Persistencia de datos con Spring Data JPA

¿Qué es JPA?

- Java Persistence API (JPA) es una especificación de **Java EE** que permite a los desarrolladores Java hacer un mapeo entre los objetos y las tablas de una base de datos (Object Relational Mapping) para facilitar la administración de los datos relacionales en las aplicaciones.

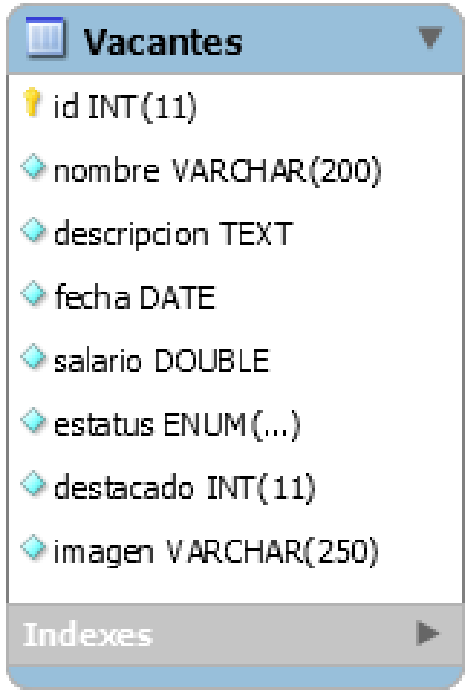
OBJECT

```
public class Vacante {

    private Integer id;
    private String nombre;
    private String descripcion;
    private Date fecha;
    private Double salario;
    private String estatus;
    private Integer destacado;
    private String imagen;
    // getters y setters ...
}
```

MAPPING

RELATIONAL



¿Qué es Spring Data JPA? (1)

- Spring Data JPA es un módulo que forma parte del proyecto **Spring Data** y básicamente nos ayuda a simplificar el desarrollo de la persistencia de datos utilizando el concepto de repositorios (algo similar al patrón de diseño DAO **D**ata **A**ccess **O**bject).
- En términos sencillos este módulo de Spring Data agrega una capa de abstracción al API de JPA (podríamos decir es una forma más sencilla y mejorada de trabajar con JPA).

Beneficios de Spring Data JPA

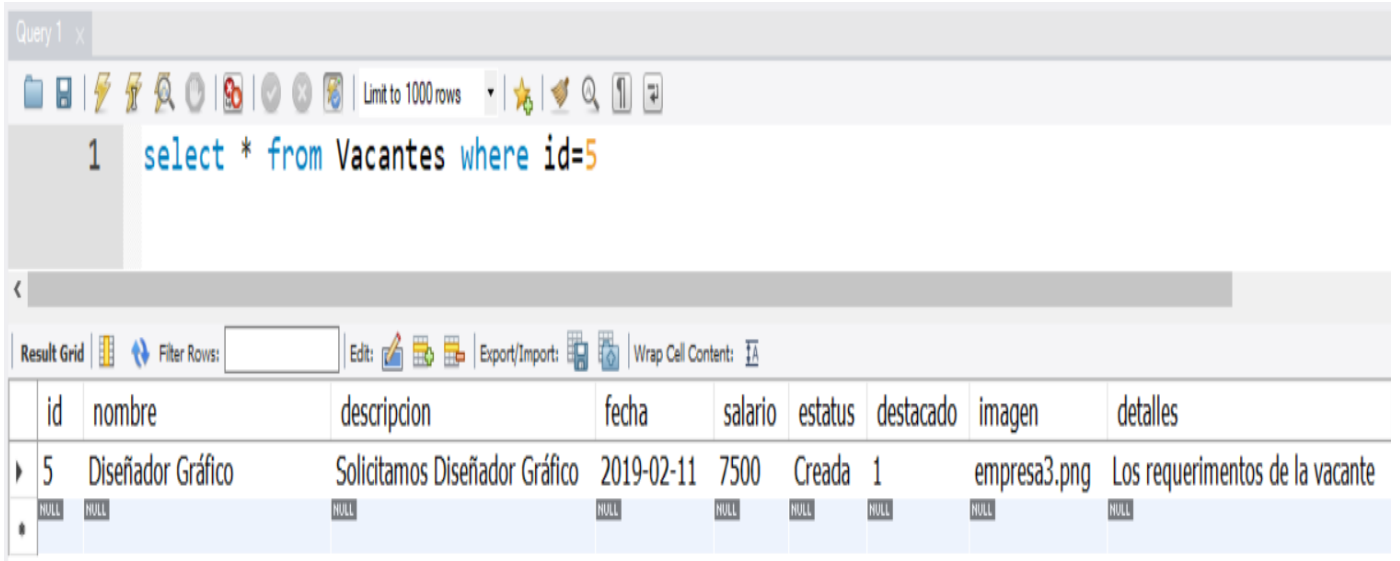
- ✓ Desarrollo ágil de la capa de persistencia de datos utilizando bases de datos relacionales.
- ✓ No es necesario escribir código SQL nativo (aunque también es posible).
- ✓ Más fácil que JDBC.
- ✓ Permite al desarrollador enfocarse más en la lógica de negocio de la aplicación y olvidarse del manejo de Excepciones (menos excepciones SQLException).
- ✓ Código más fácil de entender y mantener.

¿Qué es Spring Data JPA? (2)

Ejemplo de persistencia de datos

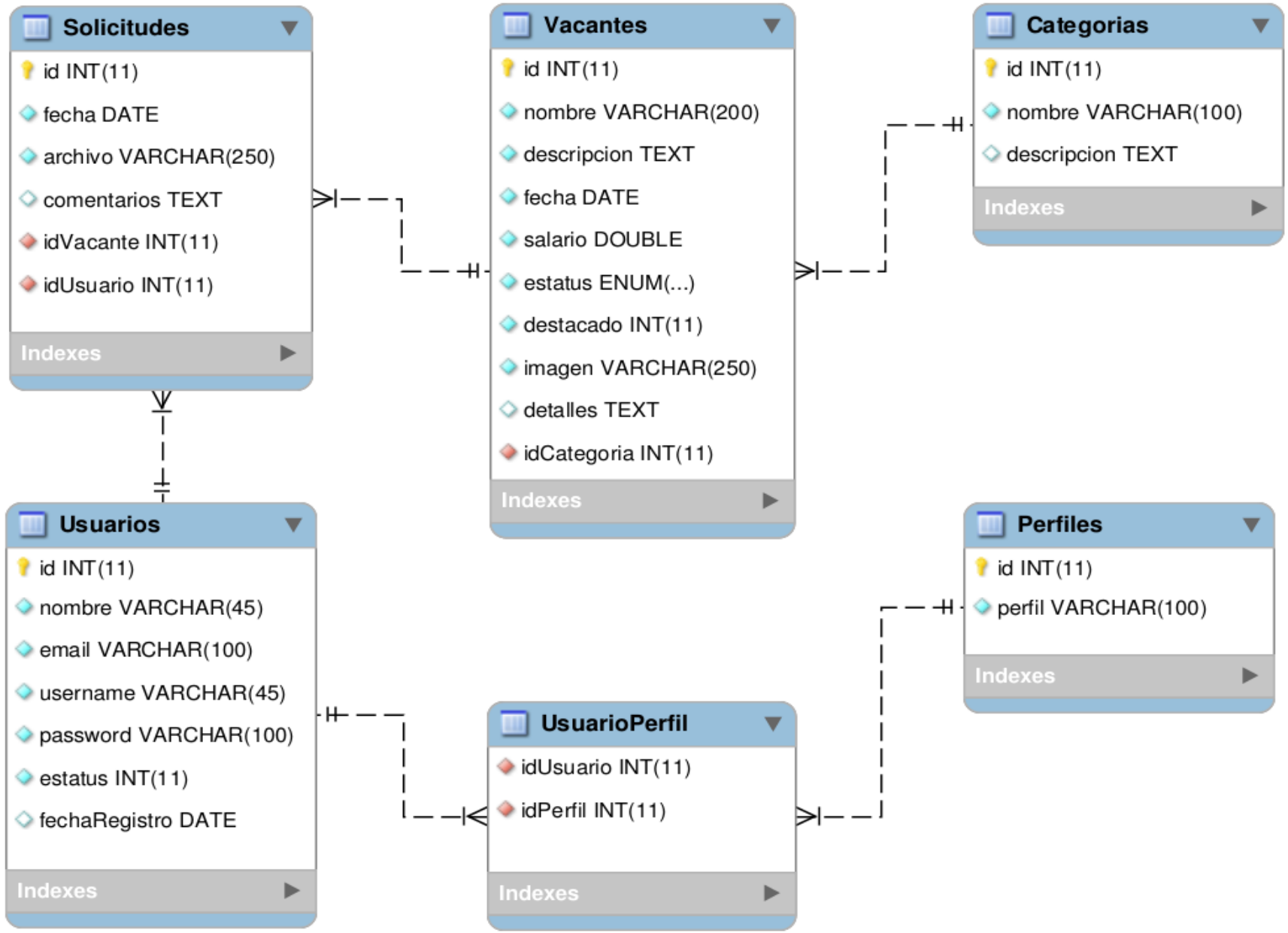
```
// 1. Crear un objeto de tipo Vacante
Vacante vacante = new Vacante();
vacante.setId(5);
vacante.setNombre("Diseñador Gráfico");
vacante.setDescripcion("Solicitamos Diseñador Gráfico");
vacante.setFecha(sdf.parse("11-02-2019"));
vacante.setSalario(7500.0);
vacante.setDestacado(1);
vacante.setImagen("empresa3.png");
vacante.setEstatus("Creada");
vacante.setDetalles("Los requerimentos de la vacante");

// 2. Persistir (guardar) objeto en la BD
repositoryVacantes.save(vacante);
```



3. Se genera todo el código SQL de forma automática y se ejecuta en la base de datos.

Estructura de la base de datos - empleosdb



Configuración del DataSource (MySQL 5.7 o inferior)

La siguientes propiedades se deben agregar para configurar el DataSource de conexión a la base de datos MySQL.

src/main/resources/application.properties

```
# DATASOURCE (MYSQL 5.7)
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/mydbname?useSSL=false
spring.datasource.username=root
spring.datasource.password=admin

#JPA
spring.jpa.generate-ddl=false
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=true
# Table names physically
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

En el archivo pom.xml por defecto viene incluido el Driver JDBC para MySQL 8. Si vamos a utilizar MySQL 5.7 o inferior, debemos especificar la versión del Driver JDBC específico. El procedimiento sería comentar la dependencia del Conector MySQL que viene por defecto y agregar la siguiente dependencia:

```
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>5.1.47</version>
</dependency>
```

Configuración del DataSource (MySQL 8.0)

La siguientes propiedades se deben agregar para configurar el DataSource de conexión a la base de datos MySQL.

src/main/resources/application.properties

```
# DATASOURCE (MYSQL 8.0)
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/mydbname?useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true
spring.datasource.username=root
spring.datasource.password=admin

#JPA
spring.jpa.generate-ddl=false
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=true
# Table names physically
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

En el archivo pom.xml por defecto viene incluido el Driver JDBC para MySQL 8. No es necesario hacer ninguna modificación.