Guía de ejercicios - Elementos de la interfaz, navegación e interacción (I)

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar la construcción de un listado usando un adaptador.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Actividad guiada: Listado

Los listados son componentes utilizados en todas las apps, incluso en el app de reloj, ¿reconoces dónde? Al crear alarmas, ¡se ordenan en un listado!
En esta actividad guiada vamos a hacer nuestro propio listado de elementos usando componentes del SDK de Android ¡Manos a la obra!

1. Vamos a crear un nuevo proyecto usando el template de Empty Activity.

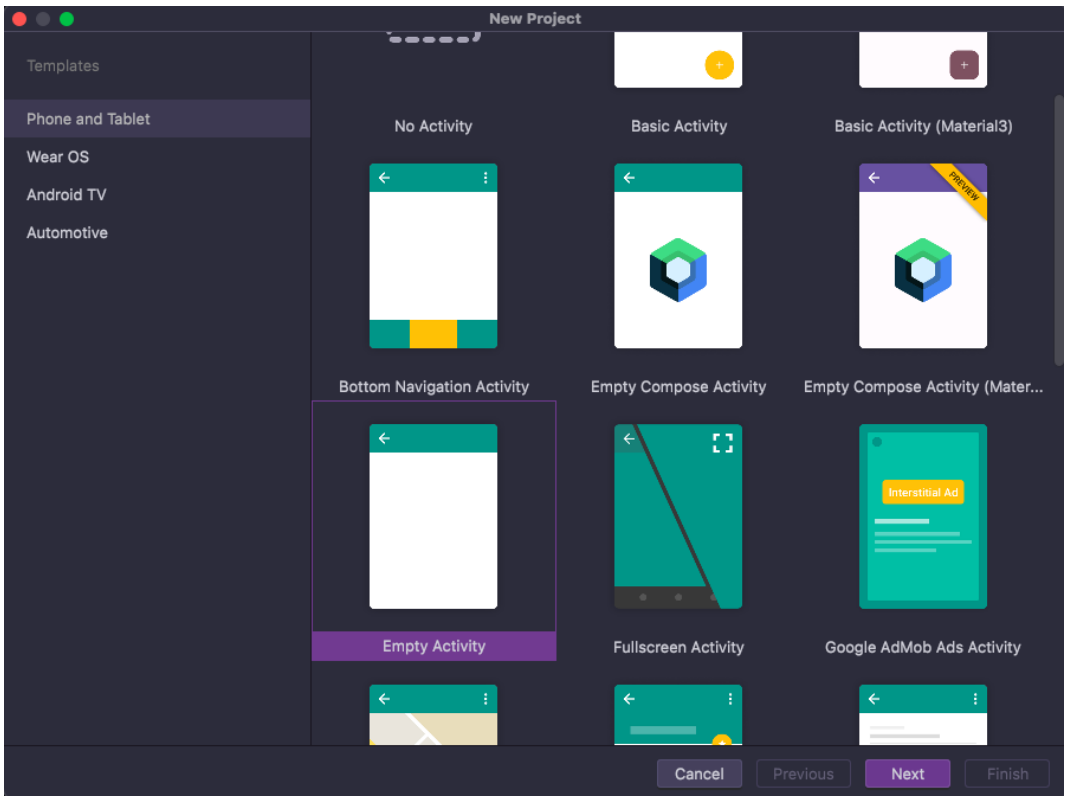

Imagen 1. Templates
Fuente: Desafío Latam.

2. Vamos a modificar el layout activity_main.xml con un ListView, arrastrando desde la paleta de vistas.

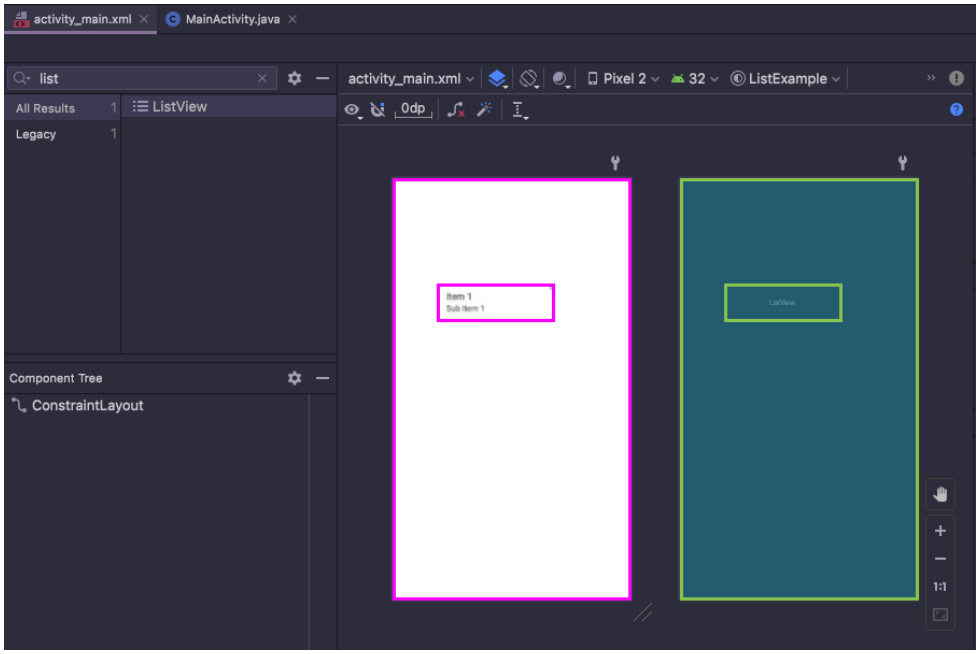

Imagen 2. List view.
Fuente: Desafío Latam.

3. Vamos a configurar el listado:
    a. Especificar el ID como `android:id="@+id/mainList"`
    b. Agregar constraints top, bottom, start, end
    c. Agregar márgenes (8dp)

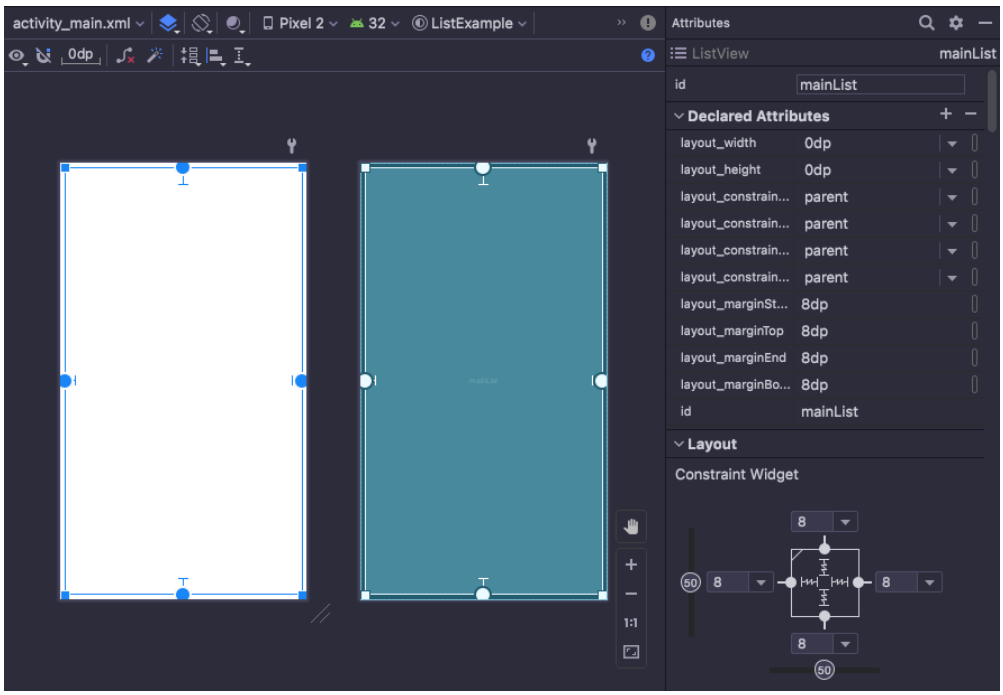

Imagen 3. Configuración.
Fuente: Desafío Latam.

4. El listado funciona como el contenedor donde se van a mostrar cada elemento, pero desconoce cuáles elementos y cómo se ven. Vamos a definir cómo se ve cada elemento, agregando el layout que será utilizado en cada elemento (item) y lo nombraremos item.xml

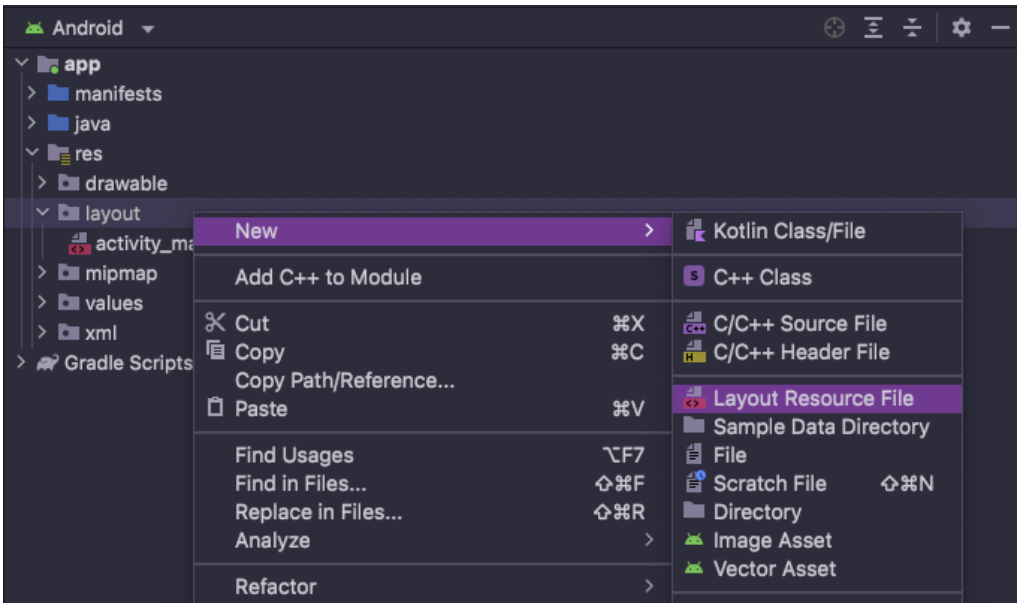

Imagen 4. Agregando el layout.
Fuente: Desafío Latam.

5. Cada elemento del listado va a tener una imagen, un título y una descripción. Para eso vamos a agregar al layout item.xml un ImageView y 2 TextView distribuidos horizontalmente.
Usaremos un CardView para redondear bordes y dar elevación.
   a. Agregar CardView que contenga un ConstraintLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.cardview.widget.CardView>
```

   b. Agregamos un nuevo vector para utilizar como imagen, usando la imagen beach access de la biblioteca de material design y cambiando el tamaño a 54dp.

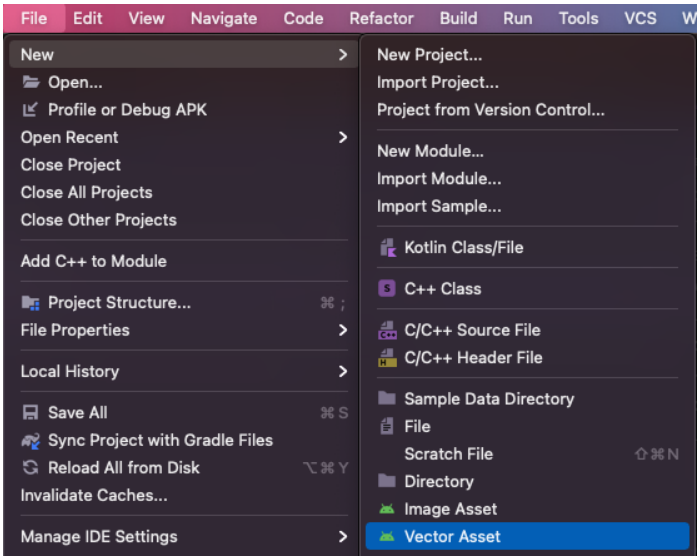

Imagen 5. Agregando nuevo vector.
Fuente: Desafío Latam.

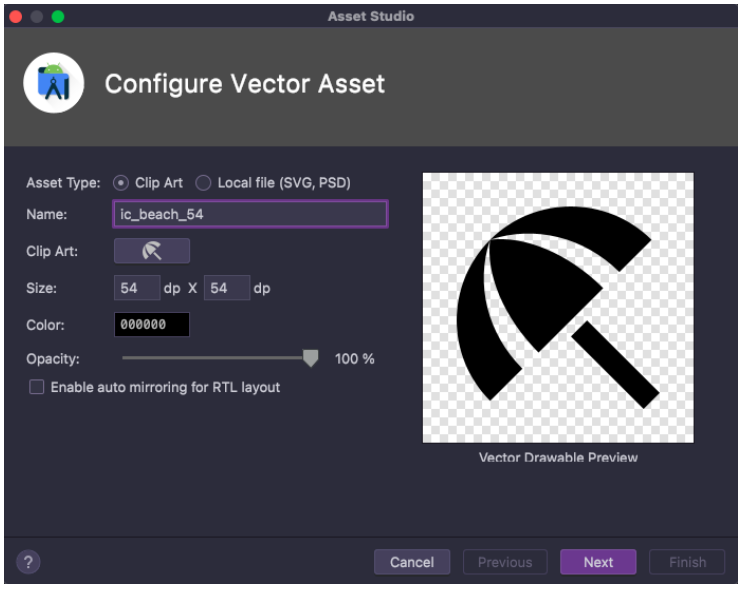

Imagen 6. Agregando el vector
Fuente: Desafío Latam.

c. Agregar un ImageView que use como recurso el vector creado en el paso anterior y agregar las constraints necesarias para que:
   i. Utilice el centro de la pantalla en forma vertical.
   ii. Tenga un margen start de 16dp que es definido en dimens.xml cómo margin_normal

dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="margin_normal">16dp</dimen>
```

```
</resources>
```

item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
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">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="@dimen/margin_normal"
            android:layout_marginTop="@dimen/margin_normal"
            android:layout_marginBottom="@dimen/margin_normal"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/ic_beach_54" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.cardview.widget.CardView>
```

d. Agregar TextView para título.
   i. Definir el texto en strings.xml.
   ii. Agregar constraints para que la parte superior e inferior estén alineadas con la imagen, y agregando una separación de 32dp.

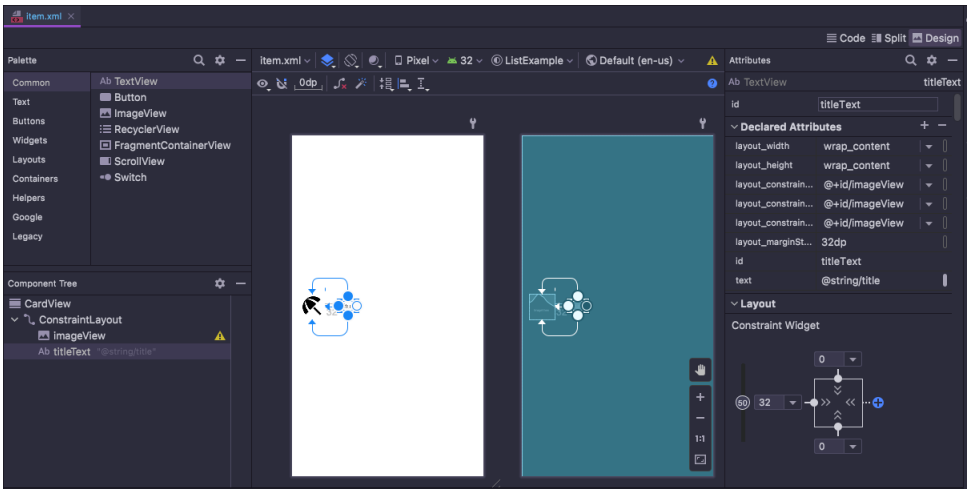

Imagen 6. Agregando TextView.
Fuente: Desafío Latam.

e. Si usamos este layout solo veremos 1 elemento de la lista a la vez, porque el CardView está usando todo el alto y ancho disponible. Debemos adecuar la altura para que dependa del contenido, cambiando la altura a:

```
android:layout_height="wrap_content"
```

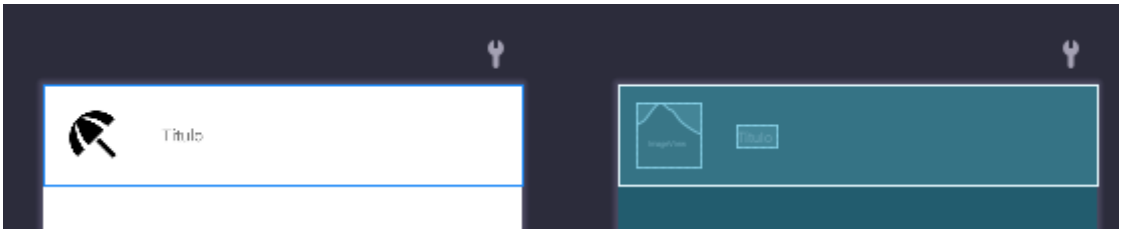

Imagen 7. Adecuando la altura.
Fuente: Desafío Latam.

f. Por último, cambiamos un poco el aspecto del CardView.
   i. Asignamos el radio de redondeo de las esquinas con `app:cardCornerRadius="12dp".`
   ii. Cambiamos la elevación por defecto a `android:elevation="6dp"` haciendo una sombra más pronunciada.

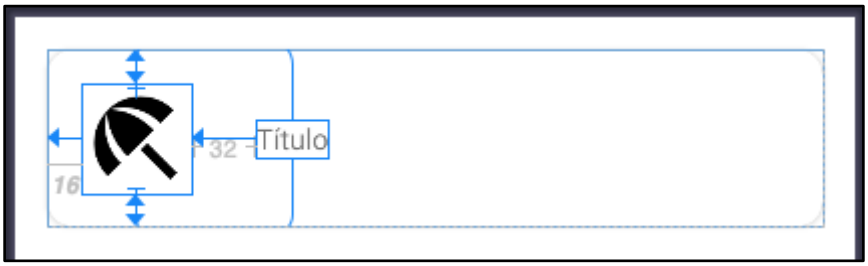

Imagen 8. Cambiando el aspecto del CardView.
Fuente: Desafío Latam.

   iii. Cambiamos el color del ConstraintLayout para que resalte, por ejemplo, usando el color `android:background="#B3E5FC"`

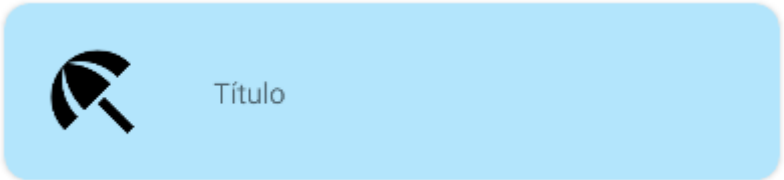

Imagen 9. Cambiando el color del ConstraintLayout.
Fuente: Desafío Latam.

6. Ahora creamos el Adaptador que va a contener la información y va a utilizar el item layout para mostrar el elemento. Vamos a generar una clase adapter privada dentro del mismo archivo de MainActivity.

```
public class MainActivity extends AppCompatActivity {

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


    private class MyAdapter extends BaseAdapter {
        private final List<String> values = new ArrayList<>();

        @Override
        public int getCount() {
            return values.size();
        }

        @Override
        public Object getItem(int i) {
            return values.get(i);
        }

        @Override
        public long getItemId(int i) {
            return 0;
        }

        @Override
        public View getView(int position, View convertView, ViewGroup container) {
            if (convertView == null) {
                convertView = getLayoutInflater().inflate(R.layout.item, container, false);
            }

            String textToShow = getItem(position).toString();
            ((TextView) convertView.findViewById(R.id.titleText)).setText(textToShow);
            return convertView;
        }

        public void setValues(List<String> newValues) {
            values.clear();
            values.addAll(newValues);
        }
    }
}
```

Este adaptador tiene como atributo de la clase la lista de valores. Cuando ListView debe mostrar el elemento número N , emplea `getView`(N) con la posición del elemento para que se construya la vista correspondiente. Para esto se infla el layout del ítem, luego se obtiene el valor seleccionado y se asigna a la vista antes de retornarla.

7. Creamos un pequeño generador de items para el listado

```
private List<String> getValues() {
    List<String> values = new ArrayList<>();
    for(int i=0; i<9; i++) {
        values.add("Elemento de listado " + i);
```

```
    }
    return values;
}
```

8. Instanciamos el adaptador y lo asociamos al listado

```
private void setUpAdapter() {
    ListView listView = (ListView) findViewById(R.id.mainList);
    MyAdapter adapter = new MyAdapter();
    adapter.setValues(getValues());
    listView.setAdapter(adapter);
}
```

9. Utilizamos el método `setUpAdapter()` en el `onCreate()` de la actividad

```
package cl.dal.listexample;

import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ListView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

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

        setUpAdapter();
    }

    private void setUpAdapter() {
        ListView listView = (ListView) findViewById(R.id.mainList);
        MyAdapter adapter = new MyAdapter();
        adapter.setValues(getValues());
        listView.setAdapter(adapter);
    }

    private List<String> getValues() {
        List<String> values = new ArrayList<>();
        for(int i=0; i<9; i++) {
            values.add("Elemento de listado " + i);
        }
        return values;
    }


    private class MyAdapter extends BaseAdapter {
        private final List<String> values = new ArrayList<>();

        @Override
        public int getCount() {
            return values.size();
        }
```

```
        @Override
        public Object getItem(int i) {
            return values.get(i);
        }

        @Override
        public long getItemId(int i) {
            return 0;
        }

        @Override
        public View getView(int position, View convertView, ViewGroup container) {
            if (convertView == null) {
                convertView = getLayoutInflater().inflate(R.layout.item, container, false);
            }

            String textToShow = getItem(position).toString();
            ((TextView) convertView.findViewById(R.id.titleText)).setText(textToShow);
            return convertView;
        }

        public void setValues(List<String> newValues) {
            values.clear();
            values.addAll(newValues);
        }
    }
}
```

10. Finalmente, al ejecutar un emulador vemos la siguiente pantalla:

Imagen 10. Pantalla terminada.
Fuente: Desafío Latam.

¡Con esto tenemos nuestro listado de elementos con un scroll vertical completamente funcional!

¡Manos a la obra! - Listado de planetas

Construye un spinner que despliegue el listado de planetas del sistema solar.
Recuerda que la información puede ser proporcionada desde el archivo de recursos arrays.xml o utilizando un adapter.

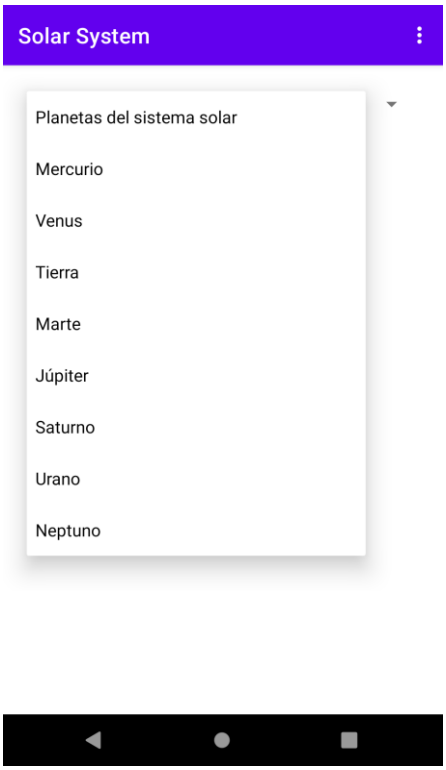

Imagen 10. Pantalla objetivo.
Fuente: Desafío Latam.

¿Y plutón?

¡Manos a la obra! - Otro listado de planetas

¿Cómo se vería el mismo listado de planetas pero en un ListView? Realízalo.

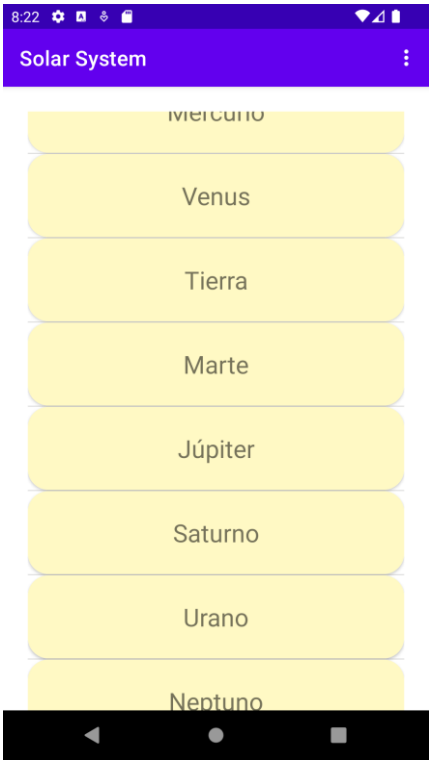

Imagen 10. Pantalla terminada con ListView.
Fuente: Desafío Latam.

Preguntas de proceso

Reflexiona:

- ¿Cómo interactúan ListView y Adapter?
- ¿Cuál es la responsabilidad de cada uno?
- ¿Qué contenido revisado hasta ahora te ha resultado más difícil?
- ¿Cuál de los contenidos revisados hasta ahora te parece útil para resolver problemas o crees que podrías ocupar a futuro?

Preguntas de cierre

- ¿Cuáles son las formas de definir la información a mostrar en una lista?
    - Recuerda que la información a mostrar puede ser estática o dinámica
    -
- ¿Qué componente tiene sentido utilizar para que el usuario seleccione su edad?

- ¿Cuál componente de los utilizados es el que más te agrada? ¿qué te permite hacer?

Solución “Manos a la obra” 1

Para el spinner se necesita:

1. Agregar la vista al layout correspondiente

```
<?xml version="1.0" encoding="utf-8"?>
<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">

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="1dp"
        android:layout_marginTop="47dp"
        android:layout_marginEnd="1dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2. Agregar el listado de planetas definidos como un string-array en el archivo arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets">
        <item>Planetas del sistema solar</item>
        <item>Mercurio</item>
        <item>Venus</item>
        <item>Tierra</item>
        <item>Marte</item>
```

```
        <item>Júpiter</item>
        <item>Saturno</item>
        <item>Urano</item>
        <item>Neptuno</item>
    </string-array>
</resources>
```

3. Crear el adaptador y asociarlo al spinner en el método onViewCreated del fragmento o en el onCreate de la actividad.

```
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(getContext(), R.array.planets,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    binding.spinner.setAdapter(adapter);
}
```

Solución “Manos a la obra” 2

Para el ListView se necesita:

1. Agregar la vista al layout

```
<?xml version="1.0" encoding="utf-8"?>
<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">

    <ListView
        android:id="@+id/listView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/margin_normal"
        android:layout_marginTop="@dimen/margin_normal"
        android:layout_marginEnd="@dimen/margin_normal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2. Crear el adapter para contener y manejar el listado de planetas

```
class MyAdapter extends BaseAdapter {
   private final List<String> values = new ArrayList<>();

   @Override
   public int getCount() {
       return values.size();
   }

   @Override
   public Object getItem(int i) {
       return values.get(i);
   }

   @Override
   public long getItemId(int i) {
       return 0;
   }

   @Override
   public View getView(int position, View convertView, ViewGroup container) {
       if (convertView == null) {
           convertView = getLayoutInflater().inflate(R.layout.item, container, false);
       }

       String textToShow = getItem(position).toString();
       ((TextView) convertView.findViewById(R.id.titleText)).setText(textToShow);
       return convertView;
   }

   public void setValues(List<String> newValues) {
       values.clear();
       values.addAll(newValues);
   }
}
```

3. Asociar el adaptador con el listado en el método onViewCreated() del fragment

```
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
   super.onViewCreated(view, savedInstanceState);

   MyAdapter adapter = new MyAdapter();
   adapter.setValues(getValues());
   binding.listView.setAdapter(adapter);
}
```