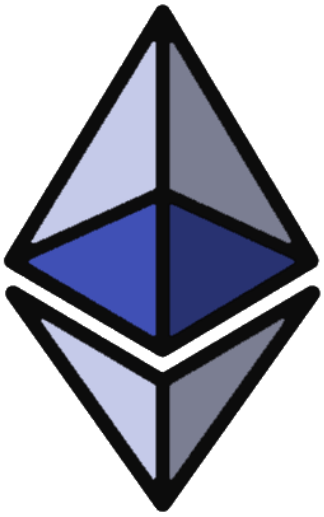
PROYECTO DE LA ASIGNATURA DE AA. TT.

Aplicaciones Telemáticas.

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación.

ETSIT-UPV



CriptoValores

Precios en tiempo real para Criptomonedas

Equipo de desarrollo:

Jordi Moros Badenes
Diego Peinado Martínez
Alejandro Martínez-Carbonell Marín

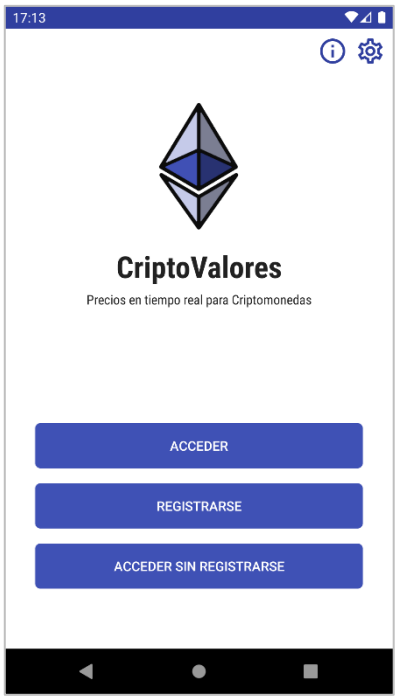
Índice

1. Acerca de la aplicación

La propuesta de diseñar una App desde cero nos ha parecido muy interesante y hemos querido ir más allá. Nuestra propuesta consiste en una aplicación de seguimiento de criptomonedas, compatible con hasta 100 criptomonedas. Recibirás información en tiempo real sobre Capitalización de mercado, posicionamiento, precios, volumen y fluctuaciones del precio en la última hora, últimas 24 horas y últimos 7 días. Además, podrás encontrar más datos curiosos sobre tus criptomonedas seleccionándolas y dispondrás en todo momento de la fecha de última actualización de los datos. Para realizar todo esto, hemos obtenido los datos de una API. Se detallará el proceso a lo largo de la memoria.

2. Actividades

2.1.MainActivity



Junto al texto se muestra la página de inicio de nuestra aplicación.

En primer lugar, se observan dos iconos en la parte superior derecha, que contienen una actividad de información acerca de la aplicación y otro icono que nos lleva a una actividad de preferencias de usuario.

A continuación, se disponen tres botones los cuales se comunican con la actividad a través de sus respectivos métodos onClick. Será así también para el resto de las actividades que contengan botones en la aplicación.

Los dos primeros botones sirven pare acceder y registrarse en la aplicación, respectivamente. El tercer botón permite acceder sin registro previo, privando al usuario de algunas funcionalidades.

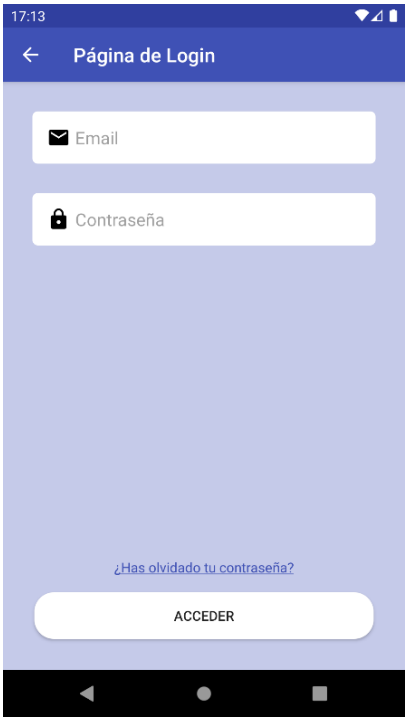
2.2.LoginActivity

La actividad de Login cumple una función doble, y es que ha sido desarrollada de forma que la misma actividad nos sirva para la función de acceder y de registrarse, dependiendo del botón que ha sido pulsado.

Esto lo logramos a través del uso del método `putExtra()` de la clase `Intent`, haciéndolo de la siguiente manera: `i.putExtra ( " register / login ",0)` De forma que en la actividad de Login vamos a recuperar el valor y, dependiendo de si es 0 o 1, mostraremos pantalla de Login o Registrarse.

Esto lo haremos ocultando algunos elementos innecesarios como es el caso del `EditText` del nombre utilizando el método `setVisibility(View.GONE)` de la correspondiente clase. Ocultaremos también la opción de recuperar contraseña en el menú de registrarse.

Esta clase dispone además de botón de atrás que ha sido programado utilizando la llamada `getSupportActionBar().setDisplayHomeAsUpEnabled (true)` y asignando que su actividad padre es la MainActivity en el `AndroidManifest.xml`.



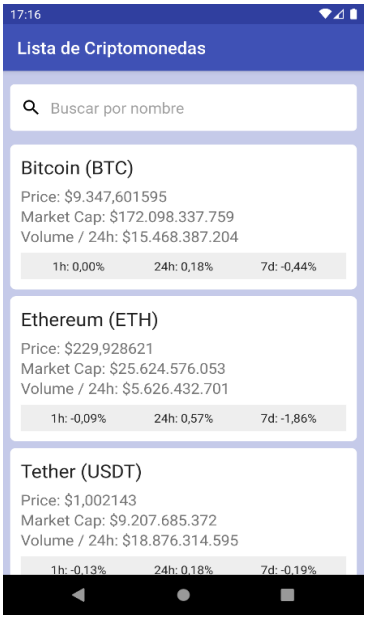
2.3. AppActivity

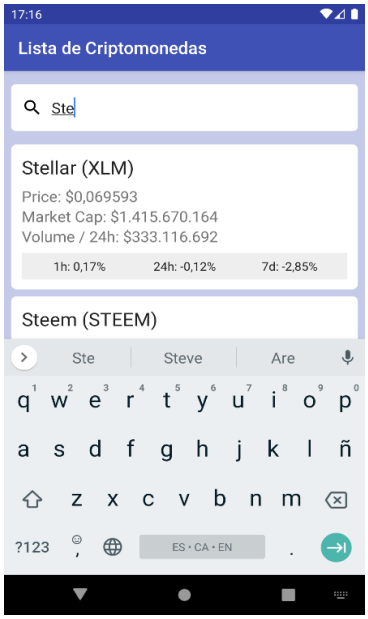
Esta actividad es la actividad "principal" de la aplicación. En ella encontramos un `RecyclerView` que nos muestra las diferentes criptomonedas y algunos valores obtenidos de la API. Este `RecyclerView` crea la lista a través del adaptador encargado de inflar las vistas correspondientes al layout xml creado para tal fin.

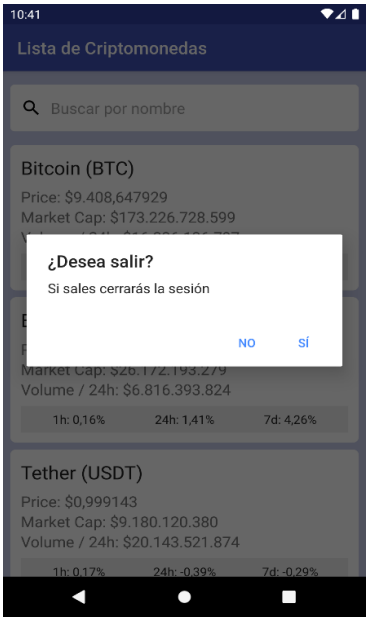
Se dispone de un buscador para filtrar una búsqueda concreta. Al pulsar en una criptomoneda de la lista, se accede a otra actividad que nos muestra más información acerca de esta.

Además, en esta actividad se ha sobreescrito el método `onBackPressed()` de forma que al intentar volver a la MainActivity, se muestra un `alertDialog` informando al usuario de si vuelve atrás tendrá que cerrar su sesión.

Si no hay conexión a internet, la aplicación de muestra un mensaje al acceder a la AppActivity. Obviamente si intentas acceder o registrarte sin conexión, te saltará un mensaje de error.



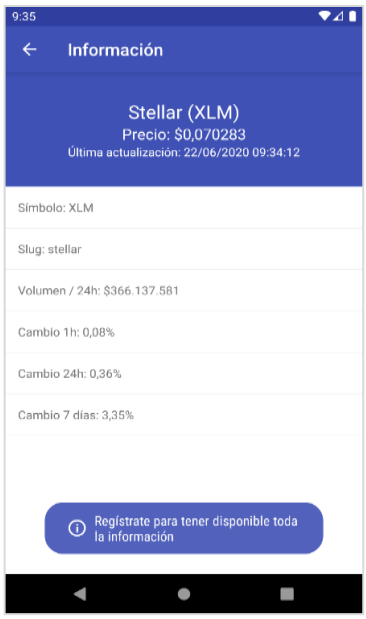




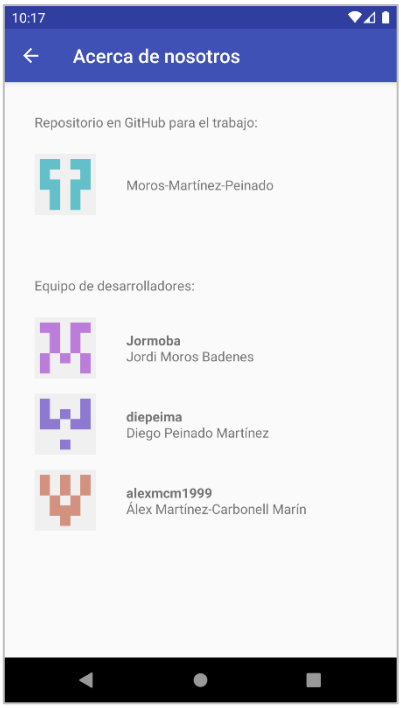
2.4.CoinPageActivity

En esta actividad vamos a encontrar absolutamente toda la información en relación con la criptomoneda. Desde su símbolo y nomenclatura hasta el valor que tiene en el mercado y la variación de su valor en los últimos días. Esta información la proporciona la API y por ello es limitada.

Además, siguiendo la práctica que se utilizó en la actividad de Login, hemos ocultado información a aquellos usuarios que no se han registrado, y además les mostramos un mensaje en pantalla para que sepan que el registro es valioso si quieren obtener más datos de la criptomoneda en cuestión. En esta actividad hemos utilizado un `ScrollView` para mostrar la información y la utilidad `GraphView` así como `LineGraphSeries` y algunos de sus métodos para mostrar unos gráficos en pantalla para el usuario registrado.



2.5. InfoActivity



Por último, hemos creado una actividad de información en la que se puede observar nuestro equipo de desarrolladores, así como el repositorio de GitHub que hemos utilizado para el control de versiones.

3. Clases y funcionalidades

3.1. Retrofit

En nuestra aplicación hemos hecho uso de Retrofit por dos motivos. Para conseguir un cliente HTTP seguro para Android y Java, y porque hace sencillo conectar a un servicio web REST traduciendo la API a interfaces Java. Mediante una librería HTTP podemos consumir de manera sencilla datos JSON o XML, que después son analizados en objetos Java. Además, sabemos que Retrofit emplea OkHttp para manejar peticiones de red y que no dispone de un convertidor JSON integrado para convertir objetos JSON a Java, en cambio dispone con soporte para librerías de convertidor JSON de las cuales hemos implementado la de Gson: *com.squareup.retrofit:converter-gson*.

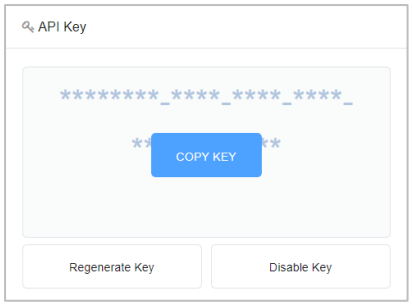
En nuestro build.gradle hemos declarado las siguientes dependencias que incluyen la librería Retrofit y tambien la librería Gson de Google para convertir JSON a POJO (Plan Old Java Objects) así como la integración de Gson de Retrofit.

```
implementation 'com.squareup.retrofit2:retrofit:2.7.1'
implementation 'com.squareup.retrofit2:converter-gson:2.7.1'
implementation 'com.google.code.gson:gson:2.8.6'
```

3.2. APIClient y APIInterface

La clase APIClient se crea con el fin de emitir peticiones de una red a una API RESTFUL con Retrofit, la cual necesita crear una instancia usando la clase Retrofit Builder y configurarla con una URL base. En cuanto a la interfaz APIInterface, esta contiene métodos que vamos a usar para ejecutar peticiones HTTP tales como @GET.

Los datos y la información de las criptomonedas se obtienen de la API CoinMarketCap mediante solicitudes HTTP que tenemos que realizar desde nuestra app. Todas las solicitudes deben validarse con una clave API. Esta clave API se obtiene tras el registro en la página web y aparece de la siguiente forma:

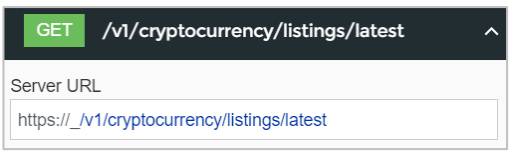


Con esta clave API podemos emplear cualquier lenguaje de programación del lado del servidor que pueda realizar solicitudes HTTP para apuntar a la API de CoinMarketCap, en nuestro caso hemos empleado Java y todas las solicitudes deben de estar dirigidas al dominio *https://pro-api.coinmarketcap.com*. Hemos realizado este paso mediante la clase APIClient como se observa a continuación.

```
public class APIClient {

    private static Retrofit retrofit = null;

    static Retrofit getClient() {
        HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();
        interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
        OkHttpClient client = new OkHttpClient.Builder().addInterceptor(interceptor).build();

        retrofit = new Retrofit.Builder()
                .baseUrl("https://pro-api.coinmarketcap.com")
                .addConverterFactory(GsonConverterFactory.create())
                .client(client)
                .build();

        return retrofit;
    }
}
```

Nuestra clave API la hemos suministrado en las llamadas de REST API a través de un encabezado personalizado llamado X-CMC_PRO_API_ KEY.

La API dispone de varios Endpoints de los cuales hemos empleado el de listings/latest y lo hemos añadido copiando el link en azul.




```
interface APIInterface {

    @Headers("X-CMC_PRO_API_KEY: 72311af0-4b64-4272-986e-53e9399ef53f")
    @GET("/v1/cryptocurrency/listings/latest?")
    Call<CryptoList> doGetUserList(@Query("limit") String page);
}
```

3.3.Firebase

Para el registro de usuario hemos utilizado la interfaz de usuario de Google, Firebase. Hemos añadido sus dependencias al gradle de nuestro proyecto para poder hacer uso de sus clases y funcionalidades.

```
implementation 'com.google.firebase:firebase-auth:19.3.1'
implementation 'com.google.firebase:firebase-database:19.3.1'
```

Para comunicar firebase con nuestra aplicación, además hemos tenido que registrarnos en la página web e introducir la clave de depuración de nuestra app. Así, ya está todo listo para funcionar.

Para el registro y login de usuario hemos utilizado los métodos de Firebase siguiendo las indicaciones de la documentación que nos proporciona Google. Hemos añadido algún comprobador básico como el de que los campos no pueden estar vacíos y la contraseña debe tener más de 6 caracteres.

4. Otros aspectos

4.1. Recuperar contraseña

Gracias a Firebase, hemos podido añadir la funcionalidad de poder recuperar nuestra contraseña en caso de perderla. Hemos creado el diálogo que se muestra a la derecha. Si introducimos un mail válido, nos mandará el correo de recuperación. El texto de mail se puede modificar desde la consola de Firebase.

El diálogo ha sido creado de la misma forma que la explicada para el cierre de sesión, además aquí hemos añadido un EditText a la vista desde la programación en Java de forma que se pueda recuperar el texto al pulsar sobre el botón de enviar.

4.2. Toasty

Esta es una librería de uso gratuita muy extendida en el mundo de la programación en Android Studio. Su función es la de sustituir los mensajes Toast vanilla de Android por unos más representativos, con iconos material design. Sus métodos permiten mostrar un Toasty personalizado y hay de varios tipos (success, error, warning, info, etc). Hemos decidido utilizar la librería para darle un toque atractivo a los mensajes. La hemos añadido también a las dependencias y a los repositorios para poder hacer uso de ella.

```
implementation 'com.github.GrenderG:Toasty:1.4.2'
```

```
maven { url "https://jitpack.io" }
```

4.3. App multilenguaje

Cabe destacar que nuestra aplicación es multilenguaje, y es que, si cambias el idioma de tu smartphone, la interfaz de usuario se va a adaptar. Eso lo hemos conseguido duplicando el archivo strings.xml y ha sido traducido para los idiomas de castellano, valenciano e inglés. Según sea el idioma de tu Android, actuará un fichero u otro, haciendo que la aplicación sea multilenguaje.

4.4. Integración con Git

La forma de trabajo con el sistema de versiones ha sido la de clonar el repositorio remoto mediante el IDE de Android Studio, una vez clonado hemos creado una rama de trabajo individual donde probar las nuevas ideas y una vez fuesen funcionales realizábamos un merge fusionando la rama de Testeo con la de Master para posteriormente realizar un commit donde se describen los cambios realizados y finalmente realizar un push para actualizar el repositorio remoto.

5. Conclusiones

Este proyecto se nos ha presentado como una oportunidad para llevar a cabo una App de propuesta libre en la cual poder volcar todo lo aprendido en esta asignatura, la cual no podíamos desaprovechar. Consideramos que la programación es muy interesante y gracias al trabajo hemos tenido un contacto mas real con la programación en sí, pero sobre todo con la manera de funcionar y de trabajar empleando un repositorio en GitHub con más participantes. Hemos puesto bastante tiempo en desarrollar la aplicación y pensamos que ha merecido la pena, mas allá de haber realizado la entrega es seguro que continuaremos desarrollándola. Queríamos agradecer a los profesores su atención y por resolver cualquier tipo de duda con claridad a lo largo de la asignatura.