Flutter todo Application avec FireBase

1. Objectif de l'application

Créer une application mobile de gestion de tâches permettant aux utilisateurs :

- De s'inscrire et se connecter via Firebase Authentication.
- De créer, consulter, modifier, et supprimer des tâches (CRUD) stockées dans Firebase Firestore.
- D'organiser les tâches par catégorie et statut (par exemple, "À faire", "En cours", "Terminé").
- De synchroniser les données en temps réel avec Firebase.
- D'ajouter des notifications pour les tâches importantes (Firebase Cloud Messaging).

---

2. Fonctionnalités principales

2.1. Gestion des utilisateurs

- **Inscription/Connexion** : Via Firebase Authentication.
    - Options : email/mot de passe, Google, et Facebook.
- **Gestion du profil utilisateur** :
    - Mise à jour du nom et de l'avatar (photo).
- **Déconnexion sécurisée**.

2.2. Gestion des tâches

- **CRUD** :
    - Créer une tâche avec un titre, une description, une date/heure limite, et une catégorie.
    - Lire/visualiser les tâches dans une liste ou une vue calendrier.
    - Modifier une tâche existante.
    - Supprimer une tâche.
- **Organisation des tâches** :
    - Trier par catégorie, priorité, ou date limite.
    - Statut des tâches : "À faire", "En cours", "Terminé".
- **Recherche et filtre** : Par mots-clés, catégorie ou date.

2.3. Notifications et rappels

- **Rappels programmés** : Envoyer une notification avant une échéance via Firebase Cloud Messaging (FCM).
- **Notifications en cas de mise à jour** : Synchronisation en temps réel.

2.4. Synchronisation et stockage

- Sauvegarde des données des tâches dans **Firebase Firestore**.
- Synchronisation en temps réel sur plusieurs appareils.

---

3. Design de l'application

3.1. Écrans principaux

1. **Écran de bienvenue** : Introduction et lien vers la connexion/inscription.
2. **Écran d'inscription/connexion** : Champs email/mot de passe et options de connexion sociale.
3. **Écran principal** :
    - Liste des tâches (vue par défaut).
    - Barre de navigation pour accéder aux catégories.
4. **Écran des détails d'une tâche** : Afficher et modifier les informations d'une tâche.
5. **Écran de création/modification de tâche** : Formulaire pour ajouter ou modifier une tâche.
6. **Écran de paramètres utilisateur** : Mise à jour du profil, langue, thème (clair/sombre), etc.

3.2. UX/UI

- Design simple et intuitif avec Flutter Material Design.
- Thème clair et sombre.
- Animation pour transitions fluides entre les écrans.

---

4. Technologie et architecture

4.1. Firebase Services

- **Firebase Authentication** : Pour la gestion des utilisateurs.
- **Firebase Firestore** : Pour le stockage et la synchronisation des tâches.
- **Firebase Cloud Messaging (FCM)** : Pour les notifications.
- **Firebase Storage** : Pour les avatars ou fichiers associés aux tâches (optionnel).

4.2. Architecture Flutter

- Gestion de l'état : Utiliser **Provider**, **GetX**, ou **Riverpod** selon la préférence.
- Structurer le projet selon le modèle **MVVM** ou **Clean Architecture** pour faciliter la maintenance.

---

5. Étapes de développement

1. **Configuration du projet** :
    - Installer Flutter et Firebase dans le projet.
    - Configurer Firebase (Authentication, Firestore, et Cloud Messaging).
2. **Implémentation des écrans** :
    - Écrans de connexion/inscription.
    - Écran principal avec liste de tâches.
    - Écran de détail et modification de tâche.
3. **Connexion Firebase** :
    - Lier Firebase Authentication aux écrans de connexion/inscription.
    - Implémenter Firestore pour créer, lire, mettre à jour, et supprimer des tâches.
4. **Gestion des tâches** :
    - Ajouter des filtres, recherches, et tri.
    - Implémenter les notifications via FCM.
5. **Test et débogage** :
    - Tests unitaires des fonctionnalités clés.
    - Tests sur différents appareils pour s'assurer de la compatibilité.
6. **Publication** :
    - Préparer l'application pour Google Play Store et Apple App Store.

---

6. Outils et ressources

- **IDE** : Visual Studio Code ou Android Studio.
- **Packages Flutter** :
    - firebase_auth pour l'authentification.
    - cloud_firestore pour Firestore.
    - firebase_messaging pour les notifications.
    - intl pour la gestion des dates et heures.

- flutter_local_notifications pour les rappels locaux.