Notas de Diseño y Desafíos

Elecciones de diseño

La elección de tecnologías y arquitectura se realizó para prototipar la pagina de detalle de productos buscando una experiencia de usuario fluida.

Arquitectura General

- Opté por una arquitectura limpia, separando el frontend y el backend, para reflejar una aplicación real.
- En un monorepo, tengo las carpetas `api/` y `webapp/`, cada una con su propia estructura organizada.
- El backend se desarrolló en **Node + Typescript + Express**, un framework ligero y muy rápido para crear APIs RESTful.
- El frontend se creó con **React + TypeScript + Vite**, un framework popular entre el mundo del desarrollo web.

Backend

- Organización en:
- `app/src/routes/` — rutas de endpoints
- `app/src/controllers/` — validaciones y logica
- `app/src/models/` — definición de schema de datos
- `app/src/data/` — persistencia de datos en **archivos JSON**
- `app/src/tests/` — pruebas unitarias
- `app/tests/` — pruebas unitarias
- `app/docs/` — colección de postman

Frontend

- Estilo visualmente inspirado en **Mercado Libre** (colores, espaciado, diseño de columnas).
- El prototipo es **responsive**.

```
src/
├── pages/              # Paginas principales
├── components/         # Componentes reutilizables
├── assets/             # Imagenes y fuentes locales
├── services/           # Solicitud de datos a la API
├── types/              # Definiciones de tipos TypeScript
└── tests/              # Pruebas unitarias
```

Principales desafios enfrentados

1) Replicar la experiencia visual de MercadoLibre:

Uno de los mayores desafíos fue diseñar una página de detalle de producto que imitara la estética y estructura de MercadoLibre. El armado de la galeria del producto, con sus respectivos atributos a su derecha, fue un desarrollo de prueba y error. Se utilizo una pagina real de MeLi y asistencia de IA para su logro.

2) Limitaciones de tiempo y priorización de funcionalidades:

Desafío: Como es común en desafíos de prototipado, el tiempo es limitado. El desafío radica en entregar una solución funcional y bien diseñada dentro de un plazo estricto. Abordaje: Se adoptó un enfoque de desarrollo iterativo, priorizando las funcionalidades core (búsqueda y visualización de detalles). Se pospusieron características secundarias o mejoras de rendimiento avanzadas para futuras iteraciones, asegurando que la solución principal fuera sólida y estuviera completa.

3) Comprender y adaptar el modelo de datos al entorno de TypeScript:

El modelo de datos del producto incluía múltiples atributos como imágenes, precios, métodos de pago, vendedor, entre otros. Interpretar este modelo, normalizarlo y luego definir tipos estrictos en TypeScript fue clave para evitar errores en tiempo de desarrollo y mantener una estructura coherente entre el backend y el frontend.

4) Integración de la API y consumo de datos asíncronos:

Desafío: Conectar la aplicación frontend con la API de backend de manera eficiente y manejar las operaciones asíncronas de obtención de datos, incluyendo estados de carga, errores y datos vacíos. Además, gestionar posibles problemas de CORS (Cross-Origin Resource Sharing) durante el desarrollo. Abordaje: Se implementó un cliente HTTP robusto en el frontend para realizar las solicitudes a la API. Se manejaron los estados de la

interfaz de usuario (loading, error, data) para proporcionar retroalimentación visual al usuario. Para CORS, se configuró el backend para aceptar solicitudes del dominio del frontend durante el desarrollo, y se consideraría un proxy o configuración de producción adecuada para el despliegue.

Diagrama de flujo