UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

MANUAL TÉCNICO

TEMA:

"AUTOMATIZACIÓN DEL PROCESO DE RECOPILACIÓN DE INFORMACIÓN PARA REALIZAR ENCUESTAS A TRAVÉS DE UNA APLICACIÓN CON DATOS GEOREFERENCIADOS UTILIZANDO LA HERRAMIENTA MAVEN"

AUTORA:

Geovana Gabriela Valladares Correa

DIRECTOR:

Ing. Mauricio Rea, Msc.

Ibarra – Ecuador

2016

Tabla de contenido

Tabla de Ilustraciones

1. EL Proyecto

EL proyecto “AUTOMATIZACIÓN DEL PROCESO DE RECOPILACIÓN DE INFORMACIÓN PARA REALIZAR ENCUESTAS A TRAVÉS DE UNA APLICACIÓN CON DATOS GEOREFERENCIADOS UTILIZANDO LA HERRAMIENTA MAVEN”, se enfoca en el desarrollo de una aplicación web con datos georeferenciados que cree, publique y tabule encuestas propuestas por un usuario encuestador que posee un monedero que almacene los depósitos de monedas de tipo FastPoll, las cuáles son necesarias para poder publicar una encuesta. También permite contestar las encuestas a través de un usuario de tipo encuestado que recibe una retribución económica (2 Fast Poll) por la información brindada.

2. Herramientas para el desarrollo

2.1. IDE Eclipse

Se utilizó el entorno de desarrollo Mars.1 Release (4.5.1).

2.2. Navicat

Navicat es un modelador de base de datos, versión 11.0.8.

2.3. PostgreSQL

Gestor de Base de Datos relacional orientado a objetos PostgreSQL y pgAdmin versión 1.20.0.

2.4. JDK

EL Java Developer Kit en sus versiones 7.0.

2.5. Prime Faces

EL framework Laravel para el diseño MVC de la aplicación web, versión 5.0.

2.6. Google Chrome

Se utilizó las herramientas para desarrollo del navegador Google Chrome, versión 49.0.2623.112 m.

2.7. Servidor web local

Se utilizó un servidor web local Apache Tomcat 7.0.65.

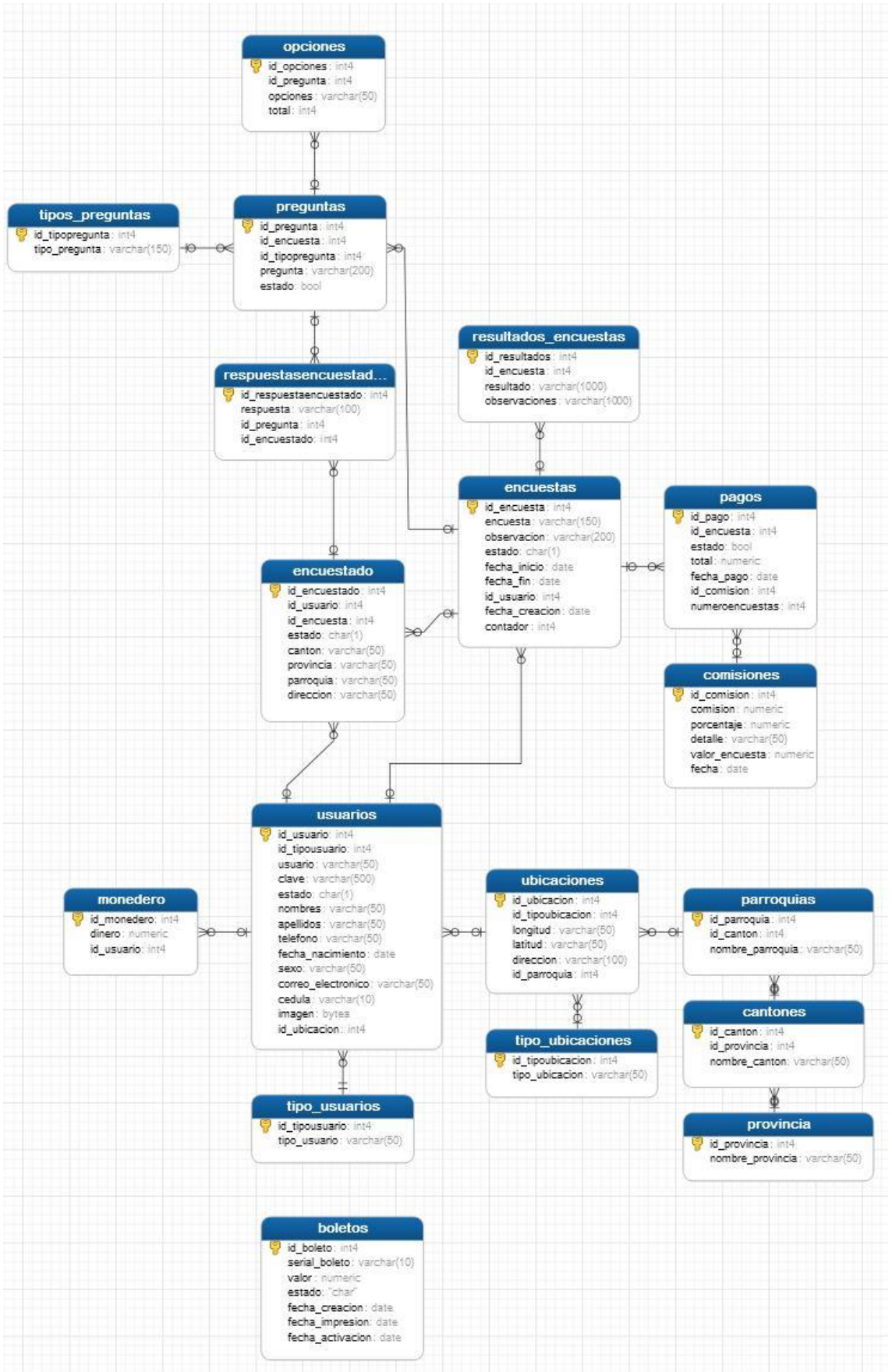
2.8. Servidor de producción

Se utilizó Openshift con un servidor Tomcat 7 (JBoss EWS 2.0) y base de datos PostgreSQL 9.2, especificaciones de hardware: 1 cpu, 512 mb en ram y 1 gb de almacenamiento.

2.9. Computador

Nombre del SO:Windows 8.1 Pro con Media Center 64-bit (6.3, Build 9600) (9600.winblue_gdr.131030-1505)
Fabricante del SO: Microsoft Corporation
Nombre del sistema: GABY-PC
Fabricante del sistema: SONY VAIO
Modelo del sistema: SVE14113ELW
Tipo de sistema PC basado en x64
Procesador: Intel(R) Core(TM) i3-2370M CPU @ 2.40GHz (4 CPUs), ~2.4GHz
Ram: 6 GB

3. Diagrama Entidad Relación


opciones
id_opciones: int4
id_pregunta: int4
opciones: varchar(50)
total: int4
tipos_preguntas
id_tipopregunta: int4
tipo_pregunta: varchar(150)
preguntas
id_pregunta: int4
id_encuesta: int4
id_tipopregunta: int4
pregunta: varchar(200)
estado: bool
respuestasencuestad...
id_respuestaencuestado: int4
respuesta: varchar(100)
id_pregunta: int4
id_encuestado: int4
resultados_encuestas
id_resultados: int4
id_encuesta: int4
resultado: varchar(1000)
observaciones: varchar(1000)
encuestas
id_encuesta: int4
encuesta: varchar(150)
observacion: varchar(200)
estado: char(1)
fecha_inicio: date
fecha_fin: date
id_usuario: int4
fecha_creacion: date
contador: int4
pagos
id_pago: int4
id_encuesta: int4
estado: bool
total: numeric
fecha_pago: date
id_comision: int4
numeroencuestas: int4
encuestado
id_encuestado: int4
id_usuario: int4
id_encuesta: int4
estado: char(1)
canton: varchar(50)
provincia: varchar(50)
parroquia: varchar(50)
direccion: varchar(50)
comisiones
id_comision: int4
comision: numeric
porcentaje: numeric
detalle: varchar(50)
valor_encuesta: numeric
fecha: date
usuarios
id_usuario: int4
id_tipousuario: int4
usuario: varchar(50)
clave: varchar(500)
estado: char(1)
nombres: varchar(50)
apellidos: varchar(50)
telefono: varchar(50)
fecha_nacimiento: date
sexo: varchar(50)
correo_electronico: varchar(50)
cedula: varchar(10)
imagen: bytea
id_ubicacion: int4
monedero
id_monedero: int4
dinero: numeric
id_usuario: int4
ubicaciones
id_ubicacion: int4
id_tipoubicacion: int4
longitud: varchar(50)
latitud: varchar(50)
direccion: varchar(100)
id_parroquia: int4
parroquias
id_parroquia: int4
id_canton: int4
nombre_parroquia: varchar(50)
cantones
id_canton: int4
id_provincia: int4
nombre_canton: varchar(50)
provincia
id_provincia: int4
nombre_provincia: varchar(50)
tipo_ubicaciones
id_tipoubicacion: int4
tipo_ubicacion: varchar(50)
tipo_usuarios
id_tipousuario: int4
tipo_usuario: varchar(50)
boletos
id_boleto: int4
serial_boleto: varchar(10)
valor: numeric
estado: "char"
fecha_creacion: date
fecha_impresion: date
fecha_activacion: date

3.1. Definición de Tablas

- Boletos

Name	Data type	Not Null?	Primary key?	Default	Comment
id_boleto	integer	Yes	Yes	nextval('seq_boletos'::regclass)	
serial_boleto	character varying(10)	No	No		
valor	numeric	No	No		Contiene el valor en dólares
estado	"char"	No	No		
fecha_creacion	date	No	No		fecha de creación del boleto
fecha_impresion	date	No	No		Fecha de impresión o emisión del boleto
fecha_activacion	date	No	No		Fecha de activación del boleto

- Cantones

Name	Data type	Not Null?	Primary key?	Default	Comment
id_canton	integer	Yes	Yes	nextval('seq_idcanton'::regclass)	Campo que almacena la clave principal de la tabla Cantones
id_provincia	integer	No	No		Campo que almacena la relación entre la tabla Provincias y Cantones
nombre_canton	character varying(50)	Yes	No		Tabla que almacena el nombre de los Cantones

- Comisiones

Name	Data type	Not Null ?	Primary key?	Default	Comment
id_comision	integer	Yes	Yes	nextval('seq_idcomisiones'::regclass)	Columna que almacena la clave primaria de la tabla comisiones
comision	numeric	No	No		Almacena el valor en dolares de la comision que se gana cuando se realiza una encuesta
porcentaje	numeric	No	No		Almacena el porcentaje que se cobra como comision por encuesta
detalle	character varying(50)	No	No		almacena alguna observacion de la comision
valor_encuesta	numeric	No	No		Valor de la encuesta

fecha	date	No	No		Fecha de establecimiento de la nueva comisión

- Encuestado

Name	Data type	Not Null?	Primary key?	Default	Comment
id_encuestado	integer	Yes	Yes	nextval('sq_idencuestado'::regclass)	
id_usuario	integer	No	No		
id_encuesta	integer	No	No		
estado	character(1)	No	No		
canton	character varying(50)	No	No		Canton al contestar la encuesta
provincia	character varying(50)	No	No		Provincia al contestar la encuesta
parroquia	character varying(50)	No	No		Parroquia al contestar la encuesta
direccion	character varying(50)	No	No		Direccion al contestar la encuesta

- Encuestas

Name	Data type	Not Null?	Primary key?	Default	Comment
id_encuesta	integer	Yes	Yes	nextval('seq_idencuesta'::regclass)	Campo que almacena la clave primaria de la tabla Encuestas
encuesta	character varying(150)	Yes	No		Campo que almacena el nombre de las encuestas
observacion	character varying(200)	No	No		Campo que almacena la descripción de la encuesta
estado	character(1)	No	No		Campo que permite visualizar si la encuesta esta o no activa en el sistema
fecha_inicio	date	No	No		Inicio encuesta
fecha_fin	date	No	No		Finalización Fecha
id_usuario	integer	No	No		Usuario
fecha_creacion	date	No	No		Fecha de creacion
contador	integer	No	No		Contabiliza el numero de encuestas a realizarse

- Monedero

Name	Data type	Not Null?	Primary key?	Default	Comment
id_monedero	integer	Yes	Yes	nextval('seq_monedero'::regclass)	Id principal
dinero	numeric	No	No		Almacena dinero
id_usuario	integer	No	No		Usuario

- Opciones

Name	Data type	Not Null ?	Primary key?	Default	Comment
id_opciones	integer	Yes	Yes	nextval('seq_idrespuesta'::regclass)	Campo que almacena la clave principal de la tabla opciones de las respuestas
id_pregunta	integer	No	No		Campo que guarda la relación entre las tablas opciones de respuesta y la tabla pregunta
opciones	character varying(50)	No	No		Campo que almacena la opción de respuesta/as para una pregunta
total	integer	No	No		ALMACENA EL TOTAL DE VECES QUE SE A CONTESTADO ESA OPCIÓN

- Pagos

Name	Data type	Not Null ?	Primary key?	Default	Comment
id_pago	integer	Yes	Yes	nextval('pago_id_pago_seq'::regclass)	Campo que almacena la clave primaria de la pago
id_encuesta	integer	No	No		Campo que almacena la relación enter las tablas encuesta y pagos
estado	boolean	No	No		Campo que almacena el estado del pago
total	numeric	No	No		Campo que almacena el total de dinero disponibleque el Encuestador paga al encuestador para realizar las encuestas
fecha_pago	date	No	No		
id_comision	integer	No	No		
numero encuestas	integer	No	No		

- Parroquias

Name	Data type	Not Null?	Primary key?	Default	Comment
id_parroquia	integer	Yes	Yes	nextval('seq_idparroquias'::regclass)	Campo que almacena la primary key de la tabla parroquia
id_canton	integer	No	No		Campo para almacenar la relación entre cantones y parroquias

nombre_parroquia	character varying(50)	No	No		Campo que almacena el nombre de la parroquia

- Preguntas

Name	Data type	Not Null?	Primary key?	Default	Comment
id_pregunta	integer	Yes	Yes	nextval('seq_idpregunta'::regclass)	Campo que almacena la clave principal de la tabla pregunta
id_encuesta	integer	No	No		campo que almacena la relación entre los tablas preguntas y encuestas
id_tipopregunta	integer	No	No		Campo que almacena la relación entre las tablas pregunta y tipopregunta
pregunta	character varying(200)	No	No		Campo que almacena las preguntas de la encuesta
estado	boolean	No	No		Columna que almacena el estado de las preguntas y permite ver si está o no activa

- Provincia

Name	Data type	Not Null?	Primary key?	Default	Comment
id_provincia	integer	Yes	Yes	nextval('seq_idprovincia'::regclass)	Columna que almacena una clave primaria de la tabla Provincia
nombre_provincia	character varying(50)	Yes	No		Columna que almacena el nombre de la provincia

- Respuestas Encuestados

Name	Data type	Not Null?	Primary key?	Default	Comment
id_respuestaencuestado	integer	Yes	Yes	nextval('seq_respuestasencuestados'::regclass)	Columna que guarda el primary key de la tabla Respuestasencuestados
respuesta	character varying(100)	No	No		Columna que almacena la respuesta del encuestado
id_pregunta	integer	No	No		Columna que almacena la relación de las tablas de preguntas y respuestas
id_encuestado	integer	No	No		

- Resultados Encuestados

Name	Data type	Not Null?	Primary key?	Default	Comment
id_resultados	integer	Yes	Yes	nextval('seq_idresultados'::regclass)	Columna que almacena la clave primaria de la tabla resultados_encuestas
id_encuesta	integer	No	No		Columna que almacena la relación entre encuestas y resultados
resultado	character varying(1000)	Yes	No		Columna que almacena los resultados de la encuesta
observaciones	character varying(1000)	No	No		Columna que permite almacenar alguna descripción de los resultados

- Tipo Preguntas

Name	Data type	Not Null?	Primary key?	Default	Comment
id_tipopregunta	integer	Yes	Yes	nextval('seq_idtipopregunta'::regclass)	Columna que permite almacenar la clave primaria de la tabla tipos_preguntas
tipo_pregunta	character varying(150)	No	No		Columna que almacena los nombres de los tipos de las preguntas que se pueden realizar en una encuesta WEb

- Tipo Ubicaciones

Name	Data type	Not Null?	Primary key?	Default	Comment
id_tipoubicacion	integer	Yes	Yes	nextval('seq_idtipoubicacion'::regclass)	Columna que almacena el id o codigo principal de la tabla tipo ubicación
tipo_ubicacion	character varying(50)	No	No		Columna que almacena el tipo de ubicación

- Tipo Usuarios

Name	Data type	Not Null?	Primary key?	Default	Comment
id_tipousuario	integer	Yes	Yes	nextval('seq_tipousuario'::regclass)	Columna que almacena el codigo primario de la tabla Tipo Usuario
tipo_usuario	character varying(50)	No	No		Columna que almacena el tipo de usuario

- Ubicaciones

Name	Data type	Not Null?	Primary key?	Default	Comment
id_ubicacion	integer	Yes	Yes	nextval('ubicacion_id_ubicacion_seq'::regclass)	columna que permite alamacenar la primary key de las tablas ubicaciones

id_tipoubicacion	integer	No	No		Colimna que muestra la relación de las tablas tipo ubicación y ubicación
longitud	character varying(50)	No	No		Columna que almacena la longitud de la ubicación
latitud	character varying(50)	No	No		Columna que almacena la latitud de la ubicación del Encuestado
direccion	character varying(100)	No	No		Columna que guarda la dirección de una persona al momento de registrarse
id_parroquia	integer	No	No		Relación entre las tablas ubicaciones y parroquia

- Usuarios

Name	Data type	Not Null?	Primary key?	Default	Comment
id_usuario	integer	Yes	Yes	nextval('seq_idusuario'::regclass)	Columna que contiene la clave primaria de la tabla Usuarios
id_tipousuario	integer	Yes	No		Columna de la relación entre la tabla tipousuario y usuarios
usuario	character varying(50)	Yes	No		Columna que tiene el nombre de usuario encuestado
clave	character varying(500)	Yes	No		Columna que contiene la clave del usuario
estado	character(1)	Yes	No		Columna que contiene el estado del usuario si esta activo o no
nombres	character varying(50)	Yes	No		Columna que contiene el nombre del usuario
apellidos	character varying(50)	Yes	No		Columna que almacena el apellidos del usuario
telefono	character varying(50)	Yes	No		Columna que almacena el telefono del usuario
fecha_nacimiento	date	No	No		Columna que almacena la fecha de nacimiento del usuario
sexo	character varying(50)	Yes	No		Columna que almacena el sexo del usuario
correo_electronico	character varying(50)	Yes	No		Columna que almacena el correo electronico del usuario
cedula	character varying(10)	No	No		Columna que almacena el número de cédula del usuario
imagen	bytea	No	No		Columna que almacena la imagen del usuario
id_ubicacion	integer	No	No		

4. Estructura del Proyecto

La estructura del proyecto se basa en el diseño Modelo Vista Controlador MVC.

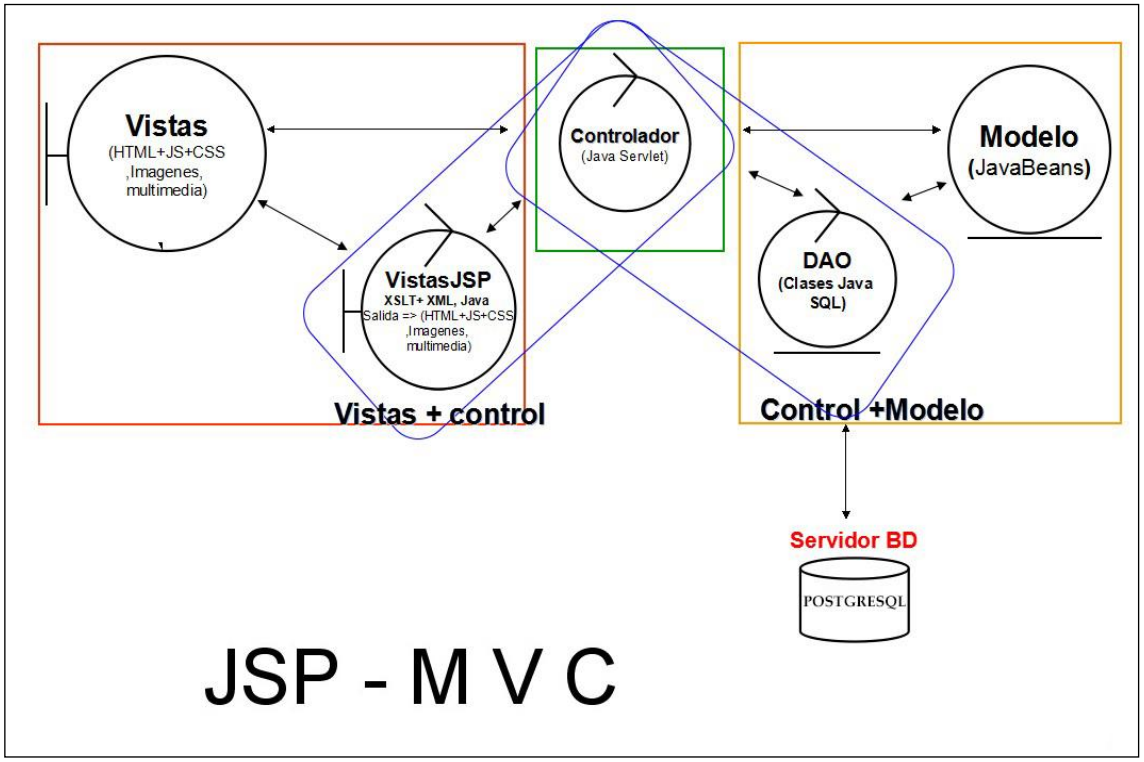


Ilustración 1 - Estructura MVC
Fuente: Propia

4.1. Modelo

4.1.1. Acceso a Datos del Objeto – Data Access Object (DAO)

- Entidad Boleto
  Esta entidad "Boleto" es un ejemplo de la estructura DAO que se implementó.

```
package encuesta.modelo;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;

import javax.persistence.*;


/**
 * The persistent class for the boletos database table.
```

```java
 *
 */
@Entity
@Table(name="boletos")
@NamedQuery(name="Boleto.findAll", query="SELECT b FROM Boleto b")
public class Boleto implements Serializable {
	private static final long serialVersionUID = 1L;
	@GeneratedValue(generator = "seq_boletos")
	@SequenceGenerator(name = "seq_boletos", sequenceName =
"seq_boletos", allocationSize = 1)

	@Id
	@Column(name="id_boleto")
	private Integer idBoleto;

	@Column(name="serial_boleto")
	private String serialBoleto;

	private BigDecimal valor;

	@Column(name="estado")
	private String estado;

	@Temporal(TemporalType.DATE)
	@Column(name = "fecha_creacion")
	private Date fechaCreacion;

	public Date getFechaCreacion() {
		return fechaCreacion;
	}

	public void setFechaCreacion(Date fechaCreacion) {
		this.fechaCreacion = fechaCreacion;
	}

	public Date getFechaImpresion() {
		return fechaImpresion;
	}

	public void setFechaImpresion(Date fechaImpresion) {
		this.fechaImpresion = fechaImpresion;
	}

	public Date getFechaActivacion() {
		return fechaActivacion;
	}

	public void setFechaActivacion(Date fechaActivacion) {
		this.fechaActivacion = fechaActivacion;
	}

	@Temporal(TemporalType.DATE)
	@Column(name = "fecha_impresion")
	private Date fechaImpresion;

	@Temporal(TemporalType.DATE)
	@Column(name = "fecha_activacion")
	private Date fechaActivacion;

	public String getEstado() {
		return estado;
```

```
	}

	public void setEstado(String estado) {
		this.estado = estado;
	}

	public Boleto() {
	}

	public Integer getIdBoleto() {
		return this.idBoleto;
	}

	public void setIdBoleto(Integer idBoleto) {
		this.idBoleto = idBoleto;
	}

	public String getSerialBoleto() {
		return this.serialBoleto;
	}

	public void setSerialBoleto(String serialBoleto) {
		this.serialBoleto = serialBoleto;
	}

	public BigDecimal getValor() {
		return this.valor;
	}

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

4.1.2. API de persistencia

Se utilizó la tecnología de persistencia Hibernate, inicialmente fue EclipseLink pero debido a problemas de compatibilidad con servidor tomcat se decidió migrar a Hibernate.

- Configuración de la persistencia:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
	xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
	<persistence-unit name="encuesta" transaction-
type="RESOURCE_LOCAL">

	<provider>org.hibernate.jpa.HibernatePersistenceProvider</provid
er>
		<class>encuesta.modelo.Boleto</class>
		<class>encuesta.modelo.Cambio</class>
		<class>encuesta.modelo.Cantones</class>
		<class>encuesta.modelo.Comisione</class>
		<class>encuesta.modelo.Encuesta</class>
		<class>encuesta.modelo.Encuestado</class>
		<class>encuesta.modelo.Monedero</class>
		<class>encuesta.modelo.Opcione</class>
		<class>encuesta.modelo.Pago</class>
		<class>encuesta.modelo.Parroquia</class>
```

```
            <class>encuesta.modelo.Pregunta</class>
            <class>encuesta.modelo.Provincia</class>
            <class>encuesta.modelo.Respuestasencuestado</class>
            <class>encuesta.modelo.ResultadosEncuesta</class>
            <class>encuesta.modelo.TipoUbicacione</class>
            <class>encuesta.modelo.TipoUsuario</class>
            <class>encuesta.modelo.TiposPregunta</class>
            <class>encuesta.modelo.Ubicacione</class>
            <class>encuesta.modelo.Usuario</class>


            <properties>
                  <property name="javax.persistence.jdbc.url"
value="jdbc:postgresql://localhost:5432/Encuesta" />
                  <property name="javax.persistence.jdbc.user"
value="postgres" />
                  <property name="javax.persistence.jdbc.password"
value="d" />
                  <property name="javax.persistence.jdbc.driver"
value="org.postgresql.Driver" />
            </properties>

<!--
            <properties> <property name="javax.persistence.jdbc.url"
            ="jdbc:postgresql://127.13.159.130:5432/encuesta"/>
<property name="javax.persistence.jdbc.user"
            ="adminxj9paxc"/> <property
name="javax.persistence.jdbc.password" value="WHXg8AkgWfT5"/>
                  <property name="javax.persistence.jdbc.driver"
value="org.postgresql.Driver"/>
                  </properties>
-->

      </persistence-unit>
</persistence>
```

Se implementan dos conexiones a la base de datos, la primera para trabajar en el entorno local, y la segunda para el ambiente de producción.

4.2. Controladores

4.2.1. Controlador “Boletos”

Se muestra a continuación la estructura del controlador utilizada para implementar la lógica de negocio. Este controlador cuenta con métodos CRUD, para gestión de objetos tipo “BOLETO”.

```
package encuesta.controlador;

import java.io.IOException;
import java.math.BigDecimal;
import java.util.Date;
import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

import encuesta.modelo.manager.ManagerDAO;
import encuesta.modelo.Boleto;
```

```
import encuesta.modelo.Monedero;
import encuesta.modelo.Usuario;

/**
 * Esta clase se utiliza para representar los valores en dólares que
pueden ser
 * utilizados para la compra de monedas encuestadoras FastPolls
 *
 * @author Gabriela Valladares
 * @version 17/04/2016
 */

@ManagedBean
@SessionScoped
public class BeanBoleto {
	private List<Boleto> listaBoleto;
	private ManagerDAO managerDAO;

	private Integer idBoleto;

	private String serialBoleto;

	private BigDecimal valor;

	private Boolean estadoexistente;

	private String estado;
mencuestadora = new BigDecimal("0.0");
valorm = new BigDecimal("0.05");

	/**
	* Constructor de la clase Boleto, también setea los valores a
null o 0
	*/
	public BeanBoleto() {
		// TODO Auto-generated constructor stub
		managerDAO = new ManagerDAO();
	();
	}

	/**
	* Obtiene la lista de todos los boletos registrados en la base
de datos
	*
	* @return Retorna la lista de boletos
	*/
	public List<Boleto> getListaBoleto() {
		listaBoleto = managerDAO.findAll(Boleto.class);
		return listaBoleto;
	}

	/**
	* Creación de un boleto
	*
	* @return Un string en blanco
	*/
	public String actionNewBoleto() {
	.out.print("Insertando");
		Boleto pr = new Boleto();
	.setIdBoleto(this.idBoleto);
	.setSerialBoleto(this.serialBoleto);
```

```
        .setValor(this.valor);
        .setEstado("A");
        fechaCreacion = new Date();
        .setFechaCreacion(fechaCreacion);
              try {
                    managerDAO.insertar(pr);
              ();

              } catch (Exception e) {
              .getCurrentInstance().addMessage(null,
                                  new
FacesMessage(FacesMessage.SEVERITY_ERROR, "Error: " + e.getMessage(),
e.getMessage()));
              }
              return "";
        }

        /**
        *
        * @param pr
        *             Este parámetro es un objeto boleto que se va a
eliminar
        * @return Un string en blanco
        */
        public String actionEliminar(Boleto pr) {
        .out.print("Eliminando");
              try {
                    managerDAO.eliminar(Boleto.class, pr.getIdBoleto());
              } catch (Exception e) {
              .getCurrentInstance().addMessage(null,
                                  new
FacesMessage(FacesMessage.SEVERITY_ERROR, "Error:" + e.getMessage(),
e.getMessage()));
              }

              return "";
        }

        /**
        * Metodo que carga las propiedas de un objeto boleto en esta
clase
        *
        * @param pr
        *             Objeto boleto que se va a cargar
        */
        public void actionCarga(Boleto pr) {
        .out.print("Cargando" + pr.getIdBoleto());
              this.idBoleto = pr.getIdBoleto();
              this.serialBoleto = pr.getSerialBoleto();
              this.valor = pr.getValor();

        }

        /**
        * Método que actualiza un boleto
        *
        * @return Un string que identifica una ruta hacia boletos
        */
        public String actionActualizar() {
        .out.print("Actualizando");
              try {
```

```
                  Boleto pr = new Boleto();
            .setIdBoleto(this.idBoleto);
            .setSerialBoleto(this.serialBoleto);
            .setValor(this.valor);

                  managerDAO.actualizar(pr);

            ();
            } catch (Exception e) {
            .getCurrentInstance().addMessage(null,
                              new
FacesMessage(FacesMessage.SEVERITY_ERROR, "Error:" + e.getMessage(),
e.getMessage()));
            }
            return "boletos";
      }

      public Integer getIdBoleto() {
            return idBoleto;
      }

      public void setIdBoleto(Integer idBoleto) {
            this.idBoleto = idBoleto;
      }

      public String getSerialBoleto() {
            return serialBoleto;
      }

      public void setSerialBoleto(String serialBoleto) {
            this.serialBoleto = serialBoleto;
      }

      public BigDecimal getValor() {
            return valor;
      }

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

      public Boolean getEstadoexistente() {
            return estadoexistente;
      }

      public void setEstadoexistente(Boolean estadoexistente) {
            this.estadoexistente = estadoexistente;
      }

      public BigDecimal getMencuestadora() {
            return mencuestadora;
      }

      public void setMencuestadora(BigDecimal mencuestadora) {
            this.mencuestadora = mencuestadora;
      }
      /**
      * Método que anula las propiedades de este bean
      */
      public void anular() {
            this.idBoleto = null;
```

```
            this.serialBoleto = null;
            this.valor = null;
      }
      /**
      * Método que comprueba si un boleto está registrado en la base
de datos
      */
      public void comprobarBoleto() {
            if (serialBoleto != null) {

            <Boleto> boletos = managerDAO.findWhere(Boleto.class,
"o.serialBoleto='" + serialBoleto + "'", null);
                  if (!boletos.isEmpty()) {

                  a = boletos.get(0).getEstado().toString();
                  .out.println("ESTADO BOLETO:" + a);
                        if (a.equals("A")) {
                              valor = boletos.get(0).getValor();
                              mencuestadora = valor.divide(valorm, 3);
                        .out.println("MONEDA TRANSFORMADA:" +
mencuestadora);
                              serialBoleto =
boletos.get(0).getSerialBoleto();
                              idBoleto = boletos.get(0).getIdBoleto();
                              estado = boletos.get(0).getEstado();
                              estadoexistente = true;
                        .out.println("BOLETO EXISTENTE");
                        } else if (a.equals("D")) {
                              JSFUtil.crearMensajeERROR("Boleto ya
Depositado");
                        .out.println("BOLETO USADO ANTERIORMENTE");
                              estadoexistente = false;
                        }
                  } else {
                        JSFUtil.crearMensajeERROR("Boleto no Existente
");
                  .out.println("BOLETO NO EXISTENTE");
                        estadoexistente = false;
                  }
            }

      }
      /**
      * Con el serial de un boleto, este método se utiliza para cargar
dinero de
      * boletos a la cuenta
      */
      public void depositarBoleto() {
      context = FacesContext.getCurrentInstance();
            Usuario usuario = (Usuario)
context.getExternalContext().getSessionMap().get("usuario");

            if (usuario != null) {
            <Monedero> monedero = managerDAO.findJPQL(
                              "Select m from Monedero m JOIN m.usuario
as u where u.idUsuario=" + usuario.getIdUsuario());
                  if (!monedero.isEmpty()) {
                        try {
                              Monedero m = (Monedero)
managerDAO.findById(Monedero.class, monedero.get(0).getIdMonedero());
```

```
                                    Boleto b = (Boleto)
managerDAO.findById(Boleto.class, idBoleto);
                              a = b.getEstado();
                              .out.println("a" + a);
                                    if (a.equals("A")) {

                                    deposito = m.getDinero();
                                    = deposito.add(mencuestadora);
                                    .setDinero(deposito);
                                    .out.println("DEPOSITADO" + deposito);
                                          managerDAO.actualizar(m);
                                    .out.println("--------------------------
" + idBoleto);

                                    .out.println(b.getEstado());
                                    .setEstado("D");
                                          managerDAO.actualizar(b);
                                    } else if (a.equals("D")) {
                                          JSFUtil.crearMensajeERROR("Boleto
ya Depositado");
                                    .out.println("BOLETO USADO
ANTERIORMENTE");
                                          estadoexistente = false;
                                    }

                              } catch (Exception e1) {
                                    // TODO Auto-generated catch block
                              .printStackTrace();
                              }

                        }
                  }
            }
}
```

4.2.2. Validadores

Utilizados para la validación de datos ingresados por usuario, en los components jsf.

Estructura de un validador utilizado para el control de las fechas:

```
package encuesta.controlador.validator;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.ConverterException;
import javax.faces.validator.FacesValidator;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;
import encuesta.controlador.BeanUsuario;

@FacesValidator(value = "validador_fecha")
public class validador_fecha implements Validator {

      @Override
```

```
	public void validate(FacesContext context, UIComponent
component, Object arg2) throws ValidatorException {

		fechaString = String.valueOf(arg2);

		fecha = null;
		fechaActual = new Date();
			long edad = 0;
			try {
			formatter = new SimpleDateFormat("dd/MM/yyyy");
			= formatter.parse(fechaString);
			.out.println("fecha:  " + fechaString + fecha.toString());
				long milisegundos = (fechaActual.getTime() -
fecha.getTime());
				long milianio = 31449600000L;
			= milisegundos / milianio;
			.out.println("Edad" + edad);

			} catch (ParseException e) {
				// TODO Auto-generated catch block
			.printStackTrace();
			}
			if (edad < 15) {
			fm = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Error:"
+ "Debe ser Mayor de 15 años", "");
				throw new ValidatorException(fm);
			}
			if (fecha.after(fechaActual)) {
			fm = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Error:"
+ "Fecha Incorrecta", "");
				throw new ValidatorException(fm);
			}

		}

}
```

4.3. Vista

4.3.1. JSFUTIL

La clase JSFUTIL es utilizada para la generación de mensajes desde los controladores hacia la Vista. Cuenta con todas las funcionalidades para la generación de mensajes.

```
package encuesta.controlador;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.context.FacesContext;

public class JSFUtil {
	/**
	* Crea un mensaje JSF
	* @param severidad Puede tomar el valor de:
	* <li>FacesMessage.SEVERITY_FATAL
	* <li>FacesMessage.SEVERITY_ERROR
	* <li>FacesMessage.SEVERITY_WARN
	* <li>FacesMessage.SEVERITY_INFO
	* @param mensaje Contenido del mensaje
	*/
```

```
	public static void crearMensaje(Severity severidad,String
mensaje){
	msg = new FacesMessage();
	.setSeverity(severidad);
	.setSummary(mensaje);
		//msg.setDetail(detalle);
	  FacesContext.getCurrentInstance().addMessage(null, msg);
	}

	public static void crearMensajeERROR(String mensaje){
		crearMensaje(FacesMessage.SEVERITY_ERROR,mensaje);
	}

	public static void crearMensajeWARN(String mensaje){
		crearMensaje(FacesMessage.SEVERITY_WARN,mensaje);
	}

	public static void crearMensajeINFO(String mensaje){
		crearMensaje(FacesMessage.SEVERITY_INFO,mensaje);
	}
}
```

4.3.2. Vista Boletos

Esta vista proporciona herramientas para la gestión de los boletos, procesa los datos del usuario y de la base de datos interactuando con el controlador.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
:ui="http://java.sun.com/jsf/facelets"
:h="http://java.sun.com/jsf/html"
:p="http://primefaces.org/ui"
:f="http://java.sun.com/jsf/core"
="/resources/adminlte/masterTemplate.xhtml"
:c="http://java.sun.com/jsp/jstl/core">


	<ui:define name="header">
		<script src="//code.jquery.com/jquery-migrate-
1.2.1.min.js"></script>
		<h:outputScript library="adminlte"

	="plugins/datatables/jquery.dataTables.min.js"></h:outputScript>
		<h:outputScript library="adminlte"

	="plugins/datatables/dataTables.bootstrap.min.js"></h:outputScri
pt>
	</ui:define>

	<ui:define name="header-panel">
	<small>CreaciÃ³n</small>
	</ui:define>

	<ui:define name="body">
		<c:if test="#{beanUsuario.permisos('admin')}" />

		<div class="box box-info">
```

```
                  <div class="box-header with-border">
                        <h3 class="box-title">Ingreso de Boletos</h3>
                  </div>
                  <!-- /.box-header -->
                  <!-- form start -->
                  <h:form class="form-horizontal" id="formInsertar">
                        <div class="box-body">
                              <div class="form-group">
                                    <label for="inputSerial"
class="col-sm-2 control-label">Serial
                                    Boleto</label>
                                    <div class="col-sm-4">
                                          <p:inputText
styleClass="form-control" required="true" id="serial"
                                          ="#{beanBoleto.serialBoleto}"
                                          ="Campo requerido" >
                                                <p:messages
for="serial"/>
                                          </p:inputText>
                                    </div>
                              </div>
                              <div class="form-group">
                                    <label class="col-sm-2 control-
label">Valor</label>
                                    <div class="col-sm-4">
                                          <p:inputText id="Valor"
styleClass="form-control"
                                          ="#{beanBoleto.valor}"
required="true"
                                          ="Campo requerido"
converterMessage="Debe ingresar un nÃºmero decimal ejemplo: 000.00">
                                                <f:convertNumber
pattern="#,###,##0.00"/>
                                                <p:messages for="Valor"
/>
                                          </p:inputText>
                                    </div>
                              </div>
                        </div>
                        <!-- /.box-body -->
                        <div class="box-footer">
                              <div class="form-group">
                                    <div class="col-sm-2">
                                          <p:commandButton
value="Insertar"
                                          ="btn btn-primary pull-right"

      ="#{beanBoleto.actionNewBoleto()}"

      ="formInsertar,:tablaParroquias">
                                          </p:commandButton>
                                    </div>
                              </div>
                        </div>
                        <!-- /.box-footer -->
                  </h:form>
            </div>


            <!-- tabla nuevo stilo
                  -->
```

```
            <h:form id="tablaParroquias">
                  <div class="box">
                        <div class="box-header">
                              <h3 class="box-title">Boletos</h3>
                        </div>
                        <!-- /.box-header -->
                        <div class="box-body">
                              <table id="example1" class="table table-
bordered table-striped">
                                    <thead>
                                          <tr>
                                                <th>CÃ³digo</th>
                                                <th>Serial</th>
                                                <th>Valor</th>
                                                <th>Estado</th>
                                                <th>CreaciÃ³n</th>
                                                <th>ImpresiÃ³n</th>
                                                <th>ActivaciÃ³n</th>
                                                <th>Eliminar</th>
                                                <th>Actualizar</th>
                                          </tr>
                                    </thead>
                                    <tbody>
                                          <c:forEach
items="#{beanBoleto.listaBoleto}" var="boleto">
                                                <tr>

      <td>#{boleto.idBoleto}</td>

      <td>#{boleto.serialBoleto}</td>

      <td>#{boleto.valor}</td>

      <td>#{boleto.estado}</td>

      <td>#{boleto.fechaCreacion}</td>

      <td>#{boleto.fechaImpresion}</td>

      <td>#{boleto.fechaActivacion}</td>

      <td><p:commandLink styleClass="btn btn-primary"

      actionListener="#{beanBoleto.actionEliminar(boleto)}"

      ="tablaParroquias">
                                                                  <i
class="fa fa-close"></i>

      </p:commandLink></td>

      <td><p:commandLink styleClass="btn btn-primary"

      ="#{beanBoleto.actionCarga(boleto)}"

      ="PF('mactualizar').show();" update=":actu">
                                                                  <i
class="fa fa-edit"></i>

      </p:commandLink></td>
                                                </tr>
```

```
                                        </c:forEach>
                                    </tbody>
                                </table>
                            </div>
                        </div>

                        <h:outputScript>function showAlert(){
                $('#example1').DataTable({
                  "paging": true,
                  "lengthChange": true,
                  "searching": true,
                  "ordering": true,
                  "info": true,
                  "autoWidth": true,
                  "bDestroy": true,
                  "language": {
                      "lengthMenu": "Mostrar _MENU_ registros por
pÃ¡gina",
                      "zeroRecords": "Lo sentimos, nada encontrado",
                      "info": "PÃ¡gina _PAGE_ de _PAGES_",
                      "infoEmpty": "No hay registros",
                      "infoFiltered": "(filtrado por _MAX_ total de
registros)",
                      "search": "Buscar ",
                      "ordering": true,
                      "paginate": {
                              "next": "Siguiente",
                              "previous": "Anterior"
                                    }
                  },
                 });
              }
              showAlert();
              </h:outputScript>
                </h:form>


                <!-- VENTANA MODAL -->
                <p:dialog id="actu" header="Modal Dialog"
widgetVar="mactualizar"
                ="true" height="500">
                      <p:panel header="ActualizaciÃ³n del Boleto">
                            <h:form>
                                  <p:panelGrid columns="2" id="panelGrid1">
                                        <h:outputText value="ID:" />
                                        <h:panelGroup>
                                              <h:outputText
value="#{beanBoleto.idBoleto}" />
                                              <h:inputHidden
value="#{beanBoleto.idBoleto}" />

                                        </h:panelGroup>
                                        <h:outputText value="SERIAL:" />
                                        <p:inputText
value="#{beanBoleto.serialBoleto}"></p:inputText>

                                        <h:outputText value="VALOR:" />
                                        <p:inputText
value="#{beanBoleto.valor}"></p:inputText>

                                  </p:panelGrid>
```

```
                                <p:commandButton id="cboleto"
value="Actualizar"
                                ="#{beanBoleto.actionActualizar()}"
                                ="PF('mactualizar').hide();"
update=":tablaParroquias"></p:commandButton>
                            </h:form>
                    </p:panel>
            </p:dialog>

            </ui:define>

</ui:composition>
```