NAVIGATION DRAWER ESTRUCTURA

Navegación mediante menú lateral, muy conocido en las aplicaciones de Android, es decir, vamos a programar el menú que aparece al clickear las tres líneas de la parte superior izquierda de la pantalla.

Al pulsarlas, aparece un menú deslizando en el lateral izquierdo donde aparecerán las opciones de navegación de la aplicación.

Para esto necesitamos una Activity especial que permite trabajar con este tipo de navegación, que en la que se conoce como ***Navigation_DrawerActivity***

Paso1:

Analizar esta nueva Activity para ver cómo personalizarla para adoptarla a nuestras necesidades.

1. MainActivity implementa una interface *OnNavigationItemSelectedListener* que es la interfaz que permite gestionar eventos click sobre el menú lateral. (Ver la ejecución)
2. También trae el menú de opciones Toolbar que vimos en aplicaciones anteriores y el botón flotante.

Paso 2:

Analizar cómo está estructurado el menú lateral.

En onCreate tenemos el elemento DrawerLayout

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener((view) → {
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
    });

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
    drawer.setDrawerListener(toggle);
    toggle.syncState();

    NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);
}
```

que se encarga de gestionar el menú lateral. Veamos cómo se encentra estructurado gráficamente.

El menú lateral está contenido en un elemento dentro del activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout 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:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

Como vemos el contenedor es un DrawerLayout con:

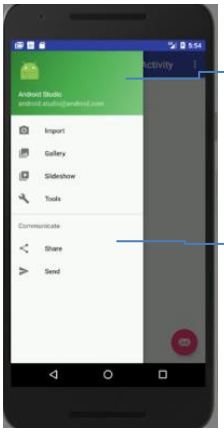
- Una etiqueta include: que tiene el contenido del activity
- Un elemento NavigationView: que contiene el menú lateral.

Por tanto, el contenido de la página estará en el include y en el NavigationView el menú lateral.

Por su parte el NavigationView se encarga, como gestor, de interconectar el contenido de la página con las opciones de este, y además de que aparezca y desaparezca dicho menú deslizante.

Paso3:

Veamos cómo se define cada parte del NavigationView. En principio veamos que carga dos elementos



Header: `app:headerLayout="@layout/nav_header_main"`

Cuerpo: **app:menu="@menu/activity_main_drawer"**

Paso3.1: Header o cabecera

Accedamos al layout nav_header_main.xml.

Este es un elementos LinearLayout con tres elementos:

- Un ImageView: con la imagen y su id
- Dos TextView: con su texto correspondiente

Como podéis ver se ha modificado la imagen y el texto para personalizarlo

Paso3.2: Cuerpo del menú

Menú con las opciones. Es más complejo que los vistos en el ejemplo anterior, ya que tenemos:

- Un grupo de opciones: están almacenadas en un grupo para que solo marque una y en cada acción se quede marcada al volver a mostrar el menú. Cada elemento tiene su id, su icono y su título
- Un elemento separado con un submenú de opciones

Paso4:

Volvamos al activity_main.xml y veamos el contenido de la página que dijimos estaba en el include, en el archivo app_bar_main.xml

Este es un layout muy similar a los vistos en el Basic Activity que tiene:

- Un Toolbar
- Un botón flotante

- En el centro el contenido de la página, almacenado este en el archivo content_main.xml

En el content_main.xml hemos dicho que estará el contenido que queremos que tenga la página.

Cuando usamos este tipo de menús, lo ideal es que carguemos en este fragmentos, por tanto, cuando pinchemos en una opción del menú lo que esperamos es que en el contenido de la página se cargue un fragmento concreto. Para ello vamos a cambiar el ConstraintLayout por un FrameLayout, y como ya hicimos anteriormente, no olvidar ponerle su identificados

```
android:id="@+id/contenedor"
```

Paso5:

Vamos a continuación a crear los fragmentos que se cargarán en función de la opción marcada. Los vamos a crear como Fragment Blank, y como vamos a hacerlo básico, desmarquemos las opciones por defecto.

Las hemos llamado ListadoAveriasFragment y ListadoTallerFragment, y en cada una de las vistas de estos cambiamos el texto que muestran.

Paso6:

Vamos a cargar los fragmentos según las opciones.

Primero vamos a cargar uno por defecto, por tanto, dentro del onCreate carguemos uno de ellos

```
getSupportFragmentManager()
        .beginTransaction()
        .add(R.id.contenedor,new ListadoAveriasFragment())
        .commit();
```

Para que se cambien en función de la opción usaremos un método que, al implementar nuestro MainActivity la interfaz *OnNavigationItemSelectedListener* nos obliga a implementar el método *onNavigationItemSelected*.

Este método recibe el elemento del menú seleccionado, saca su id y en función de este hace una opción y otra del if

Mejorando el código podemos crear un fragmento nulo y en función de la opción a la que acceda le asignaremos el fragmento a cargar. Al final del condicional, reemplazamos el fragmento, en caso de que se marcase alguna opción válida, por el correcto

```
Fragment f = null;

if (id == R.id.nav_camera) {
    f = new ListadoAveriasFragment();
} else if (id == R.id.nav_gallery) {
    f = new ListaTalleresFragment();
} else if (id == R.id.nav_slideshow) {

} else if (id == R.id.nav_manage) {

} else if (id == R.id.nav_share) {

} else if (id == R.id.nav_send) {

}

if(f!=null) {
    getSupportFragmentManager()
            .beginTransaction()
            .replace(R.id.contenedor,f)
            .commit();
}
```

Paso6:

Vamos a personalizar nuestra aplicación, tanto el header como el menú de opciones. En el onCreate tenemos el navigationView, y vamos a ver cómo acceder a sus elementos

Paso6.1: Header

Accedemos al fichero que contiene la cabecera, que como vimos es *nav_header_main.xml* y cambiamos los identificadores para hacerlos más personales

```
<ImageView
    android:id="@+id/imageViewAvatar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="16dp"
    app:srcCompat="@android:drawable/sym_def_app_icon" />

<TextView
    android:id="@+id/textViewNombre"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingTop="16dp"
    android:text="Android Studio"
    android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

<TextView
    android:id="@+id/textViewEmail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="android.studio@android.com" />
```

Ahora en el MainActivity, justo después de definir el navigationView podemos hacer los siguiente para personalizar el header del NavigationView Header:

1. Primero personalizamos el avatar. El 0 es porque podemos cargar más de una imagen, pero como solo tenemos una pues pondremos el 0

Nos descargamos un icono desde flaticon, como se vio en clases anteriores y lo cargamos creando un new→vector assert

2. Cargamos el icono

```
// Personalización del contenido del NavigationView header
ImageView ivAvatar = (ImageView) navigationView.getHeaderView( index: 0).findViewById(R.id.imageViewAvatar);
ivAvatar.setImageResource(R.drawable.ic_learning);

TextView nombre = (TextView) navigationView.getHeaderView( index: 0).findViewById(R.id.textViewNombre);
TextView email = (TextView) navigationView.getHeaderView( index: 0).findViewById(R.id.textViewEmail);
```

3. Cambiamos el texto de debajo de la imagen, poniendo a continuación del texto anterior el siguiente código

```
nombre.setText("Chari");
email.setText("ralcazar@iespuntadelverde.es");
```

   Estos pasas cambian el texto dinámicamente ya que se hace en tiempo de ejecución.

4. El color de fondo está definido en el LinearLayout, con el background. Si nos metemos dentro del color definido, que es un degradado. Podemos definir en el drawable, y podemos definir los colores en este fichero.

Paso6.2: Menú

Accedemos al fichero que contiene el menú, que como vimos es *activity_main_drawer.xml* y simplemente cambiamos las opciones que nos interesen y los iconos, que podemos seleccionar de la librería. También vamos a eliminar los que nos sobren

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_averias"
            android:icon="@drawable/ic_assignment_black_24dp"
            android:title="Averías" />
        <item
            android:id="@+id/nav_talleres"
            android:icon="@drawable/ic_directions_car_black_24dp"
            android:title="Talleres" />
    </group>
    <item android:title="Communicate">
        <menu>
            <item
                android:id="@+id/nav_share"
                android:icon="@drawable/ic_menu_share"
                android:title="Compartir" />
            <item
                android:id="@+id/nav_logout"
                android:icon="@drawable/ic_exit_to_app_black_24dp"
                android:title="Salir" />
        </menu>
    </item>
</menu>
```

Por último actualizamos en el método *onNavigationItemSelected* actualizamos los valores que hemos modificado en el fichero anterior

```java
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();
    Fragment f = null;
    if (id == R.id.nav_averias) {
        f = new ListadoAveriasFragment();
    } else if (id == R.id.nav_talleres) {
        f = new ListaTalleresFragment();
    } else if (id == R.id.nav_share) {

    } else if (id == R.id.nav_logout) {

    }
```