Manual de Usuario del Reproductor de Música `Spotifaker`

Desarrollado por: Equipo de Desarrollo Spotifaker

Noviembre 2024

1 Introducción

El presente manual describe el funcionamiento y las características de la interfaz de usuario del reproductor de música `Spotifaker`. A continuación, se detallan los botones y elementos de control.

2 Descripción de la Interfaz

La interfaz del reproductor de música cuenta con los siguientes componentes:

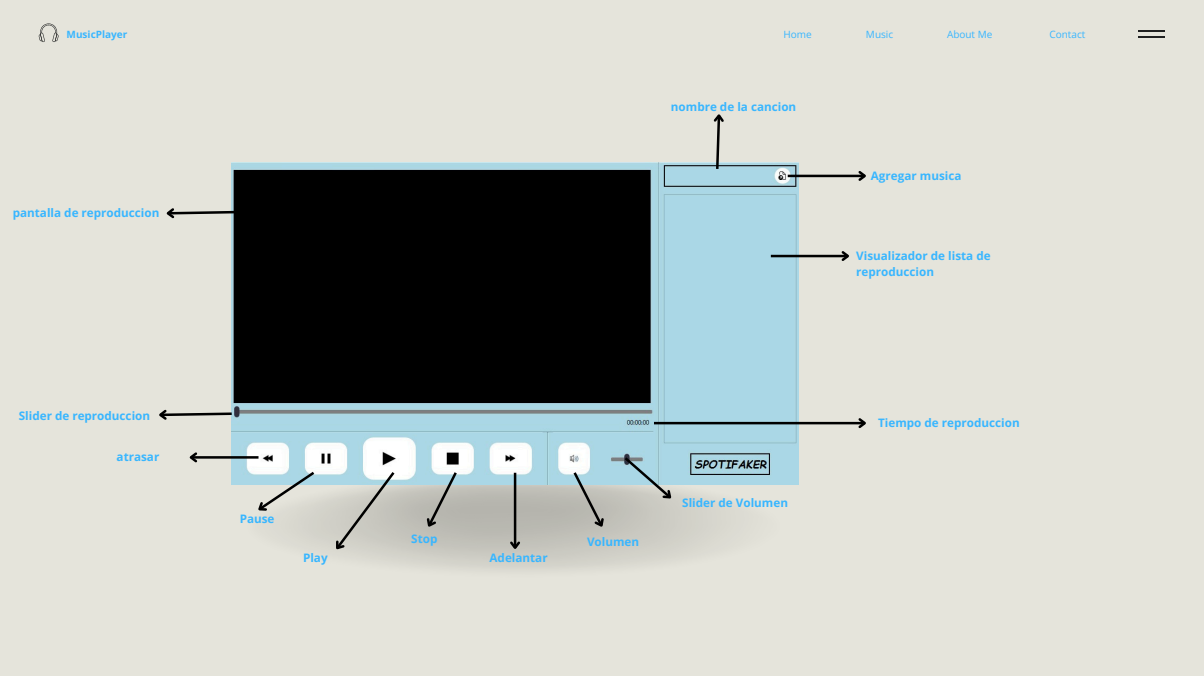

Figure 1: Interfaz del reproductor de música `Spotifaker`

- **Pantalla de visualización**: Muestra el contenido multimedia actual (por ejemplo, nombre de la canción, artista o video si es compatible).
- **Controles de reproducción**: Situados en la parte inferior, permiten controlar la reproducción de la música.
- **Barra de volumen**: Situada a la derecha de los controles de reproducción, permite ajustar el volumen.
- **Barra de progreso**: Situada en la parte inferior de la pantalla, permite visualizar y controlar la posición actual de la canción.

- **Tiempo de reproducción**: Situado en la esquina inferior derecha de la pantalla, muestra el tiempo transcurrido de la pista actual.
- **Lista de reproducción**: Muestra los archivos multimedia disponibles para reproducir y permite seleccionarlos.
- **Pantalla de Video**: Muestra contenido de video si el archivo multimedia incluye esta información.

3 Función de Cada Botón

3.1 Botones de Control

Los botones de control en el reproductor de música permiten manejar la reproducción de la pista actual. Los botones disponibles son los siguientes:

- **Rebobinar (¡¡)**: Permite retroceder a la canción anterior.
- **Pausa (——)**: Pausa la reproducción de la música.
- **Reproducir ()**: Inicia o reanuda la reproducción de la música.
- **Detener ()**: Detiene la reproducción de la música y reinicia la pista al principio.
- **Avanzar (¿¿)**: Permite avanzar a la siguiente canción.

3.1.1 Función Play

El botón de reproducción inicia o reanuda la reproducción de la música. Cuando el reproductor está en pausa o detenido, hacer clic en este botón comenzará a reproducir la canción desde donde se dejó.

Listing 1: Código de la Función Play en C++

```
void Widget::on_pushButton_play_clicked()
{
    mMediaPlayer->play();  // Inicia la reproducci n de la m sica
}
```

3.1.2 Función Pause

El botón de pausa detiene la reproducción de la música temporalmente. La música se pausa y se mantiene en la misma posición, por lo que puede reanudarse desde el mismo lugar cuando se presiona nuevamente el botón de reproducción.

Listing 2: Código de la Función Pause en C++

```
void Widget::on_pushButton_pause_clicked()
{
    mMediaPlayer->pause();  // Pausa la reproducci n de la m sica
}
```

3.1.3 Función Stop

El botón de detención detiene la música y la coloca de nuevo al inicio. Cuando el usuario presiona este botón, la canción se detiene y se reinicia si se vuelve a reproducir.

Listing 3: Código de la Función Stop en C++

```
void Widget::on_pushButton_stop_clicked()
{
    mMediaPlayer->stop();  // Detiene la reproducci n y reinicia la canci n
}
```

3.2 Control de Volumen

El control de volumen, representado con el icono de altavoz y la barra deslizante, permite ajustar el nivel de volumen de la música. Mueva el deslizador hacia la derecha para aumentar el volumen y hacia la izquierda para disminuirlo.

A continuación se muestra un fragmento de código que ilustra cómo se configura el control de volumen en el sistema:

Listing 4: Código de Control del Volumen en C++

```
void Widget::on_vol_valueChanged(int value)
{
    audioOutput->setVolume(value / 100.0); // Ajustar el volumen de 0.0 a 1.0
}

// Configuraci n inicial del volumen
ui->vol->setRange(0, 100); // Rango de volumen de 0 a 100
ui->vol->setValue(50);     // Valor inicial en 50

audioOutput = new QAudioOutput(this);
```

3.3 Barra de Progreso

La barra de progreso permite visualizar la posición actual de la canción y ajustar la misma moviendo el deslizador. A continuación, se muestra el código relacionado con la funcionalidad de la barra de progreso:

Listing 5: Código de Control del Deslizador de Reproducción en C++

```
void Widget::on_positionChanged(qint64 position)
{
    // Actualiza el slider de progreso de acuerdo con la posici n actual de la canci n
    if (mMediaPlayer->duration() > 0) {
        ui->progressSlider->setValue(static_cast<int>((position * 1000) / mMediaPlayer->
            duration()));
    }
}

void Widget::on_sliderMoved(int value)
{
    // Cambia la posici n de la canci n de acuerdo con el valor del slider
    qint64 newPosition = (value * mMediaPlayer->duration()) / 1000;
    mMediaPlayer->setPosition(newPosition);
}

void Widget::durationChanged(qint64 duration)
{
    mDuration = duration;  // Ahora en milisegundos
    ui->progressSlider->setMaximum(mDuration);
}
```

3.4 Temporizador de Música

El temporizador muestra el tiempo transcurrido de la pista actual en un formato adecuado (minutos y segundos o horas, minutos y segundos, según corresponda). A continuación, se muestra el código del temporizador de música:

Listing 6: Código del Temporizador de Música en C++

```
void Widget::updateDuration(qint64 position)
{
    qint64 currentSecond = (position / 1000);  // Convierte posici n de milisegundos a
        segundos

    if (currentSecond != lastSecond)  // Solo actualiza si el segundo ha cambiado
    {
        lastSecond = currentSecond;   // Actualiza la  ltima  posici n del segundo

        QTime currentTime((position / 3600000) % 60, (position / 60000) % 60, (position
            / 1000) % 60);
```

```
        QString format = mDuration > 3600 ? "hh:mm:ss" : "mm:ss";
        ui->label_time->setText(currentTime.toString(format));
    }
}
```

4 Lista de Reproducción

A continuación se muestra cómo se gestiona la lista de reproducción, mostrando los archivos disponibles y gestionando la selección:

Listing 7: Código de la Lista de Reproducción en C++

```
void Widget::on_listView_clicked(const QModelIndex &index)
{
    // Obtiene la ruta completa del archivo basado en el  ndice  de la lista
    QString selectedFilePath = filePathList.at(index.row());

    // Configura la fuente del archivo seleccionado y actualiza la etiqueta con el
        nombre del archivo
    mMediaPlayer->setSource(QUrl::fromLocalFile(selectedFilePath));
    ui->label_file_name->setText(fileNameList.at(index.row()));

    // Reproduce el archivo seleccionado
    mMediaPlayer->play();
}

// Listas para almacenar los nombres de archivo y sus rutas completas
QStringList fileNameList;  // Nombres de archivo para mostrar en la interfaz
QStringList filePathList;  // Rutas completas de los archivos para reproducci n

Widget::Widget(QWidget *parent)
    : QWidget(parent)            // Llama al constructor de la clase base QWidget.
    , ui(new Ui::Widget)         // Crea una nueva instancia de la interfaz de usuario
        definida en 'ui_widget.h'.
    , listModel(new QStringListModel(this))
```

4.1 Configuración de la Pantalla de Video

El reproductor también permite la visualización de contenido multimedia con video. A continuación, se detalla cómo se configura la pantalla de video:

Listing 8: Código de Configuración de la Pantalla de Video en C++

```
void Widget::setupVideoPlayer()
{
    // Inicializar el reproductor y la salida de audio.
    mMediaPlayer = new QMediaPlayer(this);
    audioOutput = new QAudioOutput(this);
    mMediaPlayer->setAudioOutput(audioOutput);

    // Busca y configura el widget de pantalla de video.
    screen = findChild<QWidget*>("screen");
    videoWidget = new QVideoWidget(screen);
    mMediaPlayer->setVideoOutput(videoWidget);  // Asocia el reproductor al videoWidget
    videoWidget->setGeometry(0, 0, screen->width(), screen->height());
}
```