Taller Diseño Android

Carolina Martinez Cortes

SENA – Servicio Nacional de Aprendizaje

Centro de la Industria la Empresa y los Servicios

Análisis y Desarrollo de Software

Ficha: 2694667

Carlos Julio Cadena

27 de mayo del 2024

Neiva / Huila

➢ LinearLayout(vertical y horizontal)

El LinearLayout es un tipo de ViewGroup en Android que se organiza sus elementos secundarios en una sola dirección ya sea vertical o horizontal. Cada elemento secundario en una LinearLayout se coloca uno tras otro, en fila (horizontal) o en columna(vertical).

➢ *Características principales:*

1. Orientación: se puede establecer la orientación en vertical (android:orientaion=”vertical”) o horizontal (android:orientation=”horizontal”).

2. Peso(Weight):permite distribuir el espacio disponible ente los elementos secundarios mediante el atributo ´android:layout_weight´.

3.Distribucion Espacial: los elementos pueden alinearse y distribuirse en función de los valores especificados en sus atributos como ´layout_gravity´ y ´gravity´.

➢ *Como se organizan los objetos dentro del contenedor:*

1. Vertical: Los elementos se colocan uno debajo del otro.

2. Horizontal: Los elementos se colocan uno al lado del otro.

- *Ventajas:*

1. Fácil de usar y entender para disposiciones simples.

2. Con el uso del atributo ´layout_weight´, se puede controlar como se distribuye el espacio ente los elementos.

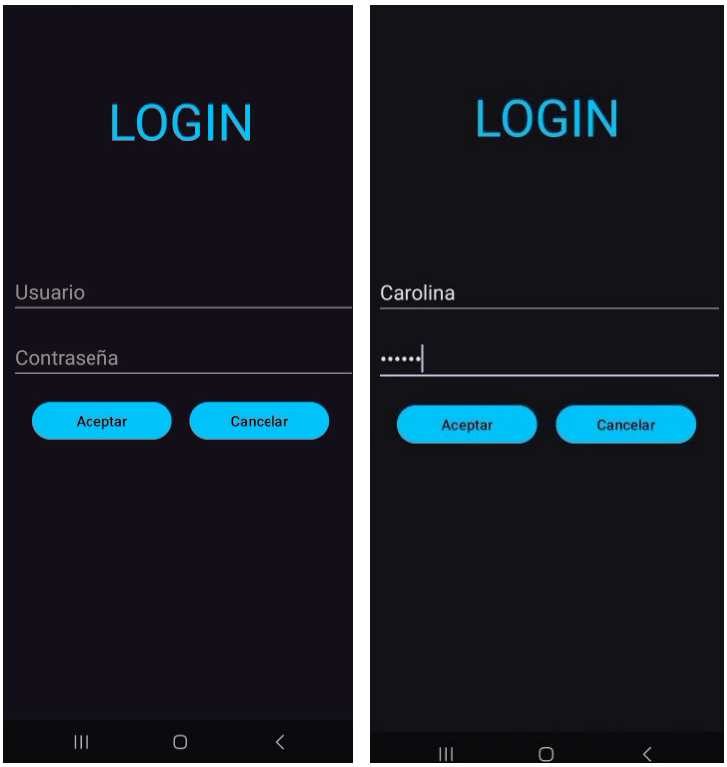
- *Desventajas:*

1. Puede ser menos eficiente en términos de rendimiento si se usa para interfaces muy complejas, ya que anidar múltiples LinearLayouts puede incrementar el tiempo de renderizado.

2. Menos flexible comparado con RelativeLayout o ConstraintLayout, que permiten disposiciones más complejas y adaptables.

3. Puede no ser ideal para aprovechar todo el espacio disponible en dispositivos con diferentes tamaños de pantalla.

- *Ejemplo de interfaz:*



➢ RelativeLayout.

El RelativeLayout es un tipo de ViewGroup en Android que permite posicionar sus elementos secundarios en relación unos con otros (por ejemplo, a lado de, debajo de) o en relación con el contenedor padre (por ejemplo, alineado a la izquierda, derecha, arriba, abajo).

➢ *Características principales:*

1. Posicionamiento Relativo: Los elementos pueden ser posicionados relativos a otros elementos dentro del mismo contenedor mediante atributos como *´layout_toRightOf´, ´ layout_below´, ´layout_alignParentTop´*.

2. Flexibilidad: Ofrece gran flexibilidad para crear disposiciones complejas sin necesidad de anidar múltiples layouts.

3. Alineación: Los elementos pueden alinearse a los bordes del contenedor padre o a otros elementos dentro del contenedor.

➢ *Como se organizan los objetos dentro del contenedor:*

1. Relaciones entre Elementos: Los elementos secundarios se posicionan en relación con otros elementos utilizando atributos específicos. Por ejemplo, un elemento puede estar a la derecha de otro (layout_toRightOf), o debajo de otro (layout_below).

2. Alineación con el Padre: Los elementos también pueden alinearse con los bordes del contenedor padre utilizando atributos como layout_alignParentTop, layout_centerHorizontal, etc.

- *Ventajas:*

1. Permite crear interfaces complejas y bien organizadas sin necesidad de anidar varios layouts.

2. Puede reducir la cantidad de anidamiento de layouts, mejorando el rendimiento en comparación con el uso excesivo de LinearLayouts anidados.

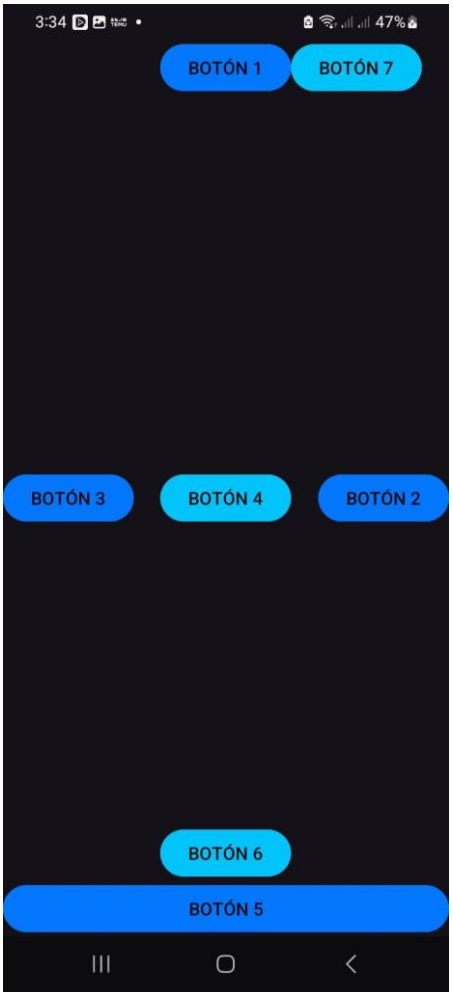
3. Facilita la creación de diseños adaptables a diferentes tamaños de pantalla y orientaciones.

- *Desventajas:*

1. Puede volverse complejo y difícil de mantener cuando se tienen muchas vistas con relaciones complicadas.

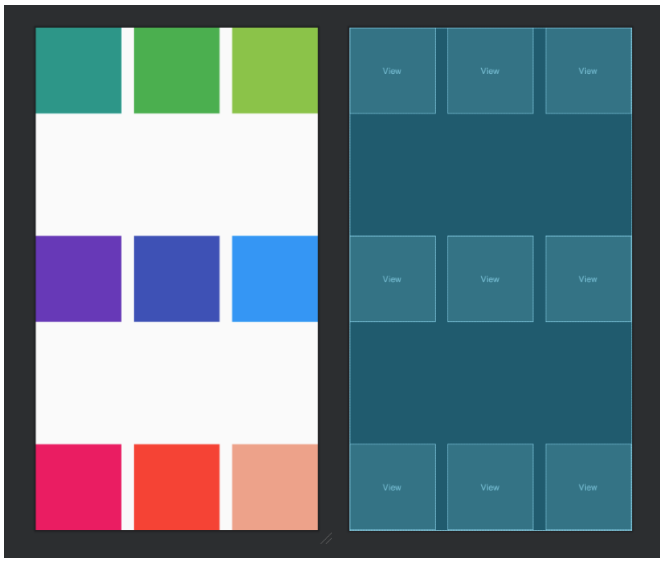
2. Aunque mejor que múltiples LinearLayouts anidados, puede ser menos eficiente que ConstraintLayout para diseños muy complejos debido a la evaluación múltiple de relaciones durante el layout.

- *Ejemplo de interfaz:*



- **FrameLayout.**

  FrameLayout es un tipo de ViewGroup en Android que se usa para conectar una sola vista o un pequeño numero de vistas secundarias. Las vistas en un FrameLayout se apilan una sobre otra, permitiendo superposiciones de elementos. Es útil para escenarios donde se desea mostrar una vista sobre otra, como una imagen con un texto encima.



- *Características principales:*

1. Apilamiento de Vistas: Los elementos secundarios se apilan uno sobre otro, con la primera vista agregada al FrameLayout en el fondo y las vistas subsecuentes apiladas encima.

2. Uso Simplicidad: Se utiliza generalmente para disposiciones simples y cuando se necesita superponer vistas.

3. Control Individual: Cada vista dentro de un FrameLayout se maneja independientemente sin afectar la posición de las otras vistas, aunque están apiladas.

- *Como se organizan los objetos dentro del contenedor:*

1. Superposición: Las vistas se organizan en capas, una sobre otra. La vista agregada primero está en la capa más baja y las vistas agregadas posteriormente se superponen encima.

2. Posicionamiento: Las vistas dentro de un FrameLayout se posicionan en la esquina superior izquierda por defecto, aunque pueden ser reposicionadas usando márgenes o mediante otras técnicas de layout.

- *Ventajas:*

    1. Muy sencillo de usar para diseños donde se necesita una simple superposición de elementos.

    2. Ideal para casos donde se necesita superponer vistas, como botones sobre una imagen.

    3. Generalmente más eficiente que otros layouts para simples apilamientos debido a su menor sobrecarga de gestión de vistas.

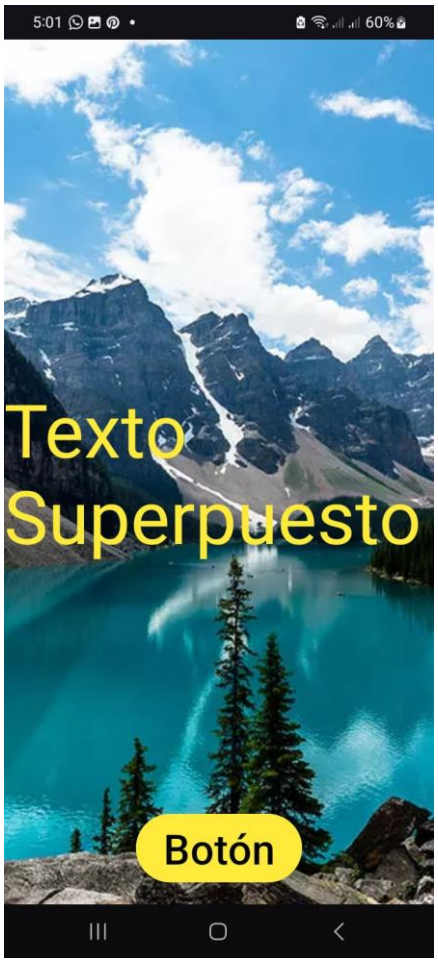
- *Desventajas:*

    1. No es adecuado para diseños complejos ya que no ofrece muchas opciones para posicionar vistas relativas entre sí.

    2. Las vistas se apilan sin reglas de organización específicas, lo que puede complicar su uso en disposiciones complejas.

    3. Puede ser menos flexible en términos de redimensionamiento y posicionamiento dinámico de vistas.

- *Ejemplo de interfaz:*



➢ ConstraintLayout.

ConstraintLayout es un tipo de ViewGroup en Android diseñado para crear layouts complejos sin anidar múltiples Layouts. Ofrece un sistema de restricción flexible que permite posicionar y dimensionar vistas en relación con otras vistas y con el contenedor principal.

- *Características principales:*

1. Restricciones (Constraints): Las vistas pueden ser posicionadas y dimensionadas utilizando restricciones relativas a otras vistas, al contenedor padre, o a guías y barreras.

2. Guiado y Barreras: Soporta elementos adicionales como guías (Guidelines) y barreras (Barriers) para ayudar en la organización y alineación de las vistas.

3. Encadenamiento (Chains): Permite agrupar vistas en una cadena (horizontal o vertical) para distribuir el espacio de manera equitativa o con proporciones específicas.

4. Dimensionamiento: Soporta dimensionamiento flexible como "wrap_content", "match_parent", y "0dp" (dimensionamiento basado en restricciones).

- *Como se organizan los objetos dentro del contenedor:*

1. Posicionamiento Relativo: Las vistas pueden ser posicionadas en relación con otras vistas usando atributos como ´layout_constraintStart_toStartOf´, ´layout_constraintEnd_toEndOf´, ´layout_constraintTop_toBottomOf´.

2. Guías y Barreras: Guías son líneas invisibles que se pueden usar como puntos de referencia para posicionar vistas. Barreras son límites dinámicos basados en el contenido de otras vistas.

3. Encadenamiento: Las vistas se pueden encadenar para crear disposiciones más complejas y equilibradas.

- *Ventajas:*

1. Ofrece un control detallado sobre el posicionamiento y dimensionamiento de vistas, permitiendo diseños complejos sin anidar layouts.

2. Minimiza la profundidad de la jerarquía de vistas, mejorando el rendimiento.

3. Facilita la creación de layouts responsivos que se adaptan bien a diferentes tamaños de pantalla y orientaciones.

4. Integración con el Editor de Diseño en Android Studio, que facilita la creación visual de layouts.

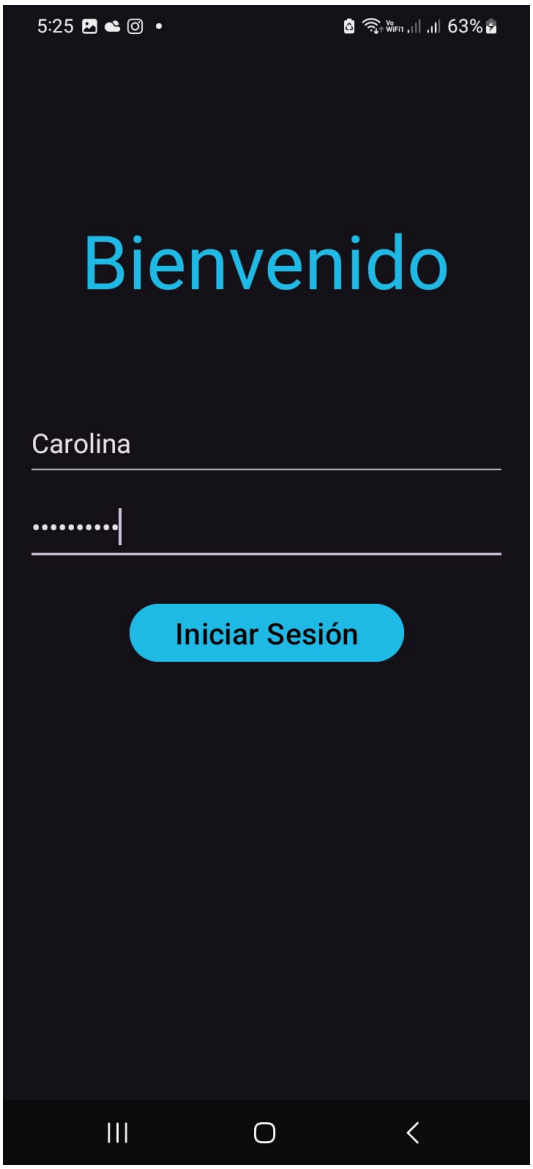
- *Desventajas:*

1. Puede ser complejo y tener una curva de aprendizaje más pronunciada comparado con otros layouts como LinearLayout o RelativeLayout.

2. Puede resultar en un código XML más extenso y detallado.

3. Para layouts extremadamente simples, puede ser excesivo y menos eficiente en comparación con otros layouts más sencillos.

- *Ejemplo de interfaz:*



➢ CardView

CardView es un contenedor especializado en Android que proporciona una envoltura con esquinas redondas y sombra, que hace que el contenido aparezca sobre la interfaz de usuario, Es una extensión de ´FrameLayout´ y se utiliza para crear tarjetas de estilo Material Design.

➢ *Características principales:*

1. Diseño de Material Design: CardView sigue las directrices de Material Design, proporcionando un aspecto moderno y visualmente atractivo.

2. Esquinas Redondeadas: Permite configurar el radio de las esquinas mediante el atributo cardCornerRadius.

3. Elevación y Sombra: Proporciona elevación y sombra para crear un efecto de profundidad mediante el atributo cardElevation.

4. Margen Interno: Se puede definir un margen interno utilizando el atributo contentPadding.

5. Compatibilidad: Compatible con versiones anteriores de Android mediante la biblioteca de soporte.

➢ *Como se organizan los objetos dentro del contenedor:*

1. Comportamiento de FrameLayout: Al ser una subclase de FrameLayout, puede contener múltiples vistas, que se apilan una sobre otra. Sin embargo, típicamente se usa para envolver un único grupo de elementos (por ejemplo, un layout que contiene texto e imágenes).

2. Diseño Interno: Dentro de una CardView, se pueden organizar los elementos utilizando cualquier otro tipo de layout (por ejemplo, LinearLayout, RelativeLayout, ConstraintLayout) para estructurar el contenido.

➢ *Ventajas:*

1. Proporciona un diseño limpio y moderno con sombras y esquinas redondeadas.

2. Ofrece varias opciones de personalización como la elevación, el radio de las esquinas y el margen interno.
3. Compatible con versiones anteriores de Android, asegurando una apariencia consistente en diferentes dispositivos.

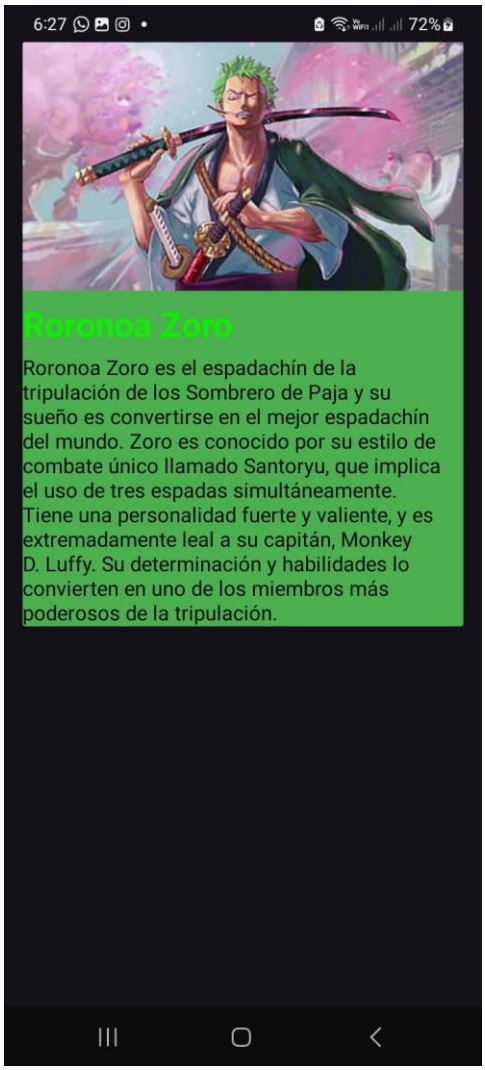
4. Facilita la creación de tarjetas visualmente atractivas sin necesidad de una configuración compleja.

- *Desventajas:*

1. El uso de sombras y elevación puede afectar el rendimiento en dispositivos más antiguos o menos potentes.

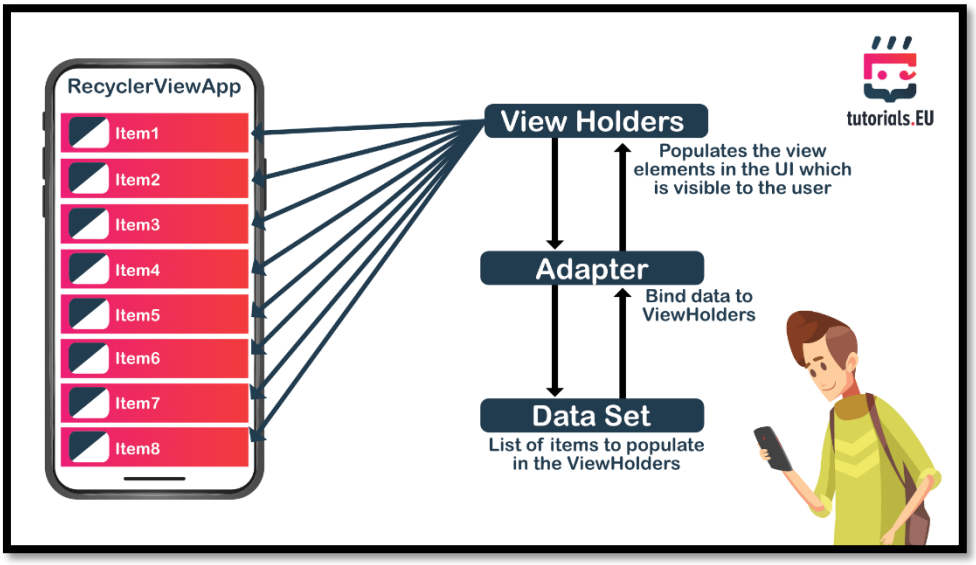
2. Menos flexible en comparación con otros ViewGroups más genéricos, ya que está diseñado específicamente para cumplir con las directrices de Material Design.

- *Ejemplo de interfaz:*



➢ RecyclerView

RecyclerView es un widget avanzado en Android para la presentación de listas grandes o conjuntos de datos que pueden desplazarse de manera eficiente. Es una mejora de ListView y GridView, diseñada para manejar y mostrar cantidades de datos de manera más eficiente.



➢ *Características principales:*

1. ViewHolder Pattern: Utiliza el patrón ViewHolder para mejorar el rendimiento al reciclar las vistas de los elementos que salen de la pantalla.

2. LayoutManager: Soporta diferentes tipos de LayoutManagers (LinearLayoutManager, GridLayoutManager, StaggeredGridLayoutManager) para controlar el diseño de los elementos.

3. Item Decoration: Permite agregar decoraciones a los elementos, como divisores de línea, mediante ItemDecoration.

4. Item Animator: Facilita la animación de las vistas de los elementos cuando cambian de estado o posición, utilizando ItemAnimator.

5. Flexibilidad y Personalización: Ofrece una alta flexibilidad y personalización en comparación con ListView y GridView.

- *Como se organizan los objetos dentro del contenedor:*

1. Adapter: El adaptador (RecyclerView.Adapter) es responsable de crear y enlazar las vistas de los elementos con los datos.

2. ViewHolder: RecyclerView.ViewHolder es una clase que se utiliza para almacenar las vistas de los elementos. Mejora el rendimiento al reciclar las vistas en lugar de inflarlas de nuevo.

3. LayoutManager: Controla cómo se disponen los elementos dentro del RecyclerView. Ejemplos incluyen LinearLayoutManager para listas verticales/horizontales y GridLayoutManager para cuadrículas.

4. Data Binding: Cada vista de elemento se enlaza con los datos en el método onBindViewHolder del adaptador

- *Ventajas:*

1. Mejora el rendimiento mediante el reciclaje de vistas y el uso del patrón ViewHolder.

2. Soporta diferentes diseños y decoraciones, proporcionando más control sobre la presentación de los datos.

3. Facilita la implementación de animaciones para cambios de estado o posición de los elementos.

4. Muy extensible y personalizable para adaptarse a diferentes necesidades de diseño y comportamiento.

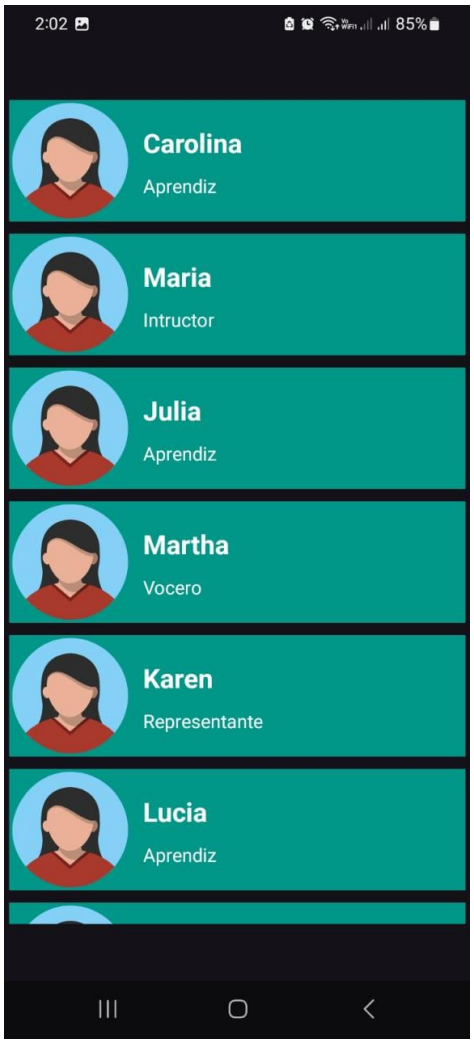
- *Desventajas:*

1. Puede ser más complejo de implementar en comparación con ListView debido a la necesidad de definir un adaptador y un ViewHolder.

2. Tiene una curva de aprendizaje más pronunciada para desarrolladores nuevos en comparación con ListView o GridView.

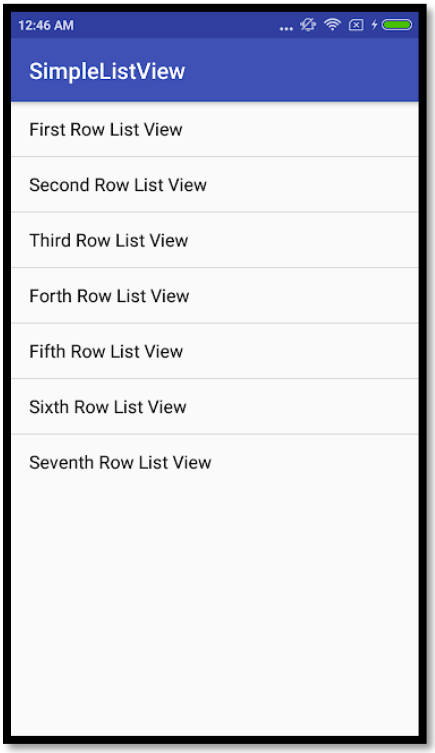
3. Código Verboso: Requiere más código para la configuración inicial y la personalización.

- *Ejemplo de interfaz:*



➢ ListView

ListView es un widget en Android que permite mostrar una lista de elementos desplazables en una sola columna. Es una de las formas mas comunes de mostrar lista de datos en Android y se utiliza principalmente para listas sencillas.



➢ *Características principales:*

1. Desplazamiento Vertical: Soporta desplazamiento vertical para listas largas.

2. Adaptadores: Utiliza adaptadores (Adapter) para enlazar los datos con las vistas. Los adaptadores comunes incluyen ArrayAdapter, SimpleAdapter y CursorAdapter.

3. View Recycling: Implementa la técnica de reciclaje de vistas para mejorar el rendimiento, reutilizando las vistas de los elementos que salen de la pantalla.

4. Item Click Listener: Permite manejar eventos de clic en los elementos de la lista mediante OnItemClickListener.

5. Compatibilidad: Es compatible con todas las versiones de Android, lo que lo hace adecuado para aplicaciones que deben soportar versiones antiguas de Android.

- *Como se organizan los objetos dentro del contenedor:*

1. Adapter: El adaptador (ListAdapter) es responsable de crear las vistas para cada elemento de la lista y enlazarlas con los datos. ArrayAdapter es uno de los adaptadores más comunes y sencillos.

2. View Recycling: Utiliza un patrón ViewHolder para mejorar el rendimiento mediante la reutilización de las vistas de los elementos.

3. XML Layout para cada Ítem: Cada elemento de la lista puede tener su propio diseño definido en un archivo XML.

- *Ventajas:*

1. Fácil de implementar para listas sencillas.

2. Funciona en todas las versiones de Android, lo que garantiza una experiencia de usuario consistente en diferentes dispositivos.

3. Soporta varios tipos de adaptadores para diferentes fuentes de datos, como arrays y bases de datos.

4. Permite personalizar los elementos de la lista mediante archivos XML de diseño.

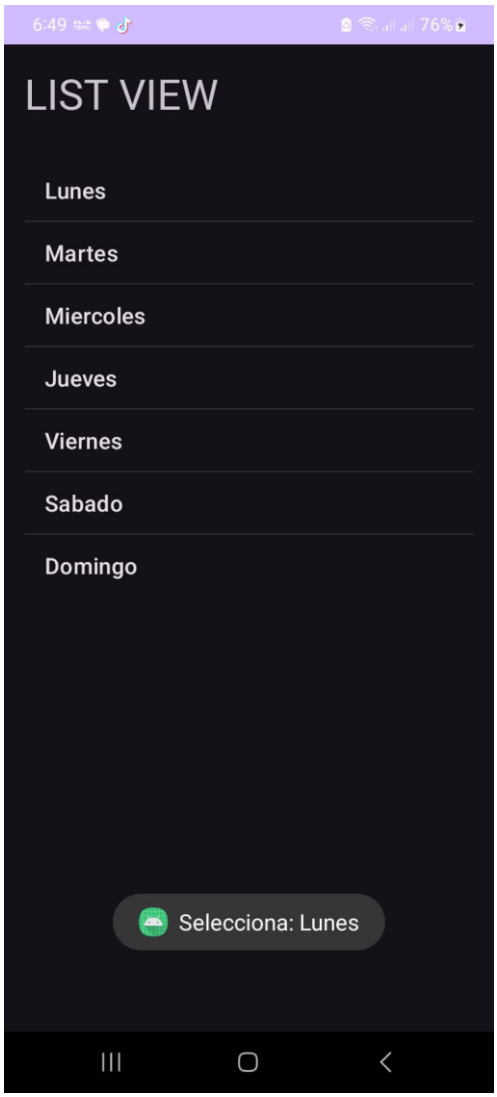
- *Desventajas:*

1. Menos flexible y más difícil de usar para diseños complejos en comparación con RecyclerView.

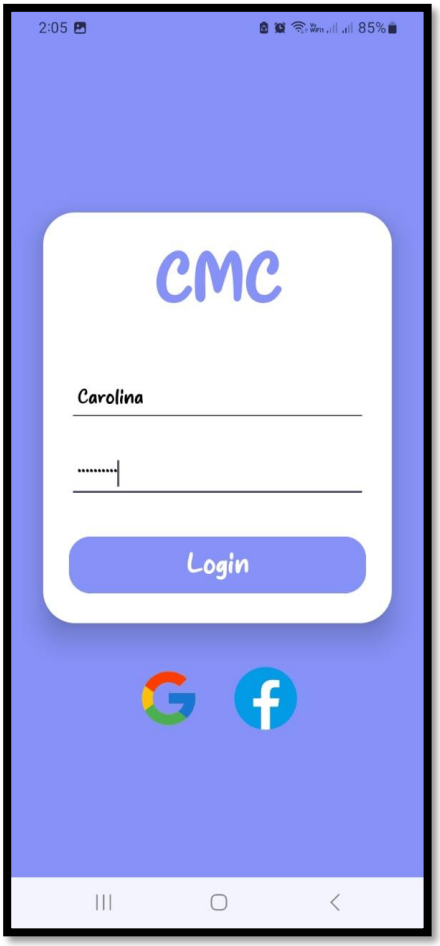
2. Menos eficiente para listas grandes en comparación con RecyclerView debido a la falta de soporte nativo para diferentes tipos de LayoutManagers y animaciones.

3. Puede volverse complejo y difícil de mantener cuando se requieren personalizaciones avanzadas de los elementos de la lista.

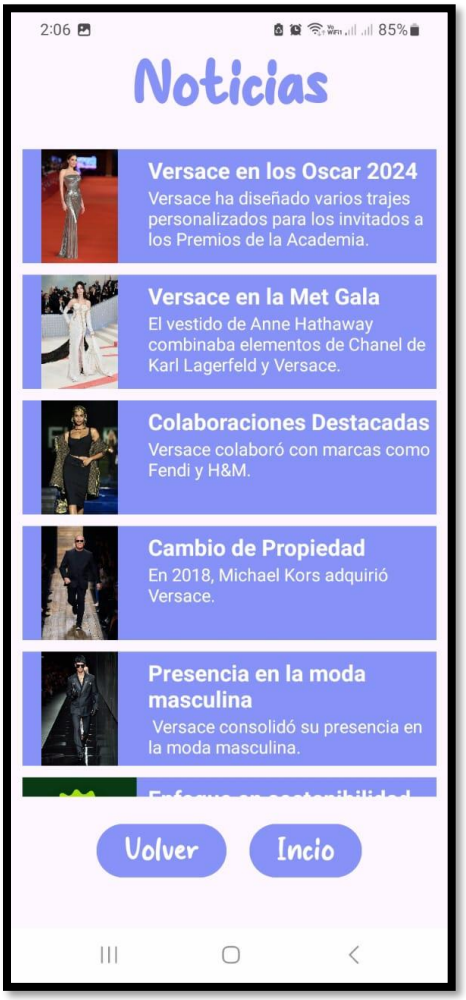
- *Ejemplo de interfaz:*



➢ Diseño de la Interfaz




2:05
85%
Lista de Productos
Trajes
Camisas
Pantalones
Jeans
Chaquetas y Abrigos
Ropa Deportiva
Ropa Interior
Vestidos
Blusas
Bolsos
Carteras
Zapatos
Volver
Noticias


2:06
85%
Noticias
Versace en los Oscar 2024
Versace ha diseñado varios trajes personalizados para los invitados a los Premios de la Academia.
Versace en la Met Gala
El vestido de Anne Hathaway combinaba elementos de Chanel de Karl Lagerfeld y Versace.
Colaboraciones Destacadas
Versace colaboró con marcas como Fendi y H&M.
Cambio de Propiedad
En 2018, Michael Kors adquirió Versace.
Presencia en la moda masculina
Versace consolidó su presencia en la moda masculina.
Volver
Incio