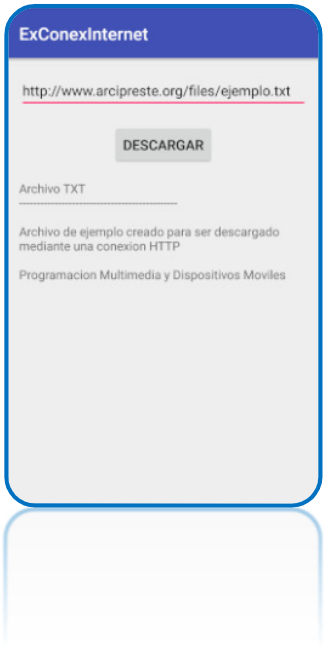
UNIDAD 4: Conexión a Internet

Para comprobar como se realiza el proceso de conexión a Internet para obtener un archivo (html, txt, etc), vamos a crear una aplicación que se conecte a un servidor web (HTTP/HTTPS), cuya URL será establecida por el usuario, y a continuación se mostrará en pantalla el contenido del archivo descargado del servidor web.

Además para evitar que el tiempo de espera pueda hacer parecer que la aplicación se ha colgado, vamos a utilizar una tarea asíncrona (AsyncTask) para que la descarga se realice en segundo plano.



SOLICITUD DE PERMISOS

Nuestra aplicación va a necesitar acceder al estado de la red y a Internet, y esto requiere que solicitemos los permisos adecuados en nuestro archivo de "Manifiesto".

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

El archivo de "Manifiesto" sería similar al siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.oscar.exconexinternet">

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

</manifest>
```

CREACIÓN DE LA INTERFAZ

Vamos a crear una interfaz muy sencilla con una única pantalla en la que añadiremos una caja de texto (*editURL*) para que el usuario introduzca la URL del archivo a descargar (tipo *textUri*), un botón (*botDescarga*) para comenzar la descarga, y una etiqueta (*textURL*) que ocupe el resto de la pantalla para mostrar el contenido del archivo descargado.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.oscar.exconexinternet.MainActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <EditText
        android:id="@+id/editURL"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="URL"
        android:tooltipText="URL"
        android:inputType="textUri"
        android:text="http://"
        android:textSize="16sp" />

    <Button
        android:id="@+id/botDescarga"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Descargar"
        android:textSize="16sp"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"/>

    <TextView
        android:id="@+id/textURL"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="16dp" />

</LinearLayout>
```

CLASE DESCARGAWEB

Como se ha explicado, queremos evitar la sensación de bloqueo de la aplicación mientras se realiza la descarga del archivo, realizando el proceso en segundo plano por medio de una tarea asíncrona. Para ello vamos a implementar una clase interna (*DescargaWeb*) que extiende a *AsyncTask* dentro de nuestro *MainActivity*, y que se lanzará mediante el método *execute()*.

Una tarea asíncrona (*AsyncTask*) se define mediante 3 tipos genéricos:

- ***Param***: define el tipo de parámetro que recibirá el método *execute* y que se pasará al método *doInBackground* de la tarea. En nuestro caso será la URL, y por tanto de tipo *String*.

- ***Progress***: tipo de unidades que se utilizará para mostrar el progreso de la tarea en segundo plano mediante el método *ProgressUpdate*. Como no vamos a mostrar el progreso utilizaremos *Void*.

- ***Result***: resultado que genera la tarea y que será enviado a *onPostExecute*. En nuestro caso es el contenido del archivo, y por tanto de tipo *String*.

Por tanto, la declaración de nuestra clase sería:

```
private class DescargaWeb extends AsyncTask <String, Void, String> {
…
}
```

Dado que la clase DescargaWeb extiende a *AsyncTask*, tendrá que implementar obligatoriamente el método *doInBackground*:

- **doInBackground**: recibe un array de cadenas (en este caso con un solo elemento en la posición 0 que será la URL), y realiza la tarea encomendada en segundo plano.
  El proceso a realizar en segundo plano será la descarga del archivo indicado por la URL y la conversión del archivo descargado a *String*, para ello se ha creado un método llamado "descargaUrl".

```
@Override
protected String doInBackground(String... urls) {
    try {
        return descargaUrl (urls[0]);
    } catch (IOException e) {
        return "URL inexistente";
    }
}
```

- **descargaUrl**: recibe un *String* con la URL del archivo a descargar y se encarga de establecer la conexión, recoger el archivo y procesarlo para convertirlo a String.

  Para crear la conexión utilizaremos HttpURLConnection (HTTP) o HttpsURLConnection (HTTPS) en función de que el *String* con la URL recibido empiece o no por "https":

```
URL url = new URL (myurl);
HttpURLConnection con;
if (myurl.startsWith("https")) {
    con = (HttpsURLConnection)url.openConnection();
} else {
    con = (HttpURLConnection)url.openConnection();
}
```

  A continuación, se establecen los parámetros de la conexión:

```
con.setReadTimeout(10000);
con.setConnectTimeout(15000);
con.setRequestMethod("GET");
con.setDoInput(true);
```

  Descargamos los datos:

```
con.connect();
int response = con.getResponseCode();
InputStream is = con.getInputStream();
```

  Convertimos los datos obtenidos a *String*, utilizando un *ByteArrayOutputStream* intermedio:

```
ByteArrayOutputStream os = new ByteArrayOutputStream();
int i;
while ( (i=is.read()) != -1) {
    os.write(i);
}
String result = os.toString("iso-8859-1");
return result;
```

  Y por último, cerramos el *InputStream*:

```
finally {
    if (is != null) {
        is.close();
    }
}
```

El método completo quedaría así:

```
private String descargaUrl(String myurl) throws IOException {
    InputStream is = null;
    try {
        // Preparamos la conexión
        URL url = new URL (myurl);
        HttpURLConnection con;
        if (myurl.startsWith("https")) {
            con = (HttpsURLConnection)url.openConnection();
        } else {
            con = (HttpURLConnection)url.openConnection();
        }
        con.setReadTimeout(10000);      // milisegundos
        con.setConnectTimeout(15000);   // milisegundos
        con.setRequestMethod("GET");
        con.setDoInput(true);

        // Descargamos los datos
        con.connect();
        int response = con.getResponseCode();
        is = con.getInputStream();

        // Convertimos los datos obtenidos (InputStream) a String
        ByteArrayOutputStream os = new ByteArrayOutputStream();
        int i;
        while ( (i=is.read()) != -1) {
            os.write(i);
        }

        // Convertimos al codigo de caracteres deseado
        // antes de devolverlos
        // String result = os.toString("UTF-8");
        String result = os.toString("iso-8859-1");
        return result;

    } catch (IOException e) {
        return "Error de lectura";
    }

    finally {
        // Cerramos el InputStream
        if (is != null) {
            is.close();
        }
    }
}
```

- ***onPostExecute***: recibe un *String* con el archivo descargado y lo muestra en la etiqueta textURL.

```
@Override
protected void onPostExecute (String result) {
    textURL.setText(result);
}
```

CLASE MAINACTIVITY

En la clase *MainActivity* declararemos dos atributos de manera que podemos acceder a ellos desde cualquier método y también desde la clase interna DescargaWeb: un *TextView* para recoger la referencia a la etiqueta en la que se mostrará el archivo y un EditText para hacer lo propio con la URL introducida por el usuario:

```
public class MainActivity extends AppCompatActivity {

    protected TextView textURL;
    protected EditText editURL;
    …
}
```

El único método de la clase será *onCreate*, que se encargará de establecer la vista de la actividad (activity_main.xml), obtener las referencias a los tres elementos de la vista: textURL, editURL y botDescarga.

En textURL, como su contenido (archivo descargado) puede exceder el tamaño asignado se establece un *scroll* mediante el método *setMovementMethod*:

```
textURL = (TextView)findViewById(R.id.textURL);
textURL.setMovementMethod(new ScrollingMovementMethod());
```

En editURL se ha establecido como contenido por defecto "http://" para que el usuario no tenga que introducirlo, así que llevaremos el cursor al final de dicho texto:

```
editURL = (EditText)findViewById(R.id.editURL);
editURL.setSelection(editURL.getText().length());
```

Por último, al botón botDescarga le añadimos un *Listener* y un método onClick que se encargará de recoger el contenido de editURL, comprobar que tiene el formato correcto y el estado de la conexión, y si todo es correcto llamar al método *execute* de nuestra clase DescargaWeb.

En caso de que la dirección introducida no tuviera el formato adecuado o de que no dispusiéramos de conexión a Internet se generaría una alerta para indicarlo.

```
Button botDescarga = (Button)findViewById(R.id.botDescarga);
botDescarga.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    String url = editURL.getText().toString().trim();
    if (!url.equals("") && URLUtil.isValidUrl(url)) {
      ConnectivityManager conMgr =
       (ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
      NetworkInfo netInfo = conMgr.getActiveNetworkInfo();
      if (netInfo != null && netInfo.isConnected()) {
        new DescargaWeb().execute(url);
      } else {
        // Alerta: no se puede establecer la conexión
        int duracion = Toast.LENGTH_SHORT;
        String mensa = "No se ha podido establecer la conexión";
        Toast alerta = Toast.makeText(MainActivity.this,mensa,duracion);
        alerta.show();
      }
    } else {
      // Alerta: falta URL o formato incorrecto
      int duracion = Toast.LENGTH_SHORT;
      String mensa = "URL incorrecta";
      Toast alerta = Toast.makeText(MainActivity.this,mensa,duracion);
      alerta.show();
    }
  }
});
```

La clase *MainActivity* quedaría así:

```
public class MainActivity extends AppCompatActivity {

  protected TextView textURL;
  protected EditText editURL;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Obtenemos la referencia al TextView donde se muestra el resultado
    // y establecemos scroll
    textURL = (TextView)findViewById(R.id.textURL);
    textURL.setMovementMethod(new ScrollingMovementMethod());

    // Obtenemos la referencia al EditText donde se recoge la URL
    // y nos situamos al final
    editURL = (EditText)findViewById(R.id.editURL);
    editURL.setSelection(editURL.getText().length());

    // Obtenemos la referencia al Botón Descargar e implementamos su acción
    Button botDescarga = (Button)findViewById(R.id.botDescarga);
    botDescarga.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        String url = editURL.getText().toString().trim();
        if (!url.equals("") && URLUtil.isValidUrl(url)) {
          ConnectivityManager conMgr =
           (ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
          NetworkInfo netInfo = conMgr.getActiveNetworkInfo();
          if (netInfo != null && netInfo.isConnected()) {
            new DescargaWeb().execute(url);
          } else {
            // Alerta: no se puede establecer la conexión
            int duracion = Toast.LENGTH_SHORT;
            String mensa = "No se ha podido establecer la conexión";
            Toast alerta = Toast.makeText(MainActivity.this,mensa,duracion);
            alerta.show();
          }
        } else {
          // Alerta: falta URL o formato incorrecto
          int duracion = Toast.LENGTH_SHORT;
          String mensa = "URL incorrecta";
          Toast alerta = Toast.makeText(MainActivity.this,mensa,duracion);
          alerta.show();
        }
      }
    });
  }

  private class DescargaWeb extends AsyncTask <String, Void, String> {
  …
  }

}
```

PRUEBAS

Para probar la aplicación introduce cualquier URL válida y verás como se muestra su contenido (HTML), por ejemplo:

http://www.google.es

También puedes probar con alguna ruta que utilice un servidor seguro (HTTPS), como:

https://www.google.es

Por último, puedes probar cómo se descarga un archivo txt utilizando la siguiente ruta:

http://www.arcipreste.org/files/ejemplo.txt