monokera

# PRUEBA TÉCNICA PARA TECH LEAD

¡Hola! Gracias por tu interés en formar parte del equipo **Monokera.** Te enviamos las instrucciones para resolver esta prueba técnica la cual está diseñada para conocer mejor tu lógica y capacidad en:

Desarrollar **APIs REST** en Rails.

- Implementar y conectar **microservicios**.
- Utilizar **PostgreSQL** para modelar datos.
- Aplicar **pruebas unitarias** y de integración.
- Implementar una arquitectura **event-driven** y manejar eventos.
- Aplicar **buenas prácticas de programación** y patrones de diseño.

## Descripción del Proyecto:

Construir una aplicación compuesta por dos microservicios: un **Servicio de Pedidos** (Order Service) y un **Servicio de Clientes** (Customer Service). El **Servicio de Pedidos** debe comunicarse con el **Servicio de Clientes** para obtener información adicional sobre los clientes cuando se crea un pedido.

## Microservicios:

1. **Servicio de Pedidos (Order Service)**:

Implementa un API en Rails para:

- **Crear pedidos**: Acepta detalles como `customer_id`, `product_name`, `quantity`, `price`, y `status`.
- **Consultar pedidos**. Permite obtener todos los pedidos asociados a un cliente específico (`customer_id`).
- Conecta con el **Servicio de Clientes** para obtener detalles adicionales del cliente al crear un pedido.
- Cada vez que se crea un pedido, el servicio debe **emitir un evento** a rabbitmq sobre la creación del pedido (nombre de la cola y routing keys a elección propia)

2. **Servicio de Clientes (Customer Service)**:

- Exponer un API en Rails con los siguientes endpoints:

- **Consultar información del cliente**: Devuelve información básica como `customer_name`, `address` y `orders_count`
- Permitir que el **Servicio de Pedidos** pueda consumir este endpoint para obtener datos adicionales del cliente.
- Debe existir un base de clientes predefinida, para que no sea necesario crear cada nuevo cliente.

## Especificaciones Técnicas

1. **Interacción entre Microservicios**:

   - Configura el **Servicio de Pedidos** para realizar una llamada HTTP al **Servicio de Clientes** al crear un pedido, obteniendo los detalles del cliente.

   - Se recomienda usar `HTTParty` o `Faraday` en Rails para simplificar las llamadas HTTP entre microservicios.

2. **Event-Driven Architecture**:

   - Cada vez que se cree un nuevo pedido, el **Servicio de Pedidos** debe generar un evento.

   - Dicho evento debe ser escuchado por **Customer Service** y actualizar el contador de pedidos hechos por el cliente (`orders_count`) asociado al pedido.

3. **Pruebas Unitarias e Integración**:

   - Implementa pruebas para asegurar:

     - La **creación y consulta de pedidos** en el **Servicio de Pedidos**.

     - La **integración y comunicación** entre el **Servicio de Pedidos** y el **Servicio de Clientes**.

     - La **generación de eventos**.

   - Utiliza **RSpec** (o MiniTest) para las pruebas y sigue buenas prácticas de organización de código.

4. **Documentación**:

   - Agrega un archivo **README** con instrucciones claras para:

- Configurar y ejecutar ambos servicios.

- Ejecutar las pruebas.

- Un diagrama breve de cómo ambos microservicios se relacionan y cómo el flujo de eventos interactúa en el sistema.

**Criterios de Evaluación**

1. **Calidad y Organización del Código**:

   - Organización y estructura.

   - Uso de patrones de diseño y principios SOLID.

2. **Dominio de Ruby on Rails y PostgreSQL**:

   - Capacidad para diseñar modelos y endpoints REST.

   - Buen manejo de base de datos en PostgreSQL.

3. **Comunicación entre Microservicios**:

   - Implementación de una comunicación sencilla y efectiva entre los microservicios.

4. **Arquitectura Event-Driven**:

   - Correcta implementación y gestión de eventos, demostrando una comprensión básica de sistemas basados en eventos.

5. **Pruebas y Cobertura**:

   - Calidad y cobertura de las pruebas.

   - Diseño de pruebas unitarias y de integración.

6. **Documentación**:

   - Claridad de la documentación y de las instrucciones.

   - Explicación de la arquitectura general y del flujo de trabajo entre microservicios y eventos.

monokera

**Entregables**

- Repositorio Git (GitHub, GitLab, etc.) con ambos microservicios.
- Documentación en README con instrucciones de configuración y ejecución.
- Scripts de migración y seeds para la base de datos.

**Fecha límite de entrega:** 28 de Julio de 2025