PROGRAMA EDUCATIVO EN:

Tecnologías de la Información en Desarrollo y Gestión de Software

Desarrollo Web Profesional

NOMBRE DEL ESTUDIANTE:

Dana Lizbeth Castañeda Sánchez

Arquitectura FrontEnd

8vo Cuatrimestre “A”

Arquitectura General del Frontend

La arquitectura se organiza en tres pilares fundamentales para garantizar un sistema escalable, accesible y de alto rendimiento.

React: El Núcleo del Sistema

Single Page Application (SPA): Se utiliza React con Vite para ofrecer una navegación fluida sin recargas de página, optimizando el tiempo del usuario.

Enrutamiento Dinámico: App.jsx actúa como el orquestador principal, definiendo rutas para la Home, el Buscador y el Login.

PWA (Progressive Web App): La arquitectura permite el acceso desde cualquier navegador y soporta cacheo offline para funcionar con conexiones intermitentes.

Componentes: Modularidad y Reutilización

La interfaz sigue una jerarquía de componentes que facilita el mantenimiento por parte del equipo:

Componentes Globales: Elementos como el Navbar son persistentes, permitiendo navegación constante.

Vistas Independientes: Cada sección del sitemap (Buscador, Directorio, Detalle de Tienda) se encapsula en su propio componente de vista.

Estilizado con Tailwind CSS: Se emplea un sistema de utilidades CSS para garantizar que el diseño sea responsivo y coherente en móviles y desktop.

Gestión de Estado: Interactividad y Datos

Estado Local (useState): Gestiona la lógica inmediata, como el filtrado de productos por precio ascendente y la captura de credenciales.

Sincronización con API: El frontend consume datos JSON del backend (Node.js) para mostrar precios y ubicaciones en tiempo real.

Estado de Autenticación: Almacena de forma segura el token JWT para permitir que las tiendas gestionen sus inventarios.

Ejemplos de Aplicaciones Web Profesionales

El diseño técnico de RAPITI adopta patrones de arquitectura utilizados por líderes de la industria para resolver problemas de escala y usabilidad:

Gmail (Persistencia de Navegación): Al igual que Gmail, RAPITI utiliza una estructura donde el marco de navegación se mantiene fijo mientras el contenido cambia dinámicamente, permitiendo al usuario buscar productos sin perder el contexto.

Trello (Componentes de Tarjeta): La visualización del directorio de tiendas y el comparador de precios utiliza una arquitectura de "tarjetas" (cards) para organizar información densa de forma legible, similar al tablero de Trello.

Figma (Retroalimentación de Estado): Se implementa un sistema de estados visuales claros. Por ejemplo, al navegar con la tecla TAB, los elementos muestran un anillo azul (ring-4 ring-blue-500), asegurando la accesibilidad profesional que caracteriza a herramientas de diseño complejas.

Flujo de Navegación Técnica

El flujo de navegación está diseñado para ser directo y reducir la fricción del comprador:

1. Búsqueda Pública: Acceso al comparador sin necesidad de registro.
2. Geolocalización: Visualización de la mejor opción en un mapa interactivo (Leaflet/OpenStreetMap).
3. Conversión (Drive-to-Store): El flujo finaliza dirigiendo al usuario físicamente a la tienda local.

Estructura de Carpetas del Frontend

Esta organización permite que un equipo trabaje en paralelo sin conflictos, separando la lógica de la interfaz.

```
rapiti-frontend/
├── public/           # Activos estáticos (iconos PWA, manifiesto)
├── src/
│   ├── api/          # Configuración de Axios/Fetch para endpoints
│   ├── assets/       # Imágenes y estilos globales (Tailwind v4)
│   ├── components/   # Componentes compartidos (Botones, Inputs, Navbar)
│   ├── features/     # Lógica por funcionalidad del negocio
│   │   ├── search/   # Buscador y comparador de precios
│   │   ├── map/      # Integración con Leaflet y OpenStreetMap
│   │   └── auth/     # Login y manejo de JWT para tiendas
│   ├── hooks/        # Lógica reutilizable (useGeolocation, useLocalStorage)
│   ├── layouts/      # Plantillas de página (MainLayout, AuthLayout)
```

```
│   └──── views/          # Páginas principales definidas en el sitemap
│       ├──── Home.jsx     # Pantalla de inicio
│       ├──── Login.jsx    # Formulario de acceso estilizado
│       └──── Buscador.jsx # Interfaz de resultados y filtros
```

Errores que Resuelve el Frontend

El frontend de RAPITI no es solo una cara bonita; está diseñado específicamente para mitigar los "puntos de dolor" identificados en Tehuacán:

Dispersión de Información (Centralización)

- Solución: El frontend agrupa en una sola vista (Buscador) los inventarios de abarrotes, farmacias y papelerías.
- Impacto: El usuario ya no necesita consultar grupos de WhatsApp desactualizados o llamar por teléfono a cada tienda.

Invisibilidad del Comercio Local (Digitalización)

- Solución: Proporciona a las tiendas de barrio una presencia digital profesional sin que necesiten conocimientos técnicos.
- Impacto: Permite que el abarrote "Don Juan" compita en "conveniencia percibida" contra cadenas como Oxxo al mostrar que tiene mejores precios.

Fricción de Adopción (Accesibilidad)

- Solución: Al ser una PWA, resuelve el error de obligar al usuario a descargar una app pesada para una consulta rápida.
- Impacto: Un comprador puede comparar el precio de la leche en menos de 2 minutos mientras camina por el Centro o San Lorenzo.

Errores Técnicos que Previene el Frontend

Gracias a la arquitectura de React y Tailwind, evitamos fallos comunes en aplicaciones web:

- Navegación Ciega: Mediante los anillos de foco azul (ring-4), evitamos que usuarios que navegan por teclado se pierdan en la interfaz.
- Datos Corruptos: El frontend valida que los precios ingresados por las tiendas sean números positivos antes de enviarlos al backend.
- Pérdida de Sesión: Gestiona el token JWT para que la tienda no tenga que loguearse repetidamente, pero protege la información sensible si el token expira.
- Inconsistencia Visual: El uso de Tailwind elimina el error de tener diferentes estilos de botones o fuentes en distintas páginas del sitemap.