ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

WEB
SPHERE(FLUTTER)

Rapport projet :

Password manager

Asmae MOUBARRIZ
Achraf AKRACHE
5IIR7-G1

Table des matières

Table de figures

Introduction

Dans le cadre de ce projet, nous avons eu l'opportunité de concevoir et de développer une application multiplateforme complète et innovante, mettant en œuvre des technologies modernes et des pratiques avancées. Ce travail nous a permis de relever des défis techniques variés, allant de l'intégration de mesures de sécurité robustes, comme le chiffrement des données et l'utilisation de Firebase, à la création d'une interface utilisateur ergonomique et fluide grâce à Flutter. Parallèlement, nous avons perfectionné nos compétences en gestion collaborative en exploitant les fonctionnalités de Git pour assurer une coordination efficace au sein de l'équipe.

Ce projet a également été l'occasion d'intégrer des fonctionnalités avancées, tout en optimisant l'expérience utilisateur pour la rendre plus intuitive. Ces réalisations témoignent non seulement de notre maîtrise des technologies de développement multiplateforme, mais aussi de notre capacité à gérer des bases de données et à intégrer des services cloud.

En somme, cette expérience nous a permis de renforcer notre savoir-faire technique, tout en développant nos compétences professionnelles, comme la gestion de projet et le travail en équipe. Elle nous prépare ainsi à aborder des projets encore plus ambitieux et à relever des défis technologiques complexes dans l'avenir.

Présentation du projet

I. Contexte du projet

Pourquoi une application de gestion de mots de passe ?

Dans un monde où les utilisateurs doivent gérer de nombreux comptes en ligne pour des services tels que les réseaux sociaux, les plateformes professionnelles ou encore les achats en ligne, il devient essentiel de garantir la sécurité des mots de passe. Cependant, la majorité des utilisateurs utilisent des mots de passe simples, souvent réutilisés sur plusieurs plateformes, ce qui augmente les risques de piratage et de vol de données.

Pour répondre à ces défis, l'idée d'une application de gestion de mots de passe est née. Ce projet vise à offrir aux utilisateurs un outil simple, sécurisé et pratique pour gérer leurs mots de passe de manière centralisée et automatisée. Il garantit également une protection des données personnelles grâce à l'utilisation d'algorithmes de chiffrement modernes.

II. Objectifs du projet

- ✓ **Sécurisation et gestion des données personnelles**

L'objectif principal de ce projet est de concevoir une application répondant aux attentes suivantes :

1. Offrir un moyen sûr pour stocker et gérer les identifiants des utilisateurs.
2. Automatiser la génération de mots de passe forts, répondant aux normes de sécurité modernes.
3. Proposer une interface intuitive et ergonomique pour une utilisation simplifiée.
4. Garantir la confidentialité et la sécurité des données grâce au chiffrement des mots de passe.

Besoins Fonctionnelles

III. Étude des besoins et fonctionnalités à intégrer

Avant de commencer le développement, nous avons mené une analyse approfondie des besoins des utilisateurs. Les fonctionnalités suivantes ont été identifiées comme essentielles :

- ✓ Authentification sécurisée pour protéger l'accès aux données.
- ✓ Gestion complète des comptes enregistrés, incluant la création, l'affichage, la mise à jour et la suppression.
- ✓ Génération automatique de mots de passe forts, pour encourager les utilisateurs à ne pas réutiliser les mêmes mots de passe.
- ✓ Affichage sécurisé des mots de passe, masqués par défaut pour éviter tout accès non autorisé en cas de consultation publique.

A. Gestion des utilisateurs (authentification et comptes)

L'authentification est au cœur de l'application. Les utilisateurs peuvent :

- ➢ Créer un compte avec une adresse email, un numéro de téléphone et un code confidentiel.
- ➢ Se connecter pour accéder à leurs informations.

B. Gestion des mots de passe

Une fois connectés, les utilisateurs peuvent :

- ➢ Ajouter des comptes en renseignant des informations comme :
- ➢ Nom de l'application (ex. : Gmail).
- ➢ Identifiant de connexion (ex. : user123@gmail.com).
- ➢ Mot de passe, généré automatiquement ou saisi manuellement.
- ➢ Mettre à jour ou supprimer des comptes existants.

Tous les mots de passe sont chiffrés avant leur stockage dans Firebase, garantissant leur confidentialité même en cas d'accès non autorisé à la base de données.

IV. Présentation des Fonctionnalités Principales

- **Connexion et authentification utilisateur**

Les utilisateurs saisissent leur email et mot de passe pour accéder à l'application. En cas d'erreur, un message clair s'affiche.

- **Liste des comptes enregistrés**

Présentation des comptes sauvegardés sous forme de liste.

Exemple :

Nom du compte : Gmail.

Identifiant : user123@gmail.com.

Mot de passe : ******** (masqué).

- **Ajout de compte avec génération automatique de mots de passe**

Fonctionnalité permettant de générer un mot de passe complexe et unique, tel que : Ht@93Xkz_

- **Mise à jour et suppression des comptes**

Deux boutons à coté de chaque Mot de passe qui permettent à l'utilisateur de modifier ou de supprimer le mot de passe

- **Chiffrement des mots de passe**

Les mots de passe sont chiffrés avant leur enregistrement dans Firebase, garantissant leur sécurité même en cas d'accès non autorisé à la base de données.

Conception et développement

V. Conception Générale

C. Diagramme de cas d'utilisation

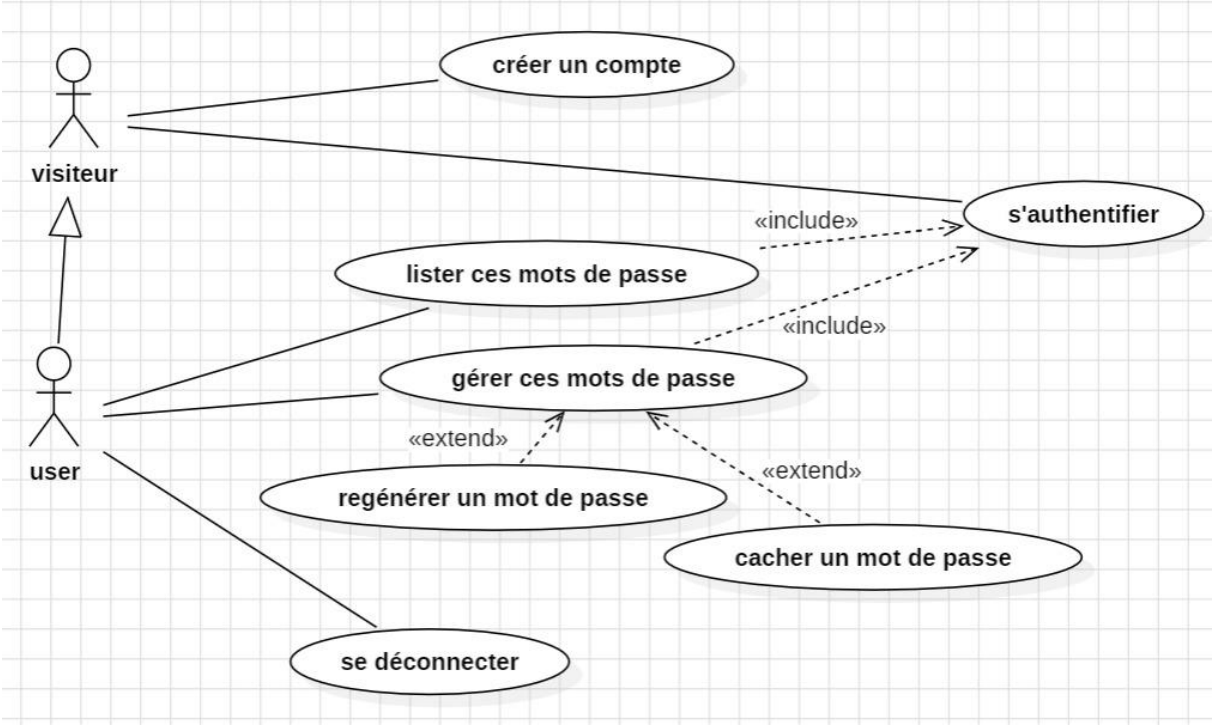


Figure 1 : Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation illustre les interactions entre deux types d'acteurs : **visiteur** et **user**. Il détaille les fonctionnalités de notre système de gestion des mots de passe.

1. **Acteurs** :
    - **Visiteur** : Représente une personne non authentifiée.
    - **User** : Représente une personne authentifiée avec des droits d'accès aux fonctionnalités de gestion des mots de passe.
2. **Cas d'utilisation** :
    - **Créer un compte** :
        - Permet aux visiteurs de s'inscrire et de devenir des utilisateurs.
    - **S'authentifier** :
        - Cas central permettant aux visiteurs et utilisateurs de se connecter.
    - **Lister ces mots de passe** :

        - Permet aux utilisateurs de visualiser tous les mots de passe stockés.
    - **Gérer ces mots de passe** :
        - Permet de mettre à jour, supprimer ou organiser des mots de passe.
    - **Cacher un mot de passe** :
        - Une fonctionnalité **étendue** de la gestion des mots de passe, spécifiquement pour masquer les informations sensibles.
    - **Régénérer un mot de passe** :
        - Aide à générer un nouveau mot de passe sécurisé en cas de besoin.
    - **Se déconnecter** :
        - Permet aux utilisateurs de se déconnecter de leur compte.
3. **Relations** :
    - **Inclure** :
        - Le cas d'utilisation **S'authentifier** est une partie essentielle de **Créer un compte** et **Lister ces mots de passe**, car l'authentification est une condition préalable.
    - **Étendre** :
        - **Cacher un mot de passe** est une extension des fonctionnalités de **Gérer ces mots de passe**, avec un comportement supplémentaire.

D. Diagramme de classe

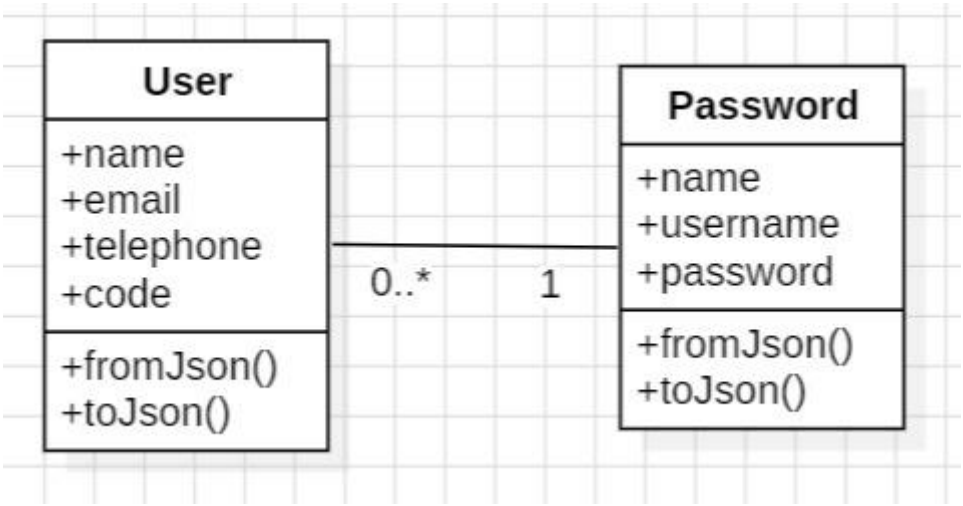


Figure 2 : Diagramme de classe

Le diagramme de classe représente deux principales classes : **User** (Utilisateur) et **Password** (Mot de passe).

1. **Classe User** :
    - **Attributs** :
        - name : Représente le nom complet de l'utilisateur.
        - email : Stocke l'adresse e-mail de l'utilisateur.
        - telephone : Contient le numéro de téléphone.
        - code : code confidentiel
    - **Méthodes** :
        - fromJson() : Méthode pour remplir l'objet User à partir d'une représentation JSON.
        - toJson() : Convertit l'objet User en une représentation JSON.
2. **Classe Password** :
    - **Attributs** :
        - name : Probablement une étiquette ou un identifiant pour le mot de passe (par exemple, "E-mail professionnel").
        - username : Stocke le nom d'utilisateur associé au mot de passe.
        - password : Contient le mot de passe réel.
    - **Méthodes** :
        - fromJson() : Comme pour User, cette méthode initialise l'objet à partir d'un format JSON.
        - toJson() : Convertit l'objet Password en un format JSON.
3. **Relation** :
    - Une relation *0... à 1** est modélisée entre User et Password.
    - Chaque utilisateur peut gérer **zéro ou plusieurs** mots de passe, mais chaque mot de passe appartient à **exactement un utilisateur**.

VI. Choix des technologies

Flutter : utilisé pour développer une interface utilisateur intuitive et multiplateforme (iOS et Android).

Firebase : choisi pour ses fonctionnalités d'authentification, de base de données en temps réel, et de gestion des utilisateurs.

VII. Structure de l'application

L'application est organisée en trois couches principales :

- Interface utilisateur : Écrans de connexion, liste des comptes, et formulaires.
- Logique métier : Gestion des fonctionnalités (CRUD) et génération automatique de mots de passe.
- Firebase : Stockage et sécurisation des données.

VIII. Méthodologie utilisée :

E. Collaboration et outils techniques

Ce projet a été réalisé en binôme, mettant en avant un travail collaboratif basé sur :

1. Les cours et travaux pratiques suivis en classe, qui ont fourni une base théorique solide.
2. La documentation technique officielle de Flutter et Firebase, utilisée pour approfondir les connaissances et résoudre des problèmes spécifiques.
3. Git pour la gestion du code source et la facilitation de la collaboration.

4. Grâce à Git, nous avons pu partager, synchroniser et gérer efficacement les modifications apportées au projet, réduisant ainsi les conflits de version.

Cette méthodologie nous a permis de structurer et d’organiser notre travail pour aboutir à une application fonctionnelle et sécurisée.

F. Structuration du projet : structure modulaire

L'architecture du projet a été pensée pour garantir une organisation claire et une maintenance facilitée. Le répertoire principal lib est structuré en plusieurs sous-dossiers, chacun jouant un rôle spécifique :

1. **models** :
   Ce dossier contient les modèles de données essentiels pour l'application.
   - Password.dart : Gestion des informations liées aux mots de passe, avec une éventuelle intégration de la logique de sécurité ou de validation.
   - User.dart : Représentation des utilisateurs de l'application, avec leurs attributs et comportements associés.
2. **pages** :
   Ce dossier regroupe les différentes pages constituant l'interface utilisateur.
   - HomePage.dart : Page principale, qui sert de point d'entrée pour l'expérience utilisateur après l'authentification.
   - LoginPage.dart : Interface dédiée à l'authentification des utilisateurs.
3. **services** :
   Ce répertoire contient les services et utilitaires nécessaires au bon fonctionnement de l'application.
   - EncryptionHelper.dart : Gestion de la sécurité, en particulier pour le chiffrement des données sensibles.
   - firestore.dart : Intégration avec la base de données Firebase pour les opérations CRUD.

- firebase_options.dart : Fichier de configuration Firebase contenant les paramètres spécifiques au projet.

Cette structure modulaire permet une séparation claire des responsabilités, rendant le code plus lisible, maintenable et évolutif. Elle favorise également une collaboration efficace en permettant à chaque membre de l'équipe de travailler sur un composant spécifique sans interférer avec les autres.

IX. Défis rencontrés

1. Intégration de Firebase avec Flutter

Configuration initiale complexe, nécessitant des ajustements dans les fichiers de projet.

Solution : Utilisation de tutoriels et de documentation pour comprendre la configuration de Firebase.

2. Mise en place du chiffrement des mots de passe

Recherche d’une bibliothèque de chiffrement adaptée à Flutter.

Solution : On a opté au Chiffrement AES, et adoption de bibliothèques comme encrypt pour implémenter le chiffrement.

3. Collaboration en binôme avec Git

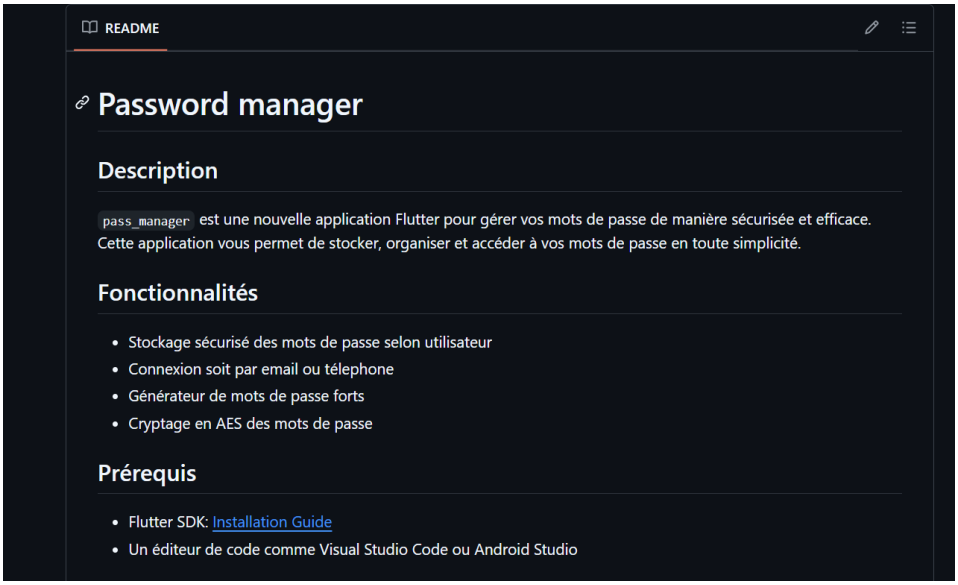
Gestion des conflits de fusion lors de la synchronisation des modifications.

Solution : Communication régulière et gestion des branches pour éviter les conflits Git.

Conclusion

Ce projet a représenté une opportunité unique de concevoir une application complète et innovante, intégrant à la fois sécurité, ergonomie et collaboration. En combinant le chiffrement des données et les services de Firebase pour garantir une sécurité optimale, l'utilisation de Flutter pour offrir une interface utilisateur ergonomique, et l'adoption de Git pour une collaboration efficace, nous avons pu développer un produit robuste et intuitif. Ce travail nous a permis de renforcer nos compétences en développement d'applications mobiles multiplateformes, en gestion de bases de données et en intégration de services cloud, tout en perfectionnant notre capacité à travailler en équipe et à gérer des projets complexes. En outre, nous avons enrichi notre application avec des fonctionnalités avancées telles que l'authentification biométrique, le partage sécurisé de mots de passe et l'optimisation de l'interface utilisateur pour offrir une expérience fluide et conviviale. En conclusion, cette expérience a été particulièrement enrichissante, tant sur le plan technique que professionnel, nous préparant à relever des défis technologiques plus ambitieux à l'avenir.

Annexes



Github link : https://github.com/asmaembr/pass_manager

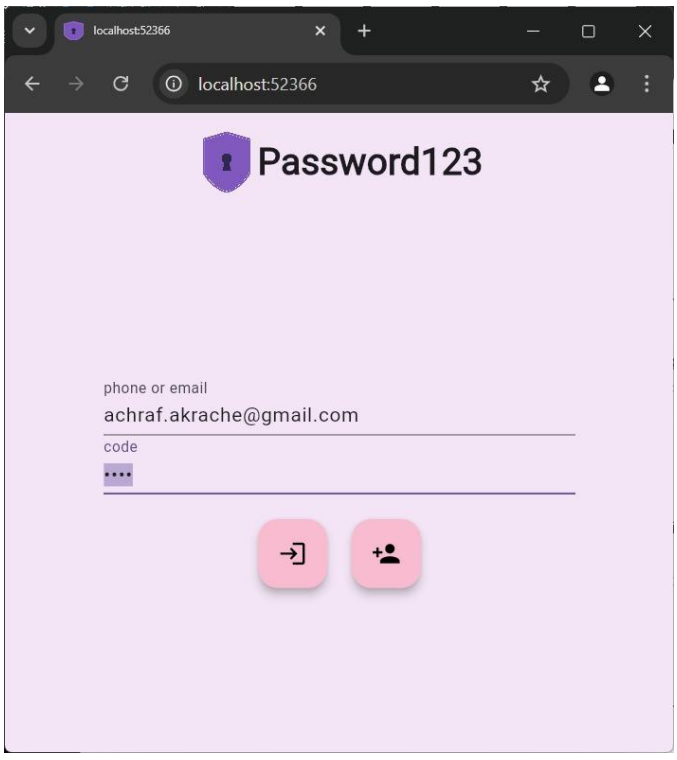


Figure 3 : Page de Connexion (Interface Web)

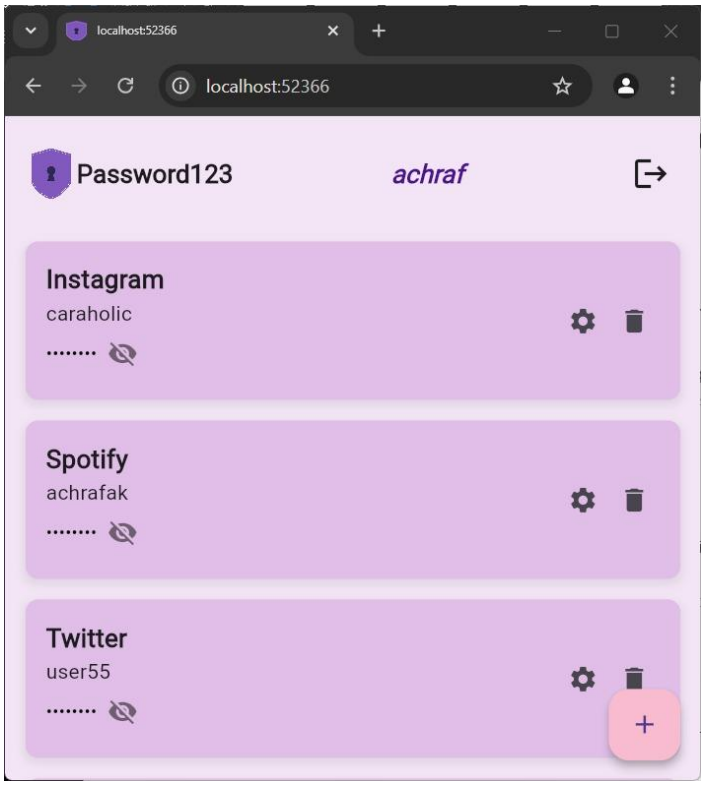


Figure 4 : Dashboard (Interface Web)

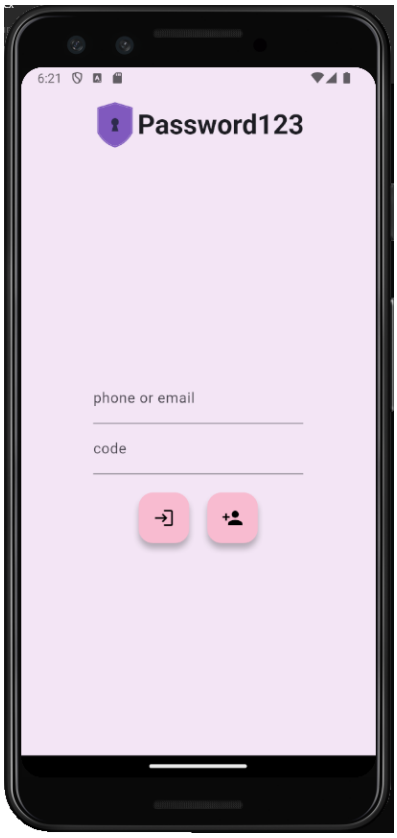


Figure 5 : Page de connexion (Interface mobile)

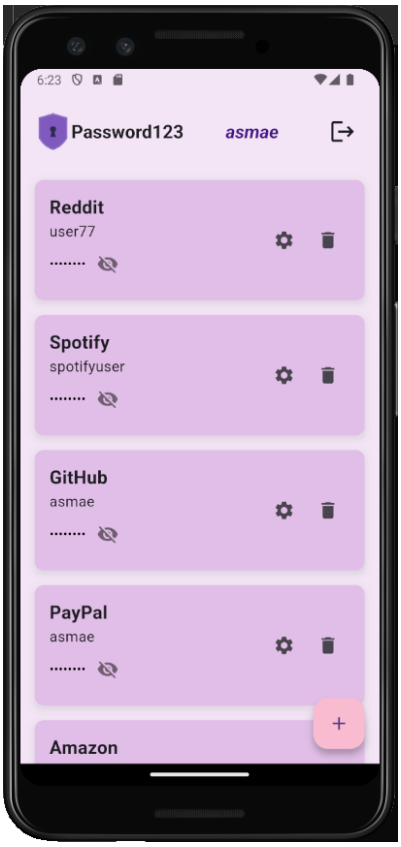


Figure 6 : Dashboard (Interface mobile)

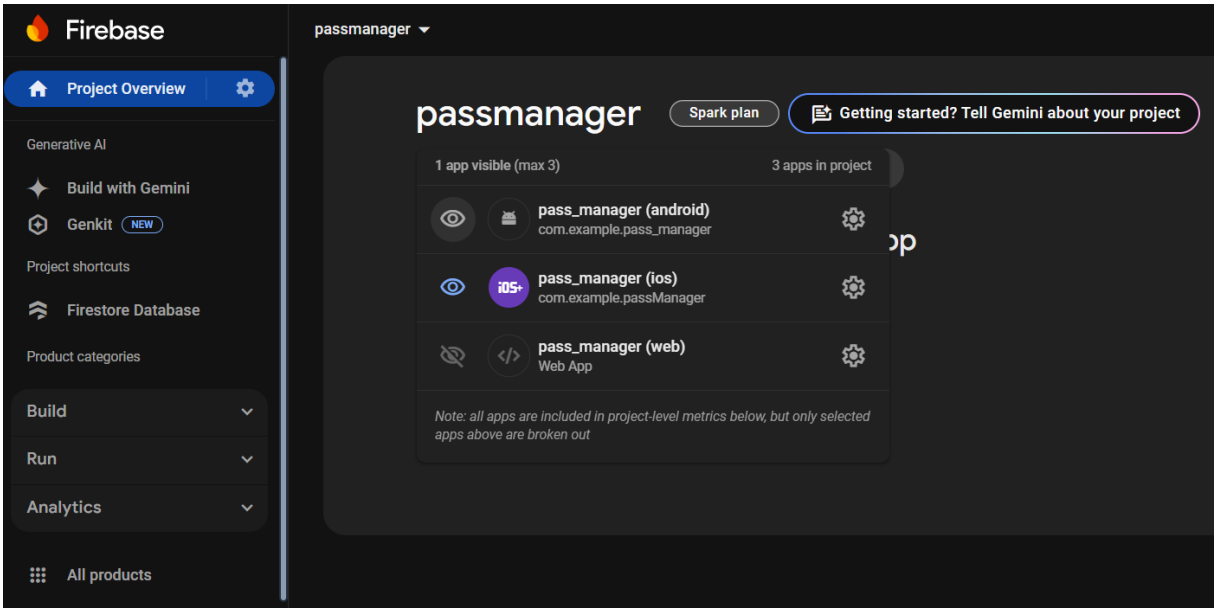


Figure 7 : Firebase console