



ESPOTIFAI

Grupo C09

Integrantes:

- Esteban de Jesus Avendano Arrubla
- Carlos Romero Rodriguez
- Paula Torres Garcia
- Ariana Sarita Vásquez Barrantes
- Aniol Verges Herrera

Contingut

1.	Resum de les especificacions del projecte.....	2
2.	Disseny de la interfície gràfica.....	2
2.1.	Contenidors	3
2.2.	LayoutManagers	8
2.3.	Components: Son elements visuals de la interfície i hereten de la classe JComponent.....	12
3.	Diagrama de classes.....	13
4.	Metodologia de desenvolupament	15
5.	Dedicació.....	17
6.	Conclusions	17
7.	Bibliografia consultada.....	18

1. Resum de les especificacions del projecte

El projecte és un programa fet per a les persones que lis agrada la música, perquè podran buscar cançons, escoltar-les i guardar-les. Per això, primer s'hauran creat un compte amb el qual podran tenir accés a aquestes diferents accions que permet el programa.

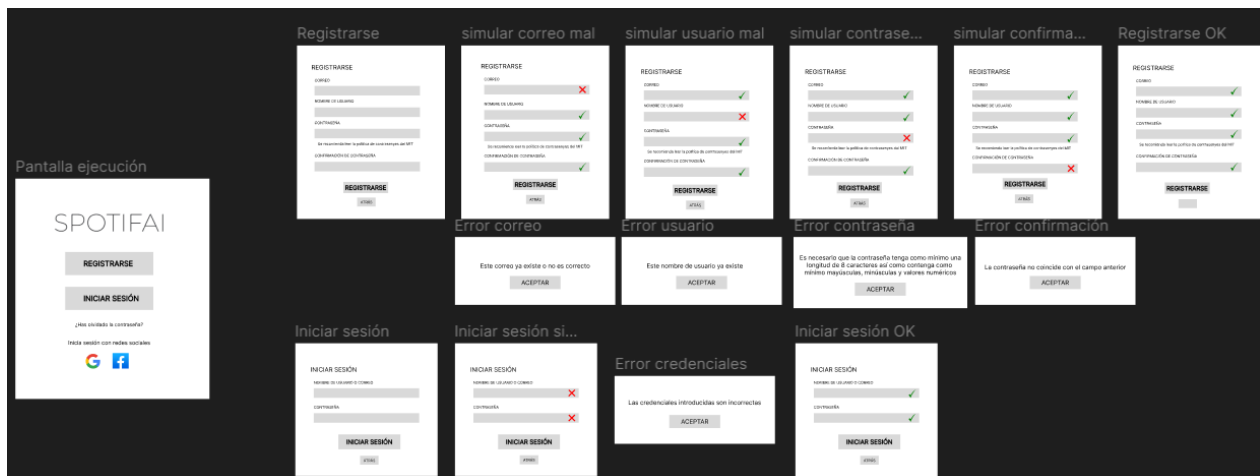
Les accions que ofereix el programa, tracten d'escoltar una varietat de cançons, les quals cada usuari pot reproduir, pausar, passar de cançó, tornar a l'anterior o tornar-la a repetir. A més, podrà visualitzar informació de les cançons, com el títol, el gènere, l'àlbum, l'autor i l'usuari propietari. Com hi ha diferents camps, es pot filtrar la busqueda d'una cançó i addicionalment es veu la seva duració, la lletra (si està disponible) i a més te la possibilitat de ficar-la dins d'una llista de reproducció.

Per tant, amb Spotifai es poden crear llistes de reproducció personalitzades, és a dir, pot afegir cançons i eliminar-les, a més en el programa es poden compartir els gustos musicals, fent que el usuari pugui veure les llistes dels altres, tenint-les actualitzades a temps real. Però a més de música, també hi ha la possibilitat de veure el gràfic d'estadístiques musicals que ofereix el programa, on es mostra la quantitat de cançons que hi ha per cada gènere musical.

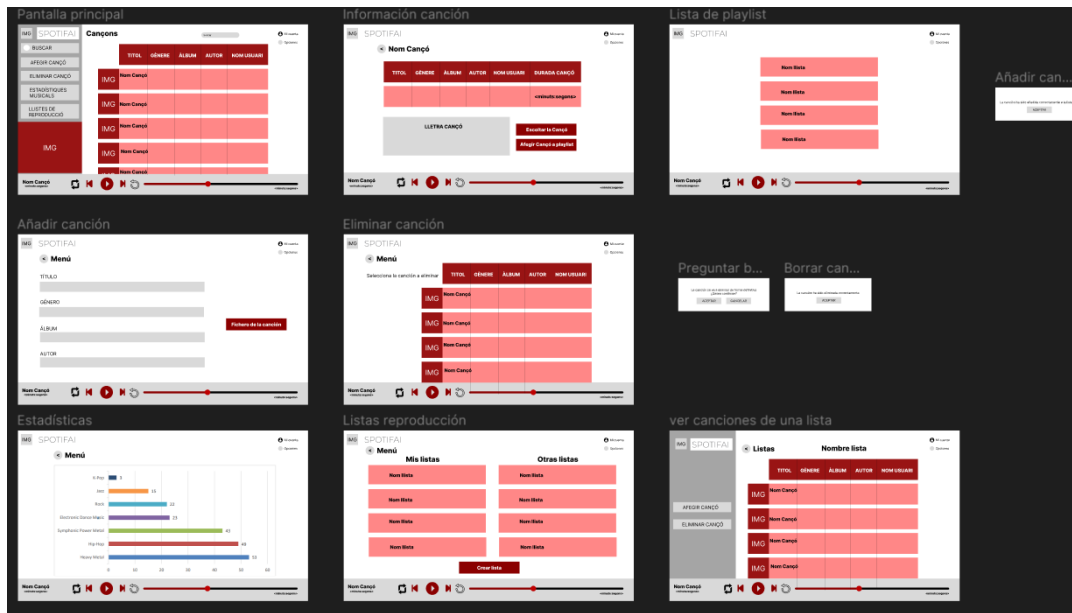
En resum, Spotifai és un programa amb contingut de música, on cada usuari pot afegir cançons que estaran disponibles per tothom que estigui a la plataforma i també pot eliminar la seva pròpia cançó, fent que no estigui disponible en el sistema. Així tots els usuaris tenen accés a una àmplia biblioteca de cançons, gaudint de la possibilitat de descobrir música d'altres usuaris, i contemplant la quantitat de categories que hi ha, calculat i representat en estadístiques.

2. Disseny de la interfície gràfica

Per començar el disseny vam establir uns mockups orientatius per saber com volíem començar a programar. Aquests son els resultats finals i com es pot veure, vam tenir canvis a l'hora de programar:



Aquí deixem el link dels mockups per a millor visualització: [PULSA AQUÍ](#)



El projecte te Swing com a biblioteca d'interfícies gràfiques d'usuari, que estén de la llibreria AWT. Això ho hem implementat al programa per tal de desenvolupar les vistes, encara que també té codi a backend per gestionar esdeveniments. Amb això en referim a les classes: contenidors, components i layouts. Quan vam fer els mockups, vam establir els tipus de classes que utilitzaríem perquè creiem necessaris, i son les següents que explicarem. Hem de comentar que mostrem un o dos exemples, per a que el document no sigui tan extens.

2.1. Contenidors

CONTENIDORS SUPERIORS

Son la base per la construcció del GUI.

➤ JFrame:

```
public class SignInView extends JFrame {
```

```
public class ListMusicView extends JFrame {
```

```
public class DeleteMusicView extends JFrame {
```

```
public class MainMenuView extends JFrame {
```

```
public class WelcomeView extends JFrame {
```

```
public class AddMusicView extends JFrame {
```

```
public class SignUpView extends JFrame implements
    SignUpObserver {
```

Canciones				
Título	Género	Autor	Album	Propietario
Cancion2	Genero2	Autor2	Album2	Hola
Cancion3	Genero3	Autor3	Album3	Hola
Cancion4	Genero4	Autor4	Album4	Hola
Cancion5	Genero5	Autor5	Album5	Hola

(Exemple visual de la classe ListMusicView que estén d'un JFrame)

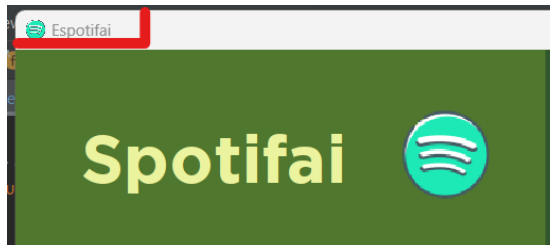
Si estenem la classe JFrame, la classe (com SignInView, DeleteMusicView, ListMusicView, etc) hereta tots els mètodes i propietats del JFrame, això permet crear una finestra que es pot mostrar a la pantalla i que depèn de la classe en la que estiguem, incloem per exemple botons o camps de text. En estendre el JFrame, és possible afegir mètodes i comportaments addicionals a la classe SignInView per personalitzar l'aparença i el comportament de la finestra segons les necessitats del programa.

```
JFrame frame = new JFrame();
```

Aquí es crea una instància de la classe JFrame anomenada frame utilitzant el constructor predeterminat new JFrame(). Creant una instància de JFrame, estem creant una finestra buida que es pot mostrar a la pantalla.

```
private final JFrame window;
```

```
this.window = new JFrame( title: "Espotifai");
```



Es declara una variable anomenada window de tipus JFrame amb el modificador private i final. Després, dins un constructor o mètode de la classe on es troba aquest codi, s'inicialitza la variable window mitjançant la creació d'una nova instància de la classe JFrame i se li assigna un títol "Espotifai".

En resum, aquest codi crea una instància nova de JFrame i l'assigna a la variable window amb el títol "Espotifai". Això permet que la finestra tingui un títol personalitzat quan es mostri a la interfície d'usuari.

```
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Quan l'usuari tanca la finestra, el sistema operatiu genera un esdeveniment de tancament que normalment ha de ser manejat per l'aplicació. En establir JFrame.EXIT_ON_CLOSE, s'indica que l'aplicació ha de finalitzar i tancar tots els recursos associats quan es tanqui la finestra. És important destacar que el mètode setDefaultCloseOperation ha de ser anomenat després d'haver creat la instància de JFrame, en aquest cas, window.

➤ Operacions:

○ Component:

```
button.setSize(120, 30);  
frame.setSize(500, 700);  
window.setSize(500, 700);  
window.setSize(1300, 800);
```

El mètode `setSize(int width, int height)` s'utilitza per establir la mida d'un component (en el cas `button` per exemple) i "window" en píxels. Els paràmetres `width` i `height` especifiquen l'amplada i l'alçada respectivament.

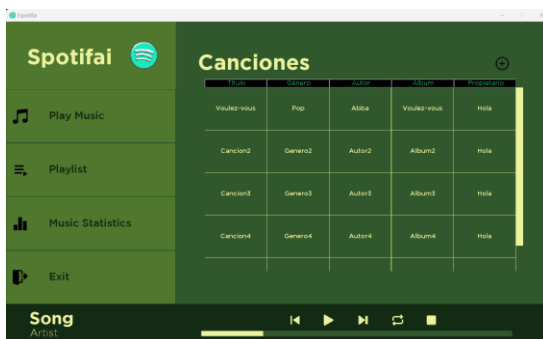
`button.setSize(120, 30);` estableix la mida del botó amb una amplada de 120 píxels i una alçada de 30 píxels. Això defineix les dimensions visuals del botó a la interfície d'usuari.



`window.setSize(int, int)` estableix la mida de la finestra (`JFrame`). Aquí, `window` fa referència a una instància de la classe `JFrame` prèviament declarada i assignada. Això defineix les dimensions visuals de la finestra a la interfície d'usuari.

```
frame.setVisible(true);  
window.setVisible(true);  
window.setVisible(true);
```

El mètode `setVisible(boolean visible)` s'utilitza per mostrar o amagar un component o una finestra a la interfície d'usuari.



`window.setVisible(true);` estableix la visibilitat de la finestra (`JFrame`) com a veritable. Aquí, `window` fa referència a una instància de la classe `JFrame` prèviament declarada i assignada. Per això és possible que es pugui veure la llista de cançons.

En trucar al mètode `setVisible(true)`, es mostra el component o la finestra a la interfície d'usuari, la qual cosa permet que siguin visibles i estiguin disponibles per interactuar-hi. Perquè el component o la finestra siguin visibles, també han d'haver estat configurats amb una mida adequada utilitzant el mètode `setSize`.

○ Window:

```
public void createViewReproductor(){
    window.add(panelPrincipal);
    window.remove(cardPanelInici);
    window.pack();
    window.setSize( width: 1300, height: 800);
    window.setLocationRelativeTo(null);
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    window.setResizable(false);
    cardLayout.show(cardPanelInici, name: "welcome");
    window.setVisible(true);
}
```

El mètode pack() en un objecte JFrame s'utilitza per redimensionar automàticament la finestra perquè s'ajusti al contingut i assegurar que tots els components siguin visibles correctament.

○ getContentPane()

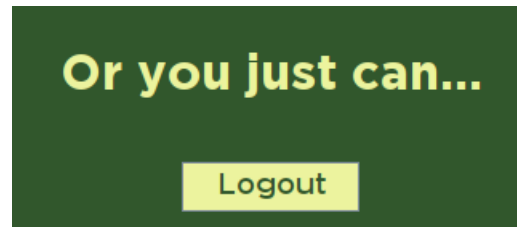
```
panelSuperiorDerecha.add(playListView.getContentPane(), "playlist");
panelSuperiorDerecha.add(musicStatisticsView.getContentPane(), "musicStatistics");
public JPanel getContentPane() {
```

El mètode getContentPane() retorna el panell de contingut d'un JFrame, cosa que permet afegir components i personalitzar l'aparença de la finestra.

El codi afegeix el panell de contingut de musicStatisticsView al contenidor panellSuperiorDreta, utilitzant una restricció de disseny específica. Això permet mostrar i organitzar el contingut del panell d'estadístiques de música (musicStatisticsView) dins d'un contenidor visual més gran (panellSuperiorDreta).

```
c.gridy = 1;
logoutButton = new JButton( text: "Logout");
logoutButton.setFont(Fonts.getMediumFont( size: 15f));
logoutButton.setBackground(UIPalette.TEXT_COLOR.getColor());
logoutButton.setForeground(UIPalette.COLOR_PRIMARIO.getColor());
logoutButton.setActionCommand(LOGOUT_COMMAND);
contentPane.add(logoutButton , c);
}

1 usage
public JPanel getContentPane() {
    return contentPane;
}
```



En aquest exemple de getContentPane() podem veure com es mostra el botó per fer el LogOut.

➤ setDefaultCloseOperation()

```
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

La trucada a aquest mètode estableix l'operació de tancament de la finestra window com a JFrame.EXIT_ON_CLOSE. Això vol dir que quan l'usuari tanqui la finestra fent clic al botó de tancament (X) a la barra de títol de la finestra, l'aplicació es tancarà del tot i finalitzarà l'execució.



CONTENIDORS INTERMEDIIS

Permeten agrupar components visuals del programa i estan dins d'un component superior.

➤ JPanel

El JPanel s'utilitza per crear un panell gràfic on es mostra el que requereix la view.

1) Exemple 1

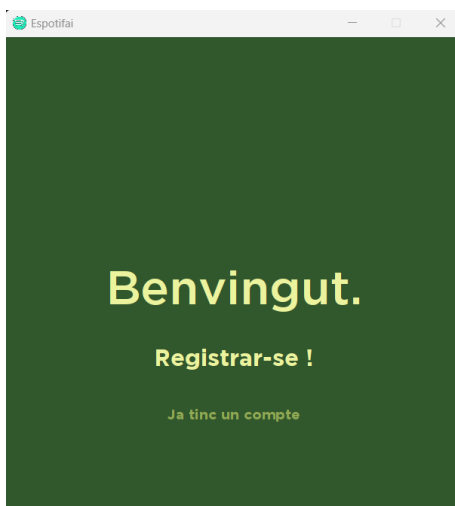
```
public class MusicStatisticsView {  
    3 usages  
    private HashMap<String, Integer> data;  
  
    13 usages  
    private final JPanel contentPane;
```

```
contentPane = new JPanel();  
contentPane.setBackground(UIPalette.COLOR_PRIMARIO.getColor());  
contentPane.setLayout(new GridBagLayout());  
GridBagConstraints c = new GridBagConstraints();  
c.fill = GridBagConstraints.BOTH;  
  
JPanel chart = new JPanel();  
//chart.add(chartPanel);  
JLabel titol = new JLabel( text: "Estadistiques.");  
titol.setFont(Fonts.getBoldFont( size: 50f));
```



En aquest cas, el JPanel s'utilitza com a contenidor per mostrar gràfics i altres elements visuals relacionats amb les estadístiques. També s'afegeix un títol al tauler utilitzant un JLabel personalitzat.

2) Exemple 2



```
private final JPanel welcomePanel;  
  
1 usage  
public WelcomeView() {  
    welcomePanel = new JPanel(new GridBagLayout());  
    welcomePanel.setBackground(UIPalette.APP_BACKGROUND.getColor());  
  
    GridBagConstraints c = new GridBagConstraints();  
    c.gridx = 0;  
    c.gridy = 0;  
    c.gridwidth = 3;
```

El JPanel anomenat welcomePanel s'utilitza com un panell de benvinguda en una interfície d'usuari.

En resum, el JPanel anomenat welcomePanel s'utilitza com a contenidor per mostrar un panell de benvinguda en una interfície d'usuari. Conté un títol, un botó per registrar-se i un altre botó per iniciar sessió, amb estils i comportaments personalitzats per a cada botó.

➤ JScrollPane

```
JScrollPane scrollPane = new JScrollPane(table);  
JScrollPane scrollPane = new JScrollPane(lyricsCancion);
```

```
JScrollPane scrollPane = new JScrollPane(lyricsCancion);  
scrollPane.setBackground(UIPalette.APP_BACKGROUND.getColor());  
scrollPane.setBorder(BorderFactory.createEmptyBorder());  
scrollPane.setPreferredSize(new Dimension( width: 750, height: 450));  
scrollPane.getViewport().setBackground(UIPalette.APP_BACKGROUND.getColor());  
scrollPane.getVerticalScrollBar().setUI(new CustomScrollBarUI());  
panelMusicInfo.add(scrollPane, c);
```

Lyrics:

People everywhere
A sense of expectation hanging in the air
Giving out a spark
Across the room, your eyes are glowing in the dark

And here we go again, we know the start, we know the end
Masters of the scene
We've done it all before
and now we're back to get some more
You know what I mean

Voulez-vous (Ah-ha)
Take it now or leave it (Ah-ha)
Now is all we get (Ah-ha)
Nothing promised, no regrets

Aquest codi, s'utilitza un JScrollPane per afegir una barra de desplaçament a un component de text, en aquest cas de lyricsCancion.

Creem un scrollPane i pasem el component de text "lyricsCancion" com argument al constructor. A més, hem dissenyat el scrollPane amb característiques específiques, com posar color de fons i eliminar la vora del scrollPane amb `BorderFactory.createEmptyBorder()`, establir dimensions i aplicar un `CustomScrollBarUI` per la visualització de la barra

En resum, el JScrollPane s'utilitza per afegir una barra de desplaçament a un component de text (lyricsCancion) i permetre que el contingut es desplaci si excedeix la mida visible del component.

- JProgressBar: S'utilitza per mostrar el progrés d'una operació, ensenyant una barra que dona l'avanç de la tasca.

```
private void addProgressBar(GridBagConstraints gbc) {  
    progressBar = new JProgressBar();  
    progressBar.setValue(40);  
    progressBar.setMaximum(200);  
    progressBar.setBackground(UIPalette.COLOR_PRIMARIO.getColor());  
    progressBar.setForeground(UIPalette.TEXT_COLOR.getColor());  
    progressBar.setBorderPainted(false);  
    gbc.gridx = 1;  
    gbc.gridy = 1; // Colocar la barra de p  
    gbc.gridwidth = 5; // Ajustar el ancho  
    gbc.ipadx = -11;  
    gbc.fill = GridBagConstraints.HORIZONTAL; // Hace que la barra  
    gbc.insets = new Insets( top: 0, left: 300, bottom: 0, right: -230);  
    contentPane.add(progressBar, gbc);  
}
```

Song
Artist



En aquest exemple, hem fet l'instància progressBar i l'hem establert el valor 40 per la barra amb un màxim de 200, a més s'ha especificat el color de fons però també de la barra, i no em inclòs la vora de la barra. La col·locació ha sigut específica per a que s'ajusti a la finestra.

2.2. LayoutManagers

Col·loquen els components de la interfície d'usuari a la finestra contenidora, especificant la posició i la mida dels components.

- **BorderLayout** : És un administrador de disseny que divideix el panell en àrees.

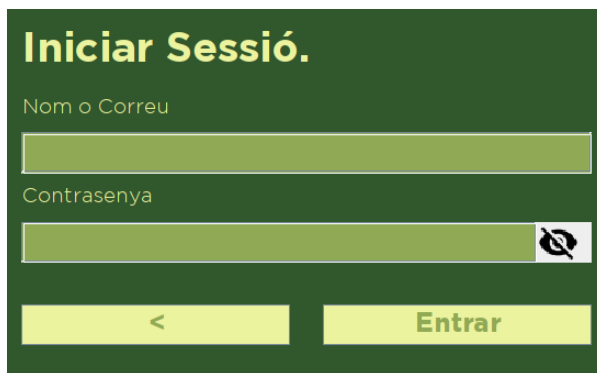
```
JPanel passwordFieldPanel = new JPanel(new BorderLayout());
passwordFieldPanel.add(jTF_contrasenya, BorderLayout.CENTER);

JToggleButton showPasswordToggle = new JToggleButton();
ImageIcon eyeClosedIcon = new ImageIcon(new ImageIcon( filename: "data/img/contra_ojo_cerrado.png").getImage().getScaledInstance( width: 30, height: 30, Image.SCALE_SMOOTH));
showPasswordToggle.setIcon(eyeClosedIcon);
showPasswordToggle.setBorder(new EmptyBorder( top: 0, left: 0, bottom: 0, right: 10));
showPasswordToggle.setContentAreaFilled(false);
showPasswordToggle.setFocusable(false);
ImageIcon eyeOpenIcon = new ImageIcon(new ImageIcon( filename: "data/img/contra_ojo_abierto.png").getImage().getScaledInstance( width: 30, height: 30, Image.SCALE_SMOOTH));

showPasswordToggle.addActionListener(e -> {
    if (showPasswordToggle.isSelected()) {
        jTF_contrasenya.setEchoChar((char) 0);
        showPasswordToggle.setIcon(eyeOpenIcon);
    } else {
        jTF_contrasenya.setEchoChar(defaultEchoChar);
        showPasswordToggle.setIcon(eyeClosedIcon);
    }
});

passwordFieldPanel.add(showPasswordToggle, BorderLayout.EAST);
panelSignin.add(passwordFieldPanel, c);

passwordFieldPanel.add(showPasswordToggle, BorderLayout.EAST);
panelSignin.add(passwordFieldPanel, c);
```



Hem fet un JPanel utilitzant el BorderLayout com el seu administrador. En aquest cas es mostra amb el component de contrasenya, col·locant-lo al centre del panel ocupant tot l'espai disponible. Utilitzem el BorderLayout.EAST per afegir el showPasswordToggle a la dreta del panel.

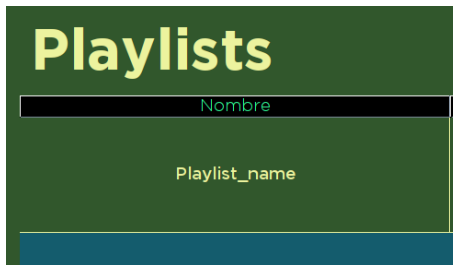
- **BoxLayout**: BoxLayout és un administrador de disseny que organitza els components en una sola columna o una sola fila.

```
contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.Y_AXIS));
// Create a new BoxLayout with a vertical orientation
setLayout(new BoxLayout(this, BoxLayout.Y_AXIS);
```

S'estableix l'administrador de disseny del contentPane (el panell de contingut) com a BoxLayout amb la

constant BoxLayout.Y_AXIS. Això vol dir que els components afegits al contentPane s'organitzaran en una única columna vertical, de dalt a baix.

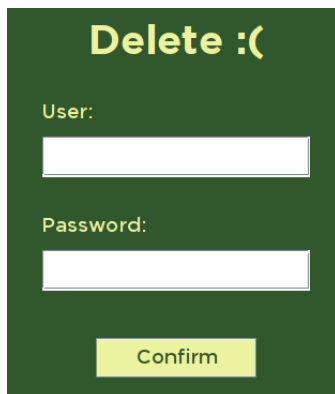
```
public PlaylistView() {
    this.contentPane = new JPanel();
    contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.Y_AXIS));
    contentPane.setBorder(BorderFactory.createEmptyBorder( top: 50, left: 50, bottom: 10, right: 10));
    Playlist = new JLabel( text: "Playlist");
    Playlist.setFont(Fonts.getBoldFont( size: 50f));
    Playlist.setForeground(Color.BLACK);
    contentPane.add(Playlist);
}
```



La playList s'afegeix al contentPane perquè té BorderLayout amb direcció vertical.

- **FlowLayout**: és un administrador de disseny per defecte d'un JPanel. Ensenya els components en una única fila (segons la mida del contenidor), o també te més files per mostrar tots.

```
JPanel buttonsPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
```



En aquest cas el FlowLayout és l'administrador de disseny de ButtonsPanel utilitzant FlowLayout.CENTER per a que els components es posin en una fila i es centrin horitzontalment dins del panel.

- **GridBagLayout**:

Alinea els components amb una graella de cel·les ocupant més d'una fila/columna.

```
welcomePanel = new JPanel(new GridBagLayout());
panelAddSong = new JPanel(new GridBagLayout());
panelDeleteSong = new JPanel(new GridBagLayout());
panelList = new JPanel(new GridBagLayout());
menuPanel.setLayout(new GridBagLayout());
contentPane.setLayout(new GridBagLayout());
contentPane.setLayout(new GridBagLayout());
contentPane.setLayout(new GridBagLayout());
panelMusicInfo = new JPanel(new GridBagLayout());
panelSignin = new JPanel(new GridBagLayout());
panelSignup = new JPanel(new GridBagLayout());
```

```
public SignupView() {
    panelSignup = new JPanel(new GridBagLayout());
    GridBagConstraints c = new GridBagConstraints();
    panelSignup.setBackground(UIPalette.APP_BACKGROUND.getColor());
    panelSignup.setBorder(new EmptyBorder( top: 0, left: 0, bottom: 0, right: 0));

    Font fTitle = Fonts.getBoldFont( 30f);
    Font fLowerText = Fonts.getLightFont( 15f);

    JLabel registerTitle = createLabel(fTitle);
    registerTitle.setForeground(UIPalette.TEXT_COLOR.getColor());
    tfEmail = createTemplateField( labelText: "Email", isPassword: false, fLowerText);
    tfEmail.setForeground(UIPalette.TEXT_COLOR.getColor());
    tfUsername = createTemplateField( labelText: "Username", isPassword: false, fLowerText);
    tfUsername.setForeground(UIPalette.TEXT_COLOR.getColor());
    tfFirstPassword = createTemplateField( labelText: "Password", isPassword: true, fLowerText);
    tfFirstPassword.setForeground(UIPalette.TEXT_COLOR.getColor());
    tfSecondPassword = createTemplateField( labelText: "Password Confirmation", isPassword: true, fLowerText);
    tfSecondPassword.setForeground(UIPalette.TEXT_COLOR.getColor());
}
```

Aquí s'ha posat el `gridBagLayout` com administrador de disseny del `panelSignup`. I hem afegit components.

- **GridLayout**: Els components que es mostren en aquest administrador de disseny tenen la mateixa mida.

```
GridLayout gridLayout = new GridLayout(12, 1);
gridLayout.setVgap(10); // indica los gaps entre las filas
panelLogout.setLayout(gridLayout); // les decimos el diseño que queremos en nuestro panel

public DeleteUserView() {
    GridBagLayout gridLayout = new GridBagLayout();
    GridBagConstraints c = new GridBagConstraints();

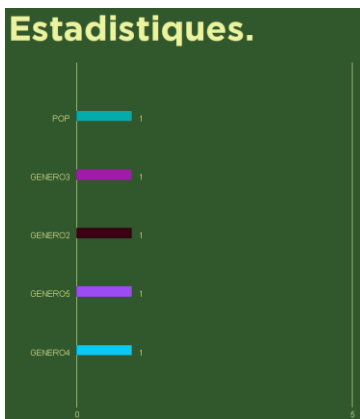
    panelLogout = new JPanel();
    panelLogout.setLayout(gridLayout);
    panelLogout.setBackground(UIPalette.COLOR_PRIMARIO.getColor());

    Font titulo = Fonts.getBoldFont( size: 30f);
    Font subtítulo = Fonts.getMediumFont( size: 15f);

    JLabel tituloLogout = new JLabel( text: "Delete :(");
    tituloLogout.setForeground(UIPalette.TEXT_COLOR.getColor());
    tituloLogout.setFont(titulo);
}
```

- **CardLayout**: Es poden veure varis components en espais de temps diferents.

```
public void setMusicStatisticsView(){
    mainPanel.CardLayout.show(panelSuperiorDerecha, name: "musicStatistics");
}
```



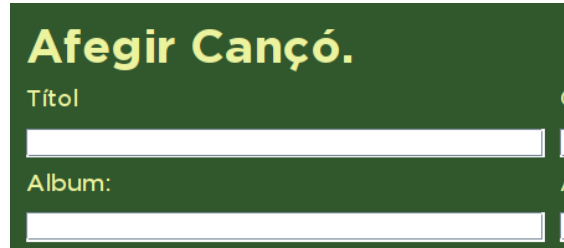
En aquest exemple, s'utilitza el `cardLayout` per canviar el panel que es mostra en el `panelSuperiorDerecha`. Amb això, s'indica que s'ha de mostrar el component `musicStatics`

2.3. Components:

Son elements visuals de la interfície i hereten de la classe JComponent.

- **JTextField**: És un component lleuger amb el que pots editar una línia de text.

```
jTFSongTitleText = new JTextField();
//jTFSongTitleText.setPreferredSize(new Dimension(50, 30));
jTFSongTitleText.setFont(fuente_petit);
c.ipady = 0;
c.gridx = 0;
c.gridy = 2;
c.gridwidth = 1;
c.insets = new Insets( top: 0, left: 10, bottom: 10, right: 0);
panelAddSong.add(jTFSongTitleText, c);
```



- **JLabel**: S'utilitza per mostrar el text o imatges. En aquest exemple l'hem fet servir per mostrar el text "url" en el panelAddSong.

```
JLabel url = new JLabel( text: "URL:");
url.setForeground(UIPalette.TEXT_COLOR.getColor());
url.setFont(fuente_petit);
c.ipady = 0;
c.gridx = 0;
c.gridy = 5;
c.gridwidth = 7;
c.insets = new Insets( top: 0, left: 10, bottom: 10, right: 0);
panelAddSong.add(url, c);
```



- **JOptionPane**: S'utilitza per crear i mostrar quadres de missatge interactives. En el següent exemple és per un missatge d'error.

```
public void wrongPasswordError() {
    JOptionPane.showMessageDialog( parentComponent: this,
        message: "<html><body><p style='width: 250px;'>Invalid password! Your password must:<br>" +
            "<ul>" +
            "<li>Be at least 8 characters long</li>" +
            "<li>Contain at least one lowercase and one uppercase letter</li>" +
            "<li>Include at least one numeric character</li>" +
            "<li>Not contain any whitespaces</li>" +
            "</ul>" +
            "Please try again.</p></body></html>",
        title: "Invalid Password", JOptionPane.ERROR_MESSAGE);
}
```

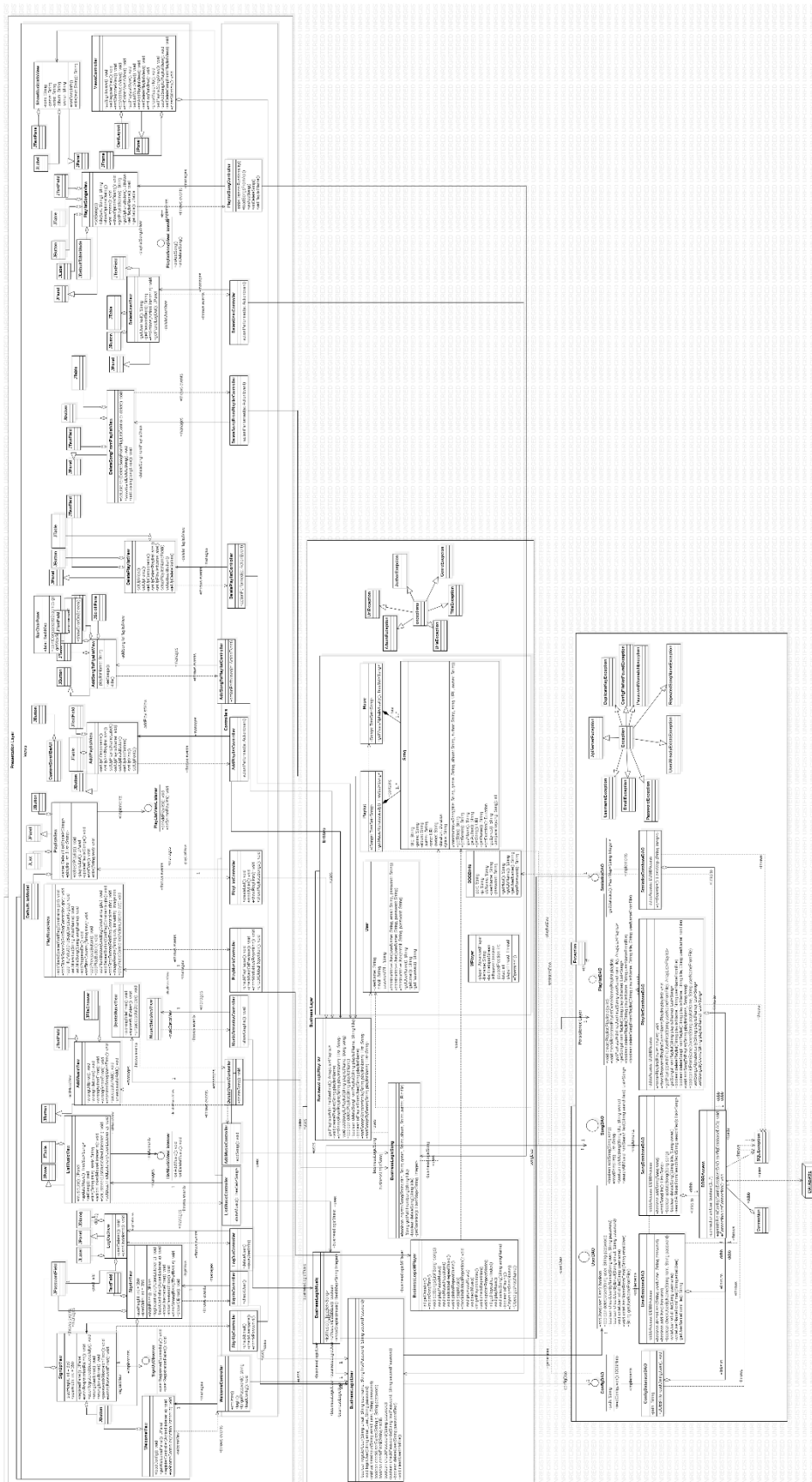


- **JFileChooser**: És un quadre de selecció d'arxius des del sistema. En aquest exemple s'afegeix al panelAddSong.

```
JFileChooser jTF_song_url = new JFileChooser();
jTF_song_url.setPreferredSize(new Dimension( width: 200, height: 350));
jTF_song_url.setFont(fuente_petit);
c.ipady = 0;
c.gridx = 0;
c.gridy = 6;
c.gridwidth = 2;
c.gridheight = 1;
c.insets = new Insets( top: 0, left: 0, bottom: 10, right: 0);
panelAddSong.add(jTF_song_url, c);
```



3. Diagrama de classes



El proyecto presentado se encuentra estructurado en una arquitectura en capas conformada por los paquetes de Presentación, Negocio y Persistencia, en el que se implementa el patrón de diseño Modelo-Vista-Controlador.

Paquete de Presentación:

En este paquete, se encuentran varias clases "View", cada una representando una funcionalidad específica de la aplicación. Se tiene también una clase "ViewController", responsable de administrar los cambios de vista mediante el uso del patrón CardLayout.

Las relaciones establecidas aquí son principalmente de dependencia con los correspondientes "Controllers" en el paquete de Negocio, lo que indica que cada "View" conoce a su "Controller" correspondiente.

Paquete de Negocio (Business):

Aquí se encuentran varias clases "Controller", cada una correspondiendo a una clase "View" del paquete de Presentación. Cada "Controller" es responsable de la lógica de negocio relacionada con su "View" correspondiente.

Además, se tienen clases específicas de lógica de negocio que realizan operaciones relacionadas con la reproducción de música, gestión de canciones, usuarios y listas de reproducción.

Las clases "Controller" tienen una relación direccional con las clases "View" correspondientes en el paquete de Presentación y con las clases de lógica de negocio en el paquete de Negocio. Esto indica que cada "Controller" instancia su "View" correspondiente y la respectiva clase de lógica de negocio.

Paquete de Persistencia:

Este paquete contiene diversas clases DAO que interactúan con la base de datos para realizar operaciones de CRUD (Crear, Leer, Actualizar, Borrar) en diferentes entidades. Cada clase DAO implementa una interfaz correspondiente, lo que sugiere una relación de generalización.

Además, se tienen las clases "APIAccess" y "DDBBAccess" que interactúan respectivamente con la API y la base de datos.

Las clases de lógica de negocio del paquete de Negocio tienen una relación de dependencia con las entidades y las clases DAO en el paquete de Persistencia. Esto significa que estas clases de lógica de negocio conocen a las entidades y las clases DAO correspondientes.

Entidades:

Las entidades representan conceptos importantes en la aplicación, incluyendo "Music", "Playlist", "Song", "User" y "MPlayer".

Excepciones personalizadas:

Se tienen excepciones personalizadas en los paquetes de Persistencia y Negocio que extienden de la clase base Exception de Java.

De manera general, las relaciones entre las clases y los paquetes en el StarUML se representan mediante líneas que indican la dirección de las relaciones. Las relaciones de dependencia se indican con una línea punteada y una flecha, mientras que las relaciones direccionales se indican con una línea sólida y una flecha.

Queremos añadir, que para una mejor visualización, hemos añadido el diagrama en la carpeta de entrega.

4. Metodologia de desenvolupament

El projecte ha tingut quatre sprints, dels quals els hem organitzat amb ajuda del nostre mentor. Ell ens suggeria les tasques més necessàries a fer i nosaltres confirmàvem si ens anava bé treballar-ho durant aquelles setmanes. A continuació, ensenyem les tasques realitzades en cada iteració i expliquem com ens hem repartir aquesta feina, però també hem de tenir en compte que potser cada tasca es va començar en un sprint i es va acabar en el següent, i contem només el sprint on el vam començar.

SPRINTS	TASQUES REALITZADES
SPRINT 1 (13/03 – 26/03)	<ul style="list-style-type: none">• Començament del “class diagram” (Es va realitzar durant el sprint 1 però també s’ha modificat durant la resta de sprints).• UI mockups (Es van realitzar durant el sprint 1 però també s’ha modificat durant la resta de sprints).• Database design• Configuration file
SPRINT 2 (11/04 – 23/04)	<ul style="list-style-type: none">• Database setup• Database information access• User sign-up GUI• User sign-up• User login GUI• Logout and account deletion GUI• User login• Logout
SPRINT 3 (24/04 – 14/05)	<ul style="list-style-type: none">• Delete user account• Main GUI• List Stored Songs• Song File Loading• Add a New Song• Delete a Song







SPRINT 4 (15/05 – 28/05)

- Project report
- Detailed Song View
- Fetching the Lyrics
- Playlist Management
- Music Statistic
- Music Playback

A cada Sprint tots els membres teníem una feina a fer, ens assignàvem una o dues tasques segons la carrega de treball, a més també hi havia molt companyerisme perquè si a algú no arribava el temps per a fer la seva part, ens comunicàvem i ens ajudàvem.

Però també per organitzar-nos, ens hem dividit las tasques segons les eines de software que hem utilitzat és a dir, si en el segon sprint un membre s'havia dedicat a base de dades, al següent sprint es dedicava a swing. Encara que no sempre ha sigut d'aquesta manera, és el procés que més s'ha intentat seguir.

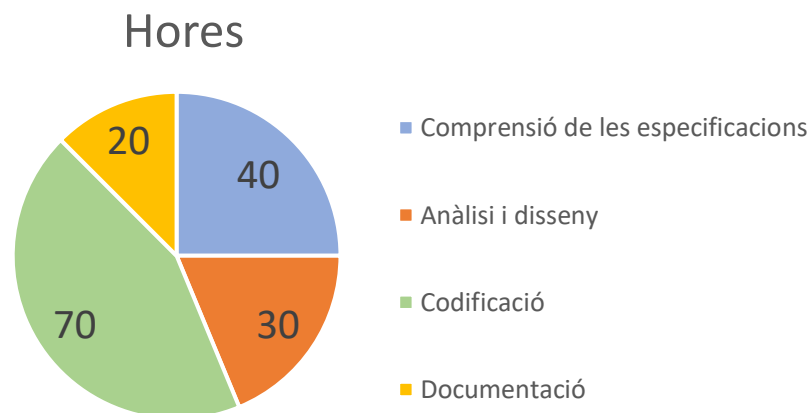
Les eines que hem utilitzat en aquest projecte son les següents:

 Jira Software	JIRA: L'hem utilitzat per organitzar el treball i fer un seguiment del que estem elaborant i de qui ho està realitzant.
 Bitbucket	BITBUCKET: L'hem utilitzat per compartir el codi i posar en comú el treball que cadascú avançava de forma personal.
	IntelliJ: L'hem utilitzat per elaborar el projecte, aquí hem programat amb java i swing i és on s'executa el programa.
 XAMPP	XAMPP: L'hem utilitzat per connectar-nos a la base de dades.
	phpMyAdmin: L'hem utilitzat per crear la base de dades i per fer les consultes MySQL.
 Figma	FIGMA: L'hem utilitzat per fer el mockup del programa, i aquest disseny creat l'hem tingut com a referència per programar.
	WORD: L'hem utilitzat per realitzar la documentació.

5. Dedicació

Si fem una estimació de temps emprat en total per la realització del projecte, considerem que arribem com a mínim a les 100 hores, tenint en compte que estem sumant el temps de treball per cada membre del grup quan treballàvem per separat, i el temps que ho fèiem junts s'ha comptat com si fóssim una única persona.

Per la **comprensió de les especificacions** hem considerat que durant tot el projecte havíem d'entendre el que demanava, així que cada setmana invertíem hores parlant i preguntant sobre aquestes especificacions, per aquesta raó durant onze setmanes, entre tots hem fet aproximadament 40 hores de comprensió. Per l'**anàlisi i disseny** assegurem que les tres primeres setmanes ens hem bolcat en acabar-



ho per tenir-lo com a estructura per poder començar a fer codi, però durant la resta del projecte també havíem d'analitzar i dissenyar per això augmenta unes poques hores més. Per la **codificació** s'asseguren quatre hores de treball en conjunt, les dues hores de classe del dijous i dues hores el divendres, per tant arribaríem a quaranta-quatre hores com a mínim. Si continuem aproximant cadascun ha treballat per el seu compte, arribant a unes setanta hores de codificació. Finalment la **documentació** l'hem realitzat en diferents dies, tenint una acumulació de vint hores dedicades entre tots.

6. Conclusions

Durant el desenvolupament d'aquest projecte de software considerem que hem arribat a aconseguir diferents objectius significatius.

Havíem de comprendre el projecte, així que a l'hora de valorar els requeriments funcionals, hem aplicat la metodologia Scrum, un repte que en el grup ha estat còmode de treballar perquè ja teníem coneixement previ. Hem trobat dificultós el primer disseny del diagrama de classes, tractant de relacionar classes i trobar elements necessaris i que no siguin no funcionals per el sistema. Però la superació d'aquest impediment ha aconseguit que el desenvolupament de codi sigui més adaptable i s'apropi més al que volíem, encara que si ens referim a similitud, no és el cas dels Mockups de la UI, que ens ha donat una idea principal per l'organització dels botons y textos, però per programar-ho a la seva totalitat ens hem adaptat a Swing més que al disseny. Per la base de dades sí que s'ha respectat més l'estructura principal,

aconseguint des d'abans de picar codi les funcionalitats necessàries i connectant-la amb el projecte, fent més efectiu el treball.

Amb aquest projecte també hem assolit creixement personal, al llarg d'aquest semestre hem après no tan sols a programar, sinó a compartir coneixements amb un grup de treball. Aconseguint experiències i habilitats amb la resolució de problemes, presa de decisions i gestió del temps. També ens hem adonat que no tothom treballa amb la mateixa metodologia i al principi va ser difícil comprendre'ns, però com teníem el mateix objectiu, hem adquirit la capacitat de treballar en un entorn divers amb empatia, on l'esforç, la dedicació i el treball constant ha merescut la pena rere veure els resultats del nostre projecte. Per això, hem generat un sentiment d'assoliment personal, més una motivació a continuar prenent nous reptes i projectes en el futur.

7. Bibliografia consultada

Bibliogúas Biblioteca Universidad de Alcalá. Citar y elaborar bibliografía: Estilo ISO 690:2010 [en línea] [fecha de consulta: 12 mayo 2023]. Disponible en:

https://uahes.libguides.com/citar_elaborar_bibliografia/iso

DuarteCorporation Tutoriales (2015). Tutorial JTable en Lenguaje Java con IntelliJ IDEA [en línea]. En: YouTube [fecha de consulta: 11 mayo 2023]. Disponible en:

<https://www.youtube.com/watch?v=fOZ1XeCWLcE&list=LL&index=2>

Java Code Geeks. Java Swing Layouts Example [en línea] [fecha de consulta: 07 abril 2023]. Disponible en: <https://examples.javacodegeeks.com/java-swing-layouts-example/>

Oracle. A Visual Guide to Layout Managers [en línea] [fecha de consulta: 25 abril 2023]. Disponible en: <https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>

Oracle. How to Make Frames(Main Windows) [en línea] [fecha de consulta: 30 marzo 2023]. Disponible en: <https://docs.oracle.com/javase/tutorial/uiswing/components/frame.html>

Oracle. How to Make Frames(Main Windows) [en línea] [fecha de consulta: 11 abril 2023]. Disponible en: <https://docs.oracle.com/javase/tutorial/uiswing/components/frame.html>

Oracle. Lesson: Laying Out Components Within a Container [en línea] [fecha de consulta: 23 abril 2023]. Disponible en: <https://docs.oracle.com/javase/tutorial/uiswing/layout/>

Oracle. Lesson: Using Swing Components [en línea] [fecha de consulta: 3 abril 2023]. Disponible en: <https://docs.oracle.com/javase/tutorial/uiswing/components/index.html>

Oracle. Using Layout Managers [en línea] [fecha de consulta: 13 mayo 2023]. Disponible en: <https://docs.oracle.com/javase/tutorial/uiswing/layout/using.html>

Oracle. Using Top-Level Containers [en línea] [fecha de consulta: 4 mayo 2023]. Disponible en: <https://docs.oracle.com/javase/tutorial/uiswing/components/toplevel.html>

Study tonight. Java Swing Components and Containers [en línea] [fecha de consulta: 5 mayo 2023]. Disponible en: <https://zetcode.com/javaswing/basicswingcomponentsII/>

Tutorialspoint. Swing Layout [en línea] [fecha de consulta: 28 marzo 2023]. Disponible en:
https://www.tutorialspoint.com/swing/swing_layouts.htm

Universidad de Alicante. La norma ISO 690:2010(E) [en línea] [fecha de consulta: 12 mayo 2023].
Disponible en: http://werken.ubiobio.cl/html/downloads/ISO_690/Guia_Breve_ISO690-2010.pdf

ZetCode. Basic Swing components [en línea] [fecha de consulta: 23 marzo 2023]. Disponible en:
<https://zetcode.com/javaswing/basicswingcomponents/>

ZetCode. Basic Swing components II [en línea] [fecha de consulta: 20 abril 2023]. Disponible en:
<https://zetcode.com/javaswing/basicswingcomponentsII/>

ZetCode. Java Swing [en línea] [fecha de consulta: 20 abril 2023]. Disponible en:
<https://zetcode.com/javaswing/>