Objetivo

- Desarrollar dos microservicios independientes que interactúen entre sí: Productos e Inventario.
- Implementar la funcionalidad básica de crear productos y realizar compras que descuenten del inventario.
- Demostrar el uso de Git Flow y buenas prácticas de desarrollo.
- Incluir pruebas unitarias y de integración como parte fundamental de la entrega.
- Aprovechar herramientas de IA para acelerar el desarrollo y mejorar la calidad del código.

Requisitos Técnicos

1. Usar el lenguaje de la vacante aplicada.
2. Implementar JSON API (https://jsonapi.org/) para todas las respuestas.
3. Docker para containerizar los servicios.
4. Base de datos: elegir entre SQLite, SQL, NoSQL (justificar la elección).

Microservicio 1: Productos

- **Modelo**: Producto con campos `id`, `nombre`, `precio`, `descripción` (opcional).
- **Funcionalidades**:
    - Crear un nuevo producto.
    - Obtener un producto específico por ID.
    - Listar todos los productos (opcional).

Microservicio 2: Inventario

- **Modelo**: Inventario con campos `producto_id`, `cantidad`.
- **Funcionalidades**:
    - Consultar la cantidad disponible de un producto específico por ID (obteniendo la información del producto desde el microservicio de productos).
    - Actualizar la cantidad disponible de un producto.
    - Registrar historial de compras (opcional).
    - Emitir un evento cuando el inventario cambie (opcional).

Flujo de Compra

1. Implementar un **endpoint de compra** que permita:
    - Recibir el ID del producto y la cantidad a comprar
    - Verificar la disponibilidad en inventario
    - Actualizar la cantidad disponible tras una compra exitosa
    - Retornar la información de la compra realizada
2. El candidato debe decidir y justificar en qué microservicio implementar este endpoint, considerando responsabilidades, acoplamiento y patrones de diseño.
3. La implementación debe mantener la consistencia de datos entre ambos servicios.
4. Manejar adecuadamente los casos de error (producto inexistente, inventario insuficiente, etc.).

Comunicación entre Microservicios

- HTTP usando JSON API.
- Autenticación básica entre servicios mediante API keys.
- Manejo de timeout y reintentos básicos.

Documentación

- Documentar los endpoints utilizando una de estas opciones:
    - Swagger / OpenAPI
    - Colección de Postman
- Incluir toda la documentación en el archivo README.md.
- Incluir diagramas de arquitectura e interacción entre servicios.

Requisitos de Testing

- Pruebas unitarias que cubran:
    - Creación de productos.
    - Gestión de inventario y proceso de compra.
    - Comunicación entre microservicios.
    - Manejo de errores (producto no encontrado, inventario insuficiente).
- Al menos una prueba de integración por microservicio.

Expectativas según Seniority

Nivel	Qué se espera
Junior	Implementación básica funcional de ambos microservicios. Docker básico para cada servicio. Pruebas unitarias básicas (≥ 40% cobertura).
Mid-level	Estructura siguiendo buenas prácticas. Docker Compose para orquestación. Manejo adecuado de errores y logs básicos. Pruebas unitarias con buena cobertura (≥ 60%). Documentación básica de API. Diagrama simple de arquitectura.
Senior	Solución robusta con arquitectura clara. Docker optimizado. Sistema de logs estructurados y health checks. Alta cobertura en pruebas (≥ 80%). Autenticación entre servicios. Documentación completa y diagrama detallado.
Líder Técnico	Todo lo anterior + patrones de diseño documentados, estrategia de versionado de API, guía de implementación y propuesta de mejoras para escalabilidad futura.

Criterios de Evaluación

- Cumplimiento con el estándar JSON API.
- Claridad en la interacción de microservicios.
- Calidad del código y buenas prácticas.
- Cobertura y calidad de las pruebas.
- Uso correcto de Git Flow.
- Calidad de la documentación.
- Uso efectivo y documentado de herramientas de IA para mejorar el desarrollo.

Entrega

- Plazo máximo: 2 días desde la recepción de la prueba.
- Repositorio público con la solución.
- Instrucciones claras para ejecutar los servicios y pruebas.

El README.md debe incluir

1. Instrucciones de instalación y ejecución.
2. Descripción de la arquitectura.
3. Decisiones técnicas y justificaciones, **incluyendo específicamente la decisión sobre dónde implementar el endpoint de compra**.
4. Diagrama de interacción entre servicios.
5. Explicación del flujo de compra implementado.
6. Documentación sobre el uso de herramientas de IA en el desarrollo (qué herramientas se utilizaron, para qué tareas específicas y cómo se verificó la calidad del código generado).

Nota importante

Prioriza la calidad sobre la cantidad de funcionalidades. Es mejor entregar menos características pero bien implementadas que muchas incompletas o con errores.