



11^a
Emisión

DIPLOMADO
Desarrollo de Sistemas
con Tecnología Java

Módulo 7
Persistencia con Spring Data

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC



Educación
Continua
1971 - 2021

JPA

- JPA es una especificación de Java, es decir, es un estándar para un framework ORM.
- JPA menciona las reglas que se deben seguir para que pueda existir la interacción con las tablas de la base de datos en forma de objetos
- En otras palabras, JPA es la teoría y algunas de las implementaciones de esta teoría son: Hibernate, EclipseLink, TopLink.



JPA

- JPA usa anotaciones para evitar usar de manera nativa sentencias SQL.
 - `@Entity`
 - indica a una clase de java que está representando una tabla de BD.
 - `@Table`
 - recibe el nombre de la tabla a la cual está mapeando la clase.
 - `@column`
 - se asigna a los atributos de la clase, no es obligatoria, se indica sólo cuando el nombre de la columna es diferente al nombre del atributo de la tabla.



JPA

- JPA usa anotaciones para evitar usar de manera nativa sentencias SQL.
 - @id y @EmbededID
 - es el atributo como llave primaria de la tabla dentro de la clase. @id se utiliza cuando es llave primaria sencilla y @EmbededID cuando es una llave primaria compuesta
 - @GeneratedValue
 - permite generar automáticamente valores para atributos de la db.
 - @OneToMany y @MatyToOne
 - permite relacionar diferentes entidades/clases.
 - También se hace uso de las anotaciones @JoinColumn.

Spring Data Repositories

- Permiten ahorrar código y tiempo de implementación al poder realizar operaciones en la base de datos sin sentencias SQL.
- Existen diferentes repositorios en Spring Data
 - CrudRepository: para realizar operaciones CRUD.
 - PagindAndSortingRepository: además de las operaciones CRUD, se puede paginar y ordenar.
 - JPARespository: agregá tareas específicas a las anteriores, como flush.

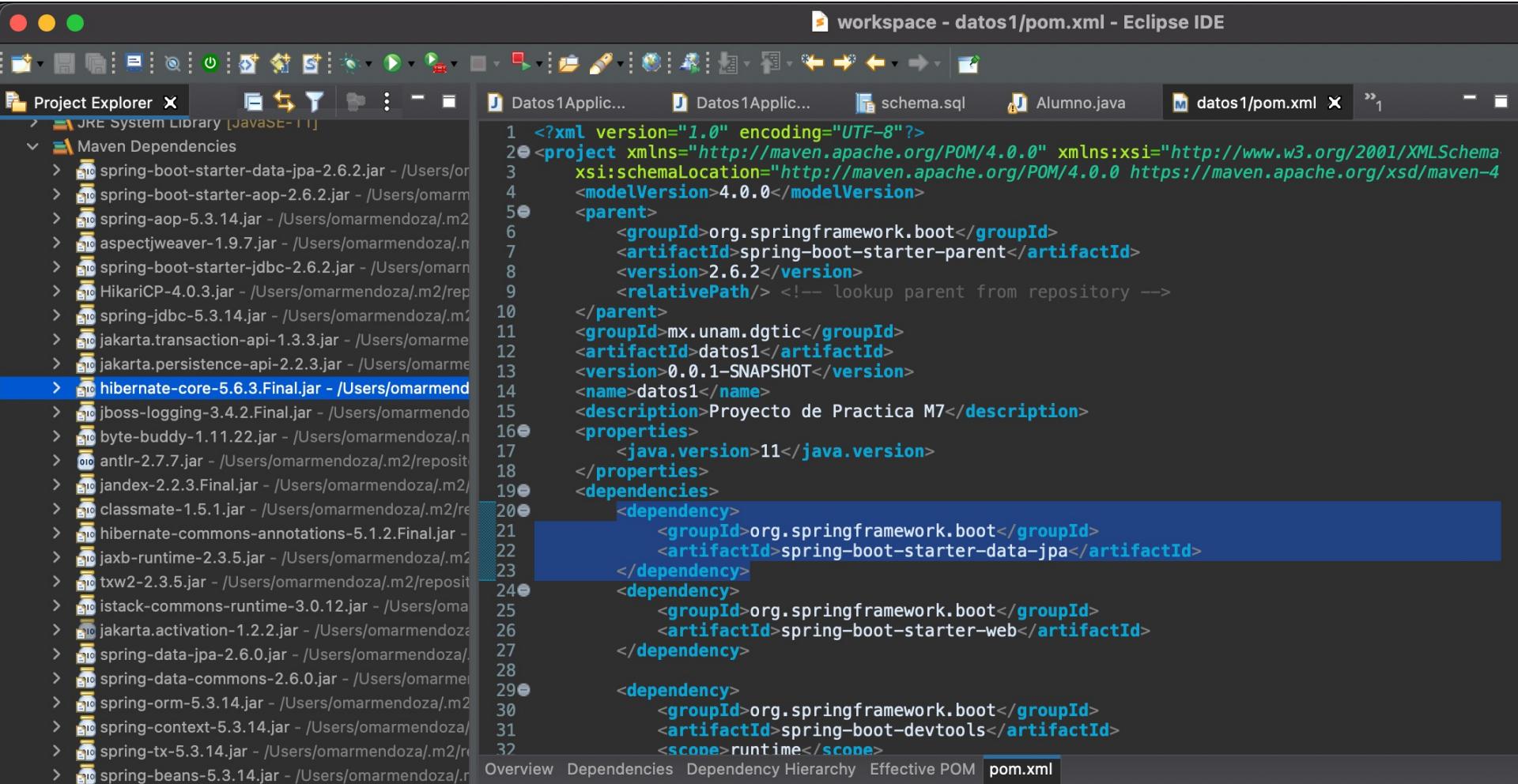
Spring Data Repositories

- A la clase que interactúa con la base de datos mediante una instancia de un repositorio, se le debe agregar alguna de las siguientes anotaciones:
 - `@Repository`
 - le indica a la clase que es la encarga de interactuar con la bd.
 - `@Component`
 - le indica que es un componente de spring.

Query Methods

- Cuando se necesitan consultas y el repositorio de Spring Data no puede hacer estas consultas, entonces los Query Method proveen la posibilidad de generar consultas mediante el nombre de los métodos.
- Son compatibles con programación funcional de Java.
- En lugar de usar *Query Methods*, se puede usar la anotación
 - `@Query`
 - para escribir consultas (select * from tabla where condición).

Visualizar dependencias



The screenshot shows the Eclipse IDE interface with the title "workspace - datos1/pom.xml - Eclipse IDE". The left side features the "Project Explorer" view, which lists various Maven dependencies under the "JRE System Library [JAVASE-11]". The right side shows the content of the "pom.xml" file in the editor. A specific dependency declaration is highlighted with a blue selection bar:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

The "pom.xml" file content is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"
  modelVersion="4.0.0">
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.2</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>mx.unam.dgtic</groupId>
  <artifactId>datos1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>datos1</name>
  <description>Proyecto de Practica M7</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scone>runtime</scone>
    </dependency>
  </dependencies>

```

Entidades

@Entity

Alumno

@matricula

nombre

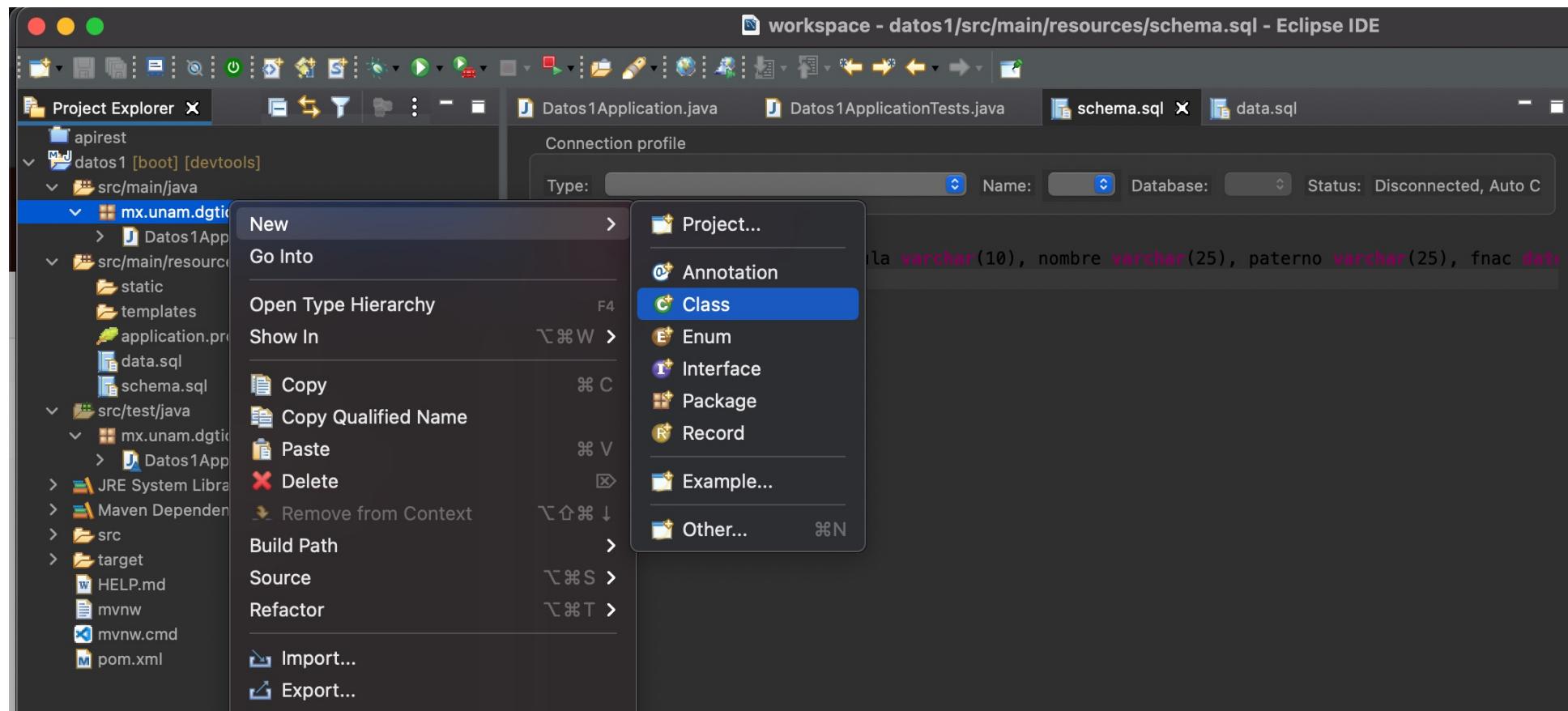
paterno

fnac

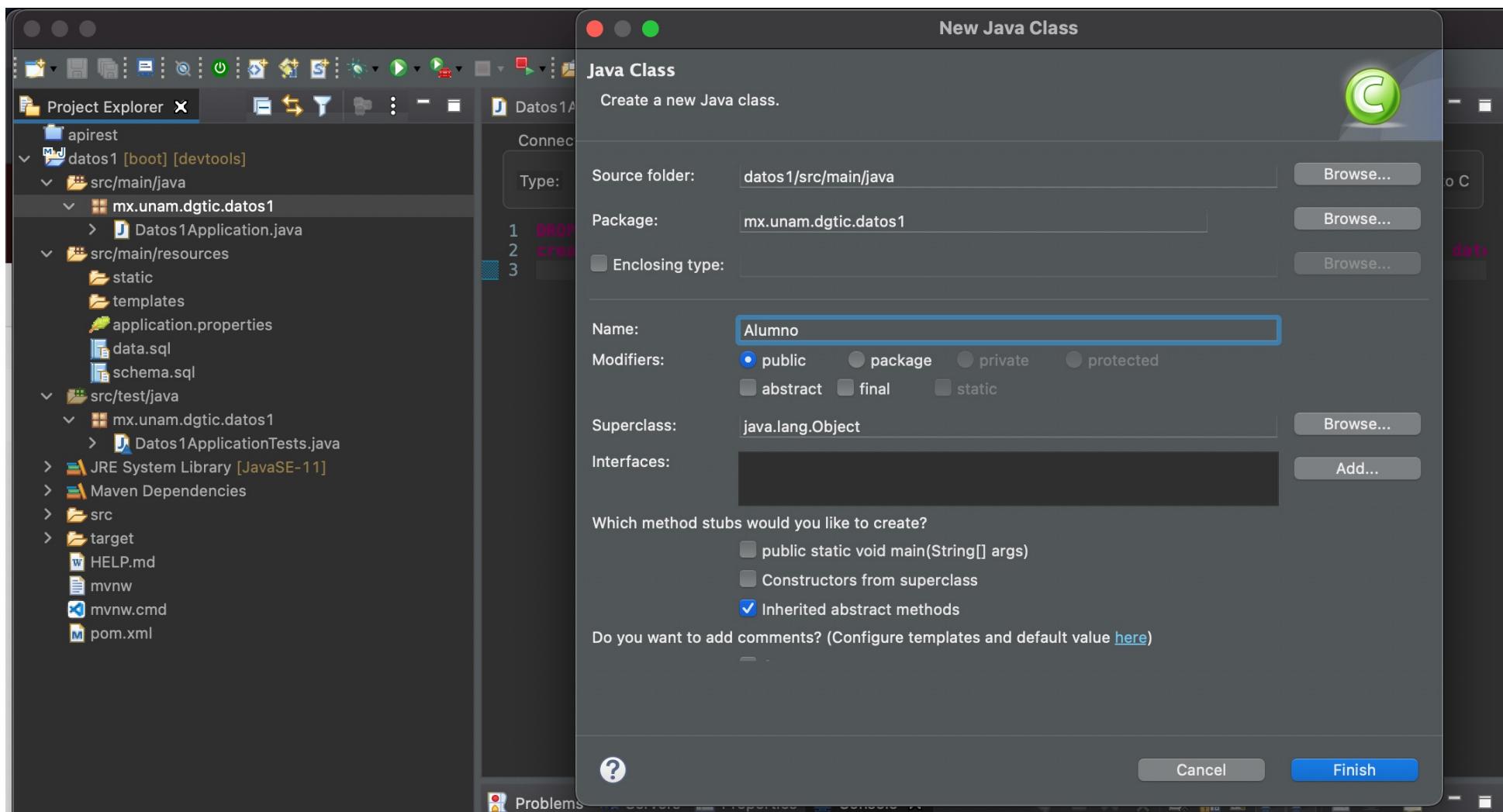
estatura



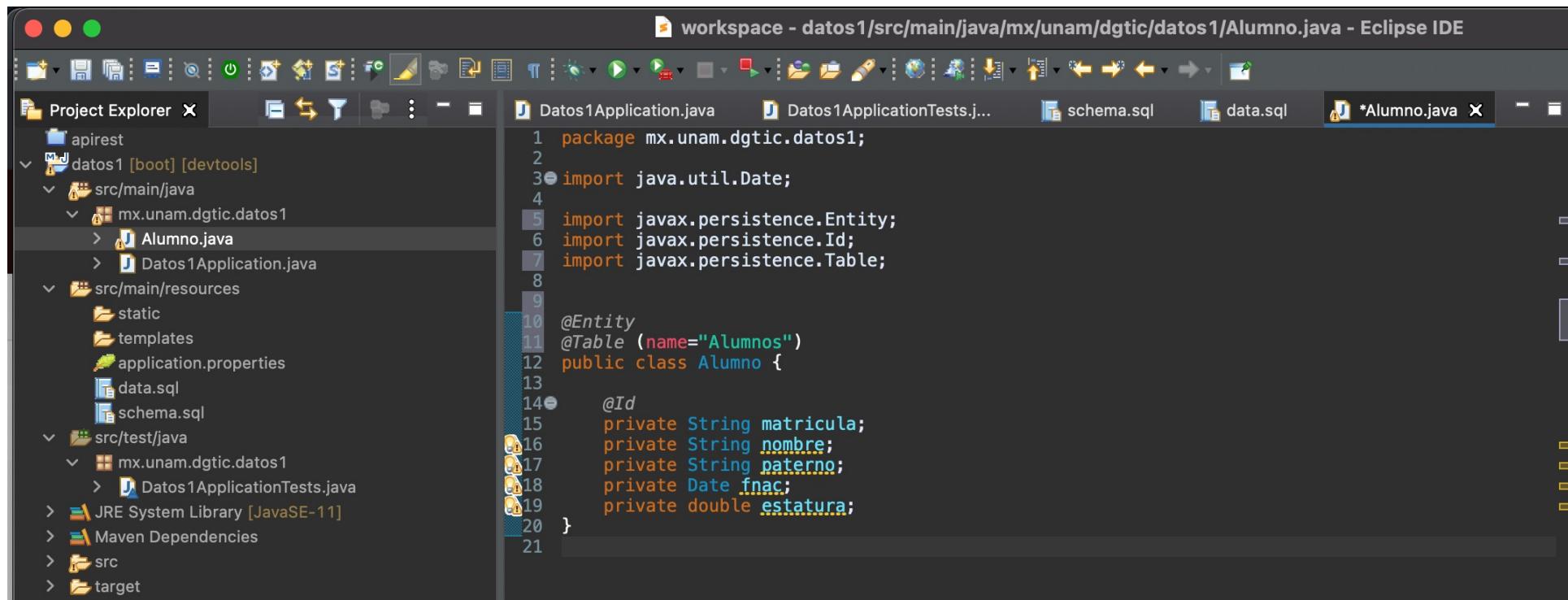
Generar una entidad



Generar una entidad



@Entity @Table

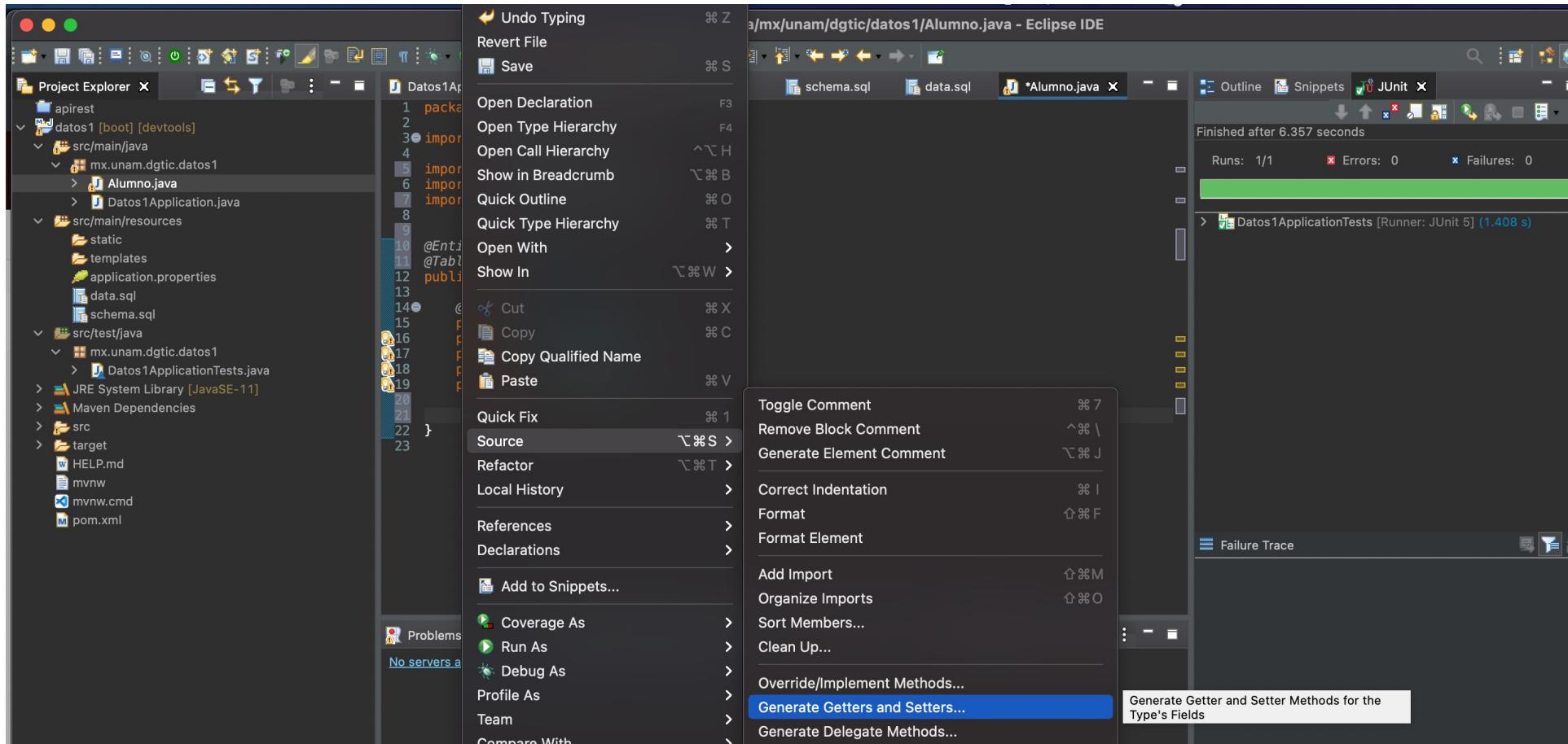


The screenshot shows the Eclipse IDE interface with the following details:

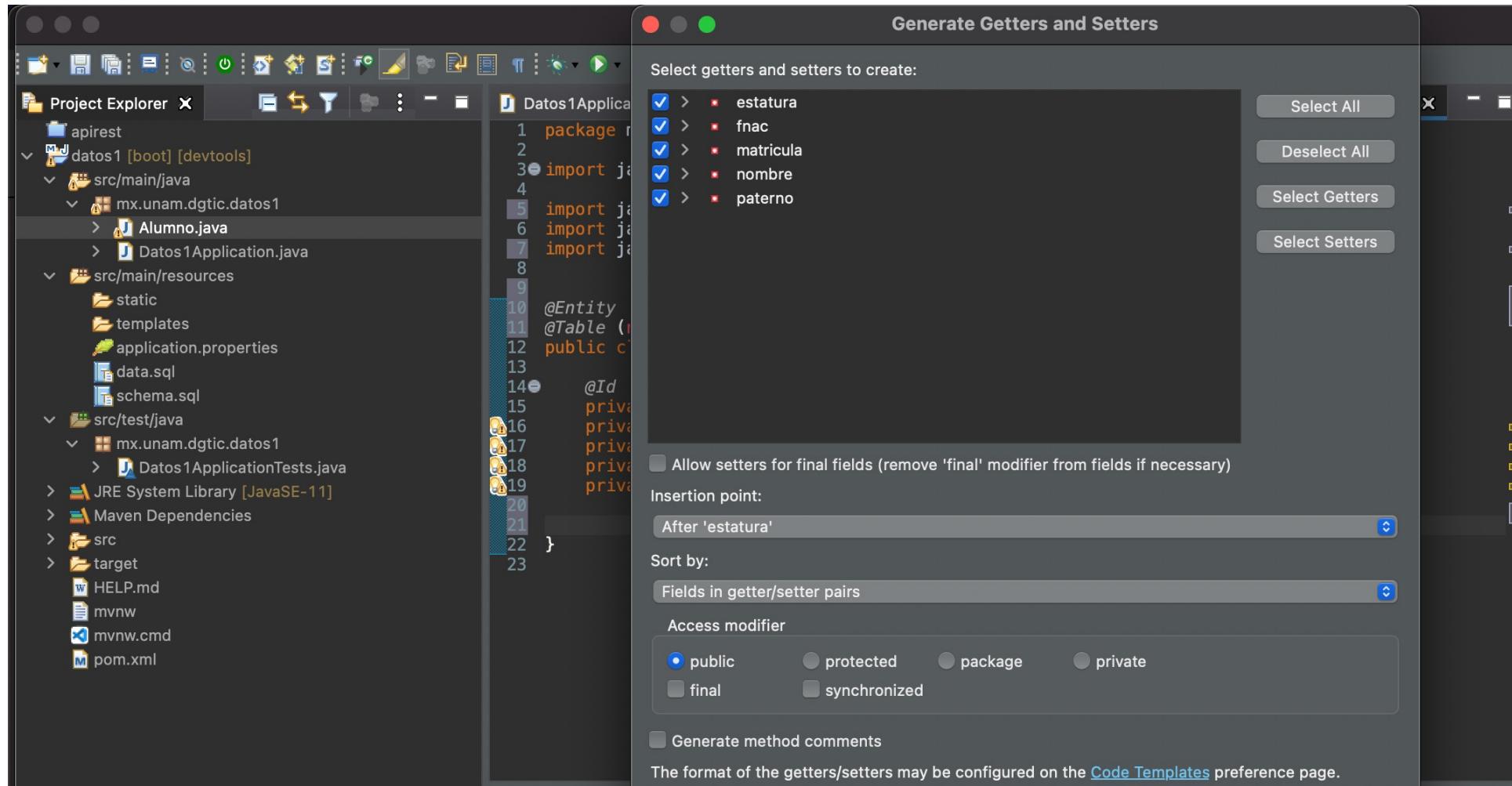
- Title Bar:** workspace - datos1/src/main/java/mx/unam/dgtic/datos1/Alumno.java - Eclipse IDE
- Project Explorer:** Shows the project structure:
 - apirest
 - datos1 [boot] [devtools]
 - src/main/java
 - mx.unam.dgtic.datos1
 - Alumno.java
 - Datos1Application.java
 - src/main/resources
 - static
 - templates
 - application.properties
 - data.sql
 - schema.sql
 - src/test/java
 - mx.unam.dgtic.datos1
 - Datos1ApplicationTests.java
 - JRE System Library [JavaSE-11]
 - Maven Dependencies
 - src
 - target
 - Code Editor:** Displays the Alumno.java code:

```
1 package mx.unam.dgtic.datos1;
2
3 import java.util.Date;
4
5 import javax.persistence.Entity;
6 import javax.persistence.Id;
7 import javax.persistence.Table;
8
9
10 @Entity
11 @Table (name="Alumnos")
12 public class Alumno {
13
14     @Id
15     private String matricula;
16     private String nombre;
17     private String paterno;
18     private Date fnac;
19     private double estatura;
20 }
21
```

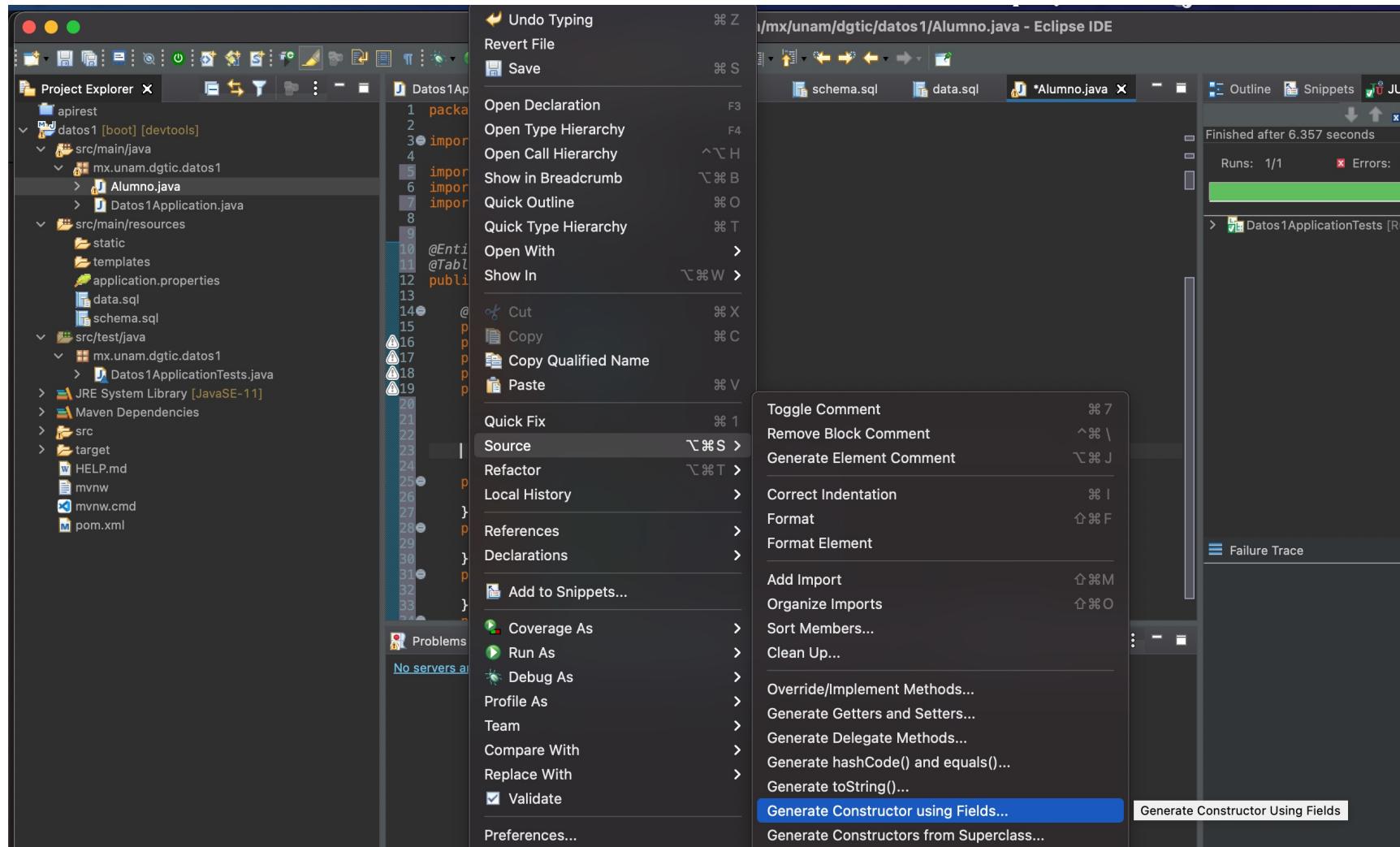
Generar Setters y Getters



Generar Setters y Getters



Generar constructores



Generar constructores

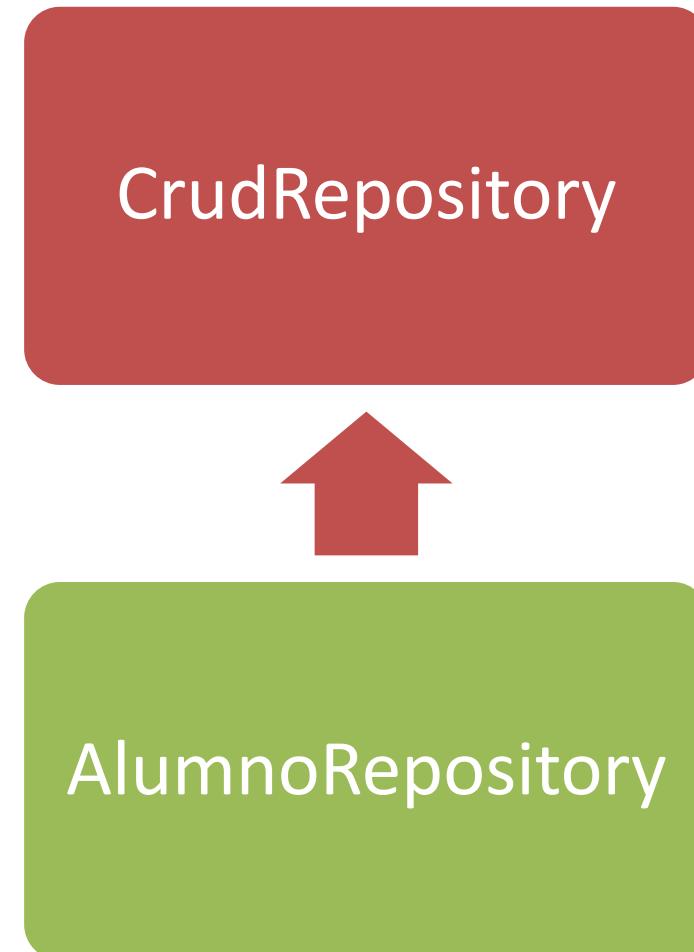
```
21
22•    public Alumno(String matricula, String nombre, String paterno, Date fnac, double estatura)
23        super();
24        this.matricula = matricula;
25        this.nombre = nombre;
26        this.paterno = paterno;
27        this.fnac = fnac;
28        this.estatura = estatura;
29    }
30
31•    public Alumno(String matricula) {
32        super();
33        this.matricula = matricula;
34    }
35
36    |
37•    public Alumno() {
38        super();
39    }
40
```



Repositorio



Repositorio



Spring Data

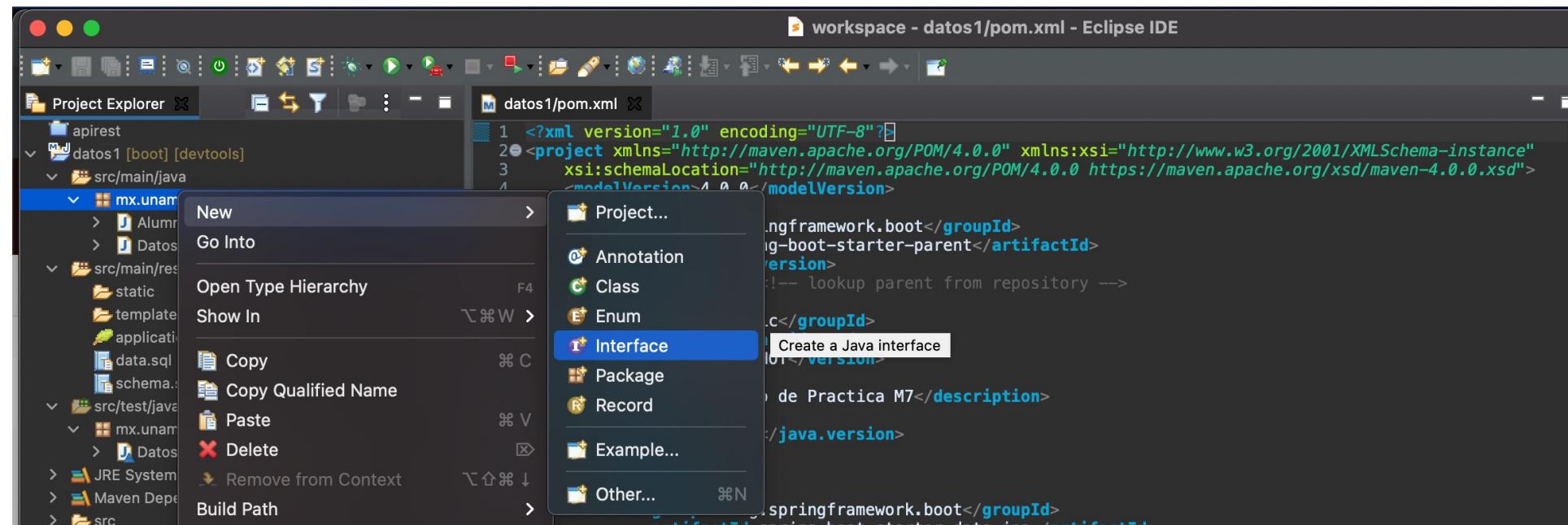
CrudRepository

```
public interface CrudRepository<T, ID> extends  
Repository<T, ID> {  
    <S extends T> S save(S entity);  
    Optional<T> findById(ID primaryKey);  
    Iterable<T> findAll();  
    long count();  
    void delete(T entity);  
    boolean existsById(ID primaryKey);  
    // ... more functionality omitted.  
}
```

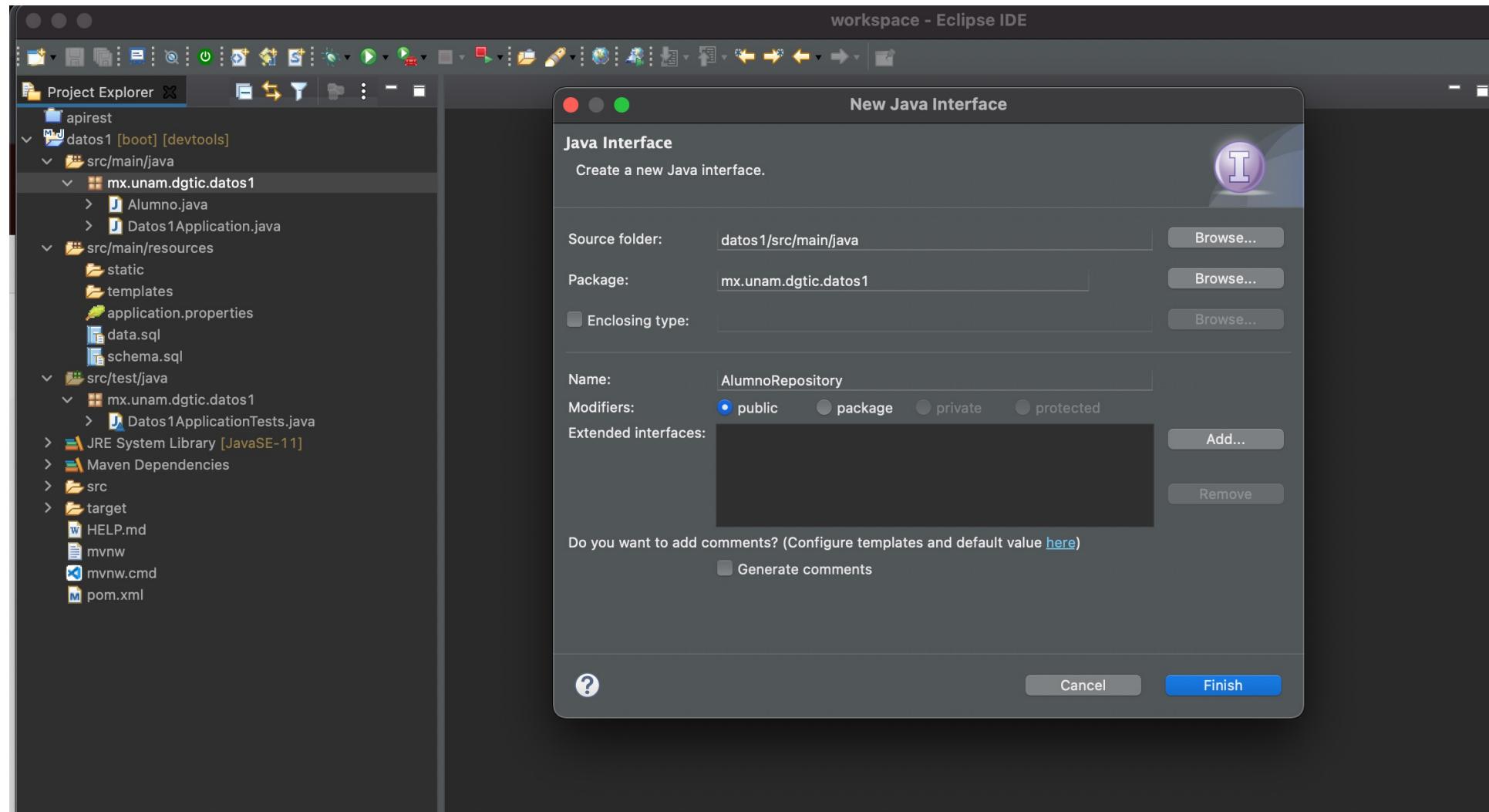
CrudRepository

Modifier and Type	Method and Description
long	<u>count()</u> Returns the number of entities available.
void	<u>delete(T entity)</u> Deletes a given entity.
void	<u>deleteAll()</u> Deletes all entities managed by the repository.
void	<u>deleteAll(Iterable<? extends T> entities)</u> Deletes the given entities.
void	<u>deleteAllById(Iterable<? extends ID> ids)</u> Deletes all instances of the type T with the given IDs.
void	<u>deleteById(ID id)</u> Deletes the entity with the given id.
boolean	<u>existsById(ID id)</u> Returns whether an entity with the given id exists.
<u>Iterable<T></u>	<u>findAll()</u> Returns all instances of the type.
<u>Iterable<T></u>	<u>findAllById(Iterable<ID> ids)</u> Returns all instances of the type T with the given IDs.
<u>Optional<T></u>	<u>findById(ID id)</u> Retrieves an entity by its id.
<S extends <u>T</u> S	<u>save(S entity)</u> Saves a given entity.
<S extends <u>T</u> <u>Iterable<S></u>	<u>saveAll(Iterable<S> entities)</u> Saves all given entities.

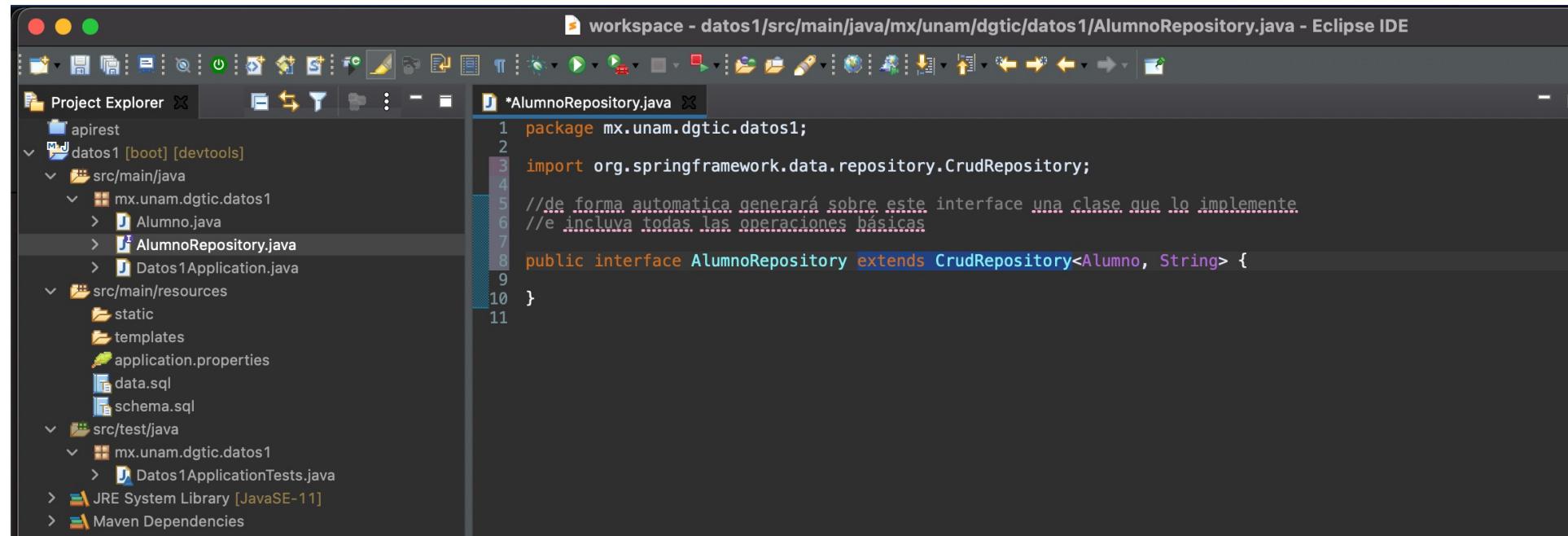
Agregar AlumnoRepository



Agregar AlumnoRepository



extends CrudRepository



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace - datos1/src/main/java/mx/unam/dgtic/datos1/AlumnoRepository.java - Eclipse IDE
- Toolbar:** Standard Eclipse toolbar with various icons for file operations, search, and project management.
- Project Explorer:** Shows the project structure:
 - apirest
 - datos1 [boot] [devtools]
 - src/main/java
 - mx.unam.dgtic.datos1
 - Alumno.java
 - AlumnoRepository.java
 - Datos1Application.java
 - src/main/resources
 - static
 - templates
 - application.properties
 - data.sql
 - schema.sql
 - src/test/java
 - mx.unam.dgtic.datos1
 - Datos1ApplicationTests.java
 - JRE System Library [JavaSE-11]
 - Maven Dependencies
 - Code Editor:** Displays the AlumnoRepository.java code:

```
1 package mx.unam.dgtic.datos1;
2
3 import org.springframework.data.repository.CrudRepository;
4
5 //de forma automatica generará sobre este interface una clase que lo implemente
6 //e incluya todas las operaciones básicas
7
8 public interface AlumnoRepository extends CrudRepository<Alumno, String> {
9
10 }
```

@Autowired

The screenshot shows the Eclipse IDE interface with the title bar "workspace - datos1/src/test/java/mx/unam/dgtic/datos1/Datos1ApplicationTests.java - Eclipse IDE". The left pane is the "Project Explorer" showing the project structure:

- apirest
- datos1 [boot] [devtools]
 - src/main/java
 - mx.unam.dgtic.datos1
 - Alumno.java
 - AlumnoRepository.java
 - Datos1Application.java
 - src/main/resources
 - static
 - templates
 - application.properties
 - data.sql
 - schema.sql
 - src/test/java
 - mx.unam.dgtic.datos1
 - Datos1ApplicationTests.java- JRE System Library [JavaSE-11]
- Maven Dependencies
- src
- target
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml

The right pane displays the Java code for `Datos1ApplicationTests.java`:

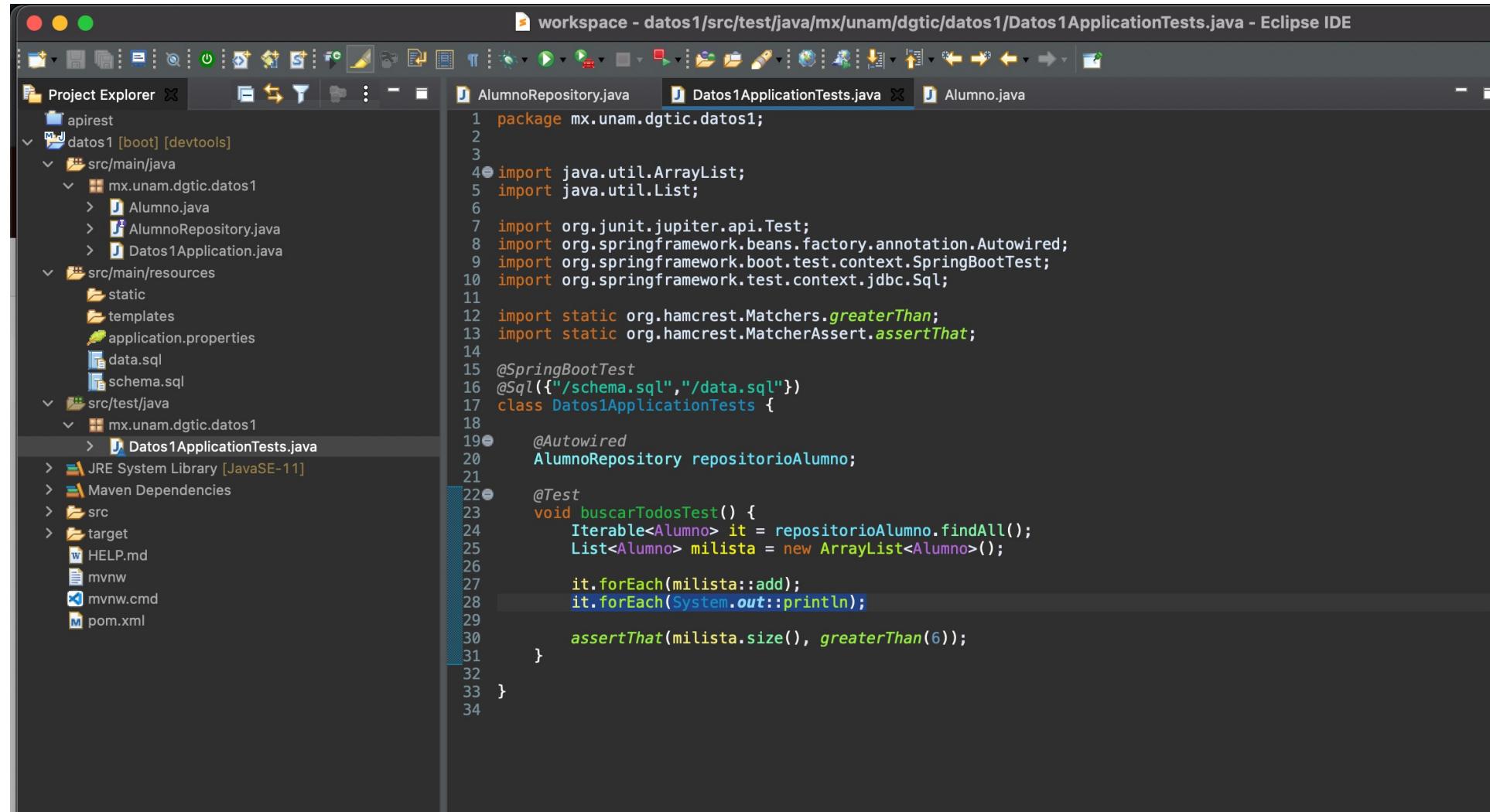
```
1 package mx.unam.dgtic.datos1;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.test.context.SpringBootTest;
6 import org.springframework.test.context.jdbc.Sql;
7
8 @SpringBootTest
9 @Sql({"/schema.sql", "/data.sql"})
10 class Datos1ApplicationTests {
11
12     @Autowired
13     AlumnoRepository repositorioAlumno;
14
15     @Test
16     void buscarTodosTest() {
17         repositorioAlumno.
18     }
19
20 }
21
```

A code completion tooltip is open at line 17, showing suggestions for the `repositorioAlumno.` part of the code. The tooltip includes:

- `count() : long - CrudRepository`
- `delete(Alumno entity) : void - CrudRepository`
- `deleteAll() : void - CrudRepository`
- `deleteAll(Iterable<? extends Alumno> entities) : void - CrudReposito...`
- `deleteAllById(Iterable<? extends String> ids) : void - CrudReposito...`
- `deleteById(String id) : void - CrudRepository`
- `equals(Object obj) : boolean - Object`
- `existsById(String id) : boolean - CrudRepository`
- `findAll() : Iterable<Alumno> - CrudRepository`
- `findAllById(Iterable<String> ids) : Iterable<Alumno> - CrudReposito...`
- `findById(String id) : Optional<Alumno> - CrudRepository`

The tooltip also contains the text "Returns the number of entities available." and "Returns: the number of entities."

Probar findAll()



The screenshot shows the Eclipse IDE interface with the following details:

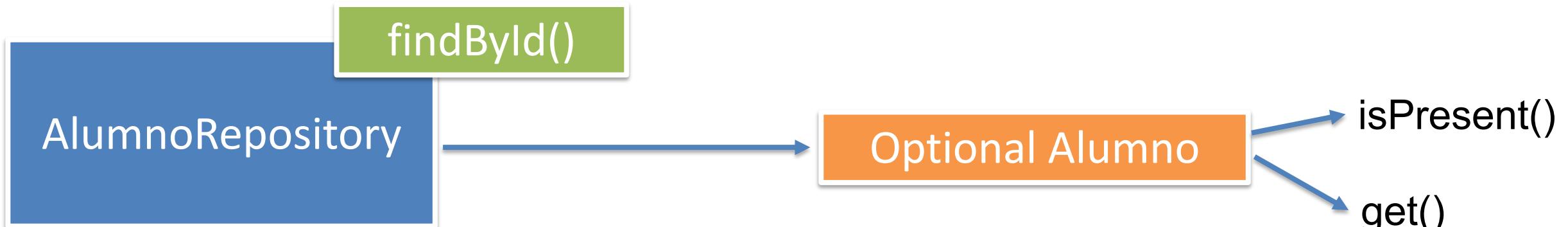
- Title Bar:** workspace - datos1/src/test/java/mx/unam/dgtic/datos1/Datos1ApplicationTests.java - Eclipse IDE
- Project Explorer:** Shows the project structure:
 - apirest
 - datos1 [boot] [devtools]
 - src/main/java
 - mx.unam.dgtic.datos1
 - Alumno.java
 - AlumnoRepository.java
 - Datos1Application.java
 - src/main/resources
 - static
 - templates
 - application.properties
 - data.sql
 - schema.sql
 - src/test/java
 - mx.unam.dgtic.datos1
 - Datos1ApplicationTests.java
 - JRE System Library [JavaSE-11]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
 - Editor:** Displays the code for `Datos1ApplicationTests.java`. The code is a Spring Boot test class that uses JUnit Jupiter and Hamcrest assertions to verify the `findAll()` method of the `AlumnoRepository`.

```
1 package mx.unam.dgtic.datos1;
2
3
4 import java.util.ArrayList;
5 import java.util.List;
6
7 import org.junit.jupiter.api.Test;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.boot.test.context.SpringBootTest;
10 import org.springframework.test.context.jdbc.Sql;
11
12 import static org.hamcrest.Matchers.greaterThan;
13 import static org.hamcrest.MatcherAssert.assertThat;
14
15 @SpringBootTest
16 @Sql({"/schema.sql", "/data.sql"})
17 class Datos1ApplicationTests {
18
19     @Autowired
20     AlumnoRepository repositorioAlumno;
21
22     @Test
23     void buscarTodosTest() {
24         Iterable<Alumno> it = repositorioAlumno.findAll();
25         List<Alumno> milista = new ArrayList<Alumno>();
26
27         it.forEach(milista::add);
28         it.forEach(System.out::println);
29
30         assertThat(milista.size(), greaterThan(6));
31     }
32
33 }
34 }
```

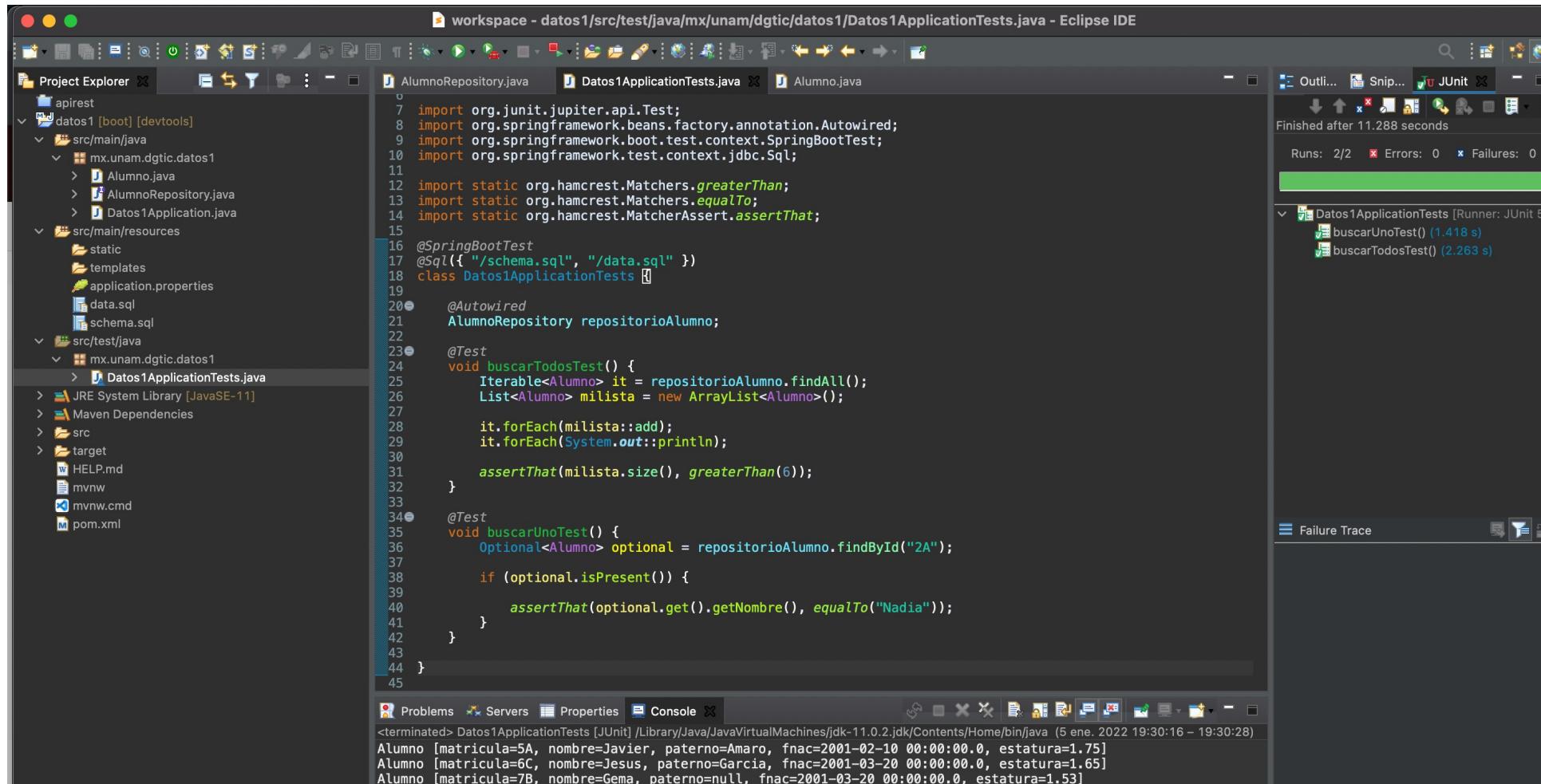
Probar findAll()

```
Alumno [matricula=1A, nombre=Juan, paterno=Lopez, fnac=2001-01-01 00:00:00.0, estatura=1.78]
Alumno [matricula=2A, nombre=Nadia, paterno=Perez, fnac=2001-01-10 00:00:00.0, estatura=1.56]
Alumno [matricula=3B, nombre=Perla, paterno=Calles, fnac=2001-01-20 00:00:00.0, estatura=1.6]
Alumno [matricula=4A, nombre=Carlos, paterno=Madero, fnac=2001-01-01 00:00:00.0, estatura=1.68]
Alumno [matricula=5A, nombre=Javier, paterno=Amaro, fnac=2001-02-10 00:00:00.0, estatura=1.75]
Alumno [matricula=6C, nombre=Jesus, paterno=Garcia, fnac=2001-03-20 00:00:00.0, estatura=1.65]
Alumno [matricula=7B, nombre=Gema, paterno=null, fnac=2001-03-20 00:00:00.0, estatura=1.53]
```

Probar findById()



Probar findById()



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Displays the project structure under "workspace - datos1/src/test/java/mx/unam/dgtic/datos1".
- Editor:** Shows the code for `Datos1ApplicationTests.java`. The code uses JUnit Jupiter and Spring Boot Test annotations to test the `AlumnoRepository`. It includes methods for testing all students and finding a specific student by ID.
- Console:** Displays the command-line output of the test run, showing three student records from the database.
- Output View:** Shows the test results: "Finished after 11.288 seconds" with 2 runs, 0 errors, and 0 failures.

```
7 import org.junit.jupiter.api.Test;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.boot.test.context.SpringBootTest;
10 import org.springframework.test.context.jdbc.Sql;
11
12 import static org.hamcrest.Matchers.greaterThan;
13 import static org.hamcrest.Matchers.equalTo;
14 import static org.hamcrest.MatcherAssert.assertThat;
15
16 @SpringBootTest
17 @Sql({"/schema.sql", "/data.sql"})
18 class Datos1ApplicationTests {
19
20    @Autowired
21    AlumnoRepository repositorioAlumno;
22
23    @Test
24    void buscarTodosTest() {
25        Iterable<Alumno> it = repositorioAlumno.findAll();
26        List<Alumno> milista = new ArrayList<Alumno>();
27
28        it.forEach(milista::add);
29        it.forEach(System.out::println);
30
31        assertThat(milista.size(), greaterThan(6));
32    }
33
34    @Test
35    void buscarUnoTest() {
36        Optional<Alumno> optional = repositorioAlumno.findById("2A");
37
38        if (optional.isPresent()) {
39
40            assertThat(optional.get().getNombre(), equalTo("Nadia"));
41        }
42    }
43
44 }
```

```
<terminated> Datos1ApplicationTests [JUnit] /Library/Java/JavaVirtualMachines/jdk-11.0.2.jdk/Contents/Home/bin/java (5ene.2022 19:30:16 - 19:30:28)
Alumno [matricula=5A, nombre=Javier, paterno=Amaro, fnac=2001-02-10 00:00:00.0, estatura=1.75]
Alumno [matricula=6C, nombre=Jesus, paterno=Garcia, fnac=2001-03-20 00:00:00.0, estatura=1.65]
Alumno [matricula=7B, nombre=Gema, paterno=null, fnac=2001-03-20 00:00:00.0, estatura=1.53]
```

Contacto

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx

