🎓 Projet : Application de Connexion Flutter avec Authentification via API REST

Introduction

Pour ce projet final, vous allez étendre et améliorer votre **application de connexion** Flutter en intégrant une authentification dynamique via une API REST, en renforçant la sécurité, et en optimisant l'expérience utilisateur. Ce projet vous permettra de démontrer votre maîtrise des concepts de développement mobile, de consommation d'API, de sécurité des données, et de design réactif.

Objectifs du Projet

1. **Intégration d'une API REST** (**Principal**) : Remplacer le système d'authentification local par des appels HTTP à une API réelle pour gérer dynamiquement les utilisateurs.
2. **Sécurité Renforcée** : Implémenter des méthodes de hachage sécurisées pour les mots de passe (**Principal**) et utiliser des tokens d'authentification (**Optionnel**).
3. **Amélioration de l'Expérience Utilisateur (UX/UI)** (**Principal**) : Ajouter des indicateurs de chargement et assurer une conception réactive pour une meilleure accessibilité.
4. **Création d'une API Simple** (**Principal**) : Développer une petite API (dans la technologie de votre choix) pour tester l'application en renvoyant des réponses JSON appropriées.

Description du Projet

Vous pouvez cloner votre projet de l'application de connexion ou si vous préférez, vous disposerez du projet de base disponible sur le GitHub (login_app_plus.git). Ce projet comprend déjà une interface de connexion et un écran de profil utilisateur utilisant un tableau associatif local pour gérer les identifiants.

Votre tâche principale est de :

1. **Remplacer le AuthService local par des appels HTTP à une API réelle.**
2. **Développer une API simple capable de gérer les demandes de connexion et de renvoyer les réponses JSON appropriées.**
3. **Améliorer la sécurité en hachant les mots de passe.**
4. **Optimiser l'interface utilisateur en ajoutant des indicateurs de chargement et en assurant une conception réactive.**

Exigences du Projet

1. Intégration d'une API REST

- **Consommation d'une API :**
    - Utilisez la bibliothèque `http` du Dart pour effectuer des requêtes **HTTP**.
    - Remplacez les vérifications locales des identifiants par des appels à l'API pour authentifier les utilisateurs.

- **Configuration de http :**
    - Ajoutez la dépendance `http: ^1.2.2` dans votre fichier `pubspec.yaml`.
    - Utilisez des méthodes asynchrones (`async`/`await`) pour gérer les appels réseau.

2. Création d'une API Simple

- **Technologie de l'API :** Vous pouvez choisir n'importe quelle technologie pour développer votre API (PHP, Node.js, Python, etc.).
- **Endpoints :**
    - **Endpoint de Connexion (/login) :**
        - **Méthode :** POST
        - **Paramètres :** email, password
        - **Réponses :**
            - **Succès :**

```
{
  "status": 200,
  "message": "Success",
  "id": 1,
  "firstName": "Jean",
  "lastName": "Dupont",
  "email": "jean@example.com",
  "role": 2
}
```

            - **Erreur :**

```
{
  "status": 400,
  "message": "Error"
}
```

- **Sécurité de l'API :** Implémentez des mesures de sécurité appropriées, telles que la validation des entrées et l'utilisation de HTTPS.

3. Sécurité Renforcée

- **Hachage des Mots de Passe :** Utilisez des méthodes de hachage sécurisées (par exemple, SHA-256) pour stocker les mots de passe. Ne stockez jamais les mots de passe en clair.
- **Implémentation de Tokens** (Optionnel) **:**
    - Après une connexion réussie, générez un token d'authentification utilisez-le pour sécuriser les requêtes futures.
    - Stockez le token de manière sécurisée sur l'appareil de l'utilisateur (par exemple, en utilisant `flutter_secure_storage`).

4. Amélioration de l'Expérience Utilisateur (UX/UI)

- **Indicateurs de Chargement :** Ajoutez des `CircularProgressIndicator` lors des processus de connexion pour informer l'utilisateur que l'application est en train de traiter sa demande.
- **Conception Réactif :** Utilisez des widgets tels que `SingleChildScrollView` pour assurer que le formulaire de connexion est accessible sur différentes tailles d'écran et orientations.

- **Personnalisation de l'Interface :** Améliorez le thème de l'application, ajoutez des animations et utilisez des images pour rendre l'interface plus attrayante.

Instructions de Soumission

1. **Code Source :** Poussez votre code final sur un dépôt GitHub en partageant le lien public.
2. **API :** Fournissez le code source de votre API simple dans le même dépôt ou dans un dépôt séparé avec des instructions claires pour la déployer et la tester.
3. **Documentation :** Incluez un fichier README.md dans votre dépôt principal décrivant le projet, les étapes pour exécuter l'application et l'API, et toute autre information pertinente (les identifiants de connexion par exemple).
4. **Démo Vidéo (Optionnel mais souhaitable) :** Vous pouvez ajouter une courte vidéo de démonstration montrant les principales fonctionnalités de votre application.

Critères de Notation

Votre projet final sera évalué en fonction des critères suivants. Chaque critère contribue à la note finale du projet.

- **Organisation du Projet (10%) :** Votre projet doit être bien organisé avec des dossiers séparés pour les écrans, services, modèles, et widgets. Chaque fichier doit avoir un nom descriptif.
- **Intégration de l'API REST (20%) :** L'application doit effectuer des appels HTTP vers une API REST réelle pour authentifier les utilisateurs. Les réponses de l'API doivent être correctement gérées et intégrées dans l'application.
- **Sécurité des Données (10%) :** Les mots de passe doivent être hachés avant d'être envoyés à l'API.
- **Expérience Utilisateur (UX/UI) (15%) :** L'application doit inclure des indicateurs de chargement pendant les processus de connexion. L'interface doit être réactive et bien présentée sur différentes tailles d'écran.
- **Fonctionnalités Bonus (15%) :** L'implémentation de fonctionnalités supplémentaires telles que le salage des mots de passe, l'utilisation de gestionnaires d'état avancés, l'utilisation de tokens d'authentification pour sécuriser les interactions futures est essentielle, ou des améliorations UI/UX supplémentaires peut être récompensée ici.
- **Qualité du Code (10%) :** Le code doit être propre, bien structuré, et bien commenté. Il doit suivre les meilleures pratiques de développement Flutter.
- **Screenshots/Capture Vidéo (10%) :** Fournissez des preuves visuelles montrant le bon fonctionnement de votre application, incluant les principaux flux et fonctionnalités.
- **Documentation (10%) :** Fournissez une documentation incluant les instructions pour exécuter l'API et l'application Flutter, ainsi qu'une description des fonctionnalités implémentées.