Reto: Gestión de Envíos y Rutas Logísticas

Una empresa de logística quiere desarrollar un sistema completo (frontend y backend) para gestionar el envío de paquetes, optimizar rutas de entrega y permitir a los clientes rastrear sus pedidos en tiempo real. La solución debe garantizar seguridad, eficiencia y escalabilidad..

Requerimientos Técnicos:

Backend

- Framework: Express o Fastify
- Lenguaje: TypeScript
- Base de datos: MySQL
- Caching: Redis
- Autenticación y Seguridad: JWT
- Patrones de Arquitectura: Clean Architecture
- Pruebas: Unitarias y de integración
- Documentación: Swagger

Frontend

- Framework: React
- Arquitectura: Microfrontends (Webpack Module Federation o Single SPA)
- Estado Global: Redux o Context API
- UI: TailwindCSS o Material-UI
- Rutas: React Router
- Seguridad: Manejo de autenticación con JWT
- Pruebas: Jest/React Testing Library

Historias de Usuario (HU):

HU1 - Registro y autenticación de usuarios

Descripción: Como usuario registrado, quiero poder iniciar sesión en la plataforma para gestionar mis envíos.

Criterios de Aceptación:

- Debe existir un endpoint para el registro de usuarios con validación de datos.
- Debe existir un endpoint de login que genera un token JWT.
- El sistema debe validar credenciales y retornar el token de autenticación.
- Formulario de login y registro con validaciones.
- Almacenar token en localStorage/SessionStorage y manejar autenticación.

HU2 - Creación de órdenes de envío

Descripción: Como usuario autenticado, quiero poder registrar un nuevo envío proporcionando los datos del paquete y la dirección de destino.
Criterios de Aceptación:

- Endpoint para registrar envíos con datos del paquete (peso, dimensiones, tipo de producto).
- Validación de que la dirección de destino es válida.
- Registro automático del estado inicial de la orden como "En espera".
- Formulario para crear envíos con validaciones.
- Notificación de confirmación al usuario.

HU3 - Asignación de rutas a los envíos

Descripción: Como administrador, quiero poder asignar un envío a una ruta y transportista para optimizar la entrega.
Criterios de Aceptación:

- Endpoint para asignar una orden de envío a una ruta específica.
- Registro del transportista asignado.
- Validación de disponibilidad de transportista y capacidad del vehículo.
- Dashboard administrativo para ver envíos y asignar rutas.
- Filtros para visualizar órdenes según estado.

HU4 - Seguimiento del estado del envío

Descripción: Como usuario autenticado, quiero poder consultar el estado de mis envíos en tiempo real.
Criterios de Aceptación:

- Endpoint para consultar el estado actual de un envío por ID.
- Uso de Redis para almacenar estados recientes y optimizar la consulta.
- Actualización de estado del pedido conforme avanza en la ruta: `En espera → En tránsito → Entregado`.
- Pantalla de seguimiento con actualización en tiempo real donde se vea todo el histórico de sus estados.
- Uso de WebSockets o polling para mantener el estado actualizado.

HU5 - Consulta avanzada de envíos y desempeño logístico

Descripción:
Como administrador logístico, quiero poder obtener reportes detallados sobre los envíos realizados, incluyendo tiempos de entrega, estado actual y transportistas involucrados, para optimizar la gestión operativa.

Criterios de Aceptación:

- Endpoint para consultar envíos con filtros avanzados (rango de fechas, estado del envío, transportista asignado).
- La consulta debe incluir métricas de desempeño, como el tiempo promedio de entrega por transportista y la cantidad de envíos completados en un período determinado.
- Optimización de la consulta mediante índices y paginación eficiente.
  Uso de **JOINs y subconsultas** en MySQL para extraer información relevante de múltiples tablas.
- Implementación de **Redis** para almacenar en caché las consultas más frecuentes y reducir la carga en la base de datos.
- Tabla con reportes y gráficos interactivos.
- Filtros para ajustar la consulta.

Instrucciones para el Entregable:

Repositorio de GitHub:

- Backend y frontend deben estar en repositorios separados.
- Debe contar con las ramas: `dev`, `test`, `main`.
- Debe existir historial de commits en `dev`.
- Se debe realizar merge de `dev` → `test` → `main`.
- Incluir un archivo `README.md` con instrucciones para instalación, configuración y ejecución.
- Incluir un enlace a un video privado (máx. 10 minutos) donde se explique y demuestre la solución el video debe aparecer el candidato siempre en un recuadro (tipo tutorial)

Criterios de Evaluación

✅ Código limpio y estructurado siguiendo principios SOLID y Clean Architecture.
✅ Uso adecuado de NodeJS y TypeScript.
✅ Eficiencia en las consultas y uso de Redis.
✅ Correcta implementación de seguridad con JWT.
✅ Uso correcto de patrones de desarrollo.
✅ Evidencia de gestión con Git (commits, branches, merges).
✅ Pruebas unitarias con buena cobertura.
✅ Elevator pitch en video explicando la solución.

Importante: El candidato está libre de realizar un diseño basado en su conocimiento y proponer piezas gráficas acorde a lo solicitado. Para referencias de colores, tipografías e imágenes podrá dirigirse a https:www.coordinadora.com, sin embargo el aspecto visual, simetría, composición, balance etc no serán tomados en cuenta para esta prueba y solo nos enfocaremos en la construcción de la interfaz.