11ª 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

DGTIC

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.
  - JPARepository: agregá tareas específicas a las anteriores, como flush.

Spring Data Repositories

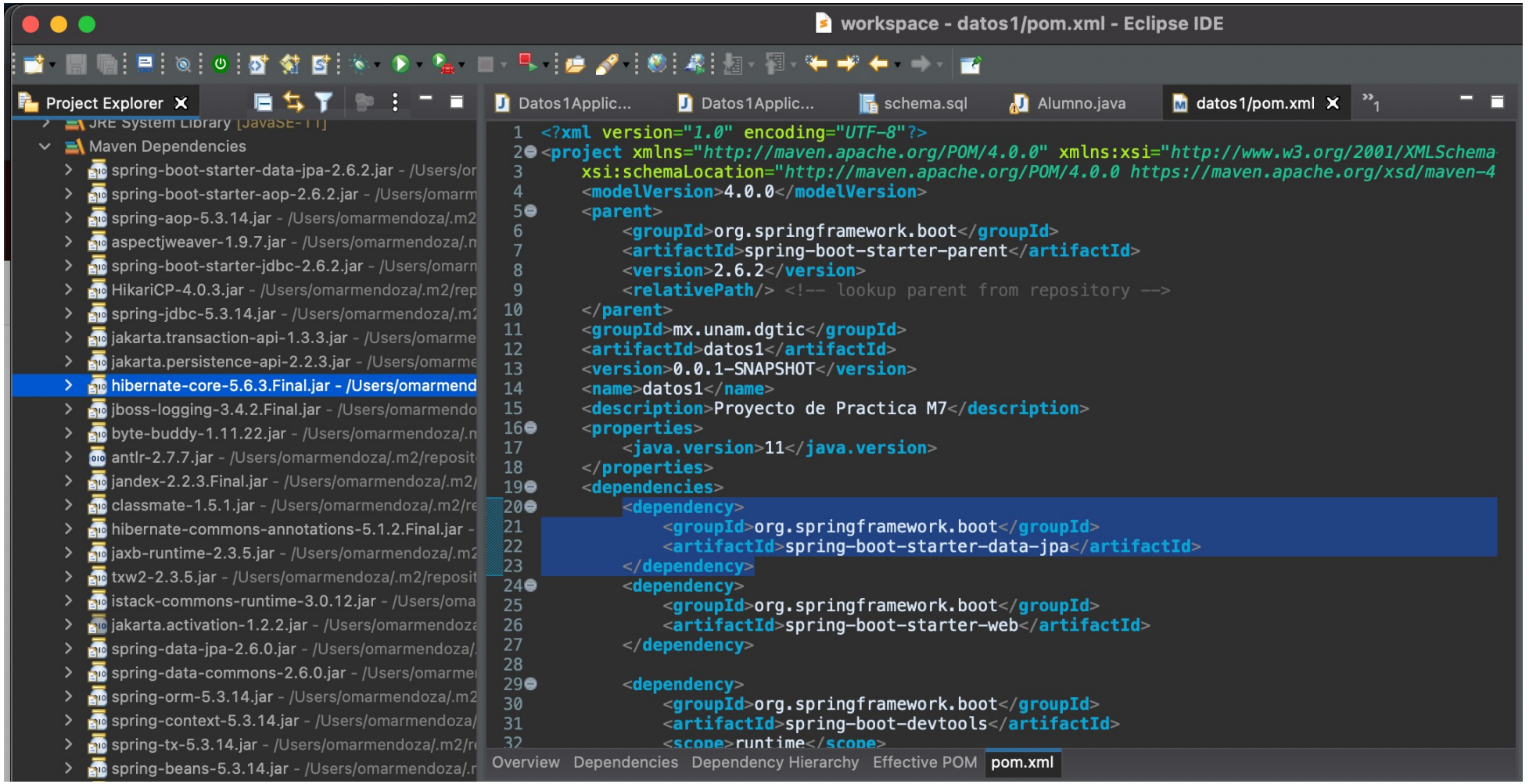
- A la clase que interactúa con la base de datos mediante una instancia de un repositorio, se le debe agregar alguna de las siguientes anotaciones:
  - @Repository
    - le indica a la clase que es la encarga de interactuar con la bd.
  - @Component
    - le indica que es un componente de spring.

Query Methods

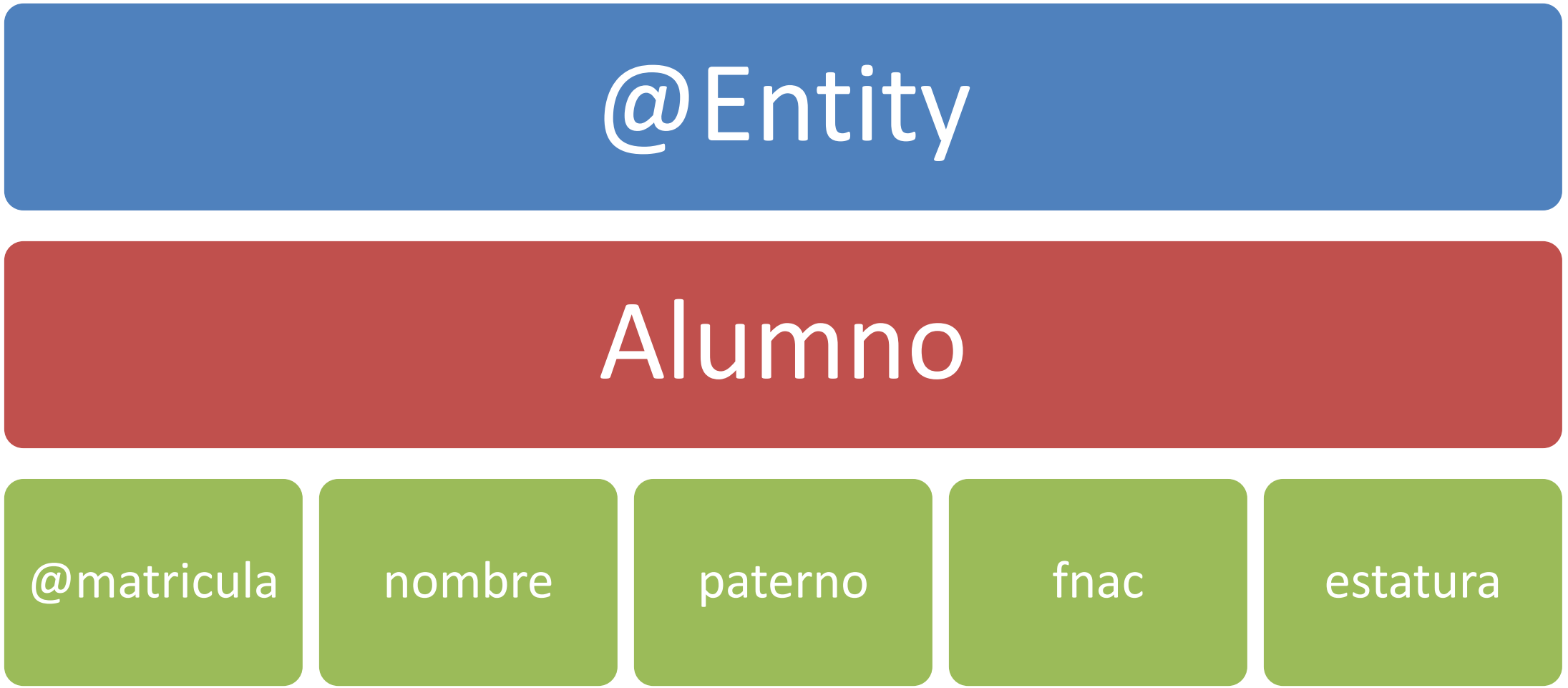
- Cuando se necesitan consultas y el repositorio de Spring Data no puede hacer estas consultas, entonces los Query Method proveen la posibilidad de generar consultas mediante el nombre de los métodos.

- 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).

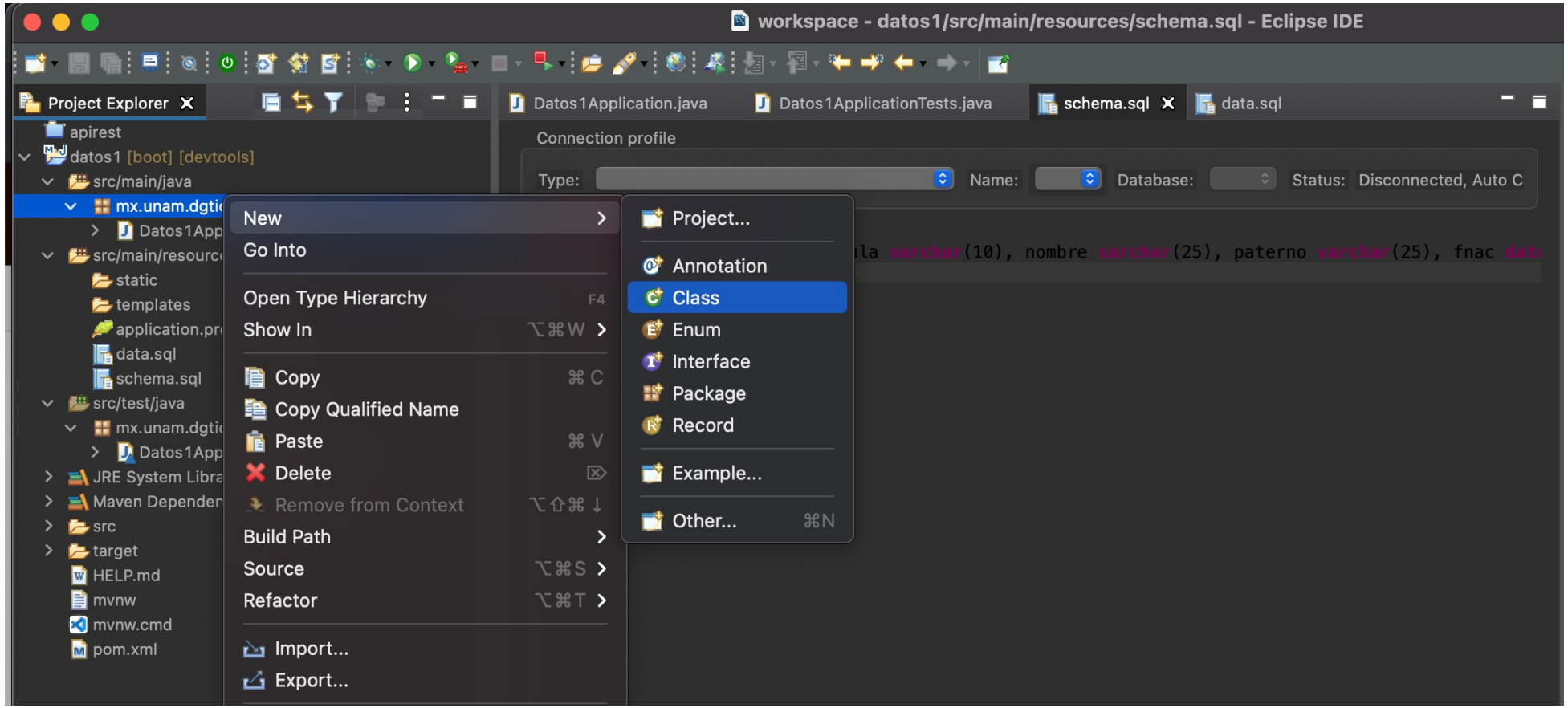
Visualizar dependencias



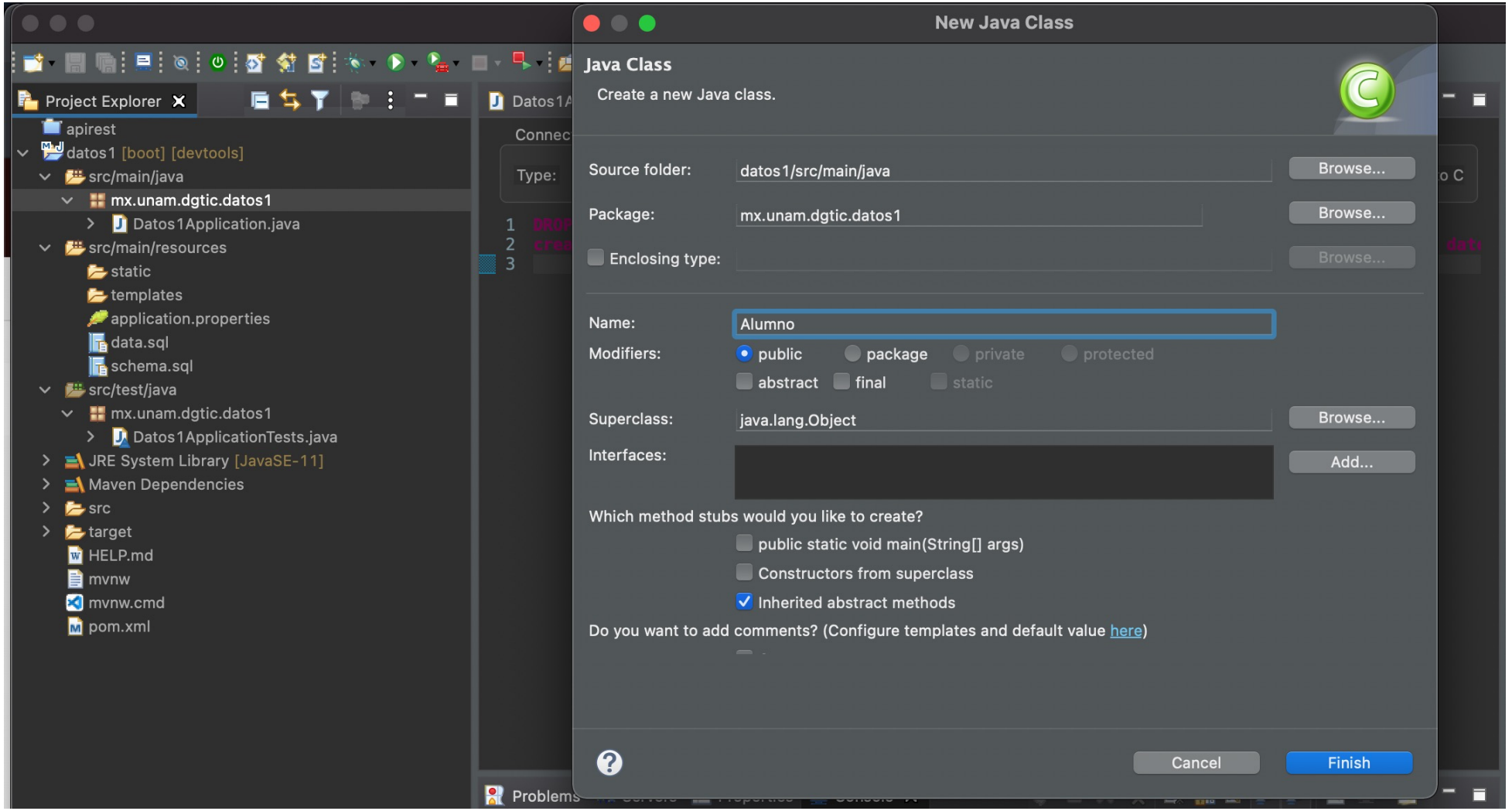
Entidades



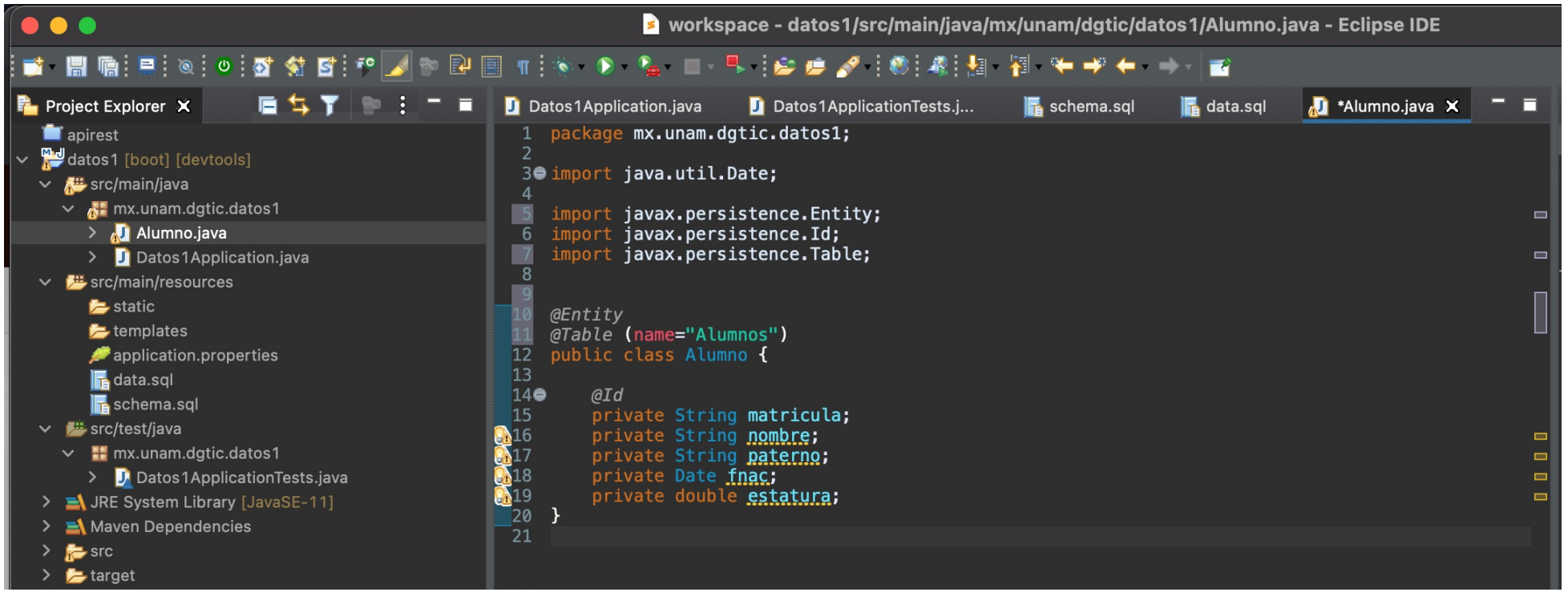
Generar una entidad



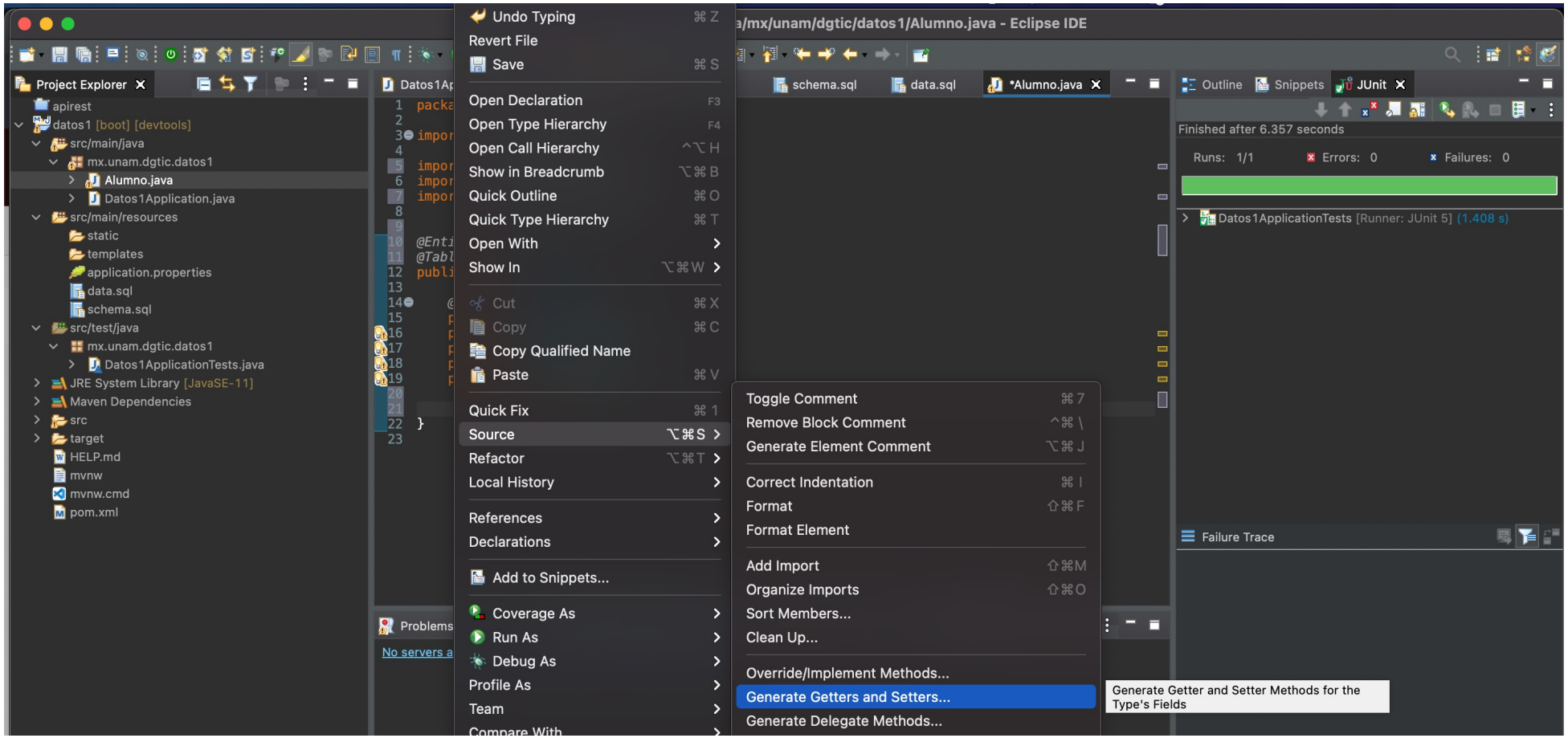
Generar una entidad



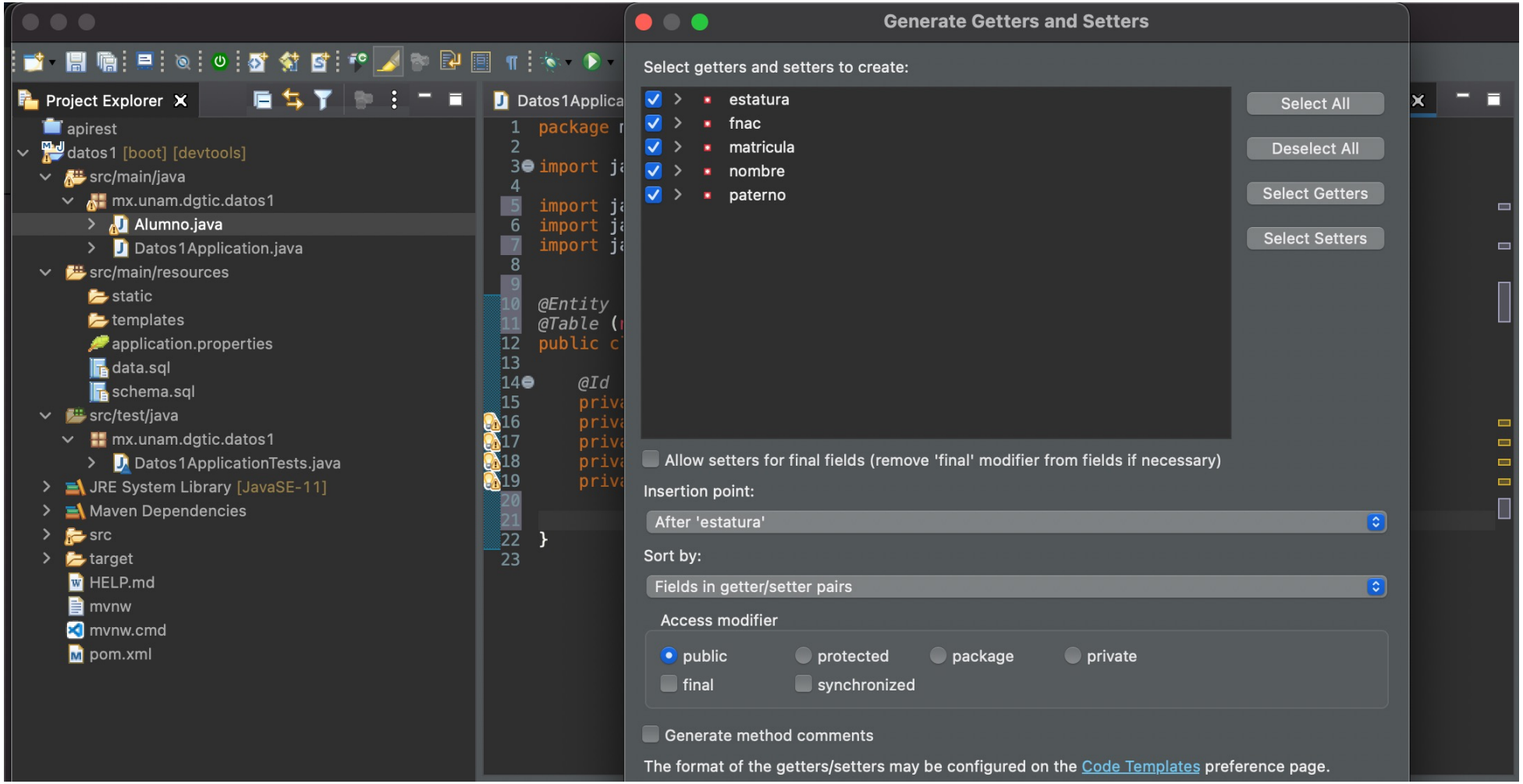
@Entity @Table



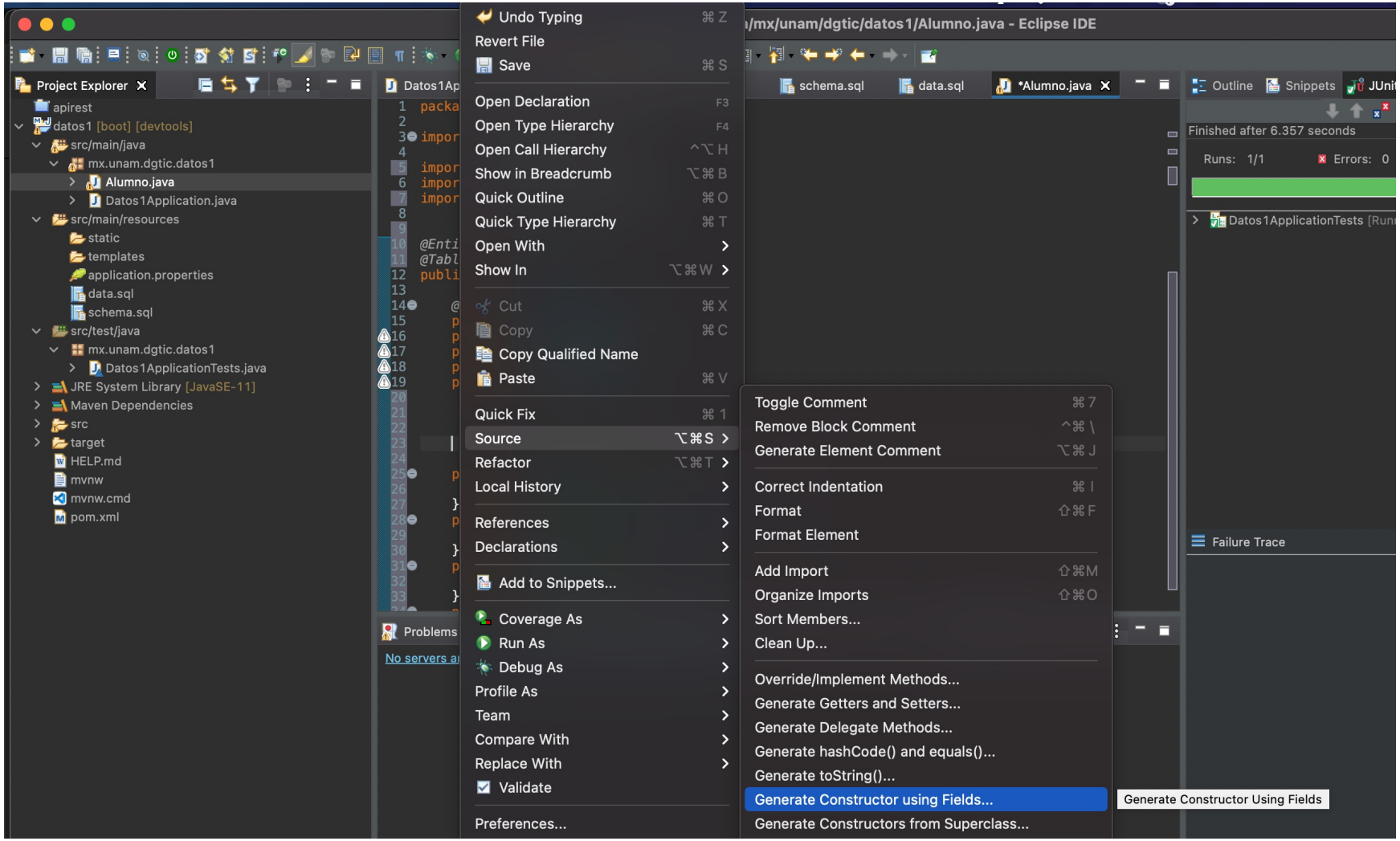
Generar Setters y Getters



Generar Setters y Getters



Generar constructores

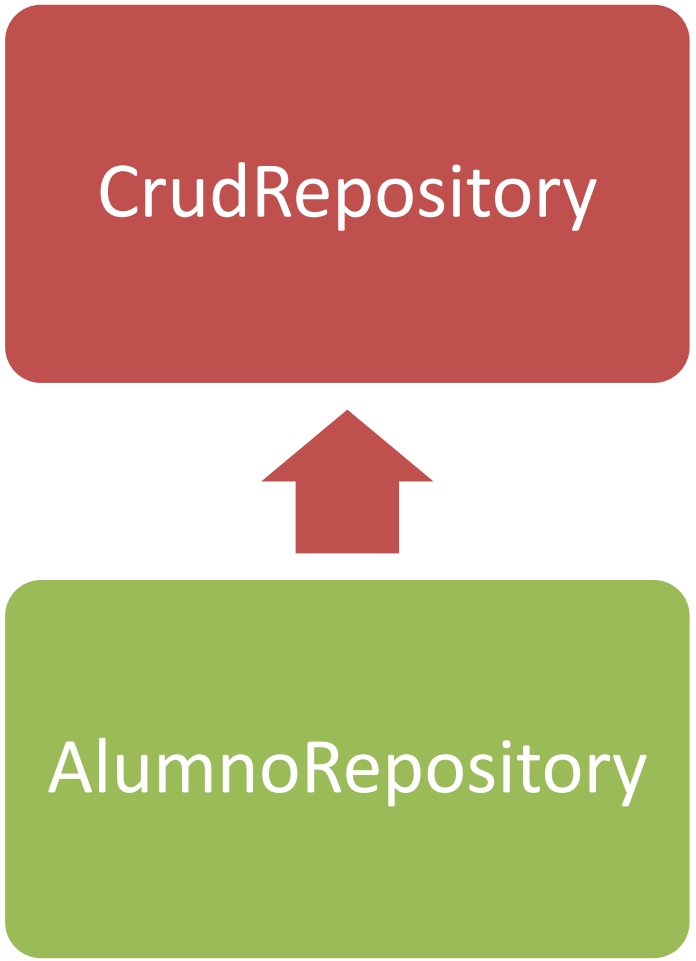


Generar constructores

```
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

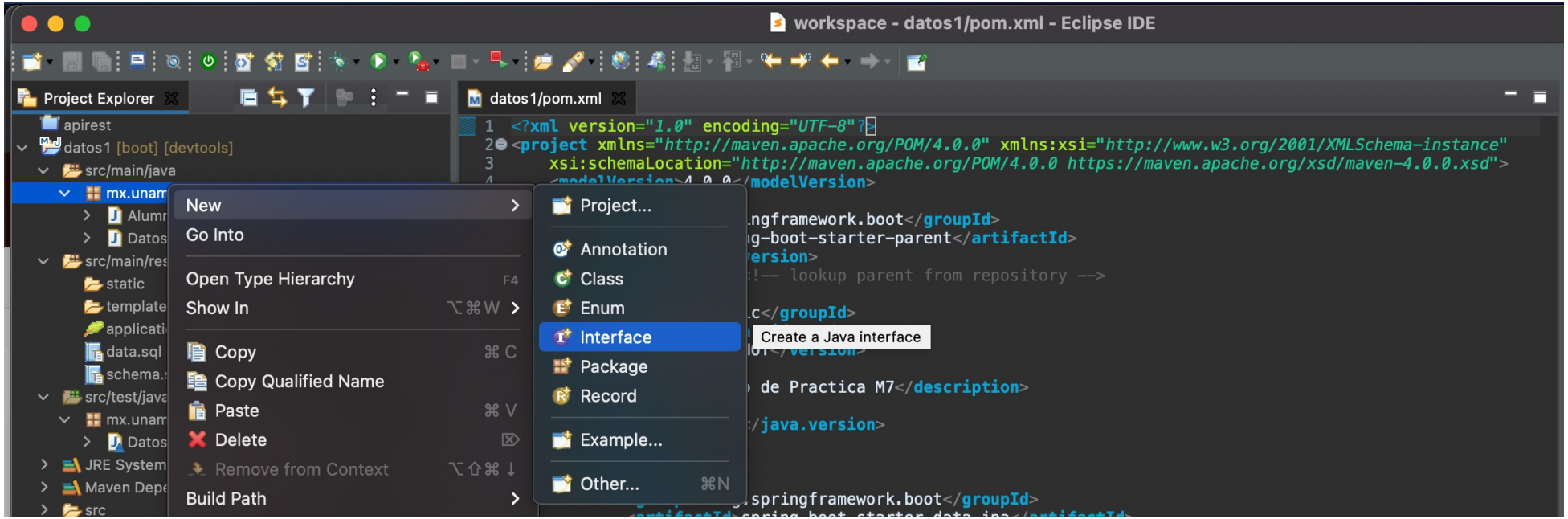
CrudRepository

```
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.
}
```

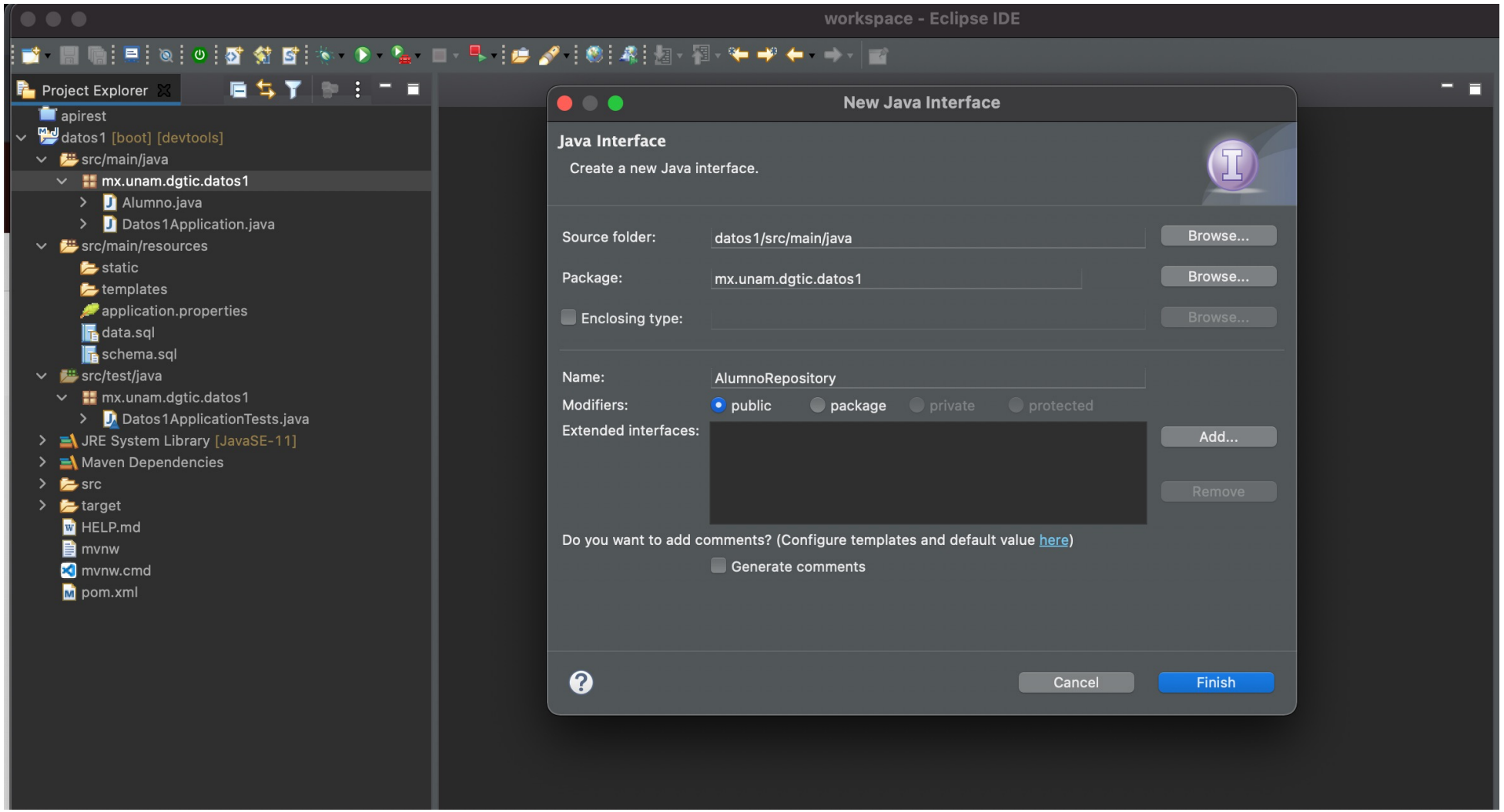
CrudRepository

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

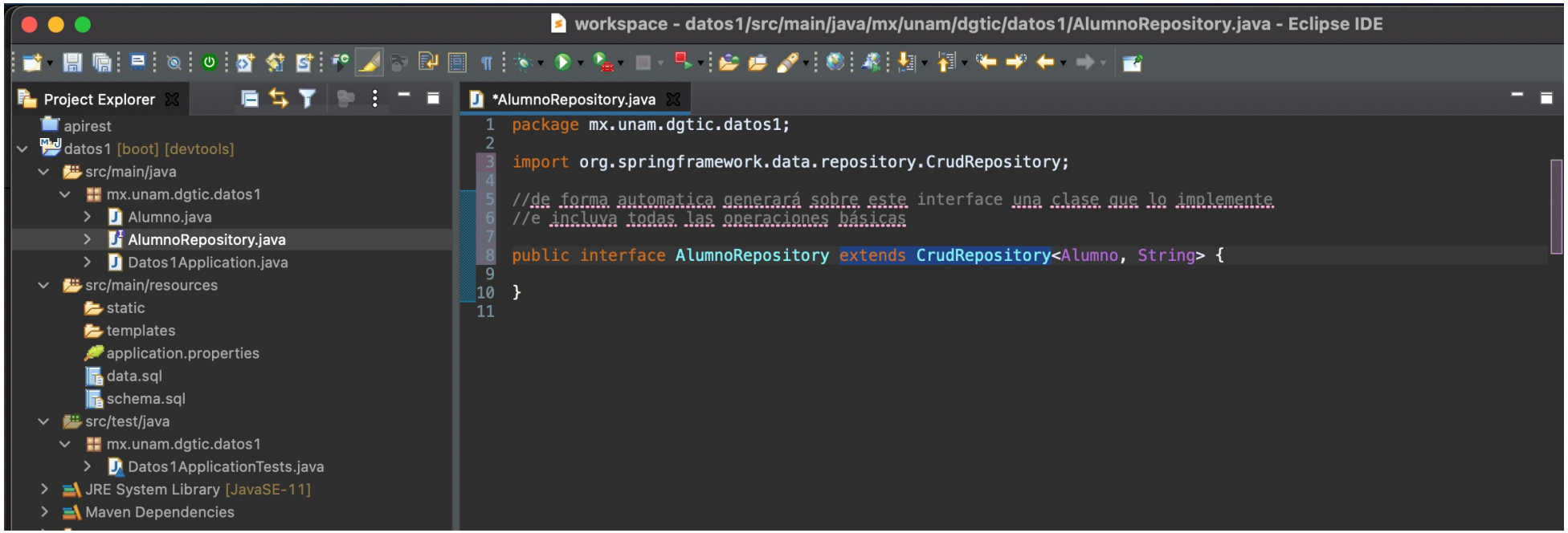
Agregar AlumnoRepository



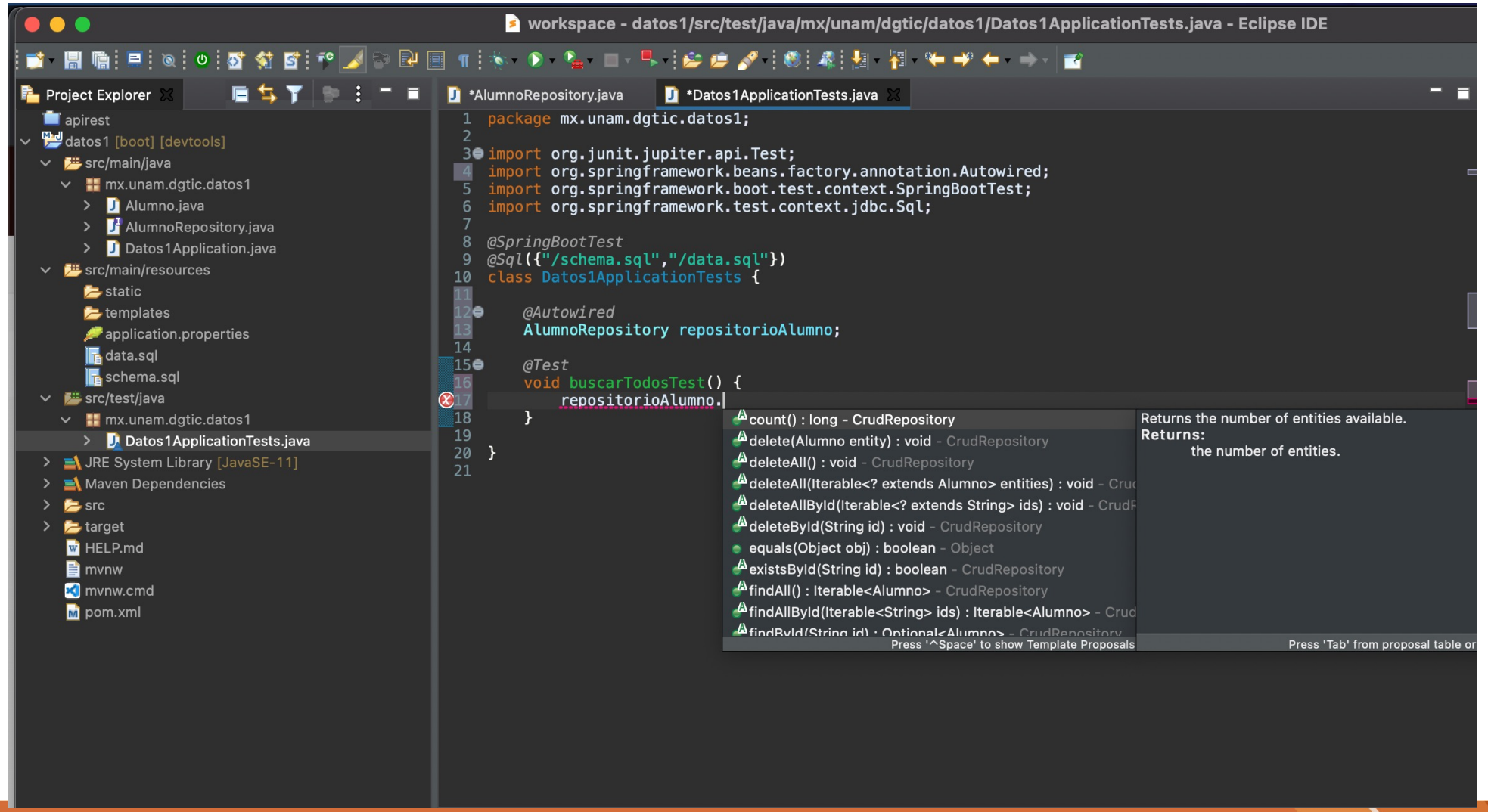
Agregar AlumnoRepository



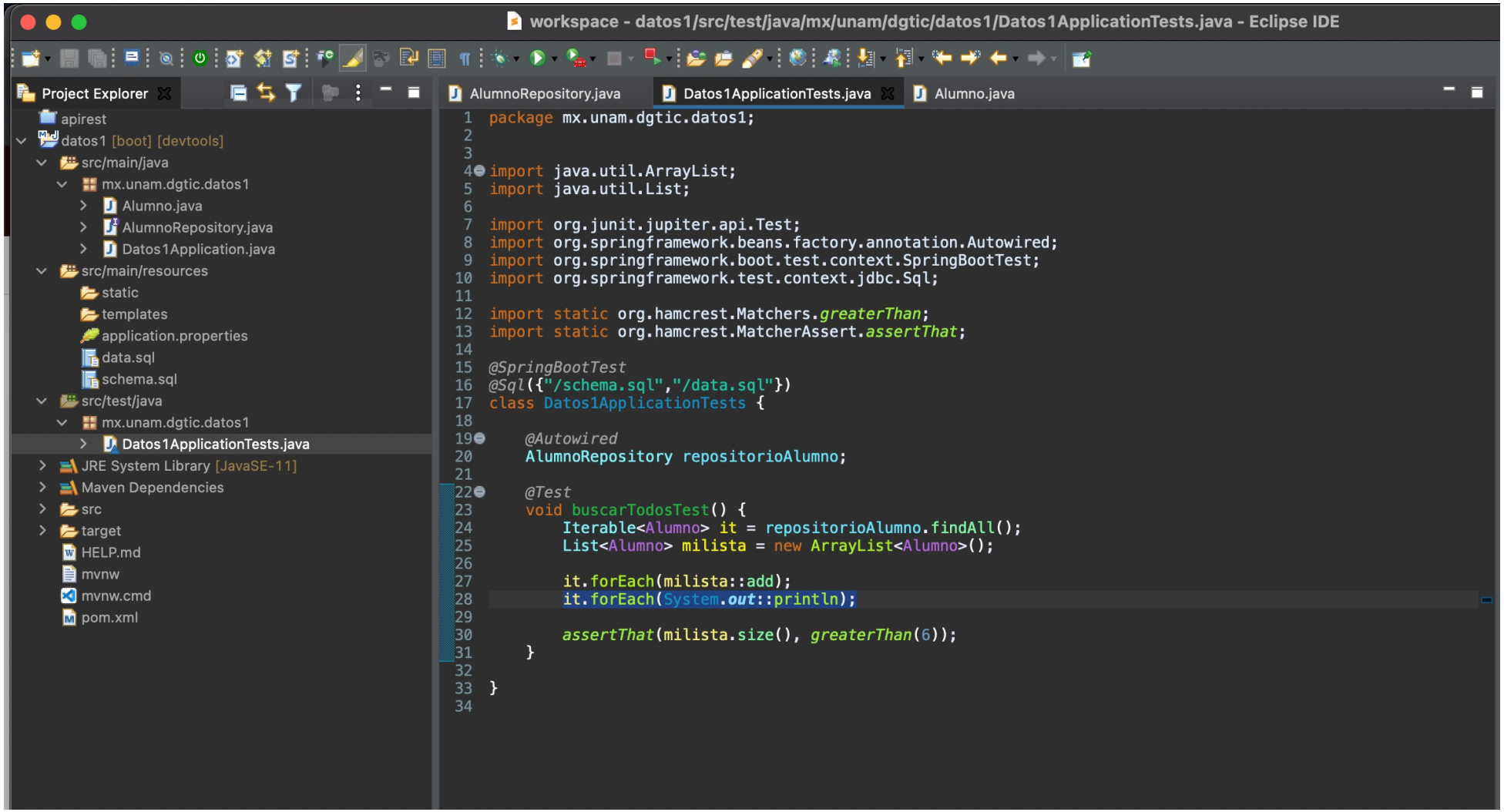
extends CrudRepository



@Autowired


workspace - datos1/src/test/java/mx/unam/dgtic/datos1/Datos1ApplicationTests.java - Eclipse IDE
Project Explorer
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
*AlumnoRepository.java
*Datos1ApplicationTests.java
package mx.unam.dgtic.datos1;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.jdbc.Sql;
@SpringBootTest
@Sql({"/schema.sql","/data.sql"})
class Datos1ApplicationTests {
@Autowired
AlumnoRepository repositorioAlumno;
@Test
void buscarTodosTest() {
repositorioAlumno.
}
}
count() : long - CrudRepository
delete(Alumno entity) : void - CrudRepository
deleteAll() : void - CrudRepository
deleteAll(Iterable<? extends Alumno> entities) : void - Crud
deleteAllById(Iterable<? extends String> ids) : void - CrudR
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> - Crud
Returns the number of entities available.
Returns:
the number of entities.
Press '^Space' to show Template Proposals
Press 'Tab' from proposal table or

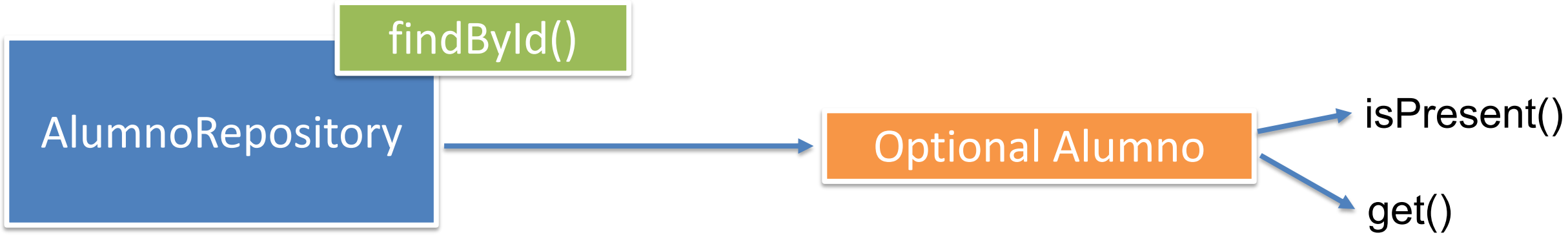
Probar findAll()



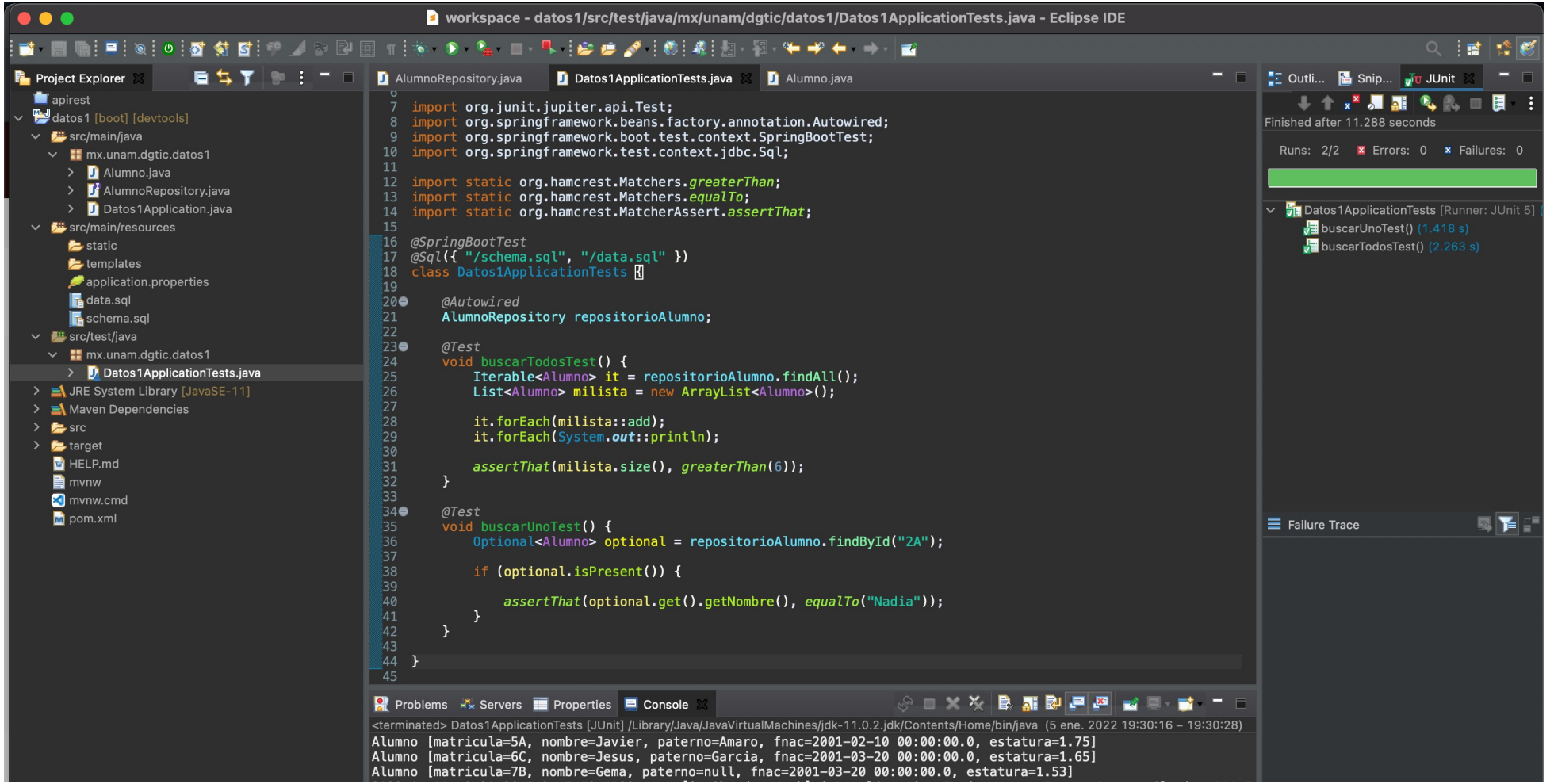
Probar findAll()

```
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()



Contacto

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx