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

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Universidad Politécnica Salesiana

Vicerrectorado Docente

Código del Formato:	GUIA-PRL-001
Versión:	VF1.0
Elaborado por:	Directores de Área del Conocimiento Integrantes Consejo Académico
Fecha de elaboración:	2016/04/01
Revisado por:	Consejo Académico
Fecha de revisión:	2016/04/06
Aprobado por:	Lauro Fernando Pesántez Avilés Vicerrector Docente
Fecha de aprobación:	2016/14/06
Nivel de confidencialidad:	Interno

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

Descripción General

Propósito


El propósito del presente documento es definir un estándar para elaborar documentación de guías de práctica de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana, con la finalidad de lograr una homogenización en la presentación de la información por parte del personal académico y técnico docente.


Alcance


El presente estándar será aplicado a toda la documentación referente a informes de prácticas de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana.

Formatos

- Formato de Guía de Práctica de Laboratorio / Talleres / Centros de Simulación – para Docentes
- Formato de Informe de Práctica de Laboratorio / Talleres / Centros de Simulación – para Estudiantes

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

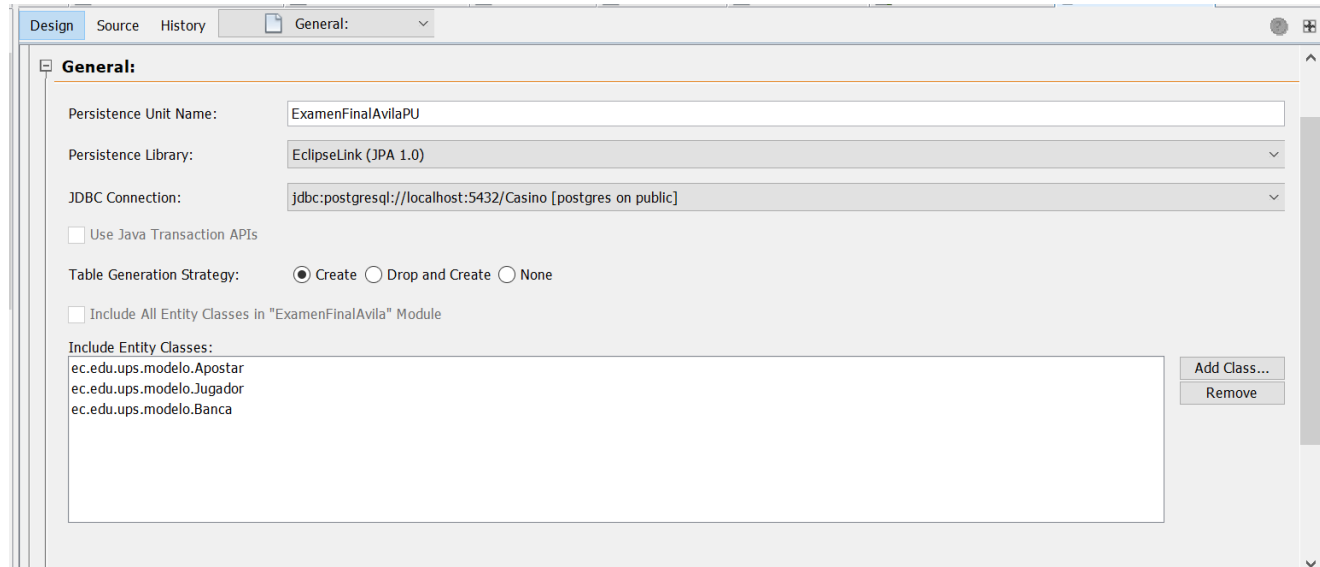
		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:		TÍTULO PRÁCTICA: Examen Final Casino	
OBJETIVO: Consolidar los conocimientos adquiridos en clase sobre Java.			
ACTIVIDADES DESARROLLADAS			
<p>1. Enunciado:</p> <p>Se desea simular los posibles beneficios de diversas estrategias de juego en un casino. La ruleta francesa es un juego en el que hay una ruleta con 37 números (del 0 al 36). Cada 2000 (tiempo parametrizable) milisegundos el croupier saca un número al azar y los diversos hilos clientes apuestan para ver si ganan. Todos los hilos empiezan con 1.000 euros y la banca (que controla la ruleta) con 50.000. Cuando los jugadores pierden dinero, la banca incrementa su saldo.</p> <ul style="list-style-type: none"> • Se puede jugar a un número concreto. Habrá 4 hilos clientes que eligen números al azar del 1 al 36 (no el 0) y restarán 10 euros de su saldo para apostar a ese ese número. Si sale su número su saldo se incrementa en 360 euros (36 veces lo apostado). • Se puede jugar a par/impar. Habrá 4 hilos clientes que eligen al azar si apuestan a que saldrá un número par o un número impar. Siempre restan 10 euros para apostar y si ganan incrementan su saldo en 20 euros. • Se puede jugar a la «martingala». Habrá 4 hilos que eligen números al azar. Elegirán un número y empezarán restando 10 euros de su saldo para apostar a ese número. Si ganan incrementan su saldo en 360 euros. Si pierden jugarán el doble de su apuesta anterior (es decir, 20, luego 40, luego 80, y así sucesivamente) • La banca acepta todas las apuestas pero nunca paga más dinero del que tiene. • Si sale el 0, todo el mundo pierde y la banca se queda con todo el dinero. <p>Adicionalmente, se deberá generar un sistema de base de datos con JPA en donde puede gestionar a los clientes o hilos jugadores, con cada una de las apuestas realizadas, los valores que se están manejando tanto de la banca como de cada cliente y gestionar la simulación es decir se puede iniciar y parar en cualquier intervalo de tiempo en la simulación, además de poder cambiar a cualquier cliente con un nuevo o un anterior y en que modalidad va a jugar. Por otro lado, es parametrizable el tiempo que se demora dar la vuelta a la ruleta con el proceso de apuesta.</p> <p>Es importante destacar que debe existir un sistema de simulación visual y un sistema de gestión de jugadores, transacciones y apuestas en donde se evidencia la apuesta, el jugador, la ruleta el numero generado y como varían los saldos de los que intervienen dentro del juego.</p> <p>Por ultimo se debe presentar dos reportes o tablas de los datos:</p> <ol style="list-style-type: none"> 1. Clientes y la banca con el numero de transacciones o apuestas realizadas, el valor de total y cuantas a perdido y cuantas veces a ganado además de la modalidad de juego. 2. Dentro de cada cliente se puede acceder al historial de apuestas y transacciones realizadas. 			

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

Se calificará el avance con los siguientes criterios de evaluación:

- JPA: 25%
- Hilos (Thread): 15 %
- Excepciones: 5%
- MVC: 10%
- Diagrama de clases: 10%
- Usabilidad – Vista - Simulación: 25%
- Programación genérica, Java 8, reflexión: 10%

1.JPA



En la imagen podemos ver que se esta utilizando JPA para el programa ya que utilizamos Entity class de las clases que vamos a necesitar una tabla, pero a lo largo del código vamos a confirmar que en verdad todo el programa esta trabajando con un JPA para el manejo de archivos en la base de datos, esto será mas notorio en el código de la clase tanto de cada uno de los entity class como en la del controlador.


2.Hilos

Para demostrar la aplicación de los Hilos en el programa vamos a ver el siguiente código de la clase Jugar ya que esta clase implementa la interface Runnable la cual después le podemos pasar a un hilo para que funcione en este caso solo veremos la clase con el método run debido a que el código donde se ejecutan los hilos se encuentra en la ventana casino a continuación el código de la ventana jugar

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

import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author NANCY
 */
public class Jugar implements Runnable{
```

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

```

private Jugador jugador;
private Banca banca;
private int dineroApuesta;
private int numeroDePartida;
int contadorDineroApuesta=0;
private List<Apostar> apuestas;

public Jugar(Jugador jugador, Banca banca, int numeroDePartida) {
    this.jugador = jugador;
    this.banca = banca;
    this.numeroDePartida = numeroDePartida;
    this.dineroApuesta=10;
    apuestas = new ArrayList();
}

public Jugador getJugador() {
    return jugador;
}

public void setJugador(Jugador jugador) {
    this.jugador = jugador;
}

public Banca getBanca() {
    return banca;
}

public void setBanca(Banca banca) {
    this.banca = banca;
}

public int getDineroApuesta() {
    return dineroApuesta;
}

public void setDineroApuesta(int dineroApuesta) {
    this.dineroApuesta = dineroApuesta;
}

public int getNumeroDePartida() {
    return numeroDePartida;
}

public void setNumeroDePartida(int numeroDePartida) {
    this.numeroDePartida = numeroDePartida;
}

public int getContadorDineroApuesta() {
    return contadorDineroApuesta;
}

public void setContadorDineroApuesta(int contadorDineroApuesta) {
    this.contadorDineroApuesta = contadorDineroApuesta;
}

```

```
}

public List<Apostar> getApuestas() {
    return apuestas;
}

public void setApuestas(List<Apostar> apuestas) {
    this.apuestas = apuestas;
}

public boolean EnQuiebra(){
    return jugador.getSaldo() >=10;
}

public synchronized void sumarApuesta(Apostar apostar){
    apuestas.add(apostar);
}

public void DisminuirDineroJugador(int saldo){
    int SaldoN = jugador.getSaldo() - saldo;
    jugador.setSaldo(SaldoN);
}

public int apostarNumero(){
    int NumeroAlazar = (int)(Math.random()*36 +1);
    DisminuirDineroJugador(10);
    banca.setSaldo(banca.getSaldo()+10);
    return NumeroAlazar;
}

public void aumentarSaldoJugador(int saldo){
    int SaldoN = jugador.getSaldo() + saldo;
    jugador.setSaldo(SaldoN);
    jugador.setVictorias(jugador.getVictorias()+1);
}

public void numeroGanador(){
    int saldoBancaMomento = banca.aumentarSaldo(Thread.currentThread());
    aumentarSaldoJugador(saldoBancaMomento);
    banca.setSaldo(banca.getSaldo()-saldoBancaMomento);
}

public String apostarPARIMPAR(){
    try {
        boolean par = Random.class.newInstance().nextBoolean();
        DisminuirDineroJugador(10);
        banca.setSaldo(banca.getSaldo()+10);
        if(par){
            return "true";
        }else{
            return "false";
        }
    } catch (InstantiationException | IllegalAccessException ex) {
        Logger.getLogger(Jugar.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```


```
}
return "false";
}

public int apostarMartingala(){
    int NumeroAlazar = (int)(Math.random()*36 + 1);
    DisminuirDineroJugador(getDineroApuesta());
    banca.setSaldo(banca.getSaldo()+getDineroApuesta());
    return NumeroAlazar;
}

public void perderMartingala(){
    setDineroApuesta(getDineroApuesta()*2);
}

@Override
public void run() {
    if(EnQuieba()){
        Apostar apuesta = new Apostar();
        switch (jugador.getTipoapuesta()){
            case("NUMERO") -> {
                int numeroRuleta = apostarNumero();
                jugador.setNumeroapostado(String.valueOf(numeroRuleta));
                if(Integer.valueOf(jugador.getNumeroruleta())== numeroRuleta){
                    numeroGanador();
                    apuesta.setCantidadapostada(10);
                    apuesta.setCodigoJugadorFk(jugador);
                    apuesta.setGanador("Jugador");
                    apuesta.setNumeroganador(jugador.getNumeroruleta());
                    apuesta.setNumeroapostado(String.valueOf(numeroRuleta));
                    apuesta.setTipoapuesta(jugador.getTipoapuesta());
                    apuesta.setNumeropartida(getNumeroDePartida());
                    apuesta.setSaldobanca(banca.getSaldo());
                    apuesta.setSaldojugador(jugador.getSaldo());
                    //System.out.println(apuesta);
                }else{
                    jugador.setDerrotas(jugador.getDerrotas()+1);
                    apuesta.setCantidadapostada(10);
                    apuesta.setCodigoJugadorFk(jugador);
                    apuesta.setGanador("Banca");
                    apuesta.setNumeroganador(jugador.getNumeroruleta());
                    apuesta.setNumeroapostado(String.valueOf(numeroRuleta));
                    apuesta.setTipoapuesta(jugador.getTipoapuesta());
                    apuesta.setNumeropartida(getNumeroDePartida());
                    apuesta.setSaldobanca(banca.getSaldo());
                    apuesta.setSaldojugador(jugador.getSaldo());
                    //System.out.println(apuesta);
                }
                apuesta.setNumeropartida(numeroDePartida);
                apuesta.setSaldojugador(jugador.getSaldo());
                apuesta.setSaldobanca(banca.getSaldo());
                sumarApuesta(apuesta);
            }
            case ("PARIMPAR") -> {
```

```
String valor = (String.valueOf(apostarPARIMPAR()));
if(Integer.valueOf(jugador.getNumeroruleta())%2==0 && valor.equals("true")){
    numeroGanador();
    apuesta.setCantidadapostada(10);
    apuesta.setCodigoJugadorFk(jugador);
    apuesta.setGanador("Jugador");
    apuesta.setNumeroganador(jugador.getNumeroruleta());
    apuesta.setNumeroapostado(valor);
    apuesta.setTipoapuesta(jugador.getTipoapuesta());
    apuesta.setNumeropartida(getNumeroDePartida());
    apuesta.setSaldobanca(banca.getSaldo());
    apuesta.setSaldojugador(jugador.getSaldo());
    //System.out.println(apuesta);
}else{
    jugador.setDerrotas(jugador.getDerrotas()+1);
    apuesta.setCantidadapostada(10);
    apuesta.setCodigoJugadorFk(jugador);
    apuesta.setGanador("Banca");
    apuesta.setNumeroganador(jugador.getNumeroruleta());
    apuesta.setNumeroapostado(valor);
    apuesta.setTipoapuesta(jugador.getTipoapuesta());
    apuesta.setNumeropartida(getNumeroDePartida());
    apuesta.setSaldobanca(banca.getSaldo());
    apuesta.setSaldojugador(jugador.getSaldo());
    //System.out.println(apuesta);
}
apuesta.setNumeropartida(numeroDePartida);
apuesta.setSaldojugador(jugador.getSaldo());
apuesta.setSaldobanca(banca.getSaldo());
sumarApuesta(apuesta);
}
default -> {
    int NumeroRuletaMartingala = apostarMartingala();
    jugador.setNumeroapostado(String.valueOf(NumeroRuletaMartingala));
    if(Integer.valueOf(jugador.getNumeroruleta())== NumeroRuletaMartingala){
        numeroGanador();
        apuesta.setCantidadapostada(this.getDineroApuesta());
        apuesta.setCodigoJugadorFk(jugador);
        apuesta.setGanador("Jugador");
        apuesta.setNumeroganador(jugador.getNumeroruleta());
        apuesta.setNumeroapostado(String.valueOf(NumeroRuletaMartingala));
        apuesta.setTipoapuesta(jugador.getTipoapuesta());
        apuesta.setNumeropartida(getNumeroDePartida());
        apuesta.setSaldobanca(banca.getSaldo());
        apuesta.setSaldojugador(jugador.getSaldo());
        //System.out.println(apuesta);
    }else{
        perderMartingala();
        jugador.setDerrotas(jugador.getDerrotas()+1);
        apuesta.setCantidadapostada(this.getDineroApuesta()/2);
        apuesta.setCodigoJugadorFk(jugador);
        apuesta.setGanador("Banca");
        apuesta.setNumeroganador(jugador.getNumeroruleta());
        apuesta.setNumeroapostado(String.valueOf(NumeroRuletaMartingala));
    }
```


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

```

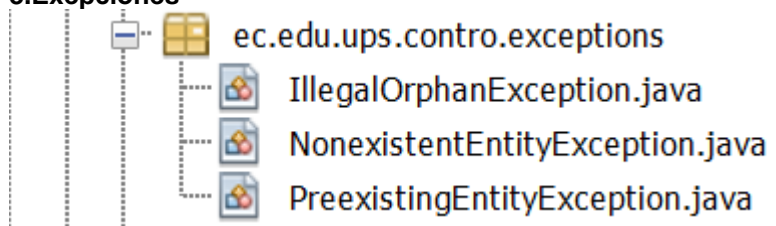
        apuesta.setTipoapuesta(jugador.getTipoapuesta());
        apuesta.setNumeropartida(getNumeroDePartida());
        apuesta.setSaldobanca(banca.getSaldo());
        apuesta.setSaldojugador(jugador.getSaldo());
        //System.out.println(apuesta);
    }
    apuesta.setNumeropartida(numeroDePartida);
    apuesta.setSaldojugador(jugador.getSaldo());
    apuesta.setSaldobanca(banca.getSaldo());
    sumarApuesta(apuesta);
}
}
}
}
}

411 Thread hiloNumero = new Thread(listajugarNumero.get(i));
412 Thread hiloParImpar = new Thread(listajugarParImpar.get(i));
413 Thread hiloMartingala = new Thread(listajugarMartingala.get(i));
414
415
416
417 hiloNumero.setName("Jugar " + i + "N");
418 hiloParImpar.setName("Jugar " + i + "PI");
419 hiloMartingala.setName("Jugar " + i + "M");
420
421 listaHilos.add(hiloNumero);
422 listaHilos.add(hiloParImpar);
423 listaHilos.add(hiloMartingala);
424
425 hiloNumero.start();
426 hiloParImpar.start();
427 hiloMartingala.start();
428

```

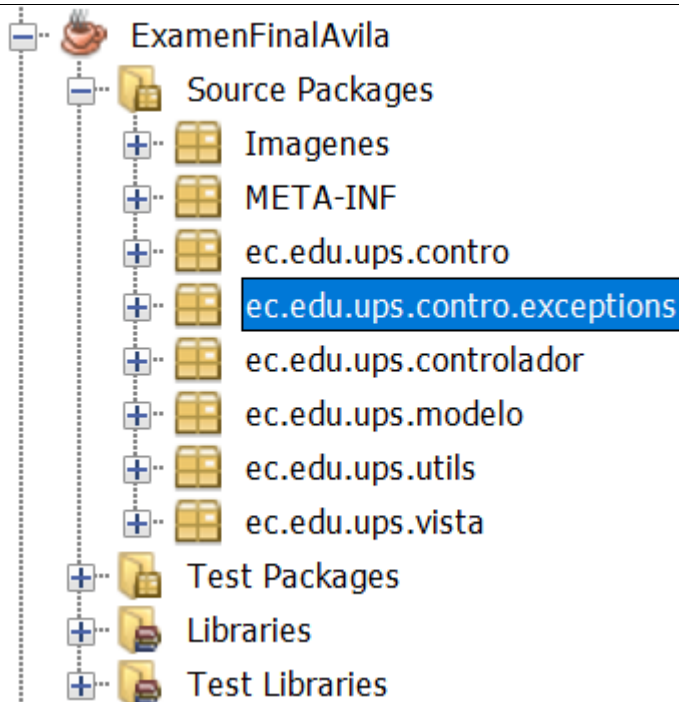
En la imagen se encuentra un fragmento del código de la ventana Casino en donde vemos como se implementan los hilos ya para inicializar y realizar lo que se encuentra en el método run.

3.Exepciones



En esta imagen podemos ver el paquete expeciones que nos ayuda a controlar ciertas excepciones del controlador

4.Patron de Diseño MVC



Como podemos ver en la imagen se cumple el patrón de diseño MVC con unos paquetes extras como el de imágenes el paquete de utils que nos ayuda a controlar la persistencia y el paquete del jpa

4. Programa.

En el programa podemos encontrar varios paquetes como podemos ver el paquete del modelo contiene las siguientes clases y vamos a ver el código de cada uno de ellas.


Paquete Modelo

Clase Apostar

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

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
```

```
/**
 *
```


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

```

* @author NANCY
*/
@Entity
@Table(name = "apostar")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Apostar.findAll", query = "SELECT a FROM Apostar a"),
    @NamedQuery(name = "Apostar.findById", query = "SELECT a FROM Apostar a WHERE a.id = :id"),
    @NamedQuery(name = "Apostar.findByGanador", query = "SELECT a FROM Apostar a WHERE a.ganador = :ganador"),
    @NamedQuery(name = "Apostar.findByNumeropartida", query = "SELECT a FROM Apostar a WHERE a.numeropartida = :numeropartida"),
    @NamedQuery(name = "Apostar.findByCantidadapostada", query = "SELECT a FROM Apostar a WHERE a.cantidadapostada = :cantidadapostada"),
    @NamedQuery(name = "Apostar.findByNumeroapostado", query = "SELECT a FROM Apostar a WHERE a.numeroapostado = :numeroapostado"),
    @NamedQuery(name = "Apostar.findByNumeroganador", query = "SELECT a FROM Apostar a WHERE a.numeroganador = :numeroganador"),
    @NamedQuery(name = "Apostar.findBySaldobanca", query = "SELECT a FROM Apostar a WHERE a.saldobanca = :saldobanca"),
    @NamedQuery(name = "Apostar.findBySaldojugador", query = "SELECT a FROM Apostar a WHERE a.saldojugador = :saldojugador"),
    @NamedQuery(name = "Apostar.findByTipoapuesta", query = "SELECT a FROM Apostar a WHERE a.tipoapuesta = :tipoapuesta"))
public class Apostar implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id")
    private Integer id;
    @Size(max = 255)
    @Column(name = "ganador")
    private String ganador;
    @Column(name = "numeropartida")
    private Integer numeropartida;
    @Column(name = "cantidadapostada")
    private Integer cantidadapostada;
    @Size(max = 255)
    @Column(name = "numeroapostado")
    private String numeroapostado;
    @Size(max = 255)
    @Column(name = "numeroganador")
    private String numeroganador;
    @Column(name = "saldobanca")
    private Integer saldobanca;
    @Column(name = "saldojugador")
    private Integer saldojugador;
    @Size(max = 255)
    @Column(name = "tipoapuesta")
    private String tipoapuesta;
    @JoinColumn(name = "codigo_jugador_fk", referencedColumnName = "id")
    @ManyToOne

```

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

```

private Jugador codigoJugadorFk;

public Apostar() {
}

public Apostar(Integer id) {
    this.id = id;
}

public Integer getId() {
    return id;
}

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

public String getGanador() {
    return ganador;
}

public void setGanador(String ganador) {
    this.ganador = ganador;
}

public Integer getNumeropartida() {
    return numeropartida;
}

public void setNumeropartida(Integer numeropartida) {
    this.numeropartida = numeropartida;
}

public Integer getCantidadapostada() {
    return cantidadapostada;
}

public void setCantidadapostada(Integer cantidadapostada) {
    this.cantidadapostada = cantidadapostada;
}


public String getNumeroapostado() {
    return numeroapostado;
}

public void setNumeroapostado(String numeroapostado) {
    this.numeroapostado = numeroapostado;
}

public String getNumeroganador() {
    return numeroganador;
}

public void setNumeroganador(String numeroganador) {

```

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

```

this.numeroganador = numeroganador;
}

public Integer getSaldobanca() {
    return saldobanca;
}

public void setSaldobanca(Integer saldobanca) {
    this.saldobanca = saldobanca;
}

public Integer getSaldojugador() {
    return saldojugador;
}

public void setSaldojugador(Integer saldojugador) {
    this.saldojugador = saldojugador;
}

public String getTipoapuesta() {
    return tipoapuesta;
}

public void setTipoapuesta(String tipoapuesta) {
    this.tipoapuesta = tipoapuesta;
}


public Jugador getCodigoJugadorFk() {
    return codigoJugadorFk;
}

public void setCodigoJugadorFk(Jugador codigoJugadorFk) {
    this.codigoJugadorFk = codigoJugadorFk;
}

@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 Apostar)) {
        return false;
    }
    Apostar other = (Apostar) object;
    if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
        return false;
    }
    return true;
}

```

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

```

@Override
public String toString() {
    return "Apostar{" + "id=" + id + ", ganador=" + ganador + ", numeropartida=" + numeropartida + ",
cantidadapostada=" + cantidadapostada + ", numeroapostado=" + numeroapostado + ", numeroganador=" +
numeroganador + ", saldobanca=" + saldobanca + ", saldojugador=" + saldojugador + ", tipoapuesta=" +
tipoapuesta + ", codigoJugadorFk=" + codigoJugadorFk + '}';
}
}

```

Como podemos ver en el código de la clase Apostar tenemos los atributos que después aran la tabla en la base de datos entonces tenemos todo lo que le hace a una apuesta por así decirlo cada atributo con su método get and set y una relación con la clase jugador, se puede visualizar el uso del JPA.

Clase Banca

```

package ec.edu.ups.modelo;

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.xml.bind.annotation.XmlRootElement;

/**
 *
 * @author NANCY
 */
@Entity
@Table(name = "banca")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Banca.findAll", query = "SELECT b FROM Banca b"),
    @NamedQuery(name = "Banca.findById", query = "SELECT b FROM Banca b WHERE b.id = :id"),
    @NamedQuery(name = "Banca.findBySaldo", query = "SELECT b FROM Banca b WHERE b.saldo = :saldo"))
public class Banca implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id")
    private Integer id;
    @Column(name = "saldo")
    private Integer saldo;

    public Banca() {
    }
}

```

```
public Banca(Integer id) {
    this.id = id;
}

public Integer getId() {
    return id;
}

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

public synchronized Integer getSaldo() {
    return saldo;
}

public synchronized void setSaldo(Integer saldo) {
    this.saldo = saldo;
}

public synchronized int aumentarSaldo(Thread thread){
    if(thread.getName().contains("N")){
        if(getSaldo())>360){
            return 360;
        }else{
            return 0;
        }
    }else if(thread.getName().contains("PI")){
        if(getSaldo())>20){
            return 20;
        }else{
            return 0;
        }
    }else{
        return 360;
    }
}

public synchronized int aumentarSaldo(Thread thread, int dineroApuesta){
    if(getSaldo())>dineroApuesta * 36){
        return dineroApuesta * 36;
    }else{
        return 0;
    }
}

@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 Banca)) {
        return false;
    }
    Banca other = (Banca) 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 "ec.edu.ups.modelo.Banca[ id=" + id + " ]";
}
}
```


Como podemos ver en el código de la clase Banca tenemos los atributos que después aran la tabla en la base de datos entonces tenemos todo lo que le hace a una Banca por así decirlo cada atributo con su método get and set, se puede visualizar el uso del JPA.

Clase Jugador

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

import java.io.Serializable;
import java.util.Collection;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

/**
 *
 * @author NANCY
 */
@Entity
@Table(name = "jugador")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Jugador.findAll", query = "SELECT j FROM Jugador j"),
```


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

```


    @NamedQuery(name = "Jugador.findById", query = "SELECT j FROM Jugador j WHERE j.id = :id"),
    @NamedQuery(name = "Jugador.findByName", query = "SELECT j FROM Jugador j WHERE j.nombre = :nombre"),
    @NamedQuery(name = "Jugador.findByNumeroapostado", query = "SELECT j FROM Jugador j WHERE j.numeroapostado = :numeroapostado"),
    @NamedQuery(name = "Jugador.findByDerrotas", query = "SELECT j FROM Jugador j WHERE j.derrotas = :derrotas"),
    @NamedQuery(name = "Jugador.findByNumeroruleta", query = "SELECT j FROM Jugador j WHERE j.numeroruleta = :numeroruleta"),
    @NamedQuery(name = "Jugador.findBySaldo", query = "SELECT j FROM Jugador j WHERE j.saldo = :saldo"),
    @NamedQuery(name = "Jugador.findByTipoapuesta", query = "SELECT j FROM Jugador j WHERE j.tipoapuesta = :tipoapuesta"),
    @NamedQuery(name = "Jugador.findByVictorias", query = "SELECT j FROM Jugador j WHERE j.victorias = :victorias"))
public class Jugador implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id")
    private Integer id;
    @Size(max = 255)
    @Column(name = "nombre")
    private String nombre;
    @Size(max = 255)
    @Column(name = "numeroapostado")
    private String numeroapostado;
    @Column(name = "derrotas")
    private Integer derrotas;
    @Size(max = 255)
    @Column(name = "numeroruleta")
    private String numeroruleta;
    @Column(name = "saldo")
    private Integer saldo;
    @Size(max = 255)
    @Column(name = "tipoapuesta")
    private String tipoapuesta;
    @Column(name = "victorias")
    private Integer victorias;
    @OneToMany(mappedBy = "codigoJugadorFk")
    private Collection<Apostar> apostarCollection;

    public Jugador() {
    }

    public Jugador(String Nombre) {
        this.nombre = Nombre;
        this.saldo=1000;
        this.victorias=0;
        this.derrotas=0;
        this.numeroapostado="-1";
        this.numeroruleta="-1";
        this.tipoapuesta="";
    }

```

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

```

}
public Jugador(Integer id) {
    this.id = id;
}

public Integer getId() {
    return id;
}

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

public String getNombre() {
    return nombre;
}

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

public String getNumeroapostado() {
    return numeroapostado;
}

public void setNumeroapostado(String numeroapostado) {
    this.numeroapostado = numeroapostado;
}

public Integer getDerrotas() {
    return derrotas;
}

public void setDerrotas(Integer derrotas) {
    this.derrotas = derrotas;
}


public String getNumeroruleta() {
    return numeroruleta;
}

public void setNumeroruleta(String numeroruleta) {
    this.numeroruleta = numeroruleta;
}

public Integer getSaldo() {
    return saldo;
}

public void setSaldo(Integer saldo) {
    this.saldo = saldo;
}

```

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

```

public String getTipoapuesta() {
    return tipoapuesta;
}

public void setTipoapuesta(String tipoapuesta) {
    this.tipoapuesta = tipoapuesta;
}

public Integer getVictorias() {
    return victorias;
}

public void setVictorias(Integer victorias) {
    this.victorias = victorias;
}

@XmlTransient
public Collection<Apostar> getApostarCollection() {
    return apostarCollection;
}


public void setApostarCollection(Collection<Apostar> apostarCollection) {
    this.apostarCollection = apostarCollection;
}

@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 Jugador)) {
        return false;
    }
    Jugador other = (Jugador) 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 "Jugador{" + "id=" + id + ", nombre=" + nombre + ", numeroapostado=" + numeroapostado + ", derrotas=" + derrotas + ", numeroruleta=" + numeroruleta + ", saldo=" + saldo + ", tipoapuesta=" + tipoapuesta + ", victorias=" + victorias + ", apostarCollection=" + apostarCollection + "}";
}

```

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

Como podemos ver en el código de la clase Jugador tenemos los atributos que después aran la tabla en la base de datos entonces tenemos todo lo que le hace a un Jugador por así decirlo cada atributo con su método get and set y con relación con la clase banca y la clase Apostar, se puede visualizar el uso del JPA.

En este paquete también se encuentra la clase de jugar, pero ya fue explicada en el punto de los hilos.

Paquete Controlador

Controlador Abstrac

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


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

/**
 *
 * @author NANCY
 */
public abstract class ControladorAbstrac <E> {
    private List <E> listaGenerica;
    private Class<E> clase;
    private EntityManager em;

    public abstract List<E> findAll();
    public abstract int codigo();

    public ControladorAbstrac() {
        listaGenerica= new ArrayList();
        //java.lang.reflect.Type t= getClass().getGenericSuperclass();
        Type t = getClass().getGenericSuperclass();
        ParameterizedType pt = (ParameterizedType) t;
        clase= (Class) pt.getActualTypeArguments()[0];
        em=JPAUtils.getEntityManager();
    }
    public ControladorAbstrac(EntityManager em) {
        listaGenerica= new ArrayList();
        //java.lang.reflect.Type t= getClass().getGenericSuperclass();
        Type t =getClass().getGenericSuperclass();
        ParameterizedType pt = (ParameterizedType) t;
        clase= (Class) pt.getActualTypeArguments()[0];
        this.em=em;
    }

    public boolean crear(E objeto){
        em.getTransaction().begin();
        em.persist(objeto);
        em.getTransaction().commit();
        listaGenerica.add(objeto);
        return true;
    }
}
```

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

```

public boolean eliminar(E objeto){
    em.getTransaction().begin();
    em.remove(em.merge(objeto));
    em.getTransaction().commit();
    listaGenerica.remove(objeto);
    return true;
}

public boolean actualizar(E objeto){
    em.getTransaction().begin();
    em.merge(objeto);
    em.getTransaction().commit();
    return true;
}

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 List<E> getListaGenerica() {
    return listaGenerica;
}

public void setListaGenerica(List<E> listaGenerica) {
    this.listaGenerica = listaGenerica;
}

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;
}
}

```

Este es el código de la clase del controlador genérico entonces aquí claramente podemos ver el uso de una programación genérica para los demás controladores y a su vez el uso de un JPA para hacer los métodos del CRUD para los objetos o datos que van hacer enviados a la base de datos es por eso que este controlador va a ser abstracto para que los demás controladores hereden los métodos de esta clase.

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

Controlador Banca

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

import ec.edu.ups.modelo.Banca;
import java.util.List;
import javax.persistence.Query;

/**
 *
 * @author NANCY
 */
public class ControladorBanca extends ControladorAbstrac<Banca>{

    @Override
    public List<Banca> findAll() {
        Query consulta = getEm().createNamedQuery("Banca.findAll");
        return consulta.getResultList();
    }

    @Override
    public int codigo() {
        return 1;
    }

}
```

Código de la clase controlador banca que con la ayuda de la programación genérica el código se reduce en su mayoría teniendo solo métodos para poder pasar el código y obtener la lista de datos de la base de datos.

Controlador Jugador


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

import ec.edu.ups.modelo.Jugador;
import java.util.Collections;
import java.util.List;
import javax.persistence.Query;

/**
 *
 * @author NANCY
 */
public class ControladorJugador extends ControladorAbstrac<Jugador>{

    @Override
    public List<Jugador> findAll() {
        Query consulta = getEm().createNamedQuery("Jugador.findAll");
        return consulta.getResultList();
    }

    @Override
```

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

```

public int codigo() {
    var lista = findAll();
    Collections.sort(lista, (Jugador j1, Jugador j2) -> j1.getId().compareTo(j2.getId()));

    if(!lista.isEmpty()){
        return lista.get(lista.size()-1).getId();
    }else{
        return 0;
    }
}
}

```

Código de la clase controlador Jugador que con la ayuda de la programación genérica el código se reduce en su mayoría teniendo solo métodos para poder pasar el código y obtener la lista de datos de la base de datos.

Controlador Apostar

```

package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Apostar;
import java.util.Collections;
import java.util.List;
import javax.persistence.Query;

/**
 *
 * @author NANCY
 */
public class ControladorApostar extends ControladorAbstrac<Apostar>{


    @Override
    public List<Apostar> findAll() {
        Query consulta = getEm().createNamedQuery("Apostar.findAll");
        return consulta.getResultList();
    }

    @Override
    public int codigo() {
        var lista = findAll();
        Collections.sort(lista, (Apostar a1, Apostar a2) -> a1.getId().compareTo(a2.getId()));
        return lista.get(lista.size()-1).getId();
    }

    public int partidas(){
        var lista = findAll();

        if(lista.isEmpty()){
            return 0;
        }else{
            Collections.sort(lista, (Apostar a1, Apostar a2) -> a1.getId().compareTo(a2.getId()));
            return lista.get(lista.size()-1).getId();
        }
    }
}

```

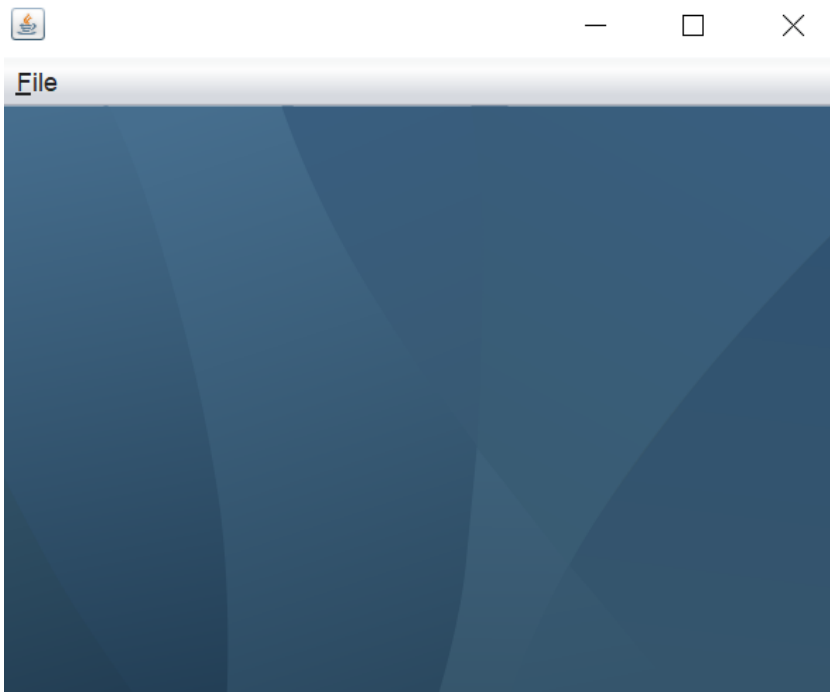
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```
}
}
```

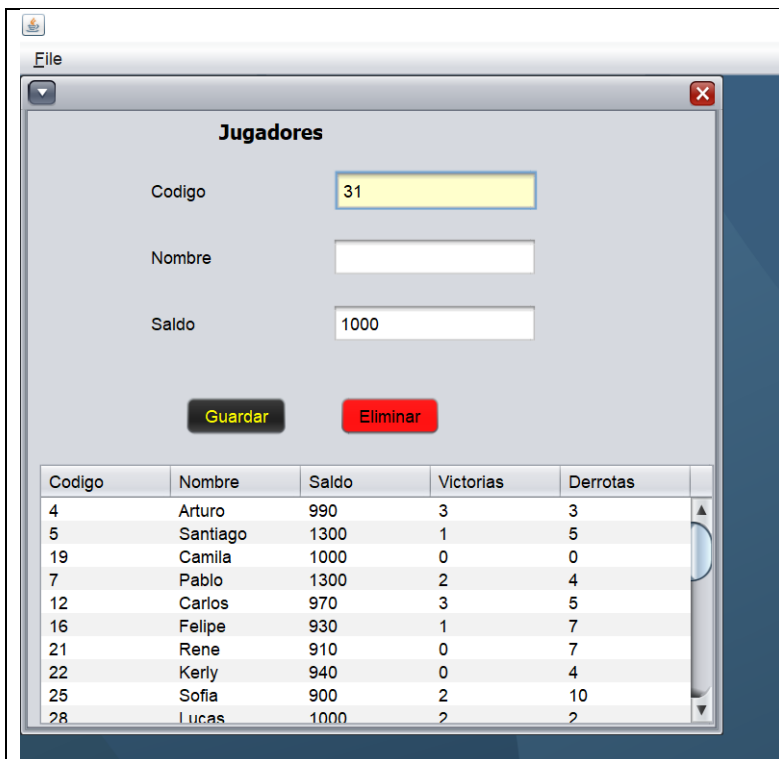
Código de la clase controlador Apostar que con la ayuda de la programación genérica el código se reduce en su mayoría teniendo solo métodos para poder pasar el código y obtener la lista de datos de la base de datos.

Paquete vista

En este paquete tenemos las ventanas que aran la parte visual del programa debido a la extensión del código se adjunta el enlace del Git en donde se encuentra este código de las ventanas.



Al ejecutar el programa nos sale la siguiente ventana en donde tenemos para seleccionar la otra ventana para continuar con el programa.



Jugadores

Codigo:

Nombre:

Saldo:

Guardar **Eliminar**

Codigo	Nombre	Saldo	Victorias	Derrotas
4	Arturo	990	3	3
5	Santiago	1300	1	5
19	Camila	1000	0	0
7	Pablo	1300	2	4
12	Carlos	970	3	5
16	Felipe	930	1	7
21	Rene	910	0	7
22	Kerly	940	0	4
25	Sofia	900	2	10
28	Lucas	1000	2	2

Si presionamos en gestión jugadores se nos abre la siguiente ventana en donde tenemos la lista de los jugadores que se encuentran en la base de datos y así podemos agregar mas jugadores o editar en este caso el saldo al crear un nuevo jugador va a ser de 1000 dólares de entrada. Y en caso de querer eliminar un jugador seleccionamos el jugador de la tabla y presionamos eliminar.



HOLLAND CASINO

Reservir Apuestas **Iniciar Juego** **Pausa**

Codigo	Nombre	Tipo de Juego

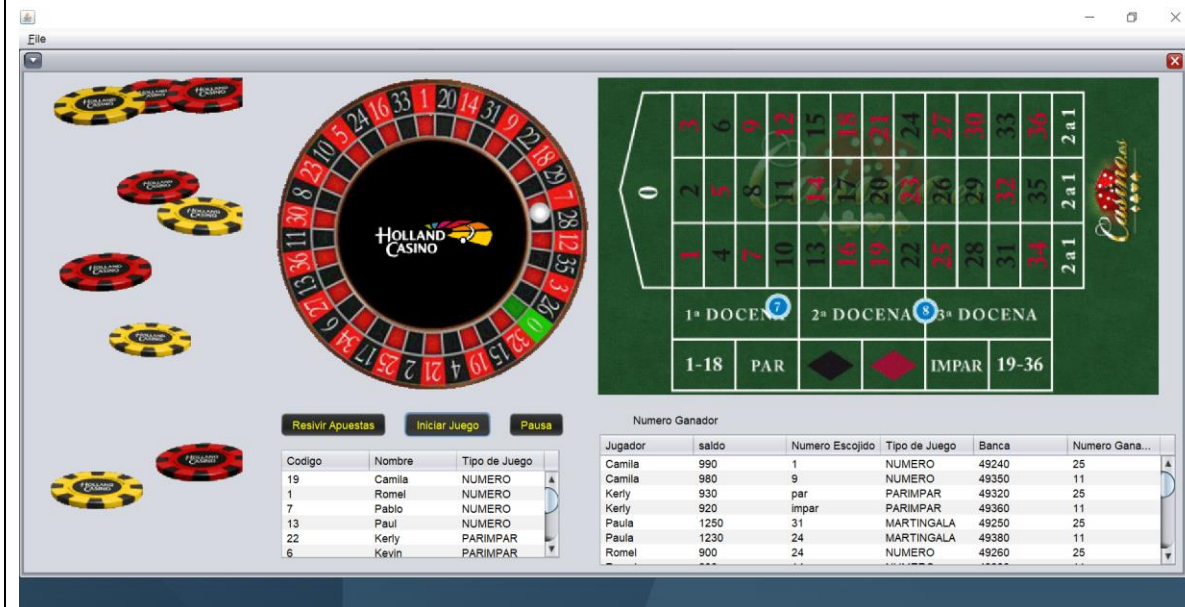
Numero Ganador


Jugador	saldo	Numero Escogido	Tipo de Juego	Banca	Numero Ganador

Si presionamos en casino se nos abre la siguiente ventana en donde está la función para jugar a la ruleta



Si presionamos en el botón de recibir apuestas se cargan aleatoriamente los jugadores que van a participar en la ruleta con la modalidad aleatoria



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



Numero	Ganador	Cantidad Apostada	Numero Apostado	Numero Ganador	Saldo Banca	Jugador	Saldo Jugador	Tipo Apuesta
1	Banca	10	7	35	49850	Romel	980	NUMERO
2	Banca	10	8	23	49920	Romel	980	NUMERO
45	Jugador	10	par	18	50030	Romel	1000	PARIMPAR
46	Jugador	10	par	10	50050	Romel	1000	PARIMPAR

Llenamos el código y presionamos buscar y se cargaran los datos de las apuestas realizadas por la personas buscada.

RESULTADO(S) OBTENIDO(S):

Como resultados obtenidos de este examen podemos encontrar el uso de los temas aprendidos en todo el ciclo en un solo programa realizando así programas mas complejos y con menos cantidad de código manejando una base de datos dentro del programa.

CONCLUSIONES:

En conclusión el uso de hilos nos ayuda realizar varios procesos a la vez en este caso varias apuestas dentro del programa y el JPA nos ayudo para poder guardar los datos en la base de datos de una manera muy fácil utilizando menos código.

RECOMENDACIONES:

No hay recomendaciones.

Estudiantes: Romel Ávila

Firma:

