

## PROJET DE middlewares

### DÉVELOPPEMENT D'UNE APPLICATION MOBILE « Smart Wait »



#### ENCADRÉ PAR :

- Pr. HAQIQ ABDELAHAY



#### RÉALISÉ PAR :

- OUSSAKEL KHADIJA
- GHRIB Mohamed Reda



# INTRODUCTION

Dans notre société moderne, nous sommes constamment en train de courir après le temps et de chercher des moyens plus efficaces de gérer notre vie quotidienne. C'est dans cette optique que nous avons développé une application mobile innovante qui vous permettra de gérer votre temps de manière encore plus efficace en vous permettant de gérer les files d'attente dans les agences et les administrations de manière simple et rapide.

Avec notre application, plus besoin de passer des heures à attendre dans une file interminable pour régler vos formalités administratives ou pour obtenir un service dans une agence. Grâce à notre système de gestion des files d'attente, vous pourrez désormais prendre rendez-vous en ligne et prendre un ticket virtuel avant de se rendre physiquement dans l'agence ou l'administration concernée, ce qui vous permettra de planifier votre temps de manière plus efficace et de réaliser vos tâches de manière plus rapide.

En utilisant notre application, vous gagnerez du temps et vous aurez l'esprit tranquille, sachant que vous n'aurez plus à vous soucier de passer des heures à attendre dans une file d'attente. Découvrez dans ce rapport comment notre application mobile de gestion de files d'attente peut vous aider à optimiser votre temps et à améliorer votre qualité de vie.

En plus de rendre l'expérience des utilisateurs plus agréable, notre application permet également aux agences et aux administrations de gérer leur flux de personnes de manière plus efficace, en leur permettant de savoir combien de personnes sont en attente et de les aiguiller vers les guichets les moins chargés.



# Table des matières :

I.	Périmètre du projet : .....	4
A.	Problématique : .....	4
B.	Solution : .....	4
II.	Planification du projet : .....	5
A.	Les différentes tâches et leur répartition : .....	5
B.	Suivi du déroulement du projet sur Trello : .....	5
C.	Gestion des versions du code source par Git : .....	7
D.	Choix des outils et des technologies : .....	7
III.	La conception du Smart Wait : .....	10
A.	Diagramme de cas d'utilisation : .....	10
B.	Élaboration de la conception de la base de données : .....	10
C.	Le logo de l'application : .....	12
D.	Élaboration de la conception de l'interface utilisateur : .....	12
IV.	Détection des personnes : .....	14
A.	Machine Learning et détection des objets : .....	14
B.	La théorie du modèle de détection : .....	16
1.	Détecteur SSD : .....	16
2.	Traqueur centroïde : .....	16
C.	Modes d'utilisations : .....	17
V.	Implémentation de l'application mobile : .....	19
A.	Les fonctionnalités de l'App : .....	19
B.	Le backend de l'application : .....	29
1.	Authentification : .....	29
2.	Réservation : .....	29
VI.	Docker et DockerHub : .....	30
A.	Création d'une image docker : .....	30
B.	Déploiement de l'image docker dans Docker Hub : .....	32

## I. Périmètre du projet :

### A. Problématique :

Les files d'attente dans les agences bancaires et les administrations peuvent être un véritable casse-tête pour les personnes qui doivent s'y rendre. Elles peuvent être longues et fastidieuses, et il n'est pas rare de passer de nombreuses heures à attendre son tour. Cela peut être particulièrement frustrant lorsque l'on a des obligations importantes et que l'on doit perdre du temps précieux à attendre. De plus, les files d'attente peuvent être coûteuses pour les entreprises et les organisations, car elles nécessitent souvent des employés supplémentaires pour gérer la demande de service.

La problématique des files d'attente est donc un véritable problème pour de nombreuses personnes, et il est important de trouver des solutions pour l'éviter au maximum.

### B. Solution :

Pour résoudre cette problématique, nous proposons Smart Wait, une application mobile de rendez-vous en ligne et de réservation de tickets. Smart Wait est la solution idéale pour vous : vous pouvez prendre rendez-vous en ligne à l'heure qui vous convient le mieux, sans avoir à passer des heures à attendre votre tour. De plus, vous pouvez également réserver un ticket en ligne, ce qui vous permet de savoir à quelle heure vous devez vous présenter et de ne pas perdre de temps à attendre. Elle est facile à utiliser et vous permet de gagner du temps et de l'énergie, tout en vous offrant la possibilité de planifier votre journée de manière efficace. En somme, Smart Wait est une solution idéale pour éviter les files d'attente et vous permettre de profiter de votre temps libre.

## II. Planification du projet :

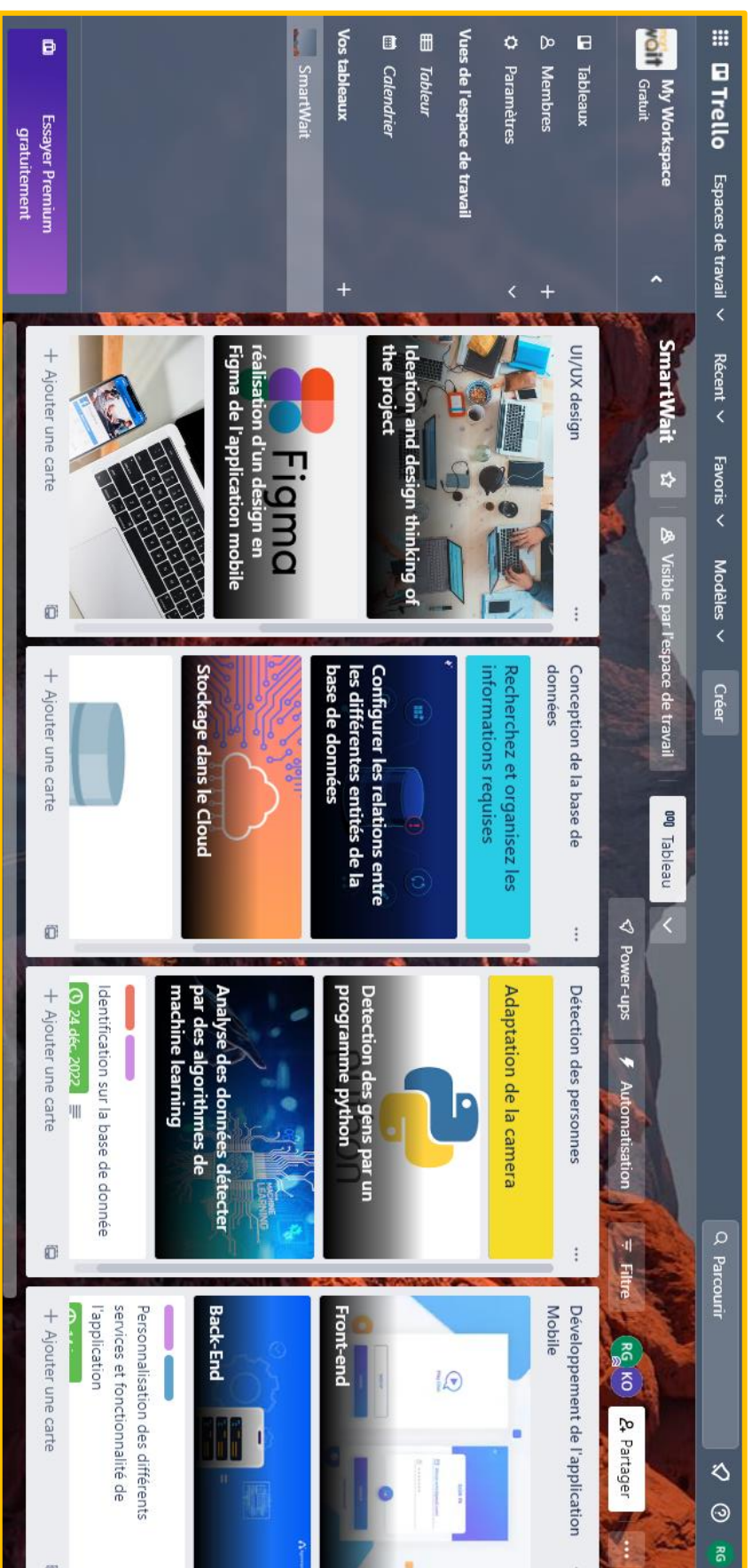
### A. Les différentes tâches et leur répartition :

Création du logo	OUSSAKEL Khadija
Réalisation des maquettes	OUSSAKEL Khadija
Conception de la base de données	OUSSAKEL Khadija -GHRIB Mohamed Reda
Détection des personnes	Ghrib Mohamed Reda
Développement de l'application mobile	OUSSAKEL Khadija
Docker et DockerHub	Ghrib Mohamed Reda

### B. Suivi du déroulement du projet sur Trello :

Pour mieux gérer notre projet, nous avons choisi l'outil Trello qui permet de planifier et de suivre l'avancement des projets. Il utilise des "tableaux" pour représenter différentes parties du projet et permet de créer des "cartes" pour chaque tâche ou élément de travail à accomplir. Il permet aussi d'ajouter des détails, des commentaires et des pièces jointes à chaque carte, et déplacer les cartes d'un tableau à l'autre pour refléter l'avancement de la tâche. Trello est un outil très visuel qui peut être utilisé de manière flexible pour différents types de projets et de travaux de planification.








### C. Gestion des versions du code source par Git :

Git un outil de versioning pour gérer les versions du code source. C'est un système de contrôle de version de code source distribué qui permet aux équipes de développement de suivre les modifications apportées au code et de gérer les versions de manière efficace. Avec Git, les développeurs peuvent travailler sur le même code source en même temps, en utilisant des branches pour isoler les différentes modifications. Ils peuvent ensuite fusionner leur travail avec la version principale du code lorsqu'ils sont prêts à le livrer. Git est largement utilisé dans l'industrie du développement logiciel et est disponible gratuitement.






### D. Choix des outils et des technologies :

Après que nous avons évalué les différents besoins du projet et nous avons déterminé quels langages et les technologies seront les plus appropriés pour répondre à ces besoins :

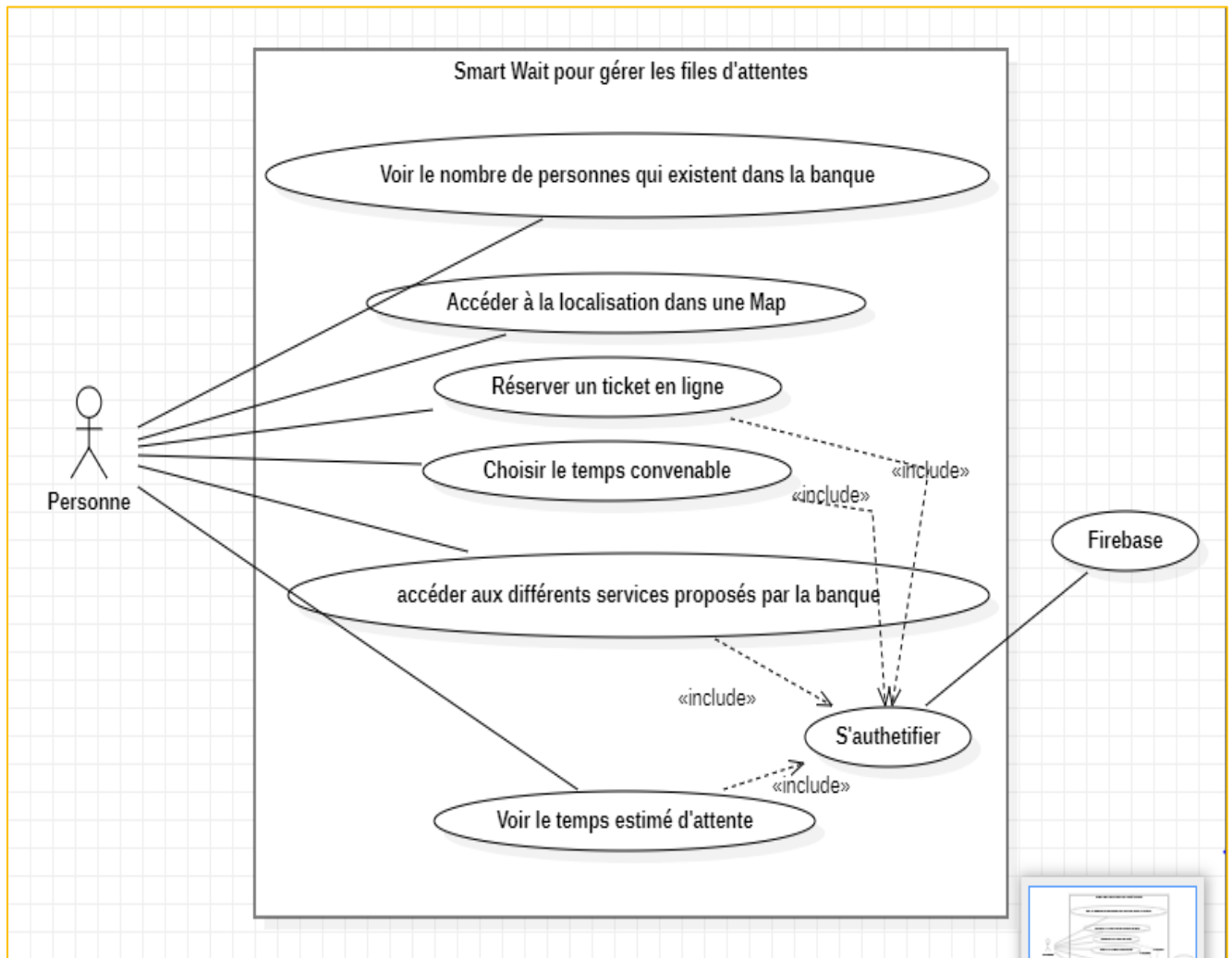
 Flutter	<p>Flutter : un Framework open-source de Google qui permet de développer des applications mobiles natives pour les plateformes iOS et Android à l'aide de Dart, un langage de programmation développé par Google.</p>
 Firebase	<p>Firebase offre un stockage en temps réel de données qui permet aux applications de stocker et de synchroniser les données en temps réel avec tous les appareils connectés. Elle offre des services d'authentification qui permettent aux applications de gérer les comptes utilisateur de manière simple et sécurisée.</p>
 python <sup>TM</sup>	<p>Python est un langage de programmation populaire qui est souvent utilisé dans le domaine du Machine Learning. Il est apprécié pour sa syntaxe simple et conviviale.</p>



	<p>OpenCV (Open Computer Vision) est une bibliothèque de traitement d'images et de vision par ordinateur open-source qui est largement utilisée dans le domaine du Machine Learning. Elle offre une large gamme d'algorithmes et de fonctionnalités pour la détection d'objets, la reconnaissance de visages, l'analyse de mouvement et bien plus encore.</p>
	<p>dlib est une bibliothèque open-source de Machine Learning et de traitement d'images qui est largement utilisée pour la détection et la reconnaissance de visages. Elle offre un ensemble d'algorithmes de pointe pour la détection de visage</p>
	<p>Docker est un outil de conteneurisation de logiciels qui permet de créer, de déployer et d'exécuter des applications de manière rapide et fiable.</p>

### III. La conception du Smart Wait :

#### A. Diagramme de cas d'utilisation :



#### B. Élaboration de la conception de la base de données :

Nous avons utilisé la base de données Firebase Realtime qui est hébergée dans le cloud. Les données sont stockées au format JSON et synchronisées en temps réel avec chaque client connecté. Contrairement à une base de données SQL, il n'y a pas de tables ou d'enregistrements. Lorsque nous ajoutons des données à l'arborescence JSON, elles deviennent un nœud dans la structure JSON existante avec une clé associée.

Nous avons donc décidé de structurer notre base de données de la façon suivante :

```
{
  "places": {
    "Banque Populaire Agdal": {
      "Camera": {
        "total_person": "10"
      },
      "adress": "X5W2+X75,Av. de France, Rabat",
      "place_name": "Banque Populaire Agdal",
      "service": {
        "Currency exchange": {
          "people_nbr": 0
        },
        "Customer service": {
          "people_nbr": 1,
          "user": {
            "kWSqmOf1kPU4RyvEnwsV5hHiAl63": {
              "Date_booking": "TimeOfDay(00:28)",
              "user_id": "kWSqmOf1kPU4RyvEnwsV5hHiAl63"
            }
          }
        }
      },
      "Loan service": {
        "people_nbr": 1,
        "user": {
          "kWSqmOf1kPU4RyvEnwsV5hHiAl63": {
            "Date_booking": "TimeOfDay(01:30)",
            "user_id": "kWSqmOf1kPU4RyvEnwsV5hHiAl63"
          }
        }
      },
      "Wire transfer": {
        "people_nbr": 0
      }
    }
  },
}
```

### C. Le logo de l'application :

A l'aide d'Adobe Illustrator, nous avons créé le logo de notre application mobile Smart Wait :

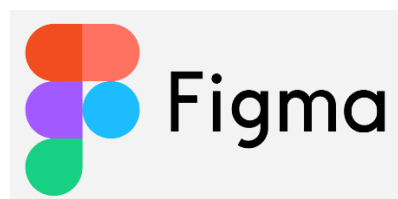


Le nom Smart Wait indique clairement le but de l'application et est facile à mémoriser. Le nom est court et simple, ce qui le rend facile à lire et à prononcer. Nous avons aussi ajouté un graphique représentant une horloge pour renforcer l'idée de gestion du temps d'attente.



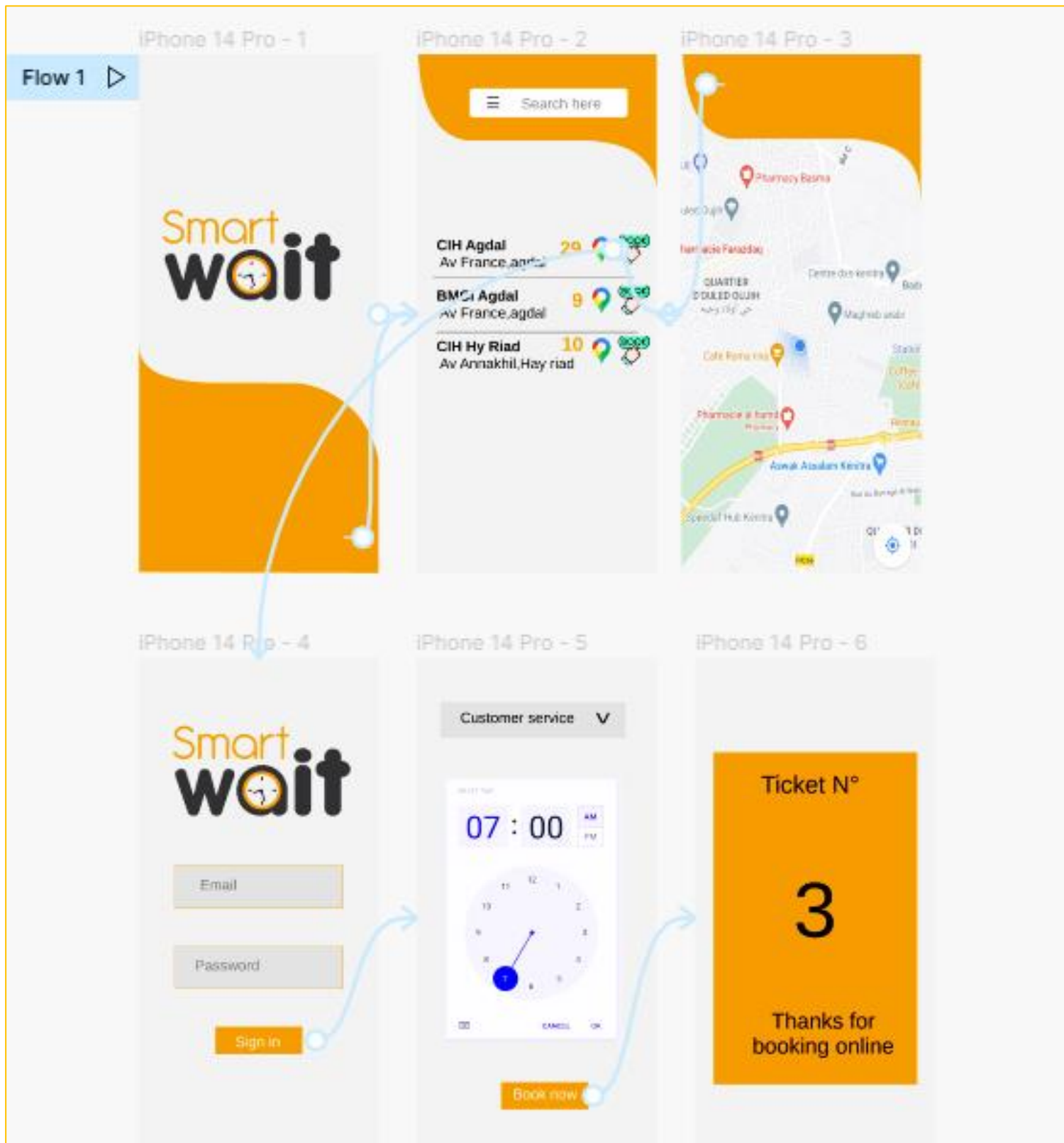
### D. Élaboration de la conception de l'interface utilisateur :

Dans cette étape, nous allons définir comment l'application sera présentée aux utilisateurs et comment ils interagiront avec elle, en utilisant des maquettes et des prototypes pour visualiser l'interface utilisateur. Nous allons réaliser ces maquettes à l'aide de Figma :



Le lien Figma :

<https://www.figma.com/file/YB5aI3NHqhXN3DV9SdtCx5/Untitled?node-id=0%3A1&t=zq9VJs36X3BoeYI5-1>



## IV. Détection des personnes :

### A. Machine Learning et détection des objets :

Dans ce projet, on a travaillé sur un programme python qui permet de détecter chaque personne qui entre dans le cadre d'une caméra vidéo en temps réel et avec une précision très élevée.

À partir d'une source vidéo, nous allons utiliser la librairie OpenCV (<https://github.com/opencv/opencv>) pour lire cette vidéo et la découper en différents frames. C'est à partir de ces frames que l'on va pouvoir détecter des personnes sur la vidéo. En effet, les modèles de machine Learning pour la détection et la classification d'objet fonctionnent sur des images issues de la vidéo. Chaque frame est alors traité par un modèle de détection développé sur TensorFlow (<https://github.com/tensorflow/tensorflow>) qui après entraînement détectera les personnes et la présence de masque sur chaque frame de la vidéo.

```
if config.Thread:
    vs = thread.ThreadingClass(config.url)

# loop over frames from the video stream
while True:

    frame = vs.read()
    frame = frame[1] if args.get("input", False) else frame

    if args["input"] is not None and frame is None:
        break

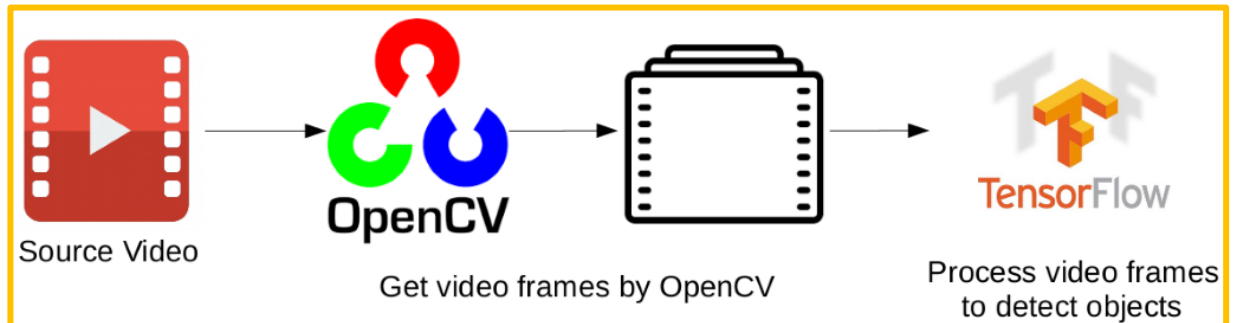
    frame = imutils.resize(frame, width = 500)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # if the frame dimensions are empty, set them
    if W is None or H is None:
        (H, W) = frame.shape[:2]

    if args["output"] is not None and writer is None:
        fourcc = cv2.VideoWriter_fourcc(*"mp4v")
        writer = cv2.VideoWriter(args["output"], fourcc, 30,
                                (W, H), True)

    status = "Waiting"
    rects = []
```

La logique de fonctionnement est indiquée sur la figure ci-dessous :



Différents modules de python ont été utilisé afin qu'on puisse importer les différentes fonctionnalités de Machine Learning (OpenCV, dlib...), ainsi de base de données, de vidéo lectures...

```
EXPLORER  ...  Run.py 3, M x
PEOPLE-COUNTING-IN-RE...
  > mobilenet_ssd
  > mylib
  > videos
  {} data.json U
  Dockerfile U
  LICENSE
  Log.csv M
  logging.csv U
  mydata.json U
  README.md
  requirements.txt M
  Run.py 3, M

1 You, 2 minutes ago | 2 authors (Sai Subhakar T and others)
2 from mylib.centroidtracker import CentroidTracker Sai Subhakar T, 2 years ago • added main
3 from mylib.trackableobject import TrackableObject
4 from imutils.video import VideoStream
5 from imutils.video import FPS
6 from mylib.mailer import Mailer
7 from mylib import config, thread
8 import time, schedule, csv
9 import numpy as np
10 import argparse, imutils
11 import time, dlib, cv2, datetime
12 from itertools import zip_longest
13 import os
14 from pprint import pprint
15 from google.cloud import storage
16 import json
17 import requests as rq
18 from flask import Flask
19 import pyrebase
20
21 import ctypes
22 firebaseConfig = {
23     "databaseURL": "https://reda-812d0-default-rtdb.europe-west1.firebaseio.com/",
24     "apiKey": "AIzaSyB0mpJborub7HRMZBno0x_3o8sp7QpCovc",
25     "authDomain": "reda-812d0.firebaseio.com",
26     "projectId": "reda-812d0",
27     "storageBucket": "reda-812d0.appspot.com",
28     "messagingSenderId": "1046303599995",
29     "appId": "1:1046303599995:web:80c3b9c40eedaf523aa1de",
30     "measurementId": "G-9ZZ5FNDBQP"
31 }
32 firebase = pyrebase.initialize_app(firebaseConfig)
33 db = firebase.database()
```

## B. La théorie du modèle de détection :

### 1. Détecteur SSD :

- Nous utilisons un SSD (Single Shot Detector) avec une architecture MobileNet. En général, il suffit d'une seule prise de vue pour détecter ce qui se trouve dans une image. C'est-à-dire, un pour générer des propositions de région, un pour détecter l'objet de chaque proposition.
- Comparé à d'autres détecteurs à 2 coups comme le R-CNN, le SSD est assez rapide.
- MobileNet, comme son nom l'indique, est un DNN conçu pour fonctionner sur des appareils à ressources limitées. Par exemple, les mobiles, les caméras IP, les scanners, etc.
- Extraire des cartes d'entités et Appliquer un filtre de convolution pour détecter les objets.

« *Les Mobile Nets sont une famille de modèles pré-entraînés de vision par ordinateur pour TensorFlow, conçus pour maximiser efficacement la précision* »

### 2. Traqueur centroïde :

- Le tracker Centroid est l'un des trackers les plus fiables.
- Pour être simple, le tracker centroïde calcule le centroïde des boîtes englobantes.
- Autrement dit, les points englobantes sont les coordonnées (x, y) des objets dans une image.
- Une fois les coordonnées obtenues par notre SSD, le tracker calcule le centroïde (centre) de la box. En d'autres termes, le centre d'un objet.
- Ensuite, un identifiant unique est attribué à chaque objet particulier détecté, pour le suivi sur la séquence de trames.



```

for (objectID, centroid) in objects.items():
    # object ID
    to = trackableObjects.get(objectID, None)

    if to is None:
        to = TrackableObject(objectID, centroid)
    else:

        y = [c[1] for c in to.centroids]
        direction = centroid[1] - np.mean(y)
        to.centroids.append(centroid)
        if not to.counted:

            if direction < 0 and centroid[1] < H // 2:
                totalUp += 1
                empty.append(totalUp)
                to.counted = True

            elif direction > 0 and centroid[1] > H // 2:
                totalDown += 1
                empty1.append(totalDown)
                p.append(len(empty1)-len(empty))

```

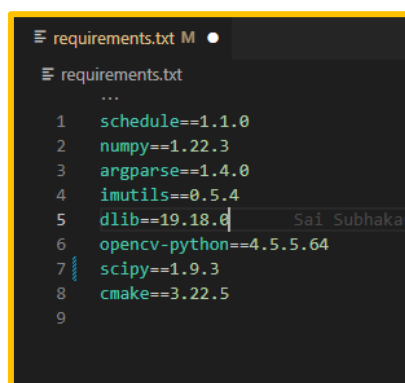
### C. Modes d'utilisations :

**Cloner le Github repository dans la machine locale :**

- « git clone https// »
- « Cd ... »

**Installation des différentes librairie et module :**

- « pip install -r requirement.txt »



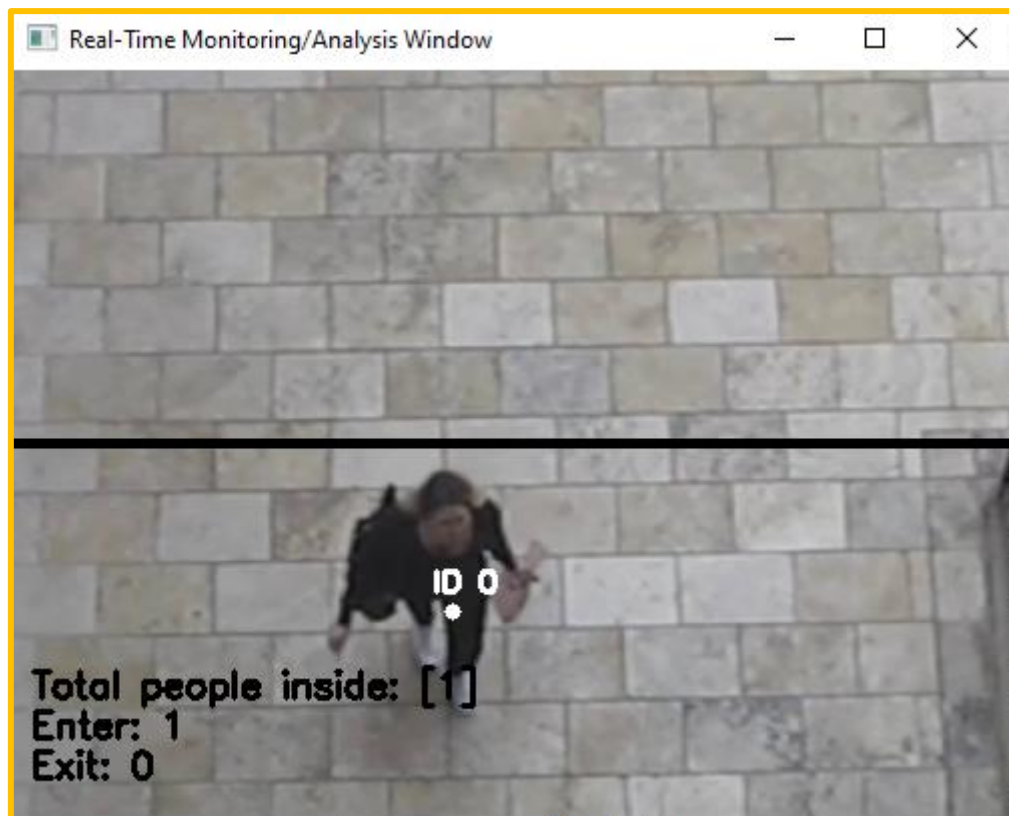
```

requirements.txt M
requirements.txt
...
1 schedule==1.1.0
2 numpy==1.22.3
3 argparse==1.4.0
4 imutils==0.5.4
5 dlib==19.18.0 Sai Subhakar
6 opencv-python==4.5.5.64
7 scipy==1.9.3
8 cmake==3.22.5
9

```

Pour détecter, suivre et compter les clients entrant et sortant nous avons tracé une ligne à une certaine distance en pixel afin qu'il soit adapté à chaque magasin et à chaque banque comme étant une entrée ou une porte virtuelle.

Il suffit de lancer le script principal pour le comptage « Run.py » avec la commande *python Run.py*, et la vidéo commence l'enregistrement et la détection.



Une fois qu'une personne a été détectée, la liste total people est incrémentée ou décrétementée dépendant de l'entrer ou la sortie et elle sera envoyée directement à la base de données de la banque sous format JSON.

La base de données est hébergée dans le serveur Cloud de Firebase qui sera ensuite traitée et affichée dans l'une des fonctionnalités de l'application mobile.

## V. Implémentation de l'application mobile :

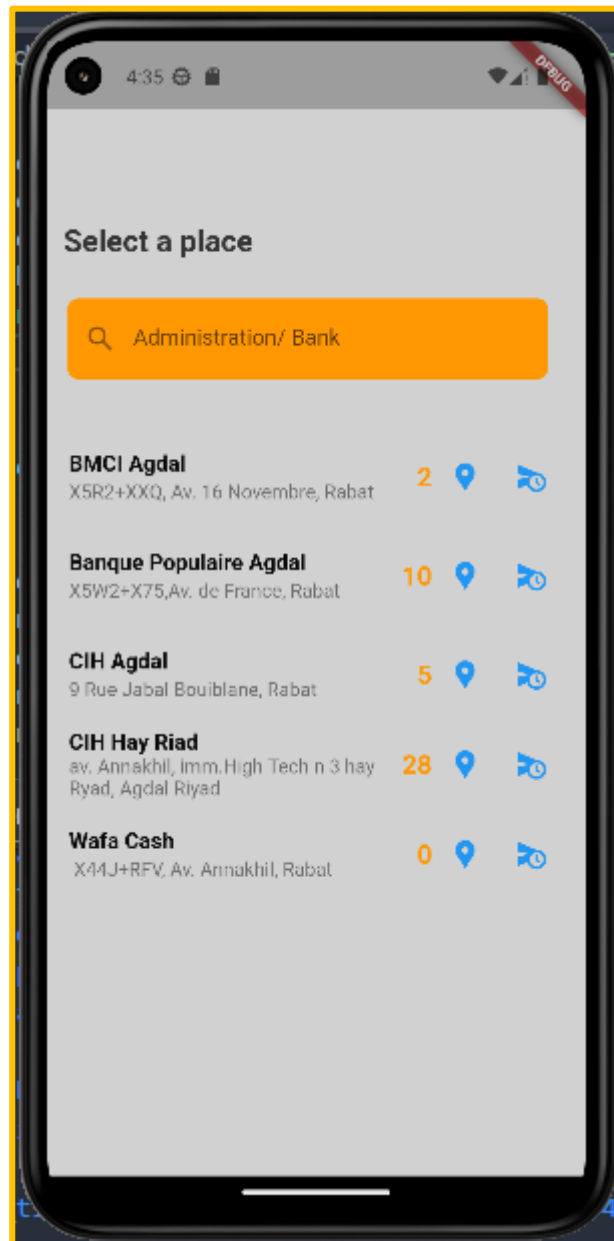
### A. Les fonctionnalités de l'App :

Notre application affiche au départ un Splash screen, avec le logo de Smart Wait, qui s'affiche pendant quelques secondes avant que l'interface utilisateur principale ne soit affichée, et est utilisé pour fournir une indication de chargement à l'utilisateur :

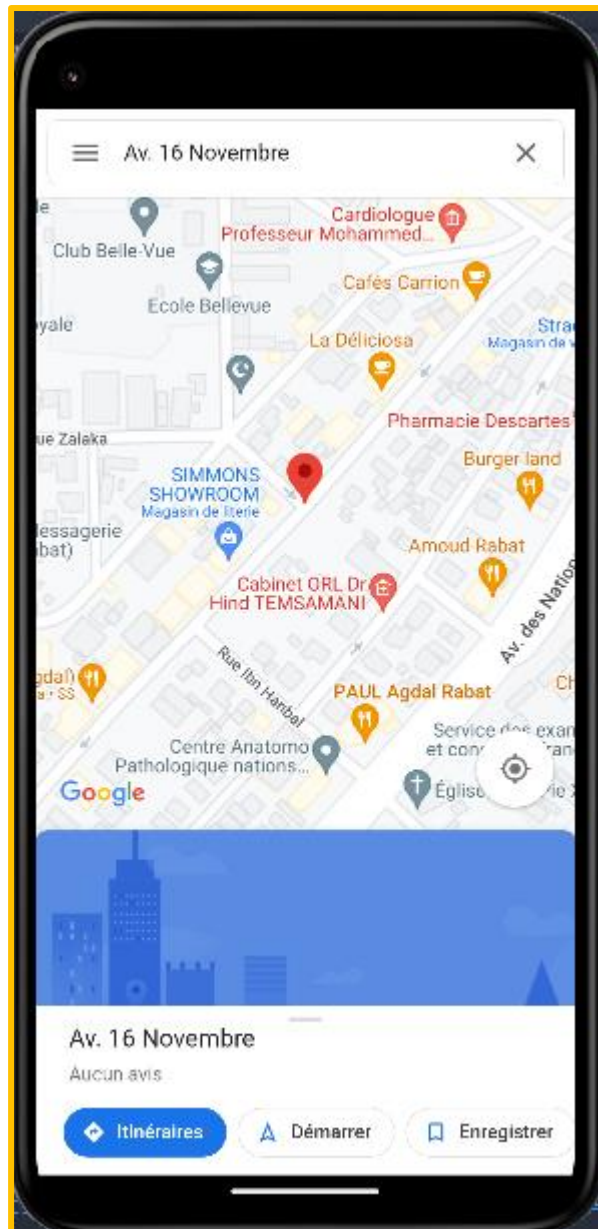


Ensuite l'interface utilisateur principale s'affiche, dans cette interface on affiche chaque place (Banque ou Administration) stocké dans Firebase avec le nombre de personnes qui y existent . Ce dernier a été calculé par le programme python à l'aide des deux bibliothèques Dlib et OpenCV .

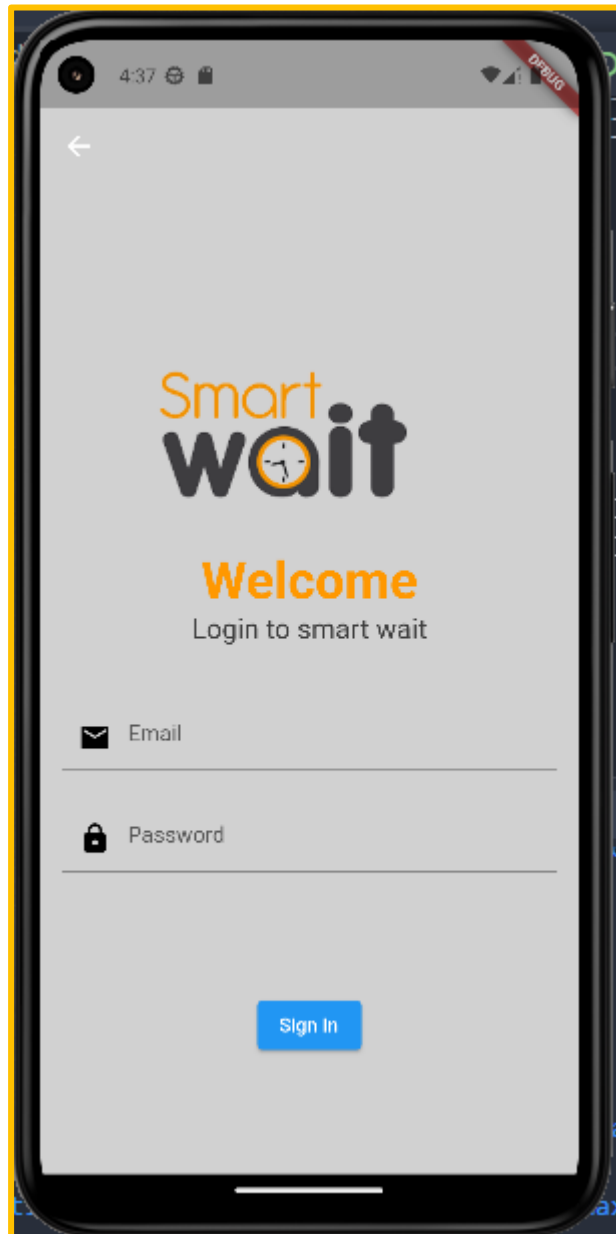
Le client peut donc à travers cette interface cliquer sur deux icônes : l'une représentant une épingle de localisation et l'autre la réservation en ligne.



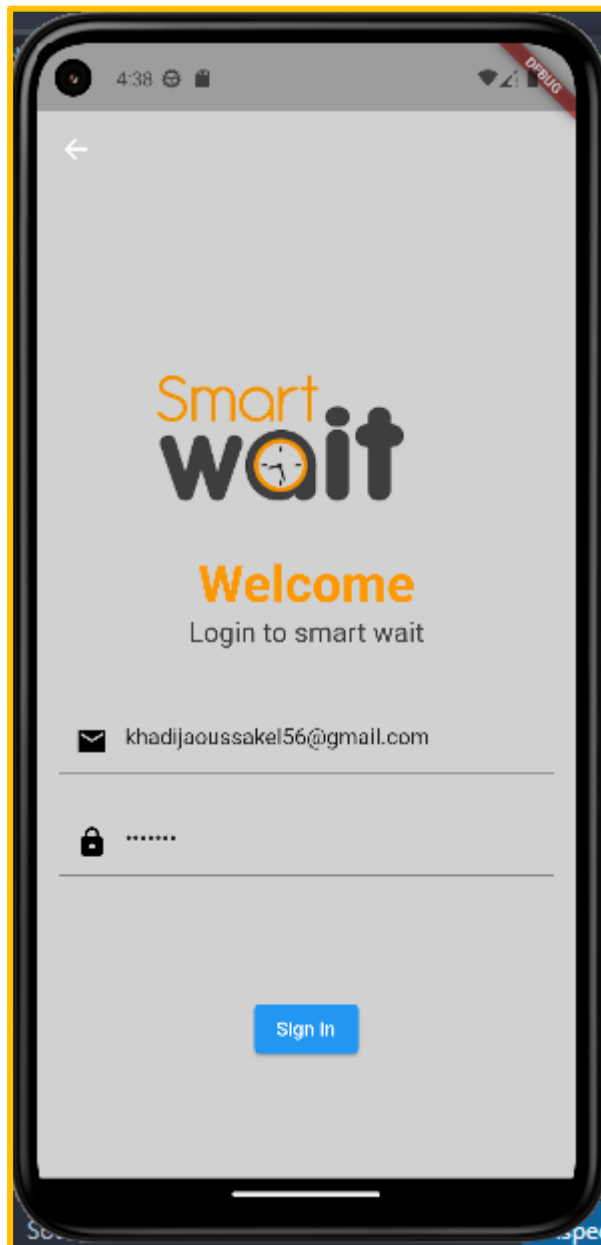
Si l'utilisateur clique sur l'icône d'épingle de localisation, il sera redirigé vers une carte qui prendra en paramètre l'adresse de la place choisie par l'utilisateur :



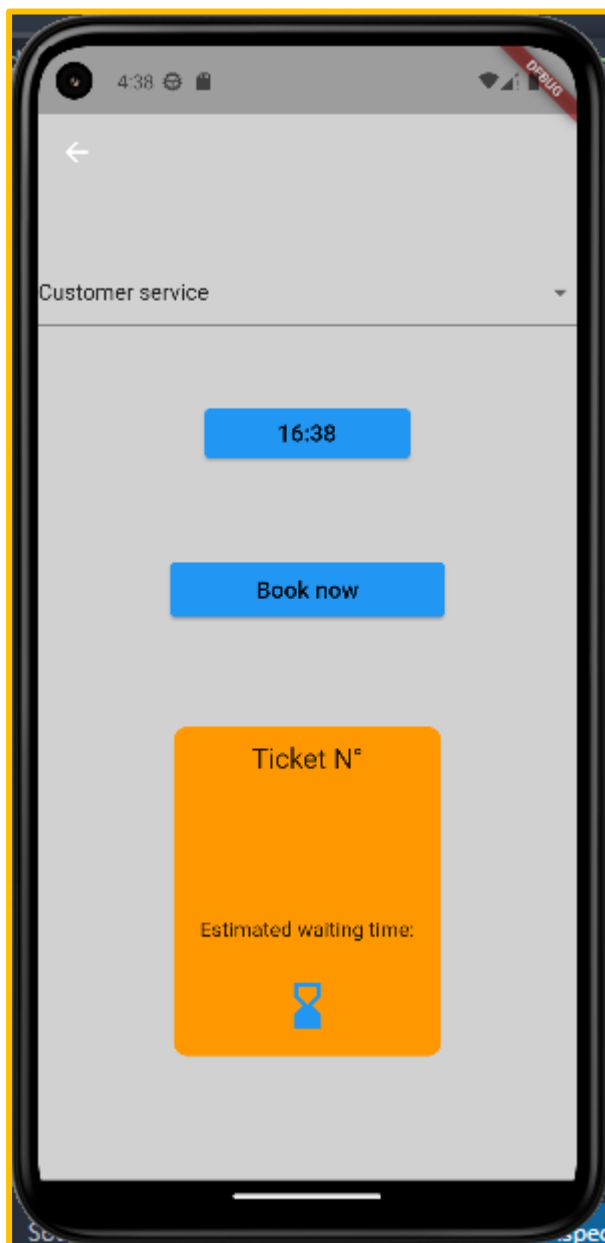
Si l'utilisateur clique sur l'icône de réservation en ligne, il va être redirigé vers une page de connexion qui demande son email et son mot de passe déjà fournis par sa banque :



Une fois il saisit les données nécessaires, il clique sur le bouton « Sign in ». Il est important de vérifier et de valider les informations entrées par l'utilisateur afin de s'assurer que toutes les informations sont correctes et complètes avant de procéder à la réservation.



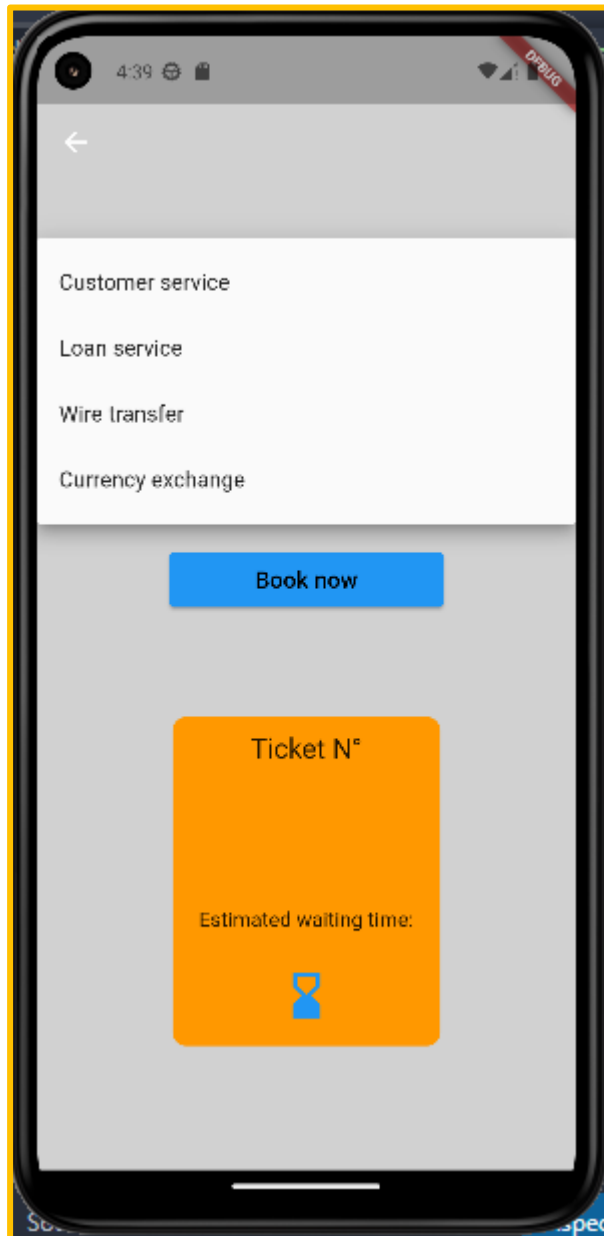
Maintenant l'utilisateur arrive sur la page de réservation où il peut prendre rendez-vous en ligne à l'heure qui lui convient et au service qu'il souhaite :



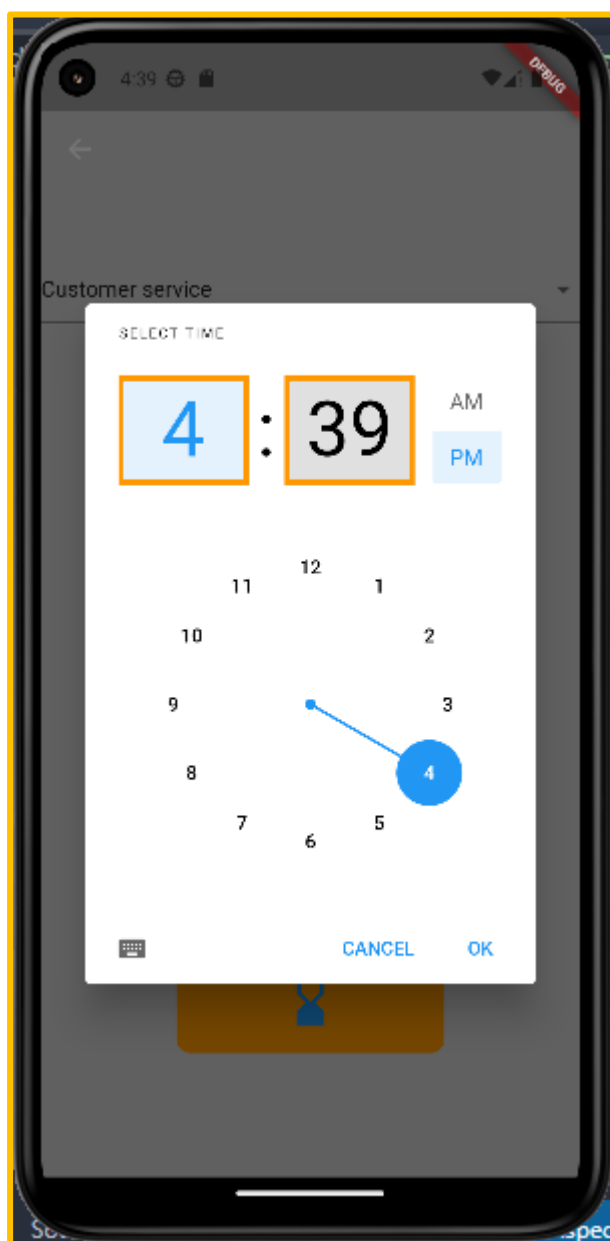


Ici on a listé les différents services d'une banque, à savoir :

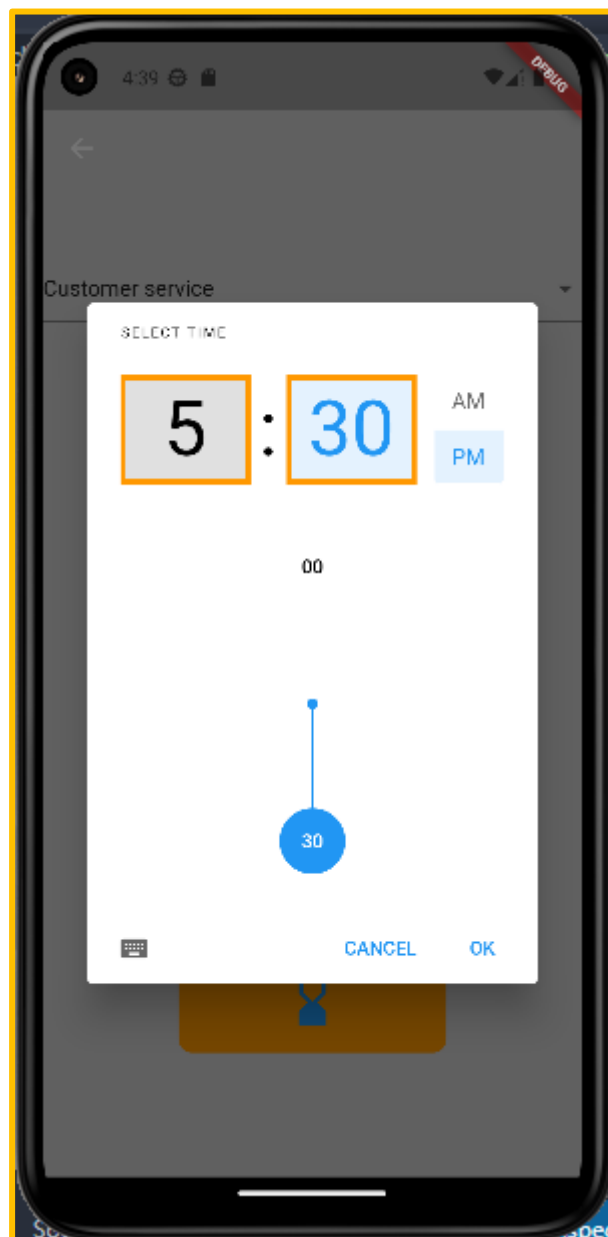
- ✕ Customer service
- ✕ Loan service
- ✕ Wire transfer
- ✕ Currency exchange



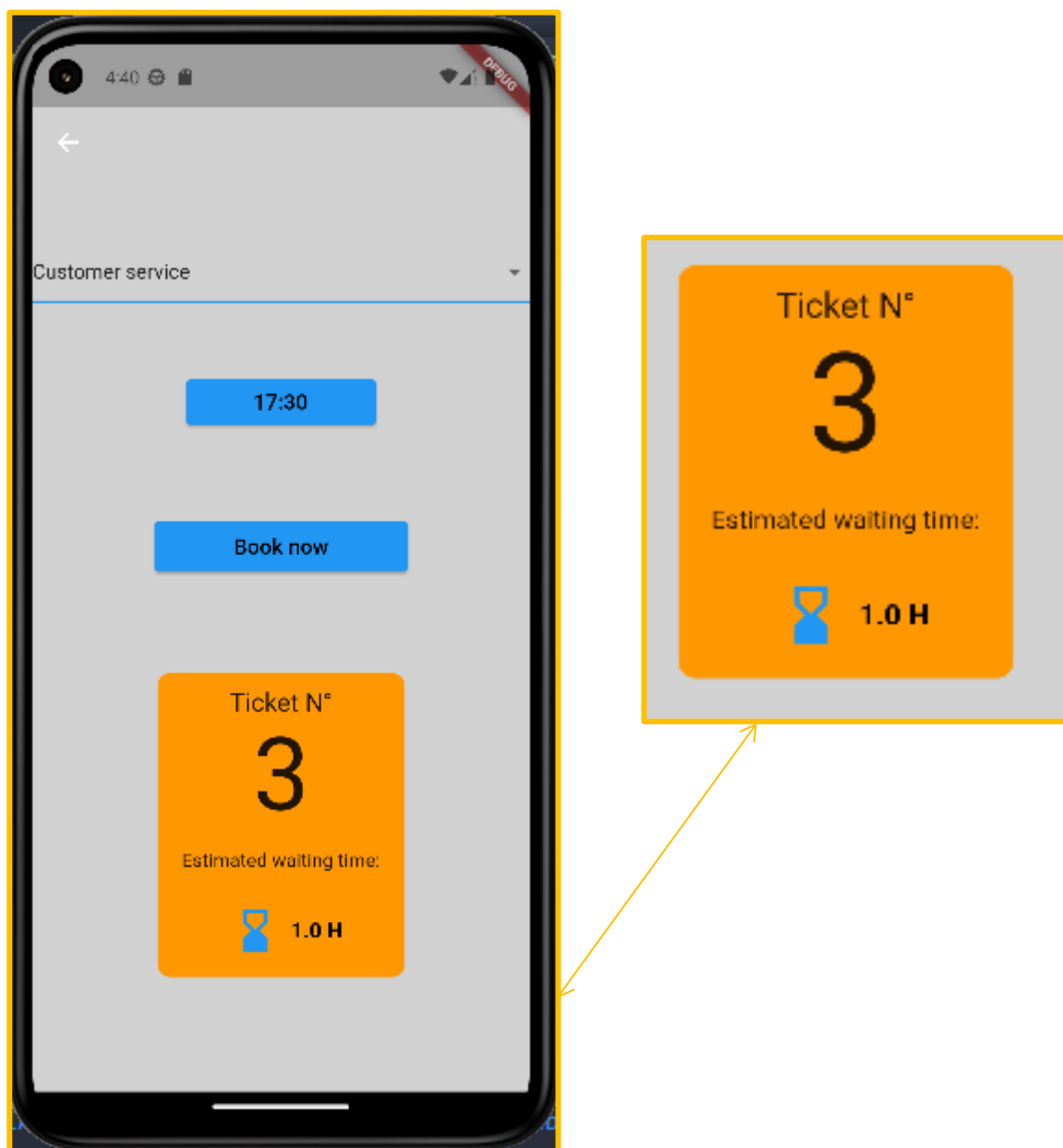
Après avoir choisi le service, l'utilisateur doit choisir le temps pour se rendre physiquement dans l'agence ou l'administration concernée et on a fixé la marge du choix à un jour :



Pour bien gérer la file d'attente, on a utilisé la propriété `IntervalTime` de l'objet `TimePicker` qui détermine l'intervalle de temps entre chaque pas de la sélection de l'heure dans le sélecteur de temps. Dans notre cas on a estimé que chaque personne prendra 30min de passage, et on a fixé l'Intervalle à 30, l'utilisateur ne pourra donc sélectionner que les heures entières et les demi-heures dans le sélecteur de temps.



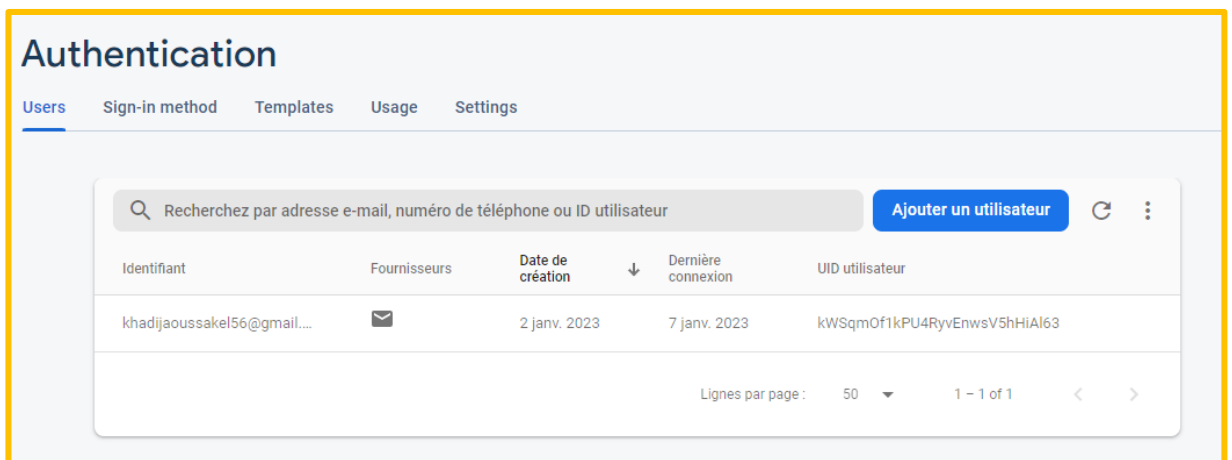
Pour finaliser la réservation, l'utilisateur clique sur le bouton « book now ». Les informations de réservation seront affichées sur le ticket, à savoir son numéro dans la file d'attente et le temps estimé d'attente :



## B. Le backend de l'application :

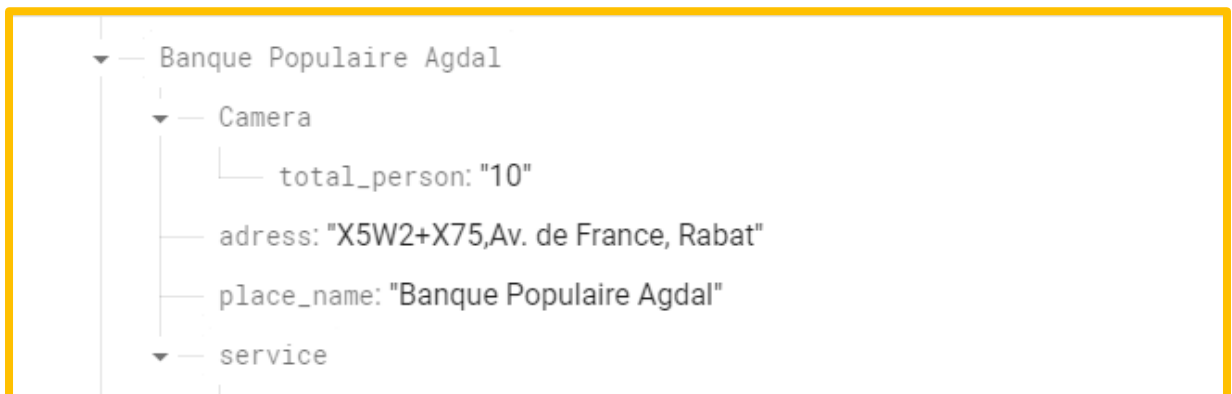
### 1. Authentification :

Pour gérer l'authentification des utilisateurs de notre application de manière simple et sécurisée, on a utilisé Firebase Authentication un service de gestion des utilisateurs proposé par Google dans sa plateforme Firebase. Il prend en charge plusieurs méthodes d'authentification, et on a utilisé l'authentification par e-mail et mot de passe :

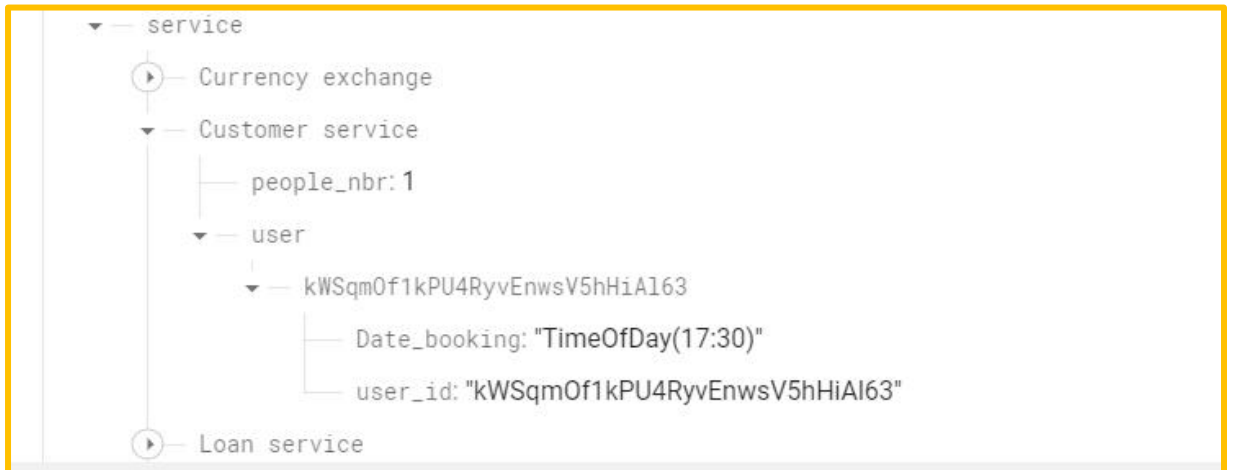


### 2. Réservation :

L'application accède en temps réel aux données stockées dans Firebase. Une fois connecté, l'utilisateur aura accès aux derniers changements de la base de données. Le nombre de personnes est modifié grâce au programme python :



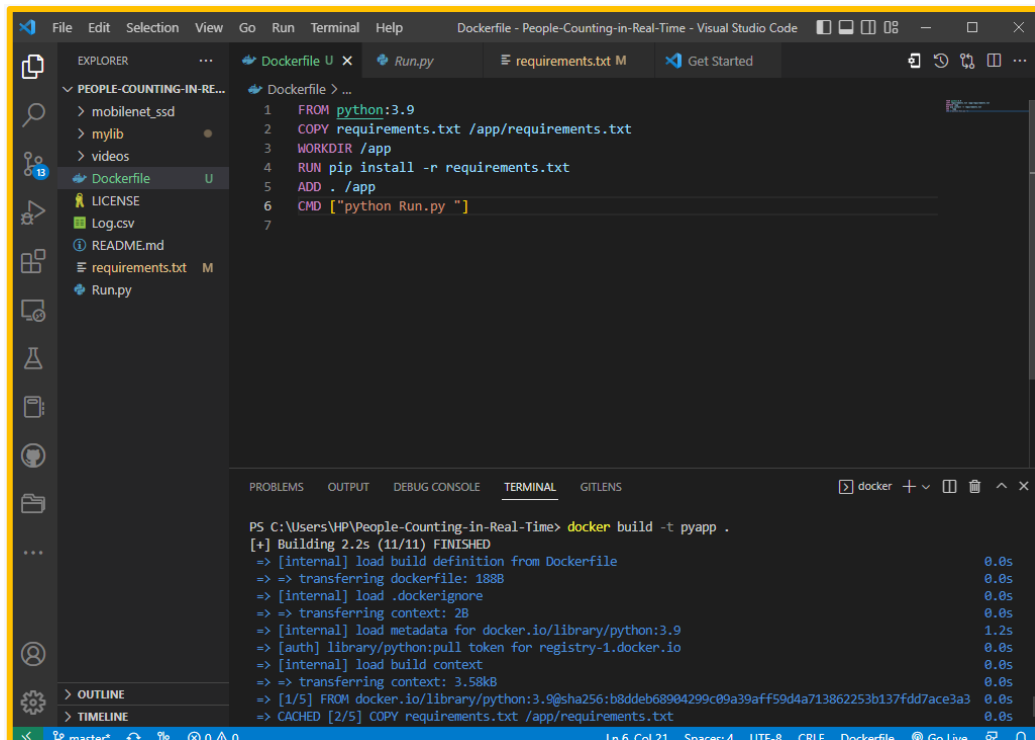
Et lorsque l'utilisateur réserve un ticket en ligne dans un service donnée, le programme en flutter permet d'accéder au service et de savoir le nombre de personnes qui ont réservé pour le même service, et attribue le numéro en file d'attente à l'utilisateur tout en incrémentant la valeur du nombre de personnes. De plus le programme stockera l'ID de l'utilisateur et la date de réservation dans le service concerné :



## VI. Docker et DockerHub :

### A. Création d'une image docker :

- On crée un fichier Dockerfile dans le répertoire de notre application Python.
- Le Dockerfile est un fichier qui contient des instructions pour Docker sur la façon de construire une image Docker. Un Dockerfile commence par spécifier une base d'image à partir de laquelle on crée notre image, puis ajoute des fichiers, exécute des commandes, et définit des paramètres d'environnement pour configurer l'image.



- Puis on exécute la commande **Docker build -t python\_app** dans le répertoire contenant le Dockerfile. Cette commande crée une image docker qui contient des conteneurs d'exécution du projet.
- Après on exécute la commande **Docker run -d python\_app** afin de lancer un conteneur.

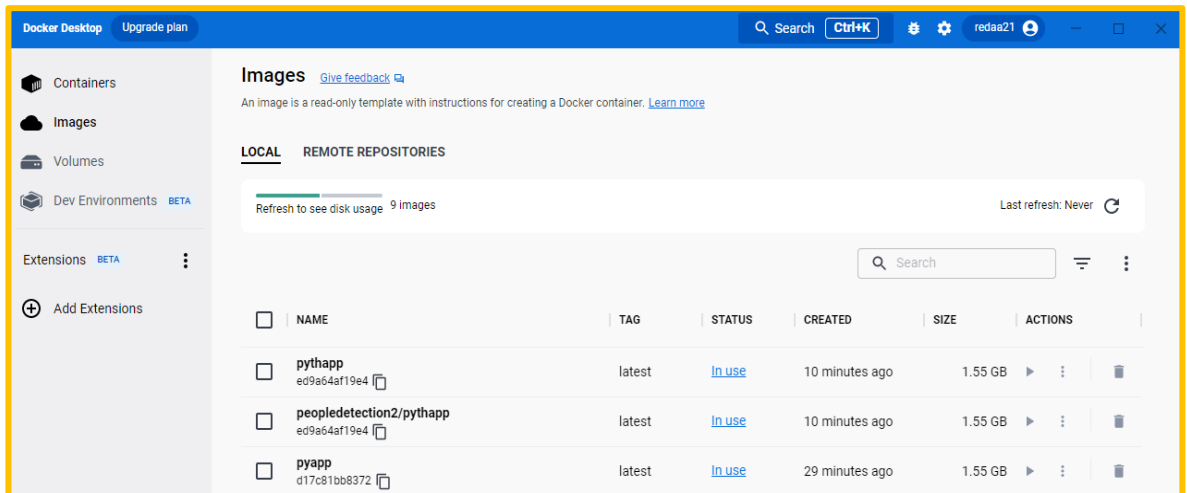
```

PS C:\Users\HP\People-Counting-in-Real-Time> docker build -t python_app .
[+] Building 3.6s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 329B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.9
=> [auth] library/python:pull token for registry-1.docker.io
=> => transferring context: 3.72kB
=> CACHED [2/5] COPY requirements.txt /app/requirements.txt
=> CACHED [3/5] WORKDIR /app
=> CACHED [4/5] RUN pip install -r requirements.txt
=> [5/5] ADD . /app
=> exporting to image
=> => exporting layers
PS C:\Users\HP\People-Counting-in-Real-Time> docker run -d python_app
7df60aee912341eda8ab0810487103bc0a40885790b69744ceb1902459b2f5c
PS C:\Users\HP\People-Counting-in-Real-Time> docker login
Authenticating with existing credentials...
Login Succeeded

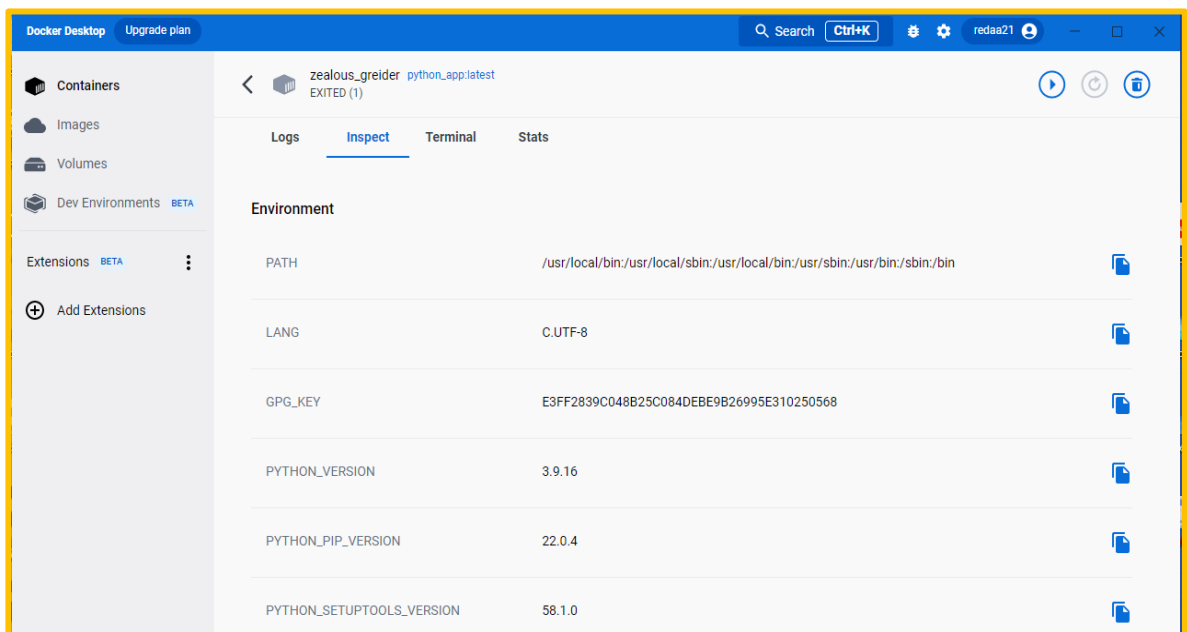
Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at
https://docs.docker.com/go/access-tokens/

```

Dans Docker on trouve ceci :



Après le lancement du conteneur par **docker run** :



## B. Déploiement de l'image docker dans Docker Hub :

DockerHub est un service cloud-based qui permet de stocker et gérer les images Docker. On peut l'utiliser ainsi pour gérer et organiser des groupes d'utilisateurs qui veulent collaborer sur des référentiels partagés.



- Après la création d'un repo en DockerHub, on s'authentifie localement par la commande `docker login`
- Ensuite, on exécute la commande suivante **`docker tag python_app redaa21/python_app`** le tag utilisé nous permet d'identifier notre image en DockerHub
- Enfin, on exécute la commande **`docker push redaa21/python_app`** afin d'exporter l'image au repository de DockerHub, cela simplifiera le déploiement après de l'image dans différents environnements et son exécution dans n'importe quelle machine.

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
PS C:\Users\HP\People-Counting-in-Real-Time> docker tag python_app redaa21/python_app
PS C:\Users\HP\People-Counting-in-Real-Time> docker push redaa21/python_app
Using default tag: latest
The push refers to repository [docker.io/redaa21/python_app]
e5d3c024d45a: Preparing
012f4b70fcda: Preparing
012f4b70fcda: Pushed
302100ac5738: Pushed
26f5bb747440: Pushed
2383c46b6fbc: Pushed
7739e9ebcd9a: Pushed
248397b6b856: Pushed
fa1175420e6f: Pushed
bb2453e12947: Pushed
7354e83da007: Pushed
c284f546974c: Pushed
4efcd4003c84: Pushed
latest: digest: sha256:c3af4e2f77f2fa71bbcef23b8f49a92ce173a50a8c22abec8a8183739d7d4763 size: 3056
PS C:\Users\HP\People-Counting-in-Real-Time>

```

Dans DockerHub on trouve notre repo qui contient notre image :

**Description**  
*This repository does not have a description*  
 Last pushed: 4 minutes ago

To push a new tag to this repository,  
`docker push redaa21/python_app:tagname`


**Tags**  
 This repository contains 1 tag(s).  

Tag	OS	Type	Pulled	Pushed
latest		Image	---	4 minutes ago

VULNERABILITY SCANNING - DISABLED  
[Enable](#)


**Automated Builds**  
 Manually pushing images to Hub? Connect your account Bitbucket to automatically build and tag new images when code is updated, so you can focus your time on creating.  
 Available with Pro, Team and Business subscriptions.  
[Upgrade](#) [Learn more](#)

**README**  
 Repository description is empty. Click [here](#) to edit.

 [Search Docker Hub](#)

[Explore](#) [Repositories](#) [Organizations](#) [Help](#)

[Upgrade](#)

 [redaa21](#)

redaa21

Search by repository name

All Content

Create repository

redaa21 / **python\_app**

Contains: Image | Last pushed: 3 minutes ago

Not Scanned

0

0

Public

redaa21 / **smartwait\_1**

Contains: No content | Last pushed: an hour ago

Not Scanned

0

0

Public

# RÉSUMÉ

En tant que groupe, nous avons développé une application mobile de gestion des files d'attente dans les banques et les administrations en utilisant Flutter comme cadre de développement, l'apprentissage automatique pour la gestion de la file d'attente et l'API JSON stockée dans Firebase pour la gestion des données. Nous avons utilisé des bibliothèques de vision par ordinateur telles que OpenCV et Dlib, ainsi que TensorFlow, pour détecter le nombre de personnes présentes dans les banques et les administrations en temps réel.

Nous avons mis en place une interface utilisateur conviviale permettant aux utilisateurs de s'inscrire et de prendre un ticket virtuel en temps réel, ainsi que de recevoir des informations sur le temps qu'il leur reste avant de pouvoir être servis.

L'apprentissage automatique a joué un rôle crucial dans la gestion de la file d'attente en utilisant des algorithmes de détection. La détection du nombre de personnes présentes dans les banques et les administrations en temps réel a également contribué à améliorer l'efficacité de la gestion de la file d'attente.

En utilisant Firebase comme solution de stockage de données, nous avons été en mesure de gérer efficacement les données de l'application et de fournir une expérience utilisateur fluide sans temps de chargement excessif.

En conclusion, le développement de cette application mobile de gestion de files d'attente a été un projet passionnant qui nous a permis de mettre en pratique nos compétences en développement de logiciels et de travailler en équipe pour créer un produit utile pour les utilisateurs. L'utilisation de l'apprentissage automatique et de la vision par ordinateur pour détecter le nombre de personnes a été un ajout précieux à notre application et a contribué à son efficacité. Nous sommes convaincus que cette application a le potentiel de faciliter grandement la gestion des files d'attente dans les banques et les administrations, et nous espérons qu'elle sera largement adoptée dans un proche avenir.

