**Idée de Projet : Application de suivi d'habitudes minimaliste**

* **Concept :** Une application simple où l'utilisateur peut ajouter des habitudes (ex: boire 8 verres d'eau, faire 30 min d'exercice) et marquer s'il les a complétées chaque jour.
* **Objectifs de codage :**
  + Utiliser une **base de données légère** (comme SQLite) pour stocker les habitudes et l'historique de complétion.
  + Gérer l'**état de l'interface utilisateur** (affichage des habitudes, mise à jour des icônes).
  + Afficher une **série** (streak) de jours consécutifs réussis.
* **Framework Python recommandé :** **Kivy** ou **KivyMD** (pour un look Material Design plus moderne).

**🧑‍💻 Explication Détaillée des Rôles**

**🚀 Sprint 0 : L'environnement**

| **Rôle** | **Tâches** | **Explication et Objectif** |
| --- | --- | --- |
| **Dév. A (Backend)** | **Mise en place Kivy** | Objectif : Assurer que les outils de base pour exécuter le projet sont installés. Installer Python et le framework Kivy via la ligne de commande (pip install kivy). |
| **Dév. B (Frontend)** | **Initialisation du Dépôt** et **Fichier Kivy de base** | Objectif : Créer la maison du code. Mettre en place un dépôt Git partagé. Créer le fichier main.py avec la classe App Kivy de base (juste une fenêtre vide qui se lance) et le fichier de configuration .kv. |
| **A & B** | **Test de l'Environnement** | Objectif : Vérification mutuelle. S'assurer que les deux postes peuvent cloner, modifier, pousser et exécuter le code de base sans erreur. |

**🧱 Sprint 1 : La structure des données**

| **Rôle** | **Tâches** | **Explication et Objectif** |
| --- | --- | --- |
| **Dév. A (Backend)** | **Modèle de données SQLite** et **Logique CRUD (Données)** | Objectif : Créer la fondation des données. Écrire le code Python (probablement une classe) pour se connecter au fichier .db, créer les tables nécessaires, et fournir les fonctions pour : **1.** Ajouter une habitude, **2.** Récupérer la liste des habitudes, et **3.** Marquer une habitude comme faite pour la date du jour. *Ce code n'a pas d'interface.* |
| **Dév. B (Frontend)** | **Structure de l'Interface (KV)** et **$(\star)$ Connexion Vue/Modèle** | Objectif : Préparer les emplacements visuels. Définir le layout principal de l'application dans le fichier .kv (ex : un BoxLayout ou GridLayout). Mettre un bouton "Ajouter" et l'espace pour la liste. L'étape $(\star)$ est cruciale : il faut intégrer la classe de gestion de données de Dév. A dans la classe principale Kivy pour que le Frontend puisse appeler les fonctions du Backend (ex : *self.db.get\_all\_habits()*). |

**🖼️ Sprint 2 : L'interface utilisateur**

| **Rôle** | **Tâches** | **Explication et Objectif** |
| --- | --- | --- |
| **Dév. A (Backend)** | **Widget Personnalisé (HabitItem)**, **Liste dynamique (RecycleView)**, et **Rendu Visuel de l'État** | Objectif : Afficher les données. Créer le petit bloc Kivy (le *widget*) qui représente une seule habitude (nom + état). Utiliser ensuite un RecycleView (optimisé pour les longues listes) pour charger et afficher dynamiquement tous ces widgets en utilisant les données fournies par ses fonctions de Backend. Enfin, il met à jour visuellement le widget après le clic. |
| **Dév. B (Frontend)** | **Fenêtre d'Ajout d'Habitude** et **$(\star)$ Gestion du Clic** | Objectif : Gérer l'entrée utilisateur. Construire la fenêtre Pop-up/Modale pour que l'utilisateur puisse saisir le nom d'une nouvelle habitude. S'assurer que le bouton "Ajouter" appelle la fonction add\_habit du Backend. L'étape $(\star)$ est l'interaction clé : quand l'utilisateur touche un HabitItem (créé par Dév. A), Dév. B intercepte ce clic et appelle la fonction mark\_completed\_today du Backend. |

**📊 Sprint 3 : Le "Streak" et finition**

| **Rôle** | **Tâches** | **Explication et Objectif** |
| --- | --- | --- |
| **Dév. A (Backend)** | **Navigation (ScreenManager)** et **Écran des Paramètres/Historique** | Objectif : Ajouter la complexité de l'application. Mettre en place la navigation entre plusieurs écrans (ex : Écran d'accueil et Écran des paramètres) en utilisant le ScreenManager de Kivy. Créer la structure et les éléments de base du nouvel écran pour pouvoir, par exemple, supprimer les habitudes. |
| **Dév. B (Frontend)** | **Calcul du "Streak"** et **Affichage du "Streak"** | Objectif : Implémenter la logique métier complexe. Écrire le code Python pour déterminer le nombre de jours consécutifs où l'habitude a été complétée. C'est une logique purement métier (Backend), mais attribuée ici à Dév. B pour équilibrer la charge. Ensuite, s'assurer que ce nombre est transmis et affiché correctement par le widget d'habitude (créé par Dév. A). |
| **A & B** | **Test et Polissage final** | Objectif : Valider le produit. Tester toutes les fonctionnalités (ajout, complétion, streak, navigation) pour s'assurer qu'il n'y a pas de bugs. Examiner le code de l'autre pour une revue croisée. |

**Ce découpage est clair et permet de bien séparer les préoccupations :**

* **Dév. A** fournit les **données** et la **capacité à les modifier**.
* **Dév. B** fournit les **boutons**, les **écrans** et les **interactions** nécessaires.