Université des sciences et de la technologie

Houari-Boumédiène

Rapport Technique du Projet Flutter – Authentification app

Module: ***Développement mobile***

Projet final

Réalisé par:

KHOUAS Ihsene Noor
SADAOUI Sara Rahma

Tables des matières

1. Introduction Générale

2. Fonctionnement Détaillé de l'Application(implémentation)
- 2.1 Inscription
- 2.2 Connexion classique (email / mot de passe)
- 2.3 Connexion par reconnaissance faciale
- 2.4 Accès et modification du profil

3. Mise en Place de l'Infrastructure Supabase
- 3.1 Authentification Supabase
- 3.2 Table des utilisateurs
- 3.3 Espace de stockage (Supabase Storage)

4. Détails Techniques sur les Classes et Dépendances
- 4.1 Structure de l'Application Flutter
- 4.2 Dépendances Flutter Utilisées
- 4.3 Composants Personnalisés et Gestion des Erreurs

5. Conclusion

1. Introduction Générale

Dans le cadre de notre projet, nous avons développé une application mobile Android en utilisant Flutter. Cette application met en œuvre un système d'authentification sécurisé basé sur deux approches complémentaires : une méthode classique par email et mot de passe, et une méthode plus moderne reposant sur la reconnaissance faciale.

L'objectif principal de notre application est d'offrir un moyen pratique et sécurisé pour qu'un utilisateur puisse créer un compte en associant une image de référence, puis se connecter ultérieurement en utilisant cette même image.

2. Fonctionnement Détaillé de l'Application(implémentation)

2.1 Inscription

L'inscription commence par un formulaire classique. Après validation du nom, de l'email et du mot de passe, l'application déclenche une demande d'accès à l'appareil photo ou à la galerie. Une fois l'image choisie, celle-ci est compressée et convertie en données binaires, puis envoyée vers Supabase Storage dans un dossier spécifique (le bucket faces). Chaque image est nommée de façon unique, souvent à partir de l'adresse email de l'utilisateur, afin d'en garantir l'identifiabilité.

Une fois l'image stockée, l'application utilise la bibliothèque supabase_flutter pour créer un compte dans Supabase Auth avec les identifiants fournis. Dès que l'opération est confirmée, l'application effectue une insertion manuelle dans la base de données Supabase, dans la table users, en enregistrant l'email, le nom et l'URL de l'image.

Cette double insertion (Auth + table users) garantit une séparation claire entre les informations d'authentification et les données personnalisées de profil.

2.2 Connexion classique (email / mot de passe)

L'écran de connexion offre la possibilité de se connecter avec l'adresse email et le mot de passe enregistrés. Si les informations sont valides, Supabase Auth valide la session.

2.3 Connexion par reconnaissance faciale

Cette fonctionnalité repose sur l'usage de Face++, un service tiers de reconnaissance faciale.

Voici comment elle fonctionne :

- Une nouvelle photo du visage est prise via la caméra.
- L'image enregistrée lors de l'inscription (URL depuis Supabase Storage) est récupérée.
- Ces deux images sont envoyées à l'API Face++, qui retourne un indice de confiance (score de similarité).
- Si le score est supérieur à 80, l'authentification est considérée comme réussie.
- Un son .wav est ensuite joué (via just_audio) pour notifier l'utilisateur d'un succès ou d'un échec.

2.4 Accès et modification du profil

Une fois connecté, l'utilisateur peut modifier:

- son nom,
- sa photo de profil,
- son mot de passe.

Toute modification exige la saisie du mot de passe actuel, même si l'utilisateur est entré via la reconnaissance faciale. Cela assure un niveau de sécurité renforcé.

La nouvelle photo est uploadée dans Supabase Storage, et l'URL mise à jour dans la table users. Le mot de passe est mis à jour via Supabase Auth.

2.5 Historique des Connexions

Une table appelée logs est utilisée pour tracer chaque connexion utilisateur. À chaque connexion (via mot de passe ou FaceID), une nouvelle entrée est enregistrée avec :

- l'UUID de l'utilisateur,
- l'adresse IP,
- la méthode utilisée (mot de passe ou reconnaissance faciale),
- la date et l'heure.

Depuis la page d'accueil, l'utilisateur peut consulter cet historique pour voir l'ensemble des connexions à son compte.

3. Mise en Place de l'Infrastructure Supabase

Un travail important a été réalisé dans Supabase pour préparer le backend.

3.1 Authentification Supabase

Le système d'authentification par email et mot de passe a été activé dans la console Supabase. À la demande de l'application, Supabase crée un nouvel utilisateur avec un identifiant unique (UUID) associé à l'email et au mot de passe fournis. L'application utilise la session retournée pour authentifier toutes les requêtes suivantes.

3.2 Table des utilisateurs

Une table users a été créée manuellement dans la base de données

Supabase (PostgreSQL). Elle contient les champs suivants :

- id : identifiant utilisateur (associé à celui de Supabase Auth),
- email : email unique de l'utilisateur,
- nom : nom complet,
- photo_url : lien public vers la photo dans Supabase Storage.

Cette table permet de lier chaque utilisateur à ses données personnelles, de les afficher dans l'application, et de les modifier si nécessaire.

3.3 Table logs

Elle trace toutes les connexions utilisateur :

- user_id,
- ip_address,
- method (password ou face_id),
- timestamp.

Elle renforce la transparence et la sécurité.

3.4 Espace de stockage (Supabase Storage)

Un bucket de stockage nommé faces a été configuré dans Supabase. Chaque image de visage est enregistrée avec un nom unique et est accessible en lecture publique via une URL. Ce choix d'ouverture permet à l'application Flutter de télécharger directement les fichiers sans authentification supplémentaire, ce qui simplifie l'architecture tout en maintenant un bon niveau de sécurité.

4. Détails Techniques sur les Classes et Dépendances

4.1 Structure de l'Application Flutter

4. Détails Techniques sur les Classes et Dépendances

4.1 Structure Principale

- **main.dart :** point d'entrée de l'application, initialise Supabase et dirige l'utilisateur selon sa session.
- **signup_screen.dart :** gère l'inscription et le téléchargement de la photo.
- **auth_service.dart :** centralise toutes les opérations liées à Supabase Auth, insertion dans users, et enregistrement des logs.
- **faceplusplus_service.dart :**

    - Capture la nouvelle image via image_picker,
    - Récupère l'image de référence (URL),
    - Appelle l'API Face++ pour comparer les deux images,
    - Analyse l'indice retourné (succès si > 80),
    - Joue un son via just_audio.

- **edit_profile_screen.dart :** permet de modifier les données du profil après saisie du mot de passe.
- **home_screen.dart :** propose une interface utilisateur claire, avec accès à l'historique des connexions (logs).

4.2 Dépendances Flutter Utilisées

Voici les principales dépendances déclarées dans pubspec.yaml avec leur utilité concrète dans le projet :

- **supabase_flutter :** bibliothèque principale pour connecter l'application à Supabase (authentification, base de données, stockage).
- **image_picker :** permet de capturer une photo avec la caméra ou d'en sélectionner une depuis la galerie.
- **image :** permet de décoder, manipuler, et comparer les pixels des images (reconnaissance faciale).
- **http :** utilisée en soutien pour certains appels réseau (optionnel ici car Supabase gère la plupart via son SDK).
- **provider :** utilisé pour gérer l'état global de l'application (par exemple savoir si un utilisateur est connecté).
- **shared_preferences :** permet de mémoriser des informations localement (ex. : email si l'option "Se souvenir de moi" est activée).
- **form_field_validator et email_validator :** facilitent la vérification des données entrées par l'utilisateur dans les champs de formulaire.
- **just_audio :** lecture des sons .wav
- **flutter_screenutil :** utilisée pour adapter les tailles d'éléments à différentes résolutions d'écran.
- **google_fonts :** permet d'intégrer facilement des polices personnalisées.
- **flutter_svg :** rend possible l'affichage d'icônes vectorielles au format SVG.

4.3 Composants Personnalisés et Gestion des Erreurs

L'application utilise également :

- Des snackbars ou dialogs pour afficher les messages d'erreur (ex : image non sélectionnée, email invalide, mot de passe incorrect…),
- Des fonctions asynchrones avec await pour garantir que les opérations réseau (auth, envoi de fichiers, récupération de données)

soient bien terminées avant de continuer,
- Des gestes utilisateurs bien guidés (ex. : retour sonore, messages explicatifs, transitions entre écrans).

4.4 Sécurité et UX

- Revalidation par mot de passe avant toute modification,
- Communication sécurisée avec API tierce (Face++),
- Enregistrement des connexions dans logs,
- Feedback utilisateur clair (sons, messages, transitions),
- Appels réseau asynchrones avec gestion des erreurs et des délais.

5. Conclusion

Ce projet nous a permis de concevoir une application mobile Android complète, fonctionnelle, et sécurisée, mettant en œuvre à la fois des technologies modernes d'authentification et des mécanismes de traitement d'image pour la reconnaissance faciale. Grâce à Flutter, nous avons pu développer une interface utilisateur fluide, intuitive et responsive. Le choix de Supabase comme backend s'est avéré pertinent, car il nous a offert un environnement intégré capable de gérer l'authentification, la base de données, et le stockage de fichiers avec une grande simplicité d'intégration.