**Persistence**

<?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="JuegoPreguntantasPU" transaction-type="RESOURCE\_LOCAL">

<provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>

<class>entity.Cuentausuario</class>

<class>entity.Cuentainvitado</class>

<class>entity.Setpregunta\_1</class>

<class>entity.Cuentausuario\_1</class>

<class>entity.Pregunta\_1</class>

<class>entity.Setpregunta</class>

<class>entity.Partida</class>

<class>entity.Pregunta</class>

<class>entity.Respuesta</class>

<class>entity.Respuesta\_1</class>

<class>entity.Categoria</class>

<properties>

<property name="javax.persistence.jdbc.url" value="jdbc:mysql://192.168.43.91:3306/juegopr?zeroDateTimeBehavior=convertToNull"/>

<property name="javax.persistence.jdbc.user" value="pregunton"/>

<property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>

<property name="javax.persistence.jdbc.password" value="PR3GUNT0N"/>

</properties>

</persistence-unit>

</persistence>

**Clase JugadorConectadoEnvio**

package clasesutilidad;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 06/11/2018 \*

\* Nombre de la clase JugadorConectadoEnvio \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class JugadorConectadoEnvio {

private String nombre;

private String socketid;

private int puntaje;

/\*\*

\* Constructor de la clase.

\*/

public JugadorConectadoEnvio() {

//Vacio para invocacion.

}

/\*\*

\* @return the nombre

\*/

public String getNombre() {

return nombre;

}

/\*\*

\* @param nombre the nombre to set

\*/

public void setNombre(String nombre) {

this.nombre = nombre;

}

/\*\*

\* @return the socketid

\*/

public String getSocketid() {

return socketid;

}

/\*\*

\* @param socketid the socketid to set

\*/

public void setSocketid(String socketid) {

this.socketid = socketid;

}

/\*\*

\* @return the puntaje

\*/

public int getPuntaje() {

return puntaje;

}

/\*\*

\* @param puntaje the puntaje to set

\*/

public void setPuntaje(int puntaje) {

this.puntaje = puntaje;

}

/\*\*

\* Este metodo sirve para conservar el puntaje de cada jugador.

\* @param puntaje

\*/

public void sumarPuntos(int puntaje) {

this.puntaje = this.puntaje + puntaje;

}

/\*\*

\* Este metodo sirve para que al hacerlo string retorne solo el nombre.

\* @return el nombre.

\*/

@Override

public String toString() {

return this.nombre;

}

}

**Clase PreguntaEnvio**

package clasesutilidad;

import java.io.Serializable;

import java.util.List;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Eduardo Rosas Rivera \*

\* @since 10/12/2018 \*

\* Nombre de la clase PreguntaEnvio \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class PreguntaEnvio implements Serializable{

private String pregunta;

private int tipoPregunta;

private String imagen;

private List<RespuestaEnvio> respuestas;

private byte[] imagenByte;

/\*\*

\* Constructor vacio de la clase.

\*/

public PreguntaEnvio() {

//Vacio para invocacion.

}

/\*\*

\* @return the pregunta

\*/

public String getPregunta() {

return pregunta;

}

/\*\*

\* @param pregunta the pregunta to set

\*/

public void setPregunta(String pregunta) {

this.pregunta = pregunta;

}

/\*\*

\* @return the tipoPregunta

\*/

public int getTipoPregunta() {

return tipoPregunta;

}

/\*\*

\* @param tipoPregunta the tipoPregunta to set

\*/

public void setTipoPregunta(int tipoPregunta) {

this.tipoPregunta = tipoPregunta;

}

/\*\*

\* @return the imagen

\*/

public String getImagen() {

return imagen;

}

/\*\*

\* @param imagen the imagen to set

\*/

public void setImagen(String imagen) {

this.imagen = imagen;

}

/\*\*

\* @return the respuestas

\*/

public List<RespuestaEnvio> getRespuestas() {

return respuestas;

}

/\*\*

\* @param respuestas the respuestas to set

\*/

public void setRespuestas(List<RespuestaEnvio> respuestas) {

this.respuestas = respuestas;

}

/\*\*

\* @return the respuestas

\*/

public byte[] getImagenByte() {

return imagenByte;

}

/\*\*

\* fija la el arreglo de bytes.

\* @param imagenByte

\*/

public void setImagenByte(byte[] imagenByte) {

this.imagenByte = imagenByte;

}

}

**Clase PuntajeEnvio**

package clasesutilidad;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 10/11/2018 \*

\* Nombre de la clase PuntajeEnvio \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class PuntajeEnvio {

private String usuario;

private int puntaje;

/\*\*

\* Contructor vacio de la clase.

\*/

public PuntajeEnvio() {

//Vacio para invocacion.

}

/\*\*

\* @return the usuario

\*/

public String getUsuario() {

return usuario;

}

/\*\*

\* @param usuario the usuario to set

\*/

public void setUsuario(String usuario) {

this.usuario = usuario;

}

/\*\*

\* @return the puntaje

\*/

public int getPuntaje() {

return puntaje;

}

/\*\*

\* @param puntaje the puntaje to set

\*/

public void setPuntaje(int puntaje) {

this.puntaje = puntaje;

}

}

**Clase RespuestaEnvio**

package clasesutilidad;

import java.io.Serializable;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Eduardo Rosas Rivera \*

\* @since 10/11/2018 \*

\* Nombre de la clase RespuestaEnvio \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class RespuestaEnvio implements Serializable{

private String respuesta;

private int tipoRespuesta;

private String imagen;

private int puntaje;

private byte[] imagenByte;

/\*\*

\* Constructor de la clase.

\*/

public RespuestaEnvio() {

//Vacio para invocacion.

}

/\*\*

\* @return the respuesta

\*/

public String getRespuesta() {

return respuesta;

}

/\*\*

\* @param respuesta the respuesta to set

\*/

public void setRespuesta(String respuesta) {

this.respuesta = respuesta;

}

/\*\*

\* @return the tipoRespuesta

\*/

public int getTipoRespuesta() {

return tipoRespuesta;

}

/\*\*

\* @param tipoRespuesta the tipoRespuesta to set

\*/

public void setTipoRespuesta(int tipoRespuesta) {

this.tipoRespuesta = tipoRespuesta;

}

/\*\*

\* @return the imagen

\*/

public String getImagen() {

return imagen;

}

/\*\*

\* @param imagen the imagen to set

\*/

public void setImagen(String imagen) {

this.imagen = imagen;

}

/\*\*

\* @return the imagenByte

\*/

public byte[] getImagenByte() {

return imagenByte;

}

/\*\*

\* @param imagenByte the imagen to set

\*/

public void setImagenByte(byte[] imagenByte) {

this.imagenByte = imagenByte;

}

/\*\*

\* @return the puntaje

\*/

public int getPuntaje() {

return puntaje;

}

/\*\*

\* @param puntaje the puntaje to set

\*/

public void setPuntaje(int puntaje) {

this.puntaje = puntaje;

}

@Override

public String toString() {

return "La respuestas es: " + respuesta + ", el puntaje es " + puntaje;

}

}

**Clase Categoria**

package entity;

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;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 26/10/2018 \*

\* Nombre de la clase Categoria \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@Entity

@Table(name = "categoria")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "Categoria.findAll", query = "SELECT c FROM Categoria c")

, @NamedQuery(name = "Categoria.findAllCategoria", query = "SELECT c.nombre FROM Categoria c")

, @NamedQuery(name = "Categoria.findCategoriaById", query = "SELECT c.nombre FROM Categoria c WHERE c.idcategoria = :idcategoria")

, @NamedQuery(name = "Categoria.findIdByCategoria", query = "SELECT c.idcategoria FROM Categoria c WHERE c.nombre = :categoria")

, @NamedQuery(name = "Categoria.findByIdcategoria", query = "SELECT c FROM Categoria c WHERE c.idcategoria = :idcategoria")

, @NamedQuery(name = "Categoria.findByCategoria", query = "SELECT c FROM Categoria c WHERE c.nombre = :categoria")})

public class Categoria implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Basic(optional = false)

@Column(name = "idcategoria")

private Integer idcategoria;

@Column(name = "categoria")

private String nombre;

public Categoria() {

//Vacio para instanciacion.

}

public Categoria(Integer idcategoria) {

this.idcategoria = idcategoria;

}

public Integer getIdcategoria() {

return idcategoria;

}

public void setIdcategoria(Integer idcategoria) {

this.idcategoria = idcategoria;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

@Override

public int hashCode() {

int hash = 0;

hash += (idcategoria != null ? idcategoria.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

if (!(object instanceof Categoria)) {

return false;

}

Categoria other = (Categoria) object;

return (this.idcategoria == null && other.idcategoria != null) || (this.idcategoria != null && !this.idcategoria.equals(other.idcategoria));

}

@Override

public String toString() {

return idcategoria.toString();

}

}

**Clase CuentaInvitado**

package entity;

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;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 26/10/2018 \*

\* Nombre de la clase CuentaInvitado \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@Entity

@Table(name = "cuentainvitado")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "Cuentainvitado.findAll", query = "SELECT c FROM Cuentainvitado c")

, @NamedQuery(name = "Cuentainvitado.findMaximo", query = "SELECT MAX(c.idcuentainvitado) FROM Cuentainvitado c")

, @NamedQuery(name = "Cuentainvitado.findAllByCorreoelectronico", query = "SELECT c.correoelectronico FROM Cuentainvitado c WHERE UPPER(c.correoelectronico) = :correoelectronico")

, @NamedQuery(name = "Cuentainvitado.findAllByNombre", query = "SELECT c.nombreusuario FROM Cuentausuario c WHERE c.nombreusuario = :nombre")

, @NamedQuery(name = "Cuentainvitado.findAllByCodigo", query = "SELECT c.codigo FROM Cuentainvitado c WHERE c.codigo = :codigo")

, @NamedQuery(name = "Cuentainvitado.findByIdcuentainvitado", query = "SELECT c FROM Cuentainvitado c WHERE c.idcuentainvitado = :idcuentainvitado")

, @NamedQuery(name = "Cuentainvitado.findByNombre", query = "SELECT c FROM Cuentainvitado c WHERE c.nombre = :nombre")

, @NamedQuery(name = "Cuentainvitado.findByCorreoelectronico", query = "SELECT c FROM Cuentainvitado c WHERE c.correoelectronico = :correoelectronico")

, @NamedQuery(name = "Cuentainvitado.findByCodigo", query = "SELECT c FROM Cuentainvitado c WHERE c.codigo = :codigo")})

public class Cuentainvitado implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Basic(optional = false)

@Column(name = "idcuentainvitado")

private Integer idcuentainvitado;

@Column(name = "nombre")

private String nombre;

@Column(name = "correoelectronico")

private String correoelectronico;

@Column(name = "codigo")

private String codigo;

public Cuentainvitado() {

//Vacio para instanciacion

}

public Cuentainvitado(Integer idcuentainvitado) {

this.idcuentainvitado = idcuentainvitado;

}

public Integer getIdcuentainvitado() {

return idcuentainvitado;

}

public void setIdcuentainvitado(Integer idcuentainvitado) {

this.idcuentainvitado = idcuentainvitado;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getCorreoelectronico() {

return correoelectronico;

}

public void setCorreoelectronico(String correoelectronico) {

this.correoelectronico = correoelectronico;

}

public String getCodigo() {

return codigo;

}

public void setCodigo(String codigo) {

this.codigo = codigo;

}

@Override

public int hashCode() {

int hash = 0;

hash += (idcuentainvitado != null ? idcuentainvitado.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

if (!(object instanceof Cuentainvitado)) {

return false;

}

Cuentainvitado other = (Cuentainvitado) object;

return (this.idcuentainvitado == null && other.idcuentainvitado != null) || (this.idcuentainvitado != null && !this.idcuentainvitado.equals(other.idcuentainvitado));

}

@Override

public String toString() {

return "entity.Cuentainvitado[ idcuentainvitado=" + idcuentainvitado + " ]";

}

}

**Clase CuentaUsuario**

package entity;

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.OneToMany;

import javax.persistence.Table;

import javax.xml.bind.annotation.XmlRootElement;

import javax.xml.bind.annotation.XmlTransient;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 26/10/2018 \*

\* Nombre de la clase Cuentausuario \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@Entity

@Table(name = "cuentausuario")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "Cuentausuario.findAll", query = "SELECT c FROM Cuentausuario c")

, @NamedQuery(name = "Cuentausuario.findByIdcuentausuario", query = "SELECT c FROM Cuentausuario c WHERE c.idcuentausuario = :idcuentausuario")

, @NamedQuery(name = "Cuentausuario.findByContrasenia", query = "SELECT c FROM Cuentausuario c WHERE c.contrasenia = :contrasenia")

, @NamedQuery(name = "Cuentausuario.findByNombreusuario", query = "SELECT c FROM Cuentausuario c WHERE c.nombreusuario = :nombreusuario")

, @NamedQuery(name = "Cuentausuario.findByCorreoelectronico", query = "SELECT c FROM Cuentausuario c WHERE c.correoelectronico = :correoelectronico")})

public class Cuentausuario implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Basic(optional = false)

@Column(name = "idcuentausuario")

private Integer idcuentausuario;

@Column(name = "contrasenia")

private String contrasenia;

@Column(name = "nombreusuario")

private String nombreusuario;

@Column(name = "correoelectronico")

private String correoelectronico;

@OneToMany(mappedBy = "idcuentausuario")

private Collection<Setpregunta> setpreguntaCollection;

public Cuentausuario() {

//Vacio para instanciacion.

}

public Cuentausuario(Integer idcuentausuario) {

this.idcuentausuario = idcuentausuario;

}

public Integer getIdcuentausuario() {

return idcuentausuario;

}

public void setIdcuentausuario(Integer idcuentausuario) {

this.idcuentausuario = idcuentausuario;

}

public String getContrasenia() {

return contrasenia;

}

public void setContrasenia(String contrasenia) {

this.contrasenia = contrasenia;

}

public String getNombreusuario() {

return nombreusuario;

}

public void setNombreusuario(String nombreusuario) {

this.nombreusuario = nombreusuario;

}

public String getCorreoelectronico() {

return correoelectronico;

}

public void setCorreoelectronico(String correoelectronico) {

this.correoelectronico = correoelectronico;

}

@XmlTransient

public Collection<Setpregunta> getSetpreguntaCollection() {

return setpreguntaCollection;

}

public void setSetpreguntaCollection(Collection<Setpregunta> setpreguntaCollection) {

this.setpreguntaCollection = setpreguntaCollection;

}

@Override

public int hashCode() {

int hash = 0;

hash += (idcuentausuario != null ? idcuentausuario.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

if (!(object instanceof Cuentausuario)) {

return false;

}

Cuentausuario other = (Cuentausuario) object;

return ((this.idcuentausuario == null && other.idcuentausuario != null) || (this.idcuentausuario != null && !this.idcuentausuario.equals(other.idcuentausuario)));

}

@Override

public String toString() {

return "entity.Cuentausuario[ idcuentausuario=" + idcuentausuario + " ]";

}

}

**Clase Partida**

package entity;

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.persistence.Transient;

import javax.xml.bind.annotation.XmlRootElement;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 26/10/2018 \*

\* Nombre de la clase Partida \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@Entity

@Table(name = "partida")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "Partida.findAll", query = "SELECT p FROM Partida p")

, @NamedQuery(name = "Partida.findBySet", query = "SELECT p.nombre FROM Partida p WHERE p.idsetpregunta.idsetpregunta = :idsetpregunta")

, @NamedQuery(name = "Partida.findByIdset", query = "SELECT p FROM Partida p WHERE p.idsetpregunta.idsetpregunta = :idsetpregunta")

, @NamedQuery(name = "Partida.findByIdpartida", query = "SELECT p FROM Partida p WHERE p.idpartida = :idpartida")

, @NamedQuery(name = "Partida.findByNombre", query = "SELECT p FROM Partida p WHERE p.nombre = :nombre")})

public class Partida implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Basic(optional = false)

@Column(name = "idpartida")

private Integer idpartida;

@Column(name = "nombre")

private String nombre;

@JoinColumn(name = "idsetpregunta", referencedColumnName = "idsetpregunta")

@ManyToOne

private Setpregunta idsetpregunta;

@Transient

private int totalPreguntas;

public Partida() {

//Vacio para instanciacion.

}

public Partida(Integer idpartida) {

this.idpartida = idpartida;

}

public Integer getIdpartida() {

return idpartida;

}

public void setIdpartida(Integer idpartida) {

this.idpartida = idpartida;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public Setpregunta getIdsetpregunta() {

return idsetpregunta;

}

public void setIdsetpregunta(Setpregunta idsetpregunta) {

this.idsetpregunta = idsetpregunta;

}

@Override

public int hashCode() {

int hash = 0;

hash += (idpartida != null ? idpartida.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

if (!(object instanceof Partida)) {

return false;

}

Partida other = (Partida) object;

return ((this.idpartida == null && other.idpartida != null) || (this.idpartida != null && !this.idpartida.equals(other.idpartida)));

}

@Override

public String toString() {

return nombre;

}

/\*\*

\* Este metodo fija el total de preguntas con las que cuenta una partida.

\* @param totalPreguntas total de preguntas que existen en la partida.

\*/

public void setTotalPreguntas(int totalPreguntas) {

this.totalPreguntas = totalPreguntas;

}

public int getTotalPreguntas() {

return totalPreguntas;

}

}

**Clase Pregunta**

package entity;

import java.io.IOException;

import java.io.Serializable;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Collection;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

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.OneToMany;

import javax.persistence.Table;

import javax.persistence.Transient;

import javax.xml.bind.annotation.XmlRootElement;

import javax.xml.bind.annotation.XmlTransient;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 26/10/2018 \*

\* Nombre de la clase Pregunta \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@Entity

@Table(name = "pregunta")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "Pregunta.findAll", query = "SELECT p FROM Pregunta p")

, @NamedQuery(name = "Pregunta.findByIdpregunta", query = "SELECT p FROM Pregunta p WHERE p.idpregunta = :idpregunta")

, @NamedQuery(name = "Pregunta.findByPregunta", query = "SELECT p FROM Pregunta p WHERE p.preguntaContenido = :pregunta")

, @NamedQuery(name = "Pregunta.findByIdsetpregunta", query = "SELECT p FROM Pregunta p WHERE p.idsetpregunta = :idsetpregunta")})

public class Pregunta implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Basic(optional = false)

@Column(name = "idpregunta")

private Integer idpregunta;

@Column(name = "pregunta")

private String preguntaContenido;

@Column(name = "tipoPregunta")

private Integer tipoPregunta;

@Column(name = "idsetpregunta")

private Integer idsetpregunta;

@OneToMany(mappedBy = "idpregunta")

private Collection<Respuesta> respuestaCollection;

@Transient

private byte[] imagen;

@Transient

private List<Respuesta> respuestas;

public Pregunta() {

//Vacio para instanciacion.

}

public Pregunta(Integer idpregunta) {

this.idpregunta = idpregunta;

}

public Integer getIdpregunta() {

return idpregunta;

}

public void setIdpregunta(Integer idpregunta) {

this.idpregunta = idpregunta;

}

public String getPregunta() {

return preguntaContenido;

}

public void setPregunta(String pregunta) {

this.preguntaContenido = pregunta;

}

public Integer getTipoPregunta() {

return tipoPregunta;

}

public void setTipoPregunta(Integer tipoPregunta) {

this.tipoPregunta = tipoPregunta;

}

public Integer getIdsetpregunta() {

return idsetpregunta;

}

public void setIdsetpregunta(Integer idsetpregunta) {

this.idsetpregunta = idsetpregunta;

}

@XmlTransient

public Collection<Respuesta> getRespuestaCollection() {

return respuestaCollection;

}

public void setRespuestaCollection(Collection<Respuesta> respuestaCollection) {

this.respuestaCollection = respuestaCollection;

}

public void setRespuestas(List<Respuesta> respuestas) {

this.respuestas = respuestas;

}

public List<Respuesta> getRespuestas() {

return respuestas;

}

public void setRespuesta(Respuesta respuesta) {

this.respuestas.add(respuesta);

}

public Respuesta getRespuesta(int indice) {

return respuestas.get(indice);

}

@Override

public int hashCode() {

int hash = 0;

hash += (idpregunta != null ? idpregunta.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

if (!(object instanceof Pregunta)) {

return false;

}

Pregunta other = (Pregunta) object;

return ((this.idpregunta == null && other.idpregunta != null) || (this.idpregunta != null && !this.idpregunta.equals(other.idpregunta)));

}

@Override

public String toString() {

return "entity.Pregunta[ idpregunta=" + idpregunta + " ]";

}

/\*\*

\* @return the imagen

\*/

public byte[] getImagen() {

return imagen;

}

/\*\*

\* @param imagen the imagen to set

\*/

public void setImagen(byte[] imagen) {

this.imagen = imagen;

}

/\*\*

\* Este metodo crea un arreglo de bytes para poder ser enviado por nodeJs

\* @return imagen convertida en un arreglo de bytes.

\*/

public byte[] crearArregloImagen() {

byte[] imagenPregunta = null;

try {

Path path = Paths.get(preguntaContenido);

byte[] contenido = Files.readAllBytes(path);

imagen = contenido;

} catch (IOException ex) {

Logger.getLogger(Pregunta.class.getName())

.log(Level.SEVERE, null, ex);

}

return imagenPregunta;

}

}

**Clase Respuesta**

package entity;

import java.io.IOException;

import java.io.Serializable;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Objects;

import java.util.logging.Level;

import java.util.logging.Logger;

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.persistence.Transient;

import javax.xml.bind.annotation.XmlRootElement;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 26/10/2018 \*

\* Nombre de la clase Respuesta \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@Entity

@Table(name = "respuesta")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "Respuesta.findAll", query = "SELECT r FROM Respuesta r")

, @NamedQuery(name = "Respuesta.findByIdpregunta", query = "SELECT r FROM Respuesta r WHERE r.idpregunta.idpregunta = :idpregunta")

, @NamedQuery(name = "Respuesta.findByIdrespuesta", query = "SELECT r FROM Respuesta r WHERE r.idrespuesta = :idrespuesta")

, @NamedQuery(name = "Respuesta.findByRespuesta", query = "SELECT r FROM Respuesta r WHERE r.respuestaContenido = :respuesta")

, @NamedQuery(name = "Respuesta.findByPuntaje", query = "SELECT r FROM Respuesta r WHERE r.puntaje = :puntaje")

, @NamedQuery(name = "Respuesta.findByTipoRespuesta", query = "SELECT r FROM Respuesta r WHERE r.tipoRespuesta = :tipoRespuesta")})

public class Respuesta implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Basic(optional = false)

@Column(name = "idrespuesta")

private Integer idrespuesta;

@Column(name = "respuesta")

private String respuestaContenido;

@Column(name = "puntaje")

private Integer puntaje;

@Column(name = "tipoRespuesta")

private Integer tipoRespuesta;

@JoinColumn(name = "idpregunta", referencedColumnName = "idpregunta")

@ManyToOne

private Pregunta idpregunta;

@Transient

private byte[] imagen;

public Respuesta() {

//Vacio para instanciacion.

}

public Respuesta(Integer idrespuesta) {

this.idrespuesta = idrespuesta;

}

public Integer getIdrespuesta() {

return idrespuesta;

}

public void setIdrespuesta(Integer idrespuesta) {

this.idrespuesta = idrespuesta;

}

public String getRespuesta() {

return respuestaContenido;

}

public void setRespuesta(String respuesta) {

this.respuestaContenido = respuesta;

}

public Integer getPuntaje() {

if(Objects.equals(puntaje, null)) {

this.puntaje = 0;

}

return puntaje;

}

public void setPuntaje(Integer puntaje) {

this.puntaje = puntaje;

}

public Integer getTipoRespuesta() {

return tipoRespuesta;

}

public void setTipoRespuesta(Integer tipoRespuesta) {

this.tipoRespuesta = tipoRespuesta;

}

public Pregunta getIdpregunta() {

return idpregunta;

}

public void setIdpregunta(Pregunta idpregunta) {

this.idpregunta = idpregunta;

}

@Override

public int hashCode() {

int hash = 0;

hash += (idrespuesta != null ? idrespuesta.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

if (!(object instanceof Respuesta)) {

return false;

}

Respuesta other = (Respuesta) object;

return ((this.idrespuesta == null && other.idrespuesta != null) || (this.idrespuesta != null && !this.idrespuesta.equals(other.idrespuesta)));

}

@Override

public String toString() {

return "entity.Respuesta[ idrespuesta=" + idrespuesta + " ]";

}

/\*\*

\* @return the imagen

\*/

public byte[] getImagen() {

return imagen;

}

/\*\*

\* @param imagen the imagen to set

\*/

public void setImagen(byte[] imagen) {

this.imagen = imagen;

}

/\*\*

\* Este metodo crea un arreglo de bytes para poder ser enviado por nodeJs

\* @return imagen convertida en un arreglo de bytes.

\*/

public byte[] crearArregloImagen() {

byte[] imagenBytes = null;

try {

Path path = Paths.get(respuestaContenido);

byte[] contenido = Files.readAllBytes(path);

imagen = contenido;

} catch (IOException ex) {

Logger.getLogger(Respuesta.class.getName())

.log(Level.SEVERE, null, ex);

}

return imagenBytes;

}

}

**Clase SetPregunta**

package entity;

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.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import javax.xml.bind.annotation.XmlRootElement;

import javax.xml.bind.annotation.XmlTransient;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 26/10/2018 \*

\* Nombre de la clase Setpregunta \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@Entity

@Table(name = "setpregunta")

@XmlRootElement

@NamedQueries({

@NamedQuery(name = "Setpregunta.findAll", query = "SELECT s FROM Setpregunta s")

, @NamedQuery(name = "Setpregunta.findIdByIdCuentaUsuario", query = "SELECT s.idcategoria FROM Setpregunta s WHERE s.idcuentausuario.idcuentausuario = :idcuentausuario")

, @NamedQuery(name = "Setpregunta.findAllByIdCuentaUsuario", query = "SELECT s FROM Setpregunta s WHERE s.idcuentausuario.idcuentausuario = :idcuentausuario")

, @NamedQuery(name = "Setpregunta.findAllByIdsetpregunta", query = "SELECT s FROM Setpregunta s WHERE s.idsetpregunta = :idsetpregunta")

, @NamedQuery(name = "Setpregunta.findAllByIdcategoria", query = "SELECT s FROM Setpregunta s WHERE s.idcategoria = :idcategoria")

, @NamedQuery(name = "Setpregunta.findMaxIdsetpregunta", query = "SELECT MAX(s.idsetpregunta) FROM Setpregunta s")

, @NamedQuery(name = "Setpregunta.findByIdsetpregunta", query = "SELECT s FROM Setpregunta s WHERE s.idsetpregunta = :idsetpregunta")

, @NamedQuery(name = "Setpregunta.findByIdcategoria", query = "SELECT s FROM Setpregunta s WHERE s.idcategoria = :idcategoria")})

public class Setpregunta implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Basic(optional = false)

@Column(name = "idsetpregunta")

private Integer idsetpregunta;

@Column(name = "idcategoria")

private Integer idcategoria;

@JoinColumn(name = "idcuentausuario", referencedColumnName = "idcuentausuario")

@ManyToOne

private Cuentausuario idcuentausuario;

@OneToMany(mappedBy = "idsetpregunta")

private Collection<Partida> partidaCollection;

public Setpregunta() {

//Vacio para instanciacion.

}

public Setpregunta(Integer idsetpregunta) {

this.idsetpregunta = idsetpregunta;

}

public Integer getIdsetpregunta() {

return idsetpregunta;

}

public void setIdsetpregunta(Integer idsetpregunta) {

this.idsetpregunta = idsetpregunta;

}

public Integer getIdcategoria() {

return idcategoria;

}

public void setIdcategoria(Integer idcategoria) {

this.idcategoria = idcategoria;

}

public Cuentausuario getIdcuentausuario() {

return idcuentausuario;

}

public void setIdcuentausuario(Cuentausuario idcuentausuario) {

this.idcuentausuario = idcuentausuario;

}

@XmlTransient

public Collection<Partida> getPartidaCollection() {

return partidaCollection;

}

public void setPartidaCollection(Collection<Partida> partidaCollection) {

this.partidaCollection = partidaCollection;

}

@Override

public int hashCode() {

int hash = 0;

hash += (idsetpregunta != null ? idsetpregunta.hashCode() : 0);

return hash;

}

@Override

public boolean equals(Object object) {

if (!(object instanceof Setpregunta)) {

return false;

}

Setpregunta other = (Setpregunta) object;

return ((this.idsetpregunta == null && other.idsetpregunta != null) || (this.idsetpregunta != null && !this.idsetpregunta.equals(other.idsetpregunta)));

}

@Override

public String toString() {

return "entity.Setpregunta[ idsetpregunta=" + idsetpregunta + " ]";

}

}

**Enviar Invitacion.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.paint.\*?>

<?import javafx.scene.text.\*?>

<?import javafx.geometry.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.EnviarInvitacionController">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="351.0" minWidth="10.0" prefWidth="147.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="493.0" minWidth="10.0" prefWidth="453.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<Label alignment="CENTER" contentDisplay="CENTER" prefHeight="17.0" prefWidth="619.0" text="%lbl.enviar\_invitacion" textAlignment="CENTER" textFill="WHITE" GridPane.columnSpan="2">

<font>

<Font size="25.0" />

</font>

</Label>

<Label alignment="CENTER" prefHeight="17.0" prefWidth="632.0" text="%lbl.instruccion\_invitacion" textFill="WHITE" GridPane.columnSpan="2" GridPane.halignment="CENTER" GridPane.rowIndex="1" GridPane.valignment="TOP">

<GridPane.margin>

<Insets />

</GridPane.margin>

</Label>

<Label text="%lbl.correo\_electronico" textFill="WHITE" GridPane.halignment="LEFT" GridPane.rowIndex="2" GridPane.valignment="TOP">

<GridPane.margin>

<Insets left="20.0" />

</GridPane.margin>

</Label>

<TextField fx:id="txtCorreoElectronico" GridPane.columnIndex="1" GridPane.rowIndex="2" GridPane.valignment="TOP">

<GridPane.margin>

<Insets right="20.0" />

</GridPane.margin>

</TextField>

<GridPane GridPane.columnIndex="1" GridPane.rowIndex="3">

<children>

<Button fx:id="btnCancelar" mnemonicParsing="false" onAction="#cancelar" style="-fx-background-color: #F67280;" text="%btn.cancelar" GridPane.halignment="RIGHT" />

<Button fx:id="btnEnviarInvitacion" mnemonicParsing="false" onAction="#enviarInvitacion" style="-fx-background-color: #F8B195;" text="%btn.enviar\_invitacion" GridPane.columnIndex="2" GridPane.halignment="RIGHT" />

</children>

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="341.0" minWidth="10.0" prefWidth="290.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="146.0" minWidth="0.0" prefWidth="19.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="139.0" minWidth="10.0" prefWidth="123.0" />

</columnConstraints>

<padding>

<Insets right="20.0" />

</padding>

<rowConstraints>

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

</rowConstraints>

</GridPane>

</children>

</GridPane>

**Clase EnviarInvitacionController**

package juegopreguntantas;

import entity.Cuentainvitado;

import entity.Cuentausuario;

import java.io.IOException;

import java.net.URL;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import javafx.stage.Stage;

import java.util.Date;

import java.util.Locale;

import java.util.Properties;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javax.mail.Authenticator;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

import persistencia.PersistenciaCuentaInvitado;

import utilidades.UtilidadCadenas;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez \*

\* @since 7/11/2018 \*

\* Nombre de la clase EnviarInvitacionController \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class EnviarInvitacionController implements Initializable {

@FXML

private TextField txtCorreoElectronico;

@FXML

private Button btnCancelar;

@FXML

private Button btnEnviarInvitacion;

private Cuentausuario cuenta;

private String idioma;

@Override

public void initialize(URL url, ResourceBundle rb) {

UtilidadCadenas cadena = new UtilidadCadenas();

cadena.excluirEspacios(txtCorreoElectronico);

cadena.limitarCampos(txtCorreoElectronico, 35);

}

/\*\*

\* Este metodo es para cancelar el inicio de la partida y regresar a la

\* ventana anterior de Inicio de partida

\* @param event del click del mouse

\*/

@FXML

private void cancelar(ActionEvent event) {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle("juegopreguntantas.lang/lang");

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("MenuPrincipal.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

MenuPrincipalController controller = loader.getController();

controller.recibirParametros(cuenta, idioma);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Menu principal");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(EnviarInvitacionController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo es para enviar una invitacion por correo electronico para

\* unirse a una partida, cuando es exitoso envía un mensaje de confirmacion

\* @param event del click del mouse

\*/

@FXML

private void enviarInvitacion(ActionEvent event) throws MessagingException {

if (!txtCorreoElectronico.getText().isEmpty()) {

try {

PersistenciaCuentaInvitado invitadoBD = new PersistenciaCuentaInvitado();

if (invitadoBD.comprobarCorreo(txtCorreoElectronico.getText())) {

mostrarCorreoRepetido();

} else {

Cuentainvitado nuevoInvitado = new Cuentainvitado();

nuevoInvitado.setNombre(invitadoBD.crearNombre());

nuevoInvitado.setCorreoelectronico(txtCorreoElectronico.getText());

nuevoInvitado.setCodigo(invitadoBD.crearCodigo());

String deCorreo = "juego.preguntantas@gmail.com";

final String contrasenia = "pr3gunt0n";

Properties properties = crearProperties();

Authenticator auth = new Authenticator() {

@Override

public PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(deCorreo

, contrasenia);

}

};

Session sesion = Session.getInstance(properties, auth);

Message mensaje = crearContenidoInvitacion(sesion

, nuevoInvitado);

mostrarInvitadoExito(mensaje, nuevoInvitado);

volverAlMenu();

}

} finally {

txtCorreoElectronico.clear();

}

} else {

mostrarCorreoVacio();

}

}

/\*\*

\* Este metodo es para mostrar una ventana de error si intenta el usuario

\* volver a invitar al mismo usuario

\*/

private void mostrarCorreoRepetido() {

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle("Error");

alert.setHeaderText(null);

alert.setContentText("Ya se había invitado a este correo");

alert.showAndWait();

}

/\*\*

\* Este metodo es para hacer todos los put que necesitan las properties

\* @return El el properties para la sesion

\*/

private Properties crearProperties() {

Properties properties = new Properties();

String host = "smtp.gmail.com";

String puerto = "587";

properties.put("mail.smtp.host", host);

properties.put("mail.smtp.starttls.enable", "true");

properties.put("mail.smtp.ssl.trust", "smtp.gmail.com");

properties.put("mail.smtp.auth", "true");

properties.put("mail.smtp.port", puerto);

properties.put("mail.smtp.debug", "true");

return properties;

}

/\*\*

\* Este metodo es para crear el mensaje que se va a enviar por correo

\* @param sesion Session para enviar mensaje

\* @param nuevoInvitado Cuenta de invitado con los datos para el mensaje

\* @return El mensaje que se enviara por correo

\*/

private Message crearContenidoInvitacion(Session sesion,

Cuentainvitado nuevoInvitado) {

Message mensaje = new MimeMessage(sesion);

try {

InternetAddress[] address

= {new InternetAddress(nuevoInvitado.getCorreoelectronico())};

mensaje.setRecipients(Message.RecipientType.TO, address);

mensaje.setSubject("Invitacion a jugar Preguntantas");

String saludo = "Hola futuro pregunton\n\n";

String cuerpo = "Haz sido invitado a jugar Preguntantas\n\n";

String despedida = "¡Ingresa con usuario: "

+ nuevoInvitado.getNombre() + " y contraseña "

+ nuevoInvitado.getCodigo() + " !";

String contenidoCorreo = saludo + cuerpo + despedida;

mensaje.setSentDate(new Date());

mensaje.setText(contenidoCorreo);

} catch (MessagingException ex) {

Logger.getLogger(EnviarInvitacionController.class.getName())

.log(Level.SEVERE, null, ex);

}

return mensaje;

}

/\*\*

\* Este metodo sirve para mostrar una ventana en caso de intentar enviar una

\* invitación sin introducir un correo

\*/

private void mostrarCorreoVacio() {

Alert alert = new Alert(AlertType.WARNING);

alert.setTitle("Revisa");

alert.setHeaderText(null);

alert.setContentText("Introduce un correo primero");

alert.showAndWait();

}

/\*\*

\* Este metodo es para regresar al menú principal una vez enviada una

\* invitación

\*/

public void volverAlMenu(){

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle("juegopreguntantas.lang/lang");

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("MenuPrincipal.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

MenuPrincipalController controller = loader.getController();

controller.recibirParametros(cuenta, idioma);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Menu principal");

stage.setScene(scene);

stage.show();

btnEnviarInvitacion.getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(EnviarInvitacionController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo es para mostrar una ventana en caso de exito

\* @param mensaje Message que se va a enviar por correo

\* @param nuevoInvitado Cuenta de invitado que se guardara en BD

\*/

private void mostrarInvitadoExito(Message mensaje, Cuentainvitado nuevoInvitado) {

PersistenciaCuentaInvitado invitadoBD = new PersistenciaCuentaInvitado();

if (invitadoBD.crearInvitado(nuevoInvitado)) {

try {

Transport.send(mensaje);

Alert alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Invitacion exitosa");

alert.setHeaderText(null);

alert.setContentText("Invitación enviada");

alert.showAndWait();

} catch (MessagingException ex) {

Logger.getLogger(EnviarInvitacionController.class.getName())

.log(Level.SEVERE, null, ex);

mostrarInvitadoFracaso();

invitadoBD.eliminarInvitado(nuevoInvitado);

}

} else {

mostrarInvitadoFracaso();

}

}

/\*\*

\* Este metodo es para mostrar una ventana en caso de fracaso

\*/

private void mostrarInvitadoFracaso() {

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle("Error");

alert.setHeaderText(null);

alert.setContentText("No se pudo enviar la invitacion, checa el correo "

+ "introducido o la conexion y vuelve a intentar");

alert.showAndWait();

}

/\*\*

\* Metodo que recibe el objeto de cuenta de usuario o invitado del

\* Controlador de la pantalla que la invocó

\* @param usuario Cuenta de usuario registrado

\* @param idioma Idioma del properties

\*/

public void recibirParametros(Object usuario, String idioma){

this.idioma = idioma;

this.cuenta = (Cuentausuario)usuario;

}

}

**EnvioDePreguntas.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.EnvioDePreguntasController">

<children>

<ProgressBar fx:id="pbProcesoEnvio" layoutX="85.0" layoutY="200.0" prefHeight="18.0" prefWidth="431.0" style="-fx-background-color: #F8B195;" />

<Label layoutX="193.0" layoutY="87.0" text="%lbl\_enviandoDatos" textFill="WHITE">

<font>

<Font size="24.0" />

</font>

</Label>

<Button fx:id="btnIniciar" disable="true" layoutX="274.0" layoutY="308.0" mnemonicParsing="false" onAction="#iniciar" style="-fx-background-color: #F67280;" text="%btn\_iniciar" visible="false" />

</children>

</AnchorPane>

**Clase EnvioDePreguntas**

package juegopreguntantas;

import clasesutilidad.PreguntaEnvio;

import clasesutilidad.RespuestaEnvio;

import com.google.gson.Gson;

import entity.Cuentausuario;

import entity.Pregunta;

import entity.Respuesta;

import io.socket.client.IO;

import io.socket.client.Socket;

import io.socket.emitter.Emitter;

import java.io.IOException;

import java.net.URISyntaxException;

import java.net.URL;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.List;

import java.util.Locale;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ProgressBar;

import javafx.stage.Stage;

import persistencia.PersistenciaPregunta;

import javax.xml.bind.DatatypeConverter;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Eduardo Rosas Rivera \*

\* @since 12/12/2018 \*

\* Nombre de la clase EnvioDePreguntasController \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class EnvioDePreguntasController implements Initializable {

@FXML

private ProgressBar pbProcesoEnvio;

@FXML

private Button btnIniciar;

private final Socket socket;

private Cuentausuario usuario;

private String idioma;

private List<Pregunta> preguntas;

/\*\*

\* Initializes the controller class.

\* @param url

\* @param rb

\*/

@Override

public void initialize(URL url, ResourceBundle rb) {

//No es necesario hacer algo

}

/\*\*

\* Constructor de la clase.

\* @throws URISyntaxException

\*/

public EnvioDePreguntasController() throws URISyntaxException {

ResourceBundle propiedadesCliente = ResourceBundle.getBundle(

"utilidades.conexiones");

String ipServidor = propiedadesCliente.getString(

"key.ipServidor");

socket = IO.socket("http://" + ipServidor + ":4000");

socket.on(Socket.EVENT\_CONNECT, new Emitter.Listener() {

@Override

public void call(Object... os) {

//No es necesario hacer algo.

}

});

socket.connect();

}

/\*\*

\* Este metodo envia las preguntas de la partida a todos los jugadores.

\* @param event Clic en el boton iniciar.

\*/

@FXML

private void iniciar(ActionEvent event) {

abrirVentanaResponderPregunta();

socket.emit("iniciarPartida");

socket.disconnect();

btnIniciar.getScene().getWindow().hide();

}

/\*\*

\* Este metodo recibe informacion de otras ventanas que instancian a esta.

\* @param usuario Cuentausuario que inicio la partida.

\* @param idioma idioma en que se abrira la ventana.

\* @param idSetPregunta id del set de preguntas a recuperar.

\*/

public void recibirParametros(Cuentausuario usuario, String idioma,

int idSetPregunta) {

this.usuario = usuario;

this.idioma = idioma;

recuperarPreguntas(idSetPregunta);

enviarPreguntas();

}

/\*\*

\* Este metodo recupera todas la preguntas de un set de preguntas.

\* @param idSetPregunta id del set de preguntas.

\*/

private void recuperarPreguntas(int idSetPregunta) {

PersistenciaPregunta recuperadorPreguntas = new PersistenciaPregunta();

List<Pregunta> preguntasDelSet =

recuperadorPreguntas.recuperarPreguntaConRespuestas(

idSetPregunta);

for(int i = 0; i < preguntasDelSet.size(); i++) {

if(preguntasDelSet.get(i).getTipoPregunta() == 2) {

preguntasDelSet.get(i).crearArregloImagen();

}

for(int j = 0; j < 4; j++) {

if(preguntasDelSet.get(i).getRespuesta(j).getTipoRespuesta() == 2){

preguntasDelSet.get(i).getRespuesta(j).crearArregloImagen();

}

}

}

preguntas = preguntasDelSet;

}

/\*\*

\* Este metodo sirve para enviar una pregunta a los jugadores.

\*

\* @param pregunta pregunta a enviar.

\*/

public void enviarPreguntaCompleta(Pregunta pregunta) {

Gson preguntaGSON = new Gson();

PreguntaEnvio preguntaAEnviar = convertirPregunta(pregunta);

List<RespuestaEnvio> respuestasDePregunta

= convertirRespuesta(pregunta.getRespuestas());

preguntaAEnviar.setRespuestas(respuestasDePregunta);

String out = preguntaGSON.toJson(preguntaAEnviar);

socket.emit("envioPregunta", out);

}

/\*\*

\* Este metodo convierte la pregunta que va a ser enviada a un objeto de

\* tipo JSONObject.

\*

\* @param pregunta Pregunta a ser enviada.

\* @return Pregunta convertida en tipo JSONObject.

\*/

private PreguntaEnvio convertirPregunta(Pregunta pregunta) {

PreguntaEnvio pregEnvio = new PreguntaEnvio();

pregEnvio.setPregunta(pregunta.getPregunta());

pregEnvio.setTipoPregunta(pregunta.getTipoPregunta());

if (pregunta.getTipoPregunta() == 2) {

byte[] imagen = crearArregloImagen(pregunta.getPregunta());

String imagenCodificada = DatatypeConverter.printBase64Binary(imagen);

pregEnvio.setImagen(imagenCodificada);

} else {

pregEnvio.setImagen(null);

}

return pregEnvio;

}

/\*\*

\* Este metodo convierte una lista de Respuestas a un arreglo de tipo

\* JSONArray.

\* @param respuestas lista de respuestas a enviar.

\* @return Arreglo de respuestas de tipoJSON.

\*/

private ArrayList<RespuestaEnvio> convertirRespuesta(List<Respuesta>

respuestas) {

ArrayList<RespuestaEnvio> respuestasAEnviar = new ArrayList<>();

for(int i = 0; i < respuestas.size(); i++) {

RespuestaEnvio nuevaRespuestaEnvio1 = new RespuestaEnvio();

nuevaRespuestaEnvio1.setRespuesta(respuestas.get(i).getRespuesta());

nuevaRespuestaEnvio1.setPuntaje(respuestas.get(i).getPuntaje());

nuevaRespuestaEnvio1.setTipoRespuesta(respuestas.get(i)

.getTipoRespuesta());

if (respuestas.get(i).getTipoRespuesta() == 2) {

byte[] imagen =

crearArregloImagen(respuestas.get(i).getRespuesta());

String imagenCodificada =

DatatypeConverter.printBase64Binary(imagen);

nuevaRespuestaEnvio1.setImagen(imagenCodificada);

}

respuestasAEnviar.add(nuevaRespuestaEnvio1);

}

return respuestasAEnviar;

}

/\*\*

\* Este metodo crea un arreglo de bytes para poder ser enviado por nodeJs

\* @param ruta ruta del archivo.

\* @return imagen convertida en un arreglo de bytes.

\*/

public byte[] crearArregloImagen(String ruta) {

byte[] imagen = null;

try {

Path path = Paths.get(ruta);

byte[] contenido = Files.readAllBytes(path);

imagen = contenido;

} catch (IOException ex) {

Logger.getLogger(EnvioDePreguntasController.class.getName())

.log(Level.SEVERE, null, ex);

}

return imagen;

}

/\*\*

\* Este metodo sirve para abrir la ventana Responder Pregunta.

\*/

private void abrirVentanaResponderPregunta() {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle

= ResourceBundle.getBundle("juegopreguntantas.lang/lang");

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource(

"ResponderPregunta.fxml"));

loader.setResources(resourceBundle);

Parent responder = loader.load();

ResponderPreguntaController controller = loader.getController();

controller.recibirParametros(usuario, idioma, true);

Scene scene = new Scene(responder);

Stage stage = new Stage();

stage.setScene(scene);

stage.show();

} catch (Exception ex) {

Logger.getLogger(EnvioDePreguntasController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Este metodo envia todas las preguntas por nodejs.

\*/

private void enviarPreguntas() {

int proceso = 0;

for(int i = 0; i < preguntas.size(); i++) {

enviarPreguntaCompleta(preguntas.get(i));

proceso++;

}

if(proceso == preguntas.size()) {

btnIniciar.setVisible(true);

btnIniciar.setDisable(false);

}

}

}

**EsperarJugadores.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.\*?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.text.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.EsperarJugadoresController">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints maxHeight="216.0" minHeight="0.0" prefHeight="78.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="283.0" minHeight="10.0" prefHeight="240.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<Label alignment="TOP\_CENTER" contentDisplay="CENTER" prefHeight="17.0" prefWidth="603.0" text="%lbl.jugadores\_unidos" textAlignment="CENTER" textFill="WHITE">

<font>

<Font size="25.0" />

</font>

</Label>

<ListView fx:id="listJugadores" prefHeight="257.0" prefWidth="410.0" GridPane.rowIndex="1">

<GridPane.margin>

<Insets left="70.0" right="70.0" />

</GridPane.margin>

</ListView>

<Button fx:id="btnIniciar" disable="true" mnemonicParsing="false" onAction="#iniciar" prefWidth="99.0" style="-fx-background-color: #F8B195;" text="%btn.iniciar" visible="false" GridPane.halignment="RIGHT" GridPane.rowIndex="2">

<GridPane.margin>

<Insets right="20.0" />

</GridPane.margin>

</Button>

<Button fx:id="btnCancelar" mnemonicParsing="false" onAction="#cancelar" prefWidth="99.0" style="-fx-background-color: #F67280;" text="%btn.cancelar" GridPane.rowIndex="2">

<GridPane.margin>

<Insets left="20.0" />

</GridPane.margin>

</Button>

</children>

</GridPane>

**Clase EsperarJugadoresController**

package juegopreguntantas;

import clasesutilidad.JugadorConectadoEnvio;

import entity.Cuentainvitado;

import entity.Cuentausuario;

import entity.Partida;

import entity.Setpregunta;

import io.socket.client.IO;

import io.socket.client.Socket;

import io.socket.emitter.Emitter;

import java.io.IOException;

import java.net.URISyntaxException;

import java.net.URL;

import java.util.List;

import java.util.Locale;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.application.Platform;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ListView;

import javafx.stage.Stage;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import persistencia.PersistenciaPartida;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 06/11/2018 \*

\* Nombre de la clase EsperarJugadoresController \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class EsperarJugadoresController implements Initializable {

@FXML

private ListView<JugadorConectadoEnvio> listJugadores;

@FXML

private Button btnIniciar;

@FXML

private Button btnCancelar;

private final Socket socket;

private Object cuenta;

private Cuentausuario usuario;

private Cuentainvitado invitado;

private String idioma;

private ObservableList<JugadorConectadoEnvio> conectados =

FXCollections.observableArrayList();

ObservableList<JugadorConectadoEnvio> conectadosCopia =

FXCollections.observableArrayList();

private int entrada;

private static final String RECURSO = "juegopreguntantas.lang/lang";

@Override

public void initialize(URL url, ResourceBundle rb) {

listJugadores.setItems(conectados);

}

/\*\*

\* Constructor de la clase, se mantiene a la espera de llamadas.

\* @throws URISyntaxException

\*/

public EsperarJugadoresController() throws URISyntaxException {

ResourceBundle propiedadesCliente = ResourceBundle.getBundle(

"utilidades.conexiones");

String ipServidor = propiedadesCliente.getString(

"key.ipServidor");

socket = IO.socket("http://" + ipServidor + ":5000");

socket.on(Socket.EVENT\_CONNECT, new Emitter.Listener() {

@Override

public void call(Object... os) {

//No es necesario hacer algo.

}

}).on("recuperoPreguntones", new Emitter.Listener() {

@Override

public void call(Object... os) {

JSONArray arregloRecuperado = (JSONArray) os[0];

conectadosCopia = conectados;

for (int i = 0; i < conectadosCopia.size(); i++) {

conectadosCopia.get(i).getNombre();

}

Platform.runLater(() -> conectados.clear());

for (int i = 0; i < arregloRecuperado.length(); i++) {

try {

JSONObject jugador =

(JSONObject) arregloRecuperado.get(i);

JugadorConectadoEnvio conectado =

new JugadorConectadoEnvio();

String nombre = jugador.getString("nombre");

conectado.setNombre(nombre);

Platform.runLater(() -> {

agregarConectado(conectado);

btnCancelar.setDisable(true);

if(entrada > -1) {

btnIniciar.setVisible(true);

btnIniciar.setDisable(false);

}

});

} catch (JSONException ex) {

Logger.getLogger(EsperarJugadoresController.class

.getName()).log(Level.SEVERE, null, ex);

}

}

}

}).on("inicioPartida", new Emitter.Listener() {

@Override

public void call(Object... os) {

if (entrada == -1) {

Platform.runLater(() -> {

abrirVentanaResponderPregunta();

socket.disconnect();

btnCancelar.getScene().getWindow().hide();

});

}

}

});

socket.connect();

}

/\*\*

\* Este metodo es para cancelar el inicio de la partida y regresar a la

\* ventana anterior de Inicio de partida

\*

\* @param event del click del mouse

\*/

@FXML

private void cancelar(ActionEvent event) {

eliminarPartida();

socket.emit("vaciarLista");

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("InicioPartida.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

InicioPartidaController controller = loader.getController();

controller.recibirParametros(usuario, idioma);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Inicio de partida");

stage.setScene(scene);

stage.show();

socket.disconnect();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(EsperarJugadoresController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo es para cancelar iniciar la partida para todos los jugadores

\* @param event del click del mouse

\*/

@FXML

private void iniciar(ActionEvent event) {

socket.emit("iniciarPartida");

eliminarPartida();

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("EnvioDePreguntas.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

EnvioDePreguntasController controller = loader.getController();

controller.recibirParametros(usuario, idioma, entrada);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Inicio de partida");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(EsperarJugadoresController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Metodo que recibe el objeto de cuenta de usuario o invitado del

\* Controlador de la pantalla que la invocó

\* @param cuenta objeto de la cuenta.

\* @param idioma Idioma del properties

\* @param entrada si es -1 entra como jugador, si es 2 entra como monitor.

\*/

public void recibirParametros(Object cuenta, String idioma, int entrada){

Locale.setDefault(new Locale(idioma));

this.idioma = idioma;

if(cuenta instanceof Cuentausuario) {

this.usuario = (Cuentausuario) cuenta;

} else {

this.invitado = (Cuentainvitado) cuenta;

}

this.cuenta = cuenta;

this.entrada = entrada;

if(entrada == -1) {

btnCancelar.setDisable(true);

btnCancelar.setVisible(false);

unirseAPartida();

socket.emit("recuperarPreguntones");

}

}

/\*\*

\* Este metodo verifica que no exista el nuevo jugador en la lista.

\* @param nuevo jugador para el envio y recibo.

\*/

private void agregarConectado(JugadorConectadoEnvio nuevo) {

if(!conectados.contains(nuevo)) {

conectados.add(nuevo);

}

}

/\*\*

\* Este metodo elimina la partida iniciada de la base de datos para que deje

\* de aparecer como disponibles para otros usuarios.

\*/

private void eliminarPartida() {

PersistenciaPartida partidaBD = new PersistenciaPartida();

Setpregunta nuevoSet = new Setpregunta();

nuevoSet.setIdsetpregunta(entrada);

List<Partida> partidas = partidaBD.recuperarPartida(nuevoSet);

int id = partidas.get(0).getIdpartida();

partidaBD.destroyPartida(id);

}

/\*\*

\* Este metodo abre la ventana de responder pregunta.

\*/

private void abrirVentanaResponderPregunta() {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("ResponderPregunta.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

ResponderPreguntaController controller = loader.getController();

controller.recibirParametros(cuenta, idioma, false);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("ResponderPregunta");

stage.setScene(scene);

stage.show();

} catch (IOException e) {

Logger.getLogger(EsperarJugadoresController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo registra un usuario en una lista de nodejs.

\*/

private void unirseAPartida() {

try {

JugadorConectadoEnvio conectado = new JugadorConectadoEnvio();

if (cuenta instanceof Cuentausuario) {

conectado.setNombre(usuario.getNombreusuario());

} else {

conectado.setNombre(invitado.getNombre());

}

JSONObject nuevoConectado = new JSONObject();

nuevoConectado.put("idSocket", socket.id());

nuevoConectado.put("nombre", conectado.getNombre());

socket.emit("registrarPregunton", nuevoConectado);

} catch (JSONException ex) {

Logger.getLogger(EsperarJugadoresController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

}

**EsperarPartida.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.\*?>

<?import javafx.geometry.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.EsperarPartidaController">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<Label alignment="CENTER" prefHeight="17.0" prefWidth="348.0" text="%lbl.categorias" textFill="WHITE" GridPane.rowIndex="1" />

<Label alignment="CENTER" prefHeight="17.0" prefWidth="470.0" text="%lbl.partidas" textFill="WHITE" GridPane.columnIndex="1" GridPane.rowIndex="1" />

<ComboBox fx:id="cbCategorias" onAction="#mostrarPartidas" prefHeight="25.0" prefWidth="475.0" style="-fx-background-color: #F8B195;" GridPane.halignment="CENTER" GridPane.rowIndex="2" GridPane.valignment="TOP">

<GridPane.margin>

<Insets left="20.0" right="20.0" />

</GridPane.margin>

</ComboBox>

<ComboBox fx:id="cbPartidas" prefHeight="25.0" prefWidth="353.0" style="-fx-background-color: #F8B195;" GridPane.columnIndex="1" GridPane.halignment="CENTER" GridPane.rowIndex="2" GridPane.valignment="TOP">

<GridPane.margin>

<Insets left="20.0" right="20.0" />

</GridPane.margin>

</ComboBox>

<Label alignment="CENTER" prefHeight="17.0" prefWidth="977.0" text="%lbl.partida\_en\_breve" textFill="WHITE" GridPane.columnSpan="2" GridPane.rowIndex="3">

<font>

<Font size="25.0" />

</font>

</Label>

<Button fx:id="btnUnirse" mnemonicParsing="false" onAction="#unirse" style="-fx-background-color: #F67280;" text="%btn.unirse" GridPane.columnIndex="1" GridPane.halignment="CENTER" GridPane.rowIndex="4" />

<Button fx:id="btnCancelar" mnemonicParsing="false" onAction="#cancelar" style="-fx-background-color: #F8B195;" text="%btn.cancelar" GridPane.halignment="CENTER" GridPane.rowIndex="4" />

</children>

</GridPane>

**Clase EsperarPartidaController**

package juegopreguntantas;

import entity.Partida;

import entity.Setpregunta;

import java.io.IOException;

import java.net.URISyntaxException;

import java.net.URL;

import java.util.List;

import java.util.Locale;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.stage.Stage;

import persistencia.PersistenciaCategoria;

import persistencia.PersistenciaPartida;

import persistencia.PersistenciaSetpregunta;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 06/11/2018 \*

\* Nombre de la clase EsperarPartidaController \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class EsperarPartidaController implements Initializable {

@FXML

private ComboBox<String> cbCategorias;

@FXML

private ComboBox<Partida> cbPartidas;

@FXML

private Button btnCancelar;

@FXML

private Button btnUnirse;

private Object cuenta;

private String idioma;

private static final String RECURSO = "juegopreguntantas.lang/lang";

@Override

public void initialize(URL url, ResourceBundle rb) {

//No es necesario hacer algo.

}

/\*\*

\* Constructor de la clase.

\* @throws URISyntaxException

\*/

public EsperarPartidaController() throws URISyntaxException {

//Vacio para instanciacion.

}

/\*\*

\* Este metodo se usa para unirse a una partida.

\* @param event Clic en el boton Unirse.

\*/

@FXML

private void unirse(ActionEvent event) {

if (validarCampos()) {

abrirVentanaJugadoresConectados();

btnCancelar.getScene().getWindow().hide();

}

}

/\*\*

\* Este metodo es para cancelar el inicio de la partida y regresar a la

\* ventana anterior de Inicio de partida

\* @param event del click del mouse

\*/

@FXML

private void cancelar(ActionEvent event) {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("MenuPrincipal.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

MenuPrincipalController controller = loader.getController();

controller.recibirParametros(cuenta, idioma);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Menu principal");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(EsperarPartidaController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo verifica que se hayan seleccionado los comboBox de las

\* categorias y partidas.

\* @return

\*/

private boolean validarCampos() {

boolean permiso = true;

if(cbCategorias.getSelectionModel().isEmpty()) {

permiso = false;

}

if(cbPartidas.getSelectionModel().isEmpty()) {

permiso = false;

}

return permiso;

}

/\*\*

\* Este metodo es para mostrar las partidas del set de preguntas

\* seleccionado anteriormente

\* @param event del click del mouse

\*/

@FXML

private void mostrarPartidas(ActionEvent event) {

PersistenciaPartida partidaBD = new PersistenciaPartida();

PersistenciaSetpregunta setPreguntaBD = new PersistenciaSetpregunta();

PersistenciaCategoria categoriaBD = new PersistenciaCategoria();

int id = categoriaBD.recuperarIdCategoria(cbCategorias.getSelectionModel().getSelectedItem());

List<Setpregunta> sets = setPreguntaBD.recuperarSetCategoria(id);

ObservableList<Partida> partidas = FXCollections.observableArrayList();

List<Partida> par;

for (int i = 0; i < sets.size(); i++) {

par = partidaBD.recuperarPartida(sets.get(i));

for (int j = 0; j < par.size(); j++) {

partidas.add(par.get(j));

}

}

cbPartidas.setItems(partidas);

}

/\*\*

\* Metodo que para mostrar las categorias de los set de pregunta que ha

\* hecho el usuario

\*/

public void mostrarCategorias(){

PersistenciaCategoria cate = new PersistenciaCategoria();

List<String> categoriasMostrar = cate.recuperarCategorias();

cbCategorias.getItems().addAll(categoriasMostrar);

if (cbCategorias.getItems().isEmpty()) {

cbCategorias.setDisable(true);

cbPartidas.setDisable(true);

}

}

/\*\*

\* Metodo que recibe el objeto de cuenta de usuario o invitado del

\* Controlador de la pantalla que la invocó

\* @param cuenta objeto invitado o usuario registrado

\* @param idioma Idioma del properties

\*/

public void recibirParametros(Object cuenta, String idioma){

this.idioma = idioma;

this.cuenta = cuenta;

mostrarCategorias();

}

/\*\*

\* Este metodo es para abrir la ventana de jugadores conectados.

\*/

public void abrirVentanaJugadoresConectados() {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle

= ResourceBundle.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("EsperarJugadores.fxml"));

loader.setResources(resourceBundle);

Parent responder = loader.load();

EsperarJugadoresController controller = loader.getController();

controller.recibirParametros(cuenta, idioma, -1);

Scene scene = new Scene(responder);

Stage stage = new Stage();

stage.setScene(scene);

stage.show();

} catch (Exception ex) {

Logger.getLogger(EsperarPartidaController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

}

**InicioPartida.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.\*?>

<?import javafx.geometry.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<?import java.lang.\*?>

<?import java.util.\*?>

<?import javafx.scene.\*?>

<?import javafx.scene.control.\*?>

<?import javafx.scene.layout.\*?>

<?import javafx.scene.text.\*?>

<?import javafx.scene.control.cell.\*?>

<?import javafx.collections.\*?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.InicioPartidaController">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="147.0" />

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="453.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<Label alignment="CENTER" prefHeight="36.0" prefWidth="608.0" text="%lbl.inicio\_partida" textFill="WHITE" GridPane.columnSpan="2">

<font>

<Font size="25.0" />

</font>

</Label>

<Label text="%lbl.nombre\_partida" textFill="WHITE" GridPane.rowIndex="1">

<GridPane.margin>

<Insets left="20.0" />

</GridPane.margin>

</Label>

<TextField fx:id="txtNombrePartida" GridPane.columnIndex="1" GridPane.rowIndex="1">

<GridPane.margin>

<Insets right="20.0" />

</GridPane.margin>

</TextField>

<Button fx:id="btnCancelar" mnemonicParsing="false" onAction="#cancelar" style="-fx-background-color: #F67280;" text="%btn.cancelar" GridPane.rowIndex="3">

<GridPane.margin>

<Insets left="20.0" />

</GridPane.margin>

</Button>

<Button fx:id="btnSiguiente" mnemonicParsing="false" onAction="#seguirEsperarJugadores" prefWidth="90.0" style="-fx-background-color: #F8B195;" text="%btn.siguiente" GridPane.columnIndex="1" GridPane.halignment="RIGHT" GridPane.rowIndex="3">

<GridPane.margin>

<Insets right="20.0" />

</GridPane.margin>

</Button>

<ComboBox fx:id="cbCategoria" prefHeight="25.0" prefWidth="290.0" style="-fx-background-color: #F8B195;" GridPane.columnIndex="1" GridPane.rowIndex="2" GridPane.valignment="CENTER">

<GridPane.margin>

<Insets left="20.0" right="10.0" />

</GridPane.margin>

</ComboBox>

<Label alignment="CENTER" prefHeight="17.0" prefWidth="285.0" text="%lbl.selec\_categoria" textFill="WHITE" GridPane.columnIndex="1" GridPane.rowIndex="2" GridPane.valignment="TOP" />

</children>

</GridPane>

**Clase InicioPartidaController**

package juegopreguntantas;

import entity.Cuentausuario;

import entity.Setpregunta;

import java.io.IOException;

import java.net.URL;

import java.util.List;

import java.util.Locale;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.application.Platform;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.TextField;

import javafx.stage.Stage;

import persistencia.PersistenciaPartida;

import persistencia.PersistenciaSetpregunta;

import utilidades.UtilidadCadenas;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 10/11/2018 \*

\* Nombre de la clase InicioPartidaController \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class InicioPartidaController implements Initializable {

@FXML

private TextField txtNombrePartida;

@FXML

private ComboBox<String> cbCategoria;

@FXML

private Button btnCancelar;

@FXML

private Button btnSiguiente;

private Cuentausuario usuario;

private String idioma;

private Setpregunta setPregunta;

private static final String RECURSO = "juegopreguntantas.lang/lang";

@Override

public void initialize(URL url, ResourceBundle rb) {

UtilidadCadenas cadena = new UtilidadCadenas();

cadena.limitarCampos(txtNombrePartida, 30);

excluirEspacios();

}

/\*\*

\* Constructor de la clase.

\*/

public InicioPartidaController() {

//Vacio para instanciacion

}

/\*\*

\* Este metodo es para cancelar el inicio de una partida

\* @param event del click del mouse

\*/

@FXML

private void cancelar(ActionEvent event) {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("MenuPrincipal.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

MenuPrincipalController controller = loader.getController();

controller.recibirParametros(usuario, idioma);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Menu Principal");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(InicioPartidaController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo es para continuar a la ventana de espera de jugadores para

\* iniciar la partida, la cual se registra antes

\* @param event del click del mouse

\*/

@FXML

private void seguirEsperarJugadores(ActionEvent event) {

if (validarCampos() && registrarPartida()) {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("EsperarJugadores.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

EsperarJugadoresController controller

= loader.getController();

controller.recibirParametros(usuario, idioma,

setPregunta.getIdsetpregunta());

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Espera de jugadores");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(InicioPartidaController.class

.getName()).log(Level.SEVERE, null, e);

}

}

}

/\*\*

\* Este metodo valida que el usuario haya seleccionado correctamente los

\* campos.

\* @return true si los campos necesarios estan seleccionados, false si no lo

\* son.

\*/

private boolean validarCampos() {

boolean permiso = false;

if(!txtNombrePartida.getText().isEmpty() &&

!cbCategoria.getSelectionModel().isEmpty()) {

permiso = true;

}

return permiso;

}

/\*\*

\* Metodo que para mostrar las categorias de los set de pregunta que ha

\* hecho el usuario

\*/

public void mostrarCategorias() {

PersistenciaSetpregunta setPreguntaBD = new PersistenciaSetpregunta();

List<String> categorias = setPreguntaBD.recuperarCategorias(usuario);

Platform.runLater(() -> cbCategoria.getItems().addAll(categorias));

if (categorias.isEmpty()) {

cbCategoria.setDisable(true);

}

}

/\*\*

\* Metodo que para registrar una partida

\* @return

\*/

private boolean registrarPartida() {

PersistenciaPartida partidaBD = new PersistenciaPartida();

PersistenciaSetpregunta setPreguntaBD = new PersistenciaSetpregunta();

List<Setpregunta> sets = setPreguntaBD.recuperarSetPregunta(usuario);

List<String> categorias = setPreguntaBD.recuperarCategorias(usuario);

setPregunta = sets.get(categorias.indexOf(cbCategoria.getValue()));

return partidaBD.crearPartida(txtNombrePartida.getText(), setPregunta);

}

/\*\*

\* Este metodo recibe los parametros de la ventana anterior y los fija

\* como atributo.

\* @param usuario usuario registrado.

\* @param idioma idioma del properties.

\*/

public void recibirParametros(Cuentausuario usuario, String idioma) {

Locale.setDefault(new Locale(idioma));

this.idioma = idioma;

this.usuario = usuario;

mostrarCategorias();

}

/\*\*

\* Este metodo sirve para que los textField nieguen la entrada a espacios.

\*/

private void excluirEspacios() {

txtNombrePartida.textProperty().addListener(

(observable, oldValue, newValue) -> {

if (newValue.contains(" ")) {

txtNombrePartida.setText(oldValue);

}

});

}

}

**Clase JuegoPreguntantas**

package juegopreguntantas;

import java.util.Locale;

import java.util.ResourceBundle;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 12/09/2018 \*

\* Nombre de la clase JuegoPreguntantas \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class JuegoPreguntantas extends Application {

@Override

public void start(Stage stage) throws Exception {

Locale.setDefault(new Locale("es"));

ResourceBundle resourceBundle =

ResourceBundle.getBundle("juegopreguntantas.lang/lang");

Parent root =

FXMLLoader.load(getClass().getResource("VentanaLogIn.fxml"),

resourceBundle);

Scene scene = new Scene(root);

stage.setScene(scene);

stage.show();

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

launch(args);

}

}

**MenuPrincipal.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.MenuPrincipalController">

<children>

<Label fx:id="lUser" layoutX="43.0" layoutY="27.0" text="Label" textFill="WHITE" AnchorPane.leftAnchor="43.0" AnchorPane.topAnchor="27.0">

<font>

<Font size="36.0" />

</font>

</Label>

<Button fx:id="btnRegistrar" layoutX="62.0" layoutY="147.0" mnemonicParsing="false" onAction="#registrarPregunta" prefHeight="39.0" prefWidth="172.0" style="-fx-background-color: #F8B195;" text="%btn.registrarPregunta" AnchorPane.leftAnchor="62.0">

<font>

<Font size="18.0" />

</font>

</Button>

<Button fx:id="btnUnirse" layoutX="339.0" layoutY="147.0" mnemonicParsing="false" onAction="#uniseAPartida" prefHeight="39.0" prefWidth="172.0" style="-fx-background-color: #F8B195;" text="%btn.unirse" AnchorPane.rightAnchor="89.0">

<font>

<Font size="18.0" />

</font>

</Button>

<Button fx:id="btnIniciar" layoutX="62.0" layoutY="254.0" mnemonicParsing="false" onAction="#iniciarPartida" prefHeight="39.0" prefWidth="172.0" style="-fx-background-color: #F8B195;" text="%btn.iniciar" AnchorPane.bottomAnchor="107.0" AnchorPane.leftAnchor="62.0">

<font>

<Font size="18.0" />

</font>

</Button>

<Button fx:id="btnInvitar" layoutX="339.0" layoutY="254.0" mnemonicParsing="false" onAction="#invitar" prefHeight="39.0" prefWidth="172.0" style="-fx-background-color: #F8B195;" text="%btn.invitar" AnchorPane.bottomAnchor="107.0" AnchorPane.rightAnchor="89.0">

<font>

<Font size="18.0" />

</font>

</Button>

<Button fx:id="btnCerrar" layoutX="492.0" layoutY="15.0" mnemonicParsing="false" onAction="#cerrarSesion" style="-fx-background-color: #F67280;" text="%btn.cerrarSesion" AnchorPane.rightAnchor="20.0" AnchorPane.topAnchor="20.0" />

</children>

</AnchorPane>

Clase MenuPrincipalController

package juegopreguntantas;

import entity.Cuentainvitado;

import entity.Cuentausuario;

import java.io.File;

import java.io.IOException;

import java.net.URL;

import java.util.Locale;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.stage.Stage;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Eduardo Rosas Rivera \*

\* @since 29/10/2018 \*

\* Nombre de la clase MenuPrincipalController \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class MenuPrincipalController implements Initializable {

@FXML

private Label lUser;

@FXML

private Button btnCerrar;

@FXML

private Button btnRegistrar;

@FXML

private Button btnUnirse;

@FXML

private Button btnIniciar;

@FXML

private Button btnInvitar;

private static final String RECURSO = "juegopreguntantas.lang/lang";

private Object cuenta;

private Cuentausuario usuario;

private Cuentainvitado invitado;

private String idioma;

/\*\*

\* Constructor de la clase.

\*/

public MenuPrincipalController() {

//Vacio para instanciacion

}

/\*\*

\* Initializes the controller class.

\* @param url

\* @param rb

\*/

@Override

public void initialize(URL url, ResourceBundle rb) {

crearCarpetas();

}

/\*\*

\* Este metodo abre la ventana para registrar preguntas y sets.

\*/

@FXML

private void registrarPregunta(ActionEvent event) {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("RegistrarPregunta.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

RegistrarPreguntaController controller = loader.getController();

controller.recibirParametros(cuenta, idioma);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Registro de pregunta");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(MenuPrincipalController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo es para ir a la ventana para unirse a una partida disponible

\* @param event del click del mouse

\*/

@FXML

private void uniseAPartida(ActionEvent event) {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("EsperarPartida.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

EsperarPartidaController controller = loader.getController();

controller.recibirParametros(cuenta, idioma);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Espera de inicio de partida");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(MenuPrincipalController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo es para ir a la ventana para iniciar una partida

\* @param event del click del mouse

\*/

@FXML

private void iniciarPartida(ActionEvent event) {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("InicioPartida.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

InicioPartidaController controller = loader.getController();

controller.recibirParametros(usuario, idioma);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Iniciar partida");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(MenuPrincipalController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo invoca a la pantalla EnviarInvitacion y le pasa como

\* parametro el objeto y el idioma.

\* @param event Clic en el boton invitar.

\*/

@FXML

private void invitar(ActionEvent event) {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("EnviarInvitacion.fxml"));

loader.setResources(resourceBundle);

Parent registro;

try {

registro = loader.load();

EnviarInvitacionController controller = loader.getController();

controller.recibirParametros(cuenta, idioma);

Scene scene = new Scene(registro);

Stage stage = new Stage();

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException ex) {

Logger.getLogger(VentanaLogInController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Metodo que cierra la sesion del usuario, si se trata de un usuario

\* invitado invoca borrarCuentaInvitado para elimianr su cuenta de la BD.

\* @param event Clic en el boton Cerrar Sesion.

\*/

@FXML

private void cerrarSesion(ActionEvent event) {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle =

ResourceBundle.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("VentanaLogIn.fxml"));

loader.setResources(resourceBundle);

try {

Parent logIn = loader.load();

VentanaLogInController controller = loader.getController();

controller.setIdioma(idioma);

Scene scene = new Scene(logIn);

Stage stage = new Stage();

stage.setScene(scene);

stage.show();

((Node) event.getSource()).getScene().getWindow().hide();

} catch (IOException ex) {

Logger.getLogger(MenuPrincipalController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Metodo que recibe el objeto de cuenta de usuario o invitado del

\* Controlador de la pantalla de Login.

\* @param cuenta Cuenta de invitado o usuario registrado.

\* @param idioma idioma del properties.

\*/

public void recibirParametros(Object cuenta, String idioma) {

Locale.setDefault(new Locale(idioma));

this.idioma = idioma;

this.cuenta = cuenta;

if(cuenta instanceof Cuentausuario) {

this.usuario = (Cuentausuario) cuenta;

lUser.setText(usuario.getNombreusuario());

} else {

this.invitado = (Cuentainvitado) cuenta;

btnIniciar.setDisable(true);

btnRegistrar.setDisable(true);

btnInvitar.setDisable(true);

lUser.setText(invitado.getNombre());

}

}

/\*\*

\* Este metodo verifica que exista que las carpetas necesarias para

\* las partidas existan.

\*/

private void crearCarpetas() {

File carpetaJuego = new File("PartidaActual");

File carpetaPartidas = new File("MisPartidas");

if (!carpetaJuego.exists()) {

carpetaJuego.mkdir();

}

if (!carpetaPartidas.exists()) {

carpetaPartidas.mkdir();

}

}

}

**RegistrarPregunta.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.\*?>

<?import javafx.scene.image.\*?>

<?import javafx.scene.canvas.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<?import javafx.collections.\*?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.RegistrarPreguntaController">

<children>

<Label layoutX="33.0" layoutY="105.0" text="%lbl.respuesta" textFill="WHITE" />

<TextField fx:id="txtPregunta" layoutX="121.0" layoutY="58.0" prefHeight="25.0" prefWidth="297.0" />

<Button fx:id="btnRespuesta2" layoutX="188.0" layoutY="169.0" mnemonicParsing="false" onAction="#seleccionarImagen" prefHeight="85.0" prefWidth="85.0" style="-fx-background-color: #F8B195;" text="+" />

<Button fx:id="btnRespuesta1" layoutX="51.0" layoutY="169.0" mnemonicParsing="false" onAction="#seleccionarImagen" prefHeight="85.0" prefWidth="85.0" style="-fx-background-color: #F8B195;" text="+" />

<Button fx:id="btnRespuesta4" layoutX="462.0" layoutY="169.0" mnemonicParsing="false" onAction="#seleccionarImagen" prefHeight="85.0" prefWidth="85.0" style="-fx-background-color: #F8B195;" text="+" />

<Button fx:id="btnRespuesta3" layoutX="326.0" layoutY="169.0" mnemonicParsing="false" onAction="#seleccionarImagen" prefHeight="85.0" prefWidth="85.0" style="-fx-background-color: #F8B195;" text="+" />

<ChoiceBox fx:id="cbCategoriaSet" layoutX="246.0" layoutY="309.0" prefHeight="25.0" prefWidth="196.0" style="-fx-background-color: #F8B195;">

<items>

<FXCollections fx:factory="observableArrayList">

<String fx:value="Arte" />

<String fx:value="Conocimiento general" />

<String fx:value="Colores" />

<String fx:value="Deportes" />

<String fx:value="Entretenimiento" />

<String fx:value="Historia" />

</FXCollections>

</items>

</ChoiceBox>

<ChoiceBox fx:id="cbRespuestaCorrecta" layoutX="33.0" layoutY="307.0" prefHeight="25.0" prefWidth="196.0" style="-fx-background-color: #F8B195;">

<items>

<FXCollections fx:factory="observableArrayList">

<String fx:value="1a" />

<String fx:value="2a" />

<String fx:value="3a" />

<String fx:value="4a" />

</FXCollections>

</items>

</ChoiceBox>

<Label layoutX="452.0" layoutY="281.0" text="%lbl.no\_set" textFill="WHITE" />

<Button fx:id="btnPregunta" layoutX="433.0" layoutY="30.0" mnemonicParsing="false" onAction="#seleccionarImagen" prefHeight="53.0" prefWidth="85.0" style="-fx-background-color: #F8B195;" text="+" />

<TextField fx:id="txtRespuesta1" layoutX="33.0" layoutY="134.0" prefHeight="25.0" prefWidth="120.0" />

<TextField fx:id="txtRespuesta2" layoutX="170.0" layoutY="134.0" prefHeight="25.0" prefWidth="120.0" />

<TextField fx:id="txtRespuesta3" layoutX="308.0" layoutY="134.0" prefHeight="25.0" prefWidth="120.0" />

<TextField fx:id="txtRespuesta4" layoutX="444.0" layoutY="134.0" prefHeight="25.0" prefWidth="120.0" />

<Button fx:id="btnAgregarPregunta" layoutX="537.0" layoutY="30.0" mnemonicParsing="false" onAction="#agregarPregunta" style="-fx-background-color: #F8B195;" text="+">

<tooltip>

<Tooltip text="%tt.agregar\_pregunta" />

</tooltip></Button>

<Button fx:id="btnEliminarPregunta" disable="true" layoutX="537.0" layoutY="56.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="25.0" style="-fx-background-color: #F67280;" text="-" visible="false">

<tooltip>

<Tooltip text="%tt.eliminar\_pregunta" />

</tooltip></Button>

<Button fx:id="btnEliminarSet" disable="true" layoutX="537.0" layoutY="307.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="25.0" style="-fx-background-color: #F67280;" text="-" visible="false">

<tooltip>

<Tooltip text="%tt.eliminar\_set" />

</tooltip></Button>

<GridPane layoutX="355.0" layoutY="350.0">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="173.0" minWidth="10.0" prefWidth="90.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="120.0" minWidth="10.0" prefWidth="27.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="95.0" minWidth="10.0" prefWidth="90.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<Button fx:id="btnRegistrar" mnemonicParsing="false" onAction="#registrarSet" prefHeight="25.0" prefWidth="134.0" style="-fx-background-color: #F8B195;" text="%btn.registrar\_set" GridPane.columnIndex="2" />

<Button fx:id="btnCancelar" mnemonicParsing="false" onAction="#cancelar" style="-fx-background-color: #F67280;" text="%btn.regresar" />

</children>

</GridPane>

<HBox layoutX="33.0" layoutY="280.0" prefHeight="17.0" prefWidth="411.0">

<children>

<Label prefHeight="17.0" prefWidth="214.0" text="%lbl.respuesta\_correcta" textFill="WHITE" />

<Label text="%lbl.categoria\_set" textFill="WHITE" />

</children>

</HBox>

<HBox layoutX="33.0" layoutY="28.0" prefHeight="17.0" prefWidth="385.0">

<children>

<Label prefHeight="17.0" prefWidth="100.0" text="%lbl.no\_pregunta" textFill="WHITE" />

<Label prefHeight="17.0" prefWidth="310.0" text="%lbl.pregunta" textFill="WHITE" />

</children>

</HBox>

<Separator layoutX="24.0" layoutY="96.0" prefHeight="0.0" prefWidth="553.0" />

<Separator layoutX="236.0" layoutY="271.0" orientation="VERTICAL" prefHeight="76.0" prefWidth="0.0" />

<Separator layoutX="237.0" layoutY="270.0" prefHeight="3.0" prefWidth="333.0" />

<Label layoutX="419.0" layoutY="60.0" text="?" textFill="WHITE">

<font>

<Font name="System Bold" size="12.0" />

</font>

</Label>

<Label layoutX="114.0" layoutY="60.0" text="¿" textFill="WHITE">

<font>

<Font name="System Bold" size="12.0" />

</font>

</Label>

<Button fx:id="btnCancelarImagen1" graphicTextGap="1.0" layoutX="33.0" layoutY="169.0" mnemonicParsing="false" onAction="#cancelarImagen" prefHeight="85.0" prefWidth="0.0" style="-fx-background-color: #F67280;" text="X" visible="false">

<font>

<Font size="10.0" />

</font>

</Button>

<Button fx:id="btnCancelarImagen2" graphicTextGap="1.0" layoutX="170.0" layoutY="169.0" mnemonicParsing="false" onAction="#cancelarImagen" prefHeight="85.0" prefWidth="0.0" style="-fx-background-color: #F67280;" text="X" visible="false">

<font>

<Font size="10.0" />

</font>

</Button>

<Button fx:id="btnCancelarImagen3" graphicTextGap="1.0" layoutX="308.0" layoutY="169.0" mnemonicParsing="false" onAction="#cancelarImagen" prefHeight="85.0" prefWidth="0.0" style="-fx-background-color: #F67280;" text="X" visible="false">

<font>

<Font size="10.0" />

</font>

</Button>

<Button fx:id="btnCancelarImagen4" graphicTextGap="1.0" layoutX="444.0" layoutY="169.0" mnemonicParsing="false" onAction="#cancelarImagen" prefHeight="85.0" prefWidth="0.0" style="-fx-background-color: #F67280;" text="X" visible="false">

<font>

<Font size="10.0" />

</font>

</Button>

<Button fx:id="btnCancelarPregunta" graphicTextGap="1.0" layoutX="432.0" layoutY="30.0" mnemonicParsing="false" onAction="#cancelarImagen" prefHeight="53.0" prefWidth="0.0" style="-fx-background-color: #F67280;" text="X" visible="false">

<font>

<Font size="10.0" />

</font>

</Button>

<Label fx:id="lblNoPregunta" alignment="CENTER" contentDisplay="CENTER" layoutX="37.0" layoutY="61.0" prefHeight="17.0" prefWidth="60.0" style="-fx-background-color: #F8B195;" text="00">

<font>

<Font name="System Bold" size="12.0" />

</font>

</Label>

<Label fx:id="lblNoSet" alignment="CENTER" contentDisplay="CENTER" layoutX="474.0" layoutY="313.0" prefHeight="17.0" prefWidth="60.0" style="-fx-background-color: #F8B195;" text="00">

<font>

<Font name="System Bold" size="12.0" />

</font>

</Label>

</children>

</AnchorPane>

**Clase RegistrarPreguntaController**

package juegopreguntantas;

import entity.Cuentausuario;

import entity.Pregunta;

import entity.Respuesta;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.net.URL;

import java.util.ArrayList;

import java.util.List;

import java.util.Locale;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.beans.value.ObservableValue;

import javafx.embed.swing.SwingFXUtils;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.ChoiceBox;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.stage.FileChooser;

import javafx.stage.Stage;

import javax.imageio.ImageIO;

import persistencia.PersistenciaSetpregunta;

import utilidades.UtilidadCadenas;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* @version 1.0 \*/

/\* @author Puxka Acosta Domínguez \*/

/\* @since 17/11/2018 \*/

/\* Nombre de la clase ResponderPreguntaController \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class RegistrarPreguntaController implements Initializable {

@FXML

private TextField txtPregunta;

@FXML

private Button btnRespuesta2;

@FXML

private Button btnRespuesta1;

@FXML

private Button btnRespuesta4;

@FXML

private Button btnRespuesta3;

@FXML

private Button btnCancelarImagen1;

@FXML

private Button btnCancelarImagen2;

@FXML

private Button btnCancelarImagen3;

@FXML

private Button btnCancelarImagen4;

@FXML

private ChoiceBox<String> cbCategoriaSet;

@FXML

private ChoiceBox<String> cbRespuestaCorrecta;

@FXML

private Button btnPregunta;

@FXML

private Button btnCancelarPregunta;

@FXML

private TextField txtRespuesta1;

@FXML

private TextField txtRespuesta2;

@FXML

private TextField txtRespuesta3;

@FXML

private TextField txtRespuesta4;

@FXML

private Label lblNoPregunta;

@FXML

private Button btnAgregarPregunta;

@FXML

private Button btnEliminarPregunta;

@FXML

private Button btnEliminarSet;

@FXML

private Label lblNoSet;

@FXML

private Button btnRegistrar;

@FXML

private Button btnCancelar;

private Cuentausuario cuenta;

private String idioma;

List<Pregunta> preguntas = new ArrayList<>();

@Override

public void initialize(URL url, ResourceBundle rb) {

UtilidadCadenas cadena = new UtilidadCadenas();

cadena.excluirSimbPregunta(txtPregunta);

cadena.limitarCampos(txtPregunta, 80);

cadena.limitarCampos(txtRespuesta1, 30);

cadena.limitarCampos(txtRespuesta2, 30);

cadena.limitarCampos(txtRespuesta3, 30);

cadena.limitarCampos(txtRespuesta4, 30);

}

/\*\*

\* Este metodo es para cancelar el inicio de la partida y regresar al menu

\* @param event del click del mouse al boton cancelar

\*/

@FXML

private void cancelar(ActionEvent event) {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle

.getBundle("juegopreguntantas.lang/lang");

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass()

.getResource("MenuPrincipal.fxml"));

loader.setResources(resourceBundle);

Parent esperaJugadores = loader.load();

MenuPrincipalController controller = loader.getController();

controller.recibirParametros(cuenta, idioma);

Scene scene = new Scene(esperaJugadores);

Stage stage = new Stage();

stage.setTitle("Menu principal");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException e) {

Logger.getLogger(RegistrarPreguntaController.class.getName())

.log(Level.SEVERE, null, e);

}

}

/\*\*

\* Este metodo es para registrar un set de preguntas

\* @param event del click del mouse al boton registrar set

\*/

@FXML

private void registrarSet(ActionEvent event) {

if ((Integer.parseInt(lblNoPregunta.getText()) > 0) &&

!cbCategoriaSet.getSelectionModel().isEmpty()) {

PersistenciaSetpregunta setPreguntaBD = new PersistenciaSetpregunta();

if (setPreguntaBD.crearSetPregunta(cbCategoriaSet.

getSelectionModel().getSelectedItem(), cuenta, preguntas)) {

int noUltimoSet = Integer.parseInt(lblNoSet.getText()) + 1;

lblNoSet.setText(Integer.toString(noUltimoSet));

preguntas.clear();

lblNoPregunta.setText("00");

mostrarCreacionExito();

} else {

mostrarCreacionFracaso();

}

} else if(Integer.parseInt(lblNoPregunta.getText()) == 0){

mostrarCampoVacio("agregar almenos una pregunta");

} else {

mostrarCampoVacio("elegir la categoria del set");

}

}

/\*\*

\* Este metodo es para agregar una pregunta con respuesta a la base de datos

\* , como también a la aplicación, en caso de que sea una imagen.

\* @param event del click del mouse al boton "+" de agregar pregunta

\*/

@FXML

private void agregarPregunta(ActionEvent event) {

List<String> paths = listaPaths();

List<Respuesta> respuestas = listaRespuesta(paths);

Pregunta nuevaPregunta = new Pregunta();

boolean exito = false;

if (!respuestas.isEmpty()) {

if (txtPregunta.getText().isEmpty() &&

!btnPregunta.getText().equals("+")) {

nuevaPregunta.setRespuestaCollection(respuestas);

nuevaPregunta.setPregunta(paths.get(0));

nuevaPregunta.setTipoPregunta(2);

preguntas.add(nuevaPregunta);

guardarImagen(paths.get(0), btnPregunta);

exito = true;

} else if (btnPregunta.getText().equals("+") &&

!txtPregunta.getText().isEmpty()) {

nuevaPregunta.setRespuestaCollection(respuestas);

nuevaPregunta.setPregunta("¿" + txtPregunta.getText() + "?");

nuevaPregunta.setTipoPregunta(1);

preguntas.add(nuevaPregunta);

txtPregunta.clear();

exito = true;

}

if (exito){

if(respuestas.get(0).getTipoRespuesta() == 1) {

txtRespuesta1.clear();

txtRespuesta2.clear();

txtRespuesta3.clear();

txtRespuesta4.clear();

} else {

guardarImagen(paths.get(1), btnRespuesta1);

guardarImagen(paths.get(2), btnRespuesta2);

guardarImagen(paths.get(3), btnRespuesta3);

guardarImagen(paths.get(4), btnRespuesta4);

}

limpiarBotones();

int noUltimaPregunta = Integer.

parseInt(lblNoPregunta.getText()) + 1;

lblNoPregunta.setText(Integer.toString(noUltimaPregunta));

}

}

if (!exito){

mostrarCampoVacio("llenar uno o más campos de la pregunta");

}

}

/\*\*

\* Este metodo es para seleccionar una imagen que tenga el usuario en su

\* computadora

\* @param event del click del mouse al boton "+" de los botones de respuesta

\* o pregunta

\*/

@FXML

private void seleccionarImagen(ActionEvent event) {

FileChooser fileChooser = new FileChooser();

FileChooser.ExtensionFilter imageFilter = new FileChooser.

ExtensionFilter("Image Files", "\*.jpg", "\*.png");

fileChooser.getExtensionFilters().add(imageFilter);

fileChooser.setTitle("Selecciona una imagen");

Stage stage = new Stage();

File file = fileChooser.showOpenDialog(stage);

if (file != null) {

Image image = new Image(file.toURI().toString());

ImageView imageView = new ImageView(image);

imageView.setFitWidth(85);

imageView.setFitHeight(85);

((Button) event.getSource()).setText("");

((Button) event.getSource()).setGraphic(imageView);

activarCancelarImagen((Button) event.getSource());

bloquearCampoTexto((Button) event.getSource());

}

}

/\*\*

\* Este metodo es para cancelar la imagen que se había subido

\* @param event del click del mouse "x" de los botones de respuesta o

\* pregunta

\*/

@FXML

private void cancelarImagen(ActionEvent event) {

if (((Button) event.getSource()).equals(btnCancelarImagen1)) {

btnRespuesta1.setGraphic(null);

btnRespuesta1.setText("+");

btnCancelarImagen1.setVisible(false);

txtRespuesta1.setDisable(false);

} else if (((Button) event.getSource()).equals(btnCancelarImagen2)) {

btnRespuesta2.setGraphic(null);

btnRespuesta2.setText("+");

btnCancelarImagen2.setVisible(false);

txtRespuesta2.setDisable(false);

} else if (((Button) event.getSource()).equals(btnCancelarImagen3)) {

btnRespuesta3.setGraphic(null);

btnRespuesta3.setText("+");

btnCancelarImagen3.setVisible(false);

txtRespuesta3.setDisable(false);

} else if (((Button) event.getSource()).equals(btnCancelarImagen4)) {

btnRespuesta4.setGraphic(null);

btnRespuesta4.setText("+");

btnCancelarImagen4.setVisible(false);

txtRespuesta4.setDisable(false);

} else if (((Button) event.getSource()).equals(btnCancelarPregunta)) {

btnPregunta.setGraphic(null);

btnPregunta.setText("+");

btnCancelarPregunta.setVisible(false);

txtPregunta.setDisable(false);

}

}

/\*\*

\* Este metodo es para comprobar que se subieron todas las imagenes de las

\* respuestas, en caso de que el usuario haya elegido subir imagenes

\* @return Si es verdadero o no que se seleccionaron las imagenes para las

\* respuestas

\*/

private boolean comprobarImagenLleno() {

return !btnRespuesta1.getText().equals("+") &&

!btnRespuesta2.getText().equals("+") &&

!btnRespuesta3.getText().equals("+") &&

!btnRespuesta4.getText().equals("+") &&

!cbRespuestaCorrecta.getSelectionModel().isEmpty();

}

/\*\*

\* Este metodo es para comprobar que se ingresaron todos los textos de las

\* respuestas, en caso de que el usuario haya elegido escribir respuestas

\* @return Si es verdadero o no que se escribieron las respuestas

\*/

private boolean comprobarTextoLleno() {

return !txtRespuesta1.getText().isEmpty() &&

!txtRespuesta2.getText().isEmpty() &&

!txtRespuesta3.getText().isEmpty() &&

!txtRespuesta4.getText().isEmpty() &&

!cbRespuestaCorrecta.getSelectionModel().isEmpty();

}

/\*\*

\* Este metodo es para activar los botones para cancelar una imagen, que

\* cuando no hay una seleccionada, estos botones estan ocultos.

\* @param imagenRespuesta boton que tiene la imagen

\*/

private void activarCancelarImagen(Button imagenRespuesta) {

if (imagenRespuesta.equals(btnRespuesta1)) {

btnCancelarImagen1.setVisible(true);

} else if (imagenRespuesta.equals(btnRespuesta2)) {

btnCancelarImagen2.setVisible(true);

} else if (imagenRespuesta.equals(btnRespuesta3)) {

btnCancelarImagen3.setVisible(true);

} else if (imagenRespuesta.equals(btnRespuesta4)) {

btnCancelarImagen4.setVisible(true);

} else if (imagenRespuesta.equals(btnPregunta)) {

btnCancelarPregunta.setVisible(true);

}

}

/\*\*

\* Este metodo es para bloquear el campo de texto de la respuesta en caso de

\* que se haya elegido subir una imagen, para evitar el error del usuario

\* @param imagenRespuesta boton que tiene la imagen

\*/

private void bloquearCampoTexto(Button imagenRespuesta) {

if (imagenRespuesta.equals(btnRespuesta1)) {

txtRespuesta1.setDisable(true);

txtRespuesta1.clear();

} else if (imagenRespuesta.equals(btnRespuesta2)) {

txtRespuesta2.setDisable(true);

txtRespuesta2.clear();

} else if (imagenRespuesta.equals(btnRespuesta3)) {

txtRespuesta3.setDisable(true);

txtRespuesta3.clear();

} else if (imagenRespuesta.equals(btnRespuesta4)) {

txtRespuesta4.setDisable(true);

txtRespuesta4.clear();

} else if (imagenRespuesta.equals(btnPregunta)) {

txtPregunta.setDisable(true);

txtPregunta.clear();

}

}

/\*\*

\* Este metodo es para guardar en una lista las rutas o paths para las

\* imagenes que se hayan subido como respuesta

\* @return Lista con las rutas de las imagenes

\*/

private List<String> listaPaths(){

List<String> paths = new ArrayList<>();

PersistenciaSetpregunta setPreguntaBD = new PersistenciaSetpregunta();

int noSet = setPreguntaBD.recuperarUltimaId() + 1;

paths.add(".\\imagenes\\" + noSet + "\\" + lblNoPregunta.getText() +

"\\imagenPregunta.png");

for(int i = 1; i < 5; i++){

paths.add(".\\imagenes\\" + noSet + "\\" + lblNoPregunta.getText() +

"\\imagenRespuesta" + i + ".png");

}

return paths;

}

/\*\*

\* Este metodo es para guardar en una lista las respuestas llenas

\* @param path La listade de rutas de las imagenes

\* @return Lista con las respuestas dadas por el usuario

\*/

private List<Respuesta> listaRespuesta(List<String> path){

List<Respuesta> respuestas = new ArrayList<>();

boolean respuestasLleno = false;

if (!cbRespuestaCorrecta.getSelectionModel().isEmpty()) {

Respuesta respuesta1 = new Respuesta();

Respuesta respuesta2 = new Respuesta();

Respuesta respuesta3 = new Respuesta();

Respuesta respuesta4 = new Respuesta();

respuesta1.setPuntaje(0);

respuesta2.setPuntaje(0);

respuesta3.setPuntaje(0);

respuesta4.setPuntaje(0);

switch (cbRespuestaCorrecta.getSelectionModel().getSelectedItem()) {

case "1a":

respuesta1.setPuntaje(1);

break;

case "2a":

respuesta2.setPuntaje(1);

break;

case "3a":

respuesta3.setPuntaje(1);

break;

case "4a":

respuesta4.setPuntaje(1);

break;

default:

break;

}

if (comprobarTextoLleno()) {

respuestasLleno = true;

respuesta1.setRespuesta(txtRespuesta1.getText());

respuesta2.setRespuesta(txtRespuesta2.getText());

respuesta3.setRespuesta(txtRespuesta3.getText());

respuesta4.setRespuesta(txtRespuesta4.getText());

respuesta1.setTipoRespuesta(1);

respuesta2.setTipoRespuesta(1);

respuesta3.setTipoRespuesta(1);

respuesta4.setTipoRespuesta(1);

} else if (comprobarImagenLleno()) {

respuestasLleno = true;

respuesta1.setRespuesta(path.get(1));

respuesta2.setRespuesta(path.get(2));

respuesta3.setRespuesta(path.get(3));

respuesta4.setRespuesta(path.get(4));

respuesta1.setTipoRespuesta(2);

respuesta2.setTipoRespuesta(2);

respuesta3.setTipoRespuesta(2);

respuesta4.setTipoRespuesta(2);

}

if (respuestasLleno) {

respuestas.add(respuesta1);

respuestas.add(respuesta2);

respuestas.add(respuesta3);

respuestas.add(respuesta4);

}

}

return respuestas;

}

/\*\*

\* Este metodo es para quitar las imagenes que tengan los botones y

\* regresarlos a su estado original, como tambien a su boton de cancelar y

\* el campo de texto

\*/

public void limpiarBotones(){

btnRespuesta1.setGraphic(null);

btnRespuesta2.setGraphic(null);

btnRespuesta3.setGraphic(null);

btnRespuesta4.setGraphic(null);

btnPregunta.setGraphic(null);

btnRespuesta1.setText("+");

btnRespuesta2.setText("+");

btnRespuesta3.setText("+");

btnRespuesta4.setText("+");

btnPregunta.setText("+");

btnCancelarImagen1.setVisible(false);

btnCancelarImagen2.setVisible(false);

btnCancelarImagen3.setVisible(false);

btnCancelarImagen4.setVisible(false);

btnCancelarPregunta.setVisible(false);

txtRespuesta1.setDisable(false);

txtRespuesta2.setDisable(false);

txtRespuesta3.setDisable(false);

txtRespuesta4.setDisable(false);

txtPregunta.setDisable(false);

}

/\*\*

\* Este metodo es para guardar la imagen que fue introducida por el usuario

\* @param path Ruta de la imagen

\* @param botonImagen El boton que contiene la imagen

\*/

public void guardarImagen(String path, Button botonImagen){

File outputFile = new File(path);

ImageView imageViewAdjusted = (ImageView) botonImagen.getGraphic();

imageViewAdjusted.setFitWidth(150);

imageViewAdjusted.setFitHeight(150);

outputFile.getParentFile().mkdirs();

BufferedImage bImage = SwingFXUtils.

fromFXImage(imageViewAdjusted.snapshot(null, null), null);

try {

ImageIO.write(bImage, "png", outputFile);

imageViewAdjusted.setFitWidth(85);

imageViewAdjusted.setFitHeight(85);

} catch (IOException | NullPointerException ex) {

Logger.getLogger(RegistrarPreguntaController.class.

getName()).log(Level.SEVERE, null, ex);

} finally {

imageViewAdjusted.setFitWidth(85);

imageViewAdjusted.setFitHeight(85);

}

}

/\*\*

\* Metodo que recibe el objeto de cuenta de usuario o invitado del

\* Controlador de la pantalla que la invocó

\* @param usuario Cuenta de usuario registrado

\* @param idioma Idioma del properties

\*/

public void recibirParametros(Object usuario, String idioma){

this.idioma = idioma;

this.cuenta = (Cuentausuario)usuario;

}

/\*\*

\* Este metodo impide que el campo de texto sea mayor a un numero de

\* caracteres fijo.

\* @param txtField textField a limitar

\* @param maximo numero maximo de caracteres permitidos.

\*/

public void limitarCampos(javafx.scene.control.TextField txtField,

int maximo) {

txtField.textProperty().addListener((final ObservableValue<?

extends String> ov, final String oldValue,

final String newValue) -> {

if (txtField.getText().length() > maximo) {

String s = txtField.getText().substring(0, maximo);

txtField.setText(s);

}

});

}

/\*\*

\* Este metodo es para mostrar una ventana en caso de fracaso de creacion de

\* un set

\*/

private void mostrarCreacionFracaso() {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Error");

alert.setHeaderText(null);

alert.setContentText("Se ha perdido conexión con el servidor"

+ ", prueba de nuevo");

alert.showAndWait();

}

/\*\*

\* Este metodo es para mostrar una ventana en caso de algun campo vacio

\* @param campo El campo que está vacío

\*/

private void mostrarCampoVacio(String campo) {

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setTitle("Revisa");

alert.setHeaderText(null);

alert.setContentText("Falta " + campo);

alert.showAndWait();

}

/\*\*

\* Este metodo es para mostrar una ventana en caso de exito de creacion de

\* un set

\*/

private void mostrarCreacionExito() {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Exito");

alert.setHeaderText(null);

alert.setContentText("Set creado con exito");

alert.showAndWait();

}

}

**RegistrarUsuario.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.\*?>

<?import javafx.scene.text.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.RegistrarUsuarioController">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="356.0" minWidth="10.0" prefWidth="147.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="493.0" minWidth="10.0" prefWidth="453.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints maxHeight="216.0" minHeight="0.0" prefHeight="78.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="267.0" minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints maxHeight="267.0" minHeight="10.0" prefHeight="40.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="40.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="40.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="40.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<Label alignment="TOP\_CENTER" contentDisplay="CENTER" prefHeight="17.0" prefWidth="603.0" text="%lbl.registrar\_usuario" textAlignment="CENTER" textFill="#fffbfb" GridPane.columnSpan="2">

<font>

<Font size="25.0" />

</font>

</Label>

<Label text="%lbl.nombre\_usuario" textFill="WHITE" GridPane.halignment="LEFT" GridPane.rowIndex="2">

<GridPane.margin>

<Insets left="20.0" />

</GridPane.margin>

</Label>

<Label layoutX="10.0" layoutY="49.0" text="%lbl.contrasenia" textFill="WHITE" GridPane.halignment="LEFT" GridPane.rowIndex="4">

<padding>

<Insets left="20.0" />

</padding>

</Label>

<Label alignment="CENTER" layoutX="10.0" layoutY="49.0" text="%lbl.correo\_electronico" textFill="WHITE" GridPane.halignment="LEFT" GridPane.rowIndex="3">

<padding>

<Insets left="20.0" />

</padding>

</Label>

<TextField fx:id="txtNombreUsuario" GridPane.columnIndex="1" GridPane.rowIndex="2">

<GridPane.margin>

<Insets right="20.0" />

</GridPane.margin>

</TextField>

<TextField fx:id="txtCorreoElectronico" layoutX="157.0" layoutY="105.0" GridPane.columnIndex="1" GridPane.rowIndex="3">

<GridPane.margin>

<Insets right="20.0" />

</GridPane.margin>

</TextField>

<GridPane GridPane.columnIndex="1" GridPane.rowIndex="6">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="341.0" minWidth="10.0" prefWidth="322.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="146.0" minWidth="0.0" prefWidth="19.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="112.0" minWidth="10.0" prefWidth="94.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

</rowConstraints>

<children>

<Button fx:id="btnCancelar" mnemonicParsing="false" onAction="#cancelar" style="-fx-background-color: #F8B195;" text="%btn.cancelar" GridPane.halignment="RIGHT" />

<Button fx:id="btnRegistrar" mnemonicParsing="false" onAction="#registrar" style="-fx-background-color: #F67280;" text="%btn.registrar" GridPane.columnIndex="2" GridPane.halignment="RIGHT" />

</children>

<padding>

<Insets right="20.0" />

</padding>

</GridPane>

<Label alignment="CENTER" prefHeight="17.0" prefWidth="656.0" text="%lbl.instruccion\_usuario" textFill="WHITE" GridPane.columnSpan="2" GridPane.rowIndex="1" GridPane.valignment="TOP" />

<Label fx:id="lMensaje" prefHeight="17.0" prefWidth="359.0" GridPane.columnIndex="1" GridPane.rowIndex="5" />

<PasswordField fx:id="pfPassword" layoutX="157.0" layoutY="105.0" prefHeight="25.0" prefWidth="425.0" GridPane.columnIndex="1" GridPane.rowIndex="4">

<GridPane.margin>

<Insets right="20.0" />

</GridPane.margin>

</PasswordField>

</children>

</GridPane>

**Clase RegistrarUsuarioController**

package juegopreguntantas;

import persistencia.PersistenciaCuentaUsuario;

import entity.Cuentausuario;

import java.io.IOException;

import java.net.URL;

import java.util.Date;

import java.util.Locale;

import java.util.Objects;

import java.util.Properties;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.stage.Stage;

import javax.mail.Authenticator;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.AddressException;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

import utilidades.UtilidadCadenas;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Eduardo Rosas Rivera \*

\* @since 02/11/2018 \*

\* Nombre de la clase RegistrarUsuarioController \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class RegistrarUsuarioController implements Initializable {

@FXML

private TextField txtNombreUsuario;

@FXML

private TextField txtCorreoElectronico;

@FXML

private PasswordField pfPassword;

@FXML

private Button btnCancelar;

@FXML

private Button btnRegistrar;

@FXML

private Label lMensaje;

private static final String RECURSO = "juegopreguntantas.lang/lang";

private String idioma;

/\*\*

\* Initializes the controller class.

\* @param url

\* @param rb

\*/

@Override

public void initialize(URL url, ResourceBundle rb) {

UtilidadCadenas cadena = new UtilidadCadenas();

cadena.limitarCampos(txtNombreUsuario, 26);

cadena.limitarCampos(pfPassword, 26);

cadena.limitarCampos(txtCorreoElectronico, 60);

excluirEspacios();

}

/\*\*

\* Este metodo regresa a la pantalla de login.

\* @param event Clic en cancelar.

\*/

@FXML

private void cancelar(ActionEvent event) {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle =

ResourceBundle.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("VentanaLogIn.fxml"));

loader.setResources(resourceBundle);

try {

Parent logIn = loader.load();

VentanaLogInController controller = loader.getController();

controller.setIdioma(idioma);

Scene scene = new Scene(logIn);

Stage stage = new Stage();

stage.setTitle("Inicio de sesion");

stage.setScene(scene);

stage.show();

((Node) event.getSource()).getScene().getWindow().hide();

} catch (IOException ex) {

Logger.getLogger(VentanaLogInController.class.getName()).log(

Level.SEVERE, null, ex);

}

}

/\*\*

\* Este metodo registra a un usuario en la base de datos.

\* @param event Clic en el boton Registrar.

\*/

@FXML

private void registrar(ActionEvent event) {

String nombre = txtNombreUsuario.getText().trim();

UtilidadCadenas cadena = new UtilidadCadenas();

String contraseniaNueva =

cadena.hacerHashAContrasenia(pfPassword.getText().trim());

String email = txtCorreoElectronico.getText().trim();

if (validarCampos()) {

if (verificarRegistroUsuario()) {

Cuentausuario usuario = new Cuentausuario();

usuario.setNombreusuario(nombre);

usuario.setContrasenia(contraseniaNueva);

usuario.setCorreoelectronico(email);

txtNombreUsuario.clear();

pfPassword.clear();

try {

String deCorreo = "juego.preguntantas@gmail.com";

final String contrasenia = "pr3gunt0n";

Properties properties = crearProperties();

Authenticator auth = new Authenticator() {

@Override

public PasswordAuthentication

getPasswordAuthentication() {

return new PasswordAuthentication(deCorreo,

contrasenia);

}

};

Session sesion = Session.getInstance(properties, auth);

Message mensaje = crearContenidoRegistro(sesion,

usuario);

mostrarRegistroExito(mensaje, usuario);

volverALogin();

} finally {

txtCorreoElectronico.clear();

}

}

} else {

lMensaje.setText(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_llenarCampos"));

}

}

/\*\*

\* Este metodo fija el idioma que enviara de retorno en caso de presionar el

\* boton cancelar.

\* @param idioma idioma fijado en la pantalla LogIn.

\*/

public void setIdioma(String idioma) {

this.idioma = idioma;

}

/\*\*

\* Este metodo valida que no exista algun campo vacio.

\* @return

\*/

private boolean validarCampos() {

boolean permiso = true;

String nombre = txtNombreUsuario.getText().trim();

String contrasenia = pfPassword.getText().trim();

String email = txtCorreoElectronico.getText().trim();

if(nombre.length() == 0) {

permiso = false;

}

if(contrasenia.length() == 0) {

permiso = false;

}

if(email.length() == 0) {

permiso = false;

}

return permiso;

}

/\*\*

\* Este metodo verifica si alguno de los dos metodos verificadores

\* obtienen true.

\* @return true si el usuario se puede registrar, false si no puede.

\*/

private boolean verificarRegistroUsuario() {

boolean permiso = true;

if(!verificarRegistrosNombre()) {

lMensaje.setText(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_NombreExistente"));

permiso = false;

}

if(!verificarRegistrosEmail()) {

lMensaje.setText(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_emailExistente"));

permiso = false;

}

return permiso;

}

/\*\*

\* Este metodo verifica que no exista ese nombre en la base de datos.

\* @return true si no existe, false si existe.

\*/

private boolean verificarRegistrosNombre() {

boolean permiso = false;

String user = txtNombreUsuario.getText().trim();

PersistenciaCuentaUsuario persistencia =

new PersistenciaCuentaUsuario();

Cuentausuario usuario = persistencia.getCuentaUsuarioNombre(

user.toUpperCase());

if(Objects.equals(usuario, null)) {

permiso = true;

}

return permiso;

}

/\*\*

\* Este metodo verifica que el correo electronico no exista en la

\* base de datos.

\* @return true si no existe, false si existe.

\*/

private boolean verificarRegistrosEmail() {

boolean permiso = false;

String email = txtCorreoElectronico.getText().trim();

PersistenciaCuentaUsuario persistencia =

new PersistenciaCuentaUsuario();

Cuentausuario usuario =

persistencia.getCuentaUsuarioEmail(email.toUpperCase());

if(Objects.equals(usuario, null)) {

permiso = true;

}

return permiso;

}

/\*\*

\* Este metodo es para hacer todos los put que necesitan las properties

\* @return El el properties para la sesion

\*/

private Properties crearProperties() {

Properties properties = new Properties();

String host = "smtp.gmail.com";

String puerto = "587";

properties.put("mail.smtp.host", host);

properties.put("mail.smtp.starttls.enable", "true");

properties.put("mail.smtp.ssl.trust", "smtp.gmail.com");

properties.put("mail.smtp.auth", "true");

properties.put("mail.smtp.port", puerto);

properties.put("mail.smtp.debug", "true");

return properties;

}

/\*\*

\* Este metodo es para crear el mensaje que se va a enviar por correo

\* @param sesion Session para enviar mensaje

\* @param nuevoInvitado Cuenta de invitado con los datos para el mensaje

\* @return El mensaje que se enviara por correo

\*/

private Message crearContenidoRegistro(Session sesion,

Cuentausuario nuevousuario) {

Message mensaje = new MimeMessage(sesion);

try {

InternetAddress[] address = {

new InternetAddress(nuevousuario.getCorreoelectronico())};

mensaje.setRecipients(Message.RecipientType.TO, address);

mensaje.setSubject(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_registro"));

String saludo = java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_holaUsuario");

String cuerpo = java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_registroExitoso")

+ java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_formaExitosa");

String contenidoCorreo = saludo + cuerpo;

mensaje.setSentDate(new Date());

mensaje.setText(contenidoCorreo);

} catch (AddressException ex) {

Logger.getLogger(EnviarInvitacionController.class.getName())

.log(Level.SEVERE, null, ex);

} catch (MessagingException ex) {

Logger.getLogger(EnviarInvitacionController.class.getName())

.log(Level.SEVERE, null, ex);

}

return mensaje;

}

/\*\*

\* Este metodo es para mostrar una ventana en caso de exito

\* @param mensaje Message que se va a enviar por correo

\* @param nuevoInvitado Cuenta de invitado que se guardara en BD

\*/

private void mostrarRegistroExito(Message mensaje,

Cuentausuario nuevoUsuario) {

PersistenciaCuentaUsuario usuarioBD =

new PersistenciaCuentaUsuario();

if (usuarioBD.crearUsuario(nuevoUsuario)) {

try {

Transport.send(mensaje);

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_tituloExito"));

alert.setHeaderText(null);

alert.setContentText(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_confirmacionEnvio")

+ java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_email"));

alert.showAndWait();

} catch (MessagingException ex) {

Logger.getLogger(EnviarInvitacionController.class.getName())

.log(Level.SEVERE, null, ex);

usuarioBD.eliminarUsuario(nuevoUsuario);

mostrarRegistroFracaso();

}

} else {

mostrarRegistroFracaso();

}

}

/\*\*

\* Este metodo es para mostrar una ventana en caso de fracaso al registrar.

\*/

private void mostrarRegistroFracaso() {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Error");

alert.setHeaderText(null);

alert.setContentText("Se ha perdido conexión con el servidor"

+ ", prueba de nuevo");

alert.showAndWait();

}

/\*\*

\* Este metodo sirve para volver a la pantalla de Login.

\*/

private void volverALogin() {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("VentanaLogIn.fxml"));

loader.setResources(resourceBundle);

try {

Parent logIn = loader.load();

VentanaLogInController controller = loader.getController();

controller.setIdioma(idioma);

Scene scene = new Scene(logIn);

Stage stage = new Stage();

stage.setScene(scene);

stage.show();

btnRegistrar.getScene().getWindow().hide();

} catch (IOException ex) {

Logger.getLogger(ResponderPreguntaController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Este metodo limita niega la posibilidad de escribir espacios en un

\* TextField.

\*/

private void excluirEspacios() {

txtNombreUsuario.textProperty().addListener(

(observable, oldValue, newValue) -> {

if (newValue.contains(" ")) {

txtNombreUsuario.setText(oldValue);

}

});

txtCorreoElectronico.textProperty().addListener(

(observable, oldValue, newValue) -> {

if (newValue.contains(" ")) {

txtCorreoElectronico.setText(oldValue);

}

});

pfPassword.textProperty().addListener(

(observable, oldValue, newValue) -> {

if (newValue.contains(" ")) {

pfPassword.setText(oldValue);

}

});

}

}

**ResponderPregunta.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.paint.\*?>

<?import javafx.geometry.\*?>

<?import javafx.scene.chart.\*?>

<?import javafx.scene.text.\*?>

<?import javafx.scene.image.\*?>

<?import javafx.scene.control.\*?>

<?import java.lang.\*?>

<?import javafx.scene.layout.\*?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="602.0" prefWidth="789.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.ResponderPreguntaController">

<children>

<TextField fx:id="tfMensaje" layoutX="8.0" layoutY="362.0" prefHeight="25.0" prefWidth="149.0" AnchorPane.bottomAnchor="10.0" AnchorPane.leftAnchor="2.0" />

<Button fx:id="btnEnviar" layoutX="149.0" layoutY="377.0" mnemonicParsing="false" onAction="#enviar" prefHeight="25.0" prefWidth="52.0" style="-fx-background-color: #F67280;" text="%btn.enviar" AnchorPane.bottomAnchor="10.0" AnchorPane.leftAnchor="149.0" />

<TextArea fx:id="txtChat" editable="false" layoutX="2.0" layoutY="2.0" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="352.0" prefWidth="198.0" wrapText="true" AnchorPane.bottomAnchor="46.0" AnchorPane.leftAnchor="2.0" AnchorPane.topAnchor="2.0" />

<ImageView fx:id="imgA" disable="true" fitHeight="150.0" fitWidth="150.0" layoutX="266.0" layoutY="267.0" onMouseClicked="#seleccionarA" pickOnBounds="true" preserveRatio="true" visible="false" AnchorPane.bottomAnchor="185.0" AnchorPane.leftAnchor="266.0" AnchorPane.rightAnchor="373.0">

<image>

<Image url="@../../PartidaActual/1.png" />

</image></ImageView>

<ImageView fx:id="imgB" disable="true" fitHeight="150.0" fitWidth="150.0" layoutX="266.0" layoutY="438.0" onMouseClicked="#seleccionarB" pickOnBounds="true" preserveRatio="true" visible="false" AnchorPane.bottomAnchor="14.0" AnchorPane.rightAnchor="373.0">

<image>

<Image url="@../../PartidaActual/2.png" />

</image></ImageView>

<ImageView fx:id="imgC" disable="true" fitHeight="150.0" fitWidth="150.0" layoutX="565.0" layoutY="267.0" onMouseClicked="#seleccionarC" pickOnBounds="true" preserveRatio="true" visible="false" AnchorPane.bottomAnchor="185.0" AnchorPane.leftAnchor="565.0">

<image>

<Image url="@../../PartidaActual/3.png" />

</image></ImageView>

<ImageView fx:id="imgD" disable="true" fitHeight="150.0" fitWidth="150.0" layoutX="565.0" layoutY="438.0" onMouseClicked="#seleccionarD" pickOnBounds="true" preserveRatio="true" visible="false" AnchorPane.bottomAnchor="14.0" AnchorPane.rightAnchor="74.0">

<image>

<Image url="@../../PartidaActual/4.png" />

</image>

<viewport>

<Rectangle2D />

</viewport></ImageView>

<ImageView fx:id="imgPregunta" fitHeight="150.0" fitWidth="150.0" layoutX="401.0" layoutY="95.0" pickOnBounds="true" preserveRatio="true" AnchorPane.bottomAnchor="364.0" AnchorPane.leftAnchor="401.0" AnchorPane.rightAnchor="238.0" AnchorPane.topAnchor="95.0" />

<Separator layoutX="296.0" layoutY="44.0" prefHeight="4.0" prefWidth="268.0" AnchorPane.leftAnchor="296.0" AnchorPane.rightAnchor="104.0" />

<Label fx:id="lblPuntaje" layoutX="296.0" layoutY="21.0" prefHeight="27.0" prefWidth="73.0" text="%lbl.puntaje" textFill="WHITE" AnchorPane.leftAnchor="296.0" AnchorPane.rightAnchor="281.0" AnchorPane.topAnchor="21.0">

<font>

<Font size="18.0" />

</font>

</Label>

<Label fx:id="lblTiempo" layoutX="442.0" layoutY="21.0" textFill="WHITE" AnchorPane.leftAnchor="480.0" AnchorPane.rightAnchor="108.0">

<font>

<Font size="18.0" />

</font>

</Label>

<Label fx:id="lbPuntajeNum" layoutX="416.0" layoutY="21.0" text="0" textFill="RED" AnchorPane.leftAnchor="416.0" AnchorPane.rightAnchor="224.0" AnchorPane.topAnchor="21.0">

<font>

<Font size="18.0" />

</font>

</Label>

<Label fx:id="lblPreguntaNo" layoutX="325.0" layoutY="59.0" text="%lbl.no\_pregunta" textFill="WHITE" AnchorPane.leftAnchor="325.0" AnchorPane.rightAnchor="182.0">

<font>

<Font size="18.0" />

</font>

</Label>

<Label fx:id="lblNumeroPregunta" layoutX="476.0" layoutY="59.0" text="0" textFill="WHITE" AnchorPane.leftAnchor="476.0">

<font>

<Font size="18.0" />

</font>

</Label>

<Label fx:id="lblCategoria" layoutX="381.0" layoutY="86.0" textFill="WHITE" AnchorPane.leftAnchor="381.0" AnchorPane.rightAnchor="189.0" />

<Label fx:id="lblPregunta" contentDisplay="CENTER" layoutX="380.0" layoutY="108.0" prefHeight="106.0" prefWidth="549.0" textAlignment="CENTER" textFill="WHITE" AnchorPane.leftAnchor="220.0" AnchorPane.rightAnchor="20.0">

<font>

<Font size="18.0" />

</font>

</Label>

<CheckBox fx:id="chbOpcionA" disable="true" layoutX="211.0" layoutY="349.0" mnemonicParsing="false" onMouseClicked="#seleccionarA" text="CheckBox" textFill="WHITE" visible="false">

<font>

<Font size="14.0" />

</font></CheckBox>

<CheckBox fx:id="chbOpcionB" disable="true" layoutX="211.0" layoutY="388.0" mnemonicParsing="false" onMouseClicked="#seleccionarB" text="CheckBox" textFill="WHITE" visible="false">

<font>

<Font size="14.0" />

</font></CheckBox>

<CheckBox fx:id="chbOpcionC" disable="true" layoutX="211.0" layoutY="428.0" mnemonicParsing="false" onMouseClicked="#seleccionarC" text="CheckBox" textFill="WHITE" visible="false">

<font>

<Font size="14.0" />

</font></CheckBox>

<CheckBox fx:id="chbOpcionD" disable="true" layoutX="211.0" layoutY="467.0" mnemonicParsing="false" onMouseClicked="#seleccionarD" text="CheckBox" textFill="WHITE" visible="false">

<font>

<Font size="14.0" />

</font></CheckBox>

<Pane fx:id="pPanelGrafica" layoutX="201.0" layoutY="5.0" prefHeight="595.0" prefWidth="589.0" style="-fx-background-color: #355C7D;" visible="false" AnchorPane.bottomAnchor="2.0" AnchorPane.leftAnchor="200.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="5.0">

<children>

<BarChart fx:id="bcGraficaPuntaje" animated="false" layoutX="3.0" layoutY="1.0" prefHeight="585.0" prefWidth="550.0" title="%bar\_puntajes">

<xAxis>

<CategoryAxis fx:id="caJugadores" animated="false" label="%bar\_jugadores" side="BOTTOM" tickLabelFill="WHITE" />

</xAxis>

<yAxis>

<NumberAxis fx:id="caPuntos" animated="false" label="%bar\_puntos" side="LEFT" tickLabelFill="WHITE" />

</yAxis>

<padding>

<Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />

</padding>

</BarChart>

</children>

</Pane>

<Pane fx:id="pPanelGanador" layoutX="202.0" layoutY="10.0" prefHeight="585.0" prefWidth="583.0" style="-fx-background-color: #355C7D;" visible="false">

<children>

<Label fx:id="lblAnunciador" layoutX="151.0" layoutY="33.0" prefHeight="56.0" prefWidth="338.0" text="%lbl.el\_ganador\_es" textAlignment="CENTER" textFill="WHITE">

<font>

<Font size="36.0" />

</font>

</Label>

<Label fx:id="lblGanador" disable="true" layoutX="19.0" layoutY="150.0" prefHeight="85.0" prefWidth="561.0" textAlignment="CENTER" textFill="#f8b195" visible="false">

<font>

<Font size="48.0" />

</font>

</Label>

<Label fx:id="lblBuenJuego" layoutX="17.0" layoutY="381.0" prefHeight="160.0" prefWidth="567.0" text="%lbl.buen\_juego" textAlignment="CENTER" textFill="#f67280">

<font>

<Font size="64.0" />

</font>

</Label>

<Button fx:id="btnvolver" layoutX="266.0" layoutY="529.0" mnemonicParsing="false" onAction="#volverAMenu" style="-fx-background-color: #F67280;" text="%btn.exit" />

</children>

</Pane>

<Pane fx:id="pnlEspera" layoutX="202.0" layoutY="3.0" prefHeight="595.0" prefWidth="583.0" style="-fx-background-color: #355C7D;">

<children>

<ProgressBar fx:id="pbEspera" layoutX="27.0" layoutY="289.0" prefHeight="18.0" prefWidth="533.0" />

<Label layoutX="138.0" layoutY="161.0" text="%btn\_recibiendo\_preguntas" textFill="WHITE">

<font>

<Font size="24.0" />

</font>

</Label>

</children>

</Pane>

</children>

</AnchorPane>

**Clase ResponderPreguntaController**

package juegopreguntantas;

import clasesutilidad.JugadorConectadoEnvio;

import entity.Cuentainvitado;

import entity.Cuentausuario;

import java.io.File;

import java.net.URL;

import java.util.ArrayList;

import java.util.Locale;

import java.util.ResourceBundle;

import java.util.Timer;

import java.util.TimerTask;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.application.Platform;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.chart.BarChart;

import javafx.scene.control.Button;

import javafx.scene.control.CheckBox;

import javafx.scene.control.Label;

import javafx.scene.control.ProgressBar;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.Pane;

import clasesutilidad.PreguntaEnvio;

import clasesutilidad.PuntajeEnvio;

import clasesutilidad.RespuestaEnvio;

import com.google.gson.Gson;

import io.socket.client.IO;

import io.socket.client.Socket;

import io.socket.emitter.Emitter;

import java.io.BufferedOutputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStream;

import java.net.URISyntaxException;

import java.util.Collections;

import java.util.List;

import java.util.Objects;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.chart.XYChart;

import javafx.stage.Stage;

import javax.xml.bind.DatatypeConverter;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import utilidades.UtilidadCadenas;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Eduardo Rosas Rivera \*

\* @since 28/11/2018 \*

\* Nombre de la clase ResponderPreguntaController \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class ResponderPreguntaController implements Initializable {

@FXML

private TextField tfMensaje;

@FXML

private Button btnEnviar;

@FXML

private TextArea txtChat;

@FXML

private ImageView imgPregunta;

@FXML

private ImageView imgA;

@FXML

private ImageView imgB;

@FXML

private ImageView imgC;

@FXML

private ImageView imgD;

@FXML

private Label lblPuntaje;

@FXML

private Label lblTiempo;

@FXML

private Label lbPuntajeNum;

@FXML

private Label lblPreguntaNo;

@FXML

private Label lblCategoria;

@FXML

private Label lblPregunta;

@FXML

private CheckBox chbOpcionA;

@FXML

private CheckBox chbOpcionB;

@FXML

private CheckBox chbOpcionC;

@FXML

private CheckBox chbOpcionD;

@FXML

private Pane pPanelGrafica;

@FXML

private Pane pPanelGanador;

@FXML

private Label lblAnunciador;

@FXML

private Label lblNumeroPregunta;

@FXML

private Label lblGanador;

@FXML

private Label lblBuenJuego;

@FXML

private BarChart<?, ?> bcGraficaPuntaje;

@FXML

private ProgressBar pbRestante;

@FXML

private Button btnVolver;

@FXML

private Pane pnlEspera;

@FXML

private ProgressBar pbEspera;

private static final Integer TIEMPORESPUESTA = 15;

private static final Integer TIEMPOGRAFICA = 10;

private final ArrayList<String> mensajesChat = new ArrayList<>();

private final Socket socket;

private Object cuenta;

private String idioma;

private Cuentausuario usuario;

private Cuentainvitado invitado;

private int tiempoRestante;

private int tiempoTotalGrafica;

private List<PreguntaEnvio> preguntas = new ArrayList<>();

private List<RespuestaEnvio> respuestas = new ArrayList<>();

private Timer timer;

private Timer timerGraph;

private PreguntaEnvio preguntaActual;

private boolean activacion = true;

private final ObservableList<JugadorConectadoEnvio> participantes =

FXCollections.observableArrayList();

private int contador = 0;

private boolean creador;

/\*\*

\* Initializes the controller class.

\* @param url

\* @param rb

\*/

@Override

public void initialize(URL url, ResourceBundle rb) {

socket.emit("recuperarPreguntones");

activarPanelGanador(false);

activarRespuestaEscrita(false);

UtilidadCadenas cadena = new UtilidadCadenas();

cadena.limitarCampos(tfMensaje, 70);

}

/\*\*

\* Constructor de la clase.

\* @throws URISyntaxException

\*/

public ResponderPreguntaController() throws URISyntaxException {

ResourceBundle propiedadesCliente = ResourceBundle.getBundle(

"utilidades.conexiones");

String ipServidor = propiedadesCliente.getString("key.ipServidor");

socket = IO.socket("http://" + ipServidor + ":4000");

socket.on(Socket.EVENT\_CONNECT, new Emitter.Listener() {

@Override

public void call(Object... os) {

//No es necesario hacer algo.

}

}).on("reciboMensaje", new Emitter.Listener(){

@Override

public void call(Object... os) {

String mensajeRecibido = (String) os[0];

mostrarMensaje(mensajeRecibido);

}

}).on("reciboPregunta", new Emitter.Listener() {

@Override

public void call(Object... os) {

String preguntaRecibida = (String) os[0];

Platform.runLater(() -> recibirPregunta(preguntaRecibida));

}

}).on("inicioPartida", new Emitter.Listener() {

@Override

public void call(Object... os) {

if(!creador) {

Platform.runLater(() -> {

activacion = true;

desactivarPanelEspera(activacion);

verificarPregunta();

});

}

}

}).on("recuperoPreguntones", new Emitter.Listener() {

@Override

public void call(Object... os) {

JSONArray recuperados = (JSONArray) os[0];

Platform.runLater(() -> recuperarPreguntones(recuperados));

}

}).on("recibirRespuesta", new Emitter.Listener() {

@Override

public void call(Object... os) {

String respuesta = (String) os[0];

Platform.runLater(() -> recibirRespuesta(respuesta));

}

}).on("finalizoPartida", new Emitter.Listener() {

@Override

public void call(Object... os) {

if(creador) {

Platform.runLater(() -> {

activarPanelGanador(true);

creador = false;

});

}

}

});

socket.connect();

}

/\*\*

\* Este metodo es para disminuir la lista de preguntas

\*/

private void disminuirPreguntas() {

preguntas.remove(0);

verificarPregunta();

}

/\*\*

\* Este metodo verifica que la pregunta tiene algo, de ser nula termina el

\* juego.

\*/

private void verificarPregunta() {

if(preguntas.isEmpty()) {

socket.emit("finalizarPartida");

activarPanelGanador(true);

} else {

preguntaActual = preguntas.get(0);

Platform.runLater(() -> lblNumeroPregunta.setText(Integer.toString(

Integer.parseInt(lblNumeroPregunta.getText()) + 1)));

respuestas = preguntaActual.getRespuestas();

definirTipoPregunta();

}

}

/\*\*

\* Este metood sirve para definir si la pregunta recibida es texto(1) o es

\* imagen(2).

\*/

private void definirTipoPregunta() {

if (preguntaActual.getTipoPregunta() == 1) {

Platform.runLater(() ->

lblPregunta.setText(preguntaActual.getPregunta()));

imgPregunta.setVisible(false);

imgPregunta.setDisable(true);

} else {

try {

String ruta = "PartidaActual\\";

String imagenPregunta;

File filePregunta = new File(preguntaActual.getPregunta());

imagenPregunta = ruta + filePregunta.getName();

crearImagen(imagenPregunta, preguntaActual.getImagenByte());

filePregunta = new File(imagenPregunta);

Image imagePregunta = new Image(filePregunta.toURI()

.toString());

imgPregunta.setImage(imagePregunta);

} catch (IOException ex) {

Logger.getLogger(ResponderPreguntaController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

definirTipoRespuesta();

}

/\*\*

\* Este metodo verifica si la primera respuesta es de tipo imagen, de ser

\* asi activa los imageView, de otra forma, activa los checkBox.

\*/

private void definirTipoRespuesta() {

if(respuestas.get(0).getTipoRespuesta() == 2) {

activarRespuestaImagen(true);

activarRespuestaEscrita(false);

fijarImagenes();

} else {

activarRespuestaEscrita(true);

activarRespuestaImagen(false);

fijarTextos();

}

}

/\*\*

\* Este metodo fija las imagenes en los imageView en caso deque las

\* respuestas sean imagen.

\*/

private void fijarImagenes() {

try {

String ruta = "PartidaActual\\";

String imagenA;

String imagenB;

String imagenC;

String imagenD;

File file1 = new File(respuestas.get(0).getRespuesta());

File file2 = new File(respuestas.get(1).getRespuesta());

File file3 = new File(respuestas.get(2).getRespuesta());

File file4 = new File(respuestas.get(3).getRespuesta());

imagenA = ruta + file1.getName();

crearImagen(imagenA, respuestas.get(0).getImagenByte());

file1 = new File(imagenA);

imagenB = ruta + file2.getName();

crearImagen(imagenB, respuestas.get(1).getImagenByte());

file2 = new File(imagenB);

imagenC = ruta + file3.getName();

crearImagen(imagenC, respuestas.get(2).getImagenByte());

file3 = new File(imagenC);

imagenD = ruta + file4.getName();

crearImagen(imagenD, respuestas.get(3).getImagenByte());

file4 = new File(imagenD);

Image imageA = new Image(file1.toURI().toString());

Image imageB = new Image(file2.toURI().toString());

Image imageC = new Image(file3.toURI().toString());

Image imageD = new Image(file4.toURI().toString());

imgA.setImage(imageA);

imgB.setImage(imageB);

imgC.setImage(imageC);

imgD.setImage(imageD);

contarRegresivamente();

} catch (IOException ex) {

Logger.getLogger(ResponderPreguntaController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Este metodo habilita los imageView cuando se trata de una pregunta con

\* una respuesta de imagenes.

\* @param permiso true si deben ser visibles y activos, false si no.

\*/

private void activarRespuestaImagen(boolean permiso) {

imgA.setVisible(permiso);

imgA.setDisable(!permiso);

imgB.setVisible(permiso);

imgB.setDisable(!permiso);

imgC.setVisible(permiso);

imgC.setDisable(!permiso);

imgD.setVisible(permiso);

imgD.setDisable(!permiso);

}

/\*\*

\* Este metodo fija el texto de la pregunta y las respuestas.

\*/

private void fijarTextos() {

chbOpcionA.setSelected(false);

chbOpcionB.setSelected(false);

chbOpcionC.setSelected(false);

chbOpcionD.setSelected(false);

Platform.runLater(() -> chbOpcionA.setText(respuestas.get(0).

getRespuesta()));

Platform.runLater(() -> chbOpcionB.setText(respuestas.get(1).

getRespuesta()));

Platform.runLater(() -> chbOpcionC.setText(respuestas.get(2).

getRespuesta()));

Platform.runLater(() -> chbOpcionD.setText(respuestas.get(3).

getRespuesta()));

contarRegresivamente();

}

/\*\*

\* Este metodo habilita los checkBox cuando se trata de una pregunta con

\* respuestas escritas.

\* @param permiso true si deben ser visibles y activos, false si no.

\*/

private void activarRespuestaEscrita(boolean permiso) {

chbOpcionA.setDisable(!permiso);

chbOpcionA.setVisible(permiso);

chbOpcionB.setDisable(!permiso);

chbOpcionB.setVisible(permiso);

chbOpcionC.setDisable(!permiso);

chbOpcionC.setVisible(permiso);

chbOpcionD.setDisable(!permiso);

chbOpcionD.setVisible(permiso);

}

/\*\*

\* Este metodo sirve para enviar un mensaje en el chat.

\* @param event Clic en el boton Enviar.

\*/

@FXML

private void enviar(ActionEvent event) {

String nombre;

String mensaje = tfMensaje.getText().trim();

if (cuenta instanceof Cuentausuario) {

nombre = usuario.getNombreusuario();

} else {

nombre = invitado.getNombre();

}

tfMensaje.clear();

if (!Objects.equals(mensaje, "")) {

String mensajeEnviado = nombre + ": " + mensaje;

socket.emit("envioMensaje", mensajeEnviado);

mostrarMensaje(mensajeEnviado);

}

}

/\*\*

\* Este metodo sirve para imprimir en el TextArea del chat todos lo mensajes

\* escritos.

\* @param mensaje mensaje a enviar.

\*/

private void mostrarMensaje(String mensaje) {

mensajesChat.add(mensaje);

String chatContenido;

StringBuilder sb = new StringBuilder();

for(int i = 0; i < mensajesChat.size(); i++){

sb.append(mensajesChat.get(i)).append("\n");

}

chatContenido = sb.toString();

txtChat.setText(chatContenido);

}

@FXML

private void seleccionarA(MouseEvent event) {

bloquearRespuestas();

asignarPuntaje(respuestas.get(0));

}

@FXML

private void seleccionarB(MouseEvent event) {

bloquearRespuestas();

asignarPuntaje(respuestas.get(1));

}

@FXML

private void seleccionarC(MouseEvent event) {

bloquearRespuestas();

asignarPuntaje(respuestas.get(2));

}

@FXML

private void seleccionarD(MouseEvent event) {

bloquearRespuestas();

asignarPuntaje(respuestas.get(3));

}

/\*\*

\* Este metodo desabilita el panel que muestra al ganador.

\* @param visibilidad true si seran visibles los componentes, false si no.

\* @param activacion true si estaran desactivados, false si no.

\*/

private void activarPanelGanador(boolean visibilidad) {

pPanelGanador.setVisible(visibilidad);

pPanelGanador.setDisable(!visibilidad);

lblGanador.setDisable(!visibilidad);

lblAnunciador.setDisable(!visibilidad);

lblBuenJuego.setDisable(!visibilidad);

lblGanador.setVisible(visibilidad);

lblAnunciador.setVisible(visibilidad);

lblBuenJuego.setVisible(visibilidad);

if(visibilidad) {

mostrarNombreGanador();

}

}

/\*\*

\* Este metodo muestra el nombre del jugador que tuvo el mayor puntaje y es

\* el ganador

\*/

public void mostrarNombreGanador() {

ArrayList<Integer> puntajes = new ArrayList<>();

for(int i = 0; i < participantes.size(); i++) {

puntajes.add(participantes.get(i).getPuntaje());

}

List<String> ganador = new ArrayList<>();

Collections.max(puntajes);

for(int j = 0; j < participantes.size(); j++) {

if(puntajes.get(puntajes.size() - 1) == participantes.get(j).getPuntaje()) {

ganador.add(participantes.get(j).getNombre());

}

}

StringBuilder sb = new StringBuilder();

String winner;

if(ganador.size() > 1) {

for(int e = 0; e < ganador.size(); e++) {

sb.append(ganador.get(e)).append(", ");

}

winner = sb.toString();

} else {

winner = ganador.get(0);

}

final String ganadorText = winner;

Platform.runLater(() -> lblGanador.setText(ganadorText));

}

/\*\*

\* Este metodo deshabilita el panel que muestra la grafica de los jugadores.

\* @param visibilidad true si seran visibles los componentes, false si no.

\* @param activacion true si estaran desactivados, false si no.

\*/

private void activarPanelGrafica(boolean visibilidad) {

pPanelGrafica.setVisible(visibilidad);

bcGraficaPuntaje.setVisible(visibilidad);

pPanelGrafica.setDisable(!visibilidad);

bcGraficaPuntaje.setDisable(!visibilidad);

if (visibilidad) {

crearGrafica();

}

}

/\*\*

\* Metodo que bloquea los botones de respuestas una vez que ha respondido

\* para que no pueda volver a responder

\*/

public void bloquearRespuestas() {

imgA.setDisable(true);

imgB.setDisable(true);

imgC.setDisable(true);

imgD.setDisable(true);

chbOpcionA.setDisable(true);

chbOpcionB.setDisable(true);

chbOpcionC.setDisable(true);

chbOpcionD.setDisable(true);

}

/\*\*

\* Este metodo es para asignar el puntaje segun la respuesta elegida

\* @param respuestaSeleccionada respuesta seleccionada.

\*/

public void asignarPuntaje(RespuestaEnvio respuestaSeleccionada){

PuntajeEnvio puntajeE = new PuntajeEnvio();

puntajeE.setPuntaje(respuestaSeleccionada.getPuntaje());

String nombre;

if(cuenta instanceof Cuentausuario) {

nombre = usuario.getNombreusuario();

} else {

nombre = invitado.getNombre();

}

puntajeE.setUsuario(nombre);

Gson puntajeEnvio = new Gson();

String out = puntajeEnvio.toJson(puntajeE);

socket.emit("enviarRespuesta", out);

timer.cancel();

}

/\*\*

\* Este metodo es para crear la gráfica con las puntuaciones hasta el

\* momento de llamar la gráfica

\*/

private void crearGrafica() {

XYChart.Series datosGrafica = new XYChart.Series();

for(int i = 0; i < participantes.size(); i++) {

datosGrafica.getData().add(new XYChart.Data(

participantes.get(i).getNombre(),

participantes.get(i).getPuntaje()));

}

bcGraficaPuntaje.getData().setAll(datosGrafica);

}

/\*\*

\* Este metodo lleva la cuenta regresiva para responder una pregunta y anima

\* el label.

\* @return el segundo en que el usuario respondio la pregunta.

\*/

private void contarRegresivamente() {

timer = new Timer();

tiempoRestante = TIEMPORESPUESTA;

timer.schedule(new TimerTask() {

@Override

public void run() {

if (tiempoRestante > 0) {

Platform.runLater(() -> lblTiempo.setText(

Integer.toString(tiempoRestante)));

tiempoRestante--;

} else {

timer.cancel();

RespuestaEnvio sinSeleccion = new RespuestaEnvio();

sinSeleccion.setPuntaje(-1);

asignarPuntaje(sinSeleccion);

}

}

}, 1000, 1000);

}

/\*\*

\* Este metodo lleva la cuenta regresiva de mostrar la gráfica.

\*/

private void contarRegresivamenteGrafica() {

timerGraph = new Timer();

tiempoTotalGrafica = TIEMPOGRAFICA;

timerGraph.schedule(new TimerTask() {

@Override

public void run() {

if (tiempoTotalGrafica > 0) {

Platform.runLater(() -> activarPanelGrafica(true));

tiempoTotalGrafica--;

} else {

timerGraph.cancel();

activarPanelGrafica(false);

disminuirPreguntas();

}

}

}, 1000, 1000);

}

@FXML

private void volverAMenu(ActionEvent event) {

try {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle.getBundle(

"juegopreguntantas.lang/lang");

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("MenuPrincipal.fxml"));

loader.setResources(resourceBundle);

Parent menu = loader.load();

MenuPrincipalController controller = loader.getController();

controller.recibirParametros(cuenta, idioma);

Scene scene = new Scene(menu);

Stage stage = new Stage();

stage.setScene(scene);

stage.show();

socket.disconnect();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException ex) {

Logger.getLogger(ResponderPreguntaController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Este metodo crea la imagen en la ruta especificada.

\* @param ruta ruta de donde se creara la imagen.

\* @param imagen arreglo de bytes que compone la imagen.

\* @throws IOException

\*/

private void crearImagen(String ruta, byte[] imagen) throws IOException {

try(OutputStream out =

new BufferedOutputStream(new FileOutputStream(ruta))) {

out.write(imagen);

} catch (IOException ex) {

Logger.getLogger(ResponderPreguntaController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Este metodo desactiva el panel de espera del envio de las preguntas.

\*/

private boolean desactivarPanelEspera(boolean activacion) {

pnlEspera.setVisible(!activacion);

pnlEspera.setDisable(activacion);

pbEspera.setVisible(!activacion);

pnlEspera.setDisable(activacion);

return activacion;

}

/\*\*

\* Este metodo sirve para cambiar el puntaje del label.

\*/

private void cambiarPuntaje() {

String nombreUsuario;

if(cuenta instanceof Cuentausuario) {

nombreUsuario = usuario.getNombreusuario();

} else {

nombreUsuario = invitado.getNombre();

}

for(int i = 0; i < participantes.size(); i++) {

if(nombreUsuario.equals(participantes.get(i).getNombre())) {

String puntajeNuevo = Integer.toString(participantes.get(i)

.getPuntaje());

final String puntajeFinal = puntajeNuevo;

Platform.runLater(() -> lbPuntajeNum.setText(puntajeFinal));

}

}

}

/\*\*

\* Este metodo recibe una pregunta del objeto JSON.

\* @param preguntaRecibida Pregunta recibida en string

\*/

private void recibirPregunta(String preguntaRecibida) {

Gson preguntaGson = new Gson();

PreguntaEnvio nueva

= preguntaGson.fromJson(preguntaRecibida,

PreguntaEnvio.class);

if (nueva.getTipoPregunta() == 2) {

nueva.setImagenByte(DatatypeConverter.parseBase64Binary(nueva.getImagen()));

}

for (int i = 0; i < nueva.getRespuestas().size(); i++) {

if (nueva.getRespuestas().get(i).getTipoRespuesta() == 2) {

nueva.getRespuestas().get(i).setImagenByte(

DatatypeConverter.parseBase64Binary(

nueva.getRespuestas().get(i).getImagen()));

}

}

preguntas.add(nueva);

}

/\*\*

\* Este metodo recupera la lista de juegadores del sevidor.

\* @param recuperados JSONArray recuperado.

\*/

private void recuperarPreguntones(JSONArray recuperados) {

for (int i = 0; i < recuperados.length(); i++) {

try {

JSONObject jugador = (JSONObject) recuperados.get(i);

String nombre = jugador.getString("nombre");

JugadorConectadoEnvio nuevo

= new JugadorConectadoEnvio();

nuevo.setNombre(nombre);

nuevo.setPuntaje(0);

Platform.runLater(() -> participantes.add(nuevo));

} catch (JSONException ex) {

Logger.getLogger(ResponderPreguntaController.class

.getName()).log(Level.SEVERE, null, ex);

}

}

}

/\*\*

\* Este metodo sirve para recibir una respuesta y anadirla a la lista.

\* @param respuesta respuesta recuperada en string.

\*/

private void recibirRespuesta(String respuesta) {

Gson recibida = new Gson();

PuntajeEnvio nuevoPuntaje

= recibida.fromJson(respuesta, PuntajeEnvio.class);

int indice = 0;

for (int i = 0; i < participantes.size(); i++) {

if (participantes.get(i).getNombre().equals(nuevoPuntaje.getUsuario())) {

indice = i;

}

}

if (nuevoPuntaje.getPuntaje() == 1) {

if (contador == 0) {

participantes.get(indice).sumarPuntos(2);

} else {

participantes.get(indice).sumarPuntos(1);

}

} else {

participantes.get(indice).sumarPuntos(nuevoPuntaje.getPuntaje());

}

cambiarPuntaje();

contador++;

if (contador == participantes.size()) {

if (!creador) {

contarRegresivamenteGrafica();

} else {

crearGrafica();

}

contador = 0;

}

}

/\*\*

\* Metodo que recibe el objeto de cuenta de usuario o invitado del

\* Controlador de la pantalla de Login.

\* @param cuenta Cuenta de invitado o usuario registrado.

\* @param idioma idioma del properties.

\* @param creador metodo que valida si el jugador es el creador o no.

\*/

public void recibirParametros(Object cuenta, String idioma, boolean creador) {

Locale.setDefault(new Locale(idioma));

this.idioma = idioma;

this.cuenta = cuenta;

this.creador = creador;

if (creador) {

activacion = false;

desactivarPanelEspera(true);

activarPanelGrafica(true);

socket.emit("vaciarLista");

}

if (cuenta instanceof Cuentausuario) {

this.usuario = (Cuentausuario) cuenta;

} else {

this.invitado = (Cuentainvitado) cuenta;

}

}

}

**VentanaLogIn.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.\*?>

<?import javafx.geometry.\*?>

<?import javafx.scene.paint.\*?>

<?import javafx.scene.image.\*?>

<?import javafx.scene.layout.\*?>

<?import java.lang.\*?>

<?import javafx.scene.control.\*?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #355C7D;" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="juegopreguntantas.VentanaLogInController">

<children>

<ImageView fx:id="imgLogin" fitHeight="220.0" fitWidth="220.0" layoutX="29.0" layoutY="85.0" pickOnBounds="true" preserveRatio="true" AnchorPane.bottomAnchor="95.0" AnchorPane.leftAnchor="29.0" AnchorPane.topAnchor="85.0">

<viewport>

<Rectangle2D />

</viewport>

<image>

<Image url="@../../../../../Tecnologias/Pantallas/Login.png" />

</image>

</ImageView>

<TextField fx:id="tfUser" layoutX="301.0" layoutY="140.0" prefHeight="25.0" prefWidth="271.0" AnchorPane.leftAnchor="300.0" AnchorPane.rightAnchor="40.0" />

<Label fx:id="lbUser" layoutX="301.0" layoutY="123.0" text="%lbl.usuario" textFill="WHITE" AnchorPane.leftAnchor="301.0" />

<Label layoutX="301.0" layoutY="186.0" text="%lbl.contrasenia" textFill="WHITE" AnchorPane.leftAnchor="301.0" />

<Button fx:id="btnLogin" layoutX="501.0" layoutY="280.0" mnemonicParsing="false" onAction="#ingresar" style="-fx-background-color: #F67280;" text="%btn.login" textAlignment="CENTER" AnchorPane.rightAnchor="40.0" />

<Button fx:id="btnRegistrar" layoutX="302.0" layoutY="280.0" mnemonicParsing="false" onAction="#registrar" style="-fx-background-color: #F8B195;" text="%btn.registrar" textAlignment="CENTER" AnchorPane.leftAnchor="302.0" />

<Button fx:id="btnCerrar" layoutX="14.0" layoutY="361.0" mnemonicParsing="false" onAction="#cerrar" style="-fx-background-color: #F8B195;" text="%btn.exit" textAlignment="CENTER" AnchorPane.bottomAnchor="14.0" AnchorPane.leftAnchor="14.0" />

<Label fx:id="lTittle" layoutX="141.0" layoutY="14.0" text="%lbl.title" textAlignment="CENTER" AnchorPane.leftAnchor="141.0" AnchorPane.rightAnchor="137.0" AnchorPane.topAnchor="14.0">

<font>

<Font name="Arial Unicode MS" size="36.0" />

</font>

</Label>

<CheckBox fx:id="chbInvitado" layoutX="301.0" layoutY="244.0" mnemonicParsing="false" text="%chb.invitado" textFill="WHITE" AnchorPane.leftAnchor="301.0" />

<Label fx:id="lMensaje" layoutX="300.0" layoutY="85.0" prefHeight="17.0" prefWidth="260.0" textFill="WHITE" AnchorPane.leftAnchor="300.0" AnchorPane.rightAnchor="40.0" AnchorPane.topAnchor="85.0" />

<PasswordField fx:id="pfPassword" layoutX="302.0" layoutY="203.0" prefHeight="25.0" prefWidth="260.0" AnchorPane.leftAnchor="302.0" AnchorPane.rightAnchor="38.0" />

<ComboBox fx:id="cbCambiarIdioma" layoutX="412.0" layoutY="361.0" onAction="#cambiarIdioma" prefWidth="150.0" promptText="%cb.cambiar\_idioma" style="-fx-background-color: #FFFFFF;" />

</children>

</AnchorPane>

**Clase VentanaLogInController**

package juegopreguntantas;

import persistencia.PersistenciaCuentaUsuario;

import entity.Cuentainvitado;

import entity.Cuentausuario;

import java.io.IOException;

import java.net.URL;

import java.util.Locale;

import java.util.Objects;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.Button;

import javafx.scene.control.CheckBox;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.image.ImageView;

import javafx.stage.Stage;

import persistencia.PersistenciaCuentaInvitado;

import utilidades.UtilidadCadenas;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Eduardo Rosas Rivera \*

\* @since 26/10/2018 \*

\* Nombre de la clase VentanaLogInController \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class VentanaLogInController implements Initializable {

@FXML

private ImageView imgLogin;

@FXML

private TextField tfUser;

@FXML

private PasswordField pfPassword;

@FXML

private Label lbUser;

@FXML

private Button btnLogin;

@FXML

private Button btnRegistrar;

@FXML

private Button btnCerrar;

@FXML

private Label lTittle;

@FXML

private CheckBox chbInvitado;

@FXML

private ComboBox<String> cbCambiarIdioma;

@FXML

private Label lMensaje;

private static final String RECURSO = "juegopreguntantas.lang/lang";

private final ObservableList idiomas =

FXCollections.observableArrayList("Español", "English");

private Object cuenta;

private String idioma = "es";

/\*\*

\* Initializes the controller class.

\* @param url

\* @param rb

\*/

@Override

public void initialize(URL url, ResourceBundle rb) {

UtilidadCadenas cadena = new UtilidadCadenas();

cbCambiarIdioma.setItems(idiomas);

cadena.limitarCampos(tfUser, 26);

cadena.limitarCampos(pfPassword,36);

cadena.excluirEspacios(tfUser);

cadena.excluirEspacios(pfPassword);//tfUser|pfPassword

}

/\*\*

\* Este metodo lleva a la ventana del menu principal.

\* @param event clic en el boton Ingresar.

\* @throws IOException error al abrir ventana.

\*/

@FXML

private void ingresar(ActionEvent event) throws IOException {

if (validarCampos() && validarAcceso()) {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle.getBundle(

RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("MenuPrincipal.fxml"));

loader.setResources(resourceBundle);

Parent menu = loader.load();

MenuPrincipalController controller = loader.getController();

controller.recibirParametros(cuenta, idioma);

Scene scene = new Scene(menu);

Stage stage = new Stage();

stage.setTitle("Menu principal");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

}

}

/\*\*

\* Metodo que invoca la pantalla de registro.

\* @param event Clic en el boton Registrar.

\*/

@FXML

private void registrar(ActionEvent event) {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle = ResourceBundle.getBundle(

RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("RegistrarUsuario.fxml"));

loader.setResources(resourceBundle);

Parent registro;

try {

registro = loader.load();

RegistrarUsuarioController controller = loader.getController();

controller.setIdioma(idioma);

Scene scene = new Scene(registro);

Stage stage = new Stage();

stage.setTitle("Registro de usuario");

stage.setScene(scene);

stage.show();

((Node) (event.getSource())).getScene().getWindow().hide();

} catch (IOException ex) {

Logger.getLogger(VentanaLogInController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Metodo que cierra el programa.

\* @param event Clic en cerrar.

\*/

@FXML

private void cerrar(ActionEvent event) {

System.exit(0);

}

/\*\*

\* Metodo que valida si ambos campos fueron fueron llenados.

\* @return true si ambos estan llenos, false si alguno esta vacio.

\*/

private boolean validarCampos() {

boolean validador = true;

if(Objects.equals(tfUser.getText().trim(), "")) {

validador = false;

lMensaje.setText(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_llenarCampos"));

}

if(Objects.equals(pfPassword.getText().trim(), "")) {

validador = false;

lMensaje.setText(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_llenarCampos"));

}

return validador;

}

/\*\*

\* Metodo que valida si el nombre de usuario y la contrase;a introducidos

\* por el usuario concuerdan con alguno registrado en la base de datos.

\* @return true si los datos son validos, false si no coinciden.

\*/

private boolean validarIngresoUsuario() {

boolean ingresoExitoso = false;

String usuarioR = tfUser.getText().trim();

UtilidadCadenas cadena = new UtilidadCadenas();

String contrasenia =

cadena.hacerHashAContrasenia(pfPassword.getText().trim());

PersistenciaCuentaUsuario usuarioBD = new PersistenciaCuentaUsuario();

try {

Cuentausuario usuario = usuarioBD.getCuentaUsuarioNombre(usuarioR

.toUpperCase());

if (usuario != null) {

if (contrasenia.equals(usuario.getContrasenia())) {

cuenta = usuario;

ingresoExitoso = true;

lMensaje.setText(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_iniciandoSesion"));

} else {

lMensaje.setText(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_incorrectos"));

}

} else {

lMensaje.setText(java.util.ResourceBundle.getBundle(RECURSO)

.getString("string\_noEncontrado"));

}

} catch (Exception ex) {

mostrarMensajeError();

Logger.getLogger(VentanaLogInController.class.getName())

.log(Level.SEVERE, null, ex);

}

return ingresoExitoso;

}

/\*\*

\* Metodo que valida si el nombre de usuario y la contrase;a

\* introducidos por el usuario

\* consuerdan con alguno de invitado registrado en la base de datos

\* @return true si los datos son validos, false si no coinciden.

\*/

private boolean validarIngresoInvitado() {

boolean ingresoExitoso = false;

String usuario = tfUser.getText().trim();

String contrasenia = pfPassword.getText().trim();

try {

PersistenciaCuentaInvitado persistencia

= new PersistenciaCuentaInvitado();

Cuentainvitado invitado

= persistencia.getCuentaInvitado(usuario.toUpperCase());

if (invitado != null) {

if (contrasenia.equals(invitado.getCodigo())) {

ingresoExitoso = true;

cuenta = invitado;

lMensaje.setText(java.util.ResourceBundle.getBundle(

RECURSO)

.getString("string\_iniciandoSesion"));

} else {

lMensaje.setText(java.util.ResourceBundle.getBundle(

RECURSO)

.getString("string\_incorrectos"));

}

} else {

lMensaje.setText(java.util.ResourceBundle.getBundle(

RECURSO)

.getString("string\_noEncontrado"));

}

} catch (Exception ex) {

Logger.getLogger(VentanaLogInController.class.getName())

.log(Level.SEVERE, null, ex);

mostrarMensajeError();

}

return ingresoExitoso;

}

/\*\*

\* Metodo que valida si el usuario se trata de una cuenta invitada o

\* una cuenta registrada.

\* @return true si el usuario esta en la base de datos.

\*/

private boolean validarAcceso() {

boolean ingresoExitoso = false;

if(chbInvitado.isSelected()) {

if(validarIngresoInvitado()) {

ingresoExitoso = true;

}

} else {

if(validarIngresoUsuario()) {

ingresoExitoso = true;

}

}

return ingresoExitoso;

}

/\*\*

\* Este metodo sirve para cambiar el idioma de la ventana.

\* @param event Clic en una opcion del cdIdioma.

\*/

@FXML

private void cambiarIdioma(ActionEvent event) {

String idiomaLogin;

if(cbCambiarIdioma.getSelectionModel().getSelectedItem()

.equals("English")) {

idiomaLogin = "en";

} else {

idiomaLogin = "es";

}

abrirLogin(idiomaLogin, event);

}

/\*\*

\* Este metodo sirve para abrir la ventana con un nuevo idioma.

\* @param idioma idioma con el que se abrira la ventana.

\* @param event Clic en cdIdioma.

\*/

private void abrirLogin(String idioma, ActionEvent event) {

Locale.setDefault(new Locale(idioma));

ResourceBundle resourceBundle =

ResourceBundle.getBundle(RECURSO);

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("VentanaLogIn.fxml"));

loader.setResources(resourceBundle);

try {

Parent logIn = loader.load();

VentanaLogInController controller = loader.getController();

controller.setIdioma(idioma);

Scene scene = new Scene(logIn);

Stage stage = new Stage();

stage.setScene(scene);

stage.show();

((Node) event.getSource()).getScene().getWindow().hide();

} catch (IOException ex) {

Logger.getLogger(VentanaLogInController.class.getName())

.log(Level.SEVERE, null, ex);

}

}

/\*\*

\* Este metodo sirve para fijar solo el idioma con el que se abrira la

\* ventana.

\* @param idioma idioma con el que iniciara la ventana.

\*/

public void setIdioma(String idioma) {

this.idioma = idioma;

}

/\*\*

\* Este metodo muestra un mensaje de error en caso de fallo.

\*/

private void mostrarMensajeError() {

Alert alert = new Alert(AlertType.ERROR);

alert.setHeaderText("Error");

alert.setTitle("Erro de conexion.");

alert.setContentText("Se ha perdido la conexion con la base de datos.");

}

}

**Clase Persistenia**

package persistencia;

import java.util.HashMap;

import java.util.Map;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* @version 1.0 \*/

/\* @author Puxka Acosta Domínguez Eduardo Rosas Rivera \*/

/\* @since 07/11/2018 \*/

/\* Nombre de la clase PersistenciaCuentaInvitado \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class Persistencia {

/\*\*

\* Este metodo es para trabajar con las entidades de la base de datos

\* @return El EntityManager

\*/

public EntityManager administrarEntidades() {

Map<String, String> properties = new HashMap<>();

properties.put("javax.persistence.jdbc.user", "pregunton");

properties.put("javax.persistence.jdbc.password", "PR3GUNT0N");

EntityManagerFactory emf = javax.persistence.Persistence

.createEntityManagerFactory("JuegoPreguntantasPU", properties);

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

return em;

}

}

**Clase PersistenciaCategoria**

package persistencia;

import entity.Categoria;

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.persistence.EntityManager;

import javax.persistence.Query;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 26/10/2018 \*

\* Nombre de la clase PersistenciaCategoria \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class PersistenciaCategoria extends Persistencia {

/\*\*

\* Este metodo es para recuperar las categorias deacuerdo a un id

\* @param idCategoria El id de una categoria

\* @return La lista de categorias

\*/

public List<String> recuperarCategoriasSet(List<Integer> idCategoria) {

EntityManager em = administrarEntidades();

List<String> categorias = new ArrayList<>();

try {

for(int i = 0; i < idCategoria.size(); i++) {

Query query = em.createNamedQuery("Categoria.findCategoriaById",

Categoria.class).setParameter("idcategoria", idCategoria.get(i));

categorias.add((i + 1) + ".- " + query.getResultList().get(0).toString());

}

} catch (NullPointerException e) {

Logger.getLogger(PersistenciaCategoria.class.getName())

.log(Level.SEVERE, null, e);

}

return categorias;

}

/\*\*

\* Este metodo es para recuperar todas las categorias de la base de datos

\* @return Lista con las categorias

\*/

public List<String> recuperarCategorias() {

EntityManager em = administrarEntidades();

List<String> categorias = new ArrayList<>();

try {

Query query = em.createNamedQuery("Categoria.findAllCategoria",

Categoria.class);

categorias = query.getResultList();

} catch (NullPointerException e) {

Logger.getLogger(PersistenciaCategoria.class.getName())

.log(Level.SEVERE, null, e);

}

return categorias;

}

/\*\*

\* Este metodo es para recuperar el id de una categoria por medio de su

\* nombre

\* @param categoria La categoria

\* @return El id de la categoria

\*/

public int recuperarIdCategoria(String categoria) {

EntityManager em = administrarEntidades();

List<Integer> categorias;

int idCategoria = 0;

try {

Query query = em.createNamedQuery("Categoria.findIdByCategoria",

Categoria.class).setParameter("categoria", categoria);

categorias = query.getResultList();

if(!categorias.isEmpty()) {

idCategoria = categorias.get(0);

}

} catch (NullPointerException e) {

Logger.getLogger(PersistenciaCategoria.class.getName())

.log(Level.SEVERE, null, e);

}

return idCategoria;

}

}

**Clase PersistenciaCuentaInvitado**

package persistencia;

import entity.Cuentainvitado;

import java.util.List;

import java.util.Random;

import java.util.UUID;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.persistence.NoResultException;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Query;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* @version 1.0 \*/

/\* @author Puxka Acosta DomÃ-nguez y Eduardo Rosas Rivera \*/

/\* @since 07/11/2018 \*/

/\* Nombre de la clase PersistenciaCuentaInvitado \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class PersistenciaCuentaInvitado extends Persistencia {

private static final String PERSISTENCIA = "JuegoPreguntantasPU";

/\*\*

\* Este metodo es para crear una cuenta de invitado en la base de datos

\* @param nuevoInvitado Cuenta de invitado a insertar

\* @return Si es verdadero o no el exito de la creacion de la cuenta

\*/

public boolean crearInvitado(Cuentainvitado nuevoInvitado) {

boolean exito = false;

EntityManager em = administrarEntidades();

try {

em.persist(nuevoInvitado);

em.getTransaction().commit();

exito = true;

} catch (Exception e) {

Logger.getLogger(PersistenciaCuentaInvitado.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

} finally {

em.close();

}

return exito;

}

/\*

\* Este metodo es para crear el nombre al invitado

\* @return El nombre de la cuenta de invitado

\*/

public String crearNombre() {

EntityManager em = administrarEntidades();

String nombre = null;

try {

int id = (int) em.createNamedQuery("Cuentainvitado.findMaximo").getFirstResult();

nombre = "Pregunton" + (id + 1);

} catch(NullPointerException e) {

Logger.getLogger(PersistenciaCuentaInvitado.class.getName())

.log(Level.SEVERE, null, e);

nombre = "Pregunton0";

} finally {

nombre = comprobarNombre(nombre);

}

return nombre;

}

/\*\*

\* Este metodo es para crear el codigo que usara de contraseÃ±a el invitado

\* @return El codigo de la cuenta de invitado

\*/

public String crearCodigo() {

String codigo = UUID.randomUUID().toString();

if (comprobarCodigo(codigo)) {

crearCodigo();

}

return codigo;

}

/\*\*

\* Este metodo es para comprobar que el correo que se ha ingresado es

\* diferente al que se ha invitado antes

\* @param correoElectronico Correo electronico ingresado por el usuario

\* @return Si es verdadero o no la repeticion del correo electronico en la

\* base de datos

\*/

public boolean comprobarCorreo(String correoElectronico) {

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery(

"Cuentainvitado.findAllByCorreoelectronico")

.setParameter("correoelectronico", correoElectronico);

List<String> correoRepetido = query.getResultList();

return !correoRepetido.isEmpty();

}

/\*\*

\* Este metodo es para comprobar que el codigo es diferente al de otra cuenta

\* de invitado

\* @param codigo El codigo creado

\* @return Si es verdadero o no la repeticion del codigo en la base de datos

\*/

public boolean comprobarCodigo(String codigo) {

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery("Cuentainvitado.findAllByCodigo")

.setParameter("codigo", codigo);

List<String> codigoRepetido = query.getResultList();

boolean datoRepetido = !codigoRepetido.isEmpty();

return datoRepetido;

}

/\*\*

\* Este metodo es para comprobar que el nombre es diferente al de una cuenta

\* de usuario, agregando un caracter en caso de que lo sea

\* @param nombre El nombre de la cuenta de invitado

\* @return El nombre de la cuenta de invitado diferente al de una cuenta

\* de usuario

\*/

public String comprobarNombre(String nombre) {

PersistenciaCuentaUsuario usuarioRegistrado

= new PersistenciaCuentaUsuario();

if (!usuarioRegistrado.comprobarNombreDiferente(nombre)) {

nombre = nombre + (char) (48 + new Random().nextInt(47));

comprobarNombre(nombre);

}

return nombre;

}

/\*\*

\* Este metodo es para eliminar una cuenta de invitado en la base de datos

\* @param invitado Cuenta de invitado a eliminar

\* @return true si se realizo exitosamente, false sino.

\*/

public boolean eliminarInvitado(Cuentainvitado invitado) {

boolean exito = false;

EntityManager em = administrarEntidades();

try {

if (!em.contains(invitado)) {

invitado = em.merge(invitado);

}

em.remove(invitado);

em.getTransaction().commit();

exito = true;

} catch (Exception e) {

Logger.getLogger(PersistenciaCuentaInvitado.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

} finally {

em.close();

}

return exito;

}

/\*\*

\* Constructor de la clase.

\*/

public PersistenciaCuentaInvitado() {

//vacio para instanciacion

}

/\*\*

\* Metodo que registra una Cuentainvitado en la base de datos.

\* @param ci Cuenta invitado

\*/

public void registrarCuentaInvitado(Cuentainvitado ci) {

PersistenciaCuentaInvitado persistencia =

new PersistenciaCuentaInvitado();

persistencia.persist(ci);

}

/\*\*

\* Metodo que guarda objetos en la base de datos.

\* @param object objeto a guadar en la base de datos.

\*/

public void persist(Object object) {

EntityManagerFactory emf = javax.persistence.Persistence.

createEntityManagerFactory(PERSISTENCIA);

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

try {

em.persist(object);

em.getTransaction().commit();

} catch (Exception e) {

Logger.getLogger(PersistenciaCuentaInvitado.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

} finally {

em.close();

}

}

/\*\*

\* Metodo usado para eliminar una cuenta de un usuario registrado en la base

\* de datos.

\* @param id identificador de la cuenta a eliminar.

\*/

public void destroyCuentaInvitado(long id) {

EntityManagerFactory emf = javax.persistence.Persistence.

createEntityManagerFactory(PERSISTENCIA);

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

try {

Cuentainvitado cu= em.find(Cuentainvitado.class, id);

em.remove(cu);

em.getTransaction().commit();

} catch (Exception e) {

Logger.getLogger(PersistenciaCuentaInvitado.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

} finally {

em.close();

}

}

/\*\*

\* Metodo que busca el nombre ingresado por el usuario en la base de datos

\* de la tabla de invitados.

\* @param nombreinvitado nombre ingresado por el usuario en la pantalla de

\* login.

\* @return true si la cuenta de tipo invitado existe en la base de datos,

\* false si no existe.

\*/

public Cuentainvitado getCuentaInvitado(String nombreinvitado) {

EntityManagerFactory emf = javax.persistence.Persistence.

createEntityManagerFactory(PERSISTENCIA);

EntityManager em = emf.createEntityManager();

Cuentainvitado invitado = new Cuentainvitado();

try {

invitado = em.createNamedQuery(

"Cuentainvitado.findByNombre", Cuentainvitado.class)

.setParameter("nombre", nombreinvitado).getSingleResult();

} catch (NoResultException noResult) {

invitado = null;

} finally {

em.close();

}

return invitado;

}

}

**Clase PersistenciaCuentaUsuario**

package persistencia;

import entity.Cuentausuario;

import javax.persistence.NoResultException;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Query;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* @version 1.0

/\* @author Puxka Acosta DomÃ-nguez y Eduardo Rosas Rivera

/\* @since 07/11/2018

/\* Nombre de la clase EnviarInvitacionController

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class PersistenciaCuentaUsuario extends Persistencia {

private static final String PERSISTENCIA = "JuegoPreguntantasPU";

/\*\*

\* Este metodo es para comprobar que el nombre de invitado no es el mismo

\* que el de una cuenta de usuario

\* @param nombreInvitado El nombre de la cuenta de invitado

\* @return Si es verdadero o no la repeticion del nombre en la base de datos

\*/

public boolean comprobarNombreDiferente(String nombreInvitado) {

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery("Cuentainvitado.findAllByNombre")

.setParameter("nombre", nombreInvitado);

List<String> nombreInvitadoRepetido = query.getResultList();

return nombreInvitadoRepetido.isEmpty();

}

/\*\*

\* Este metodo recupera una cuentausuario por su id.

\* @param id id de la cuenta a recuperar.

\* @return cuentausuario correspondiente al id.

\*/

public Cuentausuario recuperarUsuario(int id){

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery("Cuentausuario.findByIdcuentausuario",

Cuentausuario.class).setParameter("idcuentausuario", id);

List<Cuentausuario> usuario = query.getResultList();

return usuario.get(0);

}

public PersistenciaCuentaUsuario() {}

/\*\*

\* Este metodo es para crear una cuenta de usuario en la base de datos

\* @param nuevoUsuario Cuenta de usuario a registrar

\* @return Si es verdadero o no el exito de la creacion de la cuenta

\*/

public boolean crearUsuario(Cuentausuario nuevoUsuario) {

boolean exito = false;

EntityManager em = administrarEntidades();

try {

em.persist(nuevoUsuario);

em.getTransaction().commit();

exito = true;

} catch (Exception e) {

Logger.getLogger(PersistenciaCuentaUsuario.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

} finally {

em.close();

}

return exito;

}

/\*\*

\* Metodo usado para eliminar una cuenta de un usuario registrado en la

\* base de datos.

\* @param id identificador de la cuenta a eliminar.

\*/

public void destroyCuentaUsuario(long id) {

EntityManager em = administrarEntidades();

em.getTransaction().begin();

try {

Cuentausuario cu= em.find(Cuentausuario.class, id);

em.remove(cu);

em.getTransaction().commit();

} catch (Exception e) {

Logger.getLogger(PersistenciaCuentaUsuario.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

} finally {

em.close();

}

}

/\*\*

\* Este metodo es para eliminar una cuenta de invitado en la base de datos

\* @param usuario Cuenta de usuario a eliminar

\* @return true si se elimino correctamente

\*/

public boolean eliminarUsuario(Cuentausuario usuario) {

EntityManager em = administrarEntidades();

boolean exito = false;

try {

if (!em.contains(usuario)) {

usuario = em.merge(usuario);

}

em.remove(usuario);

em.getTransaction().commit();

exito = true;

} catch (Exception e) {

Logger.getLogger(PersistenciaCuentaUsuario.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

}

return exito;

}

/\*\*

\* Metodo que busca un usuario en la base de datos con el nombre que

\* el usuario ingrese.

\* @param nombreusuario en el TextField de la pantalla del Login.

\* @return true si existe en la base de datos, false si no existe.

\*/

public Cuentausuario getCuentaUsuarioNombre(String nombreusuario) {

EntityManagerFactory emf = javax.persistence.Persistence.

createEntityManagerFactory(PERSISTENCIA);

EntityManager em = emf.createEntityManager();

Cuentausuario cuenta;

try {

cuenta = (Cuentausuario) em.createNamedQuery(

"Cuentausuario.findByNombreusuario",

Cuentausuario.class)

.setParameter("nombreusuario", nombreusuario)

.getSingleResult();

} catch (NoResultException noResult) {

cuenta = null;

}

return cuenta;

}

/\*\*

\* Este metodo recupera al usuario con su email.

\* @param email email del usuario a buscar.

\* @return el usuario a buscar, null si no hay ninguna ocurrencia.

\*/

public Cuentausuario getCuentaUsuarioEmail(String email) {

EntityManagerFactory emf = javax.persistence.Persistence.

createEntityManagerFactory(PERSISTENCIA);

EntityManager em = emf.createEntityManager();

Cuentausuario cuenta;

try {

cuenta = (Cuentausuario) em.createNamedQuery(

"Cuentausuario.findByCorreoelectronico",

Cuentausuario.class)

.setParameter("correoelectronico", email)

.getSingleResult();

} catch (NoResultException noResult) {

cuenta = null;

}

return cuenta;

}

}

**Clase PersistenciaPartida**

package persistencia;

import entity.Partida;

import entity.Setpregunta;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.persistence.EntityManager;

import javax.persistence.Query;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 10/11/2018 \*

\* Nombre de la clase PersistenciaPartida \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class PersistenciaPartida extends Persistencia {

/\*\*

\* Este metodo es para crear una paertida en la base de datos

\* @param nombre nombre de la partida.

\* @param idSetPregunta ide del set.

\* @return Si es verdadero o no el exito de la creacion de la partida

\*/

public boolean crearPartida(String nombre, Setpregunta idSetPregunta) {

boolean exito = false;

EntityManager em = administrarEntidades();

Partida nuevaPartida = new Partida();

nuevaPartida.setIdsetpregunta(idSetPregunta);

nuevaPartida.setNombre(nombre);

try {

em.persist(nuevaPartida);

em.getTransaction().commit();

exito = true;

} catch (Exception e) {

Logger.getLogger(PersistenciaPartida.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

} finally {

em.close();

}

return exito;

}

/\*\*

\* Este metodo es para recuperar los nombres de las partidas del set de

\* preguntas seleccionada anteriormente

\* @param setPregunta Set de pregunta seleccionado

\* @return La lista de nombres de las partidas

\*/

public List<String> recuperarNombre(Setpregunta setPregunta) {

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery("Partida.findBySet")

.setParameter("idsetpregunta", setPregunta.getIdsetpregunta());

List<String> nombres = query.getResultList();

return nombres;

}

/\*\*

\* Este metodo es para recuperar una partida

\* @param setPregunta Setpregunta que tiene una partida

\* @return Una partida

\*/

public List<Partida> recuperarPartida(Setpregunta setPregunta) {

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery("Partida.findByIdset", Partida.class)

.setParameter("idsetpregunta", setPregunta.getIdsetpregunta());

return query.getResultList();

}

/\*\*

\* Metodo usado para eliminar una partida registrada en la base de datos.

\* @param id identificador de la partida a eliminar.

\*/

public void destroyPartida(int id) {

EntityManager em = administrarEntidades();

try {

Partida partida = em.find(Partida.class, id);

em.remove(partida);

em.getTransaction().commit();

} catch (Exception e) {

Logger.getLogger(PersistenciaPartida.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

} finally {

em.close();

}

}

}

**Clase PersistenciaPregunta**

package persistencia;

import entity.Pregunta;

import entity.Respuesta;

import java.util.Collection;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.persistence.EntityManager;

import javax.persistence.Query;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 17/11/2018 \*

\* Nombre de la clase PersistenciaPregunta \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class PersistenciaPregunta extends Persistencia {

/\*\*

\* Este metodo es para crear una pregunta en la base de datos

\* @param nuevaPregunta Pregunta a insertar

\* @return Si es verdadero o no el exito de la creacion de la pregunta

\*/

public boolean crearPregunta(Pregunta nuevaPregunta) {

boolean exito = false;

EntityManager em = administrarEntidades();

PersistenciaRespuesta respuestaBD = new PersistenciaRespuesta();

Collection<Respuesta> respuestas =

nuevaPregunta.getRespuestaCollection();

nuevaPregunta.setRespuestaCollection(null);

try {

em.persist(nuevaPregunta);

em.getTransaction().commit();

if (respuestaBD.crearRespuesta(respuestas,

nuevaPregunta)) {

exito = true;

}

} catch (Exception e) {

Logger.getLogger(PersistenciaPregunta.class.getName())

.log(Level.SEVERE, null, e);

em.getTransaction().rollback();

}

return exito;

}

/\*\*

\* Este metodo es para guardar la relacion con el set de preguntas

\* @param numPreguntas numero de preguntas.

\*/

public void guardarSetPregunta(int numPreguntas){

EntityManager em = administrarEntidades();

PersistenciaSetpregunta setPreguntaBD = new PersistenciaSetpregunta();

int idSetPregunta = setPreguntaBD.recuperarUltimaId();

Query query = em.createNamedQuery("Pregunta.findAll", Pregunta.class);

List<Pregunta> preguntas = query.getResultList();

int indice = preguntas.size() - 1;

int limite = preguntas.size() - numPreguntas;

for (int i = indice; i > limite; i--) {

em = administrarEntidades();

Pregunta pregunta = em.find(Pregunta.class,

preguntas.get(i).getIdpregunta());

pregunta.setIdsetpregunta(idSetPregunta);

em.getTransaction().commit();

}

}

/\*\*

\* Este metodo es para recuperar las preguntas de un set de preguntas;

\* @param idSetPregunta El id del set de preguntas

\* @return Una lista de preguntas del set de preguntas

\*/

public List<Pregunta> recuperarPregunta(int idSetPregunta) {

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery("Pregunta.findByIdsetpregunta",

Pregunta.class).setParameter("idsetpregunta", idSetPregunta);

return query.getResultList();

}

/\*\*

\* Este metodo recupera todas las preguntas junto con las respuestas, de un

\* set de preguntas.

\* @param idSetPregunta id del set a recuperar.

\* @return Lista de preguntas con sus respuestas, del set.

\*/

public List<Pregunta> recuperarPreguntaConRespuestas(int idSetPregunta) {

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery("Pregunta.findByIdsetpregunta",

Pregunta.class).setParameter("idsetpregunta", idSetPregunta);

List<Pregunta> preguntas = query.getResultList();

for(int i = 0; i < preguntas.size(); i++) {

PersistenciaRespuesta respuestaBD = new PersistenciaRespuesta();

List<Respuesta> respuestas;

respuestas = respuestaBD.recuperarRespuestasDePregunta(

preguntas.get(i).getIdpregunta());

preguntas.get(i).setRespuestas(respuestas);

}

return preguntas;

}

}

**Clase PersistenciaRespuesta**

package persistencia;

import entity.Pregunta;

import entity.Respuesta;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.persistence.EntityManager;

import javax.persistence.Query;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 12/11/2018 \*

\* Nombre de la clase PersistenciaRespuesta \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class PersistenciaRespuesta extends Persistencia {

/\*\*

\* Este metodo es para crear una pregunta en la base de datos

\* @param respuestas Respuesta a insertar

\* @param pregunta La pregunta a la que pertenecen las preguntas

\* @return Si es verdadero o no el exito de la creacion de la respuesta

\*/

public boolean crearRespuesta(Collection<Respuesta> respuestas,

Pregunta pregunta) {

boolean exito = false;

List<Respuesta> nuevaRespuesta = new ArrayList();

nuevaRespuesta.addAll(respuestas);

try {

for (int i = 0; i < nuevaRespuesta.size(); i++) {

EntityManager em = administrarEntidades();

nuevaRespuesta.get(i).setIdpregunta(pregunta);

em.persist(nuevaRespuesta.get(i));

em.getTransaction().commit();

}

exito = true;

} catch (Exception e) {

Logger.getLogger(PersistenciaRespuesta.class.getName())

.log(Level.SEVERE, null, e);

}

return exito;

}

/\*\*

\* Este metodo es para recuperar las respuestas de preguntas

\* @param preguntas La preguntas de la que se van a recuperar sus respuestas

\* @return Una lista de respuestas de las preguntas

\*/

public List<Respuesta> recuperarRespuesta(List<Pregunta> preguntas){

List<Respuesta> respuestas = new ArrayList<>();

List<Respuesta> resultadoConsulta;

Query query;

EntityManager em = administrarEntidades();

for (int i = 0; i < preguntas.size(); i++) {

query = em.createNamedQuery("Respuesta.findByIdpregunta",

Respuesta.class).setParameter("idpregunta",

preguntas.get(i).getIdpregunta());

resultadoConsulta = query.getResultList();

for (int j = 0; j < resultadoConsulta.size(); j++) {

respuestas.add(resultadoConsulta.get(j));

}

}

return respuestas;

}

/\*\*

\* Este metodo recupera todas las respuestas de una pregunta.

\* @param idPregunta id de la pregunta a recuperar.

\* @return lista de respuestas de solo una pregunta.

\*/

public List<Respuesta> recuperarRespuestasDePregunta(int idPregunta) {

EntityManager em = administrarEntidades();

List<Respuesta> respuestas;

Query query = em.createNamedQuery("Respuesta.findByIdpregunta",

Respuesta.class).setParameter("idpregunta", idPregunta);

respuestas = query.getResultList();

return respuestas;

}

}

**Clase PersistenciaSetpregunta**

package persistencia;

import entity.Cuentausuario;

import entity.Pregunta;

import entity.Setpregunta;

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.persistence.EntityManager;

import javax.persistence.Query;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @version 1.0 \*

\* @author Puxka Acosta Domínguez y Eduardo Rosas Rivera \*

\* @since 12/11/2018 \*

\* Nombre de la clase PersistenciaSetpregunta \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public class PersistenciaSetpregunta extends Persistencia {

/\*\*

\* Este metodo es para recuperar las categorias de los set de preguntas que

\* haya en la base de datos

\* @param usuario Cuenta del usuario que esta usando el juego

\* @return La lista de categorias de los set de pregunta

\*/

public List<String> recuperarCategorias(Cuentausuario usuario) {

EntityManager em = administrarEntidades();

PersistenciaCategoria categoriasSet = new PersistenciaCategoria();

List<String> categorias = new ArrayList<>();

try {

Query query = em.createNamedQuery("Setpregunta.findIdByIdCuentaUsuario",

Setpregunta.class).setParameter("idcuentausuario",

usuario.getIdcuentausuario());

List<Integer> idCategoria = query.getResultList();

categorias = categoriasSet.recuperarCategoriasSet(idCategoria);

} catch (NullPointerException e) {

Logger.getLogger(PersistenciaSetpregunta.class.getName())

.log(Level.SEVERE, null, e);

} finally {

em.close();

}

return categorias;

}

/\*\*

\* Este metodo es para recuperar los set de preguntas del usuario creador

\* @param usuario Cuenta del usuario que esta usando el juego

\* @return una lista de los sets de preguntas del usuario.

\*/

public List<Setpregunta> recuperarSetPregunta(Cuentausuario usuario) {

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery("Setpregunta.findAllByIdCuentaUsuario",

Setpregunta.class).setParameter("idcuentausuario",

usuario.getIdcuentausuario());

return query.getResultList();

}

/\*\*

\* Este metodo es para recuperar un set de preguntas por su categoria

\* @param categoria Categoria del set de pregunta

\* @return Un set de preguntas

\*/

public Setpregunta recuperarSetPregunta(String categoria) {

EntityManager em = administrarEntidades();

String[] idSetPregunta = categoria.split(".- ");

Query query = em.createNamedQuery("Setpregunta.findAllByIdsetpregunta",

Setpregunta.class).setParameter("idsetpregunta",

Integer.parseInt(idSetPregunta[0]));

List<Setpregunta> setPregunta = query.getResultList();

return setPregunta.get(0);

}

/\*\*

\* Este metodo es para crear un set de preguntas

\* @param categoria La categoria del set

\* @param cuenta La cuenta del usuario creador del set de preguntas

\* @param preguntas Las preguntas del set de preguntas

\* @return Si la creacion es exitosa o no

\*/

public boolean crearSetPregunta(String categoria, Cuentausuario cuenta,

List<Pregunta> preguntas) {

boolean exito = false;

EntityManager em = administrarEntidades();

PersistenciaCategoria categoriaBD = new PersistenciaCategoria();

PersistenciaPregunta preguntaBD = new PersistenciaPregunta();

Setpregunta setPregunta = new Setpregunta();

setPregunta.setIdcuentausuario(cuenta);

try {

int idCategoria = categoriaBD.recuperarIdCategoria(categoria);

setPregunta.setIdcategoria(idCategoria);

em.persist(setPregunta);

em.getTransaction().commit();

for(int i = 0; i < preguntas.size(); i++){

preguntas.get(i).setIdsetpregunta(recuperarUltimaId());

if(preguntaBD.crearPregunta(preguntas.get(i))){

exito = true;

}

}

} catch (NullPointerException e) {

Logger.getLogger(PersistenciaSetpregunta.class.getName())

.log(Level.SEVERE, null, e);

} finally {

em.close();

}

return exito;

}

/\*\*

\* Este metodo es para recuperar la id del último set de preguntas

\* @return El id del set de preguntas

\*/

public int recuperarUltimaId(){

EntityManager em = administrarEntidades();

int indice;

try {

Query query = em.createNamedQuery("Setpregunta.findMaxIdsetpregunta",

Setpregunta.class);

indice = (int) query.getSingleResult();

} catch(NullPointerException e) {

indice = 0;

}

return indice;

}

/\*\*

\* Este metodo recupera todos los sets de pregunta que concuerden con un

\* mismo id de una categoria.

\* @param idCategoria id de la categoria de interes.

\* @return una lista de setpregunta con la misma id categoria.

\*/

public List<Setpregunta> recuperarSetCategoria(int idCategoria) {

List<Setpregunta> setsPregunta;

EntityManager em = administrarEntidades();

Query query = em.createNamedQuery("Setpregunta.findAllByIdcategoria",

Setpregunta.class).setParameter("idcategoria", idCategoria);

setsPregunta = query.getResultList();

em.close();

return setsPregunta;

}

}

**Clase UtilidadCadenas**

package utilidades;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.beans.value.ObservableValue;

/\*\*

\*

\* @author Eduar

\*/

public class UtilidadCadenas {

/\*\*

\* Este metodo convierte la contrasenia ingresada por el usuario en una

\* cadena hash.

\* @param contrasenia contrasenia a hacer hash.

\* @return el codigo hash de la contrasenia.

\*/

public String hacerHashAContrasenia(String contrasenia) {

String contraseniaHash = null;

try {

MessageDigest md = MessageDigest.getInstance("SHA-512");

byte[] messageDigest = md.digest(contrasenia.getBytes());

BigInteger codigo = new BigInteger(1, messageDigest);

contraseniaHash = codigo.toString(16);

while (contraseniaHash.length() < 32) {

contraseniaHash = new StringBuilder().append("0").

append(contraseniaHash).toString();

}

} catch (NoSuchAlgorithmException ex) {

Logger.getLogger(UtilidadCadenas.class.getName())

.log(Level.SEVERE, null, ex);

}

return contraseniaHash;

}

/\*\*

\* Este metodo impide que el campo de texto sea mayor a un numero de

\* caracteres fijo.

\* @param tf textField a limitar

\* @param maximo numero maximo de caracteres permitidos.

\*/

public void limitarCampos(javafx.scene.control.TextField tf, int maximo) {

tf.textProperty().addListener((final ObservableValue<? extends

String> ov, final String oldValue, final String newValue) -> {

if (tf.getText().length() > maximo) {

String s = tf.getText().substring(0, maximo);

tf.setText(s);

}

});

}

/\*\*

\* Este metodo sirve para que los textField nieguen la entrada a espacios.

\* @param tf textField a limitar

\*/

public void excluirEspacios(javafx.scene.control.TextField tf) {

tf.textProperty().addListener(

(observable, oldValue, newValue) -> {

if (newValue.contains(" ")) {

tf.setText(oldValue);

}

});

}

/\*\*

\* Este metodo sirve para que los textFiel nieguen de los simbolos de

\* interrogación.

\* @param tf textField a limitar

\*/

public void excluirSimbPregunta(javafx.scene.control.TextField tf) {

tf.textProperty().addListener(

(observable, oldValue, newValue) -> {

if (newValue.contains("¿") || newValue.contains("?")) {

tf.setText(oldValue);

}

});

}

}