Programación Aplicada
Tema: Base de Datos Java.

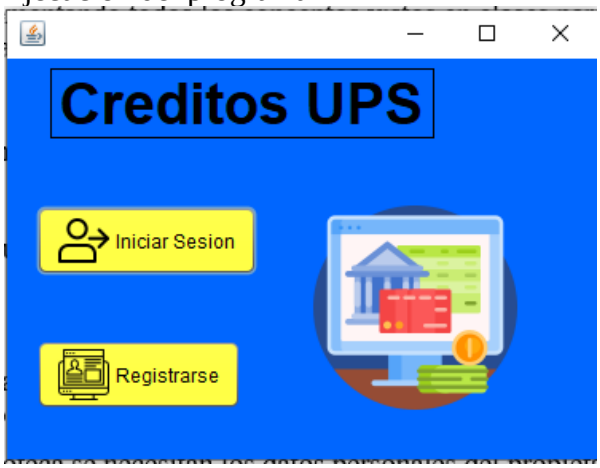
Prueba 3 29/01/2021

Objetivo:

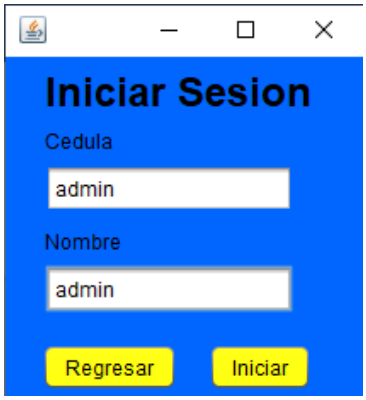
- Consolidar los conocimientos adquiridos en clase sobre JPA.

Enunciado:

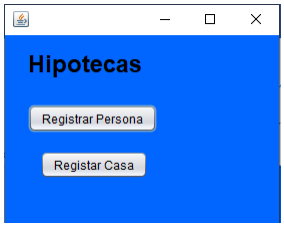
Realizar un sistema implementando todos los conceptos vistos en clases para gestionar la hipoteca de las casas con las siguientes características:
Ejecución del programa



Incio de sesion del admin



Ventana del admin

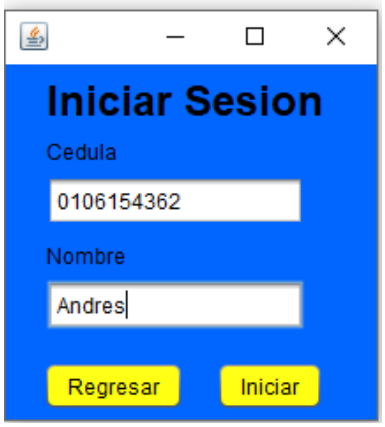


Registro de una persona y de una casa
Persona

Prueba 3 29/01/2021

Casa

Inicio de sesión de una persona

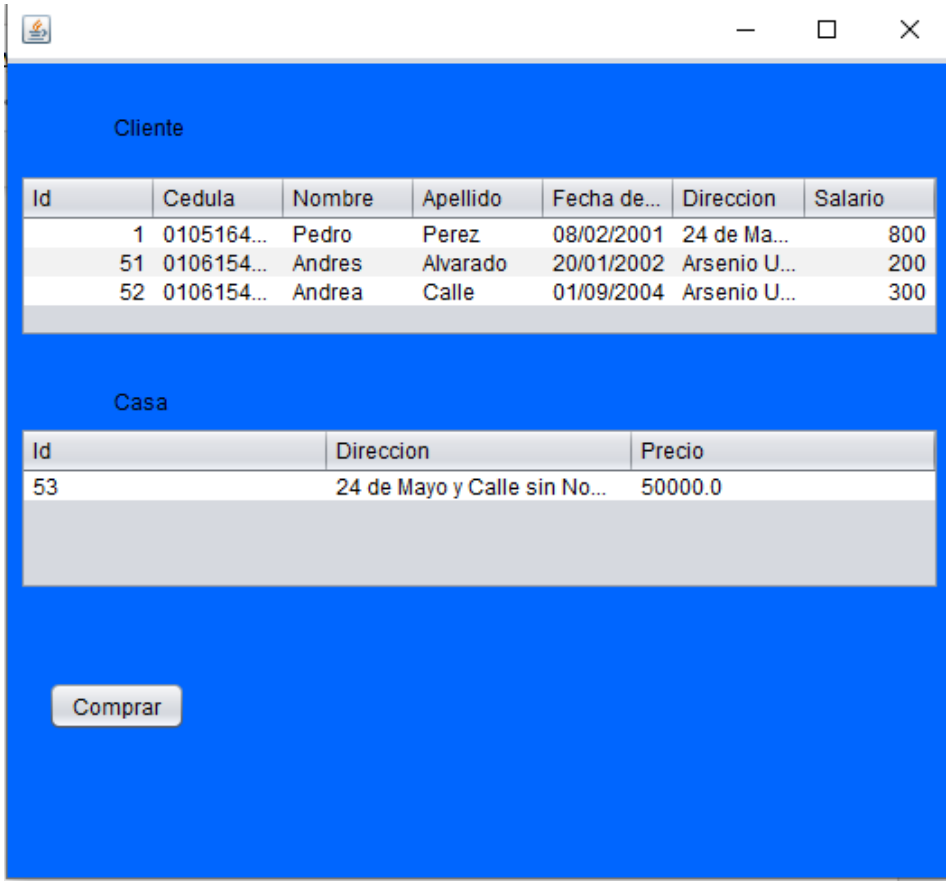


Ventana de la persona

Prueba 3 29/01/2021

- Las personas compran casas y se convierten en propietarios.

Compra de la casa



Base de Datos

Antes de la Compra

Programación Aplicada
Tema: Base de Datos Java.

Prueba 3 29/01/2021

SELECT * FROM "public".ca... ×

Max. rows: 100 Fetched Rows: 1 Match

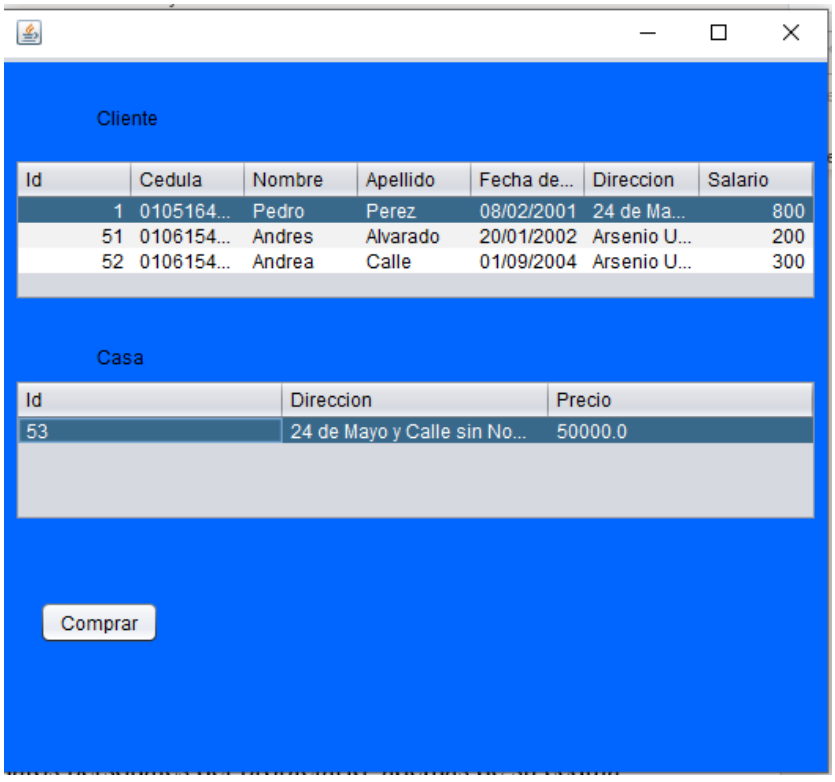
#	id	direccion	precio	fk_propietario
1	53	24 de Mayo y Calle sin Nombre	50.000	<NULL>

SELECT * FROM "public".pe... ×

Max. rows: 100 Fetched Rows: 3 Matching Rows:

#	id	apellido	cedula	direccion	fechanacimiento
1	1	Perez	0105164362	24 de Mayo y calle sin nombre	08/02/2001
2	51	Alvarado	0106154362	Arsenio Ullauri y Jose Serrano	20/01/2002
3	52	Calle	0106154370	Arsenio Ullauri y Jose Serrano	01/09/2004

Después de la Compra

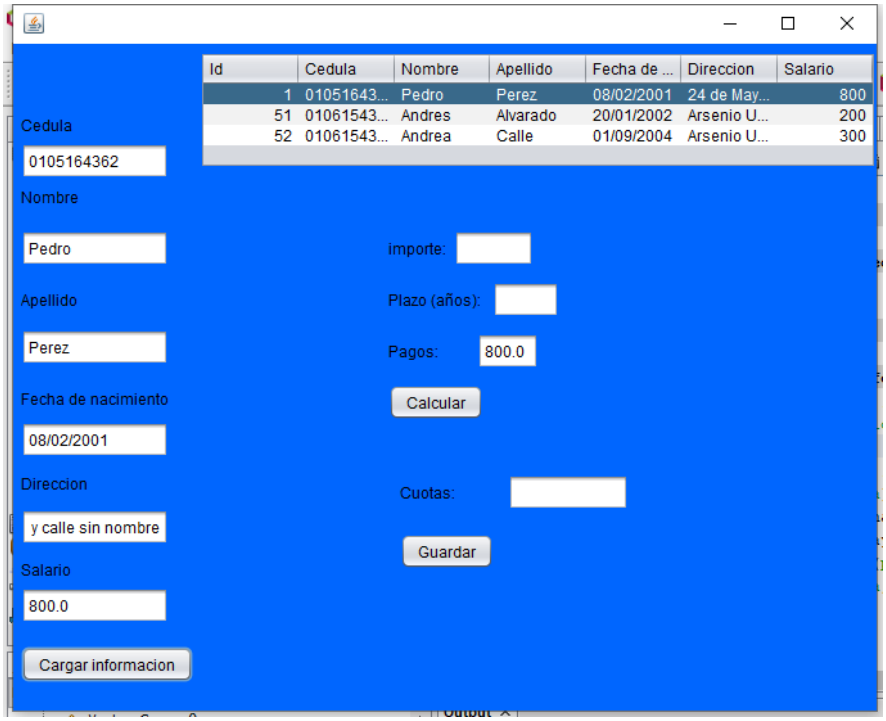


SELECT * FROM "public".ca... ×

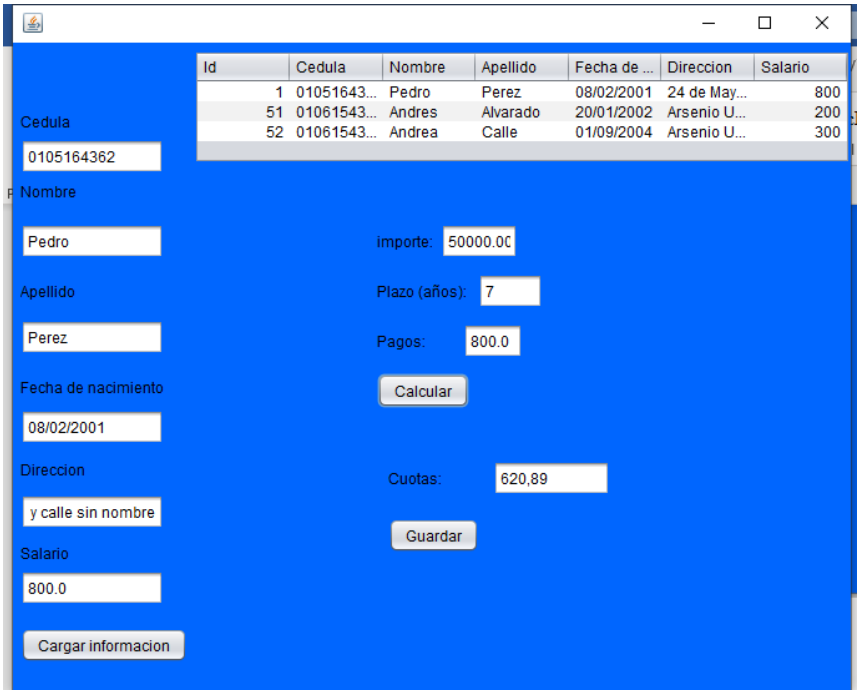
Max. rows: 100 Fetched Rows: 1 Matchi

#	id	direccion	precio	fk_propietario
1	53	24 de Mayo y Calle sin Nombre	50.000	1

- Para pagarlas es habitual que el propietario formalice un préstamo hipotecario con una entidad bancaria.

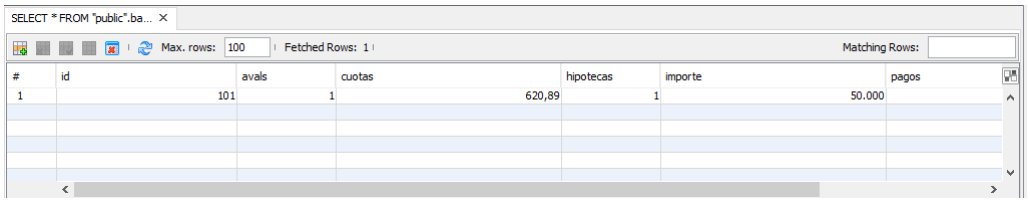


crédito

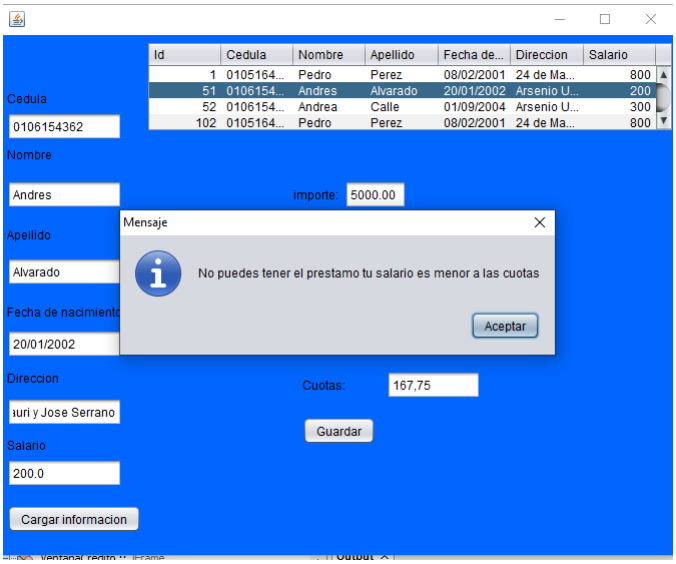


- El banco toma la casa en forma de aval en caso de impago de las mensualidades.

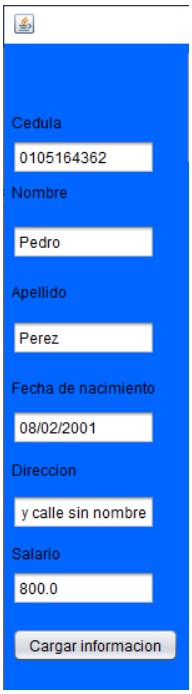
Prueba 3 29/01/2021



- En el caso de que el capital fiado supera el valor de tasación de la casa y el sueldo del propietario no es suficiente, el banco suele exigir la presencia de un avalista (garante).

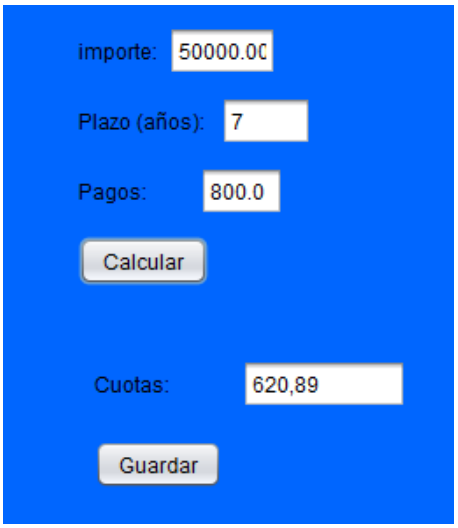


- Para formalizar la hipoteca se necesitan los datos personales del propietario, además de su cédula , dirección de la casa, su dirección, nombres, apellidos y fecha de nacimiento y del garante de ser necesario.



Prueba 3 29/01/2021

- El capital de la hipoteca se ajusta teniendo en cuenta el valor de tasación de la casa y los datos de dirección.



- JPA
  El abstract controlador para usar el JPA

```
public abstract class AbstractControlador <E> {
private List<E> lista;
private Class<E> clase;
private EntityManager em;

 public AbstractControlador() {
       lista = new ArrayList();
       Type t =getClass().getGenericSuperclass();
       ParameterizedType pt=(ParameterizedType)t;
       clase = (Class) pt.getActualTypeArguments()[0];
       em = JPAUtils.getEntityManager();
   }
   public AbstractControlador(EntityManager em) {
       lista = new ArrayList();
       Type t =getClass().getGenericSuperclass();
       ParameterizedType pt=(ParameterizedType)t;
       clase = (Class) pt.getActualTypeArguments()[0];
       this.em = em;
   }
public E crear(E objeto) throws Exception{
this.validar(objeto);
em.getTransaction().begin();
em.persist(objeto);
em.getTransaction().commit();
lista.add(objeto);
return objeto;
}
```

```
public boolean eliminar(E objeto){
em.getTransaction().begin();
em.remove(em.merge(objeto));
em.getTransaction().commit();
lista.remove(objeto);
return true;
}
public E actualizar(E objeto)throws Exception{
   this.validar(objeto);
em.getTransaction().begin();
objeto=em.merge(objeto);
em.getTransaction().commit();
lista= this.buscarTodo();
return objeto;
}
public E buscar(Object id){
return(E) em.find(clase, id);
}
public List<E> buscarTodo(){
return em.createQuery("Select t from "+clase.getSimpleName()+" t ").getResultList();
}


public abstract boolean validar(E objeto)throws Exception;

   public List<E> getLista() {
      return lista;
   }

   public void setLista(List<E> lista) {
      this.lista = lista;
   }

   public Class<E> getClase() {
      return clase;
   }

   public void setClase(Class<E> clase) {
      this.clase = clase;
   }

   public EntityManager getEm() {
      return em;
   }

   public void setEm(EntityManager em) {
      this.em = em;
   }
```

```
}
Un ejemplo de los entity class aplicados
@Entity
@NamedQuery(name ="consultaCedula",query = "Select p from Persona p where p.cedula= :cedula")
@NamedQuery(name ="consultaNombre",query = "Select p from Persona p where
p.nombre= :nombre")
public class Persona implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
//partes de la tabla
@Column
@NotNull
private String cedula;
@Column
private String nombre;
@Column
private String apellido;
@Column
private String fechaNacimiento ;
@Column
private String direccion;
@Column
private Double salario;
//relacion
@OneToMany(mappedBy="persona", cascade=CascadeType.ALL)

private List<Casa>lsitaCasa;

@ManyToOne
@JoinColumn(name ="fk_banco")
private Banco banco;
//getters y setters

    public Persona() {
    }

    public Banco getBanco() {
        return banco;
    }

    public void setBanco(Banco banco) {
        this.banco = banco;
    }
```

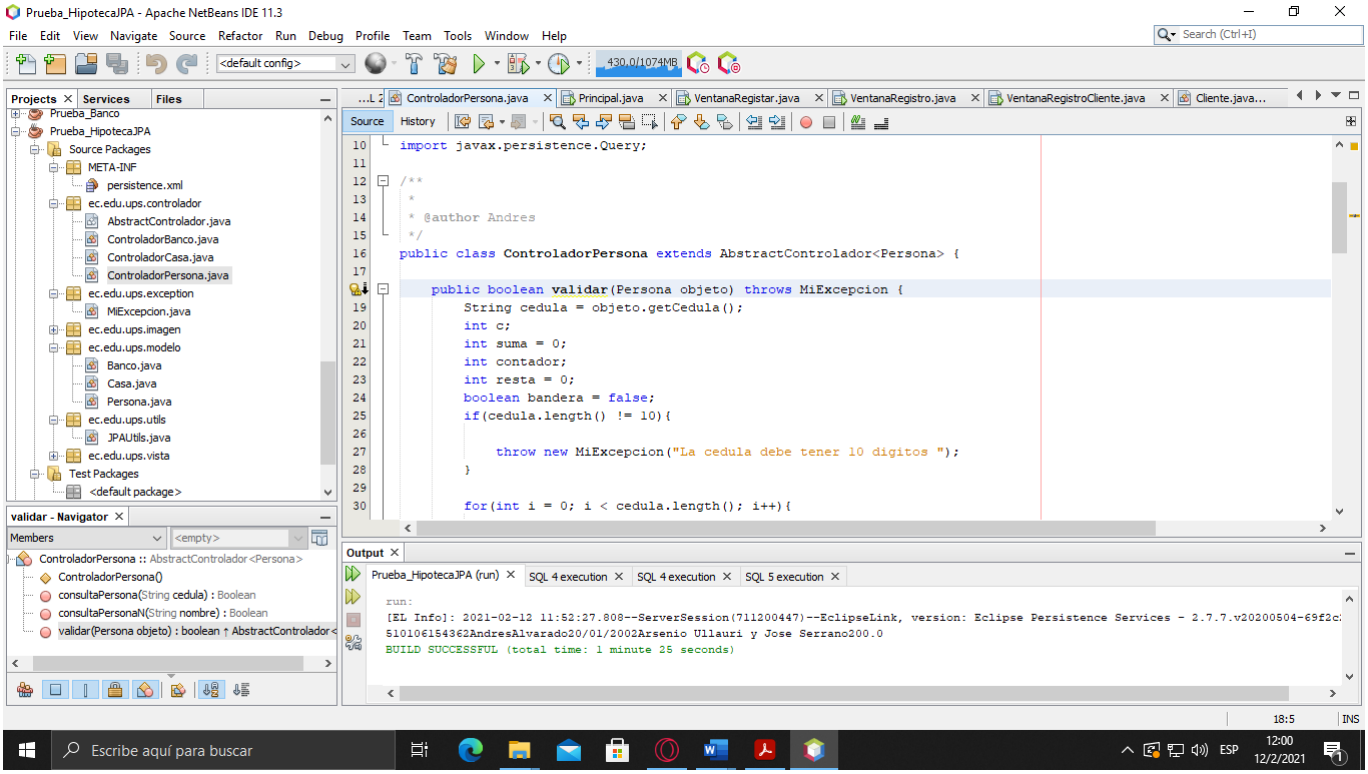
```
    public Persona(Long id, String cedula, String nombre, String apellido, String fechaNacimiento,
String direccion, Double salario, Banco banco) {
        this.id = id;
        this.cedula = cedula;
        this.nombre = nombre;
        this.apellido = apellido;
        this.fechaNacimiento = fechaNacimiento;
        this.direccion = direccion;
        this.salario = salario;
        this.banco = banco;
    }

    public Persona(Long id, String cedula, String nombre, String apellido, String fechaNacimiento,
String direccion, Double salario) {
        this.id = id;
        this.cedula = cedula;
        this.nombre = nombre;
        this.apellido = apellido;
        this.fechaNacimiento = fechaNacimiento;
        this.direccion = direccion;
        this.salario = salario;
    }


    public List<Casa> getLsitaCasa() {
        return lsitaCasa;
    }

    public void setLsitaCasa(List<Casa> lsitaCasa) {
        this.lsitaCasa = lsitaCasa;
    }

    public String getCedula() {
        return cedula;
    }

    public void setCedula(String cedula) {
        this.cedula = cedula;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
```

```
public String getFechaNacimiento() {
    return fechaNacimiento;
}

public void setFechaNacimiento(String fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
}


public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public Double getSalario() {
    return salario;
}

public void setSalario(Double salario) {
    this.salario = salario;
}




public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

@Override
public int hashCode() {
    int hash = 0;
```

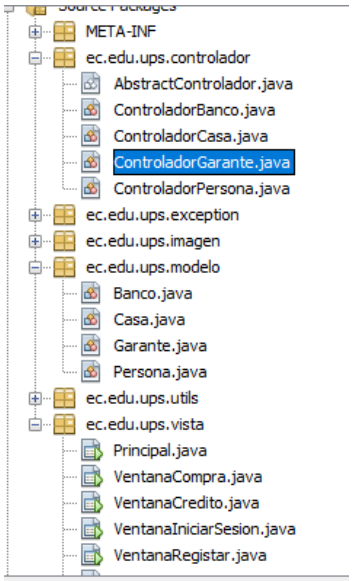
```
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Persona)) {
            return false;
        }
        Persona other = (Persona) object;
        if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Persona{" + "id=" + id + ", cedula=" + cedula + ", nombre=" + nombre + ",
fechaNacimiento=" + fechaNacimiento + ", apellido=" + apellido + ", direccion=" + direccion + ",
salario=" + salario + '}';
    }



}
```

- Excepciones: 10%
  Las excepciones con el método validar donde se aplico

Prueba 3 29/01/2021



- MVC: 20%



Diagrama

Prueba 3 29/01/2021