DGTIC UNAM
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

REDEC UNAM

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS
DE INFORMACIÓN Y COMUNICACIÓN

DIPLOMADO

Desarrollo de sistemas con tecnología Java

Módulo 8

Persistencia con Spring Data

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx

Anotaciones de configuración

- Usar **anotaciones** como @SpringBootApplication, @PropertySource, y @EnableJpaRepositories permite configurar de manera declarativa una aplicación Spring Boot sin necesidad de archivos XML o configuraciones manuales detalladas.

- El archivo **application.properties** facilita la centralización de las propiedades de configuración, como los detalles de conexión a la base de datos, la configuración de JPA, y otros parámetros.

Anotaciones de configuración

@SpringBootApplication

- Marca la clase como la principal de la aplicación Spring Boot.
- Combina tres anotaciones: @Configuration, @EnableAutoConfiguration, y @ComponentScan.

@PropertySource("classpath.properties")

- Define la ubicación de un archivo de propiedades externo, como application.properties, que se utiliza para configurar la aplicación.

@EnableTransactionManagement

- Habilita el manejo de transacciones en la aplicación, lo cual es fundamental cuando se utilizan bases de datos y JPA.

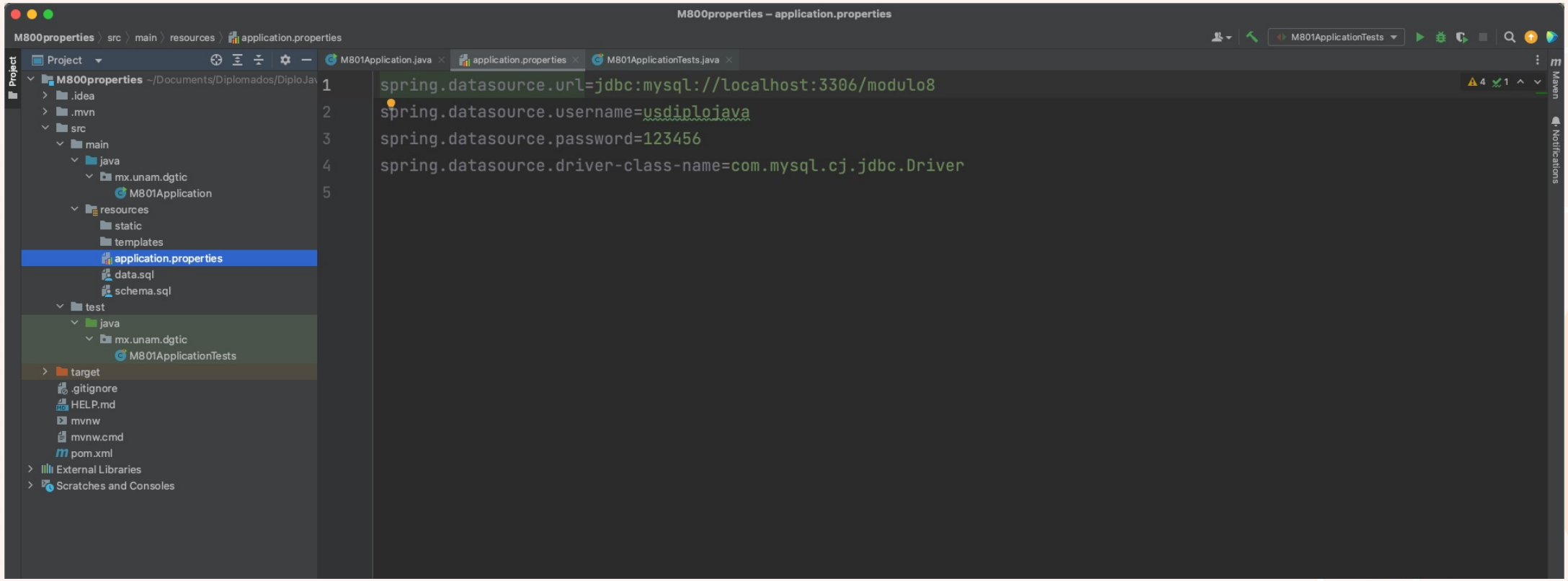
@EnableJpaRepositories(basePackages = "mx.unam.dgtic.datos")

- Activa la creación automática de repositorios JPA y escanea el paquete especificado en busca de interfaces de repositorio.

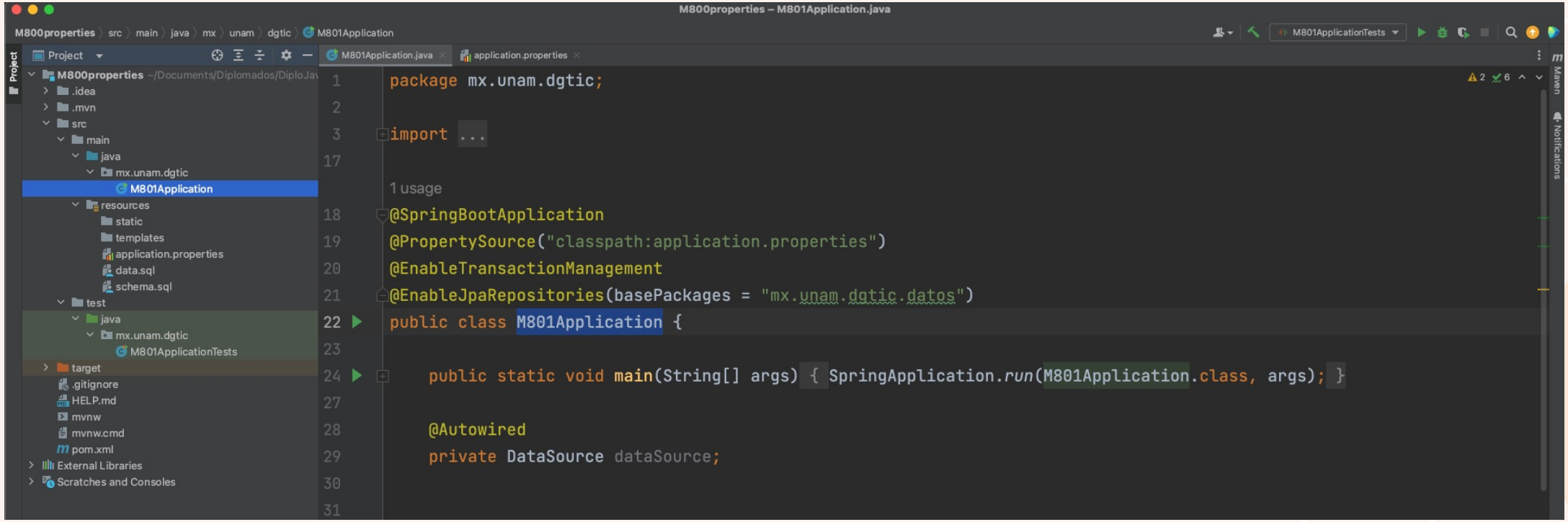
@Autowired

- Inyecta automáticamente dependencias en los componentes de Spring. En este caso, inyecta el DataSource para la conexión a la base de datos.

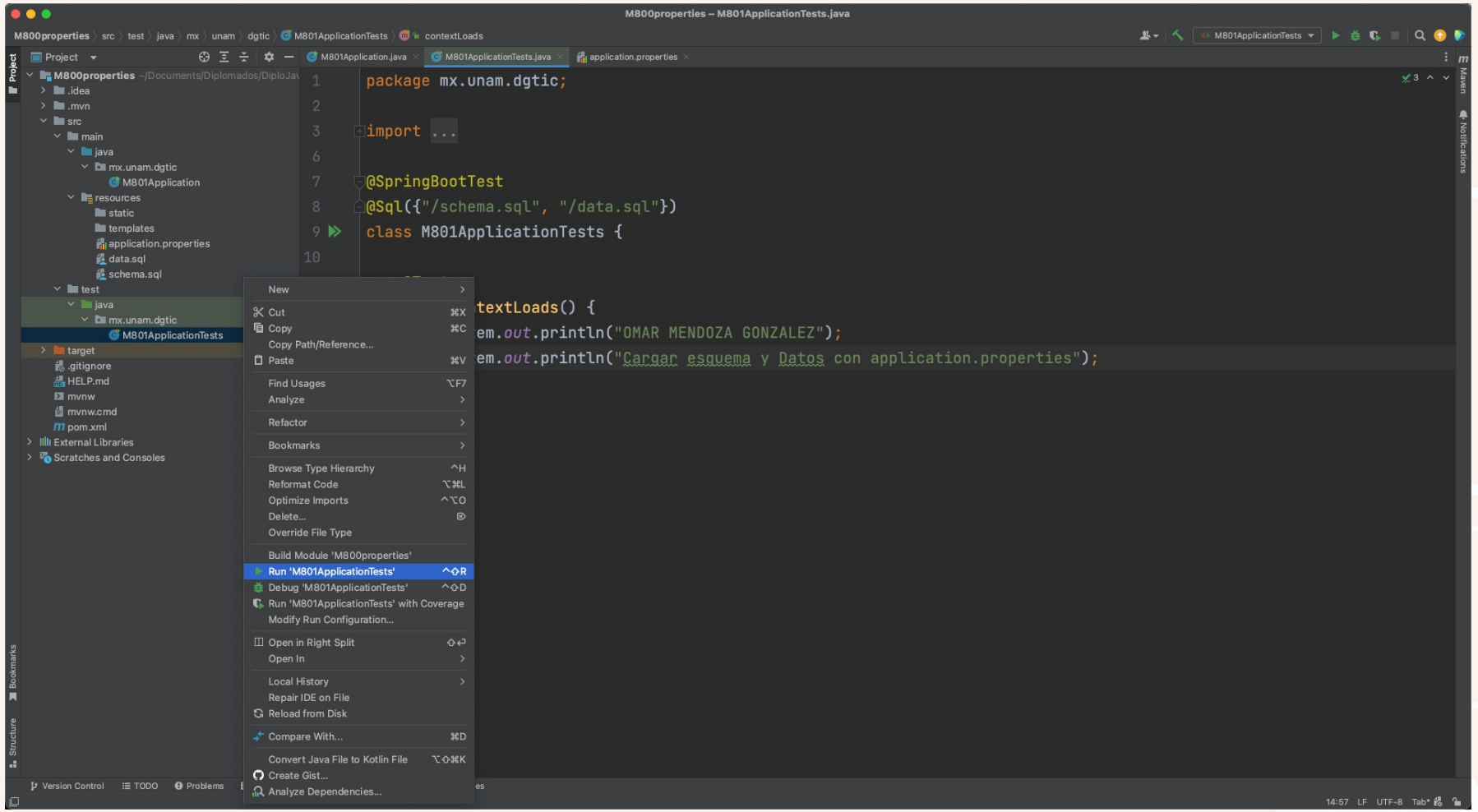
application.properties



M801Application



Ejecutar una UnitTest



JPA

- JPA (Java Persistence API) es una especificación de Java que define un estándar para frameworks ORM.
- Establece las reglas para interactuar con las tablas de una base de datos a través de objetos en Java.
- En otras palabras, JPA es la teoría que describe cómo deben ser las interacciones, algunas implementaciones de esta especificación incluyen
  - **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**
    - Define el nombre de la tabla en la BD a la que la clase está mapeada.
    - Se utiliza cuando el nombre de la tabla es diferente al de 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

- **@Id y @EmbeddedId**
  - Define el atributo que actúa como PK de la tabla dentro de la clase.
    - **@Id**: Se utiliza para llaves primarias simples.
    - **@EmbeddedId**: Se utiliza para llaves primarias compuestas.
- **@GeneratedValue**
  - Permite generar automáticamente valores para atributos PK.
- **@OneToMany y @ManyToOne**

  Establecen relaciones entre entidades/clases.
  - **@OneToMany**: Relación uno a muchos.
  - **@ManyToOne**: Relación muchos a uno.
- **@JoinColumn**
  - Define la columna en la tabla que se usará para unir dos entidades relacionadas.

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

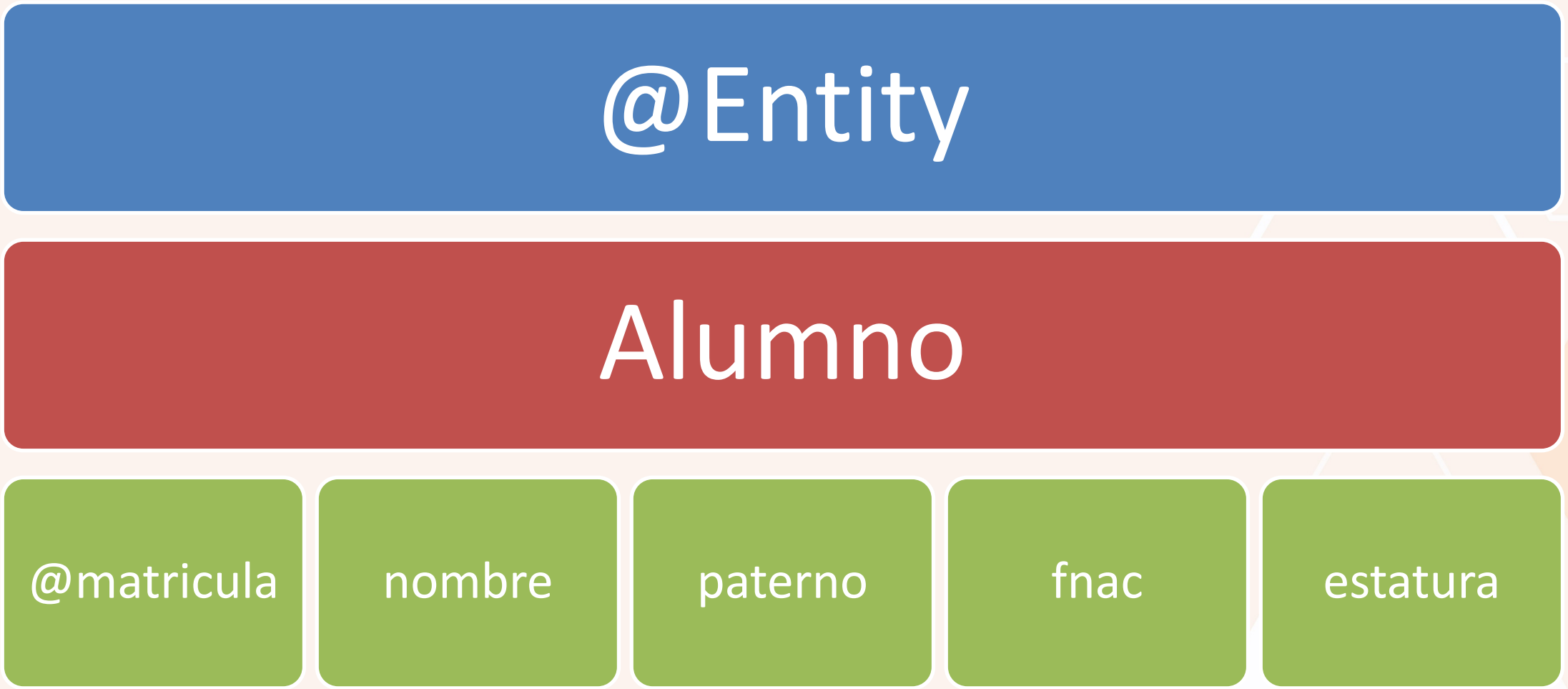
- A la clase que interactúa con la base de datos mediante una instancia de un repositorio, se le debe agregar alguna de las siguientes anotaciones:
  - @Repository
    - le indica a la clase que es la encarga de interactuar con la bd.
  - @Component
    - le indica que es un componente de spring.

Query Methods

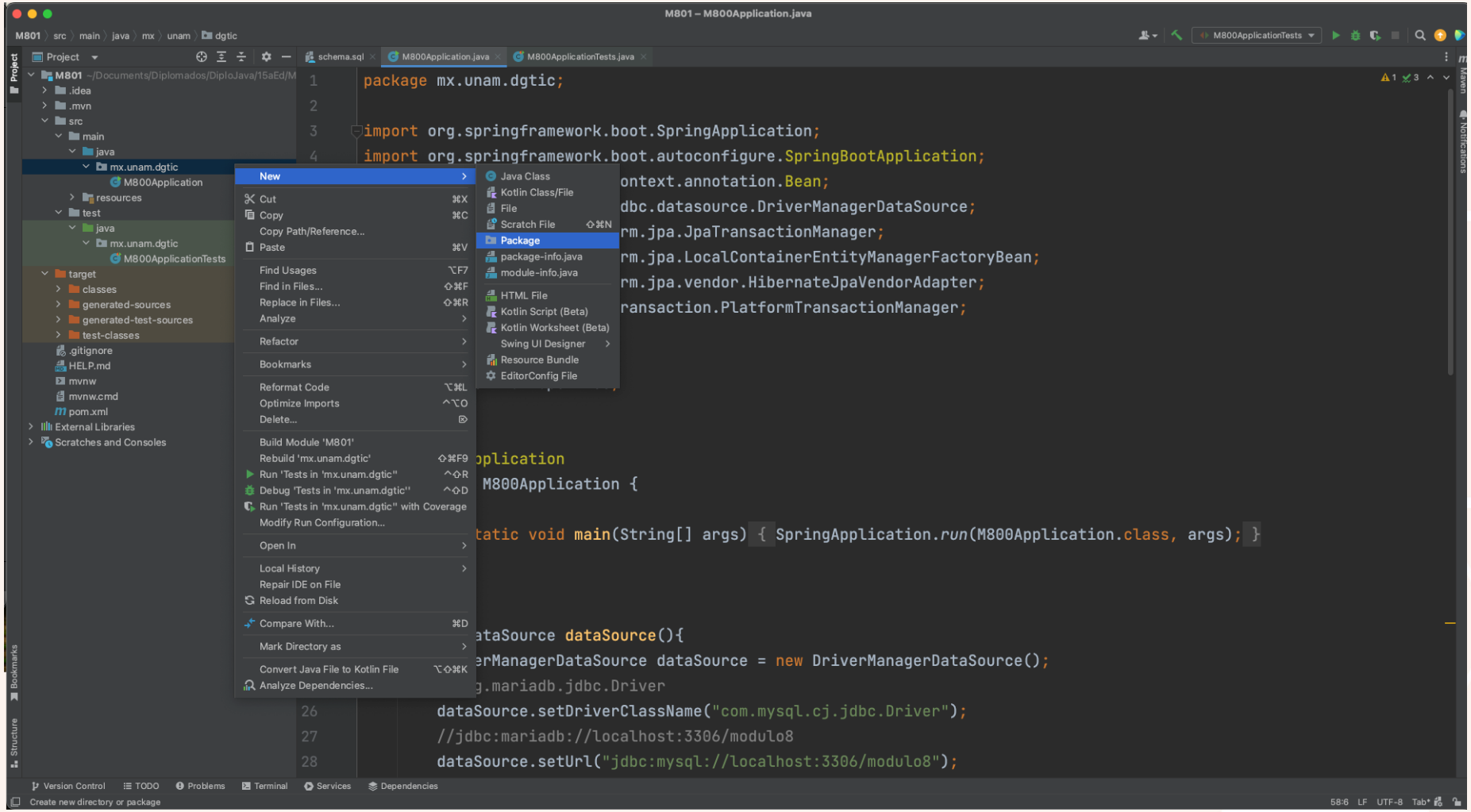
- Cuando se necesitan consultas y el repositorio de Spring Data no puede hacer estas consultas, entonces los Query Method proveen la posibilidad de generar consultas mediante el nombre de los métodos.

- 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).

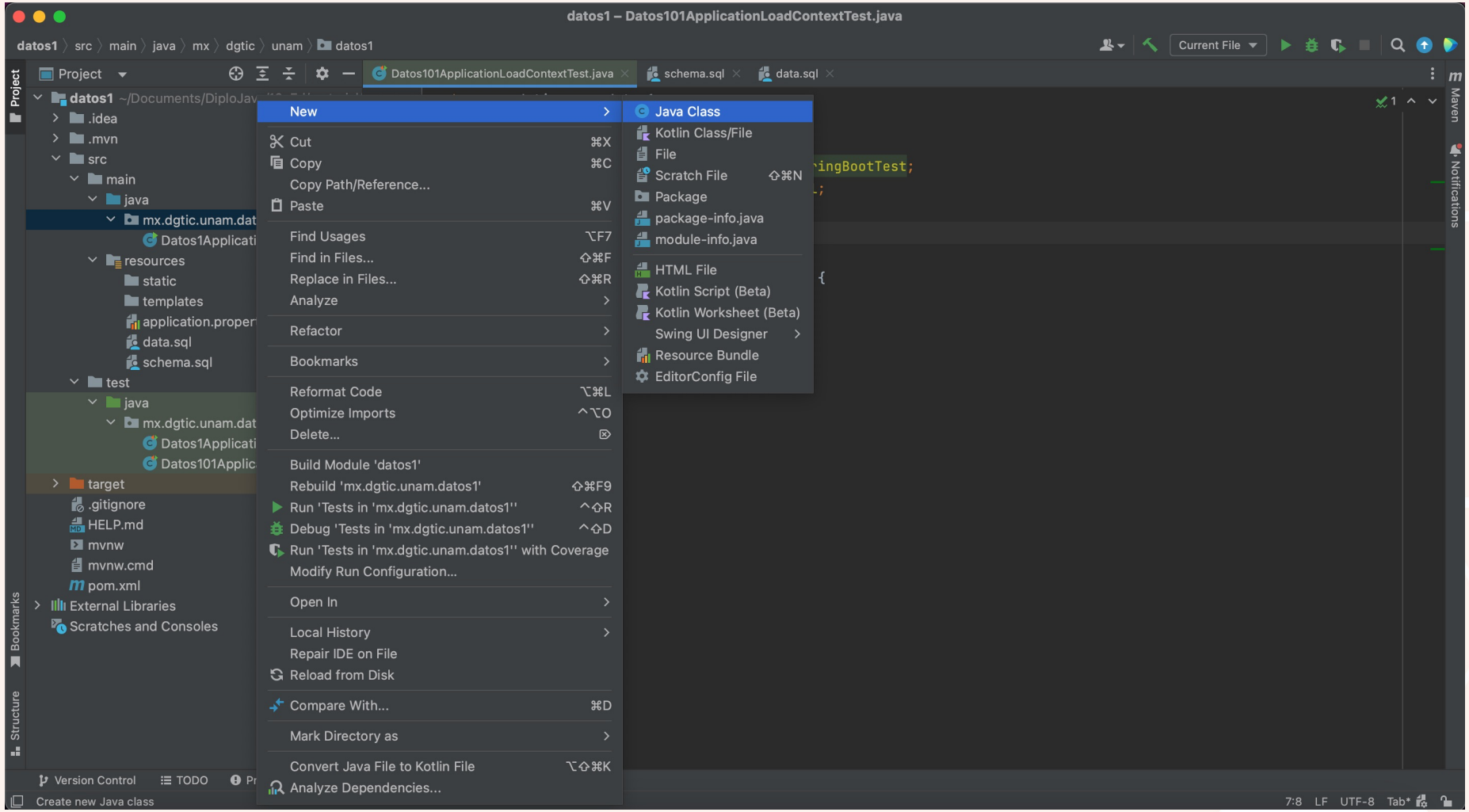
Entidades



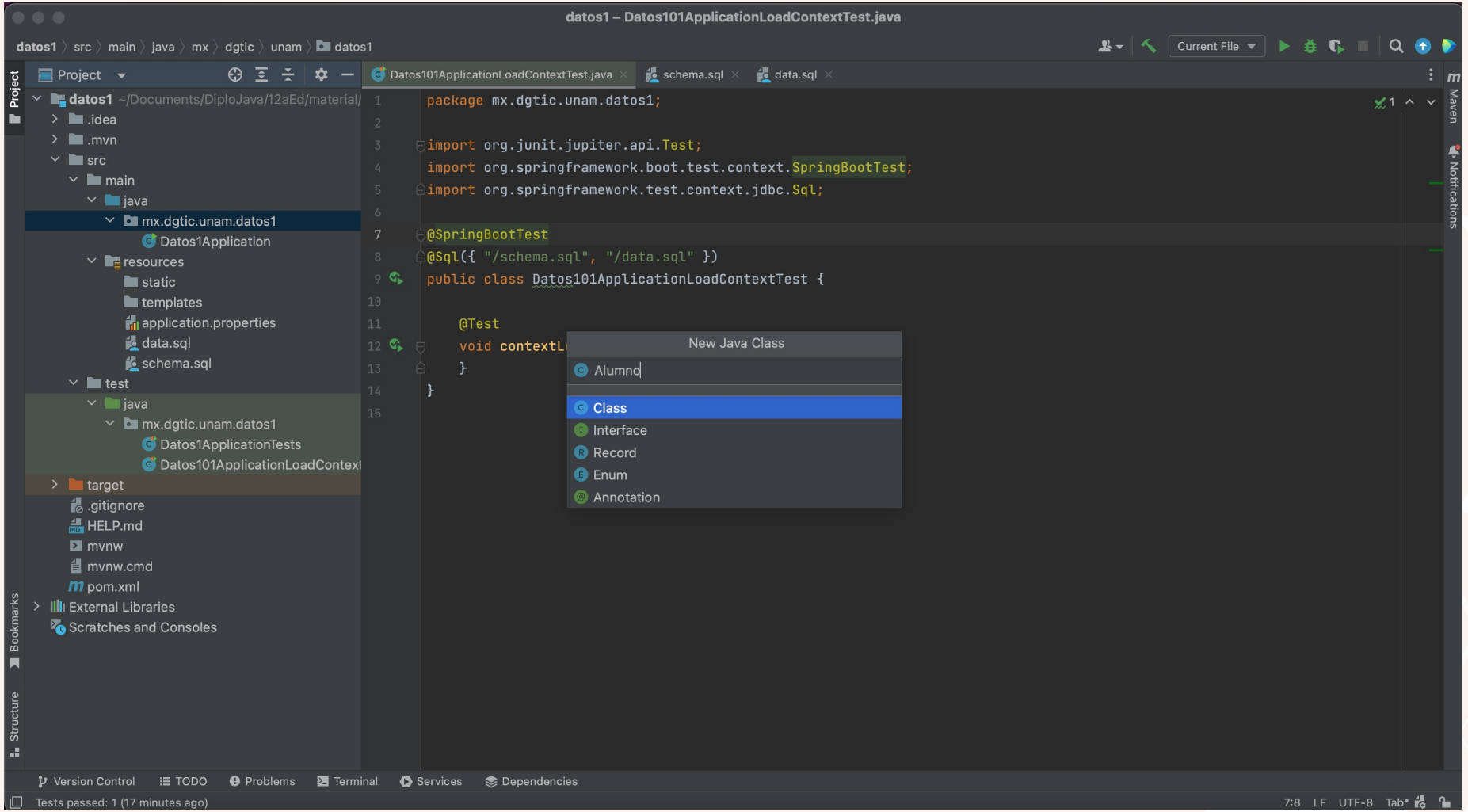
Crear el paquete datos



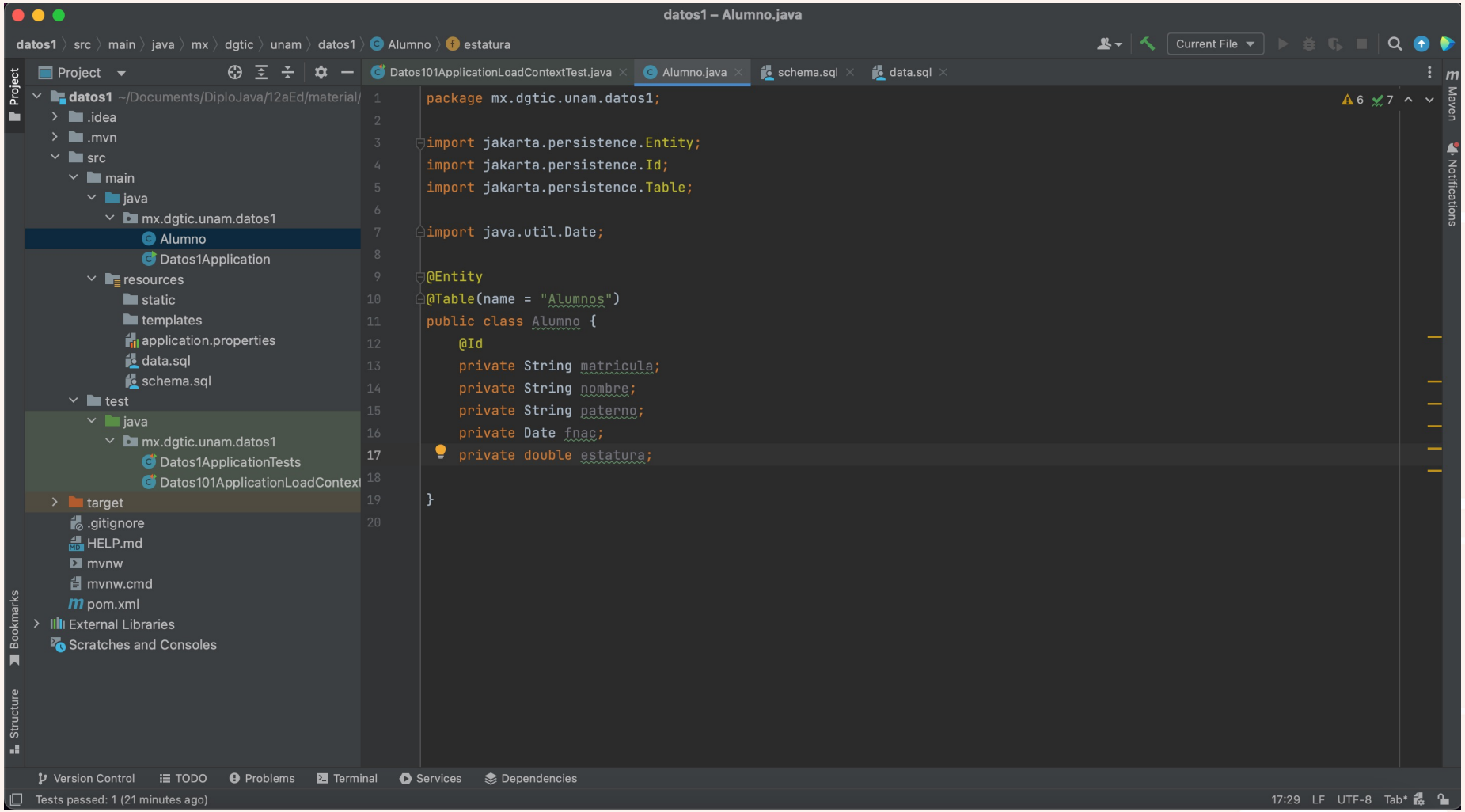
Generar una entidad



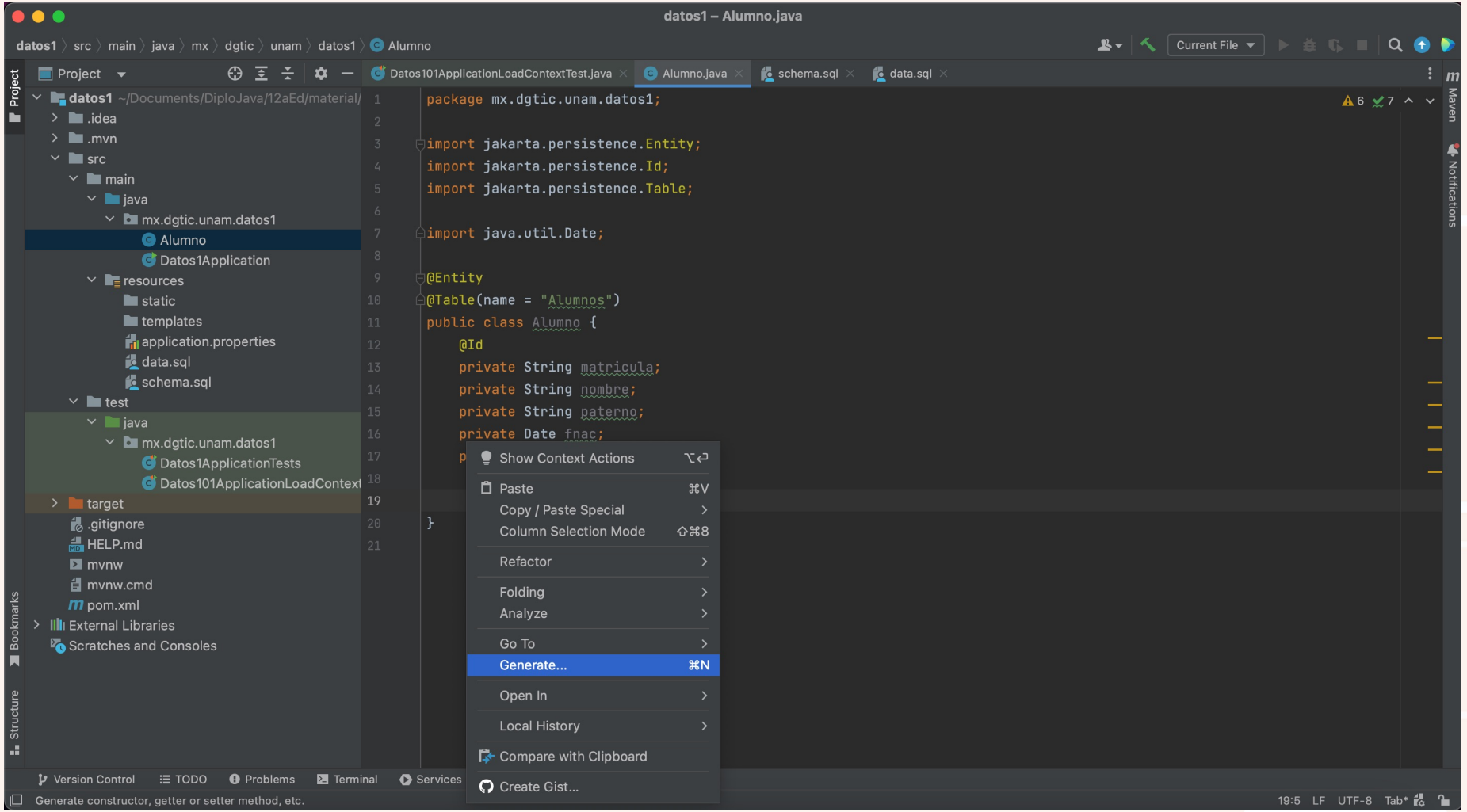
Generar una entidad



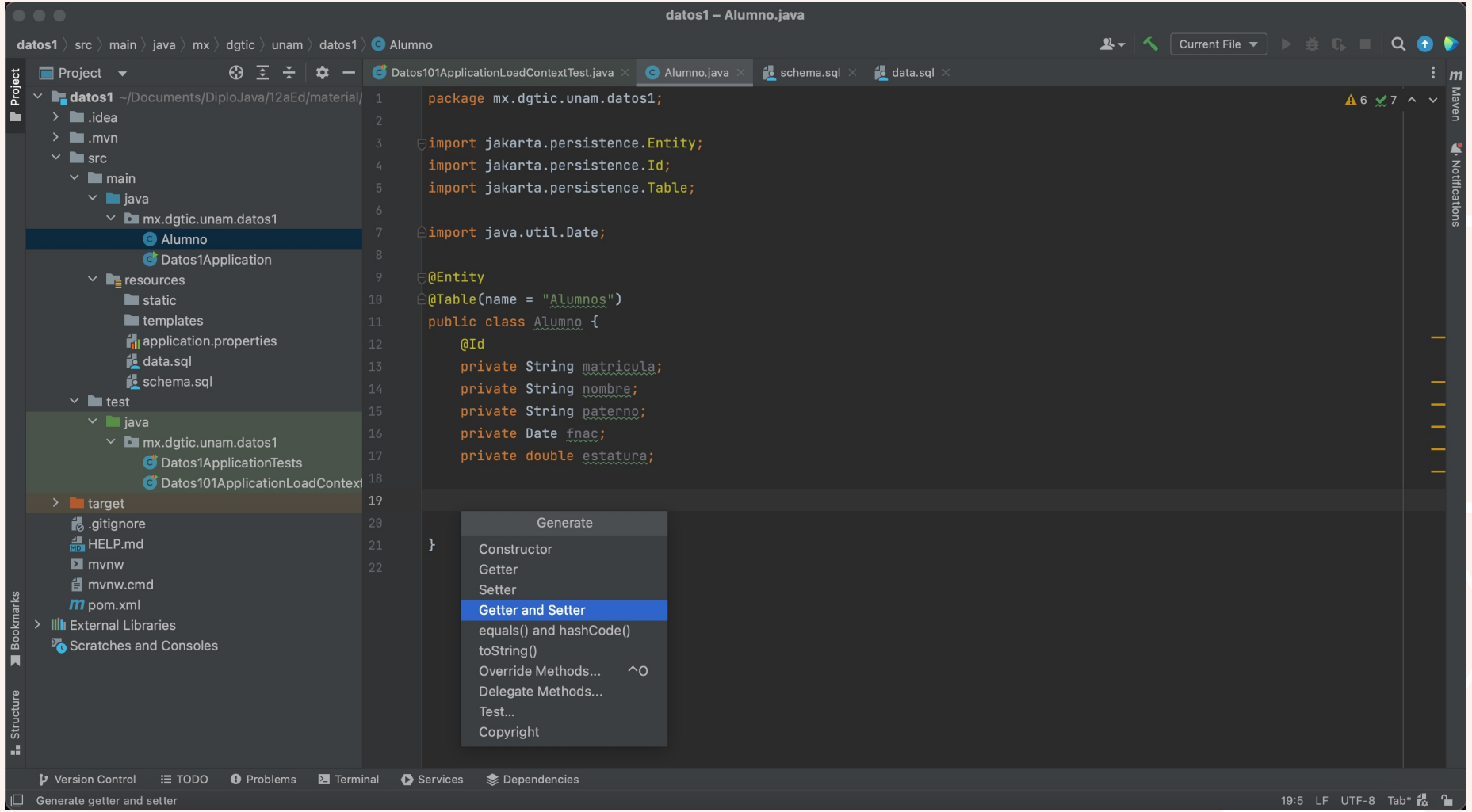
@Entity @Table



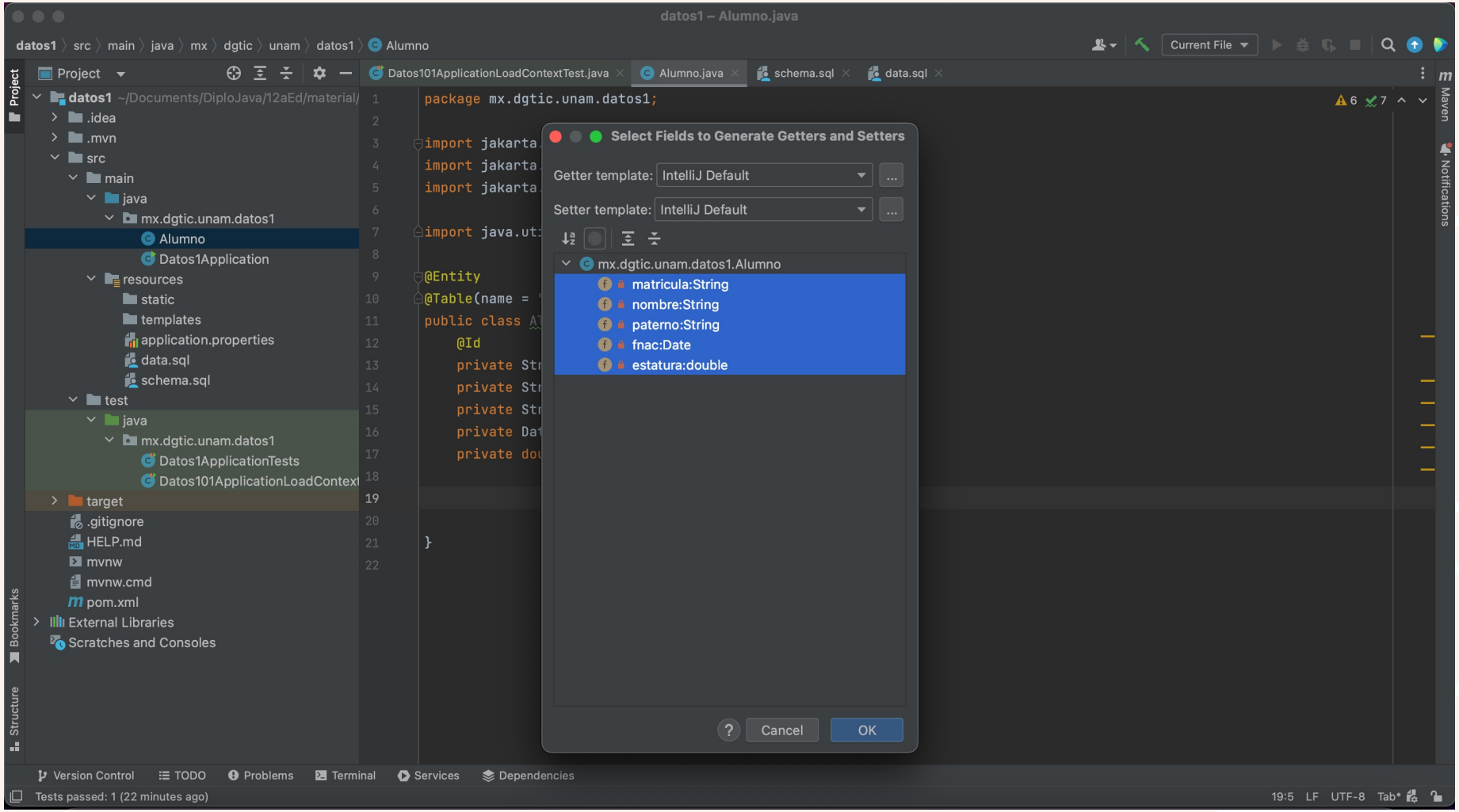
Generar Setters y Getters



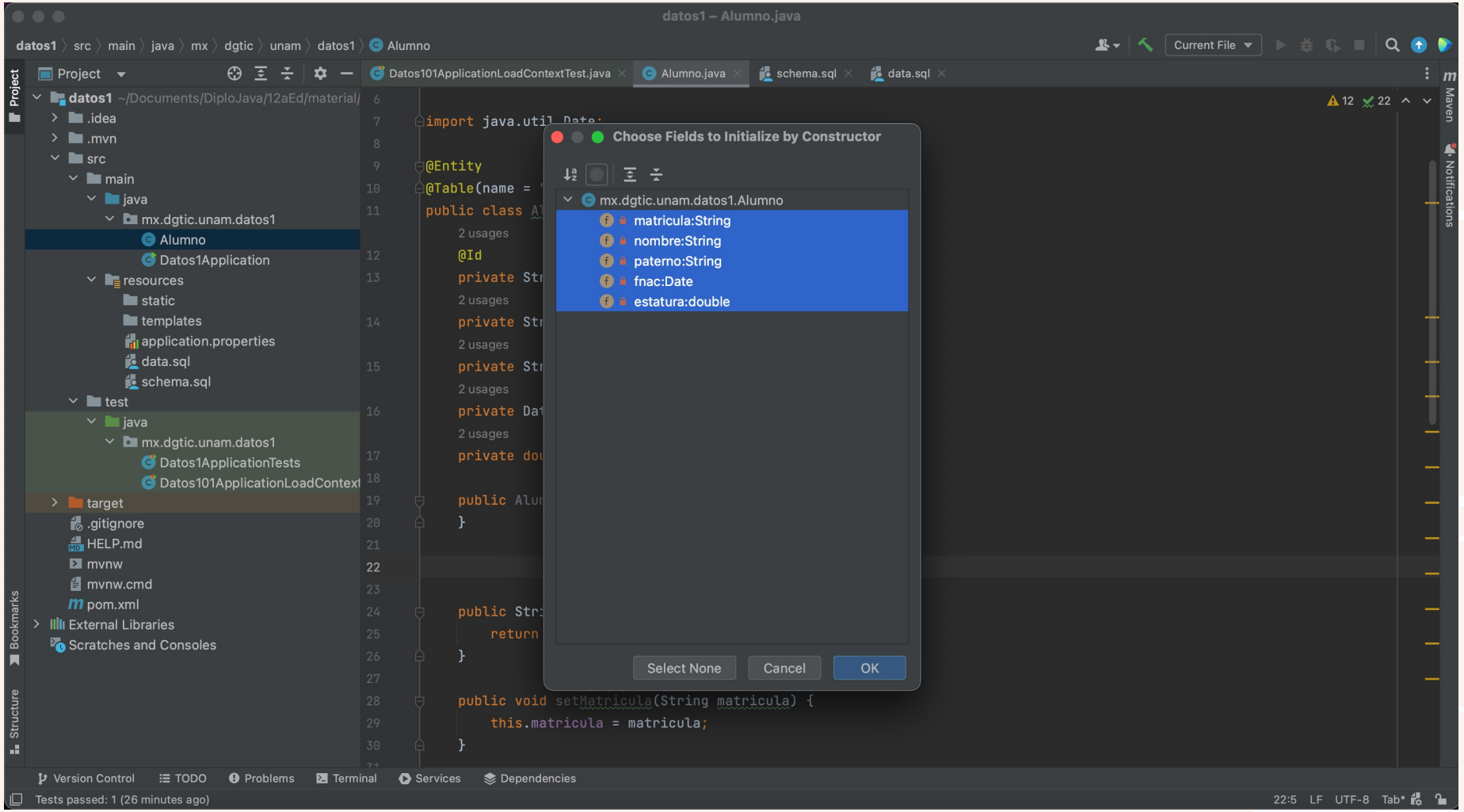
Generar Setters y Getters



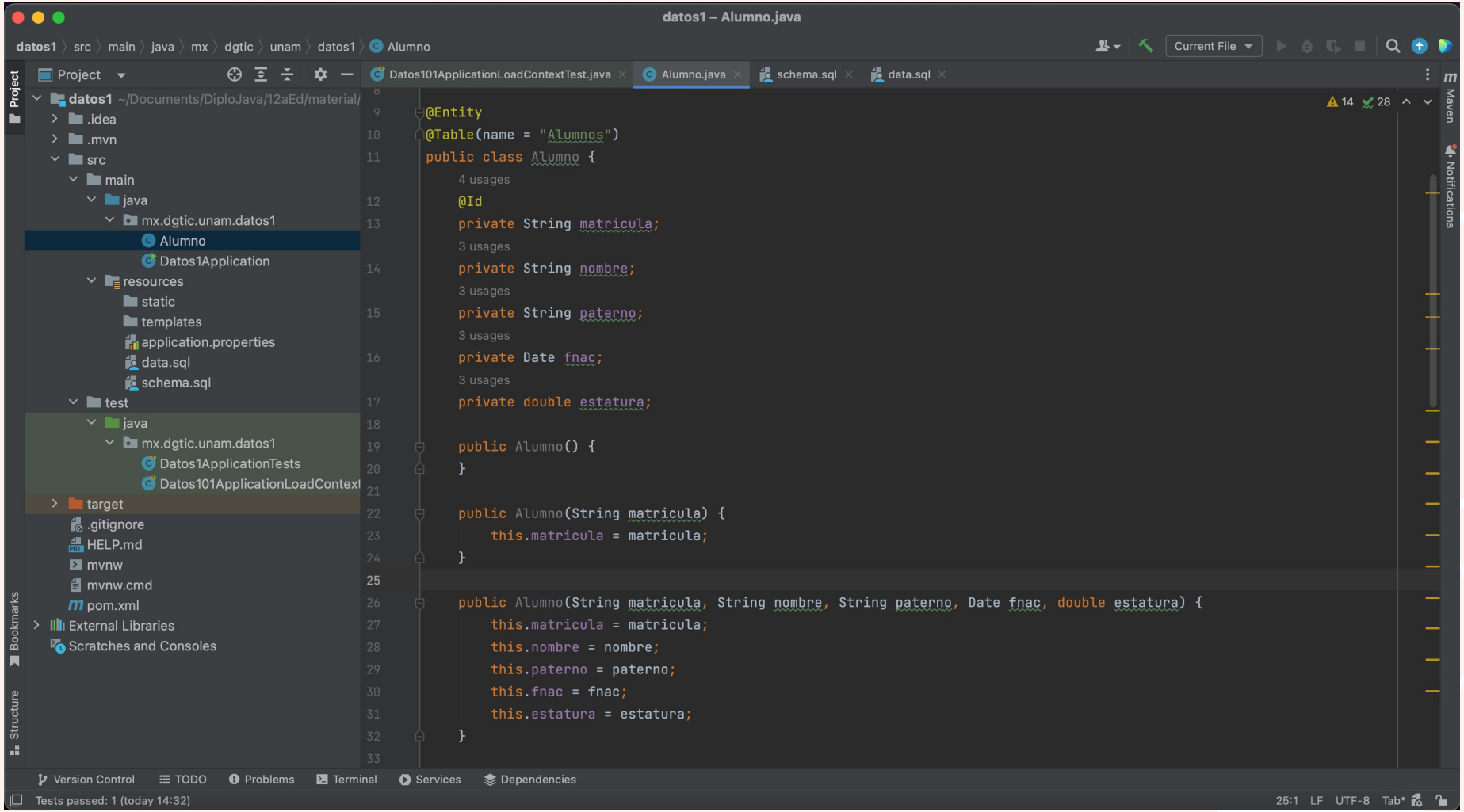
Generar Setters y Getters



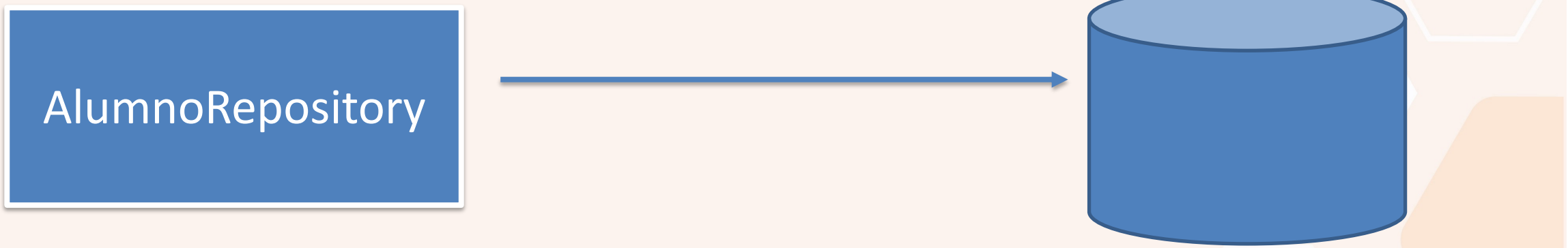
Generar constructores



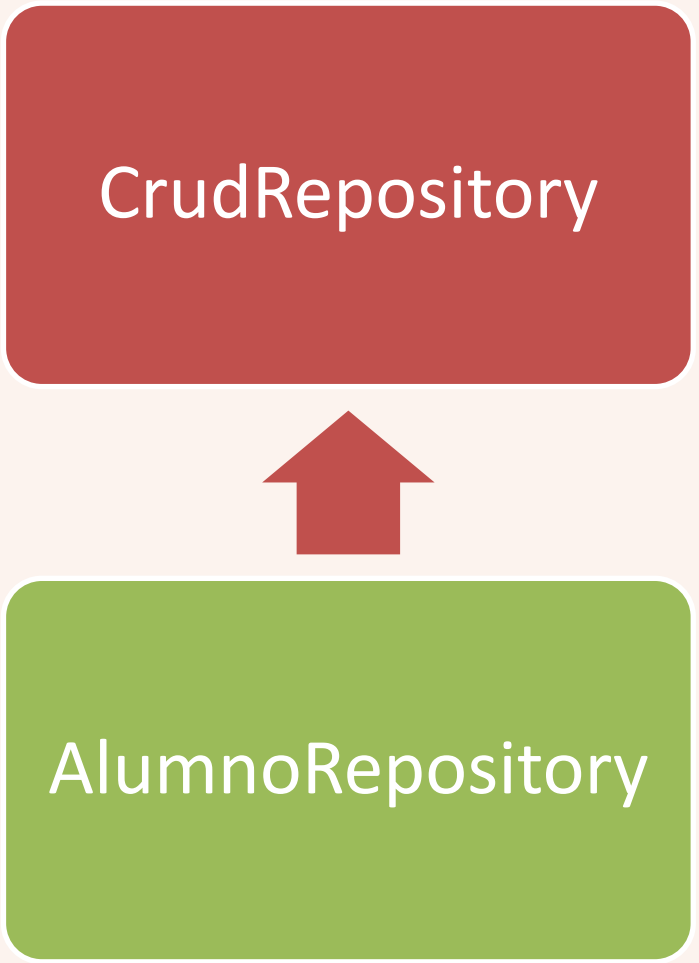
Generar constructores



Repositorio



Repositorio



Spring Data

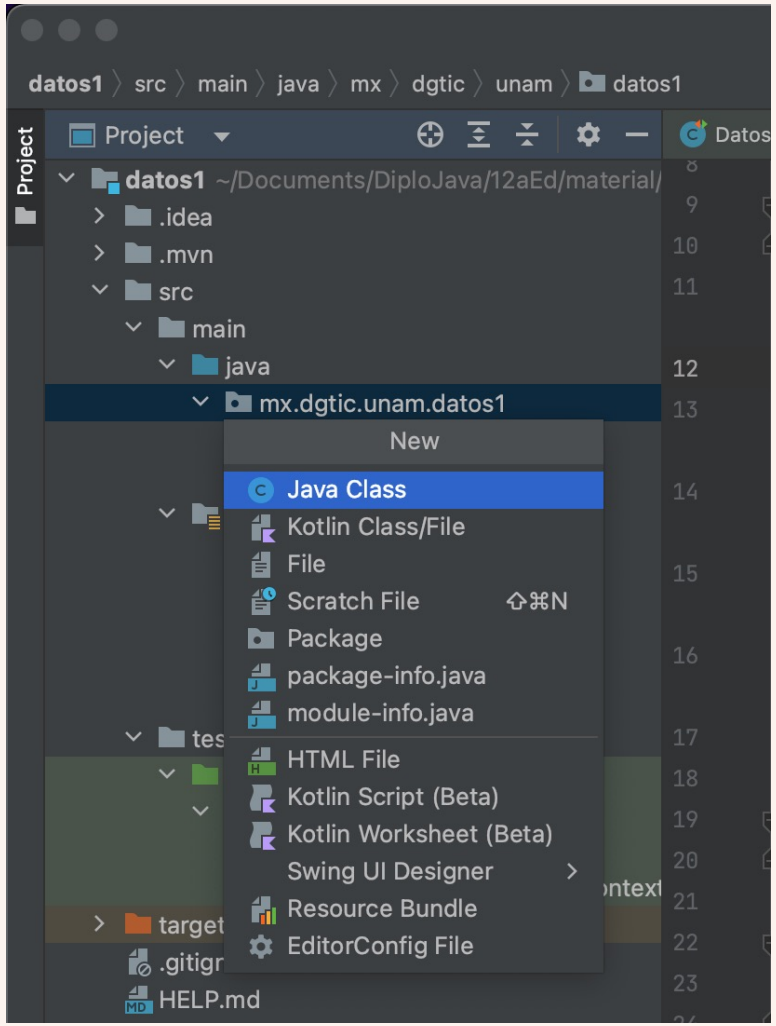
CrudRepository

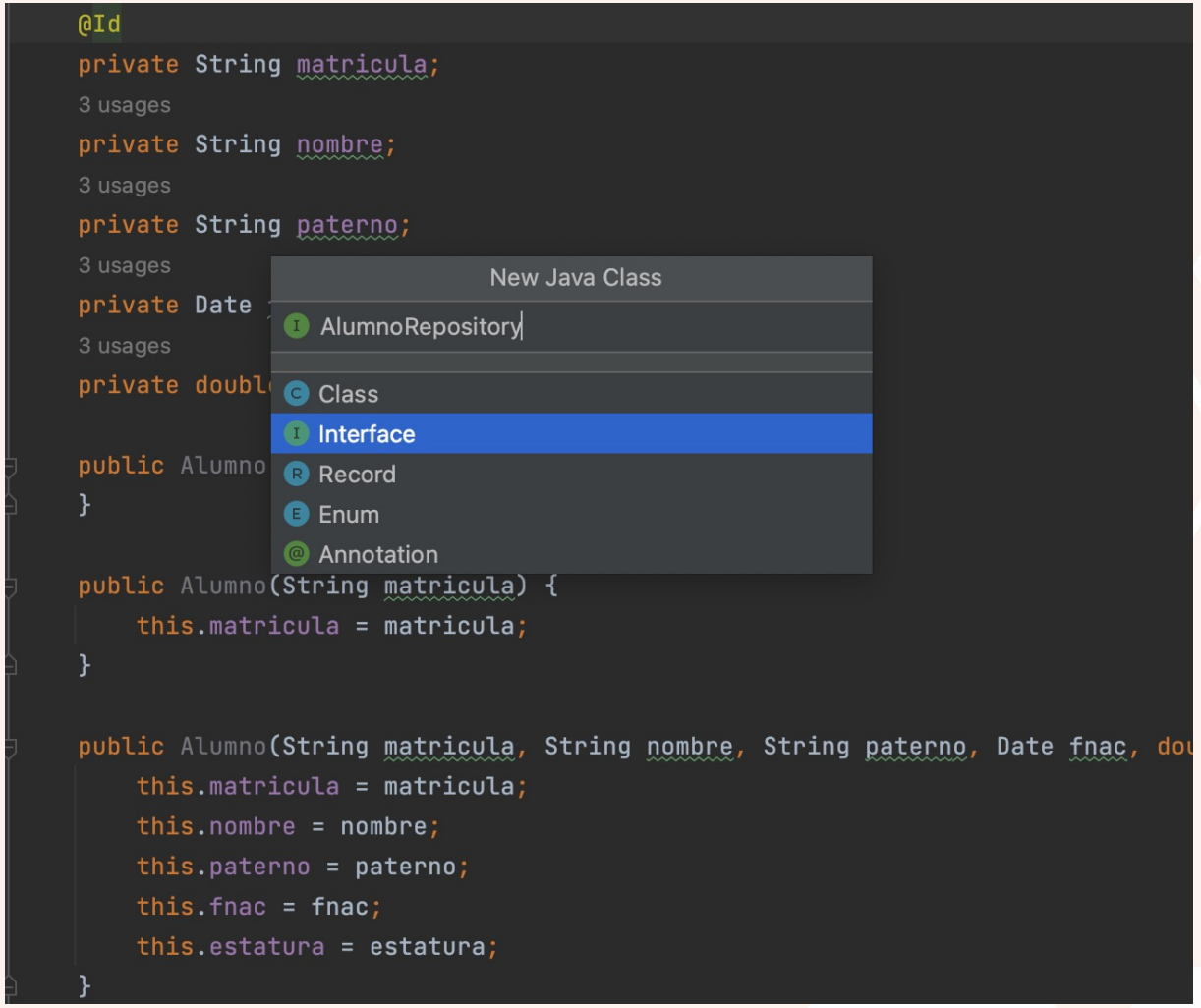
```
public interface CrudRepository<T, ID> extends
Repository<T, ID> {
  <S extends T> S save(S entity);
  Optional<T> findById(ID primaryKey);
  Iterable<T> findAll();
  long count();
  void delete(T entity);
  boolean existsById(ID primaryKey);
  // … more functionality omitted.
}
```

CrudRepository

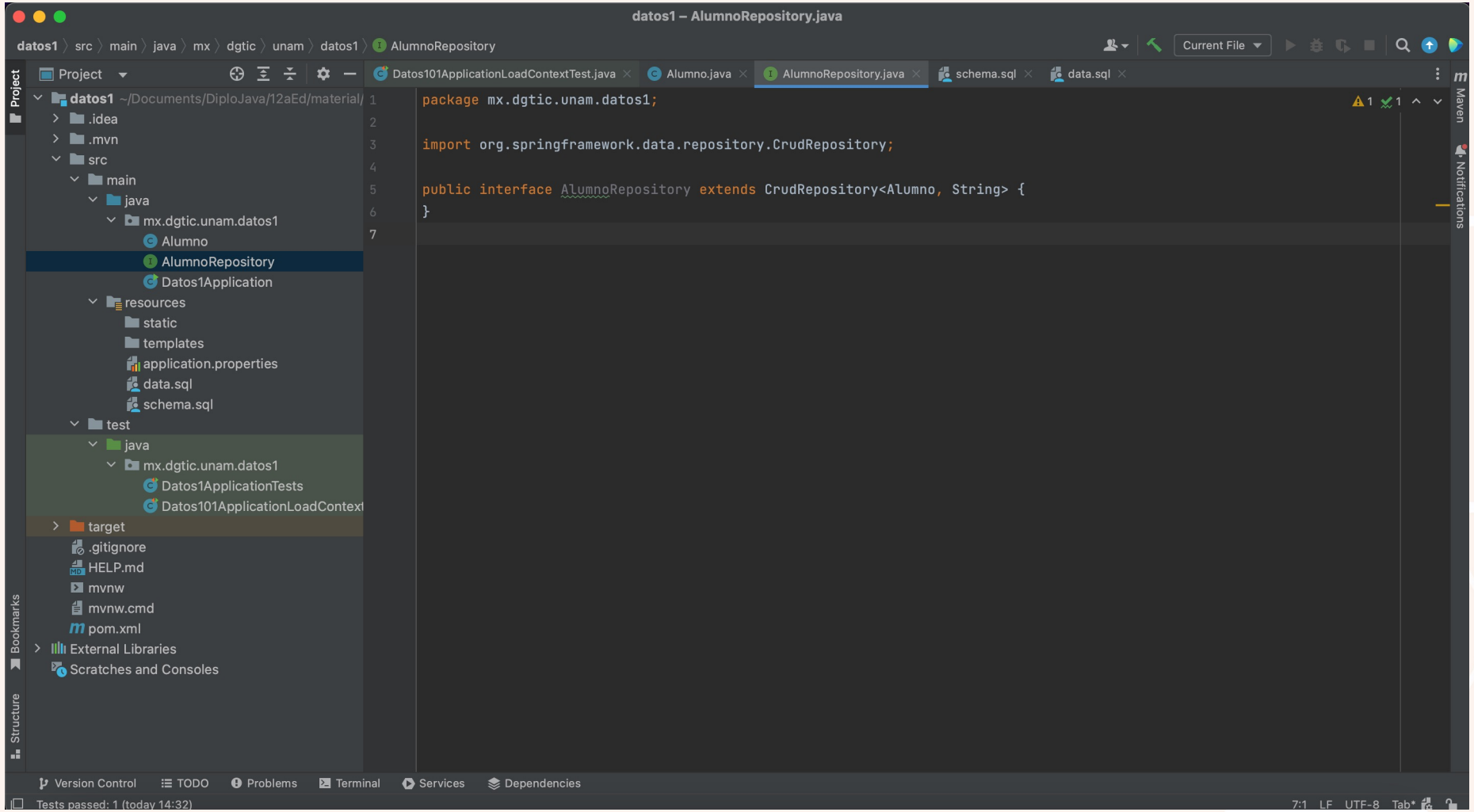
Modifier and Type	Method and Description
long	**count**()Returns the number of entities available.
void	**delete**(T entity)Deletes a given entity.
void	**deleteAll**()Deletes all entities managed by the repository.
void	**deleteAll**(Iterable<? extends T> entities)Deletes the given entities.
void	**deleteAllById**(Iterable<? 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

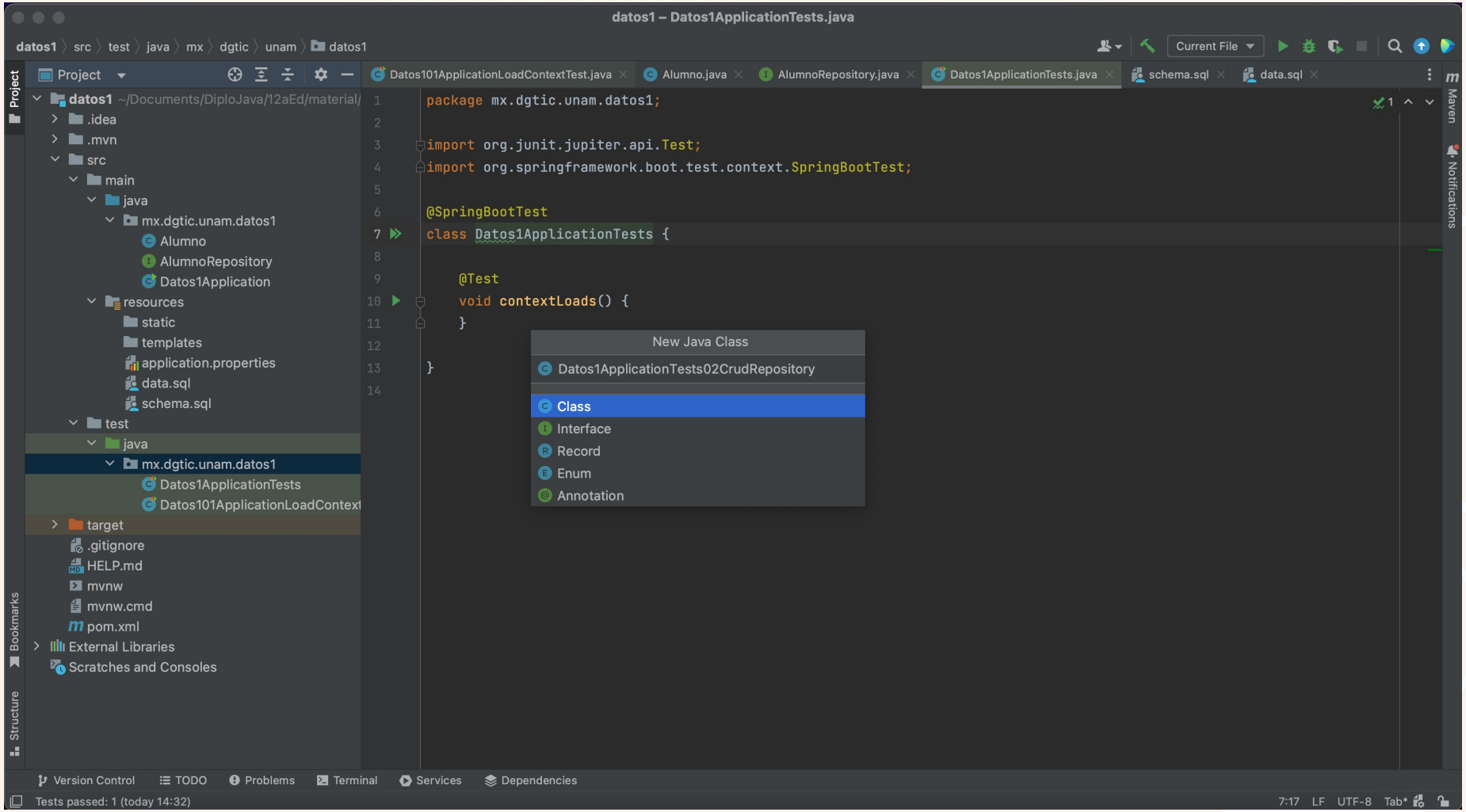




extends CrudRepository

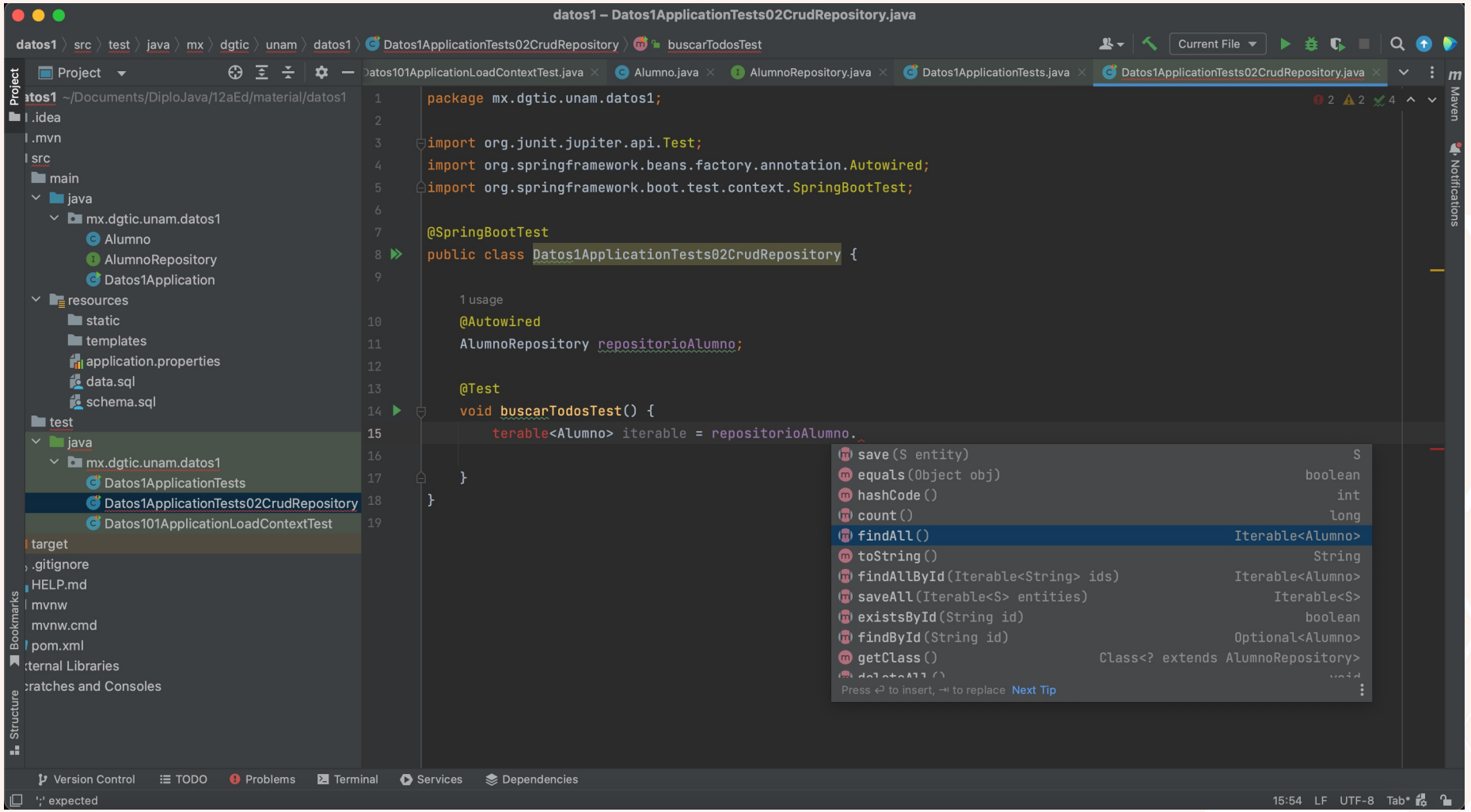


Agregar una nueva JUnit

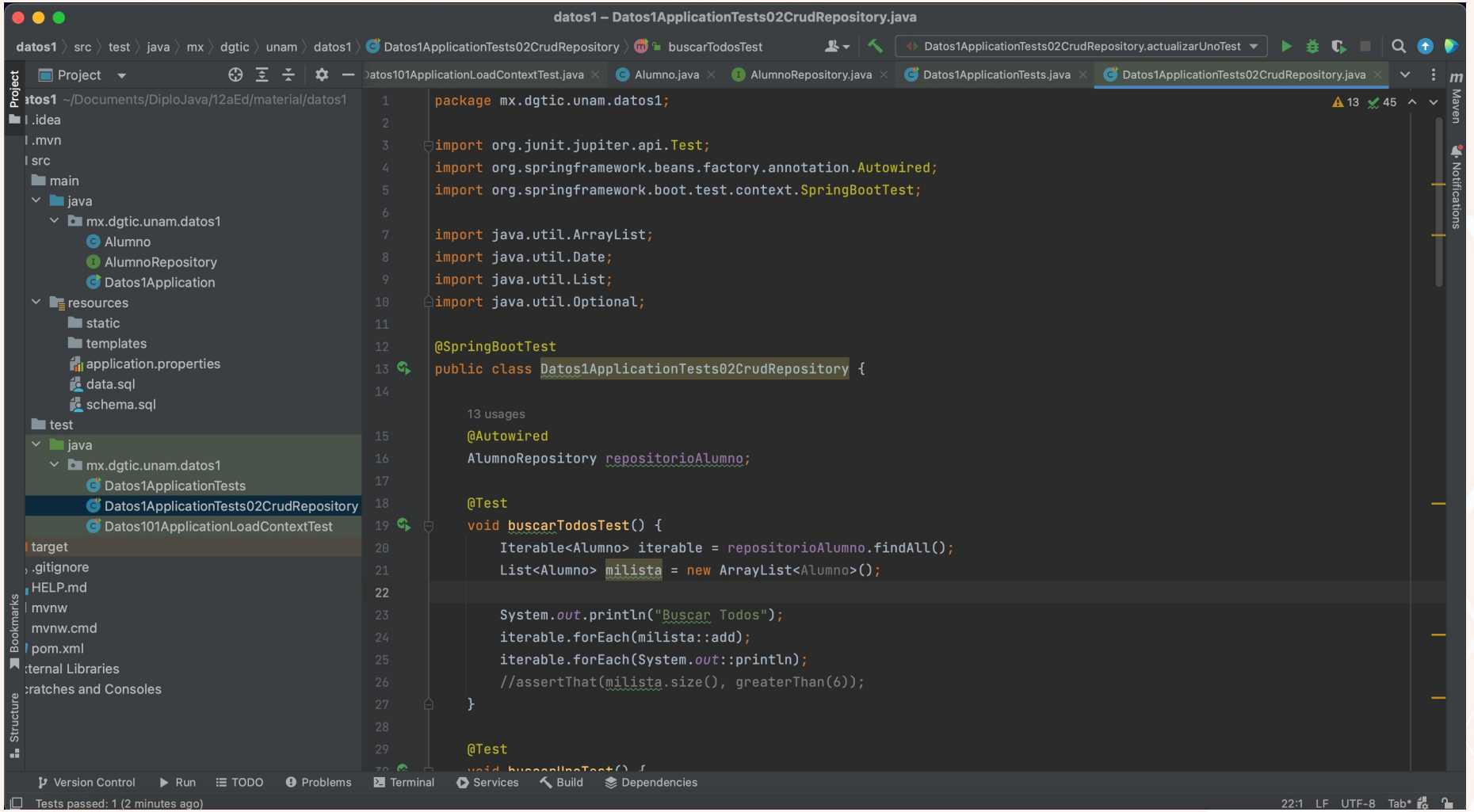


DGTIC UNAM
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

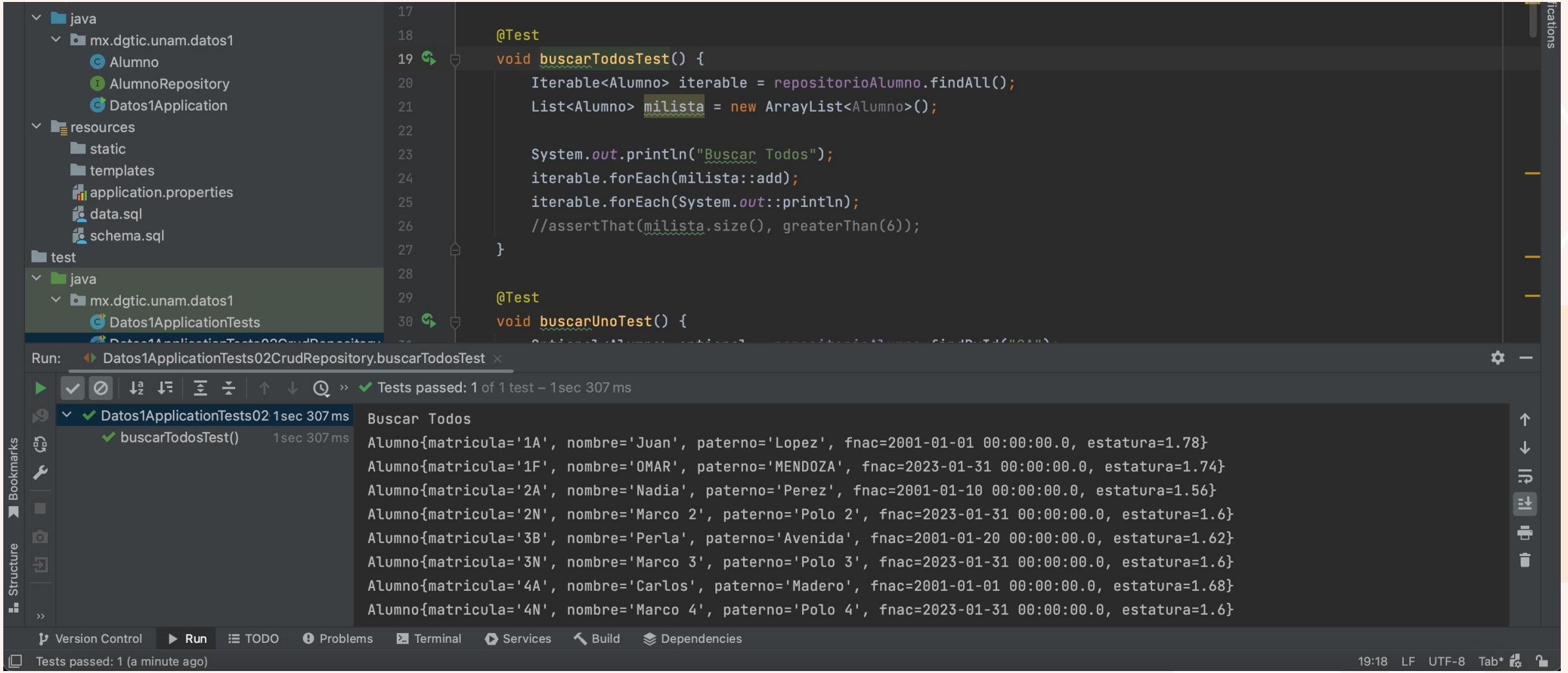
@Autowired



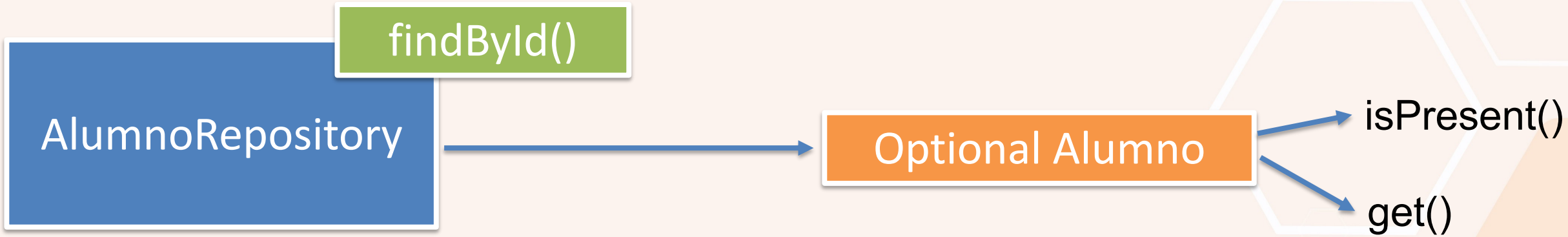
Probar findAll()



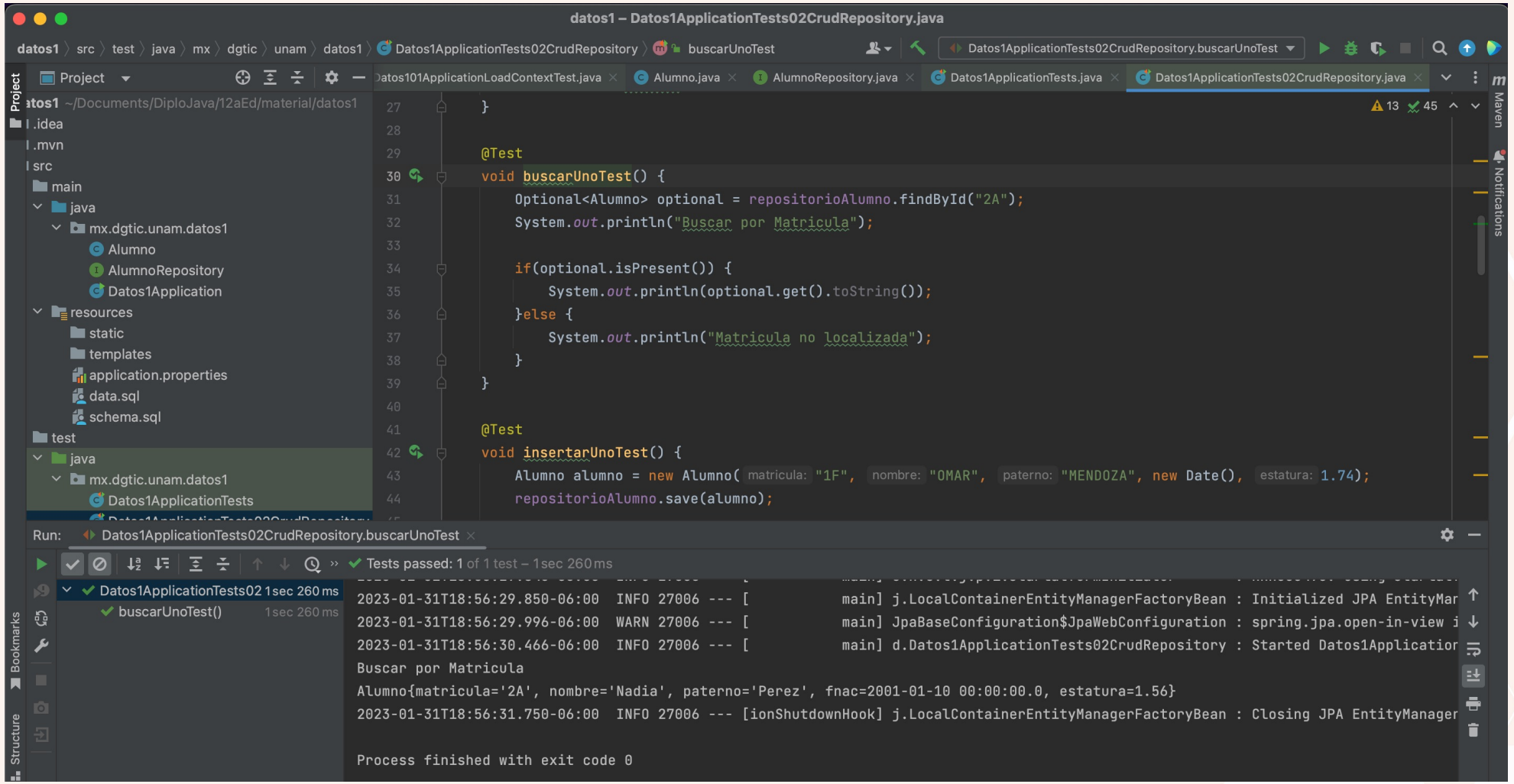
Probar findAll()



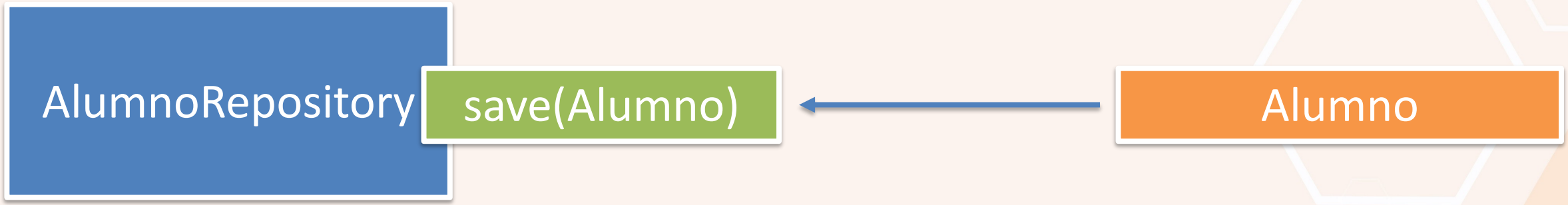
Probar findById()



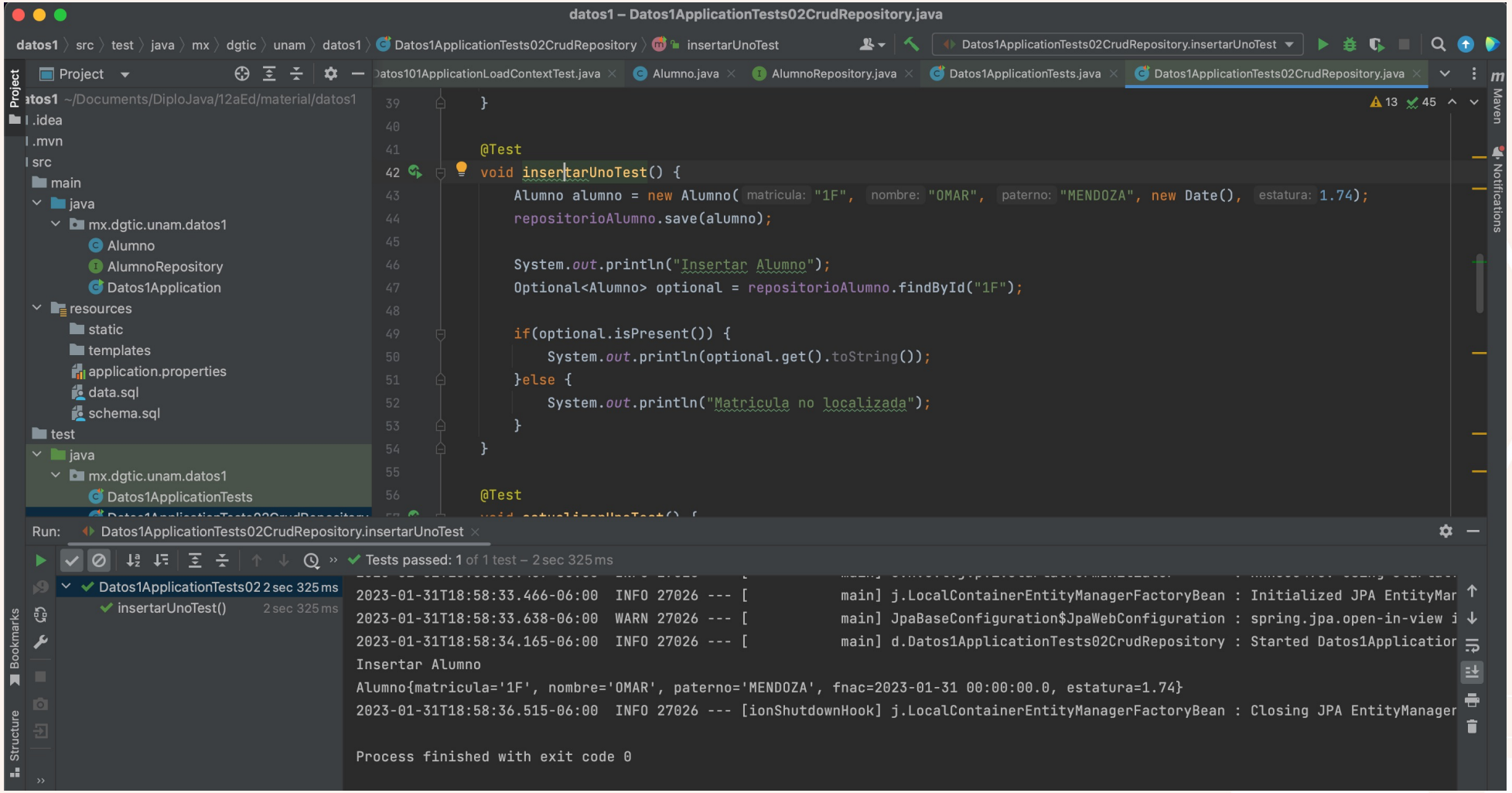
Probar findById()



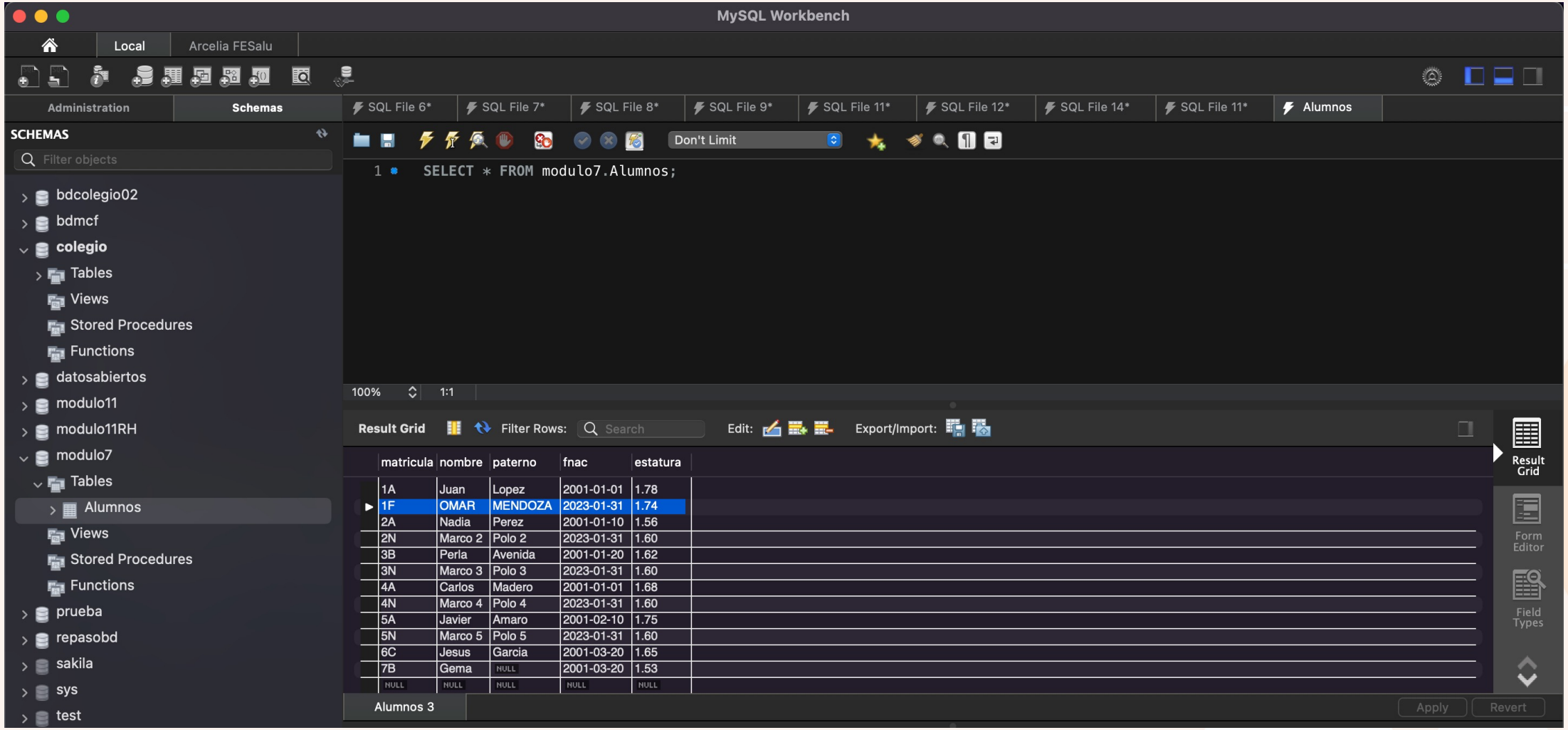
save()



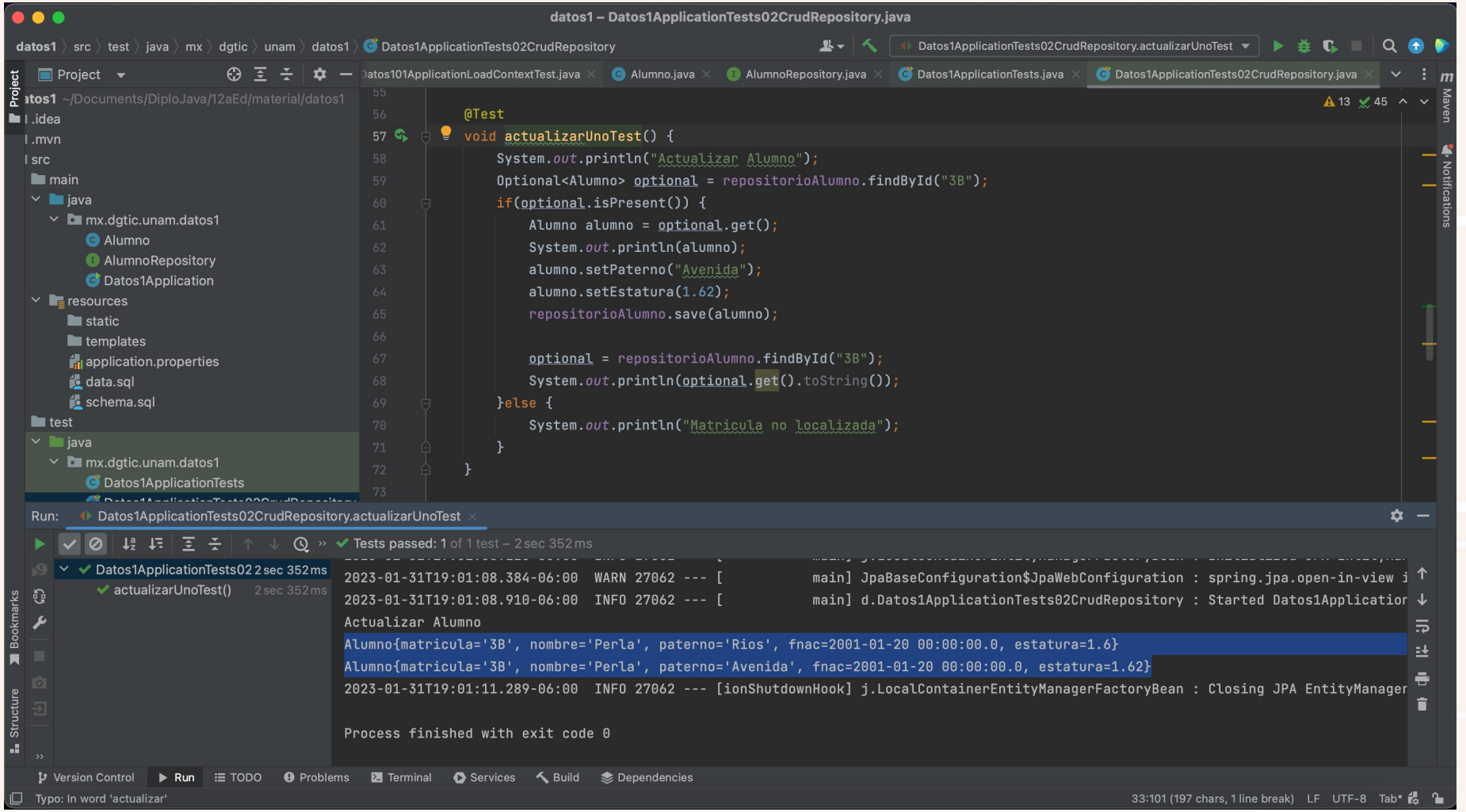
Probar save()


datos1 – Datos1ApplicationTests02CrudRepository.java
Project
Alumno.java
AlumnoRepository.java
Datos1ApplicationTests.java
Datos1ApplicationTests02CrudRepository.java
@Test
void insertarUnoTest() {
Alumno alumno = new Alumno(matricula: "1F", nombre: "OMAR", paterno: "MENDOZA", new Date(), estatura: 1.74);
repositorioAlumno.save(alumno);
System.out.println("Insertar Alumno");
Optional<Alumno> optional = repositorioAlumno.findById("1F");
if(optional.isPresent()) {
System.out.println(optional.get().toString());
}else {
System.out.println("Matricula no localizada");
}
}
@Test
Run: Datos1ApplicationTests02CrudRepository.insertarUnoTest
Tests passed: 1 of 1 test – 2 sec 325 ms
Datos1ApplicationTests02 2 sec 325 ms
insertarUnoTest() 2 sec 325 ms
2023-01-31T18:58:33.466-06:00 INFO 27026 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityMar
2023-01-31T18:58:33.638-06:00 WARN 27026 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view i
2023-01-31T18:58:34.165-06:00 INFO 27026 --- [main] d.Datos1ApplicationTests02CrudRepository : Started Datos1Applicatior
Insertar Alumno
Alumno{matricula='1F', nombre='OMAR', paterno='MENDOZA', fnac=2023-01-31 00:00:00.0, estatura=1.74}
2023-01-31T18:58:36.515-06:00 INFO 27026 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManager
Process finished with exit code 0

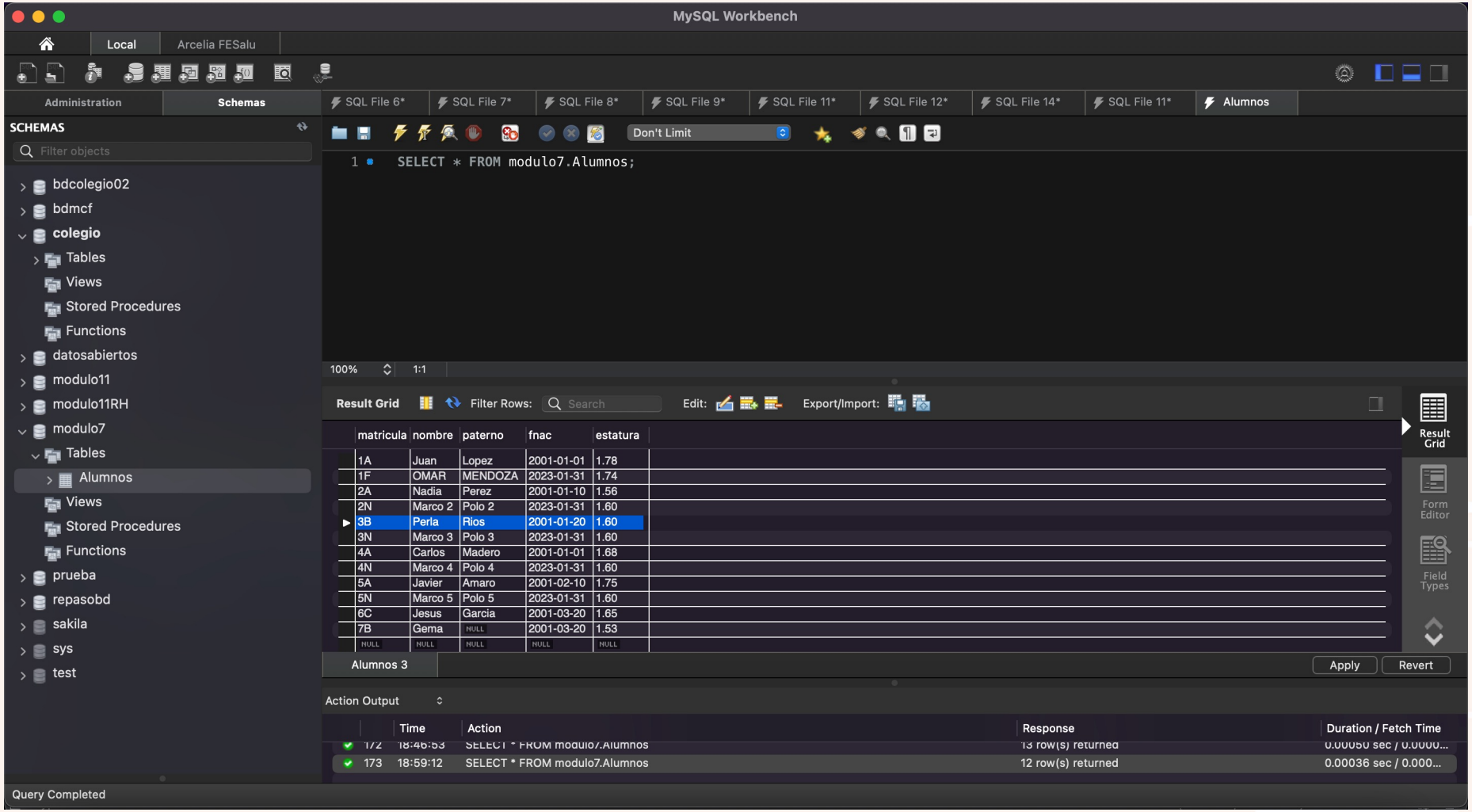
Probar save()


MySQL Workbench
Local
Arcelia FESalu
Administration
Schemas
SQL File 6*
SQL File 7*
SQL File 8*
SQL File 9*
SQL File 11*
SQL File 12*
SQL File 14*
SQL File 11*
Alumnos
SCHEMAS
Filter objects
bdcolegio02
bdmcf
colegio
Tables
Views
Stored Procedures
Functions
datosabiertos
modulo11
modulo11RH
modulo7
Tables
Alumnos
Views
Stored Procedures
Functions
prueba
repasobd
sakila
sys
test
Don't Limit
1 SELECT * FROM modulo7.Alumnos;
100%
1:1
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
matricula nombre paterno fnac estatura
1A Juan Lopez 2001-01-01 1.78
1F OMAR MENDOZA 2023-01-31 1.74
2A Nadia Perez 2001-01-10 1.56
2N Marco 2 Polo 2 2023-01-31 1.60
3B Perla Avenida 2001-01-20 1.62
3N Marco 3 Polo 3 2023-01-31 1.60
4A Carlos Madero 2001-01-01 1.68
4N Marco 4 Polo 4 2023-01-31 1.60
5A Javier Amaro 2001-02-10 1.75
5N Marco 5 Polo 5 2023-01-31 1.60
6C Jesus Garcia 2001-03-20 1.65
7B Gema NULL 2001-03-20 1.53
NULL NULL NULL NULL NULL
Result Grid
Form Editor
Field Types
Alumnos 3
Apply
Revert

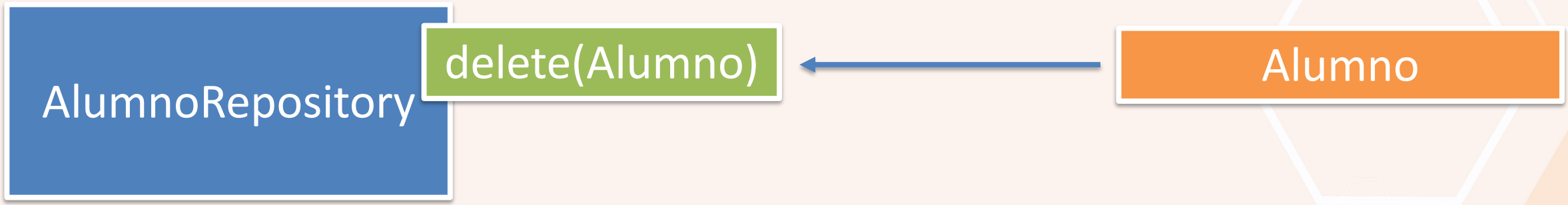
Probar save() actualizar



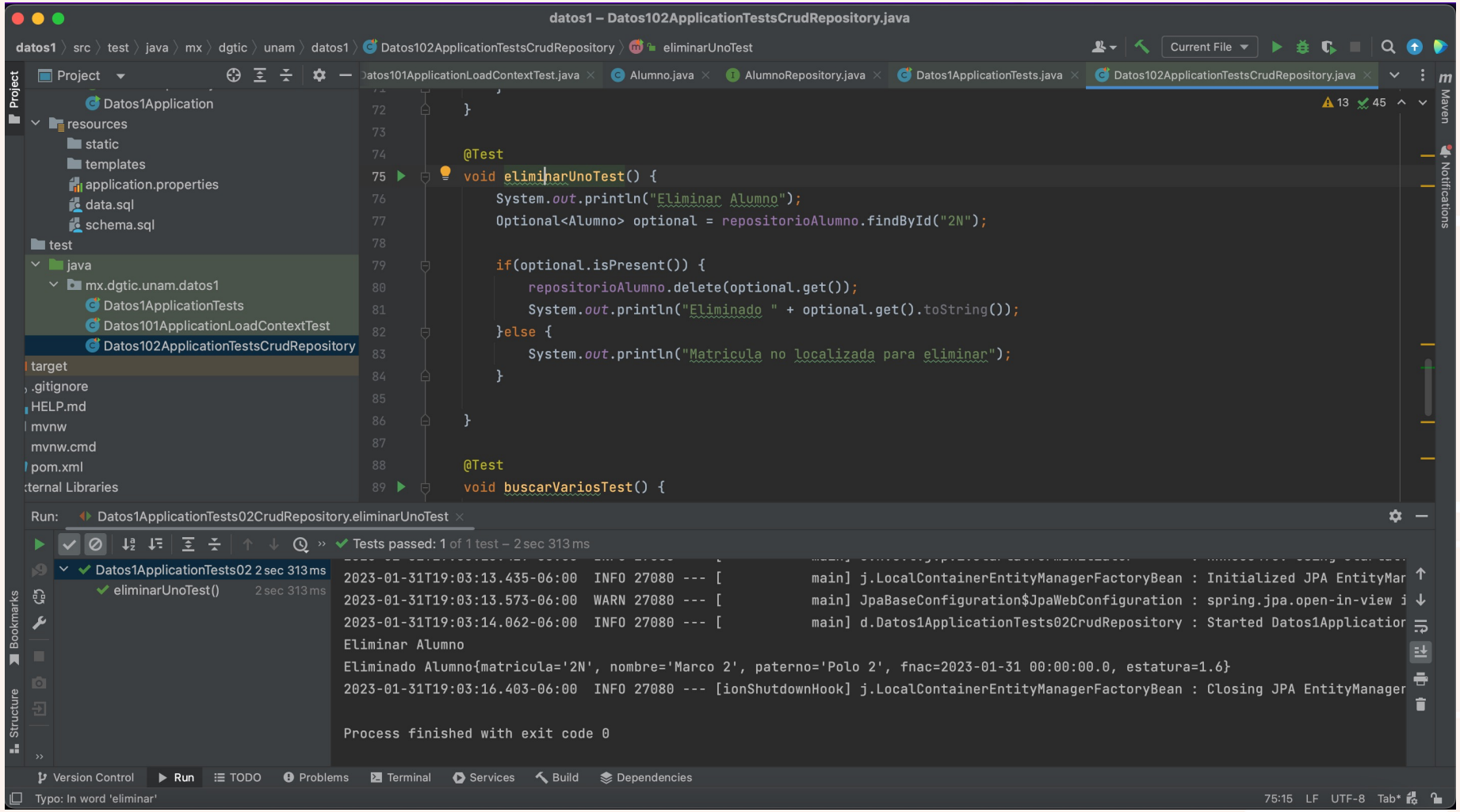
Probar save() actualizar



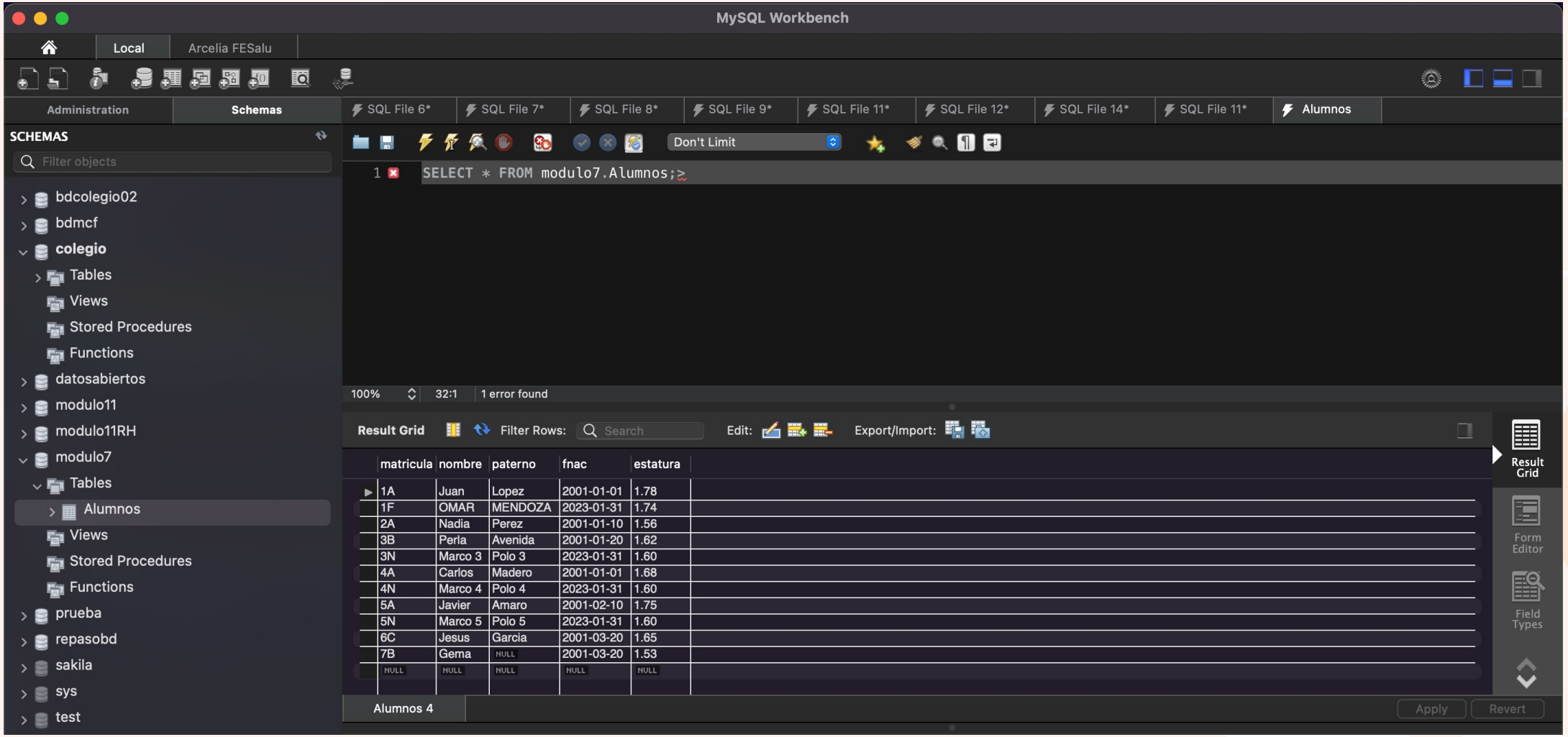
delete()



Probar delete()

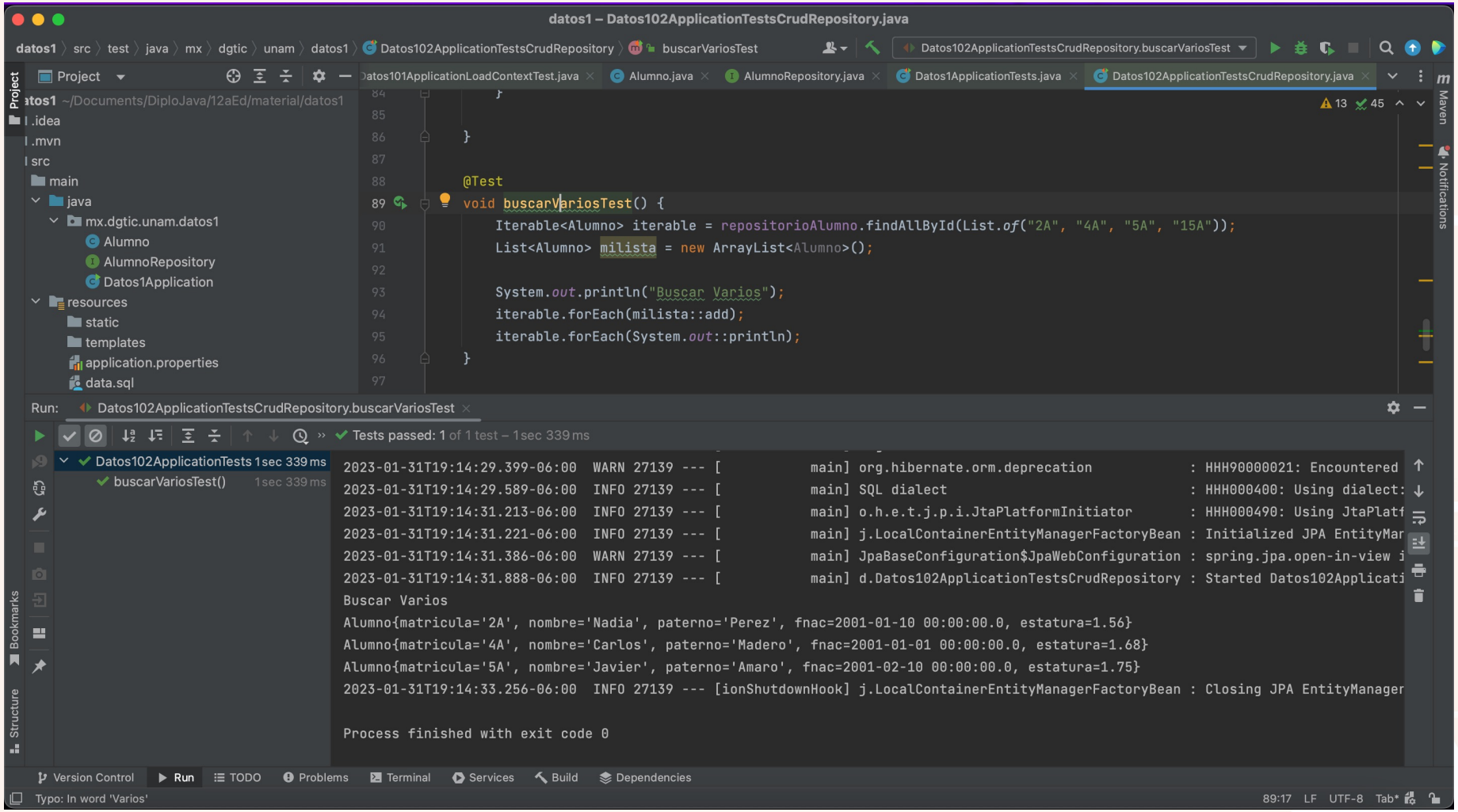


Probar delete()

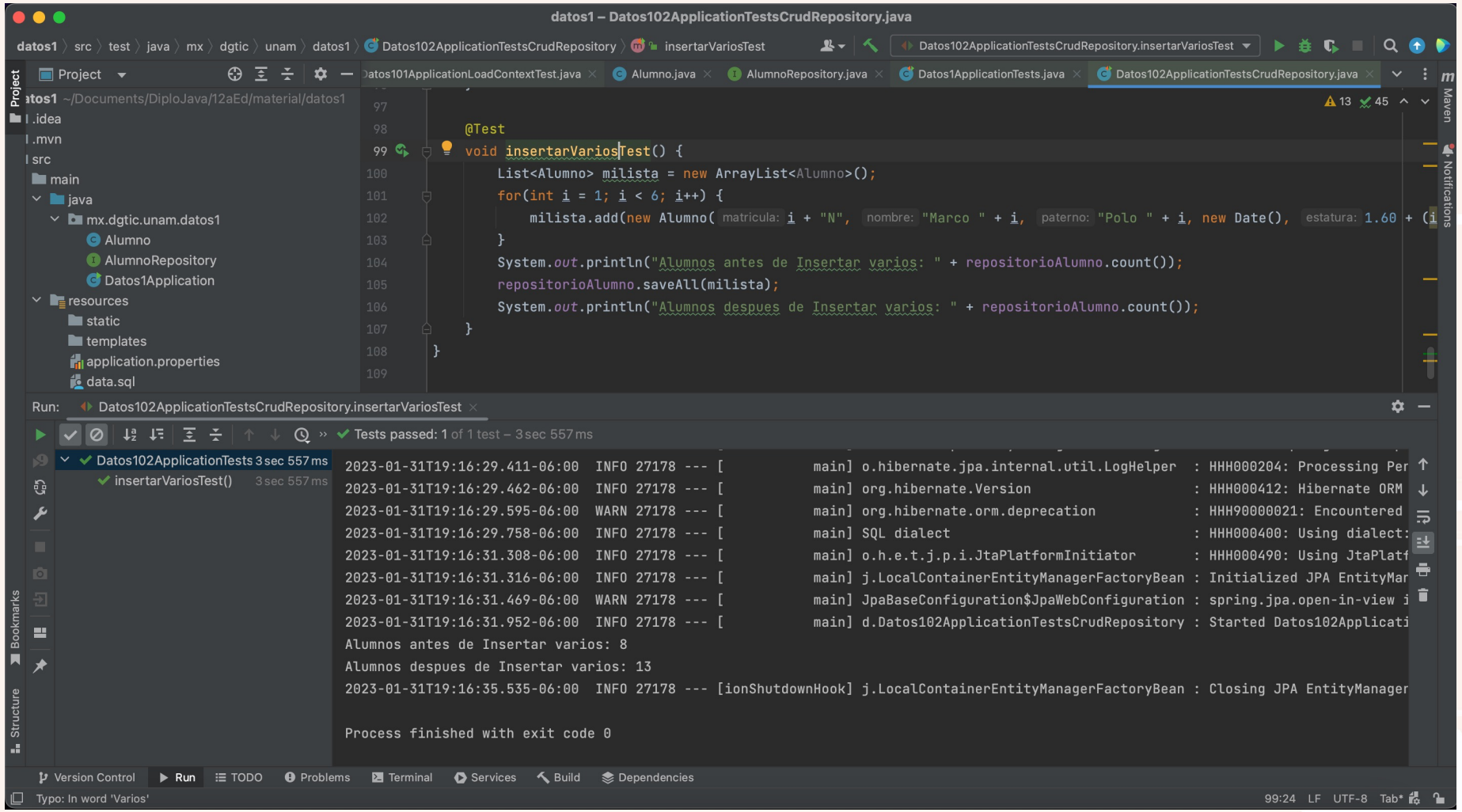


Colecciones

findAllById()



saveAll()


datos1 – Datos102ApplicationTestsCrudRepository.java
@Test
void insertarVariosTest() {
List<Alumno> milista = new ArrayList<Alumno>();
for(int i = 1; i < 6; i++) {
milista.add(new Alumno(matricula: i + "N", nombre: "Marco " + i, paterno: "Polo " + i, new Date(), estatura: 1.60 + (i
}
System.out.println("Alumnos antes de Insertar varios: " + repositorioAlumno.count());
repositorioAlumno.saveAll(milista);
System.out.println("Alumnos despues de Insertar varios: " + repositorioAlumno.count());
}
}
Run: Datos102ApplicationTestsCrudRepository.insertarVariosTest
Tests passed: 1 of 1 test – 3 sec 557 ms
Datos102ApplicationTests 3 sec 557 ms
insertarVariosTest() 3 sec 557 ms
2023-01-31T19:16:29.411-06:00 INFO 27178 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing Per
2023-01-31T19:16:29.462-06:00 INFO 27178 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM
2023-01-31T19:16:29.595-06:00 WARN 27178 --- [main] org.hibernate.orm.deprecation : HHH90000021: Encountered
2023-01-31T19:16:29.758-06:00 INFO 27178 --- [main] SQL dialect : HHH000400: Using dialect:
2023-01-31T19:16:31.308-06:00 INFO 27178 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatf
2023-01-31T19:16:31.316-06:00 INFO 27178 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityMar
2023-01-31T19:16:31.469-06:00 WARN 27178 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view i
2023-01-31T19:16:31.952-06:00 INFO 27178 --- [main] d.Datos102ApplicationTestsCrudRepository : Started Datos102Applicati
Alumnos antes de Insertar varios: 8
Alumnos despues de Insertar varios: 13
2023-01-31T19:16:35.535-06:00 INFO 27178 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManager
Process finished with exit code 0

deleteAll()

- Elimina todas las entidades administradas por el repositorio.

Contacto

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx