Remerciement

Tout d'abord, nous remercions Dieu, notre créateur, pour nous avoir donné la force, la volonté et le courage d'accomplir ce modeste travail.

Nous remercions notre directeur de thèse Monsieur Allaoui et Madame Kerrouche pour leurs conseils et leur encadrement du début à la fin de ce travail.

Nos remerciements vont à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail. Enfin, nous tenons à exprimer notre profonde gratitude à nos familles qui nous ont toujours soutenus et à tous ceux qui ont participé à la réalisation de cette thèse. Ainsi qu'à tous les enseignants qui ont contribué à notre formation.

Dédicaces

Nous dédions ce modeste ouvrage à nos parents bien-aimés qui nous ont soutenus tout au long de notre vie.
À nos amis qui n'ont jamais hésité à nous aider de quelque manière que ce soit.
Et à tous ceux qui nous ont enseigné tout au long de notre vie scolaire.
A nos camarades de la 3ème année LMD.
Merci à tous.

Abstract

this is the abstract

Contents

List of Figures

Liste des Abréviations

- **MVC**: Model View Controller
- **UI**: User Interface
- **SDK**: Software Development Kit
- **UML**: Unified Modeling Language
- **NoSQL**: Not only Structured Query Language
- **JSON**: Java Script Object Notation

Introduction

Au cours de la dernière décennie, nous avons assisté à une révolution dans les commerces en ligne. Les gens se sont habitués aux achats en ligne et aux réservations à distance comme la réservation de billets, l'achat d'articles et même la commande de nourriture, principalement parce que c'est plus rapide, sans effort et moins compliqué.

Lorsque quelqu'un veut acheter de la nourriture en ligne, il doit vérifier un restaurant qui fabrique la nourriture qu'il veut, puis la nourriture doit lui être livrée soit par le livreur du restaurant, soit par un établissement de livraison séparé.

Comme nous l'avons vu dans les pays étrangers en Europe ou aux États-Unis, beaucoup de restaurants et de sociétés de livraison ont leur propre application mobile. Malheureusement, tous les établissements ne peuvent pas se permettre de créer leur propre application en raison des coûts élevés de développement d'une application mobile.

Notre solution est une plateforme conçue comme une application mobile qui regroupe les clients, les restaurants et les sociétés de livraison en un seul endroit. Elle répond à tous les besoins des personnes qui veulent commander leur nourriture à distance, elle permettra également aux propriétaires de restaurants et aux sociétés de livraison d'augmenter leurs ventes et d'accélérer leur flux de travail avec les clients à distance.

Chapter 1

Etude Bibliographique

1.1 Développement Android

Nous avons choisi de réaliser une application mobile dédiée aux utilisateurs d'Android, et nous avons beaucoup de choix quant aux technologies que nous allons utiliser pour le développement.

1.1.1 Les technologies utilisées

Nous avons utilisé le framework Flutter pour les interfaces et la logique de l'application, et Cloud Firestore de Firebase pour le stockage et la gestion de la base de données.

Flutter

Un kit de développement logiciel d'interface utilisateur open-source créé par Google. Il est utilisé pour développer des applications multiplateformes pour Android, iOS, Linux, Mac, Windows, et le Web à partir d'une base de code unique. Nous avons utilisé Flutter en raison de sa structure qui favorise la vitesse de développement. Sa structure est simplement une cascade de widgets faciles à personnaliser.

Flutter utilise le langage de programmation Dart, également développé par Google. Il s'agit d'un langage orienté objet, basé sur des classes, avec une syntaxe de type de la language C.

Firebase

la plateforme mobile de Google qui aide à développer rapidement des applications de qualité. Elle dispose de nombreuses fonctionnalités, mais nous l'utilisons principalement pour son service de stockage au Cloud (Cloud Firestore).

Cloud Firestore est une base de données documentaire NoSQL (un ensemble de continuation en cascade collection-document) qui nous permet de stocker, de synchroniser et d'interroger facilement des données pour des applications mobiles et web - à l'échelle mondiale.

1.2 Méthode de travail collaborative

Cette application a été développée à partir de zéro par trois développeurs, nous devions travailler en parallèle sur le même projet, nous avons donc utilisé Git et Github.

Git

C'est un logiciel permettant de suivre les modifications apportées à un ensemble de fichiers. Il est généralement utilisé pour coordonner le travail des programmeurs qui collaborent à l'élaboration du code source lors du développement de logiciels. Ses objectifs sont la rapidité, l'intégrité des données et la prise en charge des flux de travail distribués et non linéaires.

Github

C'est un service d'hébergement de référentiel Git, mais il ajoute de nombreuses fonctionnalités qui lui sont propres. Il fournit une interface graphique basée sur le Web. Il fournit également un contrôle d'accès et plusieurs fonctions de collaboration.

1.3 Architecture du code

Nous avons utilisé l'architecture de projet MVC. Il s'agit d'un modèle de conception de logiciel couramment utilisé pour développer des interfaces utilisateur qui divisent la logique du programme en trois éléments interconnectés.

Cela permet de séparer les représentations internes de l'information de la manière dont l'information est présentée à l'utilisateur et acceptée par celui-ci.

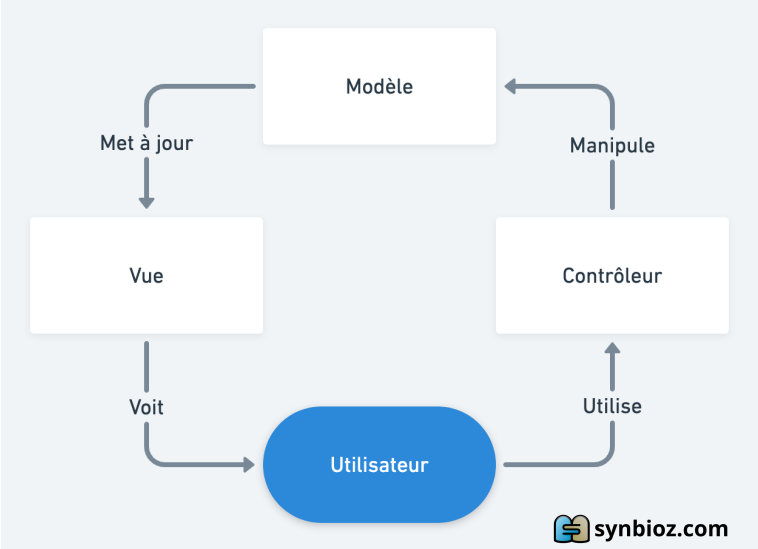

Figure 1.1: Le flux MVC

Chapter 2

Conception

2.1 Modélisation

On a choisi trois diagrammes d'UML pour modéliser l'application:

- Diagramme de classes
- Diagrammes de cas d'utilisation
- Diagrammes de séquence

Diagramme de classes

bbbbbbbbbbbrève explication

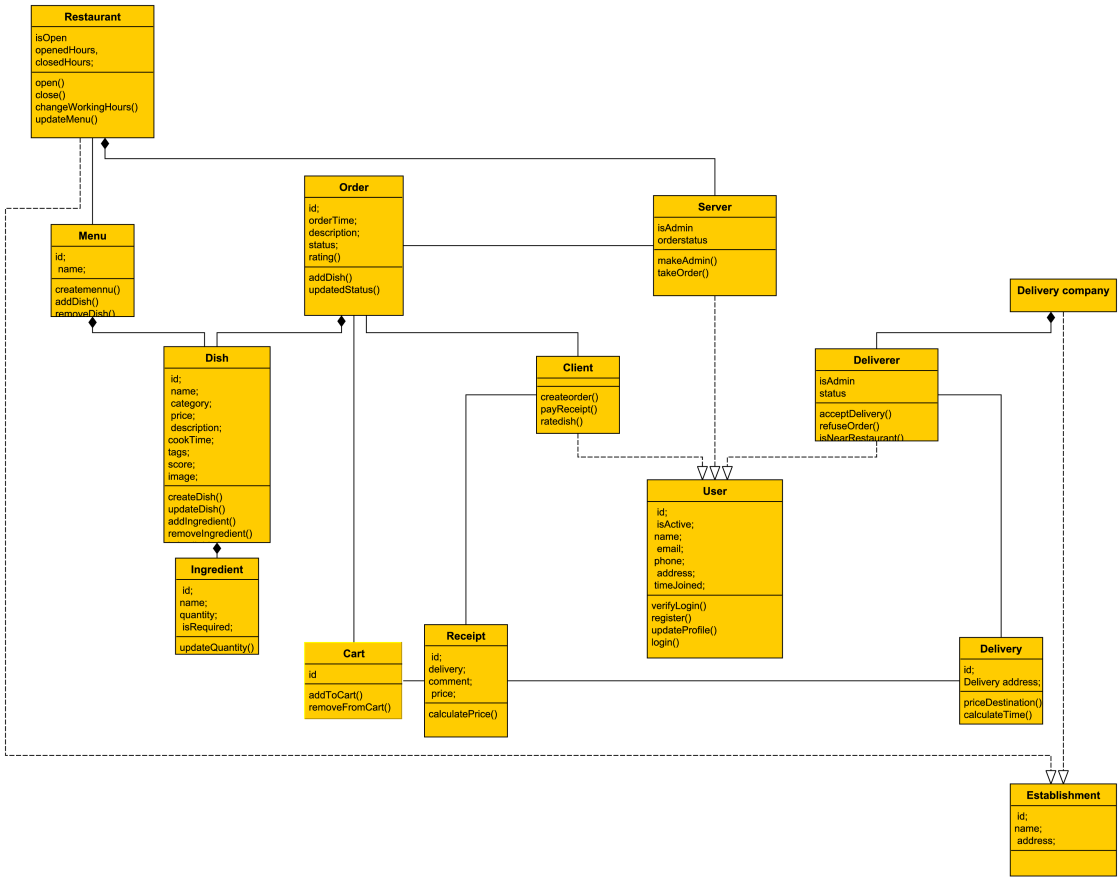

Figure 2.1: Diagramme de classes

Diagrammes de cas d'utilisation

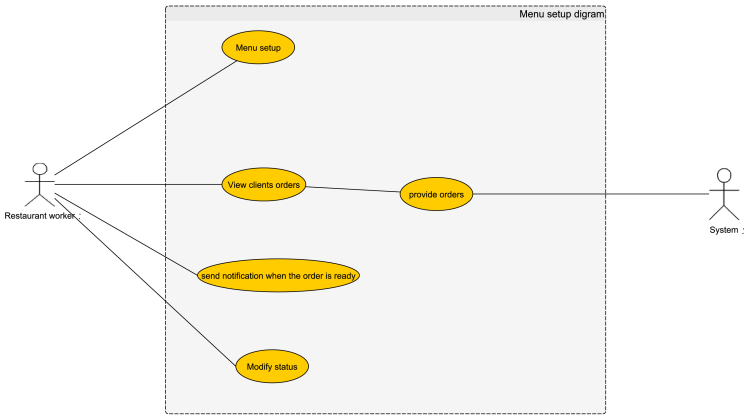

Figure 2.2: Diagramme de cas d'ustilisation du configuration du menu

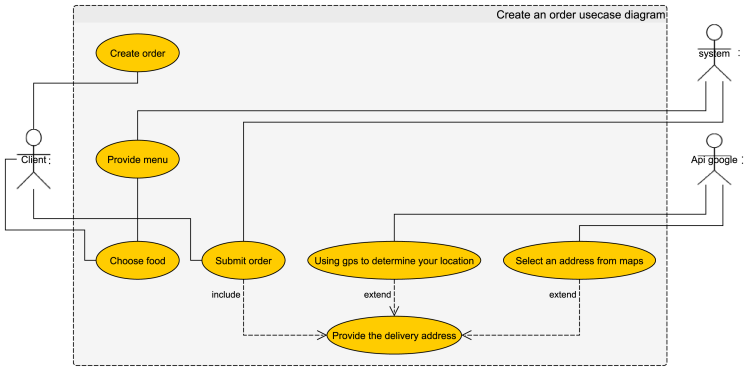

Figure 2.3: Diagramme de cas d'ustilisation du passation d'une commande

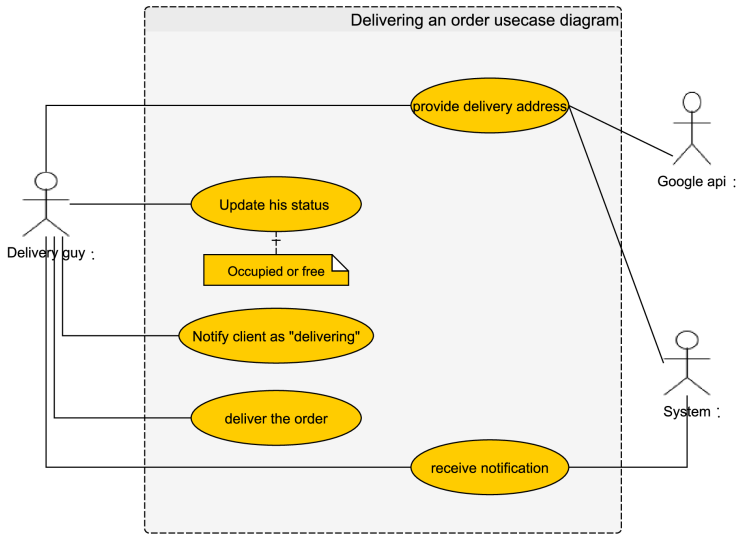

Figure 2.4: Diagramme de cas d'ustilisation du livraison d'une commande

Diagrammes de séquences

2.2 La structure du code

2.2.1 Modéles

Nous avons représenté chaque classe du diagramme de classes (Fig. 2.1) comme des classes Dart pour les traiter comme des objets. Chacune de ces classes possède deux méthodes permettant de convertir de et vers des objets JSON pour faciliter la communication avec la base de données.

2.2.2 Contrôleurs

Ils représentent la fonctionnalité principale de chaque action dans l'application, de la connexion à la mise à jour des commandes et tout ce qui est similaire.

2.2.3 Vues

Chaque écran qui apparaît à l'utilisateur est considéré comme une vue. Les vues contiennent plusieurs widgets en cascade.

2.2.4 Widgets

Nous avons dû créer des widgets personnalisés pour répondre à nos besoins, comme des boutons, des vues de liste, des vues de texte, etc.

Chapter 3

Réalisation

3.1 Startup

sign in and up

3.2 Client case

flows

3.3 Restaurant case

flows

3.4 Deliverer case

flows

Conclusion

blblabla

Bibliography