Documentation technique

1. Architecture de l'application

L'application "Gestion de Dépenses" a été développée avec Ionic et React.

Elle respecte le modèle "SPA" (Single Page Application) :

Navigation gérée par Ionic React Router

Interface basée sur les composants UI d'Ionic, adaptés pour le mobile

Les données (les dépenses) sont stockées localement via le navigateur (localStorage), ce qui rend l'application utilisable hors-ligne, sans backend.

2. Découpage en composants

L'architecture est modulaire et clairement découpée :

a. Pages

- ExpensesList

Page d'accueil affichant : la liste des dépenses, un graphe de répartition par catégorie, et le total cumulé.

- AddExpense

Page pour ajouter une nouvelle dépense.

b. Composants

- Header

En-tête réutilisable avec titre dynamique et bouton retour optionnel.

- Footer

Pied de page réutilisable, affiche le total des dépenses.

- ExpenseCard

Affiche une dépense individuelle (catégorie, montant, description, date, suppression).

- ExpensesPieChart

Graphe camembert affichant la répartition des dépenses par catégorie (utilise recharts).

- AddExpenseForm

Formulaire d’ajout de dépense, avec validation et enregistrement.

c. Modèle

- Expense

Interface TypeScript qui définit une dépense :

id, amount, category, description, date.

3. Services

- expenseService.ts

Service utilitaire qui gère l’accès aux données (CRUD) dans le localStorage :

- getExpenses() : récupère toutes les dépenses
- addExpense(expense) : ajoute une nouvelle dépense
- deleteExpense(id) : supprime une dépense par son id

4. Processus de packaging Android

Le packaging pour Android se fait en 5 étapes principales :

- Build de l’application

ionic build

Génère le code de production.

- Ajout de la plateforme Android

npx ionic cap add android

- Copie des fichiers de build

npx ionic cap copy android

- Ouverture dans Android Studio

npx ionic cap open android

L’application est prête à être compilée ou testée sur un émulateur/appareil réel.

5. Résumé technique

- React et Ionic pour l’UI et la navigation mobile
- Recharts pour l’affichage des graphiques
- TypeScript pour la robustesse du code

- localStorage pour la persistance des données (pas de backend)
- Packaging avec Capacitor pour générer un APK Android