Projet Todolist

Flutter

Objectif

Votre projet consiste à concevoir et à développer une application mobile cross-platforms avec Flutter et le langage Dart en respectant le cahier des charges ci-après.

Spécification fonctionnelles

Fonctionnalités de base

L'application permettra à un utilisateur de créer et gérer une liste de tâches :

- ***CRUD*** : création / mise à jour / suppression d'une tâche,
- Affichage dynamique de la **liste des tâches** (écran *Master*),
- Affichage des **détails d'une tâche sélectionnée** depuis l'écran *Master* (écran *Details*),
- **Tri / filtre des tâches** :
    - par statut de réalisation,
    - par tag,
    - par niveau de priorité,
    - par date limite de réalisation.
- Mise en place d'une ***API REST* avec un *CMS Headless*** *(Directus ou Supabase)*,
- **Création de compte utilisateur** (*Sign Up*) + **authentification** (*Sign In*) auprès de l'*API REST*,
- Interactions de type **CRUD** auprès de l'*API REST*.

Fonctionnalités avancées

Fonctionnalités avancées à réaliser, par ordre de priorité, uniquement si les fonctionnalités de base sont en place :

1. Gestion du ***mode offline*** (détection de l'absence de réseau + base de données locale + synchronisation avec l'API) + gestion des permissions,
2. Création de **tâches intégrant un contenu graphique** (image capturée avec la caméra et/ou issue de la bibliothèque de médias) + gestion des permissions,
3. **Géolocalisation de tâches** (soit, des tâches à réaliser à un emplacement GPS déterminé, ex : domicile, bureau…).
4. **Association de tâche à un contact** (interaction avec le carnet de contact natif *iOS / Android*),
5. **Intégration d'une tâche au calendrier** (interaction avec le calendrier natif *iOS / Android*).

Vous pourrez adapter et compléter les fonctionnalités ci-dessus selon votre inspiration.

Exigences fonctionnelles et techniques

- **Structuration du projet en plusieurs fichiers** (1 widget = 1 fichier) et **dossiers** (organisés par *feature* ou par aspect technique),
- **Emploi de classes *Dart* pour la gestion des services métier**,
- **Navigation entre les différents écrans de l'application avec un système de routes** (*Sign Up, Sign In, Master / Details*),
- **Gestion d'état applicatif local** (*Stateless / Stateful*),
- **Gestion d'état applicatif global avec un store de données** (*Provider*),
- Stockage des **variables d'environnement dans un fichier *".env"***,
- **Absence de messages d'erreur ou d'avertissement** dans l'onglet *"problèmes"* de votre IDE, respect des conventions de nommage fixées par Dart et Flutter…

- **Développement de l'application en priorité à destination d'un terminal mobile de type smartphone, orienté en mode portrait, sur la plateforme *iOS* et/ou *Android*** (plateformes web et desktop non prioritaires).

Spécifications ergonomiques

UI / UX

Vous apporterez un **soin particulier aux aspects UI / UX** de l'application afin de proposer la meilleure expérience possible à l'utilisateur :

- simplicité,
- intuitivité,
- rapidité,
- respect des standards ergonomiques mobiles / tactiles.

Pour ce faire, **vous vous inspirerez des applications faisant référence dans le domaine** (ex: *Rappels, Microsoft To do, Todoist, Evernote, Any.do,…*).

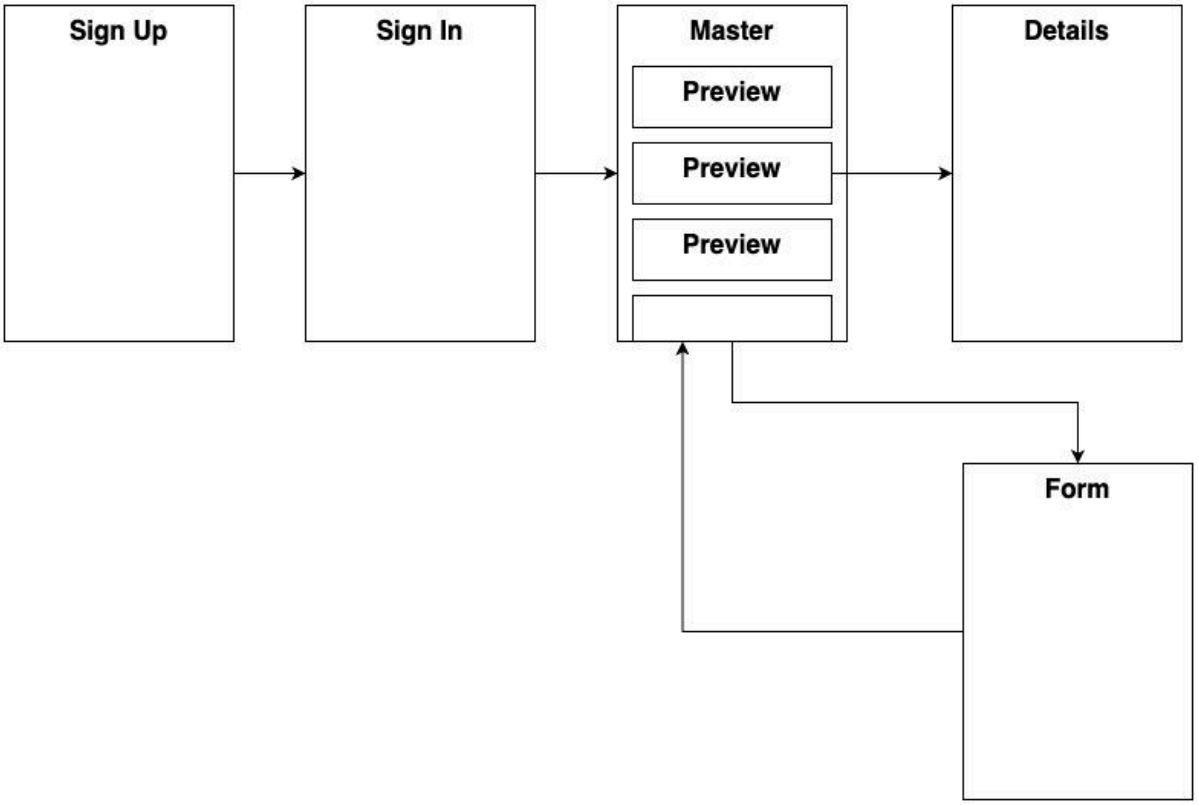
Vous personnaliserez l'interface graphique de l'application :

- Thème ***Material Design*** (ex : couleurs, bannière, mode sombre / light…),
- **Typographie**,
- **Icône** (*App Icon*)
- **Écran d'accueil** (*Splash Screen*),
- Affichage de **notifications graphiques** pour confirmer les actions réalisées,
- Affichage de **fenêtre modale** pour demander la confirmation d'une suppression,
- Emploi d'une **interaction de type *swipe*** (ex: pour supprimer ou modifier le statut d'une tâche),
- Distinction graphique entre les tâches réalisées / à réaliser,
- Distinction graphique des tâches selon leur niveau de priorité.

Navigation entre écrans

L'application comportera plusieurs écrans, accessibles via des *routes* dédiées :

- ***Sign In*** : formulaire de connexion,
- ***Sign Up*** : formulaire de création de compte,
- ***Master*** : liste de toutes les tâches,
- ***Details*** : affichage des détails d'une tâche sélectionnée depuis l'écran Master, sous forme de formulaire,
- ***Form*** : formulaire de création de tâche.



Spécifications techniques

Modèles de données

Utilisateur (User)

- identifiant,
- prénom,
- nom,
- email,
- mot de passe,
- liste de tâches dont il est l'auteur.

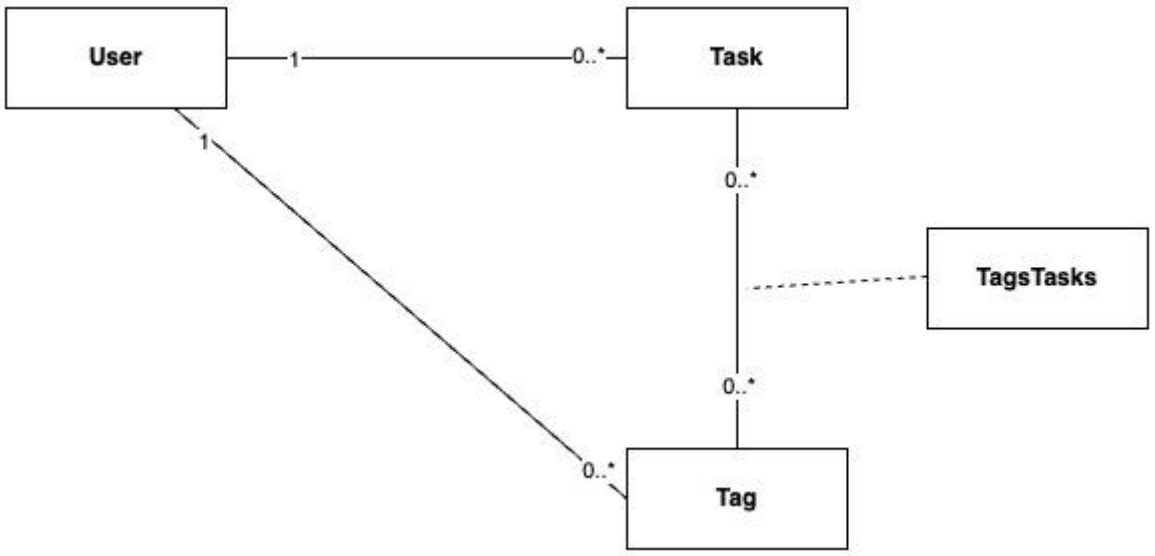
Tâche (Task)

- identifiant,
- auteur (utilisateur),
- contenu,
- tags,
- statut de réalisation (en cours / complétée),
- date de création,
- date de mise à jour,
- date limite de réalisation,
- niveau de priorité (normal, important, très important).

Tag

- nom,
- auteur,
- tâches.

Le modèle de données caractérisant une tâche sera amené à évoluer au gré des fonctionnalités ajoutées.
Vous programmerez les classes *Dart* nécessaires.


User
1
0..*
Task
1
0..*
TagsTasks
0..*
0..*
Tag

API REST

Vous mettrez en place une ***API REST*** locale ou distante à l'aide d'un ***CMS Headless*** tel que *Supabase* (en ligne) ou *Directus* (en local, avec *Docker*).

L'application interagira avec l'*API REST* afin d'authentifier l'utilisateur et gérer les données (fonctionnalités de type *CRUD*).

Authentification / Autorisation

L'utilisateur accèdera aux tâches dont il est l'auteur après authentification auprès de l'*API REST*.

A chaque requête *HTTP* à destination de l'*API REST*, **l'utilisateur devra communiquer un *access token (JWT)* afin de s'authentifier**.

En cas d'expiration, **l'utilisateur devra effectuer un renouvellement de son *access token* via un mécanisme de *refresh token*.**

Pour simplifier l'emploi de l'application, **l'utilisateur pourra choisir de rester connecté en conservant ses données de connexion dans les préférences de l'application** (espace du système de fichiers alloué à l'application sur le device mobile).

Mode Offline

Afin de permettre à l'utilisateur d'utiliser l'application en l'absence de connexion au réseau, vous enregistrerez les données initialement obtenues auprès de l'API dans une base de données locale (*SQLite ou NoSQL*).

Les modifications effectuées en local, en mode *offline*, devront être synchronisées auprès de l'*API REST* dès que l'accès au réseau sera à nouveau possible.

Packages recommandés

(Liste non-exhaustive).
Packages à installer progressivement selon nécessité.

- **Navigation** : https://pub.dev/packages/go_router
- **Store de données** : https://pub.dev/packages/provider
- **Client *HTTP*** : https://pub.dev/packages/dio
- **Variables d'environnement** : https://pub.dev/packages/flutter_dotenv
- **Personnalisation du *Splash Screen*** : https://pub.dev/packages/flutter_native_splash
- **Personnalisation de l'*App Icon*** : https://pub.dev/packages/flutter_launcher_icons
- **Préférences de l'application** : https://pub.dev/packages/shared_preferences
- **Chiffrage des données stockées dans les préférences de l'application :** https://pub.dev/packages/encrypt_shared_preferences
- **Base de données locale *SQLite*** : https://pub.dev/packages/sqflite
- **Base de données locale *NoSQL*** : https://pub.dev/packages/hive
- **Détection de la connexion au réseau** : https://pub.dev/packages/connectivity_plus
- **Caméra** : https://pub.dev/packages/camera
- **Carte géographique interactive** : https://pub.dev/packages/flutter_map

Modalités de réalisation

- **Travail individuel**,
- **Code personnel** (usage modéré de l'IA générative),
- Rendu sous forme de dépôt Git partagé à alex@shrp.dev (si dépôt privé, ajouter *shrp777* en tant que collaborateur),
- Renseigner l'identité de l'auteur de l'application dans un fichier README.md à la racine du projet,
- Délai de rendu : **vendredi 28/6/2024, 17h.**

Critères d'évaluation

- Etat fonctionnel de l'application,
- Respect des consignes,
- Qualité du code,
- Qualité *UI / UX*,
- Participation, assiduité et comportement respectueux pendant les séances de cours.