces études on peut en déduire clairement qu'il est primordial pour une entreprise aujourd'hui de proposer ses services internet pour les appareils mobiles qui représentent 88% de l'accès à internet.

De nos jours, le développement pour mobile a pris beaucoup d'ampleur et de plus en plus d'entreprises l'ont adopté pour leurs produits car c'est un moyen de créer des services innovants, d'améliorer la communication et d'augmenter leur productivité.

Du 02 septembre 2019 au 07 février 2020 (5 mois 5 jours), j'ai effectué un stage assistant ingénieur au sein de l'entreprise **Luxembourg Online SA** (située à Luxembourg). Au cours de ce stage dans le département informatique, j'ai pu mettre mes compétences de développeurs logiciels pour développer plusieurs applications mobiles pour les systèmes d'exploitations Système d'exploitation mobile basé sur le noyau linux et développé par Google (Android) et iPhone Operating System (iOS).

Luxembourg Online est l'un des principaux opérateurs luxembourgeois de télécommunications. La société est spécialisée dans la fourniture d'accès internet, la téléphonie fixe, mobile, la télévision, le développement de réseaux et d'applications informatiques.

Le service de l'entreprise qui m'a accueillit pour mon stage est le service informatique qui est dirigé par mon maître de stage M. Paul Retter. Mon stage a consisté essentiellement en le développement de plusieurs applications mobiles pour les platformes Android[1] et iOS.

Plus largement, ce stage a été l'opportunité pour moi non seulement d'approfondir mes connaissances dans le développement pour Android et de travailler sur une application grand public mais aussi d'apprendre à développer des applications pour la platforme iOS d'apple et d'approfondir mes connaissances dans le domaine.

Au delà d'enrichir mes connaissances en développement logiciel, ce stage m'a permis de comprendre certains aspect du développement notamment la programmation réactive, l'architecture logicielle, et de le mettre en pratique.

Dans l'optique de rendre compte de manière fidèle des 5 mois passés au sein de la société Luxembourg Online, il apparaît logique de présenter à titre préalable de l'état actuel des solutions internet et mobile de l'entreprise, ensuite envisager le cadre du stage : la culture d'entreprise dans la société Luxembourg Online et son apport dans la méthode de travail et la productivité et enfin préciser les différentes missions et tâches que j'ai pu efectuer au sein du service informatique, et les nombreux apports que j'ai pu en tirer.

[1] Etude réalisée par Hootsuite

1

Présententation de l'entreprise

1.1/ Naissance et évolution

La société Luxembourg Online est une société anonyme (SA) fondée en 1995 et elle est implentée uniquement sur le territoire luxembourgeois. La société est l'un des principaux opérateurs luxembourgeois de télécommunications et est spécialisée dans la fourniture d'accès internet, la téléphonie fixe, mobile, la télévision, le développement de réseaux et d'applications informatiques. Depuis sa création en 1995 la société a fait de manière récurrente à intervalle de plus ou moins 2 ans des lancements majeures de services. En 1997, elle lance sa plateforme de commerce électronique, deux ans plus tard en 1999 elle lance un accès internet gratuit à une échelle nationale. En 2001 il y a eu le lancement des forfaits internet sous forme de packages, en 2003, elle lance l'accès à internet par le câble de télévision et de l'accès internet haut-débit. En 2004 elle lance le service de préselection téléphonique Luxembourg Online pour téléphoner moins cher. En 2005, elle lance le service de téléphonie via internet et deux ans plus tard le service de téléphonie mobile. En 2011, elle lance un service de TV qui est une solution de télévision par Internet Protocol (IP) et deux ans plus tard elle lance le dégroupage en fibre optique. Enfin en 2016 elle lance le service de visiophonie.

Figure 1 – Logo de la société Luxembourg Online

1.2/ Effectif, organisation et services de la société

La société Luxembourg Online dispose d'un effectif de plus de 100 collaborateurs, répartis sur 3 sites dans le pays. Le siège de la société est situé dans la ville de Luxembourg le lieu de mon stage. Dans ces bureaux sont installés : le service informatique dans lequel j'ai travaillé, le service administratif et le service comptabilité. Les deux autres sites sont situés respectivement dans la ville de Luxembourg qui sert de boutique oú est intallé le service client et les vendeurs et dans la ville de Bertrange qui est un lieu de stockage de tous les matériels de télécommunications (équipements fibre optique, box internet, câble de raccordements, et les décodeurs TV).

La société dispose de son propre réseau internet sur le territoire Luxembourgeois et qui couvre presque la totalité du pays. Elle dispose aussi de son propre réseau de téléphonie fixe par IP et de télévision par IP. Cette autonomie permet à la société d'être totalement libre sur le développement, la commercialisation et le suivi de l'ensemble de ses services et produits. La société propose à ses clients plusieurs offres internet qui va de la connexion bas débit Digital Subscriber Line (DSL) à

une connexion très haut-débit par fibre optique. Les-dits clients peuvent de manière optionnelle souscrire au service de télévision, de téléphonie ou encore de stockage cloud.

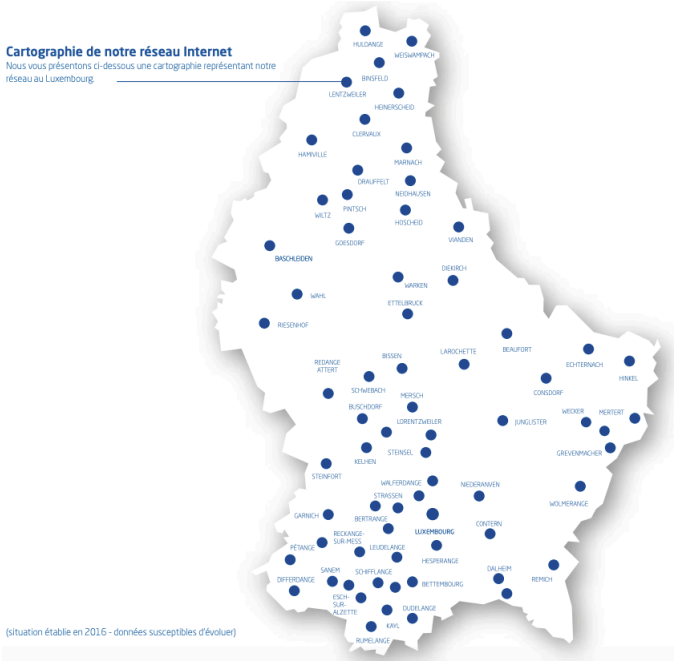

FIGURE 2 – Cartographie du réseau de fibre optique couverte par la société

Outre son activité de Fournisseur d'accès Internet (FAI), Luxembourg Online est un opérateur de téléphonie mobile et une société de service informatique qui développe des solutions informatiques pour le grand public et pour toute sorte d'entreprise.

1.3/ OUVERTURE SUR MON TRAVAIL AU COURS DU STAGE

Le service qui m'a accueilli pendant mon stage est le service informatique. J'ai travaillé dans un premier temps seul sur un premier projet et je faisais des rapports à M. **Paul RETTER** qui est mon tuteur en entreprise et j'ai ensuite travaillé sur un deuxième projet sous la direction toujours de mon tuteur mais cette fois-ci en collaboration avec d'autres ingénieurs de la société.

2

Organisation du stage

Travail réalisé

Le sujet défini avant le début de mon stage est intitulé « Développement d'applications sous Android et iOS ».

A mon arrivé dans l'entreprise j'ai eu une réunion avec mon tuteur de stage, réunion au cours de laquelle il m'a été expliqué concrètement le travail que j'effectuerai durant la période de mon stage.

L'objectif de mon stage est de développer deux applications respectivement nommé **Smart Home Viewer** en version Android puis iOS et **Resto**[1]

Le développement de l'application Smart Home Viewer avait été commencé par un stagiaire mais il n'a pas pu le terminé avant son départ. Mon travail a consisté donc à prendre en main le travail incomplet, de le terminer, de corriger les éventuels bogues après les tests et ensuite de commencer à développer l'application Resto.

Le suivi du développement est réalisé de manière hebdomadaire. Sur une semaine, je réalisais des cycles de développement court pendant lequel, je travaillais personnellement en Agile avec la méthode Kanban. A la fin de la semaine, je remplissais une fiche des travaux réalisés que je transmettais via la Gestionnaire des Ressources Humaines à mon tuteur. Je sortais aussi par la même occasion une *release* du projet que des testeurs externes à mon service mais interne à l'entreprise pouvait tester et faire des retours d'informations.

J'ai réussi à produire une *release* finale à la fin du mois de Septembre pour l'application Smart Home Viewer.

Au début du mois d'Octobre j'ai eu une nouvelle réunion majeure avec mon tuteur de stage et une équipe de développement composée de développeur Back-end, d'un administrateur de base de donnée et d'un développeur Android pour lancement du développement de l'application Resto.

Pour cette application, je me suis occupé du développement de la version iOS. L'application est composée d'une partie cliente et d'une partie Pro. Ma tâche est d'arrivée à produire une version Bêta à la fin du mois d'octobre et de faire ensuite l'application vers une version majeure vers la fin du mois de Novembre. Ensuite il sera question de travailler sur la version Pro jusqu'à la fin du stage.

Pour cette application, j'ai travaillé directement en collaboration avec le développeur Android, le développeur Backend et un designer User Interface (UI).

Dans la suite de ce document, je présenterai chacune des applications que j'ai développé, leurs spécificités, les difficultés que j'ai rencontré et les solutions que j'ai trouvé. Je ferai à la fin un bilan

[1] Nom de code de l'application. Le nom de marque n'étant pas encore choisi.

sur la totalité du travail

3

Smart Home Viewer

Smart Home Viewer est une application de télémaintenance permettant de faire le diagnostique de la connexion internet d'un modem internet.

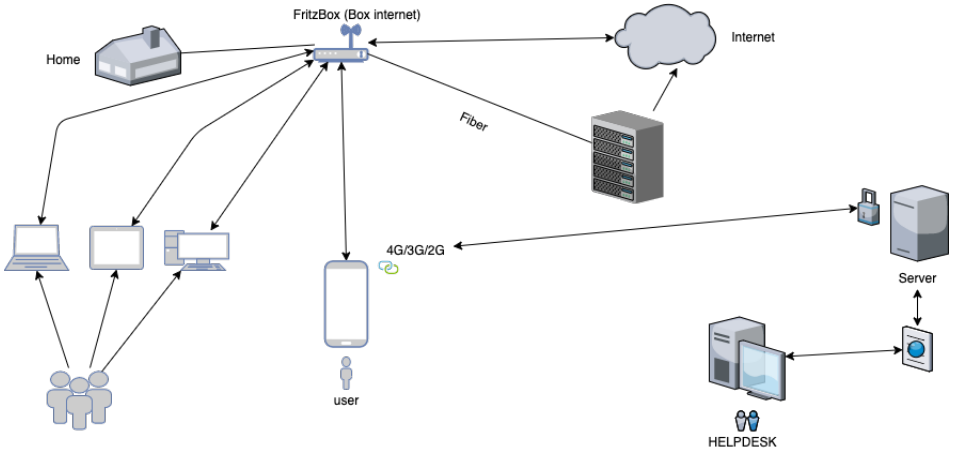

Figure 3 – Description global du système de fonctionnement de l'application

Le contexte de fonctionnement de l'application est lorsque pour une raison ou une autre, un utilisateur de la box de l'entreprise voit sa connexion internet s'interrompre. Dans ce cas, il appelle le centre d'appel de l'entreprise et l'operateur lui fournit un code de connexion pour le diagnostique de sa connexion.

Le processus de diagnostique de la connexion est décrit sur la figure 3. Pour faire fonctionner l'application, l'utilisateur devra activer son Wifi et ses données mobiles. La liaison Wifi servira à la communication avec la box et les données mobiles serviront à envoyer les données de la box au serveur.

Lorsque l'utilisateur s'authentifie avec le code que l'opérateur lui a fourni, l'opérateur, le téléphone est prêt pour servir de canal de transmission de l'information entre la box et le serveur. L'opérateur déclenche la communication entre les appareils puis le serveur envoie les requêtes nécessaires à la box pour récupérer et afficher la page d'accueil de la box directement sur le poste de l'opérateur qui pourra vérifier les paramètres et faire des modifications si nécessaire.

Dans la suite, il sera décrit le fonctionnement du système.

3.1/ Contexte du système

Dans cette section je présente le diagramme de contexte du système afin de localiser l'application dans son environnement. Il est aussi décrit les acteurs qui interagissent avec le système. Dans la suite de ce chapitre, *Modem* fait référence à la box.

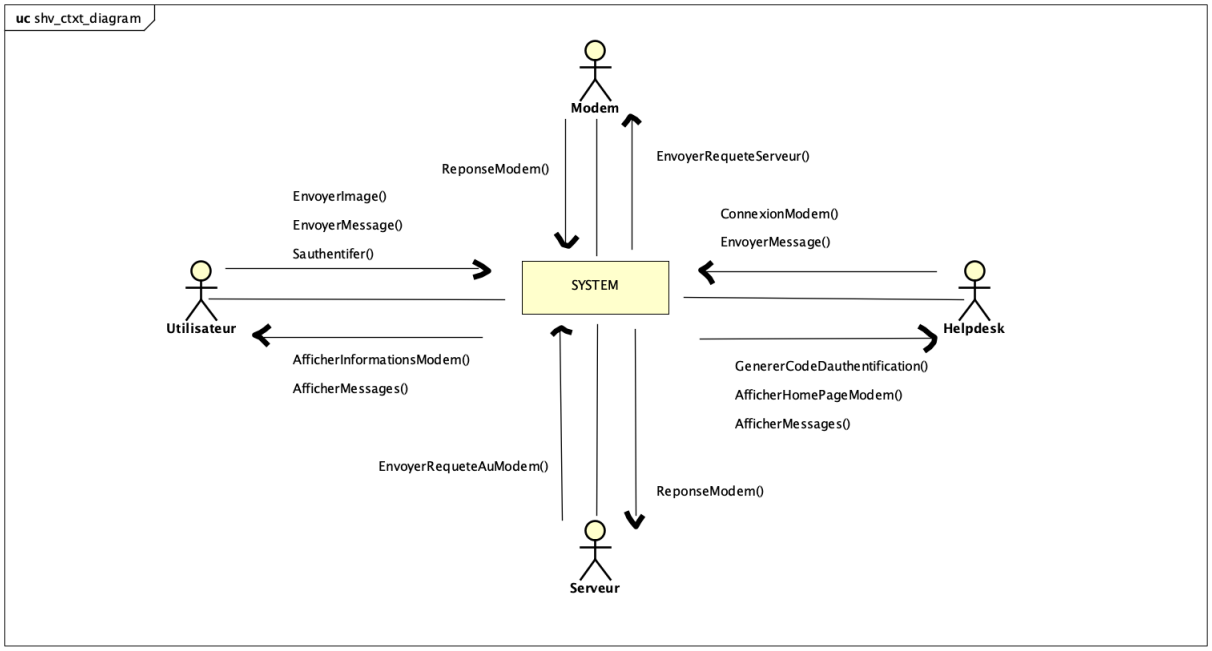

Figure 4 – Diagramme de contexte

L'utilisateur et le *helpdesk* sont les acteurs principaux du système. Afin d'initier la communication, le *helpdesk* doit fournir le code d'authentification à l'utilisateur. Le code d'authentification est généré par le système côté serveur et fourni au *helpdesk* . *GenererCodeDauthentification()* consiste à générer un code à six chiffres, l'associer à une session et à l'afficher au *helpdesk* . L'*utilisateur* se connecte au système via ce code. Il réalise cette connexion via *Sauthentifier()*. Une fois authentifier, le système côté client effectue quelques requêtes afin d'afficher à l'utilisateur, les informations de son Modem. Les informations sont affichées via *AfficherInformationsModem()*. Le *helpdesk* et l'*utilisateur* peuvent s'échanger via un chat et les fonctions principales de ce chat est l'envoie des messages et d'images. Ces fonctions sont réalisées via *EnvoyerMessage()*, *EnvoyerImage()* et *AfficherMessages()*.

Une fois la connexion réalisée par l'*utilisateur*, le *helpdesk* , peut déclencher l'affichage de la page d'accueil du modem. Si la requête est fait, le serveur envoie et reçoit des requêtes au système via les fonctions *EnvoyerRequeteAuModem()* et *ReponseModem()*. Le système récupère la requête du serveur, construit une nouvelle requête spécifique et l'envoie au modem qui répond au système en retour en fonction des éléments demandés. Les fonctions misent à contributions sont *EnvoyerRequeteServeur()* et *ReponseModem()*. Si tout se passe bien, le serveur construit à partir des éléments fournis par le modem la page d'accueil du modem et l'affiche au *helpdesk* .

Du point de vue du fonctionnement du système, les acteurs *Modem* et *Serveur* sont certes secondaires mais indispensable. C'est la connexion internet du modem qui est diagnostiquée. Le serveur a pour rôle de servir d'interface pour le *helpdesk* et de serveur pour l'application du point de vue de l'architecture client-serveur pour une application. Le système sert donc d'intermédiaire ou de moyen de communication entre le *modem* et le *serveur*.

3.2/ Caractéristique de l'utilisateur

L'application est destinée à l'utilisation des clients de la société Luxembourg Online ayant souscris à une offre internet via box ou Asymmetric Digital Subscriber Line (ADSL). L'utilisation de l'application ne requiert aucune compétence particulière à part le fait de savoir utiliser un smartphone.

3.3/ Contraintes principaux de développements

L'application a été développé pour les deux plateformes mobiles Android et iOS et selon le paradigme orienté objet. Pour ce faire j'ai utilisé le langage de programmation Java est un langage de programmation objet créé par Sun microsystems et détenu depuis 2009 par Oracle. (Java), l'environnement de développement d'application Android et l'Integrated Development Editor (IDE) Android Studio pour l'application Android et le les langages Swift est un langage de programmation objet, multiparadigme, open-source développé par Apple en 2014 (Swift) et Objective C est un langage orienté objet réflexif créé en 1983 et détenu par Apple (Objective C) avec l'environnement de développement d'application Cocoa est une API native d'Apple pour le développement orienté objet sur son système d'exploitation Mac OS X (Cocoa) Touch et l'IDE XCode pour l'application iOS.

J'ai aussi utilisé des standards Requests for comments (RFC) pour des questions réseaux.

3.4/ Besoins fonctionnels

Il s'agit ici de décrire les besoins fonctionnels du système à travers les cas d'utilisation de l'application.

Description générale

Le diagramme de la figure 4 présente une vue générale des cas d'utilisations du système. Chaque cas d'utilisation lié à un acteur, décrit un état atteint par le système au cours de l'exécution de l'application. Au cours des prochaines sections, je détaillerai plus précisément les cas d'utilisation en les décomposant par acteur.

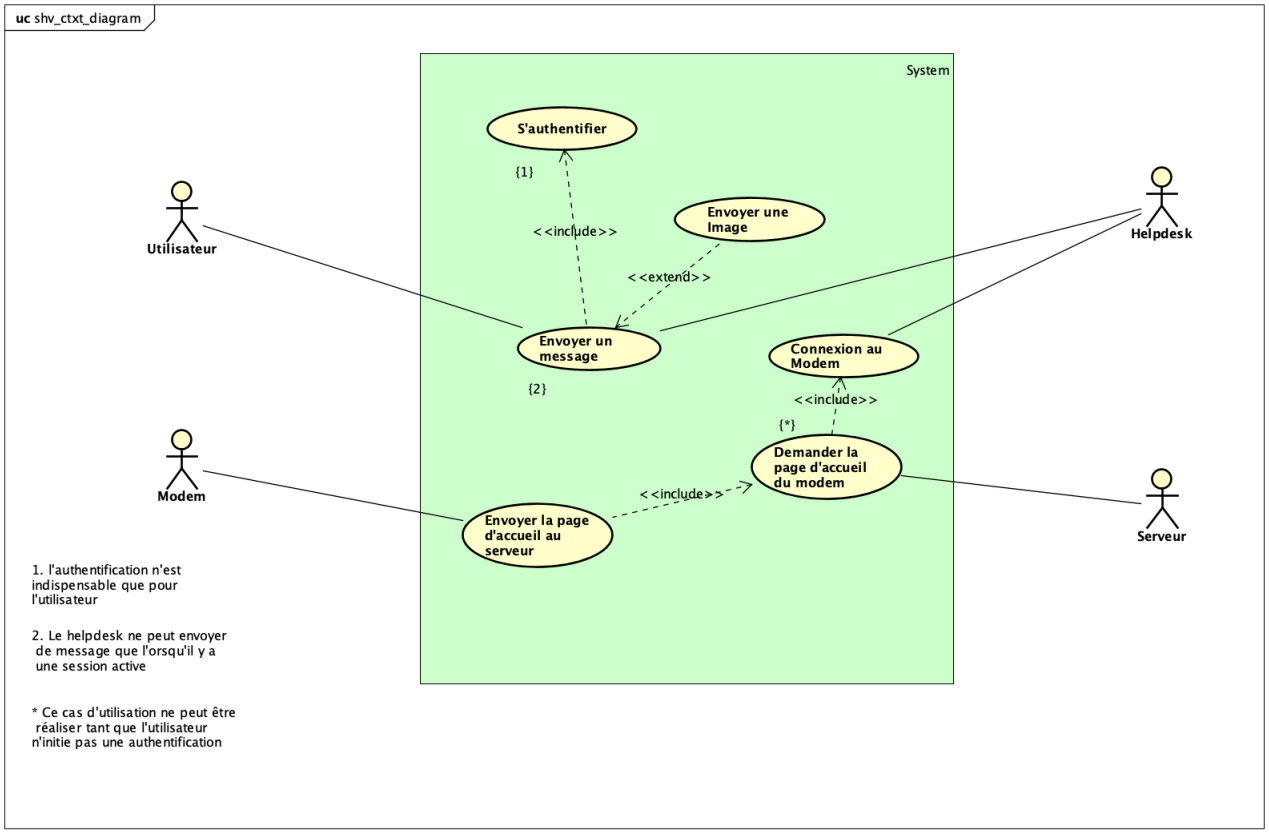

FIGURE 5 – Diagramme de cas d'utilisation

3.4.1/ DESCRIPTION DES CAS D'UTILISATIONS PAR ACTEUR

Le système est composé de quatre acteurs principaux à savoir l'*utilisateur* , le *helpdesk* , le *modem* et le *serveur* . Pour comprendre totalement le rôle joué par chacun, je procède ici à une brièvе description de chacun d'eux.

L'UTILISATEUR

Le principal rôle de l'*utilisateur* est de se créer une session en s'authentifiant sur l'application. Il pourra envoyer un message ou l'image de son appareil au *helpdesk* . Il pourra aussi lire les messages envoyés par le *helpdesk* .

LE HELPDESK

Les principaux rôles du *helpdesk* sont de générer le code d'authentification à l'*utilisateur* en démarrant la session et de se connecter au *modem* . Il pourra aussi fermer la session de l'*utilisateur* ou encore envoyer et lire les messages de ses échanges avec l'*utilisateur* .

LE MODEM

Le *modem* est l'appareil qui est diagnostiqué. Son rôle dans le fonctionnement du système est de recevoir, traiter et répondre aux requêtes envoyer par le serveur via le système.

Le Serveur

Les principaux rôles du *serveur* sont de créer et clore la session pour l'utilisateur, construire les requêtes à envoyer au *modem* et envoyer les messages.

3.4.2/ Description des cas d'utilisation

Description du cas d'utilisation de connexion à l'appareil

Résumé d'identification

Titre : Connexion au modem

Résumé : Initier la communication entre le serveur et le modem et début du diagnostique.

Acteurs : *helpdesk* (principal), *modem* , *serveur*

Date de création : 23 Septembre 2019

Date de mise à jour : 25 Septembre 2019

Version : 1.0

Description des scénarii

Pré-condition :

- L'*utilisateur* s'est authentifié et la session a été démarrée
- L'*utilisateur* a activé le wifi et les données mobile sur son téléphone et le wifi est connecté au modem à diagnostiquer.

Scénarii nominaux

1. Le *helpdesk* démarre la connexion au modem.
2. La connexion entre le *serveur* et le *modem* a été effectuée avec succès et la communication est déclenchée (A1)
3. La page d'accueil du modem est affichée dans un nouvel onglet dans le navigateur web du *helpdesk*

Scénario alternatif
A1 : Ce scénario est déclenché lors du point 2 des scénarii nominaux lorsque pour une raison la connexion est interrompue.

Le système réinitie la communication en trois tentative.

Scénario d'exception
E1 : Ce scénario est déclenché en A1 lorsque après trois tentatives de réinitialisation de la communication, la liaison est toujours cassée entre le *modem* et le *serveur* .

Le système déconnecte l'utilisateur et demande à nouveau le code d'authentification.

Post-condition :

- La page d'accueil du modem est affichée
- La session reste active

3.5/ Spécification des structures de données

Au cours de cette section, les diagrammes de classes ne concernent que la version Android de l'application mais les fonctionnalités sont les mêmes sur les deux versions de l'application.

3.5.1/ Module de communication

Pour réaliser la communication entre les différentes entités intervenant dans l'exécution de l'application, j'ai utilisé les **Socket** au niveau logiciel. Ainsi pour ce faire, j'ai appliqué le design pattern factory pour créer deux classes dans un contexte orienté objet à savoir *ModemSocket* et *ServeurSocket* afin de définir le comportement au niveau de l'application des interfaces de communication du *serveur* et du *modem* .

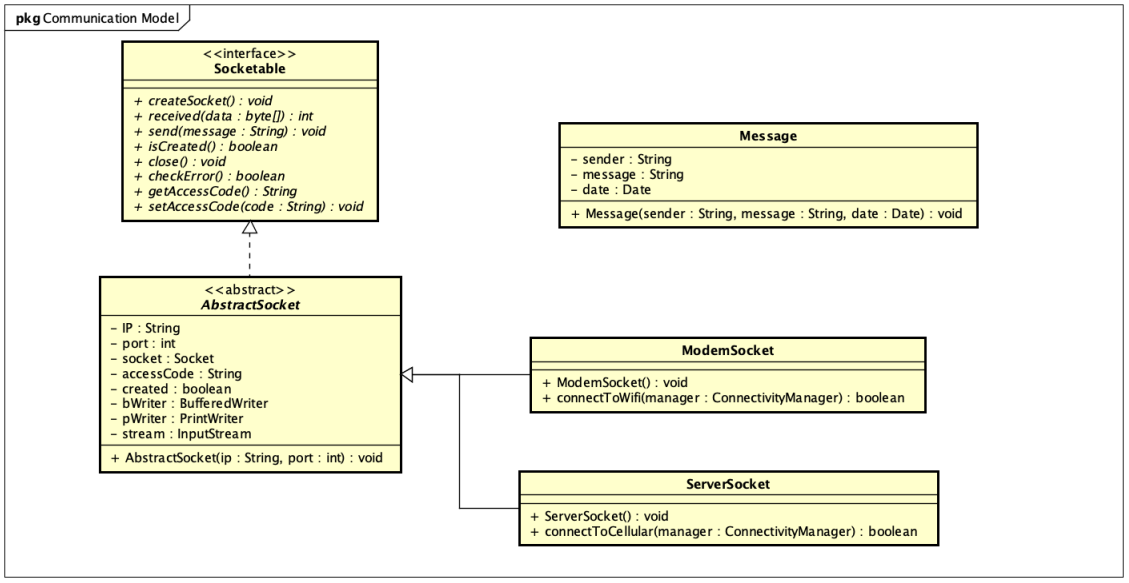

Figure 6 – Diagramme de classe du module de communication

Au lancement de l'application, un service au sens programmation, est démarré. Ce service possède les méthodes nécessaires pour authentifier et établir la connexion entre le *serveur* et le *modem* . Le diagramme de classe des services est présenté en annexe.

Dans la suite est présenté le diagramme d'activité représentant comment est géré la communication au niveau de l'application.

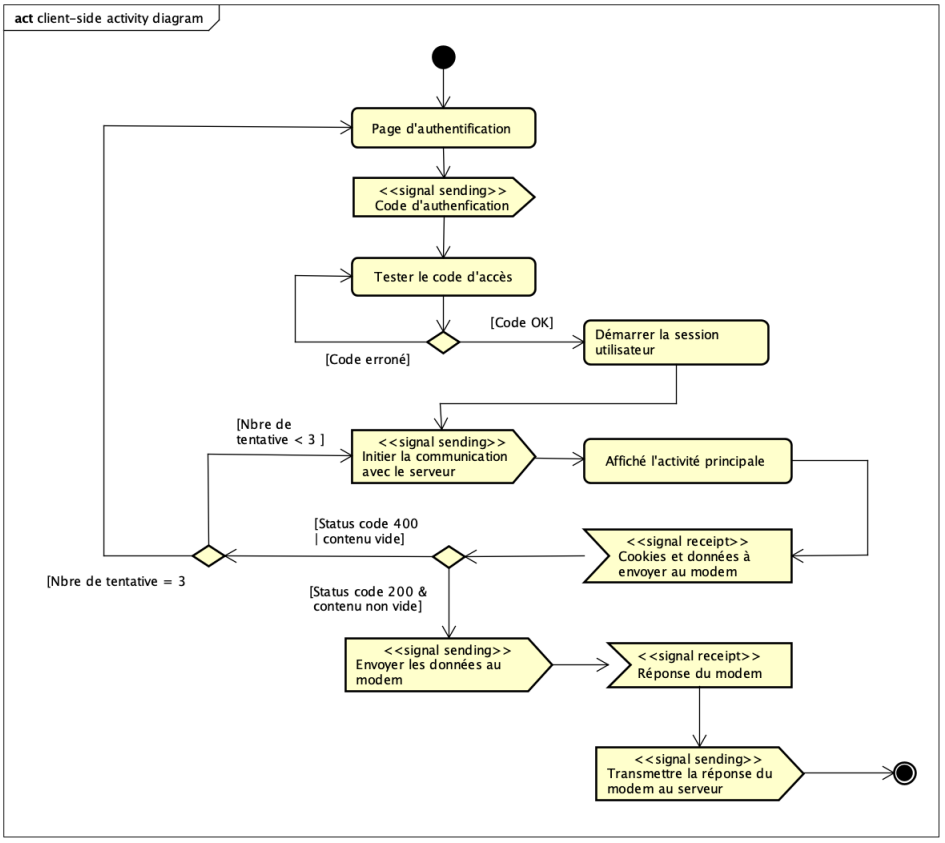

FIGURE 7 – Diagramme d'activité du module de communication

Comme décrit sur la figure 7, la communication est réalisée via le protocole Hypertext Transfer Protocol (HTTP). Ainsi une fois la session démarrée, le fonctionnement du système est lié à la bonne réception des données de la part du service de l'application et du serveur. Si le code de statut HTTP est 200, la communication est effective mais s'il est 400 le service réinitie la communication jusqu'à trois fois. Si après trois tentatives le service ne reçoit toujours pas un code 200, la communication est interrompu au niveau du programme et l'*utilisateur* est redirigé vers la page d'authentification.

3.6/ Spécification des interfaces externes

3.6.1/ Interface Matériel/logiciel

Configuration minimale sur laquelle le système peut s'exécuter

Pour faire fonctionner l'application sur un appareil mobile, l'appareil devra avoir les configurations minimales suivantes :

- Matériel : Tous les smartphones Android ou iOS supportant les configurations minimales au niveau logiciel.
- Logiciel : Tous les smartphones android supportant au moins la version 5.0 (Android Lollipop) ou les smartphones iOS supportant au moins la version 12.0 d'iOS.

Protocole d'échange

Pour le fonctionnement du système les protocoles utilisés sont le HTTP et un protocole développé par l'entreprise.

3.6.2/ Interface logiciel/logiciel

Pour développer l'application, les outils suivants ont été utilisés :

- lanagages et bibliothèques de développement
  - Java 1.8
  - Swift[2] 5.1
  - Objective C 2.0
  - Gradle est un moteur de production fonctionnant sur la plateforme Java. Il permet de construire des projets en Java, Scala, Groovy voire C++. (Gradle) 5.4.1
  - Android API 29
  - Cocoa Touch Framework
- Outils de développement
  - Android Studio 3.5
  - XCode 10.0
  - Git est un logiciel de gestion de versions décentralisé. (Git) 2.2
  - GitLab
  - Mac Mini 2019 et MacOS Mojave
  - Samsung Tab 4
  - iPad mini 2

3.6.3/ Interface homme-machine

Structure

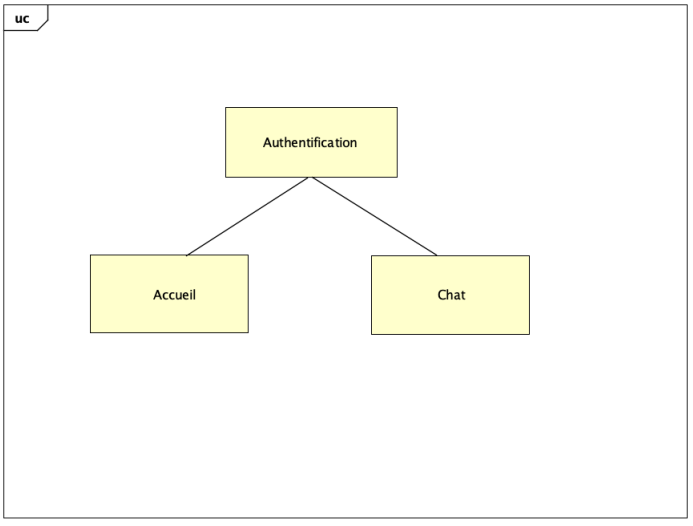

Figure 8 – Structure IHM de l'application

3.7/ Déroulement du stage

3.7.1/ Développement et difficultés rencontrés

Le développement de l'application Smart Home Viewer a été commencé en 2016 par un ancien stagiaire mais il n'a pas pu le terminé avant son départ. Lorsque j'ai repris le code, j'ai dû lire l'intégralité de son code et le compilé pour savoir exactement ce que je dois améliorer ou ajouter. Le code compilait mais l'application ne fonctionnait pas. J'ai d'abord refactoré le code, changé d'architecture et ajouté des fonctionnalités pour qu'elle fonctionne. Je dois dire que malgré le fait que l'application ne fonctionnait pas l'ancien stagiaire m'avait mâché le travail.

J'ai d'abord pris en main l'application Android et après l'avoir fait fonctionné totalement, j'ai réimplémenté les mêmes choses sur la version iOS.

Les premières difficultés sont apparus lors des premiers tests. En effet il arrivait que des fois la communication entre les appareils s'interrompaient inopinément ou encore après quelques minutes d'inactivité, la session se terminait tout seul.

Pour résoudre ces problèmes, j'ai mis en place un mécanisme basé sur le pattern Observer pour reconnecter automatiquement les appareils. Si après trois tentatives la reconnexion ne fonctionne pas, la session de l'*utilisateur* est déconnectée automatiquement.

Pour résoudre le problème qui survient à cause de l'inactivité, nous avons mis en place un protocole semblable a du ping pong pour garder la session active.

Le diagramme du modèle observateur que j'ai implémenté est disponible dans l'annexe.

3.7.2/ Planning daté du travail

Le diagramme de Gantt de la figure 9 présente ma charge de travail sur le développement de l'application Smart Home Viewer.

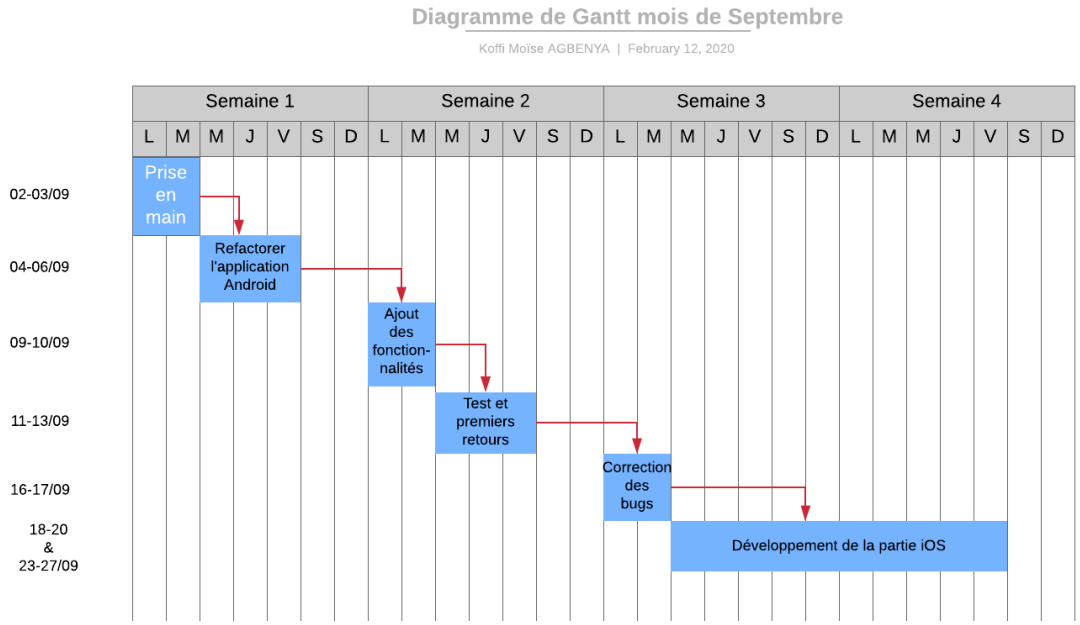

Figure 9 – Diagramme de Gantt du projet Smart Home Viewer

4

Application Resto

L'application Resto est une application destinée au grand public de consultation de restaurant à distance et de livraison de restaurant.

Les utilisations courantes de l'application sont :

- dans un restaurant comme cartes virtuelles en remplacement des cartes physiques ;
- réservation d'une table dans un restaurant ;
- commande depuis un autre lieu que le restaurant pour la livraison.

Pour trouver un restaurant sur l'application, l'utilisateur doit se géolocaliser et ensuite choisir le restaurant soit par les restaurants à proximité soit par les favoris soit par recherche via un moteur de recherche.

Un restaurant est organisé de la manière suivante : Un restaurant possède des cartes de plusieurs types et chaque carte peut avoir trois usages. Une carte peut contenir une carte ou un article. Un article peut être un plat ou une boisson.

Les trois usages d'une carte sont :

- en tant que carte de haut niveau :
  Une carte de haut niveau contient d'autres cartes et des articles
- en tant que carte d'accueil :
  Une carte d'accueil est celle qui sera affichée par défaut à l'utilisateur.
- en tant que carte autres :
  Une carte autre est une carte simple.

Dans la suite de ce chapitre il sera présenté les différents composants de l'application en détaillant les diagrammes et en présentant les travaux réalisés.

4.1/ Contexte du système

Dans cette section il est présenté le diagramme de contexte du système afin de le localiser dans son environnement, les acteurs qui interagissent avec le système.

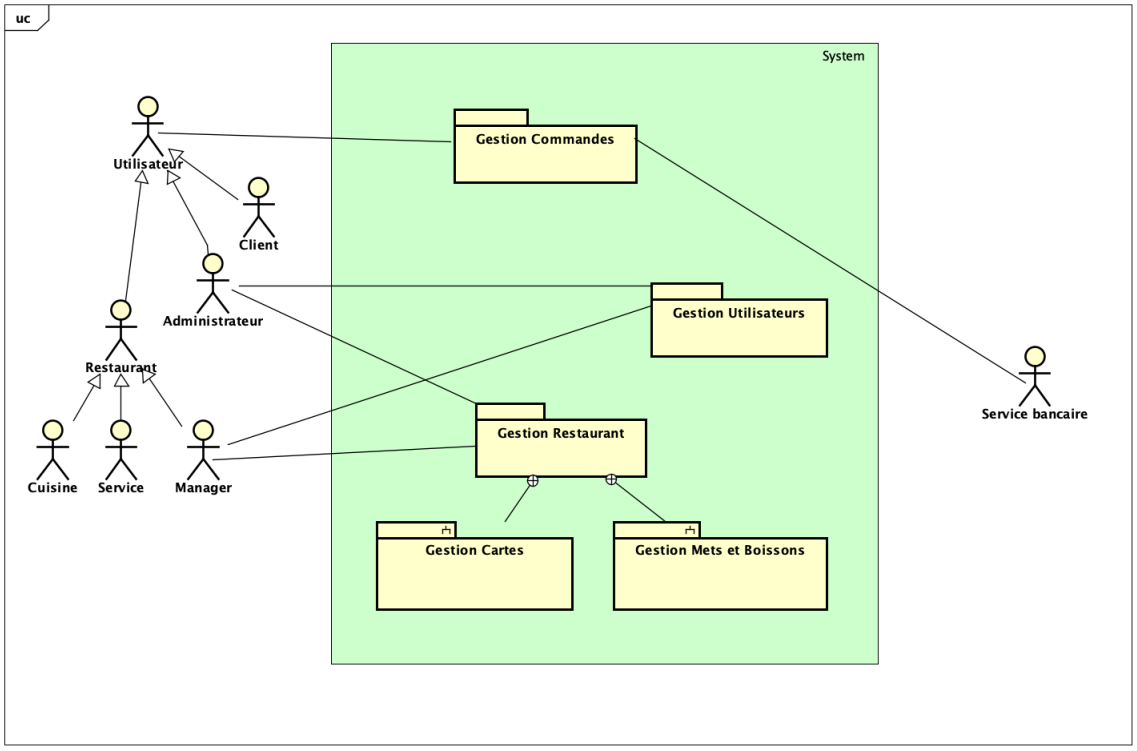

FIGURE 10 – Diagramme de contexte de l'application Resto

L'*utilisateur* est l'acteur principal. Il a plusieurs profil non cumulable. Comme indiqué sur la figure 10 les différents profil sont : le manager, le service, la cuisine, l'administrateur et le client.

Un *utilisateur* c'est-à-dire tous les profils peut gérer une commande. Un administrateur peut gérer les utilisateurs et les restaurants.

Un manager peut aussi gérer un restaurant et un seul.

L'acteur secondaire du système est la banque ou le service bancaire. Son rôle est de valider les paiements.

4.2/ CARACTÉRISTIQUE DE L'UTILISATEUR

L'application est destinée à l'utilisation du grand public et des restaurants. Elle ne requiert aucune compétence particulière à part le fait de savoir utiliser un smartphone.

4.3/ CONTRAINTES PRINCIPAUX DE DÉVELOPPEMENT

L'application a été développé pour la plateforme mobile iOS et selon le paradigme orienté objet. L'application étant assez complexe, il est important de la concevoir selon une architecture logicielle pour rendre facile son utilisation.

4.4/ BESOINS FONCTIONNELS

Il s'agit ici de décrire les besoins fonctionnels du système à travers les cas d'utilisation de l'application. Pour une meilleure étude des fonctionnalités de l'application, j'ai décomposé l'application

par paquet. Nous allons donc étudié les différents paquets.

4.4.1/ Paquet gestion des cartes

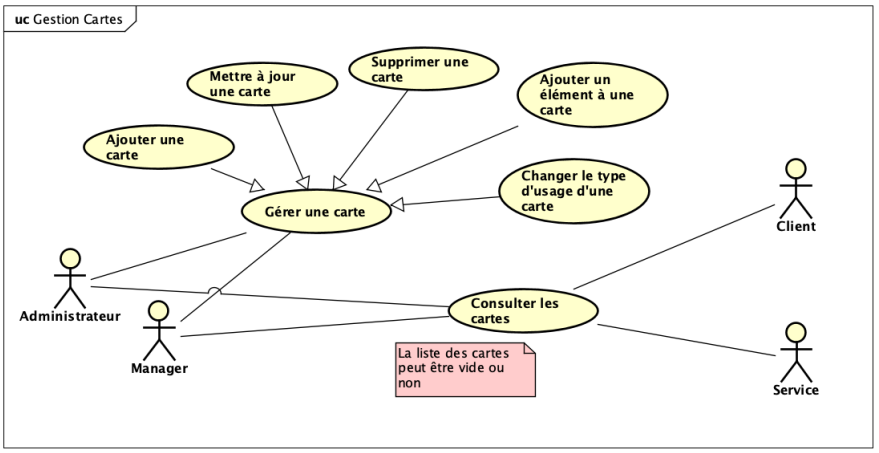

Figure 11 – Diagramme de cas d'utilisation de la gestion des cartes

Le diagramme sur la figure 11, présente une vue générale des cas d'utilisation du paquet. Chaque cas d'utilisation est lié à un acteur et décrit un état du système. Dans les prochaines sections, il sera détaillé les cas d'utilisation par acteur.

4.4.1.1/ Description des cas d'utilisation par acteur

Le système est composé de quatre acteurs principaux qui sont : l'*administrateur*, le *manager*, le *service*, et le *client*. Pour comprendre le rôle joué par chacun, nous procédons ici à une brève description de chacun d'eux.

L'*administrateur*

L'*administrateur*, peut ajouter une carte, mettre à jour les informations d'une carte, supprimer une carte et ajouter un article à une carte. Il peut aussi consulter les cartes. L'*administrateur*peut effectuer ces actions sur les cartes de tous les restaurants.

Le *manager*

Il peut effectuer les mêmes actions que l'*administrateur*mais ces actions sont limitées aux cartes du restaurant qu'il gère.

Le *service*

Il ne peut que consulter les cartes du restaurant dans lequel il est associé et voir leurs contenus.

Le *client*

Il peut consulter les cartes de tous les restaurants dans son lieu de résidence et voir leurs contenus.

4.4.1.2/ Spécification des structures de données

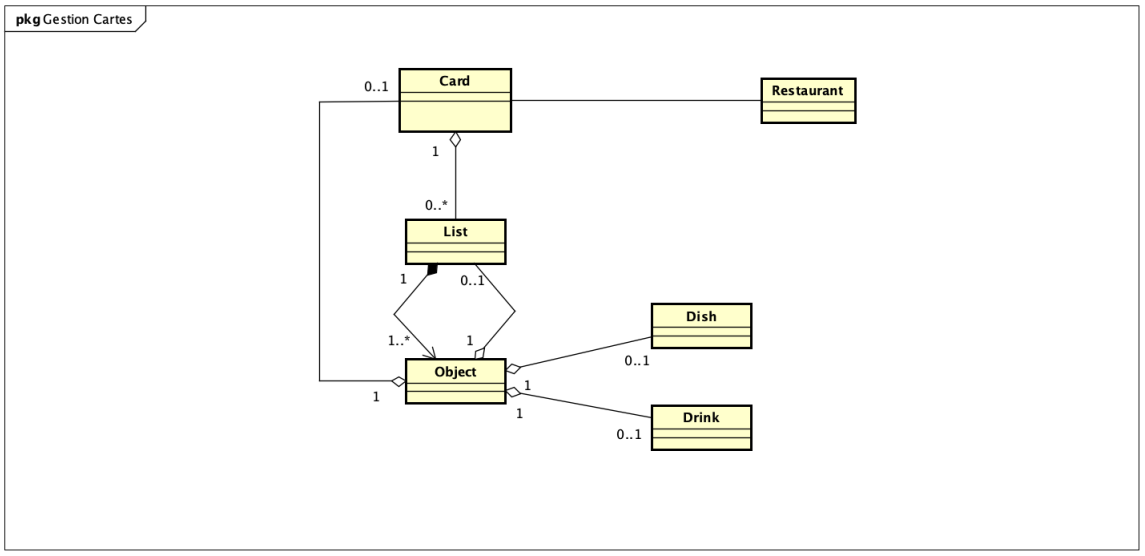

Figure 12 – Diagramme de classe de la gestion des cartes

Comme décrit sur le diagramme de la figure 12, un restaurant dispose de plusieurs cartes. Une carte possède zero ou plusieurs listes. Une liste peut exister sans carte. Une liste est composée de un ou plusieurs objet qui est soit un plat ou une boisson. Un objet peut posséder aussi une liste ou une carte.

4.4.2/ Paquet gestion d'un article

Le diagramme de la figure 13 présente une vue générale des cas d'utilisations du paquet. Chaque cas d'utilisation est lié à un acteur et décrit un état du système.

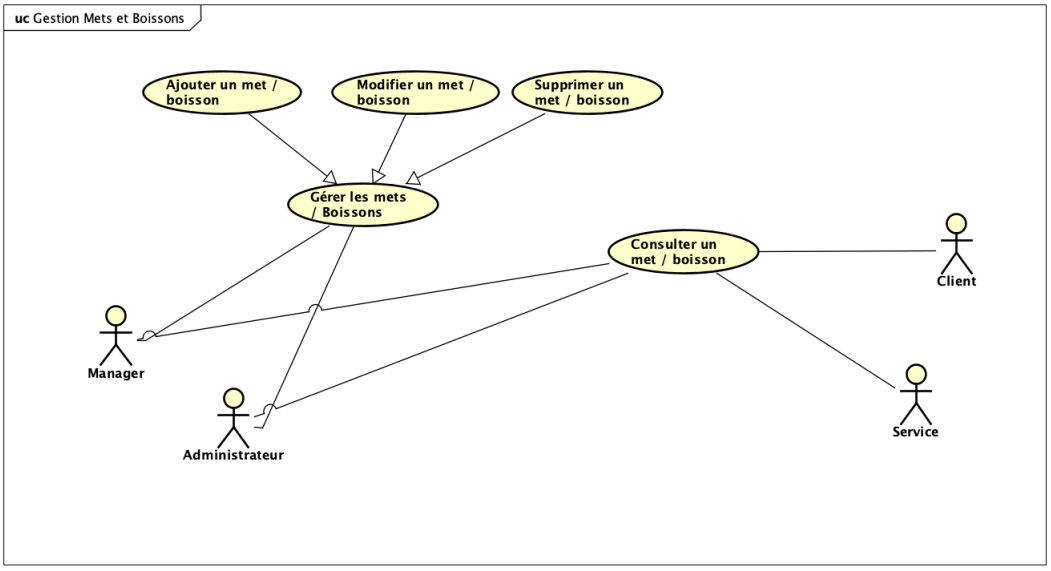

Figure 13 – Diagramme de cas d'utilisation de la gestion d'un article

4.4.2.1/ Description des cas d'utilisation par acteur

Les acteurs intervenant dans ce paquet sont les mêmes que le paquet précédent.

L'*administrateur*

Il peut ajouter, modifier les informations et supprimer un plat ou une boisson de tous les restaurants. Il peut aussi consulter les articles des restaurants.

Le *manager*

Il peut ajouter, modifier, supprimer et consulter un plat ou une boisson de son restaurant.

Le *service*

Il peut consulter les articles du restaurant auquel il est associé.

Le *client*

Il peut consulter les articles de tous les restaurants dans son lieu de résidence.

4.4.2.2/ Spécification des structures de données

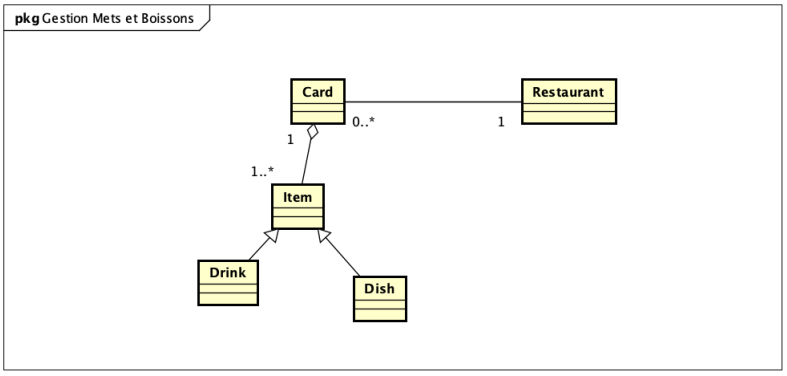

Figure 14 – Diagramme de classe de la gestion d'un article

D'après le diagramme de la figure 14, un article est soit un plat soit une boisson et il appartient à une carte.

4.4.3/ Paquet gestion des commandes

Le diagramme de la figure 15 présente une vue générale des cas d'utilisation du paquet. Chaque cas d'utilisation est lié à un acteur et décrit un état du système.

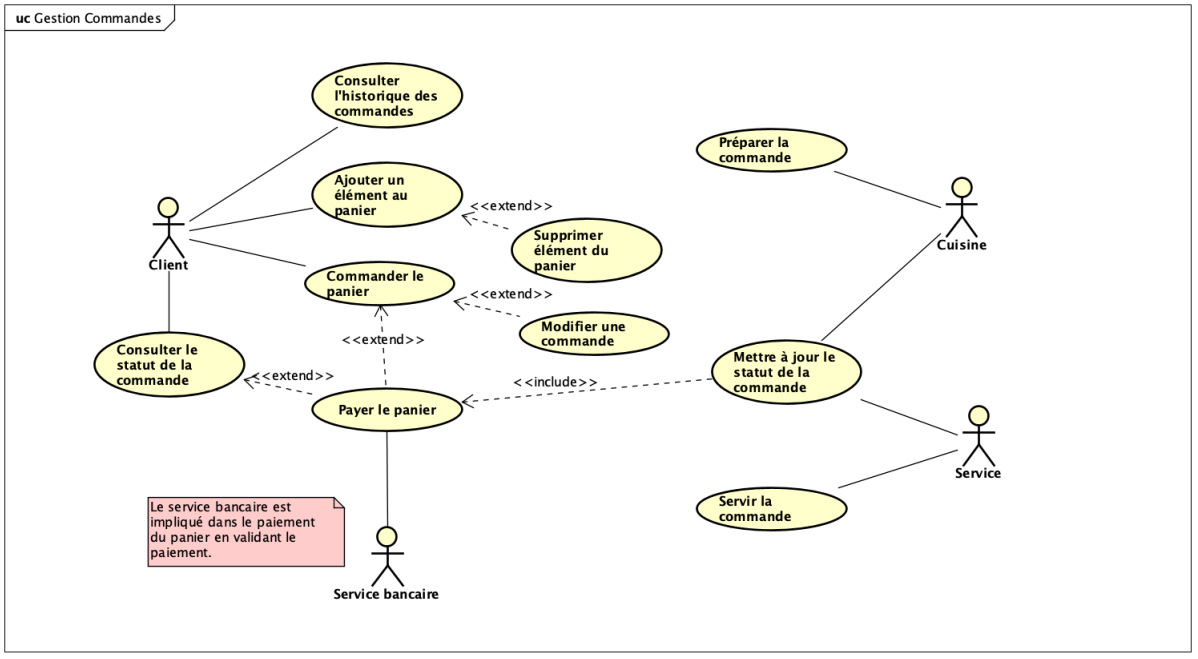

FIGURE 15 – Diagramme de cas d'utilisation de la gestion des commandes

4.4.3.1/ DESCRIPTION DES CAS D'UTILISATION PAR ACTEUR

Les principaux acteurs intervenant dans le fonctionnement de ce paquet sont : le *client*, le *service bancaire*, le *service*, et la *cuisine*.

LE *client*

Il peut ajouter un article au panier, commander et payer le panier, consulter le statut d'une commande, consulter l'historique des commandes, modifier une commande ou encore supprimer un article du panier.

LE *service bancaire*

Son rôle est de valider le paiement d'une commande.

LE *service*

Il peut mettre à jour le statut d'une commande et servir une commande.

LA *cuisine*

Elle peut préparer une commande et mettre à jour le statut d'une commande.

4.4.3.2/ Spécification des structures de données

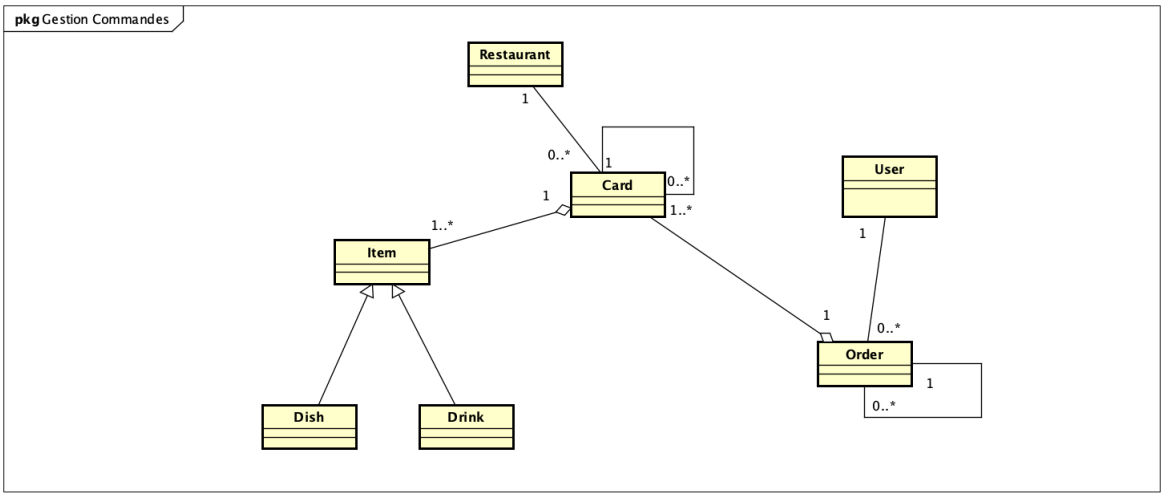

Figure 16 – Diagramme de classe de la gestion des commandes

D'après le diagramme de la figure 16, une commande contient un ou plusieurs cartes. Elle peut être liée à une autre commande pour constituer une commande de groupe.

D'une manière spécifique, une commande étant lié à un restaurant, le concept de panier aussi est lié d'un couplage fort au restaurant. Ainsi si pour une raison, le *client*sort du restaurant courant dans le cas où il a des commandes non validées, et visite un autre restaurant, le panier courant sera celui du nouveau restaurant et donc vide.

4.4.4/ Description générale

Comme défini sur la figure 10, l'application est composée de paquets et chaque paquet communique avec l'autre. L'architecture globale de l'application est une architecture client-serveur. Le serveur communique avec le client via une API REST dont le module de sécurité est le token de session lié automatiquement à un utilisateur.

4.5/ Spécification des interfaces externes

4.5.1/ Interface Matériel/logiciel

Configuration minimale sur laquelle le système peut s'exécuter

Pour faire fonctionner l'application sur un appareil mobile, l'appareil devra avoir les configurations minimales suivantes :

- Matériel : Tous les smartphones iOS supportant les configurations minimales au niveau logiciel.
- Logiciel : Tous les smartphones iOS supportant au moins la version 13.0 de l'OS.

Protocole d'échange

Pour le fonctionnement du système les protocoles utilisés sont le REST qui est basé sur les verbes HTTP.

4.5.2/ Interface logiciel/logiciel

Pour développer l'application, les outils suivants ont été utilisés :

- lanagages et bibliothèques de développement
  - Swift 5.1
  - Objective C 2.0
  - Cocoa Touch Framework
- Outils de développement
  - XCode 11.0
  - Git 2.2
  - GitLab
  - Mac Mini 2019 et MacOS Catalina
  - Simulateur iPhone 11 Pro Max

4.5.3/ Interface homme-machine

Structure

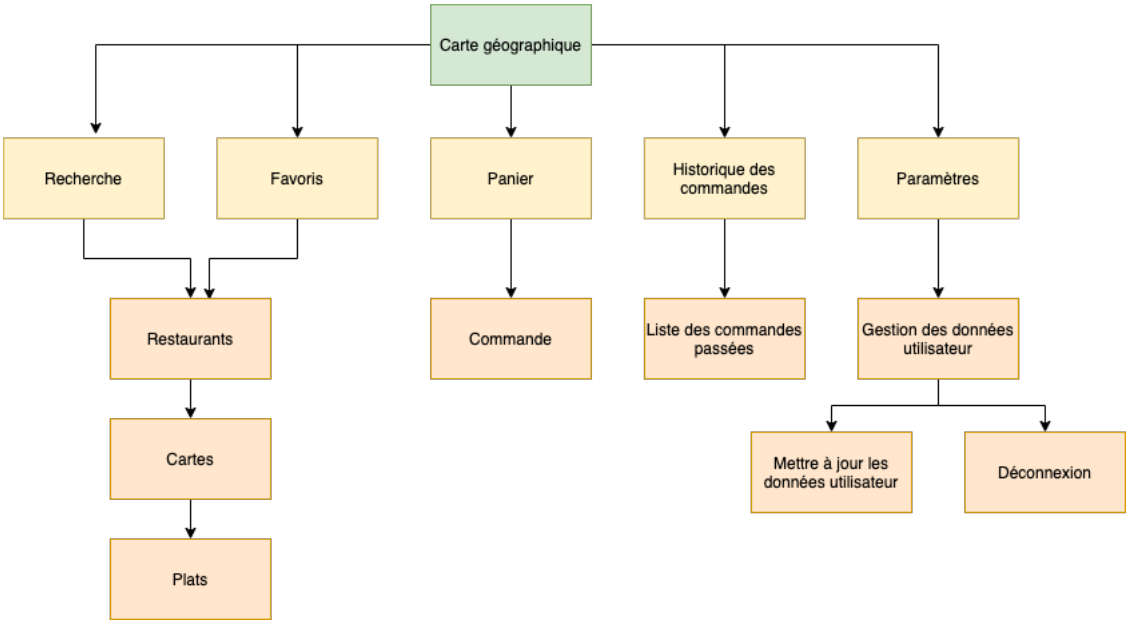

Figure 17 – Structure IHM de l'application client

Il est décrit sur la figure 17, la structure générale de l'IHM de l'application client. Le point d'entrée de l'application est la carte géographique. De cette carte, l'utilisateur peut se géolocaliser afin de réduire la zone de recherche des restaurants. En réalité il existe deux points d'entrée pour

l'application qui sont liées à deux scénarios. Le cas ou c'est un nouveau utilisateur ou un habitué. Si l'utilisateur est un nouveau, son point d'entrée est la carte mais s'il est un habitué son point d'entrée est l'onglet des favoris.

Le contenu des recherches et des favoris contient des restaurants. D'un restaurant, l'utilisateur peut consulter les cartes, les plats et effectuer une commande.

Le panier est l'onglet dans lequel l'utilisateur peut voir sa commande courante.

Une fois la commande passée, l'utilisateur peut consulter l'historique des commandes.

L'utilisateur peut gérer ses informations personnelles depuis l'onglet Paramètres ou se déconnecter.

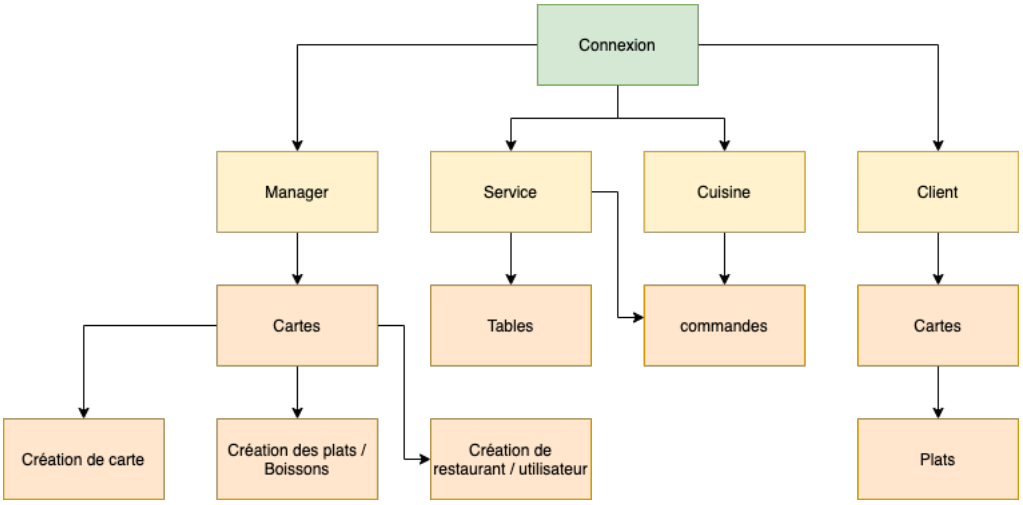

FIGURE 18 – Structure IHM de l'application pro

Il est décrit sur le diagramme de la figure 18, la structure générale de l'IHM de l'application pro. Le point d'entrée de l'application est la page de connexion. L'utilisateur connecté possède un profil. Selon son profil, l'utilisateur peut effectuer certaines actions.

Ainsi le *manager*pourra créer des cartes des plats ou encore un restaurant ou un utilisateur.

Le service peut voir l'organisation des tables et consulter les commandes.

La cuisine peut voir uniquement les commandes.

Le client peut consulter les cartes du restaurants et passer une commande.

Pour des raisons de sécurité, pour changer de profil, l'utilisateur est forcément rediriger vers un écran d'authentification.

4.6/ Déroulement du stage

4.6.1/ Développement et difficultés rencontrés

L'application Resto est composée de deux modules. Le module client et le module pro. Les deux modules ont été développés en deux applications différentes. Pour rendre l'application évolutive et pour éviter un couplage fort entre les composants, j'ai adopté une des méthodes de Clean architecture[3] développé par **Robert C. Martin**[4] appliqué au développement iOS au cours du développement. Cet architecture est nommé **VIPER**[5].

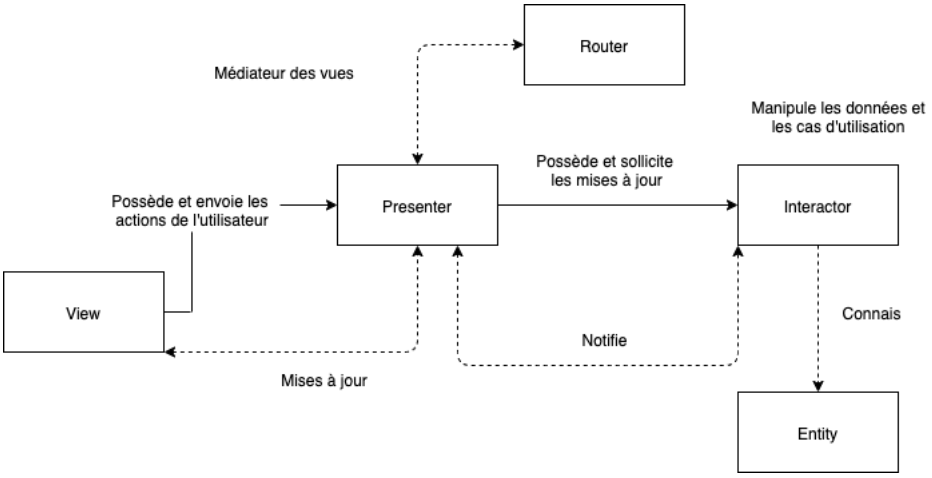

FIGURE 19 – Architecture VIPER

Chaque entité sur le diagramme, représente une classe au sens orienté objet. Ainsi la classe View est celle décrivant la vue. La classe Presenter met à jour la vue et demande à la classe Router de changer de vue si nécessaire ou à la classe Interactor de lui fournir les données nécessaires.

Comme décrit ci-haut on constate que chaque a une seule responsabilité ce qui respecte le fondement même des programmes orienté objet.

L'objectif de cet architecture est de permettre une évolutivité totale du code, une testabilité facile et un couplage très faible.

Appliqué à tous les composants de l'application, cette architecture m'a permis de la réutilisation des composants sans modifier le code. L'écriture des tests unitaires a été facile.

Un exemple de diagramme de classe d'un composant utilisant cette architecture est disponible dans l'annexe.

Les difficultés que j'ai rencontré au cours du développement résident lors de l'ajout d'un article au panier. Chaque article étant lié à une carte, c'était compliqué de réaliser la fonctionnalité d'ajout d'article au panier. Finalement j'ai résolu le problème en créant une classe générique fictive de navigation entre les cartes et j'ai adopté le pattern Observer pour gérer le panier.

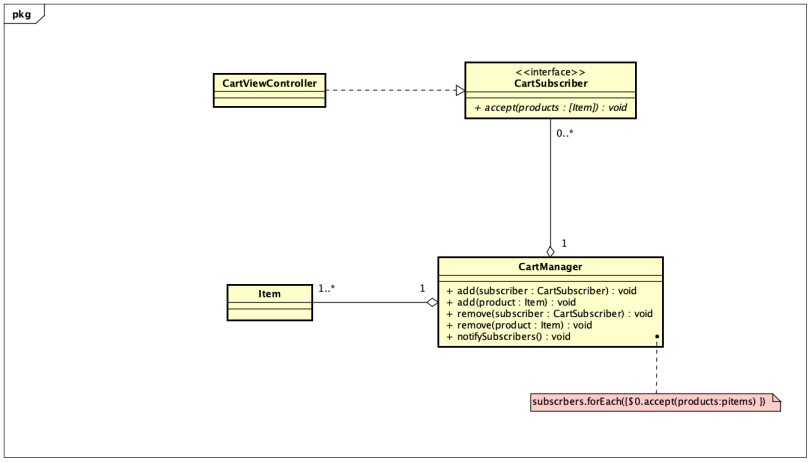

FIGURE 20 – Pattern Observer décrivant la gestion du panier

4.7/ Planning daté du travail

Le diagramme de Gantt de la figure 21 présente ma charge de travail sur le développement de l'application Resto

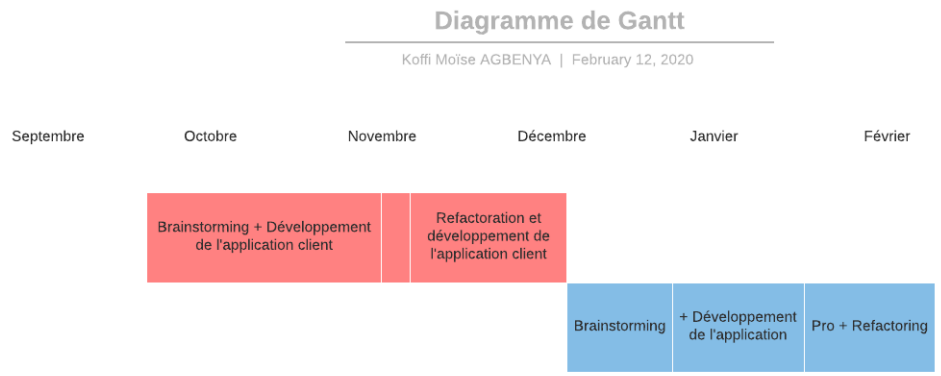

Figure 21 – Diagramme de Gantt du projet Resto

4.7.1/ Conditions de travail

Horaires

Mes horaires de travail pendant le stage étaient de 08h à 17h avec une pause de 1h soit entre 12h et 13h ou 13h et 14h. L'horaire journalier de travail est de 8h au Luxembourg.

Badges et Accès

L'accès aux bureaux requiert un badge. J'ai obtenu mon badge dès le début de mon stage.

Accès internet

L'accès à internet n'est pas restreint dans l'entreprise mais l'entreprise dispose d'un proxy pour tout échange avec l'extérieur.

Conclusion

Entreprise

L'entreprise Luxembourg Online est exceptionnel dans le sens où l'apprentissage est autorisé au cours du stage. Ainsi avant mon arrivé dans l'entreprise, je ne savais pas codé pour les plateformes de Apple mais la possibilté m'a été laissé d'apprendre la technologie afin de travailler de manière autonome. Une des particularité que j'ai aimé de l'entreprise est qu'elle développe elle même presque tous les outils utilisés. Cette façon de faire à pousser ma curiosité à explorer des domaines que je n'aurai pas exploré si je n'avais pas fait ce stage

Travail réalisé

Je suis content du travail que j'ai réalisé au cours de mon stage. Le sujet initial étant le développement d'application sous android et iOS, j'était venu avec l'handicap de ne pas savoir comment développer pour les plateformes iOS, mais j'ai su vite apprendre et produire les deux premières applications le premier mois de stage. J'ai ainsi pu travailler sur deux autres applications à la suite.

Expériences

J'ai appris énormément au cours de ce stage et j'ai appris beaucoup sur moi même. Le travail que j'ai réalisé m'a permis d'améliorer mon niveau en conception et architecture logicielle, il m'a permis d'apprendre à apprendre et assimiler en moins de 48h un langage de programmation et en 1 semaine une technologie.

Ce stage a été pour moi l'occasion de travailler sur des sujets assez intéressant et des applications grand public. Les spécificités de ces types d'applications sont souvent différentes des projets que nous réalisons dans le cadre scolaire. Ainsi j'ai appris à faire une application qui sera utilisé par un grand nombre d'utilisateur. Pour ce faire, il faut faire au préalable une conception approfondie de l'application avant de commencer le développement. J'ai ainsi à travers ce stage amélioré mes compétences en développement logiciel.

Bibliographie

[1] Google, « Android documentation ». https://developer.android.com/guide, 2019. [Online ; accessed 2019-09-02].

[2] Apple, « Swift documentation ». https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html, 2019. [Online ; accessed 2019-09-16].

[3] R. Martin, « The clean architecture ». https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html, 2012. [Online ; accessed 2019-11-12].

[4] R. Martin, *Clean Code : A Handbook of Agile Software Craftsmanship*. 2008.

[5] J. Gilbert et C. Stoll, « Architecting ios app with viper ». https://www.objc.io/issues/13-architecture/viper/, 2014. [Online ; accessed 2019-11-12].

Annexes

A/ Cycle de vie d'une activité d'Android

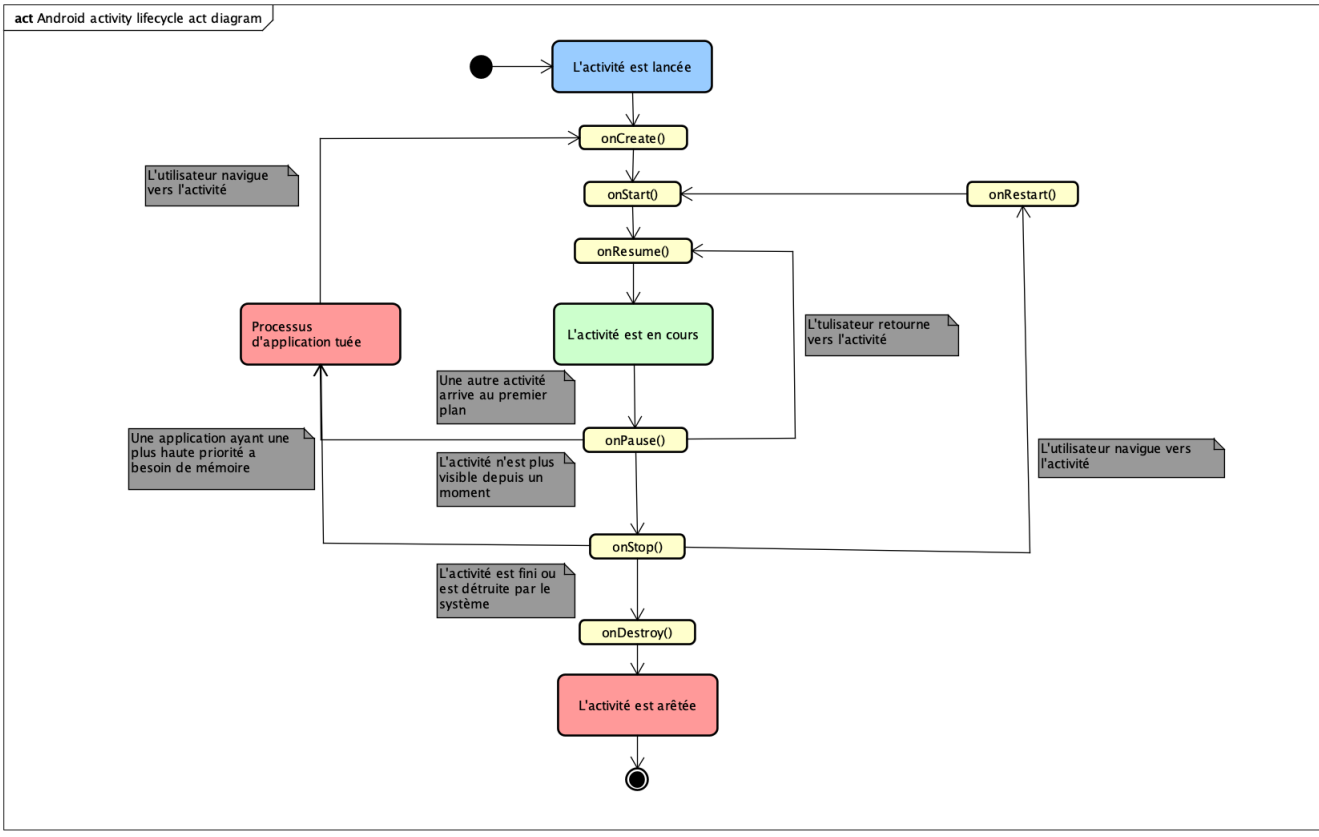

Figure 22 – Diagramme d'activité du cycle de vie d'une activité Android

B/ Diagramme de séquence du cycle de vie d'une activité

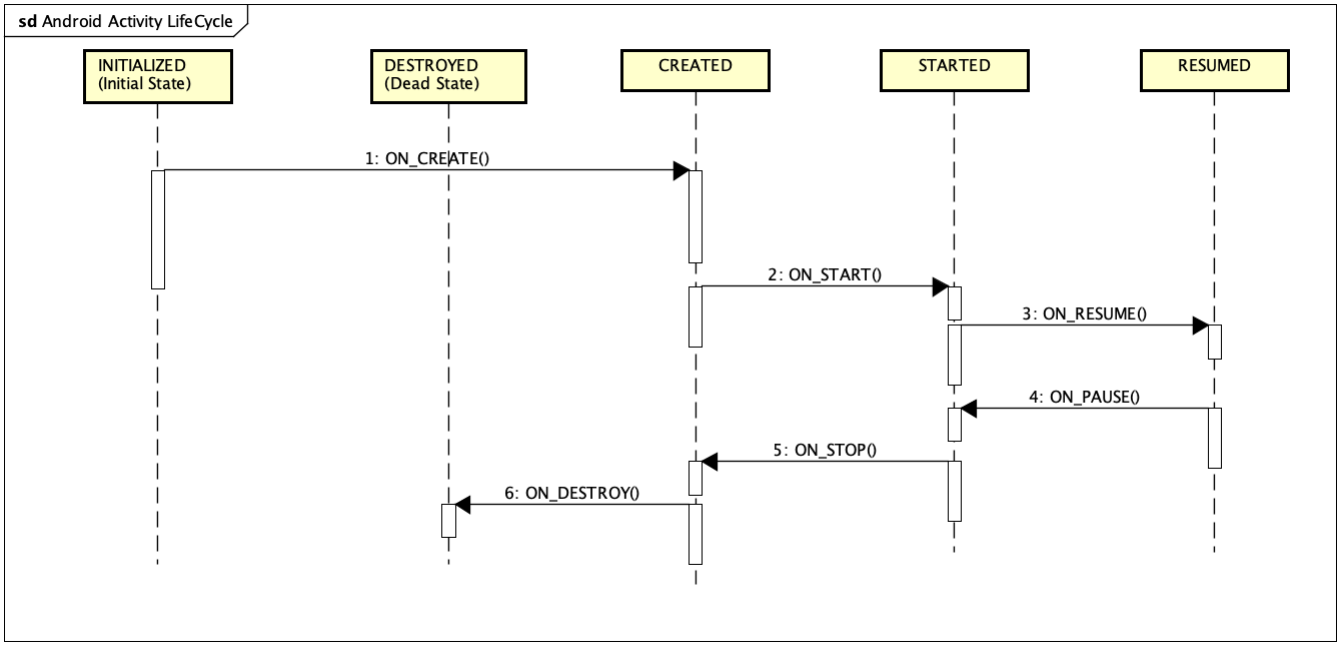

Figure 23 – Diagramme de séquence du cycle de vie d'une activité Android

C/ Diagramme d'état du cycle de vie d'une activité

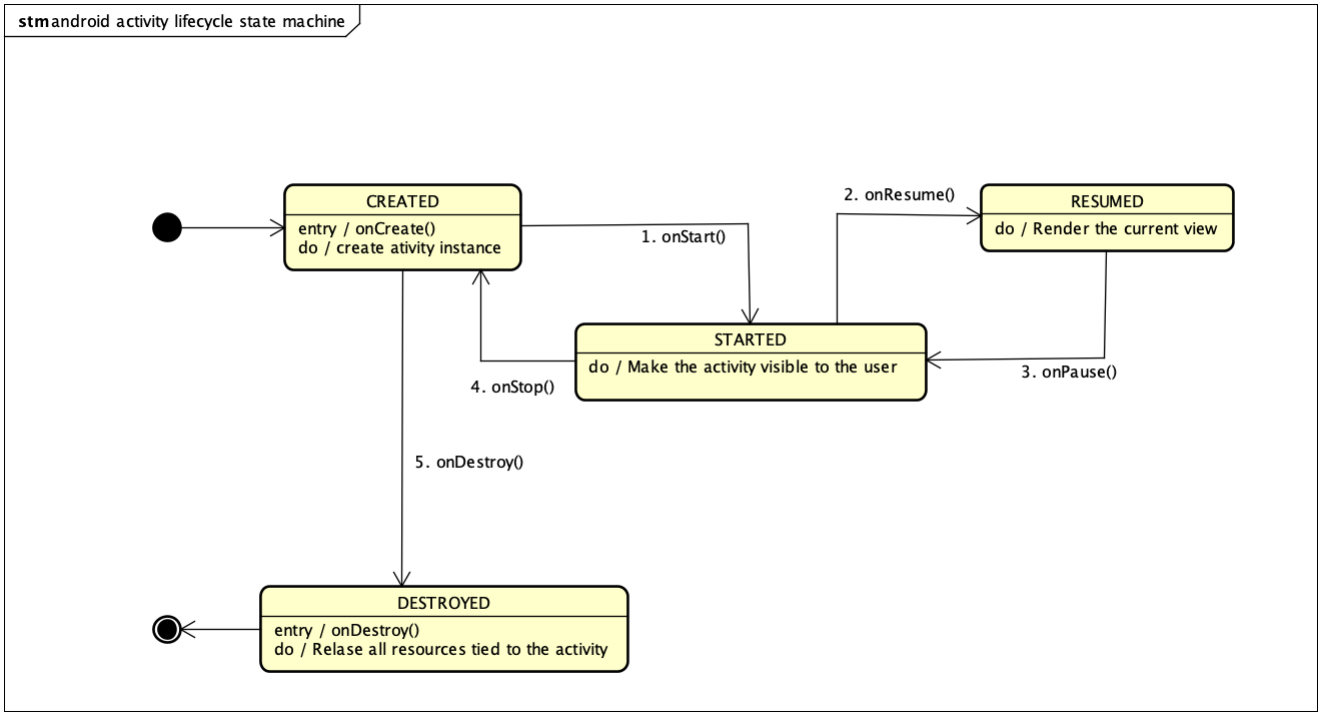

Figure 24 – Diagramme de état-transition du cycle de vie d'une activité Android

D/ Gestion du panier

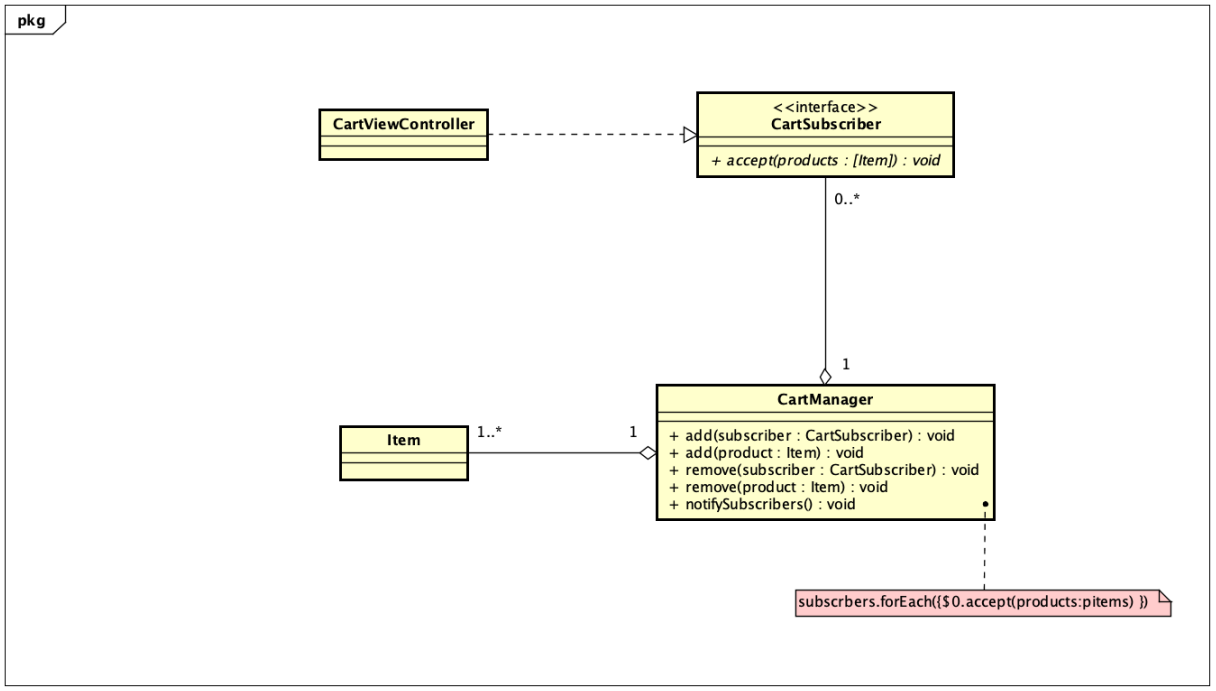

Figure 25 – Diagramme de classe de la gestion des paniers

E/ Diagramme d'activité du cycle de vie d'une activité iOS

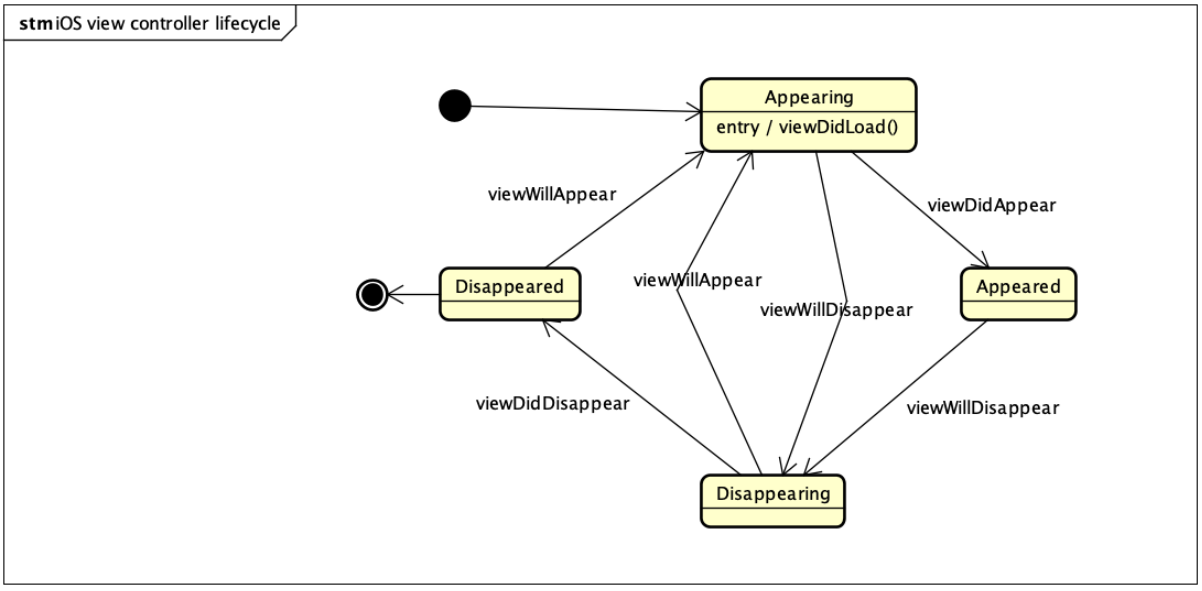

Figure 26 – Diagramme d'activité des cycles de vie

F/ Architecture Viper

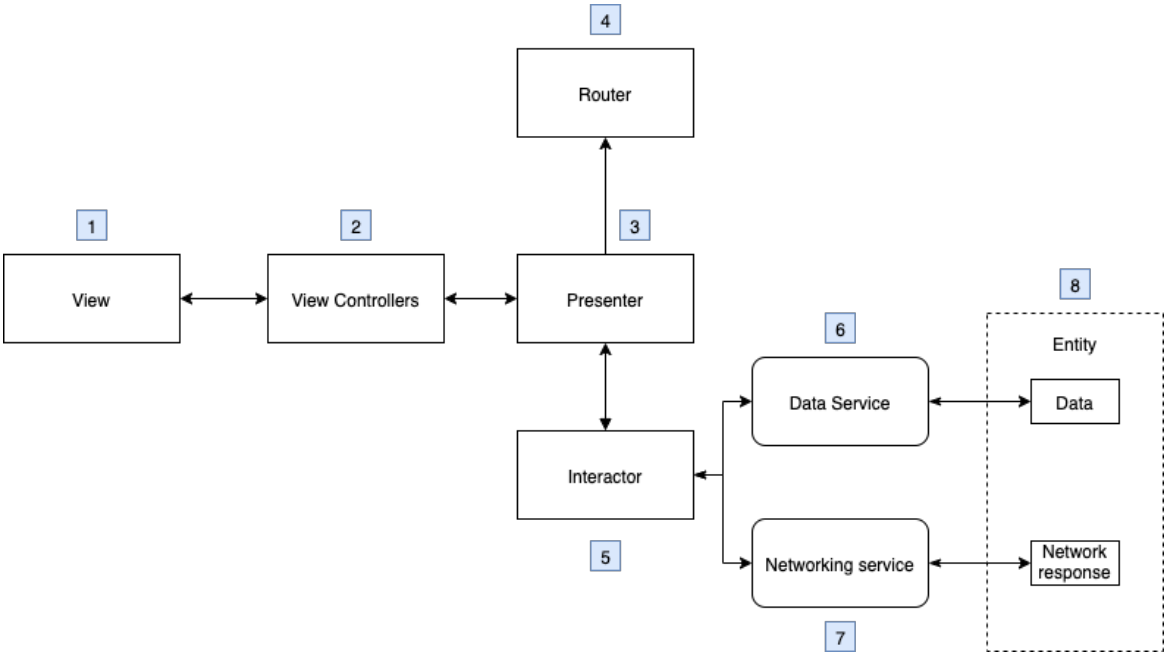

Figure 27 – Description de l'architecture VIPER

G/ Exemple d'application de l'architecture VIPER

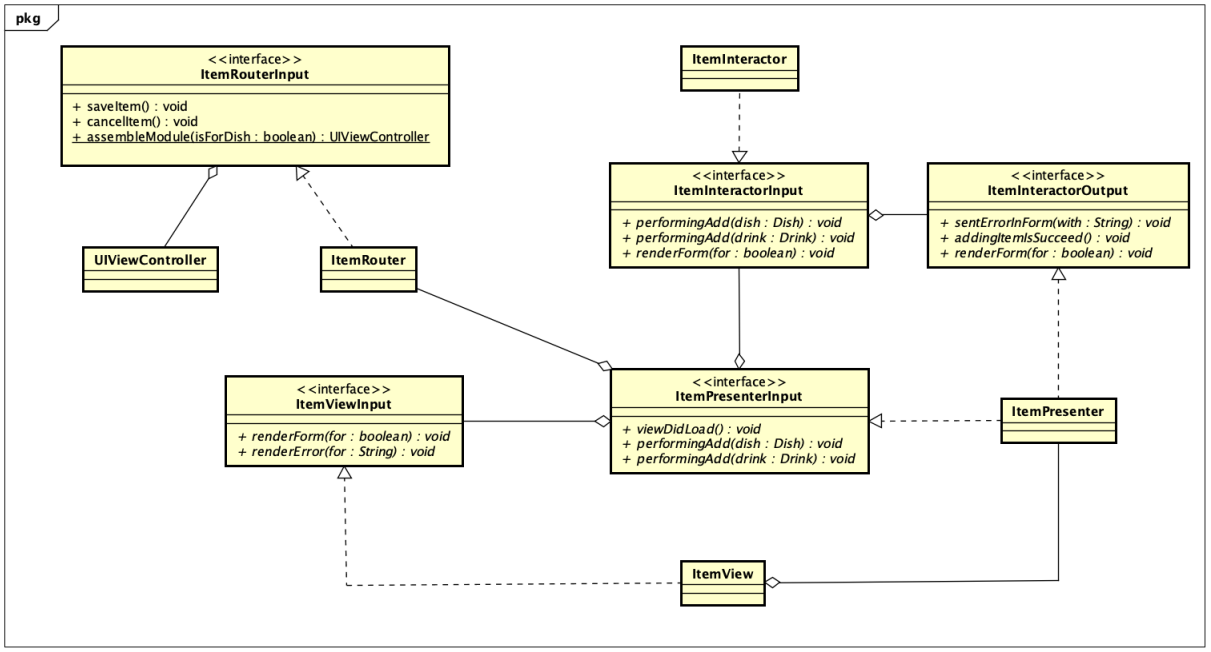

Figure 28 – Diagramme de classe d'un composant de l'application avec l'architecture VIPER

Mots clefs

Télécommunications - Informatique - Développements logiciels - Architecture logicielle - Logiciel réseau - Logiciel grand public

Koffi Moïse Agbenya

Rapport de stage ST40 - A2019

Résumé

De nos jours, de plus en plus d'entreprises proposent des solutions innovantes afin de résoudre de nombreux problèmes rencontrés par des prospects ou pour fidéliser leurs clients. Face à la montée croissante d'utilisateurs de téléphone mobile, ces solutions ciblent en plus grande partie les téléphones mobiles. Derrière la simplicité d'utilisation d'une application mobile se cache beaucoup de processus qui peuvent devenir très rapidement complexe. Au cours de mon stage, la tâche qui m'a été confiée est de développer des applications pour téléphone mobile fonctionnant sous les systèmes d'exploitations Android et iOS. Les applications que j'ai développé deux (2) au total, ont respectivement pour objectif de : faire le diagnostique à distance d'un modem-routeur ; et de réserver, commander dans ses restaurants préférés ou se les faire livrer. Ce document découlant de mon stage en tant que développeur d'application mobile au sein de Luxembourg Online SA, exhibe la totalité des activités menées et précisément sur le sujet « Développement d'application sous android et iOS » dans le cadre de ma formation à l'Université de Technologie de Belfort Montbéliard (UTBM).

LUXEMBOURG ONLINE S.A.
14 Avenue du X Septembre
L-2550 Luxembourg
www.internet.lu

utbm
université de technologie
Belfort-Montbéliard