



**11<sup>a</sup>**  
Emisión

**DIPLOMADO**  
**Desarrollo de Sistemas**  
**con Tecnología Java**

**Módulo 7**  
**Persistencia con Spring Data**

*Dr. Omar Mendoza González*

*omarmendoza564@aragon.unam.mx*



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



Educación  
Continua  
1971 - 2021

# Métodos de consultas derivadas

- Los nombres de métodos derivados tienen dos partes principales separadas por la primera palabra clave **By**
- `List<Alumno> findByNombre(String nombre)`
  - La primera parte, **find**, es el *introduction*
  - La segunda parte **ByNombre**, son los *criterios*.
- **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*



# FindByField

AlumnoRepository

findByNombre(String nombre)

findByPaterno(String paterno)

findByEstatura(double estatura)



# FindByField

AlumnoRepository

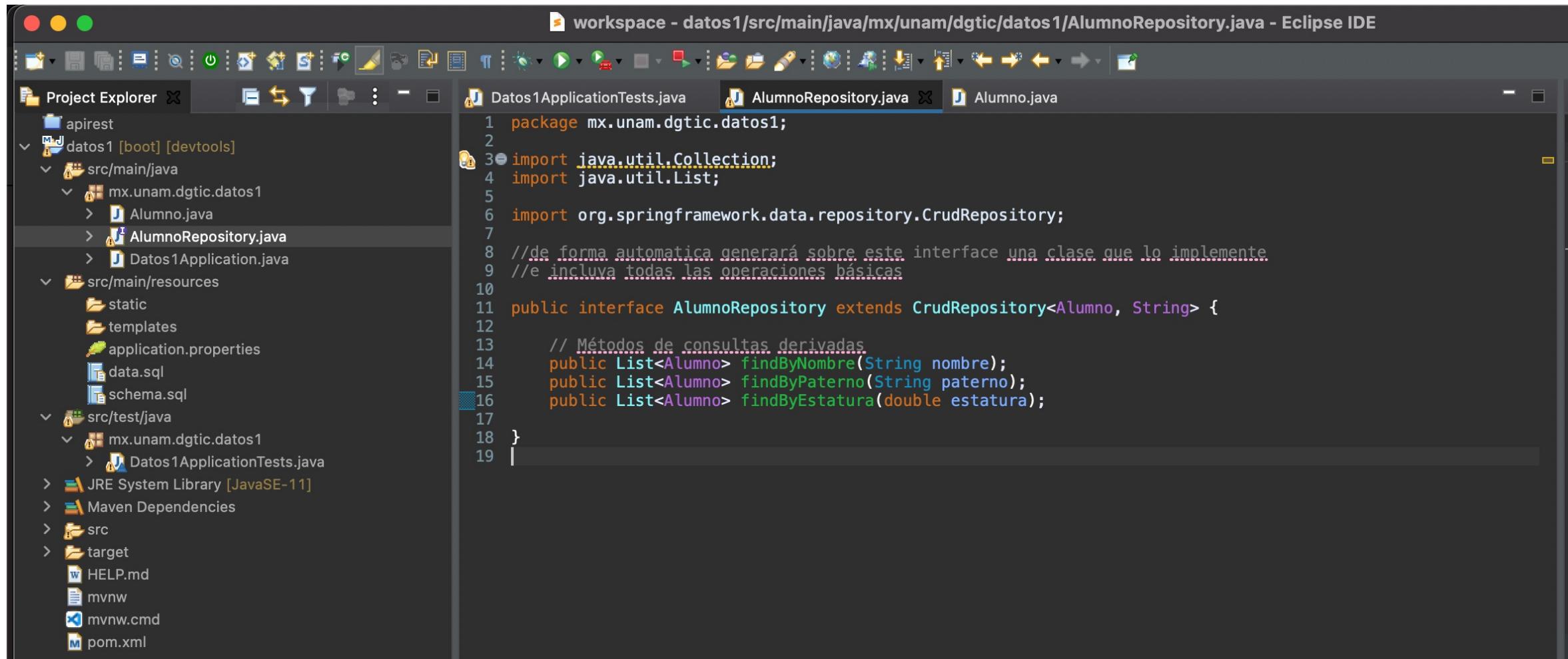
findByNombre(String nombre)

findByNombres(String nombre)

findByNombreEquals(String nombre)



# FindByField

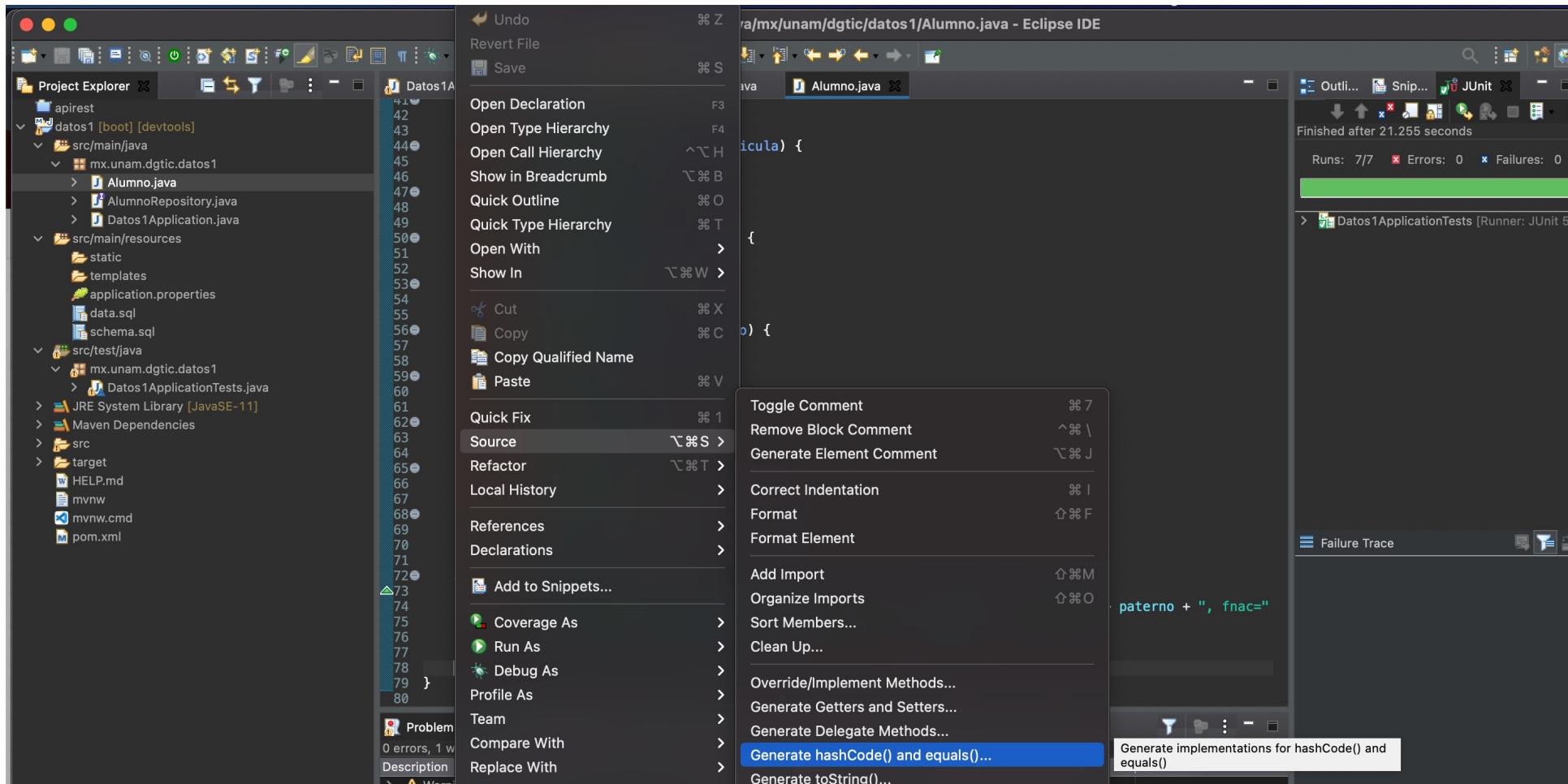


The screenshot shows the Eclipse IDE interface with the following details:

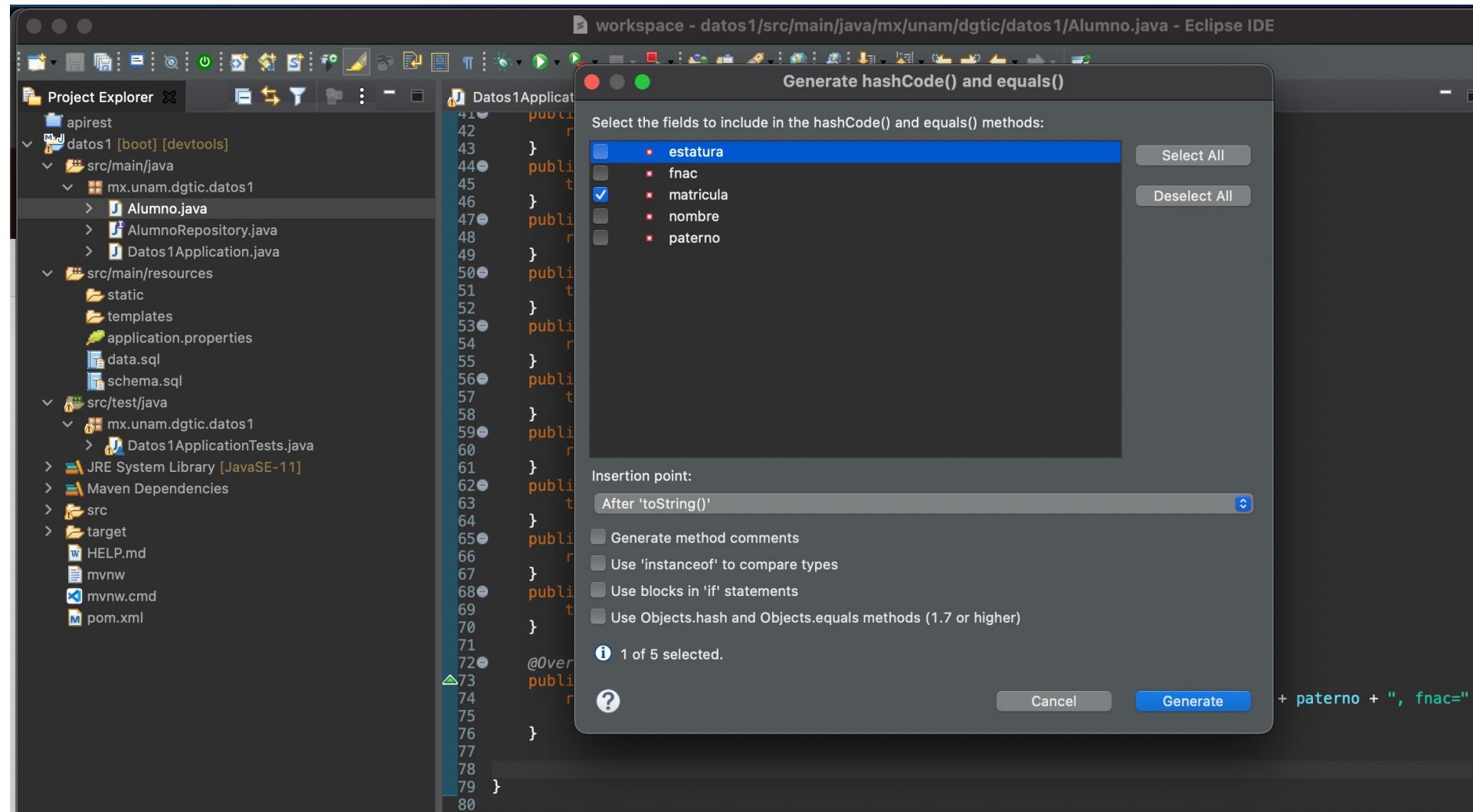
- Title Bar:** workspace - datos1/src/main/java/mx/unam/dgtic/datos1/AlumnoRepository.java - Eclipse IDE
- Project Explorer:** Shows the project structure:
  - apirest
  - datos1 [boot] [devtools]
    - src/main/java
      - mx.unam.dgtic.datos1
        - Alumno.java
        - AlumnoRepository.java
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

```
1 package mx.unam.dgtic.datos1;
2
3 import java.util.Collection;
4 import java.util.List;
5
6 import org.springframework.data.repository.CrudRepository;
7
8 //de forma automatica generará sobre este interface una clase que lo implemente
9 //e incluya todas las operaciones básicas
10
11 public interface AlumnoRepository extends CrudRepository<Alumno, String> {
12
13     // Métodos de consultas derivadas
14     public List<Alumno> findByNombre(String nombre);
15     public List<Alumno> findByPaterno(String paterno);
16     public List<Alumno> findByEstatura(double estatura);
17
18 }
19
```

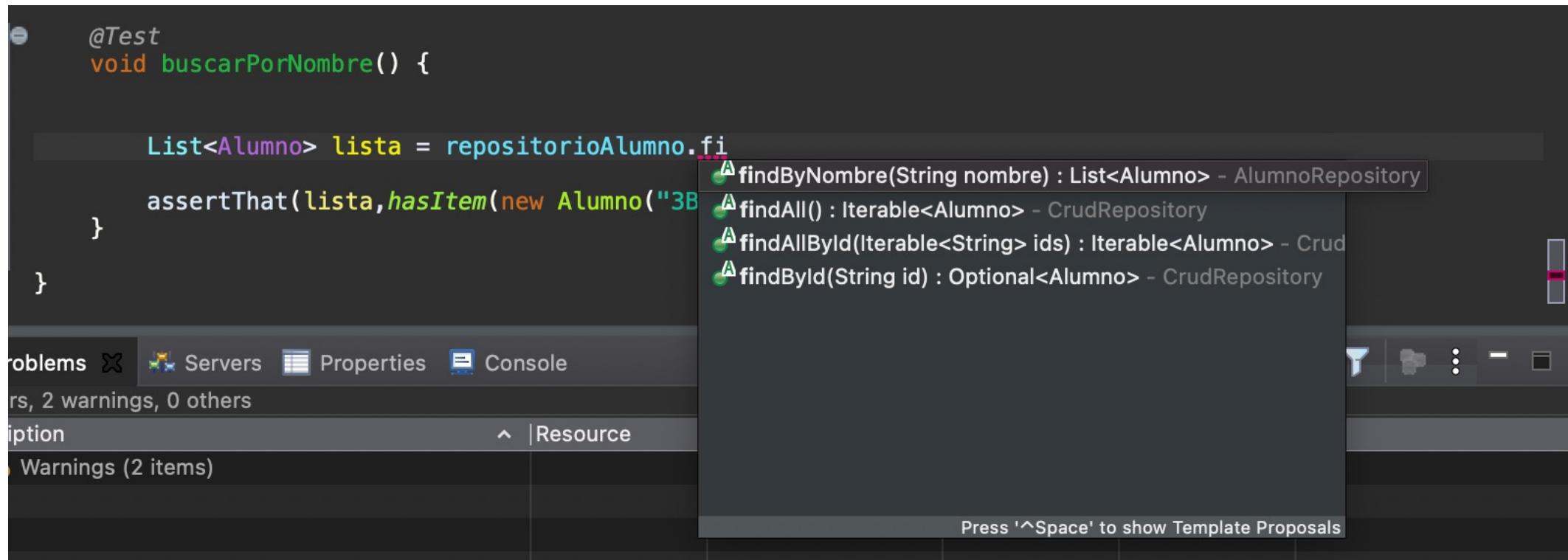
# Generate hashCode()



# Generate hashCode()



# findByNombre()



The screenshot shows a Java code editor with the following code:

```
@Test  
void buscarPorNombre() {  
  
    List<Alumno> lista = repositorioAlumno.fi  
    assertEquals(lista, hasItem(new Alumno("3B"))  
}  
  
Problems X Servers Properties Console  
0 errors, 2 warnings, 0 others  
option  
Warnings (2 items)  
Press '^Space' to show Template Proposals
```

A code completion dropdown is open over the `findByName` call, listing the following methods:

- findByNombre(String nombre) : List<Alumno> - AlumnoRepository
- findAll() : Iterable<Alumno> - CrudRepository
- findAllById(Iterable<String> ids) : Iterable<Alumno> - Crud
- findById(String id) : Optional<Alumno> - CrudRepository

# findByNombre()

```
• @Test  
void buscarPorNombre() {  
  
    List<Alumno> lista = repositorioAlumno.findByNombre(NOMBRE);  
    System.out.println("findByNombre");  
    lista.forEach(System.out::println);  
  
    assertThat(lista, hasItem(new Alumno("G1")));  
}
```

```
findByNombre  
Alumno [matricula=G1, nombre=Maria, paterno=Paterno 1, fnac=2022-01-05 00:00:00.0, estatura=2.6]  
Alumno [matricula=G2, nombre=Maria, paterno=Paterno 2, fnac=2022-01-05 00:00:00.0, estatura=3.6]  
Alumno [matricula=G3, nombre=Maria, paterno=Paterno 3, fnac=2022-01-05 00:00:00.0, estatura=4.6]  
Alumno [matricula=G4, nombre=Maria, paterno=Paterno 4, fnac=2022-01-05 00:00:00.0, estatura=5.6]  
Alumno [matricula=G5, nombre=Maria, paterno=Paterno 5, fnac=2022-01-05 00:00:00.0, estatura=6.6]
```

# findByEstatura()

```
● @Test  
void buscarPorEstatura() {  
  
    List<Alumno> lista = repositorioAlumno.findByEstatura(ESTATURA);  
    System.out.println("findByEstatura");  
    lista.forEach(System.out::println);  
  
    assertEquals(2, repositorioAlumno.findByEstatura(ESTATURA).size());  
}
```

```
findByEstatura  
Alumno [matricula=1F, nombre=Oscar, paterno=Medina, fnac=2022-01-05 00:00:00.0, estatura=1.68]  
Alumno [matricula=4A, nombre=Carlos, paterno=Madero, fnac=2001-01-01 00:00:00.0, estatura=1.68]
```

# FindByField negación

AlumnoRepository

findByNombreIsNot(String nombre)

findByNombreNot(String nombre)



# FindByField negación

```
//Negacion  
public List<Alumno> findByNombreNot(String nombre);
```

```
@Test  
void buscarPorNotNombre() {  
  
    List<Alumno> lista = repositorioAlumno.findByNombreNot(NOMBRE);  
    System.out.println("findByNombreNot");  
    lista.forEach(System.out::println);  
  
    assertThat(lista, not(hasItem(new Alumno("G1"))));  
}
```

```
findByNombreNot  
Alumno [matricula=1F, nombre=Oscar, paterno=Medina, fnac=2022-01-06 00:00:00.0, estatura=1.68]  
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]
```

# FindByFieldisNullisNotNull

AlumnoRepository

findByPaternoIsNotNull()

findByPaternoIsNotNull()



# FindByField negación

```
//IsNULL IsNotNULL  
public List<Alumno> findByPaternoIsNull();  
public List<Alumno> findByPaternoIsNotNull();
```

```
@Test  
void buscarPorPaternoNulo() {  
  
    List<Alumno> lista = repositorioAlumno.findByPaternoIsNull();  
    System.out.println("findByPaternoIsNull");  
    lista.forEach(System.out::println);  
  
    assertThat(lista.size(), greaterThan(0));  
}
```

```
findByPaternoIsNull  
Alumno [matricula=7B, nombre=Gema, paterno=null, fnac=2001-03-20 00:00:00.0, estatura=1.53]
```



# Condición de similitud

AlumnoRepository

findByNombreStartingWith(String prefix)

findByNombreEndingWith(String suffix)

findByNombreContaining(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);
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

```
//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)



| 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                         |

# Expresiones de condición múltiple AND / OR

| Keyword  | Descripción  |
|--|--|
| find...By, read...By, get...By, query...By, search...By, stream...By | Método de consulta general<br>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.<br>Esta palabra clave puede aparecer en cualquier lugar del tema entre find(y las otras palabras clave) y by. |
| ...Distinct...   | Devuelve solo resultados únicos.   |

# Ordenar los resultados

AlumnoRepository

findByNombreOrderByNombre(String nombre)

findByNombreOrderByNombreDesc(String nombre)



# Limitar los resultados

AlumnoRepository

`findFirstByOrderByEstaturaAsc()`

`findTopByOrderByEstaturaAsc()`



# 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)

