



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



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*

# Named Queries



Las **Named Queries** son un concepto central en **JPA** que permiten definir consultas reutilizables de forma estática.



Permiten declarar una consulta en la capa de persistencia y hacer referencia a ella en la capa de negocio.



**Facilitan la reutilización** de consultas, mejorando la legibilidad y el mantenimiento del código.



Se pueden definir usando la anotación **@NamedQuery** en una clase de entidad, o con **@Query** directamente en un repositorio.



Cuando define una NamedQuery, puede proporcionar una consulta **JPQL** o una consulta **SQL** nativa de formas muy similares.

# Named Queries

JPQL (Java Persistence Query Language) tiene una sintaxis similar a **SQL**, pero con diferencias clave que lo hacen adecuado para consultas orientadas a objetos.

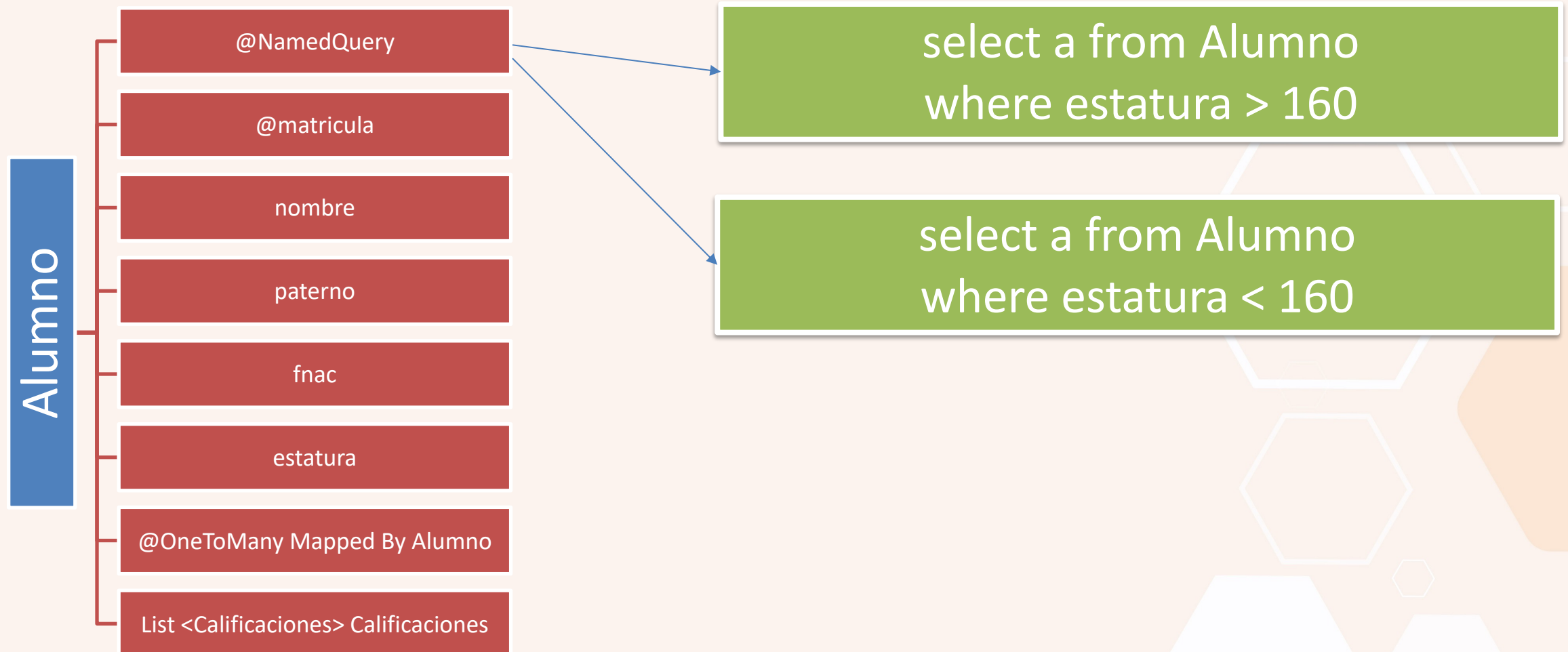
- Modelo basado en entidades.
- Cuando se ejecuta, el proveedor de persistencia genera una consulta SQL basada en las asignaciones de entidades y la declaración JPQL proporcionada.
- JPQL permite definir consultas independientes SGBD, lo que lo hace portable entre distintos SGBD.
- JPQL admite solo un **subconjunto del estándar SQL**, y **no incluye características específicas** de los diferentes SGBD

# Named Queries

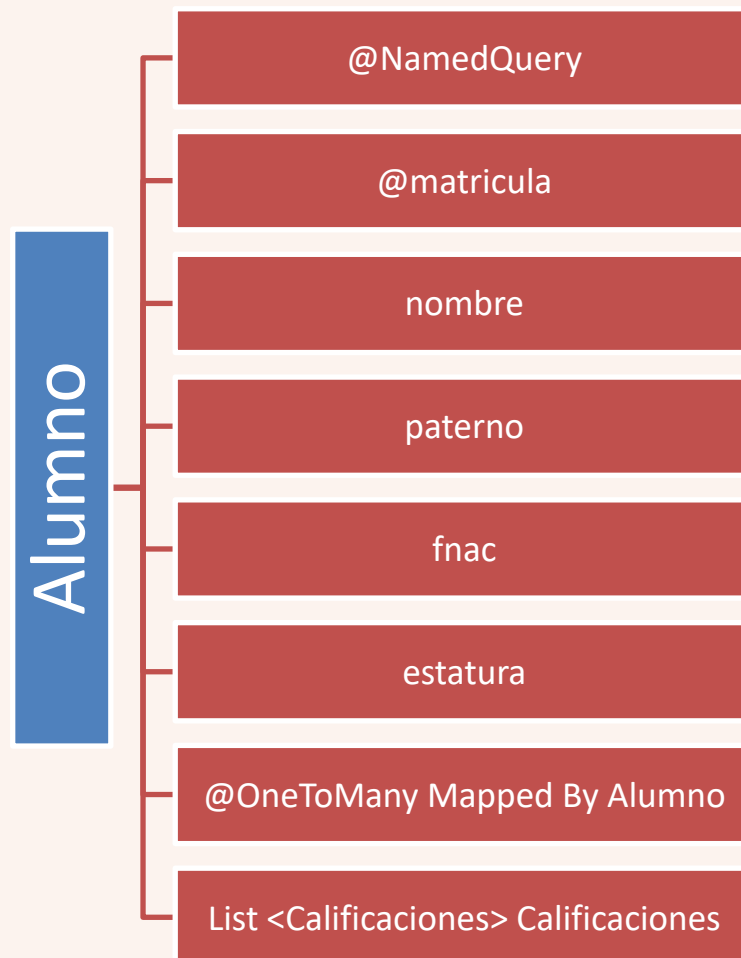
La definición de una consulta JPQL con nombre es bastante simple.

- Solo tiene que anotar una de sus clases de entidad con **@NamedQuery** y proporcionar 2 String s para el nombre y los atributos de consulta .
- El nombre de su consulta debe ser único dentro de su contexto de persistencia
- El valor del atributo **query** debe contener una declaración JPQL válida que será ejecutada en la base de datos.
- Si la consulta devuelve una entidad, la **proyección** se define de manera implícita, como en la consulta **Alumno.findByNombre** .
- En consultas más complejas como **Alumno.findByNombreAndPaterno**, se puede usar una cláusula **SELECT** para definir la proyección de manera explícita.

# Named Query y Entidades



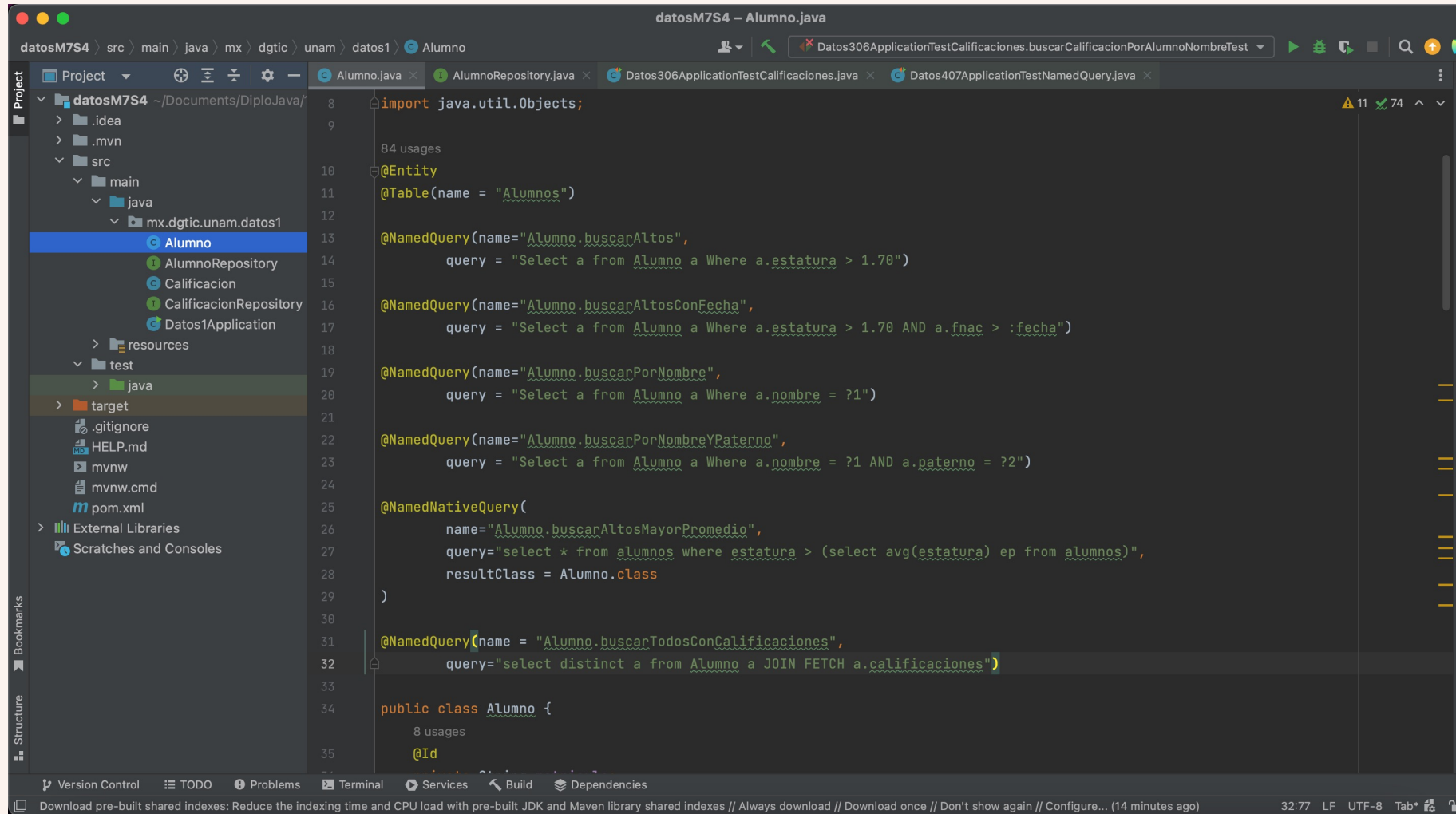
# Named Query y Parametros



select a from Alumno a  
where a.estatura > 160  
AND a.fecha > :fecha



# Named Queries (Entidad Alumno)



```
datosM7S4 - Alumno.java
datosM7S4 > src > main > java > mx > dgtic > unam > datos1 > Alumno
Alumno.java x AlumnoRepository.java x Datos306ApplicationTestCalificaciones.java x Datos407ApplicationTestNamedQuery.java x
Project
  datosM7S4 ~/Documents/DiploJava/
    .idea
    .mvn
    src
      main
        java
          mx.dgtic.unam.datos1
            Alumno
            AlumnoRepository
            Calificacion
            CalificacionRepository
            Datos1Application
          resources
          test
            java
            target
            .gitignore
            HELP.md
            mvnw
            mvnw.cmd
            pom.xml
      External Libraries
      Scratches and Consoles
Structure
Bookmarks
8 import java.util.Objects;
9
10 @Entity
11 @Table(name = "Alumnos")
12
13 @NamedQuery(name="Alumno.buscarAltos",
14             query = "Select a from Alumno a Where a.estatura > 1.70")
15
16 @NamedQuery(name="Alumno.buscarAltosConFecha",
17             query = "Select a from Alumno a Where a.estatura > 1.70 AND a.fnac > :fecha")
18
19 @NamedQuery(name="Alumno.buscarPorNombre",
20             query = "Select a from Alumno a Where a.nombre = ?1")
21
22 @NamedQuery(name="Alumno.buscarPorNombreYPaterno",
23             query = "Select a from Alumno a Where a.nombre = ?1 AND a.paterno = ?2")
24
25 @NamedNativeQuery(
26     name="Alumno.buscarAltosMayorPromedio",
27     query="select * from alumnos where estatura > (select avg(estatura) ep from alumnos)",
28     resultClass = Alumno.class
29 )
30
31 @NamedQuery(name = "Alumno.buscarTodosConCalificaciones",
32             query="select distinct a from Alumno a JOIN FETCH a.calificaciones")
33
34 public class Alumno {
35     @Id
```

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built JDK and Maven library shared indexes // Always download // Download once // Don't show again // Configure... (14 minutes ago) 32:77 LF UTF-8 Tab\*

# Named Queries (AlumnoRepository)

```
//NamedQuery
public List<Alumno> buscarAltos();
public List<Alumno> buscarAltosConFecha(Date fecha);

public List<Alumno> buscarPorNombre(String n);
public List<Alumno> buscarPorNombreYPaterno(String n, String p);

//NamedNativeQuery
public List<Alumno> buscarAltosMayorPromedio();
```



# @Query

- Permite agregar una consulta dinámica dentro de la propia clase de repositorio

```
@Query ("select avg(a.estatura) from Alumno a")  
public double findEstaturaPromedioAlumnos();
```

# AlumnoRepository

AlumnoRepository

@Query

Select avg(a.estatura) from Alumnos a

findEstaturaPromedio()

# Native Query



Las consultas SQL nativas son más potentes y flexibles que las consultas JPQL.



El proveedor de persistencia no analiza estas consultas y las envía directamente al SGBD.



Esto permite utilizar todas las funciones de SQL admitidas por la base de datos.



Pero también debe manejar los diferentes dialectos de la base de datos si necesita admitir múltiples SGBD.

# AlumnoRepository

AlumnoRepository

@Query

Select \* from alumnos

NativeQuery

findAllAlumnosNative()

# Native Query

```
//@Query en repositorio usando JPQL
1 usage

@Query ("select avg(a.estatura) from Alumno a")
public double buscarEstaturaPromedioAlumnos();

//@Query en repositorio usando SQL
1 usage

@Query (value="select distinct a.* from alumnos a\n"
        + "join calificaciones c ON(a.matricula = c.alumnos_matricula)\n"
        + "order by nombre",
        nativeQuery = true)
public List<Alumno> buscarAlumnosConCalificacion();

1 usage

@Query( value = "select * from alumnos "
        + "where nombre = ? and paterno = ? "
        + "order by nombre, paterno",
        nativeQuery = true)
public List<Alumno> buscarPorNombreYPaternoNative(String n, String p);
```

# Named Native Query

- Puede definir una consulta nativa con nombre casi de la misma manera que especifica una consulta JPQL con nombre.
  - Debe usar una *anotación* **@NamedNativeQuery**
  - El valor del atributo de consulta debe ser una declaración SQL en lugar de una declaración JPQL.
  - Puede definir una clase de entidad o una referencia a un **@SqlResultSetMapping** que se usará para mapear el resultado de su consulta.



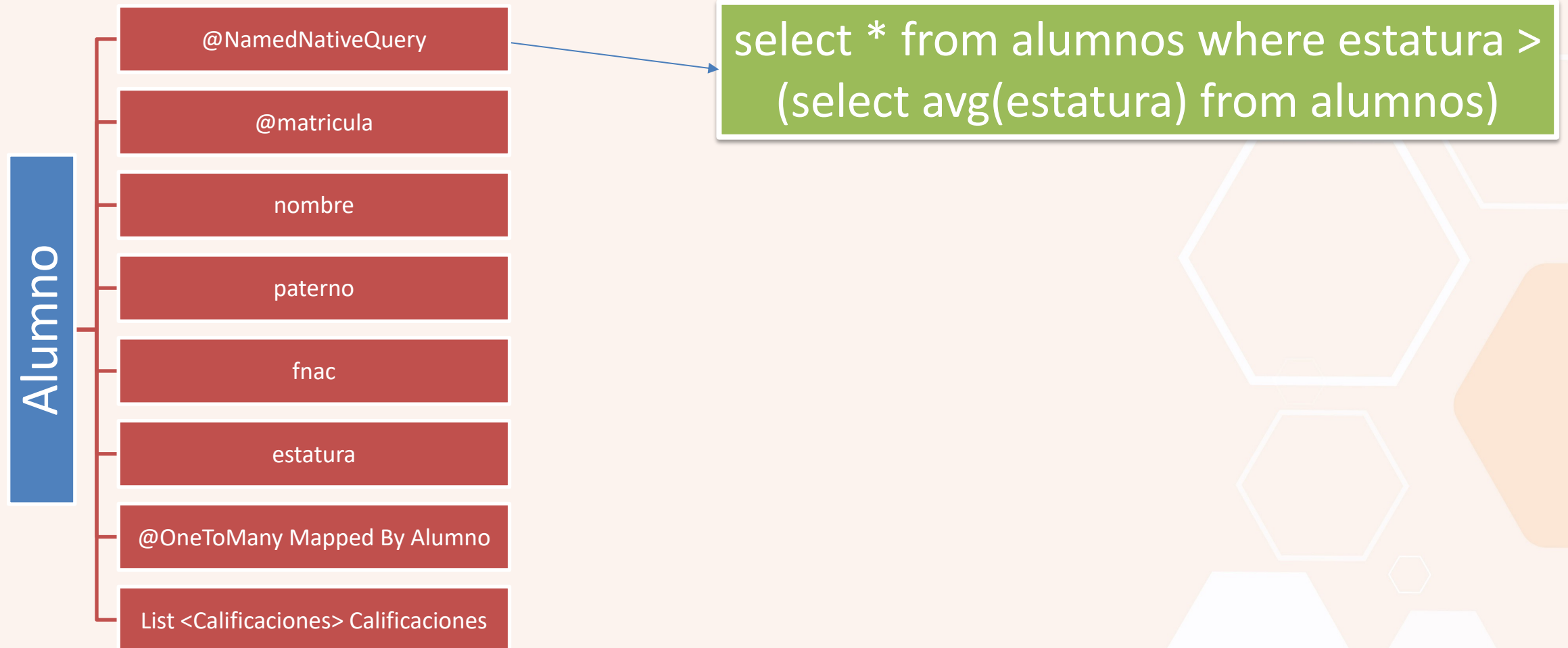
# Named Native Query

@Entity

```
@NamedNativeQuery(  
    name = "Alumno.buscarAlturaMayorPromedio",  
    query = "select * from alumnos where estatura >  
            (select avg(estatura) from alumnos)",  
    resultClass = Alumno.class)
```

```
public class Alumno { ... }
```

# Named Named Query



# AlumnoRepository

AlumnoRepository

buscarAlturaMayorPromedio()

# Contacto

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx