**Phase 1 : Fondations (Semaines 1-2)**

Objectif : Mettre en place la base technique. Focus sur l'architecture simple (StatefulWidget, modèles de données, stockage).

* **Ce que tu dois apprendre** :
  + Widgets de base (Scaffold, AppBar, Text, Button).
  + Gestion d'état simple (StatefulWidget vs StatelessWidget).
  + Modèles de données (classes Dart comme Note {id, title, content, createdAt, updatedAt}).
  + Persistance basique avec SharedPreferences (pour sauvegarder des listes JSON).
  + Structure des dossiers (comme décrite : lib/main.dart, models/note.dart, services/storage\_service.dart, etc.).
* **Tâches clés** (basées sur le doc) :
  + Crée le projet Flutter : flutter create babana\_note.
  + Définit les modèles (Note, Category).
  + Implémente StorageService avec saveNotes/loadNotes (copie le code exemple du doc et adapte).
  + Teste le stockage : Écris un petit script pour sauvegarder/charger une note.
* **Ressources** :
  + Widgets : Codelab "Building a Cupertino app with Flutter".
  + État : Tutoriel "State management" sur flutter.dev (commence par approche simple).
  + SharedPreferences : Cookbook Flutter "Persist data with SharedPreferences".
  + Livrable : À la fin, tu auras la structure du projet et les modèles.

**Phase 2 : Fonctionnalités de Base (Semaines 3-4)**

Objectif : App fonctionnelle minimale (MVP). Focus sur UI basique et CRUD (Create, Read, Update, Delete).

* **Ce que tu dois apprendre** :
  + Navigation entre écrans (Navigator.push pour passer d'un écran à l'autre).
  + Listes dynamiques (ListView pour afficher les notes).
  + Formulaires simples (TextField pour titre/contenu).
  + CRUD avec listes Dart + SharedPreferences.
  + Sauvegarde auto (utilise Timer pour save toutes les 5s).
* **Tâches clés** :
  + Écran d'accueil (home\_screen.dart) : AppBar, liste de notes (cartes avec aperçu), bouton flottant pour ajouter.
  + Éditeur de notes (note\_editor\_screen.dart) : Champs titre/contenu, boutons save/supprimer.
  + Implémente CRUD : Ajouter/modifier/supprimer notes, visualiser liste.
  + Ajoute navigation et auto-save.
* **Ressources** :
  + Navigation : Cookbook "Navigate to a new screen and back".
  + ListView : Codelab "Implicit animations" (pour listes basiques).
  + Formulaires : Tutoriel "Add a user interface for creating meals" (adaptable).
  + Livrable : MVP – App qui gère des notes basiques.

**Phase 3 : Fonctionnalités Avancées (Semaines 5-6)**

Objectif : Améliorer l'app avec organisation et personnalisation.

* **Ce que tu dois apprendre** :
  + Recherche et tri (filtrer listes avec .where, trier avec .sort).
  + Widgets avancés (Dropdown pour catégories, ColorPicker pour couleurs).
  + Épinglage (ajoute un bool isPinned dans Note).
  + Gestion de catégories (liste de catégories, assigner à notes).
* **Tâches clés** :
  + Ajoute recherche dans l'écran d'accueil.
  + Implémente tri (par date/titre).
  + Personnalisation : Couleur de note (utilise flutter\_colorpicker package), épinglage.
  + Système de catégories : Menu déroulant dans éditeur.
  + Installe packages : flutter pub add shared\_preferences intl flutter\_colorpicker.
* **Ressources** :
  + Recherche/Tri : Cookbook "Filter a list".
  + ColorPicker : Doc du package sur pub.dev.
  + Packages : Guide "Using packages" sur flutter.dev.
  + Livrable : Version Beta – Toutes fonctionnalités principales.

**Phase 4 : Finitions (Semaines 7-8)**

Objectif : Polir l'app pour qu'elle soit pro.

* **Ce que tu dois apprendre** :
  + Thèmes (ThemeData pour mode sombre/clair).
  + Animations (AnimatedContainer, Hero pour transitions).
  + Tests (unitaires avec flutter\_test pour modèles/services).
  + Optimisation (vérifie perf avec >1000 notes).
  + Documentation (ajoute commentaires dans le code).
* **Tâches clés** :
  + Ajoute mode sombre/clair (switch dans params).
  + Animations sur listes/transitions.
  + Tests : Couvre 80% du code (modèles, storage).
  + Optimise : Temps réponse <1s, APK <50MB.
  + Documente : Guide d'utilisation + comments.
* **Ressources** :
  + Thèmes : Codelab "Add a theme switcher".
  + Animations : Cookbook "Animate a widget".
  + Tests : Guide "Unit testing" sur flutter.dev.
  + Livrable : Version Finale + Documentation.

**Après le projet :**

* Teste sur vrai device (Android d'abord, comme spécifié).
* Respecte les critères d'acceptation (section 9) : Fonctionne offline, UI intuitive, perf ok.
* Si bloqué : Debug avec VS Code (breakpoints), ou cherche erreurs spécifiques en ligne.
* Évolutions futures : Migre vers une BD comme Hive si besoin (mais pas pour ce projet simple).

Ce plan suit le cahier des charges à la lettre, en progressant du simple au complexe pour que tu apprennes sans te décourager. Commence petit, code tous les jours, et n'hésite pas à me demander des détails sur une phase spécifique (ex. : code exemple pour StorageService). Bonne chance, tu vas y arriver – comme dit Churchill dans ton doc ! 🚀

une page web