SpendIQ

Proyecto Final Gestor Gastos

Plan 11 Semanas

Requerimientos del proyecto

- **Tipo de proyecto: Web App – Gestor de Gastos / Dashboard personal**
- **Tiempo disponible: 11 semanas**
- **Requisitos iniciales:**

  Definir un tema, aplicar identidad corporativa (nombre, logotipo, paleta de colores), incluir autenticación simulada, análisis de competencia con reseñas e interfaces principales.

- **Herramientas a utilizar : React + Vite, CSS Modules**

---

1) Visión y alcance (MVP)

Objetivo: Desarrollar una SPA de gestión de gastos con login simulado, CRUD de gastos y categorías, dashboard de resumen y exportación de datos.

MVP

- **Autenticación simulada (usuario/password falso con persistencia local).**
- **Listado de gastos con búsqueda, filtros por categoría y CRUD completo.**
- **Dashboard con totales mensuales, top categorías y presupuesto.**
- **Categorías personalizables con color.**
- **Persistencia local (LocalStorage).**
- **Formularios con validación (React Hook Form o validación propia).**

Extras (opcional): tema oscuro, exportar CSV/JSON, importación, estadísticas con Chart.js.

---

2) Arquitectura de contenido

Páginas clave:

- **Login: acceso simulado, redirección a Dashboard.**
- **Dashboard: resumen de gastos por mes/categoría**
- **Gastos: listado CRUD + filtros y búsqueda.**
- **Categorías: gestión CRUD de categorías.**

Definir Rutas:

- **/login**
- **/dashboard**
- **/expenses**
- **/categories**

Wireframes sugeridos:

1. **Login → formulario minimalista con feedback de error.**
2. **Dashboard → tarjetas de resumen + gráfico circular (opcional).**
3. **Gastos → tabla con filtros, botones de editar/eliminar.**
4. **Categorías → lista de chips con color y botón de añadir.**

---

3) Backlog funcional

P0 – Imprescindible: rutas, layout, listado CRUD, validación, estado global.
P1 – Deseable: filtros avanzados, presupuesto mensual, toasts, export CSV.
P2 – Opcional: tema oscuro, estadísticas con Chart.js, PWA offline.

---

4) Stack técnico

- **Bundler: Vite**
- **Lenguaje: JavaScript (JSX)**
- **Ruteo: React Router v6**
- **Estado global: Zustand (o Redux Toolkit)**
- **Estilos: Tailwind o CSS Modules**
- **Validación: React Hook Form (si permitido)**
- **Persistencia: LocalStorage**
- **Testing: Vitest + React Testing Library**

---

5) Estructura de carpetas:

DEFINIR ARBOLD E FICHEROS

---

6) Manejo de estado y datos

- **Local: useState, useEffect para formularios o componentes simples.**
- **Global: useExpenseStore, useCategoryStore, useAuthStore con Zustand.**
- **Persistencia: sincronización automática con LocalStorage.**
- **Estructura de datos:**
- **Expense = { id, date, category, concept, amount }**
- **Category = { id, name, color }**
- **User = { email, theme }**

---

7) Calidad, testing y DX

ESLint/Prettier desde el inicio.

- **Tests mínimos:**
  - **3–5 tests de componentes (render + interacción).**
  - **1–2 tests de hooks (Zustand).**
  - **1 flujo E2E (login → crear gasto → dashboard).**
- **Commits convencionales: feat:, fix:, test:, docs:.**
- **Git con ramas main + feat/* + PRs pequeños.**

PRESENTACION