

COMUNICACIÓN ENTRE APP ANDROID Y MYSQL

Ingeniería de Software con Inteligencia Artificial

Aplicaciones requeridas:

- XAMPP
- Visual Studio Code
- Navegador de Internet actualizado
- Android Studio

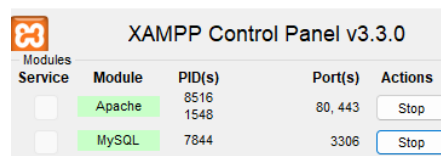
Consideraciones:

1. Una base de datos de MySQL reside en un servidor Web, normalmente alojado en Internet o Intranet (Xampp/Wamp); la misma que podrá ser aprovechada desde aplicaciones de escritorio, Web o **móvil** (esta última a través de servicios).
2. Las aplicaciones móviles residen (existen/están alojadas) en la unidad de almacenamiento del dispositivo móvil (Tablet/teléfono) por lo tanto requieren acceder a la información de la base de datos a través de un acceso especial – Internet.
3. Siga atentamente las instrucciones descritas en este manual.

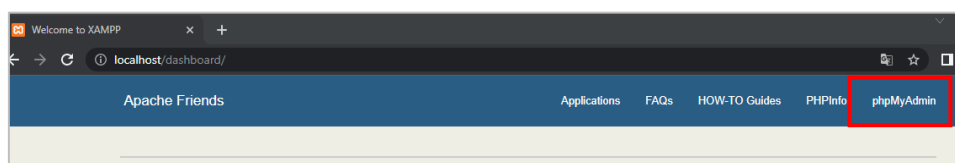
Primera parte:

CREACIÓN DE BASE DE DATOS

- a. Procederemos a iniciar XAMPP Server



- b. Luego ingresamos a nuestro navegador y escribimos <http://localhost>, seleccionamos phpMyAdmin del menú superior.



- c. Pulsamos en el botón **nueva** situado al lado izquierdo de su pantalla, escribimos el nombre de la base de datos (**restaurant**) y clic en **crear**.



- d. Construiremos una tabla llamada **platos** de **4** campos, verifique estos datos y pulse en **crear**

- e. Configure la tabla de acuerdo con los siguientes valores:

RECOMENDACIÓN: Escriba los nombres de campos en minúsculas

Nombre	Tipo	Longitud	Clave primaria	Auto increment
idplato	INT		X	X
tipo	VARCHAR	50		
nombreplato	VARCHAR	50		
precio	DECIMAL	5,2		

- f. Pulse clic en guardar para finalizar, se visualizará la estructura de la tabla tal como se indica en la siguiente imagen.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	idplato	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
2	tipo	varchar(50)	utf8mb4_general_ci		No	Ninguna			Cambiar Eliminar Más
3	nombreplato	varchar(50)	utf8mb4_general_ci		No	Ninguna			Cambiar Eliminar Más
4	precio	decimal(5,2)			No	Ninguna			Cambiar Eliminar Más

- g. ¡Hemos terminado de crear la base de datos y la tabla!, vamos a ingresar dos registros de prueba para poder mostrarlos en el navegador a través del lenguaje PHP. Posteriormente en este manual, indicaremos CÓMO ENVIAR LOS DATOS DESDE UNA APP ANDROID. Pulse clic en el botón INSERTAR situado en la parte superior.

h. Inserte los datos de prueba

Columna	Tipo	Función	Nulo	Valor
idplato	int(11)			
tipo	varchar(50)			Entrada
nombreplato	varchar(50)			Ceviche de pescado
precio	decimal(5,2)			7

☐ Ignorar

Columna	Tipo	Función	Nulo	Valor
idplato	int(11)			
tipo	varchar(50)			Plato de fondo
nombreplato	varchar(50)			Arroz con mariscos
precio	decimal(5,2)			35

Insertar como una nueva fila y luego Volver

Nota: No es necesario ingresar datos para el campo idplato

i. Pulse clic en examinar para visualizar los datos recientemente agregados

✓ Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0002 segundos.)

`SELECT * FROM `platos``

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Ordenar

Opciones extra

	idplato	tipo	nombreplato	precio
<input type="checkbox"/> Editar Copiar Borrar	1	Entrada	Ceviche de pescado	7.00
<input type="checkbox"/> Editar Copiar Borrar	2	Plato de fondo	Arroz con mariscos	35.00

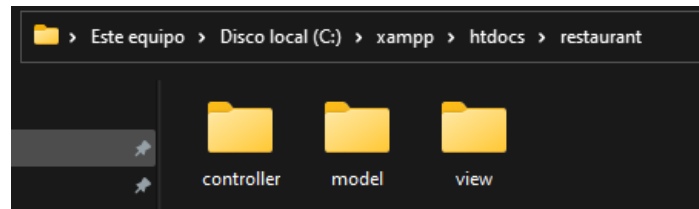
☐ Seleccionar todo Para los elementos que están marcados: [Editar](#) [Copiar](#)

Segunda parte:

APLICACIÓN WEB DE CONSULTA - PHP

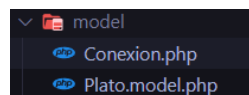
Ahora procederemos a crear una aplicación Web utilizando el lenguaje PHP para poder acceder a la información contenida en la BD y tabla creadas en la primera parte de este manual.

- Creemos una carpeta llamada **restaurant** en la ruta del servidor de XAMPP, dentro crearemos 3 directorios (**controller**, **model** y **view**) los cuales contendrán la lógica de nuestra aplicación.



Luego de crear estos directorios, abra el proyecto **restaurant** con Visual Studio Code

- Dentro de la carpeta **model** creamos los archivos: **Conexion.php** y **Plato.model.php**. Iniciamos la codificación con el archivo de conexión.



```
Conexion.php X
model > Conexion.php > PHP Intelephense > Conexion > getConexion
1  <?php
2
3  class Conexion{
4
5      //Guarda La conexión
6      protected $pdo;
7
8      //Acceder a La BD
9      private function Conectar(){
10         $cn = new PDO("mysql:host=localhost;port=3306;dbname=restaurant;charset=utf8", "root", "");
11         return $cn;
12     }
13
14     //Retornar la conexión
15     public function getConexion(){
16         try{
17             $pdo = $this->Conectar();
18             $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
19             return $pdo;
20         }catch(Exception $e){
21             die($e->getMessage());
22         }
23     }
24
25 } //Fin de la clase
26
27 ?>
```

Diagrama de anotaciones:

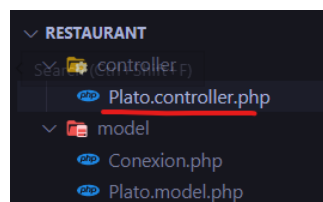
- Servidor (punta a localhost en la línea 10)
- Base de datos (punta a dbname=restaurant en la línea 10)
- Usuario (punta a root en la línea 10)
- Puerto (no cambiar) (punta a port=3306 en la línea 10)
- Codificación (punta a charset=utf8 en la línea 10)
- Contraseña (punta a la coma después de root en la línea 10)

El método **Conectar** establece la comunicación con el servidor, el método **getConexion** DEVUELVE la conexión para el método que la necesite (*insertar, listar, buscar, eliminar, etc.*)

- c. El archivo **Plato.model.php** es sencillo, se agregaron comentarios de todo lo que se está haciendo que podrá agregar de forma opcional.

```
Plato.model.php X
model > Plato.model.php > PHP Intelephense > Plato > __construct
1  <?php
2
3  //Importar la conexión
4  require_once 'Conexion.php';
5
6  //La clase plato heredará los métodos de la clase Conexion
7  class Plato extends Conexion{
8
9      //Almacena la conexión
10     private $conexion;
11
12     //Pasamos la conexión
13     public function __construct()
14     {
15         $this->conexion = parent::getConexion();
16     }
17
18     public function listarPlatos(){
19         try{
20             //Preparamos la consulta
21             $consulta = $this->conexion->prepare("SELECT * FROM platos ORDER BY idplato");
22             //Ejecutamos la consulta
23             $consulta->execute();
24
25             //Presentaremos los datos obtenidos como ARRAY asociativo
26             $tabla = $consulta->fetchAll(PDO::FETCH_ASSOC);
27             //Retornamos los datos obtenidos
28             return $tabla;
29         }catch(Exception $e){
30             die($e->getMessage());
31         }
32     }
33 }
34
35 }
36
37 ?>
```

- d. En la carpeta **controller**, cree un archivo llamado **Plato.controller.php** y agregue las siguientes instrucciones:



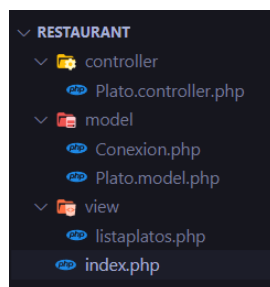
```
Plato.controller.php X
controller > Plato.controller.php > ...
1  <?php
2
3  //Incorporamos el modelo (lógica)
4  require_once '../model/Plato.model.php';
5
```

Continúa en la siguiente página

... Plato.controller.php

```
6 //Si existe alguna operación en curso
7 if (isset($_POST['op'])){
8
9     //Instanciamos la clase Plato
10    $plato = new Plato();
11
12    //Si la operación es listar...
13    if ($_POST['op'] == 'listarPlatos'){
14
15        //Ejecutamos el método y guardamos el resultado
16        $datos = $plato->listarPlatos();
17
18        //Recorremos cada registro obtenido
19        foreach($datos as $registro){
20
21            //El resultado se renderizará en el navegador
22            //como una fila de una tabla HTML
23            echo "
24            <tr>
25                <td>{$registro['idplato']}</td>
26                <td>{$registro['tipo']}</td>
27                <td>{$registro['nombreplato']}</td>
28                <td>{$registro['precio']}</td>
29            </tr>
30            ";
31
32        } //Fin foreach
33    } //Fin if ($_POST)
34
35 } //Fin de if isset()
36
37 ?>
```

- e. Ahora cree un archivo **index.php** en la raíz del proyecto, y otro archivo llamado **listaplatos.php** en la carpeta **view**. Todo el proyecto debería tener la siguiente estructura:



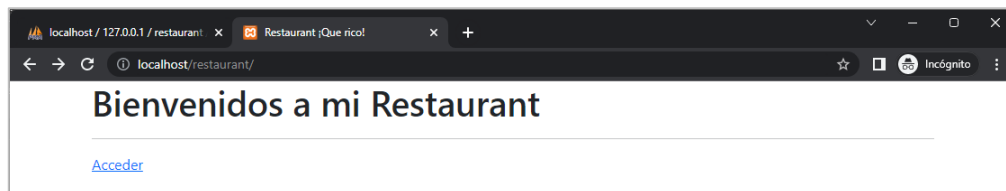
A continuación, se detallará el contenido de las vistas (interfaces) con las que el usuario interactúa, para mejorar su presentación se está utilizando Bootstrap 5, para obtener esta librería, visite la siguiente dirección:

<https://getbootstrap.com/docs/5.2/getting-started/download/>



- f. Este es el código del **index.php**, solo contendrá un mensaje de bienvenida y un enlace que nos dirija al archivo **listaplatos.php** dentro de **view**, tal como se puede apreciar a continuación:

```
index.php X
index.php > html > body > div.container > hr
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Restaurant ¡Que rico!</title>
8
9   <!-- CDN Bootstrap -->
10  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css" rel="stylesheet">
11  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.min.js" rel="script">
12
13 </head>
14 <body>
15
16   <div class="container">
17     <h1>Bienvenidos a mi Restaurant</h1>
18     <hr>
19     <a href="view/listaplatos.php">Acceder</a>
20   </div>
21
22 </body>
23 </html>
```



- g. El código para **listaplatos.php**

```
listaplatos.php X
view > listaplatos.php > html > body
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Carta</title>
8
9   <!-- CDN Bootstrap -->
10  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css" rel="stylesheet">
11  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.min.js" rel="script">
12
13 </head>
14 <body>
15
```

Se está utilizando Bootstrap para mejorar la interfaz

Dentro del cuerpo de la página, construimos una tabla HTML (observe la imagen contenida en la siguiente página).

```

15
16 <div class="container">
17   <h1>Lista de platos</h1>
18   <hr>
19
20   <table id="tabla-platos" class="table table-striped">
21     <thead>
22       <tr>
23         <th>ID</th>
24         <th>Tipo de plato</th>
25         <th>Nombre de plato</th>
26         <th>Precio</th>
27       </tr>
28     </thead>
29     <tbody>
30
31     </tbody>
32   </table>
33 </div>
34

```

Y debajo de este HTML, tenemos el bloque de instrucciones JS que permite cargar los datos de manera asíncrona (AJAX). El CDN de jQuery utilizado en la línea 35 de esta guía es la siguiente:

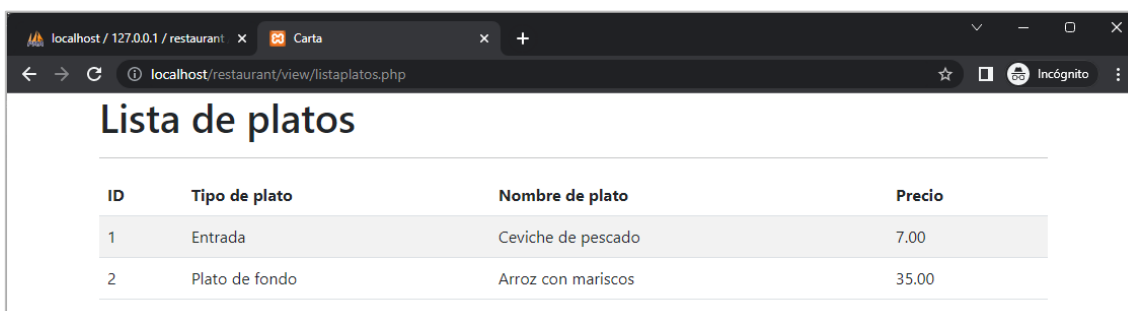
```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
```

```

34
35 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
36
37 <script>
38   $(document).ready(function(){
39
40     //Al cargar la página traemos los datos de forma asíncrona
41     $.ajax({
42       url: '../controller/Plato.controller.php',
43       type: 'POST',
44       data: 'op=listarPlatos',
45       success: function(result){
46         $("#tabla-platos tbody").html(result);
47       }
48     });
49
50   });
51 </script>
52
53 </body>
54 </html>

```

- h. Ingrese a la aplicación escribiendo <http://localhost/restaurant> seguidamente pulse clic en ingresar para ver la lista de platos con la información cargada desde la base de datos.



ID	Tipo de plato	Nombre de plato	Precio
1	Entrada	Ceviche de pescado	7.00
2	Plato de fondo	Arroz con mariscos	35.00

Tercera parte:

SERVICIO WEB PARA REGISTRO

Ahora crearemos un servicio en PHP que permita recibir los datos enviados a través de la Web. Este componente de Software será lo que utilizemos para enviar nuestros datos desde Android.

- a. Abrimos el archivo **Plato.model.php** y agregamos un nuevo método debajo del método **listarPlatos**, tal como se muestra en la siguiente imagen:



```
model > Plato.model.php > PHP Intelephense > Plato > registrarPlato
40 $tabla = $consulta->fetchAll(PDO::FETCH_ASSOC);
27 //Retornamos los datos obtenidos
28 return $tabla;
29
30 }catch(Exception $e){
31 die($e->getMessage());
32 }
33 }//Fin listarPlatos
34
35 public function registrarPlato($tipo, $nombreplato, $precio){
36 try{
37 //Preparamos la consulta, Los signos de ? son los valores que debemos pasar
38 $consulta = $this->conexion->prepare("INSERT INTO platos (tipo, nombreplato, precio) VALUES (?, ?, ?)");
39 //Ejecutamos la consulta pasándole las variables
40 $consulta->execute(array($tipo, $nombreplato, $precio));
41
42 }catch(Exception $e){
43 die($e->getMessage());
44 }
45 }// Fin registrarPlato
46
47 }//Fin de la clase
48
49 ?>
```

- b. Abrimos el archivo **Plato.controller.php**, de preferencia y para que el código se vea más ordenado, cerramos la sección de la operación **listarPlatos** y agregamos el bloque para **registrarPlato**, tal como se observa en la siguiente imagen:

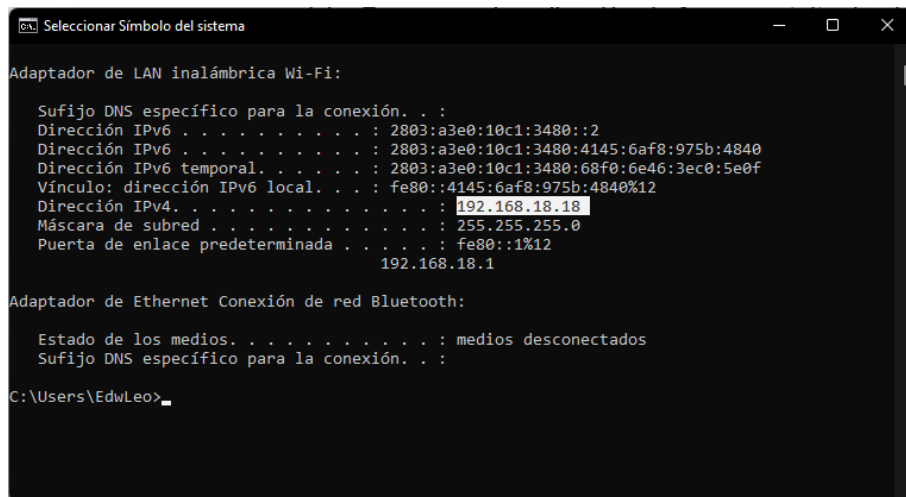


```
controller > Plato.controller.php > ...
12 //Si la operación es listar...
13 > if ($_POST['op'] == 'listarPlatos'){ ...
33 } //Fin if ($_POST)
34
35 //Si la operación es registrar...
36 if ($_POST['op'] == 'registrarPlato'){
37 $tipo = $_POST['tipo'];
38 $nombreplato = $_POST['nombreplato'];
39 $precio = $_POST['precio'];
40 $plato->registrarPlato($tipo, $nombreplato, $precio);
41 }
42
43 } //Fin de if isset()
44
45 ?>
```

- c. Opcionalmente, si dispone de la aplicación POSTMAN, podría verificar el funcionamiento de este servicio. Descargue la aplicación de forma gratuita desde este link:

<https://www.postman.com/>

Para las pruebas deberá saber cuál es su **dirección IP** local, abra el símbolo de sistema (**CMD**) y escriba el comando **ipconfig**, seguidamente verifique en la sección según corresponda (puede estar utilizando un cable de red/**Ethernet** o bien inalámbrico/**Wifi**). En mi caso la conexión es por Wifi, por lo tanto, mi dirección IP es: **192.168.18.18**



```
Seleccinon Smbolo del sistema

Adaptador de LAN inalámbrica Wi-Fi:

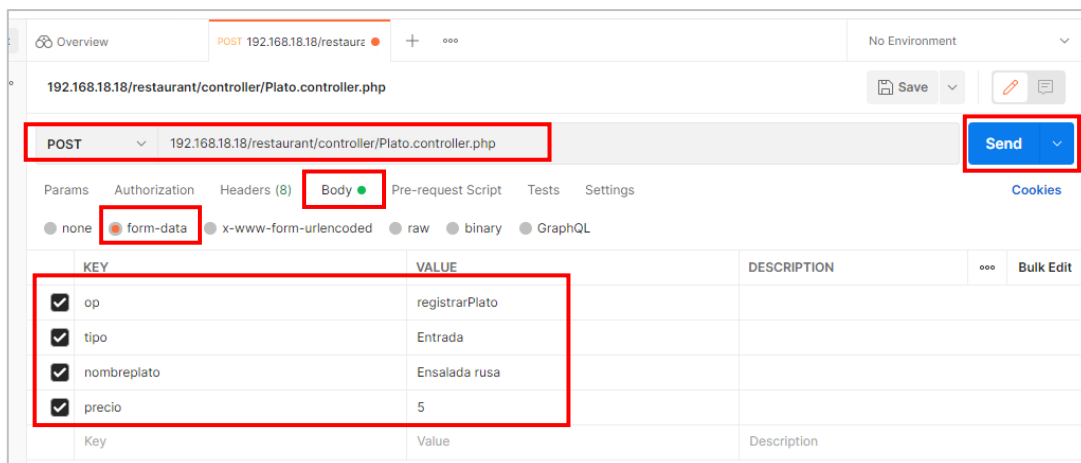
Sufijo DNS específico para la conexión. . . :
Dirección IPv6 . . . . . : 2803:a3e0:10c1:3480::2
Dirección IPv6 . . . . . : 2803:a3e0:10c1:3480:4145:6af8:975b:4840
Dirección IPv6 temporal. . . . . : 2803:a3e0:10c1:3480:68f0:6e46:3ec0:5e0f
Vínculo: dirección IPv6 local. . . . : fe80::4145:6af8:975b:4840%12
Dirección IPv4. . . . . : 192.168.18.18
Máscara de subred . . . . . : 255.255.255.0
Puerta de enlace predeterminada . . . : fe80::1%12
192.168.18.1

Adaptador de Ethernet Conexión de red Bluetooth:

Estado de los medios. . . . . : medios desconectados
Sufijo DNS específico para la conexión. . . :

C:\Users\EdwLeo>
```

- d. Establezca los parámetros necesarios para la prueba, observe la siguiente imagen:



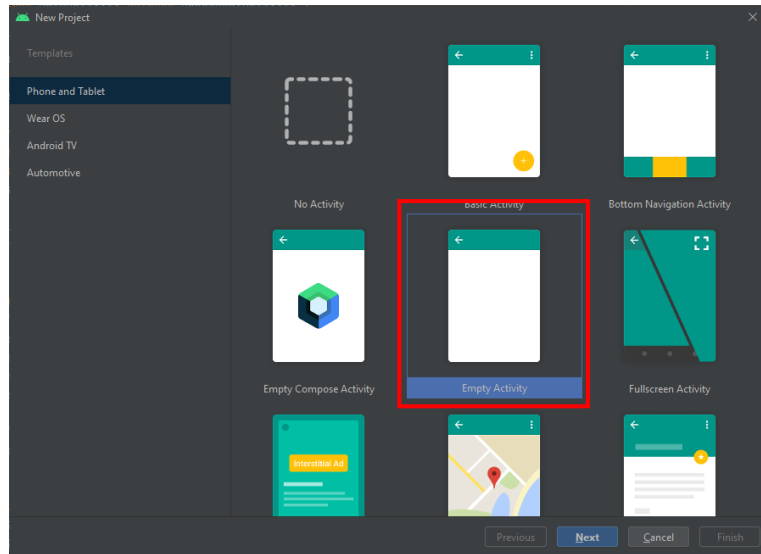
Luego de enviar el dato desde POSTMAN, este debería poder visualizarse desde el navegador



NO continúe con el siguiente apartado, si esta última operación no ha sido verificada

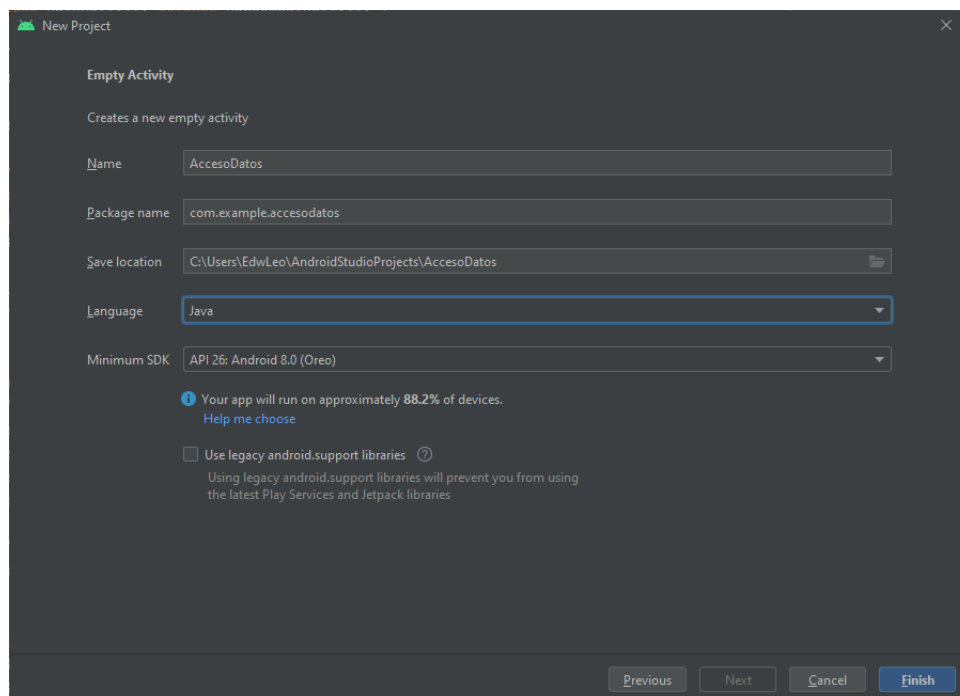
Cuarta parte:
CONSTRUYENDO APP MÓVIL (INTERFAZ)

- a. Utilizaremos Android Studio, procedemos a crear un proyecto con un Activity vacío



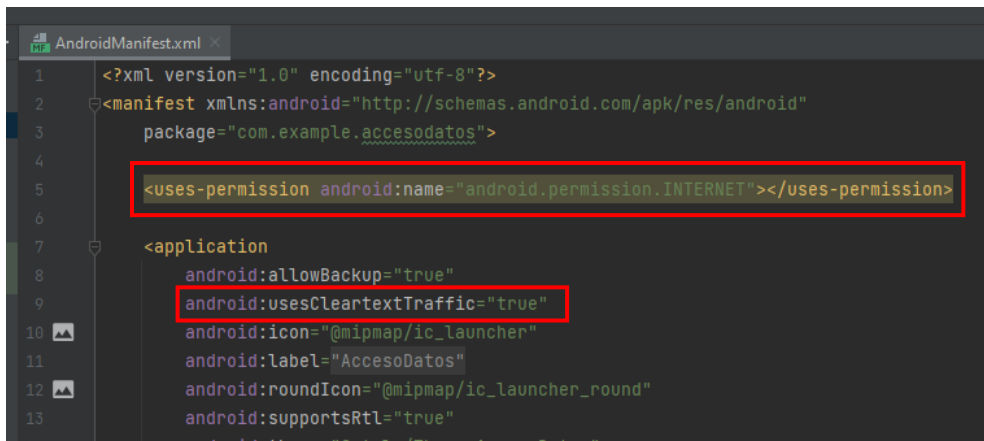
Seleccione: **Empty Activity**

- b. Asignamos el nombre del proyecto, como lenguaje JAVA y la versión mínima SDK (compatibilidad con la versión de Android), será la 8; pulsamos clic en **finish**.



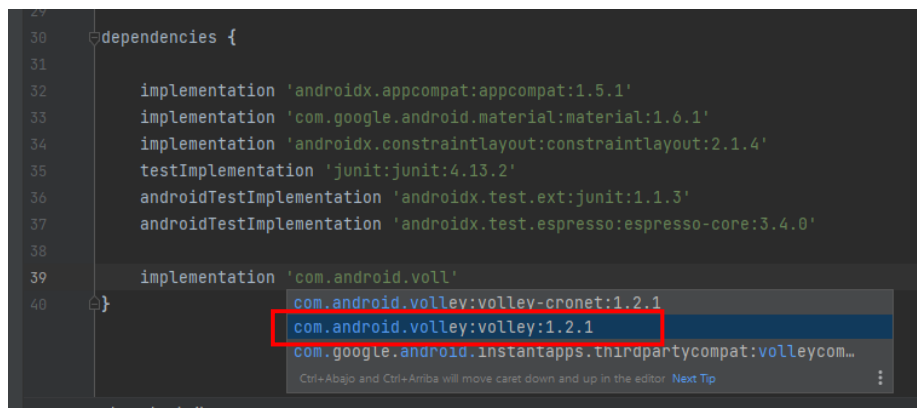
Puede desarrollar la App desde Android 5.0 o superior

- c. Una vez se finalice la creación del proyecto, abre el archivo AndroidManifest.xml y agregue la siguiente línea:



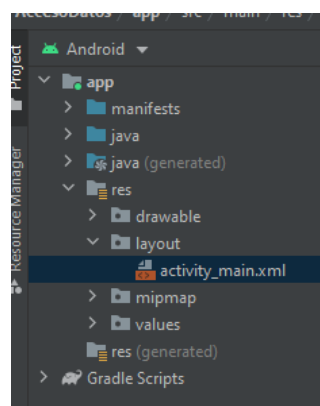
Ambas instrucciones asignan permisos para enviar o recibir datos desde Internet

- d. Seguidamente abra el módulo archivo build.gradle y agregue una nueva dependencia:



*Antes de continuar, pulse clic en el menú **Build** y seleccione **Make Project** para actualizar los cambios realizados recientemente*

- e. Ahora abra el archivo de interfaz, es decir el activity_main.xml contenido dentro de res > layout de su proyecto:



- f. Ahora cambie el constraintLayout por LinearLayout, este paso es opcional, ya que si Ud. desea, puede agregar los componentes utilizando el Layout que mejor domine.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical"
8      android:padding="50dp"
9      tools:context=".MainActivity">
10
11  </LinearLayout>

```

Agregue 3 cajas de texto o EditText

```

10
11  <EditText
12      android:id="@+id/etTipo"
13      android:layout_width="match_parent"
14      android:layout_height="wrap_content"
15      android:layout_marginBottom="20dp"
16      android:hint="Tipo de plato" />
17
18  <EditText
19      android:id="@+id/etNombreplato"
20      android:layout_width="match_parent"
21      android:layout_height="wrap_content"
22      android:layout_marginBottom="20dp"
23      android:hint="Nombre del plato" />
24
25  <EditText
26      android:id="@+id/etPrecio"
27      android:layout_width="match_parent"
28      android:layout_height="wrap_content"
29      android:layout_marginBottom="20dp"
30      android:hint="Precio" />
31

```

Y debajo un botón, eso es toda la interfaz

```

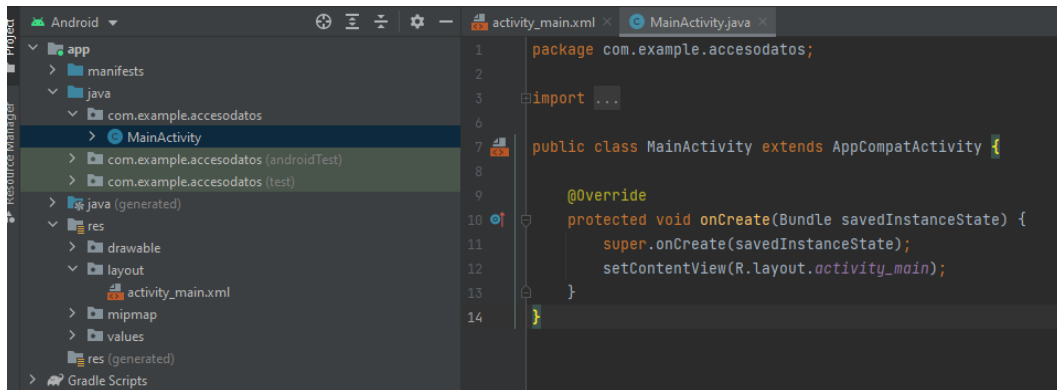
32  <Button
33      android:id="@+id/btGuardar"
34      android:text="Guardar datos"
35      android:layout_width="match_parent"
36      android:inputType="numberDecimal"
37      android:layout_height="wrap_content" />
38
39  </LinearLayout>

```

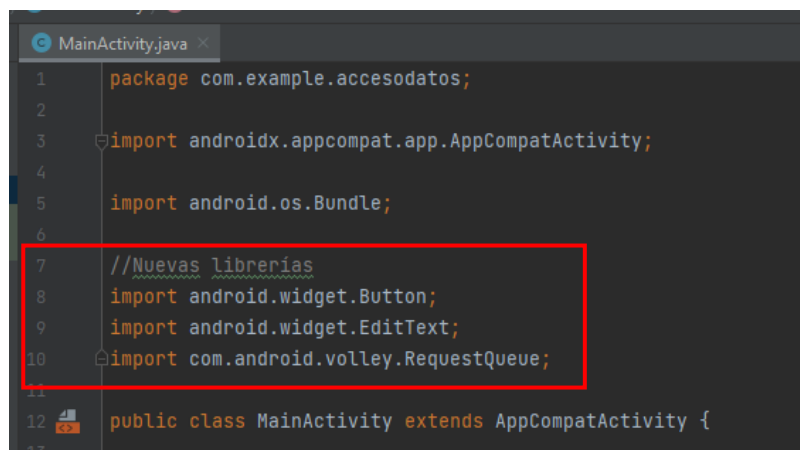
The screenshot shows the visual representation of the XML code. It features three vertically stacked text input fields with placeholder text: 'Tipo de plato', 'Nombre del plato', and 'Precio'. Below these fields is a blue button with the text 'GUARDAR DATOS' in white capital letters.

Quinta parte (FINAL):
CONSTRUYENDO APP MÓVIL (BACKEND Y PRUEBAS)

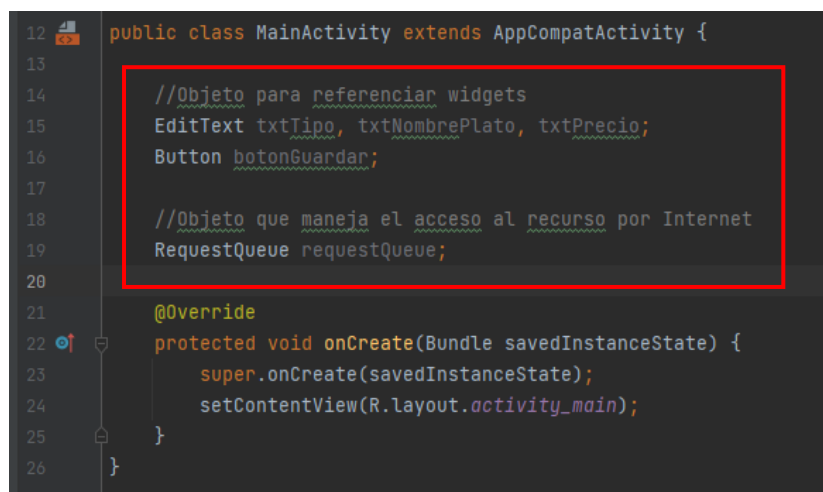
- a. Ahora solo nos queda darle comportamiento a la aplicación. Para ello abriremos el archivo MainActivity.java



- b. Inicie agregando algunas de las librerías a utilizar:



- c. Definimos los objetos a utilizar



- d. Creamos el método **loadUI** (español: cargar Interfaz Gráfica), este método sirve para poder asignar a los objetos creados anteriormente el control sobre un widget; este método se invocará en el evento onCreate (línea 26) tal como se ve en la siguiente imagen:

```
21      @Override
22      protected void onCreate(Bundle savedInstanceState) {
23          super.onCreate(savedInstanceState);
24          setContentView(R.layout.activity_main);
25
26          loadUI();
27      }
28
29      private void loadUI(){
30          txtTipo = findViewById(R.id.etTipo);
31          txtNombrePlato = findViewById(R.id.etNombreplato);
32          txtPrecio = findViewById(R.id.etPrecio);
33
34          botonGuardar = findViewById(R.id.btGuardar);
35      }
36  }
```

- e. Ahora asignaremos el evento clic para el botón, debería quedar tal como en la imagen:

```
22      @Override
23      protected void onCreate(Bundle savedInstanceState) {
24          super.onCreate(savedInstanceState);
25          setContentView(R.layout.activity_main);
26
27          loadUI();
28
29          //Asignando evento clic al botón
30          botonGuardar.setOnClickListener(new View.OnClickListener() {
31              @Override
32              public void onClick(View view) {
33
34              }
35          });
36      }
37  }
```

- f. Debajo de loadUI, crearemos uno nuevo, este, contendrá los procedimientos necesarios para enviar los datos a través de Internet, se ejecuta desde evento onClick

```
29          //Asignando evento clic al botón
30          botonGuardar.setOnClickListener(new View.OnClickListener() {
31              @Override
32              public void onClick(View view) {
33                  enviarDatosWS();
34              }
35          });
36      }
37
38      private void loadUI(){
39          txtTipo = findViewById(R.id.etTipo);
40          txtNombrePlato = findViewById(R.id.etNombreplato);
41          txtPrecio = findViewById(R.id.etPrecio);
42
43          botonGuardar = findViewById(R.id.btGuardar);
44      }
45
46      private void enviarDatosWS(){
47
48      }
49  }
```

g. Funcionamiento de enviarDatosWS

```

58 private void enviarDatosWS(){
59
60     //Leemos las cajas de texto
61     String tipo = txtTipo.getText().toString();
62     String nombrePlato = txtNombrePlato.getText().toString();
63     String precio = txtPrecio.getText().toString();
64
65     //Preparamos la consulta
66     RequestQueue servicio = Volley.newRequestQueue( context, this);
67
68     //Creamos una consulta basada en cadena (texto)
69     StringRequest respuesta = new StringRequest(
70         Request.Method.POST,
71         url: "http://192.168.18.18/restaurant/controller/Plato.controller.php",
72         new Response.Listener<String>() {
73             @Override
74             public void onResponse(String response) {
75                 Toast.makeText(getApplicationContext(), text: "Enviado correctamente", Toast.LENGTH_LONG).show();
76             }
77         },
78         new Response.ErrorListener() {
79             @Override
80             public void onErrorResponse(VolleyError error) {
81                 Toast.makeText(getApplicationContext(), text: "Error", Toast.LENGTH_LONG).show();
82             }
83         }
84     );
85
86     @Override
87     protected Map<String, String> getParams() throws AuthFailureError {
88         Map<String, String> params = new HashMap<>();
89
90         params.put( k: "op", v: "registrarPlato");
91         params.put( k: "tipo", tipo);
92         params.put( k: "nombreplato", nombrePlato);
93         params.put( k: "precio", precio);
94
95         return params;
96     };
97
98     servicio.add(respuesta);
99 } //Fin enviarDatosWS

```

Resultado obtenido:

The screenshot displays two side-by-side views. On the left, a web browser window shows a table titled 'Lista de platos' with the following data:

ID	Tipo de plato	Nombre de plato	Precio
1	Entrada	Ceviche de pescado	7.00
2	Plato de fondo	Arroz con mariscos	35.00
3	Entrada	Ensalada rusa	5.00
4	Sopas	Sopa a la minuta	12.00

On the right, a mobile app interface titled 'AccesoDatos' is shown. It features three input fields labeled 'Sopas', 'Sopa a la minuta', and '12'. Below these fields is a purple button labeled 'Guardar datos'. A keyboard is visible at the bottom of the screen.