



# Desafío Técnico

## Backend Developer

### con Conocimientos en Java

***En esta prueba queremos que puedas demostrar tus conocimientos y capacidades de aprender algo nuevo y ponerlo en práctica***

#### Contexto:

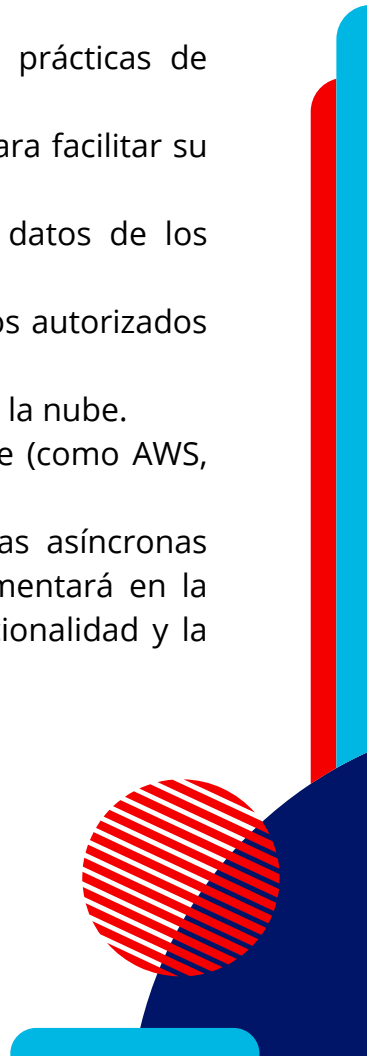
Una empresa está desarrollando un sistema de gestión de clientes para ofrecer una mejor experiencia a sus usuarios. Se requiere un microservicio que permita manejar el registro, consulta y análisis de datos de clientes. El objetivo es que este servicio sea escalable, seguro y cumpla con las mejores prácticas de desarrollo de software.

#### Requerimientos:

##### 1. Funcionalidades principales:

- Crear nuevos clientes mediante un endpoint que permita registrar nombre, apellido, edad y fecha de nacimiento.
- Consultar un conjunto de métricas sobre los clientes existentes, como el promedio de edad y la desviación estándar de las edades.
- Listar todos los clientes registrados con sus datos completos y un cálculo derivado, como una fecha estimada para un evento futuro basado en los datos del cliente (por ejemplo, esperanza de vida).

##### 2. Requisitos técnicos:

- Diseña el servicio utilizando un enfoque basado en buenas prácticas de diseño y patrones de arquitectura.
  - Documenta la API con herramientas estándar de la industria para facilitar su consumo.
  - Implementa un modelo de persistencia para almacenar los datos de los clientes.
  - Aplica principios de seguridad para garantizar que solo usuarios autorizados puedan interactuar con el servicio.
  - Diseña el sistema considerando su despliegue en un entorno en la nube.
  - Configurar el despliegue del servicio en un entorno en la nube (como AWS, Azure o GCP).
  - Implementa una solución de mensajería para gestionar tareas asíncronas relacionadas con los clientes. La funcionalidad que se implementará en la cola es de libre elección y criterio. Se deberá justificar la funcionalidad y la implementación en el README.
- 

### 3. Aspectos clave a evaluar:

- Estructura del proyecto: Se espera una separación clara de responsabilidades entre las capas del sistema.
- Gestión de errores: El servicio debe manejar adecuadamente las excepciones y devolver mensajes claros a los consumidores de la API.
- Pruebas: Se debe incluir un conjunto de pruebas que valide la funcionalidad implementada.
- Logs y monitoreo: Implementa un mecanismo que permita registrar y monitorear la actividad del servicio.
- Escalabilidad: Diseña el servicio de manera que pueda adaptarse a un aumento en el volumen de datos o peticiones.

### 4. Criterios de éxito:

- El microservicio debe ser funcional y cumplir con los requerimientos mencionados.
- Debe ser posible entender el diseño y las decisiones tomadas a través del código, documentación o comentarios.
- Se valorará el uso de patrones de diseño (por ejemplo, Factory, Builder, o Repository) y patrones de arquitectura (por ejemplo, RESTful o event-driven).
- La implementación debe ser eficiente y seguir las mejores prácticas para aplicaciones en Java y Spring Boot.

### Entrega:

- Sube el código a un repositorio público o privado (con acceso para los evaluadores).
- Incluye un archivo README.md explicando cómo ejecutar el proyecto, cómo realizar pruebas y cualquier otra información relevante.
- Describe brevemente en el README.md las decisiones arquitectónicas tomadas, los patrones de diseño aplicados y las herramientas utilizadas.

### Extras (Opcionales):

- Exponer métricas en un formato compatible con sistemas de monitoreo.