UNIVERSIDAD DE GUAYAQUIL

FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS

INGENIERÍA DE SOFTWARE

Micrófono - VoxBridge

BALLADARES LUIS

BAUZ GIOVANNI

JORDAN ANDERSON

MORALES STEWARD

TAFFUR KEVIN

VERA JORGE

ING. MARLON ADRIAN CALDERON GAVILANES

DESARROLLO DE APLICACIONES MÓVILES

SOF-S-NO-8-6

GUAYAQUIL, ENERO 11, 2024

Tabla de Contenido

VoxBridge

Objetivo

Brindar al usuario la capacidad de redactar texto en su dispositivo móvil haciendo uso de su voz como herramienta principal para expresar sus ideas.

Alcance

La aplicación por desarrollar (VoxBridge) les permitirá a las personas redactar texto en sus dispositivos móviles de forma sencilla mediante la voz, considerando la digitalización de todos los procesos en la actualidad, consideramos que esta aplicación puede ser una herramienta bastante útil para personas con discapacidades que les impidan redactar texto haciendo uso del teclado en pantalla, a su vez también, una forma cómoda de redactar textos. Se le ofrece al usuario la posibilidad de que el texto que ha redactado pueda ser traducido a múltiples idiomas, eso lo elige él mismo, de esa forma, puede redactar lo expresado por voz y hacerlo accesible para más personas sin importar el idioma. La versión de Android necesaria para ejecutar esta aplicación será Android Nougat, esta es la versión 7 del sistema operativo, por lo que versiones anteriores a esta no serán compatibles con la funcionalidad presentada.

Manual Técnico

1. Introducción

1.1 Objetivo

El objetivo principal de VoxBridge es proporcionar a los usuarios la capacidad de redactar texto en sus dispositivos móviles utilizando la voz como herramienta principal. La aplicación busca ser una solución eficiente y accesible para aquellos con dificultades para

escribir mediante el teclado en pantalla, al tiempo que ofrece la posibilidad de traducir el texto a varios idiomas.

1.2 Alcance

VoxBridge permite a los usuarios redactar texto de manera sencilla mediante el reconocimiento de voz en dispositivos Android con la versión 7 (Nougat) o superior. El alcance de la aplicación abarca tanto a personas con discapacidades que dificultan el uso del teclado, como a aquellos que buscan una forma cómoda de redactar textos.

2. Características Clave

2.1 Reconocimiento de Voz

VoxBridge integra un sistema de reconocimiento de voz que permite a los usuarios dictar texto en lugar de escribirlo manualmente. Esto mejora la accesibilidad para personas con discapacidades y proporciona una forma eficiente de redactar textos.

2.2 Traducción a Múltiples Idiomas

La aplicación ofrece la capacidad de traducir el texto redactado a varios idiomas. Los usuarios pueden elegir el idioma de destino, lo que hace que sus mensajes sean accesibles para una audiencia más amplia.

3. Requisitos del Sistema

3.1 Versión de Android

VoxBridge requiere Android Nougat (versión 7) o superior para ejecutarse. Las versiones anteriores a Android Nougat no serán compatibles con las funcionalidades de reconocimiento de voz y traducción presentadas.

4. Desarrollo de la Aplicación

4.1 Tecnologías Utilizadas

La aplicación está desarrollada en el entorno de Android, utilizando el lenguaje de programación Java. Se hace uso de las bibliotecas de Android para integrar funciones de reconocimiento de voz y traducción.

4.2 Estructura del Código

El código fuente de VoxBridge está organizado en clases y paquetes de acuerdo con las mejores prácticas de desarrollo de Android. Las principales funcionalidades, como el reconocimiento de voz, la traducción y la interfaz de usuario, están modularizadas para facilitar el mantenimiento y la expansión.

4.3 Integración de Servicios Externos

VoxBridge utiliza servicios externos para el reconocimiento de voz y la traducción. Se integra con las API proporcionadas por Android para el reconocimiento de voz y la API de Google Translate para la traducción de texto.

5. Consideraciones Finales

VoxBridge es una aplicación diseñada para mejorar la accesibilidad y la comodidad en la redacción de texto mediante la voz. Su enfoque en la traducción a múltiples idiomas amplía su utilidad para una variedad de usuarios. Este manual técnico proporciona una visión general del desarrollo y las características clave de la aplicación. Para obtener información detallada sobre la implementación, se recomienda revisar el código fuente y la documentación técnica asociada.

Código

AndroidManifest.xml

El principal componente de la aplicación es el micrófono, por lo que debemos asegurarnos de que se tienen permisos para acceder a este en el dispositivo, de igual forma, para poder interpretar y traducir el texto, se deberá hacer uso de internet, por lo que el permiso para este último también deberá ser expresado en el archivo AndroidManifest.xml, tal como se muestra en la ilustración.

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
```

En este mismo archivo debemos especificar las actividades que nuestra aplicación usará, tal como se muestra en la ilustración, en nuestro caso hacemos uso de dos actividades, una de ellas es el Splash Screen que será mostrada al iniciar la aplicación y la segunda que sería la funcionalidad de la aplicación como tal.

```
<activity
    android:name=".MainActivity"
    android:exported="false" />
<activity
    android:name=".SplashActivity"
    android:theme="@style/Theme.Design.NoActionBar"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

MainActivity.java

La clase MainActivity es la actividad principal de la aplicación. Esta aplicación utiliza reconocimiento de voz y traducción de texto para proporcionar una interfaz que permite a los usuarios hablar en un idioma específico, reconocer el habla y luego traducir el texto resultante a otro idioma de su elección.

La clase MainActivity extiende AppCompatActivity y contiene variables miembros para varios elementos de la interfaz de usuario, objetos de reconocimiento de voz (SpeechRecognizer), un traductor (Translator), y otras variables necesarias para la funcionalidad de la aplicación.

Atributos Principales:

imageButton: Botón para activar/desactivar el reconocimiento de voz.

editText: Campo de texto para la entrada de voz o texto.

targetlang: TextView para mostrar la traducción.

srclangchoose: Botón para seleccionar el idioma de origen.

trgtlangchoose: Botón para seleccionar el idioma de destino.

transbtn: Botón para iniciar la traducción.

count: Variable para gestionar el estado del reconocimiento de voz.

speechRecognizer: Objeto para el reconocimiento de voz.

translator: Objeto para la traducción de texto.

progressDialog: Indicador de progreso para la traducción.

languageArrayList: Lista de modelos de idiomas disponibles.

sourceLanguageCode, targetLanguageCode: Códigos de idioma de origen y destino.

TAG: Etiqueta para mensajes de registro.

Otros elementos como botones adicionales y vistas de texto.

```
ImageButton imageButton;
4 usages
EditText editText;
3 usages
TextView targetlang;
4 usages
Button srclangchoose;
4 usages
Button trgtlangchoose;
2 usages
Button transbtn;
3 usages
int count = 0;
4 usages
SpeechRecognizer speechRecognizer;
3 usages
Translator translator;
9 usages
ProgressDialog progressDialog;

no usages
private Button openInstructionsButton;

9 usages
private static final String TAG = "MAIN_TAG";

8 usages
private ArrayList<ModelLanguage> languageArrayList = null;

3 usages
private String sourceLanguageCode = "es";
3 usages
```

Métodos Principales:

onCreate(Bundle savedInstanceState): Método principal que se llama cuando la actividad es creada. Inicializa elementos de la interfaz de usuario, solicita permisos, y configura los controladores de eventos.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    imageButton = findViewById(R.id.button);
    editText = findViewById(R.id.editartexto);
    targetlang = findViewById(R.id.targetlang);
    srclangchoose = findViewById(R.id.srclangchoose);
    trgtlangchoose = findViewById(R.id.trgtlangchoose);
    transbtn = findViewById(R.id.transbtn);
    ImageButton copyButton = findViewById(R.id.copyButton);
Resto de tu lógica con el ImageButton
```

copyTextToClipboard(): Copia el texto traducido al portapapeles.

```
private void copyTextToClipboard() {
    String translatedText = targetlang.getText().toString().trim();

    if (!translatedText.isEmpty()) {
        ClipboardManager clipboard = (ClipboardManager) getSystemService(Context.CLIPBOARD_SERVICE);
        ClipData clip = ClipData.newPlainText( label: "Texto Traducido", translatedText);
        clipboard.setPrimaryClip(clip);

        showToast("Texto copiado al portapapeles");
    }
}
```

validateData(): Valida que haya datos de entrada antes de iniciar la traducción.

```
private void validateData() {
    String sourceLanguageText = editText.getText().toString().trim();
    Log.d(TAG, msg: "validateData: sourceLanguageText: " + sourceLanguageText);

    if (!sourceLanguageText.isEmpty()) {
        startTranslation();
    }
}
```

startTranslation(): Inicia el proceso de traducción utilizando la API de traducción de Google ML Kit.

```
private void startTranslation() {
    progressDialog = new ProgressDialog( context: this);
    progressDialog.setTitle("Por favor, espere");
    progressDialog.setCanceledOnTouchOutside(false);
    progressDialog.setMessage("Procesando el lenguaje ....");
    progressDialog.show();
```

loadAvailableLanguages(): Carga la lista de idiomas disponibles utilizando Google ML Kit.

```
private void loadAvailableLanguages() {
    languageArrayList = new ArrayList<>();
    String[] languageCodeList = TranslateLanguage.getAllLanguages().toArray(new String[0]);
```

sourceLanguageChoose(), targetLanguageChoose(): Muestran menús emergentes para seleccionar el idioma de origen y destino.

```
private void sourceLanguageChoose() {
    showLanguagePopupMenu(srclangchoose, new PopupMenu.OnMenuItemClickListener() {
        @Override
```

```
private void targetLanguageChoose() {
    showLanguagePopupMenu(trgtlangchoose, new PopupMenu.OnMenuItemClickListener() {
        @Override
```

showLanguagePopupMenu(View anchorView, PopupMenu.OnMenuItemClickListener listener): Muestra un menú emergente de idiomas.

```
private void showLanguagePopupMenu(View anchorView, PopupMenu.OnMenuItemClickListener listener) {
    PopupMenu popupMenu = new PopupMenu( context: this, anchorView);

    for (int i = 0; i < languageArrayList.size(); i++) {
        popupMenu.getMenu().add(Menu.NONE, i, i, languageArrayList.get(i).getLanguageTitle());
    }

    popupMenu.show();

    popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
        @Override
        public boolean onMenuItemClick(MenuItem menuItem) {
            int position = menuItem.getItemId();
            listener.onMenuItemClick(menuItem);
            return true;
```

Otros métodos relacionados con el reconocimiento de voz y la gestión de eventos.

ModelLanguage.java

La clase ModelLanguage es un modelo simple que representa un idioma. Contiene atributos para el código del idioma y su nombre o título. Esta clase es utilizada para almacenar información sobre los idiomas disponibles en la aplicación.

Atributos:

languageCode: Almacena el código del idioma (por ejemplo, "es" para español, "en" para inglés).

languageTitle: Almacena el nombre o título del idioma (por ejemplo, "español", "English").

```
3 usages
public class ModelLanguage {
    3 usages
    private String languageCode;
    3 usages
    private String languageTitle;
```

Constructores:

ModelLanguage(String languageCode, String languageTitle): Constructor que permite crear una instancia de ModelLanguage con valores iniciales para el código y el título del idioma.

```
1 usage
public ModelLanguage(String languageCode, String languageTitle) {
    this.languageCode = languageCode;
    this.languageTitle = languageTitle;
}
```

Métodos de Acceso:

getLanguageCode(): Devuelve el código del idioma.

setLanguageCode(String languageCode): Establece el código del idioma.

getLanguageTitle(): Devuelve el título del idioma.

setLanguageTitle(String languageTitle): Establece el título del idioma.

```
public String getLanguageCode() { return languageCode; }

no usages
public void setLanguageCode(String languageCode) { this.languageCode = languageCode; }

3 usages
public String getLanguageTitle() { return languageTitle; }

no usages
public void setLanguageTitle(String languageTitle) { this.languageTitle = languageTitle; }
```

SplashActivity.java

La clase SplashActivity es una actividad que se utiliza para mostrar una pantalla de presentación (splash screen) durante un breve período de tiempo antes de redirigir al usuario a la actividad principal (MainActivity). La pantalla de presentación es comúnmente utilizada para proporcionar una transición visual suave entre el lanzamiento de la aplicación y su interfaz principal.

Estructura de la Clase:

La clase SplashActivity extiende AppCompatActivity y contiene un método onCreate donde se configura la pantalla de presentación.

```
3 usages
public class SplashActivity extends AppCompatActivity {
```

Métodos Principales:

onCreate(Bundle savedInstanceState): Método principal que se llama cuando la actividad es creada. En este método, se establece el contenido de la actividad como el diseño de la pantalla de presentación (activity_splash.xml). Se utiliza un temporizador (Handler con postDelayed)

para esperar durante 3000 milisegundos (3 segundos) antes de iniciar la actividad principal (MainActivity) y finalizar la actividad de la pantalla de presentación.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_splash);
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            Intent intent = new Intent( packageContext: SplashActivity.this,MainActivity.class);
            startActivity(intent);
            finish();
        }
    }, delayMillis: 3000);
```

Layout

activity_main.xml

El diseño utiliza ConstraintLayout como contenedor principal, permitiendo la creación de interfaces de usuario más complejas mediante restricciones entre las diferentes vistas. A continuación, se describen las vistas clave presentes en el diseño:

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    tools:context=".MainActivity"
    tools:ignore="ExtraText">
```

1. ImageView – logoImageView: Un componente de imagen que muestra el logotipo de la aplicación en la parte superior de la pantalla.

```
<ImageView
    android:id="@+id/logoImageView"
    android:layout_width="250dp"
    android:layout_height="216dp"
    android:src="@drawable/logo"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/appNameTextView"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintVertical_bias="0.2"
    tools:ignore="MissingConstraints" />
```

2. TextView – appNameTextView: Un componente de texto que muestra el nombre de la aplicación ("VoxBridge") debajo del logotipo.

```
<TextView
    android:id="@+id/appNameTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="VoxBridge"
    android:textSize="24sp"
    app:layout_constraintTop_toBottomOf="@+id/logoImageView"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintVertical_bias="0.0"
    tools:ignore="MissingConstraints" />
```

3. EditText – editartexto: Un campo de entrada de texto que permite al usuario ingresar texto o ver el resultado de la traducción.

```
<EditText
    android:id="@+id/editartexto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Presione el ícono del micrófono para convertir a texto"
    android:textColor="#000000"
    android:textColorHint="#252525"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

4. ImageButton – button: Un botón de imagen que representa el micrófono y se utiliza para iniciar y detener el reconocimiento de voz.

```
<ImageButton
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="84dp"
    android:src="@drawable/baseline_mic_off_24"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editartexto"
    app:layout_constraintVertical_bias="0.18" />
```

5. LinearLayout – options: Un contenedor horizontal que alberga dos botones (srclangchoose y trgtlangchoose) y una flecha para seleccionar idiomas de origen y destino.

```
<LinearLayout
    android:id="@+id/options"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_above="@+id/transbtn"
    android:layout_marginTop="204dp"
    android:gravity="center"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editartexto">
```

6. Button – srclangchoose: Un botón para seleccionar el idioma de origen.

```
<Button
    android:id="@+id/srclangchoose"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="5dp"
    android:layout_weight="1"
    android:text="Español"
    app:cornerRadius="20dp" />
```

7. ImageView: Una imagen de flecha hacia adelante que indica la dirección de traducción entre idiomas.

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="6.5dp"
    android:src="@drawable/baseline_arrow_forward_ios_24"></ImageView>
```

8. Button – trgtlangchoose: Un botón para seleccionar el idioma de destino.

```
<Button
    android:id="@+id/trgtlangchoose"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Inglés"
    app:cornerRadius="20dp" />
```

9. Button – transbtn: Un botón que inicia el proceso de traducción del texto ingresado.

```
<Button
    android:id="@+id/transbtn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="24dp"
    android:text="Traducir"
    app:cornerRadius="20dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent" />
```

10. TextView – targetlang: Un componente de texto que muestra el resultado de la traducción.

```
<TextView
    android:id="@+id/targetlang"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text=""
    android:textSize="18sp"
    android:textColor="#000000"
    app:layout_constraintTop_toBottomOf="@+id/editartexto"
    tools:layout_editor_absoluteX="0dp" />
```

11. ImageButton – copyButton: Un botón de imagen que permite copiar el texto traducido al portapapeles.

```
<ImageButton
    android:id="@+id/copyButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/copiar"
    android:layout_marginTop="204dp"
    app:layout_constraintTop_toBottomOf="@+id/appNameTextView"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498" />
```

12. FrameLayout – instructionsContainer: Un contenedor que alberga las instrucciones de uso y un botón para cerrarlas.

```
<FrameLayout
    android:id="@+id/instructionsContainer"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginTop="16dp"
    android:layout_marginBottom="16dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0">
```

13. RelativeLayout: Un contenedor de diseño relativo dentro de instructionsContainer.

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#E0E0E0"
    android:padding="16dp">
```

14. TextView – instructionsTextView: Un componente de texto que contiene las instrucciones detalladas para el uso de la aplicación.

```
<TextView
    android:id="@+id/instructionsTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Instrucciones:\n\n1. Presiona el icono del micrófono para dictar al traductor o escribe
    android:textColor="#000000"
    android:textSize="16sp" />
```

15. Button – closeButton: Un botón para cerrar las instrucciones.

```
<Button
    android:id="@+id/closeButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/instructionsTextView"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="16dp"
    android:layout_marginBottom="16dp"
    android:text="Cerrar" />
```

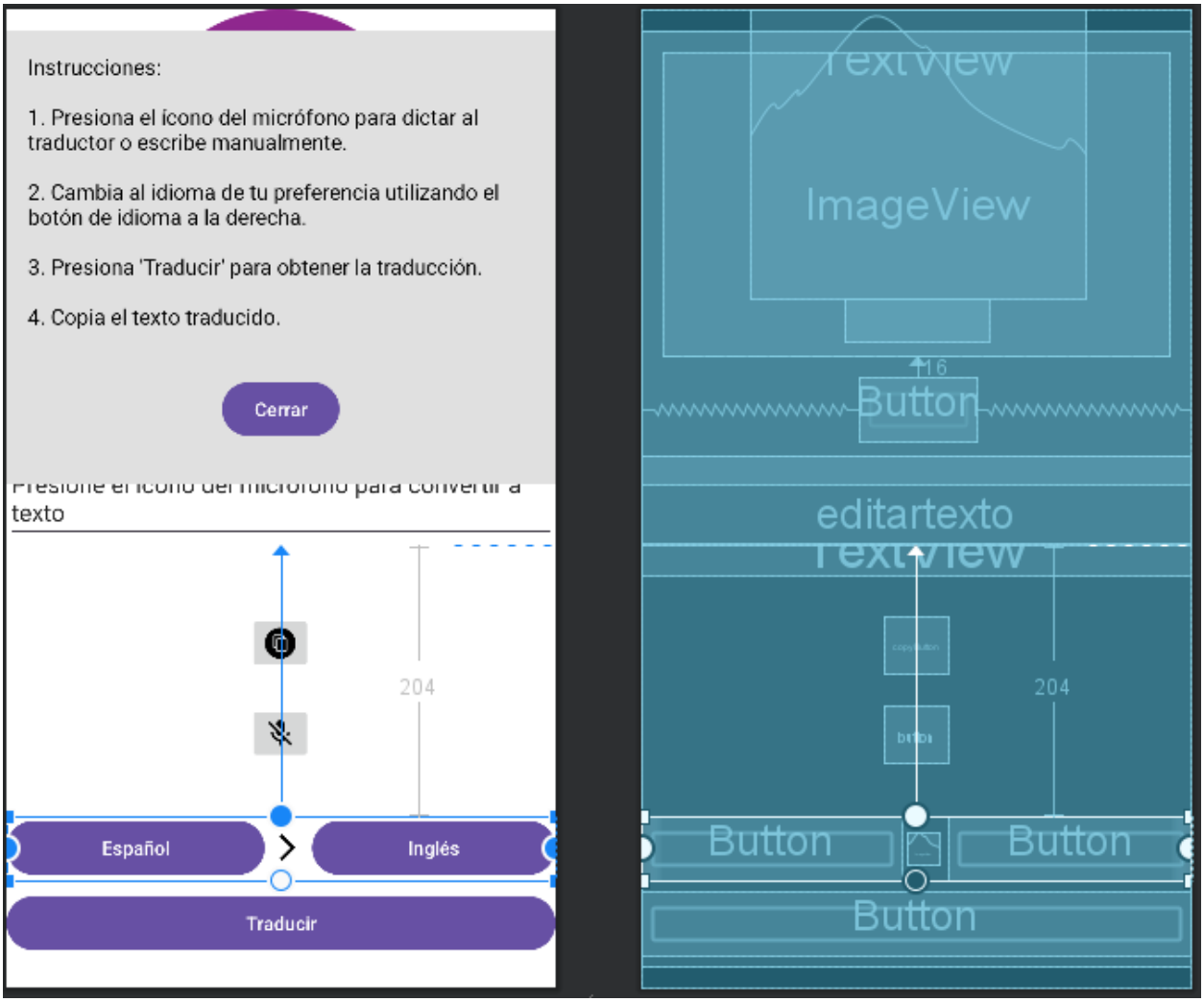
Notas Adicionales:

El atributo tools:ignore se usa para ignorar advertencias específicas de lint durante el desarrollo.

Se utilizan restricciones (app:layout_constraint*) para posicionar y alinear las vistas de manera dinámica en relación con otras vistas.

Se especifican márgenes y pesos en algunos elementos para garantizar un diseño visualmente equilibrado.

Diseño



activity_splash.xml

El diseño utiliza RelativeLayout como contenedor principal, permitiendo la superposición de vistas de manera relativa. A continuación, se describen las vistas clave presentes en el diseño:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    tools:context=".SplashActivity"
```

1. ImageView – imageView: Un componente de imagen que muestra el logotipo de la aplicación en el centro de la pantalla.

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="361dp"
    android:layout_height="326dp"
    android:layout_centerInParent="true"
    app:srcCompat="@drawable/logo" />
```

2. TextView – titleTextView: Un componente de texto que muestra el nombre de la aplicación ("VoxBridge") debajo del logotipo. Este texto se visualiza con un tamaño grande y en negrita para destacar.

```
<TextView
    android:id="@+id/titleTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/imageView"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"
android:text="VoxBridge"
android:textSize="38sp"
android:textColor="#000000"
android:textStyle="bold" />
```

Notas Adicionales:

Se utiliza el atributo layout_centerInParent="true" en la ImageView para centrarla vertical y horizontalmente en el RelativeLayout.

El atributo layout_below en el TextView asegura que este se posicione debajo de la ImageView.

Se especifica un margen en la parte superior del TextView (layout_marginTop) para ajustar la distancia entre el logotipo y el título.

Se aplican estilos de texto (textSize, textColor, textStyle) al TextView para mejorar la presentación visual.

Diseño: