

ESCUELA POLITECNICA NACIONAL
FACULTAD DE SISTEMAS
AGORITMO II – PROYECTO PELICULAS

INTEGRANTES:

Arias Benítez Jeremy I.

Illicachi Manzano Johan S.

Rodriguez Benavides Lenin D.

Tibanta Miranda Alexander F.

FECHA:

17/01/2023

PERIODO ADEMICO:

2022-B

Algoritmos y Estructura de datos II

Objetivos

Objetivo General

1. Desarrollar una aplicación en java que permita gestionar una tienda de películas de manera eficiente, aplicando los conceptos de programación orientada a objetos y el uso de diferentes métodos de ordenamiento y búsqueda, a través de un programa de administración de Películas.

Objetivos Específicos

1. Diseñar y crear una estructura de datos que represente una película y contenga al menos los atributos id de la película, nombre, año, calificación, género y estado.
2. Implementar métodos para permitir a la aplicación realizar las siguientes acciones: ingresar nuevas películas, eliminar una película determinada, listar las películas por año de creación, buscar una película utilizando el atributo nombre y mostrar la información asociada a la misma, imprimir en pantalla las 10 películas con la mejor calificación, mostrar la información de las películas de acuerdo con el género (elección del usuario), mostrar las películas ordenadas alfabéticamente (nombre de la película), y crear un método seleccionarPelículas que permita al usuario seleccionar las películas que podría alquilar (máximo 5), guardar el id de las películas que seleccione el usuario en una cola temporal, y mostrar la información almacenada en la cola una vez que el usuario haya terminado de seleccionar las películas que le interesa o cuando haya llegado al máximo permitido (5).
3. Modificar el atributo estado de “Disponible” a “Alquilado” de aquellas películas que el usuario haya alquilado.
4. Utilizar el entorno de desarrollo Eclipse para crear y probar la aplicación.
5. Trabajar en equipo con tus compañeros de grupo para completar el proyecto de manera eficiente y colaborativa.

Algoritmos y Estructura de datos II

Introducción

En la actualidad, es común encontrar plataformas y aplicaciones que nos permiten acceder a una amplia variedad de películas y series de televisión. Sin embargo, a veces puede ser difícil encontrar una aplicación que permita gestionar de manera eficiente una tienda de películas, especialmente si se desea utilizar diferentes métodos de ordenamiento y búsqueda para manipular y acceder a la información de manera rápida y sencilla.

Por ello, en este proyecto se ha desarrollado una aplicación en Java que permite gestionar una tienda de películas de manera eficiente, utilizando los conceptos de programación orientada a objetos y diferentes métodos de ordenamiento y búsqueda. La aplicación se ha creado utilizando el entorno de desarrollo Eclipse y ha sido probada y depurada para garantizar su correcto funcionamiento. Además, se ha trabajado en equipo con compañeros de grupo para completar el proyecto de manera eficiente y colaborativa.

En resumen, esta aplicación de gestión de películas ofrece una amplia variedad de funcionalidades para que el usuario pueda administrar de manera eficiente su tienda de películas y acceder a la información de manera rápida y sencilla.

Marco teórico

El ordenamiento secuencial y la búsqueda binaria son algoritmos de ordenamiento y búsqueda utilizados en la programación y en bases de datos. El ordenamiento secuencial es un algoritmo de ordenamiento que ordena los elementos de una colección de datos mediante la comparación de cada elemento con todos los demás elementos. Este algoritmo se basa en la comparación de pares de elementos y se repite hasta que todos los elementos estén ordenados. El ordenamiento secuencial es un algoritmo de ordenamiento simple pero ineficiente, ya que su tiempo de ejecución aumenta a medida que aumenta el tamaño de la colección de datos. Por otro lado, la búsqueda binaria es un algoritmo de búsqueda que se utiliza para encontrar un elemento específico en una colección de datos ordenados. Este algoritmo utiliza una técnica de división y conquista para reducir el espacio de búsqueda. En cada paso, se divide la colección de datos en dos partes y se elige la parte en la que se encuentra el elemento buscado. La búsqueda binaria es más eficiente que la búsqueda secuencial ya que su tiempo de ejecución es logarítmico en lugar de lineal. Es importante mencionar que estos métodos de búsqueda y ordenamientos son los más eficientes dentro del SQL ya que estas estructuras de datos están diseñadas para buscar y ordenar datos de manera óptima. Desarrollo

Algoritmos y Estructura de datos II

Descripción

Programa

Este programa está desarrollado en Java y tiene como objetivo facilitar la gestión de una tienda de películas. Utiliza principios de programación orientada a objetos, así como diversos métodos de ordenamiento y búsqueda para manipular y acceder a la información de manera eficiente y sencilla.

La aplicación cuenta con una estructura de datos que representa a cada película y almacena sus atributos básicos: id, nombre, año, calificación, género y estado. Además, ofrece métodos para realizar diversas acciones y consultas, tales como: ingresar nuevas películas, eliminar películas, listar películas por año, buscar películas por nombre, mostrar las 10 películas mejor calificadas, mostrar películas por género seleccionado por el usuario, mostrar películas ordenadas alfabéticamente por nombre, y seleccionar películas para alquilar (máximo 5), guardar el id de las películas que seleccione el usuario en una cola temporal, y mostrar la información almacenada en la cola una vez que el usuario haya terminado de seleccionar las películas que le interesa o cuando haya llegado al máximo permitido (5).

Además, se ha modificado el atributo estado de “Disponible” a “Alquilado” de aquellas películas que el usuario haya alquilado.

Para el desarrollo y prueba de la aplicación se ha utilizado el entorno de desarrollo Eclipse y se ha trabajado en equipo con compañeros de grupo para completar el proyecto de manera eficiente y colaborativa.

Herramientas

Se han utilizado diversas herramientas y tecnologías, tales como el lenguaje Java, el entorno NetBeans, el sistema operativo Windows 10, y la base de datos MySQL junto con la librería MySQL JDBC Driver. Además, se han utilizado algunas extensiones de librerías adicionales, como Calendar, Date, StringTokenizer, Pattern y Matcher.

Diseño y Arquitectura

Clase “Interacciones”:

Funciones principales utilizadas en la clase, utilizan para realizar operaciones en la base de datos, como insertar filas en una tabla, mostrar los resultados de una consulta en una JTable y buscar códigos en una tabla.

- **busquedaDespliegue:** Este método muestra los resultados de una consulta en una tabla JTable. Recibe como parámetros una JTable, el nombre de la tabla que se está consultando y la sentencia SQL que se está utilizando para realizar la consulta.
- **busquedaCod:** Este método busca un código en una tabla de la base de datos. Recibe como parámetros el nombre de la tabla que se está consultando, la sentencia SQL que se está utilizando para realizar la consulta y el nombre de la columna que se está buscando. Devuelve el código encontrado como un entero.
- **getConnection:** Este método se utiliza para conectarse a la base de datos. Devuelve un objeto Connection que se puede utilizar para realizar consultas y operaciones en la base de datos.

Algoritmos y Estructura de datos II

- conectar: Este método se utiliza para establecer una conexión con la base de datos. Devuelve un objeto Connection que se puede utilizar para realizar consultas y operaciones en la base de datos.
- busquedaArray: Este método realiza una búsqueda en una tabla específica de la base de datos y almacena los resultados en una lista. Recibe como parámetros el nombre de la tabla que se está consultando y la sentencia SQL que se está utilizando para realizar la consulta. Devuelve la lista con los resultados encontrados.

busquedaDespliegue

```
public void busquedaDespliegue(JTable jTablela, String selectTabla, String SQL) {
    try {
        conection = con1.getConnection();
        st = conection.createStatement();
        rs = st.executeQuery(SQL);
        if (selectTabla.equals("pelicula")) {
            Object[] pelicula = new Object[9];
            DefaultTableModel tabla = new javax.swing.table.DefaultTableModel(
                new Object[][] {},
                new String[] { "Código", "Nombre", "Estreno", "Idioma", "Puntaje", "Sinopsis",
                    "Genero", "Director", "Estado" });
            while (rs.next()) {
                pelicula[0] = rs.getInt("idPelicula");
                pelicula[1] = rs.getString("nombrePelicula");
                pelicula[2] = rs.getString("Estreno");
                pelicula[3] = rs.getString("idioma");
                pelicula[4] = rs.getString("PuntuacionSobre5");
                pelicula[5] = rs.getString("sinopsis");
                pelicula[6] = rs.getString("genero");
                pelicula[7] = rs.getString("directo");
                pelicula[8] = rs.getString("Disponibilidad");
                tabla.addRow(pelicula);
            }
            jTablela.setModel(tabla);
        }
        if (selectTabla.equals("prestamoPelicula")) {
            Object[] prestamoPelicula = new Object[8];
            DefaultTableModel tabla = new javax.swing.table.DefaultTableModel(
                new Object[][] {},
```

Algoritmos y Estructura de datos II

```

        new String[]{"IdPrestamo", "IdPelicula", "Titulo", "Id Usuario", "Nombre",
"Apellido", "FechaPrestamo", "FechaDevolucion"});

        while (rs.next()) {
            prestamoPelicula[0] = rs.getInt("idPrestamo");
            prestamoPelicula[1] = rs.getInt("IdPelicula");
            prestamoPelicula[2] = rs.getString("Titulo");
            prestamoPelicula[3] = rs.getInt("idEstudiante");
            prestamoPelicula[4] = rs.getString("NombreEstudiante");
            prestamoPelicula[5] = rs.getString("ApellidoEstudiante");
            prestamoPelicula[6] = rs.getString("FechaPrestamo");
            prestamoPelicula[7] = rs.getString("FechaDevolucion");
            tabla.addRow(prestamoPelicula);
        }
        jTable1.setModel(tabla);
    }
} catch (SQLException e) {
    Logger.getLogger(Interaccion.class.getName()).log(Level.SEVERE, null, e);
}
}

-----

public ArrayList busquedaArray(String selectTabla, String SQL) {
    ArrayList lista = new ArrayList();
    try {
        lista = new ArrayList();
        conection = con1.getConnection();
        st = conection.createStatement();
        rs = st.executeQuery(SQL);
        if (selectTabla.equals("pelicula")) {
            while (rs.next()) {
                Pelicula pelicula = new Pelicula();
                pelicula.setIdPelicula(rs.getInt("idPelicula"));
                pelicula.setNombrePelicula(rs.getString("nombrePelicula"));
                pelicula.setEstreno(Integer.parseInt(rs.getString("Estreno")));
                pelicula.setIdioma(rs.getString("idioma"));
                pelicula.setPuntuacionSobre5(Integer.parseInt(rs.getString("PuntuacionSobre5")));
                pelicula.setSinopsis(rs.getString("sinopsis"));
                pelicula.setGenero(rs.getString("genero"));
            }
        }
    }
}

```

Algoritmos y Estructura de datos II

```

        pelicula.setDirector(rs.getString("directo"));
        lista.add(pelicula);
    }
}
} catch (SQLException e) {
    Logger.getLogger(Interaccion.class.getName()).log(Level.SEVERE, null, e);
}
return lista;
}

-----
//Actualizar y eliminar datos tablas.
public String prepararActualizar(ArrayList<String> atributosActualizar) {
    String parametroCambio = "";
    Iterator i = atributosActualizar.iterator();
    while (i.hasNext()) {
        parametroCambio += i.next() + ",";
    }
    parametroCambio = parametroCambio.substring(0, parametroCambio.length() - 1);
    return parametroCambio;
}

public boolean actualizarEliminarTablas(String SQL) {
    int pos;
    try {
        PreparedStatement PS = con1.getConnection().prepareStatement(SQL);
        pos = PS.executeUpdate();
        System.out.println(pos);
        if (pos > 0) {
            return true;
        }
    } catch (SQLException e) {
        Logger.getLogger(Interaccion.class.getName()).log(Level.SEVERE, null, e);
        return false;
    }

    return false;
}

```

Algoritmos y Estructura de datos II

```
//Fin Actualizar tablas.
```

Clase “JPPeliculas”

JPPeliculas es una clase que extiende de javax.swing.JPanel, lo que significa que es un panel de Java con una interfaz gráfica de usuario. Contiene una serie de métodos y variables privadas, como una instancia de la clase Interacción y de las clases JFPeliculasInsentar, JFEstudianteActualizar y JFEstudianteEliminar para interactuar con las funcionalidades de inserción, actualización y eliminación de películas en una base de datos. Además, utiliza variables selectTabla y SQL para realizar operaciones en una base de datos. El método initComponents se utiliza para inicializar los componentes de la interfaz de usuario. Los métodos relacionados con el evento del botón se utilizan para habilitar o deshabilitar la edición de campos de texto específicos en la interfaz de usuario, y el método jTextFieldBusquedaFocusLost se utiliza para comprobar la validez de un campo de texto específico en la interfaz de usuario.

Clase “JFEstudianteActualizar”

JFEstudianteActualizar extiende de javax.swing.JFrame, lo que significa que es una ventana de aplicación de Java con una interfaz gráfica de usuario. Contiene una serie de métodos y variables privadas, como "x" e "y", que se utilizan para mover la ventana al arrastrarla con el mouse. También tiene una instancia de la clase Interacción y Comprobaciones, y utiliza una variable SQL y tablaActualizar para realizar operaciones en una base de datos. Los métodos relacionados con el evento del mouse se utilizan para mover la ventana al arrastrarla con el mouse. Los métodos relacionados con el evento del botón se utilizan para habilitar o deshabilitar la edición de campos de texto específicos en la interfaz de usuario, y el método jTextFieldGenero_ActualizarFocusLost se utiliza para comprobar la validez de un campo de texto específico en la interfaz de usuario.

```
public JFEstudianteActualizar() {
    initComponents();
    jComboBoxFiltrarEstudiante.setBackground(Color.WHITE);
    setLocationRelativeTo(this);
}

-----

private void jTextFieldBuscar_ActualizarEstKeyTyped(java.awt.event.KeyEvent evt) {
    char variable = evt.getKeyChar();
    int select = this.jComboBoxFiltrarEstudiante.getSelectedIndex();
    switch (select) {
        case 0: {
            if (Character.isLetter(variable)) {
                getToolkit().beep();
                evt.consume();
            }
        }
    }
}
```


Algoritmos y Estructura de datos II

```

        }
        break;
    }
}

private void jTFBuscar_ActualizarEstKeyReleased(java.awt.event.KeyEvent evt) {
    String datoBusqueda = this.jTFBuscar_ActualizarEst.getText();
    String columna = "count(*)";
    int select = jComboBoxFiltrarEstudiante.getSelectedIndex();
    switch (select) {
        case 0: {
            this.SQL = "Select count(*) from pelicula where idPelicula like '" + datoBusqueda +
            ""';";

            if (atributos.busquedaCod(tablaActualizar, SQL, columna) == 0) {
                getToolkit().beep();

            } else {
                this.SQL = "SELECT * FROM " + tablaActualizar + " WHERE idPelicula like '" +
            datoBusqueda + ""';";
                atributos.despliegueFields(SQL,        tablaActualizar,        jTFCodigo_Actualizar,
            jTFNombre_Actualizar, jTFEstreno_Actualizar,
                jTFIdioma_Actualizar,    jTFPuntaje_Actualizar,    jTFSinopsis_Actualizar,
            jTFGenero_Actualizar, jTFDirector_Actualizar, "actualizar");
            }
            break;
        }
        case 1: {
            this.SQL = "SELECT * FROM " + tablaActualizar + " WHERE nombrePelicula like
            '%" + datoBusqueda + "%'";
            atributos.despliegueFields(SQL,        tablaActualizar,        jTFCodigo_Actualizar,
            jTFNombre_Actualizar, jTFEstreno_Actualizar,
                jTFIdioma_Actualizar,    jTFPuntaje_Actualizar,    jTFSinopsis_Actualizar,
            jTFGenero_Actualizar, jTFDirector_Actualizar, "actualizar");
            break;
        }
    }
}

```

Algoritmos y Estructura de datos II

```
private void jBRegistrarActionPerformed(java.awt.event.ActionEvent evt) {
    if (this.jTFBuscar_ActualizarEst.getText().isEmpty()) {

    } else {
        ArrayList<String> atributosActualizar = new ArrayList<>();
        String parametroBusqueda = this.jTFCodigo_Actualizar.getText();
        if (this.jChBNombre_Actualizar.isSelected()) {
            atributosActualizar.add("nombrePelicula=" + jTFNombre_Actualizar.getText() + "");
        }
        if (this.jChBEstreno_Actualizar.isSelected()) {
            atributosActualizar.add("Estreno=" + jTFEstreno_Actualizar.getText() + "");
        }
        if (this.jChBPuntaje_Actualizar.isSelected()) {
            atributosActualizar.add("PuntuacionSobre5=" + jTFPuntaje_Actualizar.getText() +
""");
        }
        if (this.jChBIdioma_Actualizar.isSelected()) {
            atributosActualizar.add("idioma=" + jTFIdioma_Actualizar.getText() + "");
        }
        if (this.jChBSinopsis_Actualizar.isSelected()) {
            atributosActualizar.add("sinopsis=" + jTFSinopsis_Actualizar.getText() + "");
        }
        if (this.jChBGenero_Actualizar.isSelected()) {
            atributosActualizar.add("genero=" + jTFGenero_Actualizar.getText() + "");
        }
        if (this.jChBDirector_Actualizar.isSelected()) {
            atributosActualizar.add("directo=" + jTFDirector_Actualizar.getText() + "");
        }
        String parametroCambio = atributos.prepararActualizar(atributosActualizar);
        this.SQL = "UPDATE " + tablaActualizar + " SET " + parametroCambio + " WHERE
idPelicula LIKE " + parametroBusqueda;
        if (atributos.actualizarEliminarTablas(this.SQL) == true) {
            this.setVisible(false);
            limpiarCampos();
        } else {

        }
    }
}
```

Algoritmos y Estructura de datos II

```
}
```

Clase “JFEstudianteEliminar”

JFEstudianteEliminar es una clase de interfaz gráfica de usuario (GUI) en Java que se encarga de eliminar películas de una base de datos. La clase contiene métodos para mover la ventana en la pantalla, para validar que el usuario esté ingresando el tipo de datos correcto, y para buscar y eliminar películas en la base de datos. La clase también utiliza una instancia de la clase Interaccion para interactuar con la base de datos y una instancia de la clase Comprobaciones para verificar la validez de los datos ingresados por el usuario.

```
public JFEstudianteActualizar() {
    initComponents();
    jComboBoxFiltrarEstudiante.setBackground(Color.WHITE);
    setLocationRelativeTo(this);
}

-----
private void jTFBuscar_EliminarEstKeyReleased(java.awt.event.KeyEvent evt) {
    int select = jComboBoxFiltrarEstudiante.getSelectedIndex();
    String datoBusqueda = this.jTFBuscar_EliminarEst.getText();
    switch (select) {
        case 0: {
            this.SQL = "Select count(*) from pelicula where idPelicula like '" + datoBusqueda +
";";
            if (atributos.busquedaCod(tablaEliminar, SQL, "count(*)" ) == 0) {
                getToolkit().beep();
            } else {
                this.SQL = "SELECT * FROM " + tablaEliminar + " WHERE idPelicula like '" +
datoBusqueda + ";";
                atributos.despliegueFields(SQL,        tablaEliminar,        jTFCodigo_EliminarEst,
jTFNombres_EliminarEst,
                jTFApellidos_EliminarEst, jTFCelu_EliminarEst, jTFSectorEst_EliminarEst,
                jTFFacultad_EliminarEst,  jTFCorreo_EliminarEst,  jTFDirector_Eliminar,
"actualizar");
            }
            break;
        }
        case 1: {
            this.SQL = "SELECT * FROM " + tablaEliminar + " WHERE nombrePelicula like '%" +
datoBusqueda + "%'";
```

Algoritmos y Estructura de datos II

```

        atributos.despliegueFields(SQL,        tablaEliminar,        jTFCodigo_EliminarEst,
jTFNombres_EliminarEst,
        jTFApellidos_EliminarEst, jTFCelu_EliminarEst, jTFSectorEst_EliminarEst,
        jTFFacultad_EliminarEst,    jTFCorreo_EliminarEst,    jTFDirector_Eliminar,
"actualizar");
        break;
    }
}
}

```

Clase “JFEstudianteInsertar”

JFPeliculasInsentar es una clase de interfaz gráfica de usuario (GUI) que permite al usuario ingresar y registrar información sobre películas en una base de datos. Utiliza la clase Interacción para interactuar con la base de datos y realizar operaciones como la inserción de nuevos registros. Esta clase también tiene la capacidad de seleccionar imágenes para asociar con la película ingresada a través de un JFileChooser. También tiene características como movimiento de ventana y cierre de ventana.

```

public JFPeliculasInsentar() {
    initComponents();
    setLocationRelativeTo(this);
    this.jTFNombre.requestFocus();
    this.SQL = "Select idPelicula from pelicula order By idpelicula DESC limit 1;";
    int nuevoCod = atributos.busquedaCod(tablaInsertar, SQL, "idPelicula") + 1;
    this.jTFCodigo.setText(nuevoCod + "");
}

-----

private void jBRegistrarActionPerformed(java.awt.event.ActionEvent evt) {
    //ImagenPosters mimagen = new ImagenPosters();

    String nombre = jTFNombre.getText();
    String apellido = jTFEstreno.getText();
    String sector = jTFPuntuacion.getText();
    String facultad = jTFSinopsis.getText();
    String correo = jTFGenero.getText();
    if (this.jTFCodigo.getText().isEmpty() || nombre.isEmpty() || apellido.isEmpty()
        || this.jTFIdioma.getText().isEmpty() || sector.isEmpty() || facultad.isEmpty() ||
        correo.isEmpty()) {

```

Algoritmos y Estructura de datos II

```

    } else {

        int codigoEst = Integer.parseInt(jTFCodigo.getText());
        String celular = jTFIdioma.getText();
        String director = jTFDirector.getText();

        //datosBasura
        int codigoLibro = 0;
        int stock = 0;
        int disponibilidad = 0;
        //this.SQL="INSERT INTO pelicula values(?,?,?,?,?,?,?,?)";
        this.SQL = "INSERT INTO pelicula values(?,?,?,?,?,?,?,?)";
        if (atributos.insertarTabla(SQL, tablaInsertar, codigoEst, nombre, apellido, celular,
            sector, facultad, correo,
                codigoLibro, null, null, null, null, stock, disponibilidad, null, null, director)) {

            this.setVisible(false);
        } else {

        }
    }

    String url = lblurl.getText();
    if (url.trim().length() != 0 && nombre.trim().length() != 0) {
        atributos.guardarImagen(ruta, nombre);
    } else {
        JOptionPane.showMessageDialog(null, "No debe dejar los campos vacios: Nombre e
Imagen");
    }
}

```

Clase “JPMejores”

JPMejores es un panel de Java que se encarga de mostrar las películas mejor calificadas en una tabla. Utiliza la clase Interaccion para obtener los datos de las películas de una base de datos y la clase Pelicula para almacenar estos datos. La clase también utiliza la clase Collections para ordenar las películas de acuerdo con su calificación. El método llenarTabla() se encarga de llenar la tabla con las películas mejor calificadas. La tabla también tiene características de diseño como el color y tipo de letra del encabezado de la tabla.

Algoritmos y Estructura de datos II

```

public class JPMejores extends javax.swing.JPanel {

    Interaccion atributos = new Interaccion();
    private String selectTabla = "pelicula";
    private String SQL;
    private Pelicula pelicula;

    public JPMejores() {
        initComponents();
        jTableTablaEstudiantes.getTableHeader().setBackground(new Color(127, 0, 0));
        jTableTablaEstudiantes.getTableHeader().setFont(new Font("Segoe UI Symbol",
Font.BOLD, 12));
        jTableTablaEstudiantes.getTableHeader().setForeground(Color.WHITE);
        this.SQL = "select * from " + selectTabla;
        llenarTabla();
    }

    private void llenarTabla() {
        ArrayList<Pelicula> listaPeliculas = atributos.busquedaArray(selectTabla, SQL);
        Collections.sort(listaPeliculas);
        DefaultTableModel modelo = new DefaultTableModel(new String[]{"Código", "Nombre",
"Estreno", "Idioma", "Puntaje", "Sinopsis", "Genero", "Director"}, 10);
        jTableTablaEstudiantes.setModel(modelo);

        TableModel modeloDatos = jTableTablaEstudiantes.getModel();
        for (int i = 0; i < 10; i++) {
            pelicula = listaPeliculas.get(i);
            modeloDatos.setValueAt(pelicula.getIdPelicula(), i, 0);
            modeloDatos.setValueAt(pelicula.getNombrePelicula(), i, 1);
            modeloDatos.setValueAt(pelicula.getEstreno(), i, 2);
            modeloDatos.setValueAt(pelicula.getIdioma(), i, 3);
            modeloDatos.setValueAt(pelicula.getPuntuacionSobre5(), i, 4);
            modeloDatos.setValueAt(pelicula.getSinopsis(), i, 5);
            modeloDatos.setValueAt(pelicula.getGenero(), i, 6);
            modeloDatos.setValueAt(pelicula.getDirector(), i, 7);
        }
    }
}

```

Algoritmos y Estructura de datos II

Clase "JPRentas"

JPRentas es un panel de Java que se encarga de mostrar las películas que están siendo prestadas en una tabla. Utiliza la clase Interaccion para obtener los datos de las películas prestadas de una base de datos y la clase JFRentar y JFDevolver para realizar el proceso de préstamo y devolución de películas. El método `busquedaDespliegue()` se encarga de llenar la tabla con las películas prestadas. También tiene un botón para iniciar un nuevo préstamo de películas.

```
public class JPRentas extends javax.swing.JPanel {

    Interaccion interaccion = new Interaccion();
    JFRentar prestamo = new JFRentar();
    JFDevolver devolver = new JFDevolver();
    String selectTabla = "prestamoPelicula";
    String SQL;
    public JPRentas() {
        initComponents();
        Interaccion interaccion = new Interaccion();
        this.SQL = "select * from prestamoPelicula";
        interaccion.busquedaDespliegue(jTableTablaRentas, selectTabla, SQL);
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jTableTablaRentas = new javax.swing.JTable();
        jButtonNuevoPres = new javax.swing.JButton();

        setBackground(new java.awt.Color(255, 255, 255));

        jTableTablaRentas.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null, null},
                {null, null, null, null, null, null, null, null},
            },
            new String [] {
            }
        ));
    }
}
```

Algoritmos y Estructura de datos II

```

        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null},
        {null, null, null, null, null, null, null, null}
    },
    new String [] {
        "Id Prestamo", "Id Pelicula", "Titulo", "Id Usuario", "Nombre", "Apellido", "Fecha
Prestamo", "Fecha Devolucion"
    }
});
jScrollPane1.setViewportView(jTableTablaRentas);

jButtonNuevoPres.setText("PRESTAR");
jButtonNuevoPres.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        jButtonNuevoPresMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        jButtonNuevoPresMouseExited(evt);
    }
});
jButtonNuevoPres.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButtonNuevoPresActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
this.setLayout(layout);

```


Algoritmos y Estructura de datos II

```

        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup())
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(layout.createSequentialGroup()
                            .addGap(37, 37, 37)
                            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
663, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(46, 46, 46)
                                .addComponent(jButtonNuevoPres)))
                            .addContainerGap(527, Short.MAX_VALUE))
                    );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                    .addGap(29, 29, 29)
                    .addComponent(jButtonNuevoPres)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 33,
Short.MAX_VALUE)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(26, 26, 26))
                );
    } // </editor-fold>

    private void jButtonNuevoPresActionPerformed(java.awt.event.ActionEvent evt) {
        prestamo.setVisible(true);
    }

    private void jButtonNuevoPresMouseEntered(java.awt.event.MouseEvent evt) {
        this.jButtonNuevoPres.setText("PRESTAR");
    }

    private void jButtonNuevoPresMouseExited(java.awt.event.MouseEvent evt) {
        this.jButtonNuevoPres.setText("prestar");
    }

```

Algoritmos y Estructura de datos II

```
// Variables declaration - do not modify

private javax.swing.JButton jButtonNuevoPres;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTableTablaRentas;

// End of variables declaration

}
```

Clase “JFRentas”

JFRentar es una clase de Java que se encarga de manejar la interfaz gráfica de usuario para registrar un préstamo de una película. Utiliza la clase Fecha para obtener la fecha actual del sistema y la clase Interaccion para interactuar con una base de datos. El método `jBRegistrarActionPerformed` se encarga de registrar el préstamo de la película en la base de datos. La clase también utiliza la clase `JOptionPane` para mostrar mensajes de error o confirmación al usuario. El código de préstamo se genera automáticamente y se verifica la disponibilidad de la película antes de registrar el préstamo. Al finalizar el préstamo, la disponibilidad de la película se actualiza en la base de datos.

```
public JFRentar() {
    initComponents();
    this.fecha1 = fecha.fechaRegistro();
    this.jTFFechaPrestamo.setText(this.fecha1);
    this.SQL = "Select idPrestamo from prestamoPelicula order By idPrestamo DESC limit 1;";
    int nuevoCod = atributos.busquedaCod("prestamoPelicula", SQL, "idPrestamo") + 1;
    this.jTFCodigoPrestamo.setText(nuevoCod + "");
}

-----

private void jTFBuscar_ActualizarPeliKeyReleased(java.awt.event.KeyEvent evt) {
    this.selectTabla = "pelicula";
    String datoBusqueda = this.jTFBuscar_ActualizarPeli.getText();
    String columna = "count(*)";
    int select = jCBCodoNomLib.getSelectedIndex();
    switch (select) {
        case 0: {
            this.SQL = "Select count(*) from pelicula where idPelicula like " + datoBusqueda + " ";
            if (atributos.busquedaCod(selectTabla, SQL, columna) == 0) {
                getToolkit().beep();
            } else {

```

Algoritmos y Estructura de datos II

```

        this.SQL = "SELECT * FROM " + selectTabla + " WHERE idPelicula like '" +
datoBusqueda + "'";
        atributos.despliegueFields(SQL,          selectTabla,          jTFCodigo_Actualizar,
jTFTitulo_Actualizar, null,
            null, null, null, null, null, "actualizar");
    }
    break;
}
case 1: {
    this.SQL = "SELECT * FROM " + selectTabla + " WHERE nombrePelicula like '%" +
datoBusqueda + "%'";
    atributos.despliegueFields(SQL,          selectTabla,          jTFCodigo_Actualizar,
jTFTitulo_Actualizar, null,
        null, null, null, null, null, "actualizar");
    break;
}
}
}
}

```

Algoritmos y Estructura de datos II

Implementación

Durante el proceso de implementación, se trabajó en equipo para dividir el trabajo y coordinar el avance del proyecto. Cada miembro del equipo se encargó de una funcionalidad específica, como la adición de películas, la renta de películas, la creación de tablas en la base de datos MySQL, la visualización del top 10 de mejores películas y la búsqueda de películas. Esto permitió aprovechar al máximo las habilidades y conocimientos de cada uno, y agilizó el proceso de implementación del proyecto. Además, se realizaron reuniones periódicas para revisar el avance y asegurarse de que cada miembro estuviera cumpliendo con sus tareas y para resolver dudas o problemas. También se estableció una comunicación constante a través de una aplicación de mensajería para solucionar problemas de manera rápida. Gracias a esto, se logró completar el proyecto de manera satisfactoria y se cumplieron todos los objetivos específicos establecidos al inicio del proyecto.

Sin embargo, también se enfrentaron algunos desafíos y problemas durante el proceso que tuvieron que ser superados. Por ejemplo, al tratar de conectar las clases entre sí para compartir datos de manera eficiente, se utilizaron métodos set y get para establecer y obtener valores de las variables en diferentes clases. Además, al trabajar con la base de datos, se tuvieron que solucionar problemas de conectividad y sincronización de datos, los cuales fueron resueltos importando la librería JDBC Driver y utilizando la clase DriverManager para establecer la conexión. También se utilizaron expresiones regulares para validar la entrada de datos y evitar errores de programación.

Después del proceso de implementación, se llevó a cabo una fase de pruebas y depuración de la aplicación para asegurarse de que funcionaba de manera adecuada y sin errores. Se probaron todas las funcionalidades y se verificó que los resultados obtenidos fueran los esperados. Además, se hizo una revisión del código para identificar y corregir errores o problemas de programación.

Conclusión

En conclusión, el proyecto de Películas ha sido un éxito gracias al trabajo en equipo y al esfuerzo de todos los miembros del grupo. Se han cumplido todos los objetivos establecidos al inicio del proyecto y se ha desarrollado una aplicación eficiente y fácil de usar para gestionar una tienda de películas. Además, se han aplicado y adquirido conocimientos en programación orientada a objetos y el uso de métodos de ordenamiento y búsqueda, así como la conexión a bases de datos y la validación de datos.

En el futuro, se pueden realizar mejoras y ampliaciones al programa, como la integración de un sistema de pago en línea para facilitar la renta de películas o la incorporación de un sistema de recomendaciones de películas basado en el historial de películas vistas por el usuario. De cualquier manera, este proyecto ha sido una valiosa oportunidad para aplicar y profundizar los conocimientos adquiridos en el curso de programación orientada a objetos y el uso de bases de datos en Java. También ha sido una oportunidad para trabajar en equipo y aprender a coordinar el trabajo para alcanzar objetivos comunes de manera eficiente. Además, se ha podido desarrollar habilidades en la resolución de problemas y la depuración de código, lo que sin duda será de gran utilidad en el futuro.

En resumen, el proyecto de películas ha sido una experiencia enriquecedora y ha permitido aplicar y mejorar los conocimientos adquiridos en el curso de programación orientada a objetos y el uso de bases de datos en Java. A pesar de los desafíos y problemas enfrentados durante el proceso de implementación, gracias al trabajo en equipo y al esfuerzo de todos los miembros del grupo, se ha logrado completar la aplicación de manera satisfactoria y cumplir con todos los objetivos establecidos al inicio del proyecto.

Algoritmos y Estructura de datos II

Anexos

Enlace al proyecto: <https://github.com/JexDev13/FilmFinder>



IMAGEN 1. MAIN PRINCIPAL



IMAGEN 3. ACTUALIZAR INFORMACION DE PELICULA



IMAGEN 2. INSERTAR NUEVA PELICULA

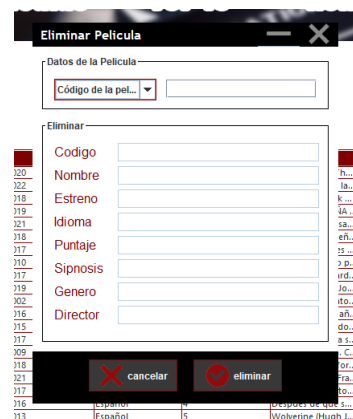


IMAGEN 4. ELIMINAR PELICULA

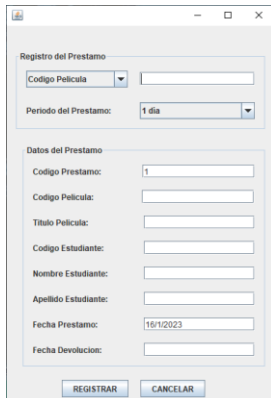


IMAGEN 6. VENTANA TOP 10



IMAGEN 5. VENTANA RENTAS

Algoritmos y Estructura de datos II



Registro del Prestamo

Codigo Pelicula:

Periodo del Prestamo:

Datos del Prestamo

Codigo Prestamo:

Codigo Pelicula:

Titulo Pelicula:

Codigo Estudiante:

Nombre Estudiante:

Apellido Estudiante:

Fecha Prestamo:

Fecha Devolucion:

IMAGEN 7. RENTAR PELICULA