

 UNIVERSIDAD DON BOSCO	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERIA ESCUELA DE COMPUTACION
CICLO III	DESAFIO PRACTICO #1 Lugar de Ejecución: Centro de Cómputo Estudiante: Paola Matilde Orellana Castillo OC250609 MATERIA: Desarrollo de Aplicaciones con Web Frameworks DWF404

1. Lógica de Negocio – Cálculo del Promedio

En la capa de servicio implementé la lógica necesaria para calcular el promedio general de mis materias registradas en la base de datos.

Primero obtengo todas las materias utilizando el método `findAll()` del repositorio. Este método retorna una lista de objetos `Materia`.

Posteriormente utilice la API de Streams de Java para procesar la información:

- `mapToDouble(Materia::getNota)` permite extraer únicamente el valor numérico de cada nota.
- `average()` calcula el promedio de las notas.
- `orElse(0.0)` evita errores en caso de que la lista esté vacía.

Este enfoque permite que el cálculo sea dinámico y automático, ya que si se agregan nuevas materias, el promedio se recalcula sin necesidad de modificar el código.

2. Análisis de Inyección de Dependencias

La inyección del repositorio en el servicio fue realizada mediante el constructor, evitando el uso de `new` o `@Autowired` en atributos.

Las ventajas de este enfoque son:

- Reduce el acoplamiento entre clases.
- Permite que Spring gestione automáticamente las dependencias.
- Facilita la creación de pruebas unitarias.
- Garantiza que el servicio no pueda instanciarse sin su dependencia obligatoria.

Si se hubiera utilizado new, el servicio quedaría fuertemente acoplado al repositorio, afectando la mantenibilidad y dificultando las pruebas.

3. Comportamiento de Perfiles (dev vs prod)

Se configuraron dos perfiles en el proyecto para manejar diferentes entornos de ejecución:

Perfil dev

Utiliza una base de datos H2 en memoria (jdbc:h2:mem).

Esto significa que los datos se eliminan automáticamente al cerrar la aplicación.

Perfil prod

Utiliza una base de datos H2 persistente (jdbc:h2:file).

En este caso, los datos se almacenan en un archivo físico, por lo que permanecen después de cerrar la aplicación.

La activación del perfil se realiza mediante:

```
spring.profiles.active=dev
```

Esto permite cambiar entre entornos sin modificar el código fuente, únicamente alterando la configuración.

4. Uso de Lombok

En la entidad Materia utilicé las siguientes anotaciones de Lombok:

- `@Getter`
- `@Setter`
- `@NoArgsConstructor`
- `@AllArgsConstructor`
- `@Builder`

Estas anotaciones reducen la cantidad de código repetitivo (boilerplate), ya que generan automáticamente los métodos necesarios.

Esto mejora la legibilidad del proyecto y agiliza el desarrollo.

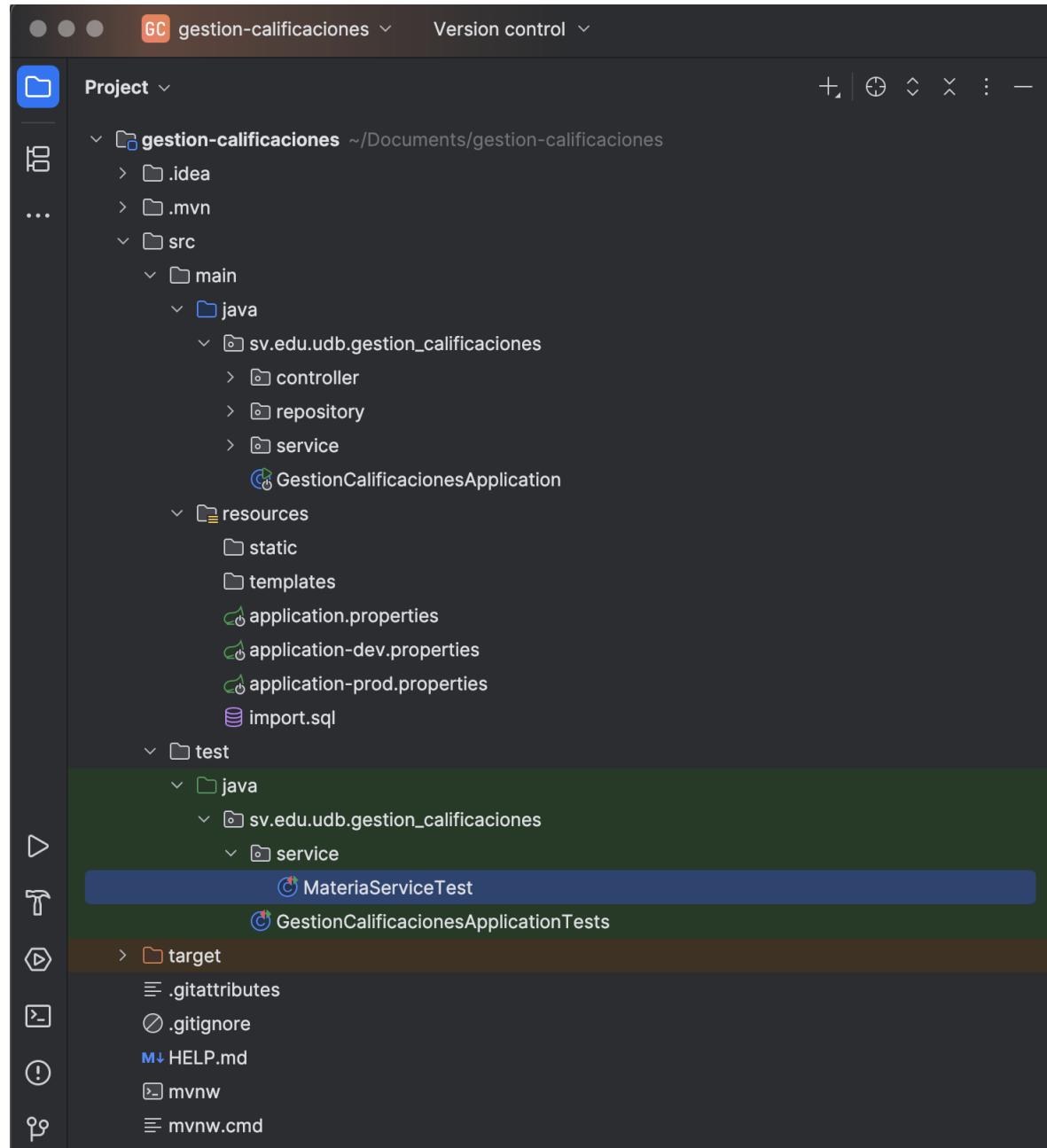
5. Pruebas Unitarias

Se implementaron dos pruebas utilizando JUnit 5:

1. Una prueba que verifica que la lista de materias no sea nula.
2. Una prueba que valida que el promedio calculado coincida con los datos cargados desde import.sql.

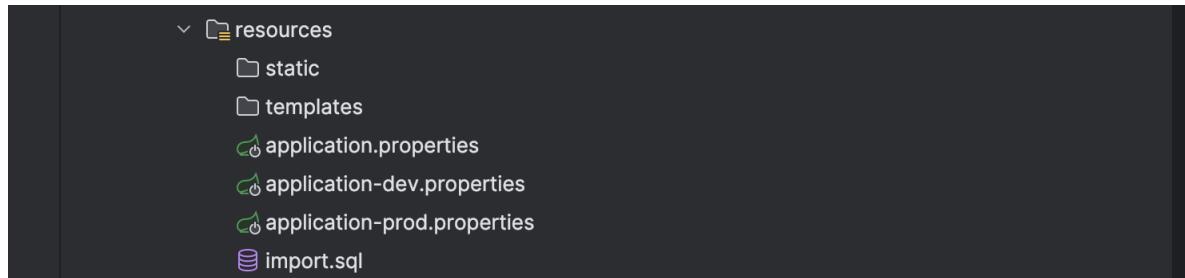
Ambas pruebas se ejecutaron correctamente y finalizaron sin errores, lo que confirma que la lógica implementada funciona como se espera.

6. CAPTURAS DEL SISTEMA



The screenshot shows a Java code editor interface. The main window displays the file `MateriaService.java`. The code defines a service class `MateriaService` that interacts with a `MateriaRepository` to retrieve a list of `Materia` objects and calculate their average note using a Stream API. Several code completion suggestions are visible in a tooltip over the `OptionalDouble` return type of the `average()` method.

```
1 package sv.edu.udb.gestion_calificaciones.service;
2
3 import org.springframework.stereotype.Service;
4 import sv.edu.udb.gestion_calificaciones.repository.MateriaRepository;
5 import sv.edu.udb.gestion_calificaciones.repository.domain.Materia;
6
7 import java.util.List;
8
9 @Service 4 usages
10 public class MateriaService {
11
12     private final MateriaRepository repository; 3 usages
13
14     public MateriaService(MateriaRepository repository) {
15         this.repository = repository;
16     }
17
18     public List<Materia> obtenerMaterias() { 2 usages
19         return repository.findAll();
20     }
21
22     public double calcularPromedio() { 2 usages
23         List<Materia> materias = repository.findAll();
24
25         return materias.stream() Stream<Materia>
26             .mapToDouble(Materia::getNota) DoubleStream
27             .average() OptionalDouble
28             .orElse( other: 0.0);
29     }
30 }
```



The screenshot shows an IDE interface with a code editor window titled 'import.sql'. The code contains several INSERT INTO statements for a 'materia' table, each specifying a name, note, and cycle. A warning message at the top states 'No data sources are configured to run this SQL and provide advanced code assistance.' The code is as follows:

```
1  INSERT INTO materia (nombre_materia, nota, ciclo) VALUES ('Administración de Servicios en la Nube', 8.7, 'Ciclo I-2026');
2  INSERT INTO materia (nombre_materia, nota, ciclo) VALUES ('Diseño y Programación de Software Multiplataforma', 9.1, 'Ciclo I-2026');
3  INSERT INTO materia (nombre_materia, nota, ciclo) VALUES ('Desarrollo de Aplicaciones Web con Software Interpretado en el Servidor', 8.5, 'Ciclo I-2026');
4  INSERT INTO materia (nombre_materia, nota, ciclo) VALUES ('Desarrollo de Aplicaciones Web con Frameworks en Plataformas Propietarias', 9.3, 'Ciclo I-2026');
5  INSERT INTO materia (nombre_materia, nota, ciclo) VALUES ('Proyecto Integrador', 8.9, 'Ciclo I-2026');
```

The screenshot shows a web browser window with the URL 'localhost:8080/materias'. The page displays a JSON response with five objects, each representing a subject (materia) with its name, note, and cycle. The JSON is as follows:

```
[{"id":1,"nombreMateria":"Administración de Servicios en la Nube","nota":8.7,"ciclo":"Ciclo I-2026"}, {"id":2,"nombreMateria":"Diseño y Programación de Software Multiplataforma","nota":9.1,"ciclo":"Ciclo I-2026"}, {"id":3,"nombreMateria":"Desarrollo de Aplicaciones Web con Software Interpretado en el Servidor","nota":8.5,"ciclo":"Ciclo I-2026"}, {"id":4,"nombreMateria":"Desarrollo de Aplicaciones Web con Frameworks en Plataformas Propietarias","nota":9.3,"ciclo":"Ciclo I-2026"}, {"id":5,"nombreMateria":"Proyecto Integrador","nota":8.9,"ciclo":"Ciclo I-2026"}]
```

The screenshot shows a web browser window with the URL 'localhost:8080/materias/promedio'. The page displays a single numerical value '8.9'.

The screenshot shows a terminal window with two separate test runs for the 'MateriaServiceTest' class. Both runs show one test passed. The first run took 254 ms and the second took 385 ms. The output includes Java startup logs and the test results.

Run MateriaServiceTest.listaNoDebeSerNula x MateriaServiceTest.promedioDebeSerCorrecto x

MateriaServiceTest (sv.edu.udb.gestion_calificaciones.ser 254 ms) 1 test passed 1 test total, 254 ms

```
java -jar /Users/paolacostillo/Library/Java/JavaVirtualMachines/openjdk-25/Contents/Home/bin/java ...  
23:06:28.939 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could  
23:06:28.992 [main] INFO org.springframework.boot.test.context.SpringBootTestBootstrapper -- Found @Spr
```

MateriaServiceTest (sv.edu.udb.gestion_calificaciones.ser 385 ms) 1 test passed 1 test total, 385 ms

```
java -jar /Users/paolacostillo/Library/Java/JavaVirtualMachines/openjdk-25/Contents/Home/bin/java ...  
23:06:26.543 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could  
23:06:26.613 [main] INFO org.springframework.boot.test.context.SpringBootTestBootstrapper -- Found @Spr
```