**Precio Del Euro**

Documentación

Contenidos

1. Información Funcional.
2. Información Tecnica.
3. Imagenes de la Aplicación.

## Información Funcional

El desarrollo de esta aplicación está basado en Android Studio, utiliza un Web Service del sitio http://dolarhoyinfo.com y recupera los valores del euro frente a otras divisas, el link a Google play:

<https://play.google.com/store/apps/details?id=com.betamobilemonkey.preciodeleuro>

Consta de una pantalla principal de cotizaciones, una calculadora y una pantalla de contacto, junto a un menú tipo sándwich y una splash screen.

Los países cuales consulta la información son Reino Unido, Usa, México, Argentina, Colombia, India, Brasil, Australia y China.

## Información Técnica

Los detalles técnicos se detallan a continuación

1. **Sdk utilizado:** compileSdkVersion 28
2. **Dependencias:**

**'com.android.support:appcompat-v7:28.0.0'  
'com.android.support.constraint:constraint-layout:1.1.3'  
'com.android.support:support-v4:28.0.0'  
'junit:junit:4.12'  
'com.android.support.test:runner:1.0.2'  
'com.android.support.test.espresso:espresso-core:3.0.2'  
'com.android.support:design:28.0.0'  
'com.android.support:cardview-v7:28.0.0'  
'com.google.firebase:firebase-ads:9.8.0'**

1. **Componentes:**

Activity Principal

MainActivity.java

MainActivity.xml (Basado en un drawerLayout)

**DrawerLayout**

El drawerLayout es un tipo de Layout (contenedor) donde se permite traer y cerrar diferentes vistas desde los bordes de la pantalla del teléfono. Generalmente se utiliza con un NavigationView dentro (como menú lateral desplegable). Ademas en este tipo de layout podemos incluir una toolbar y el espacio para renderizar fragments. Por ello nuestro MainActivity.xml es de este tipo.

**NavigationView**

Menu lateral deslizable donde se seleccionan diferentes ítems, una vez seleccionado se crea Fragment, que es donde renderizaremos y agregaremos lógica.

**Toolbar**

Este widget es el más utilizado en Android para mostrar una barra sobre la aplicación, aquí podremos ver los títulos de cada fragment que se renderiza, como también un icono conocido como hamburguesa, el cual se agrega de la siguiente manera:getSupportActionBar().setHomeAsUpIndicator(R.drawable.***ic\_home***);

El comando getSupportActionBar trae la barra, y mediante el set y la dirección de la imagen se agrega este icono.

**Splash Screen**

Es la pantalla inicial de la APP, mientras carga el inicio esta es mostrada, se realiza en el AndroidManifest.xml

<**activity android:name=".Activities.MainActivity"  
 android:theme="@style/SplashTheme"**>

A su vez, en el archive Styles.xml hay que crear este ítem con la imagen correspondiente y con el parent AppTheme.

**Admob**

La forma de mostrar el banner es la siguiente:

**mBottomBanner** = (AdView) view.findViewById(R.id.***av\_bottom\_banner***);  
**if** (**mBottomBanner** != **null**) {  
 AdRequest adRequest = **new** AdRequest.Builder().build();  
 **mBottomBanner**.loadAd(adRequest);  
}

Esto busca el banner y lo carga, el bottonBanner es de la clase Adview, y se referencia en el FragmentCotizaciones, y esto se implementa via la dependencia firebase-ads:9.8.0.

**AdaptadorCotizaciones**

Este clase java creada por nosotros hereda de BaseAdapter por lo cual hereda métodos como getCount donde podremos saber cuántos ítems tiene, en que ítem estamos entre otros. La función de esto es poder mostrar en una sola línea diferentes tipos de datos.

El constructor recibe un layout, un context y un list con las cotizaciones, y por otro lado al pedir un ítem en particular se infla el componente *list\_itemcard\_cotizaciones.xml, se* setea la imagen, la descripción y la cotización en particular de uno.

**Consulta Json**

Utilizaremos dos clases RestHelper (esta realiza la consulta REST) y GetCotizacionesWS (la cual procesa los datos recibidos), ya en el Fragment CotizacionesFragment se asignan los valores recibidos si la transacción fue correcta.

## Imágenes de la aplicación









