

	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: COMPUTACION		ASIGNATURA: PROGRAMACION APLICADA
NRO. PRÁCTICA:		TÍTULO PRÁCTICA: JPA
OBJETIVO ALCANZADO: <u>Enunciado:</u> 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]. Bibliografía [1] https://www.produbanco.com.ec/banca-minorista/cr%C3%A9ditos/hipotecario/simulador-de-cr%C3%A9dito-hipotecario/		
RESULTADO(S) OBTENIDO(S): a) INTERFAZ GRAFICA Se creo una interfaz grafica donde tenemos la opción de inicio de sesión, un apartado para los datos del simulador donde podemos observar que los datos del cliente y del garante se visualizan en la parte derecha. En caso de no existir un cliente, tendremos una opción para agregarlo a la base de datos.		

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

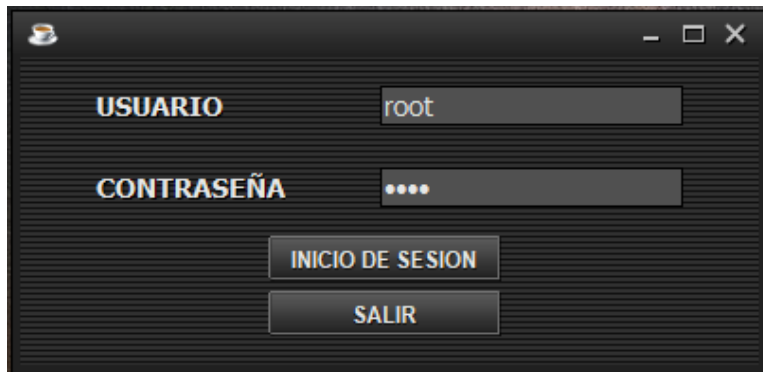



Figura 1 Interfaz Inicio de Sesión



Figura 2 Interfaz Datos Simulación

En esta interfaz tenemos la posibilidad de agregar los datos del cliente, además del capital he interés. Los datos del cliente serán reflejados al dar enter, esto hace que en la parte derecha se carguen sus datos al igual que en caso de necesitar garante.

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

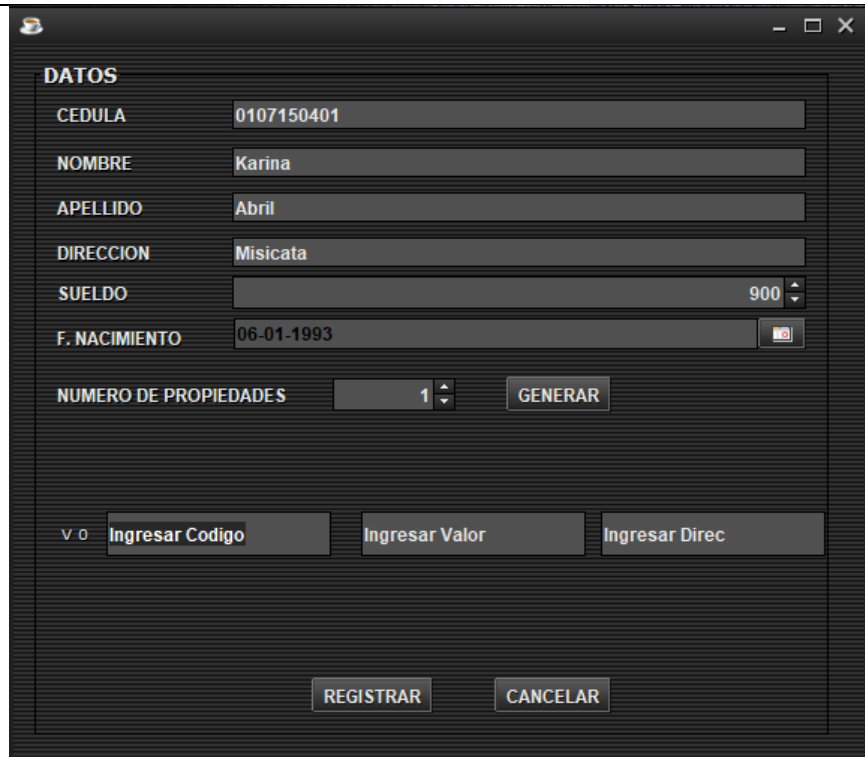


Figura 3 Interfaz Agregar Cliente

En este apartado tenemos la posibilidad de agregar un nuevo cliente, con todos sus datos y propiedades.

TABLA DE PRESTAMO
VALOR TOTAL: \$10000.0
INTERES: %9.5
CUOTAS: 24



N. DE CUOTA	CAPITAL AL INICIO DEL PERIODO	AMORTIZACION	INTERESES DEL PERIODO	CUOTA
1	10000.0	416.67	79.17	495.84
2	9583.33	416.67	75.87	492.54
3	9166.66	416.67	72.57	489.24
4	8749.99	416.67	69.27	485.94
5	8333.32	416.67	65.97	482.64
6	7916.65	416.67	62.67	479.34
7	7499.98	416.67	59.37	476.04
8	7083.31	416.67	56.08	472.75
9	6666.64	416.67	52.78	469.45
10	6249.97	416.67	49.48	466.15
11	5833.3	416.67	46.18	462.85
12	5416.63	416.67	42.88	459.55
13	4999.96	416.67	39.58	456.25
14	4583.29	416.67	36.28	452.95
15	4166.62	416.67	32.99	449.66
16	3749.95	416.67	29.69	446.36
17	3333.28	416.67	26.39	443.06
18	2916.61	416.67	23.09	439.76
19	2499.94	416.67	19.79	436.46
20	2083.27	416.67	16.49	433.16
21	1666.6	416.67	13.19	429.86
22	1249.93	416.67	9.9	426.57
23	833.26	416.67	6.6	423.27
24	416.59	416.67	3.3	419.97

Figura 4 PDF TABLA

Al poner generar, se nos genera un PDF con todos los datos del préstamo según el sistema Alemán.

b) ESQUEMA BASE DE DATOS

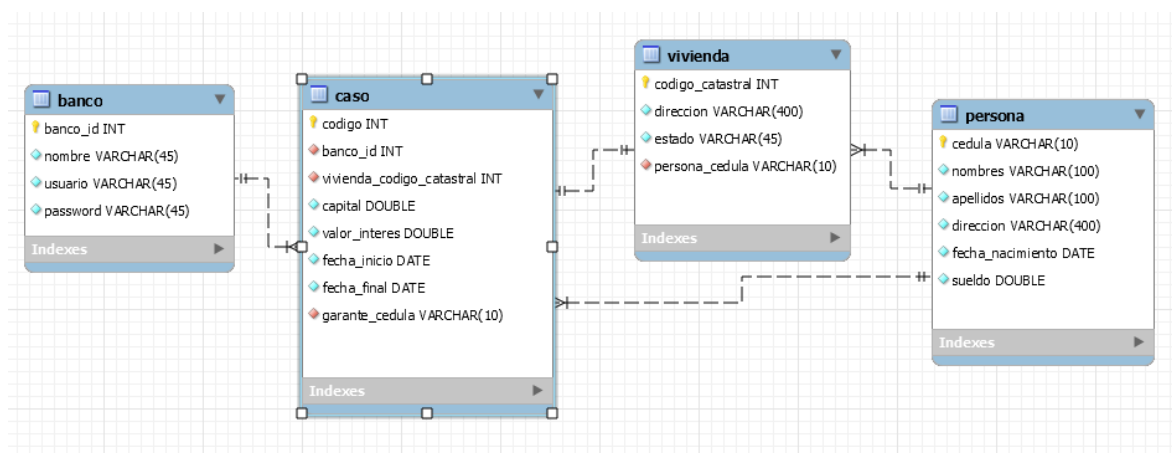



Figura 5 Esquema Base de Datos

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

Para la base de datos se creó tres tablas.

- BANCO
- CASO
- PERSONA
- VIVIENDA

Banco. _ Almacena los datos del banco, como es usuario, contraseña.

Caso. _ Almacena los registros del préstamo, donde se enlaza con la Primary Key de Vivienda para obtener el dueño y con la Primary Key de Persona para el garante.

Persona. _ Dispone todos los datos de los clientes y garantes.

Vivienda. _ Tiene todos los datos referentes al terreno y tambien la Primary key del propietario.

Modelos.

Al disponer de una herramienta con el JPA, podemos generar los modelos desde la base de Datos o Generar la base de Datos desde los modelos, estos disponen de los mismos atributos que las tablas de referencia.

Banco

```

@Entity
@Table(name = "banco")
@NamedQueries({
    @NamedQuery(name = "Banco.findAll", query = "SELECT b FROM Banco b"),
    @NamedQuery(name = "Banco.findById", query = "SELECT b FROM Banco b WHERE b.bancoId = :bancoId"),
    @NamedQuery(name = "Banco.findByName", query = "SELECT b FROM Banco b WHERE b.nombre = :nombre"),
    @NamedQuery(name = "Banco.findByUsuario", query = "SELECT b FROM Banco b WHERE b.usuario = :usuario"),
    @NamedQuery(name = "Banco.findByPassword", query = "SELECT b FROM Banco b WHERE b.password = :password")})
public class Banco implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "banco_id")
    private Integer bancoId;
    @Basic(optional = false)
    @Column(name = "nombre")
    private String nombre;
    @Basic(optional = false)
    @Column(name = "usuario")
    private String usuario;
    @Basic(optional = false)
    @Column(name = "password")
    private String password;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "bancoId")
    private Collection<Caso> casoCollection;

```

Caso

```
@Entity
@Table(name = "caso")
@NamedQueries({
    @NamedQuery(name = "Caso.findAll", query = "SELECT c FROM Caso c"),
    @NamedQuery(name = "Caso.findByCodigo", query = "SELECT c FROM Caso c WHERE c.codigo = :codigo"),
    @NamedQuery(name = "Caso.findByCapital", query = "SELECT c FROM Caso c WHERE c.capital = :capital"),
    @NamedQuery(name = "Caso.findByValorInteres", query = "SELECT c FROM Caso c WHERE c.valorInteres = :valorInteres"),
    @NamedQuery(name = "Caso.findByFechaInicio", query = "SELECT c FROM Caso c WHERE c.fechaInicio = :fechaInicio"),
    @NamedQuery(name = "Caso.findByFechaFinal", query = "SELECT c FROM Caso c WHERE c.fechaFinal = :fechaFinal"),
    @NamedQuery(name = "Caso.findByGaranteId", query = "SELECT c FROM Caso c WHERE c.garanteId = :garanteId"))
public class Caso implements Serializable {


    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "codigo")
    private Integer codigo;
    @Basic(optional = false)
    @Column(name = "capital")
    private double capital;
    @Basic(optional = false)
    @Column(name = "valor_interes")
    private double valorInteres;
    @Basic(optional = false)
    @Column(name = "fecha_inicio")
    @Temporal(TemporalType.DATE)
    private Date fechaInicio;
    @Basic(optional = false)
    @Column(name = "fecha_final")
    @Temporal(TemporalType.DATE)
    private Date fechaFinal;
    @Column(name = "garante_id")
    private String garanteId;
    @JoinColumn(name = "banco_id", referencedColumnName = "banco_id")
    @ManyToOne(optional = false)
    private Banco bancoId;
    @JoinColumn(name = "vivienda_codigo_catastral", referencedColumnName = "codigo_catastral")
    @ManyToOne(optional = false)
    private Vivienda viviendaCodigoCatastral;
```

Persona

```
@Entity
@Table(name = "persona")
@NamedQueries({
    @NamedQuery(name = "Persona.findAll", query = "SELECT p FROM Persona p"),
    @NamedQuery(name = "Persona.findByCedula", query = "SELECT p FROM Persona p WHERE p.cedula = :cedula"),
    @NamedQuery(name = "Persona.findByNombres", query = "SELECT p FROM Persona p WHERE p.nombres = :nombres"),
    @NamedQuery(name = "Persona.findByApellidos", query = "SELECT p FROM Persona p WHERE p.apellidos = :apellidos"),
    @NamedQuery(name = "Persona.findByDireccion", query = "SELECT p FROM Persona p WHERE p.direccion = :direccion"),
    @NamedQuery(name = "Persona.findByFechaNacimiento", query = "SELECT p FROM Persona p WHERE p.fechaNacimiento = :fechaNacimiento"),
    @NamedQuery(name = "Persona.findBySueldo", query = "SELECT p FROM Persona p WHERE p.sueldo = :sueldo"))
public class Persona implements Serializable {

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "personaCedula")
    private Collection<Vivienda> viviendaCollection;

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "cedula")
    private String cedula;
    @Basic(optional = false)
    @Column(name = "nombres")
    private String nombres;
    @Basic(optional = false)
    @Column(name = "apellidos")
    private String apellidos;
    @Basic(optional = false)
    @Column(name = "direccion")
    private String direccion;
    @Basic(optional = false)
    @Column(name = "fecha_nacimiento")
    @Temporal(TemporalType.DATE)
    private Date fechaNacimiento;
    @Basic(optional = false)
    @Column(name = "sueldo")
    private double sueldo;
```

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

Vivienda

```

@Entity
@Table(name = "vivienda")
@NamedQueries({
    @NamedQuery(name = "Vivienda.findAll", query = "SELECT v FROM Vivienda v"),
    @NamedQuery(name = "Vivienda.findByCodigoCatastral", query = "SELECT v FROM Vivienda v WHERE v.codigoCatastral = :codigoCatastral"),
    @NamedQuery(name = "Vivienda.findByDireccion", query = "SELECT v FROM Vivienda v WHERE v.direccion = :direccion"),
    @NamedQuery(name = "Vivienda.findByEstado", query = "SELECT v FROM Vivienda v WHERE v.estado = :estado"),
    @NamedQuery(name = "Vivienda.findByAvaluo", query = "SELECT v FROM Vivienda v WHERE v.avaluo = :avaluo"))
public class Vivienda implements Serializable {

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "viviendaCodigoCatastral")
    private Collection<Caso> casoCollection;

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "codigo_catastral")
    private Integer codigoCatastral;
    @Basic(optional = false)
    @Column(name = "direccion")
    private String direccion;
    @Basic(optional = false)
    @Column(name = "estado")
    private String estado;
    @Basic(optional = false)
    @Column(name = "avaluo")
    private double avaluo;
    @JoinColumn(name = "persona_cedula", referencedColumnName = "cedula")
    @ManyToOne(optional = false)
    private Persona personaCedula;

```

Se puede Visualizar mejor el código y sus librerías en el repositorio de git Hub que contiene toda la información.

<https://github.com/psidrovo/PruebaPractica3Banca.git>

CONCLUSIONES:

Esta prueba nos ayudo a entender la importancia de la utilización de frameworks, que no únicamente nos facilita las cosas, sino que nos ayuda a mejorar nuestro trabajo y tareas, gestionando de una forma mejor nuestras clases y conexiones a base de datos. Al igual que esta y otras herramientas, están para ayudarnos a estandarizar ciertas formas de programación y hacer que nuestros códigos sean mas legibles y pueda cualquiera hacer una mejora al código de una forma ágil y sencilla.

Nombre de estudiante: Paul Sebastian Idrovo Berrezueta



Firma de estudiante: _____