BABANA NOTE

Application Mobile de Prise de Notes

CAHIER DES CHARGES

Projet d'apprentissage Flutter

Version	1.0
Date	19 septembre 2025
Statut	Projet d'apprentissage
Plateforme	Android & iOS
Framework	Flutter

Développé avec Flutter & Dart

Table des matières

1 Introduction et Contexte

1.1 Présentation du projet

Babana Note est une application mobile de prise de notes développée dans le cadre d'un projet d'apprentissage de Flutter. Cette application vise à offrir une expérience utilisateur simple et intuitive pour la création, l'organisation et la gestion de notes personnelles.

1.2 Contexte d'apprentissage

Ce projet s'inscrit dans une démarche d'apprentissage de la programmation mobile avec Flutter. Il permettra de maîtriser :

— Les concepts fondamentaux de Flutter et Dart
— La gestion d'état simple avec StatefulWidget
— La persistance de données avec des structures simples
— Le design d'interfaces utilisateur modernes
— Les bonnes pratiques de développement mobile

1.3 Public cible

— **Utilisateur principal** : Développeur en apprentissage
— **Niveau technique** : Débutant à intermédiaire

2 Objectifs du Projet

2.1 Objectifs pédagogiques

1. Maîtriser les widgets Flutter essentiels
2. Comprendre la gestion d'état simple avec StatefulWidget
3. Implémenter la persistance de données avec des structures simples
4. Créer une interface utilisateur responsive
5. Gérer les états de l'application de manière basique
6. Implémenter la navigation entre écrans

2.2 Objectifs fonctionnels

1. Créer une application fonctionnelle de prise de notes
2. Offrir une expérience utilisateur fluide
3. Assurer la sauvegarde automatique des données
4. Permettre l'organisation des notes par catégories
5. Implémenter une fonction de recherche

2.3 Objectifs techniques

1. Code propre et bien documenté
2. Architecture modulaire et maintenable
3. Performance optimisée
4. Compatibilité Android
5. Tests unitaires et d'intégration

3 Analyse des Besoins

3.1 Besoins fonctionnels

3.1.1 Gestion des notes

— **BF001** : Créer une nouvelle note
— **BF002** : Modifier une note existante
— **BF003** : Supprimer une note
— **BF004** : Visualiser la liste des notes
— **BF005** : Rechercher dans les notes

3.1.2 Organisation

— **BF006** : Trier les notes (date, titre)

3.1.3 Personnalisation

— **BF007** : Changer la couleur d'une note
— **BF008** : Épingler des notes importantes
— **BF009** : Mode sombre/clair

3.2 Besoins non fonctionnels

3.2.1 Utilisabilité

— Interface intuitive et moderne
— Navigation simple et logique
— Feedback visuel pour toutes les actions
— Accessibilité pour les utilisateurs malvoyants

3.2.2 Fiabilité

— Sauvegarde automatique des données
— Récupération en cas de fermeture inattendue
— Gestion des erreurs gracieuse

4 Spécifications Fonctionnelles

4.1 Écran d'accueil

Fonction	Description
Liste des notes	Affichage de toutes les notes sous forme de cartes avec aperçu du contenu
Bouton d'ajout	Bouton flottant pour créer une nouvelle note
Barre de recherche	Champ de recherche en haut de l'écran
Menu de navigation	Accès aux catégories et paramètres
Tri et filtres	Options de tri par date, titre, catégorie

4.2 Éditeur de notes

Fonction	Description
Champ titre	Zone de saisie pour le titre de la note
Éditeur de texte	Zone de saisie multiligne pour le contenu
Sélecteur de couleur	Palette pour choisir la couleur de la note
Sélecteur de catégorie	Menu déroulant pour assigner une catégorie
Sauvegarde auto	Sauvegarde automatique toutes les 5 secondes
Boutons d'action	Sauvegarder, partager, supprimer

5 Spécifications Techniques

5.1 Architecture de l'application

5.1.1 Pattern architectural

L'application utilisera une architecture simple basée sur :

- **StatefulWidget** pour la gestion d'état locale
- **Structures de données** (List, Map) pour stocker les notes
- **SharedPreferences** pour la persistance simple
- **Séparation claire** entre widgets et logique métier

5.1.2 Structure des dossiers

```
lib/
 main.dart
 models/
    note.dar
```

```
screens/
   home_screen.dart
   note_editor_screen.dart
widgets/
   note_card.dart
   color_picker.dart
services/
   storage_service.dart
utils/
    constants.dart
    helpers.dart
```

5.2 Technologies utilisées

5.2.1 Framework et langages

- **Flutter** : Framework de développement mobile
- **Dart** : Langage de programmation
- **Version Flutter** : 3.16+ (stable)
- **Version Dart** : 3.2+

5.2.2 Packages principaux

Package	Version	Usage
shared_preferences	^2.2.0	Stockage simple des données
intl	^0.18.0	Formatage des dates
flutter_colorpicker	^1.0.3	Sélecteur de couleurs

5.3 Stockage des données

5.3.1 Structures de données

$$\text{Note} = \{id, title, content, createdAt, updatedAt\}$$

5.3.2 Implémentation du stockage

```
class StorageService {
  static const String _notesKey = 'notes';
  static const String _categoriesKey = 'categories';

  // Sauvegarde des notes dans SharedPreferences
  Future<void> saveNotes(List<Note> notes) async {
    final prefs = await SharedPreferences.getInstance();
    final notesJson = notes.map((note) => note.toJson()).toList();
    await prefs.setString(_notesKey, jsonEncode(notesJson));
  }
```

```
  // Chargement des notes depuis SharedPreferences
  Future<List<Note>> loadNotes() async {
    final prefs = await SharedPreferences.getInstance();
    final notesString = prefs.getString(_notesKey);
    if (notesString == null) return [];

    final notesList = jsonDecode(notesString) as List;
    return notesList.map((json) => Note.fromJson(json)).toList();
  }
}
```

6 Interface Utilisateur

6.1 Design System

6.1.1 Palette de couleurs

- **Primaire** : #2196F3 (Bleu Material)
- **Secondaire** : #FF9800 (Orange)
- **Surface** : #FFFFFF (Blanc)
- **Background** : #F5F5F5 (Gris clair)
- **Error** : #F44336 (Rouge)

6.1.2 Typographie

- **Police principale** : Roboto
- **Titre principal** : 24sp, Bold
- **Titre secondaire** : 20sp, Medium
- **Corps de texte** : 16sp, Regular
- **Caption** : 12sp, Regular

6.1.3 Espacement

- **Marge externe** : 16dp
- **Marge interne** : 8dp
- **Espacement entre éléments** : 12dp
- **Rayon de bordure** : 8dp

6.2 Wireframes des écrans principaux

6.2.1 Écran d'accueil

- AppBar avec titre "ZeBabana Note" et icône de recherche
- Barre de recherche rétractable
- Liste de cartes représentant les notes
- Bouton d'action flottant pour ajouter une note
- Drawer pour navigation vers les catégories

6.2.2 Éditeur de note

- AppBar avec boutons retour et sauvegarde
- Champ de saisie pour le titre
- Zone de texte multiligne pour le contenu
- Barre d'outils avec options de formatage
- Sélecteurs de couleur et catégorie en bas

7 Planning et Livrables

7.1 Phases de développement

7.1.1 Phase 1 : Fondations (Semaines 1-2)

- Configuration de l'environnement Flutter
- Création de la structure du projet
- Mise en place de l'architecture simple avec StatefulWidget
- Création des modèles de données (Note, Category)
- Configuration du stockage avec SharedPreferences

7.1.2 Phase 2 : Fonctionnalités de base (Semaines 3-4)

- Écran d'accueil avec liste des notes
- Éditeur de notes basique
- CRUD des notes avec structures de données simples
- Sauvegarde automatique avec SharedPreferences
- Navigation entre écrans

7.1.3 Phase 3 : Fonctionnalités avancées (Semaines 5-6)

- Système de catégories
- Fonction de recherche
- Personnalisation des couleurs
- Tri et filtrage des notes
- Épinglage des notes importantes

7.1.4 Phase 4 : Finitions (Semaines 7-8)

- Thème sombre/clair
- Animations et transitions
- Tests unitaires et d'intégration
- Optimisation des performances
- Documentation du code

7.2 Livrables

Livrable	Date	Description
Architecture	Semaine 2	Structure du projet et modèles de données
MVP	Semaine 4	Version minimale fonctionnelle
Version Beta	Semaine 6	Toutes les fonctionnalités implémentées
Version Finale	Semaine 8	Application complète et testée
Documentation	Semaine 8	Guide d'utilisation et documentation technique

8 Contraintes et Risques

8.1 Contraintes techniques

— **Compatibilité** : Android 5.0+ et iOS 11.0+
— **Taille** : Application $<$ 50 MB
— **Performance** : Temps de réponse $<$ 1 seconde
— **Hors ligne** : Fonctionnement sans connexion internet

8.2 Contraintes de développement

— **Temps** : 8 semaines de développement
— **Ressources** : Développeur unique en apprentissage
— **Budget** : Projet personnel sans budget
— **Outils** : Utilisation d'outils gratuits uniquement

8.3 Risques identifiés

Risque	Probabilité	Impact	Mitigation
Courbe d'apprentissage Flutter	Élevée	Moyen	Formation progressive, documentation
Gestion des données complexes	Faible	Moyen	Structures simples, tests réguliers
Problèmes de performance	Faible	Élevé	Tests réguliers, optimisation continue
Limitation du stockage simple	Moyenne	Faible	Migration future vers base de données

9 Critères d'Acceptation

9.1 Critères fonctionnels

1. L'utilisateur peut créer, modifier et supprimer des notes
2. L'utilisateur peut organiser ses notes par catégories

3. L'utilisateur peut rechercher dans ses notes
4. L'application sauvegarde automatiquement les modifications
5. L'interface est intuitive et responsive

9.2 Critères techniques

1. L'application fonctionne sur Android et iOS
2. Le code respecte les bonnes pratiques Flutter
3. L'architecture est modulaire et maintenable
4. Les tests couvrent au moins 80% du code
5. La documentation est complète et à jour

9.3 Critères de performance

1. Temps de démarrage $<$ 3 secondes
2. Temps de réponse $<$ 1 seconde pour les actions courantes
3. Gestion fluide de 1000+ notes
4. Consommation mémoire $<$ 100 MB
5. Taille de l'APK $<$ 50 MB

10 Conclusion

Ce cahier des charges définit les spécifications complètes pour le développement de l'application mobile **Babana Note**. Ce projet d'apprentissage permettra de maîtriser les concepts fondamentaux de Flutter tout en créant une application utile et fonctionnelle.

L'approche progressive du développement, organisée en 4 phases distinctes, permettra un apprentissage structuré et l'acquisition de compétences solides en développement mobile.

Le respect de ce cahier des charges garantira la livraison d'une application de qualité, bien architecturée et maintenable, constituant une excellente base pour des projets Flutter plus complexes.

« Le succès, c'est d'aller d'échec en échec sans perdre son enthousiasme. »
Winston Churchill