

Instituto tecnológico de Oaxaca

Ing.\_Sistemas computacionales

Materia: Tópicos Avanzados de Programación

Reporte de componente (Reproducción de texto a voz)

Integrantes:

Montesinos Diaz Victor Alfredo

Muños Audelo Erick Eli



# Contenido

<b>Introducción .....</b>	<b>3</b>
<b>Lector de texto a voz.....</b>	<b>4</b>
<b>Creación .....</b>	<b>5</b>
<b>Visualización .....</b>	<b>9</b>
<b>Conclusión.....</b>	<b>12</b>

## **Introducción**

En este proyecto, exploraremos los componentes visuales de una interfaz de usuario que facilita el uso de un programa, permitiendo que el usuario interactúe de manera intuitiva con sus funciones. Este reporte documenta lo aprendido en la materia de Tópicos Avanzados de Programación, aplicado en el desarrollo de un programa que emplea JFrame para el diseño de la interfaz gráfica.

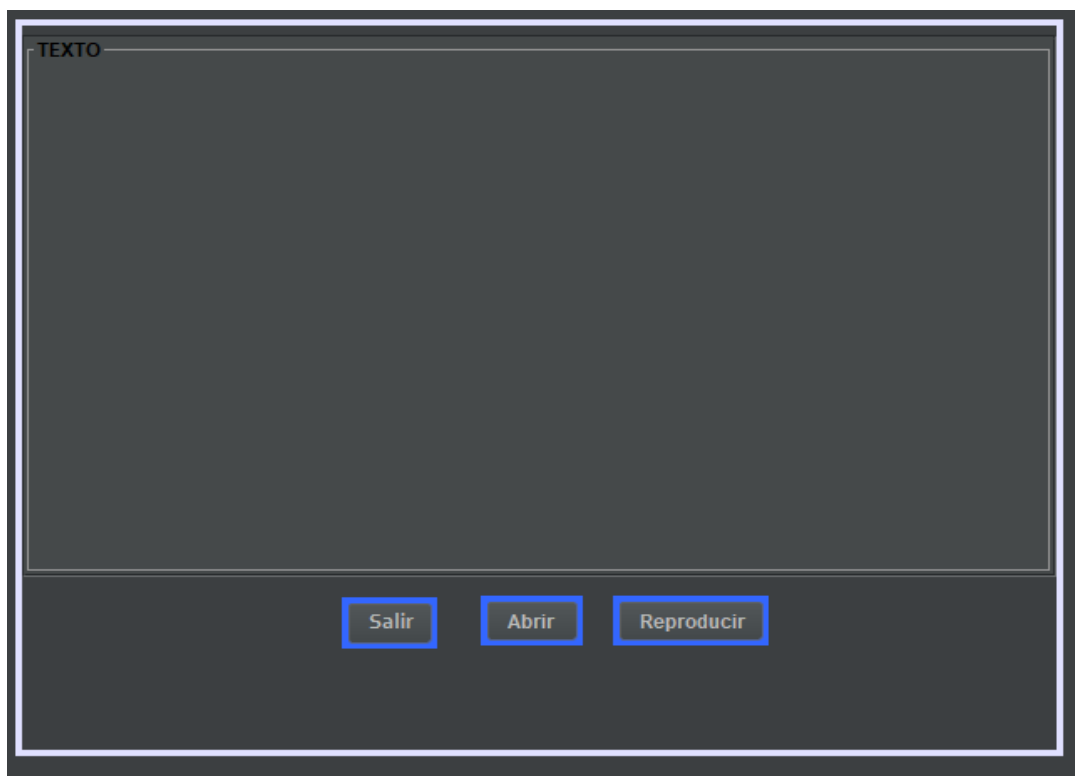
El programa contará con botones y un área de texto (TextArea) para mostrar el contenido de un archivo. El usuario podrá abrir y visualizar archivos, los cuales se leerán mediante una voz generada desde una librería especializada. Esta característica permite que el archivo seleccionado se lea en voz alta, mejorando la accesibilidad y la experiencia de uso.

## Lector de texto a voz

Este programa implementará un mecanismo que permite abrir archivos desde una ventana de interfaz y reproducir una voz que leerá cada línea del texto contenido en el archivo.

Al presionar el botón "Abrir", el usuario será dirigido a una ventana para seleccionar un archivo de su preferencia. Una vez elegido y confirmado con la opción "Abrir", el contenido del archivo se mostrará en la interfaz principal. A continuación, al hacer clic en el botón "Reproducir", una voz automatizada comenzará a leer el texto en voz alta, línea por línea, proporcionando una experiencia auditiva del contenido.

Este flujo de acciones facilita tanto la visualización como la escucha del archivo, optimizando la accesibilidad y la interacción en la aplicación.



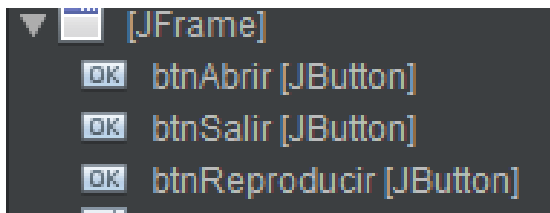
## Creación

Ocuparemos lo siguiente

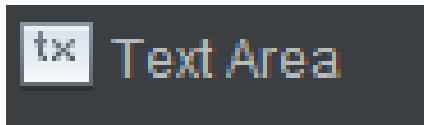
**1.1.-**Tres botones los cuales los llamaremos Abrir, Salir y Reproducir cada uno con su variable llamado “btnAbrir”, “btnSalir” y “btnReproducir”



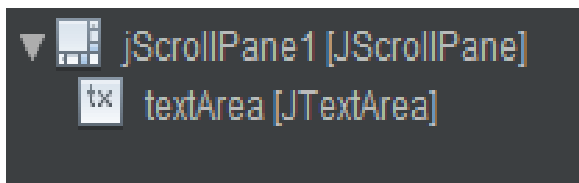
**1.2.-**Como a continuación tiene que quedar como en la siguiente forma, como en la imagen.



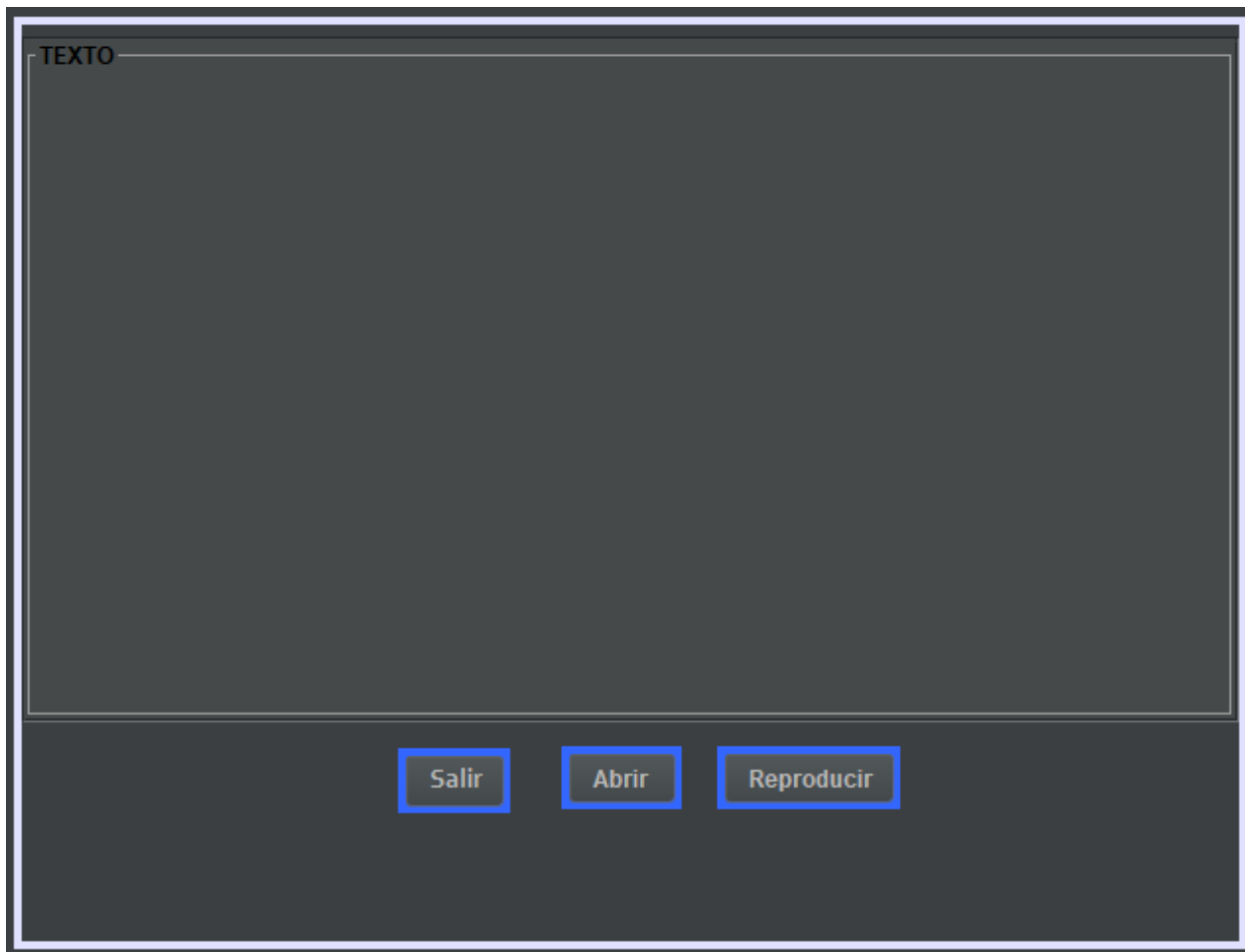
**1.3.-**Después colocaremos lo que es un Text Area con esto nos ayudara a visualizar lo que contenga el archivo al seleccionarlo.



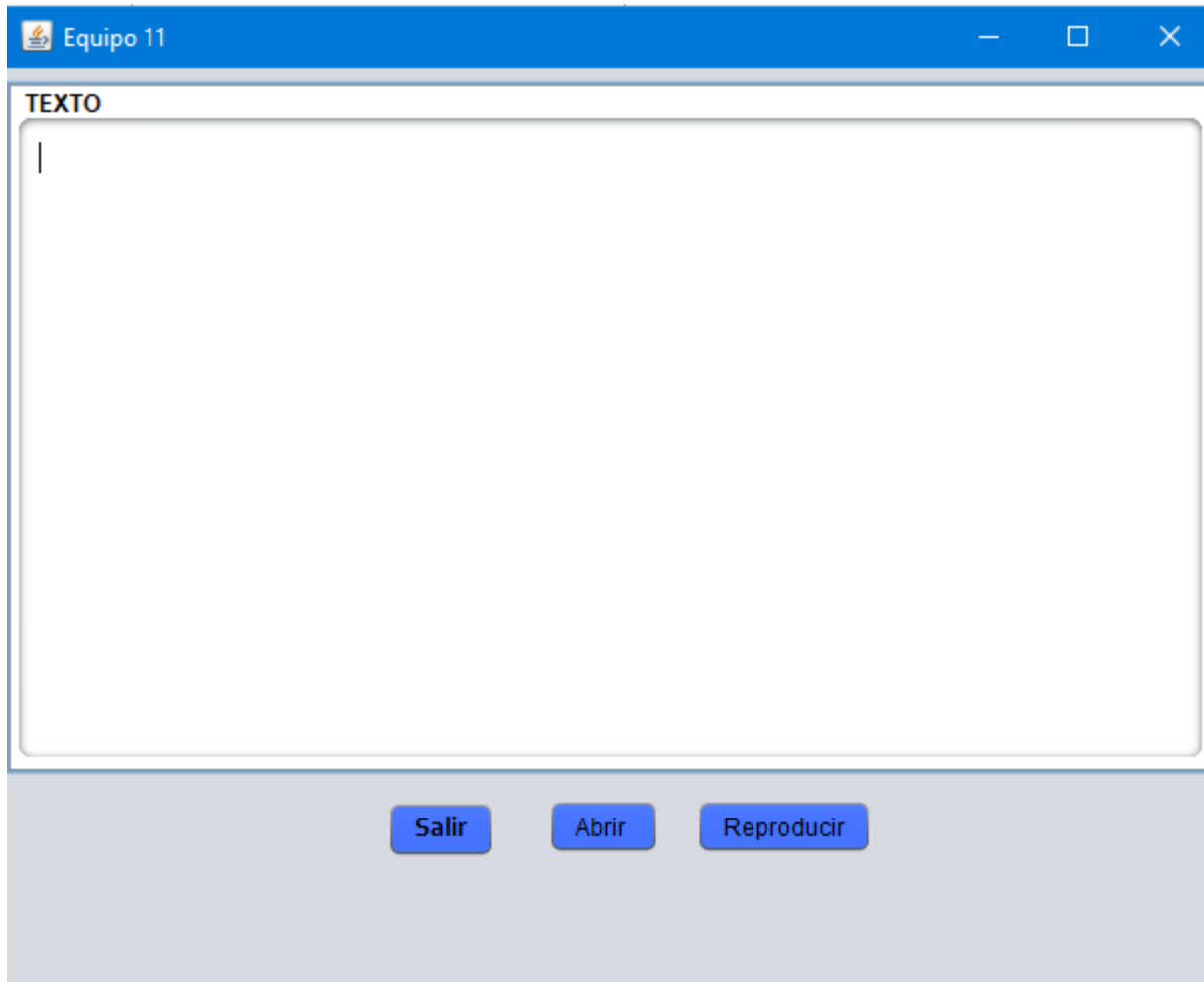
**1.4.-**Tendremos que llamar el variable como” textArea” para poderlo llamar para poder visualizar el contenido.



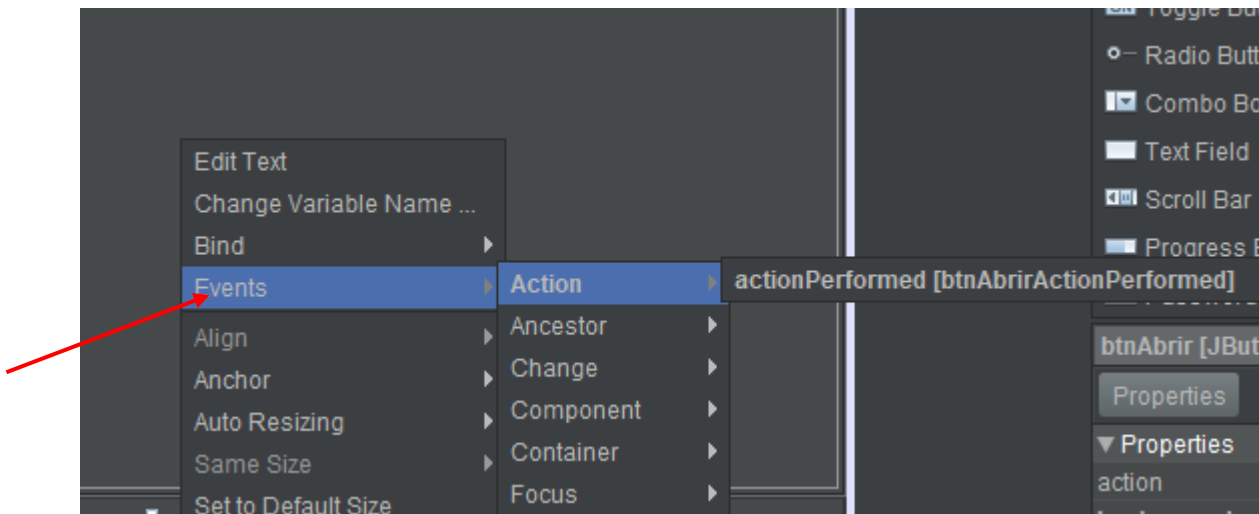
1.5.-Te tiene que quedar como en la imagen.



1.6.- Al ejecutar el programa, la interfaz permitirá personalizar el aspecto de los botones. Haciendo clic derecho sobre un botón y seleccionando "Propiedades", el usuario podrá acceder a la opción de "Background" para modificar el color del botón según su preferencia. Esta funcionalidad facilita la personalización visual, permitiendo al usuario ajustar los colores de los botones para mejorar la estética o la accesibilidad de la aplicación.



1.7.-Ahora lo siguiente que hacer es el funcionamiento del programa donde se podrá abrir el archivo tenemos que colocarnos donde esta nuestro botón de abrir y darle clic derecho y darle donde dice Events>Action>actionPerformend.



1.8.- colocaremos el código donde nos llevó el actionPerformed

```
JFileChooser fileChooser = new JFileChooser();

int resultado = fileChooser.showOpenDialog(this);

if (resultado == JFileChooser.APPROVE_OPTION) {
    File archivoSeleccionado = fileChooser.getSelectedFile();

    try (BufferedReader br = new BufferedReader(new FileReader(archivoSeleccionado))) {
        textArea.setText("");
        String linea;
        while ((linea = br.readLine()) != null) {
            textArea.append(linea + "\n");
        }
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this, "Error al abrir el archivo: " + e.getMessage(), "Error",
        JOptionPane.ERROR_MESSAGE);
    }
}
```



### 1.9.- explicación del código

Se comprueba si el usuario aprobó la selección de un archivo comparando el resultado con `JFileChooser.APPROVE_OPTION`. Si es así, el archivo elegido se obtiene mediante `getSelectedFile()` y se guarda en `archivoSeleccionado`. A continuación, se utiliza un `BufferedReader` en un bloque *try-with-resources* para leer el archivo línea por línea, asegurando que el `BufferedReader` se cierre automáticamente al finalizar, incluso si ocurre un error.

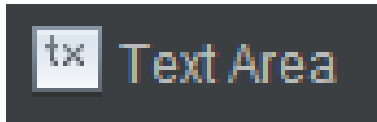
**1.10.-** Será lo mismo para el botón de salir le tendremos que seguir con `Events>Action>actionPerformed` el código de línea es sencillo.

```
private void btnSalirActionPerformed(java.awt.event.ActionEvent evt) {  
    System.exit(0);  
}
```

Esto tiene como funcionamiento como el número 0 indica que el programa finalizó sin errores

## Visualización

**2.1.-** Para poder ver el texto que se pueda ver en el programa ocuparemos un `JTextArea`



**2.2.-** para que el texto pueda leerse solo con una voz utilizaremos lo cual será un botón para poder reproducir ocuparemos un botón y lo que aremos es lo siguiente `Events>Action>ActionPerformed` y colocaremos el siguiente código.


```
private void btnReproducirActionPerformed(java.awt.event.ActionEvent evt) {  
    String texto = textArea.getText();  
    if (!texto.isEmpty()) {  
        reproducirTexto(texto);  
    } else {  
        JOptionPane.showMessageDialog(this, "No hay texto para reproducir.", "Advertencia",  
JOptionPane.WARNING_MESSAGE);  
    }  
}
```

**2.3.-** Este método se activa al hacer clic en el botón "Reproducir". Primero, obtiene el texto del área de texto (textArea). Si el texto no está vacío, llama al método reproducirTexto(texto) para reproducirlo. Si el área de texto está vacía, muestra un mensaje de advertencia diciendo "No hay texto para reproducir".

**2.4.-** Se agregará un método dentro de private void btnReproducirActionPerformed lo cual será el siguiente código.

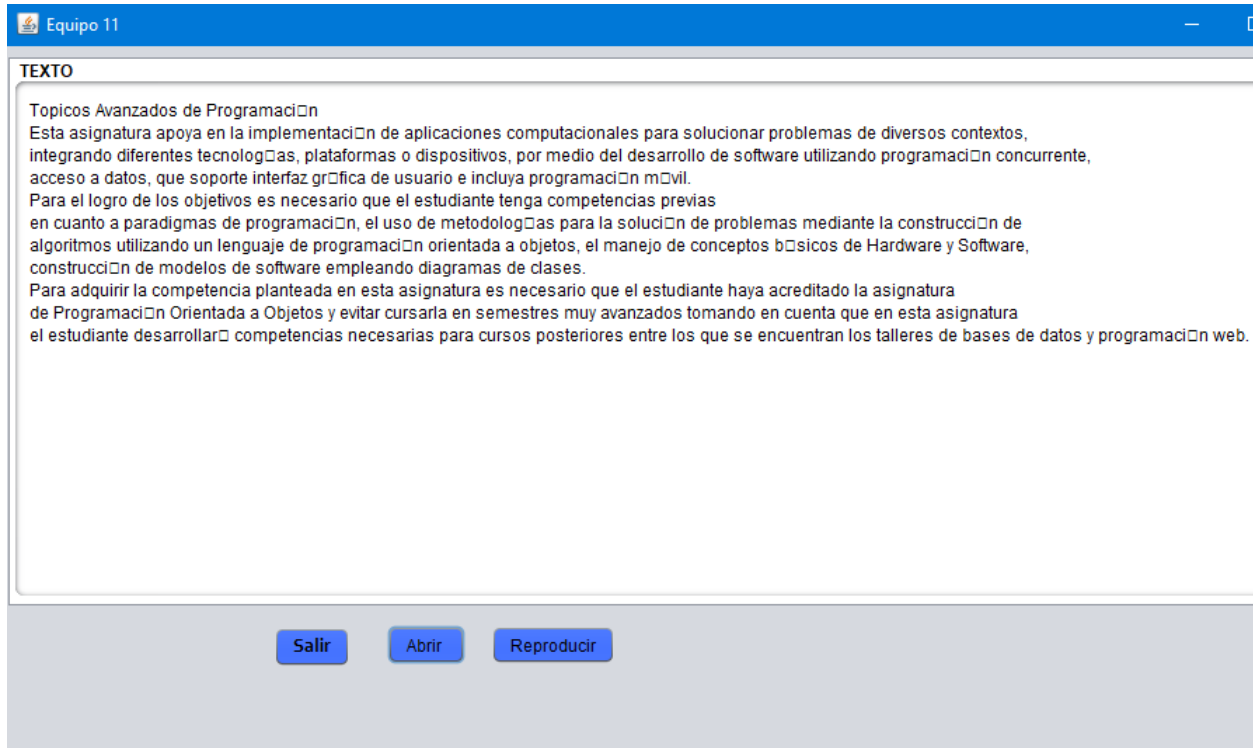
```
private void btnReproducirActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String texto = textArea.getText();  
  
    if (!texto.isEmpty()) {  
  
        reproducirTexto(texto);  
  
    } else {  
  
        JOptionPane.showMessageDialog(this, "No hay texto para reproducir.", "Advertencia", JOptionPane.WARNING_MESSAGE);  
  
    }  
}
```

```
private void reproducirTexto(String texto) {  
  
    Voice voice;  
    VoiceManager voiceManager = VoiceManager.getInstance();  
    voice = voiceManager.getVoice("kevin");  
  
    if (voice != null) {  
        voice.allocate();  
        voice.speak(texto);  
        voice.deallocate();  
    } else {  
        JOptionPane.showMessageDialog(this, "No se pudo encontrar la voz.", "Error", JOptionPane.ERROR_MESSAGE);  
    }  
}
```



**2.5.-** Este método reproducirTexto utiliza una voz sintética para leer el texto en voz alta. Primero, obtiene una instancia de VoiceManager y selecciona la voz llamada "kevin". Si la voz está disponible, la inicializa (allocate), lee el texto (speak), y luego la libera (deallocate). Si la voz no se encuentra, muestra un mensaje de error informando al usuario.

**2.6.-** así tendrá que lucir el programa al ejecutarlo



## **Conclusión**

La interfaz de usuario es un componente fundamental en el diseño de cualquier programa, ya que determina en gran medida la facilidad de uso y la accesibilidad para el usuario. Este programa ha sido diseñado con una interfaz sencilla e intuitiva, permitiendo al usuario cargar y escuchar archivos de texto sin complicaciones. Gracias a los botones "Abrir", "Reproducir" y "Salir", la interacción es clara y directa, mejorando la experiencia general de uso.

El uso de la síntesis de voz facilita aún más la accesibilidad, ya que permite que el texto sea escuchado en lugar de solo leído. En resumen, esta aplicación destaca cómo una interfaz bien diseñada puede simplificar la interacción con el software, haciendo que las funciones sean fáciles de usar y accesibles para cualquier usuario.