

# Arquitecturas de Software para Aplicaciones Empresariales

## Relaciones en JPA y Spring Boot

### PROGRAMA DE INGENIERIA DE SISTEMAS

Ing. Daniel Eduardo Paz Perafán ([danielp@Unicauca.edu.co](mailto:danielp@Unicauca.edu.co))

Ing. Pablo A. Magé ([pmage@Unicauca.edu.co](mailto:pmage@Unicauca.edu.co))



En esta sesión veremos como mapear las relaciones (1 a 1, 1 a muchos y muchos a muchos) entre tablas a las clases Entity.

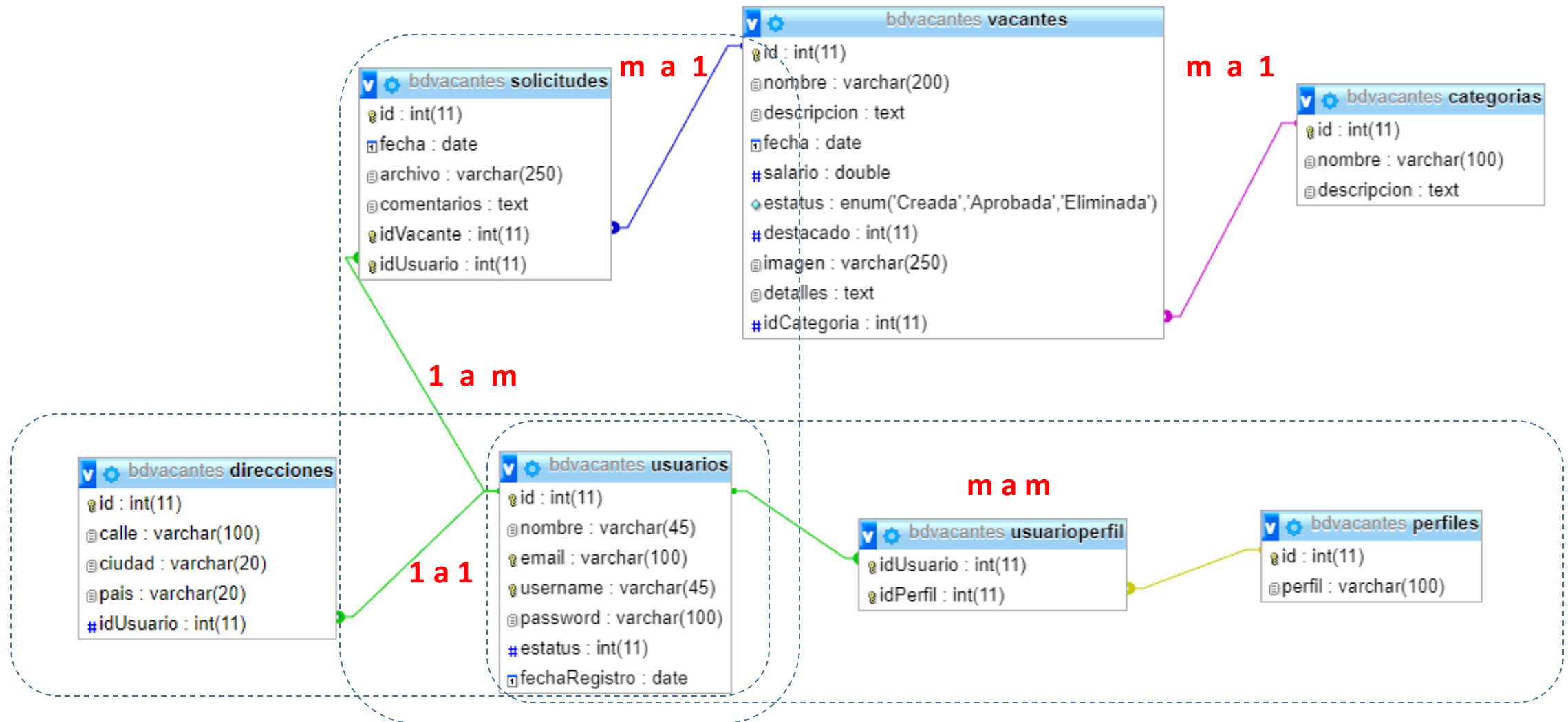
Para lograr el mapeo se utilizaran las siguientes anotaciones @oneToOne, @manyToOne, @oneToMany y @manyToMany.

En la siguiente sesión veremos ligados, transacciones y acciones en cascada

## **Explicación de la base de datos que utilizaremos para trabajar en los ejemplos**

# Base de datos para trabajar con Spring JPA

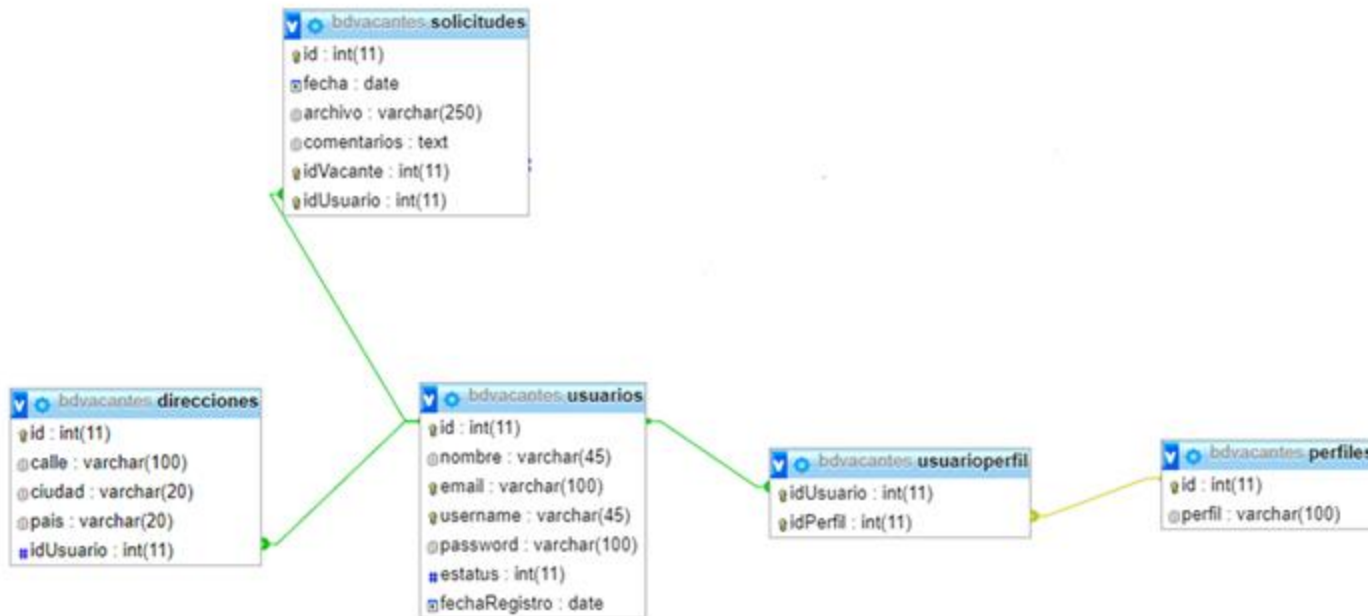
UNIVERSIDAD DEL CAUCA – FIET  
DEPARTAMENTO DE SISTEMAS



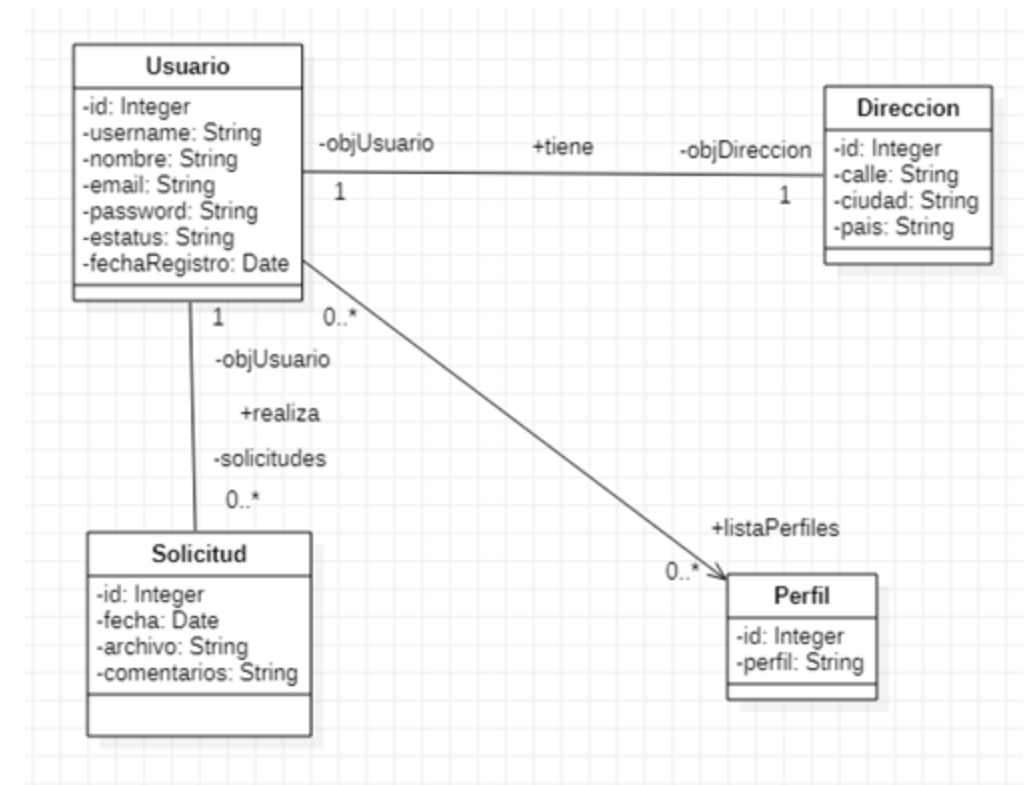
# Mapeos de relaciones entre tablas y clases

UNIVERSIDAD DEL CAUCA – FIET  
DEPARTAMENTO DE SISTEMAS

## Modelo relacional



## Mapeo entre clases



- ❖ Una transacción de base de datos es un conjunto de instrucciones que se ejecutan en bloque.
- ❖ Por ejemplo, hago una consulta, modifico un registro A en la base de datos y elimino un registro B. Si en alguna de estas instrucciones se produce un error todo el proceso se echa atrás. De esta manera si luego consulto la base de datos veré que el registro A no ha sido alterado.
- ❖ Este proceso de «tirar atrás» las instrucciones realizadas se le dice hacer un *rollback*, mientras que el proceso de confirmar todas las instrucciones en bloque una vez hemos visto que no se ha producido ningún error se le llama hacer un *commit*.

Cuando se anota una clase `@transaccional`, significa que todos los métodos públicos de esta clase son transacciones abiertas. Se aplica a cada método público individual. Spring ignora los métodos privados y protegidos.

```
@SpringBootApplication
@Transactional
public class EjemploRelacionesJpaApplication implements CommandLineRunner{

    @Autowired
    private DireccionesRepository servicioBDDirecciones;

    @Autowired
    private UsuariosRepository servicioBDUsuarios;

    @Autowired
    private SolicitudesRepository servicioDBSolicitudes;

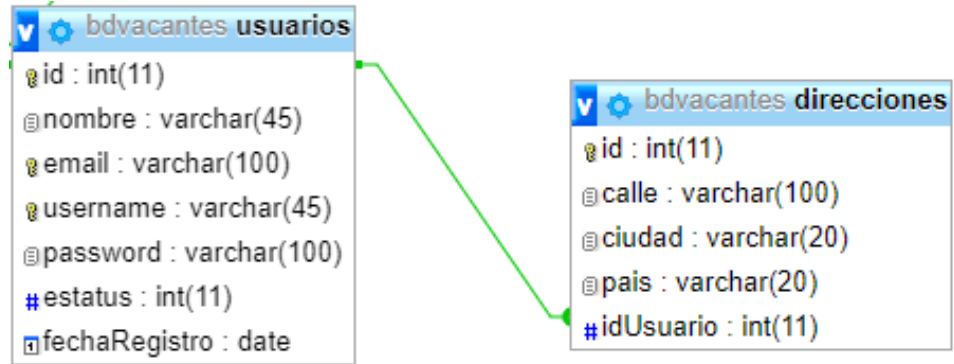
    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(EjemploRelacionesJpaApplication.class, args);
    }
}
```

## **Maapeo de la relación 1 a 1**



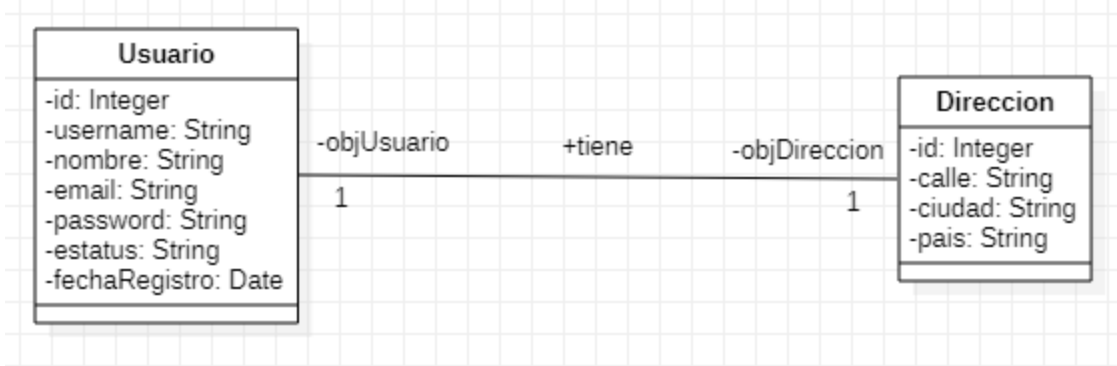
# Relación 1 a 1

Tablas de la base de datos



Entre la tabla usuarios y la tabla direcciones hay una relación de 1 a 1

Modelo



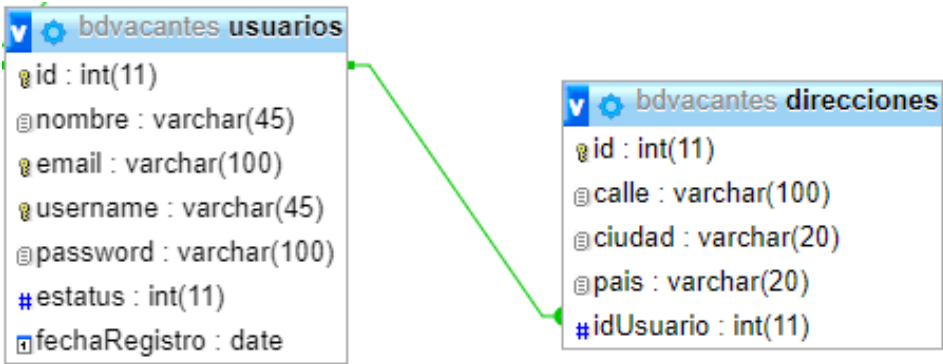
Los roles se convierten en atributos derivados

Entre la clase Usuario y la clase Direccion hay una relación de 1 a 1

En las clases no se deben colocar atributos asociados a claves foraneas

Una relación uno a uno en Java se presenta cuando el objeto de origen tiene un atributo que hace referencia a otro objeto de destino y (si) ese objeto de destino tuviera la relación inversa con el objeto de origen, también sería una relación uno a uno.

## Tablas de la base de datos



Entre la tabla usuarios y la tabla direcciones hay una relación de 1 a 1

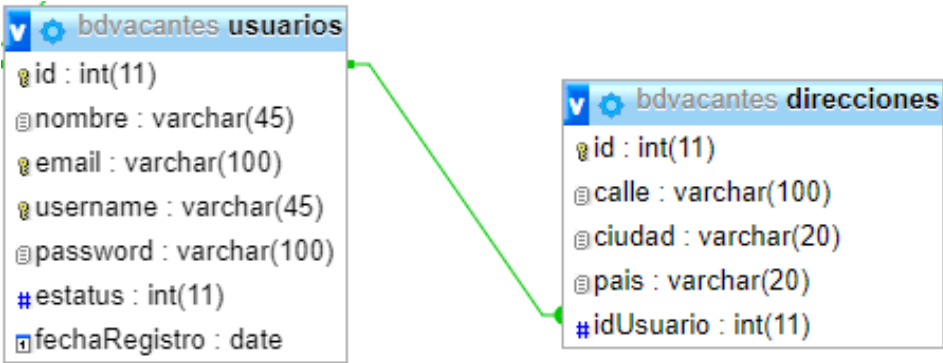
## Registros de la tabla usuarios

<div><div>←</div><div>T</div><div>→</div></div>			<div>▼</div>	id	nombre	email	username	password	estatus	fechaRegistro
<input type="checkbox"/>	<div><div></div>Editar</div>	<div><div></div>Copiar</div>	<div><div></div>Borrar</div>	1	Juan Perez	juanp@unicauca.edu.co	juanp	123	1	2020-10-01
<input type="checkbox"/>	<div><div></div>Editar</div>	<div><div></div>Copiar</div>	<div><div></div>Borrar</div>	2	Calatina Serna	catalinaS@unicauca.edu.co	catalinas	123	2	2020-10-02
<input type="checkbox"/>	<div><div></div>Editar</div>	<div><div></div>Copiar</div>	<div><div></div>Borrar</div>	3	Andres Sanchez	andres@unicauca.edu.co	andres	123	3	2020-10-03

## Registros de la tabla direcciones

						id	calle	ciudad	pais	idUsuario	
<input type="checkbox"/>		Editar		Copiar		Borrar	1	calle 5 20 A 33	Bogota	Colombia	1
<input type="checkbox"/>		Editar		Copiar		Borrar	2	calle 4 5 N 20	Cali	Colombia	2
<input type="checkbox"/>		Editar		Copiar		Borrar	3	calle 6 25 A 70	Medellin	Colombia	3

## Tablas de la base de datos

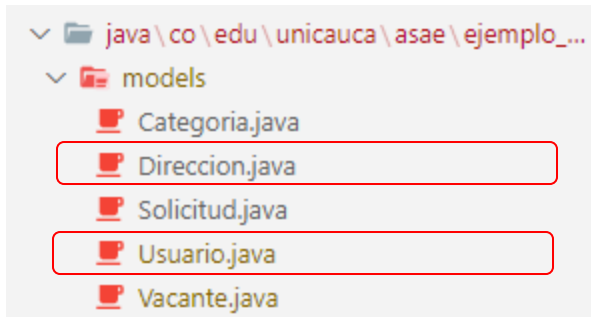


Entre la tabla usuarios y la tabla direcciones hay una relación de 1 a 1

**select \* from usuarios INNER JOIN direcciones ON usuarios.id=direcciones.idUsuario**

id	nombre	email	username	password	estatus	fechaRegistro	id	calle	ciudad	pais	idUsuario
1	Juan Perez	juanp@unicauca.edu.co	juanp	123	1	2020-10-01	1	calle 5 20 A 33	Bogota	Colombia	1
2	Calatina Serna	catalinaS@unicauca.edu.co	catalinas	123	2	2020-10-02	2	calle 4 5 N 20	Cali	Colombia	2
3	Andres Sanchez	andres@unicauca.edu.co	andres	123	3	2020-10-03	3	calle 6 25 A 70	Medellin	Colombia	3

## Crear las clases del modelo



```
package co.edu.unicauca.asae.core.modelo;

public class Direccion {

    private Integer id;
    private String calle;
    private String ciudad;
    private String pais;

    private Usuario objUsuario;

    public Integer getId() { }

    public void setId(Integer id) { }

    public String getCalle() { }

    public void setCalle(String calle) { }

    public String getCiudad() { }

    public void setCiudad(String ciudad) { }

    public String getPais() { }

    public void setPais(String pais) { }

    public Usuario getObjUsuario() { }

    public void setObjUsuario(Usuario objUsuario) { }

}
```

```
package co.edu.unicauca.asae.core.modelo;

import java.util.Date;
```

```
public class Usuario {

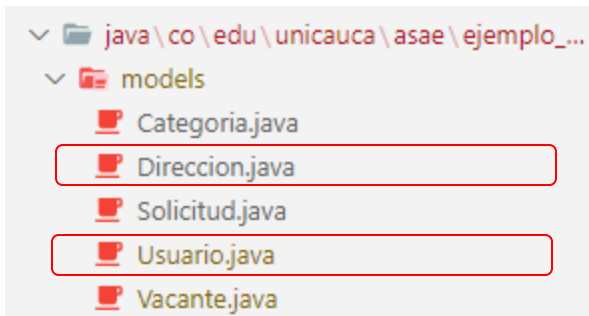
    private Integer id;
    private String username;
    private String nombre;
    private String email;
    private String password;
    private Integer estatus;
    private Date fechaRegistro;

    private Direccion objDireccion;

    public Integer getId() { }
    public void setId(Integer id) { }
    public String getUsername() { }
    public void setUsername(String username) { }
    public String getNombre() { }
    public void setNombre(String nombre) { }
    public String getEmail() { }
    public void setEmail(String email) { }
    public String getPassword() { }
    public void setPassword(String password) { }
    public Integer getEstatus() { }
    public void setEstatus(Integer estatus) { }
    public Date getFechaRegistro() { }
    public void setFechaRegistro(Date fechaRegistro) { }
    public Direccion getObjDireccion() { }
    public void setObjDireccion(Direccion objDireccion) { }

}
```

## Asociar las clases del modelo con las tablas de la BD



```
package co.edu.unicauca.asae.core.modelo;

import javax.persistence.Entity;

@Entity
@Table(name="direcciones")
public class Direccion {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String calle;
    private String ciudad;
    private String pais;

    private Usuario objUsuario;

    public Integer getId() {}
    public void setId(Integer id) {}
    public String getCalle() {}
    public void setCalle(String calle) {}
    public String getCiudad() {}
    public void setCiudad(String ciudad) {}
    public String getPais() {}
    public void setPais(String pais) {}
    public Usuario getObjUsuario() {}
    public void setObjUsuario(Usuario objUsuario) {}

}
```

```
package co.edu.unicauca.asae.core.modelo;

import java.util.Date;

@Entity
@Table(name = "Usuarios")
public class Usuario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String username;
    private String nombre;
    private String email;
    private String password;
    private Integer estatus;
    private Date fechaRegistro;

    public Integer getId() {}
    public void setId(Integer id) {}
    public String getUsername() {}
    public void setUsername(String username) {}
    public String getNombre() {}
    public void setNombre(String nombre) {}
    public String getEmail() {}
    public void setEmail(String email) {}
    public String getPassword() {}
    public void setPassword(String password) {}
    public Integer getEstatus() {}
    public void setEstatus(Integer estatus) {}
    public Date getFechaRegistro() {}
    public void setFechaRegistro(Date fechaRegistro) {}

}
```

## Utilizar la notación @oneToOne y @JoinColumn

```
package co.edu.unicauca.asae.core.modelo;

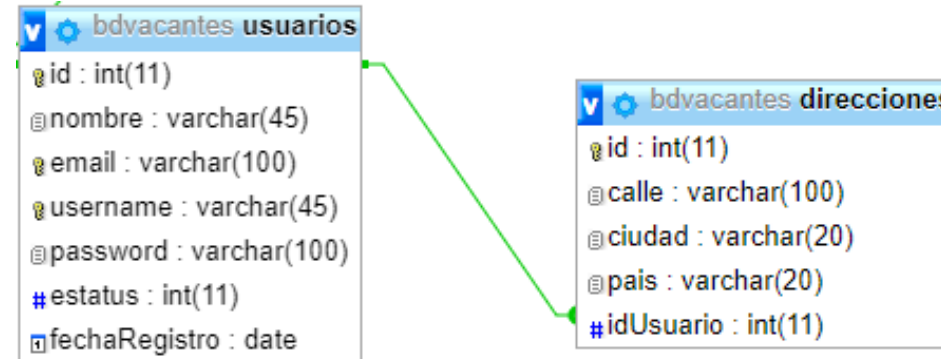
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name="direcciones")
public class Direccion {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String calle;
    private String ciudad;
    private String pais;

    @OneToOne
    @JoinColumn(name="idUser")
    private Usuario objUsuario;

    public Integer getId() {}
    public void setId(Integer id) {}
    public String getCalle() {}
    public void setCalle(String calle) {}
    public String getCiudad() {}
    public void setCiudad(String ciudad) {}
    public String getPais() {}
    public void setPais(String pais) {}
    public Usuario getObjUsuario() {}
    public void setObjUsuario(Usuario objUsuario) {}
}
```



En JPA, una relación uno a uno se define mediante la anotación **@OneToOne**

En JPA, la anotación **@JoinColumn** se utiliza para definir cual columna contiene la clave foránea

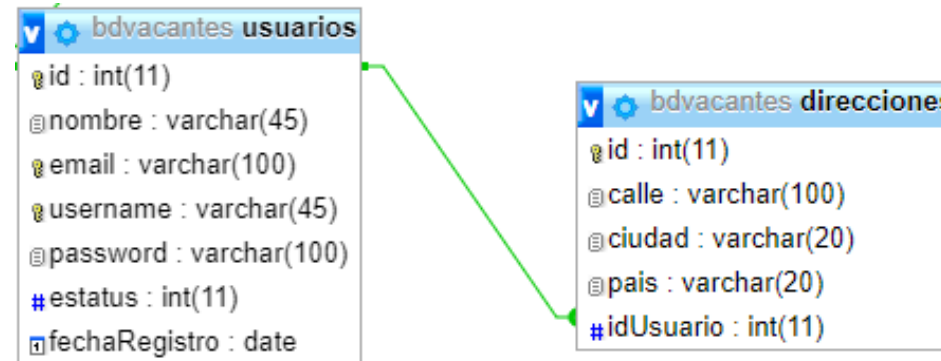


## Utilizar la notación @oneToOne y la propiedad mappedBy

```
import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "Usuarios")
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String username;
    private String nombre;
    private String email;
    private String password;
    private Integer estatus;
    private Date fechaRegistro;

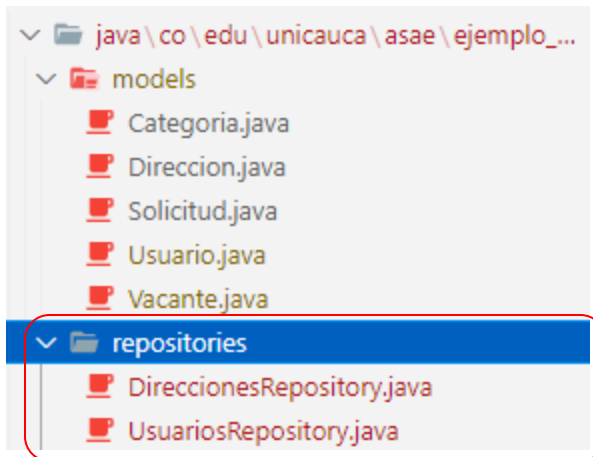
    @OneToOne(mappedBy = "objUsuario")
    private Direccion objDireccion;
}
```



En JPA, una relación uno a uno se define mediante la anotación **@OneToOne**.

En JPA, la propiedad **mappedBy** de la anotación **@OneToOne** permite hacer una relación bidireccional

## Crear los repositorios



### Repositorio para gestionar direcciones

```
package co.edu.unicauca.asae.core.repository;

import org.springframework.data.repository.CrudRepository;

import co.edu.unicauca.asae.core.modelo.Direccion;

public interface DireccionesRepository extends CrudRepository<Direccion, Integer> {

}
```

### Repositorio para gestionar usuarios

```
package co.edu.unicauca.asae.core.repository;

import org.springframework.data.repository.CrudRepository;

import co.edu.unicauca.asae.core.modelo.Usuario;

public interface UsuariosRepository extends CrudRepository<Usuario, Integer> {

}
```



- Crear un método que permita listar las direcciones, las cuales en su interior tienen el usuario asociado
- Crear un método que permita listar los usuarios, los cuales en su interior tiene la dirección asociada

```
package co.edu.unicauca.asae.core;

import org.springframework.beans.factory.annotation.Autowired;

@SpringBootApplication
public class ProyectoJpaRelacionesV2Application implements CommandLineRunner{

    @Autowired
    private DireccionesRepository servicioBDdirecciones;

    @Autowired
    private UsuariosRepository serviciosBDUsuarios;

    public static void main(String[] args) {
        SpringApplication.run(ProyectoJpaRelacionesV2Application.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        consultarUsuarios();
    }
}
```

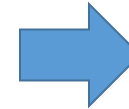
```
private void consultarDirecciones()
{
    Iterable<Direccion> listaDirecciones= this.servicioBDdirecciones.findAll();
    for (Direccion direccion : listaDirecciones) {
        System.out.println("Calle: " + direccion.getCalle());
        System.out.println("Ciudad: " + direccion.getCiudad());
        System.out.println("País: " + direccion.getPais());
        System.out.println("usuario asociado");
        System.out.println("Nombres y apellidos " + direccion.getObjUsuario().getNombre());
        System.out.println(" --- ----");
    }
}

private void consultarUsuarios()
{
    Iterable<Usuario> listaUsuarios= this.serviciosBDUsuarios.findAll();

    for (Usuario usuario : listaUsuarios) {
        System.out.println("Usuario");
        System.out.println("Nombres: " + usuario.getNombre());
        System.out.println("Dirección: ");
        System.out.println("Calle: " + usuario.getObjDireccion().getCalle());
        System.out.println("Ciudad: " + usuario.getObjDireccion().getCiudad());
        System.out.println("País: " + usuario.getObjDireccion().getPais());
    }
}
```

- Resultados

```
private void consultarDirecciones()
{
    Iterable<Direccion> listaDirecciones= this.servicioBDdirecciones.findAll();
    for (Direccion direccion : listaDirecciones) {
        System.out.println("Calle: " + direccion.getCalle());
        System.out.println("Ciudad: " + direccion.getCiudad());
        System.out.println("Pais: " + direccion.getPais());
        System.out.println("usuario asociado");
        System.out.println("Nombres y apellidos " + direccion.getObjUsuario().getNombre());
        System.out.println(" --- ----");
    }
}
```



```
Calle: calle 5 20 A 33
Ciudad: Bogota
Pais: Colombia
usuario asociado
Nombres y apellidos Juan Perez
--- ----
Calle: calle 4 5 N 20
Ciudad: Cali
Pais: Colombia
usuario asociado
Nombres y apellidos Calatina Serna
--- ----
Calle: calle 6 25 A 70
Ciudad: Medellin
Pais: Colombia
usuario asociado
Nombres y apellidos Andres Sanchez
```

```
private void consultarUsuarios()
{
    Iterable<Usuario> listaUsuarios= this.serviciosBDUsuarios.findAll();

    for (Usuario usuario : listaUsuarios) {
        System.out.println("Usuario");
        System.out.println("Nombres: " + usuario.getNombre());
        System.out.println("Dirección: ");
        System.out.println("Calle: " + usuario.getObjDireccion().getCalle());
        System.out.println("Ciudad: " + usuario.getObjDireccion().getCiudad());
        System.out.println("Pais: " + usuario.getObjDireccion().getPais());
    }
}
```



```
Usuario
Nombres: Juan Perez
Dirección:
Calle: calle 5 20 A 33
Ciudad: Bogota
Pais: Colombia
Usuario
Nombres: Calatina Serna
Dirección:
Calle: calle 4 5 N 20
Ciudad: Cali
Pais: Colombia
Usuario
Nombres: Andres Sanchez
Dirección:
Calle: calle 6 25 A 70
Ciudad: Medellin
Pais: Colombia
```

# Relación 1 a 1

- Crear un método que permita almacenar un usuario y su dirección

```
private void almacenarUsuario()
{
    Usuario user = new Usuario();
    user.setNombre("Andrea Sanchez");
    user.setEmail("Andrea@unicauca.edu.co");
    user.setFechaRegistro(new Date());
    user.setUsername("Andrea");
    user.setPassword("12345");
    user.setEstatus(1);

    Usuario objUsuarioAlmacenado=this.servicioBDUsuarios.save(user);

    System.out.println("Id generado para el usuario: " + objUsuarioAlmacenado.getId());

    Direccion objDireccion= new Direccion();
    objDireccion.setCalle("calle 8 no 23 A 34");
    objDireccion.setCiudad("palmira");
    objDireccion.setPais("Colombia");

    objDireccion.setObjUsuario(objUsuarioAlmacenado);

    Direccion objDireccionAlmacenada=this.servicioBDDirecciones.save(objDireccion);
    System.out.println("id almacenado: " + objDireccionAlmacenada.getId());
}
```

Hibernate: insert into Usuarios (email, estatus, fechaRegistro, nombre, password, username) values (?, ?, ?, ?, ?, ?)  
id generado para el usuario: 10  
Hibernate: insert into direcciones (calle, ciudad, idUsuario, pais) values (?, ?, ?, ?)  
id almacenado: 5

Tabla  
usuarios



usuarios

<div><div><div></div><div></div><div></div></div></div>			id	nombre	email	username	password	estatus	fechaRegistro
<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	1	Juan Perez	juanp@unicauca.edu.co	juanp	123	1	2020-10-01
<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	2	Catalina Serna	catalinaS@unicauca.edu.co	catalinas	123	2	2020-10-02
<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	3	Andres Sanchez	andres@unicauca.edu.co	andres	123	3	2020-10-03
<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	9	Daniel Paz	danielp2@unicauca.edu.co	danielp2	12345	1	2020-10-16
<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	<div><div><div></div><div></div><div></div></div></div>	10	Daniel Paz	danielp3@unicauca.edu.co	danielp3	12345	1	2020-10-16



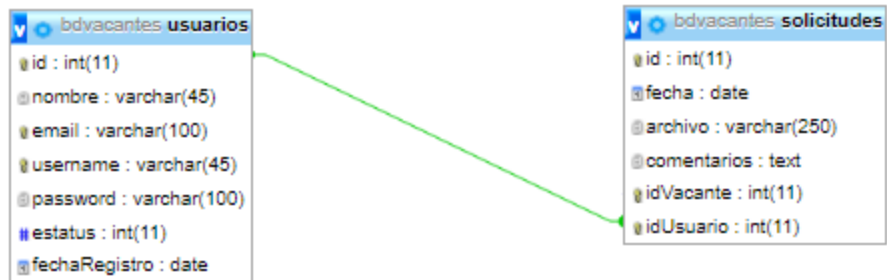
Tabla  
direcciones

direcciones

						calle	ciudad	pais	idUsuario
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	1		calle 5 20 A 33	Bogota	Colombia	1
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	2		calle 4 5 N 20	Cali	Colombia	2
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	3		calle 6 25 A 70	Medellin	Colombia	3
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	4		calle 4 no 23 A 34	Popayán	Colombia	9
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	5		calle 4 no 23 A 34	Popayán	Colombia	10

## **Maapeo de la relación muchos a 1**

## Tablas de la base de datos

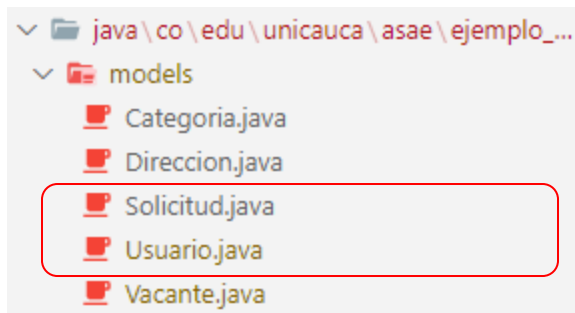


Entre la tabla usuarios y solicitudes  
hay una relación de 1 a muchos

**select \* from usuarios INNER JOIN solicitudes ON usuarios.id=solicitudes.idUsuario**

+ Opciones												
id	nombre	email	username	password	estatus	fechaRegistro	id	fecha	archivo	comentarios	idVacante	idUsuario
1	Juan Perez	juanp@unicauca.edu.co	juanp	123	1	2020-10-01	1	2021-11-09	ruta al archivo de la hoja de vida	Tengo un amplio conocimiento en análisis contable	1	1
1	Juan Perez	juanp@unicauca.edu.co	juanp	123	1	2020-10-01	2	2021-11-10	ruta al archivo de la hoja de vida	Deseo unirme al Socio Logístico con mayor presenci...	2	1
1	Juan Perez	juanp@unicauca.edu.co	juanp	123	1	2020-10-01	3	2021-11-04	ruta a la hoja de vida	Tengo experiencia como coordinador o coordinadora ...	3	1

## Crear las clases del modelo



```
package co.edu.unicauca.asae.ejemplo_relaciones_jpa.models;

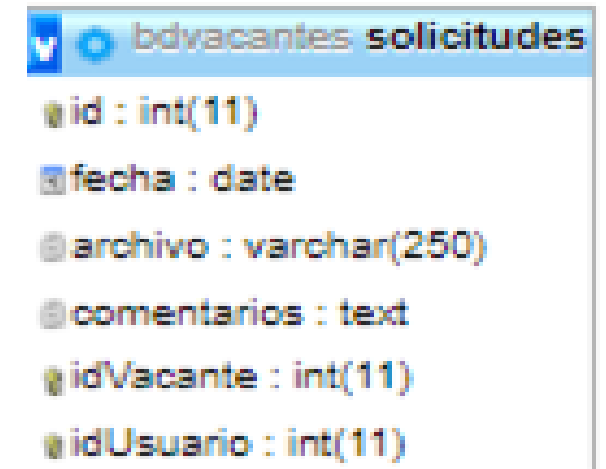
import java.util.Date;

public class Solicitud {

    private Integer id;
    private Date fecha;
    private String archivo;
    private String comentarios;
    private Usuario objUsuario;

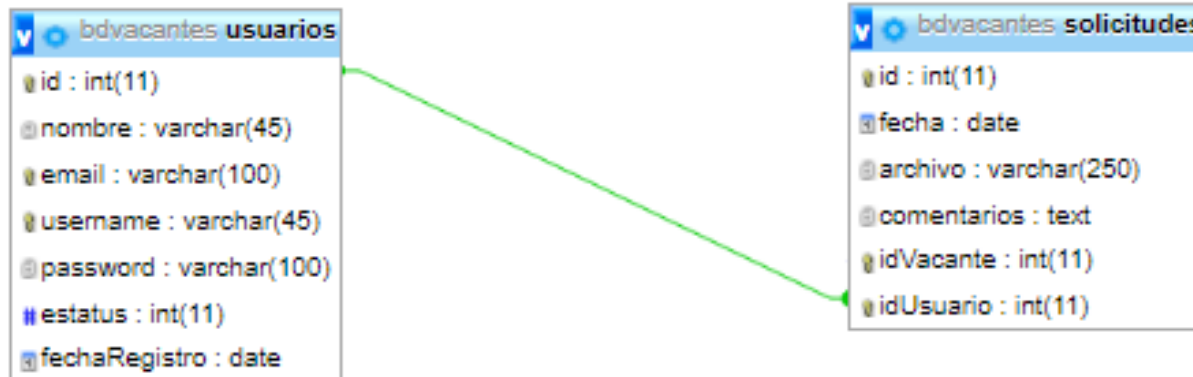
    public Integer getId() { ...
    public void setId(Integer id) { ...
    public Date getFecha() { ...
    public void setFecha(Date fecha) { ...
    public String getArchivo() { ...
    public void setArchivo(String archivo) { ...
    public String getComentarios() { ...
    public void setComentarios(String comentarios) { ...
    public Usuario getObjUsuario() { ...
    public void setObjUsuario(Usuario objUsuario) { ...

}
```



## Crear las clases del modelo

Como mapear esa relación entre la clase Vacante y Categoría



```
package co.edu.unicauca.asae.ejemplo_relaciones_jpa.models;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
```

```
public class Usuario {

    private Integer id;
    private String username;
    private String nombre;
    private String email;
    private String password;
    private Integer estatus;
    private Date fechaRegistro;
    private Direccion objDireccion;
    private List<Solicitud> solicitudes;

    public Usuario()
    {
        this.solicitudes=new ArrayList<Solicitud>();
    }

    public void agregarSolicitud(Solicitud objSolicitud)
    {
        this.solicitudes.add(objSolicitud);
    }

    public List<Solicitud> getSolicitudes()
    {
        return this.solicitudes;
    }
}
```



## Asociar las clases del modelo con las tablas de la BD

```
@Entity
@Table(name = "Usuarios")
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String username;
    private String nombre;
    private String email;
    private String password;
    private Integer estatus;
    private Date fechaRegistro;

    @OneToOne(mappedBy = "objUsuario")
    private Direccion objDireccion;

    @OneToMany(fetch = FetchType.EAGER, mappedBy="objUsuario")
    private List<Solicitud> solicitudes;

    public Usuario()
    {
        this.solicitudes=new ArrayList<Solicitud>();
    }

    public void agregarSolicitud(Solicitud objSolicitud)
    {
        this.solicitudes.add(objSolicitud);
    }
}
```

```
@Entity
@Table(name = "Solicitudes")
public class Solicitud {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private Date fecha;
    private String archivo;
    private String comentarios;

    @ManyToOne
    @JoinColumn(name="idUsuario", nullable=false)
    private Usuario objUsuario;
}
```

En JPA, una relación muchos a uno se define mediante la anotación `@ManyToOne`, `@OneToOne`

En JPA, la anotación `@JoinColumn` se utiliza para definir cual columna contiene la clave foránea

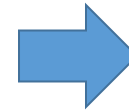
En JPA, la propiedad `FetchType.EAGER` indica que siempre que se consulte un usuario se cargaran sus solicitudes. (Se explicara detalladamente la siguiente sesión)



- Crear un método que permita listar los usuarios, junto con sus solicitudes

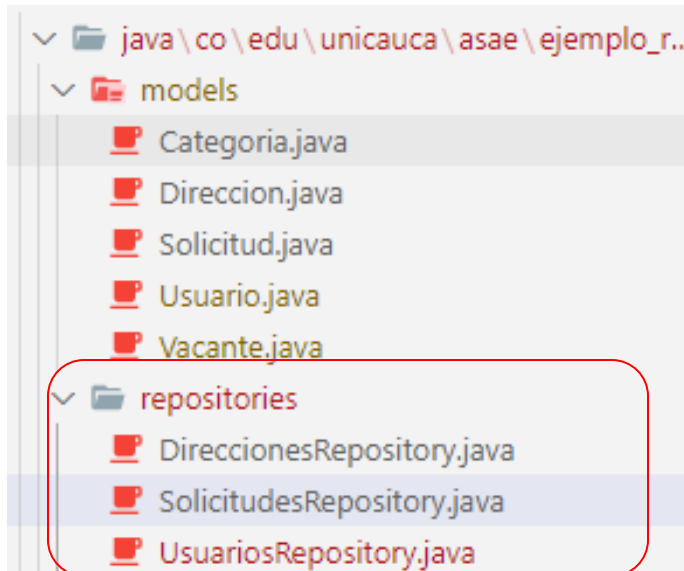
```
private void consultarUsuariosSolicitudes()
{
    Iterable<Usuario> listaUsuarios= this.servicioBDUsuarios.findAll();

    for (Usuario usuario : listaUsuarios) {
        System.out.println("Usuario");
        System.out.println("Nombres: " + usuario.getNombre());
        System.out.println("Solicitudes");
        Iterable<Solicitud> listaSolicitudes=usuario.getSolicitudes();
        for (Solicitud solicitud : listaSolicitudes) {
            System.out.println("id de la solicitud: " + solicitud.getId());
            System.out.println("Fecha de la solicitud: " + solicitud.getFecha());
            System.out.println("Ruta a la hoja de vida: " + solicitud.getArchivo());
            System.out.println("Comentarios: " + solicitud.getComentarios());
        }
        System.out.println(" ---- ---- ----");
    }
}
```



```
TERMINAL  PROBLEMAS 10 SALIDA  CONSOLA DE DEPURACIÓN
in Solicitudes solicitud2_ on usuario1_id=solicitud2_idUsuario where di
Usuario
Nombres: Juan Perez
Solicitudes
id de la solicitud: 1
Fecha de la solicitud: 2021-11-09 00:00:00.0
Ruta a la hoja de vida: ruta al archivo de la hoja de vida
Comentarios: Tengo un amplio conocimiento en análisis contable
id de la solicitud: 2
Fecha de la solicitud: 2021-11-10 00:00:00.0
Ruta a la hoja de vida: ruta al archivo de la hoja de vida
Comentarios: Deseo unirme al Socio Logístico con mayor presencia en México,
id de la solicitud: 3
Fecha de la solicitud: 2021-11-04 00:00:00.0
Ruta a la hoja de vida: ruta a la hoja de vida
Comentarios: Tengo experiencia como coordinador o coordinadora de marketing
---- ---- ----
Usuario
Nombres: Calatina Serna
Solicitudes
---- ---- ----
Usuario
Nombres: Andres Sanchez
Solicitudes
```

## Crear los repositorios



## Repositorio para gestionar Solicitudes

```
package co.edu.unicauca.asae.ejemplo_relaciones_jpa.repositories;

import org.springframework.data.repository.CrudRepository;
import co.edu.unicauca.asae.ejemplo_relaciones_jpa.models.Solicitud;

public interface SolicitudesRepository extends CrudRepository<Solicitud, Integer> {

}
```

- Crear un método que permita guardar un usuario y un conjunto de solicitudes

```
private void almacenarSolicitudes() {
    Usuario objUsuario = new Usuario();
    objUsuario.setNombre(nombre:"Tatiana Acosta");
    objUsuario.setEmail(email:"tatiana@unicauca.edu.co");
    objUsuario.setFechaRegistro(new Date());
    objUsuario.setUsername(username:"Tatiana");
    objUsuario.setPassword(password:"123456");
    objUsuario.setEstatus(estatus:2);

    Vacante objVacante1=this.servicioBDVacanes.findById(1).get();

    Vacante objVacante2=this.servicioBDVacanes.findById(2).get();

    Vacante objVacante3=this.servicioBDVacanes.findById(3).get();

    this.servicioBDUsuarios.save(objUsuario);

    Solicitud objSolicitud1 = new Solicitud();
    objSolicitud1.setObjUsuario(objUsuario);
    objSolicitud1.setFecha(new Date());
    objSolicitud1.setArchivo(archivo:"Ruta al archivo de la solicitud 1");
    objSolicitud1.setComentarios(comentarios:"Comentarios de la solicitud 1");
    objSolicitud1.setObjVacante(objVacante1);

    this.servicioDBSolicitudes.save(objSolicitud1);
}
```

## **Maapeo de la relación muchos a muchos sin una clase intermedia**

# Relación muchos a muchos



```
select usuarios.nombre, perfiles.perfil from usuarios inner JOIN usuarioperfil on usuarios.id=usuarioperfil.idUsuario INNER JOIN perfiles on usuarioperfil.idPerfil=perfiles.id
```

nombre	perfil
Andres Sanchez	SUPERVISOR
Juan Perez	ADMINISTRADOR
Calatina Serna	ADMINISTRADOR
Andres Sanchez	ADMINISTRADOR
Juan Perez	USUARIO
Calatina Serna	USUARIO

## Asociar las clases del modelo con las tablas de la BD

```
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.Table;
```

```
@Entity  
@Table(name = "Perfiles")  
public class Perfil {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer id;  
    private String perfil;  
  
    public Integer getId() {  
  
    public void setId(Integer id) {  
  
    public String getPerfil() {  
  
    public void setPerfil(String perfil) {  
  
    }  
}
```

```
import java.util.Date;
```

```
@Entity  
@Table(name = "Usuarios")  
public class Usuario {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer id;  
    private String username;  
    private String nombre;  
    private String email;  
    private String password;  
    private Integer estatus;  
    private Date fechaRegistro;
```

```
@OneToOne( mappedBy = "objUsuario")  
private Direccion objDireccion;  
  
@ManyToMany  
@JoinTable(name="UsuarioPerfil",  
            joinColumns = @JoinColumn(name="idUsuario"),  
            inverseJoinColumns = @JoinColumn(name="idPerfil")  
            )  
private List<Perfil> perfiles;
```

```
package co.edu.unicauca.asae.core.modelo;

import java.util.Date;

@Entity
@Table(name = "Usuarios")
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String username;
    private String nombre;
    private String email;
    private String password;
    private Integer estatus;
    private Date fechaRegistro;

    @OneToOne(mappedBy = "objUsuario")
    private Direccion objDireccion;

    @ManyToMany(fetch=FetchType.EAGER)
    @JoinTable(name="UsuarioPerfil",
        joinColumns = @JoinColumn(name="idUsuario"),
        inverseJoinColumns = @JoinColumn(name="idPerfil")
    )
    private List<Perfil> perfiles;
```

En JPA, una relación muchos a muchos se define mediante la anotación **@ManyToMany**

En JPA, la anotación **@JoinTable** se utiliza para definir cual tabla contiene las claves foráneas de la tabla usuario y perfiles

En JPA, la anotación **@JoinColumn** permite establecer la columna que contiene la clave foránea

## Crear un método que permita buscar un usuario

```
package co.edu.unicauca.asae.core;

import java.util.Date;

@SpringBootApplication
public class ProyectoJpaRelacionesV2Application implements CommandLineRunner{

    @Autowired
    private DireccionesRepository servicioBDdirecciones;

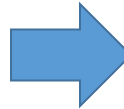
    @Autowired
    private UsuariosRepository servicioBDUsuarios;

    @Autowired
    private VacantesRepository servicioBDVacantes;

    public static void main(String[] args) {
        SpringApplication.run(ProyectoJpaRelacionesV2Application.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        buscarUsuario();
    }

    public void buscarUsuario() {
        Optional<Usuario> optional = servicioBDUsuarios.findById(1);
        if (optional.isPresent()) {
            Usuario u = optional.get();
            System.out.println("Usuario: " + u.getNombre());
            System.out.println("Perfiles asignados");
            for (Perfil p : u.getPerfiles()) {
                System.out.println(p.getPerfil());
            }
        } else {
            System.out.println("Usuario no encontrado");
        }
    }
}
```

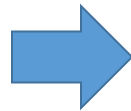


```
Hibernate: select usuario0_.id as id1_4_0_, usuario0_.email as email2_4_0_, usuario0_.estatus as estatus3_4_0_
Usuario: Juan Perez
Perfiles asignados
ADMINISTRADOR
USUARIO
```



## Crear un método que permita crear un usuario con dos perfiles

```
private void crearUsuarioConDosPerfiles() {  
    Usuario objUsuario = new Usuario();  
    objUsuario.setNombre("Daniel Paz");  
    objUsuario.setEmail("danielp@unicauca.edu.co.com");  
    objUsuario.setFechaRegistro(new Date());  
    objUsuario.setUsername("danielp");  
    objUsuario.setPassword("12345");  
    objUsuario.setEstatus(1);  
  
    Perfil objPerfil1 = new Perfil();  
    objPerfil1.setId(2);  
  
    Perfil objPerfil2 = new Perfil();  
    objPerfil2.setId(3);  
  
    objUsuario.agregarPerfil(objPerfil1);  
    objUsuario.agregarPerfil(objPerfil2);  
  
    this.servicioBDUsuarios.save(objUsuario);  
}
```



```
Hibernate: insert into Usuarios (email, `estatus`, fechaRegistro, nombre, password, username) values (?, ?, ?, ?, ?, ?)  
Hibernate: insert into UsuarioPerfil (idUserio, idPerfil) values (?, ?)  
Hibernate: insert into UsuarioPerfil (idUserio, idPerfil) values (?, ?)
```

<https://www.baeldung.com/jpa-one-to-one>

<https://www.baeldung.com/hibernate-one-to-many>

<https://www.baeldung.com/jpa-many-to-many>

<https://www.baeldung.com/hibernate-identifiers>

**Muchas gracias**  
**Preguntas**

