
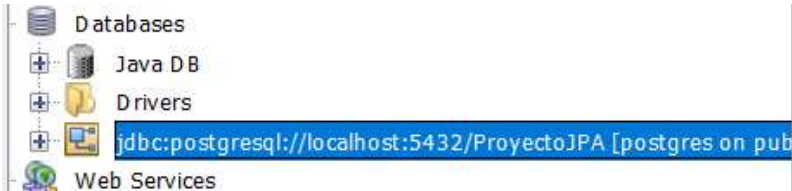

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		PRÁCTICA DE LABORATORIO	
CARRERA: Computación		ASIGNATURA: Programación Orientada a objetos	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Practica JPA	
OBJETIVO ALCANZADO: <ul style="list-style-type: none"> Consolidar los conocimientos adquiridos en clase sobre JPA 			
ACTIVIDADES DESARROLLADAS			
1. Describir al menos 10 objetos relacionados al sistema de Registro de calificaciones. Realizar un sistema implementando todos los conceptos vistos en clases para gestionar la hipoteca de las casas con las siguientes características: <ul style="list-style-type: none"> Las personas compran casas y se convierten en propietarios. Para pagarlas es habitual que el propietario formalice un préstamo hipotecario con una entidad bancaria. El banco toma la casa en forma de aval en caso de impago de las mensualidades. 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. El capital de la hipoteca se ajusta teniendo en cuenta el valor de tasación de la casa y los datos de dirección. Toda hipoteca se formaliza detallando el capital, el interés (8,99 - 16,99%) y la duración (fecha de inicio y fecha de fin). A partir de estos datos se calcula el importe de cada mensualidad para el total del tiempo que pide el préstamo. No es necesario guardar los datos del banco pero si un sistema de autenticación. Generar los datos con el sistema de amortización Alemán			
1. Link del diagrama de clases https://lucid.app/lucidchart/invitations/accept/7910c6d0-4e3c-4fef-a2fa-58751e67fe44			
2. Conexión a la base de datos 			

Prueba03-JPA

- Source Packages
 - META-INF
 - ec.edu.ups.controlador
 - AbstractControlador.java
 - ControladorCasa.java
 - ControladorPersona.java
 - ControladroUsuario.java
 - ec.edu.ups.imagenes
 - ec.edu.ups.modelo
 - Casa.java
 - Persona.java
 - Usuario.java
 - ec.edu.ups.utils
 - UtilsJPA.java
 - ec.edu.ups.vista
 - PantallaCuotas.java
 - PantallaPrincipal.java**
 - PantallaRegistrarCasa.java
 - PantallaRegistrarCliente.java
 - PantallaSolicitud.java
- Test Packages
- Generated Sources (ap-source-output)
- Libraries
 - eclipselink.jar
 - javax.annotation-api-1.3.2.jar
 - EclipseLink (JPA 2.1) - org.eclipse.persistence.core-2.5
 - EclipseLink (JPA 2.1) - org.eclipse.persistence.asm-2.5
 - EclipseLink (JPA 2.1) - org.eclipse.persistence.antlr-2.5
 - EclipseLink (JPA 2.1) - org.eclipse.persistence.jpa-2.5
 - EclipseLink (JPA 2.1) - org.eclipse.persistence.jpa.jpql-2.5
 - EclipseLink (JPA 2.1) - javax.persistence-2.1.0.jar
 - PostgresJDBC Driver - postgresql-42.2.18.jar
 - JCalendar.jar
 - JDK 11 (Default)
- Test Libraries

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Clase generica con reflexion para crear las entidades de forma generica y automatica:

```
package ec.edu.ups.controlador;

import ec.edu.ups.utils.UtilsJPA;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.List;
import javax.persistence.EntityManager;

/**
 *
 * @author Estudiantes
 */
public abstract class AbstractControlador<T> {

    private List<T> lista;
    private Class<T> clase;
    private EntityManager em;


    public AbstractControlador(EntityManager em) {
        lista = new ArrayList<>();
        Type t = getClass().getGenericSuperclass();
        ParameterizedType pt = (ParameterizedType) t;
        clase = (Class) pt.getActualTypeArguments()[0];
        this.em = em;
    }

    public AbstractControlador() {
        lista = new ArrayList<>();
        Type t = getClass().getGenericSuperclass();
        ParameterizedType pt = (ParameterizedType) t;
        clase = (Class) pt.getActualTypeArguments()[0];
        this.em = UtilsJPA.getEntityManager();
    }

    public boolean create(T objeto) {
        em.getTransaction().begin();
        em.persist(objeto);
        em.getTransaction().commit();
        lista.add(objeto);
        return true;
    }

    public T read(Object id) {
        return (T) em.find(clase, id);
    }

    public boolean update(T objeto) {
        em.getTransaction().begin();
        objeto = em.merge(objeto);
        em.getTransaction().commit();
    }
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

        lista = findAll();
        return true;
    }

    public boolean delite(T objeto) {
        em.getTransaction().begin();
        em.remove(em.merge(objeto));
        em.getTransaction().commit();
        lista.remove(objeto);
        return true;
    }

    public List<T> findAll() {
        return em.createQuery("Select t from "+ clase.getSimpleName()+ "
t").getResultList();
    }

    public List<T> getList() {
        return lista;
    }

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

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

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

    public EntityManager getEm() {
        return em;
    }

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

```


Controladores con el patron de diseño singleton y se utiliza un select para buscar en la base de datos.

```

package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Casa;
import static ec.edu.ups.utils.UtilsJPA.getEntityManager;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

/**
 *
 * @author Estudiantes
 */
public class ControladorCasa extends AbstractControlador<Casa>{

    private static ControladorCasa instance = new ControladorCasa();

    public static ControladorCasa getInstance() {
        return instance;
    }

    public Casa buscarDireccion(String direccion){

        EntityManager em = getEntityManager();

        try {
            String jpql = "Select c from Casa c where c.direccion='"+direccion+"'";
            Casa casa = (Casa) em.createQuery(jpql).getSingleResult();
            return casa;
        } catch (NoResultException e) {
            System.out.println("Error Buscar cedula");
        }
        return null;
    }
}

```

Controladores con el patron de diseño singleton y se utiliza un select para buscar en la base de datos.

```

package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Persona;
import static ec.edu.ups.utils.UtilsJPA.getEntityManager;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;

/**
 *
 * @author Estudiantes
 */
public class ControladorPersona extends AbstractControlador<Persona>{


    private static ControladorPersona instance = new ControladorPersona();

    public static ControladorPersona getInstance() {
        return instance;
    }

    public Persona buscarCedula(String cedula){

        EntityManager em = getEntityManager();
    }
}

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

    try {
        String jpql = "Select p from Persona p where p.cedula='"+cedula+"'";
        Persona persona = (Persona) em.createQuery(jpql).getSingleResult();
        return persona;
    } catch (NoResultException e) {
        System.out.println("Error Buscar cedula");
    }
    return null;
}
}

```

Controladores con el patron de diseño singleton y se utiliza un select para buscar en la base de datos.

```

package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Usuario;
import static ec.edu.ups.utils.UtilsJPA.getEntityManager;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
import javax.swing.JOptionPane;

/**
 *
 * @author Estudiantes
 */
public class ControladroUsuario extends AbstractControlador<Usuario>{

    private static ControladroUsuario instance = new ControladroUsuario();


    public static ControladroUsuario getInstance() {
        return instance;
    }

    public Object iniciarSesion(String correo, String contraseña){

        EntityManager em = getEntityManager();

        try {
            String jpql = "Select u from Usuario u where u.correo='"+correo+"'";
            Usuario usuario = (Usuario) em.createQuery(jpql).getSingleResult();
            if(usuario.getContraseña().equals(contraseña)){
                return usuario;
            }
            JOptionPane.showMessageDialog(null, "Contraseña Incorrecta");
            return null;
        } catch (NoResultException e) {
            System.out.println("Error Iniciar Sesion");
        }
    }
}

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

    }
    JOptionPane.showMessageDialog(null, "Usuario no registrado");
    return null;
}
}

```

Modelos de el programa:

Casa:

```

package ec.edu.ups.modelo;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

/**
 *
 * @author Estudiantes
 */
@Entity
public class Casa implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column
    private String ubicacion;

    @Column
    private String direccion;


    @Column
    private double valor;

    public Long getId() {
        return id;
    }

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

    public String getUbicacion() {
        return ubicacion;
    }
}

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

public void setUbicacion(String ubicacion) {
    this.ubicacion = ubicacion;
}

public String getDireccion() {
    return direccion;
}

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

public double getValor() {
    return valor;
}

public void setValor(double valor) {
    this.valor = valor;
}

@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 Casa)) {
        return false;
    }
    Casa other = (Casa) 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 "Casa{" + "id=" + id + ", ubicacion=" + ubicacion + ", direccion=" +
direccion + ", valor=" + valor + '}';
}
}

```

Persona:

```
package ec.edu.ups.modelo;
```



```
import java.io.Serializable;
import java.util.Date;
import javax.persistence.Column;

/**
 *
 * @author Estudiantes
 */
@Entity
public class Persona implements Serializable {

    private static final long serialVersionUID = 1L;
    @javax.persistence.Id
    @javax.persistence.GeneratedValue(strategy =
javax.persistence.GenerationType.AUTO)
    private Long id;

    @Column
    private String cedula;

    @Column
    private String nombres;

    @Column
    private String apellidos;

    @Column
    private Date fechaNacimiento;

    @Column
    private double sueldo;

    public Long getId() {
        return id;
    }


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

    public String getCedula() {
        return cedula;
    }

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

    public String getNombres() {
        return nombres;
    }

    public void setNombres(String nombres) {
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

        this.nombres = nombres;
    }

    public String getApellidos() {
        return apellidos;
    }

    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }

    public Date getFechaNacimiento() {
        return fechaNacimiento;
    }

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

    public double getSueldo() {
        return sueldo;
    }


    public void setSueldo(double sueldo) {
        this.sueldo = sueldo;
    }

    @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() {

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

        return "Persona{" + "id=" + id + ", cedula=" + cedula + ", nombres=" +
nombres + ", apellidos=" + apellidos + ", fechaNacimiento=" + fechaNacimiento + ",
sueldo=" + sueldo + '}';
    }

}

```

Clase Usuario:

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.edu.ups.modelo;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

/**
 *
 * @author Estudiantes
 */
@Entity
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column
    private String correo;


    @Column
    private String contraseña;

    public Long getId() {
        return id;
    }

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

    public String getCorreo() {
        return correo;
    }
}

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

public void setCorreo(String correo) {
    this.correo = correo;
}

public String getContraseña() {
    return contraseña;
}

public void setContraseña(String contraseña) {
    this.contraseña = contraseña;
}

@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 Usuario)) {
        return false;
    }
    Usuario other = (Usuario) 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 "Usuario{" + "id=" + id + ", correo=" + correo + ", contrase\u00f1a="
+ contraseña + '}';
}
}

```

Metodo para calcular las cuotas:


```

public void mostrar(){
    double cuota = prestamo/plazo;
    JLabel label;
    double[][] matriz = new double[plazo][5];
    matriz[0][0] = 0;
    matriz[0][1] = 0;
    matriz[0][2] = 0;
    matriz[0][3] = 0;
}

```

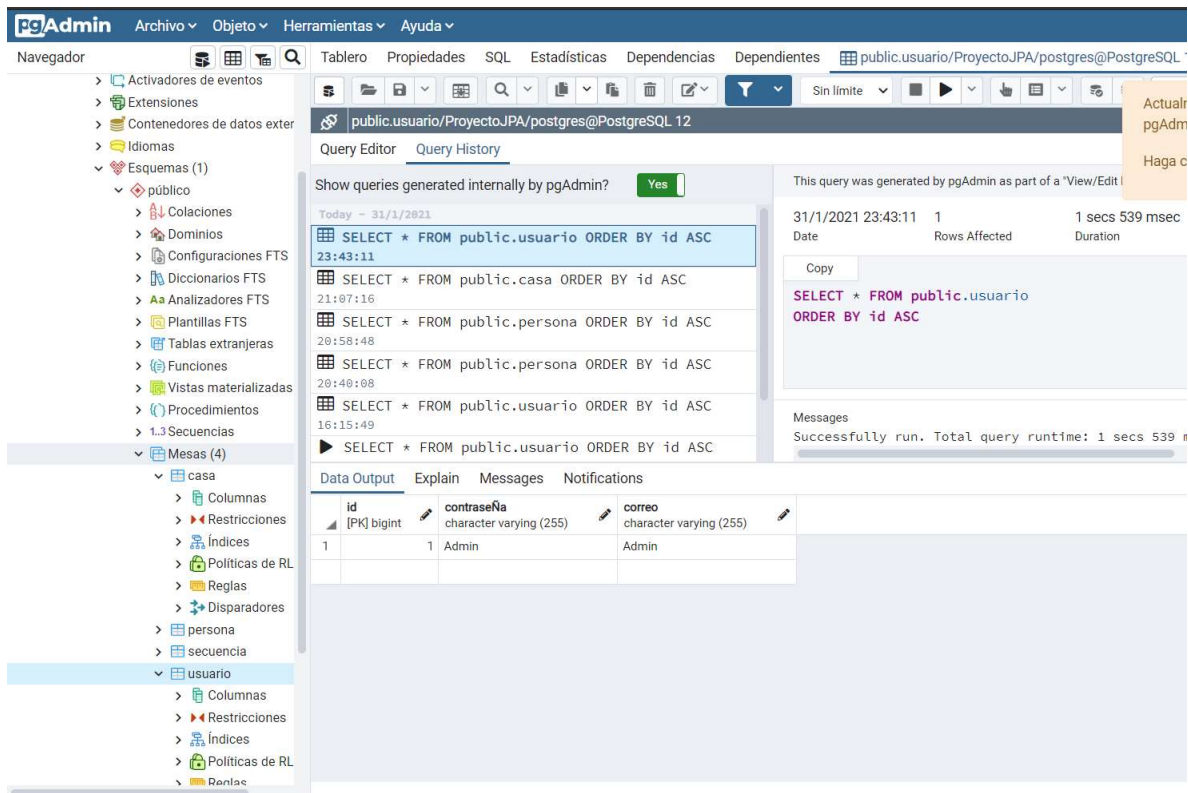
```
matriz[0][4] = prestamo;
for (int i = 1; i < plazo+1; i++) {
    for (int j = 0; j < 5; j++) {
        if(j==0){
            matriz[i][j] = i;
        }
        if(j==1){
            matriz[i][j]=cuota;
        }
        if(j==2){
            matriz[i][j] = matriz[i-1][4]*(interes/100);
        }
        if(j==3){
            matriz[i][j]= matriz[i][2]+cuota;
        }
        if(j==4){
            matriz[i][j]= matriz[i-1][4] - cuota;
        }
    }
}
String[] c = {"Mes","Cuota","Interes","Amortizativo","Capital"};
for (int i = 0; i < 5; i++) {
    label = new JLabel();
    label.setSize(110, 30);
    label.setText(c[i]);
    jPanel1.add(label);
}
jPanel1.updateUI();
for (int i = 0; i < plazo; i++) {
    for (int j = 0; j < 5; j++) {
        label = new JLabel();
        label.setSize(110, 30);
        label.setText(matriz[i][j]+"");
        jPanel1.add(label);
    }
}
jPanel1.updateUI();
}
```

Probamos la aplicación:

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



La pantalla principal la cual se tiene guardado un usuario por defecto en la base de datos:





Registro

Cedula: 0107030196

Nombres: Cristina

Apellidos: Uyaguari

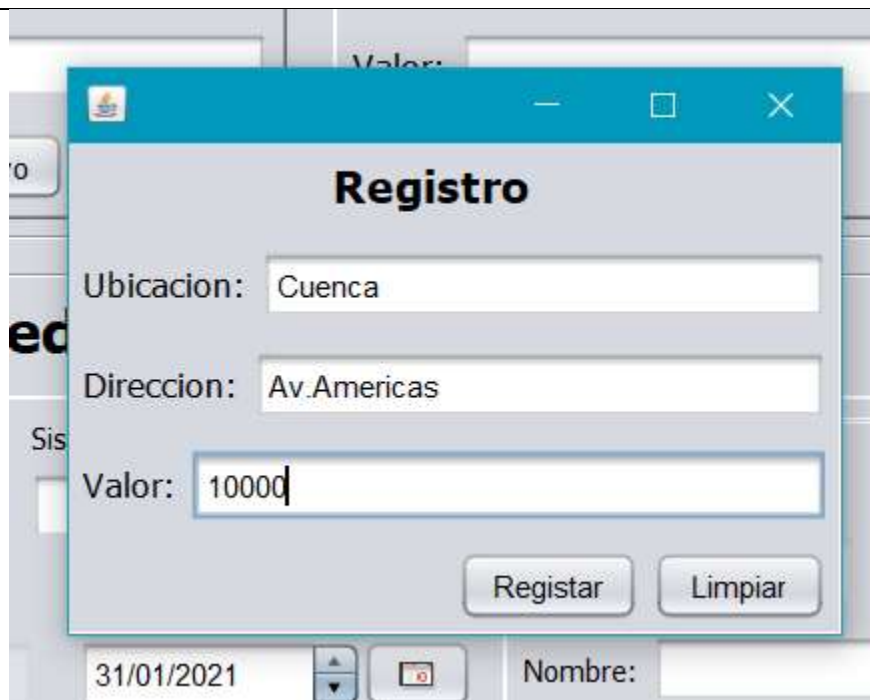
Fecha Nacimiento: 20/12/2000

Sueldo: 1500.98

Registrar Limpiar

Fecha Final

Apellido



Registro


Ubicacion: Cuenca

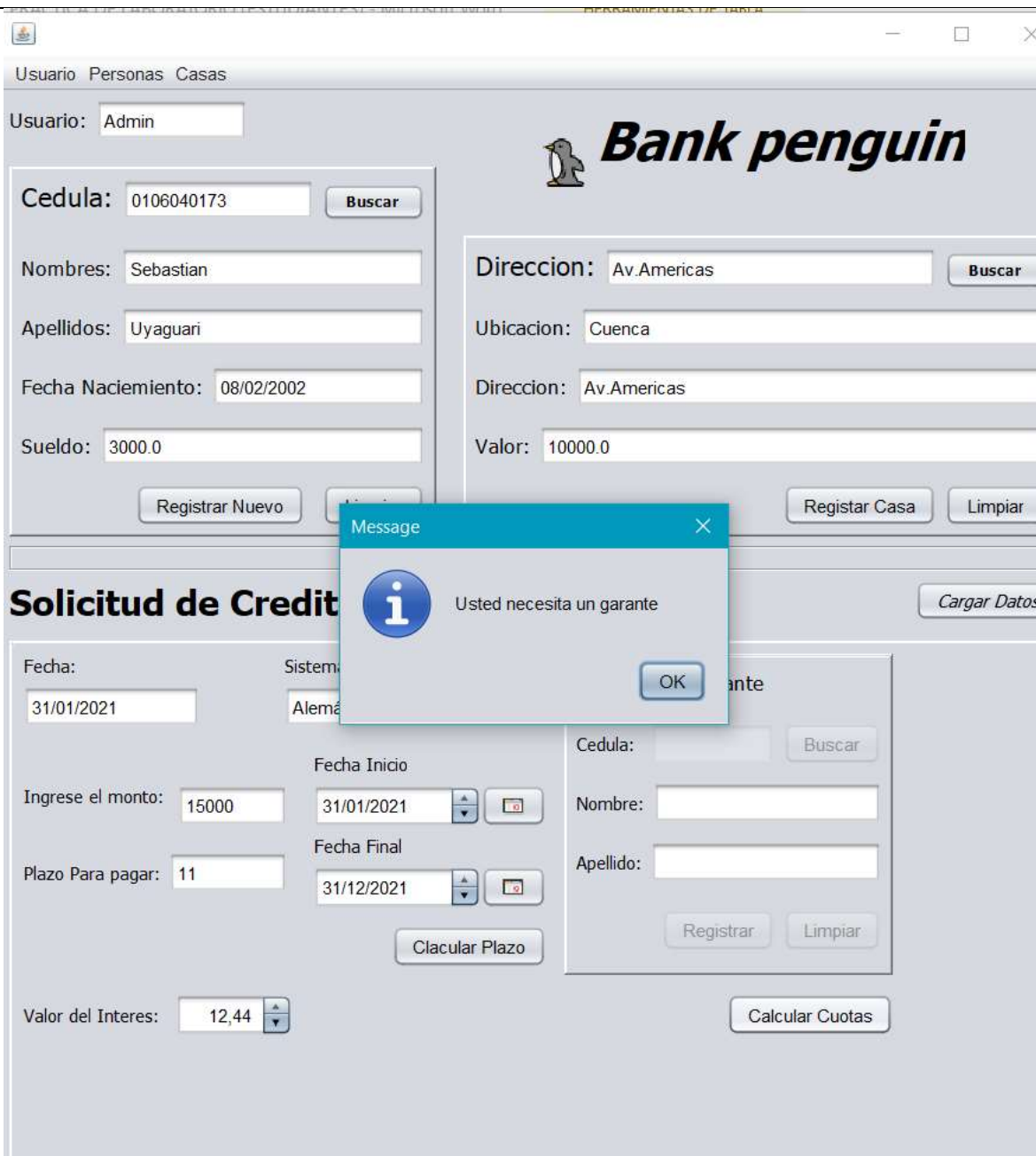
Direccion: Av.Americas

Valor: 10000

Registrar Limpiar

31/01/2021 Nombre:

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



The screenshot shows a web application interface for 'Bank penguin'. At the top, there is a navigation bar with 'Usuario', 'Personas', and 'Casas'. Below this, the 'Usuario' section shows 'Admin' as the logged-in user. The main area is divided into two columns. The left column contains a form for user registration with fields for 'Cedula' (0106040173), 'Nombres' (Sebastian), 'Apellidos' (Uyaguari), 'Fecha Nacimiento' (08/02/2002), and 'Sueldo' (3000.0). The right column contains a form for address registration with fields for 'Direccion' (Av. Americas), 'Ubicacion' (Cuenca), and 'Valor' (10000.0). A 'Message' dialog box is displayed in the center, with the text 'Usted necesita un garante' (You need a guarantor) and an 'OK' button. Below the dialog, the 'Solicitud de Credit' (Credit Application) section is visible, with fields for 'Fecha' (31/01/2021), 'Ingreso el monto' (15000), 'Plazo Para pagar' (11), 'Fecha Inicio' (31/01/2021), 'Fecha Final' (31/12/2021), 'Valor del Interes' (12,44), and 'Cedula'. There are buttons for 'Registrar Nuevo', 'Registrar Casa', 'Limpiar', 'Cargar Datos', 'Clacular Plazo', and 'Calcular Cuotas'.

Usuario: Admin

Bank penguin

Cedula: 0106040173

Nombres: Sebastian

Apellidos: Uyaguari

Fecha Nacimiento: 08/02/2002

Direccion: Av. Americas

Ubicacion: Cuenca

Direccion: Av. Americas

Cuotas

Mes	Cuota	Interes	Amortizativo	Capital
0.0	0.0	0.0	0.0	15000.0
1.0	1363.63636363637	1648.4999656677246	3012.1363293040886	13636.363636363636
2.0	1363.63636363637	1498.636332425204	2862.272696061568	12272.727272727272
3.0	1363.63636363637	1348.7726991826837	2712.4090628190474	10909.090909090908
4.0	1363.63636363637	1198.9090659401631	2562.5454295765267	9545.454545454544
5.0	1363.63636363637	1049.0454326976428	2412.6817963340063	8181.818181818181
6.0	1363.63636363637	899.1817994551224	2262.818163091486	6818.181818181816
7.0	1363.63636363637	749.3181662126019	2112.9545298489656	5454.545454545452
8.0	1363.63636363637	599.4545329700815	1963.0908966064453	4090.9090909090883
9.0	1363.63636363637	449.59089972756095	1813.2272633639247	2727.2727272727243
10.0	1363.63636363637	299.7272664850405	1663.3636301214042	1363.6363636363606


Plazo Para pagar: 11 31/12/2021

Valor del Interes: 10,99

Prueba03-JPA (run)

RESULTADO(S) OBTENIDO(S):

- Un programa funcional con ingreso de datos y con la facilidad de agregar datos a la Base creado con JPA

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

CONCLUSIONES:

- Con esta práctica sirve como introducción para la programación en base de datos la cual nos servirá para que en el futuro cercano pongamos en práctica los conocimientos adquiridos.

RECOMENDACIONES:

- Aplicar los conocimientos adquiridos con mayor frecuencia.

Nombre de estudiante: Sebastián Roberto Uyaguari Ramón

Firma de estudiante:

