Prueba Técnica
Arquitectura Microservicios

Este documento presenta una prueba técnica destinada a evaluar las habilidades y conocimientos de los candidatos en el desarrollo de Java, con un enfoque específico en el uso de Spring Boot.

Indicaciones generales

- Aplique todas las buenas prácticas, patrones, etc. que considere necesarios.
- El manejo de la capa de datos se debe realizar con JPA.
- Utilizar Spring WebFlux.
- Realizar inyección de dependencias por constructor.
- Utilizar una arquitectura limpia.
- Implementar logs.
- Utilizar lombok.
- Implementar el manejo global de excepciones (RestControllerAdvice).
- Se debe realizar como mínimo dos pruebas unitarias.
- La codificación debe estar en inglés.
- **Implementar pruebas de mutación (Opcional).**
- **Utilizar el enfoque Contract First (API First).**
- La solución se debe desplegar con Docker.
- Posterior a la entrega de este ejercicio, se estará agendando una entrevista técnica donde el candidato deberá defender la solución planteada.

Herramientas y tecnologías que se deben utilizar

- Java versión 17 o superior.
- Spring Boot en su última versión estable.
- IDE de su preferencia.
- Base de Datos Relacional.
- Postman | Swagger OPENAPI | Karate DSL.

Complejidad por Seniority

Nota: Considerar las siguientes indicaciones en base al perfil al que está aplicando, Ejemplo: si es un perfil SemiSenior solo realizar lo indicado para este perfil.

Junior: Implementar los diferentes endpoints para cumplir las funcionalidades: F1, F2, F3, no es mandatorio funcionalidades: F4, F5, F6.

SemiSenior: Separar en 2 microservicios, agrupando (Cliente, Persona) y (Cuenta, Movimientos) donde se contemple una comunicación REST entre los 2 microservicios. Cumplir las funcionalidades: F1, F2, F3, F4, F5, deseable la funcionalidad F6.

Senior: Implementar en 2 microservicios, agrupando (Cliente, Persona) y (Cuenta, Movimientos) donde se contemple una comunicación asincrónica (Kafka, MQ, SQS, etc.) entre los 2 microservicios. Cumplir las funcionalidades F1, F2, F3, F4, F5, F6, F7
La solución debe contemplar (no necesariamente implementado) factores como: rendimiento, escalabilidad, resiliencia.

Generación de Api Rest “Application Programming Interface”

- Manejar correctamente los verbos: GET, POST, PUT, y DELETE.
- Manejar correctamente los códigos de estados HTTP.

Persona

- Implementar la clase persona con los siguientes datos: nombre, genero, identificación, dirección, teléfono

Cliente

- Cliente debe manejar una entidad, que herede de la clase persona.
- Un cliente tiene: contraseña, estado.

Cuenta

- Cuenta debe manejar una entidad con los siguientes campos: número, tipo, saldo Inicial, estado.

Movimientos

- Movimientos debe manejar una entidad con los siguientes campos: fecha, tipo, valor, saldo

Funcionalidades del API

F1: Generación de CRUD (Crear, editar, actualizar y eliminar registros - Entidades: Cliente, Cuenta y Movimiento).

Los nombres de los endpoints a generar son:

- /api/v1/accounts
- /api/v1/customers
- /api/v1/movements

F2: Registro de movimientos: al registrar un movimiento en la cuenta se debe tener en cuenta lo siguiente:

- El valor de un movimiento debe ser mayor que cero.
- Un movimiento de tipo débito se resta del saldo disponible.
- Un movimiento de tipo crédito se suma al saldo disponible.
- Se debe registrar cada una de las transacciones realizadas.

F3: Registro de movimientos: Al realizar un movimiento el cual no cuente con saldo, debe alertar mediante el siguiente mensaje “Saldo no disponible”

- Defina, según su expertise, la mejor manera de capturar y mostrar el error.

F4: Reportes: Generar un reporte de “Estado de cuenta” especificando un rango de fechas y cliente.

- Este reporte debe contener:
  - Cuentas asociadas con sus respectivos saldos
  - Detalle de movimientos de las cuentas

- EL endpoint de que sebe utilizar para esto debe ser el siguiente:
  - (/reports/{client-id}?startDate=fecha&endDate=fecha)

- El servicio del reporte debe retornar la información en formato JSON o Excel.

F5: Pruebas unitarias: Implementar pruebas unitarias del servicio de movimientos.

F6: Pruebas de Integración: Implementar 1 prueba de integración.

F7: Despliegue de la solución en contenedores.

Casos de Uso (Ejemplos)

1. Creación de Usuarios.

Nombres	Dirección	Teléfono	Contraseña	estado
Jose Lema	Otavalo sn y principal	098254785	1234	True
Marianela Montalvo	Amazonas y NNUU	097548965	5678	True
Juan Osorio	13 junio y Equinoccial	098874587	1245	True

2. Creación de Cuentas de Usuario.

Numero Cuenta	Tipo	Saldo Inicial	Estado	Cliente
478758	Ahorro	2000	True	Jose Lema
225487	Corriente	100	True	Marianela Montalvo
495878	Ahorros	0	True	Juan Osorio
496825	Ahorros	540	True	Marianela Montalvo

3. Crear una nueva Cuenta Corriente para José Lema

Numero Cuenta	Tipo	Saldo Inicial	Estado	Cliente
585545	Corriente	1000	True	Jose Lema

4. Realizar los siguientes movimientos

Numero Cuenta	Tipo cuenta	Saldo Inicial	Estado	Movimiento
478758	Ahorro	2000	True	Retiro de 575
225487	Corriente	100	True	Depósito de 600
495878	Ahorros	0	True	Depósito de 150
496825	Ahorros	540	True	Retiro de 540

5. Listado de Movimiento, por fechas y por usuario.

Fecha	Cliente	Número Cuenta	Tipo	Saldo Inicial	Estado	Valor movimiento	Tipo Movimiento	Saldo Disponible
10/2/2022	Marianela Montalvo	225487	Corriente	100	True	600	Crédito	700
8/2/2022	Marianela Montalvo	496825	Ahorros	540	True	540	Débito	0

Instrucciones de despliegue

- Generar el script de base datos, entidades y esquema datos, con el nombre BaseDatos.sql.

Entregables

- La solución se debe cargar en un repositorio Git público, y se debe enviar la ruta de este repositorio.
- Compartir la especificación OPENAPI (openapi-starter permite generar el .yaml)
- Compartir la colección de las pruebas de los endpoints.
- Se debe entregar antes de la fecha y hora indicada por correo.