



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.

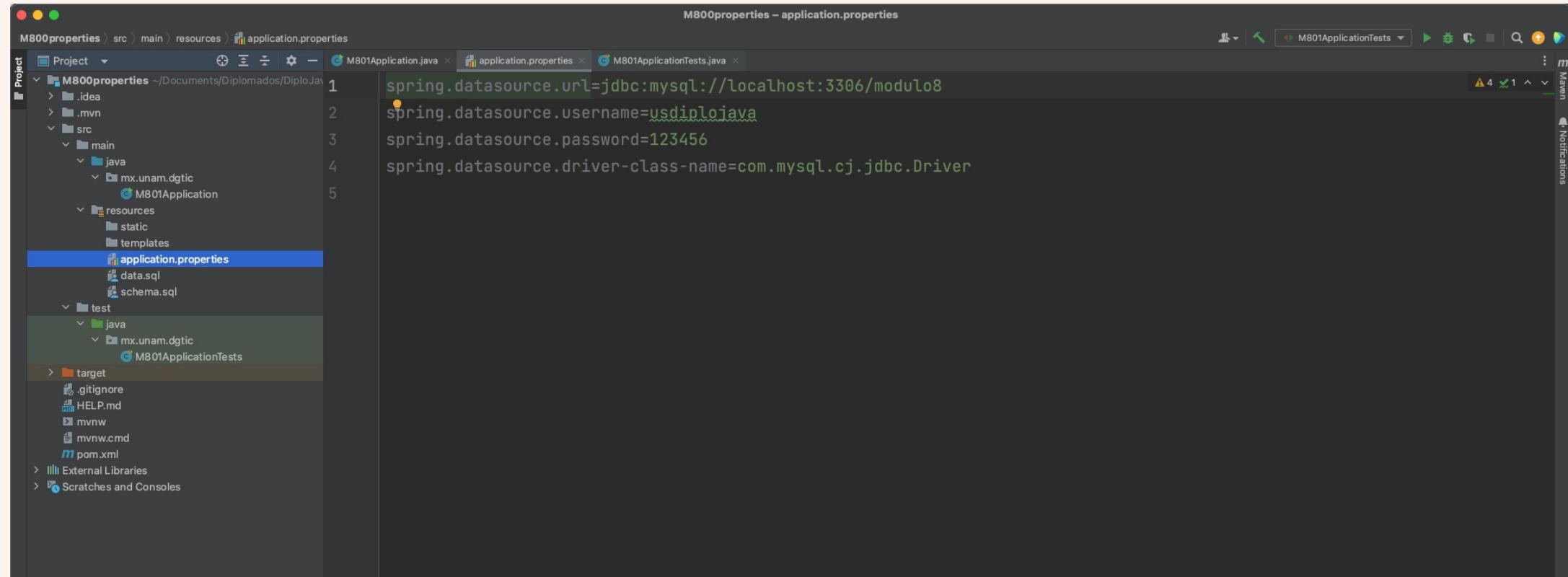
@EnableJpaRepositories(basePackages = "mx.unam.dgtic.datos")

- 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.

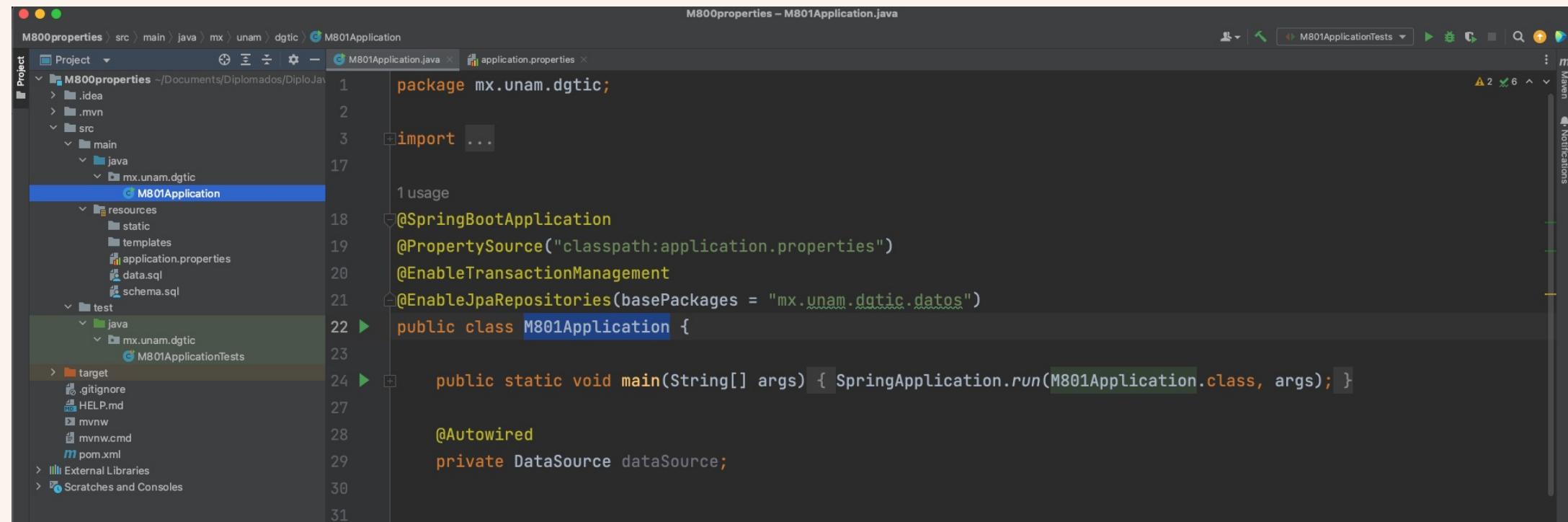
application.properties



The screenshot shows a Java project structure in an IDE. The project is named 'M800properties' and contains a 'src' directory with 'main' and 'test' packages. The 'main' package contains 'java' and 'resources' directories, with 'application.properties' being the active file. The 'test' package also contains a 'java' directory with 'M801ApplicationTests'. The 'resources' directory contains 'data.sql' and 'schema.sql'. The 'application.properties' file content is as follows:

```
spring.datasource.url=jdbc:mysql://localhost:3306/modulo8
spring.datasource.username=usdiplojava
spring.datasource.password=123456
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

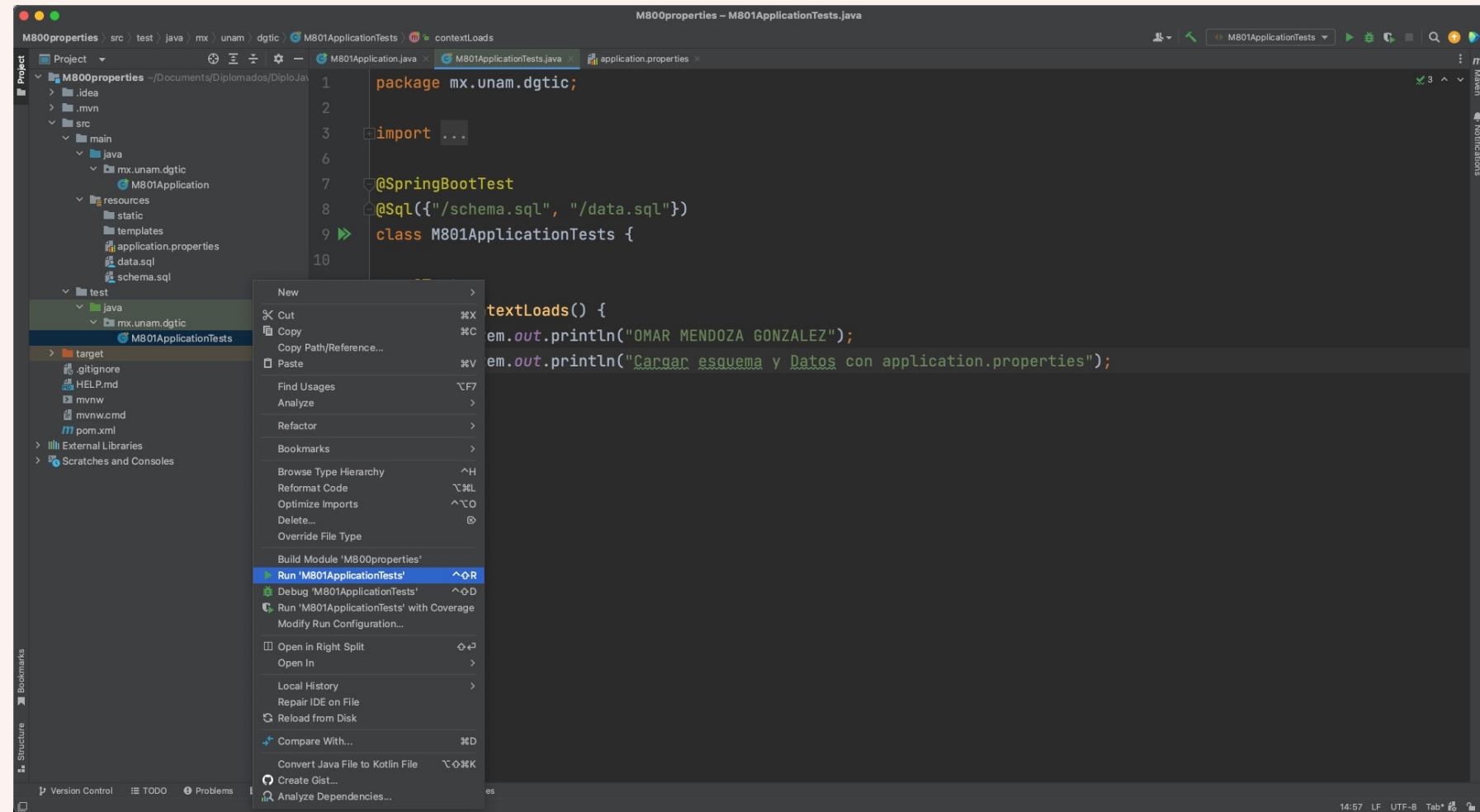
M801Application



The screenshot shows the IntelliJ IDEA interface with the project 'M800properties' open. The 'M801Application.java' file is the active editor. The code implements a Spring Boot application with annotations for configuration and transaction management. The project structure on the left includes 'src' with 'main' and 'test' directories, and 'target' for build artifacts.

```
package mx.unam.dgtic;
import ...
@SpringBootApplication
@PropertySource("classpath:application.properties")
@EnableTransactionManagement
@EnableJpaRepositories(basePackages = "mx.unam.dgtic.datos")
public class M801Application {
    public static void main(String[] args) { SpringApplication.run(M801Application.class, args); }
    @Autowired
    private DataSource dataSource;
}
```

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.
 - 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).

Entidades

@Entity

Alumno

@matricula

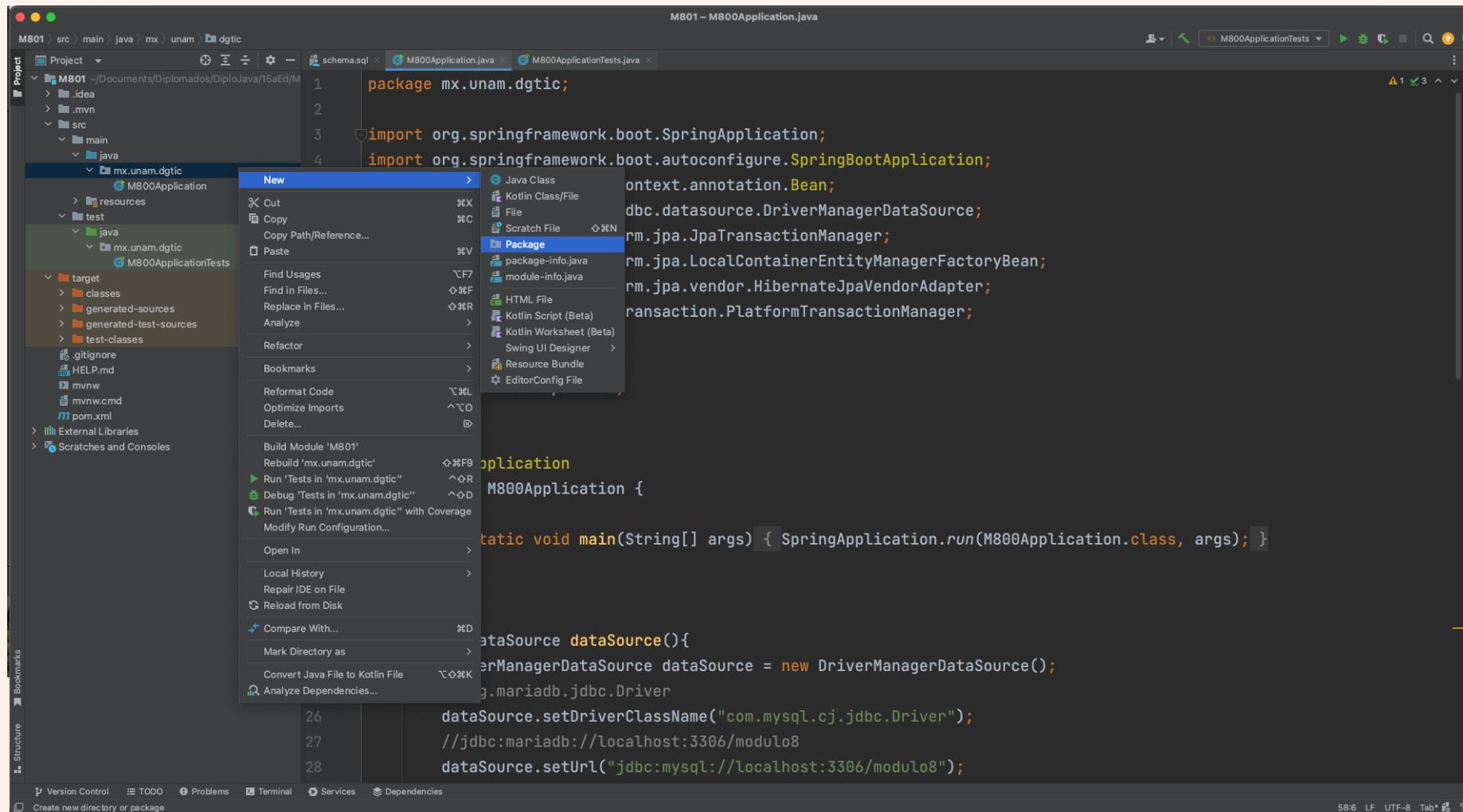
nombre

paterno

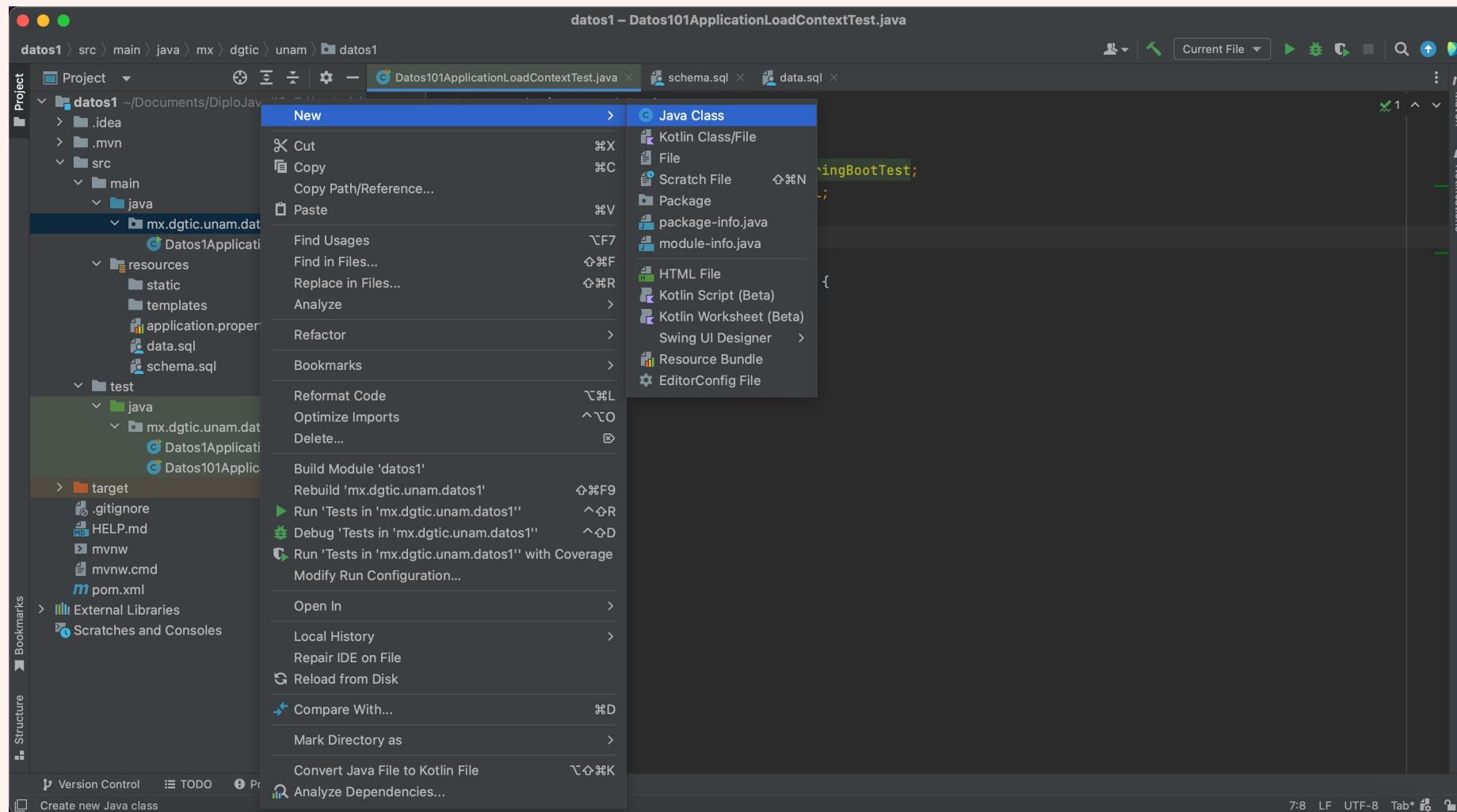
fnac

estatura

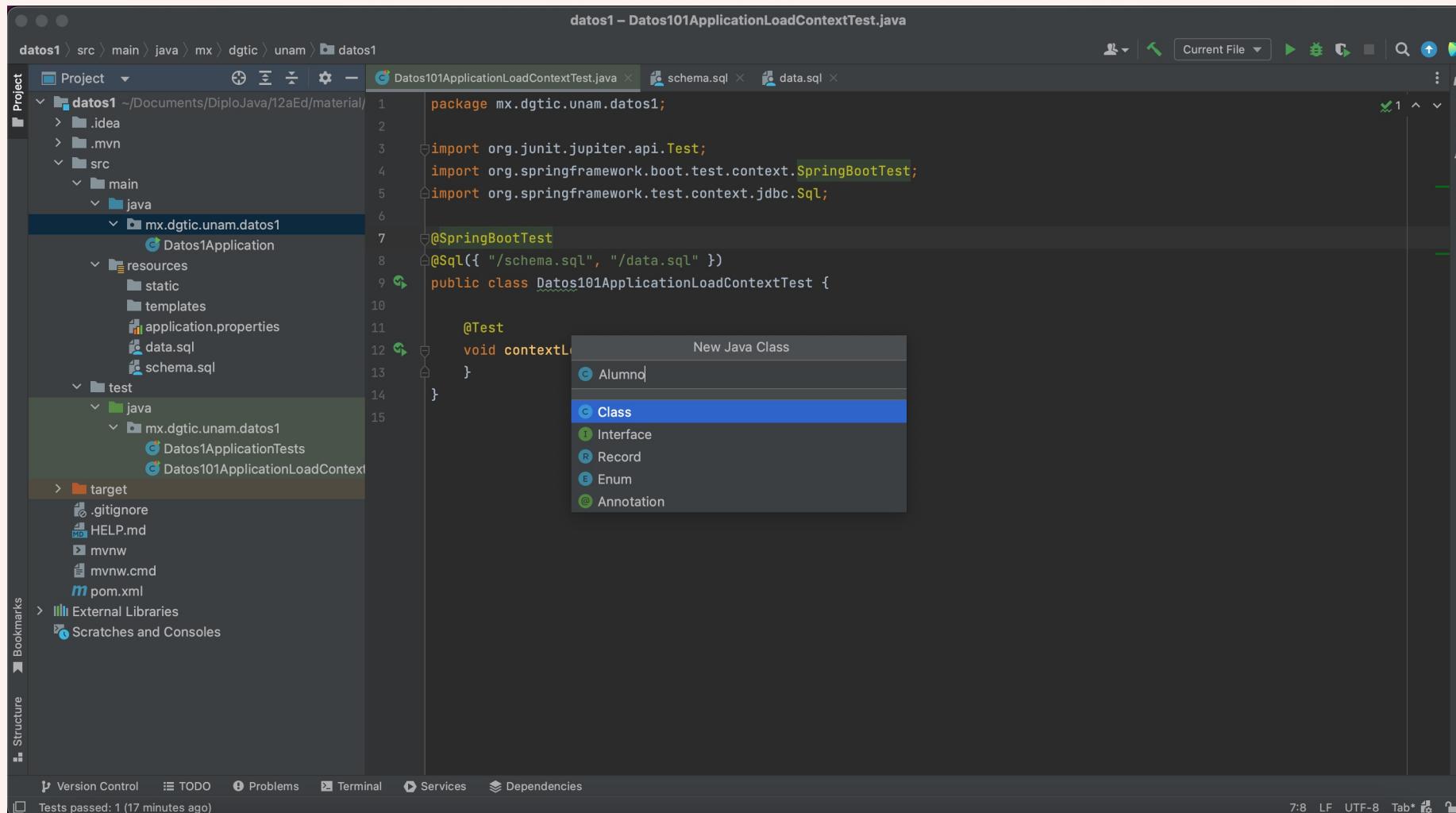
Crear el paquete datos



Generar una entidad



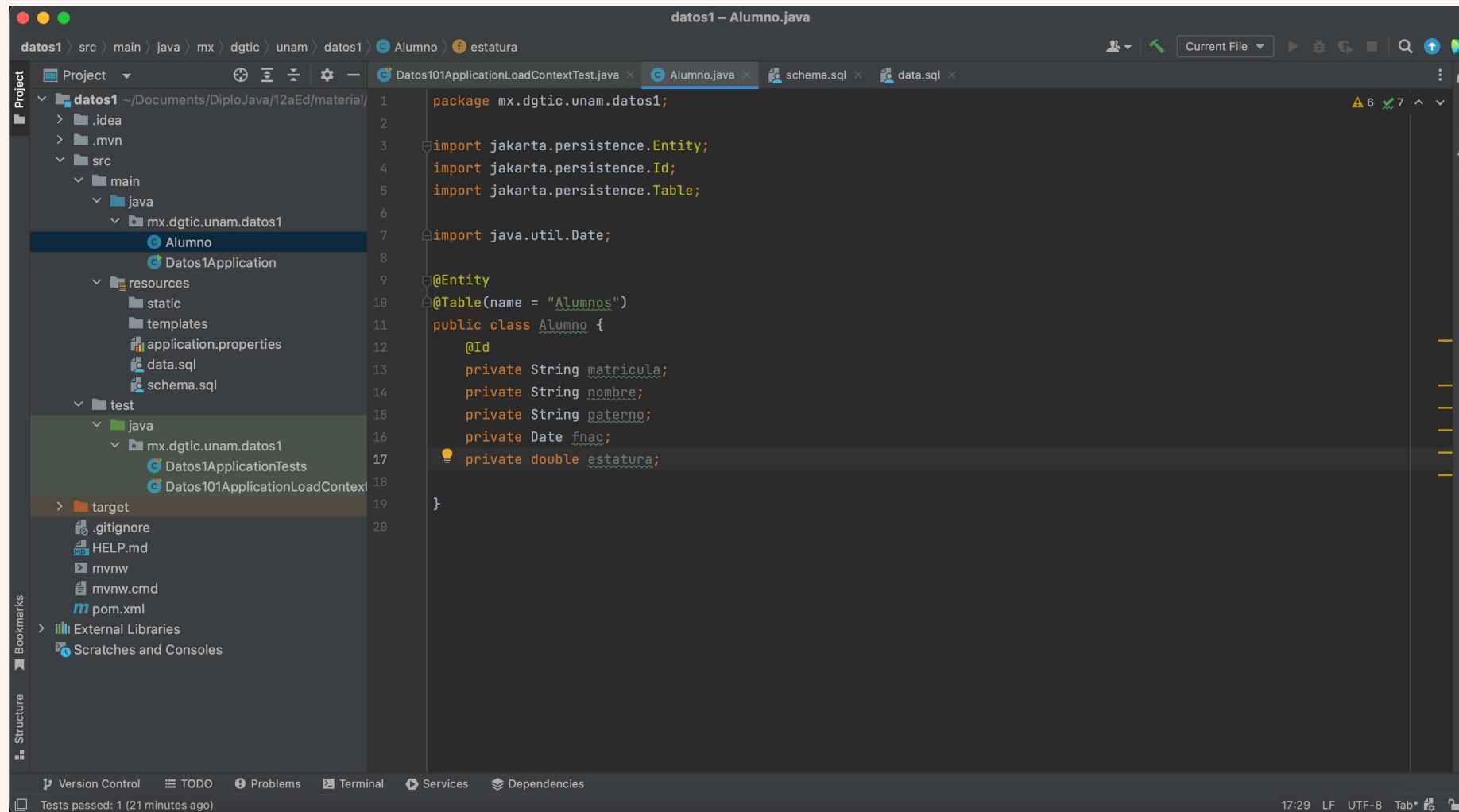
Generar una entidad



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" packages. "main" has a "java" package containing "mx.dgtic.unam.datos1" which includes "Datos1Application" and "Datos1ApplicationLoadContextTest.java". "test" has a "java" package containing "mx.dgtic.unam.datos1" which includes "Datos1ApplicationTests" and "Datos1ApplicationLoadContextTest.java".
- Code Editor:** The file "Datos1ApplicationLoadContextTest.java" is open. The cursor is at the end of the class definition, and a code completion dropdown is visible. The dropdown shows options: "New Java Class", "Alumno", "Class" (which is selected), "Interface", "Record", "Enum", and "@ Annotation".
- Toolbars and Status Bar:** The top bar shows tabs for "Datos1ApplicationLoadContextTest.java", "schema.sql", and "data.sql". The bottom status bar shows "Tests passed: 1 (17 minutes ago)", "7:8 LF UTF-8 Tab*", and other system information.

@Entity @Table



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "java" sub-directories. The "java" directory contains packages "mx.dgtic.unam.datos1" and "mx.dgtic.unam.datos1.Alumno". There are also "resources", "test", and "target" directories.
- Code Editor:** The file "Alumno.java" is open. The code defines a class "Alumno" with annotations "@Entity" and "@Table(name = "Alumnos")". It includes fields for matrícula, nombre, paterno, fnac, and estatura.
- Toolbars and Status Bar:** The status bar at the bottom shows "Tests passed: 1 (21 minutes ago)".
- Right Panel:** Shows Maven and Notifications sections.

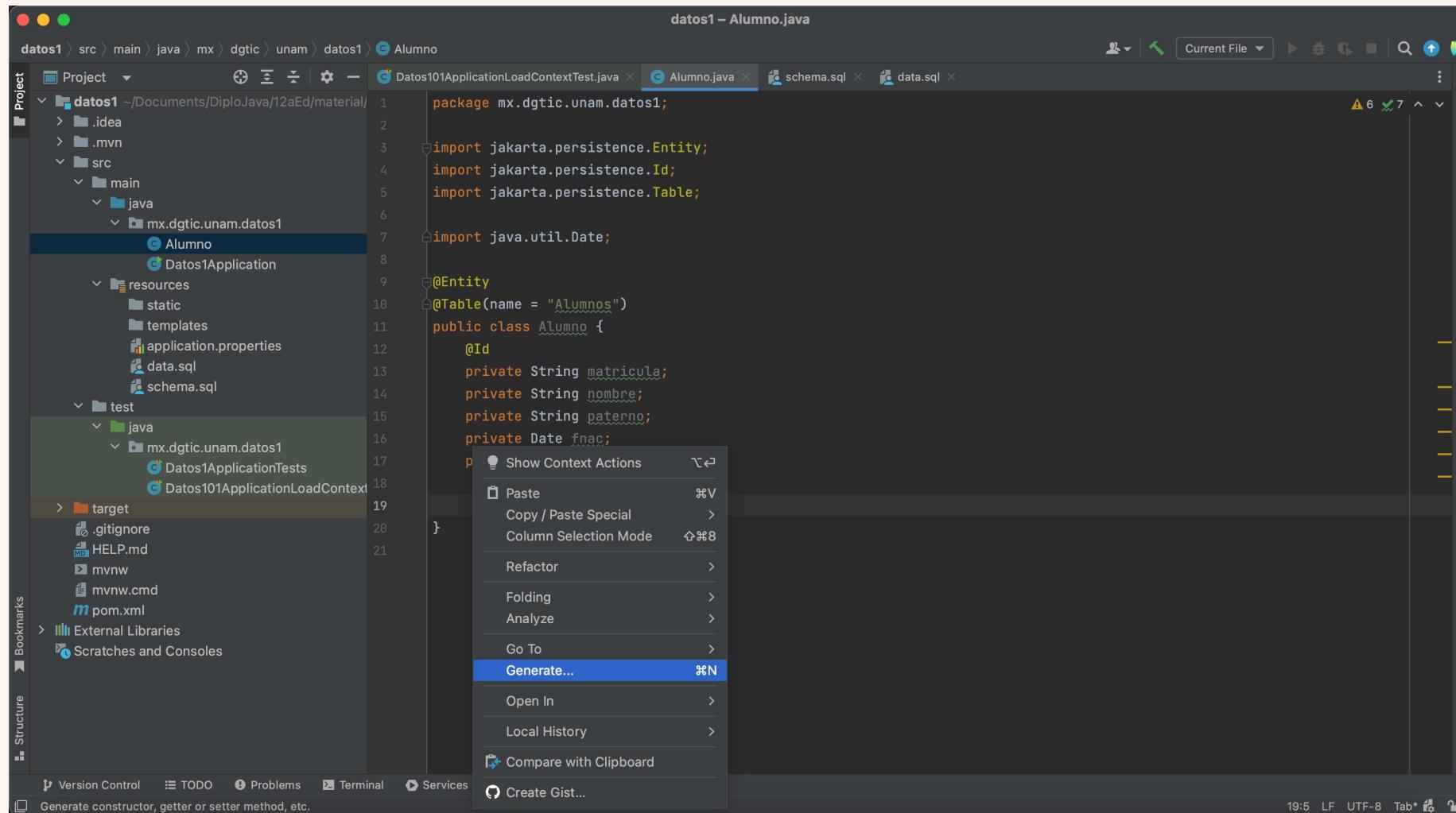
```
datos1 - Alumno.java
package mx.dgtic.unam.datos1;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

import java.util.Date;

@Entity
@Table(name = "Alumnos")
public class Alumno {
    @Id
    private String matricula;
    private String nombre;
    private String paterno;
    private Date fnac;
    private double estatura;
}
```

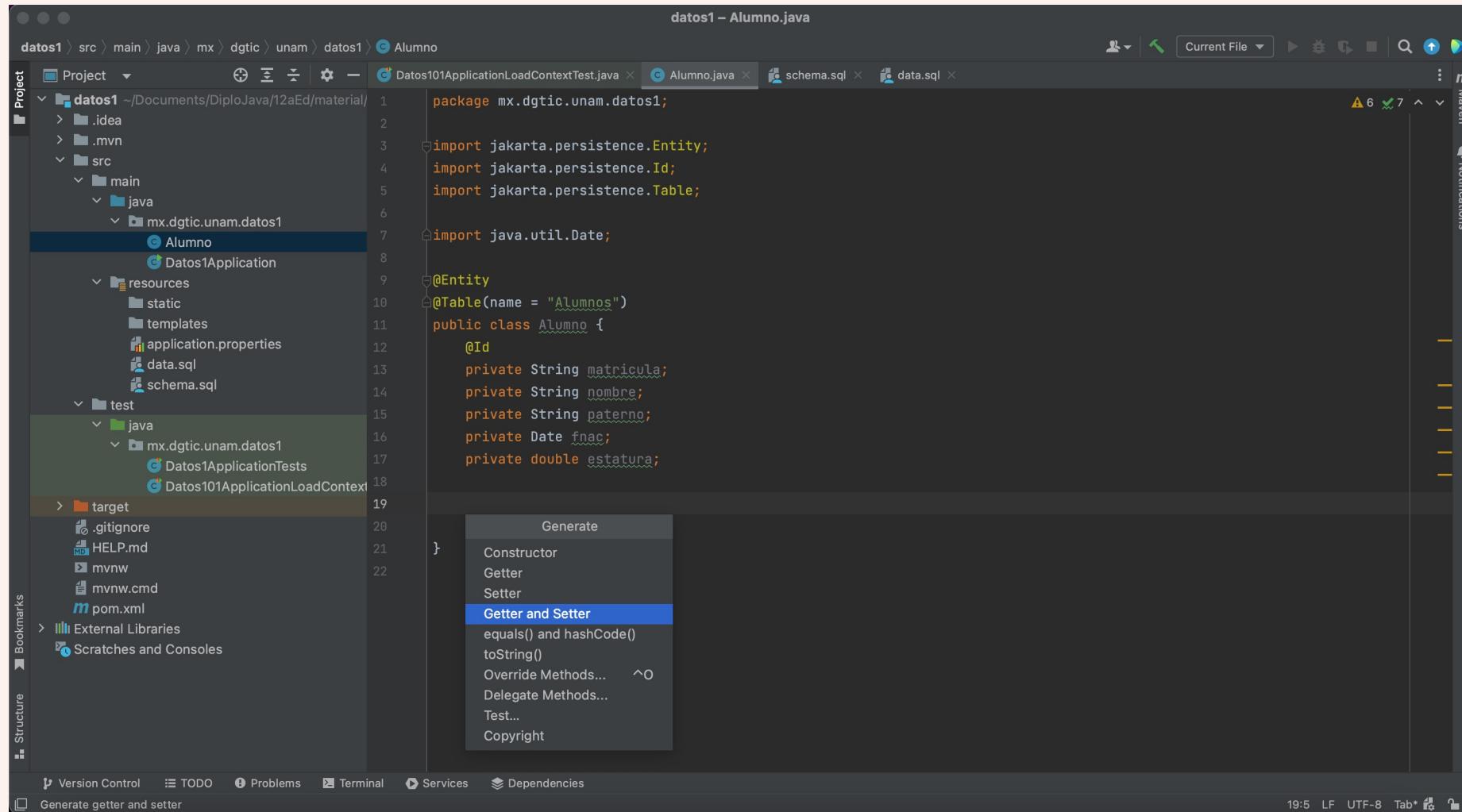
Generar Setters y Getters



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" sub-directories. "main/java" contains packages "mx.dgtic.unam.datos1" (with classes "Alumno" and "Datos1Application") and "mx.dgtic.unam.datos1" (with test classes "Datos1ApplicationTests" and "Datos101ApplicationLoadContext").
- Code Editor:** The file "Alumno.java" is open. The code defines a class "Alumno" with annotations "@Entity" and "@Table(name = "Alumnos")". It has four private fields: "matricula", "nombre", "paterno", and "fnac".
- Context Menu:** A context menu is open over the closing brace of the "Alumno" class definition. The menu items include "Show Context Actions", "Paste", "Copy / Paste Special", "Column Selection Mode", "Refactor", "Folding", "Analyze", "Go To", "Generate...", "Open In", "Local History", "Compare with Clipboard", and "Create Gist...". The "Generate..." option is highlighted.
- Toolbar:** The toolbar at the bottom includes icons for Version Control, TODO, Problems, Terminal, and Services.
- Status Bar:** The status bar at the bottom right shows the time as 19:5 LF UTF-8 Tab*.

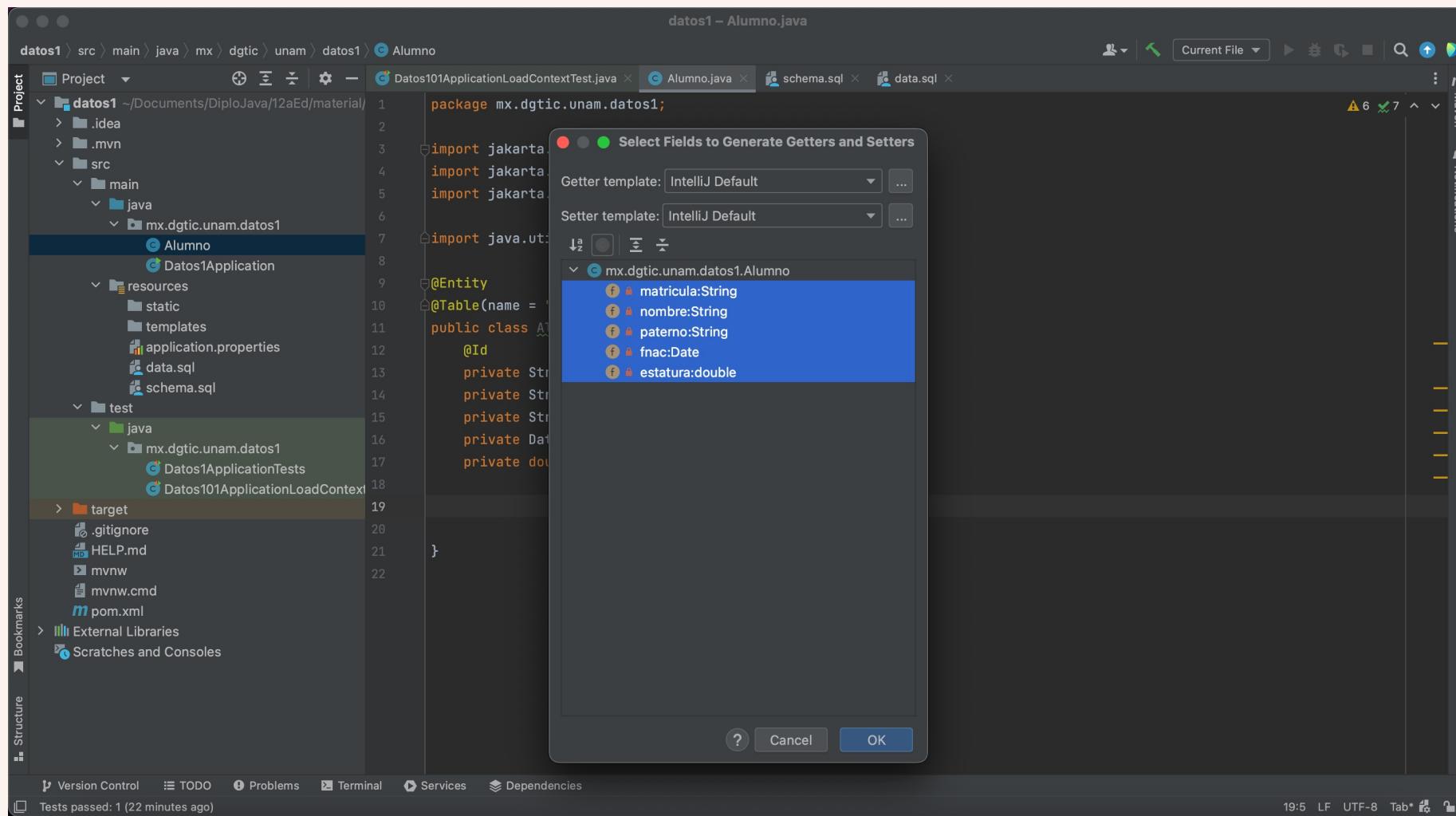
Generar Setters y Getters



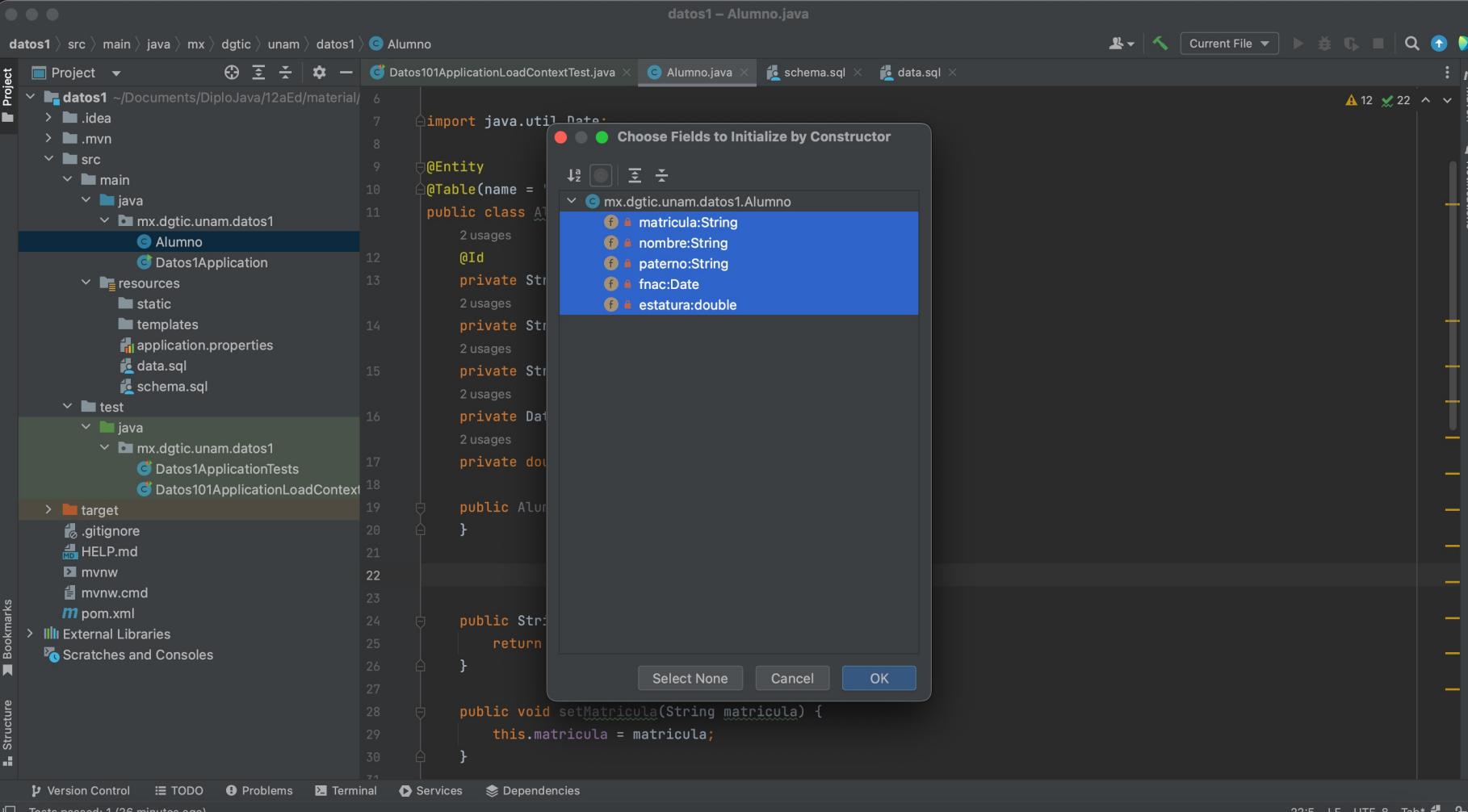
The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" packages. The "main/java" package contains "mx.dgtic.unam.datos1" which has "Alumno" and "Datos1Application" classes. The "test/java" package also contains "mx.dgtic.unam.datos1" with "Datos1ApplicationTests" and "Datos101ApplicationLoadContext" classes.
- Code Editor:** The file "Alumno.java" is open. The code defines a class "Alumno" with annotations "@Entity" and "@Table(name = "Alumnos")". It includes fields for matrícula, nombre, paterno, fnac, and estatura, each annotated with "@Id".
- Toolbars and Status Bar:** The status bar at the bottom shows "19:5 LF UTF-8 Tab*".
- Context Menu:** A context menu is open over the closing brace of the "Alumno" class definition. The menu items include "Generate", "Constructor", "Getter", "Setter", "Getter and Setter" (which is highlighted in blue), "equals() and hashCode()", "toString()", "Override Methods...", "Delegate Methods...", "Test...", and "Copyright".

Generar Setters y Getters



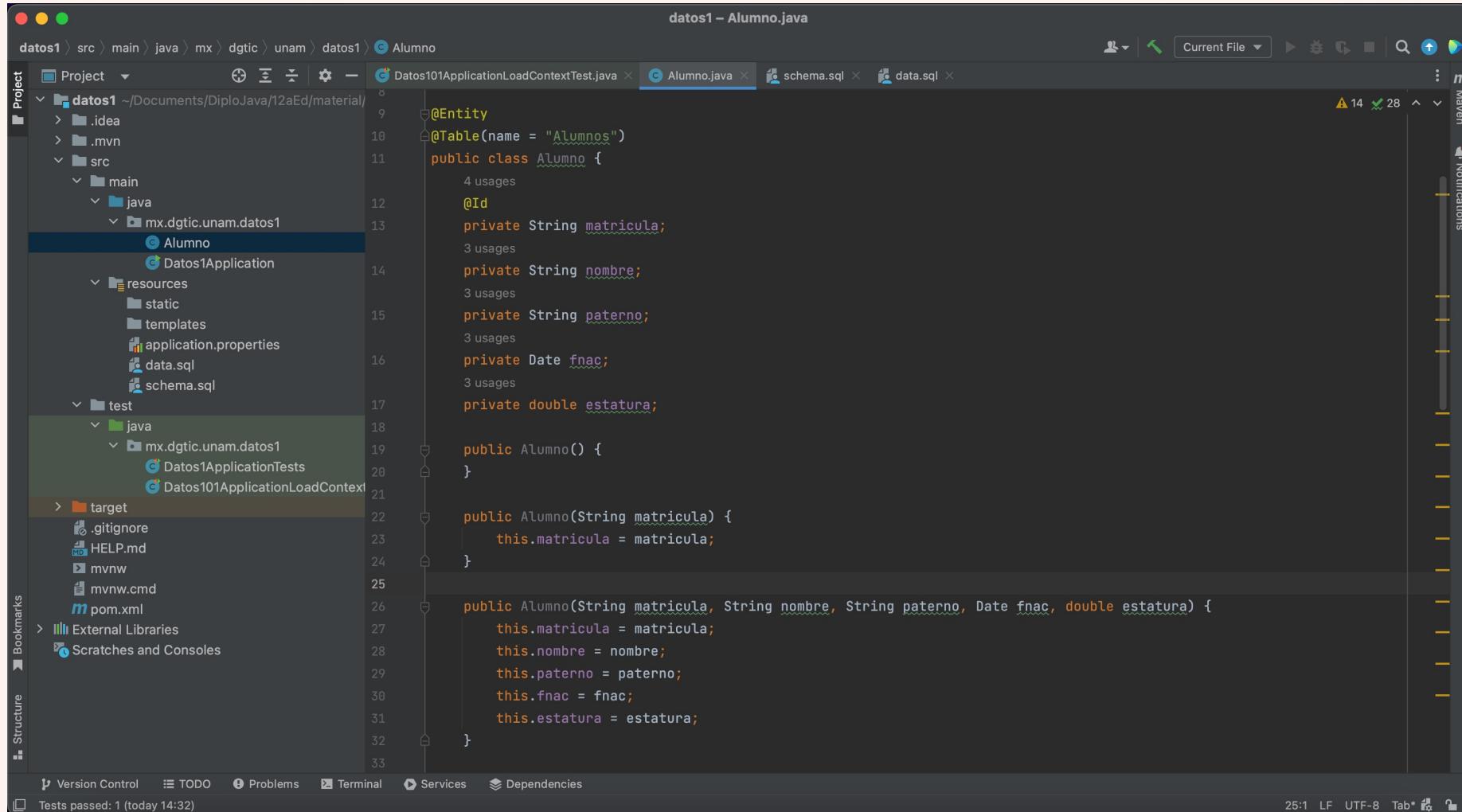
Generar constructores



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "datos1".
- Code Editor:** Displays the `Alumno.java` file content.
- Tool Window:** Shows tabs for `Datos101ApplicationLoadContextTest.java`, `Alumno.java`, `schema.sql`, and `data.sql`.
- Right Panel:** Shows build errors (12 warnings, 22 errors) and notifications.
- Central Dialog:** A modal dialog titled "Choose Fields to Initialize by Constructor" lists fields for the `Alumno` class:
 - `matricula`: String
 - `nombre`: String
 - `paterno`: String
 - `fnac`: Date
 - `estatura`: double
- Bottom Status Bar:** Shows "Tests passed: 1 (26 minutes ago)" and system information like "22:5 LF UTF-8 Tab*".

Generar constructores



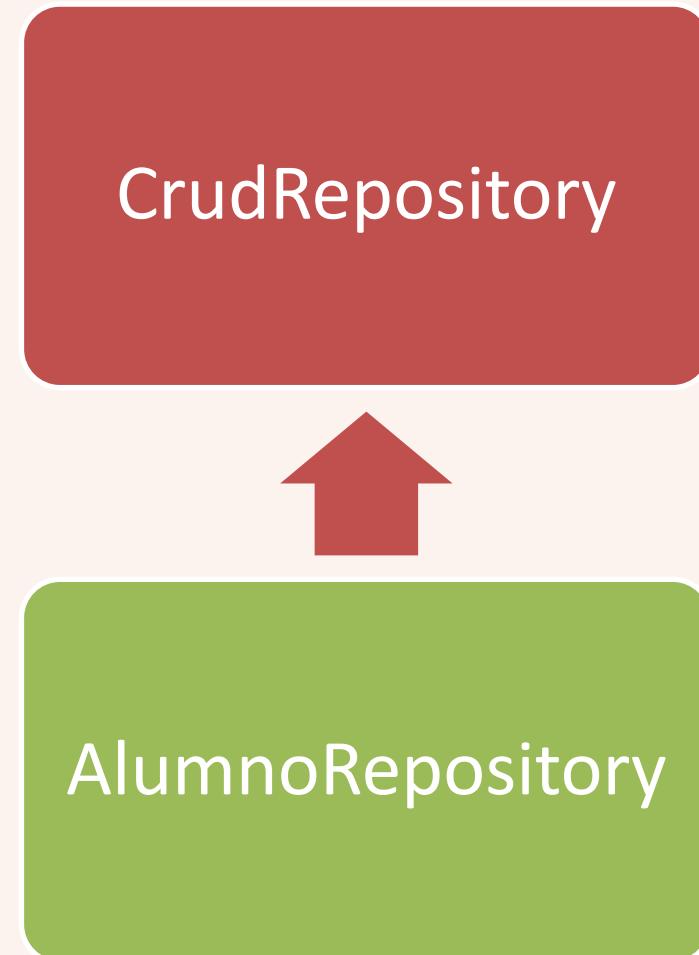
The screenshot shows the IntelliJ IDEA interface with the project 'datos1' open. The 'Alumno.java' file is the active tab, displaying Java code for an entity class. The code includes annotations for Entity and Table, and defines several private fields: matricula, nombre, paterno, fnac, and estatura. Two constructors are generated: one taking a single String parameter and another taking five parameters (String, String, String, Date, double). The code editor shows syntax highlighting and code completion. The left sidebar displays the project structure, and the bottom navigation bar shows tabs for Version Control, TODO, Problems, Terminal, Services, and Dependencies.

```
datos1 – Alumno.java
Project Current File ▾
Alumno.java
schema.sql
data.sql
Maven
Notifications
@Entity
@Table(name = "Alumnos")
public class Alumno {
    @Id
    private String matricula;
    private String nombre;
    private String paterno;
    private Date fnac;
    private double estatura;
    public Alumno() {
    }
    public Alumno(String matricula) {
        this.matricula = matricula;
    }
    public Alumno(String matricula, String nombre, String paterno, Date fnac, double estatura) {
        this.matricula = matricula;
        this.nombre = nombre;
        this.paterno = paterno;
        this.fnac = fnac;
        this.estatura = estatura;
    }
}
```

Repositorio



Repositorio



Spring Data

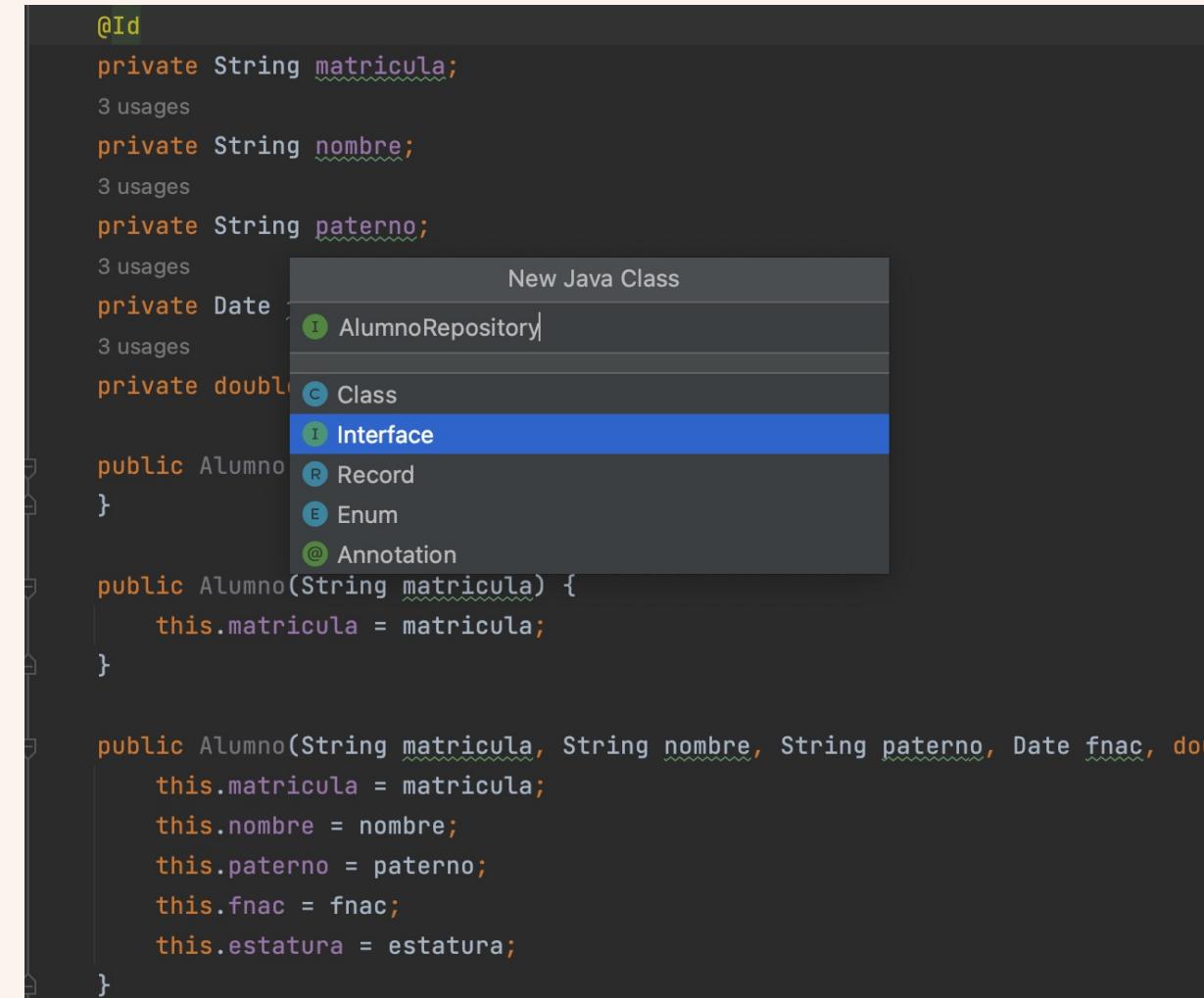
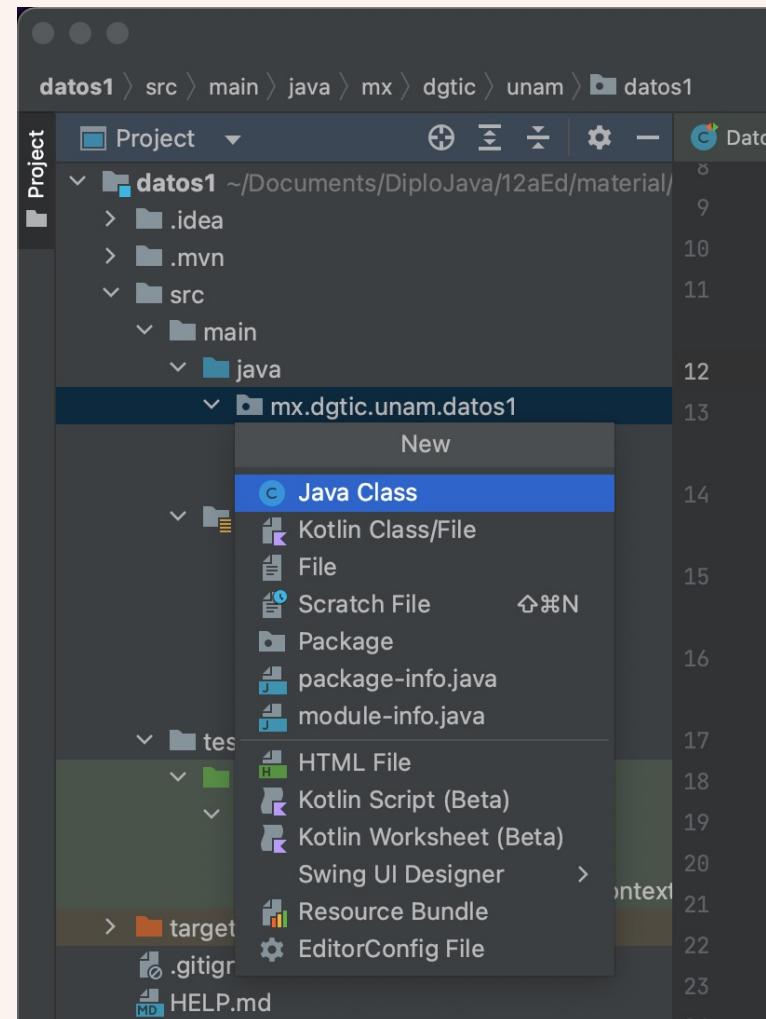
CrudRepository

```
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

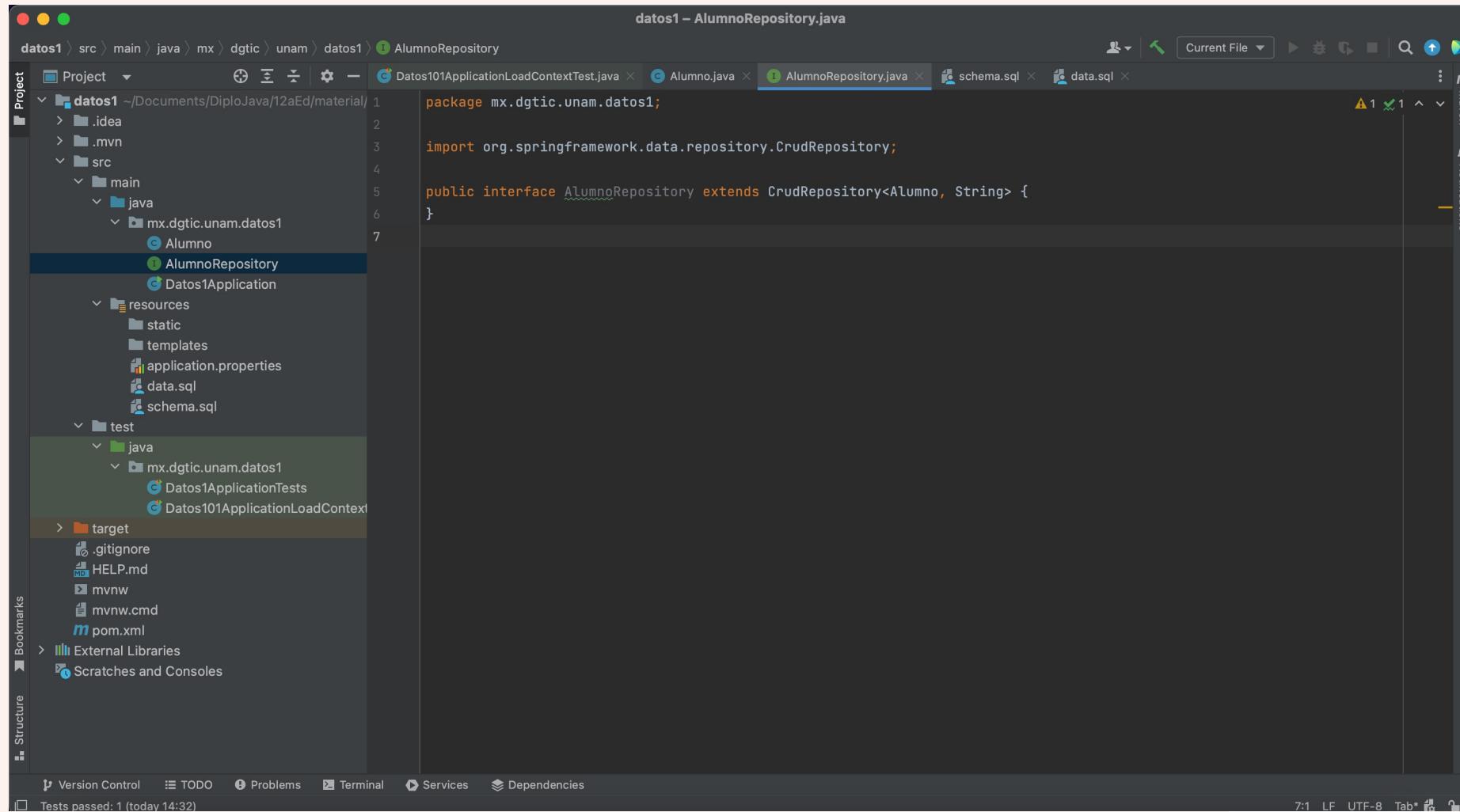


The screenshot shows the IntelliJ IDEA code editor with the following Java code:

```
@Id
private String matricula;
3 usages
private String nombre;
3 usages
private String paterno;
3 usages
private Date fnac;
3 usages
private double estatura;
public Alumno(String matricula) {
    this.matricula = matricula;
}
public Alumno(String matricula, String nombre, String paterno, Date fnac, dou
```

A context menu is open at the end of the first constructor's parameter list ('String matricula'). The 'Interface' option is selected and highlighted in blue. Other options in the menu include 'Class', 'Record', 'Enum', and '@Annotation'. The code editor shows the beginning of an 'AlumnoRepository' interface definition.

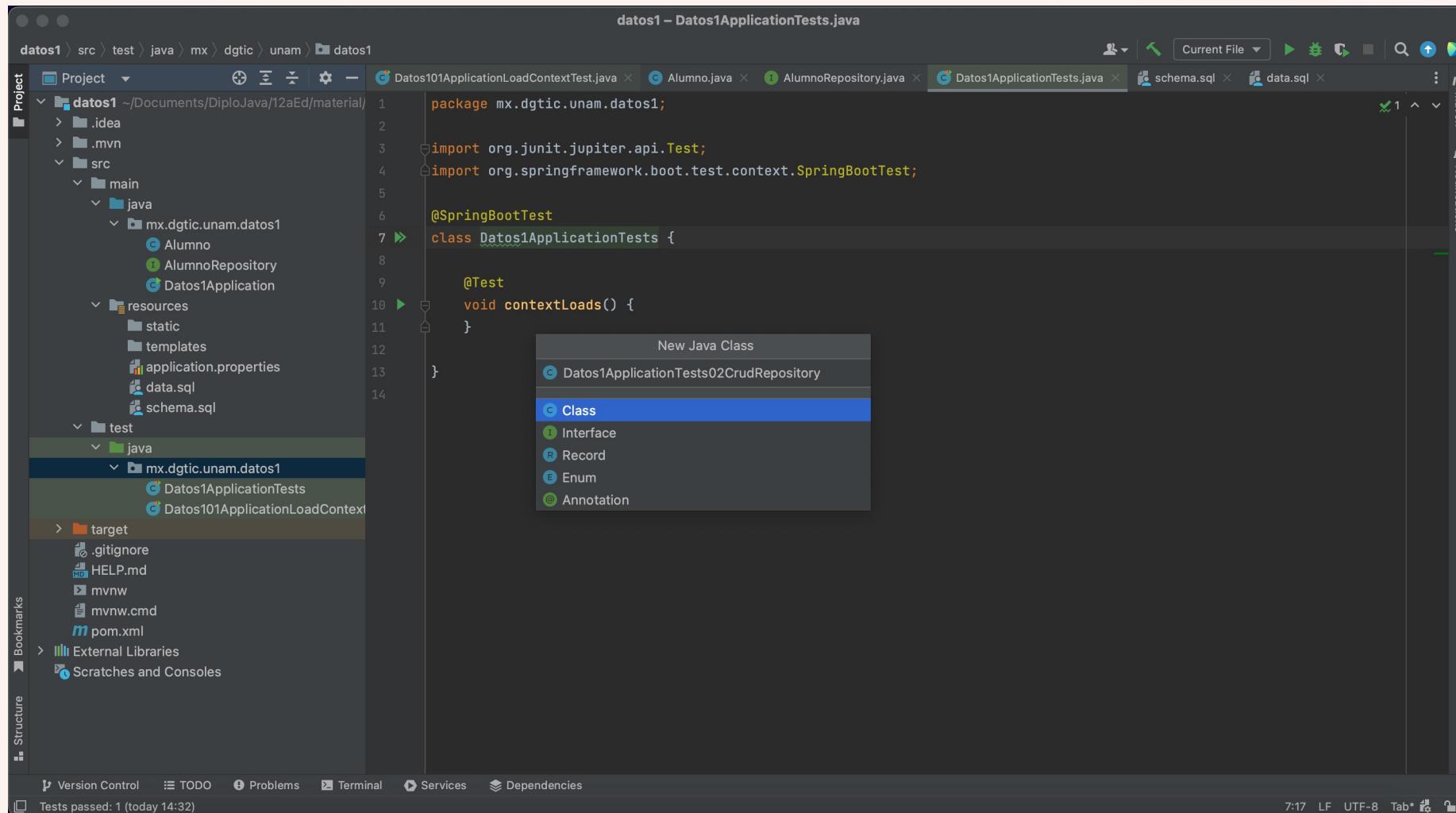
extends CrudRepository



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" packages. The "main/java/mx/dgtic/unam/datos1" package contains "Alumno" and "AlumnoRepository" classes, and "Datos1Application" which has a main method. The "test/java/mx/dgtic/unam/datos1" package contains "Datos1ApplicationTests" and "Datos101ApplicationLoadContext".
- Code Editor:** The file "AlumnoRepository.java" is open. The code defines an interface "AlumnoRepository" that extends "CrudRepository<Alumno, String>".
- Toolbars and Status Bar:** The top bar shows tabs for "Datos101ApplicationLoadContextTest.java", "Alumno.java", "AlumnoRepository.java" (which is the active tab), and "schema.sql", "data.sql". The status bar at the bottom shows "Tests passed: 1 (today 14:32)" and "7:1 LF UTF-8 Tab*".

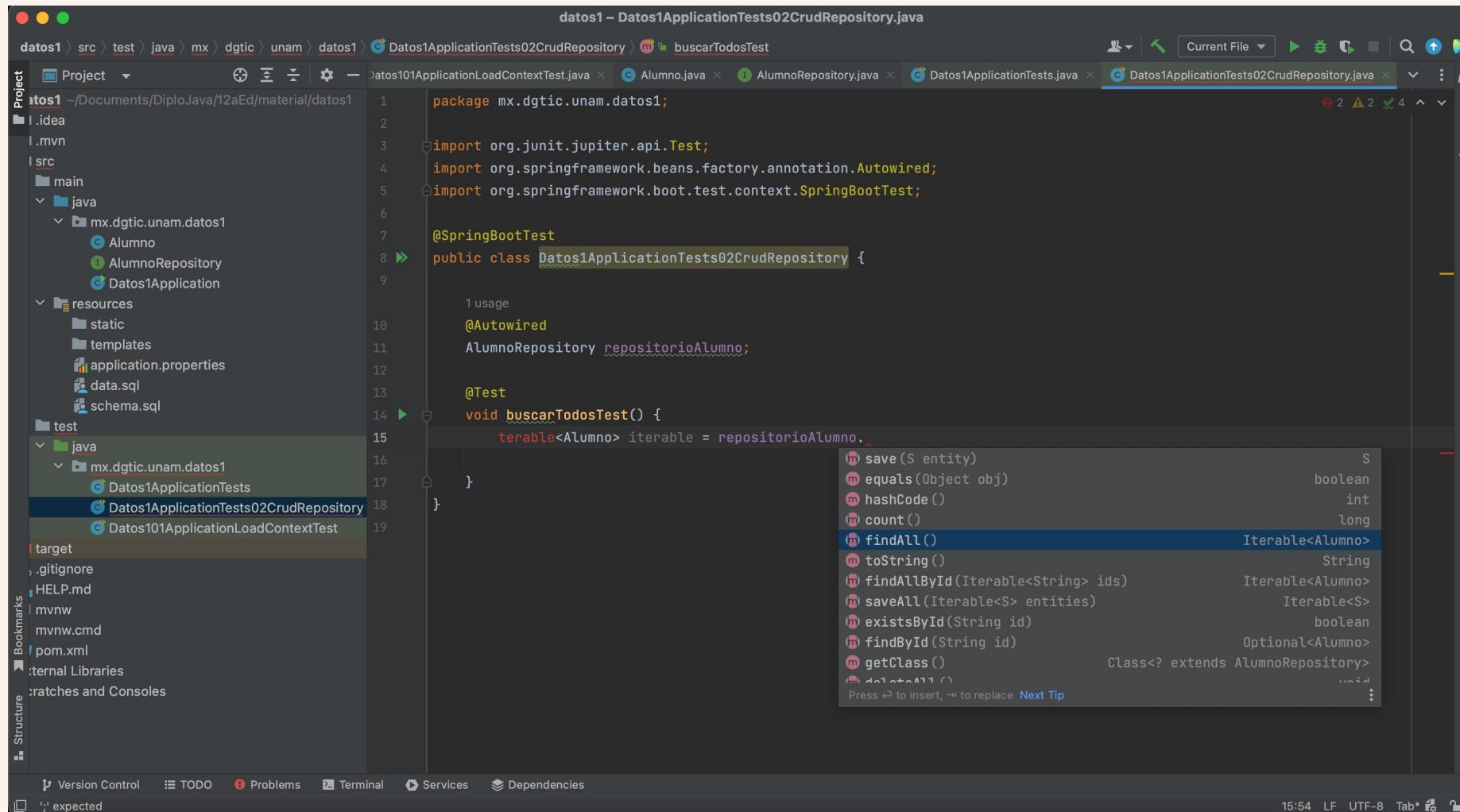
Agregar una nueva JUnit



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" sub-directories. Under "main/java", there is a package "mx.dgtic.unam.datos1" containing classes "Alumno", "AlumnoRepository", and "Datos1Application". Under "test/java", there is a package "mx.dgtic.unam.datos1" containing classes "Datos1ApplicationTests" and "Datos101ApplicationLoadContextTest".
- Code Editor:** The file "datos1 - Datos1ApplicationTests.java" is open. The code imports org.junit.jupiter.api.Test and org.springframework.boot.test.context.SpringBootTest. It defines a class "Datos1ApplicationTests" with a single test method "contextLoads()".
- Tool Window:** A "New Java Class" dialog is displayed, listing options: "Datos1ApplicationTests02CrudRepository", "Class" (which is selected), "Interface", "Record", "Enum", and "Annotation".
- Bottom Bar:** Shows tabs for "Version Control", "TODO", "Problems", "Terminal", "Services", and "Dependencies". It also displays the message "Tests passed: 1 (today 14:32)".

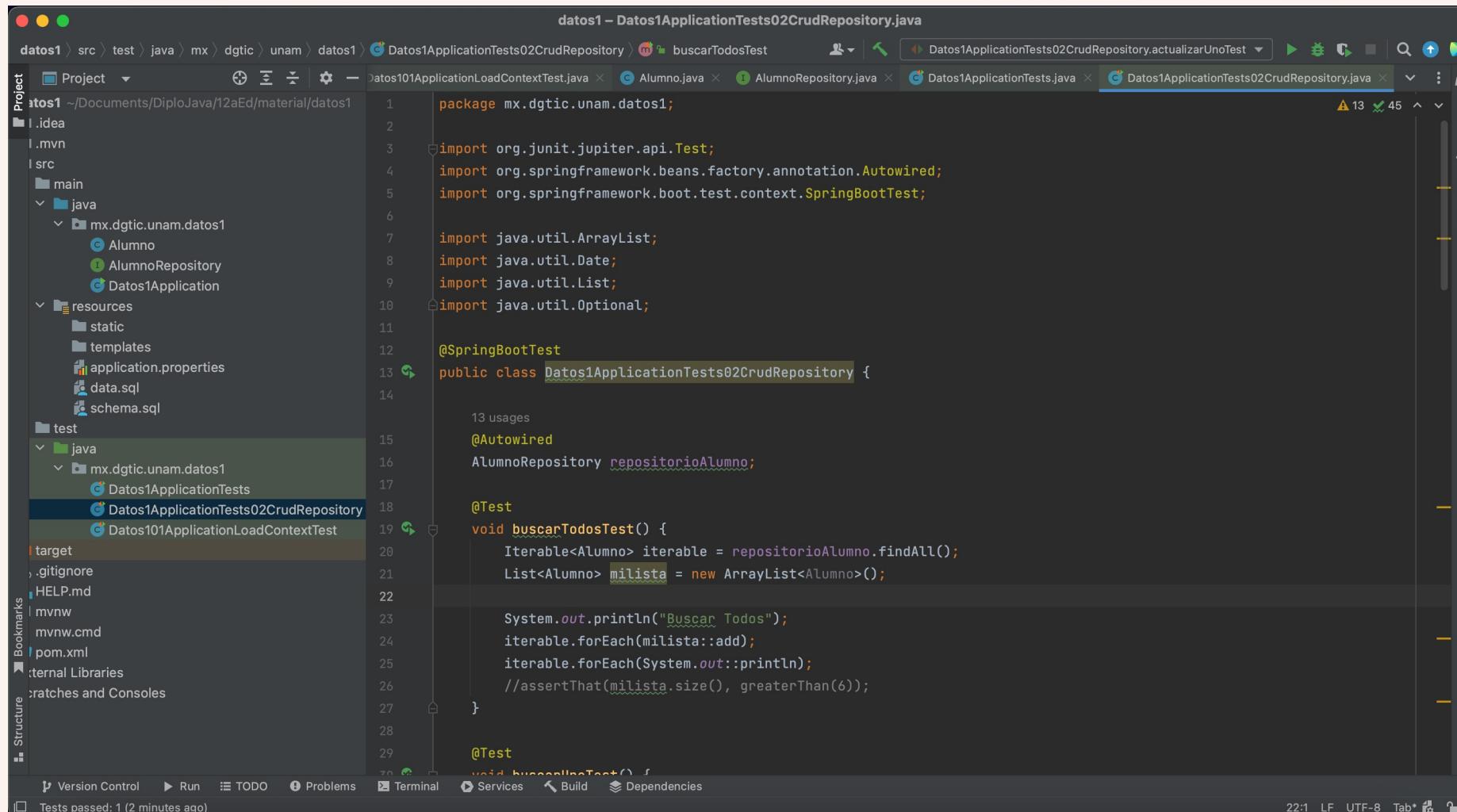
@Autowired



The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" packages. "main" contains "mx.dgtic.unam.datos1" which includes "Alumno" and "AlumnoRepository" classes, and "Datos1Application". "test" contains "java" with "mx.dgtic.unam.datos1" which includes "Datos1ApplicationTests", "Datos1ApplicationTests02CrudRepository" (the current file), and "Datos101ApplicationLoadContextTest".
- Code Editor:** The current file is "Datos1ApplicationTests02CrudRepository.java". The code defines a test class "Datos1ApplicationTests02CrudRepository" with a method "buscarTodosTest". The cursor is at the end of the line "Iterable<Alumno> iterable = repositorioAlumno.". A code completion dropdown is open, listing various methods of the Iterable interface, such as "save(S entity)", "equals(Object obj)", "hashCode()", "count()", "findAll()", "toString()", "findAllById(Iterable<String> ids)", "saveAll(Iterable<S> entities)", "existsById(String id)", "findById(String id)", "getClass()", and "getOne()".
- Toolbars and Status Bar:** The top bar shows tabs for "Current File" (highlighted), "Datos1ApplicationTests02CrudRepository.java", "Alumno.java", "AlumnoRepository.java", "Datos1ApplicationTests.java", and "Datos1ApplicationTests02CrudRepository.java". The status bar at the bottom right shows the time as "15:54", the encoding as "UTF-8", and the tab count as "Tab*".

Probar findAll()



The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" sub-directories. The "test" directory contains "java" and "resources" sub-directories. The "java" directory under "test" contains classes for "Alumno", "AlumnoRepository", "Datos1Application", and "Datos1ApplicationTests02CrudRepository".
- Code Editor:** The current file is "Datos1ApplicationTests02CrudRepository.java". The code defines a test class "Datos1ApplicationTests02CrudRepository" that includes methods for testing the "findAll()" method of the "AlumnoRepository".
- Toolbars and Status Bar:** The status bar at the bottom shows "Tests passed: 1 (2 minutes ago)" and the system time as "22:11".
- Right Panel:** The "Notifications" panel shows 45 notifications.

```
package mx.dgtic.unam.datos1;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Optional;

@SpringBootTest
public class Datos1ApplicationTests02CrudRepository {

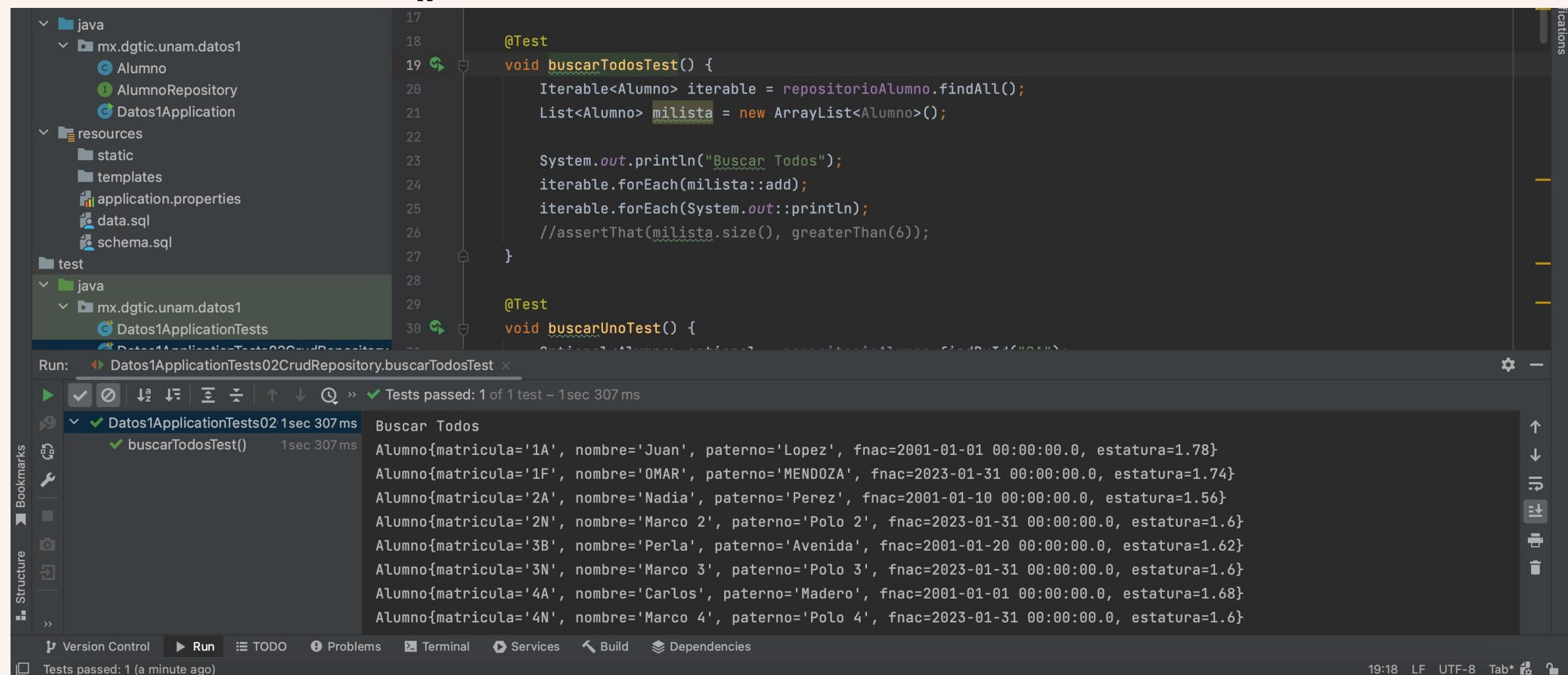
    @Autowired
    AlumnoRepository repositorioAlumno;

    @Test
    void buscarTodosTest() {
        Iterable<Alumno> iterable = repositorioAlumno.findAll();
        List<Alumno> milista = new ArrayList<Alumno>();

        System.out.println("Buscar Todos");
        iterable.forEach(milista::add);
        iterable.forEach(System.out::println);
        //assertThat(milista.size(), greaterThan(6));
    }

    @Test
    void buscarAlunoTest() {
}
```

Probar findAll()



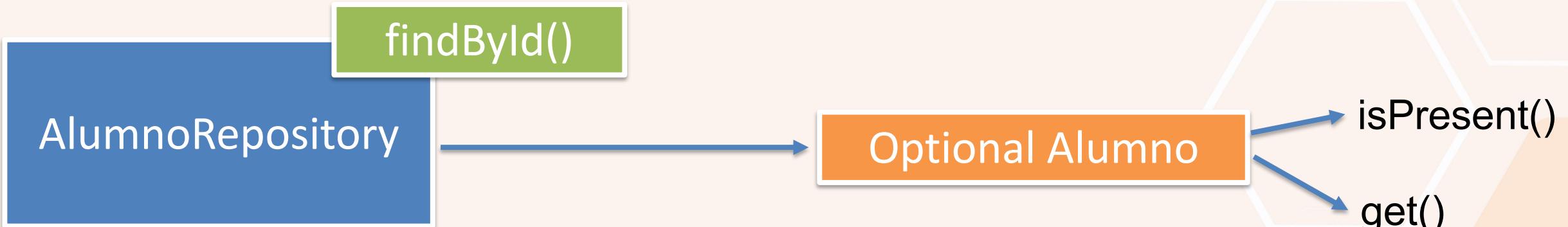
The screenshot shows an IDE interface with the following details:

- Project Structure:** The project structure on the left includes a `java` folder containing `mx.dgtic.unam.datos1` (with `Alumno`, `AlumnoRepository`, and `Datos1Application`), `resources` (with `static`, `templates`, `application.properties`, `data.sql`, and `schema.sql`), and a `test` folder containing `java` (with `mx.dgtic.unam.datos1` and `Datos1ApplicationTests`).
- Code Editor:** The main editor window displays Java test code for `AlumnoRepository`. It contains two test methods: `buscarTodosTest()` and `buscarUnoTest()`.
- Run Tab:** The "Run" tab at the bottom shows the result of the run: "Tests passed: 1 of 1 test – 1sec 307 ms".
- Terminal Output:** The terminal pane shows the output of the `buscarTodosTest()` method, which prints a list of `Alumno` objects to the console.
- Bottom Navigation:** The bottom navigation bar includes tabs for Version Control, Run, TODO, Problems, Terminal, Services, Build, and Dependencies.

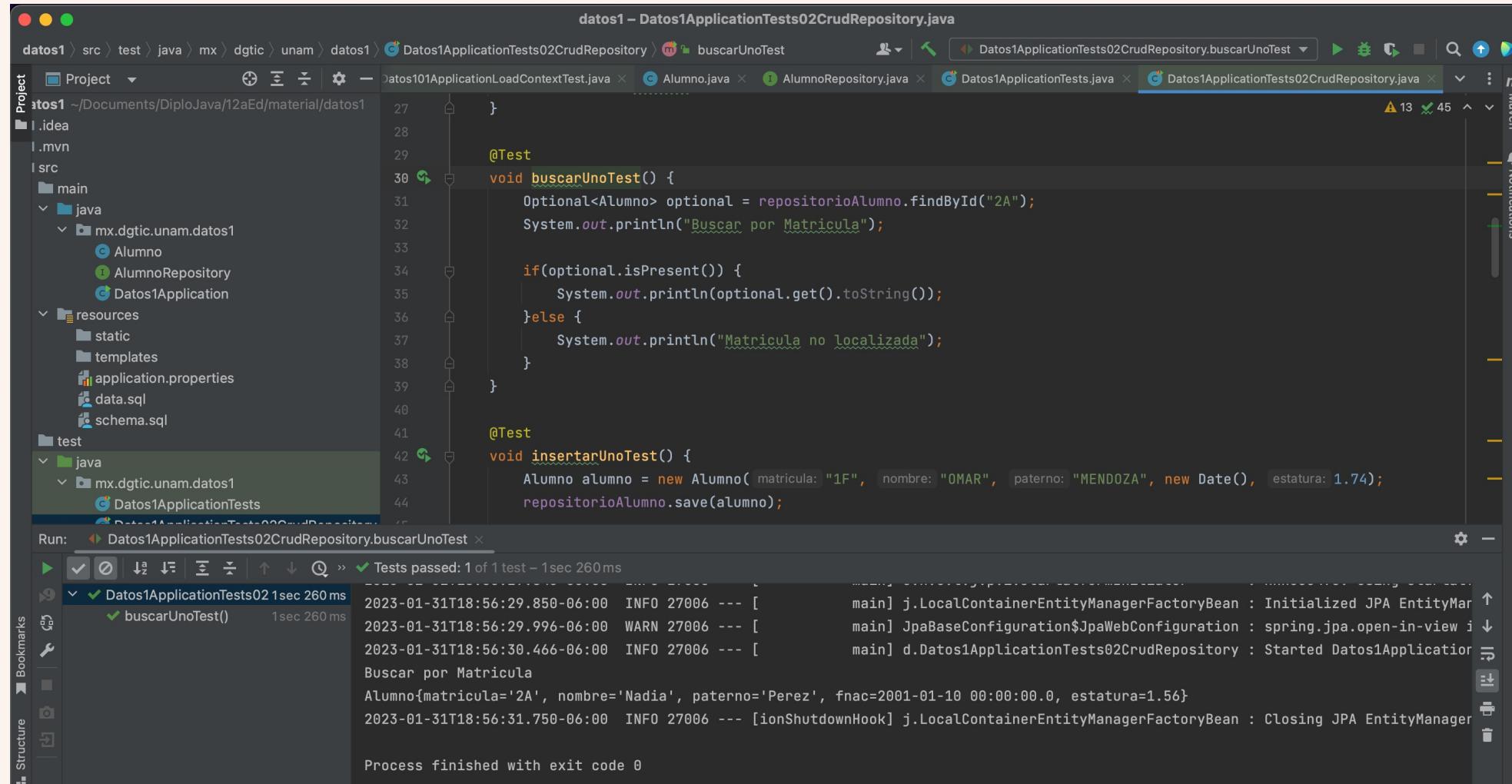
```
17
18
19  @Test
20  void buscarTodosTest() {
21      Iterable<Alumno> iterable = repositorioAlumno.findAll();
22      List<Alumno> milista = new ArrayList<Alumno>();
23
24      System.out.println("Buscar Todos");
25      iterable.forEach(milista::add);
26      iterable.forEach(System.out::println);
27  }
28
29  @Test
30  void buscarUnoTest() {
```

```
Buscar Todos
Alumno{matricula='1A', nombre='Juan', paterno='Lopez', fnac=2001-01-01 00:00:00.0, estatura=1.78}
Alumno{matricula='1F', nombre='OMAR', paterno='MENDOZA', fnac=2023-01-31 00:00:00.0, estatura=1.74}
Alumno{matricula='2A', nombre='Nadia', paterno='Perez', fnac=2001-01-10 00:00:00.0, estatura=1.56}
Alumno{matricula='2N', nombre='Marco 2', paterno='Polo 2', fnac=2023-01-31 00:00:00.0, estatura=1.6}
Alumno{matricula='3B', nombre='Perla', paterno='Avenida', fnac=2001-01-20 00:00:00.0, estatura=1.62}
Alumno{matricula='3N', nombre='Marco 3', paterno='Polo 3', fnac=2023-01-31 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='4N', nombre='Marco 4', paterno='Polo 4', fnac=2023-01-31 00:00:00.0, estatura=1.6}
```

Probar findById()



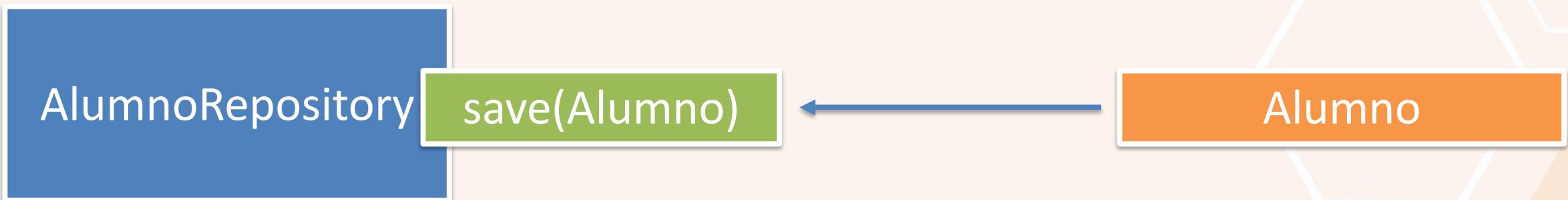
Probar findById()



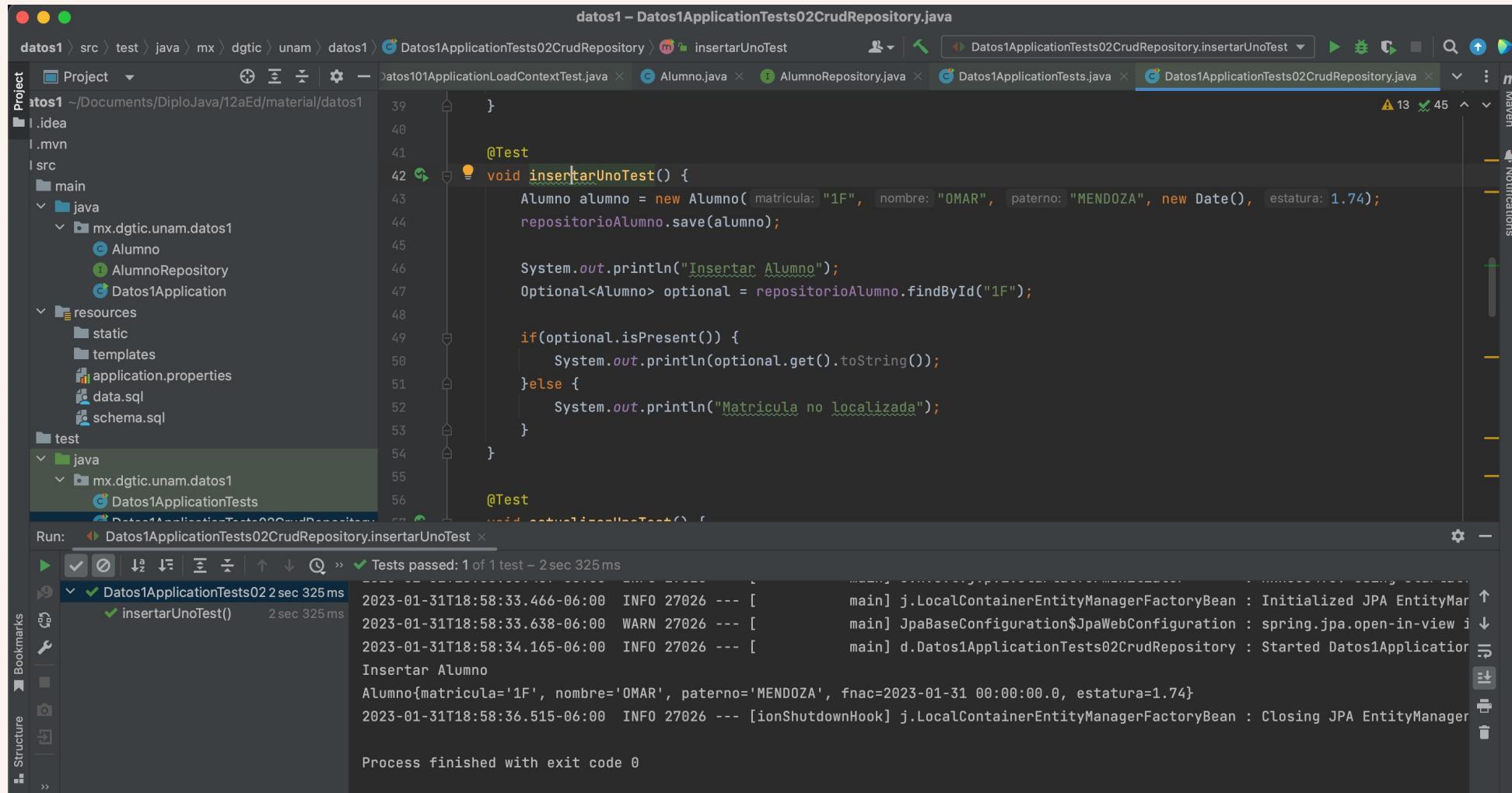
The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" packages. The "main" package has "Alumno" and "AlumnoRepository" classes, and a "Datos1Application" class. The "test" package has a "java" folder containing "mx.dgtic.unam.datos1.Datos1ApplicationTests" and a "Resources" folder with static, templates, application.properties, data.sql, and schema.sql files.
- Code Editor:** The file "Datos1ApplicationTests02CrudRepository.java" is open. It contains two test methods: "buscarUnoTest()" and "insertarUnoTest()". The "buscarUnoTest()" method uses the "findById" method from the repository to find an Alumno by matricula "2A" and prints the result if present, or a message if not found. The "insertarUnoTest()" method creates a new Alumno object and saves it to the repository.
- Run Tab:** The "Run" tab shows the run configuration "Datos1ApplicationTests02CrudRepository.buscarUnoTest" has passed, taking 1 sec 260 ms. The log output shows the successful execution of the test and the printed result: "Alumno{matricula='2A', nombre='Nadia', paterno='Perez', fnac=2001-01-10 00:00:00.0, estatura=1.56}".
- Bottom Status:** The status bar at the bottom indicates "Process finished with exit code 0".

save()



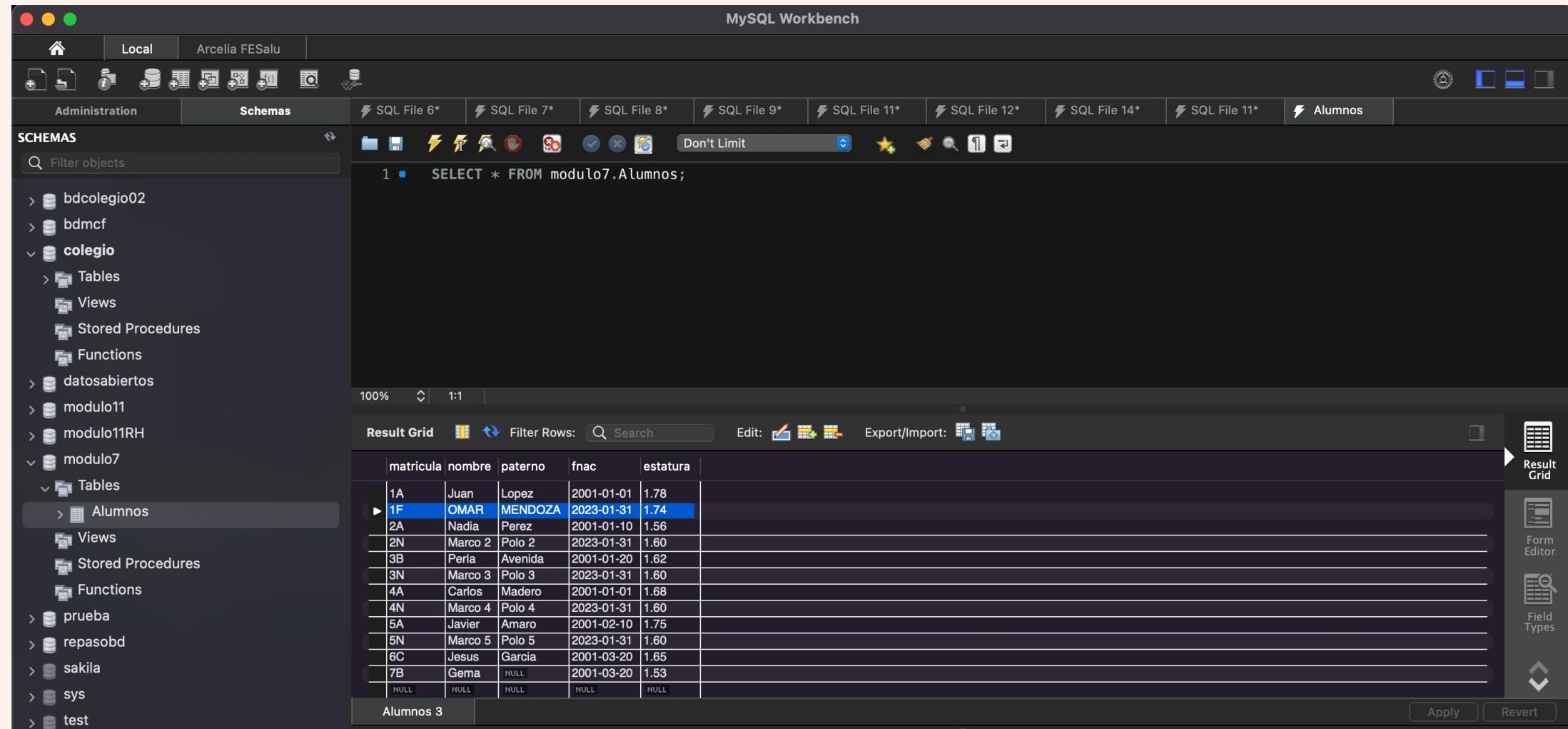
Probar save()



The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" packages. The "main/java/mx.dgtic.unam.datos1" package contains "Alumno", "AlumnoRepository", and "Datos1Application" classes. The "test/java/mx.dgtic.unam.datos1" package contains "Datos1ApplicationTests" and "Datos1ApplicationTests02CrudRepository" classes.
- Code Editor:** The editor displays the "Datos1ApplicationTests02CrudRepository.java" file, specifically the "insertarUnoTest" method. The code creates a new "Alumno" object with matrícula "1F", nombre "OMAR", paterno "MENDOZA", and estatura 1.74, then saves it using the repository's save method. It then prints the inserted alumno and checks if the optional is present.
- Run Tab:** The "Run" tab shows the test has passed: 1 of 1 test - 2 sec 325 ms.
- Output Tab:** The "Output" tab shows the terminal output of the test execution. It includes log entries from the application and repository, the printed alumno object, and the final message "Process finished with exit code 0".

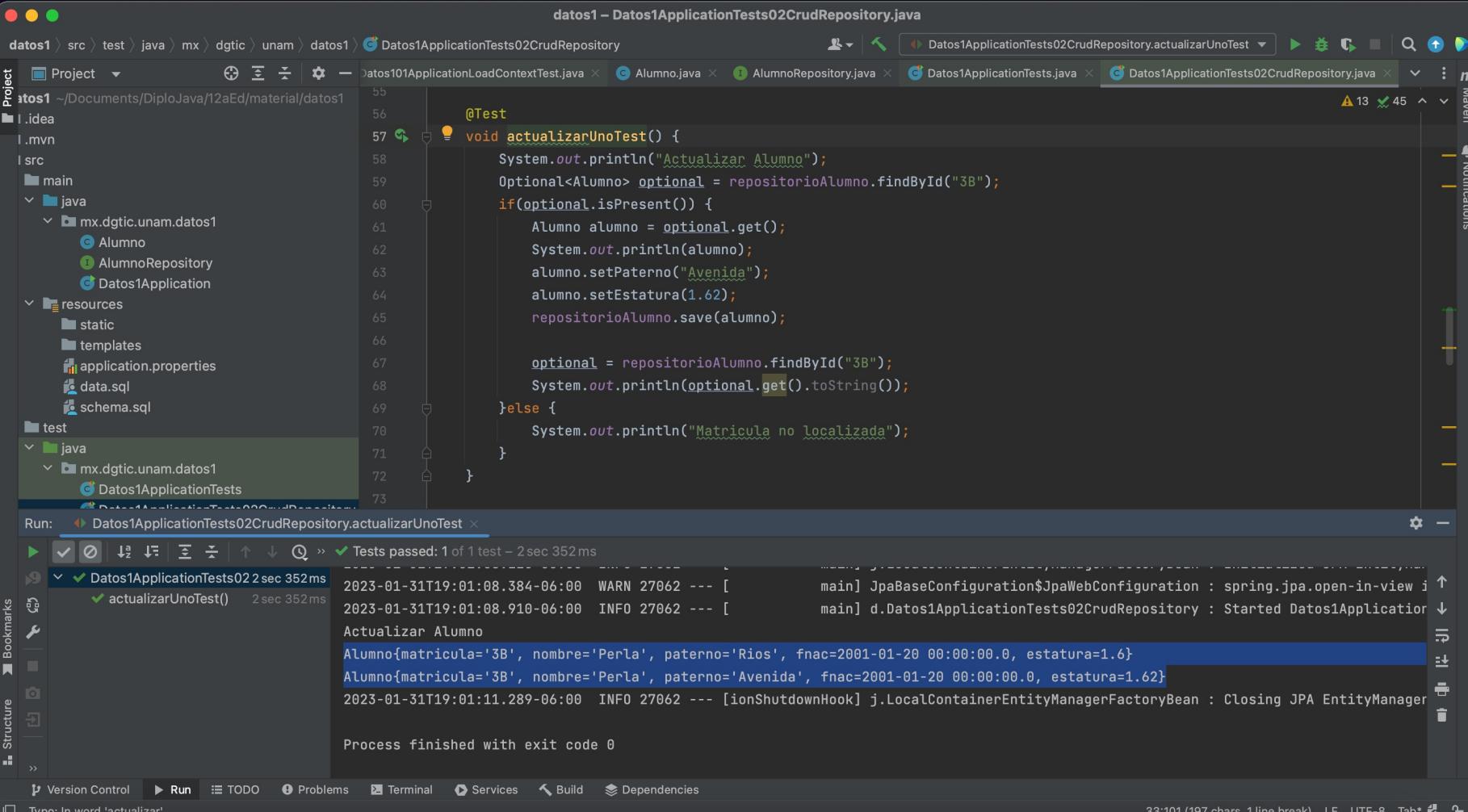
Probar save()



The screenshot shows the MySQL Workbench interface with the following details:

- Title Bar:** MySQL Workbench
- Toolbar:** Includes icons for Home, Local, Schemas, Administration, SQL File 6*, SQL File 7*, SQL File 8*, SQL File 9*, SQL File 11*, SQL File 12*, SQL File 14*, SQL File 11*, and Alumnos.
- Schemas Panel:** Shows available databases: bdcolegio02, bdmcf, colegio, datosabiertos, modulo11, modulo11RH, modulo7. The modulo7 database is selected, and its tables (Alumnos, Views, Stored Procedures, Functions) are listed.
- SQL Editor:** Contains the query: `1 • SELECT * FROM modulo7.Alumnos;`
- Result Grid Tab:** Displays the results of the query in a grid format. The columns are matricula, nombre, paterno, fnac, and estatura. The data includes rows for OMAR MENDOZA (matricula 1F), Nadia Perez (matricula 2A), Marco 2 Polo 2 (matricula 2N), Perla Avenida (matricula 3B), Marco 3 Polo 3 (matricula 3N), Carlos Madero (matricula 4A), Marco 4 Polo 4 (matricula 4N), Javier Amaro (matricula 5A), Marco 5 Polo 5 (matricula 5N), Jesus Garcia (matricula 6C), and Gema (matricula 7B).
- Right Sidebar:** Includes tabs for Result Grid, Form Editor, and Field Types, along with buttons for Apply and Revert.

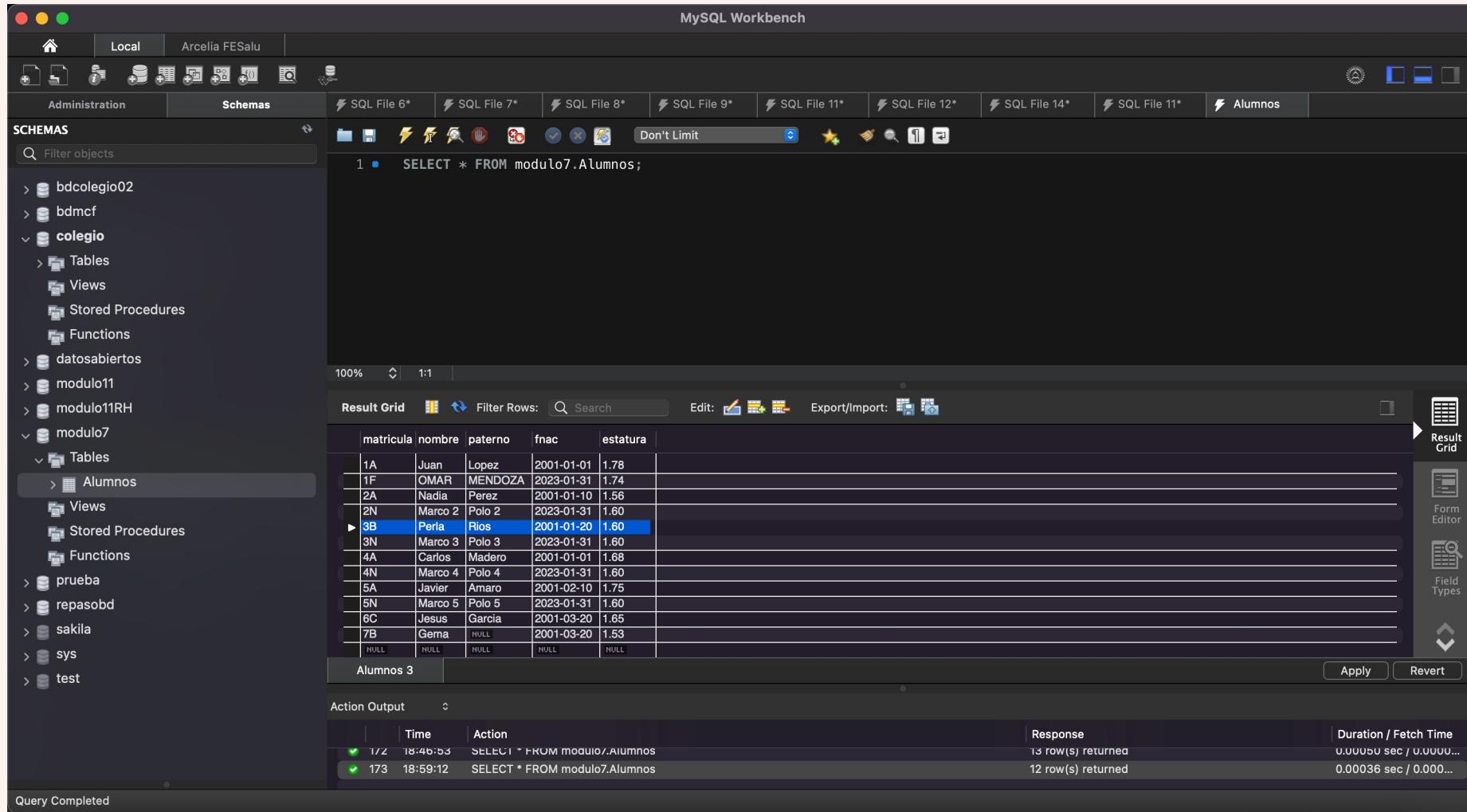
Probar save() actualizar



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "datos1". It contains a "src" directory with "main" and "test" packages. In "main", there are "mx.dgtic.unam.datos1" sub-packages containing "Alumno", "AlumnoRepository", and "Datos1Application" classes. In "test", there is a "java" package with "mx.dgtic.unam.datos1" sub-package containing "Datos1ApplicationTests" class.
- Code Editor:** The file "Datos1ApplicationTests02CrudRepository.java" is open, showing a test method "actualizarUnoTest". The code prints the current student information, updates the student's last name to "Avenida", and then saves the updated student.
- Run Tab:** The "Run" tab shows the test "Datos1ApplicationTests02CrudRepository.actualizarUnoTest" has passed, taking 2 seconds and 352ms.
- Output Tab:** The output window displays log messages and the updated student object: "Alumno{matricula='3B', nombre='Perla', paterno='Rios', fnac=2001-01-20 00:00:00.0, estatura=1.62}" and "Alumno{matricula='3B', nombre='Perla', paterno='Avenida', fnac=2001-01-20 00:00:00.0, estatura=1.62}".
- Bottom Status Bar:** Shows the file has 197 characters, 1 line break, is in UTF-8 encoding, and has tabs enabled.

Probar save() actualizar



The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree view is open, showing various databases like bdcolegio02, bdmcf, colegio, datosabiertos, modulo11, modulo11RH, modulo7, prueba, repasobd, sakila, sys, and test. The 'Alumnos' table under the modulo7 schema is selected. In the main pane, a SQL editor window displays the following query:

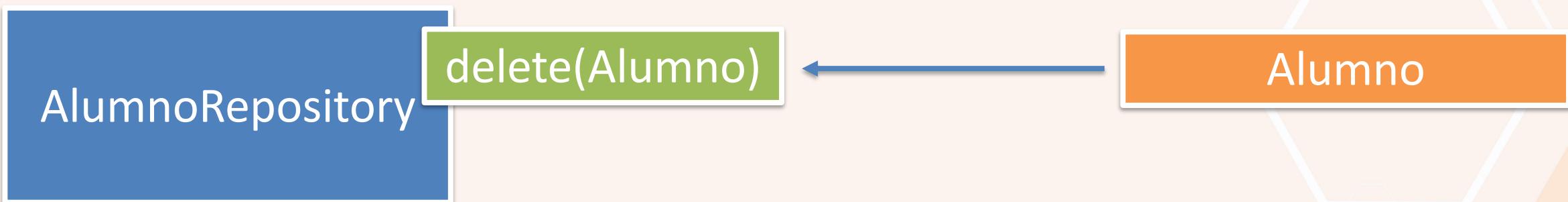
```
1 •   SELECT * FROM modulo7.Alumnos;
```

The result grid shows a list of students. One row, with matricula '3B' and nombre 'Perla', estatura '1.60', and fnac '2001-01-20', is highlighted. The 'Action Output' pane at the bottom shows two log entries:

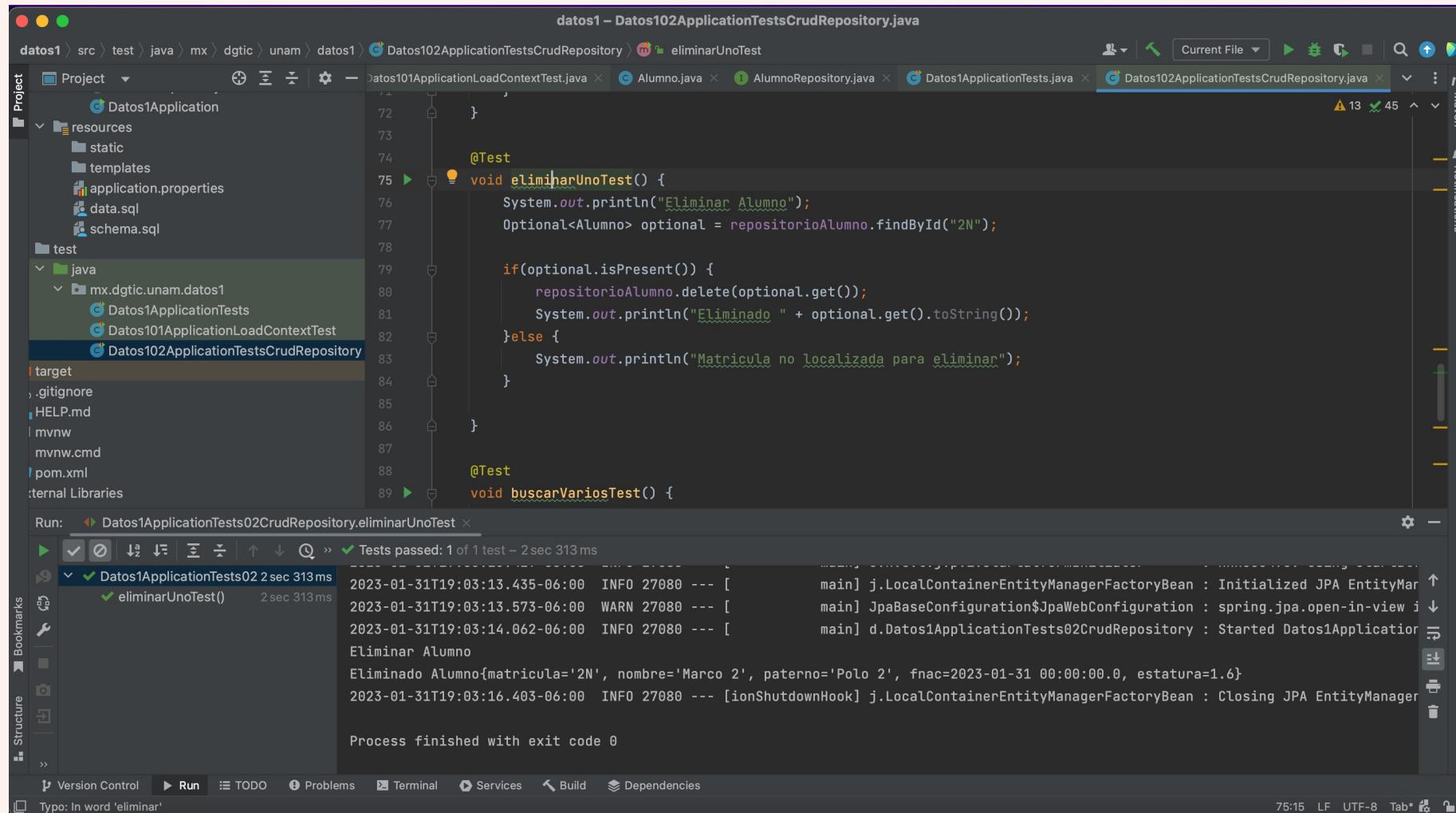
| Time | Action | Response | Duration / Fetch Time |
|----------|-------------------------------|--------------------|------------------------|
| 18:40:53 | SELECT * FROM modulo7.Alumnos | 13 row(s) returned | 0.00050 sec / 0.000... |
| 18:59:12 | SELECT * FROM modulo7.Alumnos | 12 row(s) returned | 0.00036 sec / 0.000... |

The status bar at the bottom indicates 'Query Completed'.

delete()



Probar delete()

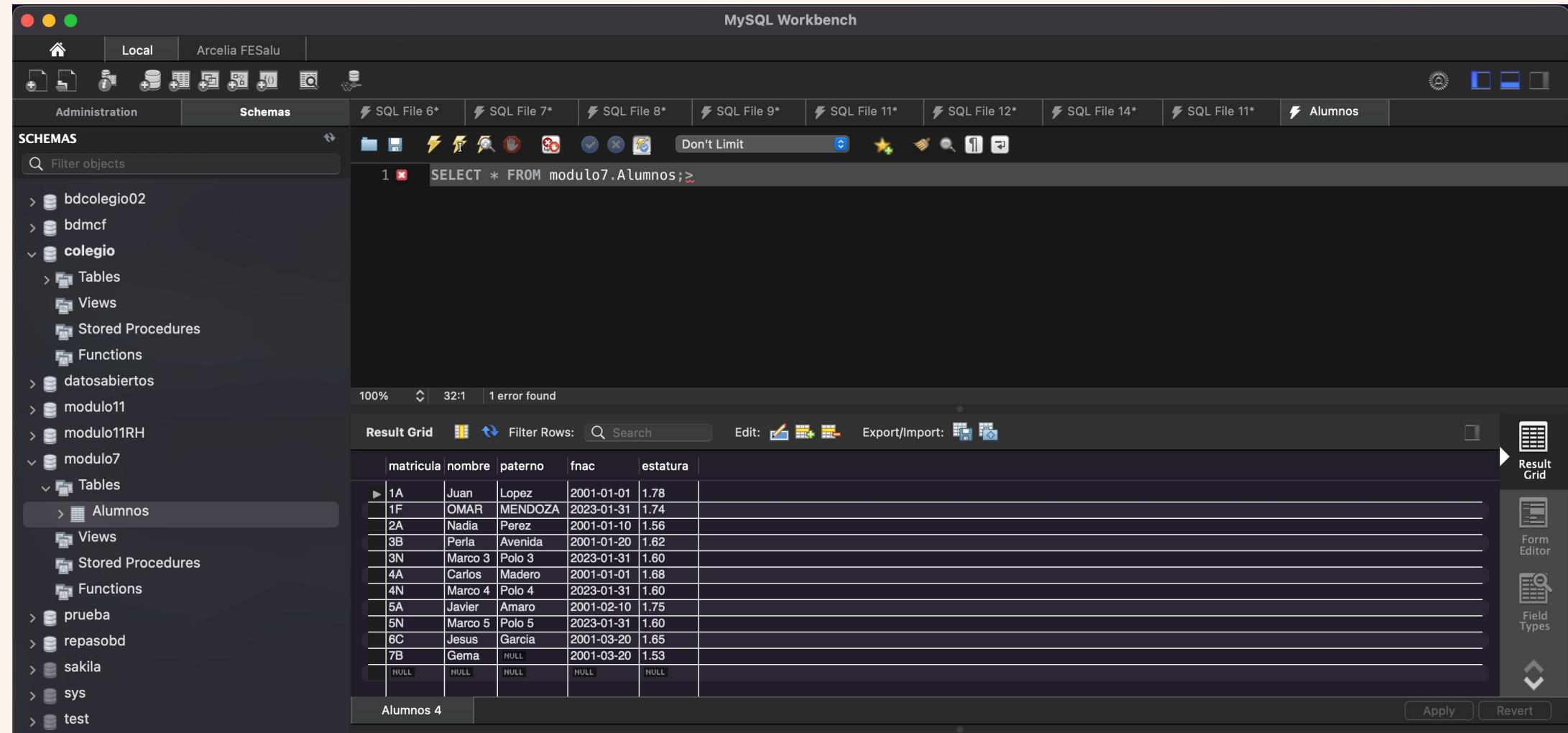


The screenshot shows the IntelliJ IDEA IDE interface with the following details:

- Project Structure:** The project is named "datos1". The "test" directory contains Java classes: "Datos1ApplicationTests", "Datos101ApplicationLoadContextTest", and "Datos102ApplicationTestsCrudRepository".
- Code Editor:** The file "Datos102ApplicationTestsCrudRepository.java" is open, showing a test method "eliminarUnoTest". The code prints "Eliminar Alumno", finds an optional student by matricula "2N", and then deletes it if present.
- Run Tab:** The "Run" tab shows the test "Datos1ApplicationTests02CrudRepository.eliminarUnoTest" has passed (1 of 1 test - 2 sec 313 ms).
- Output Tab:** The "Output" tab displays the console logs:

```
2023-01-31T19:03:13.435-06:00 INFO 27080 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'Default Unit'  
2023-01-31T19:03:13.573-06:00 WARN 27080 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Always set 'true' or 'false' to prevent automatic opening of sessions during requests.  
2023-01-31T19:03:14.062-06:00 INFO 27080 --- [main] d.Datos1ApplicationTests02CrudRepository : Started Datos1ApplicationTests02CrudRepository in 2.01 seconds (j.LocalContainerEntityManagerFactoryBean: 1.99s)  
Eliminar Alumno  
Eliminado Alumno{matricula='2N', nombre='Marco 2', paterno='Polo 2', fnac=2023-01-31 00:00:00.0, estatura=1.6}  
2023-01-31T19:03:16.403-06:00 INFO 27080 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'Default Unit'
```
- Bottom Status Bar:** Shows "Process finished with exit code 0".

Probar delete()



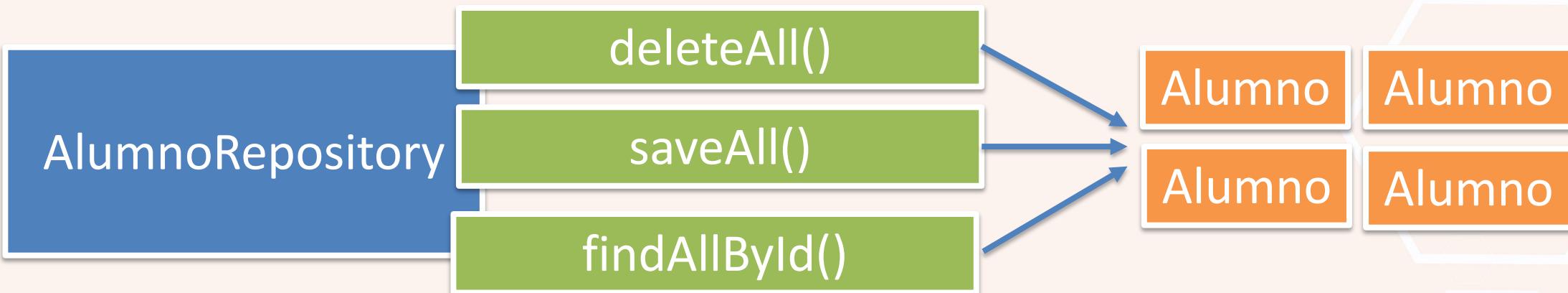
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The left sidebar lists several databases: bdcolegio02, bdmcf, colegio, datosabiertos, modulo11, modulo11RH, modulo7, prueba, repasobd, sakila, sys, and test. The colegio database is expanded, showing Tables, Views, Stored Procedures, and Functions.
- Current Schema:** The current schema is set to 'Arcelia FESalu'.
- SQL Editor:** A query window displays the SQL command: `SELECT * FROM modulo7.Alumnos;`. The status bar indicates "1 error found".
- Result Grid:** The main pane shows the data from the Alumnos table in a grid format. The columns are: matricula, nombre, paterno, fnac, and estatura. The data includes 14 rows of student information.
- Table Data:**

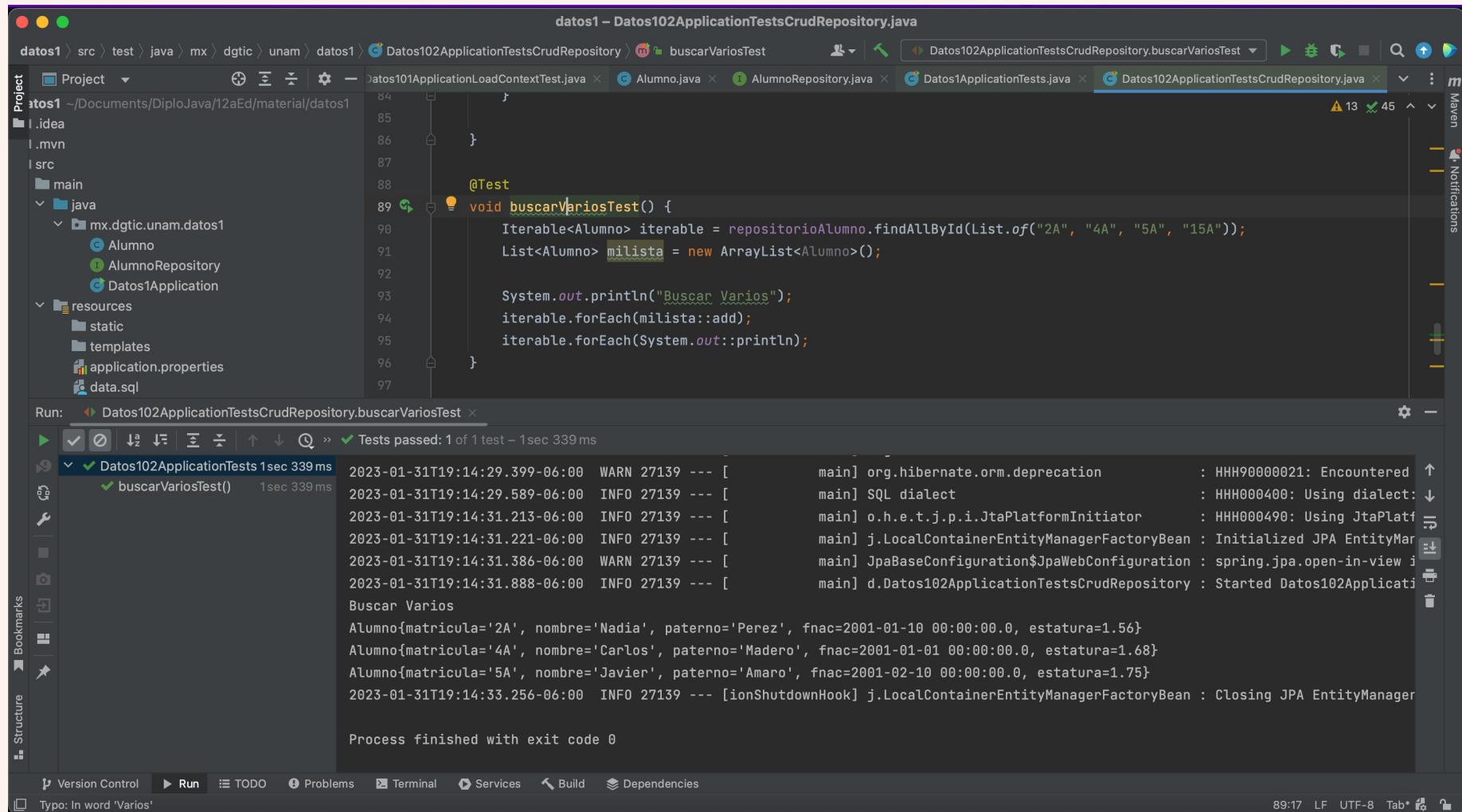
| matricula | nombre | paterno | fnac | estatura |
|-----------|---------|---------|------------|----------|
| 1A | Juan | Lopez | 2001-01-01 | 1.78 |
| 1F | OMAR | MENDOZA | 2023-01-31 | 1.74 |
| 2A | Nadia | Perez | 2001-01-10 | 1.56 |
| 3B | Perla | Avenida | 2001-01-20 | 1.62 |
| 3N | Marco 3 | Polo 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 |

- Buttons:** The bottom right of the grid area has "Apply" and "Revert" buttons.
- Right Sidebar:** This sidebar contains three buttons: "Result Grid" (selected), "Form Editor", and "Field Types".

Colecciones



findAllById()



The screenshot shows the IntelliJ IDEA interface during the execution of a unit test. The code being run is:

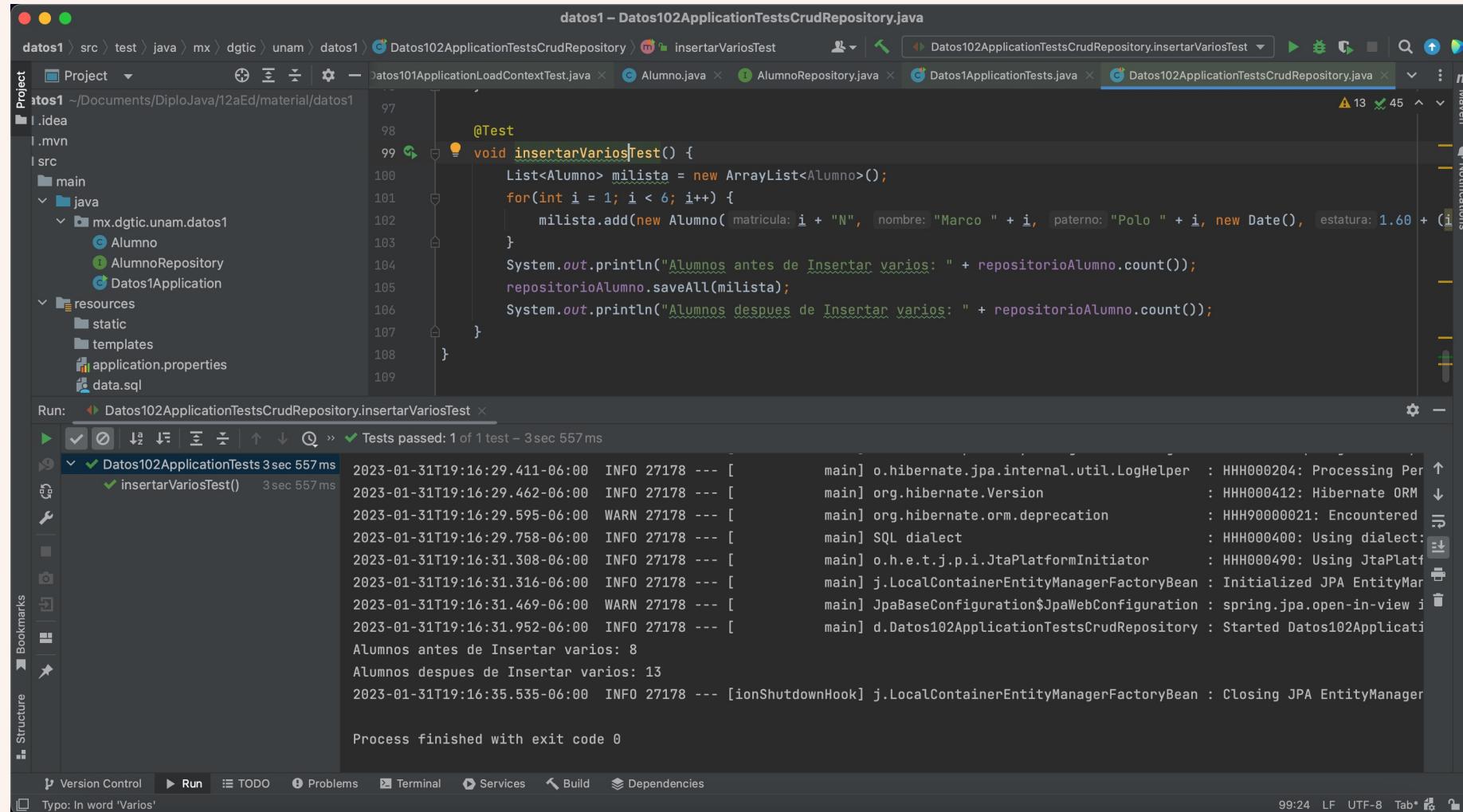
```
84     }
85 }
86 }
87 }
88 @Test
89 void buscarVariosTest() {
90     Iterable<Alumno> iterable = repositorioAlumno.findAllById(List.of("2A", "4A", "5A", "15A"));
91     List<Alumno> milista = new ArrayList<Alumno>();
92
93     System.out.println("Buscar Varios");
94     iterable.forEach(milista::add);
95     iterable.forEach(System.out::println);
96 }
97 }
```

The test passes with the message "Tests passed: 1 of 1 test - 1sec 339 ms". The output window shows the results of the database query:

```
2023-01-31T19:14:29.399-06:00  WARN 27139 --- [           main] org.hibernate.orm.deprecation          : HHH0000021: Encountered ...
2023-01-31T19:14:29.589-06:00  INFO 27139 --- [           main] SQL dialect                           : HHH000400: Using dialect: ...
2023-01-31T19:14:31.213-06:00  INFO 27139 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator      : HHH000490: Using JtaPlatform ...
2023-01-31T19:14:31.221-06:00  INFO 27139 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManager ...
2023-01-31T19:14:31.386-06:00  WARN 27139 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is set ...
2023-01-31T19:14:31.888-06:00  INFO 27139 --- [           main] d.Datos102ApplicationTestsCrudRepository : Started Datos102Application ...
Buscar Varios
Alumno{matricula='2A', nombre='Nadia', paterno='Perez', fnac=2001-01-10 00:00:00.0, estatura=1.56}
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}
2023-01-31T19:14:33.256-06:00  INFO 27139 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManager
```

The process finished with exit code 0.

saveAll()



The screenshot shows the IntelliJ IDEA IDE interface during the execution of a unit test. The test class is `Datos102ApplicationTestsCrudRepository.java`, specifically the `insertarVariosTest` method. The code creates a list of 6 `Alumno` objects and saves them using the `repositoryAlumno.saveAll` method. The test passes, taking 3 seconds.

```
datos1 - Datos102ApplicationTestsCrudRepository.java
...
97
98     @Test
99     void insertarVariosTest() {
100         List<Alumno> milista = new ArrayList<Alumno>();
101         for(int i = 1; i < 6; i++) {
102             milista.add(new Alumno( matricula: i + "N", nombre: "Marco " + i, paterno: "Polo " + i, new Date(), estatura: 1.60 + (i * 0.05) );
103         }
104         System.out.println("Alumnos antes de Insertar varios: " + repositoryAlumno.count());
105         repositoryAlumno.saveAll(milista);
106         System.out.println("Alumnos despues de Insertar varios: " + repositoryAlumno.count());
107     }
108 }
109 
```

Run: `Datos102ApplicationTestsCrudRepository.insertarVariosTest`

Tests passed: 1 of 1 test – 3 sec 557 ms

2023-01-31T19:16:29.411-06:00 INFO 27178 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [PersistenceUnit: DATOS1]
2023-01-31T19:16:29.462-06:00 INFO 27178 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM / 6.2.1.Final
2023-01-31T19:16:29.595-06:00 WARN 27178 --- [main] org.hibernate.orm.deprecation : HHH90000021: Encountered deprecated utilitarian API in org.hibernate.OrmException: ...
2023-01-31T19:16:29.758-06:00 INFO 27178 --- [main] SQL dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL5InnoDBDialect
2023-01-31T19:16:31.308-06:00 INFO 27178 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatformInitiator [org.hibernate.transaction.JBossJtaPlatform]
2023-01-31T19:16:31.316-06:00 INFO 27178 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit: DATOS1
2023-01-31T19:16:31.469-06:00 WARN 27178 --- [main] JpaBaseConfiguration\$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, non-localized ...
2023-01-31T19:16:31.952-06:00 INFO 27178 --- [main] d.Datos102ApplicationTestsCrudRepository : Started Datos102ApplicationTestsCrudRepository in 0.01 seconds (refreshing: 1)
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 EntityManagerFactory for persistence unit: DATOS1

Process finished with exit code 0

deleteAll()

- Elimina todas las entidades administradas por el repositorio.

Contacto

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx