

LABORATORIO –

Gestión de Estudiantes con Spring Boot

Construcción de un servicio REST para la gestión de estudiantes usando Spring Boot

En este laboratorio, debe implementar un **servicio web REST** utilizando **Spring Boot**, con el fin de gestionar un conjunto de estudiantes en una aplicación backend.

El propósito es reforzar conceptos de controladores REST, modelos de datos, servicios de negocio y manejo de colecciones en memoria.

El servicio permitirá realizar las operaciones básicas de un CRUD (**Crear, Leer, Actualizar y Eliminar**) sin utilizar todavía bases de datos externas.

Los datos serán almacenados en una colección interna para facilitar la comprensión de la arquitectura inicial.

Contexto del ejercicio:

La Universidad desea implementar un módulo básico para registrar estudiantes en un prototipo de plataforma académica. Por ahora, solo se necesita un servicio backend que permita registrar estudiantes y consultar su información a través de rutas HTTP.

Este laboratorio sirve como base para futuros módulos como matrículas, asignaturas o notas.

El objetivo principal es que el estudiante comprenda el flujo entre:

- **Controlador REST**
- **Servicio de lógica de negocio**
- **Modelo de datos**
- **Uso de rutas HTTP (GET, POST, PUT, DELETE)**

Para controlar el modelo se deben usar:

- Crear controladores REST usando `@RestController` y `@RequestMapping`.

`@GetMapping`, `@PostMapping`, `@PutMapping`, `@DeleteMapping`, `@Service`, Manejo de listas en memoria (`ArrayList`).

- Implementar modelos de datos en Java.
- Separar responsabilidades mediante un servicio de negocio (@Service).
- Crear endpoints que reciban y respondan datos en formato JSON.
- Usar Postman o navegador para consumir servicios REST.
- Comprender el funcionamiento básico de un CRUD antes de integrar bases de datos.

Requerimientos funcionales:

El sistema debe permitir:

1. **Listar todos los estudiantes** (GET).
2. **Buscar un estudiante por ID** (GET).
3. **Registrar un nuevo estudiante** (POST).
4. **Actualizar los datos de un estudiante existente** (PUT).
5. **Eliminar un estudiante por ID** (DELETE).

Cada estudiante debe contener como mínimo:

- id (entero)
- nombre (cadena de texto)
- edad (entero)

Los datos iniciales deben incluir **al menos dos estudiantes de ejemplo** cargados en memoria.

OJO:

- Código fuente completo del proyecto con las siguientes capas:
model
service
Controller
1. Capturas de pantalla de las pruebas de los endpoints:
 - Listar estudiantes

Gestión de Estudiantes

Listar todos los estudiantes

ID	Nombre	Edad
1	Juan Pérez	20
2	Maria López	21
3	juan	19

Registrar nuevo estudiante

I. Estudiante registrado: juan (ID: 3)

- Buscar por ID

Listar todos los estudiantes

ID	Nombre	Edad
1	Juan Pérez	20
2	Maria López	21
3	juan	19

Registrar nuevo estudiante

Buscar estudiante por ID

ID: 3, Nombre: juan , Edad: 99

- Crear estudiante

Listar todos los estudiantes

ID	Nombre	Edad
1	Juan Pérez	20
2	Maria López	21
3	juan	19

Registrar nuevo estudiante

I. Estudiante registrado: juan (ID: 3)

- Actualizar estudiante

Gestión de Estudiantes

Listar todos los estudiantes

Mostrar todos		
ID	Nombre	Edad
1	Juan Pérez	20
2	Maria López	21
3	kevin	20

Registrar nuevo estudiante

Nombre Edad Registrar
Estudiante registrado: juan (ID: 3)

Buscar estudiante por ID

Buscar
ID: 3, Nombre: juan , Edad: 99

Actualizar estudiante

ID del estudiante Nuevo nombre Nueva edad Actualizar
Estudiante actualizado: kevin

- I. ➤ Eliminar estudiante

Listar todos los estudiantes

Mostrar todos		
ID	Nombre	Edad
1	Juan Pérez	20
2	Maria López	21

Registrar nuevo estudiante

Nombre Edad Registrar
Estudiante registrado: juan (ID: 3)

Buscar estudiante por ID

Buscar
ID: 3, Nombre: juan , Edad: 99

Actualizar estudiante

ID del estudiante Nuevo nombre Nueva edad Actualizar
Estudiante actualizado: kevin

Eliminar estudiante

ID del estudiante Eliminar
Estudiante eliminado correctamente

2. Documento corto respondiendo:

- ¿Qué ventajas y desventajas tiene usar listas en memoria?

RTA

VENTAJAS

- La facilidad a la hora de realizar el código, esto debido a que se puede implementar de manera mas directa y rápida sin necesidad de utilizar dependencias, librerías o Base de datos.
- La velocidad a la hora de acceder a los datos y realizar las operaciones necesarias (lógica del negocio), será muy alta, esto debido a que los datos se encuentran en la memoria.
- Es muy bueno para realizar prototipos, hacer pruebas unitarias o un desarrollo rápido fácil y rápido sin necesidad de crear una base de datos.

DESVENTAJAS

- La facilidad con la que se pueden perder los datos, ya que al tener almacenada la información en la memoria, con solo reiniciar el equipo se perdería toda la información.
 - En aplicaciones mas grandes con una gran cantidad de datos puede que el espacio de almacenamiento no sea suficiente, lo que generaría que “colapsara”, en estos casos es mejor tener una base de datos, que tiene una mejor gestión del espacio.
 - Complejidad a la hora de hacer la aplicación de manera escalable, ya que un ArrayList es muy limitado en comparación con una base de datos, que da mayor flexibilidad e integridad cuando toca guardar registros.
 - No se podrían realizar funciones más complejas como consultas, índices, y la seguridad del sistema sería muy pobre, ya que una lista no tiene la integridad que si te puede ofrecer una base de datos.
- ¿Qué ocurriría si hubiese múltiples usuarios accediendo al sistema al mismo tiempo?

RTA

Si muchos usuarios accedieran a un sistema de manera simultánea, ocurrirían varios problemas entre ellos tenemos:

Si varios usuarios tratan de modificar los mismos datos al mismo tiempo, uno de los cambios se perderá, además que la información que se muestre a cada usuario puede estar desactualizada, esto debido a los constantes cambios y sin alguna gestión para que el sistema los muestre a todos los usuarios en el preciso momento en que se realizan las modificaciones, también se pueden generar bloqueos, esto debido a que puede que los usuarios requieran de algún recurso para poder operar, provocando que el sistema los haga esperar, hasta que el recurso quede libre (interbloqueo).

El rendimiento del sistema disminuye considerablemente, debido a que, al tener tantos usuarios en sus filas, al sistema le cuesta muchos recursos poder atenderlos a todos generando cuellos de botellas, y si el sistema no tiene una alta capacidad

podría colapsar. Si el sistema no tiene sincronización los datos podrían corromperse.

- ¿Qué mejorarías si usaras una base de datos real?

RTA

La persistencia o duración de los datos sería mayor, esto porque los datos se mantendrían guardados, aun si se reinicia el sistema o equipo, también a largo plazo los registros que se van almacenando no se pierden, sino que se guardan en una o varias tablas, también si llega a ocurrir un fallo, permite el poder recuperar los datos o al menos la mayoría de estos (siempre y cuando se tenga algún respaldo de la base de datos, lo cual es muy recomendable).

Las bases de datos también son muy flexibles en cuanto al acceso concurrente de usuarios ya sean mediante bloqueos (locks) automáticos o transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad). Esto evita condiciones de carrera y pérdida de información cuando varios usuarios escriben al mismo tiempo.

Las bases de datos son escalables, es decir, si el sistema maneja una gran cantidad de registros no tendría problemas con la memoria RAM, lo que hace que el sistema corra en óptimas condiciones y a largo plazo sea una práctica excelente.

Conclusión: Aunque realizar listas locales y almacenamiento en la memoria es más rápido no es lo más recomendable, si se va a realizar un sistema duradero y que va estar manejando y operando con datos constantemente, lo más óptimo es tomarse el tiempo de realizar un script para correr una Base de datos en cualquier lenguaje SQL, sea MySQL, PostgreSQL, Maria DB, Oracle, etc. No solo permite un mejor y más seguro almacenamiento de los datos, sino también respeta la integridad de los mismos, haciendo que sea difícil eliminar un campo importante como algún id, cédula o algún otro campo de primary key, esto por la integridad que existe.