



DIPLOMADO  
**Desarrollo de sistemas con  
tecnología Java**

**Módulo 8**  
Persistencia con Spring Data

*Dr. Omar Mendoza González*

*omarmendoza564@aragon.unam.mx*

# Consultas Derivadas en Spring Data JPA



Permiten generar **automáticamente** consultas basadas en el nombre del método en los repositorios.



Estas consultas se generan sin necesidad de escribir código SQL o JPQL explícito.



Los métodos se nombran usando una **Sintaxis intuitiva** y convenciones como **findBy**, **countBy**, **deleteBy**, etc.

# Métodos de consultas derivadas

- Los nombres de los métodos derivados tienen dos partes principales **Introductor** y **Criterios**, separadas por la primera palabra clave **By**
- `List<Alumno> findByNombre(String nombre)`
  - La primera parte, **find**, es el *introductor*
  - La segunda parte **Nombre** es el *criterio*.
- **Spring Data JPA** admite
  - **find**
  - **read**
  - **query**
  - **count**
  - **get**

# Métodos de consultas derivadas

- También pueden usarse ***Distinct***, ***First*** o ***Top*** para eliminar duplicados o limitar el result set
- List<Alumno> **findTop3ByPaterno()**
- **La parte de criterios contiene las expresiones de condición específicas de la entidad de la consulta.**
- Se pueden usar las palabras clave de condición junto con los nombres de propiedad de la entidad.
- También se pueden concatenar las expresiones con *And* y *Or*

# Métodos de consultas derivadas

Acción	Verbos Usados	Descripción	Ejemplo
Encontrar (Find)	find, read, get, query, search, stream	Recupera uno o más registros que coinciden con los criterios.	findByNombre(String nombre)
Contar (Count)	count	Devuelve el número de registros que coinciden con los criterios.	countByNombre(String nombre)
Verificar Existencia (Exists)	exists	Verifica si existe al menos un registro que coincida con los criterios, devolviendo un valor booleano.	existsByNombre(String nombre)
Eliminar (Delete)	delete, remove	Elimina uno o más registros que coinciden con los criterios.	deleteByNombre(String nombre)
Ordenar (Sort)	findBy...OrderBy...Asc, findBy...OrderBy...Desc	Recupera registros ordenados según uno o más campos, de forma ascendente o descendente.	findByNombreOrderByPaternoA sc(String nombre)
Paginar (Paging)	findByNombre(Pageable pageable)	Recupera un conjunto de resultados paginados.	findByNombre(String nombre, PageRequest.of(0, 10))
Limitar (Limit/Top)	findTop, findFirst	Devuelve el primer o los primeros registros que coinciden con los criterios.	findTop5ByNombre(String nombre)
Distintos (Distinct)	findDistinct	Recupera registros únicos que coinciden con los criterios.	findDistinctByNombre(String nombre)

# Find vs Stream

- **find**
  - El método **find** se utiliza para recuperar una lista completa de resultados que coincidan con la consulta.
  - Devuelve una **lista de objetos** de la entidad (**List<T>**).
  - Los resultados se recuperan **todos a la vez** y se almacenan en memoria, lo que puede ser costoso si el número de resultados es grande.
  - Adecuado cuando se necesitan todos los resultados a la vez y se requiere trabajar con la lista en la memoria.

```
List<Alumno> alumnos = alumnoRepository.findByNombre(String nombre);  
// Trabaja con todos los alumnos a la vez  
alumnos.forEach(System.out::println);
```

# Find vs Stream

- **stream:**
  - El método **stream** devuelve los resultados en forma de un **Stream<T>**, lo que permite un procesamiento más eficiente y escalable.
  - Devuelve un **stream de objetos** (**Stream<T>**).
  - Los datos pueden cargarse de manera **perezosa (lazy)**, es decir, se cargan según sea necesario mientras se recorre el Stream, lo que puede ser más eficiente para trabajar con grandes cantidades de datos.
  - Adecuado cuando se requiere procesar los resultados secuencialmente, sin cargar todos los datos en la memoria a la vez.

```
try (Stream<Alumno> alumnosStream = alumnoRepository.streamByNombre(String nombre))
{
    alumnosStream.forEach(System.out::println);
}
```

# Find vs Stream

Característica	find	stream
Resultado	List<T> (todos los resultados)	Stream<T> (procesa los resultados secuencialmente)
Carga de datos	Carga todos los datos a la vez	Carga datos a medida que se necesitan (lazy)
Uso de memoria	Puede ser intensivo para grandes volúmenes	Más eficiente en memoria
Uso típico	Cuando necesitas todos los resultados de una vez	Para procesar grandes volúmenes de datos gradualmente
Cierre necesario	No	Sí, hay que cerrar el Stream

# FindByField

AlumnoRepository

findByNombre(String nombre)

findByPaterno(String paterno)

findByEstatura(double estatura)

findByFecha(date fecha)

# FindByField

AlumnoRepository

findByNombre(String nombre)

findByNombres(String nombre)

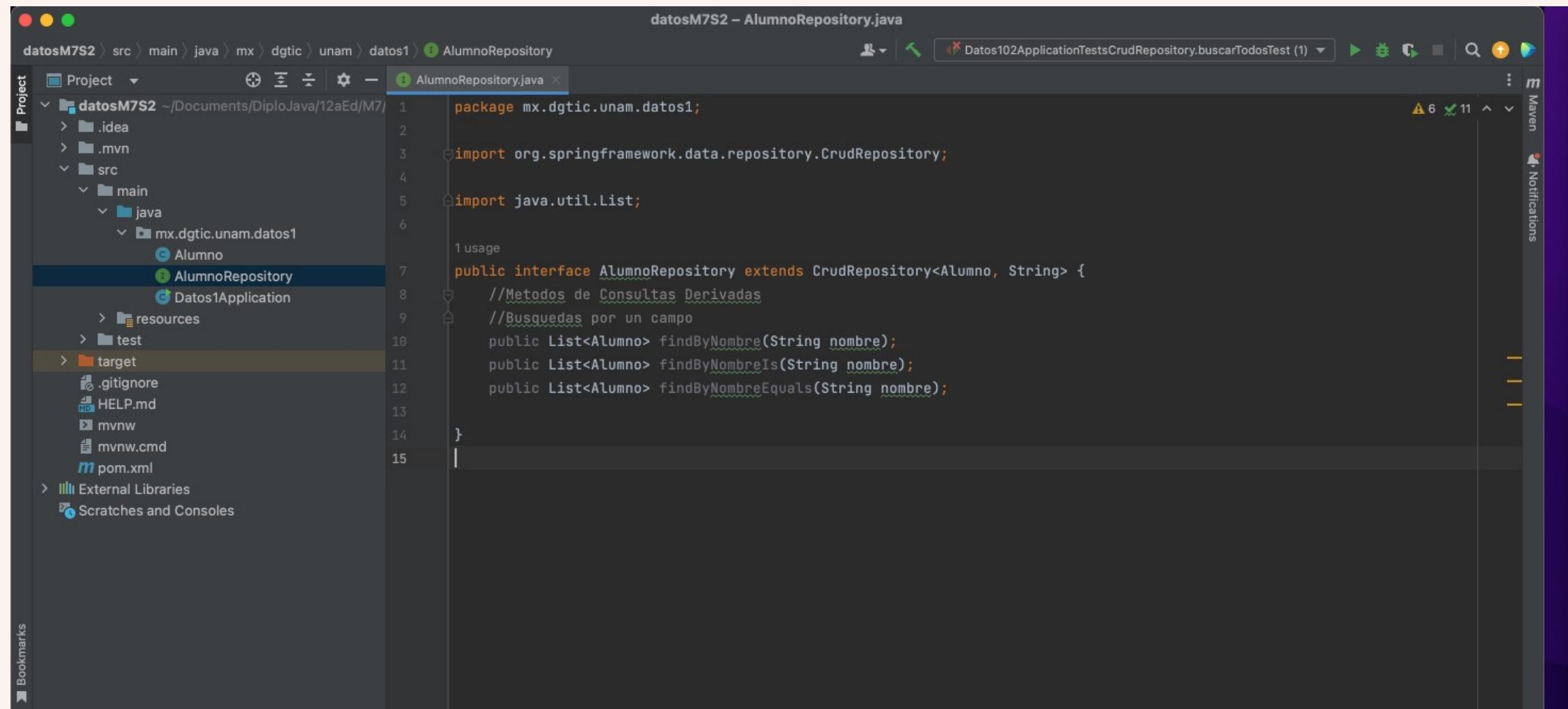
findByNombreEquals(String nombre)

# FindByField

AlumnoRepository

```
getByNombre(String nombre)  
searchByNombre(String nombre)  
streamByNombre(String nombre)  
readByNombre(String nombre)  
queryByNombre(String nombre)
```

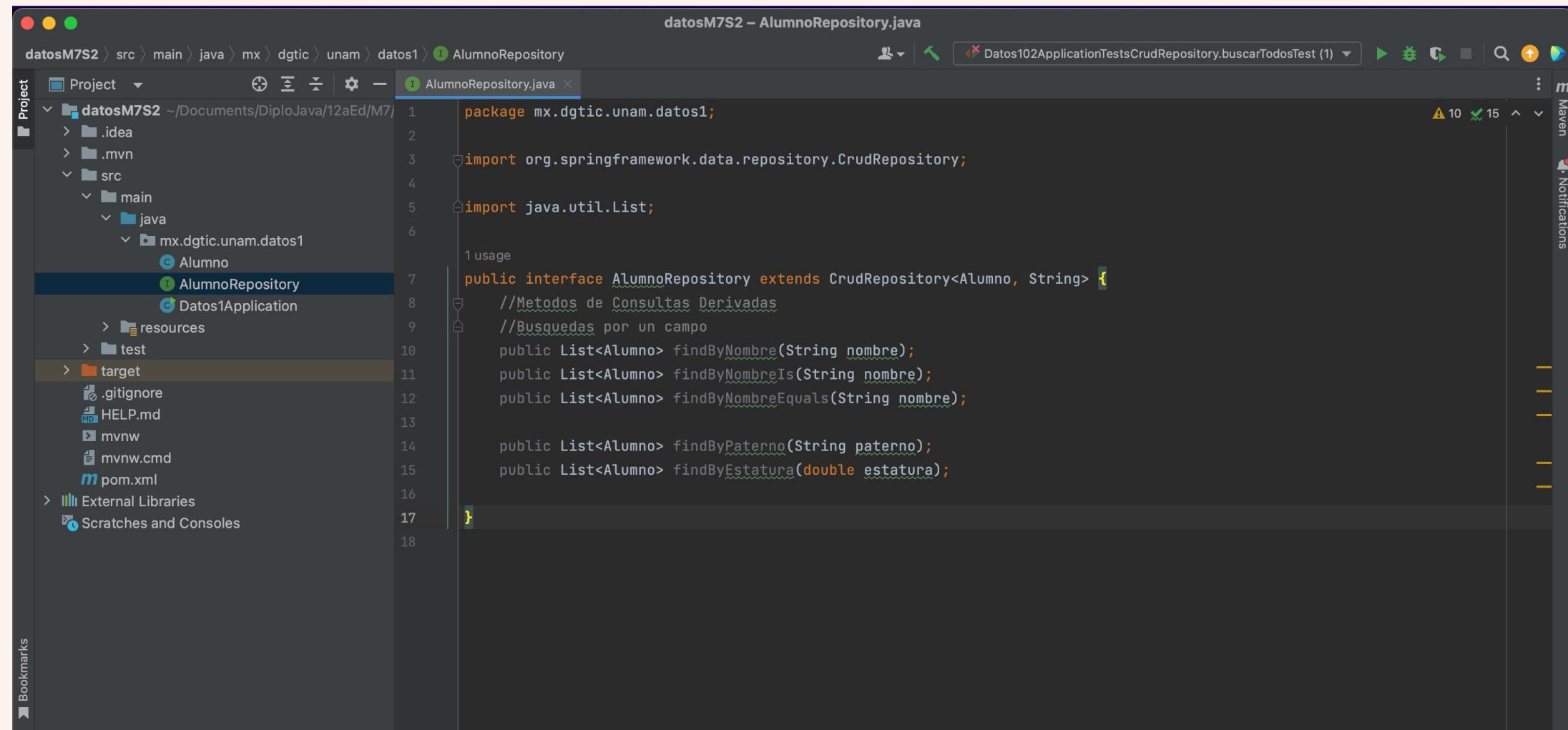
# FindByField



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

- Project Structure:** The project is named "datosM7S2". It contains a "src" directory with "main" and "java" sub-directories. The "java" directory has a package "mx.dgtic.unam.datos1" containing classes "Alumno", "AlumnoRepository", and "Datos1Application".
- Code Editor:** The file "AlumnoRepository.java" is open. The code defines an interface "AlumnoRepository" that extends "CrudRepository<Alumno, String>". It includes three methods for querying by name: "findByNombre", "findByNombreIs", and "findByNombreEquals".
- Toolbars and Status:** The top bar shows the project path "datosM7S2 > src > main > java > mx > dgtic > unam > datos1 > AlumnoRepository.java". There are tabs for "AlumnoRepository.java" and "Datos102ApplicationTestsCrudRepository.buscarTodosTest (1)". The status bar on the right shows "A 6 X 11".
- Side Panels:** The "Project" panel on the left shows the file structure. The "Maven" and "Notifications" panels are visible on the right.

# FindByField



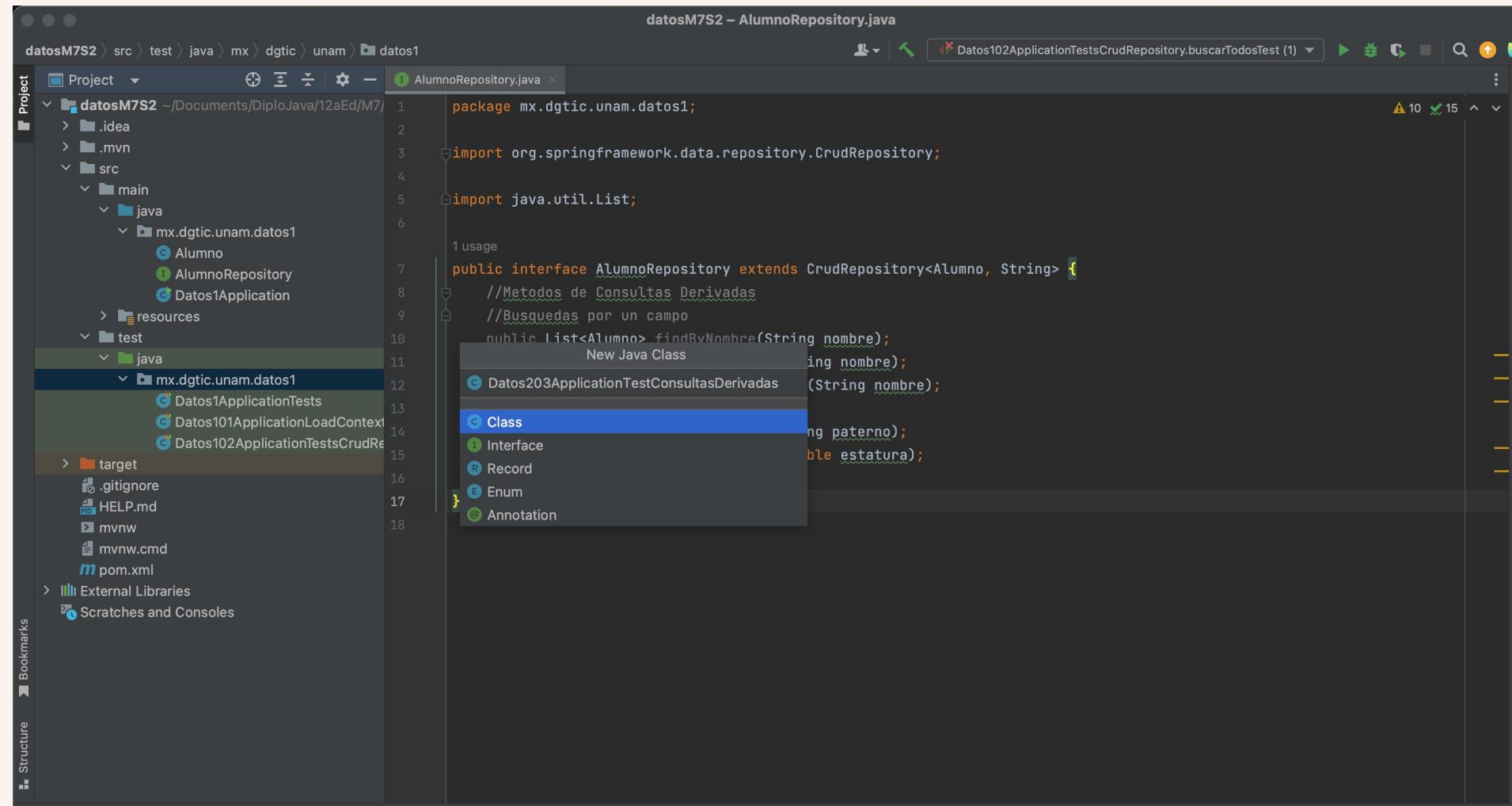
The screenshot shows the IntelliJ IDEA interface with the file `AlumnoRepository.java` open. The code defines a repository interface for the `Alumno` entity, extending `CrudRepository`. It includes methods for finding students by name, paternity, and height.

```
package mx.dgtic.unam.datos1;
import org.springframework.data.repository.CrudRepository;
import java.util.List;

public interface AlumnoRepository extends CrudRepository<Alumno, String> {
    //Metodos de Consultas Derivadas
    //Busquedas por un campo
    public List<Alumno> findByNombre(String nombre);
    public List<Alumno> findByNombreIs(String nombre);
    public List<Alumno> findByNombreEquals(String nombre);

    public List<Alumno> findByPaterno(String paterno);
    public List<Alumno> findByEstatura(double estatura);
}
```

# Prueba unitaria findByNombre()

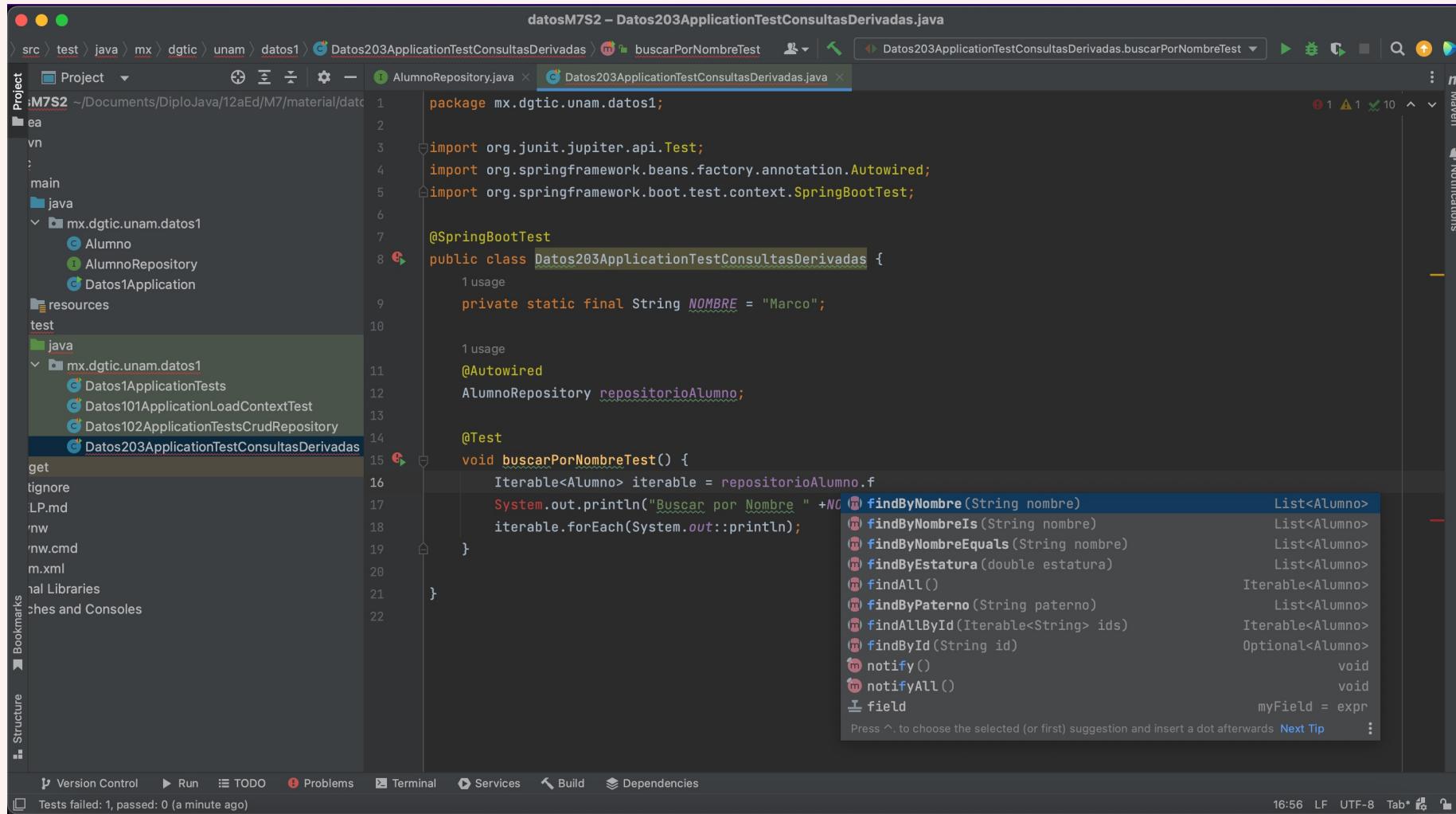


The screenshot shows the IntelliJ IDEA interface with the code completion feature open. The current file is `AlumnoRepository.java`, which contains the following code:

```
package mx.dgtic.unam.datos1;
import org.springframework.data.repository.CrudRepository;
import java.util.List;
1 usage
public interface AlumnoRepository extends CrudRepository<Alumno, String> {
    //Metodos de Consultas Derivadas
    //Busquedas por un campo
    public List<Alumno> findByNombre(String nombre);
}
```

The code completion menu is displayed at the bottom of the editor, with the option `C Class` highlighted. Other options shown include `New Java Class`, `Datos203ApplicationTestConsultasDerivadas`, `(String nombre);`, `Class`, `Interface`, `Record`, `Enum`, and `Annotation`.

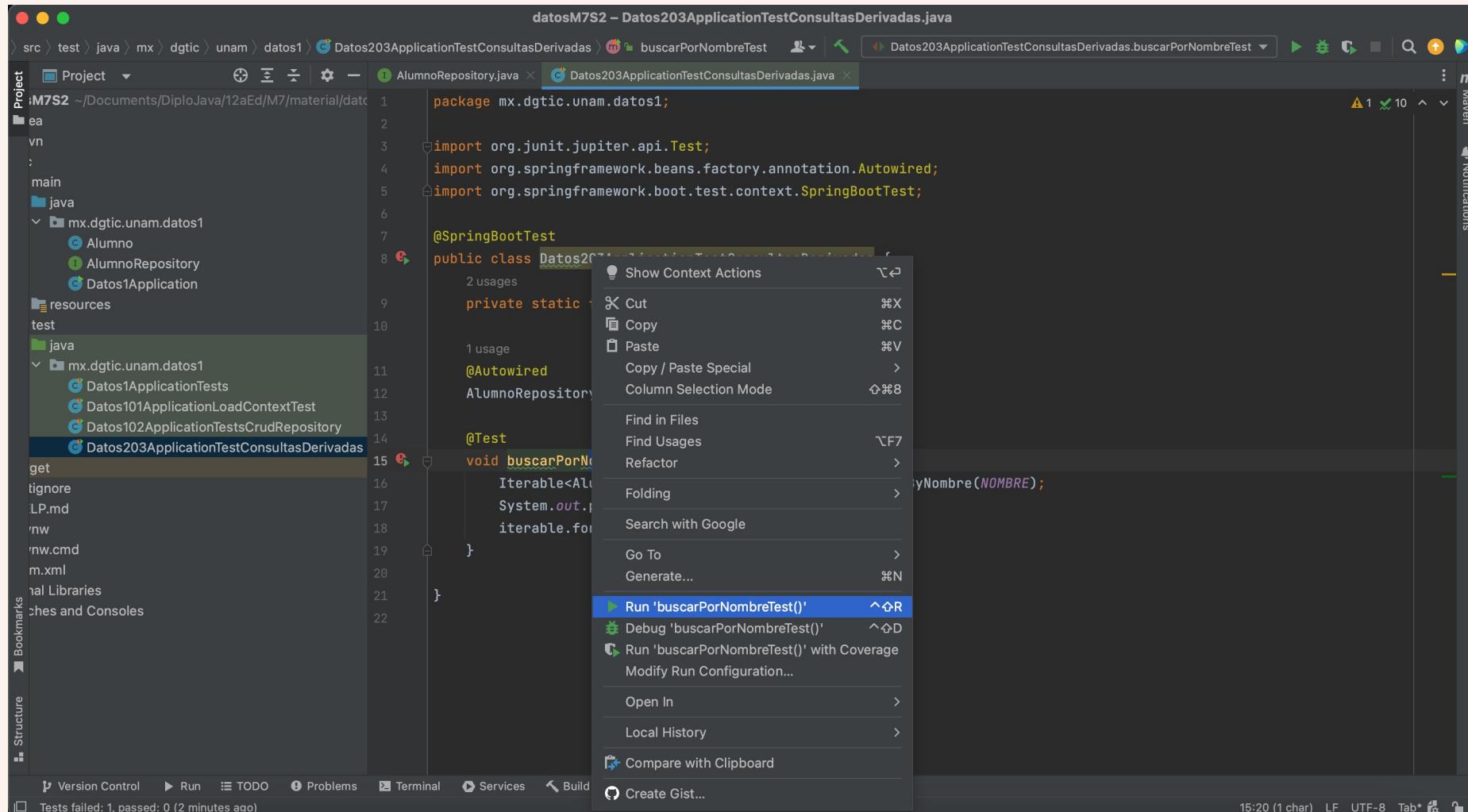
# Prueba unitaria findByNombre()



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

- Project Structure:** The project is named "iM7S2" and contains packages like "mx.dgtic.unam.datos1" and "mx.dgtic.unam.datos1.AlumnoRepository".
- Code Editor:** The file "Datos203ApplicationTestConsultasDerivadas.java" is open, showing a test class for an "AlumnoRepository".
- Method Implementation:** The code includes imports for JUnit Jupiter, Spring Framework beans, and Spring Boot Test.
- Test Method:** A test method "buscarPorNombreTest" is defined, which iterates over an iterable of "Alumno" objects and prints them to the console.
- Completion Pop-up:** A code completion pop-up is displayed for the "Iterable<Alumno> iterable = repositorioAlumno.f" line, showing various methods available for the repository interface, such as "findAll()", "findByNombre(String nombre)", and "findById(String id)".
- Status Bar:** The status bar at the bottom shows "Tests failed: 1, passed: 0 (a minute ago)" and the current time as "16:56".

# findByNombre()



The screenshot shows an IDE interface with a dark theme. The left sidebar displays a project structure for 'M7S2' containing 'src', 'test', and 'resources' directories. Under 'src/test/java/mx/dgtic/unam/datos1', there are several test classes: 'Alumno', 'AlumnoRepository', 'Datos1Application', 'Datos1ApplicationTests', 'Datos101ApplicationLoadContextTest', 'Datos102ApplicationTestsCrudRepository', and 'Datos203ApplicationTestConsultasDerivadas'. The 'Datos203ApplicationTestConsultasDerivadas' class is currently selected. The code editor shows the following Java code:

```
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;

@SpringBootTest
public class Datos203ApplicationTestConsultasDerivadas {
    @Autowired
    private static AlumnoRepository alumnoRepository;

    @Test
    void buscarPorNombre() {
        Iterable<Alumno> iterable = alumnoRepository.findAll();
        System.out.println(iterable);
    }
}
```

A context menu is open over the 'buscarPorNombre' method, listing options such as 'Cut', 'Copy', 'Paste', 'Run 'buscarPorNombreTest()' (highlighted in blue), and 'Debug 'buscarPorNombreTest()''.

# findByNombre()

```
@Test
void buscarPorNombreTest() {
    Iterable<Alumno> iterable = repositorioAlumno.findByNombre(NOMBRE);
    System.out.println("Buscar por Nombre " + NOMBRE);
    iterable.forEach(System.out::println);
}
```

The screenshot shows an IDE interface with a dark theme. On the left, there's a toolbar with various icons. The main area displays the test code above. Below it, the terminal output shows the test results and the printed output from the test method. The terminal window title bar says "Tests passed: 1 of 1 test – 1sec 334 ms". The terminal content includes:

- Test results: "2023-02-03T20:16:28.237-06:00 [INFO] JpaBaseConfiguration\$JpaWebConfiguration : Tests passed: 1 of 1 test – 1sec 334 ms"
- Test execution: "2023-02-03T20:16:28.834-06:00 [INFO] atos203ApplicationTestConsultasDerivadas :"
- Printed output: "Buscar por Nombre Gema" followed by the Alumno object details: "Alumno{matricula='7B', nombre='Gema', paterno='null', fnac=2001-03-20 00:00:00.0, estatura=1.53}"
- Shutdown message: "2023-02-03T20:16:30.196-06:00 [INFO] j.LocalContainerEntityManagerFactoryBean :"
- Final message: "Process finished with exit code 0"

# findByEstatura()

```
@Test
void buscarEstaturaTest() {
    Iterable<Alumno> iterable = repositorioAlumno.findByEstatura(ESTATURA);
    System.out.println("Buscar por estatura");
    iterable.forEach(System.out::println);
}
```

```
Buscar por estatura 1.6
Alumno{matricula='1N', nombre='Marco 1', paterno='Polo 1', fnac=2023-01-31 00:00:00.0, estatura=1.6}
Alumno{matricula='2N', nombre='Marco 2', paterno='Polo 2', fnac=2023-01-31 00:00:00.0, estatura=1.6}
Alumno{matricula='3N', nombre='Marco 3', paterno='Polo 3', fnac=2023-01-31 00:00:00.0, estatura=1.6}
Alumno{matricula='4N', nombre='Marco 4', paterno='Polo 4', fnac=2023-01-31 00:00:00.0, estatura=1.6}
Alumno{matricula='5N', nombre='Marco 5', paterno='Polo 5', fnac=2023-01-31 00:00:00.0, estatura=1.6}
```

# findByFecha()

```
@Test
void buscarFechaTest() {
    List<Alumno> alumnos;
    System.out.println("Buscar por fecha");

    try {
        alumnos = repositorioAlumno.getByFnac(new SimpleDateFormat( pattern: "yyyy-MM-dd")
            .parse( source: "2001-03-20"));
        alumnos.forEach(System.out::println);
    } catch (ParseException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Buscar por fecha

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}

2023-02-03T20:52:21.093-06:00 INFO 11959 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManager

# FindByField negación

AlumnoRepository

findByNombreIsNot(String nombre)

findByNombreNot(String nombre)

# FindByField negación

```
//Listar los diferentes a Nombre
1 usage
public List<Alumno> findByNombreNot(String nombre);

@Test
void buscarPorNombreNotTest() {
    Iterable<Alumno> iterable = repositorioAlumno.findByNombreNot(NOMBRE);
    System.out.println("Buscar por Nombre <> " +NOMBRE);
    iterable.forEach(System.out::println);
}

Buscar por Nombre <> Gema
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='1N', nombre='Marco 1', paterno='Polo 1', fnac=2023-01-31 00:00:00.0, estatura=1.6}
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}
```

# FindByFieldisNullisNotNull

AlumnoRepository

findByPaternoIsNotNull()

findByPaternoIsNotNull()

# findByPaternoIsNull()

```
//Null  
1 usage  
public List<Alumno> findByPaternoIsNull();  
1 usage  
public List<Alumno> findByPaternoIsNotNull();
```

```
@Test  
void buscarPaternoIsNullTest() {  
    Iterable<Alumno> iterable = repositorioAlumno.findByPaternoIsNull();  
    System.out.println("Buscar por paterno is null");  
    iterable.forEach(System.out::println);  
}
```

Buscar por paterno is null

Alumno{matricula='7B', nombre='Gema', paterno='null', fnac=2001-03-20 00:00:00.0, estatura=1.53}

2023-02-03T20:37:40.743-06:00 INFO 11863 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManager

Process finished with exit code 0

# Condición de similitud

AlumnoRepository

findByPaternoStartingWith(String prefix)

findByPaternoEndingWith(String suffix)

findByPaternoContaining(String infix)

# Like

```
String likePattern = "a%b%c";
```

AlumnoRepository

findByNombreLike(String likePattern)

# Condiciones de comparación

AlumnoRepository

findByEstaturaLessThan(double estatura)

findByEstaturaLessThanEqual(double estatura)

findByEstaturaGreaterThanOrEqual(double estatura)

findByEstaturaGreaterThanOrEqual(double estatura)

# Condiciones de comparación

```
//Condiciones de comparación

public List<Alumno> findByEstaturaLessThan(double estatura);
public List<Alumno> findByEstaturaLessThanEqual(double estatura);
public List<Alumno> findByEstaturaGreaterThanOrEqual(double estatura);
```

```
@Test
void buscarPorEsaturaMayorQue() {

    List<Alumno> lista = repositorioAlumno.findByEstaturaGreaterThanOrEqual(ESTATURA);
    System.out.println("findByEstaturaGreaterThanOrEqual");
    lista.forEach(System.out::println);

    assertThat(lista.size(), greaterThan(0));
}
```

```
findByEstaturaGreaterThanOrEqual
Alumno [matricula=5A, nombre=Javier, paterno=Amaro, fnac=2001-02-10 00:00:00.0, estatura=1.75]
Alumno [matricula=G1, nombre=Maria, paterno=Paterno 1, fnac=2022-01-06 00:00:00.0, estatura=2.6]
Alumno [matricula=G2, nombre=Maria, paterno=Paterno 2, fnac=2022-01-06 00:00:00.0, estatura=3.6]
Alumno [matricula=G3, nombre=Maria, paterno=Paterno 3, fnac=2022-01-06 00:00:00.0, estatura=4.6]
Alumno [matricula=G4, nombre=Maria, paterno=Paterno 4, fnac=2022-01-06 00:00:00.0, estatura=5.6]
Alumno [matricula=G5, nombre=Maria, paterno=Paterno 5, fnac=2022-01-06 00:00:00.0, estatura=6.6]
```

# Condiciones de comparación

AlumnoRepository

```
findByEstaturaBetween(double estaturaini, double estaturafin)  
findByEstaturaIn(Collection<Double> estaturas)
```

# Condiciones de comparación

```
public List<Alumno> findByEstaturaBetween(double estaturaini, double estaturafin);  
public List<Alumno> findByEstaturaIn(Collection<Double> estaturas);
```

```
@Test  
void buscarPorEsaturaIn() {  
  
    final List<Double> estaturas = Arrays.asList(1.53, 1.60, 1.68);  
  
    List<Alumno> lista = repositorioAlumno.findByEstaturaIn(estaturas);  
    System.out.println("findByEstaturaIn");  
    lista.forEach(System.out::println);  
  
    assertThat(lista.size(), greaterThan(0));  
}
```

```
findByEstaturaIn  
Alumno [matricula=1F, nombre=Oscar, paterno=Medina, fnac=2022-01-06 00:00:00.0, estatura=1.68]  
Alumno [matricula=3B, nombre=Perla, paterno=Rios, 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=7B, nombre=Gema, paterno=null, fnac=2001-03-20 00:00:00.0, estatura=1.53]
```

# Condiciones fecha

AlumnoRepository

findByFnacAfter(Date fecha)

findByFnacBefore(Date fecha)

# Condiciones fecha

```
//.....  
  
public List<Alumno> findByFnacAfter(Date fecha);  
public List<Alumno> findByFnacBefore(Date fecha);  
  
@Test  
void buscarPorFechaAntes() {  
  
    List<Alumno> lista;  
    try {  
        lista = repositorioAlumno.  
            findByFnacBefore(new SimpleDateFormat("yyyy-MM-dd").parse("2022-01-05"));  
        System.out.println("findByFnacBefore");  
        lista.forEach(System.out::println);  
  
        assertThat(lista.size(), greaterThan(0));  
    } catch (ParseException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
}  
  
findByFnacBefore  
Alumno [matricula=2A, nombre=Nadia, paterno=Perez, fnac=2001-01-10 00:00:00.0, estatura=1.56]  
Alumno [matricula=3B, nombre=Perla, paterno=Rios, 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]
```

# Expresiones de condición múltiple AND / OR

AlumnoRepository

```
findByNombreOrPaterno(String nombre, String paterno)
```

```
findByNombreAndPaterno(String nombre, String paterno)
```

# Expresiones de condición múltiple AND / OR

Keyword	Descripción
find...By, read...By, get...By, query...By, search...By, stream...By	Método de consulta general Puede ser utilizado como findBy..., findMyDomainTypeBy...o en combinación con palabras clave adicionales.
exists...By	Existe proyección, que devuelve típicamente un resultado boolean.
count...By	Cuenta la proyección que devuelve un resultado numérico.
delete...By, remove...By	Eliminar método de consulta que no devuelve ningún resultado ( void) o el recuento de eliminación.
...First<number>..., ...Top<number>...	Limita los resultados de la consulta al primero <number>de los resultados. Esta palabra clave puede aparecer en cualquier lugar del tema entre find(y las otras palabras clave) y by.
...Distinct...	Devuelve solo resultados únicos.

Logical keyword	Keyword expressions
AND	And
OR	Or
AFTER	After, IsAfter
BEFORE	Before, IsBefore
CONTAINING	Containing, IsContaining, Contains
BETWEEN	Between, IsBetween
ENDING_WITH	EndingWith, IsEndingWith, EndsWith
EXISTS	Exists
FALSE	False, IsFalse
GREATER_THAN	GreaterThan, IsGreaterThan
GREATER_THAN_EQUALS	GreaterThanEqual, IsGreaterThanOrEqual
IN	In, IsIn
IS	Is, Equals, (or no keyword)
IS_EMPTY	IsEmpty, Empty
IS_NOT_EMPTY	IsNotEmpty, NotEmpty
IS_NOT_NULL	NotNull, IsNotNull
IS_NULL	Null, IsNull
LESS_THAN	LessThan, IsLessThan
LESS_THAN_EQUAL	LessThanEqual, IsLessThanEqual
LIKE	Like, IsLike
NEAR	Near, IsNear
NOT	Not, IsNot
NOT_IN	NotIn, IsNotIn
NOT_LIKE	NotLike, IsNotLike
REGEX	Regex, MatchesRegex, Matches
STARTING_WITH	StartingWith, IsStartingWith, StartsWith
TRUE	True, IsTrue
WITHIN	Within, IsWithin

# Ordenar los resultados

AlumnoRepository

findByNombreOrderByNombre(String nombre)

findByNombreOrderByNombreDesc(String nombre)

# Limitar los resultados

AlumnoRepository

`findFirstOrderByEstaturaAsc()`

`findTopOrderByEstaturaAsc()`

# Contar los resultados

AlumnoRepository

countByEstatura(double estatura)

countByEstaturaGreaterThanOrEqualTo(double estatura)

countByNombreEndingWith(String endString)

countByNombreLike(String likeString)

# Resultados Unicos

AlumnoRepository

findAlumnoDistinctByNombre(String nombre)

# Contacto

Dr. Omar Mendoza González

[omarmendoza564@aragon.unam.mx](mailto:omarmendoza564@aragon.unam.mx)