Desafíos y Soluciones del Proyecto

Introducción

El desafío consistió en desarrollar un prototipo inspirado en MercadoLibre, enfocado en el detalle de artículos. El proyecto se dividió en dos partes: frontend y backend.

Para comenzar, analicé los requerimientos básicos con el fin de delimitar el alcance del desarrollo. A partir de esta revisión, definí el dominio de la aplicación, centrado en un único objeto llamado Article o Artículo. Este objeto permitió establecer los distintos atributos que era necesario mostrar.

Backend

Con el dominio definido, inicié el desarrollo de un servicio REST utilizando Java y Spring Boot. El enfoque fue implementar un diseño limpio, basado en arquitectura hexagonal, organizando la aplicación en capas: domain, application, infrastructure y shared. Esta estructura facilita la separación de responsabilidades, permitiendo aislar la lógica de negocio de las interfaces de usuario y de la capa de datos. Todas las dependencias están dirigidas hacia el núcleo (core) del dominio.

Agregué la lógica propia del dominio, como el cálculo de calificaciones basado en reseñas. Para optimizar el desarrollo, incorporé dependencias como Lombok (para generación automática de métodos getter y setter) y MapStruct (para el mapeo entre objetos).

En la capa de application definí los casos de uso. Implementé dos métodos principales: uno para obtener el detalle de un artículo, y otro para listar todos los artículos disponibles, necesario para tener una referencia de los mismos.

En infrastructure, desarrollé los componentes necesarios para la comunicación con clientes externos y definí el repositorio de datos. En la capa shared, ubiqué elementos transversales, comunes a varias capas, que por necesidades del framework no podían limitarse a una sola.

Realicé pruebas unitarias y de integración para validar el comportamiento de cada capa de forma independiente, permitiendo detectar errores antes de la integración total.

Finalmente, incorporé un archivo Dockerfile para contenerizar el backend, que posteriormente combiné con otro para el frontend mediante un docker-compose.yml, permitiendo el despliegue conjunto de ambos servicios.

Durante la validación, utilicé herramientas de inteligencia artificial como Gemini para revisar la estructura del proyecto. Esto me ayudó a identificar una interfaz que debía trasladarse a la capa de aplicación como un caso de uso. El resto del análisis no presentó observaciones.

Frontend

Para el frontend seleccioné Angular 18. Comencé definiendo el modelo de dominio y los componentes necesarios. Utilicé herramientas de inteligencia artificial para agilizar la

generación de código, lo que me permitió avanzar rápidamente en la definición de componentes y estilos.

El primer componente fue el encargado de mostrar el detalle del artículo, seguido de uno para listar artículos, que serviría como punto de partida para seleccionar un artículo específico.

Desarrollé un componente reutilizable para mostrar calificaciones con estrellas, ya que este formato se repetía en distintas secciones como el artículo, las recomendaciones y el vendedor.

Implementé una capa de servicios para la comunicación con el backend, usando HttpClient de Angular y RxJS para el manejo de observables. También utilicé ActivatedRoute para capturar parámetros de la URL y redirigir correctamente entre componentes.

Para dar un diseño similar al de MercadoLibre, añadí un header fijo con el logo y colores de la marca, así como una galería de imágenes, modales para métodos de pago, y diversos ajustes de estilo para lograr un diseño responsive.

Al final del desarrollo, incorporé un gestor de errores, que permite mostrar mensajes adecuados cuando el backend no está disponible o si falla la búsqueda de un artículo. También realicé pruebas unitarias usando mocks para validar el comportamiento de los componentes.

Integración y Despliegue

Para una entrega unificada, agrupé ambos proyectos (frontend y backend) en un repositorio principal que contiene dos submódulos de Git independientes, conservando así el historial de cambios de cada uno.

Agregué un archivo docker-compose.yml que permite compilar, construir y levantar ambos servicios sin necesidad de instalar manualmente Angular, Java o Node.js; únicamente se requiere tener Docker instalado.

Desafíos Relevantes

Uno de los desafíos más interesantes fue el diseño web. Representó un reto creativo importante, pero gracias al uso adecuado de herramientas y buenas prácticas, logré obtener un diseño funcional, estético y fiel al objetivo propuesto.