

# **¿QUÉ HAGO HOY?**

Pablo Blázquez Bautista

Rafa

Colegio Santísima Trinidad, Salamanca

2021

## **1. Estudio del problema y análisis del sistema**

1.1. Introducción

1.2. Finalidad

1.3. Objetivos

## **2. Modelado de la solución**

2.1. Recursos humanos

2.2. Recursos hardware

2.3. Recursos software

## **3. Planificación.**

3.1. Diseño del proyecto

## **4. Fase de pruebas**

4.1. Pruebas realizadas.

## **5. Conclusiones finales**

5.1. Grado de cumplimiento de los objetivos fijados.

5.2. Propuesta de modificaciones o ampliaciones futuras del sistema implementado.

## **6. Documentación del sistema desarrollado**

6.1. Manual de Instalación.

6.2. Manual de uso.

## **7. Bibliografía**

## **1. Estudio del problema y análisis del sistema**

### **1.1. Introducción**

El proyecto “QUÉ HAGO HOY” consiste en una aplicación desarrollada en el sistema operativo de Android.

Esta aplicación esta desarrollada para un sistema operativo Android 9.0 (Pie).

### **1.2. Finalidad**

La finalidad de esta aplicación consiste en que cualquier persona que tenga en mente la realización de planes de ocio pueda visualizar y solicitar los planes de ocio que más deseé en un día concreto, así como poder escoger los servicios de ocio y comida de una manera aleatoria para que no tenga la necesidad de pensar en que servicio y restaurante desea escoger.

### **1.3. Objetivos**

El objetivo principal de la aplicación es que la persona o grupo de personas puedan seleccionar sus planes de ocio sin la tediosa tarea de tener que estar pensando cual es la que desean ya que esto en algunas ocasiones puede llevar a discusión, por lo que la aplicación les facilita si así lo desean de la herramienta de solicitar los servicios y restaurantes de manera aleatoria y así poder disfrutar de un plan de ocio nuevo.

## **2. Modelado de la solución**

### **2.1. Recursos humanos**

### **2.2. Recursos hardware**

Ordenador.

Teléfono móvil.

### 2.3. Recursos software

IDE Android Studio, para la elaboración del código que permitirá la creación de la aplicación en el sistema operativo de Android.

Xampp, sistema de gestión de bases de datos donde se creará y almacenará la base de datos necesaria para el inicio y registro de los usuarios así como para almacenar los servicios solicitados.

Laravel, framework necesario para la conexión entre nuestra aplicación y la base de datos.

Además se ha utilizado la IDE de NetBeans para elaborar el código de Laravel.

## 3. Planificación.

### 3.1. Diseño del proyecto

Como primer paso se ha estudiado los elementos o recursos de software necesarios los cuales son:

- Una IDE para creación del código Android para la elaboración de la aplicación.
- Un sistema de gestión de bases de datos.
- Una herramienta que nos permita conectar nuestra aplicación con la base de datos.

Como IDE de creación del código de la aplicación Android se ha utilizado “Android Studio”, utilizando en él, el lenguaje de programación Java.

Para el sistema de gestión de base de datos se ha utilizado “Xampp”, puesto que gracias a él se puede crear la base de datos así como el sistema de conexión de la base de datos con nuestra aplicación que se ha hecho con “Laravel” utilizando código PHP.

Una vez teniendo fijadas estas herramientas procedemos a crear nuestra aplicación empezando “Android Studio”.

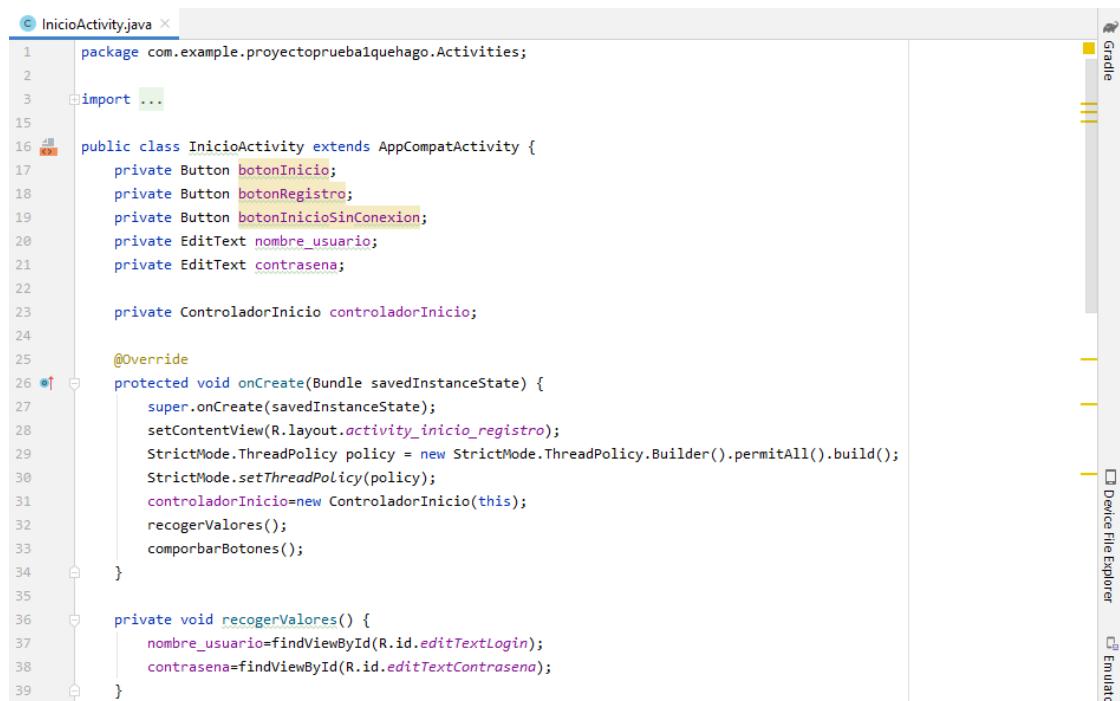
El primer paso de Android será crear un sistema de acceso y de registro mediante usuario y contraseña, para ello creamos una pagina de “Inicio” y otra de “Registro”, así como sus controladores y sus layout.xml.

INICIO

El objetivo del apartado de inicio consiste en que muestre la vista inicial, la cual se abre desde MainActivity que únicamente se encarga de crear esta vista, dentro de esta vista inicial se iniciara sesión si ya está creado el usuario.

Podemos encontrar dos botones de inicio, uno que realiza el inicio normal, es decir se ponen el usuario y la contraseña y se accede a la aplicación, y otro con el que podemos acceder a la aplicación sin la necesidad de iniciar con el usuario y contraseña, esto se hace debido a que el servidor Xampp está instalado en el ordenador por lo que de no tener acceso al servidor no se podrá acceder, para solucionar este fallo se deberá subir el servidor a la nube y así se podrá tener acceso siempre que se tenga conexión a internet.

Inicio Activity:



The screenshot shows the Android Studio interface with the code editor open to the `InicioActivity.java` file. The code defines an activity with several private fields for buttons and edit texts, and an instance of a custom controller. It overrides the `onCreate` method to set the content view and initialize the controller. A separate method `recogerValores` is used to find views by ID. The right side of the screen shows the standard Android Studio toolbars for Gradle, Device File Explorer, and Emulator.

```
1 package com.example.proyectopruebalquehago.Activities;
2
3 import ...
4
5 public class InicioActivity extends AppCompatActivity {
6     private Button botonInicio;
7     private Button botonRegistro;
8     private Button botonInicioSinConexion;
9     private EditText nombre_usuario;
10    private EditText contrasena;
11
12    private ControladorInicio controladorInicio;
13
14    @Override
15    protected void onCreate(Bundle savedInstanceState) {
16        super.onCreate(savedInstanceState);
17        setContentView(R.layout.activity_inicio_registro);
18        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
19        StrictMode.setThreadPolicy(policy);
20        controladorInicio=new ControladorInicio(this);
21        recogerValores();
22        comprobarBotones();
23    }
24
25    private void recogerValores() {
26        nombre_usuario=findViewById(R.id.editTextLogin);
27        contrasena=findViewById(R.id.editTextContrasena);
28    }
29}
```

```
40
41     private void comprobarBotones() {
42         botonInicio=findViewById(R.id.idButtonInicioSesion);
43         botonRegistro=findViewById(R.id.idButtonRegistro);
44         botonInicioSinConexion=findViewById(R.id.idButtonInicioSesionSinConexion);
45
46         botonInicio.setOnClickListener(new View.OnClickListener() {
47             @Override
48             public void onClick(View v) { iniciarApp(); }
49         });
50
51         botonRegistro.setOnClickListener(new View.OnClickListener() {
52             @Override
53             public void onClick(View v) {
54                 Toast.makeText( context: InicioActivity.this, text: "Has Pulsado Registro", Toast.LENGTH_SHORT).show();
55                 Intent intent= new Intent( packageContext: InicioActivity.this, RegistroActivity.class);
56                 startActivity(intent);
57             }
58         });
59         botonInicioSinConexion.setOnClickListener(new View.OnClickListener() {
60             @Override
61             public void onClick(View v) { iniciarAppSinConexion(); }
62         });
63     }
64
65
66
67
68
69
70     private void iniciarAppSinConexion() {
71         Intent intent= new Intent( packageContext: InicioActivity.this, AppActivity.class);
72         startActivity(intent);
73     }
74
75
76     private void iniciarApp() {
77         if (controladorInicio.comprobacionInicio(nombre_usuario.getText(),contrasena.getText())){
78             Intent intent= new Intent( packageContext: InicioActivity.this, AppActivity.class);
79             startActivity(intent);
80         }else{
81             Toast.makeText( context: InicioActivity.this, text: "Acceso Incorrecto", Toast.LENGTH_SHORT).show();
82         }
83     }
84 }
85
```

## Controlador Inicio:

```
1 package com.example.proyectopruebalquehago.Controladores;
2
3 import ...
4
5 public class ControladorInicio {
6     private InicioActivity inicio;
7     private boolean inicioCorrecto=false;
8
9
10    public ControladorInicio(InicioActivity inicio) { this.inicio = inicio; }
11
12    public boolean comprobacionInicio(Editable nombreUsuario, Editable contrasena){
13
14        inicioCorrecto=false;
15        URL url;
16        HttpURLConnection conexion = null;
17
18        try {
19            url = new URL( spec: "http://192.168.1.138/Android/AppPabloProyectoQueHago/public/login/"+nombreUsuario+"/"+contrasena);
20            System.out.println("*****"+url+"*****");
21            conexion = (HttpURLConnection) url.openConnection();
22            conexion.connect();
23
24
25            //BufferedReader br = new BufferedReader(new InputStreamReader(conexion.getInputStream()));
26            //InputStreamReader in = new InputStreamReader(conexion.getInputStream());
27            //BufferedReader br = new BufferedReader(in);
28
29        } catch (IOException e) {
30            e.printStackTrace();
31        }
32
33    }
34
35
36
37
38
39}
```

```

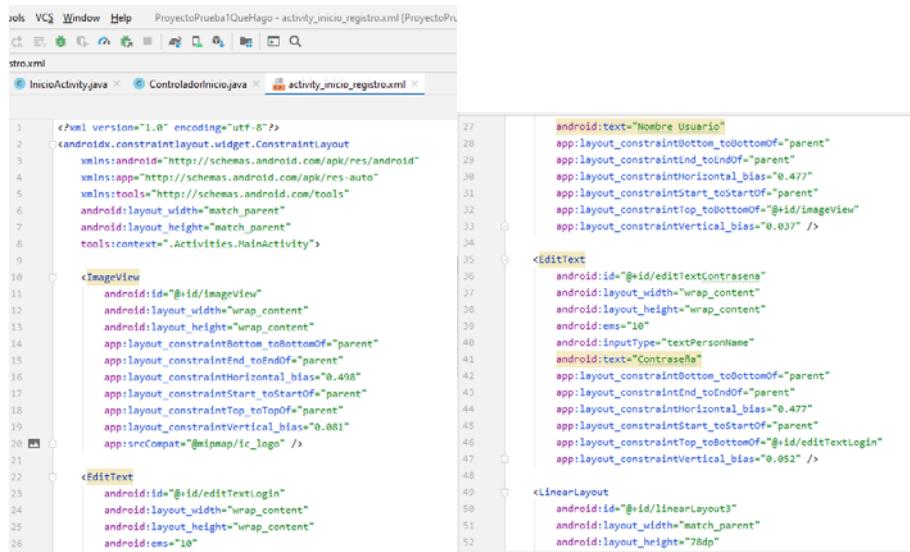
    40     String line = br.readLine();
    41     Log.d( tag: "HTTP-GET", line);
    42     if (line.equals("11")){
    43         inicioCorrecto=true;
    44     }else{
    45         inicioCorrecto=false;
    46     }
    47
    48     } catch (IOException e) {
    49         e.printStackTrace();
    50     } finally {
    51         if (conexion != null) {
    52             conexion.disconnect();
    53         }
    54     }
    55
    56
    57     return inicioCorrecto;
    58 }
    59
    60

```

Como podemos observar el método “comprobacionInicio()” se conecta con la base de datos a través de la función de `HttpUrlConection`, el cual directamente ejecuta una ruta, la cual se encarga de enviar la información necesario, además podremos ver que la ruta tiene la ip 192.168.1.138, que es la ip que tengo actualmente como usuario de una red doméstica, lo que impide la llegada o salida de información si no estoy conectado a esa red o el servidor Xampp no se encuentra activado.

Esto pasa de igual manera con el resto de funciones que depende del servicio de `HttpUrlConection` de nuestra aplicación.

### Layout Inicio:



```

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
    android:orientation="horizontal"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextContrasena"
    app:layout_constraintVertical_bias="0.135"

<Button
    android:id="@+id/idButtonIniciarSesion"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Iniciar Sesión" />

<Button
    android:id="@+id/idButtonRegistro"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Registrarse" />
</LinearLayout>
<Button
    android:id="@+id/idButtonInicioSessionSinConexion"
    android:layout_width="wrap_content"

```

Vista de Inicio desde la aplicación:



## REGISTRO

El objetivo del apartado de Registro es que el cliente pueda crear un usuario en el que establecerá el nombre y apellidos, estos deberán escribirse sin espacios y sin acentos, el nombre de usuario, email y la fecha de nacimiento, este apartado no hace comprobaciones de edad ya que los servicios a los que se pueden acceder están adaptados a todos los públicos.

Registro Activity:

```
1 package com.example.proyectopruebalquehago.Activities;
2
3 import ...
4
5
6 public class RegistroActivity extends AppCompatActivity {
7     private ControladorRegistro controladorRegistro;
8
9     private EditText et_nombre_y_apellidos;
10    private EditText et_nombre_usuario;
11    private EditText et_contrasena;
12    private EditText et_email;
13    private EditText et_fecha_nacimiento;
14
15    private Editable nombre_y_apellidos;
16    private Editable nombre_usuario;
17    private Editable contrasena;
18    private Editable email;
19    private Editable fecha_nacimiento;
20
21    private Button boton_Registrarse;
22
23    private String JsonCompleto;
24
25    @Override
26    protected void onCreate(Bundle savedInstanceState) {
27        super.onCreate(savedInstanceState);
28        setContentView(R.layout.activity_registro);
29
30        controladorRegistro=new ControladorRegistro( registroActivity: this);
31        pulsarBotonRegistro();
32        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
33    }
34
35    @Override
36    public boolean onSupportNavigateUp() {
37        onBackPressed();
38        return false;
39    }
40
41    private void pulsarBotonRegistro() {
42        boton_Registrarse = findViewById(R.id.idBotonRegistrarseRegistro);
43        boton_Registrarse.setOnClickListener(new View.OnClickListener() {
44            @Override
45            public void onClick(View v) {
46                recoger_datos();
47                controladorRegistro.enviar_BBDD(nombre_y_apellidos,nombre_usuario,contrasena,email,fecha_nacimiento);
48            }
49        });
50    }
51
52
53    private void recoger_datos() {
54        et_nombre_y_apellidos = findViewById(R.id.editTextTextNombreApellido);
55        et_nombre_usuario = findViewById(R.id.editTextTextNombreUsuario);
56        et_contrasena = findViewById(R.id.editTextTextPassword);
57        et_email = findViewById(R.id.editTextTextEmailAddress);
58        et_fecha_nacimiento = findViewById(R.id.editTextDate);
59        boton_Registrarse = findViewById(R.id.idBotonRegistrarseRegistro);
60
61        nombre_y_apellidos = et_nombre_y_apellidos.getText();
62        nombre_usuario = et_nombre_usuario.getText();
63        contrasena = et_contrasena.getText();
64        email = et_email.getText();
65        fecha_nacimiento = et_fecha_nacimiento.getText();
66    }
67
68 }
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86 }
```

Controlador Registro:

```

  package com.example.proyectoprueba1quehago.Controladores;

import ...

public class ControladorRegistro {
    private RegistroActivity registroActivity;

    public ControladorRegistro(RegistroActivity registroActivity) {
        this.registroActivity = registroActivity;
    }

    public void enviar_BBDD(Editable nombre_y_apellidos, Editable nombre_usuario, Editable contrasena, Editable email, Editable fecha_nacimiento) {
        URL url;
        HttpURLConnection conexion = null;

        try {
            url = new URL (spec: "http://192.168.1.138/Android/AppPabloProyectoQueHago/public/registro/" + nombre_y_apellidos + "/" + nombre_usuario + "/" + contrasena + "/" + email + "/" + fecha_nacimiento);
            System.out.println("*****" + url + "*****");
            conexion = (HttpURLConnection) url.openConnection();
            conexion.connect();

            InputStreamReader in = new InputStreamReader(conexion.getInputStream());
            BufferedReader br = new BufferedReader(in);

            String line = br.readLine();
            System.out.println("-----" + line + "-----");
            Log.d( tag: "HTTP-GET", line);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (conexion != null) {
                conexion.disconnect();
            }
        }
    }
}

```

## Activity Registro:

```

<?xml version="1.0" encoding="utf-8">
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="REGISTRO"
        android:textSize="36sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.076" />

    <LinearLayout
        android:id="@+id/linearLayout4"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:orientation="horizontal">

        <LinearLayout
            android:id="@+id/linearLayout2"
            android:layout_width="match_parent"
            android:layout_height="80dp"
            android:orientation="horizontal"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/linearLayout4"
            app:layout_constraintVertical_bias="0.084">

            <TextView
                android:id="@+id/textView2"
                android:layout_width="190dp"
                android:layout_height="wrap_content"
                android:text="Nombre Usuario"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toStartOf="@+id/editTextTextNombreApellido"
                app:layout_constraintHorizontal_bias="0.571"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                app:layout_constraintVertical_bias="0.212" />

            <EditText
                android:id="@+id/editTextTextNombreApellido"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginEnd="24dp"
                android:ems="7"
                android:inputType="textPersonName" />
        

```

```

105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204

```

Resource Manager L: Structure R: Variants

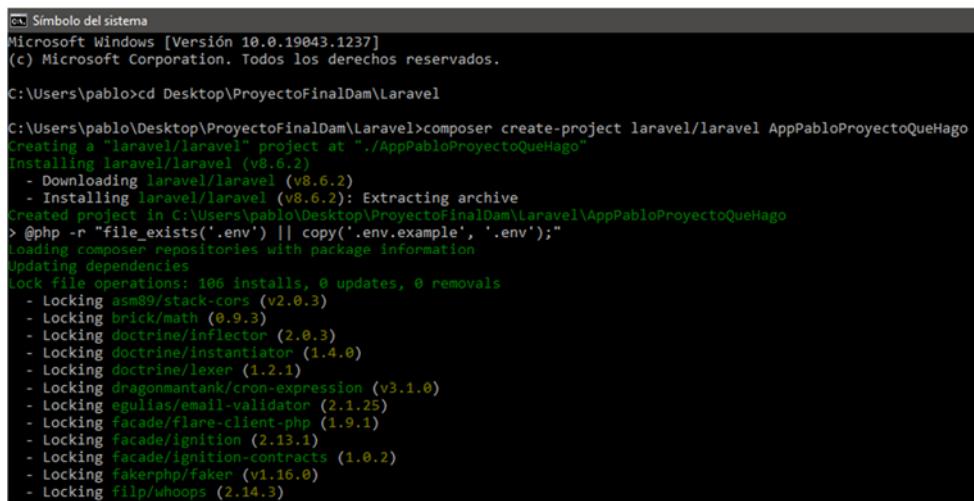
Vista de Registro desde la aplicación:



A la vez que creamos estos dos apartados debemos de crear nuestro servidor de base de datos, como ya hemos dicho se ha creado mediante la herramienta Xampp, para ello debemos de instalarlo desde la página oficial <https://www.apachefriends.org/es/index.html>.

Una vez descargado e instalado procedemos a crear nuestro repositorio o carpeta, donde se almacenarán y crearán los archivos necesarios.

Para ello deberemos abrir el CMD de Windows y colorarnos en la ubicación de “htcdocs” dentro de la carpeta de Xampp y una vez echo creamos nuestro proyecto de Laravel.



```
C:\ Símbolo del sistema
Microsoft Windows [Versión 10.0.19043.1237]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\pablo>cd Desktop\ProyectoFinalDam\Laravel

C:\Users\pablo\Desktop\ProyectoFinalDam\Laravel>composer create-project laravel/laravel AppPabloProyectoQueHago
Creating a "laravel/laravel" project at "./AppPabloProyectoQueHago"
Installing laravel/laravel (v8.6.2)
- Downloading laravel/laravel (v8.6.2)
- Installing laravel/laravel (v8.6.2): Extracting archive
created project in C:\Users\pablo\Desktop\ProyectoFinalDam\Laravel\AppPabloProyectoQueHago
> @php -r '$file_exists('.env') || copy('.env.example', '.env');'
Loading composer repositories with package information
Updating dependencies
Lock file operations: 106 installs, 0 updates, 0 removals
- Locking asmb89/stack-cors (v2.0.3)
- Locking brick/math (0.9.3)
- Locking doctrine/inflector (2.0.3)
- Locking doctrine/instantiator (1.4.0)
- Locking doctrine/lexer (1.2.1)
- Locking dragonmantank/cron-expression (v3.1.0)
- Locking egulias/email-validator (2.1.25)
- Locking facade/flare-client-php (1.9.1)
- Locking facade/ignition (2.13.1)
- Locking facade/ignition-contracts (1.0.2)
- Locking fakerphp/faker (v1.16.0)
- Locking filp/whoops (2.14.3)
```

Una vez echo este paso ya tendremos nuestro proyecto laravel que nos permitirá crear nuestra base de datos y crear la conexión con nuestra aplicación.

A continuación ubicamos el archivo “web.php” de nuestro proyecto laravel. Para configurar este proyecto se ha utilizado NetBeans.

Desde nuestro navegador nos dirigimos a la ruta “localhost/phpMyAdmin” y nos abrirá el servidor de phpMyAdmin para la base de datos, y creamos la base de datos, la cual hemos llamado “bbddproyectopabloquehago”, y ahora debemos conectarla, para ello abrimos el archivo “.env” y donde pone “DB\_DATABASE” escribimos el nombre de la base de datos “DB\_DATABASE=bbddproyectopabloquehago”.

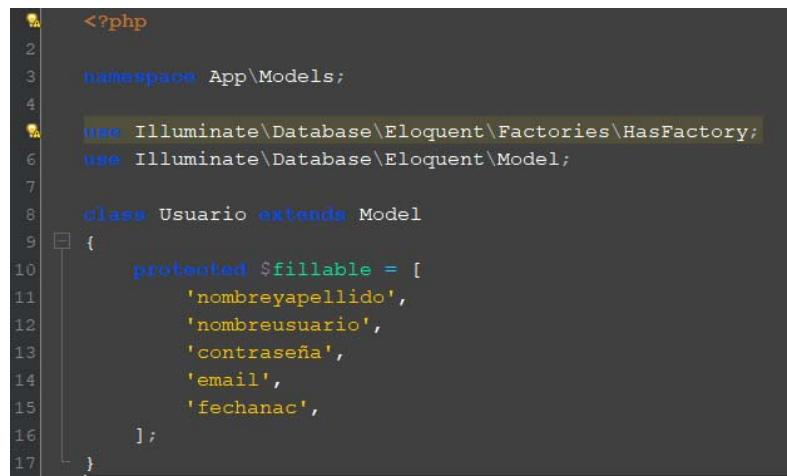
Ahora debemos de crear un controlador para gestionar las peticiones, el cual hemos llamado “PeticionController.php”, para crearlo abrimos de nuevo el CMD de Windows y nos ubicamos en el destino. Para crearlo necesitamos el comando “php artisan make:controller PeticionController”

Una vez creado le abrimos y procederemos a crear las funciones necesarios, las cuales son:

- registrarUsuario().

- logearUsuario().
- comprobarUsuario().

Antes de crear y usar estas funciones debemos de crear el modelo necesario para su utilización, el cual en este caso es el modelo de Usuario. Para crearlo abrimos de nuevo el CMD de Windows escribimos lo siguiente “php artisan make:model Usuario” y nos creara el modelo, ahora debemos adaptarlo a nuestras necesidades, por lo que queremos los mismos datos que pedimos en el apartado de Registro.

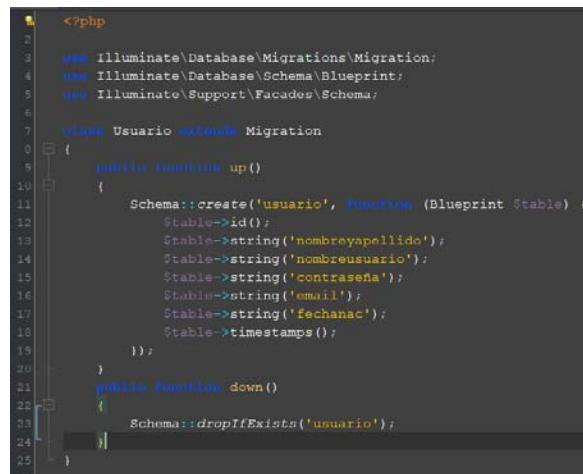


```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Usuario extends Model
9 {
10     protected $fillable = [
11         'nombreyapellido',
12         'nombreusuario',
13         'contraseña',
14         'email',
15         'fechanac',
16     ];
17 }

```

Ahora para conectarlo con la base de datos y crear la tabla debemos de crear una migración adaptada a este Usuario. Por lo que de nuevo en nuestro CMD de Windows escribimos “php artisan make:migration usuario”, y abrimos el archivo que se ha creado ubicado en la carpeta database/migrations y procedemos a modificarlo de igual manera que hemos hecho con el modelo.



```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 class Usuario extends Migration
8 {
9     public function up()
10    {
11        Schema::create('usuario', function (Blueprint $table) {
12            $table->id();
13            $table->string('nombreyapellido');
14            $table->string('nombreusuario');
15            $table->string('contraseña');
16            $table->string('email');
17            $table->string('fechanac');
18            $table->timestamps();
19        });
20    }
21    public function down()
22    {
23        Schema::dropIfExists('usuario');
24    }
25 }

```

Una vez echo debemos de ejecutar la migración, para ello utilizamos el comando “php artisan migrate” en el CMD de Windows y nos creara la tabla. Ahora ya podemos empezar a gestionar las funciones que he mencionado anteriormente.

```
function registrarUsuario($nombreyapellido,$nombreusuario,$contraseña,  
    $user =new Usuario();  
    $user->nombreyapellido=$nombreyapellido;  
    $user->nombreusuario=$nombreusuario;  
    $user->contraseña=$contraseña;  
    $user->email=$email;  
    $user->fechanac=$fechanac;  
    $user->save();  
    return "Usuario Registrado";  
}
```

```
function logearUsuario($nombreusuario,$contraseña) {  
    $usuario=Usuario::All();  
    $incorrecto="00";  
    $correcto="11";  
    $esteUsuario="00";  
    $resultado="";  
    foreach($usuario as $user){  
        if ($user->nombreusuario==$nombreusuario) {  
            if ($user->contraseña==$contraseña) {  
                $esteUsuario = $correcto;  
            }  
        }  
        if ($esteUsuario == $correcto) {  
            $resultado="11";  
        } else{  
            $resultado="00";  
        }  
        return $resultado;  
    }  
}
```

```
function comprobarUsuario($nombreusuario){  
    $usuario=Usuario::All();  
    $incorrecto="0";  
    $correcto="1";  
    $esteUsuario="0";  
    foreach($usuario as $user){  
        if ($user->nombreusuario==$nombreusuario) {  
            $esteUsuario = $correcto;  
        }  
    }  
    if ($esteUsuario == $correcto) {  
        return $correcto;  
    } else{  
        return $incorrecto;  
    }  
}
```

No debemos olvidarnos de poner la ruta del modelo:

```
namespace App\Http\Controllers;

use App\Models\Usuario;
use App\Models\Guardar;
```

Ahora que tenemos las funciones creadas, desde el archivo “web.php” vamos a crear las rutas.

```
Route::get('registro/{nombreyapellido}/{nombreusuario}/{contraseña}/{email}/{fechanac}',  
[App\Http\Controllers\PeticionController::class,'registrarUsuario']);  
  
Route::get('login/{nombreusuario}/{contraseña}',  
[App\Http\Controllers\PeticionController::class,'logearUsuario']);  
  
Route::get('comprobar/{nombreusuario}',  
[App\Http\Controllers\PeticionController::class,'comprobarUsuario']);
```

Finalizado el apartado de Inicio y de Registro y la conexión, el siguiente paso es crear la aplicación y su funcionamiento.

## App

El objetivo de este apartado es el de gestionar la aplicación en sí, la cual se encargará de recoger los servicios de ocio, para mostrarlos se deberá escoger una fecha en concreto y además se podrá distinguir entre aquellos que son por la mañana o por la tarde, solo se podrá escoger un servicio por la mañana y otro por la tarde. Además a través de este apartado se gestiona la opción de escoger servicios aleatorios y de guardar los servicios solicitados.

Los datos que ofrecen los servicios de ocio se obtendrán de la pagina de datos del portal de datos abiertos del ayuntamiento de Madrid  
“<https://datos.madrid.es/portal/site/egob/>” catálogo de datos y API, se ha decidido que sea únicamente de la ciudad de Madrid puesto que solo he podido encontrar y acceder a los datos de esta página.

## AppActivity:

```

public class AppActivity extends AppCompatActivity {
    private ControladorApp controladorApp;
    private AdaptadorPersonalizadoServicio adaptadorPersonalizadoServicio;
    private AdaptadorPersonalizadoRestaurante adaptadorPersonalizadoRestaurante;
    private GridView gridView;
    private ArrayList<String> listaJsonServicio = new ArrayList<>();
    private String distritoSeleccionado = "ARGANDUZUELA";
    private boolean gridViewVisible=false;
    private LinearLayout layoutServicio;
    private String link=""; 
    private String fechaEscogida="";
    private String fechaEscogidaDia="";
    private String fechaEscogidaMes="";
    private String fechaEscogidaAnio="";
    private ActividadCulturalExtensa actividadCulturalExtensa;
    private ImageButton botonManana;
    private ImageButton botonComida;
    private ImageButton botonTarde;
    private ImageButton botonCena;
    private Boolean primerInicio=true;
    private Boolean pantallaManana=true;
    private Boolean pantallaComida=false;
    private Boolean pantallaTarde=false;
    private Boolean pantallaCena=false;
    private String nombreRestauranteComidaSolicitado="";
    private String nombreRestauranteCenaSolicitado="";
    private String nombreServicioMananaSolicitado="";
    private String nombreServicioTardeSolicitado="";
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_app);
    gridView=findViewByIdById(R.id.gridView);
    controladorApp=new ControladorApp(getApplicationContext());
    recogerBotones();
    Toast.makeText(context, AppActivity.this, text: "POR FAVOR ESPERE UNOS SEGUNDOS HASTA QUE SE RECOZAN TODOS LOS SERVICIOS", Toast.LENGTH_LONG).show();
    Toast.makeText(context, AppActivity.this, text: "POR FAVOR SELECCIONE UNA FECHA.", Toast.LENGTH_SHORT).show();

    controladorApp.recogerTodosLosServicios();

    registerForContextMenu(gridView);
    comprobarBotones();
    if (primerInicio){
        botonManana.setImageResource(R.mipmap.ic_logo_manana_pulsado_foreground);
    }

    seleccionaServicio();
}

@Override
protected void onPause() { super.onPause(); }

private void recogerBotones() {
    botonManana=(ImageButton) findViewByIdById(R.id.idButtonManana);
    botonComida=(ImageButton) findViewByIdById(R.id.idButtonComida);
    botonTarde=(ImageButton) findViewByIdById(R.id.idButtonTarde);
    botonCena=(ImageButton) findViewByIdById(R.id.id.idButtonCena);
}

```

seleccionaDia(): recoge el valor obtenido abriendo el calendario que nos permite escoger la fecha, también se encarga de generar el propio calendario, este método se ejecutara pulsando el botón que aparece en el menú con forma de calendario.

```

public void seleccionaDia(MenuItem item) {
    primerInicio=false;
    gridViewVisible=false;
    Calendar calendario=Calendar.getInstance();
    int fecha_anio=calendario.get(calendario.YEAR);
    int fecha_mes=calendario.get(calendario.MONTH);
    int fecha_dia=calendario.get(calendario.DAY_OF_MONTH);

    DatePickerDialog dpd=new DatePickerDialog(context: AppActivity.this, new DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
            fechaEscogida = dayOfMonth + "-" + month + "-" + year;
            fechaEscogidaDia=dayOfMonth+"";
            fechaEscogidaMes=month+"";
            fechaEscogidaAnio=year+"";
            Toast.makeText(context: AppActivity.this, text: "Esta es La fecha: "+fechaEscogida, Toast.LENGTH_SHORT).show();
        }
    },fecha_anio,fecha_mes,fecha_dia);
    dpd.show();
}

```

crearGridViewAdaptado(): esta función se encarga de agregar al ArrayList de servicios que se mostraran en el GridView, aquellos que compartan la fecha que ha escogido el usuario.

```

136     private void crearGridViewAdaptado() {
137         controladorApp.vaciarListaGridView();
138         for (int i=0;i<controladorApp.getLista_AC_Extenso().size();i++){
139             String dia=controladorApp.separarDia(controladorApp.getLista_AC_Extenso().get(i).getFecha());
140             if (dia.equals(fechaEscogidaDia)){
141                 controladorApp.anadirListaGridView(controladorApp.getLista_AC_Extenso().get(i),i);
142             }
143         }
144         mostrarGridView();
145         adaptadorPersonalizadoServicio.notifyDataSetChanged();
146         controladorApp.setPopMenuPulsado(true);
147     }

```

mostrarGridView(): se encarga de mostrar el GridView con los datos que queremos, como los que se realizan por la mañana y por la tarde y así poder distinguirlos.

```

151     private void mostrarGridView(){
152         if(pantallaManana){
153             controladorApp.vaciarListaGridView_Manana();
154             for (int i=0;i<controladorApp.getLista_GridView().size();i++){
155                 if (Integer.parseInt(controladorApp.separarHora(controladorApp.getLista_GridView().get(i).getFecha()))<14 && Integer.parseInt(controladorApp.separarHora(controladorApp.getLista_GridView().get(i).getFecha()))>=10){
156                     controladorApp.anadirListaGridView_Manana(controladorApp.getLista_GridView().get(i),i);
157                 }
158             }
159             adaptadorPersonalizadoServicio=new AdaptadorPersonalizadoServicio(context: this,R.layout.list_item_layout_servicio2,controladorApp.getLista_GridView_Manana());
160             gridView.setAdapter(adaptadorPersonalizadoServicio);
161             gridViewVisible=true;
162         }
163         if(pantallaTarde){
164             controladorApp.vaciarListaGridView_Tarde();
165             for (int i=0;i<controladorApp.getLista_GridView().size();i++){
166                 if (Integer.parseInt(controladorApp.separarHora(controladorApp.getLista_GridView().get(i).getFecha()))>=14 || Integer.parseInt(controladorApp.separarHora(controladorApp.getLista_GridView().get(i).getFecha()))<=18){
167                     controladorApp.anadirListaGridView_Tarde(controladorApp.getLista_GridView().get(i),i);
168                 }
169             }
170             adaptadorPersonalizadoServicio=new AdaptadorPersonalizadoServicio(context: this,R.layout.list_item_layout_servicio2,controladorApp.getLista_GridView_Tarde());
171             gridView.setAdapter(adaptadorPersonalizadoServicio);
172             gridViewVisible=true;
173         }
174     }

```

seleccionaServicio(): Este método tiene como objetivo que al pulsar sobre cualquier servicio mostrado en el GridView, y al pulsar sobre él nos abra un popMenu que nos muestre más información sobre dicho servicio.

```

176     private void seleccionaServicio() {
177         gridView.setOnItemClickListener((parent, view, position, id) -> {
178             // TODO Auto-generated method stub
179             if (controladorApp.isServicioCulturalSeleccionado()){
180                 abrirPopMenu(position);
181             }else{
182                 abrirPopMenuRestaurante(position);
183             }
184         });
185         gridView.setOnItemLongClickListener((parent, view, position, id) -> {
186             // TODO Auto-generated method stub
187             if (controladorApp.isServicioCulturalSeleccionado()){
188                 //abrirPopMenu(position);
189             }else{
190                 //abrirPopMenuRestaurante(position);
191             }
192             return false;
193         });
194     }
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217

```

abrirPopMenu(): Se encarga de abrir el popMenu, creando el modelo y enviándole los datos necesarios.

```

199     private void abrirPopMenu(int position) {
200         Intent intent=new Intent(packageContext: AppActivity.this,PopUpActividadCultural.class);
201         if(pantallaManana){
202             intent.putExtra( name: "nombre", controladorApp.getLista_GridView_Manana().get(position).getTitulo());
203             intent.putExtra( name: "descripcion", controladorApp.getLista_GridView_Manana().get(position).getDescription());
204             intent.putExtra( name: "localizacion", controladorApp.getLista_GridView_Manana().get(position).getLocalizacion());
205             intent.putExtra( name: "precio", controladorApp.getLista_GridView_Manana().get(position).getPrecio());
206             intent.putExtra( name: "fecha", controladorApp.getLista_GridView_Manana().get(position).getFecha());
207         }
208         if (pantallaTarde){
209             intent.putExtra( name: "nombre", controladorApp.getLista_GridView_Tarde().get(position).getTitulo());
210             intent.putExtra( name: "descripcion", controladorApp.getLista_GridView_Tarde().get(position).getDescription());
211             intent.putExtra( name: "localizacion", controladorApp.getLista_GridView_Tarde().get(position).getLocalizacion());
212             intent.putExtra( name: "precio", controladorApp.getLista_GridView_Tarde().get(position).getPrecio());
213             intent.putExtra( name: "fecha", controladorApp.getLista_GridView_Tarde().get(position).getFecha());
214         }
215         startActivity(intent);
216     }
217

```

```

220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239

    //AL igual que el anterior se encarga de mostrar el gridView, pero este unicamente para los Restaurantes.
    public void mostrarGridViewComidas(ArrayList<Restaurante> listaRestaurantes) {
        adaptadorPersonalizadoRestaurante=new AdaptadorPersonalizadoRestaurante( context: this,R.layout.list_item_layout_servicio2,listaRestaurantes);
        gridView.setAdapter(adaptadorPersonalizadoRestaurante);
    }

    //Igual que el anterior pero para los restaurantes.
    private void abrirPopUpRestaurante(int position) {
        Intent intent=new Intent( packageContext: AppActivity.this,PopUpRestaurante.class);

        intent.putExtra( name: "nombre", controladorApp.getListaRestaurantes().get(position).getNombre());
        intent.putExtra( name: "localizacion", controladorApp.getListaRestaurantes().get(position).getLocalizacion());
        intent.putExtra( name: "precio", controladorApp.getListaRestaurantes().get(position).getPrecio());
        intent.putExtra( name: "distrito", controladorApp.getListaRestaurantes().get(position).getDistrito());
        intent.putExtra( name: "productos", controladorApp.getListaRestaurantes().get(position).getProductos());
        intent.putExtra( name: "puntuacion", controladorApp.getListaRestaurantes().get(position).getPuntuacion());
        startActivity(intent);
    }
}

```

`comprobarBotones()`: Se encarga constante de saber en qué pantalla queremos situarnos, si en la Mañana, comida ..., cuando abres una, esta se encarga de mostrar una imagen diferenciada y devolver a los botones su imagen original para distinguir óptimamente en qué pantalla nos encontramos.

```

244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313

private void comprobarBotones() {
    botonManana.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            pantallaManana=true;
            pantallaComida=false;
            pantallaTarde=false;
            pantallaCena=false;
            botonManana.setImageResource(R.mipmap.ic_logo_manana_pulsado_foreground);
            botonComida.setImageResource(R.mipmap.ic_logo_comida_foreground);
            botonTarde.setImageResource(R.mipmap.ic_logo_tarde1_foreground);
            botonCena.setImageResource(R.mipmap.ic_logo_cena_foreground);
            controladorApp.setServicioCulturalSeleccionado(true);
            mostrarGridView();
            //crearListViewAdaptado();
        }
    });

    botonComida.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            primerInicio=false;
            pantallaManana=false;
            pantallaComida=true;
            pantallaTarde=false;
            pantallaCena=false;
            botonManana.setImageResource(R.mipmap.ic_logo_manana1_foreground);
            botonComida.setImageResource(R.mipmap.ic_logo_comida_pulsado_foreground);
            botonTarde.setImageResource(R.mipmap.ic_logo_tarde1_foreground);
            botonCena.setImageResource(R.mipmap.ic_logo_cena_foreground);
            controladorApp.setServicioCulturalSeleccionado(false);
            controladorApp.mostrarRestaurantes();
        }
    });

    botonTarde.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            primerInicio=false;
            pantallaManana=false;
            pantallaComida=false;
            pantallaTarde=true;
            pantallaCena=false;
            botonManana.setImageResource(R.mipmap.ic_logo_manana1_foreground);
            botonComida.setImageResource(R.mipmap.ic_logo_comida_foreground);
            botonTarde.setImageResource(R.mipmap.ic_logo_tarde_pulsado_foreground);
            botonCena.setImageResource(R.mipmap.ic_logo_cena_foreground);
            controladorApp.setServicioCulturalSeleccionado(true);
            mostrarGridView();
            //crearListViewAdaptado();
        }
    });

    botonCena.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            primerInicio=false;
            pantallaManana=false;
            pantallaComida=false;
            pantallaTarde=false;
            pantallaCena=true;
            botonManana.setImageResource(R.mipmap.ic_logo_manana1_foreground);
            botonComida.setImageResource(R.mipmap.ic_logo_comida_foreground);
            botonTarde.setImageResource(R.mipmap.ic_logo_tarde1_foreground);
            botonCena.setImageResource(R.mipmap.ic_logo_cena_pulsado_foreground);
            controladorApp.setServicioCulturalSeleccionado(false);
            controladorApp.mostrarRestaurantes();
        }
    });
}

```

Otros métodos necesarios:

```

    public void mensaje(String mensaje) {
        Toast.makeText(context, AppActivity.this, mensaje, Toast.LENGTH_SHORT).show();
    }

    public String getNombreRestauranteComidaSolicitado() {
        return nombreRestauranteComidaSolicitado;
    }

    public String getNombreRestauranteCenaSolicitado() { return nombreRestauranteCenaSolicitado; }

    public String getNombreServicioMananaSolicitado() { return nombreServicioMananaSolicitado; }

    public String getNombreServicioTardeSolicitado() { return nombreServicioTardeSolicitado; }
}

```

## Controlador App:

```

    package com.example.proyectoaprendizajehecho.Controladores;

import ...

public class ControladorApp {
    private AppCompatActivity appActivity;
    private DatosServiciosActividadesCulturalesExtenso datosServiciosActividadesCulturalesExtenso;
    private DatosServiciosActividadesCulturales datosServiciosActividadesCulturales;
    private boolean servicioCulturalSeleccionado=false;
    private boolean popMenuPulsado=false;
    private String url="";
    private BBDDRestaurante bbddRestaurante;

    private ArrayList<ActividadCulturalExtensa> lista_GridView=new ArrayList<>();
    ArrayList<ActividadCulturalExtensa> lista_GridView_Manana=new ArrayList<>();
    ArrayList<ActividadCulturalExtensa> lista_GridView_Tarde=new ArrayList<>();

    private ArrayList<ActividadCulturalExtensa> lista_AC_Extensa=new ArrayList<>();
    private ArrayList<Restaurante> listaRestaurantes=new ArrayList<>();

    //Manana
    private String tituloManana="";
    private String descriptionManana="";
    private String fechaManana="";
    private String localizacionManana="";
    private String precioManana="";
    //comida
    private String nombreComida="";
    private String localizacionComida="";
    private String precioComida="";
    private String distritoComida="";
    private String productosComida="";
    private String puntuacionComida="";
    //tarde
    private String tituloTarde="";
    private String descriptionTarde="";
    private String fechaTarde="";
    private String localizacionTarde="";
    private String precioTarde="";
    //cena
    private String nombreCena="";
    private String localizacionCena="";
    private String precioCena="";
    private String distritoCena="";
    private String productosCena="";
    private String puntuacionCena="";

    public ControladorApp(AppCompatActivity appActivity) {
        this.appActivity = appActivity;
        bbddRestaurante=new BBDDRestaurante();
        recogerRestaurantes();
    }
}

```

**recogerTodosLosServicios():** Se encarga de recoger los datos ofrecidos por la API, la cual nos muestra únicamente el nombre(título) de la actividad el id(link) que nos permite abrir un nuevo archivo desde el que podemos acceder al resto de información de la actividad por lo que en este apartado recogemos todos los servicios con la información básica, creamos nuestro modelo datosServiciosActividadesCulturales que se encargara de recoger cada uno de los servicios y almacenarlos en un ArrayList y llamamos a recogerTodosLosDatosAC().

```

    public void recogerTodosLosServicios(){
        servicioCulturalSeleccionado=true;
        Servicios service= API.getAPI().create(Servicios.class);
        Call<DatosServiciosActividadesCulturales> distritoCall=service.getServicios( distrito_nombre: "");
        distritoCall.enqueue(new Callback<DatosServiciosActividadesCulturales>() {
            @Override
            public void onResponse(Call<DatosServiciosActividadesCulturales> call, Response<DatosServiciosActividadesCulturales> response) {
                datosServiciosActividadesCulturales = response.body();
                recogerTodosLosDatosAC();
            }
        });

        @Override
        public void onFailure(Call<DatosServiciosActividadesCulturales> call, Throwable t) {
            appActivity.mensaje("Error");
        }
    });
}

```

recogerTodosLosDatosAC(): Desde este método recorremos el contenido del JSON obtenido guardado en un ArrayList y llamaremos a la función llamaActividadCultural().

```
private void recogerTodosLosDatosAC(){
    for (int i=0;i<datosServiciosActividadesCulturales.getListaDatosServicios().size();i++){
        popNuevoulado=false;
        url=datosServiciosActividadesCulturales.getListaDatosServicios().get(i).getId();
        llamaActividadCultural();
    }
}
```

llamaActividadCultural(): Esta función se encarga de recoger los datos extensos que podemos encontrar abriendo el id(link), y así poder recoger el total de la información de cada actividad y almacenarla en un ArrayList de un nuevo modelo que consistirá en la Actividad Cultural Extensa.

```
private void llamaActividadCultural() {
    ActividadCulturalService service=API.getApi().create(ActividadCulturalService.class);
    Call<DatosServiciosActividadesCulturalesExtenso> actividadCulturalExtenso=service.getActividadCulturalExtenso(link_actividad_cultural:url+"");
    actividadCulturalExtenso.enqueue(new Callback<DatosServiciosActividadesCulturalesExtenso>() {
        @Override
        public void onResponse(Call<DatosServiciosActividadesCulturalesExtenso> call, Response<DatosServiciosActividadesCulturalesExtenso> response) {
            datosServiciosActividadesCulturalesExtenso = response.body();
            lista_AC_Extenso.add(datosServiciosActividadesCulturalesExtenso.get(0));
        }

        @Override
        public void onFailure(Call<DatosServiciosActividadesCulturalesExtenso> call, Throwable t) {
            appActivity.mensaje("Error");
        }
    });
}
```

separarHora(): Tanto este método como el de separarDia sirven para separar los valores de Dia y Hora.

```
public String separarHora(String fecha) {
    String fechadia="";
    String fechahora="";
    String[] listaFecha = fecha.split( regex: " " );
    fechadia = listaFecha[0];
    fechahora = listaFecha[1];

    String fechahoralimpia="";

    String[] listaFecha2 = fechahora.split( regex: ":" );
    fechahoralimpia = listaFecha2[0];
    return fechahoralimpia;
}

public String separarDia(String fecha) {
    String fechadia="";
    String fechahora="";
    String[] listaFecha = fecha.split( regex: " " );
    fechadia = listaFecha[0];
    fechahora = listaFecha[1];

    String fechadialimpia="";

    String[] listaFecha2 = fechadia.split( regex: "-" );
    fechadialimpia = listaFecha2[2];
    return fechadialimpia;
}
```

recogerRestaurantes(): Este método recoge los datos de los restaurantes que están creados de manera manual en la clase BBDDRestaurantes.

```
private void recogerRestaurantes(){
    listaRestaurantes=bbddRestaurantes.getListaRestaurantes();
}

public void mostrarRestaurantes() { appActivity.mostrarGridViewComidas(listaRestaurantes); }
```

recogerValoresGuardar(): Recoger Valores Guardar sirve para una vez tengamos los servicios solicitados este elimine los caracteres inválidos (',(),/,...) que no se pueden añadir en la barra de búsqueda a la hora de guardar todos los servicios seleccionados en la base de datos.

```

  public void recogerValoresGuardar() {
    if (appActivity.getNombreServicioMananaSolicitado()!=null){
      for (int i=0;i<lista_GridView.size();i++){
        if (lista_GridView.get(i).getTitulo().equals(appActivity.getNombreServicioMananaSolicitado())){

          String resultado=lista_GridView.get(i).getTitulo();
          if (resultado.equals("") && resultado!=null){
            tituloManana=resultado.replaceAll( regex: "\\\s+", replacement: "_" );
            resultado=tituloManana;
            tituloManana=resultado.replaceAll( regex: "/", replacement: "_" );
            resultado=tituloManana;
            tituloManana=resultado.replaceAll( regex: "\\", replacement: "_" );
            resultado=tituloManana;
            tituloManana=resultado.replaceAll( regex: "\\\\", replacement: "_" );
            resultado=tituloManana;
            tituloManana=resultado.replaceAll( regex: "\\\\", replacement: "_" );
            resultado=Normalizer.normalize(tituloManana, Normalizer.Form.NFD);
            tituloManana = resultado.replaceAll( regex: "[\u0301{InCombiningDiacriticalMarks}]", replacement: "" );
          }else{
            tituloManana="nada";
          }

          resultado=lista_GridView.get(i).getDescription();
          if (resultado.equals("") && resultado!=null){...}else{
            descripcionManana="nada";
          }

          resultado=lista_GridView.get(i).getFecha();
          if (resultado.equals("") && resultado!=null){...}else{
            fechaManana="nada";
          }

          String resultado2=lista_GridView.get(i).getLocalizacion();
          if (resultado.equals("") && resultado!=null){...}else{
            localizacionManana="nada";
          }
          if (localizacionManana.equals("")){
            localizacionManana="nada";
          }

          resultado=lista_GridView.get(i).getPrecio();
          if (resultado.equals("") && resultado!=null){...}else{
            precioManana="nada";
          }
        }
      }
    }
    if (appActivity.getNombreRestauranteComidaSolicitado()!=null){
      for (int i=0;i<listaRestaurantes.size();i++){
        if (listaRestaurantes.get(i).getNombre().equals(appActivity.getNombreRestauranteComidaSolicitado())){

          String resultado=listaRestaurantes.get(i).getNombre();
          if (resultado.equals("") && resultado!=null){
            nombreComida=resultado.replaceAll( regex: "\\\s+", replacement: "_" );
            resultado=nombreComida;
            nombreComida=resultado.replaceAll( regex: "/", replacement: "_" );
            resultado=nombreComida;
            nombreComida=resultado.replaceAll( regex: "\\", replacement: "_" );
            resultado=nombreComida;
            nombreComida=resultado.replaceAll( regex: "\\\\", replacement: "_" );
            resultado=nombreComida;
            nombreComida=resultado.replaceAll( regex: "\\\\", replacement: "_" );
            resultado=Normalizer.normalize(nombreComida, Normalizer.Form.NFD);
            nombreComida = resultado.replaceAll( regex: "[\u0301{InCombiningDiacriticalMarks}]", replacement: "" );
          }else{
            nombreComida="nada";
          }

          resultado=listaRestaurantes.get(i).getLocalizacion();
          if (resultado.equals("") && resultado!=null){...}else{
            localizacionComida="nada";
          }

          resultado=listaRestaurantes.get(i).getPrecio();
          if (resultado.equals("") && resultado!=null){...}else{
            precioComida="nada";
          }

          resultado=listaRestaurantes.get(i).getDistrito();
          if (resultado.equals("") && resultado!=null){...}else{
            distritoComida="nada";
          }

          resultado=listaRestaurantes.get(i).getProductos();
          if (resultado.equals("") && resultado!=null){...}else{
            productosComida="nada";
          }

          resultado=listaRestaurantes.get(i).getProductos();
          if (resultado.equals("") && resultado!=null){...}else{
            puntuacionComida="nada";
          }
        }
      }
    }
  }
}

```

```

        }

        if (appActivity.getNombreServicioTardeSolicitado()!=null){
            for (int i=0;i<lista_GridView.size();i++){
                if (lista_GridView.get(i).getTitulo().equals(appActivity.getNombreServicioTardeSolicitado())){

                    String resultado=lista_GridView.get(i).getTitulo();
                    if (resultado.equals("") && resultado!=null){
                        tituloTarde=resultado.replaceAll( regex: "\\\s+", replacement: "_");
                        resultado=tituloTarde;
                        tituloTarde=resultado.replaceAll( regex: "/", replacement: "_");
                        resultado=tituloTarde;
                        tituloTarde=resultado.replaceAll( regex: "\\", replacement: "_");
                        resultado=tituloTarde;
                        tituloTarde=resultado.replaceAll( regex: "\\\\", replacement: "_");
                        resultado=tituloTarde;
                        tituloTarde=resultado.replaceAll( regex: "\\\?", replacement: ".");
                        resultado = Normalizer.normalize(tituloTarde, Normalizer.Form.NFD);
                        tituloTarde = resultado.replaceAll( regex: "[\\p{InCombiningDiacriticalMarks}]", replacement: "");
                    }else{
                        tituloTarde="nada";
                    }

                    resultado=lista_GridView.get(i).getDescription();
                    if (resultado.equals("") && resultado!=null){...}else{
                        descriptionTarde="nada";
                    }

                    resultado=lista_GridView.get(i).getFecha();
                    if (resultado.equals("") && resultado!=null){...}else{
                        fechaTarde="nada";
                    }

                    String resultado2=lista_GridView.get(i).getLocalizacion();
                    if (resultado.equals("") && resultado!=null){...}else{
                        localizacionTarde="nada";
                    }
                    if (localizacionTarde.equals("")){
                        localizacionTarde="nada";
                    }

                    resultado=lista_GridView.get(i).getPrecio();
                    if (resultado.equals("") && resultado!=null){...}else{
                        precioTarde="nada";
                    }
                }
            }
        }

        if (appActivity.getNombreRestauranteCenaSolicitado()!=null){
            for (int i=0;i<listaRestaurantes.size();i++){
                if (listaRestaurantes.get(i).getNombre().equals(appActivity.getNombreRestauranteCenaSolicitado())){

                    String resultado=listaRestaurantes.get(i).getNombre();
                    if (resultado.equals("") && resultado!=null){
                        nombreCena=resultado.replaceAll( regex: "\\\s+", replacement: "_");
                        resultado=nombreCena;
                        nombreCena=resultado.replaceAll( regex: "/", replacement: "_");
                        resultado=nombreCena;
                        nombreCena=resultado.replaceAll( regex: "\\", replacement: "_");
                        resultado=nombreCena;
                        nombreCena=resultado.replaceAll( regex: "\\\\", replacement: "_");
                        resultado=nombreCena;
                        nombreCena=resultado.replaceAll( regex: "\\\?", replacement: ".");
                        resultado = Normalizer.normalize(nombreCena, Normalizer.Form.NFD);
                        nombreCena = resultado.replaceAll( regex: "[\\p{InCombiningDiacriticalMarks}]", replacement: "");
                    }else{
                        nombreCena="nada";
                    }

                    resultado=listaRestaurantes.get(i).getLocalizacion();
                    if (resultado.equals("") && resultado!=null){...}else{
                        localizacionCena="nada";
                    }

                    resultado=listaRestaurantes.get(i).getPrecio();
                    if (resultado.equals("") && resultado!=null){...}else{
                        precioCena="nada";
                    }

                    resultado=listaRestaurantes.get(i).getDistrito();
                    if (resultado.equals("") && resultado!=null){...}else{
                        distritoCena="nada";
                    }

                    resultado=listaRestaurantes.get(i).getProductos();
                    if (resultado.equals("") && resultado!=null){...}else{
                        productosCena="nada";
                    }

                    resultado=listaRestaurantes.get(i).getProductos();
                    if (resultado.equals("") && resultado!=null){...}else{
                        puntuacionCena="nada";
                    }
                }
            }
        }
    }
}

```

Otras funciones necesarias, así como los getter y setter:

```

public void vaciarListaGridView() { lista_GridView.clear(); }

public void anadirListaGridView(ActividadCulturalExtensa actividadCulturalExtensa, int i) {
    lista_GridView.add(lista_AC_Extenса.get(i));
}

public void vaciarListaGridView_Manana() { lista_GridView_Manana.clear(); }

public void anadirListaGridView_Manana(ActividadCulturalExtensa actividadCulturalExtensa, int i) {
    lista_GridView_Manana.add(lista_GridView.get(i));
}

public void anadirListaGridView_Tarde(ActividadCulturalExtensa actividadCulturalExtensa, int i) {
    lista_GridView_Tarde.add(lista_GridView.get(i));
}

```

## Activity App:

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/gridview">

    <GridView
        android:id="@+id/idGridView"
        android:layout_width="394dp"
        android:layout_height="588dp"
        android:layout_marginLeft="10dp"
        android:columnWidth="90dp"
        android:gravity="center"
        android:horizontalSpacing="10dp"
        android:numColumns="2"
        android:verticalSpacing="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.43" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:background="@color/ColorSecondaryText"
        android:orientation="horizontal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayoutServicio2">

        <ImageButton
            android:id="@+id/buttonManana"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@color/ColorSecondaryText"
            app:srcCompat="@mipmap/ic_logo_manana_foreground" />

        <ImageButton
            android:id="@+id/buttonComida"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@color/ColorSecondaryText"
            app:srcCompat="@mipmap/ic_logo_comida_foreground" />

        <ImageButton
            android:id="@+id/buttonTarde"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@color/ColorSecondaryText"
            app:srcCompat="@mipmap/ic_logo_tarde_foreground" />

    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>

```

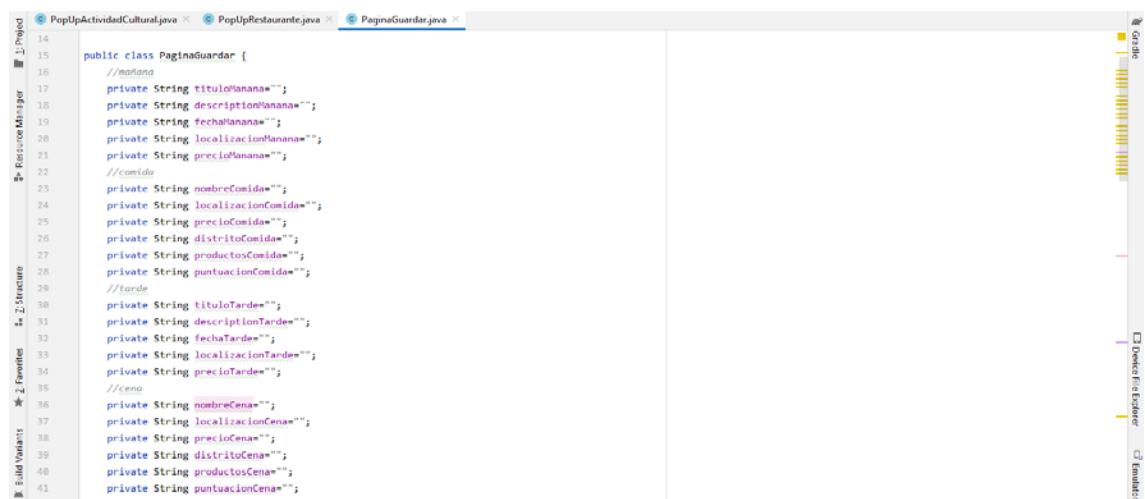
En resumen lo que hace este apartado es una vez se inicia se recogen los resultados de la API, una vez se han cargado los datos, lo cual tarda un poco de tiempo escogeremos una fecha, una vez seleccionada la fecha podremos pulsar sobre el botón de actualizar y podremos visualizar si hay servicios que se mostraran en el GridView de la pantalla de mañana ya que es la primera que aparece por defecto, luego dependiendo de que botón pulsemos, es decir si escogemos por mañana, comida, tarde o cena podremos visualizar el contenido.

Los servicios de comida y cena no se recogen de ninguna API, estos se han escrito manualmente en la clase de BBDDRestaurantes, ya que no he encontrado ninguna API ni servicio que pueda aportarlos.

Ahora que ya vemos los servicios y los restaurantes si pulsamos sobre el que queramos nos aparecerá una nueva ventana pequeña en la que podremos ver más información acerca de dicho servicio o restaurante, y si mantenemos una pulsación sobre él nos aparecerá el menú de solicitar, el cual si pulsamos agregará dicho servicio o restaurante.

También podemos encontrar en el menú el botón de 3 puntos que pulsando sobre el encontramos “Automático” y “Guardar”, si pulsamos sobre automático selecciona los servicios de manera automática en la fecha que seleccionamos anteriormente, los restaurantes no dependen de la fecha escogida, y por último si pulsamos sobre guardar nos guardara en nuestra base de datos los servicios y restaurantes escogidos, debemos de escoger un servicio y un restaurante en cada pantalla para que se guarde.

#### Página Guardar:



The screenshot shows a Java code editor with the file `PaginaGuardar.java` open. The code defines a class `PaginaGuardar` with private fields for breakfast, lunch, and dinner services. The breakfast section includes fields for title, description, date, location, and price. The lunch section includes fields for name, location, price, district, products, and rating. The dinner section includes fields for name, location, price, district, products, and rating. The code is annotated with comments indicating the sections: `//desayuno`, `//comida`, and `//tarde`.

```
14
15 public class PaginaGuardar {
16     //desayuno
17     private String tituloManana="";
18     private String descripcionManana="";
19     private String fechaManana="";
20     private String localizacionManana="";
21     private String precioManana="";
22     //comida
23     private String nombreComida="";
24     private String localizacionComida="";
25     private String precioComida="";
26     private String distritoComida="";
27     private String productosComida="";
28     private String puntuacionComida="";
29     //tarde
30     private String tituloTarde="";
31     private String descripcionTarde="";
32     private String fechaTarde="";
33     private String localizacionTarde="";
34     private String precioTarde="";
35     //cena
36     private String nombreCena="";
37     private String localizacionCena="";
38     private String precioCena="";
39     private String distritoCena="";
40     private String productosCena="";
41     private String puntuacionCena="";
```

```

    public PaginaGuardar(String tituloManana, String descripcionManana, String fechaManana, String localizacionManana, String precioManana, String nombreComida, String localizacionComida) {
        this.tituloManana = tituloManana;
        this.descripcionManana = descripcionManana;
        this.fechaManana = fechaManana;
        this.localizacionManana = localizacionManana;
        this.precioManana = precioManana;
        this.nombreComida = nombreComida;
        this.localizacionComida = localizacionComida;
        this.precioComida = precioComida;
        this.distritoComida = distritoComida;
        this.productosComida = productosComida;
        this.puntuacionComida = puntuacionComida;
        this.tituloTarde = tituloTarde;
        this.descripcionTarde = descripcionTarde;
        this.fechaTarde = fechaTarde;
        this.localizacionTarde = localizacionTarde;
        this.precioTarde = precioTarde;
        this.nombreCena = nombreCena;
        this.localizacionCena = localizacionCena;
        this.precioCena = precioCena;
        this.distritoCena = distritoCena;
        this.productosCena = productosCena;
        this.puntuacionCena = puntuacionCena;

        comprobacionInicio();
    }

    private void comprobacionInicio(){
        URL url;
        HttpURLConnection conexion = null;
        try {
            url = new URL("http://192.168.1.138/Android/AppPabloProyectoQueMago/public/guardar?"+tituloManana +"/"+
                    descripcionManana +"/"+fechaManana +"/"+localizacionManana +"/"+precioManana +"/"+nombreComida +"/"+
                    localizacionComida +"/"+precioComida +"/"+distritoComida +"/"+productosComida +"/"+puntuacionComida +"/"+
                    tituloTarde +"/"+descripcionTarde +"/"+fechaTarde +"/"+localizacionTarde +"/"+precioTarde +"/"+
                    nombreCena +"/"+localizacionCena +"/"+precioCena +"/"+distritoCena +"/"+productosCena +"/"+puntuacionCena);
            System.out.println("*****url*****");
            conexion = (HttpURLConnection) url.openConnection();
            conexion.connect();
            InputStreamReader in = new InputStreamReader(conexion.getInputStream());
            BufferedReader br = new BufferedReader(in);
            String line = br.readLine();
            System.out.println("-----"+line+"-----");
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (conexion != null) {
                conexion.disconnect();
            }
        }
    }
}

```

### PopUpActividadCultural:

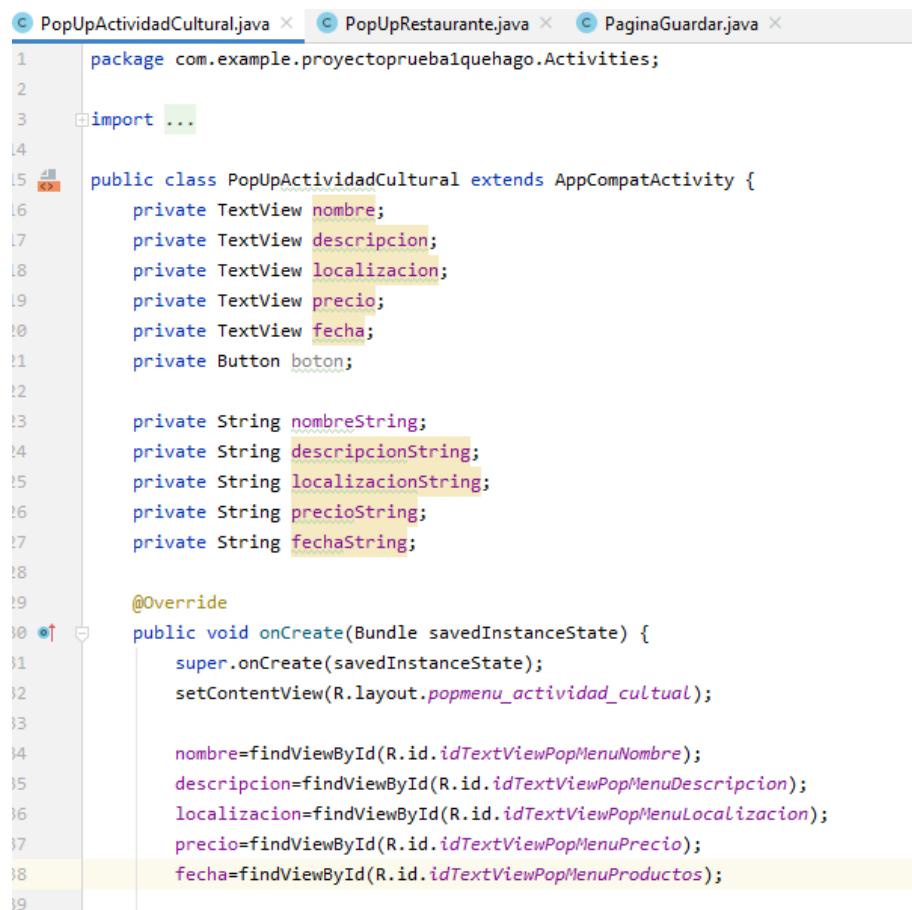
Puesto que esta vista no se encarga de realizar ninguna operación ni ninguna función de administración se ha decidido no crear un controlador ya que únicamente es para la visualización.

Para crear esta ventana y que se muestre en pequeño sobre la de App se ha utilizado la función de DisplayMetrics para generar la ventana y deberemos darle un tamaño que corresponde con un porcentaje del tamaño normal de ancho y alto de la pantalla.

Echo esto debemos de crear un nuevo estilo para que el fondo que no cubre el popMenu sea transparente, para ello abrimos el apartado styles.xml que está dentro de la carpeta de values. Y a continuación se lo agregamos a nuestro pop menu en el AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTeme.TemaPopUp" parent="Theme.AppCompat.Light.NoActionBar">
        <item name="android:windowIsFloating">true</item>
        <item name="android:windowIsTranslucent">true</item>
        <item name="android:windowCloseOnTouchOutside">true</item>
    </style>
</resources>

<activity android:name=".Activities.PopUpActividadCultural"
    android:theme="@style/AppTeme.TemaPopUp"/>
<activity android:name=".Activities.PopUpRestaurante"
    android:theme="@style/AppTeme.TemaPopUp"/>
```



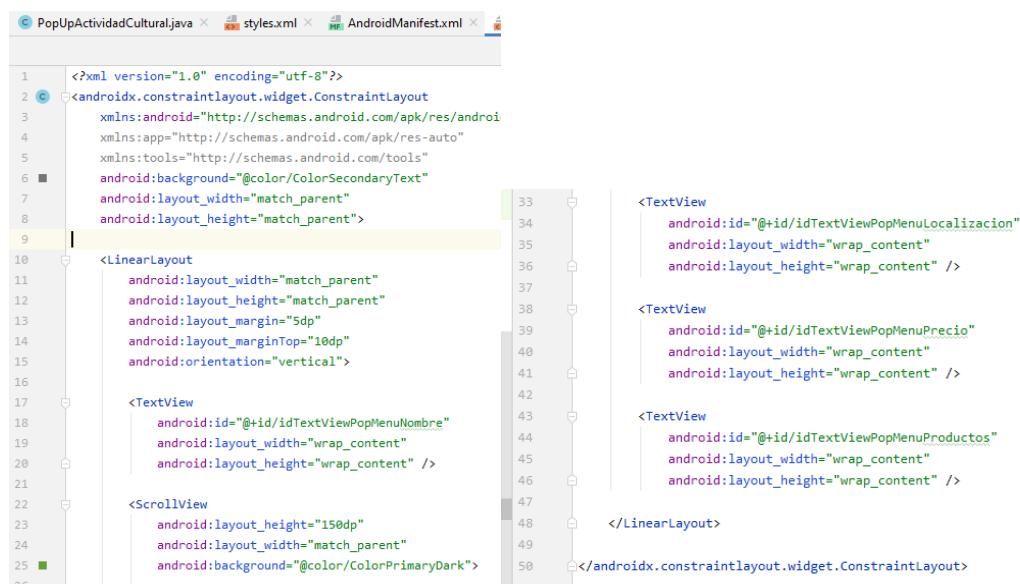
```
PopUpActividadCultural.java × PopUpRestaurante.java × PaginaGuardar.java ×
1 package com.example.proyectoprueba1quehago.Activities;
2
3 import ...
4
5 public class PopUpActividadCultural extends AppCompatActivity {
6     private TextView nombre;
7     private TextView descripcion;
8     private TextView localizacion;
9     private TextView precio;
10    private TextView fecha;
11    private Button boton;
12
13    private String nombreString;
14    private String descripcionString;
15    private String localizacionString;
16    private String precioString;
17    private String fechaString;
18
19    @Override
20    public void onCreate(Bundle savedInstanceState) {
21        super.onCreate(savedInstanceState);
22        setContentView(R.layout.popmenu_actividad_cultual);
23
24        nombre=findViewById(R.id.idTextViewPopMenuNombre);
25        descripcion=findViewById(R.id.idTextViewPopMenuDescripcion);
26        localizacion=findViewById(R.id.idTextViewPopMenuLocalizacion);
27        precio=findViewById(R.id.idTextViewPopMenuPrecio);
28        fecha=findViewById(R.id.idTextViewPopMenuProductos);
29    }
30}
```

```

41
42     Bundle bundle=getIntent().getExtras();
43     if (bundle!=null && bundle.getString( key: "nombre")!=null){
44         nombreString=bundle.getString( key: "nombre");
45         descripcionString=bundle.getString( key: "descripcion");
46         localizacionString=bundle.getString( key: "localizacion");
47         precioString=bundle.getString( key: "precio");
48         fechaString=bundle.getString( key: "fecha");
49
50         if (precioString=="1"){
51             precioString="GRATIS";
52         }
53
54         nombre.setText(nombreString);
55         descripcion.setText(descripcionString);
56         precio.setText(precioString);
57         localizacion.setText(localizacionString);
58         fecha.setText(fechaString);
59
60     }else{
61         Toast.makeText( context: PopUpActividadCultural.this, text: "ERROR POPMENU", Toast.LENGTH_SHORT).show();
62     }
63
64     DisplayMetrics medidasVentana = new DisplayMetrics();
65     getWindowManager().getDefaultDisplay().getMetrics(meidasVentana);
66
67     int ancho=medidasVentana.widthPixels;
68     int alto=medidasVentana.heightPixels;
69
70     getWindow().setLayout((int) (ancho*0.65),(int) (alto*0.45));
71
72     public void solicitarActividadCultural(View view) {
73         Intent intent=new Intent( packageContext: PopUpActividadCultural.this, AppActivity.class);
74         intent.putExtra( name: "nombreServicioSolicitado",nombreString);
75         startActivity(intent);
76     }
77
78     public void regresarApp(View view) {
79         Intent intent=new Intent( packageContext: PopUpActividadCultural.this, AppActivity.class);
80         startActivity(intent);
81     }
82 }
83

```

### PopMenuActividadCulturual Layout:



```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:background="@color/ColorSecondaryText"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent">
9
10     <LinearLayout
11         android:layout_width="match_parent"
12         android:layout_height="match_parent"
13         android:layout_margin="5dp"
14         android:layout_marginTop="10dp"
15         android:orientation="vertical">
16
17         <TextView
18             android:id="@+id/idTextViewPopMenuNombre"
19             android:layout_width="wrap_content"
20             android:layout_height="wrap_content" />
21
22         <ScrollView
23             android:layout_height="150dp"
24             android:layout_width="match_parent"
25             android:background="@color/ColorPrimaryDark">
26
27             <Textview
28                 android:id="@+id/idTextViewPopMenuLocalizacion"
29                 android:layout_width="wrap_content"
30                 android:layout_height="wrap_content" />
31
32             <Textview
33                 android:id="@+id/idTextViewPopMenuPrecio"
34                 android:layout_width="wrap_content"
35                 android:layout_height="wrap_content" />
36
37             <Textview
38                 android:id="@+id/idTextViewPopMenuProductos"
39                 android:layout_width="wrap_content"
40                 android:layout_height="wrap_content" />
41
42         </ScrollView>
43     </LinearLayout>
44
45 </androidx.constraintlayout.widget.ConstraintLayout>
46
47
48
49
50

```

## PopUpRestaurante:

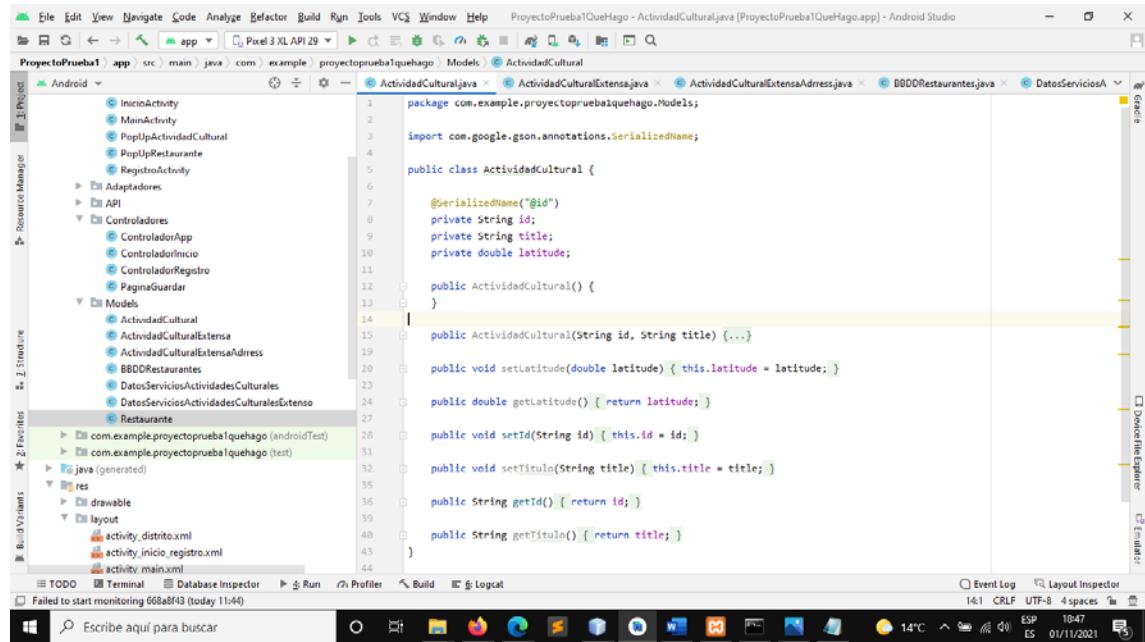
```
1 package com.example.proyecto prueba que hago.Activities;
2
3 import ...
4
5 public class PopUpRestaurante extends AppCompatActivity {
6     private TextView nombre;
7     private TextView localizacion;
8     private TextView precio;
9     private TextView distrito;
10    private TextView productos;
11    private TextView puntuacion;
12    private String nombreString;
13    private String localizacionString;
14    private String distritoString;
15    private String precioString;
16    private String productosString;
17    private String puntuacionString;
18
19    @Override
20    protected void onCreate(Bundle savedInstanceState) {
21        super.onCreate(savedInstanceState);
22        setContentView(R.layout.popmenu_restaurante);
23
24        nombre=findViewById(R.id.idTextViewPopMenuNombreRes);
25        localizacion=findViewById(R.id.idTextViewPopMenuLocalizacionRes);
26        distrito=findViewById(R.id.idTextViewPopMenuNombreDistritoRes);
27        precio=findViewById(R.id.idTextViewPopMenuPrecioRes);
28        productos=findViewById(R.id.idTextViewPopMenuProductosRes);
29        puntuacion=findViewById(R.id.idTextViewPopMenuPuntuacionRes);
30
31
32        Bundle bundle=getIntent().getExtras();
33        if (bundle!=null && bundle.getString( key: "nombre")!=null){
34            nombreString=bundle.getString( key: "nombre");
35            localizacionString=bundle.getString( key: "localizacion");
36            distritoString=bundle.getString( key: "distrito");
37            precioString=bundle.getString( key: "precio");
38            productosString=bundle.getString( key: "productos");
39            puntuacionString=bundle.getString( key: "puntuacion");
40
41            nombre.setText("Nombre: "+nombreString);
42            localizacion.setText("Localizacion: "+localizacionString);
43            distrito.setText("Distrito: "+distritoString);
44            precio.setText("Precio: "+precioString);
45            productos.setText("Productos: "+productosString);
46            puntuacion.setText("Puntuacion: "+puntuacionString);
47
48        }else{
49            Toast.makeText( context: PopUpRestaurante.this, text: "ERROR PORMENU", Toast.LENGTH_SHORT).show();
50        }
51
52        DisplayMetrics medidasVentana = new DisplayMetrics();
53        getWindowManager().getDefaultDisplay().getMetrics(meidasVentana);
54
55        int anchomedidasVentana.widthPixels;
56        int alto=medidasVentana.heightPixels;
57
58        getWindow().setLayout((int) (ancho*0.65),(int) (alto*0.45));
59    }
60
61 }
```

## PopUpRestaurante Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:background="@color/ColorSecondaryText"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="5dp"
    android:layout_marginTop="10dp"
    android:orientation="vertical">
<TextView
    android:id="@+id/idTextViewPopMenuNombreRes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/idTextViewPopMenuLocalizacionRes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/idTextViewPopMenuPrecioRes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/idTextViewPopMenuNombreDistritoRes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/idTextViewPopMenuProductosRes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/idTextViewPopMenuPuntuacionRes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## Modelos utilizados:

### Actividad Cultural:



```
package com.example.proyectoprueba1quehago.Models;

import com.google.gson.annotations.SerializedName;

public class ActividadCultural {

    @SerializedName("@id")
    private String id;
    private String title;
    private double latitude;

    public ActividadCultural() {
    }

    public ActividadCultural(String id, String title) {
    }

    public void setLatitude(double latitude) {
        this.latitude = latitude;
    }

    public double getLatitude() {
        return latitude;
    }

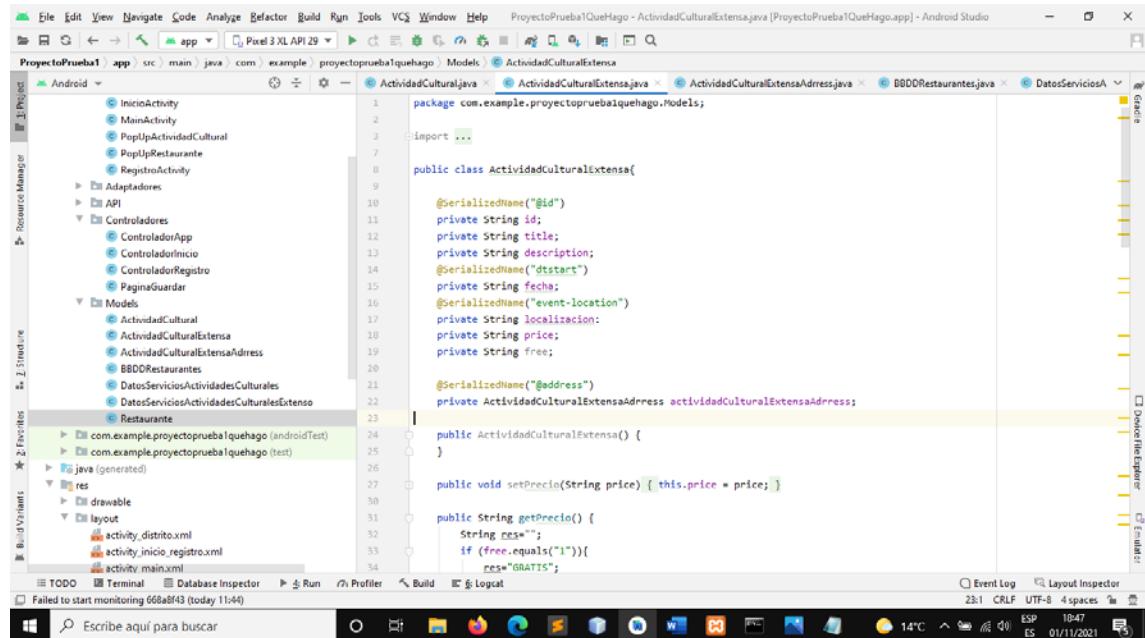
    public void setId(String id) {
        this.id = id;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }
}
```

### Actividad Cultural Extensa:



```
package com.example.proyectoprueba1quehago.Models;

import ...;

public class ActividadCulturalExtensa {

    @SerializedName("@id")
    private String id;
    private String title;
    private String description;
    @SerializedName("dtstart")
    private String fecha;
    @SerializedName("event-location")
    private String localizacion;
    private String price;
    private String free;

    @SerializedName("address")
    private ActividadCulturalExtensaAddress actividadCulturalExtensaAddress;

    public ActividadCulturalExtensa() {
    }

    public void setPrecio(String price) {
        this.price = price;
    }

    public String getPrecio() {
        String res="";
        if (free.equals("1")){
            res="GRATIS";
        }
        return res;
    }
}
```

```
public String getPrecio() {
    String res="";
    if (free.equals("1")){
        res="GRATIS";
    }else {
        res=price;
    }
    return res;
}

public void setLocalizacion(String localizacion) { this.localizacion = localizacion; }

public String getLocalizacion() {
    String res="";
    if (localizacion!=null) {
        res=localizacion;
    }else{
        res="";
    }
    return res;
}

public void setFecha(String fecha) { this.fecha = fecha; }

public String getFecha() { return fecha; }

public ActividadCulturalExtensa(String id, String title) {
    this.id = id;
    this.title = title;
}
```

```
public void setId(String id) { this.id = id; }

public void setTitle(String title) { this.title = title; }

public String getId() { return id; }

public String getTitle() { return title; }

public void setDescription(String description) { this.description = description; }

public String getDescription() {
    String res="";
    if (description!=null){
        res=description;
    }else{
        res="";
    }
    return res;
}

public void setActividadCulturalExtensaAddress(ActividadCulturalExtensaAddress actividadCulturalExtensaAddress) {
    this.actividadCulturalExtensaAddress = actividadCulturalExtensaAddress;
}

public ActividadCulturalExtensaAddress getActividadCulturalExtensaAddress() {
    return actividadCulturalExtensaAddress;
}

public static ActividadCulturalExtensaAddress parseJson(JSONObject response) {
    return null;
}
```

```
String res="";
        if (description==null){
            res=description;
        }else{
            res="";
        }
        return res;
    }

    public void setActividadCulturalExtensaAddress(ActividadCulturalExtensaAddress actividadCulturalExtensaAddress) {
        this.actividadCulturalExtensaAddress = actividadCulturalExtensaAddress;
    }

    public ActividadCulturalExtensaAddress getActividadCulturalExtensaAddress() {
        return actividadCulturalExtensaAddress;
    }

    public static ActividadCulturalExtensaAddress parseJson(String response){
        Gson gson=new GsonBuilder().create();
        ActividadCulturalExtensaAddress datos=gson.fromJson(response,ActividadCulturalExtensaAddress.class);
        return datos;
    }
}
```

## BBDDRestaurantes:

```
package com.example.proyectoquehago.Models;

import ...;

public class BBDDRestaurantes {
    ArrayList<Restaurante> listaRestaurantes;

    public BBDDRestaurantes() {
        listaRestaurantes=new ArrayList<>();
        llenarLista();
    }

    private void llenarLista() {
        //Restaurante Centro
        Restaurante restaurante;
        restaurante=new Restaurante( nombre: "Pandino", localizacion: "Justicia", precio: "", distrito: "Centro", productos: "carne q");
        listaRestaurantes.add(restaurante);
        restaurante=new Restaurante( nombre: "El Tormo", localizacion: "Palacio", precio: "30-45 euros", distrito: "Centro", productos: "paella");
        listaRestaurantes.add(restaurante);
        restaurante=new Restaurante( nombre: "Maridaje Divino", localizacion: "Palacio", precio: "-20 euros", distrito: "Centro", productos: "bebidas");
        listaRestaurantes.add(restaurante);
        restaurante=new Restaurante( nombre: "Livin'Japan", localizacion: "Embarcadores", precio: "", distrito: "Centro", productos: "japones");
        listaRestaurantes.add(restaurante);
        restaurante=new Restaurante( nombre: "Acuarela Bistró Bar", localizacion: "Palacio", precio: "", distrito: "Centro", productos: "bebidas");
        listaRestaurantes.add(restaurante);
        restaurante=new Restaurante( nombre: "La Esquina Del Real", localizacion: "Sol", precio: "45-60 euros", distrito: "Centro", productos: "bebidas");
        listaRestaurantes.add(restaurante);
        restaurante=new Restaurante( nombre: "Barganzo", localizacion: "Justicia", precio: "", distrito: "Centro", productos: "pita");
        listaRestaurantes.add(restaurante);
    }
}
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help ProyectoPruebaQueHago - BDDDRestaurantes.java [ProyectoPruebaQueHago.app] - Android Studio

```

1 Project
  app
    sampledata
    manifests
      AndroidManifest.xml
    java
      com.example.proyectopruebaquehago
        Activities
          AppActivity
          InicioActivity
          MainActivity
          PopupActividadCultural
          PopupRestaurante
          RegistroActivity
        Adaptoadores
        API
        Controladores
          ControladorApp
          ControladorInicio
          ControladorRegistro
          PáginaGuardar
        Models
          ActividadCultural
          ActividadCulturalExtensa
          ActividadCulturalExtensaAddress
          BDDDRestaurantes
          DatosServiciosActividadesCulturales
          DatosServiciosActividadesCulturalesExtenso
          Restaurante
      com.example.proyectopruebaquehago (androidTest)
    build.gradle
    settings.gradle
  build Variants
  Favorites
  Structure
  Build Variants
  Logcat
  Failed to start monitoring 668a8f43 (today 11:44)
  Event Log Layout Inspector
  7:14 CRLF UTF-8 4 spaces
  14°C ESP 10:50 ES 01/11/2021
  Escribe aquí para buscar
  
```

```

    restaurante=new Restaurante( nombre: "Maridaje Divino", localizacion: "Palacio", precio: ">20 euros", distrito: "Centro", productos: listaRestaurantes.add(restaurante);
    restaurante=new Restaurante( nombre: "Liven Japan", localizacion: "Cabajadores", precio: "", distrito: "Centro", productos: listaRestaurantes.add(restaurante);
    restaurante=new Restaurante( nombre: "Acuarela Bistró Bar", localizacion: "Palacio", precio: "", distrito: "Centro", productos: listaRestaurantes.add(restaurante);
    restaurante=new Restaurante( nombre: "La Esquina Del Real", localizacion: "Sol", precio: "45-60 euros", distrito: "Centro", productos: listaRestaurantes.add(restaurante);
    restaurante=new Restaurante( nombre: "Barganzo", localizacion: "Justicia", precio: "", distrito: "Centro", productos: listaRestaurantes.add(restaurante);
    restaurante=new Restaurante( nombre: "Chila", localizacion: "Palacio", precio: "20-30 euros", distrito: "Centro", productos: listaRestaurantes.add(restaurante);
    restaurante=new Restaurante( nombre: "Cebo", localizacion: "Cortes", precio: "+100 euros", distrito: "Centro", productos: listaRestaurantes.add(restaurante);
    restaurante=new Restaurante( nombre: "Sake Bar Shua Shua", localizacion: "Justicia", precio: "30-45 euros", distrito: "Centro", productos: listaRestaurantes.add(restaurante);

    //Restaurantes
}

public ArrayList<Restaurante> getListarRestaurantes() { return listaRestaurantes; }
  
```

## Datos Servicios Actividades Culturales:

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help ProyectoPruebaQueHago - DatosServiciosActividadesCulturales.java [ProyectoPruebaQueHago.app] - Android Studio

```

1 Project
  app
    sampledata
    manifests
      AndroidManifest.xml
    java
      com.example.proyectopruebaquehago
        Activities
          AppActivity
          InicioActivity
          MainActivity
          PopupActividadCultural
          PopupRestaurante
          RegistroActivity
        Adaptoadores
        API
        Controladores
          ControladorApp
          ControladorInicio
          ControladorRegistro
          PáginaGuardar
        Models
          ActividadCultural
          ActividadCulturalExtensa
          ActividadCulturalExtensaAddress
          BDDDRestaurantes
          DatosServiciosActividadesCulturales
          DatosServiciosActividadesCulturalesExtenso
          Restaurante
      com.example.proyectopruebaquehago (androidTest)
    build.gradle
    settings.gradle
  build Variants
  Favorites
  Structure
  Build Variants
  Logcat
  Failed to start monitoring 668a8f43 (today 11:44)
  Event Log Layout Inspector
  10:51 CRLF UTF-8 4 spaces
  14°C ESP 10:50 ES 01/11/2021
  Escribe aquí para buscar
  
```

```

    public class DatosServiciosActividadesCulturales {

        //private JsonObject graph;
        @SerializedName("id")
        private int id;
        private String c;
        private String link;
        private String title;

        private ActividadCultural actividadCultural;

        @SerializedName("graph")
        private ArrayList<ActividadCultural> listaDatosServicios = new ArrayList<>();

        public DatosServiciosActividadesCulturales() {
            //parseJSON();
        }

        public ArrayList<ActividadCultural> getListarDatosServicios() { return listaDatosServicios; }

        public void setListarDatosServicios(ArrayList<ActividadCultural> listaDatosServicios) {
            this.listaDatosServicios = listaDatosServicios;
        }

        public String getTitle() { return title; }

        public ActividadCultural getActividadCultural() { return actividadCultural; }
    }
  
```

The screenshot shows the Android Studio interface with the project 'ProyectoPrueba1' open. The code editor displays the class `DatosServiciosActividadesCulturales`. The code implements a `ActividadCultural` interface, defining methods like `gettitle()`, `getActividadCultural()`, `setActividadCultural()`, `setC()`, `getc()`, `setId()`, `getId()`, `setlink()`, `getLink()`, and a static method `parseJson()` for parsing JSON responses.

```
public String gettitle() { return title; }

public ActividadCultural getActividadCultural() { return actividadCultural; }

public void setActividadCultural(ActividadCultural actividadCultural) {
    this.actividadCultural = actividadCultural;
}

public void setC(String c) { this.c = c; }

public String getc() { return c; }

public void setId(int id) { this.id = id; }

public int getId() { return id; }

public void setlink(String link) { this.link = link; }

public String getLink() { return link; }

public static ActividadCultural parseJson(String response){
    Gson gson=new GsonBuilder().create();
    ActividadCultural datos=gson.fromJson(response,ActividadCultural.class);
    return datos;
}
```

## Datos Servicios Actividades Culturales Extenso:

The screenshot shows the Android Studio interface with the project 'ProyectoPrueba1' open. The code editor displays the class `DatosServiciosActividadesCulturalesExtenso`. This class extends `ActividadCulturalExtenso` and includes private fields for `id`, `c`, `link`, and `title`. It also includes a private field `graph` and a private field `listadoDatosServicios` of type `ArrayList<ActividadCulturalExtenso>`. The class has a constructor that calls `parseJSON()` and a method `getListaDatosServicios()` that returns the list of services.

```
private JsonObject graph;
@SerializedName("@id")
private int id;
private String c;
private String link;
private String title;

private ActividadCulturalExtenso actividadCulturalExtenso;

@SerializedName("graph")
private ArrayList<ActividadCulturalExtenso> listadoDatosServicios = new ArrayList<>();

public DatosServiciosActividadesCulturalesExtenso() {
    //parseJSON();
}

public ArrayList<ActividadCulturalExtenso> getListaDatosServicios() {
    return listadoDatosServicios;
}

public void setListaDatosServicios(ArrayList<ActividadCulturalExtenso> listaDatosServicios) {
    this.listadoDatosServicios = listaDatosServicios;
}

public String getTitle() { return title; }
```

```
public String getTitle() { return title; }

public ActividadCulturalExtensa getActividadCultural() { return actividadCulturalExtensa; }

public void setActividadCultural(ActividadCulturalExtensa actividadCultural) {
    this.actividadCulturalExtensa = actividadCulturalExtensa;
}

public void setC(String c) { this.c = c; }

public String getC() { return c; }

public void setId(int id) { this.id = id; }

public int getId() { return id; }

public void setLink(String link) { this.link = link; }

public String getLink() { return link; }

public static ActividadCulturalExtensa parseJson(String response) {
    Gson gson = new GsonBuilder().create();
    ActividadCulturalExtensa datos = gson.fromJson(response, ActividadCulturalExtensa.class);
    return datos;
}
```

## Restaurante:

```
package com.example.proyectoquehago.Models;

import java.util.ArrayList;

public class Restaurante {
    private String nombre;
    private String localizacion;
    private String precio;
    private String distrito;
    private String productos;
    private String puntuacion;

    public Restaurante(String nombre, String localizacion, String precio, String distrito, String productos, String puntuacion) {
        this.nombre = nombre;
        this.localizacion = localizacion;
        this.precio = precio;
        this.distrito = distrito;
        this.productos = productos;
        this.puntuacion = puntuacion;
    }

    public void setPrecio(String precio) { this.precio = precio; }

    public String getPrecio() { return precio; }

    public void setDistrito(String distrito) { this.distrito = distrito; }

    public String getDistrito() { return distrito; }
}
```

```

public void setPrecio(String precio) { this.precio = precio; }

public String getPrecio() { return precio; }

public void setDistrito(String distrito) { this.distrito = distrito; }

public String getDistrito() { return distrito; }

public void setNombre(String nombre) { this.nombre = nombre; }

public void setLocalizacion(String localizacion) { this.localizacion = localizacion; }

public void setProductos(String productos) { this.productos = productos; }

public void setPuntuacion(String puntuacion) { this.puntuacion = puntuacion; }

public String getNombre() { return nombre; }

public String getLocalizacion() { return localizacion; }

public String getProductos() { return productos; }

public String getPuntuacion() { return puntuacion; }

```

## 4. Fase de pruebas

### 4.1. Pruebas realizadas.

Prueba actualizar sin fecha ....

## 5. Conclusiones finales

### 5.1. Grado de cumplimiento de los objetivos fijados.

El grado de cumplimiento es satisfactorio ya que se consigue los objetivos planteados desde el principio que son los de proporcionar la visualización de servicio de ocio al usuario y poder este seleccionarlos de una manera aleatoria.

Entre los desagrados finales son principalmente derivados de la falta de encuentro de más API o servicios que aporten datos, para así poder proporcionar una mayor cantidad de actividades al usuario, así como el echo de que únicamente sean actividades culturales de la ciudad de Madrid, ya que como he dicho no he podido encontrar más API que me proporcione datos de otras ciudades.

Otro desagrado es el echo de que el servidor de manejo de la base de datos sea gestionado de forma local, lo indicado será crear un servidor de manera que este alojado en la nube y siempre

que se tenga conexión se pueda tener acceso a este. Así como la falta de seguridad que tenemos en nuestra base de datos ya que esta conexión se realiza mediante Http y noHttps.

### 5.2. Propuesta de modificaciones o ampliaciones futuras del sistema implementado.

Entre las modificaciones que haría, la principal sería la de mejorar el método de guardar los servicios solicitados, creando un apartado de usuario en que pueda visualizar los servicios solicitados en cada fecha. Además en el apartado de Usuario se podría crear apartados de configuración para poder cambiar los datos personales que quiera el usuario así como añadir foto u otros datos.

Además una ampliación muy importante sería la de que los servicios que se solicitan se solicitaran verdaderamente en la página de cada actividad y/o restaurante, para así poder no solo ver los servicios si no que solicitarlos correctamente.

Entre otras mejoras a añadir están las de mejorar la visualización de la aplicación, ya que esta es muy básica y con unos colores no demasiado acordes.

## 6. Documentación del sistema desarrollado

### 6.1. Manual de Instalación.

Para instalar esta aplicación se puede hacer de dos maneras, una teniendo acceso al archivo APK generado de la aplicación, que permite la instalación en dispositivos móviles de manera sencilla.

Otra es a través del propio Android Studio que permite a través de ejecutar la aplicación la obtención en el dispositivo móvil que tengamos conectado a nuestro ordenador a la hora de ejecutar la aplicación desde Android Studio.

Estas son las dos únicas formas de instalación puesto que esta aplicación no será subida a ninguna tienda de aplicaciones así como a ninguna página web para su descarga.

### 6.2. Manual de uso.

Para manejar esta aplicación primero debemos una vez abierta es registrar un usuario (siempre que se tenga acceso al servidor de la base de datos, si no se puede acceder directamente desde el botón de “Acceso-Sin Conexión”), para ello pulsaremos sobre el botón de registrarse para poder acceder a dicho apartado, una vez aquí simplemente insertamos datos(IMPORTANTE, no se deberían poner espacios ni acentos ni ningún otro carácter especial salvo @, en ninguno de los EditText) y pulsamos sobre registrarse y pulsamos la flecha del menú para volver al apartado inicial.

Creado el usuario procedemos a escribir los datos de nombre de usuario y contraseña y procedemos a pulsar el botón iniciar sesión, este comprobara si es correcto, si es así iniciará la aplicación.

Comprobado el usuario y abierta la aplicación tendremos acceso a ella y se empezará a buscar los servicios a la API, lo primero será escoger la fecha que queramos, para ello pulsaremos sobre el botón de calendario que tenemos en el menú de arriba y seleccionamos la fecha.

Ahora que tenemos los datos recogidos de la API y la fecha debemos de pulsar sobre botón de actualizar que se encuentra a la derecha del botón de calendario, y aparecerán los distintos servicios. Estos se mostrarán si no tocamos nada más en la pagina de mañana ya que es la que esta establecida por defecto, si queremos visualizar los servicios de la tarde o los de comida y cena pulsamos sobre su botón correspondiente que se encuentra debajo de la pantalla.

Ahora que se nos muestran los servicios podemos visualizar más información pulsando sobre ellos y nos abrirá el popMenu con la información correspondiente al servicio o restaurante pulsado, en este aparecerán el nombre la descripción, ubicación, precio y fecha, en algunos la información no está completa, pero como es la información obtenida de la API, esta viene así por defecto. Para cerrar este popMenu únicamente debemos de pulsar sobre el espacio transparente que esta fuera del popMenu.

Si el servicio que queremos, este u otro, y queremos solicitarlo únicamente debemos de mantener una pulsación sobre el para pulsar sobre el menú de “Solicitar”, pulsado este, si queremos cambiar de servicio no hay que hacer nada

únicamente pulsamos sobre el nuevo servicio que queremos, NO esta la opción de borrar el servicio y dejarlo vacío, ya que se requiere al menos un servicio o restaurante de cada pantalla, si por algún casual no se seleccionan los servicios o restaurante para cada aparato no se enviara la información de servicios solicitados a la base de datos aunque nos aparezca el mensaje de “Guardado”.

También contamos con la opción de generar servicios aleatorios que se podrá hacer pulsando sobre el botón de 3 puntos que aparece arriba a la derecha y pulsando sobre “Automático”, cuando pulsemos sobre el no aparecerá ningún mensaje, simplemente se escogerán los servicios aleatorios, si hemos escogido esta opción, pero queremos, por ejemplo elegir personalmente el servicio de por la tarde, simplemente le solicitamos el que queramos y se cambiara por el otro, y los demás se quedaran como la función automática a escogido.

Por último tenemos el botón de guardar, el cual se encargara de enviar los servicios solicitados a la base de datos, además mostrara mediante un Toast (un mensaje temporal flotante en pantalla) los datos de los servicios solicitados.