Informe Ejercicio 1b-TP5

En el presente trabajo práctico, realizamos la implementación de un sistema de registro de estudiantes basado en microservicios a partir del monolito hecho en el ejercicio 9-Tp2 usando Spring Boot (Java). La estructura se basa en un sistema distribuido que comprende dos microservicios principales: el microservicio de estudiantes y el microservicio de carreras, los cuales interactúan para gestionar la información del registro académico.

Diseño Orientado a Microservicios

Se adoptó un enfoque orientado a microservicios con los siguientes objetivos y decisiones clave:

- **Modularización del sistema:** Se hizo una separación de funcionalidades en microservicios que permite manejar independientemente la lógica de estudiantes y carreras.
- **Escalabilidad y mantenimiento**: Esta arquitectura facilita que cada servicio pueda escalar y actualizarse sin afectar al otro.
- **Comunicación entre servicios**: Se implementa comunicación entre microservicios mediante RESTful APIs, donde cada servicio expone sus propios endpoints.

Microservicios Implementados

Para la implementación, elegimos dos microservicios:

- **Microservicio de Estudiantes**: Gestiona la información de los estudiantes, incluyendo datos personales y criterios de consulta específicos.
- **Microservicio de Carreras**: Administra los datos de carreras e interactúa con el servicio de estudiantes para validar y obtener información sobre la inscripción de estudiantes en carreras específicas. A su vez también administra la creación y obtención de inscripciones.

Implementación de Funcionalidades

Endpoints del Microservicio de Estudiantes

Este servicio proporciona los siguientes endpoints REST:

- **POST /estudiantes**: Dar de alta a un estudiante.
- **GET /estudiantes**: Recuperar todos los estudiantes
- **GET /estudiantes/{id}:** Recuperar un estudiante basado en su número de libreta universitaria.
- **GET /estudiantes/genero/{genero}**: Recuperar todos los estudiantes por género.
- **GET /estudiantes/ordenados/{criterio}**: Recuperar todos los estudiantes basados en un criterio de ordenamiento simple
- **GET /estudiantes/ciudad/{ciudad}:** Recuperar todos los estudiantes que viven en la ciudad indicada.

- **DELETE /estudiantes/{id}:** Eliminar un estudiante basado en su número de libreta universitaria.

Endpoints del Microservicio de Carreras

Este servicio proporciona los siguientes endpoints REST:

- **POST /inscripciones:**Matricular un estudiante en una carrera, donde se valida la existencia del estudiante.
- **POST /carreras:** Dar de alta una carrera.
- **GET /inscripciones:** Recuperar todas las inscripciones.
- **GET /carreras**: Recuperar todas las carreras.
- **GET /carreras/{id}:** Recuperar la carrera asociada a su número de id.
- **GET /carreras/{ciudad}/{carreraId}:** Recuperar estudiantes inscritos en una determinada carrera, filtrados por ciudad de residencia.
- **GET /carreras/generarReporte:** Generar un reporte de las carreras, que para cada carrera incluya información de los inscriptos y egresados por año. Ordenadas alfabéticamente y cronológicamente por año.
- **DELETE /carreras/{id}:** Eliminar una carrera basado en su número de id.

Comunicación e Interacción entre Microservicios

Para las operaciones de inscripción y consultas específicas, el microservicio de carreras interactúa con el microservicio de estudiantes mediante una solicitud REST. Por ejemplo:

- **Verificación de inscripción**: El servicio de carreras consulta al servicio de estudiantes para verificar si el estudiante existe antes de procesar la inscripción.
- **Filtrado por ciudad**: El servicio de carreras solicita al servicio de estudiantes los datos necesarios para filtrar los estudiantes por ciudad de residencia.
- **Generar reporte:**El servicio de carreras realiza una solicitud al servicio de estudiantes para obtener una lista completa de estudiantes inscritos. Esta lista se utiliza para generar un informe detallado que incluye la cantidad de inscriptos y egresados por año en cada carrera.

Implementación Técnica

- **Tecnología**: Se empleó Spring Boot para el desarrollo de ambos microservicios.
- **DTOs**: Ambos servicios utilizan DTOs para mejorar la transferencia de datos y evitar exponer la estructura interna.
- **Configuración y despliegue**: Los microservicios están configurados para ejecutarse independientemente y al mismo tiempo. La base de datos se gestiona a través de contenedores Docker para facilitar el entorno de desarrollo y pruebas.

Pruebas de Endpoints con Postman

Para probar los endpoints REST, se recomienda:

1.Configurar los datos de la base en los archivos `application.properties` de cada servicio y levantar la base de datos.
2.Levantar ambos microservicios simultáneamente.
3.Utilizar Postman para realizar solicitudes a cada uno de los endpoints y verificar su correcto funcionamiento.