Listas y Adaptadores - Parte II

Multi ViewHolder

Competencias

- Aprender a usar más de un ViewHolder en Un RecyclerView.
- Aprender a mostrar diferentes tipos de ítems en una aplicación
- Uso del método getItemViewType

Motivación

Enseñar que un RecyclerView se pueden mostrar diferentes ítems usando diferentes clases de ViewHolder y vistas XML item_list, esto debido a que en un Dataset ArrayList/List pueden venir diferentes elementos entres sus ítems con un atributo que los diferencie uno de otro y esa data debe reflejarse en un solo RecyclerView con ajustes y cambios en sus métodos con el fin de cumplir de cumplir con los requerimientos de la aplicación.

Para que se usa un Multi ViewHolder en un RecyclerView

Cuando necesitamos mostrar diferentes vistas de los ítems que forma parte del RecyclerView, estas vistas pueden tener los mismos o diferentes atributos. Un ejemplo de Multi ViewHolder en un RecyclerView podría ser cuando estamos desarrollando un chat, ya que se usa un Dataset (ArrayList/List) que contiene los mensajes de dos personas y esta lista de mensajes se pueden visualizar en un mismo timeline que estaría representado por un RecyclerView. Otro ejemplo lo observamos en la imagen a continuación donde uno de los ítems contienen texto y una imagen, pero otro contienen un texto y un botón.

Los ítems pueden ser completamente distintos, la solución que conoceremos a continuación aplica para todos los casos.

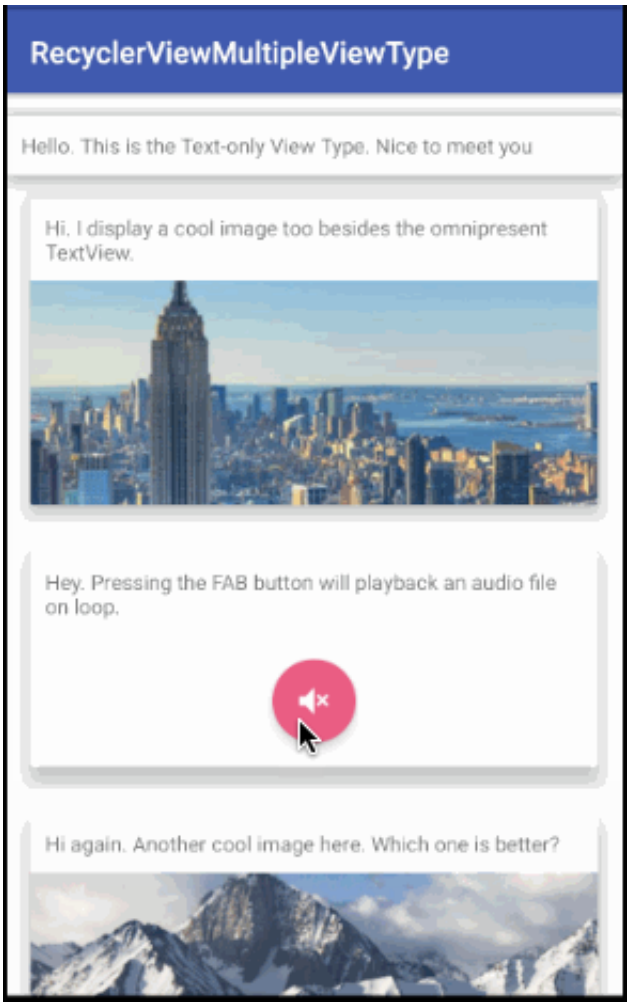

Imagen 1. Multi ViewHolder

Construir una Multi ViewHolder

Para visualizar una Multi ViewHolder en un RecyclerView se deben construir dos o más archivos item_list XML, cada archivo XML va a tener un nombre distinto según sean los requerimientos de la aplicación, al igual se deben crear 2 o más clases ViewHolder en donde cada una se va enlazar con su respectivo archivo layout item_list, a su vez estas clases ViewHolder van a ir enlazadas a una única clase Adapter en donde se realizarán las diferentes operaciones de cada uno de estos ViewHolder.

Veamos un ejemplo de una agenda diaria de una persona en donde se van a efectuar diferentes actividades.

1. Lo primero que veremos serán el POJO de que vamos a utilizar para definir los ítems de la lista que se visualizará en el RecyclerView.

```
package com.myrecyclerview.recyclerviewwithheaderfooter;

public class Item {
    private String mContent;
    private int mIcon;
    private int mImage;
    private String mTime;
    private boolean isHeader = false;
    private boolean withoutImage = false;
    private boolean withImage = false;
    private boolean isFooter = false;
    public Item(String mContent, int mIcon, int mImage, String mTime) {
        this.mContent = mContent;
        this.mIcon = mIcon;
        this.mImage = mImage;
        this.mTime = mTime;
        withImage = true;
    }
    public Item(String mContent, int mIcon, String mTime) {
        this.mContent = mContent;
        this.mIcon = mIcon;
        this.mTime = mTime;
        withoutImage = true;
    }

    public Item(String mTime) {
        this.mContent = null;
        this.mIcon = -1;
        this.mImage = -1;
        this.mTime = mTime;
        isHeader = true;
    }
```

```
    public Item() {
        this.mContent = null;
        this.mIcon = -1;
        this.mImage = -1;
        this.mTime = null;
        isFooter = true;
    }
    public String getContent() {
        return mContent;
    }
    public int getIcon() {
        return mIcon;
    }
    public int getImage() {
        return mImage;
    }
    public String getTime() {
        return mTime;
    }
    public String getmContent() {
        return mContent;
    }
    public boolean isHeader() {
        return isHeader;
    }
    public boolean isWithoutImage() {
        return withoutImage;
    }
    public boolean isWithImage() {
        return withImage;
    }
    public boolean isFooter() {
        return isFooter;
    }
}
```

2. Ahora veremos la fuente de datos estática que alimentara al RecyclerView de nuestra aplicación.

```
private void initData() {
    mItems = new ArrayList<>();
    mItems.add(new Item("Hoy"));
    mItems.add(new Item(
            "Ya acordé una rica cena con mi pololo, ¿será un momento de comida
japonesa?",
            R.mipmap.deliver_food,
            R.drawable.bento,
            "7:30 pm"));
    mItems.add(new Item(
            "Me toca cocinar en casa, quizás un delicioso pollo al horno.",
            R.mipmap.cooker,
            R.drawable.cooking_pot,
            "12:30 pm"));
    mItems.add(new Item(
            "Despertando para un rico café.",
            R.mipmap.cafe,
            "7:00 AM"));
    mItems.add(new Item());
    mItems.add(new Item("Ayer"));
    mItems.add(new Item(
            "Celebrando el cumpleaños de mi hermana, ya son 30 años.",
            R.mipmap.beer_bottle,
            R.drawable.birthday_cake,
            "8:00 pm"
    ));
    mItems.add(new Item(
            "En el trabajo, a veces tomo una merienda en las tardes, hoy toco
un Cupcake.",
            R.mipmap.cupcake,
            "4:30 pm"
    ));
    mItems.add(new Item(
            "Invite a almorzar a mi polola, por cumplir otro mes juntos.",
            R.mipmap.waiter,
            R.drawable.spaghetti,
            "12:15 pm"
    ));
    mItems.add(new Item(
            "Me desperté muy tarde hoy... muy tarde. Sale algo rápido.",
            R.mipmap.bread,
            "8:30 am"
    ));
```

```
    mItems.add(new Item());
    mItems.add(new Item("Domingo 16-Agosto-2015"));
    mItems.add(new Item(
            "Me encanta cenar en la calle. Cena con mi pareja.",
            R.mipmap.restaurant_pickup,
            R.drawable.noodles,
            "7:20 pm"
    ));
    mItems.add(new Item(
            "Parrilla con los amigos.",
            R.mipmap.sun,
            R.drawable.kebab,
            "3:00 pm"
    ));
    mItems.add(new Item());
    mItems.add(new Item("Sabado 15-Agosto-2015"));
    mItems.add(new Item(
            "Bar con los amigos, tomando unas cervezas.",
            R.mipmap.beer,
            "8:40 pm"
    ));
    mItems.add(new Item(
            "Adoro ir a McDonalds\'s.",
            R.mipmap.french_fries,
            R.drawable.hamburger,
            "7:15 pm"
    ));
    mItems.add(new Item(
            "Almuerzo con unos amigos.",
            R.mipmap.vegetarian_food,
            R.drawable.wrap,
            "12:40 pm"
    ));
    mItems.add(new Item());
}
```

3. Ahora vamos a crear los archivos item_list XML que van a formar parte del RecyclerView para ellos nos posicionamos en la columna del lado izquierdo del entorno de desarrollo de Android Studio en la sección **res → layout** hacemos click derecho y nos aparece una ventana de menú donde seleccionamos la opción **new → Layout resource file**, luego nos aparece una ventana llamada **New Resource File** en donde le colocamos el nombre a nuestro archivo item_list y seleccionamos que tipo de layout será la vista padre de la siguiente manera:

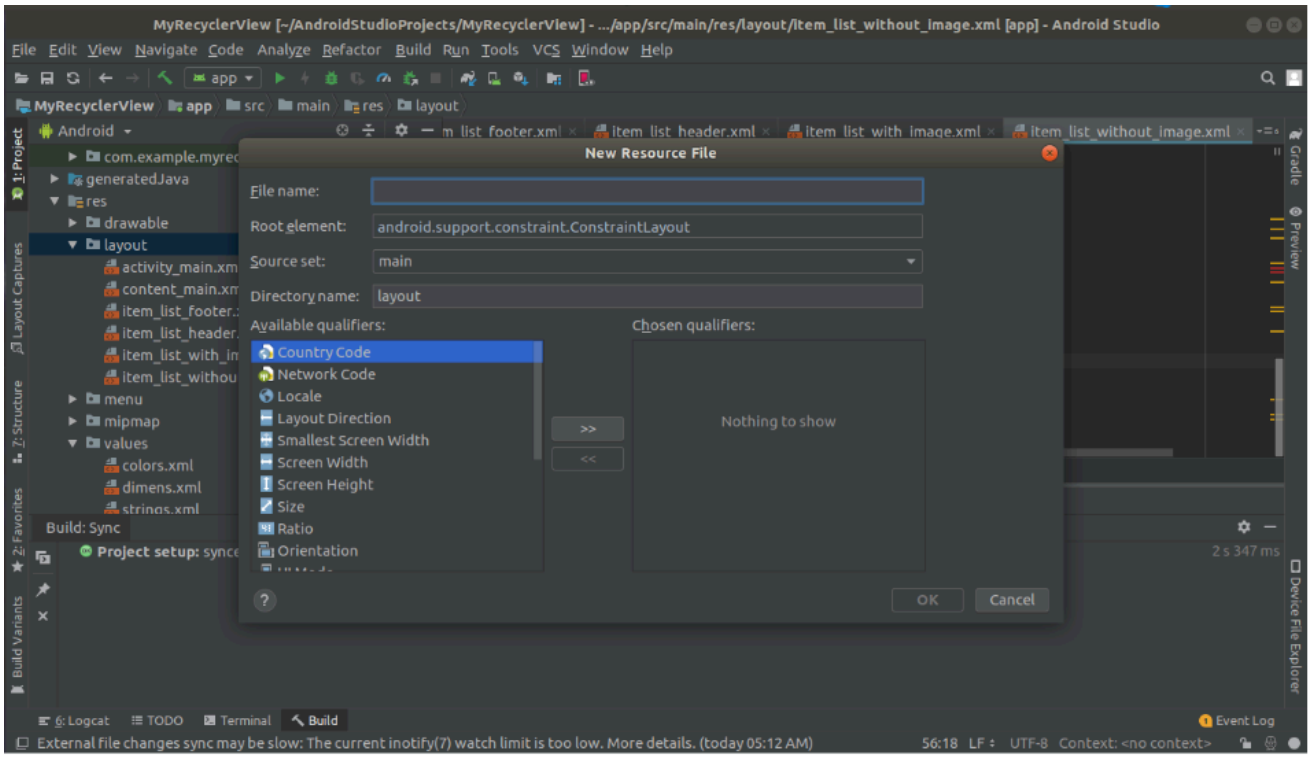

Imagen 2. Agregar recurso.

4. A continuación vamos a crear cuatro archivos xml para que podamos mostrar elementos distintos en nuestro recyclerView. Como mencionamos anteriormente la cantidad de archivos xml distintos va a depender de los requerimientos visuales de nuestra aplicación.

item_list_footer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="20dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:background="@android:color/white">
    <View
        android:layout_width="1dp"
        android:layout_height="match_parent"
        android:background="@android:color/darker_gray"
        android:layout_marginLeft="24dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"/>
</RelativeLayout>
```

item_list_header.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="70dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical"
    android:background="@android:color/white">
    <RelativeLayout
        android:id="@+id/container_top"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:paddingBottom="8dp">
```

```
        <View
            android:id="@+id/line_top"
            android:layout_width="1dp"
            android:layout_height="wrap_content"
            android:background="@android:color/darker_gray"
            android:layout_marginLeft="24dp"
            android:layout_alignParentTop="true"
            android:layout_alignParentLeft="true"
            android:layout_above="@+id/circle_top"/>
        <View
            android:id="@+id/circle_top"
            android:layout_width="8dp"
            android:layout_height="8dp"
            android:padding="8dp"
            android:layout_marginLeft="20dp"
            android:background="@drawable/circle_line_top"
            android:layout_alignParentLeft="true"
            android:layout_alignParentBottom="true"/>
    </RelativeLayout>
    <RelativeLayout
        android:id="@+id/container_bottom"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1">
        <View
            android:id="@+id/line_bottom"
            android:layout_width="1dp"
            android:layout_height="wrap_content"
            android:background="@android:color/darker_gray"
            android:layout_marginLeft="24dp"
            android:layout_alignParentLeft="true"
            android:layout_toEndOf="@+id/circle_bottom"
            android:layout_marginTop="12dp"/>
        <View
            android:id="@+id/circle_bottom"
            android:layout_width="12dp"
            android:layout_height="12dp"
            android:padding="8dp"
            android:layout_marginLeft="18dp"
            android:background="@drawable/circle_line_bottom"
            android:layout_alignParentLeft="true"
            android:layout_marginTop="6dp"/>
```

```
        <TextView
              android:id="@+id/content_header"
              android:layout_toRightOf="@id/circle_bottom"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_weight="1"
              android:text="Today"
              android:textColor="@android:color/darker_gray"
              android:textSize="18sp"
              android:textStyle="bold"
              android:layout_marginLeft="16dp" />
    </RelativeLayout>
</LinearLayout>
```

item_list_with_image.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:background="@android:color/white">
    <View
        android:layout_width="1dp"
        android:layout_height="150dp"
        android:layout_marginLeft="24dp"
        android:background="@android:color/darker_gray"
        android:layout_alignParentLeft="true" />
    <ImageView
        android:id="@+id/icon"
        android:layout_width="22dp"
        android:layout_height="22dp"
        android:padding="2dp"
        android:layout_marginLeft="13dp"
        android:background="@drawable/circle_background"
        android:src="@mipmap/beer"
        android:scaleType="centerInside"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="6dp"/>
```

```
    <TextView
        android:id="@+id/content_item"
        android:layout_toRightOf="@id/icon"
        android:layout_toLeftOf="@id/time_item"
        android:layout_alignTop="@id/icon"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ya acordé una rica cena con mi novio, ¿será un momento de
comida japonesa?"
        android:maxLines="4"
        android:ellipsize="end"
        android:gravity="start"
        android:textColor="@color/text_normal"
        android:textSize="16sp"
        android:layout_marginRight="8dp"
        android:layout_marginLeft="8dp"/>
    <ImageView
        android:id="@+id/image"
        android:layout_below="@+id/content_item"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:src="@drawable/wrap"
        android:layout_marginTop="5dp"
        android:scaleType="centerInside"
        android:layout_centerHorizontal="true"/>
    <TextView
        android:id="@+id/time_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/icon"
        android:text="10:00 PM"
        android:textColor="@color/text_list"
        android:textSize="13sp"
        android:layout_marginRight="6dp"
        android:layout_alignParentRight="true"/>
</RelativeLayout>
```

item_list_without_image.xml

```
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:background="@color/window_background">
    <View
        android:layout_width="1dp"
        android:layout_height="80dp"
        android:layout_marginLeft="24dp"
        android:background="@color/darkGray"
        android:layout_alignParentLeft="true"/>
    <ImageView
        android:id="@+id/icon"
        android:layout_width="22dp"
        android:layout_height="22dp"
        android:padding="2dp"
        android:layout_marginLeft="13dp"
        android:background="@drawable/circle_background"
        android:src="@mipmap/beer"
        android:scaleType="centerInside"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="6dp"/>
    <TextView
        android:id="@+id/content_item"
        android:layout_toRightOf="@+id/icon"
        android:layout_toLeftOf="@+id/time_item"
        android:layout_alignTop="@+id/icon"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Ya acordé una rica cena con mi novio, ¿será un momento de
comida japonesa?"
        android:gravity="start"
        android:maxLines="4"
        android:ellipsize="end"
        android:textColor="@color/text_normal"
        android:textSize="16sp"
        android:layout_marginRight="8dp"
        android:layout_marginLeft="8dp" />
```

```
    <TextView
          android:id="@+id/time_item"
          android:layout_alignTop="@+id/icon"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="10:00 PM"
          android:textColor="@color/text_list"
          android:textSize="12sp"
          android:layout_marginRight="6dp"
          android:layout_alignParentRight="true"/>
</RelativeLayout>
```

5. El siguiente paso es crear la clase adapter y dentro de ella crear los cuatro viewholder personalizados para cada uno de estos item_list que ya definimos
6. A continuación podemos ver como declaramos nuestra clase adapter con sus variables y su respectivo constructor.

```
public class DailyMenuAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
   private LayoutInflater mLayoutInflater;
    private ArrayList<Item> mItems;
    public static final int HEADER = 1;
    public static final int NORMAL_WITHOUT_IMAGE = 2;
    public static final int NORMAL_WITH_IMAGE = 3;
    public static final int FOOTER = 4;
    private Context mContext;

    public DailyMenuAdapter(Context context, ArrayList<Item> items) {
        mContext = context;
        mLayoutInflater = mLayoutInflater.from(context);
        mItems = items;
    }
```

7. Ahora procedemos a crear las cuatro clases ViewHolder

```
public class Header extends RecyclerView.ViewHolder {
  private TextView mTime;
    public Header(View itemView) {
        super(itemView);
        mTime = (TextView) itemView.findViewById(R.id.content_header);
    }
    public void setTime(String text) {
        mTime.setText(text);
    }
}

public class ItemWithoutImage extends RecyclerView.ViewHolder {
    private ImageView mIcon;
    private TextView mContent;
    private TextView mTime;
    public ItemWithoutImage(View itemView) {
        super(itemView);
        mIcon = (ImageView) itemView.findViewById(R.id.icon);
        mContent = (TextView) itemView.findViewById(R.id.content_item);
        mTime = (TextView) itemView.findViewById(R.id.time_item);
    }
    public void setIcon(int image) {
        mIcon.setImageResource(image);
    }
    public void setContent(String text) {
        mContent.setText(text);
    }
    public void setTime(String text) {
        mTime.setText(text);
    }
}
```

```
public class ItemWithImage extends RecyclerView.ViewHolder implements
View.OnClickListener {
    private ImageView mIcon;
    private TextView mContent;
    private TextView mTime;
    private ImageView mImage;
    public ItemWithImage(View itemView) {
        super(itemView);
        mIcon = (ImageView) itemView.findViewById(R.id.icon);
        mContent = (TextView) itemView.findViewById(R.id.content_item);
        mTime = (TextView) itemView.findViewById(R.id.time_item);
        mImage = (ImageView) itemView.findViewById(R.id.image);
        mImage.setOnClickListener(this);
    }
    public void setIcon(int image) {
        mIcon.setImageResource(image);
    }
    public void setContent(String text) {
        mContent.setText(text);
    }
    public void setTime(String text) {
        mTime.setText(text);
    }
    public void setImage(int image) {
        mImage.setTag(image);
        mImage.setImageResource(image);
    }

    @Override
    public void onClick(View v) {
        switch ((Integer) v.getTag()) {
            case R.drawable.bento:
                Toast.makeText(mContext, "¡Tiempo de sushi!",
Toast.LENGTH_SHORT).show();
                break;
            case R.drawable.birthday_cake:
                Toast.makeText(mContext, "Una torta de fresa, rellena de dulce
de fresa con capa de fresa. ¡Umm!", Toast.LENGTH_SHORT).show();
                break;
            case R.drawable.cooking_pot:
                Toast.makeText(mContext, "Detesto la sopa. ;)",
Toast.LENGTH_SHORT).show();
                break;
```

```
            case R.drawable.hamburger:
                Toast.makeText(mContext, "¿Quien no ama McDonald\'s?",
Toast.LENGTH_SHORT).show();
                break;
            case R.drawable.kebab:
                Toast.makeText(mContext, "Sabrosas las brochetas.",
Toast.LENGTH_SHORT).show();
                break;
            case R.drawable.spaghetti:
                Toast.makeText(mContext, "No soy amante de la comida
italiana.", Toast.LENGTH_SHORT).show();
                break;
            case R.drawable.noodles:
                Toast.makeText(mContext, "Tampoco amante de los fideos.",
Toast.LENGTH_SHORT).show();
                break;
            case R.drawable.wrap:
                Toast.makeText(mContext, "¡Ohh shawarma!",
Toast.LENGTH_SHORT).show();
                break;
        }
    }
}

public class Footer extends RecyclerView.ViewHolder {
    public Footer(View itemView) {
        super(itemView);
    }
}
```

8. Definimos nuestro método getItemCount()

```
@Override
public int getItemCount() {
    return mItems.size();
}
```

9. Ahora es el turno de definir el nuevo método getItemType el cual nos permitirá controlar y retornar como se van a mostrar los ítem en el ViewHolder

```
@Override
public int getItemViewType(int position) {
    if (mItems.get(position).isHeader())
        return HEADER;
    else if (mItems.get(position).isWithoutImage())
        return NORMAL_WITHOUT_IMAGE;
    else if (mItems.get(position).isWithImage())
        return NORMAL_WITH_IMAGE;
    else
        return FOOTER;
}
```

10. Definimos el método onCreateViewHolder con las múltiples View Holder

```
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int
viewType) {
    View mItemView;
    switch (viewType) {
        case HEADER:
            mItemView = mLayoutInflater.inflate(R.layout.custom_header,
viewGroup, false);
            return new Header(mItemView);
        case NORMAL_WITHOUT_IMAGE:
             mItemView = mLayoutInflater.inflate(
                         R.layout.custom_item_list_without_image,
                         viewGroup, false);
             return new ItemWithoutImage(mItemView);
        case NORMAL_WITH_IMAGE:
            mItemView = mLayoutInflater.inflate(
                        R.layout.custom_item_list_with_image,
                        viewGroup, false);
            return new ItemWithImage(mItemView);
        case FOOTER:
            mItemView = mLayoutInflater.inflate(
                        R.layout.custom_footer, viewGroup, false);
            return new Footer(mItemView);
        default:
            return null;
    }
}
```

En el ejemplo anterior utilizamos un switch donde cada elemento del case nos permite determinar el tipo de vista que se va a mostrar en el recyclerView.

11. Ahora es el turno de definir el onBindViewHolder

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position)
{
    switch (getItemViewType(position)) {
        case HEADER:
            Header mHeader = (Header) viewHolder;
            mHeader.setTime(mItems.get(position).getTime());
            break;
        case NORMAL_WITHOUT_IMAGE:
            ItemWithoutImage mItemWithout = (ItemWithoutImage) viewHolder;
            mItemWithout.setIcon(mItems.get(position).getIcon());
            mItemWithout.setContent(mItems.get(position).getContent());
            mItemWithout.setTime(mItems.get(position).getTime());
            break;
        case NORMAL_WITH_IMAGE:
            ItemWithImage mItemWith = (ItemWithImage) viewHolder;
            mItemWith.setIcon(mItems.get(position).getIcon());
            mItemWith.setContent(mItems.get(position).getContent());
            mItemWith.setTime(mItems.get(position).getTime());
            mItemWith.setImage(mItems.get(position).getImage());
            break;
    }
}
```

En este método, dependiendo del tipo de dato procederemos a enlazar los datos de nuestro objeto con su respectivo viewHolder, es por ello que implementamos un switch nuevamente.

12. Podemos visualizar el código completo de la clase DailyMenuAdapter

```
package com.example.myrecyclerview.adapter;
public class DailyMenuAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    private LayoutInflater mLayoutInflater;
    private ArrayList<Item> mItems;
    public static final int HEADER = 1;
    public static final int NORMAL_WITHOUT_IMAGE = 2;
    public static final int NORMAL_WITH_IMAGE = 3;
    public static final int FOOTER = 4;
    private Context mContext;
    public DailyMenuAdapter(Context context, ArrayList<Item> items) {
        mContext = context;
        mLayoutInflater = mLayoutInflater.from(context);
        mItems = items;
    }
    public class Header extends RecyclerView.ViewHolder {
        private TextView mTime;
        public Header(View itemView) {
            super(itemView);
            mTime = (TextView) itemView.findViewById(R.id.content_header);
        }
        public void setTime(String text) {
            mTime.setText(text);
        }
    }
    public class ItemWithoutImage extends RecyclerView.ViewHolder {
        private ImageView mIcon;
        private TextView mContent;
        private TextView mTime;
        public ItemWithoutImage(View itemView) {
            super(itemView);
            mIcon = (ImageView) itemView.findViewById(R.id.icon);
            mContent = (TextView) itemView.findViewById(R.id.content_item);
            mTime = (TextView) itemView.findViewById(R.id.time_item);
        }
        public void setIcon(int image) {
            mIcon.setImageResource(image);
        }
        public void setContent(String text) {
            mContent.setText(text);
        }
        public void setTime(String text) {
            mTime.setText(text);
```

```
        }
    }
    public class ItemWithImage extends RecyclerView.ViewHolder implements
View.OnClickListener {
        private ImageView mIcon;
        private TextView mContent;
        private TextView mTime;
        private ImageView mImage;
        public ItemWithImage(View itemView) {
            super(itemView);
            mIcon = (ImageView) itemView.findViewById(R.id.icon);
            mContent = (TextView) itemView.findViewById(R.id.content_item);
            mTime = (TextView) itemView.findViewById(R.id.time_item);
            mImage = (ImageView) itemView.findViewById(R.id.image);
            mImage.setOnClickListener(this);
        }
        public void setIcon(int image) {
            mIcon.setImageResource(image);
        }
        public void setContent(String text) {
            mContent.setText(text);
        }
        public void setTime(String text) {
            mTime.setText(text);
        }
        public void setImage(int image) {
            mImage.setTag(image);
            mImage.setImageResource(image);
        }
        @Override
        public void onClick(View v) {
            switch ((Integer) v.getTag()) {
                case R.drawable.bento:
                    Toast.makeText(mContext, "¡Tiempo de sushi!",
Toast.LENGTH_SHORT).show();
                    break;
                case R.drawable.birthday_cake:
                    Toast.makeText(mContext, "Una torta de fresa, rellena de
dulce de fresa con capa de fresa. ¡Umm!", Toast.LENGTH_SHORT).show();
                    break;
                case R.drawable.cooking_pot:
                    Toast.makeText(mContext, "Detesto la sopa. ;)",
Toast.LENGTH_SHORT).show();
                    break;
```

```
                case R.drawable.hamburger:
                    Toast.makeText(mContext, "¿Quien no ama McDonald\'s?",
Toast.LENGTH_SHORT).show();
                    break;
                case R.drawable.kebab:
                    Toast.makeText(mContext, "Sabrosas las brochetas.",
Toast.LENGTH_SHORT).show();
                    break;
                case R.drawable.spaghetti:
                    Toast.makeText(mContext, "No soy amante de la comida
italiana.", Toast.LENGTH_SHORT).show();
                    break;
                case R.drawable.noodles:
                    Toast.makeText(mContext, "Tampoco amante de los fideos.",
Toast.LENGTH_SHORT).show();
                    break;
                case R.drawable.wrap:
                    Toast.makeText(mContext, "¡Ohh shawarma!",
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    }
    public class Footer extends RecyclerView.ViewHolder {
        public Footer(View itemView) {
            super(itemView);
        }
    }
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int
viewType) {
        View mItemView;
        switch (viewType) {
            case HEADER:
                mItemView = mLayoutInflater.inflate(R.layout.custom_header,
viewGroup, false);
                return new Header(mItemView);
            case NORMAL_WITHOUT_IMAGE:
                mItemView =
mLayoutInflater.inflate(R.layout.custom_item_list_without_image, viewGroup,
false);
                return new ItemWithoutImage(mItemView);
```

```
            case NORMAL_WITH_IMAGE:
                mItemView =
mLayoutInflater.inflate(R.layout.custom_item_list_with_image, viewGroup,
false);
                return new ItemWithImage(mItemView);
            case FOOTER:
                mItemView = mLayoutInflater.inflate(R.layout.custom_footer,
viewGroup, false);
                return new Footer(mItemView);
            default:
                return null;
        }
    }
    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int
position) {
        switch (getItemViewType(position)) {
            case HEADER:
                Header mHeader = (Header) viewHolder;
                mHeader.setTime(mItems.get(position).getTime());
                break;
            case NORMAL_WITHOUT_IMAGE:
                ItemWithoutImage mItemWithout = (ItemWithoutImage) viewHolder;
                mItemWithout.setIcon(mItems.get(position).getIcon());
                mItemWithout.setContent(mItems.get(position).getContent());
                mItemWithout.setTime(mItems.get(position).getTime());
                break;
            case NORMAL_WITH_IMAGE:
                ItemWithImage mItemWith = (ItemWithImage) viewHolder;
                mItemWith.setIcon(mItems.get(position).getIcon());
                mItemWith.setContent(mItems.get(position).getContent());
                mItemWith.setTime(mItems.get(position).getTime());
                mItemWith.setImage(mItems.get(position).getImage());
                break;
        }
    }
    @Override
    public int getItemCount() {
        return mItems.size();
    }
```

```
    @Override
    public int getItemViewType(int position) {
        if (mItems.get(position).isHeader())
            return HEADER;
        else if (mItems.get(position).isWithoutImage())
            return NORMAL_WITHOUT_IMAGE;
        else if (mItems.get(position).isWithImage())
            return NORMAL_WITH_IMAGE;
        else
            return FOOTER;
    }
}
```

13. Acá podemos apreciar el código XML perteneciente al layout activity_daily_menu en donde se encuentra el RecyclerView de esta aplicación

```
<RelativeLayout
   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"
    android:background="@color/window_background"
    tools:context=".DailyMenuActivity">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recycler"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

14. Ahora pasamos al MainActivity de la aplicación que es la clase principal de la aplicación en donde tenemos internamente la clase que va a alimentar al RecyclerView con data estática y la nacionalización e implementación del RecyclerView.

```
package com.example.myrecyclerview;
public class MainActivity extends AppCompatActivity {
    private RecyclerView mRecycler;
    private DailyMenuAdapter mAdapter;
    private ArrayList<Item> mItems;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });
        initData();
        init();
    }
    private void init() {
        mRecycler = (RecyclerView) findViewById(R.id.recycler);
        mAdapter = new DailyMenuAdapter(this, mItems);
        mRecycler.setAdapter(mAdapter);
        mRecycler.setHasFixedSize(true);
        mRecycler.setLayoutManager(new LinearLayoutManager(this));
    }
```

```
    private void initData() {
        mItems = new ArrayList<>();
        mItems.add(new Item("Hoy"));
        mItems.add(new Item(
                "Ya cuadre una rica cena con mi novio, ¿será un momento de
comida japonesa? Espero le de antojos :)",
                R.mipmap.deliver_food,
                R.drawable.bento,
                "7:30 pm"));
        mItems.add(new Item(
                "Me toca cocina en casa, quizás un delicioso pollo al horno.",
                R.mipmap.cooker,
                R.drawable.cooking_pot,
                "12:30 pm"));
        mItems.add(new Item(
                "Despertando para un rico café.",
                R.mipmap.cafe,
                "7:00 AM"));
        mItems.add(new Item());
        mItems.add(new Item("Ayer"));
        mItems.add(new Item(
                "Celebrando el cumpleaños de mi hermana, ya son 30 años.",
                R.mipmap.beer_bottle,
                R.drawable.birthday_cake,
                "8:00 pm"
        ));
        mItems.add(new Item(
                "En el trabajo, a veces tomo una merienda en las tardes, hoy
toco un \"Cupcake\".",
                R.mipmap.cupcake,
                "4:30 pm"
        ));
        mItems.add(new Item(
                "Invite a almorzar a mi novio, por cumplir otro mes juntos.",
                R.mipmap.waiter,
                R.drawable.spaghetti,
                "12:15 pm"
        ));
        mItems.add(new Item(
                "Me desperte muy tarde hoy... muy tarde. Sale algo rápido.",
                R.mipmap.bread,
                "8:30 am"
        ));
        mItems.add(new Item());
        mItems.add(new Item("Domingo 16-Agosto-2015"));
```

```
        mItems.add(new Item(
                "Me encanta cenar en la calle. Cena con mi pareja.",
                R.mipmap.restaurant_pickup,
                R.drawable.noodles,
                "7:20 pm"
        ));
        mItems.add(new Item(
                "Parrilla con los amigos.",
                R.mipmap.sun,
                R.drawable.kebab,
                "3:00 pm"
        ));



        mItems.add(new Item());
        mItems.add(new Item("Sabado 15-Agosto-2015"));
        mItems.add(new Item(
                "Bar con los amigos, tomando unas cervezas.",
                R.mipmap.beer,
                "8:40 pm"
        ));
        mItems.add(new Item(
                "Adoro ir a McDonalds\'s.",
                R.mipmap.french_fries,
                R.drawable.hamburger,
                "7:15 pm"
        ));
        mItems.add(new Item(
                "Almuerzo con unos amigos.",
                R.mipmap.vegetarian_food,
                R.drawable.wrap,
                "12:40 pm"
        ));
        mItems.add(new Item());
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

15. Obteniendo como resultado el siguiente RecyclerView con dos ViewHolder diferentes (ver imagen a continuación)

Imagen 3. Resultado de la aplicación.