|  |
| --- |
|  |
| Conception et réalisation d’une application Flutter de géolocalisation dynamique et de réservation de Taxis |
| Travail de fin de formation en vue de l’obtention du titre professionnel Concepteur développeur d’applications Niveau II |
|  |
| **Zakaria SACI** |
| **24/06/2021** |

**RÉSUMÉ**

Le monde entier fait face actuellement à un problème de transport dû à l’urbanisation. Depuis quelques années, nous vivons une forte concentration de la population en milieu urbain. C’est la raison pour laquelle, nous assistons à une augmentation de la demande du transport au niveau des villes. Donc, la question de transport représente un maillon fort dans le processus de développement é

Aussi, le rôle du taxi a radicalement évolué dans la mobilité urbaine avec d’une part des besoins croissants pour des services personnalisés que les taxis sont les plus aptes à offrir, et d’autre part, la perspective de voir se desserrer les deux freins à leur développement : la disponibilité et le coût relativement élevé.

En outre, avec l’avènement du web mobile et la généralisation des smartphones, les applications mobiles sont devenues un point central de notre vie numérique. Elles se sont multipliées sur nos différents devices mobiles, et ont même tendance à devenir la norme sur desktop.

Notre travail consiste en « la conception et la réalisation d’une application mobile sous Android pour la réservation de taxis ». Il s’agit de mettre en place un système pour permettre à un client de rechercher un taxi de façon conviviale, facile et fiable.

Pour mettre en œuvre notre système, nous avons utilisé le processus Unifié (UP), qui se base sur l’UML comme langage de modélisation conçu pour fournir une méthode normalisée pour la conception. L’application que nous avons réalisée utilise un ensemble de technologies d’actualité et elle est pourvue de fonctionnalités nécessaires et appropriées aux besoins des clients et des chauffeurs de taxi.

**Mots clés** : Application mobile, Flutter, NoSQL, Firebase, Service Web et JSON.

**ABSTRACT**

The whole world is currently facing a transport problem due to urbanization. In recent years, we have a high concentration of the population in urban areas. That’s why we are seeing an increase in the demand for transport at the city level. Therefore, the transportation issue represents a strong link in the process of economic and social development of a city.

Also, the role of the taxi has changed radically in urban mobility with, on one hand, growing needs for personalized services that taxis are best able to offer, and on the other hand, the prospect of loosening the two brakes to their development : vailability and relatively high cost.

In addition, with the advent of the mobile web and the spread of smartphones, mobile apps have become a focal point of our digital life. They have multiplied on our different mobile devices, and even tend to become the standard on desktop.

Our work is "the design and implementation of an Android mobile application for booking taxis". It is about setting up a system to allow a customer to search for a taxi in a user-friendly, easy and reliable way.

To implement our system, we have used the Unified Process (UP), which is based on UML as a modeling language designed to provide a standardized method for design. The application we have made uses a set of topical technologies and is equipped with the necessary features that are appropriate to the needs of customers and taxi drivers.

**Key words** : Mobile application, Flutter, NoSQL, Firebase, Web service and JSON.

**Cahier des charges**

1. **Contexte**

Taxi Amiens est une jeune entreprise de transport de voyageur intra et inter urbain. L’entreprise dispose d’un groupement de taxieur Ce dernier souhaite numériser l’accès à leurs services par l’utilisation de mode de communication numérique au moyen de téléphone mobile.

On cherche à développer une solution numérique permettant de mettre en relation les utilisateurs.

2. **Objectifs**

L’objectif principal est **de développer une application mobile Cross\_plateforme.** Cet outil numérique permet la mise en contact en temps réel et instantané de voyageur souhaitant se déplacer et des transporteurs disponibles pour cette mission.

La solution que nous sommes chargés de développer pour le groupement doit répondre impérativement à certaines attentes qui ont été clairement énoncées lors du premier rendez-vous avec le gérant de Taxi\_Amiens.

1. Voyageur et transporteurs doivent pouvoir se géolocalisés en temps réel.
2. La solution doit permettre au voyageur d’afficher sa position géographique en coordonnée postales.
3. La solution doit permettre au transporteur de se mettre en disponibilité ou non pour des missions potentielles dans un rayon géographique raisonnable.
4. La solution doit permettre au voyageurs d’identifier l’itinéraire de déplacement du point de départ jusqu’au point d’arrivée, la distance de déplacement et le cout de ce service.
5. La solution doit permettre au voyageur d’identifier les transporteurs disponibles pour cette mission dans un rayon de proximité raisonnable.
6. La solution doit permettre l’accès des utilisateurs (Client, Taxieur) possédant un terminal Android\_OS ou iPHONE\_OS.

3. **Contraintes**

Exigences de sécurité de l’application.

TODO Liste

**Spécification des besoins**

**Identification des acteurs :** Les acteurs que comporte notre système sont :

1. **Client** / Voyageur (Riders): voyageur souhaitant un taxi pour leurs déplacements.
2. **Taxieur** / Transporteur (Drivers) : transporteur disposant de disponibilité à proximité du client.

**1. Besoins fonctionnels**

Dans cette partie nous détaillons les fonctionnalités, que le système doit fournir aux différents acteurs qui se présentent comme suit :

1. Inscription.
2. Authentification
3. Localisation de l’utilisateur.
4. Réservation d’un taxi en temps réel.
5. Traçage de l’itinéraire.
6. Estimation du temps et de la distance.
7. Gestion des demandes de réservation.
8. Suivre l’approche du taxieur en temps réel.
9. Notifier le client lorsque le taxieur arrive.
10. Consultation de l’historique.
11. Noter l’utilisateur.

**2. Besoins non fonctionnels**

Il s’agit des besoins qui caractérisent le système. Ce sont des besoins en matière de performance, de type de matériel ou de type de conception. Pour cela notre futur système doit répondre aux caractéristiques suivantes :

1. Ergonomie.
2. Rapidité de traitement.
3. Facilité d’utilisation.
4. Sécurité.
5. 5. Besoins d’écoconception de l’application.

**Conditions et mode d’utilisation**

1. Facilité et simplicité d’utilisation de l’interface de l’application, que ce soit pour le client ou pour le chauffeur de taxi.
2. Plusieurs moyens d’inscription par Facebook, Google ou par mot de passe et email
3. Lors d’une demande de réservation, la source du client sera localisée automatiquement et dans le cas où le client décide d’indiquer un lieu de rendez-vous.
4. Le chauffeur de taxi pourra accepter ou refuser une demande de réservation.
5. Si le taxieur ne répond pas à une demande de réservation envoyé par le client au bout de 3 minutes, la demande de ce dernier sera expirée. De même pour le client, s’il ne confirme pas le prix indiqué par l’application
6. Possibilité de consulter un guide d’utilisation « Comment ça marche » qui est utile lors de la première utilisation de l’application.
7. Le client ne peut pas envoyer une demande de réservation à un autre taxieur s’il en a déjà envoyé une.
8. Possibilité de récupérer l’historique d’une course.
9. Le client sera notifié lorsque le chauffeur de taxi arrive à destination en plus de suivre sa position sur la carte.
10. un utilisateur bloqué sera sanctionné d’une durée de 6mois.

**Proposition de réalisation de la solution**

Nous définissons l’environnement de développementet les outils de collaboratifs de travail en étroite relation avec l’entreprise d’accueil et les besoins exprimés si dessue.

Afin de répondre aux besoins de l’entreprise et de réalisation de l’application répondant aux spécifications fonctionnelle détaillée dans ce document, nous nous proposons de réaliser la solution à l’aide des technologies suivantes :

*Définition de l’environnement de développement*

1. Flutter : Framework permettant de développer des applications cross\_plattforme pour mobile Android et iOS.
2. Android Studiot : IDE
3. Firebase:
4. Google Cloud Platform :
5. Trello : outil de gestion de projet collaboratif en ligne
6. GitHub et Bitbucket : service d'hébergement de référentiel de code source basé sur Git.

**Gestion du projet Taxi\_Application.**

Nous adoptons une approche de développent itérative pour la création des fonctionnalités en langage de programmation DART et avec le Framework Flutter définie au préalable comme technologie de réalisation de l’application.

Le suivi des tâches du projet est mis en œuvre en fonction de la procédure de démarche qualité de qui consiste en, tout développent de fonctionnalités doit être suivit de test de validation.

Nous utilisons Trello comme outil de gestion de projet collaboratif en ligne qui mous a permis Planifier et suivre les tâches de développement d'une application selon la méthode Kanban.

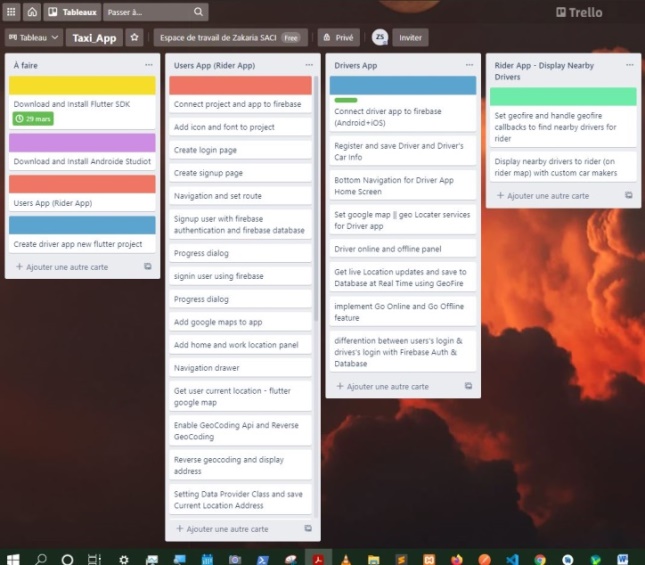
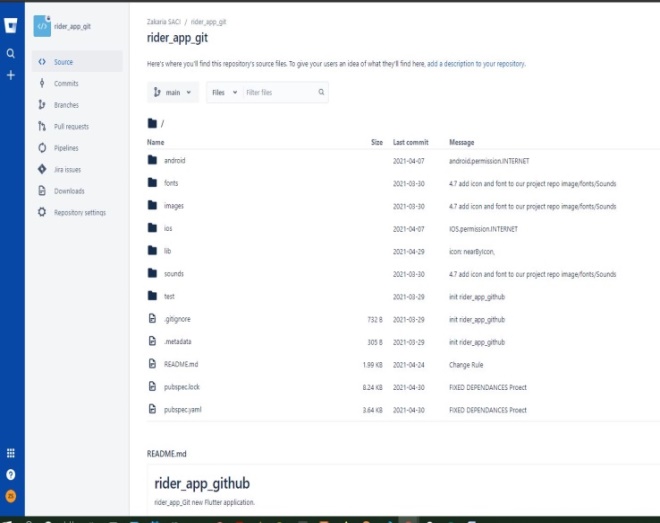
La méthode Kanban

Il s’agit d’une méthode agile très utilisée des services informatiques pour la gestion du développement et du déploiement de logiciels.

Cette méthode permet de visualiser le flux de travail, de limiter le nombre de tâches en cours et de prioriser les tâches.

logiciels centralisés :

Un serveur conserve les anciennes versions des fichiers et les développeurs s’y connectent pour prendre connaissance des fichiers qui ont été modifiés par d’autres personnes et pour y envoyer leurs modifications.



**Phase de Conception** :

Cette partie sera consacrée pour la conception de notre système, on utilisera le **diagramme d’activité**, les **diagrammes de séquences** pour représenter les scenarios, ainsi que le **diagramme de classe** pour représenter les objets manipulés par les utilisateurs, et enfin le **diagramme de déploiement** qui représente l’architecture de notre système.

**Diagramme d’activités**

La figure représente le diagramme global d’activités du système.

Authentification/Enregistrement

Affiche Google Map

Affiché Position Actuelle

Affiché Proposition de déplacement

Indiqué le Prix / Distance de déplacement

Affiché disponibilité Taxieur à proximité

Demande de réservation

Déclaré disponibilité pour missions

FIGURE - Diagramme global d’activités.

**Diagramme de cas d’utilisation**

La figure représente le diagramme de cas d’utilisation global de notre système.

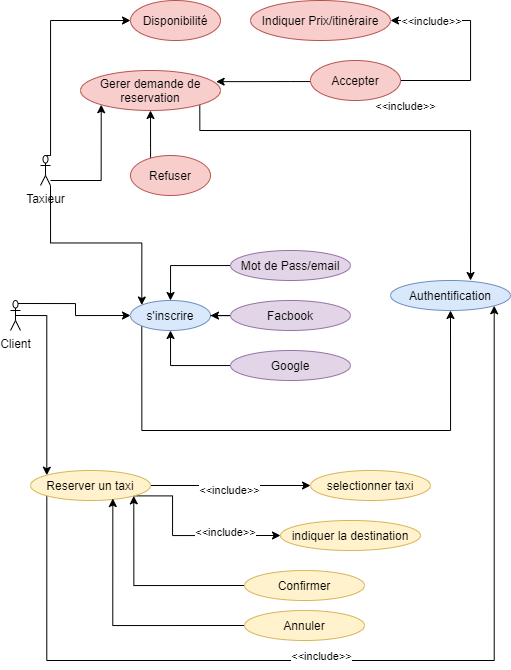


FIGURE – Diagramme de cas d’utilisation global.

**Diagramme de classe**

**Diagramme de déploiement :**

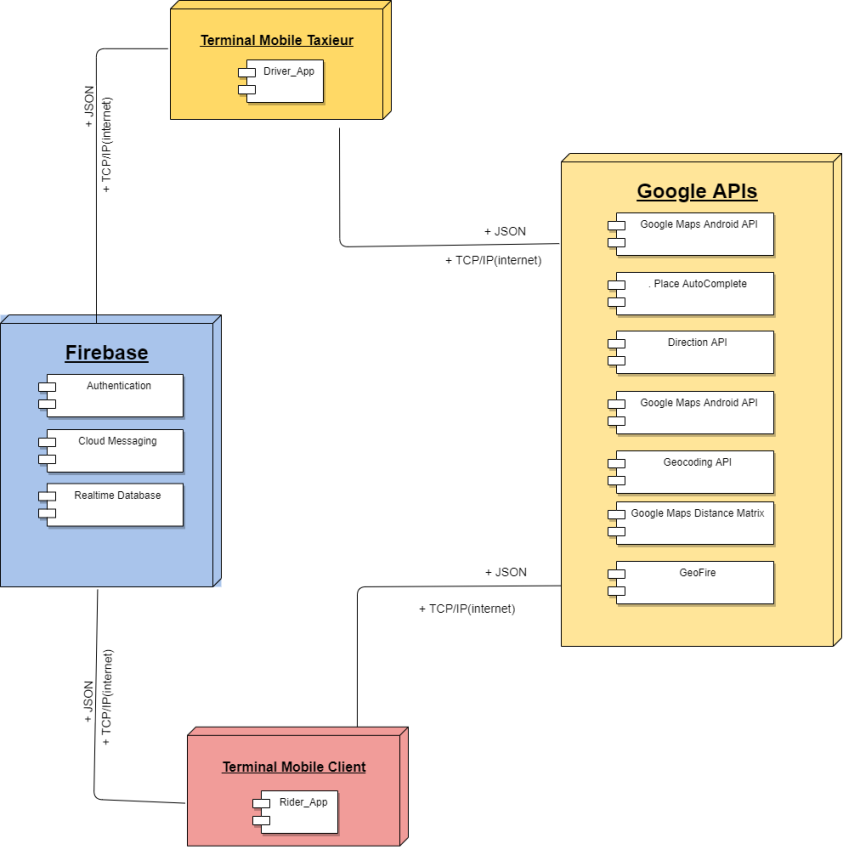
****

FIGURE – Diagramme de déploiement

Le diagramme de déploiement illustré dans la figure démontre le matériel nécessaire qu’on a utilisé dans notre projet et qui représente un terminal mobile, en plus des différentes API utilisés et des services de firebase auxquels on a eu recours, sans oublier l’utilisation de l’internet qui a été indispensable, de plus les échanges de données se sont fait en format JSON.

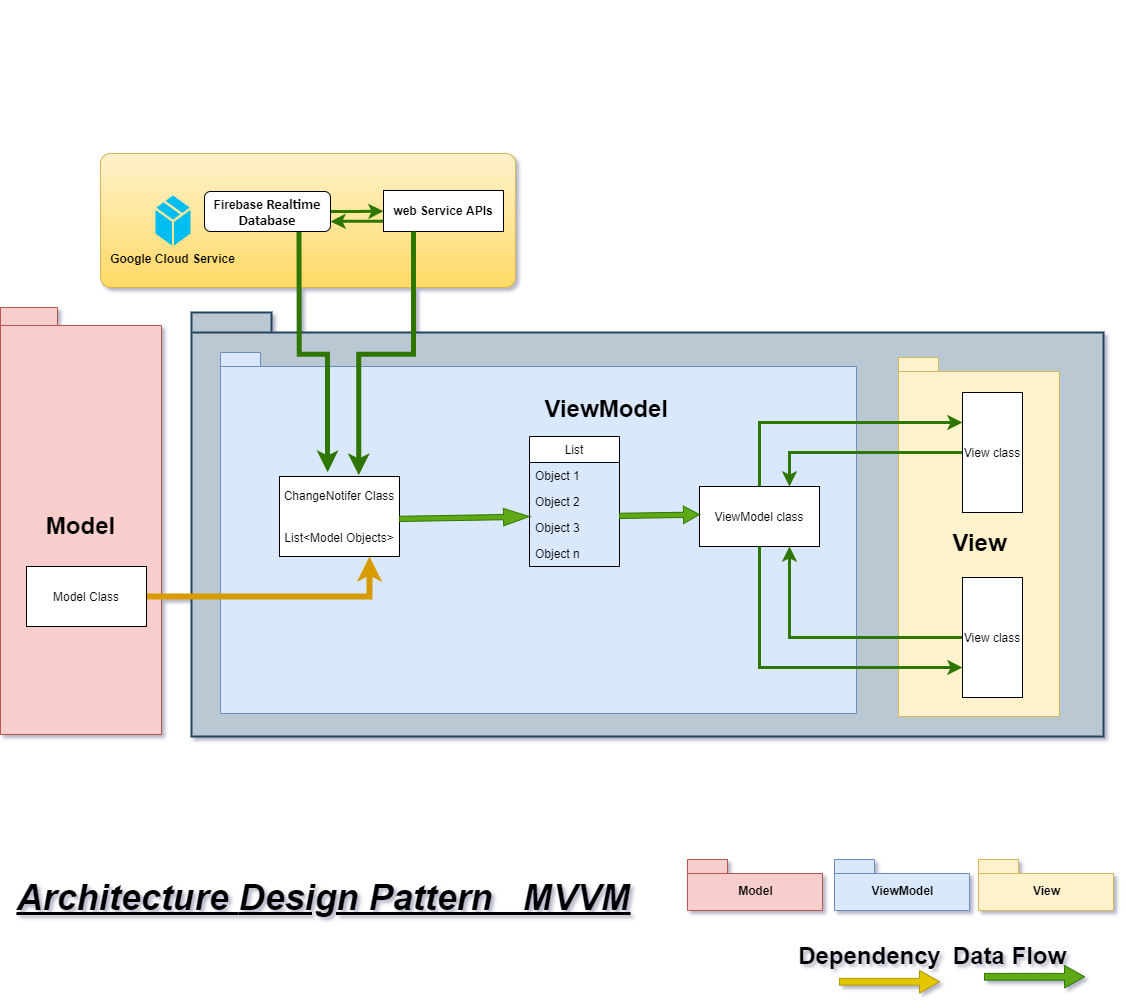
**Patrons de conceptions**

Flutter est un framework multiplateforme qui nous permet d'écrire des applications iOS et Android à l'aide d’un seul code source.

MVVM prend en charge une approche événementielle, qui va de pair avec l’architecture Fluttre car de nombreux composants de flutter sont exécutés en fonction d'événements.

**MVVM** est utile pour déplacer la logique métier de la vue vers ViewModel et Model, ViewModel est le médiateur entre View et Model qui transporte tous les événements utilisateur et renvoie le résultat.

Il y a trois éléments clés qui découlent de l'application de MVVM :

* **Maintenabilité** : La couche de présentation et la logique sont faiblement couplées, car ce code est facilement maintenable et réutilisable. Comme la base de code augmentera au fil du temps, cela nous aidera à les distinguer.
* **Testabilité** : Le ViewModel est plus facile à tester unitairement grâce à la logique de séparation MVVM.
* **Extensibilité** : - Cette architecture vous donne l'assurance qui permet au code d'être extensible au fil du temps.

**Firebase comme SaaS (*Software as a Service*)**

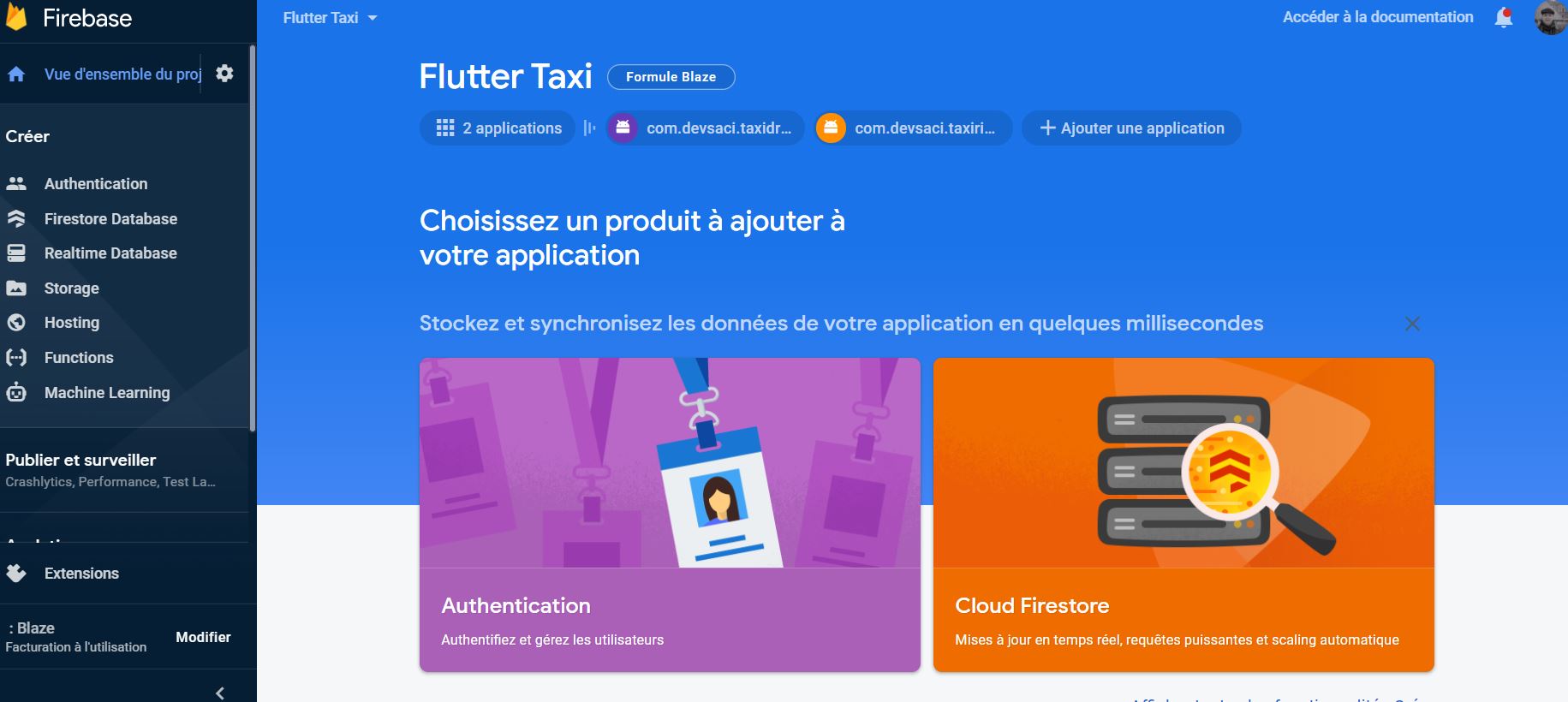
Firebase dispose de 2 types de bases de données :

1. Base de données en temps réel
2. Cloud Firestore

L'une des différences les plus importantes entre elles est le prix.

1ère facturation DB basée sur le trafic, 2ème DB dépend du nombre de lectures/écritures.

Firebase propose un hébergement statique, c'est très simple et gratuit. Nous pouvons stocker jusqu'à 1 Go gratuitement.

Pour les applications à page unique, c'est comme un hébergement illimité. De plus, l'hébergement Firebase fournira un certificat SSL gratuit avec mise à jour automatique.

**Software Development Kit (SDK)**

Firebase fournit un grand nombre de SDK nécessaires à la création d'une application Web ou mobile. Il existe deux types de SDK.

**Côté client**, qui peut être utilisé en toute sécurité sur l'interface utilisateur. Nous pouvons restreindre l'accès à certains utilisateurs et ils ne seront pas autorisés à accéder aux données protégées en lecture des bases de données. Le SDK côté client offre tout ce dont nous avons besoin : pour l’authentification des utilisateurs, accès aux bases de données, mises à jour en temps réel, téléchargement de fichiers, notifications push…

**Côté back-end** . C'est admin SDK, qui a toujours un accès complet à toutes les données de notre projet.

L’objectif premier de Firebase est de nous libérer de la complexité de création et de la maintenance d’une architecture serveur, tout en vous garantissant une scalabilité à toute épreuve et une simplicité dans l’utilisation.

Lorsqu’on développe une application, qu’elle soit destinée au grand public ou réservée à un usage interne à l’entreprise, certaines fonctionnalités sont systématiquement requises, telles que la gestion des utilisateurs, de la connexion et des notifications.

La gestion de ces fonctionnalités est fastidieuse, répétitive vu que notre SI se compose de deux applications, et critiques en termes de sécurité, dans la mesure où l’on va stocker des mots de passe.

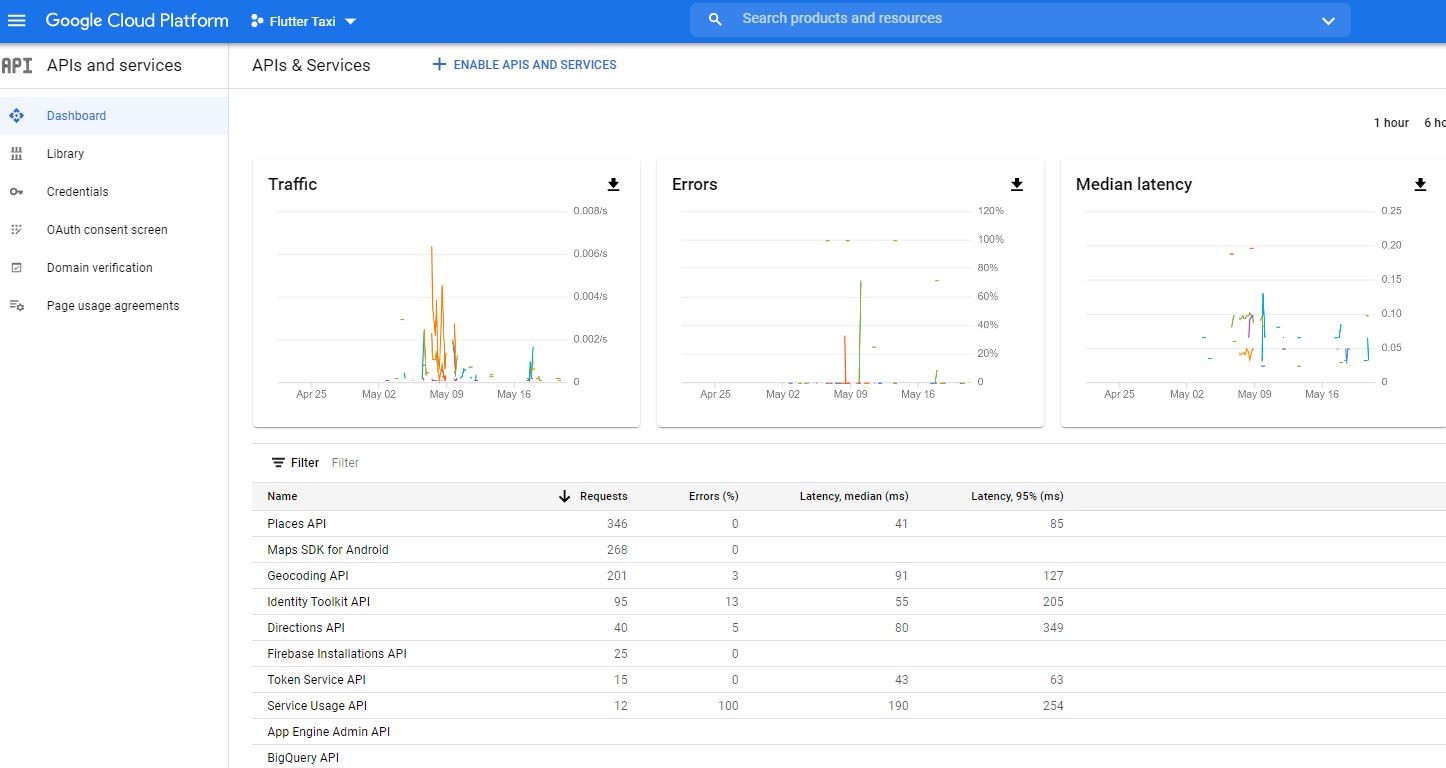
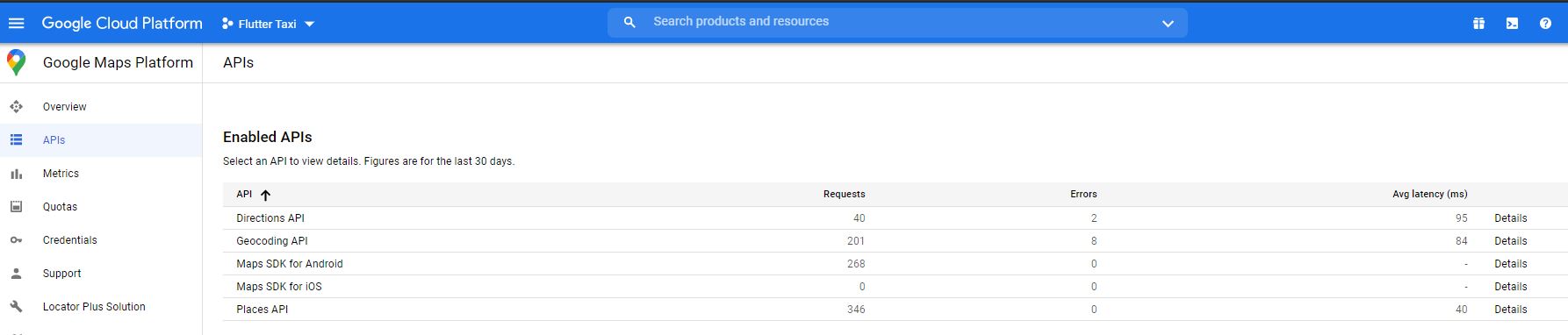
**A. Les services Firebase utilisés :**

1. **Realtime Database** : La base de données Firebase Realtime Database est une base de données
2. NoSQL hébergée dans le cloud qui vous permet de stocker et de synchroniser des données entre
3. nos utilisateurs en temps réel.
4. **Authentication** : Solution permettant de créer et gérer facilement des moyens d’authentification variés (Google, Facebook, Email, etc...) dans le but de sécuriser l’accès à une application mobile et authentifier les utilisateurs.
5. **Cloud Messaging :** Fournit un flux de communication fiable et économe en batterie entre le serveur (Firebase) et les appareils distants (où l’application est installée) dans l’objectif d’envoyer et recevoir des messages de notifications.

**B. Les services web utilisés :**

Dans le cadre de notre application, nous avons utilisé quelques services web existants.

1. **Google Maps Distance Matrix :** Le service Google Maps Distance Matrix API fournit les distances et les durées des trajets pour une matrice de points de départ et de destinations.
2. **Place AutoComplete :** Le service Place Autocomplete (saisie-semi-automatique de lieu) est un service Web qui renvoie des prédictions de lieu en réponse à une requête HTTP. La requête spécifie une chaîne de recherche textuelle et des limites géographiques facultatives. Le service permet de fournir la fonctionnalité de saisie semiautomatique pour les recherches géographiques textuelles et renvoie des lieux tels que des entreprises, des adresses et des points d’intérêt au fur et à mesure de la saisie par l’utilisateur.
3. **Google Maps Android API :** un service web qui sert à ajouter Google Maps à notre application Flutter.
4. **Direction API :** Google Maps Directions API est un service qui calcule des itinéraires entre des points géographiques.
5. **Geocoding API :** Google Maps Geocoding API est un service qui effectue le géocodage et le géocodage inversé d’adresses.
6. ***Géocodage :*** est le processus qui permet de convertir des adresses (comme une adresse postale) en coordonnées géographiques (comme la latitude et la longitude) que vous pouvez ensuite utiliser pour placer des marqueurs sur une carte ou pour positionner la carte.
7. ***Géocodage inversé :*** est le processus de conversion de coordonnées géographiques en adresses lisibles. Le service de géocodage inversé de Google Maps Geocoding API vous permet également de retrouver l’adresse correspondant à un identifiant de lieu donné
8. **GeoFire :** ensemble de bibliothèques open-source nous permettent de stocker et interroger un ensemble de clés en fonction de leur emplacement géographique.
9. GeoFire utilise la base de données Firebase pour le stockage des données, ce qui permet de mettre à jour les résultats de la requête en temps réel à mesure qu’ils changent.
10. GeoFire charge de manière sélective uniquement les données à proximité de certains emplacements, en gardant l’applications légères et réactives, même avec des ensembles de données extrêmement volumineux**.**

****

**Phase de réalisation**

Application Client/Taxieur (Rider/Driver) :

1. Connecté le projet Taxi\_App à Firebase (connect project and app to firebase)

Connecté l’application à firebase : firebase\_core: ^0.5.0+1

Authentification par mail et password : firebase\_auth: ^0.18.1+2

Stockage Firebase : firebase\_database: ^4.1.1

**Fonctionnalités Application Client/Taxieur :**

1. Code de navigation et routage

2. Authentification.

3. Inscription (Registration)

3. S’inscrire via firebase authentication sur firebase database.

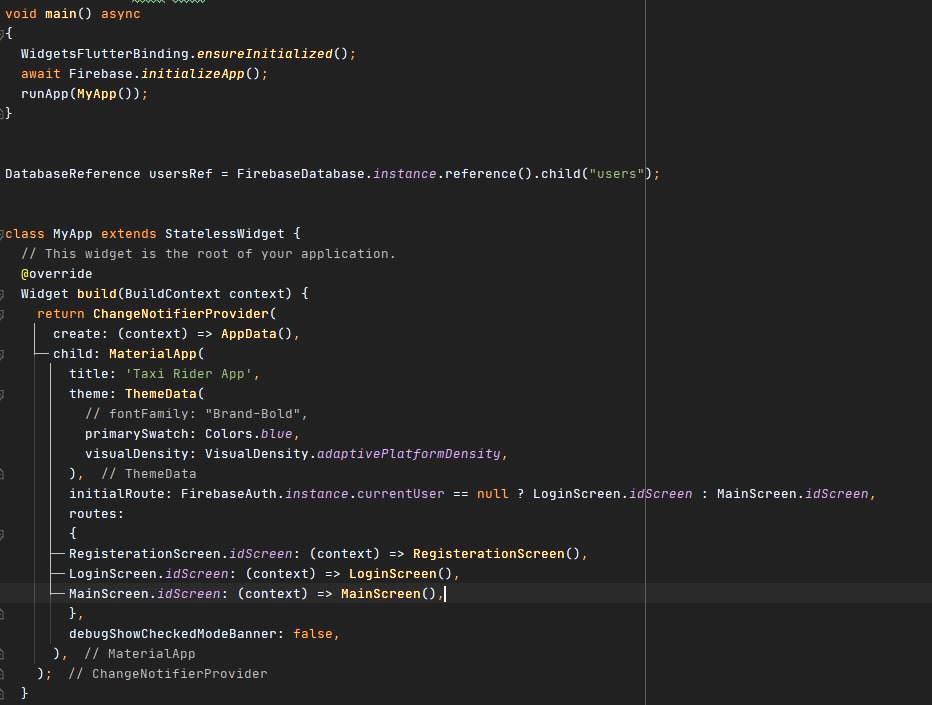
4. Affiché Google Maps.

5. Affiché La localisation Domicile / Travail (Positioned : Panel Search Ui).

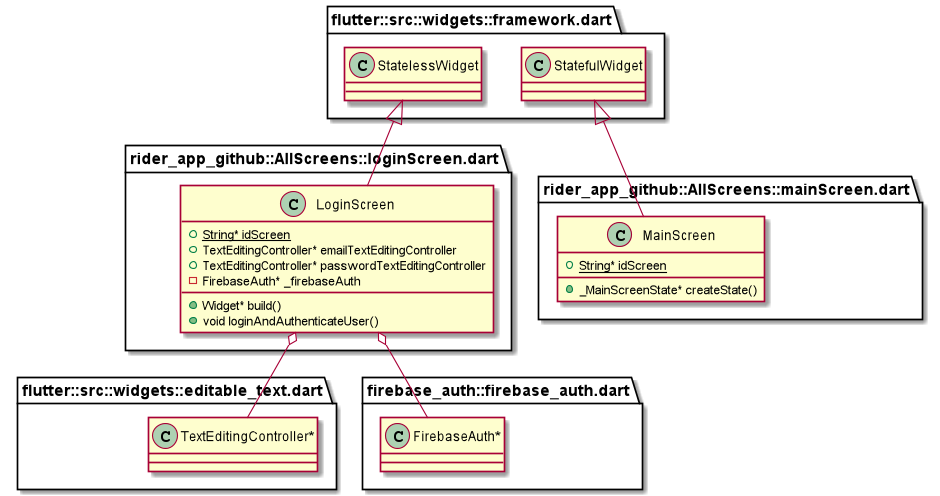
6. Drawer pour la navigation entre les vues

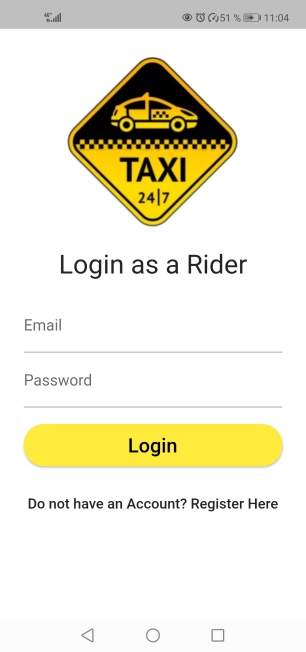
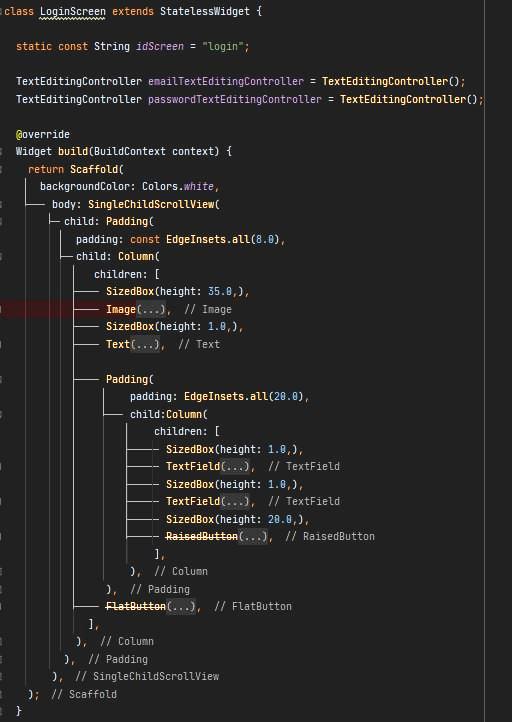
7. Affichage de la localisation actuelle de l’utilisateur.

**1. Code de navigation et routage**



1. **Authentification (create login page)**



void loginAndAuthenticateUser(BuildContext context) async

**Sécurité**

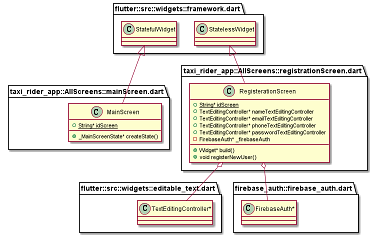
***plug-in de connexion Google pour l'authentification ( email, Password)***

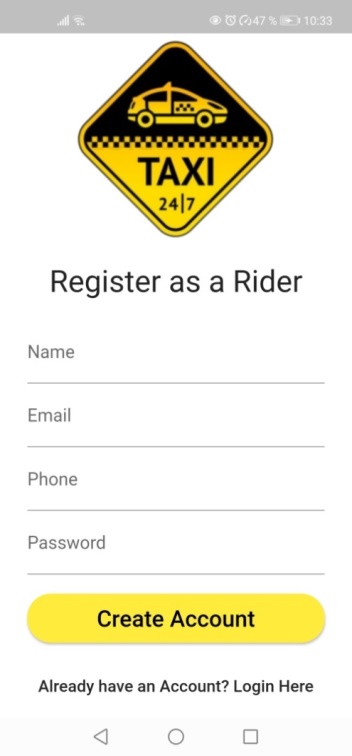
***firebase\_auth: ^0.18.1+2.***

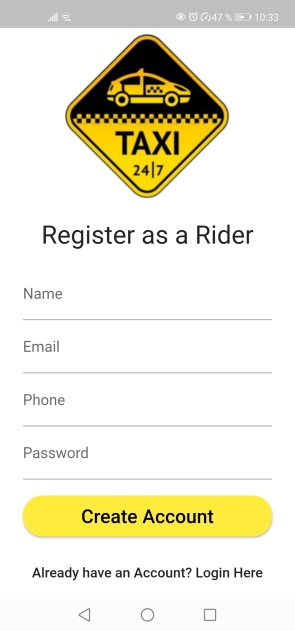
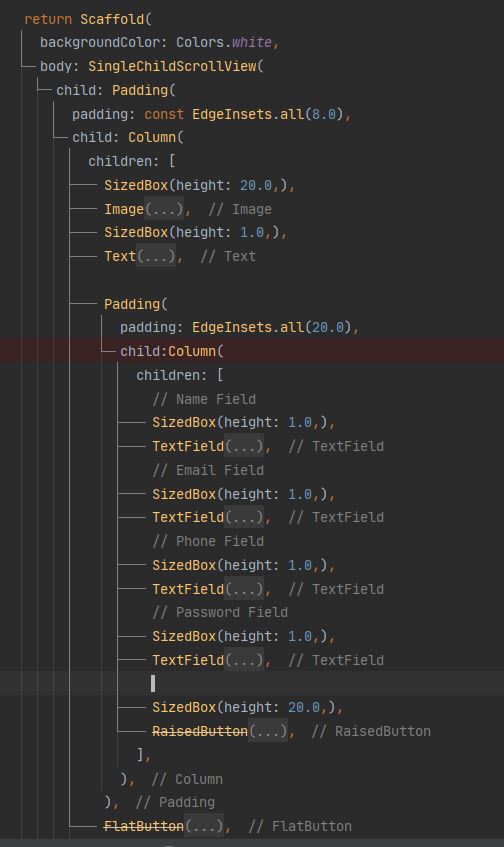


**3. S’inscrire (Enregistrement)**

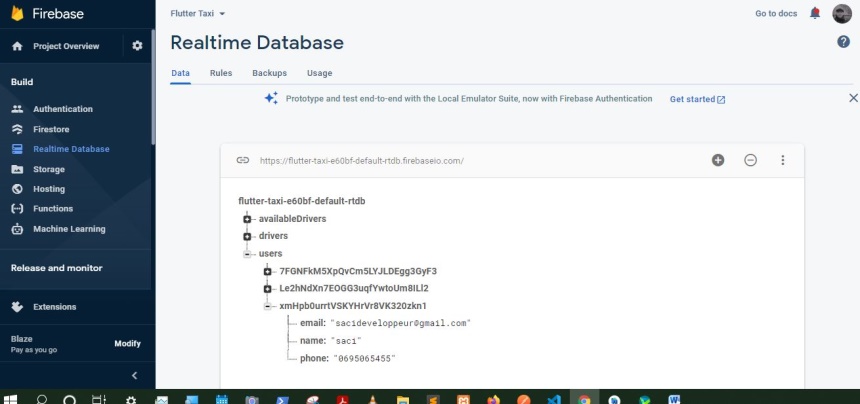
Le diagramme de Classe de la fonctionnalité s’Enregistré permet de décrire les relations en la vue Mainscreen, la Classe RegistrationScreen et les Classe implémenté en agrégation en relation avec la base de donnée (Firebase Auth) et le contrôleur permettant la gestion des imputs d’enregistrement des utilisateurs.

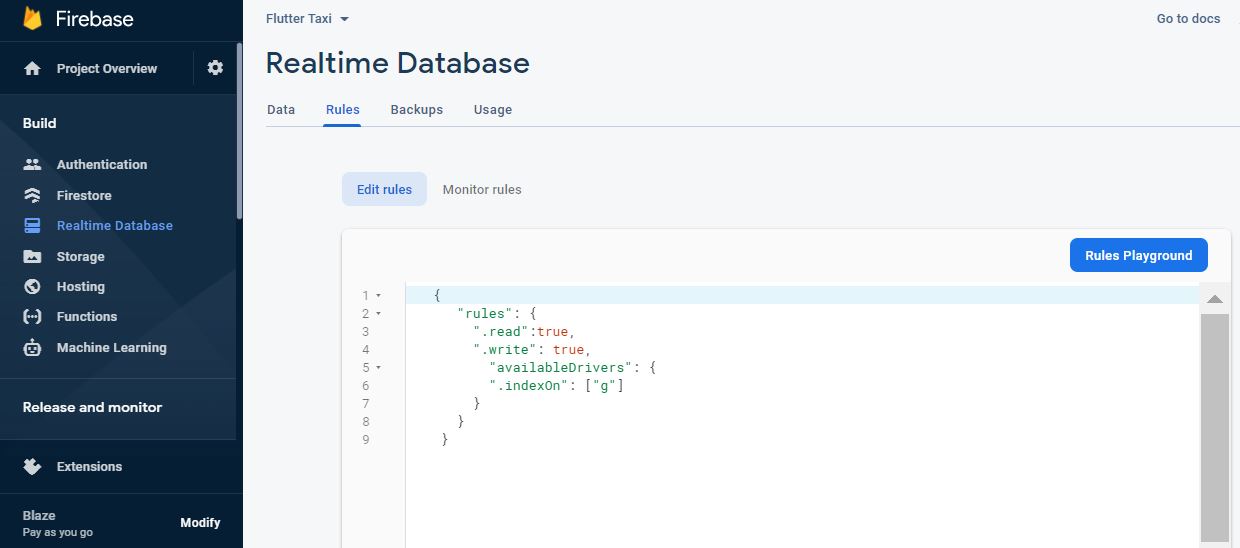


Le passage de la charte graphique produit sur diagrams.net a la vue sur mobile à Androïde Sys avec 6.8 pouces d’écran est représenté ci-dessous



**4. Signup user with firebase authentication and firebase database.**

****

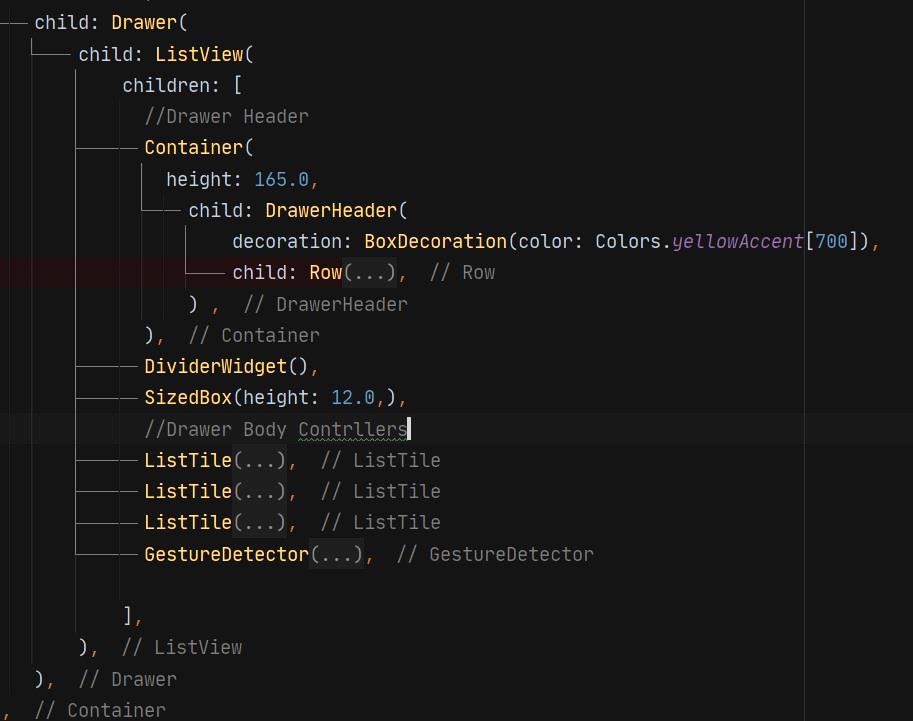
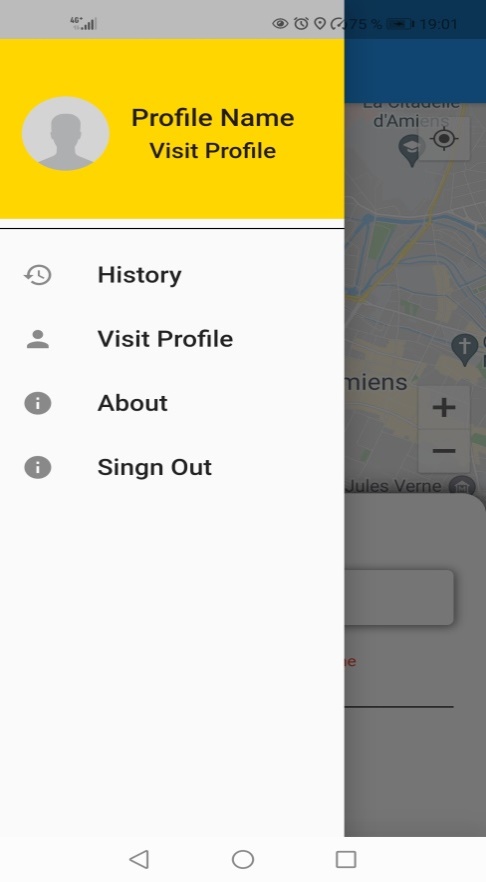


**5. Intégré Google Map à la vue mainScreen**

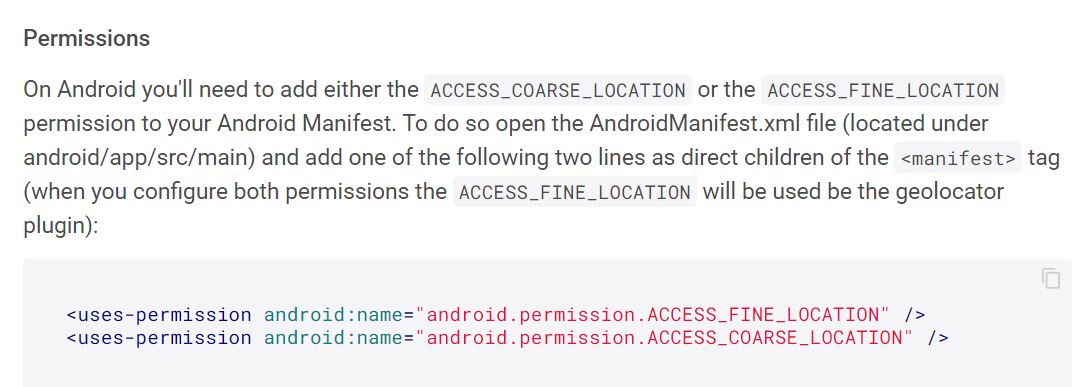
**google\_maps\_flutter: ^1.0.3**

**6. Navigation avec le widget drawer**

La traduction de la charte graphique du drawer en code donne :

****

**7.1 Autorisation d’accès à la position géographique.**

****

**L’application doit demander l’autorisation avant de permettre l’accées à la position actuelle de l’utilisateur. Pour cela, mous devons configurer le fichier androideManifest.xml.**

**7.2 Affichage de la localisation actuelle en temps réel**

**8. Activation de l’API GeoCoding Api et Reverse GeoCoding**

**(**enable GeoCoding Api and Reverse GeoCoding)

Activation du service facturation.

Format de requête d'API de géocodage

Une requête API de géocodage prend la forme suivante :

[**https://maps.googleapis.com/maps/api/geocode/*outputFormat*?*parameters***](https://maps.googleapis.com/maps/api/geocode/outputFormat?parameters)

**https://maps.googleapis.com/maps/api/geocode/json?latlng=49.894067,2.295753&key= YOUR\_API\_KEY**

Le format json(recommandé) indique une sortie en JavaScript Object Notation (JSON) ;

HTTPS est requis pour les demandes qui utilisent une clé API et recommandé pour les demandes qui utilisent un ID client. HTTPS est également requis pour les applications qui incluent des données utilisateur sensibles - telles que l'emplacement d'un utilisateur - dans les demandes.

**9. Affichage de l’adresse géolocalisée**

**(reverse geocoding and display address) :**

Plugin http: ^0.12.2

**class RequestAssistant**

**static Future<dynamic> getRequest(String url) async**

**class AssistantMethods**

**static Future<String> searchCoordinateAddress(Position position, context) async**

**String url = «**[**https://maps.googleapis.com/maps/api/geocode/json?latlng=${position.latitude},${position.longitude}&key=$mapKey**](https://maps.googleapis.com/maps/api/geocode/json?latlng=$%7bposition.latitude%7d,$%7bposition.longitude%7d&key=$mapKey)**»**

**10. setting Data Provider Class and save Current Location Address**

**Publier l’application sur Play Store**

**1. Ajout d'une icône de lanceur**

**2. Signer l'application**

Génération de clé Android Studio sous Windows

a. Commande

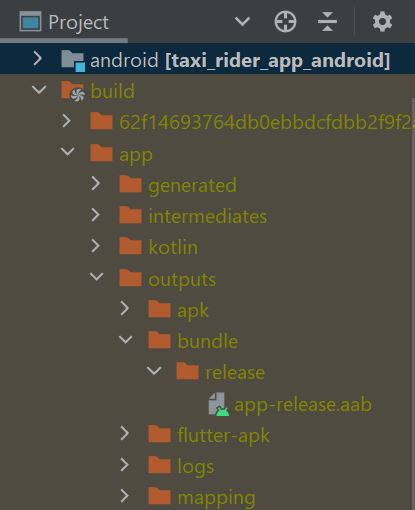
keytool -genkey -v -keystore C:\Users\Zakaria\AndroidStudioProjects\TaxiAppProject\upload-keystore.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 10000 -alias upload

b. Création des Fichiers upload-keystore.jks et key.properties.

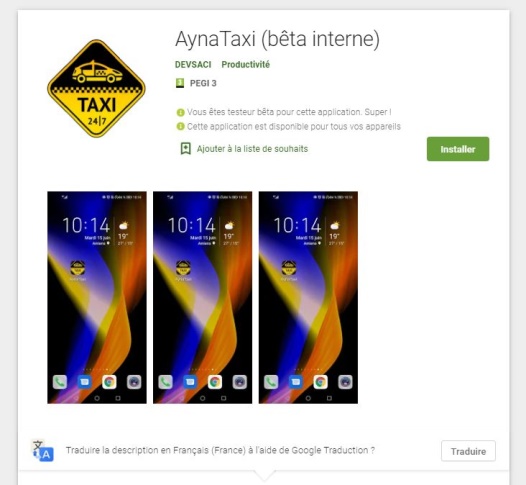
c. Configuration de build.gradle in app.

d. Stoper le Revers-Ingennering avec R8 activé par défaut lors de la création App bundle.

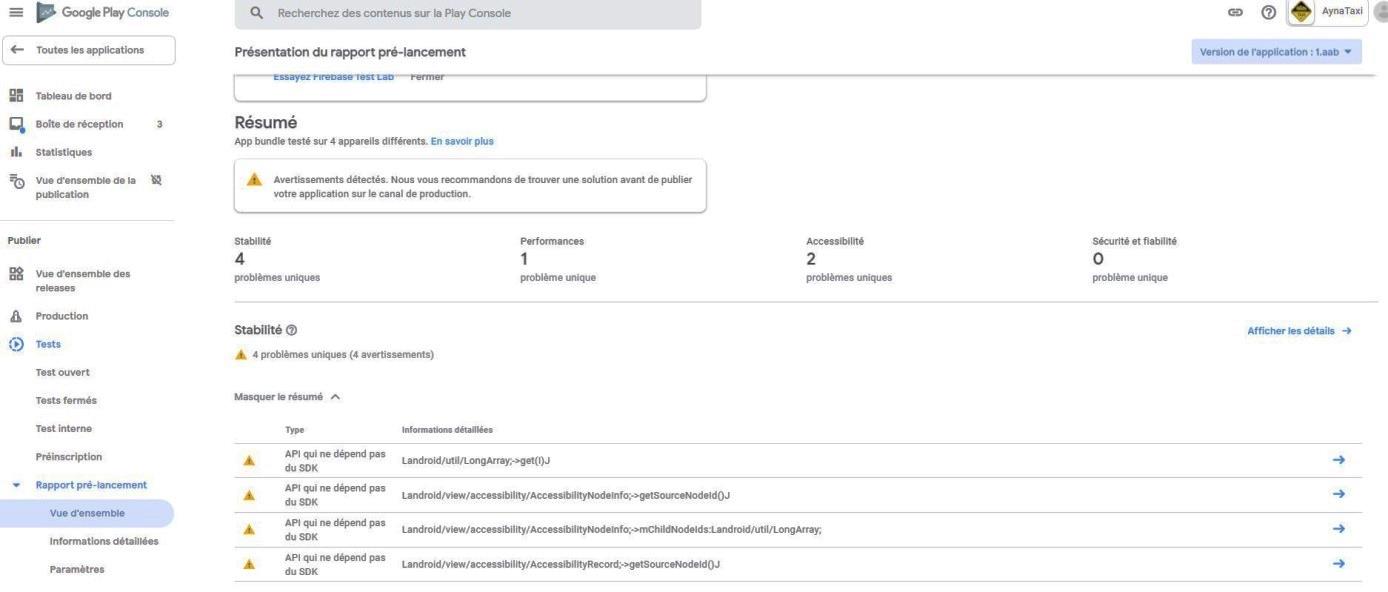
**3. Construire Appbundle** avec la commande flutter build appbundle.



**La publication de l’application sur google Store en version teste permet de vérifier les fonctionnalités.**

****

En téléchargeant AynaTaxi sur mobile Android, nous remarquons que les tests réalisés sur google store ont relevés quartes problèmes de fonctionnalités.

****

Ces dysfonctionnements n’apparaissent pas lors des tests de fonctionnalités de l’application sur mobile connecté puis découplé de Android Studiot.

Bibliographie

JOSEF GABAY, DAVID GABAY, « UML2 Analyse et Conception », Université de Québec, 1re édition, 2009.

PASCAL ROQUES, «Les cahiers du programmeurs UML2 modéliser une application web », EYROLLES, 4e édition, 2008.

RUDI BRUCHEZ, « Les base de données NoSQL et le Big Data comprendre et mettre en œuvre », EYROLLES, 2e édition, 2015.

LAURENT DEBRAUWER, « Design patterns en java », ENI, 3e édition, 2013.

Site Web bibliographie : Consulté Mars à Mai 2021

<https://flutter.dev/>

<https://dart.dev/>

<https://developers.google.com>

<https://console.firebase.google.com>

<https://pub.dev/>

<https://console.cloud.google.com/>

<https://play.google.com/>

<https://levelup.gitconnected.com/saas-with-firebase>