

# IPDownApp

a RED-TEAM tool



DESARROLLADA POR **ÁLVARO CAÑADA LORENZO** Y **ENRIQUE SÁNCHEZ VICENTE**.

# Servicios de IPDownApp:

- ▶ IPDownApp ofrece:
  - **Información** sobre tu **IP pública y privada**.
  - **Información geográfica** sobre tu **IP pública** y sobre cualquier dirección que se desee.
  - Posibilidad de **realizar un ataque DDOS** a la dirección que quiera el usuario. (Solamente realizar **con fines laborales o educativos!**)

## API utilizada:

- ▶ La API usada es: <https://ipgeolocation.io>
  - Gracias a su base de datos de información sobre IP's, obtenemos valores de nuestra propia IP pública y de las direcciones que nosotros queramos.

# Uso de la librería Retrofit 2:

- ▶ Para esta App usaremos la librería de Retrofit 2, por lo que debemos realizar una configuración previa:
  - Primero incluimos las implementaciones de librerías en el fichero **build.gradle**, en el apartado **dependencies**:

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'  
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'  
implementation 'com.google.code.gson:gson:2.8.9'
```

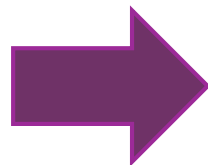
# Uso de la librería Retrofit 2:

- ▶ Para esta App usaremos la librería de Retrofit 2, por lo que debemos realizar una configuración previa:

- Segundo, creamos una clase API, en la cual vamos a configurar la conexión con el servidor e incluimos en nuestro código la constante BASE URL

```
BASE_URL = "https://api.ipgeolocation.io/";
```

- En la **clase API**, hacemos la llamada con Retrofit de la siguiente manera:



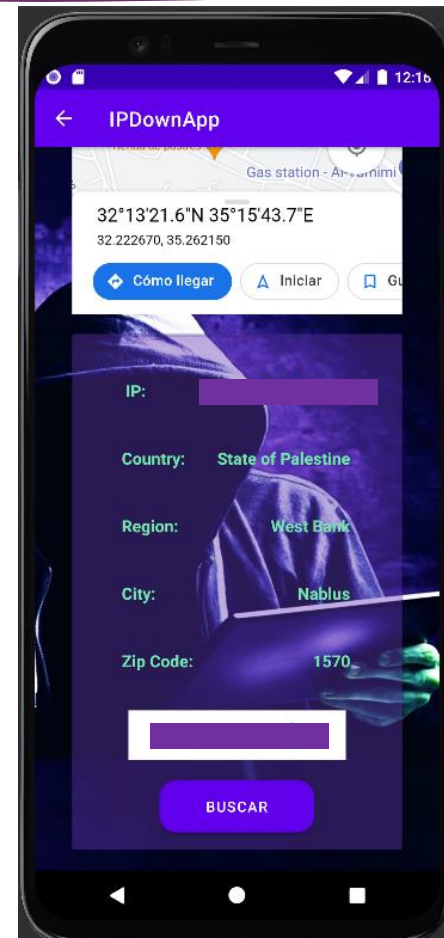
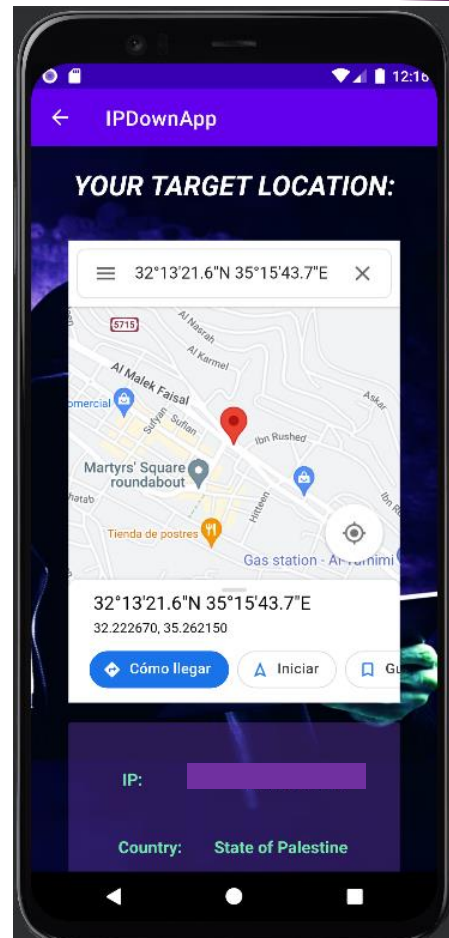
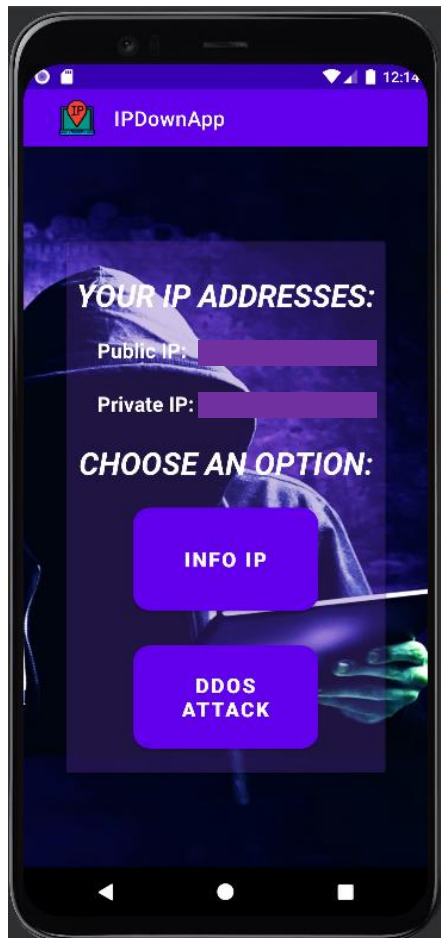
# Llamada en la clase API:

```
public static Retrofit getApi(String baseUrl){  
  
    Retrofit retrofit = new Retrofit.Builder()  
        .baseUrl(baseUrl)  
        .addConverterFactory(GsonConverterFactory.create())  
        .build();  
  
    return retrofit;  
  
}
```

En la interfaz para el servicio de la API  
añadimos:

```
@GET("ipgeo")  
Call<IPInfo> infoIp(@Query("apiKey") String key, @Query("ip") String ip);  
  
@GET("ipgeo")  
Call<IPInfo> infoIpPropia(@Query("apiKey") String key);
```

# Interfaces de usuario:

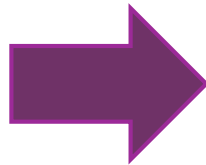




# Para futuras implementaciones:

- ▶ Usaremos un **proxy-list** para variar de servidor y dirección IP y así evitar ser detectados por los proveedores de servicio.
- ▶ Usaremos los **proxy-list gratuitos**: <https://free-proxy-list.net/>  
(Estos se comprueban y actualizan cada 10 minutos).

Ejemplo:



# Ejemplo de conexión a través de proxy en Java:

```
URLConnection.openConnection():
```

```
URL weburl = new URL (URL_STRING);
```

```
Proxy webProxy = nuevo Proxy (Proxy.Type.HTTP, new InetSocketAddress ("127.0.0.1", 8080));
```

```
HttpURLConnection webProxyConnection = (HttpURLConnection) weburl.openConnection(webProxy);
```

**(Deberíamos variar la dirección IP y puerto de nuestro InetSocketAddress a través del proxy list).**

# Repositorio de GitHub:

► <https://github.com/AlvaroCLorenzo/IPDownApp>

