UNIVERSIDAD AUTÓNOMA DE SAN LUÍS POTOSÍ
FACULTAD DE INGENIERÍA

PROYECTO:

SISTEMA DE PUNTO DE VENTA PARA UNA GALERIA DE ARTE.

MANUAL DEL PROGRAMADOR

REALIZADO POR:
MIGUEL ANGEL GALICIA TORREZ

MATERIA:
INGENIERIA DE SOFTWARE B

PROFESOR:
ING. ALICIA ARRIETA VITA

21 de mayo de 2013

Documentación de las clases

Referencia de la Clase ArtistasExposicion

Métodos públicos

- **ArtistasExposicion** ()
- **ArtistasExposicion** (String u, String p, long id)
- void **CargaArtistas** ()
- void **InsertArtExpo** ()
- void **DeleteArtistaExp** ()

Métodos públicos estáticos

- static void **main** (String args[])

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Clase implementada para controlar los artistas que participaran en alguna exposición.

Documentación del constructor y destructor

galeriadearte.ArtistasExposicion (String *u*, String *p*, long *id*)

Parámetros:

u	Nombre del usuario
p	Contraseña del usuario
id	Codigo de la exposicion

```
                                                {

        initComponents();

        con = new Conexion(u, p, ArtistasExposicion,"select * from EXPOSICIONES.det_artistas_exposicion C where
C.id_exposicion="+id);

        this.etIdExp.setText(Long.toString(id));
        this.setLocationRelativeTo(this);
        this.Artistas = new int[200];
        this.CargaArtistas();
        this.id=id;
    }
```

Documentación de las funciones miembro

void galeriadearte.ArtistasExposicion.CargaArtistas ()

Este método se encarga de inicializar un control ComboBox con los nombres de los artistas y a su vez guarda la clave primaria de cada uno de ellos en un arreglo.

```
                                    {

        numArtistas = 0;

        try
        {
              con.EstableceConexion();
              rsArtistas = con.getSentencia().executeQuery("select A.Id_Artista, (A.Nombre||' '||A.Apellidos) AS Nombre
from ARTISTAS.Artistas A");

              while(rsArtistas.next())
              {
                 cbArtistas.addItem(rsArtistas.getObject(2));
                 Artistas[numArtistas++] = Integer.parseInt( String.valueOf(rsArtistas.getObject(1)));
              }

              con.cierraConexion();
        }catch(Exception e)
        {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
    }
```

void galeriadearte.ArtistasExposicion.DeleteArtistaExp ()

Método para eliminar un artista en alguna exposición

```
                                    {

        int opc;
        int fila=this.ArtistasExposicion.getSelectedRow();

        //Se obtiene el codigo de la exposición
        long idV=Long.parseLong(String.valueOf(this.ArtistasExposicion.getModel().getValueAt(fila, 1)));

        if(ArtistasExposicion.getSelectedRow() >= 0)
        {
            opc = JOptionPane.showConfirmDialog(null, "¿Esta seguro que quiere eliminar el registro?", "Eliminación",
JOptionPane.YES_NO_OPTION);
              if(opc == 0)
              {
                  //Se define la consulta
                  sql = "delete from EXPOSICIONES.det_artistas_exposicion where id_exposicion =
"+this.etIdExp.getText()+"and id_Artista = "+idV ;
                  con.setSQL(sql);
                  con.setQueryTabla("select * from EXPOSICIONES.det_artistas_exposicion C where
C.id_exposicion="+this.id);
                  //Ejecución de la cosnulta
                  con.EjecutaSentencia();
              }
        }else {
            JOptionPane.showMessageDialog(null, "Seleccione un registro","Eliminación",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

void galeriadearte.ArtistasExposicion.InsertArtExpo ()

Método para insertar un artistas en alguna exposición

```
                                    {

        codigoArtista = Artistas[this.cbArtistas.getSelectedIndex()];
        sql = "insert into EXPOSICIONES.det_artistas_exposicion values ("+this.etIdExp.getText()+
","+this.codigoArtista+ ");";
        con.setSQL(sql);
        con.setQueryTabla("select * from EXPOSICIONES.det_artistas_exposicion C where C.id_exposicion="+this.id);
        con.EjecutaSentencia();
    }
```

static void galeriadearte.ArtistasExposicion.main (String *args*[])[static]

Parámetros:

args	the command line arguments

```
                                        {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ArtistasExposicion.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(ArtistasExposicion.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ArtistasExposicion.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ArtistasExposicion.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ArtistasExposicion().setVisible(true);
            }
        });
    }
```

Referencia de la Clase Conexion

Métodos públicos

- **Conexion** (String u, String p, JTable t, String qT)
- void **setQueryTabla** (String sql)
  Establece la consulta SQL para llenar la tabla.
- void **setSQL** (String sql)
  Establece la consulta DML para ser ejecutar en la base de datos.
- Statement **getSentencia** ()
- DefaultTableModel **getModelo** ()
- String **getUser** ()
- String **getPassword** ()
- void **setTabla** (JTable t)
- void **EstableceConexion** ()
- void **EjecutaSentencia** ()
- void **ActualizaTabla** ()
- void **cierraConexion** () throws SQLException

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Esta clase se utiliza para poder hacer la conexión con la base de datos así como ejecutar las consultas de inserción, eliminación y modificación de datos en las entidades de la base de datos.

Documentación del constructor y destructor

galeriadearte.Conexion.Conexion (String *u*, String *p*, JTable *t*, String *qT*)

Parámetros:

u	Nombre del usuario
p	Contraseña del usuario
t	Tabla que contiene los registros de algún formulario
qT	Consulta SQL que será ejecutada para llenar la tabla en el formulario

```
                                                    {

        user = u;
        password = p;
        tabla = t;
        queryTabla = qT;
        url = "jdbc:postgresql://localhost:5432/GaleriaDeArte";
        driver = "org.postgresql.Driver";

        ActualizaTabla();
    }
```

Documentación de las funciones miembro

void galeriadearte.Conexion.ActualizaTabla ()

Método encargado de actualizar el contenido de alguna tabla perteneciente en un formulario.

```
                                    {

    int nCol;

     try
     {
          //Se establece la conexion con la base de datos
           EstableceConexion();

           modelo = new DefaultTableModel();
           tabla.setModel(modelo);

           rs = sentencia.executeQuery(queryTabla);//Se ejcuta la consulta
           rsMd = rs.getMetaData();
           nCol = rsMd.getColumnCount();

           //Se inicializa el encabezado de la tabla
           for(int i=1; i <= nCol; i++) {
               modelo.addColumn(rsMd.getColumnLabel(i));
           }

           //Con este ciclo se agregan los registros a la tabla
           while(rs.next())
           {
               Object [] tupla = new Object[nCol];

               for(int i =0; i < nCol; i++) {
                     tupla[i] = rs.getObject(i+1);
                }

               modelo.addRow(tupla);
           }

           cierraConexion();

     }catch(Exception e)
     {
          JOptionPane.showMessageDialog(null, e.getMessage());
     }
    }
```

void galeriadearte.Conexion.cierraConexion () throws SQLException

Se cierra la conexión una vez finalizando alguna ejecución de sentencia sql en la base de datos.

```
                                                {

          if(sentencia != null )
          {
               sentencia.close();
          }
          if(conexion != null)
          {
               conexion.close();
          }
    }
```

void galeriadearte.Conexion.EjecutaSentencia ()

Este método se encarga de hacer la ejecución de las consultas de tipo SQL y DML en la base de datos.

```
                                    {

        try
        {
            EstableceConexion();
            sentencia.executeUpdate(sql);
            cierraConexion();
            ActualizaTabla();
        }catch(SQLException e)
        {
            JOptionPane.showMessageDialog(null, e.getMessage());
        }
    }
```

void galeriadearte.Conexion.EstableceConexion ()

Método encargado de hacer la conexión con la base de datos

```
                                    {

        try
        {
            //Se carga el controlador de postgresql para permitir la conexion
            Class.forName(driver);
            try
            {
                conexion = DriverManager.getConnection(url,user,password);//Creacion del objeto conexion
                sentencia = conexion.createStatement();//Se define un objeto de la clase Statement para permitir
ejecutar las consultas SQL
            }catch(SQLException ex)
            {
                JOptionPane.showMessageDialog(null,ex.getMessage());
            }
        }catch(Exception e)
        {
            JOptionPane.showMessageDialog(null,e.getMessage());
        }
    }
```

Referencia de la Clase FArtistas

Herencias **galeriadearte.FPersonas**.

Métodos públicos

- **FArtistas** ()
- **FArtistas** (String u, String c)
- void **InsertArtista** ()
- void **DeleteArtista** ()
- void **UpdateArtista** ()

Métodos públicos estáticos

- static void **main** (String args[])

Otros miembros heredados

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Esta clase se utiliza para que los formularios FClientes, FVendedores y FArtistas se deriven de ella, ya que cuentan con atributos comunes. La clase contiene un conjunto de métodos públicos que nos ayudaran a hacer la conexión con la base de datos y poder ejecutar algunas consultas de tipo SQL y DML

Documentación del constructor y destructor

galeriadearte.FArtistas.FArtistas ()

```
                    {
        super();
    }
```

galeriadearte.FArtistas.FArtistas (String *u*, String *c*)

Parámetros:

u	Nombre del usuario
c	Constraseña del usuario

Se llena la tabla con los registros actuales del artista

```
                                    {

        //Llamado al cosntructor padre, para inicializar los atributos heredados
        super(u,c,"select * from ARTISTAS.Artistas");

        initComponents();

        //Método de la clase padre (Persona)
        ActualizaTabla(tablaA);
        Limpia();
        setLocationRelativeTo(this);
    }
```

Documentación de las funciones miembro

void galeriadearte.FArtistas.DeleteArtista ()

Método para dar de baja un Artista de la base de datos

```
                                    {
        int opc;

        if(tablaA.getSelectedRow() >= 0)
        {
            opc = JOptionPane.showConfirmDialog(null, "¿Esta seguro que quiere eliminar el registro?", "Eliminación",
JOptionPane.YES_NO_OPTION);
            if(opc == 0)
            {
                //Consulta DML para la eleminacion de un Artista
                sql = "DELETE FROM ARTISTAS.Artistas WHERE Id_Artista = "+this.etIdArtista.getText();
                EjecutaSentencia(tablaA);
            }
        }else {
            JOptionPane.showMessageDialog(null, "Seleccione un registro","Eliminación",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

void galeriadearte.FArtistas.InsertArtista ()

Método para registrar un artista en la base de datos.

```
                                    {
        int dia, mes, anio;
        String fecha;

        dia   = Calendar.getCalendar().get(5);
        mes   = Calendar.getCalendar().get(2)+1;
        anio =Calendar.getCalendar().get(1);

        fecha = dia+"/"+mes+"/"+anio;

        //Se hacen las validaciones previas a la insercion
        if(this.camposVacios() == 0)
        {
            if(anio <= 2000)
            {
                //Consulta DML para almacenar un registro del Artista
                sql = " INSERT INTO ARTISTAS.Artistas VALUES (default,'"
                        + this.tbNombreA.getText() + "','"
                        + this.tbApellidosA.getText()+ "','"
                        + fecha+"','"
                        + this.tbDireccionA.getText()+"','"
                        + this.tbTelefonoA.getText()+"','"
                        + this.tbPaisA.getText()+"','"
                        + this.tbEmailA.getText()+"','"
                        + this.tbReseñaA.getText()+"');";

                EjecutaSentencia(tablaA);
            }
            else
            {
                JOptionPane.showMessageDialog(null, "La fecha de nacimiento debe ser menor al año
2000","Inserción", JOptionPane.ERROR_MESSAGE);
            }
        }
        else
        {
            JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

static void galeriadearte.FArtistas.main (String *args*[])[static]

Parámetros:

args	the command line arguments

Reimplementado de **galeriadearte.FPersonas** (*p.31*).

```
                                        {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FArtistas.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FArtistas.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FArtistas.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FArtistas.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FArtistas().setVisible(true);
            }
        });
    }
```

void galeriadearte.FArtistas.UpdateArtista ()

Método para actualizar los datos de un Artista

Se ejecuta las sentencias DML

```
                                    {
        int dia, mes, anio;
        String fecha;

        dia   = Calendar.getCalendar().get(5);
        mes   = Calendar.getCalendar().get(2)+1;
        anio =Calendar.getCalendar().get(1);

        fecha = dia+"/"+mes+"/"+anio;

        if(this.camposVacios() == 0)
        {
            //Consulta DML para modificar los datos del algun Artistas
            sql = "UPDATE ARTISTAS.Artistas SET Id_Artista="
                    + this.etIdArtista.getText()+",Nombre='"
                    + this.tbNombreA.getText() + "',Apellidos='"
                    + this.tbApellidosA.getText()+ "',Fecha_Nacimiento='"
                    + fecha+"',Direccion='"
                    + this.tbDireccionA.getText()+"',Telefono='"
                    + this.tbTelefonoA.getText()+"',Pais='"
                    + this.tbPaisA.getText()+"',Email='"
                    + this.tbEmailA.getText()+"',Resena='"
                    + this.tbReseñaA.getText()+"'"
                    + "WHERE Id_Artista ="+this.etIdArtista.getText();
```

```
                EjecutaSentencia(tablaA);
            }else
            {
                JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
                tbReseñaA.setFocusTraversalKeysEnabled(true);
            }
        }
```

Referencia de la Clase FBackupRestore

Métodos públicos

- **FBackupRestore** ()
- void **pgBackUp** ()
- void **pgRestore** (String host, String puerto, String bDatos, String **path**)
- void **escribirProcess** (Process process) throws Exception

Métodos públicos estáticos

- static void **main** (String args[])

Atributos protegidos

- JFileChooser **seleccion**
- Process **p**
- ProcessBuilder **pb**
- String **path**
- String **nameFile**

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Clase utilizada para implementar el backup y restore de la base de datos. Estas operaciones se hacen de manera manual.

Documentación del constructor y destructor

galeriadearte.FBackupRestore.FBackupRestore ()

```
                                {
        initComponents();

        this.setLocationRelativeTo(this);
        seleccion = new JFileChooser();
        //Se establece el filtro del OpenFileDialogo
        seleccion.setFileFilter(new FileNameExtensionFilter("Text files (*.sql)","sql"));
    }
```

Documentación de las funciones miembro

void galeriadearte.FBackupRestore.escribirProcess (Process *process*) throws Exception

Este método se utiliza para imprimir el contenido de un archivo, que contiene el resultado un backup o un restore.

Parámetros:

process	Proceso externo, encargado de ejecutar los scripts de backup y restore

```
                                                {

        BufferedInputStream bufferIs = new BufferedInputStream(process.getInputStream());
```

```
        InputStreamReader isReader = new InputStreamReader( bufferIs );
        BufferedReader reader = new BufferedReader(isReader);
        String line = "";
        progreso.setText(line);

        while (true){
         line = reader.readLine();
            if (line == null)
            {
                break;
            }
            progreso.setText(progreso.getText()+"\n"+line);
        }
    }
```

static void galeriadearte.FBackupRestore.main (String *args*[])[static]

Parámetros:

args	the command line arguments

```
                                    {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FBackupRestore.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FBackupRestore.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FBackupRestore.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FBackupRestore.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FBackupRestore().setVisible(true);
            }
        });
    }
```

void galeriadearte.FBackupRestore.pgBackUp ()

Método encargado de hacer el respaldo de la base de datos

```
                                {

        //Se establecen los atributos necesarios para ejecutar el script pg_dump, encargado de hacer el backup
        String comando = "C:\\Program Files (x86)\\PostgreSQL\\9.2\\bin\\\\pg_dump.exe";
        String ubicacion = String.valueOf(seleccion.getSelectedFile().toString())+".sql";
        String host = "localhost";
        String puerto = "5432";
        String usuario = "postgres";
```

```
        String password = "postgres";
        String bDatos = "GaleriaDeArte";

        try {

            //Se crea un proceso externo a la aplicacion, con la finalidad de ejecutar el comando pg_dump
            pb = new ProcessBuilder(comando, "--verbose", "--inserts", "--column-inserts", "-f", ubicacion);
            pb.environment().put("PGHOST", host);
            pb.environment().put("PGPORT", puerto);
            pb.environment().put("PGUSER", usuario);
            pb.environment().put("PGPASSWORD", password);
            pb.environment().put("PGDATABASE", bDatos);
            pb.redirectErrorStream(true);
            p = pb.start();

            escribirProcess(p);
            progreso.setText(progreso.getText()+"\n"+"Terminando backup...\n");
          } catch (Exception e) {
              progreso.setText("backup \n"+e.getMessage()+"\n");
          }
    }
```

void galeriadearte.FBackupRestore.pgRestore (String *host*, String *puerto*, String *bDatos*, String *path*)

Método encargado de hacer la restauracion de la base de datos

Parámetros:

host	Servidor donde reside la base de datos
puerto	Puerto del servidor
bDatos	Nombre de la base de datos
path	Ubicacion de archivo backup que se desea respaldar

```
                                                                    {

        //Ruta del script
        String comando = "C:\\Program Files (x86)\\PostgreSQL\\9.2\\bin\\\\psql.exe";

        try {

            pb = new ProcessBuilder(comando,"-f", path, "-U", "postgres");
            pb.environment().put("PGPASSWORD", "postgres");
            pb.environment().put("PGHOST", host);
            pb.environment().put("PGPORT", puerto);
            pb.environment().put("PGDATABASE", bDatos);
            pb.redirectErrorStream(true);
            p = pb.start();

            escribirProcess(p);
            progreso.setText(progreso.getText()+"\n"+"Terminando Restore...\n");
     } catch (Exception e) {
          progreso.setText("Restore \n"+e.getMessage()+"\n");
     }

    }
```

Referencia de la Clase FClientes

Herencias **galeriadearte.FPersonas**.

Métodos públicos

- **FClientes** ()
- **FClientes** (String u, String p)
- void **InsertClient** ()
- void **DeleteClient** ()
- void **UpdateClient** ()

Métodos públicos estáticos

- static void **main** (String args[])

Otros miembros heredados

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Esta clase es utilizada para gestionar los datos los clientes cuenta con un conjunto de métodos que nos van a permitir operar la información de este.

Documentación del constructor y destructor

galeriadearte.FClientes(String *u*, String *p*)

Parámetros:

u	Nombre del usuario
p	Contraseña del usuario

```
                                    {

        super(u,p,"select * from VENTAS.Clientes");
        initComponents();

        this.setLocationRelativeTo(this);
        this.ActualizaTabla(TablaClientes);
        this.Limpia();
    }
```

Documentación de las funciones miembro

void galeriadearte.FClientes.DeleteClient ()

Método para dar de baja un cliente de la base de datos

```
                                {
        int opc;

        //Se checa que se haya seleccionado un registro de algun cliente.
```

```
        if(TablaClientes.getSelectedRow() >= 0)
        {
            opc = JOptionPane.showConfirmDialog(null, "¿Esta seguro que quiere eliminar el registro?", "Eliminación",
JOptionPane.YES_NO_OPTION);
            if(opc == 0)
            {
                //Se prepara la consulta de eleminación y despues es ejecutada
                sql = "delete from VENTAS.Clientes where Id_Cliente = "+this.IdCliente.getText();
                EjecutaSentencia(TablaClientes);
            }
        }else {
            JOptionPane.showMessageDialog(null, "Seleccione un registro","Eliminación",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

void galeriadearte.FClientes.InsertClient ()

Método para insertar un cliente a la base de datos.

```
                                {

        if(this.camposVacios() == 0)
        {
            //Se prepara la sentencia DML para insertar un cliente a la base de datos
            sql = "insert into VENTAS.Clientes values (default,'"
            + this.Nombre.getText() + "','"
            + this.ApellidoPaterno.getText()+ "','"
            + this.ApellidoMaterno.getText()+"','"
            + this.Direccion.getText()+"','"
            + this.Telefono.getText()+"','"
            + this.Ciudad.getText()+"','"
            + this.CP.getText()+"','"
            + this.Email.getText()+"');";

            //Se ejecuta la consulta
            EjecutaSentencia(TablaClientes);
        }else
        {
            JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

static void galeriadearte.FClientes.main (String *args*[])[static]

Parámetros:

args	the command line arguments

Reimplementado de **galeriadearte.FPersonas** (*p.31*).

```
                                        {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FClientes.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {
```

```
            java.util.logging.Logger.getLogger(FClientes.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FClientes.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FClientes.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FClientes().setVisible(true);
            }
        });
    }
```

void galeriadearte.FClientes.UpdateClient ()

Método para actualizar los datos de un cliente

```
                                    {

        if(this.camposVacios() == 0)
        {
            //Consulta DML para modificar los datos de algun cliente
            sql = "update VENTAS.Clientes set Id_Cliente="
            + this.IdCliente.getText()+",Nombre='"
            + this.Nombre.getText() + "',Apellido_Paterno='"
            + this.ApellidoPaterno.getText()+ "',Apellido_Materno='"
            + this.ApellidoMaterno.getText()+"',Direccion='"
            + this.Direccion.getText()+"',Telefono='"
            + this.Telefono.getText()+"',Ciudad='"
            + this.Ciudad.getText()+"',CP='"
            + this.CP.getText()+"',Email='"
            + this.Email.getText()+"'"
            + "where Id_Cliente ="+this.IdCliente.getText();

            EjecutaSentencia(TablaClientes);
        }else
        {
            JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

Referencia de la Clase FComisiones

Métodos públicos

- **FComisiones** ()
- **FComisiones** (String u, String p)
- void **IniComboBox** (JComboBox cb, String query)

Métodos públicos estáticos

- static void **main** (String args[])

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Clase implementada para realizar un reporte sobre las comisiones de la galeria y las ventas realizadas por los vendedores durante un mes en específico.

Documentación del constructor y destructor

galeriadearte.FComisiones (String *u*, String *p*)

Parámetros:

u	Nombre del usuario
p	Contraseña del usuario

```
                                    {
        initComponents();

        //Reserva de espacio para los Id de Exposición, Clientes y Vendedores
        ID = new int[100];

        con = new Conexion(u, p,tablaComisones, "select E.Titulo, E.Fecha_Apertura, E.Fecha_Cierre,
sum(C.Comision_Galeria) Comision_Total from EXPOSICIONES.Exposicion E left join
VENTAS.ComisionesPorExposicion C on E.Id_Exposicion = C.Codigo_Exposicion group by E.Titulo,
E.Fecha_Apertura, E.Fecha_Cierre");
        setTitle("Reporte de Comisiones");
        setLocationRelativeTo(this);
        IniComboBox(this.cbVendedores,"SELECT V.Id_Vendedor,(V.Nombre||' '||V.Apellido_Paterno) AS Name
FROM VENTAS.Vendedores V");
    }
```

Documentación de las funciones miembro

void galeriadearte.FComisiones.IniComboBox (JComboBox *cb*, String *query*)

Parámetros:

cb	Control utilizado para almacenar los nombres de los vendedores
query	Consulta SQL necesaria para recuperar los nombres de los vendedores y su clave primaria

```
                                    {
```

```
        ResultSet rsAux;
        int cont = 0;
        try
        {
            con.EstableceConexion();
            rsAux = con.getSentencia().executeQuery(query);

            while(rsAux.next())
            {
                cb.addItem(rsAux.getObject(2));
                this.ID[cont++] = Integer.parseInt( String.valueOf(rsAux.getObject(1)));
            }

            con.cierraConexion();

        }catch(Exception e)
        {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
    }
```

static void galeriadearte.FComisiones.main (String *args*[])[static]

Parámetros:

args	the command line arguments

```
                                        {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FComisiones.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FComisiones.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FComisiones.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FComisiones.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FComisiones().setVisible(true);
            }
        });
    }
```

Referencia de la Clase FExposicion

Métodos públicos

- **FExposicion** ()
- **FExposicion** (String u, String p)
- void **InsertExposicion** ()
- void **DeleteExposicion** ()
- void **UpdateExposicion** ()
- int **ValidaFechas** ()

Métodos públicos estáticos

- static void **main** (String args[])

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Esta clase es utilizada para gestionar las exposiciones que se haran en la galeria de arte

---

Documentación del constructor y destructor

galeriadearte.FExposicion (String *u*, String *p*)

Parámetros:

u	Nombre del usuario
p	Contraseña del usuario

```
                                    {

        initComponents();

        con = new Conexion(u, p, TablaExposiciones, "select * from EXPOSICIONES.Exposicion");

        this.fCierre = new int[3];
        this.fApertura = new int[3];
        this.setLocationRelativeTo(this);
        this.Limpia();
    }
```

Documentación de las funciones miembro

void galeriadearte.FExposicion.DeleteExposicion ()

Método encargado de elemininar una exposicion de forma física

```
                                    {

        String sql;
        int opc;

        if(TablaExposiciones.getSelectedRow() >= 0)
        {
```

```
            opc = JOptionPane.showConfirmDialog(null, "¿Esta seguro que quiere eliminar el registro?", "Eliminación",
JOptionPane.YES_NO_OPTION);
            if(opc == 0)
            {
                //Se define la consulta con el lenguaje DML para poder eleminar el registro
                sql = "delete from EXPOSICIONES.Exposicion where Id_Exposicion = "+this.IdExposicion.getText();
                con.setSQL(sql);
                con.EjecutaSentencia();
            }
        }else {
            JOptionPane.showMessageDialog(null, "Seleccione un registro","Eliminación",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

void galeriadearte.FExposicion.InsertExposicion ()

Método para registrar exposiones de la galeria de arte

```
                                    {
        String sql;

        if(this.camposVacios() == 0)
        {
            IniFechaCierre();
            IniFechaApertura();
            /*
             * Si la fecha es correcta, se prepara la sentencia de SQL para
             * poder hacer la insercion en la base de datos
             */
            if(this.ValidaFechas() == 1)
            {
                sql = "insert into EXPOSICIONES.Exposicion values (default,'"
                        + this.Titulo.getText() + "','"
                        + this.Descripcion.getText()+ "','"
                        + this.fechaApertura+"','"
                        + this.fechaCierre+"');";

                //Se establece la sentecia sql al objeto con y se ejecuta
                con.setSQL(sql);
                con.EjecutaSentencia();
            }
            else{
                JOptionPane.showMessageDialog(null, "La fecha de apertura debe ser menor a la fecha de
cierre","Inserción", JOptionPane.ERROR_MESSAGE);
            }
        }else
        {
            JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

static void galeriadearte.FExposicion.main (String *args*[])[static]

Parámetros:

args	the command line arguments

```
                                    {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
```

```
                        break;
                    }
                }
            } catch (ClassNotFoundException ex) {
                java.util.logging.Logger.getLogger(FExposicion.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
            } catch (InstantiationException ex) {
                java.util.logging.Logger.getLogger(FExposicion.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
            } catch (IllegalAccessException ex) {
                java.util.logging.Logger.getLogger(FExposicion.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
            } catch (javax.swing.UnsupportedLookAndFeelException ex) {
                java.util.logging.Logger.getLogger(FExposicion.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
            }
            //</editor-fold>

            /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
                    new FExposicion().setVisible(true);
                }
            });
        }
```

void galeriadearte.FExposicion.UpdateExposicion ()

Método para modificar los datos de alguna exposición

```
                                        {
            String sql;

            //Se efectuan las validaciones necesarias
            if(this.camposVacios() == 0)
            {
                IniFechaCierre();
                IniFechaApertura();
                if(this.ValidaFechas() == 1)
                {
                    //Si todo es correcto se declara una cadena, que contenga la sentencia DML para actualizar los datos
de la exposicón
                    sql = "update EXPOSICIONES.Exposicion set Id_Exposicion="
                    + this.IdExposicion.getText()+",Titulo='"
                    + this.Titulo.getText() + "',Descripcion='"
                    + this.Descripcion.getText()+ "',Fecha_Apertura='"
                    + this.fechaApertura+"',Fecha_Cierre='"
                    + this.fechaCierre+ "'where Id_Exposicion ="+this.IdExposicion.getText();

                    con.setSQL(sql);
                    con.EjecutaSentencia();
                }
                else
                {
                    JOptionPane.showMessageDialog(null, "La fecha de apertura debe ser menor a la fecha de
cierre","Inserción", JOptionPane.ERROR_MESSAGE);
                }
            }else
            {
                JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
            }
        }
```

int galeriadearte.FExposicion.ValidaFechas ()

Este método es utlizado para validar que la fecha de apertura sea menor a la fecha de cierre

Devuelve:

0 si la fecha de apertura es mayor en caso contrario retorna 1

```
                                {

        int res = 0;

        if(fApertura[2] < fCierre[2])
        {
            res = 1;
        }
        else
            if(fApertura[2] == fCierre[2])
            {
                if(fApertura[1] < fCierre[1])
                {
                    res = 1;
                }
                else
                    if(fApertura[1] == fCierre[1])
                    {
                            if(fApertura[0] < fCierre[0])
                            {
                                res = 1;
                            }
                    }
            }

        return(res);
    }
```

Referencia de la Clase FObras

Métodos públicos

- **FObras** ()
- **FObras** (String u, String p)
- void **CargaArtistas** ()
- void **InsertObra** ()
- void **DeleteObra** ()
- void **UpdateObra** ()
- int **getArtista** (int tuplaSel)
- int **ValidaFechas** ()

Métodos públicos estáticos

- static void **main** (String args[])

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Clase para controlar las obras de los Artistas

Documentación del constructor y destructor

galeriadearte.FObras (String *u*, String *p*)

Parámetros:

u	Nombre del usuario
p	Contraseña del usuario

```
                                {

        initComponents();

        Artistas = new int[200];
        fCreacion = new int[3];
        fIngreso = new int[3];

        //Se crea un objeto de la clase Cenexion, para interactuar con la base de datos
        con = new Conexion(u, p, tablaO, "select * from ARTISTAS.Obras");

        IniFechaIngreso();
        CargaArtistas();
        setLocationRelativeTo(this);
        Estado = 0;
    }
```

Documentación de las funciones miembro

void galeriadearte.FObras.CargaArtistas ()

Metodo para cargar los nombres de los Artistas existentes en la base de datos, en un control de tipo ComboBox

```
                                    {

        numArtistas = 0;

        try
        {
              con.EstableceConexion();
              rsArtistas = con.getSentencia().executeQuery("select A.Id_Artista, (A.Nombre|| ' ' || A.apellidos ) as
Nombre   from ARTISTAS.Artistas A");

              while(rsArtistas.next())
              {
                 cbArtista.addItem(rsArtistas.getObject(2));
                 Artistas[numArtistas++] = Integer.parseInt( String.valueOf(rsArtistas.getObject(1)));
              }
              con.cierraConexion();

        }catch(Exception e)
        {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
    }
```

void galeriadearte.FObras.DeleteObra ()

Método para eleminar una obra de la base de datos

```
                                {
        String sql;
        int opc;

        if(tablaO.getSelectedRow() >= 0)
        {
             opc = JOptionPane.showConfirmDialog(null, "¿Esta seguro que quiere eliminar el registro?", "Eliminación",
JOptionPane.YES_NO_OPTION);
             if(opc == 0)
             {
                  //Se establece la consulta para eliminar el registri de una obra
                  sql = "delete from ARTISTAS.Obras where Id_Obra = "+this.etIdObra.getText();
                  con.setSQL(sql);
                  con.EjecutaSentencia();//Ejecucion de la consulta en la base de datos
             }
        }else {
             JOptionPane.showMessageDialog(null, "Seleccione un registro","Eliminación",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

int galeriadearte.FObras.getArtista (int *tuplaSel*)

Método para obtener un indice asociado a un artista

Parámetros:

tuplaSel	Registro seleccionado en la tabla de la interfacez

Devuelve:

Un valor entero que indica el indice del control ComboBox, este indice es utilizado para modificar el elemento actual del control

```
                                    {

        int index = -1;

        this.codigoArtista = Integer.valueOf(String.valueOf(con.getModelo().getValueAt(tuplaSel, 1)));

        for(int i = 0; i < this.numArtistas; i++)
             if (Artistas[i] == this.codigoArtista )
             {
                 index = i;
                 break;
```

```
            }

        return(index);
    }
```

void galeriadearte.FObras.InsertObra ()

Métodoo para registrar un obra en la base de datos

```
                                {
        String sql;

        //Validacion previas a la insercion
        if(this.tbTitulo.getText().trim().length() != 0 && this.tbPrecio.getText().length() != 0)
        {
            IniFechaCreacion();
            if(this.ValidaFechas() == 1)
            {
                //Definicion de la consulta DML para registrar una obra
                this.codigoArtista = Artistas[this.cbArtista.getSelectedIndex()];
                sql = "insert into ARTISTAS.Obras values (default,"
                        + this.codigoArtista + ",'"
                        + this.tbTitulo.getText()+ "','"
                        + this.cbTipo.getSelectedItem()+"','"
                        + this.cbEstilo.getSelectedItem()+"','"
                        + this.cbMedio.getSelectedItem()+"',"
                        + this.tbPrecio.getText()+",'"
                        + this.fechaCreacionO+"','"
                        + this.etFechaIngreso.getText()+"',"
                        + this.Estado+");";

                con.setSQL(sql);//Se establece la consulta al objeto conexion
                con.EjecutaSentencia();//Se ejecuta la sentencia

            }else{
                JOptionPane.showMessageDialog(null, "La fecha de creación debe ser menor a la fecha de
ingreso","Inserción", JOptionPane.ERROR_MESSAGE);
            }
        }else{
                JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
            }
    }
```

static void galeriadearte.FObras.main (String *args*[])[static]

Parámetros:

args	the command line arguments

```
                                {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FObras.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FObras.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FObras.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FObras.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FObras().setVisible(true);
            }
        });
    }
```

void galeriadearte.FObras.UpdateObra ()

Método para modificar los datos de una obra

```
                                {
        String sql;

        //Se valida que los cambios de los datos sean correctos
        if(!this.tbTitulo.getText().trim().isEmpty() && this.tbPrecio.getText().length() != 0)
        {
            IniFechaCreacion();
            if(this.ValidaFechas() == 1)
            {
                //Se establece la consulta para hacer la modificación de los datos de la obra
                 sql = "update ARTISTAS.Obras set Id_Artista="
                        + this.codigoArtista + ",Titulo='"
                        + this.tbTitulo.getText()+ "',Tipo='"
                        + this.cbTipo.getSelectedItem()+"',Estilo='"
                        + this.cbEstilo.getSelectedItem()+"',Medio='"
                        + this.cbMedio.getSelectedItem()+"',Precio="
                        + this.tbPrecio.getText()+",Fecha_Creacion='"
                        + this.fechaCreacionO+"',Fecha_Ingreso='"
                        + this.etFechaIngreso.getText()+"',Estado="
                        + this.Estado+ "where Id_Obra ="+this.etIdObra.getText()+";";
                con.setSQL(sql);
                con.EjecutaSentencia();
            }else
            {
                JOptionPane.showMessageDialog(null, "La fecha de creación debe ser menor a la fecha de
ingreso","Inserción", JOptionPane.ERROR_MESSAGE);
            }
        }else
         {
                JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
         }

    }
```

int galeriadearte.FObras.ValidaFechas ()

Este método compara que la fecha de creacion de la obra sea menor a la fecha de ingreso

Devuelve:

res = 0 si la fecha de creacion es mayor a la fecha de ingreso res = 1 si la fecha de creacion es menor a la fecha de ingreso

```
                                {

        int res = 0;

        if(fCreacion[2] < fIngreso[2])
        {
            res = 1;
        }
        else
```

```
            if(fCreacion[2] == fIngreso[2])
            {
                if(fCreacion[1] < fIngreso[1])
                {
                    res = 1;
                }
                else
                    if(fCreacion[1] == fIngreso[1])
                    {
                        if(fCreacion[0] < fIngreso[0])
                        {
                            res = 1;
                        }
                    }
            }

        return(res);
    }
```

Referencia de la Clase FPersonas

Heredado por **galeriadearte.FArtistas**, **galeriadearte.FClientes** y **galeriadearte.FVendedores**.

Métodos públicos

- **FPersonas** ()
- **FPersonas** (String **user**, String paswoord, String consulta)
- void **EstableceConexion** ()
- void **EjecutaSentencia** (JTable t)
- void **ActualizaTabla** (JTable t)

Métodos públicos estáticos

- static void **main** (String args[])

Atributos protegidos

- Connection **conexion**
- Statement **sentencia**
- DefaultTableModel **modelo**
- ResultSetMetaData **rsMd**
- ResultSet **rs**
- String **driver**
- String **url**
- String **sql**
- String **user**
- String **password**

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Clase padre utilizada para que las clases Clientes, Artistas y Vendedores se deriven de ella. Esta clase cuenta con un conjunto de métodos que permitirán llevar a cabo la conexión hacia la base de datos y poder ejecutar algunas consultas de tipo SQL y DML en la base de datos.

Documentación del constructor y destructor

galeriadearte.FPersonas (String *user*, String *paswoord*, String *consulta*)

Parámetros:

user	Nombre del usuario
paswoord	Contraseña del usuario
consulta	Consulta asociada a una tabla que almacenera los registros de alguna de las entidades(Clientes,Vendedores,Artistas)

```
                                                                {
        this.user = user;
```

```
        this.password = paswoord;
        url = "jdbc:postgresql://localhost:5432/GaleriaDeArte";
        driver = "org.postgresql.Driver";
        queryTabla = consulta;
    }
```

Documentación de las funciones miembro

void galeriadearte.FPersonas.ActualizaTabla (JTable *t*)

Método utilizado para actulizar el contenido de la tabla asociada a uno de los formularios derivados del formulario Persona.

Parámetros:

t	Tabla perteneciente a uno de estos formularios: Clientes, Artistas, Vendedores

```
                                        {

    int nCol;

    try
    {
          EstableceConexion();
          modelo = new DefaultTableModel();
          t.setModel(modelo);

          rs = sentencia.executeQuery(queryTabla);
          rsMd = rs.getMetaData();
          nCol = rsMd.getColumnCount();

          for(int i=1; i <= nCol; i++) {
              modelo.addColumn(rsMd.getColumnLabel(i));
          }

          while(rs.next())
          {
              Object [] tupla = new Object[nCol];

              for(int i =0; i < nCol; i++) {
                    tupla[i] = rs.getObject(i+1);
               }

              modelo.addRow(tupla);
          }

          sentencia.close();
          conexion.close();

    }catch(Exception e)
    {
              JOptionPane.showMessageDialog(this, e.getMessage());
    }
}
```

void galeriadearte.FPersonas.EjecutaSentencia (JTable *t*)

Esta método se encarga de ejecutar las consultas de tipo DML. El resultado de esa consulta se vera reflejado en al tabla que es pasada como parametro formal.

Parámetros:

t	Tabla que sera llenada por los registros de la consulta resultante definida en el cosntructor

```
                                        {
```

```
        try
        {
            EstableceConexion();
            sentencia.executeUpdate(sql);
            sentencia.close();
            conexion.close();
            ActualizaTabla(t);
        }catch(SQLException e)
        {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
    }
```

void galeriadearte.FPersonas.EstableceConexion ()

Este método crea una conexion con la base de datos. En primera instancia carga en memoria el controlador de postgresql que nos ofrecera un conjunto de clases abstractas y métodos estaticos cuya funcionalidad es facilitarnos la comunicacion con la base de datos

```
                                    {
        try
        {
            Class.forName(driver);//Se carga el controlador de pstgresql
            try
            {
                conexion = DriverManager.getConnection(url, this.user, this.password);//Se crea un objeto de la clase
conexion
                sentencia = conexion.createStatement();
            }catch(SQLException ex)
            {
                JOptionPane.showMessageDialog(null,ex.getMessage());
            }
        }catch(Exception e)
        {
             JOptionPane.showMessageDialog(null,e.getMessage());
        }
    }
```

static void galeriadearte.FPersonas.main (String *args*[])[static]

Parámetros:

args	the command line arguments

Reimplementado en **galeriadearte.FVendedores** (*p.37*), **galeriadearte.FArtistas** (*p.10*) y **galeriadearte.FClientes** (*p.16*).

```
                                    {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FPersonas.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FPersonas.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (IllegalAccessException ex) {
```

```
            java.util.logging.Logger.getLogger(FPersonas.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FPersonas.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FPersonas().setVisible(true);
            }
        });
    }
```

Referencia de la Clase FUsuarios

Métodos públicos

- **FUsuarios** ()
- void **Init** ()
- void **login** ()

Métodos públicos estáticos

- static void **main** (String args[])

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Clase para controlar el login de los usuarios

Documentación del constructor y destructor

galeriadearte.FUsuarios.FUsuarios ()

```
                        {
        initComponents();

        this.setLocationRelativeTo(this);
    }
```

Documentación de las funciones miembro

void galeriadearte.FUsuarios.Init ()

```
                        {
        this.tbUsuario.setText("");
        this.tbPassword.setText("");
    }
```

void galeriadearte.FUsuarios.login ()

Método utilizado para validar los usuarios en el sistema

```
                        {
        String use;
        String cont;

        if(this.tbUsuario.getText().length() > 0)
        {
            use = tbUsuario.getText().toLowerCase();
            cont = this.tbPassword.getText();
            if( cont.length() > 0)
            {
                //Se verifica que el usuario sea parte de la base de datos
                //Si es un usuario correcto se hace una llamada a la ventana principal de sistema.
                try
                {
                    Class.forName("org.postgresql.Driver");
                    try
```

```
                    {
                        DriverManager.getConnection("jdbc:postgresql://localhost:5432/GaleriaDeArte", use,cont);
                        new Principal(this,use,cont).setVisible(true);

                    }catch(SQLException ex)
                    {
                         JOptionPane.showMessageDialog(null,ex.getMessage());
                    }
                    }catch(Exception e)
                    {
                          JOptionPane.showMessageDialog(null,e.getMessage());
                    }

            }else
            {
                JOptionPane.showMessageDialog(null, "Escriba la contraseña","Login",
JOptionPane.INFORMATION_MESSAGE);
            }

        }else
        {
            JOptionPane.showMessageDialog(null, "Escriba el nombre de usuario","Login",
JOptionPane.INFORMATION_MESSAGE);
        }
    }
```

static void galeriadearte.FUsuarios.main (String *args*[])[static]

Parámetros:

args	the command line arguments

```
                                        {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FUsuarios.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FUsuarios.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FUsuarios.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FUsuarios.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FUsuarios().setVisible(true);
            }
        });
    }
```

Referencia de la Clase galeriadearte.FVendedores

Herencias **galeriadearte.FPersonas**.

Métodos públicos

- **FVendedores** ()
- **FVendedores** (String u, String p)
- void **InsertVendedor** ()
- void **DeleteVendedor** ()
- void **UpdateVendedor** ()

Métodos públicos estáticos

- static void **main** (String args[])

Otros miembros heredados

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Esta clase es utilizada para gestionar los datos de los vendedores cuenta con un conjunto de métodos que nos van a permitir operar la información de este.

Definición en la línea 21 del archivo FVendedores.java.

Documentación del constructor y destructor

galeriadearte.FVendedores (String *u*, String *p*)

Parámetros:

u	Nombre del usuario
p	Nombre de su contraseña

Definición en la línea 35 del archivo FVendedores.java.

```
                                        {

        super(u,p,"select * from VENTAS.Vendedores");
        initComponents();

        this.fCreacion = new int[3];
        this.fIngreso = new int[3];

        this.ActualizaTabla(TablaVendedores);
        this.IniFechaIngreso();
        this.setLocationRelativeTo(this);
    }
```

Documentación de las funciones miembro

void galeriadearte.FVendedores.DeleteVendedor ()

Método para eleminar en vendedor de la base de datos

```
                                        {
        int opc;

        //Se verifica que se haya seleccionado el registro del vendedor
        if(TablaVendedores.getSelectedRow() >= 0)
        {
            opc = JOptionPane.showConfirmDialog(null, "¿Esta seguro que quiere eliminar el registro?", "Eliminación",
JOptionPane.YES_NO_OPTION);
            if(opc == 0)
            {
                sql = "delete from VENTAS.Vendedores where Id_Vendedor = "+this.IdVendedor.getText();
                EjecutaSentencia(TablaVendedores);
            }
        }else {
            JOptionPane.showMessageDialog(null, "Seleccione un registro","Eliminación",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

void galeriadearte.FVendedores.InsertVendedor ()

Método para registrar un vendedor a la base de datos

```
                                        {
        int dia, mes, anio;
        String fecha;

        dia   = FechaNacimiento.getCalendar().get(5);
        mes   = FechaNacimiento.getCalendar().get(2)+1;
        anio =FechaNacimiento.getCalendar().get(1);

        fecha = dia+"/"+mes+"/"+anio;

        //Validaciones previas al registro
        if(this.camposVacios() == 0)
        {
            if(this.validaFecha(anio)==1)
            {
                String s = this.Porcentaje.getText();
                int porcentaje = Integer.parseInt(s);

                //Se prepara la consulra DML para hacer la inserción del vendedor
                sql = "insert into VENTAS.Vendedores values (default,'"
                + this.Nombre.getText() + "','"
                + this.ApellidoPaterno.getText()+ "','"
                + this.ApellidoMaterno.getText()+ "','"
                + fecha+"','"
                + this.Direccion.getText()+"','"
                + this.Telefono.getText()+"','"
                + this.Ciudad.getText()+"','"
                + this.CP.getText()+"','"
                + this.Email.getText()+"','"
                + this.FechaIncorporacion.getText()+"','"
                + porcentaje+"');";

                EjecutaSentencia(TablaVendedores);
            }
              else
            {
                  JOptionPane.showMessageDialog(null, "La fecha de nacimiento debe ser anterior o igual al año
2000","Inserción", JOptionPane.ERROR_MESSAGE);

            }
        }else
```

```
        {
            JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
        }
    }
```

static void galeriadearte.FVendedores.main (String *args*[])[static]

Parámetros:

args	the command line arguments

Reimplementado de **galeriadearte.FPersonas** (*p.31*).

```
                                        {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FVendedores.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FVendedores.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FVendedores.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FVendedores.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FVendedores().setVisible(true);
            }
        });
    }
```

void galeriadearte.FVendedores.UpdateVendedor ()

Método para actualizar los datos de un vendedor

```
                                    {
        int dia, mes, anio;
        String fecha;

        dia   = FechaNacimiento.getCalendar().get(5);
        mes   = FechaNacimiento.getCalendar().get(2)+1;
        anio =FechaNacimiento.getCalendar().get(1);

        fecha = dia+"/"+mes+"/"+anio;

        if(camposVacios() == 0)
        {
            sql = "update VENTAS.Vendedores set Id_Vendedor="
            + this.IdVendedor.getText()+",Nombre='"
            + this.Nombre.getText() + "',Apellido_Paterno='"
            + this.ApellidoPaterno.getText()+ "',Apellido_Materno='"
```

```
                + this.ApellidoMaterno.getText()+ "',Fecha_Nacimiento='"
                + fecha+"',Direccion='"
                + this.Direccion.getText()+"',Telefono='"
                + this.Telefono.getText()+"',Ciudad='"
                + this.Ciudad.getText()+"',CP='"
                + this.CP.getText()+"',Email='"
                + this.Email.getText()+"',Fecha_Incorporacion='"
                + this.FechaIncorporacion.getText()+"',Porcentaje_Comision='"
                + this.Porcentaje.getText()+"'"
                + "where Id_Vendedor ="+this.IdVendedor.getText();

                this.EjecutaSentencia(TablaVendedores);
            }else
            {
                JOptionPane.showMessageDialog(null, "No puede haber campos vacios","Inserción",
JOptionPane.ERROR_MESSAGE);
            }
        }
```

Referencia de la Clase galeriadearte.FVentas

Métodos públicos

- **FVentas** ()
- **FVentas** (String u, String p)
- void **InicializaNoVenta** ()
- void **EjecutaSentencia** ()
- void **IniComboBox** (JComboBox cb, int f, String query)
- void **EstableceConexion** ()
- void **IniTablaObrasDispo** ()

Métodos públicos estáticos

- static void **main** (String args[])

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Clase utilizada para llevar a cabo las ventas en la galeía

Documentación del constructor y destructor

galeriadearte.FVentas (String *u*, String *p*)

Parámetros:

u	Nombre del usuario
p	Contraseña del usuario

```
                                        {
        initComponents();

        this.user = u;
        this.password = p;
        this.url = "jdbc:postgresql://localhost:5432/GaleriaDeArte";
        this.driver = "org.postgresql.Driver";
        this.TotalVenta = 0;
        this.bandera = 0;
        this.ID_Venta = 1;

        //Reservar de espacio para los Id de Exposición, Clientes y Vendedores
        ID = new int[3][]; //Fila 0=Exposiciones, 1=Clientes y 2=Vendedores
        for(int i = 0; i < 3; i++){
            ID[i] = new int[200];
        }

        //Cargar los ComboBox
        IniComboBox(this.cbExposicion,0,"SELECT E.Id_Exposicion, E.Titulo FROM EXPOSICIONES.Exposicion E");
        IniComboBox(this.cbClientes,1,"SELECT C.Id_Cliente, (Nombre || ' ' || apellido_paterno || ' ' || apellido_materno)
as NombreCompleto   FROM VENTAS.Clientes C");
        IniComboBox(this.cbVendedores,2,"SELECT V.Id_Vendedor,(Nombre || ' ' || apellido_paterno || ' ' ||
apellido_materno) as NombreCompleto FROM VENTAS.Vendedores V WHERE V.Id_Vendedor IN ( SELECT
C.Codigo_Vendedor FROM VENTAS.ComisionesPorExposicion C WHERE C.Codigo_Exposicion ="+ID[0][0]+")");
        this.bandera = 1;

        //Llenar tablas de Obras Disponibles
```

```
        IniTablaObrasDispo();
        IniFechaCreacion();
        InicializaNoVenta();
        setLocationRelativeTo(this);
    }
```

Documentación de las funciones miembro

void galeriadearte.FVentas.EjecutaSentencia ()

Este método se encarga de ejecutar algunas de las sentencias SQL o DML.

```
                                    {

        try
        {
            EstableceConexion();
            sentencia.executeUpdate(sql);
            sentencia.close();
            conexion.close();
        }catch(SQLException e)
        {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
    }
```

void galeriadearte.FVentas.EstableceConexion ()

Este método se encarga de hacer la conexión con la base. Para ello se carga el driver jdbc de postgresql.

```
                                     {

        try
        {
            Class.forName(driver);
            try
            {
                conexion = DriverManager.getConnection(url, this.user,this.password);
                sentencia = conexion.createStatement();
            }catch(SQLException ex)
            {
                  JOptionPane.showMessageDialog(null,ex.getMessage());
            }
        }catch(Exception e)
        {
             JOptionPane.showMessageDialog(null,e.getMessage());
        }
    }
```

void galeriadearte.FVentas.InicializaNoVenta ()

Este método es utilizado para controlar el codigo de la venta

```
                                    {

        ResultSet r,r2;
        Object dato;
        try
        {
            EstableceConexion();
            r = sentencia.executeQuery("select Max(Codigo)+1 from VENTAS.Ventas");

            while(r.next())
            {
                dato = r.getObject(1);
                //En caso que el numero de venta es igual a cero, se inicializa la secuencia de la llave primari que
pertenece a la venta
```

```
                if(dato == null)
                {
                    r2 = sentencia.executeQuery("SELECT setval('ventas.ventas_codigo_seq',1000,'t');");
                    ID_Venta = 1001;
                    break;
                 }else
                {
                    this.ID_Venta = Integer.parseInt( String.valueOf(r.getObject(1)));
                }
            }

            etNoVenta.setText(String.valueOf(ID_Venta));
            sentencia.close();
            conexion.close();
        }catch(SQLException e)
        {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
    }
```

void galeriadearte.FVentas.IniComboBox (JComboBox *cb*, int *f*, String *query*)

Este método es para inicializar algun control ComboBox con el nombre de las entidades derivadas del formulario Persona.

Parámetros:

cb	Control de tipo ComboBox puede ser (Exposicion,Clientes o Vendedores)
f	Indice entre(0-2) que indica la fila en la matriz, donde se guardara las llaves primarias de alguna de las entidades
query	Consulta SQL asociada al ComboBox y a la entidad{.

```
                                                    {

        ResultSet rsAux;
        int cont = 0;

        try
        {
            EstableceConexion();
            rsAux = sentencia.executeQuery(query);

            while(rsAux.next())
            {
               cb.addItem(rsAux.getObject(2));
               this.ID[f][cont++] = Integer.parseInt( String.valueOf(rsAux.getObject(1)));
            }

            sentencia.close();
            conexion.close();
        }catch(Exception e)
        {
           JOptionPane.showMessageDialog(this, e.getMessage());
        }
    }
```

void galeriadearte.FVentas.IniTablaObrasDispo ()

Este método se encarga de inicializar la tabla que contendra las obras disponibles.

```
                                        {

        int nCol;

        try
        {
            EstableceConexion();//Se hace la conexion con la base de datos
            modelo = new DefaultTableModel();
            tablaObrasDisp.setModel(modelo);
```

```
                //Se define la consulta sql para el llenado de la tabla
                rs = sentencia.executeQuery("select O.Id_Obra, O.Titulo, O.Precio "
                                            + "from ARTISTAS.Obras O inner join Exposiciones.det_artistas_exposicion E
on O.Id_Artista = E.Id_Artista "
                                            + "where E.Id_Exposicion = "+this.ID[0][this.cbExposicion.getSelectedIndex()]+"
and O.Estado = 0;");
                rsMd = rs.getMetaData();
                nCol = rsMd.getColumnCount();

                //Se crean los encabezados de la tabal
                for(int i=1; i <= nCol; i++) {
                    modelo.addColumn(rsMd.getColumnLabel(i));
                }

                while(rs.next())
                {
                    Object [] tupla = new Object[nCol];

                    for(int i =0; i < nCol; i++) {
                          tupla[i] = rs.getObject(i+1);
                     }
                    modelo.addRow(tupla);
                }
                sentencia.close();
                conexion.close();
            }catch(Exception e)
            {
                  JOptionPane.showMessageDialog(this, e.getMessage());
            }

        }
```

static void galeriadearte.FVentas.main (String *args*[])[static]

Parámetros:

args	the command line arguments

```
                                        {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FVentas.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FVentas.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FVentas.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FVentas.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FVentas().setVisible(true);
            }
        });
    }
```

Referencia de la Clase ReporteVenta

Métodos públicos

- **ReporteVenta** (String user, String password)
- void **IniciaReporte** (String user, String password)
- void **EjecutaReporte** (long venta, String exp, String cliente, String vendedor, String total) throws SQLException

---

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Clase utilizada para implementar un reporte.

Documentación del constructor y destructor

galeriadearte.ReporteVenta.ReporteVenta (String *user*, String *password*)

Parámetros:

user	Nombre del usuario
password	Contraseña del usuario

Definición en la línea 35 del archivo ReporteVenta.java.

```
                                                {

        this.IniciaReporte(user, password);
    }
```

Documentación de las funciones miembro

void galeriadearte.ReporteVenta.EjecutaReporte (long *venta*, String *exp*, String *cliente*, String *vendedor*, String *total*) throws SQLException

Este método se utiliza para ejecutar el reporte. Mostrandolo en un archivo pdf

Parámetros:

venta	Clave de venta
exp	Título de la exposición
cliente	Nombre del cliente
vendedor	Nombre del vendedor
total	Total de la venta

Excepciones:

SQLException	Manejo de expeciones de SQL

```
{

        JasperReport reporte; JasperPrint
        jasperPrint; Map<String, Object>
        parametros;
        JRExporter exporter = new JRPdfExporter();
        JasperViewer jV;
```

```
        try
        {

            reporte = (JasperReport)JRLoader.loadObject("boucher.jasper");//Se carga el formato del reporte
            parametros = new HashMap<String, Object>();

            //Se agregan los parametros que utlizara el reporte, en un mapa
            parametros.put("venta", venta);
            parametros.put("exposicion", exp);
            parametros.put("cliente",cliente);
            parametros.put("vendedor",vendedor);
            parametros.put("total",total);

            //Este objeto es la representacion de nuestro reporte
            jasperPrint = JasperFillManager.fillReport(reporte, parametros, conexion);
            jV = new JasperViewer(jasperPrint,false);

            jV.setTitle("Reporte de ventas");
            jV.setVisible(true);

            exporter.setParameter(JRExporterParameter.JASPER_PRINT,jasperPrint);
            exporter.setParameter(JRExporterParameter.OUTPUT_FILE,new java.io.File("ventaPdf.pdf"));
            exporter.exportReport();

            conexion.close();

        }catch(JRException E)
        {
             E.printStackTrace();
        }
    }
```

void galeriadearte.ReporteVenta.IniciaReporte (String *user*, String *password*)

Este método se encarga de hacer la conexion con la base de datos.

Parámetros:

user	Nombre del usuario
password	Contraseña del usuario

```
                                                    {

        try
        {
            Class.forName("org.postgresql.Driver");//Se carga el driver de postgresql
            try
            {
                conexion = DriverManager.getConnection("jdbc:postgresql://localhost:5432/GaleriaDeArte",
user,password);
            }catch(SQLException ex)
            {
                JOptionPane.showMessageDialog(null,ex.getMessage());
            }
        }catch(Exception e)
        {
             JOptionPane.showMessageDialog(null,e.getMessage());
        }
    }
```

Referencia de la Clase VendedoresExposicion

Métodos públicos

- **VendedoresExposicion** ()
- **VendedoresExposicion** (String u, String p, long id)
- void **CargaVendedores** ()
- void **InsertVendedorExp** ()
- void **DeleteVendedorExp** ()

Métodos públicos estáticos

- static void **main** (String args[])

Descripción detallada

Autor: Miguel Angel Galicia Torrez

Clase encargada de agregar los vendedores que participaran en alguna exposición

---

Documentación del constructor y destructor

galeriadearte.VendedoresExposicion (String *u*, String *p*, long *id*)

Parámetros:

u	Nombre del usuario
p	Contraseña del usuario
id	Clave primaria de la exposición

```
                                                {
        initComponents();

        con = new Conexion(u, p, VendedoresExposicion, "select * from VENTAS.comisionesporexposicion C where
C.codigo_exposicion="+ id);

        this.etIdExp.setText(Long.toString(id));
        this.setLocationRelativeTo(this);
        this.Vendedores = new int[200];
        this.CargaVendedores();
        this.id=id;
    }
```

Documentación de las funciones miembro

void galeriadearte.VendedoresExposicion.CargaVendedores ()

Este método se encarga de cargar los nombres de los vendedores en un comboBox y a su vez su llave primaria almacenándola en un arreglo

```
                                {

        this.numArtistas = 0;

        try
        {
```

```
                con.EstableceConexion();
                rsArtistas = con.getSentencia().executeQuery("select V.Id_Vendedor, (V.Nombre||' '||V.Apellido_Paterno)
AS Nombre   from VENTAS.Vendedores V");

                while(rsArtistas.next())
                {
                   cbVendedores.addItem(rsArtistas.getObject(2));
                   Vendedores[numArtistas++] = Integer.parseInt( String.valueOf(rsArtistas.getObject(1)));
                }

                con.cierraConexion();

            }catch(Exception e)
            {
                JOptionPane.showMessageDialog(this, e.getMessage());
            }
        }
```

void galeriadearte.VendedoresExposicion.DeleteVendedorExp ()

Método para eliminar un vendedor en alguna exposición. La eliminación es de manera física y lógica.

```
                                        {
            int opc;
            int fila=this.VendedoresExposicion.getSelectedRow();

            //Se obtiene la llave primaria de la exposición.
            long idV=Long.parseLong(String.valueOf(this.VendedoresExposicion.getModel().getValueAt(fila, 1)));

            if(VendedoresExposicion.getSelectedRow() >= 0)
            {
                opc = JOptionPane.showConfirmDialog(null, "¿Esta seguro que quiere eliminar el registro?", "Eliminación",
JOptionPane.YES_NO_OPTION);
                if(opc == 0)
                {
                    sql = "delete from VENTAS.comisionesporexposicion where codigo_exposicion =
"+this.etIdExp.getText()
                            + "and codigo_vendedor = "+idV ;
                    con.setSQL(sql);
                    con.setQueryTabla("select * from VENTAS.comisionesporexposicion C where
C.codigo_exposicion="+this.id);
                    con.EjecutaSentencia();
                }
            }else {
                JOptionPane.showMessageDialog(null, "Seleccione un registro","Eliminación",
JOptionPane.ERROR_MESSAGE);
            }
        }
```

void galeriadearte.VendedoresExposicion.InsertVendedorExp ()

Método para insertar un vendedor en la exposición en forma fisica y logica

```
                                        {
            codigoArtista = Vendedores[this.cbVendedores.getSelectedIndex()];
                    sql = "insert into VENTAS.comisionesporexposicion values ("
                            +this.etIdExp.getText()+ ",'"
                            + this.codigoArtista+ "',"
                            + 0+","
                            + 0+");";
                    con.setSQL(sql);
                    con.setQueryTabla("select * from VENTAS.comisionesporexposicion C where
C.codigo_exposicion="+this.id);
                    con.EjecutaSentencia();
        }
```

static void galeriadearte.VendedoresExposicion.main (String *args*[])[static]

Parámetros:

args	the command line arguments

```
                                        {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(VendedoresExposicion.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(VendedoresExposicion.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(VendedoresExposicion.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(VendedoresExposicion.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new VendedoresExposicion().setVisible(true);
            }
        });
```