GUNES Omer / Groupe 2C1

# R3.03 - Analyse

# Documentation de l'application FlutterFlow

TABLE DES MATIERES

Cadre du TP ..... 4
Cahier des charges du TP ..... 4
Description de l'application ..... 4
Fonctionnalites de l'application ..... 5
Les Technologies ..... 5
Back-End ..... 6
Fonctionnement generale ..... 6
Gestion des donnees ..... 7
API et Serveur ..... 8
Design d'API en production ..... 8
Design d'API que je voulais effectuer ..... 9
Fichier .ENV ..... 13
Documentation ..... 13
Diagramme Technique ..... 14
Flow Chart ..... 14
Diagramme cas utilisation ..... 14
Diagramme de classe ..... 15
Diagramme de séquence ..... 15
Schema de la base de données ..... 16
Diagramme non-Technique (gestion projet) ..... 17
Diagramme de Gantt ..... 17
Interface utilisateur ..... 18
Charte graphique ..... 18
Présentation des differentes pages ..... 19
Connexion / Inscription ..... 19
Mot de passe oublié ..... 20
Page principale ..... 21
Les articles ..... 22
Photothèque ..... 23

Carte Fidelite ..... 24
Profil ..... 26
Carte de fidélité ..... 27
Changement de mot de passe ..... 28
Changement d’adresse mail ..... 29
Custom Function ..... 30
Glossaire ..... 31
Table des figures ..... 32

# CADRE DU TP

## CAHIER DES CHARGES DU TP

Dans ce TP, nous devons produire une application en utilisant l'outil de production FlutterFlow. L'application qu'on va produire doit être prêt à être mis en production. Il doit être finalisé et sans bugs. L'application doit contenir :

- Système de connexion et d'inscription
- Utilisation d'une base de données (FireBase)
- Utilisation d'au moins d'une API
- Utilisation d'une « Custom Function »
- Être sans bugs

## DESCRIPTION DE L'APPLICATION

Pour ce TP, j'ai choisi de me lancer dans un projet concret : le développement d'une application pour l'Association Culturelle Turque de Quimper. Ce projet, que je prévoyais de débuter en React Native d'ici début 2025, me permettra de poser les bases de la partie serveur et API dès maintenant. Utiliser FlutterFlow pour créer ce projet me permet de créer une « maquette » fonctionnelle du projet.

L'application est destinée dans un premier temps aux membres de l'association. Par la suite, elle sera déployée pour un public plus large.

## FONCTIONNALITES DE L'APPLICATION

L'application va permettre différentes choses, qui seront utiles aux membres de l'association dans un premier temps.

Voici les différentes fonctionnalités :

- Création de compte utilisateur
- Connexion à un compte utilisateur existant
- Modification de mot de passe utilisateur
- Permettre de consulter les dernières actualités
- Afficher sa carte de membre (permettant d'avoir des réductions dans l'épicerie de l'association)
- Voir ses derniers achats effectué à l'épicerie
- Possibilité de réinitialiser le mot de passe (si oublié)

## LES TECHNOLOGIES

Les technologies utilisées pour la mise en œuvre de se projet sont :

- JavaScript : Pour la création de l'API qui va permettre à l'application FlutterFlow de récupérer les données directement de la base de données MySQL. Le code JavaScript est situé sur un serveur distant ce qui permet l'accès à celle-ci de partout.

- Python : Le code python est un script permettant la récupération du ticket de caisse envoyé par la caisse enregistreuse. Le script va ensuite déterminer l'adresse électronique, le nom, le prénom du client mais également le montant, la date et l'heure de l'achat.

- FlutterFlow : L'application est créer avec FlutterFlow, un outil permettant de créer des applications en Low-Code.

- MySQL : La base de données va stockée les achats effectué par les clients, va également stocker une table utilisateur qui va contenir les cartes de fidélité et les mails clients.

- FireBase : Système de gestion de base de données appartenant à Google, va stocker les différents compte utilisateur.

# BACK-END

## FONCTIONNEMENT GENERALE

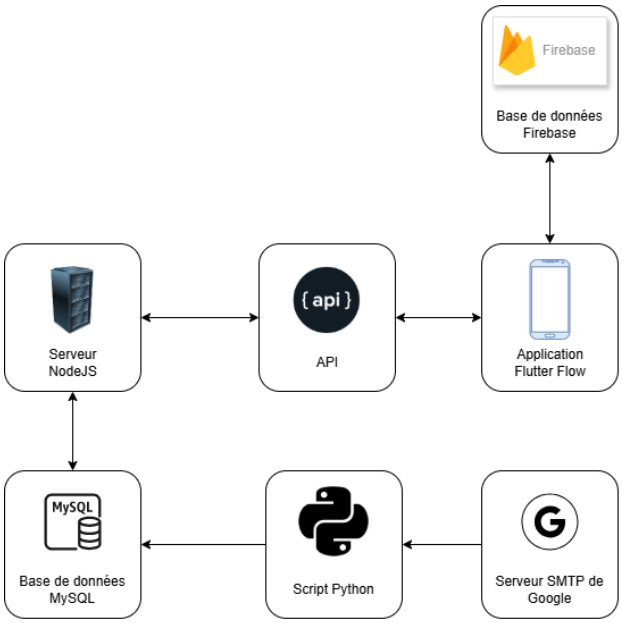



**Figure 1 - Fonctionnement Générale du projet**

Tout d'abord l'application va gérer l'inscription et la connexion des utilisateurs avec la base de données Firebase.

Pour les autres fonctionnalité comme l'affichage de la carte membre, ou l'historique d'achat on va utilisé une API, qui elle va interagir avec un serveur NodeJS. Ce serveur permettra d'assurer une interaction fluide et sécurisée entre l'application et les données stockées, tout en offrant la possibilité d'étendre facilement les fonctionnalités futures.

Le script Python va récupérer les tickets de caisse via les mails qu'on reçoit. Les mails vont être récupérer directement via les serveurs de Google.

Grâce à cette architecture modulaire et polyvalente, l'application sera capable de répondre à des besoins variés tout en assurant une évolutivité pour des ajouts futures.

## GESTION DES DONNEES

L'application repose sur l'intégration de deux bases de données distinctes pour répondre à des besoins spécifiques. La première base de données est Firebase, utilisée dans la gestion des utilisateurs et les fonctionnalités liées à l'authentification. Grâce à Firebase l'application bénéficie de :

- La connexion des utilisateurs
- L'inscription des nouveaux utilisateurs
- La déconnexion des utilisateurs
- La suppression de compte
- Changement de mot de passe
- Réinitialisation de mot de passe

La deuxième base de données, quant à elle, est une base de données plus « classique », elle est dédié au stockage des informations de transactions et les données utilisateurs. Elle est constitué de cette façon :

**Table des transactions (Achat) :**

- Identifiant de la transaction (générer automatiquement)
- Identité du client (nom et/ou prénom)
- Montant de la transaction
- Adresse e-mail de l'utilisateur lié à l'achat
- Date de l'achat
- Heure de l'achat

**Table des utilisateurs (Utilisateur) :**

- Adresse e-mail de l'utilisateur
- Code barre de la carte de membre de l'utilisateur

## API ET SERVEUR

Le serveur est hébergé directement chez moi, on peut y accéder via le lien `http://omergs.com:55555/`. Pour utiliser l'API vous devrez mettre la route que vous voulez après le /.

L'application FlutterFlow va utilisé une API pour communiqué via le serveur NodeJS. La communication avec cet API nécessite un token, sans ce token le serveur répondra uniquement `{"error":"Accès refusé : token invalide"}` .

Le token est directement intégré dans les requêtes de l'application FlutterFlow. Le token doit rester secret dans un "vrai" projet, dans le cadre du TP le token est :
`r304FyEkdtQTSuLyLZvIieLSlCrA12BLAHAJbkngmxw7EIUADXb7gEPlDcOf8vryJ2oYbLYvMOyMuOiODJPsDZV5ETsjyOhSOUfletAuFromageFZ8bLTYezXUoC5wS53RX68QkYGJafoyFZ3DS7fonFN40LFETxlQZqU1pxGPtHaFEURrcrj0k2Kn5tSwb7k0Yf8LUaiNRFKC9G6KXtupl7u6ppstxqh6B9SGk9CJXhNmtJX2CrpDDreV4JgoK5eJccAAiyu2tlP8KONRADtLVCjDNPh`

## DESIGN D'API EN PRODUCTION

Comme FlutterFlow est vraiment super bien fait, les routes ci-dessous était mon idée et ils étaient déjà implémenté sur mon serveur NodeJS. Mais j'ai récemment découvert qu'on pouvait mettre uniquement 2 appels API sur une application FlutterFlow, pour en mettre plusieurs faut payé, donc pour contrer cela j'ai du réfléchir à une solution. Au final j'ai aboutie à une solution un peu bricolé mais qui marche. J'ai créer une seul API accessible via :
http://omergs.com:55555/flutterFlow

Nous pouvons envoyé 3 paramètres à cet API, premièrement le type d'opération qu'on veut effectuer, nous avons ensuite le mail, puis nous avons le nombre de ligne (utile uniquement pour la partie gestion de l'historique d'achat).


Figure 2 - Requête de l'application vers /flutterFlow

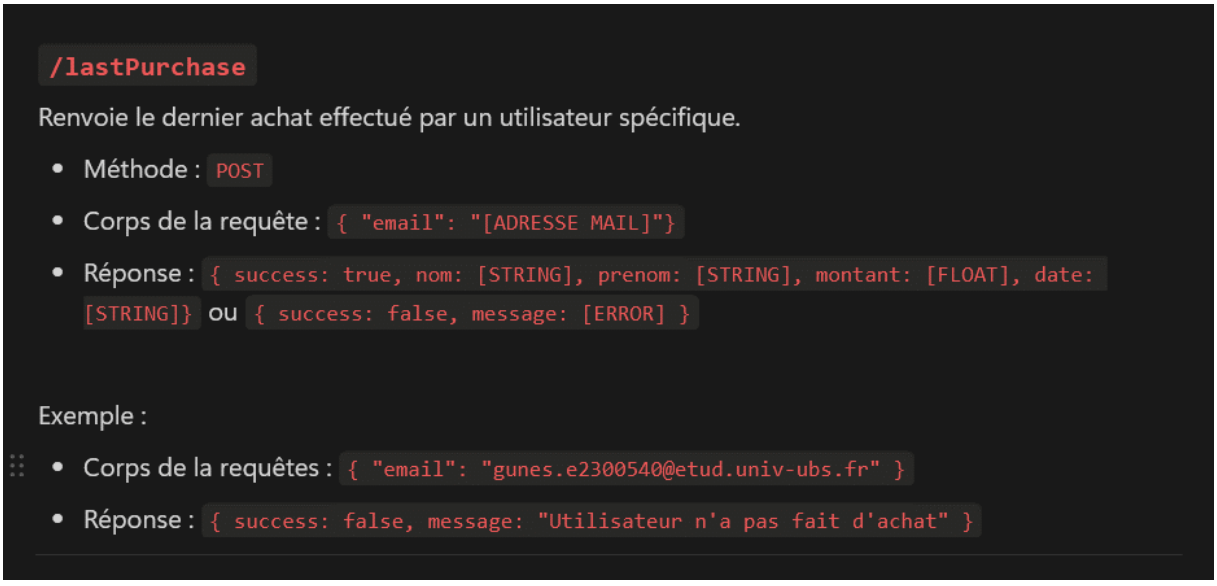
### TYPE D'OPERATION :

Chaque type d'opération fait référence à une route de l'ancien modèles. Voici les types d'opérations :

- **registerUser :** Fait comme la route /registerUser
- **deleteUser :** Fait comme la route /deleteUser
- **lastPurchase :** Fait comme la route /lastPurchase
- **fidelityCard :** Fait comme la route /fidelityCard
- **purchase :** Fait comme la route /purchase
- **allPurchase :** Fait comme la route /allPurchase
- **totalNumberPurchase :** Fait comme la route /totalNumberPurchase
- **default** : Si le type correspond à aucune des types ci-dessus nous allons renvoyé : `({ success: false, message: 'Type inconnu.' });`

## DESIGN D'API QUE JE VOULAIS EFFECTUER

### /LASTPURCHASE



## /FIDELITYCARD

**`/fidelityCard`**

Renvoie le numéro associé à la carte de fidélité.

- Méthode : `POST`
- Corps de la requête : `{ "email": "[ADRESSE MAIL]"}`
- Réponse : `{ success: true, barcode: [STRING] }` ou `{ success: false, message: [STRING] }`

Exemple :

- Corps de la requêtes : `{ "email": "gunes.e2300540@etud.univ-ubs.fr" }`
- Réponse : `{ success: true, barcode: "2356059" }`

## /PURCHASE

**`/purchase`**

Renvoie X achat pour l'utilisateur dont l'adresse électronique a été envoyer.

- Méthode : `POST`
- Corps de la requête : `{ "email": "[ADRESSE MAIL]", "row": [INTEGER] }`
- Réponse : `{ success: true, rowCount: [INTEGER], purchase[ARRAY] }` ou `{ success: false, message: [STRING] }`

Exemple :

- Corps de la requêtes : `{ "email": "27omerf@gmail.com", "row": 10 }`
- Réponse : `{"success":true,"rowCount":5,"purchases":[{"identite":"Omer Gunes","montant":"20.00","date":"2024-04-29T14:46:35.000Z"},{"identite":"Omer Gunes","montant":"20.00","date":"2024-04-29T14:46:35.000Z"},{"identite":"Omer Gunes","montant":"20.00","date":"2024-04-29T14:46:35.000Z"},{"identite":"Omer Gunes","montant":"20.00","date":"2024-04-29T14:46:35.000Z"},{"identite":"Omer Gunes","montant":"20.00","date":"2024-04-29T14:46:35.000Z"}]}`

## /TOTALNUMBERPURCHASE

**`/totalNumberPurchase`**

Renvoie le nombre total d'achat pour un utilisateur

- Méthode : `POST`
- Corps de la requête : `{ "email": "[ADRESSE MAIL]"}`
- Réponse : `{ numberOfPurchase: [INTEGER] }`

Exemple :

- Corps de la requêtes : `{ "email": "gunes.e2300540@etud.univ-ubs.fr" }`
- Réponse : `{ numberOfPurchase: 0 }`

## /ALLPURCHASE

**`/allPurchase`**

Renvoie le nombre la totalité des achats effectué par un utilisateur

- Méthode : `POST`
- Corps de la requête : `{ "email": "[ADRESSE MAIL]"}`
- Réponse : `{ count: [INTEGER], purchase: [ARRAY] }` ou `{ success: false }`

Exemple :

- Corps de la requêtes : `{ "email": "romainperon29750@gmail.com" }`
- Réponse : `{"count":4,"purchase":[{"identite":"Romain Peron","montant":"31.50","date":"2024-12-24T14:38:35.000Z"},{"identite":"Romain Peron","montant":"31.50","date":"2024-12-24T14:38:35.000Z"},{"identite":"Romain Peron","montant":"31.50","date":"2024-12-24T14:38:35.000Z"},{"identite":"Romain Peron","montant":"31.50","date":"2024-12-24T14:38:35.000Z"}]}`

## /REGISTERUSER

**`/registerUser`**

Création d'un nouvel utilisateur, on stocke l'adresse électronique et le code barre.

- Méthode : `POST`
- Corps de la requête : `{ "email": "[ADRESSE MAIL]"}`
- Réponse : `{ success: true }` ou `{ success: false }`

Exemple :

- Corps de la requêtes : `{ "email": "gunes.e2300540@etud.univ-ubs.fr" }`
- Réponse : `{ success: true }`

## /DELETEUSER

**`/deleteUser`**

Suppression de l'utilisateur de la base de données

- Méthode : `POST`
- Corps de la requête : `{ "email": "[ADRESSE MAIL]"}`
- Réponse : `{ success: true }` ou `{ success: false }`

Exemple :

- Corps de la requêtes : `{ "email": "gunes.e2300540@etud.univ-ubs.fr" }`
- Réponse : `{ success: true }`

## FICHIER .ENV

Un fichier .env est un fichier de configuration utile pour stocker des informations nécessaires au fonctionnement d'une application. Il permet de ne pas coder les informations en dure dans le code.  Dans ce TP, j'utilise un fichier d'environnement pour le serveur NodeJS.

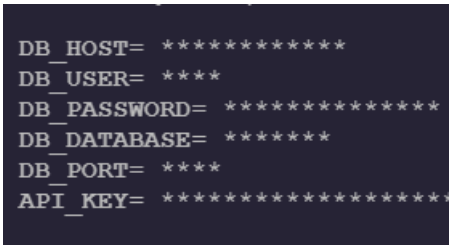



Figure 3 - Fichier .env (masqué)

**DB_HOST :** L'adresse où se situe la base de données MySQL (exemple : omergs.com, teamcroissant.fr, iutvannes.fr, 87.54.12.16...)

**DB_USER :** L'utilisateur qui va se connecter à la base de données

**DB_PASSWORD :** Le mot de passe de l'utilisateur

**DB_PORT :** Le port de la base de données (Pour MySQL le port par défaut est 3306)

**API_KEY :** Le token permettant d'accepter les requêtes venant de l'application uniquement, une personne qui ne connait pas ce token ne pourra pas interagir avec l'API.

## DOCUMENTATION

- Documentation de l'application FlutterFlow est le document ci-présent.

# DIAGRAMME TECHNIQUE

## FLOW CHART

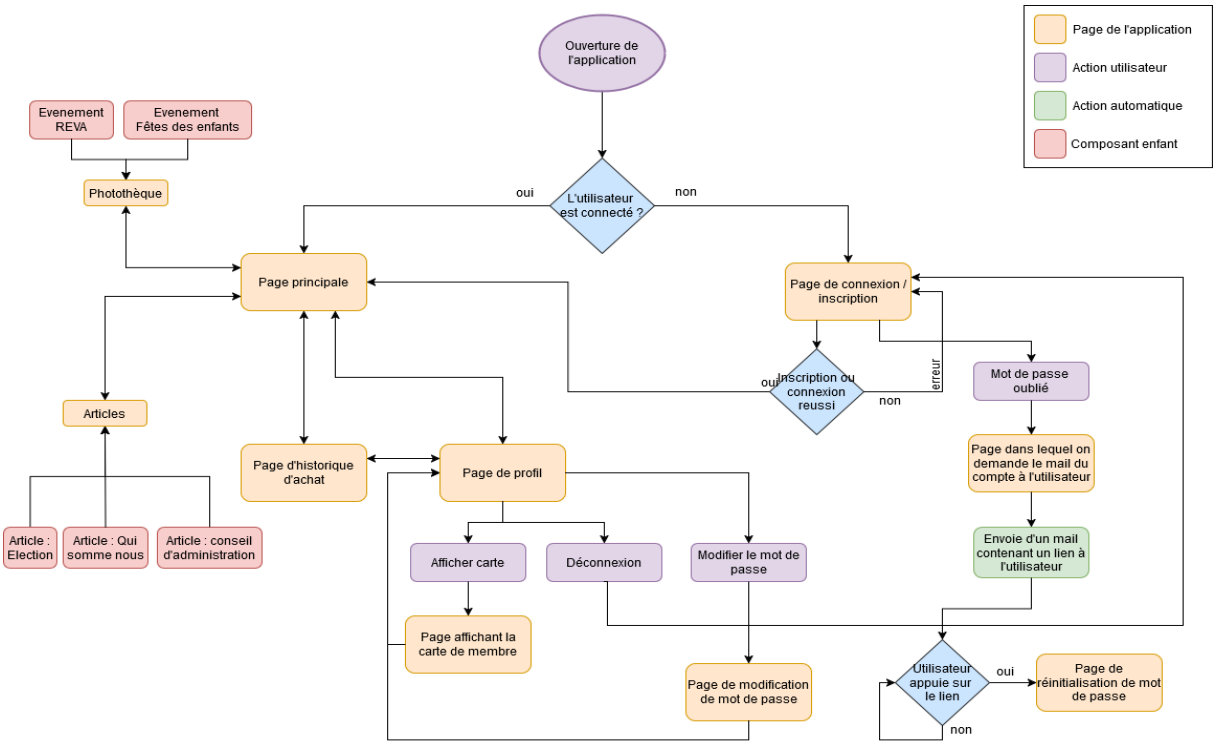



**Figure 4 - Flow Chart de l'application**

## DIAGRAMME CAS UTILISATION

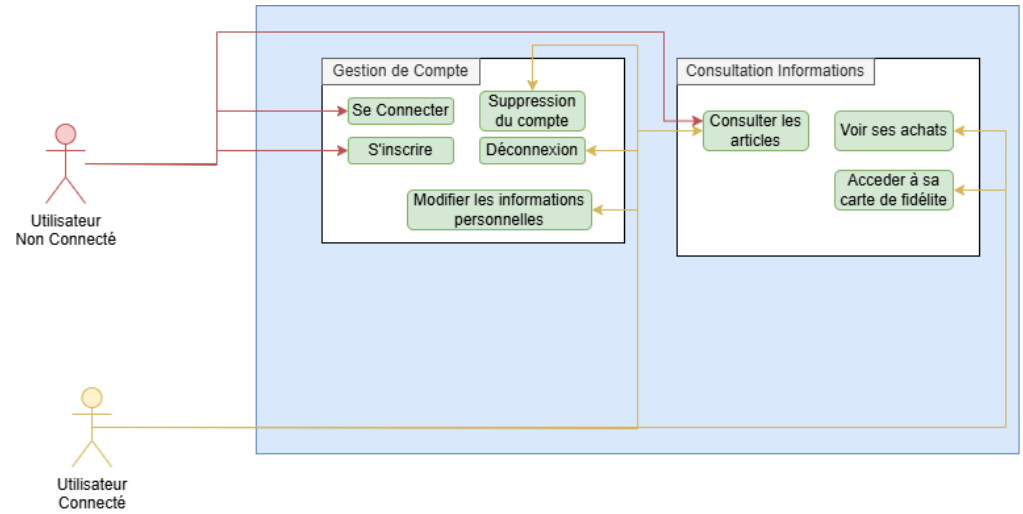



**Figure 5 - Diagramme de cas d'Utilisation**

## DIAGRAMME DE CLASSE

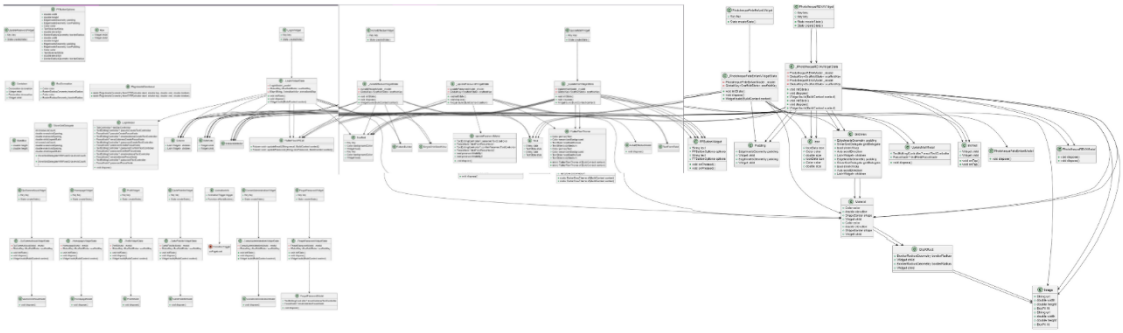


**Figure 6 - Diagramme de classe**

Pour une meilleur qualité le diagramme de classe est disponible via ce lien

## DIAGRAMME DE SEQUENCE

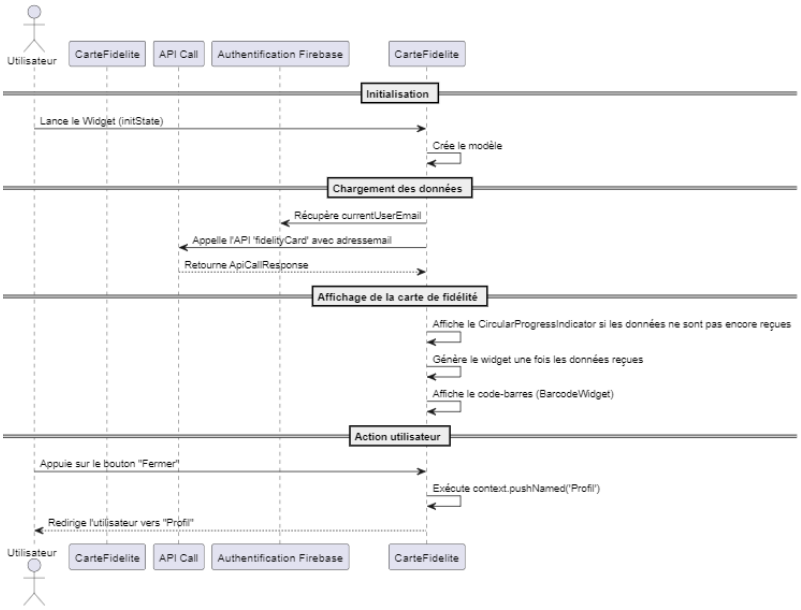



**Figure 7 - Diagramme de séquence**

## SCHEMA DE LA BASE DE DONNEES

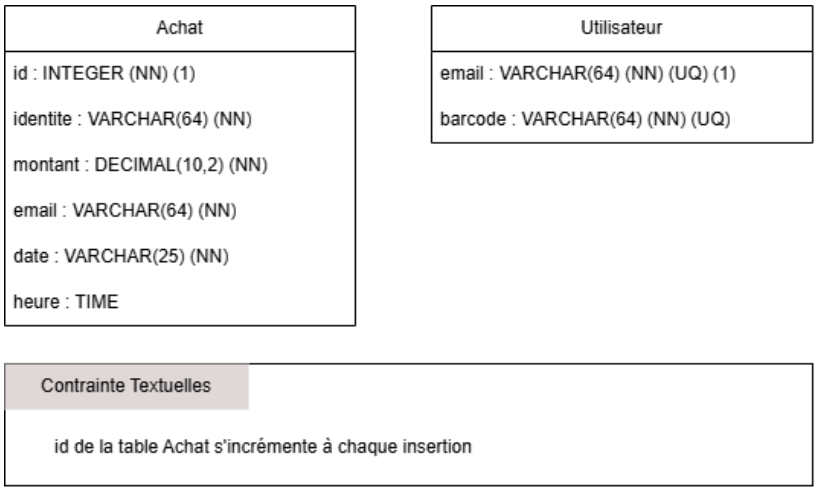



**Figure 8 - Diagramme UML de la base de données**

# DIAGRAMME NON-TECHNIQUE (GESTION PROJET)

## DIAGRAMME DE GANTT

Un projet de création d'application mobile peut devenir très vite complexes pour cela j'ai planifié mes tâches puis je les ai exécuté un par un. Voici le diagramme de Gantt avec les dates de début, les dates de fin et le nombre de jour de chaque tâches.

| | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 1 | Durée du projet | 31 days | 09.12.2024 | 09.01.25 |
| 2 | Découverte de FlutterFlow | 3 days | 09.12.2024 | 11.12.2024 |
| 3 | Création d'un formulaire de création et connexion compte | 1 day | 12.12.2024 | 12.12.2024 |
| 4 | Mise en place de la base de données Firebase | 2 days | 12.12.2024 | 12.12.2024 |
| 5 | Faire le lien entre les formulaires et la base de données | 1 day | 13.12.2024 | 13.12.2024 |
| 6 | Création / Amelioration d'une page principale | 27 day | 13.12.2024 | 26.12.2024 |
| 7 | Création d'une base de données MySQL | 1 day | 16.12.2024 | 16.12.2024 |
| 8 | Création d'un script Python pour traiter les mails | 7 days | 16.12.2024 | 22.12.2024 |
| 9 | Création / Amelioration d'une API NodeJS | 24 days | 16.12.2024 | 09.01.2025 |
| 10 | Création d'article pour la page principale | 1 day | 20.12.2024 | 20.12.2024 |
| 11 | Création d'une page de profil | 1 day | 20.12.2024 | 20.12.2024 |
| 12 | Création d'une page historique d'achat | 1 day | 20.12.2024 | 20.12.2024 |
| 13 | Lien entre l'API et la page d'historique d'achat | 2 day | 21.12.2024 | 22.12.2024 |
| 14 | Création de la fonctionnalité de carte de fidélite | 1 day | 22.12.2024 | 22.12.2024 |
| 15 | Création de la documentation | 16 days | 25.12.2024 | 10.01.2025 |
| 16 | Diagramme de cas d'utilisation | 1 day | 25.12.2024 | 25.12.2024 |
| 17 | Diagramme UML de la base de données (MySQL) | 1 day | 26.12.2024 | 26.12.2024 |
| 18 | Création de la fonctionnalité de mot de passe oublié | 2 days | 26.12.2024 | 27.12.2024 |
| 19 | Création de la fonctionnalité de changement de mot de passe | 2 days | 26.12.2024 | 27.12.2024 |
| 20 | Diagramme d'activité | 4 days | 27.12.2024 | 29.12.2024 |
| 21 | Diagramme de classe | 2 days | 30.12.2024 | 31.12.2024 |
| 22 | Diagramme de séquence | 2 days | 01.01.2025 | 02.01.2025 |
| 23 | Screenshot et explication de l'application | 2 days | 03.01.2025 | 04.01.2025 |
| 24 | | | | |
| 25 | | | | |
| 26 | | | | |

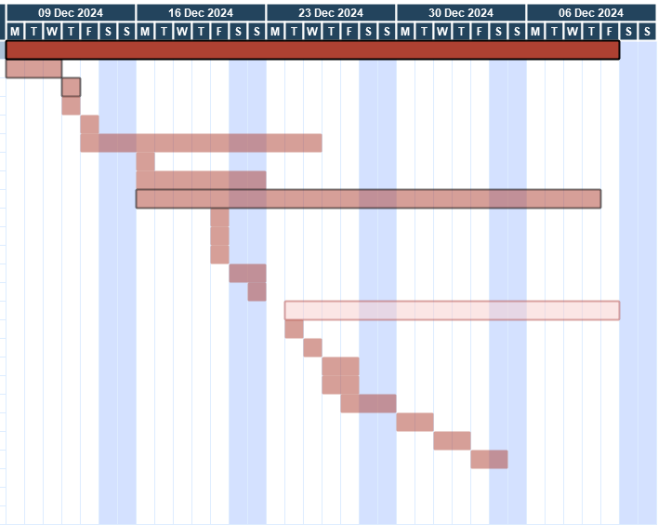



**Figure 9 - Diagramme de Gantt**

## INTERFACE UTILISATEUR

### CHARTE GRAPHIQUE

La charte graphique de l'application est sur un style minimaliste, il n'y a pas beaucoup de couleur (majorité de nuance de blanc et de noir). J'ai essayé de faire le plus simple possible en terme d'UX, c'est-à-dire qu'il n'y a pas plusieurs dizaines de boutons par page, et les boutons sont clairement visible pour les utilisateurs. L'application dispose également d'un mode clair et d'un mode sombre.

La couleur principale de l'application est la couleur bleu-violette avec le code hexadécimal (#4B39EF)

Voici la palettes de couleurs de l'application :

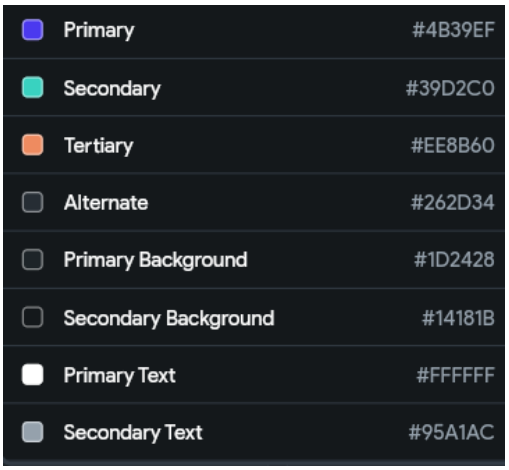



Figure 10 - Palette (#1)

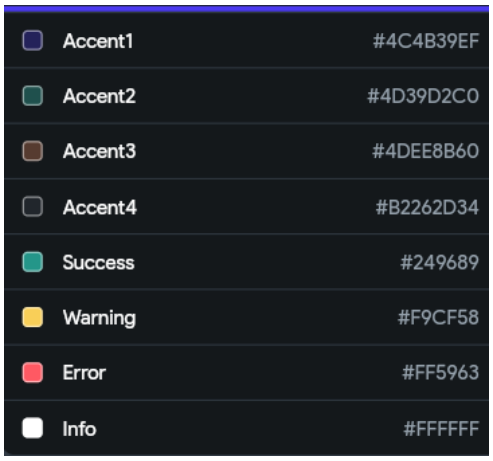



Figure 11 - Palette (#2)

# PRESENTATION DES DIFFERENTES PAGES

## CONNEXION / INSCRIPTION

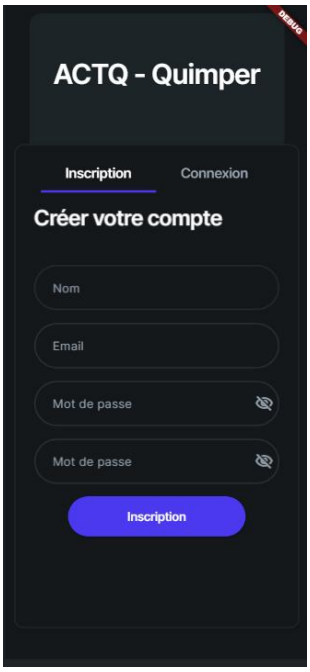



Figure 12 - Page d'inscription

La page d'inscription permet aux utilisateurs de créer leurs comptes. Pour cela il va demander le nom, l'adresse email et un mot de passe.

Lorsque l'utilisateur va appuyer sur « **Inscription** », une requête avec le type d'opération « **registerUser** » va être envoyé au serveur contenant l'adresse email de l'utilisateur. Une fois que la réponse du serveur est bon, on va créer le compte utilisateur avec Firebase. Une fois que le compte est créer l'utilisateur va être redirigé vers la page d'accueil.

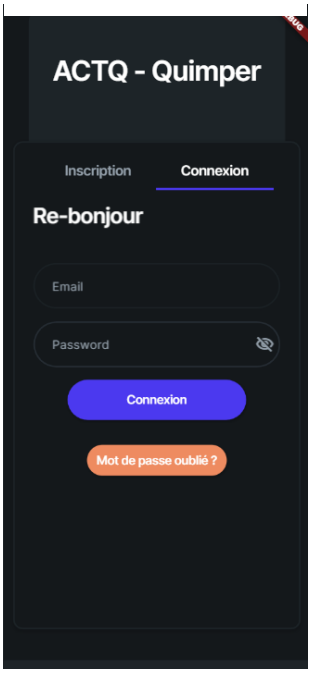



Figure 13 - Page de connexion

La page de connexion permet aux utilisateurs ayant déjà un compte de se connecter à leurs compte. Pour confirmer son identité l'utilisateur doit entrer son adresse mail avec son mot de passe. Lorsque l'utilisateur va appuyer sur « **Connexion** » on va authentifier l'utilisateur en utilisant la base de données Firebase.

Le second bouton permet d'être redirigé vers la page pour réinitialiser son mot de passe en cas d'oubli.

## MOT DE PASSE OUBLIE

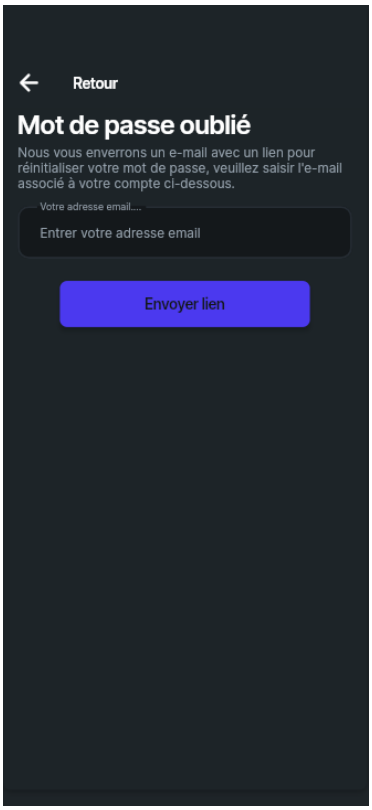



Figure 15 - Page de mot de passe oublié

Cette page permet de réinitialiser le mot de passe utilisateur. L'utilisateur doit rentré son adresse mail puis cliquez sur « **Envoyer lien** ». Une fois que l'utilisateur à cliqué sur le bouton, un mail va être envoyé à l'adresse mail présent dans le champ.

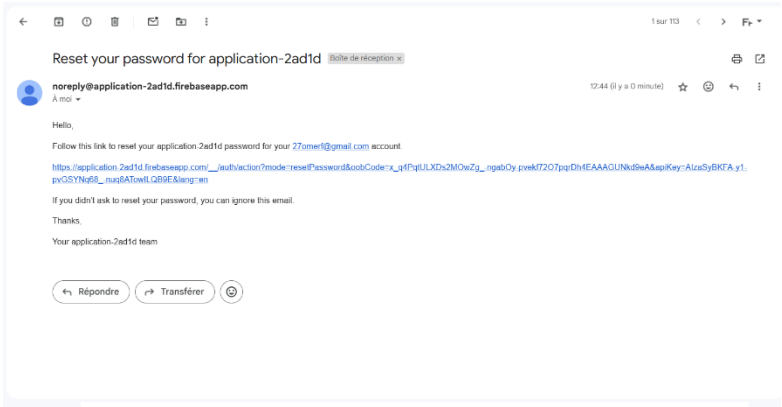



Figure 14 - Mail reçu lorsqu'on demande de réinitialiser

Le mail envoyé est comme ceci, l'utilisateur doit cliqué sur le lien pour réinitialiser son mot de passe.

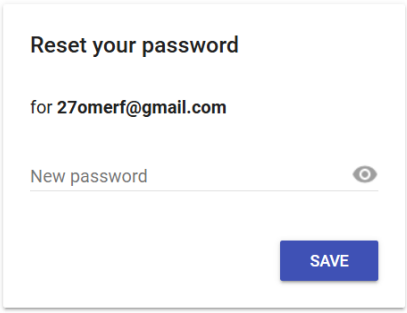



Figure 16 - Formulaire pour créer un nouveau mot de passe

Une fois que l'utilisateur clique sur le lien présent dans le mail, il va être redirigé vers une page comme ceci. Cette page est constitué uniquement d'un formulaire. L'utilisateur doit rentré son nouveau mot de passe est confirmé. Une fois qu'il clique sur « **Save** » le mot de passe appartenant à l'utilisateur dans Firebase sera mis à jour.

## PAGE PRINCIPALE


Figure 18 - Haut de la page d'accueil


Figure 17 - Bas de la page d'accueil

La page principale est constitué de trois parties, la première est un carrousel contenant les différents articles mis en avant, la seconde partie contient des photothèques de différents événement récent et la dernière partie est la barre de navigation entre les différentes page.

## LES ARTICLES


Figure 21 - Article de présentation


Figure 20 - Article concernant l'élection


Figure 19 - Article concernant le CA

Il y a un total de trois articles dans l'application, ils traitent de différents sujet d'actualités et des sujets généraux. Comme sujet d'actualité on peut retrouver l'élection du nouveau présidents de l'association. Les deux autres articles traitent des informations généraux comme le conseil d'administration et un article de présentation de l'association.

## PHOTOTHEQUE


Figure 23 - Evenement REVA


Figure 22 - Evenement Fête Enfant

La section photothèque permet de retrouver les publications avec photos publiées sur Facebook. Les utilisateurs pourront consulter les évènements antérieurs de l'association.

Dans les deux exemples que j'ai décidé d'implémenter sont, **Figure 23** un évènement pour la fête de la musique organisée par la MPT de Penhars à Quimper, l'association y avait un stand dans lequel il servait des spécialités turque et d'autre choses (Döner, thé turc, glaces italiennes.....)

Dans la **Figure 22** nous retrouvons l'évènement organisé pour la fête des enfant le 23 Avril 2024, durant lequel les enfants ont participé à un évènement. Durant cette évènement la maire de Quimper à également montré une vidéo dans laquelle elle « délivrait » son poste à une enfant (une tradition effectué en Turquie).

## CARTE FIDELITE

L'association comporte également une épicerie dans lequel se trouve une caisse enregistreuse, la caisse enregistreuse peut envoyé des mails aux clients avec leurs ticket de caisse, on envoie une copie à l'adresse mail de l'association pour suivre les habitudes d'achat des consommateurs.

En réfléchissant à comment je pourrais implémenter la fonctionnalité pour afficher les derniers achat des clients, j'ai pensé à traiter les mails reçu grâce à un script python et puis de les sauvegarder dans une base de données (dans l'application FlutterFlow j'utilise des données hypothétique car je n'ai pas le droit de traiter les données utilisateurs pour un projet qui n'est pas pour l'association).

Donc pour que ça marche on doit envoyé notre ticket de caisse (format pdf) à l'adresse mail « teamcroissantcoca@gmail.com » (c'est mon mail utilisé pour des projets utilisant les mails). Voici un exemple de ticket de caisse :

ACTQ Quimper

Epicerie Spicy

Date : 24/12/2024 à 15:38:35

Client : Omer
Mail : 27omerf@gmail.com

Vendeur : Mr. X

| | |
|---|---|
| 1x Biscuit | 7.5€ |
| 5x Eau Minérale | 23€ |
| Total | 28.58€ |

Figure 24 - Exemple de ticket de caisse

Le ticket de caisse va ensuite être traité puis les données vont être sauvegardé dans la base de données MySQL.

Les utilisateurs pourront ensuite voir leurs derniers achats directement dans l'application avec la page ci-dessous.

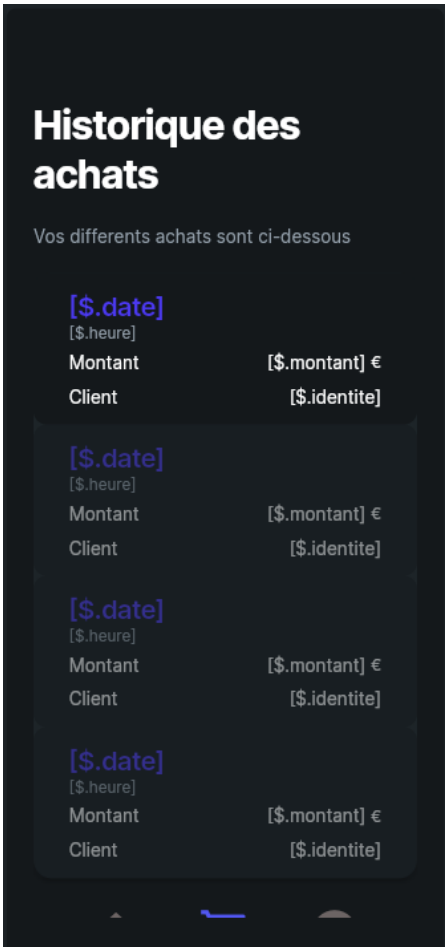



Figure 25 - Historique d'achat (Vue Code)

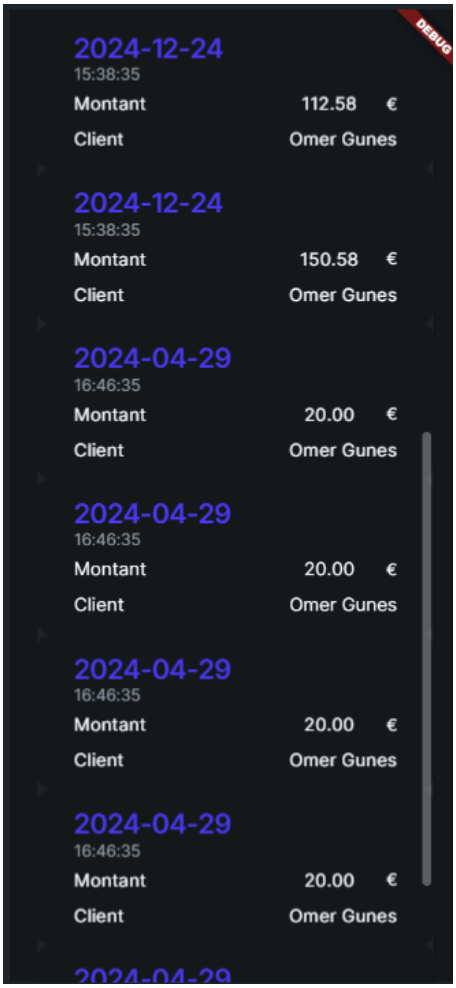



Figure 26 - Historique d'achat (Vue utilisateur)

## PROFIL

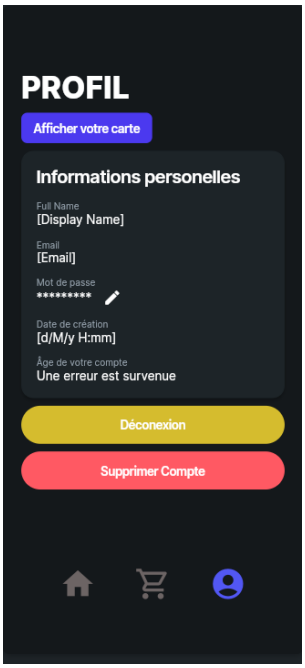



Figure 27 - Profil (Vue Code)

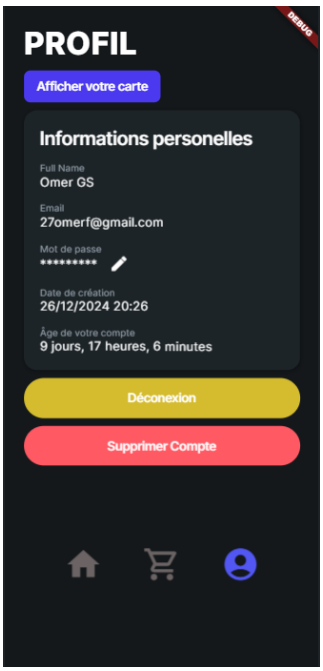



Figure 28 - Profil (Vue utilisateur)

Dans la page de profil permet d'afficher sa carte de fidélité, de voir ces informations personnelles et de pouvoir soit se déconnecter, soit de supprimer son compte. En cliquant sur déconnecter, l'utilisateur va être déconnecté et redirigé vers la page de connexion/inscription. En cliquant sur supprimer le compte de l'utilisateur va être supprimé de la base de données Firebase et MySQL.

## CARTE DE FIDELITE

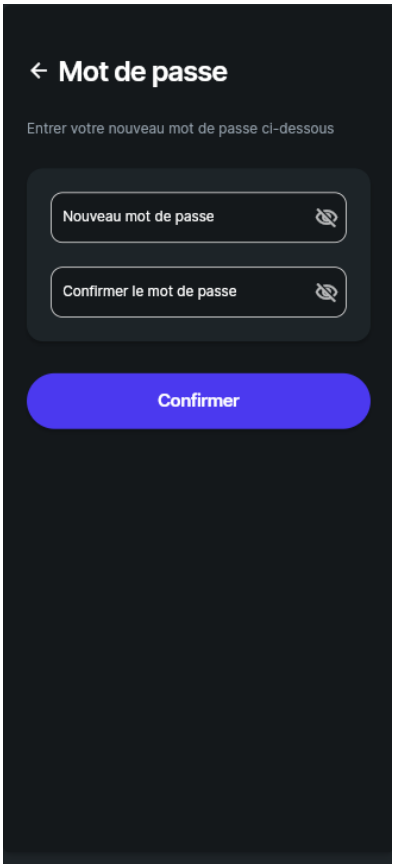

Figure 29 - Page de fidélité (Vue Code)


Figure 30 - Page de fidélité (Vue Utilisateur)

Cette page récupère le code barre enregistrer via l'API en lui donnant le type d'opération « **barcode** » et le mail de l'utilisateur. L'API va lui renvoyé le numéro associé au client et puis va l'affiché sous forme de code barre en codage 128. Compatible avec la caisse enregistreuse ABM utilisé dans l'épicerie de l'association. La génération de code barre est effectué de manière automatique à la création du compte utilisateur dans l'application.

Les possibilités d'amélioration sont plus tard de pouvoir les intégrer dans Apple Pay et dans Google Wallet pour que les utilisateurs accèdes plus simplement à la carte. On peut également passé à un système NFC, pour plus de sécurité, car ici n'importe qui peut usurper l'identité en connaissant les numéros composant le code barre.

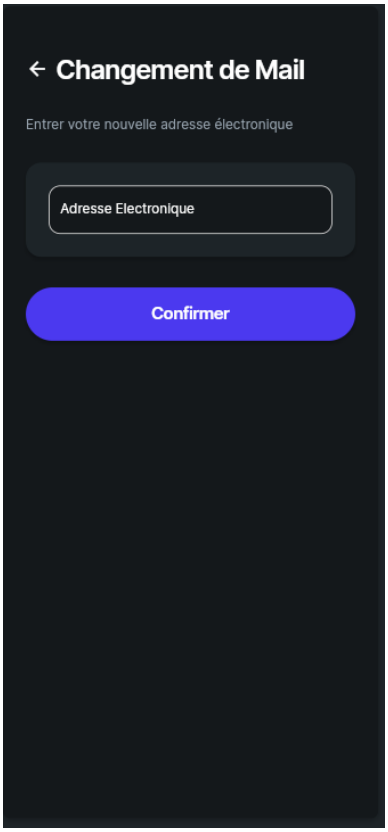
## CHANGEMENT DE MOT DE PASSE




L'utilisateur peut également changé le mot de passe de son compte en cliquant sur le crayon à côté de mot de passe dans profil utilisateur. Il va être redirigé sur cet page dans lequel il pourra changer son mot de passe.

Figure 31 - Changement de mot de passe

## CHANGEMENT D'ADRESSE MAIL




J'ai également créer cet page, son but est que l'utilisateur puisse changer son adresse mail, mais ceci ne fonctionne pas à cause d'un bug avec Firebase, l'adresse mail changé uniquement dans le serveur MySQL avec un appel API. Le serveur Firebase ne changeait pas le mail, donc j'ai décidé de mettre cette page de côté.

**Figure 32 - Changement de Mail**

## CUSTOM FUNCTION

Dans l'application FlutterFlow j'ai utilisé une custom function pour calculer le nombre de mois/jours/heure/minute écoulé depuis la création du compte de l'utilisateur dans la page de profil.

Voici le code de la custom function :

```
const pluckDeep = key => obj => key.split('.').reduce((accum, key) => accum[key],
obj)
const compose = (...fns) => res => fns.reduce((accum, next) => next(accum), res)

const unfold = (f, seed) =>import 'dart:convert';
import 'dart:math' as math;

import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:intl/intl.dart';
import 'package:timeago/timeago.dart' as timeago;
import '/flutter_flow/lat_lng.dart';
import '/flutter_flow/place.dart';
import '/flutter_flow/uploaded_file.dart';
import '/flutter_flow/custom_functions.dart';
import '/backend/backend.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import '/auth/firebase_auth/auth_util.dart';

String? accountAge(DateTime? creationDate) {
  /// MODIFY CODE ONLY BELOW THIS LINE

  if (creationDate == null) {
    return 'Date de création non fournie';
  }

  final now = DateTime.now();
  final difference = now.difference(creationDate);

  final years = difference.inDays ~/ 365;
  final months = (difference.inDays % 365) ~/ 30;
  final days = (difference.inDays % 365) % 30;
  final hours = (difference.inHours % 24);
  final minutes = (difference.inMinutes % 60);

  List<String> ageParts = [];

  // Si l'ancienneté est supérieure ou égale à un an, on affiche les années
  if (years > 0) {
    ageParts.add('$years an${years > 1 ? 's' : ''}');
  }

  // Si l'ancienneté est supérieure ou égale à un mois, on affiche les mois
  if (months > 0) {
    ageParts.add('$months mois');
  }

  // Si l'ancienneté est supérieure ou égale à un jour, on affiche les jours
  if (days > 0) {
    ageParts.add('$days jour${days > 1 ? 's' : ''}');
  }

  // Si l'ancienneté est inférieure à un an, on affiche les heures et minutes
  if (years == 0) {
    if (hours > 0) {
      ageParts.add('$hours heure${hours > 1 ? 's' : ''}');
    }
    if (minutes > 0) {
      ageParts.add('$minutes minute${minutes > 1 ? 's' : ''}');
    }
  }

  // Si aucune partie n'a été ajoutée, on retourne "moins d'une minute"
  if (ageParts.isEmpty) {
    return 'moins d\'une minute';
  }

  return ageParts.join(', ');

  /// MODIFY CODE ONLY ABOVE THIS LINE
} {
  const go = (f, seed, acc) => {
    const res = f(seed)
    return res ? go(f, res[1], acc.concat([res[0]])) : acc
  }
  return go(f, seed, [])
}
```

## GLOSSAIRE

**Token :** Un token est une chaîne de caractères unique utilisée pour authentifier une application ou un utilisateur, garantissant l'accès sécurisé à des ressources ou services spécifiques.

**API :** Une API (Interface de Programmation d'Applications) est un ensemble de protocoles permettant à plusieurs applications de communiquer entre eux.

## TABLE DES FIGURES

Figure 1 - Fonctionnement Générale du projet ..... 6
Figure 2 - Requête de l'application vers /flutterFlow ..... 8
Figure 3 - Fichier .env (masqué) ..... 13
Figure 4 - Flow Chart de l'application ..... 14
Figure 5 - Diagramme de cas d'Utilisation ..... 14
Figure 6 - Diagramme de classe ..... 15
Figure 7 - Diagramme de séquence ..... 15
Figure 8 - Diagramme UML de la base de données ..... 16
Figure 9 - Diagramme de Gantt ..... 17
Figure 10 - Palette (#1) ..... 18
Figure 11 - Palette (#2) ..... 18
Figure 12 - Page d'inscription ..... 19
Figure 13 - Page de connexion ..... 19
Figure 14 - Mail reçu lorsqu'on demande de réinitialiser ..... 20
Figure 15 - Page de mot de passe oublié ..... 20
Figure 16 - Formulaire pour créer un nouveau mot de passe ..... 20
Figure 17 - Bas de la page d'accueil ..... 21
Figure 18 - Haut de la page d'accueil ..... 21
Figure 19 - Article concernant le CA ..... 22
Figure 20 - Article concernant l'élection ..... 22
Figure 21 - Article de présentation ..... 22
Figure 22 - Evenement Fête Enfant ..... 23
Figure 23 - Evenement REVA ..... 23
Figure 24 - Exemple de ticket de caisse ..... 24
Figure 25 - Historique d'achat (Vue Code) ..... 25
Figure 26 - Historique d'achat (Vue utilisateur) ..... 25
Figure 27 - Profil (Vue Code) ..... 26
Figure 28 - Profil (Vue utilisateur) ..... 26
Figure 29 - Page de fidélité (Vue Code) ..... 27

Figure 30 - Page de fidélité (Vue Utilisateur) .....27
Figure 31 - Changement de mot de passe .....28
Figure 32 - Changement de Mail .....29