

MANUAL DE PRÁCTICAS

Nombre de la práctica	Android Studio-INTELLIJ IDEA.				1
Asignatura:	TÓPICOS AVANZADOS DE PROGRAMACIÓN	Carrera:	Ingeniería en Sistemas Computacionales	Duración de la práctica (Hrs)	50 horas

NOMBRE DEL ALUMNO: JONATHAN ZUAR HERNANDEZ MAYEN.

GRUPO: 3401.

I. Competencia(s) específica(s):

Aplica la programación orientada a objetos para resolver problemas reales y de ingeniería.

Encuadre con CACEI: Registra el (los) atributo(s) de egreso y los criterios de desempeño que se evaluarán en esta práctica.

No. atributo	Atributos de egreso del PE que impactan en la asignatura	Criterios de desempeño	
2	El estudiante diseñará esquemas de trabajo y procesos, usando metodologías congruentes en la resolución de problemas de ingeniería en sistemas computacionales	1	Identifica metodologías y procesos empleados en la resolución de problemas
		2	Diseña soluciones a problemas, empleando metodologías apropiadas al área
3	El estudiante plantea soluciones basadas en tecnologías empleando su juicio ingenieril para valorar necesidades, recursos y resultados esperados.	1	Emplea los conocimientos adquiridos para el desarrollar soluciones
		2	Analiza y comprueba resultados

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):

Actividades en casa con recursos propios

-Lab.Computo

-En casa.

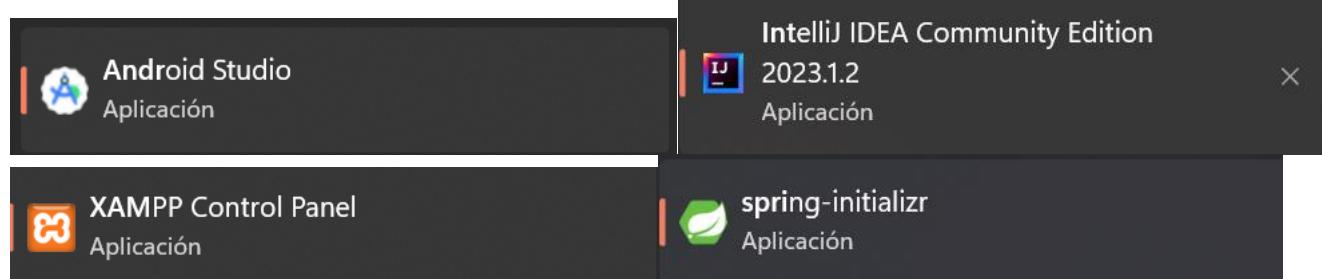
III. Material empleado:

- LAPTOP PERSONAL
- Android Studio
- MYSQL
- Java
- INTELLIJ IDEA

IV. Desarrollo de la práctica:

Primeros Pasos:

Debemos contar con las siguientes herramientas para la elaboración de este proyecto:



Spring-initializr, IntelliJ IDEA, XAMPP Y Android Studio:

Spring Initializr y IntelliJ IDEA están estrechamente relacionados y se complementan entre sí en el desarrollo de aplicaciones basadas en el framework de Spring.

Spring Initializr es una herramienta en línea que permite generar rápidamente un proyecto de Spring Boot con todas las dependencias y configuraciones necesarias. Proporciona una interfaz intuitiva donde puedes seleccionar las dependencias específicas de Spring y otras bibliotecas que necesites para tu proyecto. Al finalizar la configuración, se genera un proyecto listo para importar en tu entorno de desarrollo.

IntelliJ IDEA, por otro lado, es un poderoso IDE que ofrece soporte completo para el desarrollo de aplicaciones de Spring. Tiene una integración perfecta con Spring Initializr, lo que facilita la importación y configuración de proyectos generados por Spring Initializr en el IDE. Esto significa que puedes importar el proyecto creado con Spring Initializr directamente en IntelliJ IDEA y comenzar a trabajar en él sin problemas.

IntelliJ IDEA proporciona una amplia gama de características y herramientas específicas para el desarrollo de aplicaciones de Spring, como resaltado de sintaxis, autocompletado inteligente, navegación entre clases y métodos, refactorización de código, depuración, soporte de pruebas y muchas otras. Además, IntelliJ IDEA tiene integración con el sistema de construcción de Spring Boot, lo que permite ejecutar y depurar fácilmente tu aplicación dentro del IDE.

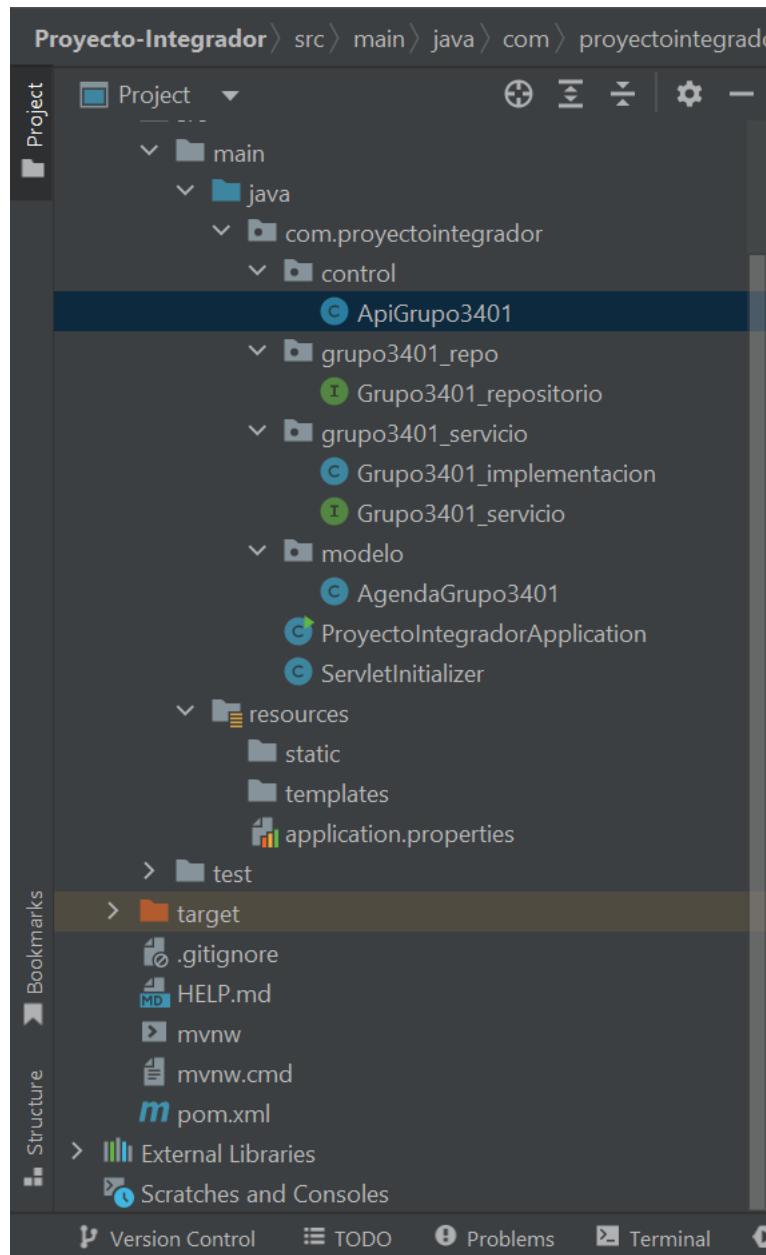
En resumen, Spring Initializr te ayuda a generar rápidamente un proyecto de Spring Boot con las dependencias necesarias, y IntelliJ IDEA te brinda un entorno de desarrollo completo y potente con características específicas para el desarrollo de aplicaciones de Spring, facilitando el trabajo con proyectos generados por Spring Initializr.

A su vez esto se complementa con XAMPP en el conectar una base de datos para consumir el servicio según nuestro entorno y esto se relaciona con Android Studio donde tenemos nuestro resto de servicio para el entorno de interfaz para consumir el servicio, todo va de la mano.

MANUAL DE PRÁCTICAS

Iniciamos Con IntelliJ IDEA:

A grandes rasgos estos son los paquetes que manejamos para llevar a cabo el proceso de nuestra creación del API.



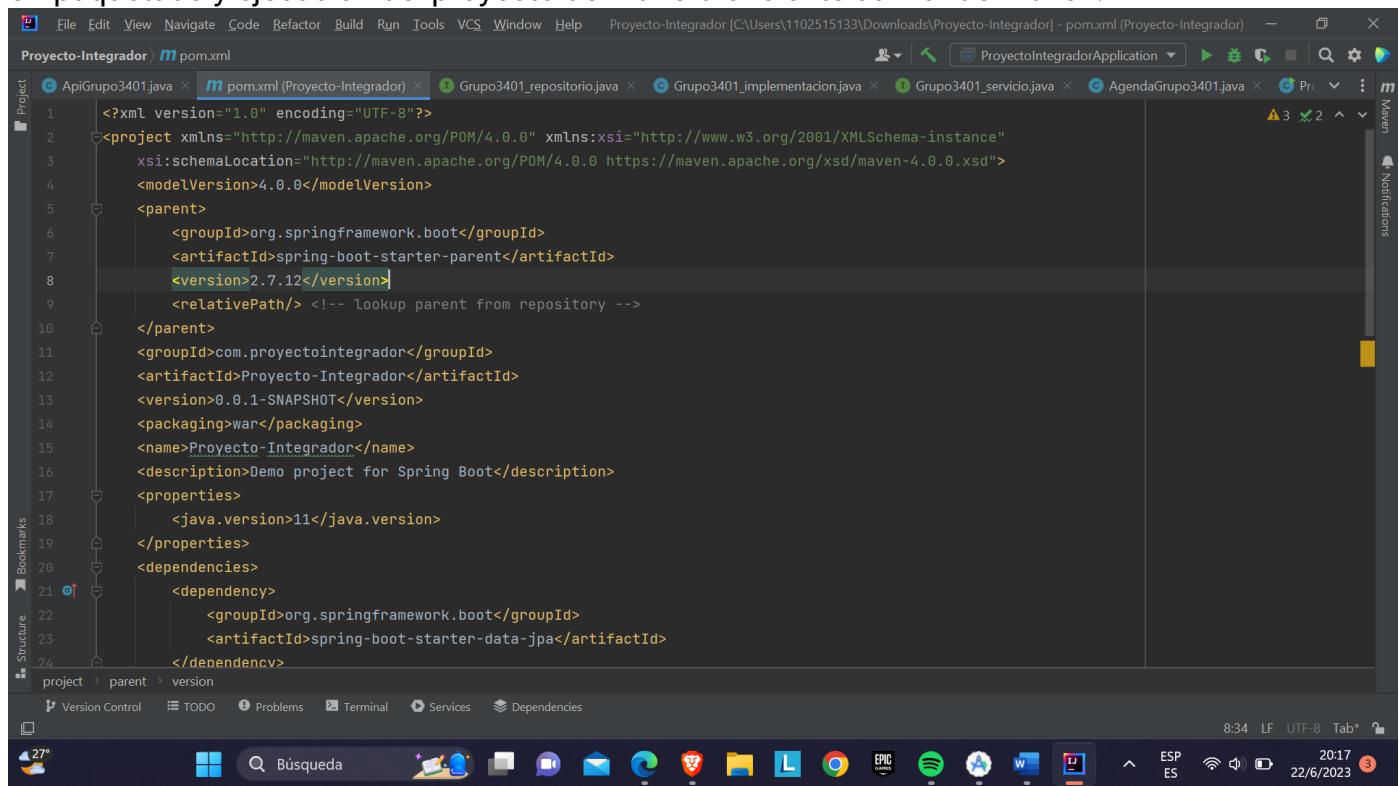
MANUAL DE PRÁCTICAS

(POM.XML)

Este es un archivo de configuración POM (Project Object Model) de Maven para un proyecto de Spring Boot. El archivo define las dependencias y la configuración del proyecto. A continuación, se presenta un resumen de las acciones que se llevan a cabo en este archivo:

- Se define la versión del modelo de proyecto (modelVersion).
- Se especifica el parent (padre) del proyecto, que es el proyecto padre de Spring Boot con la versión correspondiente.
- Se establecen el groupId, artifactId y la versión del proyecto.
- Se define el packaging del proyecto como "war".
- Se proporciona una descripción del proyecto.
- Se establece la versión de Java que se utilizará en el proyecto.
- Se enumeran las dependencias necesarias para el proyecto, incluyendo spring-boot-starter-data-jpa, spring-boot-starter-web, mysql-connector-j, spring-boot-starter-tomcat y spring-boot-starter-test.
- Se configura el plugin de Maven de Spring Boot para el empaquetado y ejecución del proyecto.
- Dentro de la sección de construcción (build), se definen los plugins necesarios, en este caso, el plugin spring-boot-maven-plugin.

En resumen, este archivo de configuración POM de Maven define las dependencias necesarias y la configuración básica para un proyecto de Spring Boot, permitiendo la construcción, empaquetado y ejecución del proyecto de manera eficiente utilizando Maven.



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.12</version>
    <relativePath/> 
  </parent>
  <groupId>com.proyectointegrador</groupId>
  <artifactId>Proyecto-Integrador</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>Proyecto-Integrador</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
```



MANUAL DE PRÁCTICAS

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
</dependencies>
```

```
<scope>provided</scope>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

```
</project>
```

MANUAL DE PRÁCTICAS

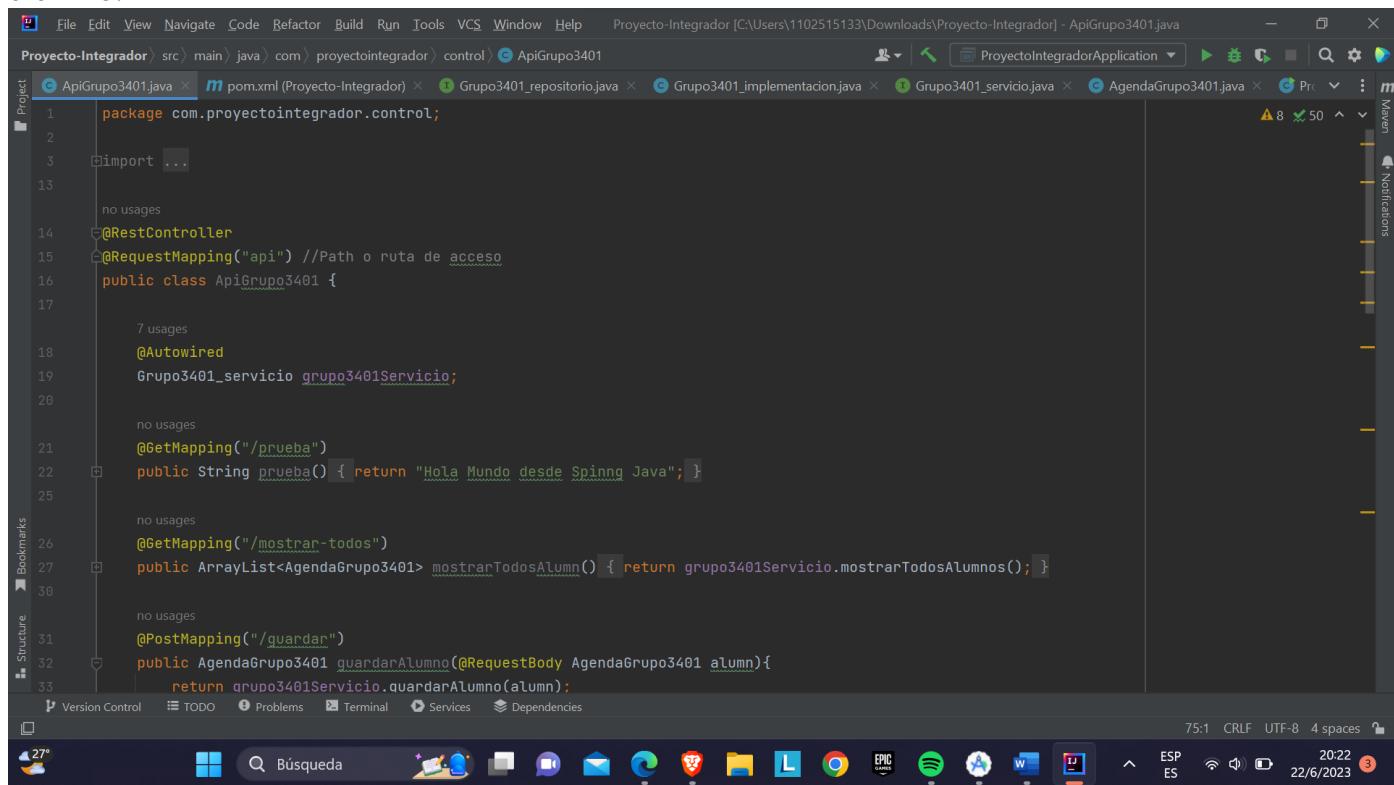
(API 3401.java)

A grandes rasgos esto es lo mas importante para realizar nuestra API.

En este código se define un controlador de una API en Spring Boot. Aquí hay un resumen muy breve de lo que se hace en cada parte:

- Se importan las clases necesarias y las dependencias.
- Se anota la clase como un controlador de la API utilizando `@RestController`.
- Se establece la ruta base para los endpoints de la API utilizando `@RequestMapping`.
- Se inyecta la dependencia de un servicio utilizando `@Autowired`.
- Se definen varios métodos para manejar diferentes endpoints de la API, como obtener una prueba, mostrar todos los alumnos, guardar un alumno, borrar un alumno por ID, mostrar un alumno por ID y actualizar un alumno.

En resumen, este código implementa un controlador de una API en Spring Boot que proporciona diferentes endpoints para realizar operaciones relacionadas con alumnos. Estos endpoints incluyen pruebas, obtener todos los alumnos, guardar, borrar, mostrar por ID y actualizar un alumno.



```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Proyecto-Integrador [C:\Users\1102515133\Downloads\Proyecto-Integrador] - ApiGrupo3401.java
Proyecto-Integrador > src > main > java > com > proyectointegrador > control > ApiGrupo3401.java
Project: ApiGrupo3401.java pom.xml (Proyecto-Integrador) Grupo3401_repositorio.java Grupo3401_implementacion.java Grupo3401_servicio.java AgendaGrupo3401.java Pro... Maven Notifications
1 package com.proyectointegrador.control;
2
3 import ...
13
14 @RestController
15 @RequestMapping("api") //Path o ruta de acceso
16 public class ApiGrupo3401 {
17
18     @Autowired
19     Grupo3401_servicio grupo3401Servicio;
20
21     @GetMapping("/prueba")
22     public String prueba() { return "Hola Mundo desde Spinn Java"; }
25
26     @GetMapping("/mostrar-todos")
27     public ArrayList<AgendaGrupo3401> mostrarTodosAlumno() { return grupo3401Servicio.mostrarTodosAlumnos(); }
30
31     @PostMapping("/guardar")
32     public AgendaGrupo3401 guardarAlumno(@RequestBody AgendaGrupo3401 alumno){
33         return grupo3401Servicio.guardarAlumno(alumno);
    }

Bookmarks Structure Version Control TODO Problems Terminal Services Dependencies 75:1 CRLF UTF-8 4 spaces 20:22 ESP ES 22/6/2023 3
```



MANUAL DE PRÁCTICAS

The screenshot shows an IDE interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help.
- Title Bar:** Proyecto-Integrador [C:\Users\1102515133\Downloads\Proyecto-Integrador] - ApiGrupo3401.java
- Project Explorer:** Shows files like ApiGrupo3401.java, pom.xml, Grupo3401_repositorio.java, Grupo3401_implementacion.java, Grupo3401_servicio.java, and AgendaGrupo3401.java.
- Code Editor:** Displays Java code for a REST API, including annotations like @GetMapping and @PostMapping.
- Toolbars:** Version Control, TODO, Problems, Terminal, Services, Dependencies.
- Status Bar:** 42:88 CRLF UTF-8 4 spaces, ESP ES 20:23 22/6/2023.

En este punto entra (XAMPP) para hacer la conexión con nuestro servidor. Aquí inicializaremos de momento solo nuestros servicios.

The XAMPP Control Panel v3.3.0 window displays the following information:

Service	Module	PID(s)	Port(s)	Actions
	Apache	19140 11772	80, 443	Stop Admin Config Logs
	MySQL	19192	3306	Stop Admin Config Logs
	FileZilla			Start Admin Config Logs
	Mercury			Start Admin Config Logs
	Tomcat			Start Admin Config Logs

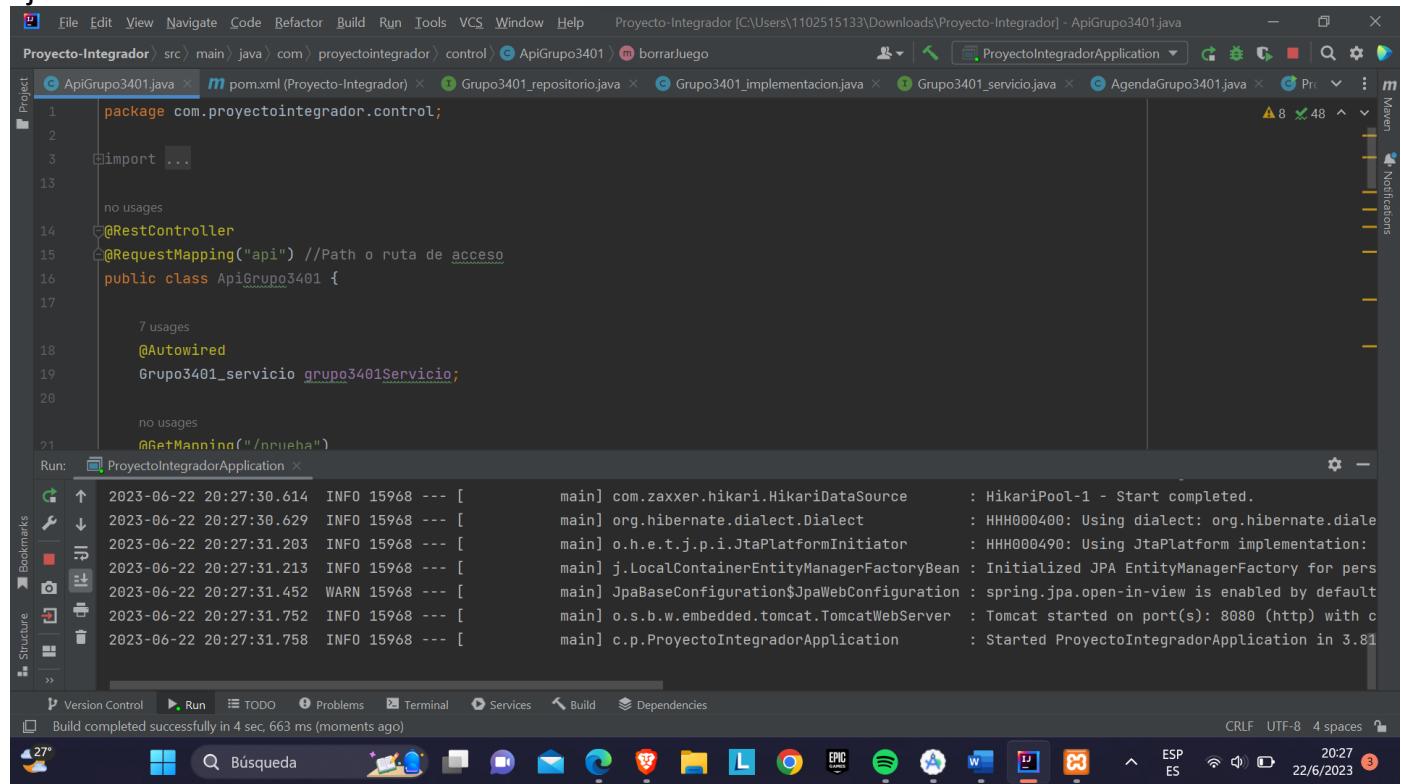
On the right side, there are links for Config, Netstat, Shell, Explorer, Services, Help, and Quit.

The bottom log window shows the following output:

```
20:25:47 [main] All prerequisites found
20:25:47 [main] Initializing Modules
20:25:47 [main] Starting Check-Timer
20:25:47 [main] Control Panel Ready
20:25:48 [Apache] Attempting to start Apache app...
20:25:49 [Apache] Status change detected: running
20:25:49 [mysql] Attempting to start MySQL app...
20:25:50 [mysql] Status change detected: running
```

MANUAL DE PRÁCTICAS

Una vez hecho esto procedemos a ejecutar nuestra API por IntelliJ IDEA, y así ser vería ya en ejecución.



```

package com.proyectointegrador.control;

import ...

@RestController
@RequestMapping("api") //Path o ruta de acceso
public class ApiGrupo3401 {

    @Autowired
    Grupo3401_servicio grupo3401Servicio;

    @GetMapping("/prueba")
}

```

Run: ProyectoIntegradorApplication

2023-06-22 20:27:30.614 INFO 15968 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.

2023-06-22 20:27:30.629 INFO 15968 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect

2023-06-22 20:27:31.203 INFO 15968 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation:

2023-06-22 20:27:31.213 INFO 15968 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistency unit

2023-06-22 20:27:31.452 WARN 15968 --- [main] JpaBaseConfiguration\$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default

2023-06-22 20:27:31.752 INFO 15968 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path /

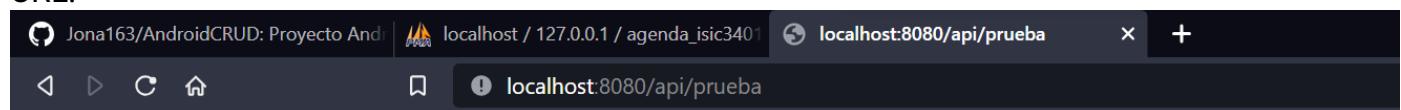
2023-06-22 20:27:31.758 INFO 15968 --- [main] c.p.ProyectoIntegradorApplication : Started ProyectoIntegradorApplication in 3.811 seconds (JVM running for 4.001)

Aquí de forma automática nos inicializo nuestro servicio de TOMCAT.

Service	Module	PID(s)	Port(s)	Actions			
	Apache	19140 11772	80, 443	Stop	Admin	Config	Logs
	MySQL	19192	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat	15968 b49b4, b49b5, b49b6,		Stop	Admin	Config	Logs

Aquí utilizaremos una herramienta mas llamada POSTMAN O podemos comprobar a través de un URL para saber si esta funcionando correctamente nuestros servicios.

URL:

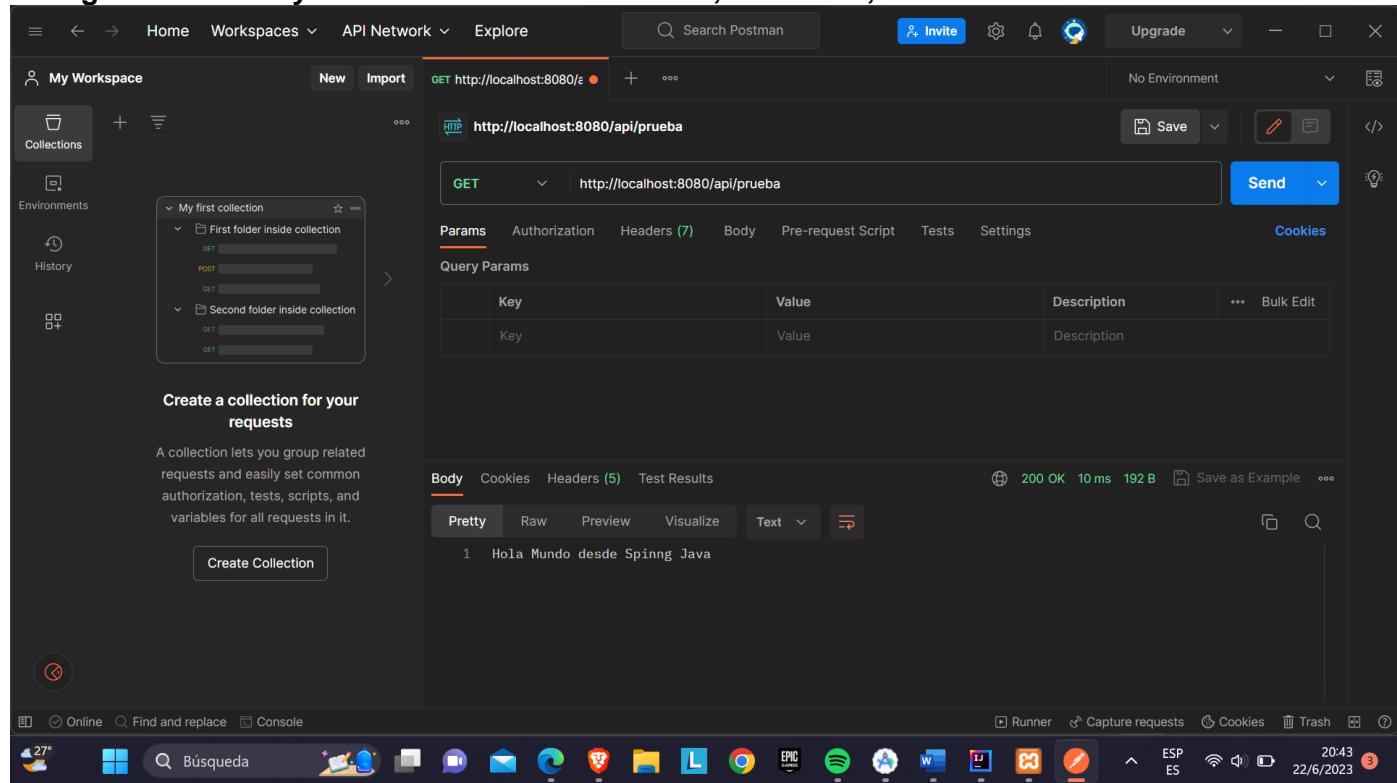


Hola Mundo desde Spinn Java

MANUAL DE PRÁCTICAS

POSTMAN:

Se ingresa los links y seleccionamos si es un GET, MAPPING, ETC.



Ahora volvemos a trabajar con nuestro XAMPP para crear nuestra base de datos para esto debemos entrar a nuestro <http://localhost/phpmyadmin/>.



Configuraciones generales

- Colejamiento de la conexión al servidor: utf8mb4_unicode_ci
- Más configuraciones

Configuraciones de apariencia

- Idioma (Language): Español - Spanish
- Tema: pmahomme

Servidor de base de datos

- Servidor: 127.0.0.1 via TCP/IP
- Tipo de servidor: MariaDB
- Conexión del servidor: No se está utilizando SSL
- Versión del servidor: 10.4.28-MariaDB - mariadb.org binary distribution
- Versión del protocolo: 10
- Usuario: root@localhost
- Conjunto de caracteres del servidor: UTF-8 Unicode (utf8mb4)

Servidor web

- Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4
- Versión del cliente de base de datos: libmysql - mysqld 8.2.4
- extensión PHP: mysqli curl mbstring
- Versión de PHP: 8.2.4

Ahora aquí creamos nuestra base de datos de acuerdo a nuestras necesidades y podemos agregar algunos datos para probar que está funcionando los servicios.

agenda_isic3401

agenda_grupo3401

agenda_grupo3401

Mostrando filas 0 - 3 (total de 4, La consulta tardó 0,0004 segundos.)

```
SELECT * FROM `agenda_grupo3401`
```

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

	id	fecha_cumple	gustos	nombre
<input type="checkbox"/>	1	tt	rt	Kona
<input type="checkbox"/>	2	24 de septiembre	cachimbear, color azul, camiones	Jonathan Zuar Hernandez Mayen
<input type="checkbox"/>	3	26 de julio	cantar, color cafe, UN BALON DE BASQUET	Alondra Cayetano Rosendo
<input type="checkbox"/>	4	11 de febrero 2003	Juli	Gael

Operaciones sobre los resultados de la consulta

Consola Copiar al portapapeles Exportar Mostrar gráfico Crear vista

MANUAL DE PRÁCTICAS

Terminado esto pasamos a nuestro Android Studio donde veremos a grandes rasgos que hicimos. **ANDROID-STUDIO.**

El código XML que has proporcionado es el archivo de manifiesto de una aplicación Android. Aquí se definen varias configuraciones y permisos para la aplicación. Algunas de las cosas que se hacen aquí son:

1. Se solicita el permiso de acceso a Internet mediante la línea `<uses-permission android:name="android.permission.INTERNET" />`. Esto permite que la aplicación pueda acceder a Internet e igual se pide el acceso al tráfico de datos para consumir nuestra API.
2. Se definen diferentes actividades de la aplicación dentro del elemento `<application>`. Por ejemplo, se definen las actividades "MainActivity", "Agenda", "Actualizar" y "Reg".
3. La actividad "MainActivity" se define como la actividad principal de la aplicación mediante las líneas:

```
...
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
...
```

Esto significa que cuando se inicie la aplicación, se mostrará la actividad "MainActivity" y se establece como punto de entrada principal.

Estas son solo algunas de las cosas que se realizan en este archivo de manifiesto. El archivo de manifiesto es esencial en una aplicación de Android, ya que proporciona información importante sobre la aplicación y su configuración.

(Build.gradle:app)

1-Plugins: Se utiliza el plugin 'com.android.application', que indica que este es un proyecto de aplicación de Android.

2-Android: Se establecen las configuraciones principales de la aplicación, como el espacio de nombres (namespace), el SDK de compilación (compileSdk), y la configuración por defecto (defaultConfig), que incluye el ID de la aplicación (applicationId), la versión mínima (minSdk), la versión objetivo (targetSdk), el código de versión (versionCode) y el nombre de la versión (versionName).

3-BuildTypes: Se define un tipo de compilación llamado "release" que desactiva la minificación (minifyEnabled) y especifica los archivos de configuración de ProGuard.

MANUAL DE PRÁCTICAS

4-CompileOptions: Se establecen las opciones de compilación, como la compatibilidad de la fuente y el destino con Java 8 (`sourceCompatibility` y `targetCompatibility`).

5-Dependencies: Aquí se especifican las dependencias del proyecto. Una de ellas es la dependencia de Volley: `implementation 'com.android.volley:volley:1.2.1'`. Volley es una biblioteca utilizada para realizar solicitudes de red en aplicaciones de Android. Esta dependencia permite utilizar las funcionalidades de Volley en el proyecto.

En resumen, este archivo de configuración de construcción establece la configuración básica de la aplicación, las opciones de compilación y las dependencias necesarias. La dependencia de Volley, en particular, permite utilizar las funcionalidades de esta biblioteca para manejar las solicitudes de red en la aplicación de Android.

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Other | build.gradle (app)
```

AgCumples > app > build.gradle

You can use the Project Structure dialog to view and edit your project configuration

```
plugins {  
    id 'com.android.application'  
}  
  
android {  
    namespace 'com.example.llll'  
    compileSdk 33  
  
    defaultConfig {  
        applicationId "com.example.llll"  
        minSdk 24  
        targetSdk 33  
        versionCode 1  
        versionName "1.0"  
  
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
    }  
  
    buildTypes {  
        android[] > compileOptions()  
    }  
}
```

Resource Manager Bookmarks Build Variants Structure

Project Bookmarks Build Variants Structure

Open (Ctrl+Alt+Mayús+S) Hide notification Device Manager Notifications Device File Explorer Running Devices

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection Layout Inspector

Failed to start monitoring emulator-5554 (today 19:15)

26:52 LF UTF-8 4 spaces 20:55 22/6/2023 3 ESP ES



GOBIERNO DEL
ESTADO DE MÉXICO



MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface with the build.gradle (app) tab selected. The code editor displays the following build configuration:

```
buildTypes {  
    release {  
        minifyEnabled false  
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
    }  
}  
compileOptions {  
    sourceCompatibility JavaVersion.VERSION_1_8  
    targetCompatibility JavaVersion.VERSION_1_8  
}  
  
dependencies {  
  
    implementation 'androidx.appcompat:appcompat:1.6.1'  
    implementation 'com.google.android.material:material:1.5.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
    implementation 'com.android.volley:volley:1.2.1'  
    testImplementation 'junit:junit:4.13.2'  
    android{> compileOptions()  
}  
}
```

The bottom part of the screenshot shows the system tray with the date and time (22/6/2023, 20:55).

The screenshot shows the Windows taskbar with pinned icons for File Explorer, Task View, Control Panel, and others. The date and time are displayed as 22/6/2023, 20:56.

MANUAL DE PRÁCTICAS

(AgendaClient.java)

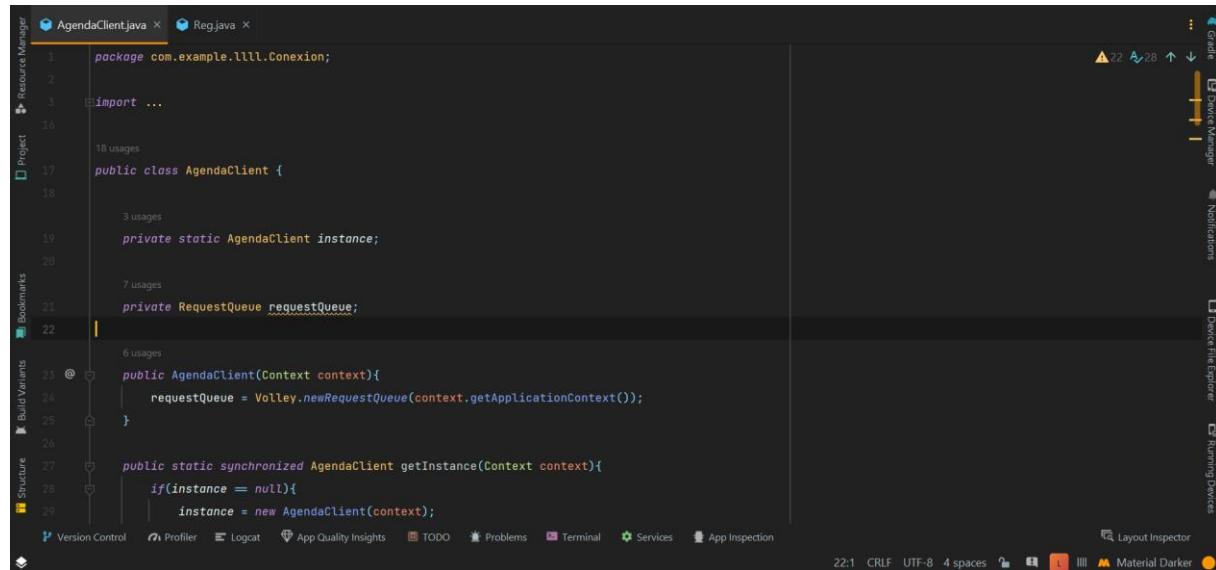
Este código representa una clase llamada "AgendaClient" que se utiliza para realizar solicitudes HTTP utilizando la biblioteca Volley en una aplicación de Android.

En resumen, las principales funciones de esta clase son:

1. Crear una cola de solicitudes de Volley en el constructor de la clase.
2. Proporcionar una instancia singleton de la clase "AgendaClient" utilizando el método "getInstance".
3. Obtener la cola de solicitudes de Volley utilizando el método "getRequestQueue".
4. Realizar una solicitud HTTP GET para encontrar un elemento por su ID utilizando el método "findById".
5. Realizar una solicitud HTTP POST para agregar un nuevo elemento utilizando el método "agregar".
6. Realizar una solicitud HTTP GET para obtener todos los elementos utilizando el método "mostrar".
7. Realizar una solicitud HTTP POST para actualizar un elemento existente utilizando el método "actualizarAlumno".
8. Realizar una solicitud HTTP DELETE para eliminar un elemento por su ID utilizando el método "delete".

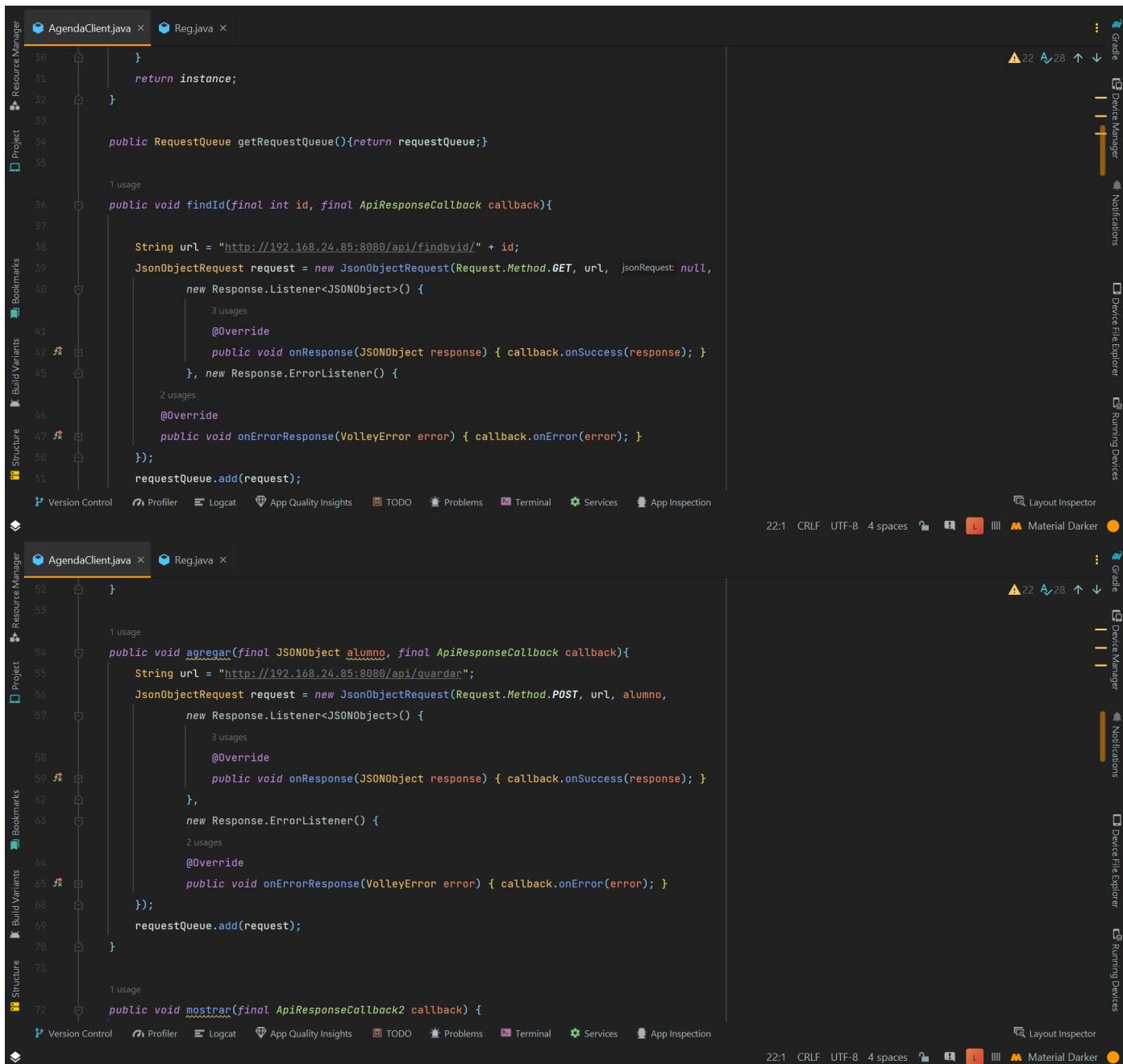
La clase también define dos interfaces de devolución de llamada ("ApiResponseCallback" y "ApiResponseCallback2") que se utilizan para manejar las respuestas exitosas y los errores de las solicitudes.

En general, esta clase proporciona métodos convenientes para interactuar con una API web y realizar operaciones comunes como buscar, agregar, mostrar, actualizar y eliminar elementos. Dentro de las definiciones de esta clase encontramos un String url, dentro se pone la dirección ip en la cual estemos trabajando ya que esto genera un socket de comunicación entre la aplicación y el servidor que levantamos.(se debe corregir cada que se conecte a la red y cambie la IP)



```
AgendaClient.java x Reg.java x
1 package com.example.lllll.Conexion;
2
3 import ...
4
5 18 usages
6
7 public class AgendaClient {
8
9     3 usages
10    private static AgendaClient instance;
11
12    7 usages
13    private RequestQueue requestQueue;
14
15    6 usages
16    @
17    public AgendaClient(Context context){
18        requestQueue = Volley.newRequestQueue(context.getApplicationContext());
19    }
20
21    public static synchronized AgendaClient getInstance(Context context){
22        if(instance == null){
23            instance = new AgendaClient(context);
24        }
25    }
26
27
28
29 }
```

MANUAL DE PRÁCTICAS



```
AgendaClient.java
38     }
39     return instance;
40   }
41
42   public RequestQueue getRequestQueue(){return requestQueue;}
43
44   1 usage
45   public void findById(final int id, final ApiResponseCallback callback){
46
47     String url = "http://192.168.24.85:8080/api/findbyid/" + id;
48     JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url, jsonRequest: null,
49       new Response.Listener<JSONObject>() {
50         3 usages
51         @Override
52         public void onResponse(JSONObject response) { callback.onSuccess(response); }
53       }, new Response.ErrorListener() {
54         2 usages
55         @Override
56         public void onErrorResponse(VolleyError error) { callback.onError(error); }
57       });
58     requestQueue.add(request);
59   }
60
61   public void agregar(final JSONObject alumno, final ApiResponseCallback callback){
62     String url = "http://192.168.24.85:8080/api/guardar";
63     JsonObjectRequest request = new JsonObjectRequest(Request.Method.POST, url, alumno,
64       new Response.Listener<JSONObject>() {
65         3 usages
66         @Override
67         public void onResponse(JSONObject response) { callback.onSuccess(response); }
68       },
69       new Response.ErrorListener() {
70         2 usages
71         @Override
72         public void onErrorResponse(VolleyError error) { callback.onError(error); }
73       });
74     requestQueue.add(request);
75   }
76
77   public void mostrar(final ApiResponseCallback2 callback) {
78 }
```



MANUAL DE PRÁCTICAS

```
String url = "http://192.168.24.85:8080/api/mostrar-todos";
JsonArrayRequest request = new JsonArrayRequest(Request.Method.GET, url, jsonRequest: null,
        new Response.Listener<JSONArray>() {
    3 usages
    @Override
    public void onResponse(JSONArray response) {
        // Procesa la respuesta JSON y llama al callback
        callback.onSuccess(response);
    }
},
new Response.ErrorListener() {
    2 usages
    @Override
    public void onErrorResponse(VolleyError error) {
        // Llama al callback con el error
        callback.onError(error);
    }
});

// Agrega la solicitud a la cola de solicitudes de Volley
requestQueue.add(request);
```



```
// Agrega la solicitud a la cola de solicitudes de Volley
requestQueue.add(request);

1 usage
public void actualizarAlumno(final int id, final JSONObject valoresActualizados, final ApiResponseCallback callback) {
    String url = "http://192.168.24.85:8080/api/actualizar/" + id;

    JsonObjectRequest request = new JsonObjectRequest(Request.Method.POST, url, valoresActualizados,
        new Response.Listener<JSONObject>() {
            3 usages
            @Override
            public void onResponse(JSONObject response) {
                // La actualización se ha realizado correctamente
                // Puedes realizar las acciones necesarias después de la actualización
                callback.onSuccess(response);
            }
},
new Response.ErrorListener() {
    2 usages
});
```



MANUAL DE PRÁCTICAS

```
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
```

```
new Response.ErrorListener() {
    ...
    @Override
    public void onErrorResponse(VolleyError error) {
        // Manejar el error de la solicitud
        callback.onError(error);
    }
};

// Agregar la solicitud a la cola de solicitudes de Volley
requestQueue.add(request);

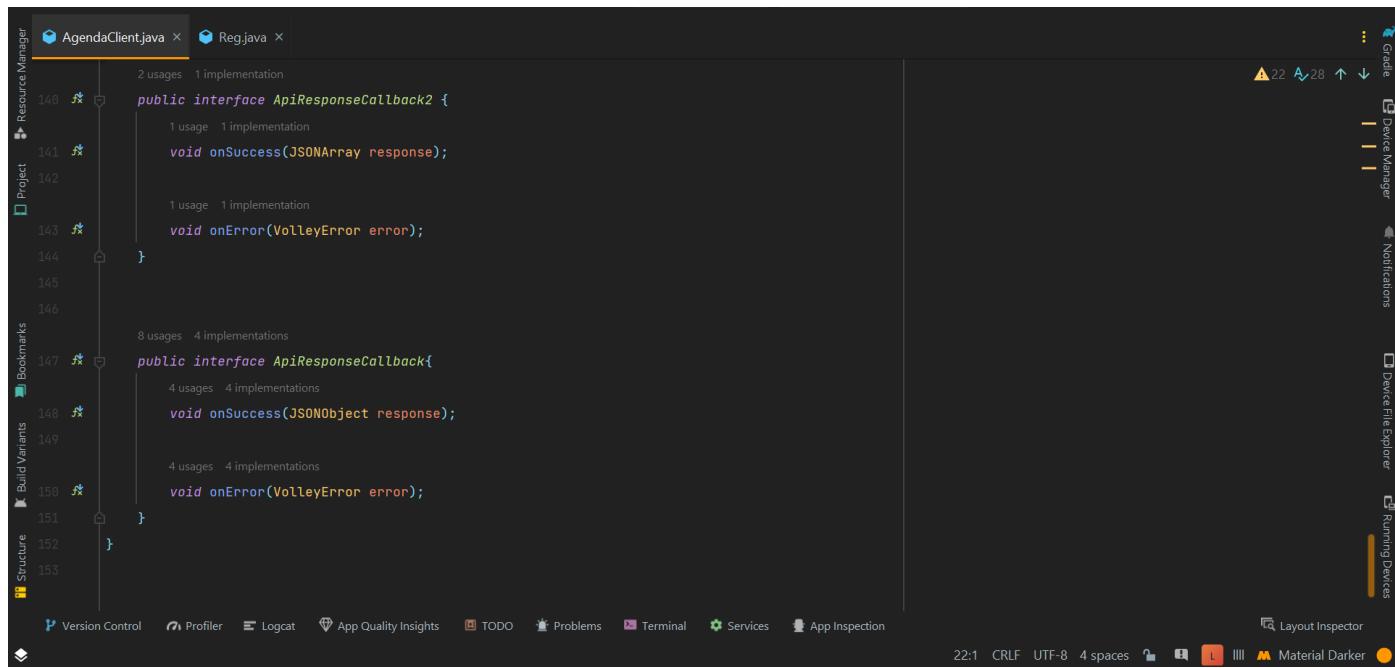
public void delete(final int id, final ApiResponseCallback callback) {
    String url = "http://192.168.24.85:8080/api/borrar/" + id;
    StringRequest request = new StringRequest(Request.Method.DELETE, url,
        new Response.Listener<String>() {
            ...
            @Override
            public void onResponse(String response) {
                // Llama al callback con una respuesta vacía
                callback.onSuccess(response);
            }
        },
        new Response.ErrorListener() {
            ...
            @Override
            public void onErrorResponse(VolleyError error) {
                // Llama al callback con el error
                callback.onError(error);
            }
        }
);

// Agrega la solicitud a la cola de solicitudes de Volley
requestQueue.add(request);
}
```

```
new Response.Listener<String>() {
    ...
    @Override
    public void onResponse(String response) {
        // Llama al callback con una respuesta vacía
        callback.onSuccess(response);
    }
},
new Response.ErrorListener() {
    ...
    @Override
    public void onErrorResponse(VolleyError error) {
        // Llama al callback con el error
        callback.onError(error);
    }
};

// Agrega la solicitud a la cola de solicitudes de Volley
requestQueue.add(request);
}
```

MANUAL DE PRÁCTICAS



The screenshot shows the Android Studio interface with two Java files open: AgendaClient.java and Reg.java. The code is related to Volley API callbacks. The AgendaClient.java file contains two interfaces: ApiResponseCallback2 and ApiResponseCallback. The Reg.java file contains a class that implements these interfaces.

```
public interface ApiResponseCallback2 {
    1 usage 1 implementation
    void onSuccess(JSONArray response);
}

1 usage 1 implementation
void onError(VolleyError error);

}

8 usages 4 implementations
public interface ApiResponseCallback{
    4 usages 4 implementations
    void onSuccess(JSONObject response);
    4 usages 4 implementations
    void onError(VolleyError error);
}
```

(AGENDA.java)

Este código representa una actividad de la aplicación de Android llamada "Agenda".

En resumen, las principales funciones de esta actividad son:

1. Configurar la interfaz de usuario (UI) mediante la asignación de elementos de diseño a variables.
2. Configurar el botón "Guardar" para agregar un nuevo elemento a la agenda cuando se hace clic en él.
3. Configurar el botón "Mostrar" para obtener y mostrar todos los elementos de la agenda cuando se hace clic en él.
4. Configurar el botón "ac" para abrir otra actividad llamada "Actualizar" después de un retraso de 500 milisegundos.
5. Configurar el botón "op" para abrir otra actividad llamada "Reg" después de un retraso de 500 milisegundos.
6. En el evento de clic del botón "Guardar", se crea un objeto de la clase "AgendaClient" y se utiliza para agregar un nuevo elemento a la agenda mediante una solicitud HTTP POST.
7. En el evento de clic del botón "Mostrar", se crea un objeto de la clase "AgendaClient" y se utiliza para obtener todos los elementos de la agenda mediante una solicitud HTTP GET. Los elementos obtenidos se procesan y se muestran en un ListView en la interfaz de usuario.
8. En el evento de clic del botón "ac", se inicia la actividad "Actualizar".
9. En el evento de clic del botón "op", se inicia la actividad "Reg".
10. Se utilizan varios métodos y clases auxiliares, como "Toast" para mostrar mensajes en pantalla, "Handler" para manejar los retrasos en la ejecución de tareas y clases relacionadas con la manipulación de solicitudes y respuestas HTTP utilizando la biblioteca Volley.

En general, esta actividad permite al usuario interactuar con la agenda, agregar nuevos elementos, mostrar los elementos existentes y acceder a otras funcionalidades de la aplicación mediante la navegación a otras actividades.



MANUAL DE PRÁCTICAS

The image shows two side-by-side screenshots of the Android Studio IDE. Both screens display Java code within the 'Code' tab of the editor.

Top Screenshot (AgendaClient.java):

```
package com.example.llll;
import ...
public class Agenda extends AppCompatActivity {
    private TextInputLayout nom;
    private TextInputLayout fecha;
    private TextInputLayout gus;
    private Button guardar;
    private Button mostrar;
    private AgendaClient agendaClient;
```

Bottom Screenshot (Agenda.java):

```
private Button op;
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_agenda);
    nom = findViewById(R.id.Nombre);
    fecha = findViewById(R.id.FechaCumple);
    gus = findViewById(R.id.Gustos);
    guardar = findViewById(R.id.btnAgregar);
    mostrar = findViewById(R.id.button2);
    ListView listView = findViewById(R.id.listView);
    ac = findViewById(R.id.btac);
    op = findViewById(R.id.opc2);
    guardar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            agendaClient = new AgendaClient(context: Agenda.this);
            try {
```

Both screenshots show the standard Android Studio interface with toolbars, status bars, and various developer tools like Layout Inspector and Device Manager visible on the right.



MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface with two code files open:

- AgendaClient.java:** Contains logic for validating input fields (nombre, fecha, gustos) and creating a new JSON object (nuevoalumno) to pass to the agendaClient.agregar() method.
- Agenda.java:** Contains methods for onSuccess and onError handling, and a catch block for JSONException. It also includes a section for setting up a click listener on a button (mostrar).

Both files include imports for Context, Toast, and VolleyError, and use the `Toast.makeText()` method to display messages to the user.



MANUAL DE PRÁCTICAS

The screenshot shows two tabs open in the code editor: `AgendaClient.java` and `Agenda.java`. The `AgendaClient.java` tab is active, displaying the following code:

```
agendaClient.mostrar(new AgendaClient.ApiResponseCallback2() {
    1 usage
    @Override
    public void onSuccess(JSONArray response) {
        try {
            StringBuilder data = new StringBuilder();

            if (response.length() == 0) {
                // No hay registros disponibles
                data.append("No hay registros disponibles");
            } else {
                for (int i = 0; i < response.length(); i++) {
                    JSONObject jsonObject = response.getJSONObject(i);

                    // Verifican si los campos en el objeto JSON están vacíos
                    if (!jsonObject.isNull("id") && !jsonObject.isNull("nombre")
                        && !jsonObject.isNull("fechaCumple") && !jsonObject.isNull("gustos")) {

                        // Obtener los valores de los campos en cada objeto JSON
                        Long id = jsonObject.getLong("id");
                        String nombre = jsonObject.getString("nombre");

```

The `Agenda.java` tab is also visible, showing the following code:

```
// Construir la cadena de texto para cada registro
String recordData = "ID: " + id + "\nNombre: " + nombre +
    "\nFecha Cumple: " + fecha + "\nGustos: " + gus;

// Agregar el registro a la cadena de texto general
data.append(recordData).append("\n\n");

}

// Verificar si no se encontraron registros válidos
if (data.length() == 0) {
    data.append("No se encontraron registros válidos");
}

// Asigna el texto al TextView
String[] records = data.toString().split( regex: "\n\n" );
ArrayAdapter<String> adapter = new ArrayAdapter<>( context: Agenda.this, android.R.layout.simple_list_item_1, records);

listView.setAdapter(adapter);

```



MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface with two code editors open. The top editor contains the `Agenda.java` file, which includes code for handling JSON exceptions and displaying error messages using `Toast.makeText`. The bottom editor contains the `AgendaClient.java` file, which sets up a click listener for a view and starts an activity using `startActivity`. Both files are part of the `Agenda` project, as indicated by the tabs and the code itself. The interface includes standard Android Studio toolbars and side panels for navigation and device management.

```
141 } catch (JSONException e) {
142     e.printStackTrace();
143     Toast.makeText(context, "Error al procesar la respuesta del servidor", Toast.LENGTH_SHORT).show();
144 }
145 }
146
147     1 usage
148     @Override
149     public void onError(VolleyError error) {
150         Toast.makeText(context, error.toString(), Toast.LENGTH_SHORT).show();
151     }
152 }
153 }
154
155 ac.setOnClickListener(new View.OnClickListener() {
156     @Override
157     public void onClick(View v) {
158         new Handler().postDelayed(new Runnable() {
159             @Override
160             public void run() {
```

```
164     }, delayMillis: 500);
165 }
166 }
167
168
169 op.setOnClickListener(new View.OnClickListener() {
170     @Override
171     public void onClick(View v) {
172         new Handler().postDelayed(new Runnable() {
173             @Override
174             public void run() {
175                 Intent en = new Intent(packageContext, Reg.class);
176                 startActivity(en);
177                 finish();
178             }
179         }, delayMillis: 500);
180     }
181 }
182 }
```



MANUAL DE PRÁCTICAS

(REG.java)

En resumen, las principales funciones de esta actividad son:

1. Configurar la interfaz de usuario (UI) mediante la asignación de elementos de diseño a variables.
 2. Configurar el botón "Buscar" para buscar un registro en la agenda según un ID especificado por el usuario.
 3. Configurar el botón "Borrar" para eliminar un registro de la agenda según un ID especificado por el usuario.
 4. Configurar el botón "Volver" para regresar a la actividad principal de la agenda después de un retraso de 500 milisegundos.
 5. En el evento de clic del botón "Buscar", se crea un objeto de la clase "AgendaClient" y se utiliza para buscar un registro en la agenda mediante una solicitud HTTP GET.
 6. En el evento de clic del botón "Borrar", se crea un objeto de la clase "AgendaClient" y se utiliza para eliminar un registro de la agenda mediante una solicitud HTTP DELETE.
 7. En el evento de clic del botón "Volver", se inicia la actividad principal de la agenda.
 8. Se utilizan varios métodos y clases auxiliares, como "Toast" para mostrar mensajes en pantalla, "Handler" para manejar los retrasos en la ejecución de tareas y clases relacionadas con la manipulación de solicitudes y respuestas HTTP utilizando la biblioteca Volley.

En general, esta actividad permite al usuario buscar y eliminar registros específicos de la agenda mediante la interacción con los elementos de la interfaz de usuario. También proporciona una opción para volver a la actividad principal de la agenda.

The screenshot shows the Android Studio interface with the following details:

- Code Editor:** The main area displays Java code for a class named `Reg`. The code includes imports, class definition, field declarations (private `TextInputLayout idd`, `Button bus`, `Button del`, `Button vol`, and `AgendaClient agendaClient`), and usage counts (12 usages for `Reg`, 3 usages for `idd`, 2 usages for `bus`, 2 usages for `del`, 2 usages for `vol`, and 4 usages for `agendaClient`). A yellow vertical bar highlights the `bus` variable.
- Toolbars and Side Panels:**
 - Project:** Shows the project structure with `src/main/java/com/example/lili/Reg.java` selected.
 - Bookmarks:** Shows a list of bookmarks.
 - Build Variants:** Shows the current build variant.
 - Structure:** Shows the code structure.
 - Version Control:** Shows the version control status.
 - Profiler:** Shows the profiler status.
 - Logcat:** Shows the logcat status.
 - App Quality Insights:** Shows the app quality insights status.
 - TODO:** Shows the TODO status.
 - Problems:** Shows the problems status.
 - Terminal:** Shows the terminal status.
 - Services:** Shows the services status.
 - App Inspection:** Shows the app inspection status.
- Device File Explorer:** A panel on the right showing connected devices: `Running Devices`.
- Bottom Bar:** Includes icons for Layout Inspector, Material Darker, and other developer tools.



MANUAL DE PRÁCTICAS

The screenshot shows two side-by-side code editors in the Android Studio IDE. Both editors have identical toolbars at the top and bottom.

Top Editor (AgendaClient.java):

```
34
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.activity_agendareg);
38         idd = findViewById(R.id.txid);
39         bus = findViewById(R.id.btnBuscar);
40         del = findViewById(R.id.btnBorrar);
41         TextView list = findViewById(R.id.txtre);
42         vol = findViewById(R.id.btnvol);
43
44         bus.setOnClickListener(new View.OnClickListener() {
45             @Override
46             public void onClick(View v) {
47                 agendaClient = new AgendaClient(context: Reg.this);
48                 try {
49                     int id = Integer.parseInt(idd.getEditText().getText().toString());
50                     agendaClient.findById(id, new AgendaClient.ApiResponseCallback() {
51                         4 usages
52                         @Override
53                         public void onSuccess(JSONObject response) {
54                             try {
55
56                             
```

Bottom Editor (Reg.java):

```
51
52     @Override
53     public void onSuccess(JSONObject response) {
54         try {
55             JSONObject jsonResponse = new JSONObject(response.toString());
56             int iden = jsonResponse.getInt(name: "id");
57             String nombre = jsonResponse.getString(name: "nombre");
58             String fecha = jsonResponse.getString(name: "fechaCumple");
59             String gustos = jsonResponse.getString(name: "gustos");
60
61             String data = "ID : " + iden + "\nNombre: " + nombre +
62                         "\nFecha Cumple: " + fecha + "\nGustos: " + gustos;
63             list.setText(data);
64         } catch (JSONException e) {
65             Toast.makeText(context: Reg.this, e.getMessage(), Toast.LENGTH_SHORT).show();
66         }
67     }
68
69     4 usages
70     @Override
71     public void onError(VolleyError error) {
72         list.setText(null);
73     }
74
75 
```



MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface with two code files open:

- AgendaClient.java:** This file contains code for a delete operation. It includes a try-catch block for `NumberFormatException`, which displays an error message if no number is entered. It also includes an `onClick` listener for a button that performs a delete operation via a callback.
- Reg.java:** This file contains code for an `onError` handler for a `VolleyError`. It displays an error message if the deletion fails. It also includes an `onClick` listener for a button that starts a new intent.

The code is written in Java and uses `Toast.makeText` for displaying messages to the user.



MANUAL DE PRÁCTICAS

```
183     vol.setOnClickListener(new View.OnClickListener() {
184         @Override
185         public void onClick(View v) {
186             new Handler().postDelayed(new Runnable() {
187                 @Override
188                 public void run() {
189                     Intent en = new Intent( mContext, Agenda.class );
190                     startActivity(en);
191                     finish();
192                 }
193             }, delayMillis: 500 );
194         }
195     });
196 }
```

(INTERFAZ) (Activity_Agenda1.xml)

En resumen, se está creando una interfaz de usuario compuesta por los siguientes elementos:

- Un LinearLayout principal con orientación vertical y fondo de color #03ECF4.
- Un TextView con ID "textView2" que muestra el texto "INICIO :" en color blanco.
- Tres TextInputLayouts con IDs "Nombre", "FechaCumple" y "Gustos", respectivamente. Cada uno contiene un TextInputEditText para que el usuario pueda ingresar datos de nombre, fecha de cumpleaños y gustos.
- Un LinearLayout horizontal que contiene dos MaterialButtons con IDs "btnAgregar" y "button2". Estos botones tienen texto, color de texto y color de fondo definidos.
- Otro LinearLayout horizontal que contiene dos MaterialButtons con IDs "btac" y "opc2". Estos botones también tienen texto, color de texto y color de fondo definidos.
- Un ScrollView que envuelve un LinearLayout vertical que contiene un ListView con ID "listView". Este ListView se utiliza para mostrar una lista de elementos.

En general, este diseño XML crea una interfaz de usuario que permite al usuario ingresar datos en campos de texto, realizar acciones como agregar, mostrar, actualizar y buscar, y muestra una lista de elementos en un ListView.

-Digamos que es como si fuera el menú principal.



MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** Shows the project tree with modules like `AgCumples`, `src/main/res/layout`, and files like `activity_actua.xml`, `activity_agenda1.xml`, `activity_agendareg.xml`, and `activity_main.xml`.
- Code Editor:** Displays the XML code for `activity_agenda1.xml`. The code includes a `LinearLayout` with a `TextView` and a `TextInputLayout` for a text input field.
- Preview:** A visual representation of the app's interface shows a screen titled "INICIO" with fields for "Nombre", "Fecha Cumplidos", and "Gustos", along with buttons for "AGREGAR", "MOSTRAR", "ACTUALIZAR", and "BORRAR Y BUSCAR".
- Bottom Bar:** Includes standard Android Studio icons for Version Control, Profiler, Logcat, App Quality Insights, TODO, Problems, Terminal, Services, and App Inspection, along with system status like weather (18°C), battery level, and date (23/6/2023).

The second screenshot is identical to the first one, showing the same project structure, code editor with XML for `activity_agenda1.xml`, preview of the "INICIO" screen, and the bottom Android Studio bar.



MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface with the project 'AgCumples' open. The code editor displays the XML layout for 'activity_agenda1.xml'. The layout includes a 'TextInputLayout' for 'Fecha Cumpleaños' with a hint 'Fecha Cumpleaños:' and text color '#FFFFFF'. It also includes another 'TextInputLayout' for 'Gustos' with a hint 'Gustos:' and text color '#FFFFFF'. Below these are two 'MaterialButton' components: one labeled 'AGREGAR' and another labeled 'MOSTRAR'. A 'Component Tree' panel on the right shows the hierarchical structure of the layout. The preview window on the right shows a teal-themed mobile application interface with a navigation bar, a list of items, and the buttons from the XML code.

18°C Despejado Búsqueda APP PIXEL 6 API 30

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Other ||| - activity_agenda1.xml [|||.app.main]

AgCumples > app > src > main > res > layout > </> activity_agenda1.xml

Android Project Bookmarks Build Variants Structure

Resource Manager

activity_actua.xml activity_agenda1.xml activity_agendareg.xml activity_main.xml

40 android:hint="Fecha Cumpleaños:"
41 android:textColor="#FFFFFF" />
42 </com.google.android.material.textfield.TextInputLayout>
43
44 <com.google.android.material.textfield.TextInputLayout
45 android:id="@+id/gustos"
46 android:layout_width="match_parent"
47 android:layout_height="wrap_content"
48 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedText
49
50 <com.google.android.material.textfield.TextInputEditText
51 android:layout_width="match_parent"
52 android:layout_height="wrap_content"
53 android:hint="Gustos:"
54 android:textColor="#FFFFFF" />
55 </com.google.android.material.textfield.TextInputLayout>
56
57 <LinearLayout
58 android:layout_width="match_parent"

Component Tree

activity_agenda1.xml

AGREGAR MOSTRAR
ACTUALIZAR BORRAR Y BUSCAR

Item 1 Sub ítem 1
Item 2 Sub ítem 2
Item 3 Sub ítem 3
Item 4 Sub ítem 4

14:32 CRLF UTF-8 4 spaces L Material Darker

Version Control Profiler Logcat App Quality Insights TODO Problems Terminal Services App Inspection

18°C Despejado Búsqueda APP PIXEL 6 API 30

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Other ||| - activity_agenda1.xml [|||.app.main]

AgCumples > app > src > main > res > layout > </> activity_agenda1.xml

Android Project Bookmarks Build Variants Structure

Resource Manager

activity_actua.xml activity_agenda1.xml activity_agendareg.xml activity_main.xml

58 android:layout_width="match_parent"
59 android:layout_height="wrap_content"
60 android:orientations="horizontal"
61 android:layout_marginTop="16dp" />
62
63 <com.google.android.material.button.MaterialButton
64 android:id="@+id btnAgregar"
65 style="@style/Widget.MaterialComponents.Button"
66 android:layout_width="0dp"
67 android:layout_height="match_parent"
68 android:layout_weight="1"
69 android:text="AGREGAR"
70 android:textColor="#FFFFFF"
71 android:backgroundTint="#2196F3"
72 android:layout_marginEnd="8dp" />
73
74 <com.google.android.material.button.MaterialButton
75 android:id="@+id/button2"
76 style="@style/Widget.MaterialComponents.Button"

Component Tree

activity_agenda1.xml

AGREGAR MOSTRAR
ACTUALIZAR BORRAR Y BUSCAR

Item 1 Sub ítem 1
Item 2 Sub ítem 2
Item 3 Sub ítem 3
Item 4 Sub ítem 4

14:32 CRLF UTF-8 4 spaces L Material Darker

Version Control Profiler Logcat App Quality Insights TODO Problems Terminal Services App Inspection

18°C Despejado Búsqueda APP PIXEL 6 API 30



MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** Shows the project tree with modules like AgCumples, com.example.lli, and java (generated), and resources like res/drawable, res/font, and res/layout.
- Code Editor:** Displays the XML code for `activity_agenda1.xml`. The code defines a linear layout with two buttons: "ACTUALIZAR" and "BORRAR Y BUSCAR".
- Preview:** A visual representation of the app's user interface, showing a screen titled "INICIO" with fields for "Nombre" and "Fecha Cumplidos", and buttons for "AGREGAR", "MOSTRAR", "ACTUALIZAR", and "BORRAR Y BUSCAR". Below these are four items with sub-items.
- Bottom Bar:** Includes icons for Version Control, Profiler, Logcat, App Quality Insights, TODO, Problems, Terminal, Services, and App Inspection, along with system status like battery level, signal strength, and date/time (14:32, 23/6/2023).

This screenshot illustrates the first step in the practical guide, which involves creating the XML layout for the agenda screen.

The second screenshot is identical to the first one, showing the same XML code for `activity_agenda1.xml` and the same preview of the app's user interface. This likely represents a continuation or a second step in the practical guide.



MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, Other, and a tab for activity_agenda1.xml. The left sidebar displays the Project structure, showing packages like AgCumples, com.example.lli, and java (generated), along with resources (res) and layouts (layout). The main editor area contains the XML code for activity_agenda1.xml, which defines a ScrollView containing a ListView. The preview pane on the right shows a mobile application interface with a header, buttons for AGREGAR, MOSTRAR, ACTUALIZAR, and BORRAR Y BUSCAR, and a list of items. The bottom status bar shows the time as 14:32, the date as 23/6/2023, and the battery level at 18°C.

```
112    android:backgroundTint="#2196F3"
113    android:layout_marginStart="8dp" />
114
115</LinearLayout>
116
117<ScrollView
118    android:layout_width="match_parent"
119    android:layout_height="0dp"
120    android:layout_weight="1"
121    android:layout_marginTop="16dp">
122
123<LinearLayout
124    android:layout_width="match_parent"
125    android:layout_height="wrap_content"
126    android:gravity="center"
127    android:orientation="vertical">
128
129<ListView
130    android:id="@+id/listView" />
131
132</LinearLayout>
133
134</ScrollView>
135
136</LinearLayout>
```

This screenshot is identical to the one above, showing the Android Studio interface with the XML code for activity_agenda1.xml and its preview pane. The code is identical, showing the structure of the ScrollView and ListView. The preview pane shows the same mobile application interface. The bottom status bar shows the time as 14:32, the date as 23/6/2023, and the battery level at 18°C.

```
112    android:backgroundTint="#2196F3"
113    android:layout_marginStart="8dp" />
114
115</LinearLayout>
116
117<ScrollView
118    android:layout_width="match_parent"
119    android:layout_height="wrap_content"
120    android:layout_weight="1"
121    android:layout_marginTop="16dp">
122
123<LinearLayout
124    android:layout_width="match_parent"
125    android:layout_height="wrap_content"
126    android:gravity="center"
127    android:orientation="vertical">
128
129<ListView
130    android:id="@+id/listView" />
131
132</LinearLayout>
133
134</ScrollView>
135
136</LinearLayout>
```

(Activity_actua.xml)

MANUAL DE PRÁCTICAS

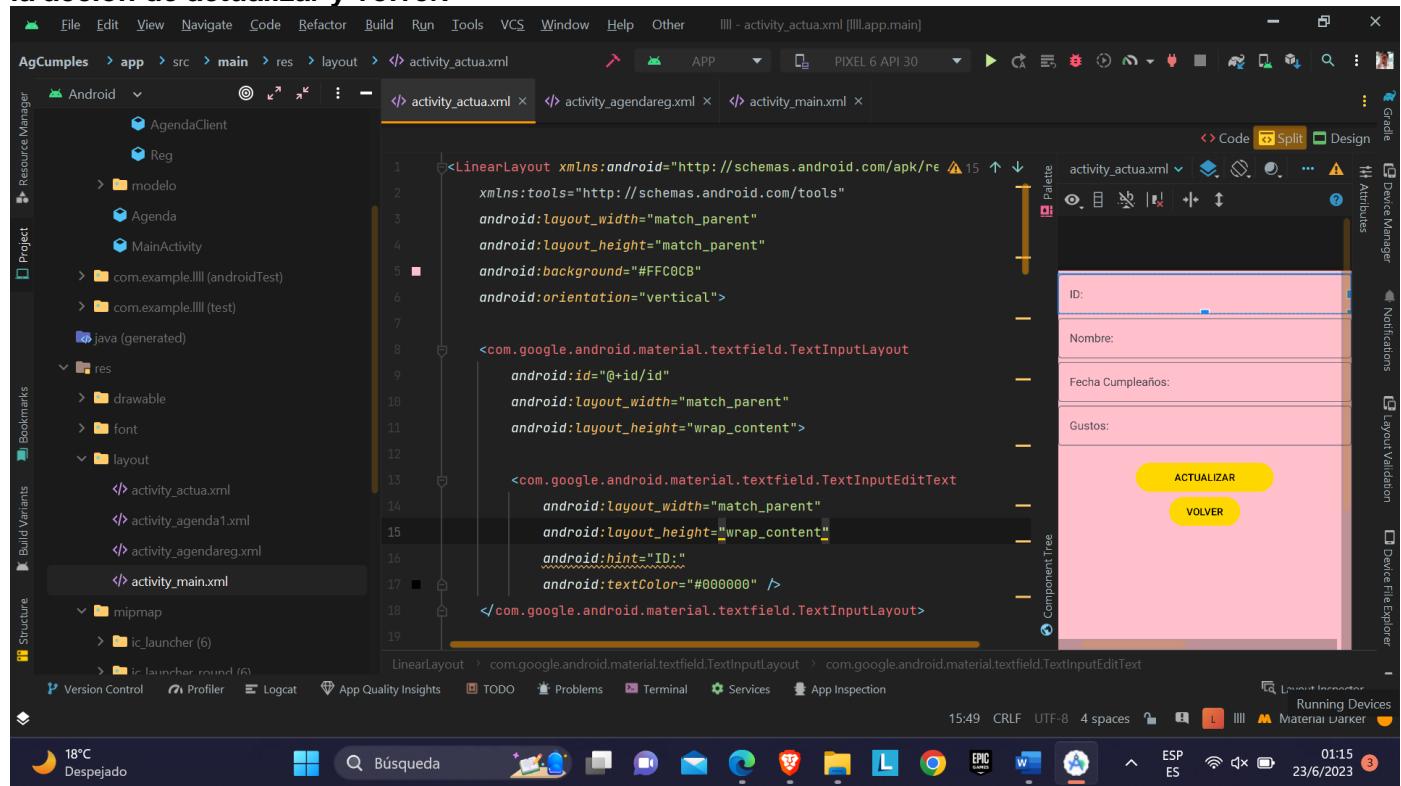
En este código XML se está creando una interfaz de usuario en Android, utilizando un **LinearLayout** como contenedor principal con orientación vertical y fondo de color #FFC0CB.

Dentro del **LinearLayout** se encuentran varios **TextInputLayouts**, cada uno con su correspondiente **TextInputEditText**. Estos elementos permiten al usuario ingresar datos en campos de texto para ID, Nombre, Fecha de Cumpleaños y Gustos.

Luego, hay un **LinearLayout horizontal** que contiene un **Button** con ID "ac", el cual se utiliza para la acción de actualizar. Tiene un texto "ACTUALIZAR" y un color de fondo y texto definidos.

Finalmente, hay otro **Button** con ID "button" que se utiliza para volver. También tiene un texto "VOLVER" y un color de fondo y texto definidos.

En resumen, esta interfaz de usuario permite al usuario ingresar datos en campos de texto, realizar la acción de actualizar y volver.



La captura de pantalla muestra la interfaz de desarrollo de Android Studio. En la parte superior, se ve el menú y las herramientas de desarrollo. Abajo, el proyecto "AgCumples" es visible, con el archivo "activity_actua.xml" abierto. El código XML es:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFC0CB"
    android:orientation="vertical">

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/id"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <com.google.android.material.textfield.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="ID:"
            android:textColor="#000000" />
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/nombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <com.google.android.material.textfield.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Nombre:" />
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/fecha"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <com.google.android.material.textfield.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Fecha Cumpleaños:" />
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/gustos"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <com.google.android.material.textfield.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Gustos:" />
    </com.google.android.material.textfield.TextInputLayout>

</LinearLayout>
```

A la derecha, el diseño visual muestra tres campos de texto y un botón amarillo "ACTUALIZAR". Debajo del botón, hay otro botón amarillo "VOLVER". Los campos de texto tienen colores y estilos definidos. La barra de herramientas abajo incluye iconos para búsqueda, terminal, servicios, etc., y muestra la fecha y hora actual.



MANUAL DE PRÁCTICAS

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/Nombre"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Nombre:"
        android:textColor="#000000" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/FechaCumpleaños"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Fecha Cumpleaños:"
        android:textColorHighlight="#673AB7"
        android:textColor="#000000" />
</com.google.android.material.textfield.TextInputLayout>
```

18°C Despejado Búsqueda APP PIXEL 6 API 30

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Other

AgCumples > app > src > main > res > layout > activity_actua.xml

Android Project Bookmarks Build Variants Structure

Resource Manager

activity_actua.xml activity_agendareg.xml activity_main.xml

19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37

Nombre:

Fecha Cumpleaños:

Gustos:

ACTUALIZAR VOLVER

Component Tree

LinearLayout > com.google.android.material.textfield.TextInputLayout > com.google.android.material.textfield.TextInputEditText

Version Control Profiler Logcat App Quality Insights TODO Problems Terminal Services App Inspection

15:49 CRLF UTF-8 4 spaces L Material Darker

ESP ES 01:15 23/6/2023

```
<com.google.android.material.textfield.TextInputEditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Nombre:"
    android:textColorHighlight="#673AB7"
    android:textColor="#000000" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/Gustos"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Gustos:"
        android:textColor="#000000" />
</com.google.android.material.textfield.TextInputLayout>
```

18°C Despejado Búsqueda APP PIXEL 6 API 30

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Other

AgCumples > app > src > main > res > layout > activity_actua.xml

Android Project Bookmarks Build Variants Structure

Resource Manager

activity_actua.xml activity_agendareg.xml activity_main.xml

37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55

Nombre:

Fecha Cumpleaños:

Gustos:

ACTUALIZAR VOLVER

Component Tree

LinearLayout > com.google.android.material.textfield.TextInputLayout > com.google.android.material.textfield.TextInputEditText

Version Control Profiler Logcat App Quality Insights TODO Problems Terminal Services App Inspection

50:40 CRLF UTF-8 4 spaces L Material Darker

ESP ES 01:15 23/6/2023



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Other ||| - activity_actua.xml

AgCumples > app > src > main > res > layout > activity_actua.xml APP PIXEL 6 API 30

Android Project Bookmarks Build Variants Structure Bookmarks Build Variants Structure

Resource Manager

AgendaClient Reg modelo Agenda MainActivity com.example.lli (androidTest) com.example.lli (test) java (generated)

res drawable font layout activity_actua.xml activity_agenda1.xml activity_agendareg.xml activity_main.xml mipmap ic_launcher (6) ic_launcher_round (6)

activity_actua.xml x activity_agendareg.xml x activity_main.xml x

```

<com.google.android.material.textfield.TextInputLayout>
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:orientation="horizontal">

    <LinearLayout
        android:id="@+id/ac"
        android:layout_width="23dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="20dp"
        android:layout_marginRight="10dp"
        android:layout_weight="1"
        android:gravity="center"
        android:text="ACTUALIZAR"
        android:backgroundTint="#FFD700" />

```

Palette Attributes Device Manager Notifications LayoutValidation Device File Explorer

ID: Nombre: Fecha Cumpleaños: Gustos: ACTUALIZAR VOLVER

Component Tree

Version Control Profiler Logcat App Quality Insights TODO Problems Terminal Services App Inspection

50:40 CRLF UTF-8 4 spaces L Material Darker

18°C Despejado Búsqueda ESP ES 01:16 23/6/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Other ||| - activity_actua.xml

AgCumples > app > src > main > res > layout > activity_actua.xml APP PIXEL 6 API 30

Android Project Bookmarks Build Variants Structure Bookmarks Build Variants Structure

Resource Manager

AgendaClient Reg modelo Agenda MainActivity com.example.lli (androidTest) com.example.lli (test) java (generated)

res drawable font layout activity_actua.xml activity_agenda1.xml activity_agendareg.xml activity_main.xml mipmap ic_launcher (6) ic_launcher_round (6)

activity_actua.xml x activity_agendareg.xml x activity_main.xml x

```

        android:textColor="#000000" />

    </LinearLayout>
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="VOLVER"
        android:backgroundTint="#FFD700"
        android:textColor="#000000" />

```

Palette Attributes Device Manager Notifications LayoutValidation Device File Explorer

ID: Nombre: Fecha Cumpleaños: Gustos: ACTUALIZAR VOLVER

Component Tree

Version Control Profiler Logcat App Quality Insights TODO Problems Terminal Services App Inspection

50:40 CRLF UTF-8 4 spaces L Material Darker

18°C Despejado Búsqueda ESP ES 01:16 23/6/2023

MANUAL DE PRÁCTICAS

(Activity_Agendareg.xml)

En este código XML se está creando una interfaz de usuario en Android utilizando un LinearLayout como contenedor principal con orientación vertical y fondo de color #03A9F4.

Dentro del LinearLayout se encuentra un TextView con ID "textView3" que muestra el mensaje "INGRESA EL ID A BUSCAR O ELIMINAR" y tiene un tamaño de texto de 20sp y color de texto blanco.

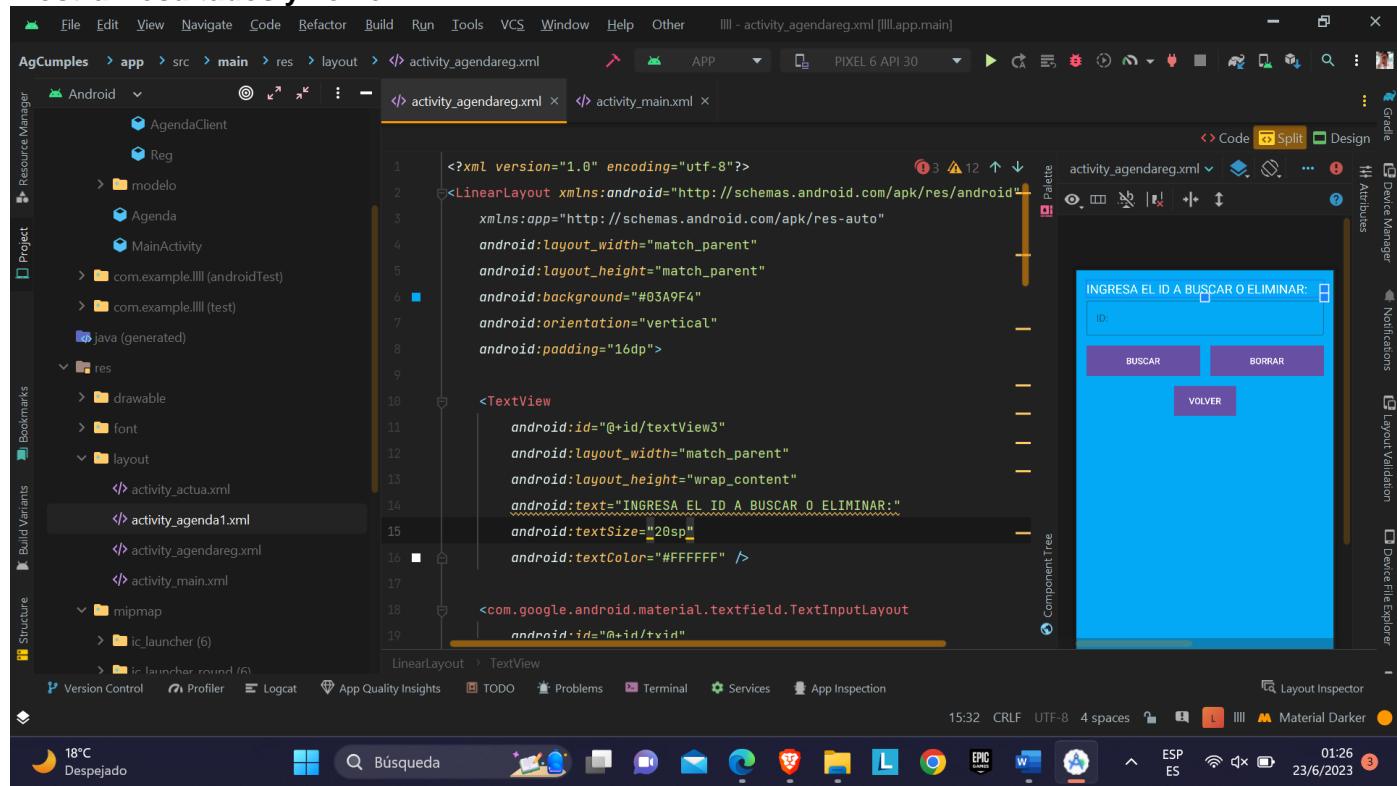
Luego, hay un TextInputLayout con ID "txid" que contiene un TextInputEditText. Este elemento permite al usuario ingresar un ID en un campo de texto.

A continuación, hay un LinearLayout horizontal que contiene dos botones. El primer botón con ID "btnBuscar" tiene el texto "BUSCAR", color de texto blanco y color de fondo #4CAF50. El segundo botón con ID "btnBorrar" tiene el texto "BORRAR", color de texto blanco y color de fondo #F44336.

Después, hay un botón con ID "btncvol" que se utiliza para volver. Tiene el texto "VOLVER", color de texto blanco y color de fondo #9E9E9E.

Finalmente, hay un TextView con ID "txtre" que se utiliza para mostrar resultados. Tiene un ancho de match_parent, una altura de 163dp, un margen superior de 20dp, un texto vacío y un color de texto blanco.

En resumen, esta interfaz de usuario permite al usuario ingresar un ID, buscar y borrar registros, mostrar resultados y volver.



El código XML es:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#03A9F4"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="INGRESA EL ID A BUSCAR O ELIMINAR:"
        android:textSize="20sp"
        android:textColor="#FFFFFF" />

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/txid">
```

La vista previa muestra:

- Un TextView con el texto "INGRESA EL ID A BUSCAR O ELIMINAR:".
- Un TextInputLayout con un TextInputEditText.
- Un LinearLayout horizontal con dos botones: "BUSCAR" (color #4CAF50) y "BORRAR" (color #F44336).
- Un botón "VOLVER" con el texto "VOLVER".
- Un TextView vacío con el id "txtre".



MANUAL DE PRÁCTICAS

```
style="@style/Widget.MaterialComponents.TextInputEditText" android:layout_width="match_parent" android:layout_height="wrap_content" android:hint="ID" android:textColor="#FFFFFF" />

<com.google.android.material.textfield.TextInputEditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="ID"
    android:textColor="#FFFFFF" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="16dp">

    <Button
        android:id="@+id/btnBuscar"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="BUSCAR"
        android:textColor="#FFFFFF"
        android:background="#4CAF50"
        android:layout_marginEnd="8dp" />

    <Button
        android:id="@+id/btnBorrar"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="BORRAR"
        android:textColor="#FFFFFF"
        android:background="#F44336"
        android:layout_marginStart="8dp" />

    <Button
        android:id="@+id/btnVolver"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="VOLVER"
        android:textColor="#FFFFFF"
        android:background="#E91E63" />

```



MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface with the project structure on the left and two XML files open in the center. The top file is `activity_main.xml`, which contains the following code:

```
<LinearLayout
    android:id="@+id/linearLayout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="VOLVER"
    android:textColor="#FFFFFF"
    android:background="#9E9E9E"
    android:layout_marginTop="16dp" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="match_parent"
    android:layout_height="163dp"
    android:layout_marginTop="20dp"
    android:text=""
    android:textColor="#FFFFFF" />
```

The bottom file is `activity_agendareg.xml`. On the right, the preview window shows a blue screen with a white text input field labeled "ID", a purple button labeled "BUSCAR", a purple button labeled "BORRAR", and a blue button labeled "VOLVER".

(activity_main.xml)

En este código XML se está creando una interfaz de usuario en Android utilizando un `LinearLayout` como contenedor principal con orientación vertical y gravedad centrada. Este código se utiliza en la actividad principal (`MainActivity`) de la aplicación.

Dentro del `LinearLayout`, se encuentra un `ImageView` con ID "imageView" que muestra una imagen de un registro. Tiene un ancho de 182dp, una altura de 167dp y visibilidad visible.

Luego, hay un `TextView` con ID "textView4" que muestra el texto "BIENVENIDO MASTER" y tiene un tamaño de texto de 24sp. El ancho del `TextView` se ajusta al ancho del contenedor principal.

A continuación, se encuentra un `ProgressBar` con ID "progressBar" que se utiliza para mostrar una barra de progreso. Tiene un estilo predefinido de `progressBarStyle` y un ancho que se ajusta al ancho del contenedor principal. El color de progreso de la barra se establece en #2196F3.

En resumen, esta interfaz de usuario muestra una imagen de registro, un mensaje de bienvenida y una barra de progreso en el centro de la pantalla.



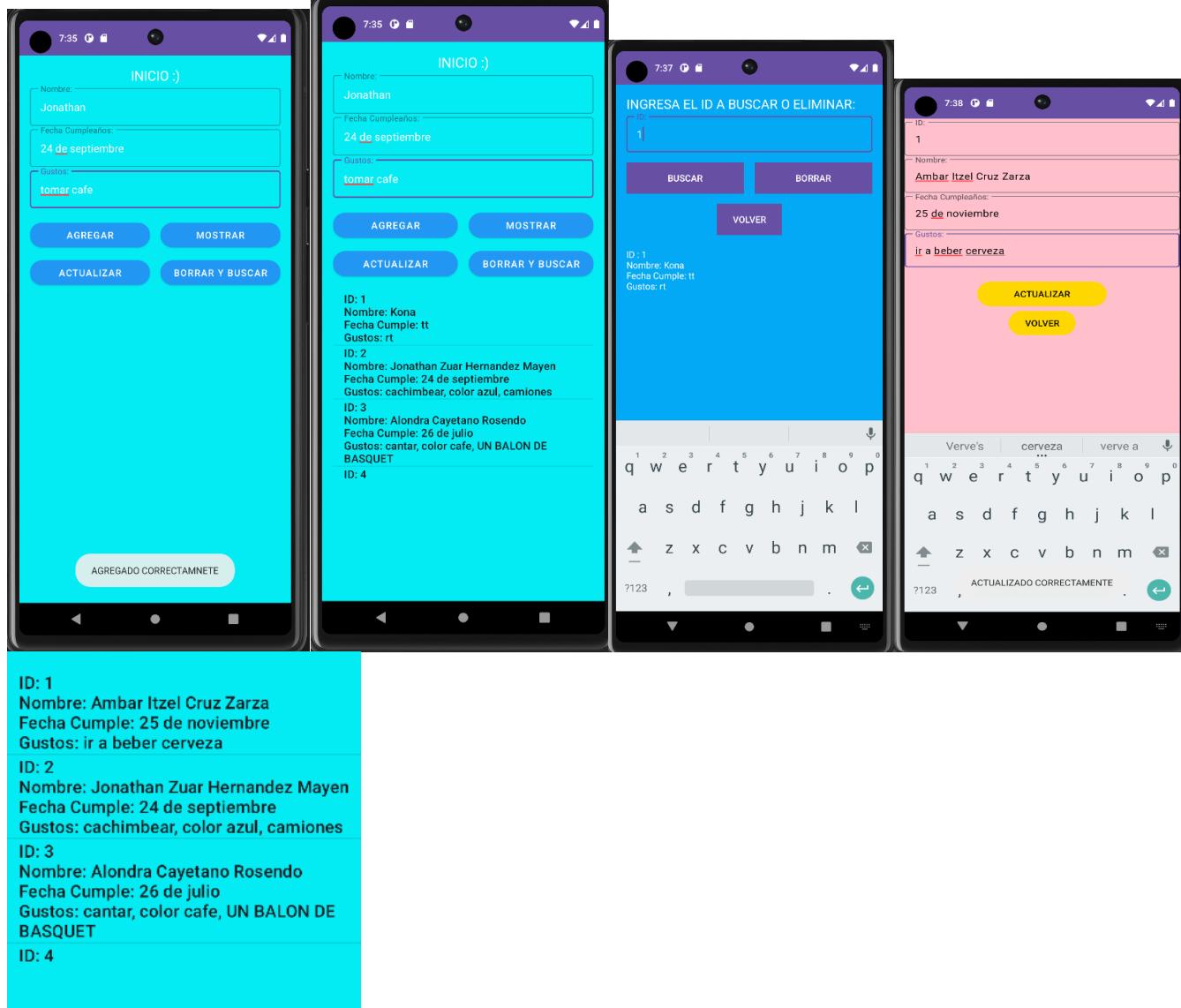
MANUAL DE PRÁCTICAS

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** Shows the file path: AgCumples > app > src > main > res > layout > activity_main.xml.
- Code Editor:** Displays the XML code for activity_main.xml. The code includes a LinearLayout with a vertical orientation, a central ImageView with a registration logo, and a TextView displaying "BIENVENIDO MASTER".
- Preview Pane:** Shows a visual representation of the layout. It features a white background with a blue progress bar at the bottom. In the center, there is a blue rounded rectangle containing the text "BIENVENIDO MASTER" in bold black font. Above it, there is a smaller blue rounded rectangle with a white registration logo.
- Bottom Bar:** Includes standard Android Studio icons for Version Control, Profiler, Logcat, App Quality Insights, TODO, Problems, Terminal, Services, and App Inspection, along with system status icons like battery level, signal strength, and network connection.

MANUAL DE PRÁCTICAS

EJECUCIÓN:



CONCLUSIÓN:

Este proyecto me ha permitido utilizar una variedad de herramientas y tecnologías para su desarrollo. Para la creación del API, utilicé IntelliJ IDEA como entorno de desarrollo, aprovechando su funcionalidad y características para facilitar la codificación y depuración del código.

Además, para el servidor local, utilicé XAMPP, una herramienta que proporciona un entorno de servidor web Apache y base de datos MySQL, lo que me permitió simular un entorno de producción para probar y ejecutar la API de manera local.

Por otro lado, para el desarrollo de la aplicación móvil, utilicé Android Studio, una herramienta específicamente diseñada para crear aplicaciones Android. Android Studio ofrece un conjunto completo de herramientas y recursos para desarrollar, depurar y empaquetar aplicaciones Android, lo que facilitó la integración y consumo de la API en el proyecto.

MANUAL DE PRÁCTICAS

Por último, para la creación del proyecto API, utilicé Spring, un framework de desarrollo de aplicaciones Java, que proporciona una arquitectura modular y flexible. Spring simplificó la creación de la API, facilitando la definición de rutas, la implementación de controladores y la gestión de datos.

En resumen, esta experiencia me permitió utilizar herramientas como IntelliJ IDEA, XAMPP, Android Studio y Spring para desarrollar tanto el API como la aplicación móvil, brindándome un enfoque práctico y aplicado en el uso de estas tecnologías para crear y consumir una API en un entorno de desarrollo completo.

GIT-HUB DEL PROYECTO:

https://github.com/Jona163/Proyecto-Creacion_API