Informe N.º2 Laboratorio Diseño y Análisis de Algoritmos

Estudiantes: Conzuelo Ardiles, Manuel Huenche, Erick Araya.
Profesor : Gonzalo Alexis Honores Vega.

17 de mayo de 2024

Índice

1. Introducción

En el marco de promover la inclusión educacional a través de la tecnología, se ha desarrollado una aplicación móvil destinada a facilitar la comunicación de personas con dificultades del habla, utilizando los principios de los Sistemas Aumentativos y Alternativos de Comunicación (SAAC). Esta aplicación ha sido creada e implementada en Android Studio, y está diseñada específicamente para ser utilizada en dispositivos con pantallas grandes, como la tablet Nexus 10.

El objetivo principal de esta aplicación es proporcionar una herramienta accesible y efectiva que permita a los usuarios construir frases mediante la selección de imágenes asociadas a palabras o conceptos. La interfaz de usuario ha sido diseñada para ser intuitiva y fácil de usar, contando con un panel de botones que, al ser presionados, muestran las imágenes correspondientes en un RecyclerView, facilitando así la creación visual de frases.

Este informe detalla el proceso completo de desarrollo de la aplicación, desde la fase inicial de diseño hasta su implementación final en un dispositivo Nexus 10. Se abordan aspectos clave como la configuración del entorno de desarrollo Android Studio, el diseño de la interfaz de usuario, la implementación de funcionalidades básicas y avanzadas, y las pruebas de accesibilidad y rendimiento.

A continuación, se describe el proceso de desarrollo y los componentes esenciales de la aplicación, destacando cómo cada elemento contribuye a la creación de una herramienta eficaz para la inclusión educativa.

2. Resumen de la aplicación

La aplicación de TeaCommunication, desarrollada utilizando Android Studio 4.2 y probada en un dispositivo virtual Nexus 10, se basa de un juego que consiste en adivinar una frase oculta mediante la selección de pictogramas que representan palabras. Cuando el usuario selecciona los pictogramas correctos, la frase se revela en la pantalla, ayudando a los usuarios a aprender y recordar frases útiles a través de la asociación de imágenes y palabras, esto frece una solución integral para aumentar la comunicacion efectiva para personas con dificultades comunicativas. Al aprovechar las herramientas y características avanzadas la aplicación garantiza una experiencia de usuario fluida y satisfactoria usando pictogramas, pantallas y botones intuitivos y faciles de usar.

A continuación, se detallan los componentes principales y los objetos clave del código que respaldan su funcionalidad:

Main Activity: FullscreenActivity es una actividad de pantalla completa en una aplicación Android que gestiona la visibilidad de la interfaz del sistema y los controles de la aplicación. Utiliza un RecyclerView para mostrar una lista horizontal de botones de imagen y permite al usuario interactuar con la pantalla para mostrar u ocultar la interfaz del sistema y los controles de la aplicación.

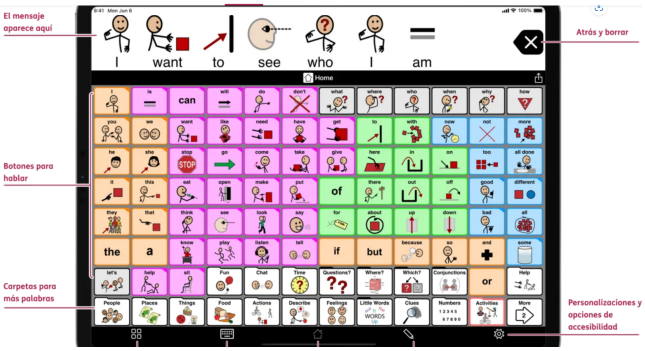


Figura 1: Base de Inspiracion Protoquo2

Los botones de imagen están configurados con listeners para responder a las interacciones del usuario.

ImageButtonAdapter es una clase en una aplicación Android que extiende RecyclerView.Adapter para manejar la visualización de una lista de botones de imagen en un RecyclerView. El constructor de la clase recibe un conjunto de recursos de imagen y un listener de clics como parámetros. Los métodos onCreateViewHolder() y onBindViewHolder() se encargan de inflar el diseño del elemento y configurar los botones de imagen con las imágenes correspondientes y el listener de clics. Además, la clase proporciona un método addItem() para agregar elementos a la lista y notificar al adaptador sobre los cambios en los datos.

3. Introducción del código

3.1. Código Principal

La clase FullscreenActivity contiene: Interacción Visual y Dinámica: Utiliza un RecyclerView en modo horizontal para mostrar botones de imagen (ImageButtons) que representan diferentes acciones o elementos. Esto permite a los usuarios seleccionar pictogramas de manera intuitiva y visual.

Secuencia de Frases: Contiene arrays (frasesSep y frases) que almacenan frases desglosadas en palabras junto con sus identificadores correspondientes. Los OnClickListeners de los botones de imagen verifican si la selección del usuario sigue la secuencia correcta de la frase, añadiendo la imagen al RecyclerView si es correcta.

Gestión de la UI: Utiliza Handlers y Runnables para ocultar y mostrar la barra de estado y la barra de navegación automáticamente después de un retraso, proporcionando una experiencia de pantalla completa sin distracciones.

Feedback al Usuario: Muestra mensajes Toast con la frase actual para proporcionar retroalimentación inmediata al usuario sobre su progreso en la construcción

de la frase.

Figura 2: Interfaz

```
public class FullscreenActivity extends AppCompatActivity {
    private RecyclerView recyclerView;
    private int[] imageResources = {
            R.drawable.acostar,
            R.drawable.cama,
            R.drawable.comer,
            R.drawable.correr,
            R.drawable.en,
            R.drawable.la,
            R.drawable.manzana,
            R.drawable.nina,
            R.drawable.nino,
            R.drawable.parque,
            R.drawable.se,
            R.drawable.uno,
            R.drawable._con,
            R.drawable.saltarcuerda,
            R.drawable.jugar,
            R.drawable.pelota
    };
    // primer valor, frase
    // segundo valor, palabra
    // tercer valor, identificador de pictograma
    String[][][] frasesSep = {
            {
                    {"El", "11"}, {"niño", "9"}, {"corre", "4"}, {"en", "5"}, {"parque", "10"}
            },
```

```
binding.dummyButton.setOnTouchListener(mDelayHideTouchListener);
ImageButton imageButtonAcostar = findViewById(R.id.imageButton_acostar);
ImageButton imageButtonCama = findViewById(R.id.imageButton_cama);
ImageButton imageButtonComer = findViewById(R.id.imageButton_comer);
ImageButton imageButtonCorrer = findViewById(R.id.imageButton_correr);
ImageButton imageButtonEn = findViewById(R.id.imageButton_en);
ImageButton imageButtonLa = findViewById(R.id.imageButton_la);
ImageButton imageButtonManzana = findViewById(R.id.imageButton_manzana);
ImageButton imageButtonNina = findViewById(R.id.imageButton_nina);
ImageButton imageButtonNino = findViewById(R.id.imageButton_nino);
ImageButton imageButtonParque = findViewById(R.id.imageButton_parque);
ImageButton imageButtonSe = findViewById(R.id.imageButton_se);
ImageButton imageButtonUno = findViewById(R.id.imageButton_uno);
ImageButton imageButtonCon = findViewById(R.id.imageButton_con);
ImageButton imageButtonSaltarCuerda = findViewById(R.id.imageButton_saltarcuerda);
ImageButton imageButtonJugar = findViewById(R.id.imageButton_jugar);
ImageButton imageButtonPelota = findViewById(R.id.imageButton_pelota);

// Asignación de listeners a cada ImageButton
imageButtonAcostar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Acción para acostar
        comp( pict: 0,frasesSep[Select]);
    }
});
```

Figura 3: Codigo Pantalla Principal

El ImageButtonAdapter es un adaptador para un RecyclerView en una aplicación Android, que maneja una lista de ImageButton. Las partes más importantes incluyen:

imageResources: Array de recursos de imagen. clickListener: Listener para manejar clics en los botones. Constructor:

Inicializa el adaptador con los recursos de imagen y el listener de clic. Métodos Clave:

onCreateViewHolder: Infla el diseño del elemento y crea un ImageButtonViewHolder.

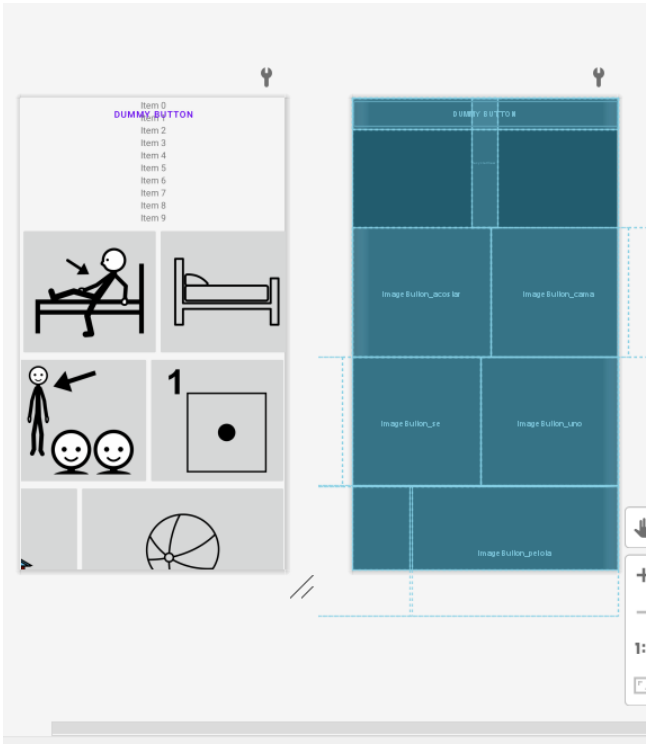


Figura 4:

onBindViewHolder: Asigna la imagen y el listener de clic al ImageButton en la posición dada.

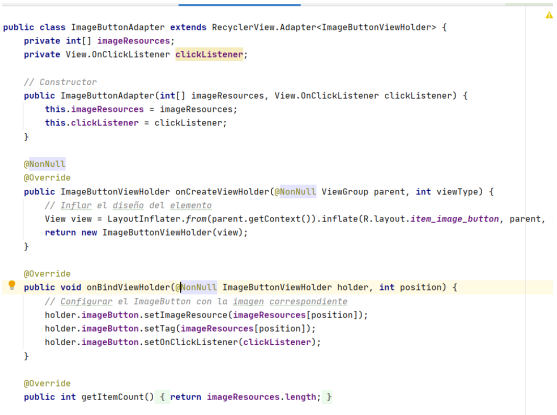
getItemCount: Devuelve la cantidad de elementos en el adaptador. addItem: Añade dinámicamente un nuevo recurso de imagen en una posición específica y notifica el cambio al RecyclerView.

```
public void addItem(int imageResource, int position) {
    if (position >= 0 && position <= imageResources.length) {
        int[] newImageResources = new int[imageResources.length + 1];
        System.arraycopy(imageResources, srcPos: 0, newImageResources, destPos: 0, position);
        newImageResources[position] = imageResource;
        System.arraycopy(imageResources, position, newImageResources, destPos: position + 1, length: imageResources.
        imageResources = newImageResources;
        notifyItemInserted(position);
    }
}
}
```

Figura 5:

La clase ImageButtonAdapter para Android gestiona una lista de botones de imagen en un RecyclerView. Incluye un array de recursos de imagen (image-Resources) y un listener de clics (clickListener). Su constructor inicializa estos atributos. Los métodos principales son onCreateViewHolder, que infla el diseño del botón desde un archivo XML; onBindViewHolder, que asigna la imagen y

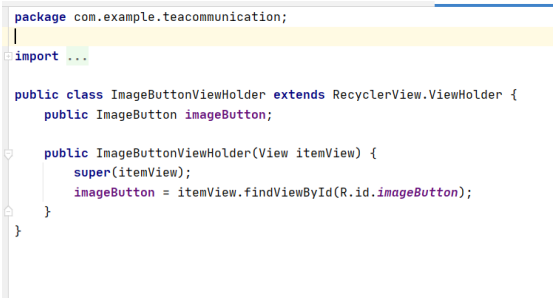
el listener de clics a cada botón según su posición; y getItemCount, que devuelve la cantidad de elementos. Además, el método addItem permite agregar una nueva imagen en una posición específica, actualizando el array de imágenes y notificando al adaptador sobre la inserción.



```java
public class ImageButtonAdapter extends RecyclerView.Adapter<ImageButtonViewHolder> {
    private int[] imageResources;
    private View.OnClickListener clickListener;

    // Constructor
    public ImageButtonAdapter(int[] imageResources, View.OnClickListener clickListener) {
        this.imageResources = imageResources;
        this.clickListener = clickListener;
    }

    @NonNull
    @Override
    public ImageButtonViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        // Inflar el diseño del elemento
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_image_button, parent,
        return new ImageButtonViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ImageButtonViewHolder holder, int position) {
        // Configurar el ImageButton con la imagen correspondiente
        holder.imageButton.setImageResource(imageResources[position]);
        holder.imageButton.setTag(imageResources[position]);
        holder.imageButton.setOnClickListener(clickListener);
    }

    @Override
    public int getItemCount() { return imageResources.length; }
```

Figura 6: ImageButtonAdapter

El código en la clase ImageButtonViewHolder de Android define un ViewHolder personalizado para un RecyclerView. Este ViewHolder contiene un solo ImageButton que se configura para mostrar una imagen en una lista de elementos. En el constructor, se usa itemView.findViewById(R.id.imageButton) para encontrar y asignar el ImageButton correspondiente al diseño del elemento de la vista. Esto permite que el adaptador (ImageButtonAdapter) utilice instancias de ImageButtonViewHolder para manejar y mostrar imágenes en un RecyclerView.



```java
package com.example.teacommunication;

import ...

public class ImageButtonViewHolder extends RecyclerView.ViewHolder {
    public ImageButton imageButton;

    public ImageButtonViewHolder(View itemView) {
        super(itemView);
        imageButton = itemView.findViewById(R.id.imageButton);
    }
}
```

Figura 7: ImageButtonViewHolder

3.2. Interfaz del usuario

Finalmente aquí tenemos las imágenes y códigos de la interfaz de usuario, la cual es la presentacion final del proyecto, usando XML y Android para crear las páginas y paneles, junto a sus respectivos códigos.

```
<LinearLayout
    android:id="@+id/linearLayout"
    android:layout_width="1279dp"
    android:layout_height="200dp"
    android:orientation="horizontal"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/recyclerView">

    <ImageButton
        android:id="@+id/imageButton_acostar"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:scaleType="fitCenter"
        app:srcCompat="@drawable/acostar" />

    <ImageButton
        android:id="@+id/imageButton_cama"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:scaleType="fitCenter"
        app:srcCompat="@drawable/cama" />
```

Utiliza un ConstraintLayout como el contenedor principal e incluye los siguientes elementos clave:

RecyclerView: Un RecyclerView que se extiende horizontalmente y tiene una altura fija de 200dp. Está centrado horizontalmente en la parte superior del diseño.

LinearLayouts: Tres LinearLayout dispuestos verticalmente uno debajo del otro, cada uno con una serie de ImageButton alineados horizontalmente. Los LinearLayout tienen una altura fija de 200dp y están configurados para alinearse con los elementos superiores del diseño.

ImageButton: Cada LinearLayout contiene varios ImageButton, cada uno con imágenes específicas asignadas mediante el atributo app:srcCompat. Estos botones tienen una disposición de escala "fitCenterz utilizan el atributo layout weight para distribuir el espacio equitativamente dentro del LinearLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:padding="2dp"
    android:background="@color/white">

    <ImageButton
        android:id="@+id/imageButton"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:background="?android:selectableItemBackgroundBorderless"
        android:scaleType="fitCenter" />
</RelativeLayout>
```

4. Conclusión

En esta sección, se presentan las conclusiones finales obtenidas tras el desarrollo y análisis de la aplicación para dispositivos Android.

4.1. Conclusión del Informe

El desarrollo de la aplicación de comunicación aumentativa y alternativa para dispositivos Android ha sido un proceso enriquecedor que nos ha permitido comprender en profundidad los principios fundamentales del diseño y desarrollo de aplicaciones móviles orientadas a la inclusión educacional. A lo largo de este proyecto, hemos explorado diversas herramientas y características proporcionadas por Android Studio, así como los principios de diseño de interfaces de usuario accesibles y la implementación de algoritmos interactivos para la creación de frases mediante pictogramas.

Al finalizar el desarrollo, hemos obtenido una aplicación funcional que cumple con los objetivos establecidos inicialmente. La aplicación ofrece una interfaz intuitiva que permite a los usuarios crear y gestionar frases mediante la selección de pictogramas, facilitando la comunicación para personas con dificultades del habla. La implementación del juego de adivinar palabras mediante pictogramas no solo hace el proceso de aprendizaje más interactivo y divertido, sino que también refuerza la asociación entre imágenes y palabras, promoviendo un aprendizaje más efectivo.

En conclusión, el desarrollo de esta aplicación nos ha permitido adquirir habilidades valiosas en el campo del desarrollo de aplicaciones móviles y una comprensión más profunda de los principios y prácticas de diseño inclusivo y análisis de algoritmos. Este proyecto sienta las bases para futuras investigaciones y desarrollos en el ámbito de la informática móvil aplicada a la inclusión educacional, contribuyendo así al avance continuo de la tecnología y la innovación en beneficio de personas con discapacidades.