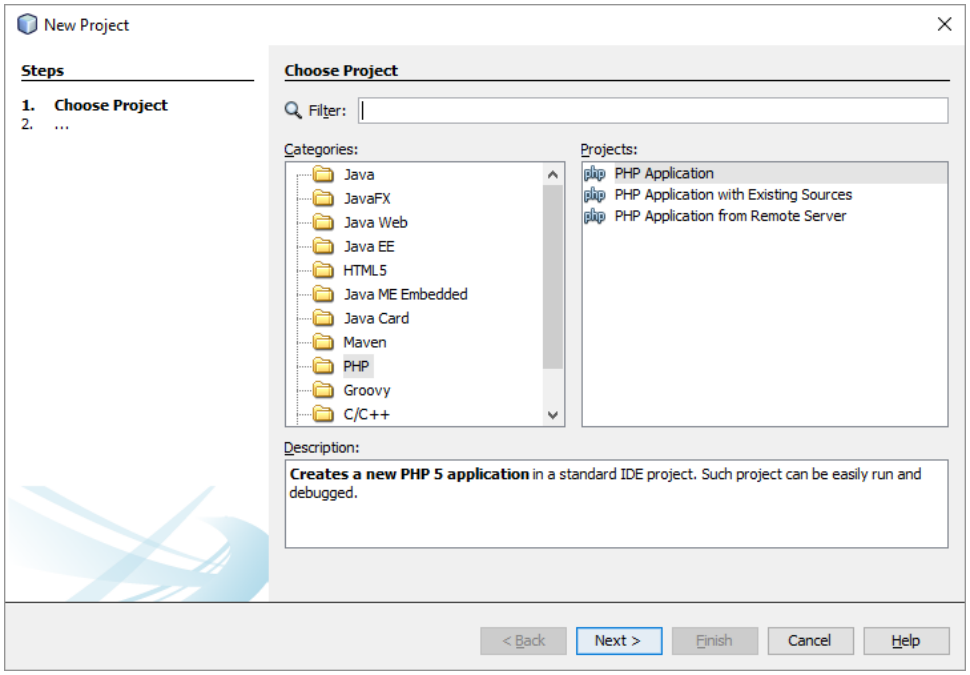
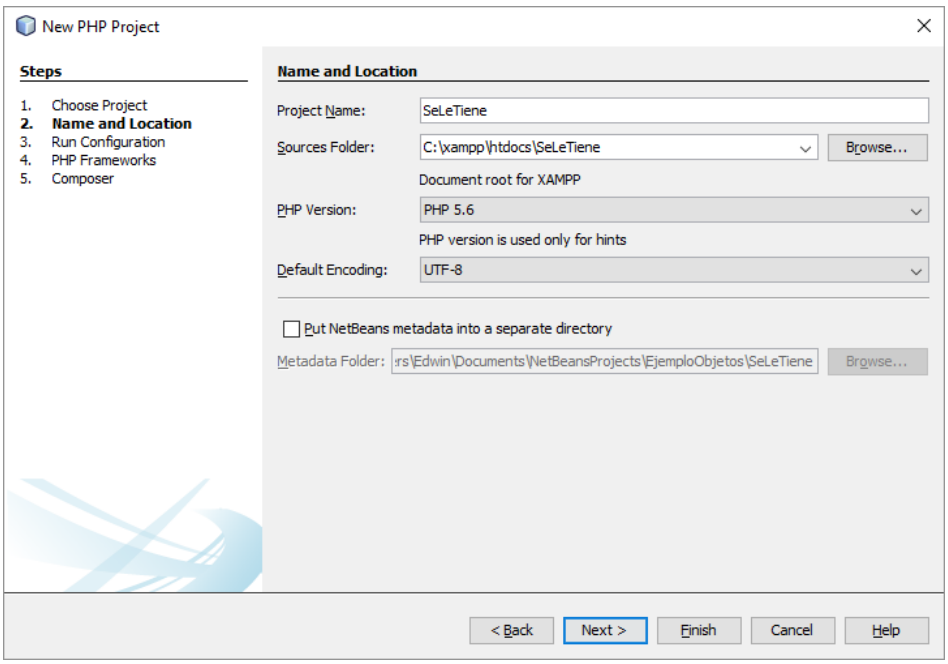
Contenido

Creación de Proyecto php en Netbeans

Crear un nuevo proyecto en Netbeans -> PHP -> PHP Application



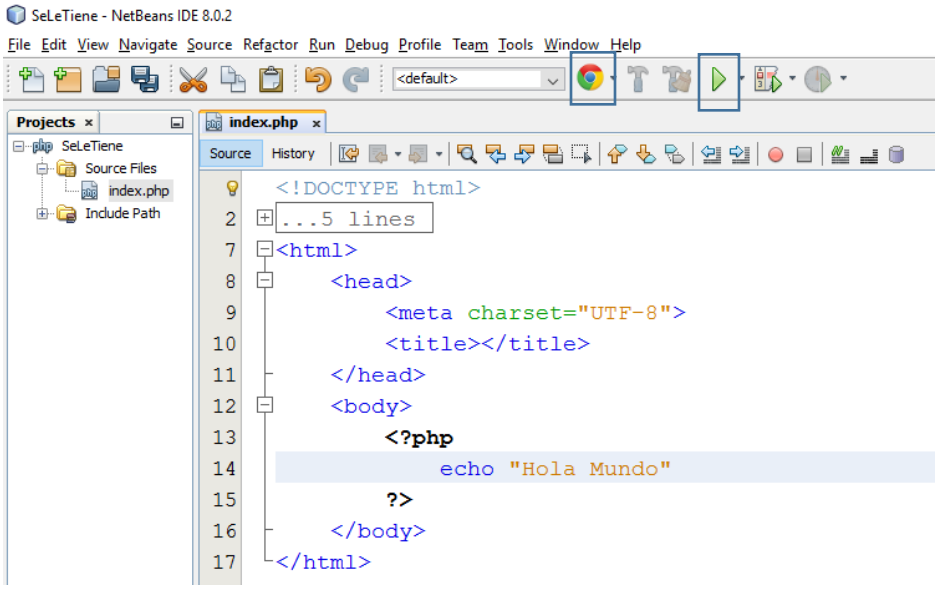
El nombre del proyecto es SeLeTiene, verificar que el campo **Sources Folder: C:\xampp\htdocs\SeLeTiene**, click en Finish.



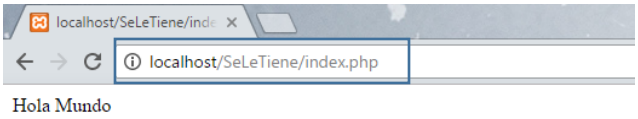
WebServices con PHP

index.php – Script de prueba

Se abrirá el archivo index.php, vamos a imprimir un mensaje de bienvenida en el navegador dentro de las llaves <?php y ?>, cambiar el navegador por el de su predilección y luego dar click en Run

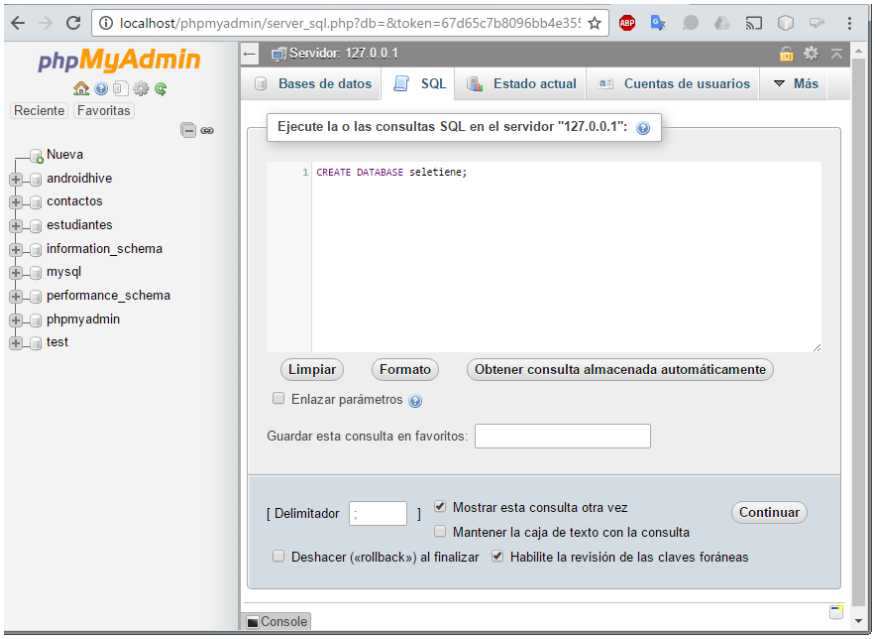


Se abre el navegador con el mensaje Hola Mundo, comprobar que si se digita directamente la url **http://localhost/SeLeTiene/index.php**

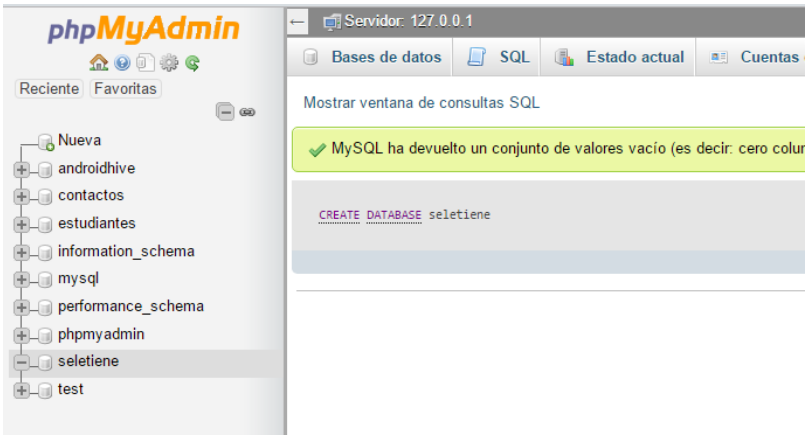


Vamos a crear la base de datos que necesitamos, para esto entramos a phpMyAdmin damos click en **"Servidor: 127.0.0.1"** y luego en SQL y digitamos la sentencia para crear la base de datos y damos click en Continuar

```
CREATE DATABASE seletiene;
```

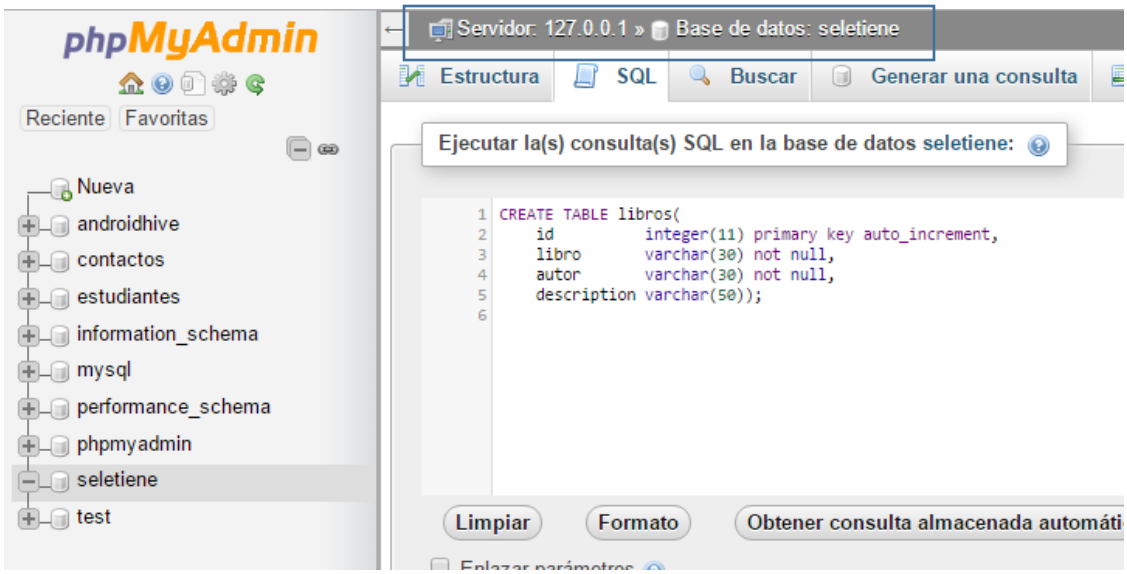


En el panel izquierdo se puede visualizar la nueva base de datos



Después de crear la base de datos procedemos a crear la tabla, para esto seleccionamos la base de datos creada y luego damos click nuevamente en **SQL**, entramos la sentencia SQL y click en **Continuar**

```
CREATE TABLE libros(
      id                  integer(11) primary key auto_increment,
      libro               varchar(30) not null,
      autor               varchar(30) not null,
      description varchar(50));
```



db_connect.php

Creamos una clase en php para conectar a la base de datos, el objetivo es abrir la conexión cuando se necesite y cerrarla cuando sea necesario, el archivo es **db_connect.php** que contiene los datos para la conexión.

```
<?php

/*
 * Variables de conexión
 */
define('DB_USER', "root"); // usuario
define('DB_PASSWORD', "123456"); // db password
define('DB_DATABASE', "seletiene"); // nombre de la base de datos
define('DB_SERVER', "localhost"); // nombre del servidor
```

```
//Connecting to Database
$con = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE) or
die('Unable to Connect');
?>
```

db_connect.php

Source History

```
<?php

/*
 * Variables de conexión
 */
define('DB_USER', "root"); // usuario
define('DB_PASSWORD', "123456"); // db password
define('DB_DATABASE', "seletiene"); // nombre de la base de datos
define('DB_SERVER', "localhost"); // nombre del servidor

//Connecting to Database
$con = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE) or
        die('Unable to Connect');
?>
```

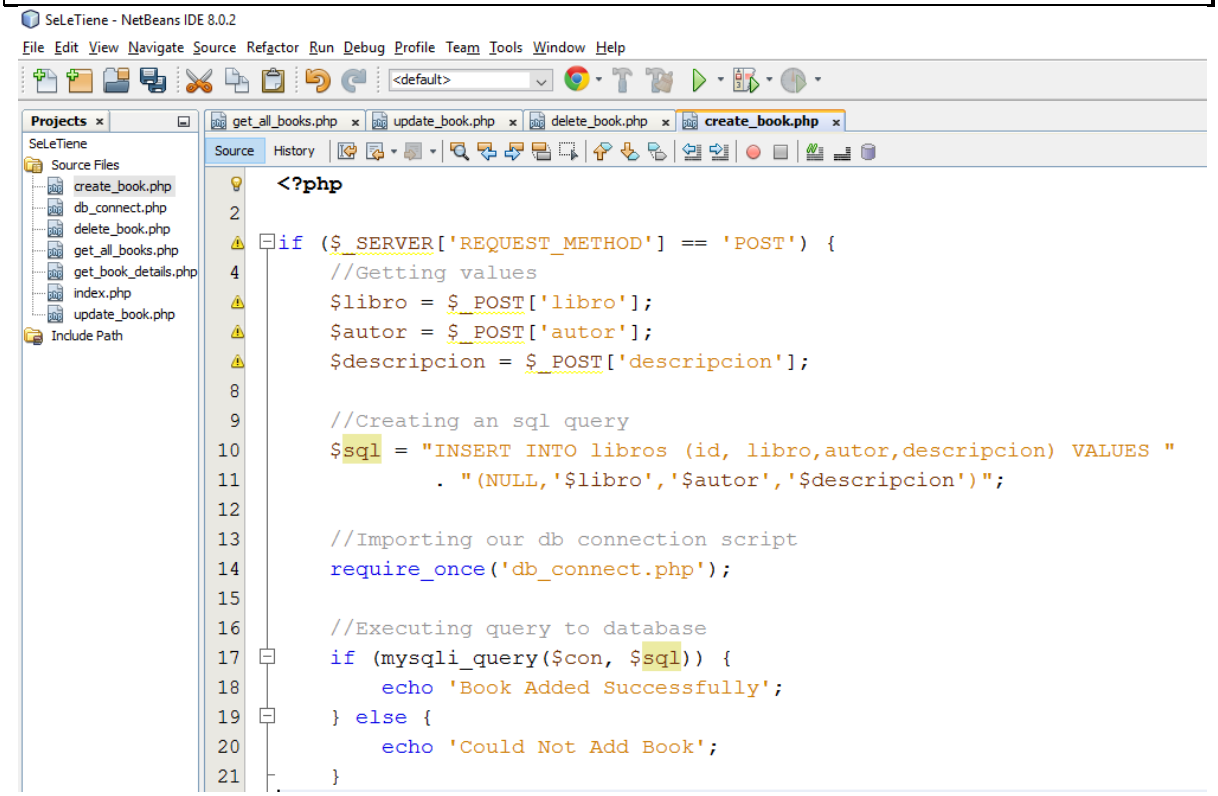
Ahora debemos crear las script en php para realizar el CRUD (Create, Read, Update, Delete) que requerimos.

create_book.php

Para crear un nuevo libro utilizaremos create_book.php

```
<?php

if ($_SERVER['REQUEST_METHOD'] == 'POST') {

    //Getting values
    $libro = $_POST['libro'];
    $autor = $_POST['autor'];
    $descripcion = $_POST['descripcion'];

    //Creating an sql query
    $sql = "INSERT INTO libros (libro,autor,descripcion) VALUES "
            . "('$libro','$autor','$descripcion')";

    //Importing our db connection script
    require_once('db_connect.php');

    //Executing query to database
    if (mysqli_query($con, $sql)) {
```

```
        echo 'Book Added Successfully';
    } else {
        echo 'Could Not Add Book';
    }

    //Closing the database
    mysqli_close($con);
}
```



get_book_details.php

El siguiente código permite realizar una búsqueda utilizando como parámetro el nombre del libro.

```
<?php

//Getting the requested id
$libro = $_GET['libro'];

//Importing database
require_once('db_connect.php');

//Creating sql query with where clause to get an specific employee
$sql = "SELECT * FROM libros WHERE libro='$libro'";

//getting result
$r = mysqli_query($con, $sql);
```

```
//pushing result to an array
$result = array();
$row = mysqli_fetch_array($r);
array_push($result, array(
    "id" => $row['id'],
    "libro" => $row['libro'],
    "autor" => $row['autor'],
    "descripcion" => $row['descripcion']
));

//displaying in json format
echo json_encode(array('result' => $result));

mysqli_close($con);
?>
```

```
<?php

//Getting the requested id
$libro = $_GET['libro'];

//Importing database
require_once('db_connect.php');

//Creating sql query with where clause to get an specific employee
$sql = "SELECT * FROM libros WHERE libro='$libro'";

//getting result
$r = mysqli_query($con, $sql);

//pushing result to an array
$result = array();
$row = mysqli_fetch_array($r);
array_push($result, array(
    "id" => $row['id'],
    "libro" => $row['libro'],
    "autor" => $row['autor'],
    "descripcion" => $row['descripcion']
));

//displaying in json format
echo json_encode(array('result' => $result));

mysqli_close($con);
?>
```

Get_all_books.php

Para obtener todos los libros

```
<?php

//Importing Database Script
require_once('db_connect.php');

//Creating sql query
$sql = "SELECT * FROM libros";

//getting result
$r = mysqli_query($con, $sql);

//creating a blank array
$result = array();

//looping through all the records fetched
while ($row = mysqli_fetch_array($r)) {

    //Pushing name and id in the blank array created
    array_push($result, array(
        "id" => $row['id'],
        "libro" => $row['libro'],
        "autor" => $row['autor'],
        "descripcion" => $row['descripcion']
    ));
}

//Displaying the array in json format
echo json_encode(array('result' => $result));

mysqli_close($con);
?>
```

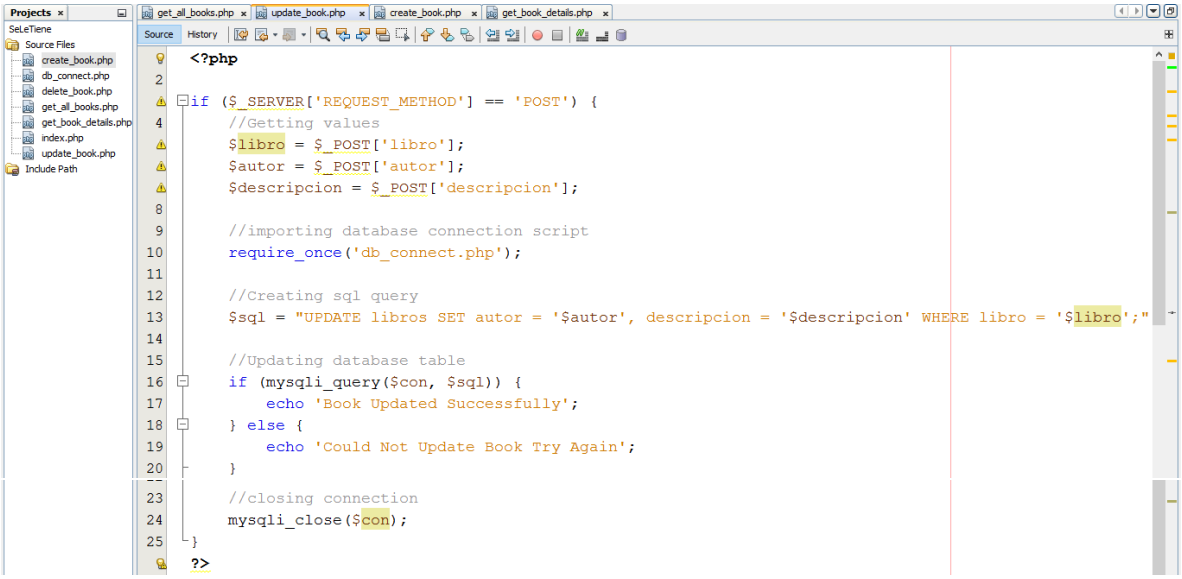
```php
<?php

//Importing Database Script
require_once('db_connect.php');

//Creating sql query
$sql = "SELECT * FROM libros";

//getting result
$r = mysqli_query($con, $sql);

//creating a blank array
$result = array();

//looping through all the records fetched
while ($row = mysqli_fetch_array($r)) {

    //Pushing name and id in the blank array created
    array_push($result, array(
        "id" => $row['id'],
        "libro" => $row['libro'],
        "autor" => $row['autor'],
        "descripcion" => $row['descripcion']
    ));
}

//Displaying the array in json format
echo json_encode(array('result' => $result));

mysqli_close($con);
?>
```

Update_book.php

Si se desea actualizar un libro

```php
<?php

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    //Getting values
    $id = $_POST['id'];
    $libro = $_POST['libro'];
    $autor = $_POST['autor'];
    $descripcion = $_POST['descripcion'];

    //importing database connection script
    require_once('db_connect.php');

    //Creating sql query
    $sql = "UPDATE libros SET autor = '$autor', descripcion = '$descripcion'
WHERE libro = '$libro';";
```

```
    //Updating database table
    if (mysqli_query($con, $sql)) {
        echo 'Book Updated Successfully';
    } else {
        echo 'Could Not Update Book Try Again';
    }

    //closing connection
    mysqli_close($con);
}
?>
```

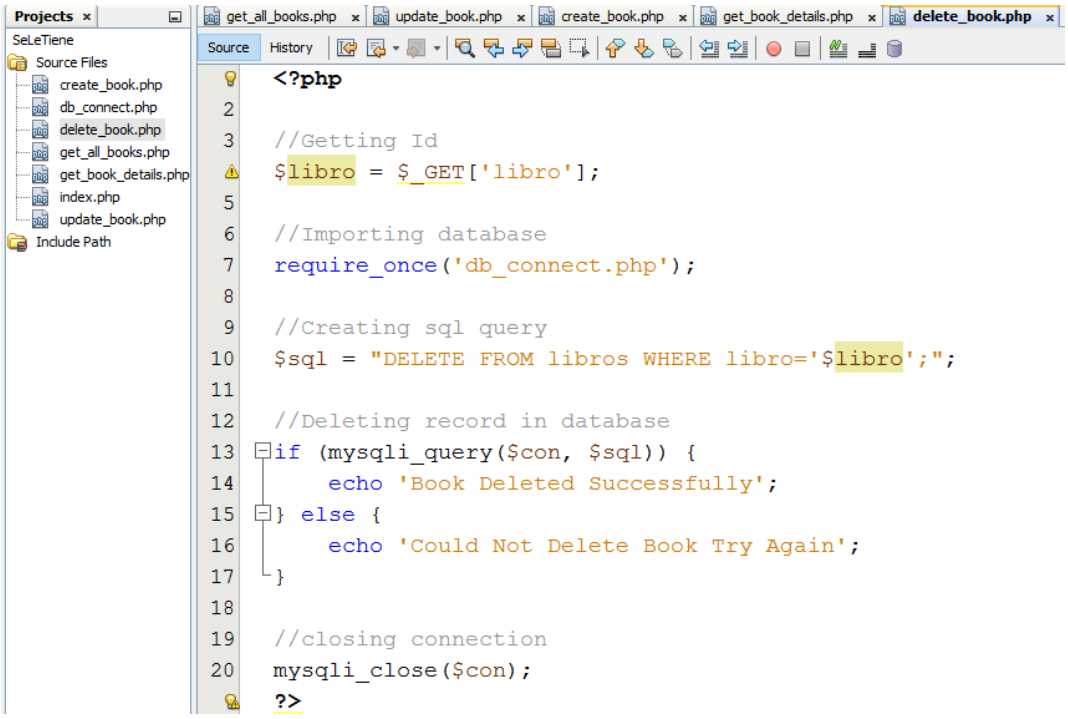


Delete_book.php

Para borrar un libro

```
<?php

//Getting Id
$libro = $_GET['libro'];

//Importing database
require_once('db_connect.php');

//Creating sql query
$sql = "DELETE FROM libros WHERE libro='$libro';";

//Deleting record in database
if (mysqli_query($con, $sql)) {
    echo 'Book Deleted Successfully';
```

```
} else {
    echo 'Could Not Delete Book Try Again';
}

//closing connection
mysqli_close($con);
?>
```



Aplicación Android

Primero debemos tener en cuenta que vamos a realizar una conexión a una base de datos externa por lo que necesitamos activar el permiso para que la aplicación se conecte a internet, en el manifest adicionamos este permiso.

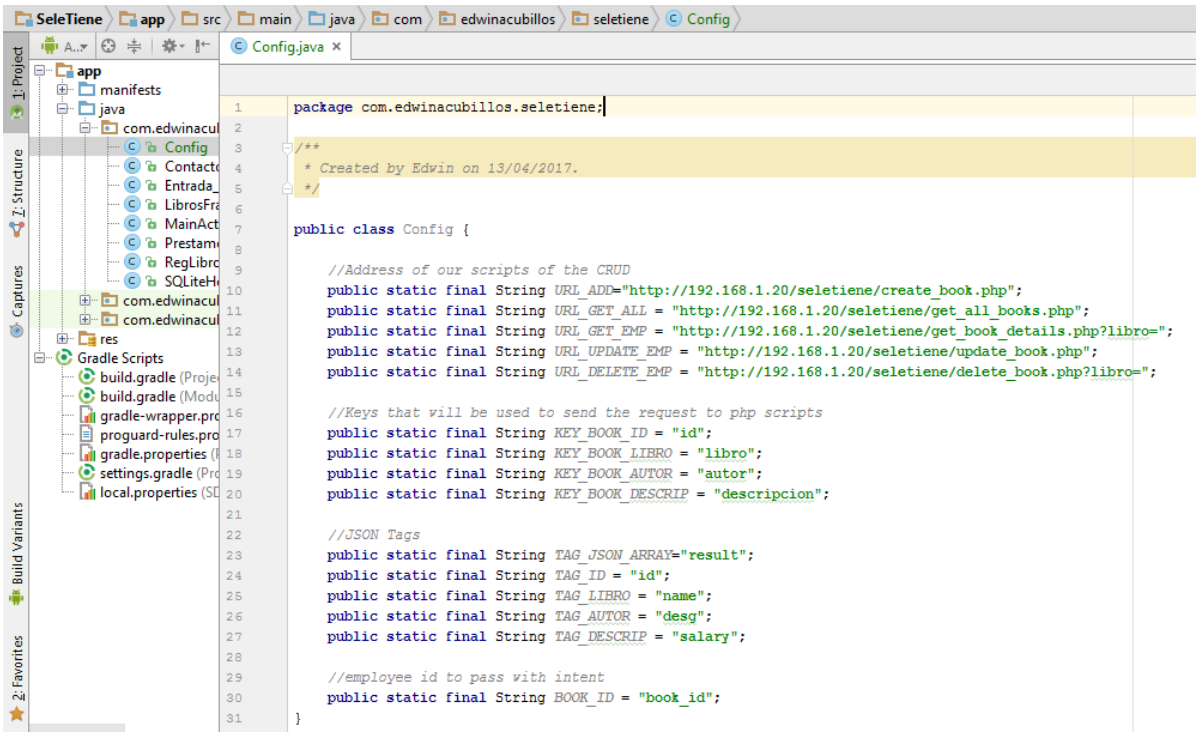
```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

```
Config.java × | RequestHandler.java × | RegLibroActivity.java × | AndroidManifest.xml ×
manifest | application
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.edwinacubillos.seletiene">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Se le Tiene"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="Se le Tiene"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".RegLibroActivity"></activity>
    </application>

</manifest>
```

Inicialmente vamos a crear una clase que contenga los String que necesitamos para realizar conexiones

```
public class Config {

    //Address of our scripts of the CRUD
    public static final String
URL_ADD="http://192.168.1.20/seletiene/create_book.php";
    public static final String URL_GET_ALL =
"http://192.168.1.20/seletiene/get_all_books.php";
    public static final String URL_GET_EMP =
"http://192.168.1.20/seletiene/get_book_details.php?libro=";
    public static final String URL_UPDATE_EMP =
"http://192.168.1.20/seletiene/update_book.php";
    public static final String URL_DELETE_EMP =
"http://192.168.1.20/seletiene/delete_book.php?libro=";

    //Keys that will be used to send the request to php scripts
    public static final String KEY_BOOK_ID = "id";
    public static final String KEY_BOOK_LIBRO = "libro";
    public static final String KEY_BOOK_AUTOR = "autor";
    public static final String KEY_BOOK_DESCRIP = "descripcion";

    //JSON Tags
    public static final String TAG_JSON_ARRAY="result";
    public static final String TAG_ID = "id";
    public static final String TAG_LIBRO = "name";
    public static final String TAG_AUTOR = "desg";
    public static final String TAG_DESCRIP = "salary";

    //employee id to pass with intent
```

```
    public static final String BOOK_ID = "book_id";
}
```



Ahora vamos a crear una nueva clase que maneje las conexiones a la red

```
public class RequestHandler {

    //Method to send httpPostRequest
    //This method is taking two arguments
    //First argument is the URL of the script to which we will send the
request
    //Other is an HashMap with name value pairs containing the data to be
send with the request
    public String sendPostRequest(String requestURL,
                                  HashMap<String, String> postDataParams)
{
        //Creating a URL
        URL url;

        //StringBuilder object to store the message retrieved from the
server
        StringBuilder sb = new StringBuilder();
        try {
            //Initializing Url
            url = new URL(requestURL);

            //Creating an httmlurl connection
            HttpURLConnection conn = (HttpURLConnection)
```

```
url.openConnection();

            //Configuring connection properties
            conn.setReadTimeout(15000);
            conn.setConnectTimeout(15000);
            conn.setRequestMethod("POST");
            conn.setDoInput(true);
            conn.setDoOutput(true);

            //Creating an output stream
            OutputStream os = conn.getOutputStream();

            //Writing parameters to the request
            //We are using a method getPostDataString which is defined
below
            BufferedWriter writer = new BufferedWriter(
                    new OutputStreamWriter(os, "UTF-8"));
            writer.write(getPostDataString(postDataParams));

            writer.flush();
            writer.close();
            os.close();
            int responseCode = conn.getResponseCode();

            if (responseCode == HttpsURLConnection.HTTP_OK) {

                BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                sb = new StringBuilder();
                String response;
                //Reading server response
                while ((response = br.readLine()) != null){
                    sb.append(response);
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
        return sb.toString();
    }

    public String sendGetRequest(String requestURL){
        StringBuilder sb =new StringBuilder();
        try {
            URL url = new URL(requestURL);
            HttpURLConnection con = (HttpURLConnection)
url.openConnection();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(con.getInputStream()));

            String s;
            while((s=bufferedReader.readLine())!=null){
                sb.append(s+"\n");
            }
```

```
        }catch(Exception e){
        }
        return sb.toString();
    }

    public String sendGetRequestParam(String requestURL, String id){
        StringBuilder sb =new StringBuilder();
        try {
            URL url = new URL(requestURL+id);
            HttpURLConnection con = (HttpURLConnection)
url.openConnection();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(con.getInputStream()));

            String s;
            while((s=bufferedReader.readLine())!=null){
                sb.append(s+"\n");
            }
        }catch(Exception e){
        }
        return sb.toString();
    }

    private String getPostDataString(HashMap<String, String> params)
throws UnsupportedEncodingException {
        StringBuilder result = new StringBuilder();
        boolean first = true;
        for (Map.Entry<String, String> entry : params.entrySet()) {
            if (first)
                first = false;
            else
                result.append("&");

            result.append(URLEncoder.encode(entry.getKey(), "UTF-8"));
            result.append("=");
            result.append(URLEncoder.encode(entry.getValue(), "UTF-8"));
        }

        return result.toString();
    }
}
```

Config.java × RequestHandler.java ×

RequestHandler

```java
package com.edwinacubillos.seletiene;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.HashMap;
import java.util.Map;

import javax.net.ssl.HttpsURLConnection;

/**
 * Created by Edwin on 13/04/2017.
 */

public class RequestHandler {

    //Method to send httpPostRequest
    //This method is taking two arguments
    //First argument is the URL of the script to which we will send the request
    //Other is an HashMap with name value pairs containing the data to be send with the request
    public String sendPostRequest(String requestURL,
                                  HashMap<String, String> postDataParams) {
        //Creating a URL
        URL url;

        //StringBuilder object to store the message retrieved from the server
        StringBuilder sb = new StringBuilder();
        try {
            //Initializing Url
            url = new URL(requestURL);

            //Creating an httmlurl connection
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();

            //Configuring connection properties
            conn.setReadTimeout(15000);
            conn.setConnectTimeout(15000);
            conn.setRequestMethod("POST");
            conn.setDoInput(true);
            conn.setDoOutput(true);

            //Creating an output stream
            OutputStream os = conn.getOutputStream();

            //Writing parameters to the request
            //We are using a method getPostDataString which is defined below
            BufferedWriter writer = new BufferedWriter(
                    new OutputStreamWriter(os, "UTF-8"));
            writer.write(getPostDataString(postDataParams));

            writer.flush();
            writer.close();
            os.close();
            int responseCode = conn.getResponseCode();

            if (responseCode == HttpsURLConnection.HTTP_OK) {

                BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                sb = new StringBuilder();
                String response;
                //Reading server response
                while ((response = br.readLine()) != null){
                    sb.append(response);
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
        return sb.toString();
    }

    public String sendGetRequest(String requestURL){
        StringBuilder sb =new StringBuilder();
        try {
```

```
82              URL url = new URL(requestURL);
83              HttpURLConnection con = (HttpURLConnection) url.openConnection();
84              BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(con.getInputStream()));
85
86              String s;
87              while((s=bufferedReader.readLine())!=null){
88                  sb.append(s+"\n");
89              }
90          }catch(Exception e){
91          }
92          return sb.toString();
93      }
94
95      public String sendGetRequestParam(String requestURL, String id){
96          StringBuilder sb =new StringBuilder();
97          try {
98              URL url = new URL(requestURL+id);
99              HttpURLConnection con = (HttpURLConnection) url.openConnection();
100             BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(con.getInputStream()));
101
102             String s;
103             while((s=bufferedReader.readLine())!=null){
104                 sb.append(s+"\n");
105             }
106         }catch(Exception e){
107         }
108         return sb.toString();
109     }
110
111     private String getPostDataString(HashMap<String, String> params) throws UnsupportedEncodingException {
112         StringBuilder result = new StringBuilder();
113         boolean first = true;
114         for (Map.Entry<String, String> entry : params.entrySet()) {
115             if (first)
116                 first = false;
117             else
118                 result.append("&");
119
120             result.append(URLEncoder.encode(entry.getKey(), "UTF-8"));
121             result.append("=");
122             result.append(URLEncoder.encode(entry.getValue(), "UTF-8"));
123         }
124
125         return result.toString();
126     }
127 }
```

addBook()

Generamos un método para guardar el nuevo libro

```
RequestHandler.java   RegLibroActivity.java   activ
gLibroActivity   onClick()

}

@Override
public void onClick(View v) {
    int id = v.getId();

    nombre = nombreLibro.getText().toString();
    autor = autorLibro.getText().toString();
    descrip = descripLibro.getText().toString();

    dataBD = new ContentValues();

    switch(id){
        case (R.id.bGuardarLibro):
            addBook();
            limpiarWidgets();
```

```
private void addBook() {

    class AddBook extends AsyncTask<Void,Void,String> {

        ProgressDialog loading;

        @Override
```

```
        protected void onPreExecute() {
            super.onPreExecute();
            loading =
ProgressDialog.show(RegLibroActivity.this,"Adding...","Wait...",false,fal
se);
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            loading.dismiss();

Toast.makeText(RegLibroActivity.this,s,Toast.LENGTH_LONG).show();
        }

        @Override
        protected String doInBackground(Void... v) {
            HashMap<String,String> params = new HashMap<>();
            params.put(Config.KEY_BOOK_LIBRO,nombre);
            params.put(Config.KEY_BOOK_AUTOR,autor);
            params.put(Config.KEY_BOOK_DESCRIP,descrip);

            RequestHandler rh = new RequestHandler();
            String res = rh.sendPostRequest(Config.URL_ADD, params);
            return res;
        }
    }

    AddBook ae = new AddBook();
    ae.execute();
}
```

Config.java × | RequestHandler.java × | RegLibroActivity.java × | activity_reg_libro.xml ×

```
139     private void addBook() {
140
141         class AddBook extends AsyncTask<Void,Void,String> {
142
143             ProgressDialog loading;
144
145             @Override
146             protected void onPreExecute() {
147                 super.onPreExecute();
148                 loading = ProgressDialog.show(RegLibroActivity.this,"Adding...","Wait...",false,false);
149             }
150
151             @Override
152             protected void onPostExecute(String s) {
153                 super.onPostExecute(s);
154                 loading.dismiss();
155                 Toast.makeText(RegLibroActivity.this,s,Toast.LENGTH_LONG).show();
156             }
157
158             @Override
159             protected String doInBackground(Void... v) {
160                 HashMap<String,String> params = new HashMap<>();
161                 params.put(Config.KEY_BOOK_LIBRO,nombre);
162                 params.put(Config.KEY_BOOK_AUTOR,autor);
163                 params.put(Config.KEY_BOOK_DESCRIP,descrip);
164
165                 RequestHandler rh = new RequestHandler();
166                 String res = rh.sendPostRequest(Config.URL_ADD, params);
167                 return res;
168             }
169         }
170         AddBook ae = new AddBook();
171         ae.execute();
172     }
173 }
```

UpdateBook()

```
private void updateBook() {

    class UpdateBook extends AsyncTask<Void,Void,String>{
        ProgressDialog loading;
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            loading =
ProgressDialog.show(RegLibroActivity.this,"Updating...","Wait...",false,f
alse);
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            loading.dismiss();

Toast.makeText(RegLibroActivity.this,s,Toast.LENGTH_LONG).show();
        }

        @Override
        protected String doInBackground(Void... params) {
            HashMap<String,String> hashMap = new HashMap<>();
            //hashMap.put(Config.KEY_BOOK_ID,id);
            hashMap.put(Config.KEY_BOOK_LIBRO,nombre);
            hashMap.put(Config.KEY_BOOK_AUTOR,autor);
```

```
            hashMap.put(Config.KEY_BOOK_DESCRIP,descrip);

            RequestHandler rh = new RequestHandler();

            String s =
rh.sendPostRequest(Config.URL_UPDATE_BOOK,hashMap);

            return s;
        }
    }
    UpdateBook ue = new UpdateBook();
    ue.execute();
}
```

```
.java ×  RequestHandler.java ×  RegLibroActivity.java ×  activity_reg_libro.xml ×

    private void updateBook() {

        class UpdateBook extends AsyncTask<Void,Void,String>{
            ProgressDialog loading;
            @Override
            protected void onPreExecute() {
                super.onPreExecute();
                loading = ProgressDialog.show(RegLibroActivity.this,"Updating...","Wait...",false,false);
            }

            @Override
            protected void onPostExecute(String s) {
                super.onPostExecute(s);
                loading.dismiss();
                Toast.makeText(RegLibroActivity.this,s,Toast.LENGTH_LONG).show();
            }

            @Override
            protected String doInBackground(Void... params) {
                HashMap<String,String> hashMap = new HashMap<>();
                //hashMap.put(Config.KEY_BOOK_ID,id);
                hashMap.put(Config.KEY_BOOK_LIBRO,nombre);
                hashMap.put(Config.KEY_BOOK_AUTOR,autor);
                hashMap.put(Config.KEY_BOOK_DESCRIP,descrip);

                RequestHandler rh = new RequestHandler();

                String s = rh.sendPostRequest(Config.URL_UPDATE_BOOK,hashMap);

                return s;
            }
        }
        UpdateBook ue = new UpdateBook();
        ue.execute();
    }
```

Delete_Book()

```
private void deleteBook(){
    class DeleteBook extends AsyncTask<Void,Void,String> {
        ProgressDialog loading;

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            loading = ProgressDialog.show(RegLibroActivity.this,
"Updating...", "Wait...", false, false);
        }
```

```
        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            loading.dismiss();
            Toast.makeText(RegLibroActivity.this, s,
Toast.LENGTH_LONG).show();
        }

        @Override
        protected String doInBackground(Void... params) {
            RequestHandler rh = new RequestHandler();
            String s = rh.sendGetRequestParam(Config.URL_DELETE_BOOK,
nombre);
            return s;
        }
    }

    DeleteBook de = new DeleteBook();
    de.execute();
}
```

```
.java ×  RequestHandler.java ×  RegLibroActivity.java ×  activity_reg_libro.xml ×
RegLibroActivity  updateBook()
        startActivity(intent);
        finish();*/
    }

    private void deleteBook(){
        class DeleteBook extends AsyncTask<Void,Void,String> {
            ProgressDialog loading;

            @Override
            protected void onPreExecute() {
                super.onPreExecute();
                loading = ProgressDialog.show(RegLibroActivity.this, "Updating...", "Wait...", false, false);
            }

            @Override
            protected void onPostExecute(String s) {
                super.onPostExecute(s);
                loading.dismiss();
                Toast.makeText(RegLibroActivity.this, s, Toast.LENGTH_LONG).show();
            }

            @Override
            protected String doInBackground(Void... params) {
                RequestHandler rh = new RequestHandler();
                String s = rh.sendGetRequestParam(Config.URL_DELETE_BOOK, nombre);
                return s;
            }
        }

        DeleteBook de = new DeleteBook();
        de.execute();
    }
```

showBook()

```
private void showBook(){
    class showBook extends AsyncTask<Void,Void,String>{
        ProgressDialog loading;
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
```

```
            loading =
ProgressDialog.show(RegLibroActivity.this,"Fetching...","Wait...",false,f
alse);
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            loading.dismiss();
            showData(s);
        }

        @Override
        protected String doInBackground(Void... params) {
            RequestHandler rh = new RequestHandler();
            String s =
rh.sendGetRequestParam(Config.URL_GET_BOOK,nombre);
            return s;
        }
    }
    showBook ge = new showBook();
    ge.execute();
}
```

```
private void showBook(){
    class showBook extends AsyncTask<Void,Void,String>{
        ProgressDialog loading;
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            loading = ProgressDialog.show(RegLibroActivity.this,"Fetching...","Wait...",false,false);
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            loading.dismiss();
            showData(s);
        }

        @Override
        protected String doInBackground(Void... params) {
            RequestHandler rh = new RequestHandler();
            String s = rh.sendGetRequestParam(Config.URL_GET_BOOK,nombre);
            return s;
        }
    }
    showBook ge = new showBook();
    ge.execute();
}
```

El formato en el que se reciben los datos es un Arreglo JSON, por lo que requerimos un objeto de este tipo para extraer la información

```
private void showData(String json){
    try {
        JSONObject jsonObject = new JSONObject(json);
        JSONArray result =
jsonObject.getJSONArray(Config.TAG_JSON_ARRAY);
```

```
        JSONObject c = result.getJSONObject(0);
        String autorE = c.getString(Config.TAG_AUTOR);
        String descripE = c.getString(Config.TAG_DESCRIP);
        Log.d("autor",autorE);
        autorLibro.setText(autorE);
        descripLibro.setText(descripE);

    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

```
private void showData(String json){
    try {
        JSONObject jsonObject = new JSONObject(json);
        JSONArray result = jsonObject.getJSONArray(Config.TAG_JSON_ARRAY);
        JSONObject c = result.getJSONObject(0);
        String autorE = c.getString(Config.TAG_AUTOR);
        String descripE = c.getString(Config.TAG_DESCRIP);
        Log.d("autor",autorE);
        autorLibro.setText(autorE);
        descripLibro.setText(descripE);

    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Finalmente para visualizar los libros en el listView personalizado hacemos una petición para obtener todos los libros al servidor

```
private void getJSON(){
    class GetJSON extends AsyncTask<Void,Void,String> {

        ProgressDialog loading;
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            loading = ProgressDialog.show(getContext(),"Fetching
Data","Wait...",false,false);
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            loading.dismiss();
            JSON_STRING = s;
            showBook();
        }

        @Override
        protected String doInBackground(Void... params) {
            RequestHandler rh = new RequestHandler();
```

```
            String s = rh.sendGetRequest(Config.URL_GET_ALL);
            return s;
        }
    }
    GetJSON gj = new GetJSON();
    gj.execute();
}
```

```
private void getJSON(){
    class GetJSON extends AsyncTask<Void,Void,String> {

        ProgressDialog loading;
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            loading = ProgressDialog.show(getContext(),"Fetching Data","Wait...",false,false);
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            loading.dismiss();
            JSON_STRING = s;
            showBook();
        }

        @Override
        protected String doInBackground(Void... params) {
            RequestHandler rh = new RequestHandler();
            String s = rh.sendGetRequest(Config.URL_GET_ALL);
            return s;
        }
    }
    GetJSON gj = new GetJSON();
    gj.execute();
}
```

y utilizamos el método showBook para extraer la información del JSON String donde esta el resultado y agregarlo a una lista para poderlo visualizar

```
private void showBook(){
    JSONObject jsonObject = null;
    ArrayList<HashMap<String,String>> list = new
ArrayList<HashMap<String, String>>();
    try {
        jsonObject = new JSONObject(JSON_STRING);
        JSONArray result =
jsonObject.getJSONArray(Config.TAG_JSON_ARRAY);

        for(int i = 0; i<result.length(); i++){
            JSONObject jo = result.getJSONObject(i);
            String id = jo.getString(Config.TAG_ID);
            String libro = jo.getString(Config.TAG_LIBRO);
            String autor = jo.getString(Config.TAG_AUTOR);
            String descrip = jo.getString(Config.TAG_DESCRIP);
            item = new Entrada_Libros(Integer.valueOf(id),
libro,autor,descrip);
            lista.add(item);
        }
```

```
    } catch (JSONException e) {
        e.printStackTrace();
    }

    Adapter adapter = new Adapter(getContext(), lista);
    listView.setAdapter(adapter);
}
```

```
private void showBook(){
    JSONObject jsonObject = null;
    ArrayList<HashMap<String,String>> list = new ArrayList<HashMap<String, String>>();
    try {
        jsonObject = new JSONObject(JSON_STRING);
        JSONArray result = jsonObject.getJSONArray(Config.TAG_JSON_ARRAY);

        for(int i = 0; i<result.length(); i++){
            JSONObject jo = result.getJSONObject(i);
            String id = jo.getString(Config.TAG_ID);
            String libro = jo.getString(Config.TAG_LIBRO);
            String autor = jo.getString(Config.TAG_AUTOR);
            String descrip = jo.getString(Config.TAG_DESCRIP);
            item = new Entrada_Libros(Integer.valueOf(id), libro,autor,descrip);
            lista.add(item);
        }

    } catch (JSONException e) {
        e.printStackTrace();
    }

    Adapter adapter = new Adapter(getContext(), lista);
    listView.setAdapter(adapter);
}
```

El adapter para cargar los datos es el personalizado que recibira un ArrayList de objetos

```
class Adapter extends ArrayAdapter<Entrada_Libros> {

    public Adapter(@NonNull Context context, ArrayList<Entrada_Libros>
libros) {
        super(context, R.layout.listitem, libros);
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView,
@NonNull ViewGroup parent) {

        Entrada_Libros libros = getItem(position);

        LayoutInflater inflater = LayoutInflater.from(getContext());
        View item = inflater.inflate(R.layout.listitem,null);

        TextView libro = (TextView) item.findViewById(R.id.tLibro);
        libro.setText(libros.getNombre());

        TextView descrip = (TextView) item.findViewById(R.id.tAutor);
```

```
        descrip.setText(libros.getAutor());

        TextView direcc = (TextView) item.findViewById(R.id.tDescrip);
        direcc.setText(libros.getDescrip());

        return item;
    }
}
```

```
class Adapter extends ArrayAdapter<Entrada_Libros> {

    public Adapter(@NonNull Context context, ArrayList<Entrada_Libros> libros) {
        super(context, R.layout.listitem, libros);
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {

        Entrada_Libros libros = getItem(position);

        LayoutInflater inflater = LayoutInflater.from(getContext());
        View item = inflater.inflate(R.layout.listitem,null);

        TextView libro = (TextView) item.findViewById(R.id.tLibro);
        libro.setText(libros.getNombre());

        TextView descrip = (TextView) item.findViewById(R.id.tAutor);
        descrip.setText(libros.getAutor());

        TextView direcc = (TextView) item.findViewById(R.id.tDescrip);
        direcc.setText(libros.getDescrip());

        return item;
    }
}
```