**Actividad N. 4**

**Aplicación móvil en Android Studio consumiendo Api**

Presentado por:

**LUIS FELIPE DIAZ CABANA**

**LUIGGI MAR SALEMI SOTO**

**Universidad de Cartagena**

**Ingeniería de software**

**Desarrollo de apps**

**2024**

**Tabla de Contenidos**

[Introducción 3](#_Toc168955565)

[Objetivo general 4](#_Toc168955566)

[Objetivos específicos 4](#_Toc168955567)

[Justificación 5](#_Toc168955568)

[Desarrollo 6](#_Toc168955569)

[Síntesis 64](#_Toc168955570)

[Bibliografía 65](#_Toc168955571)

# Introducción

Android Studio es el entorno de desarrollo integrado (IDE) oficial de Google para desarrollar aplicaciones en Android. Proporciona herramientas y recursos para crear, probar y depurar aplicaciones Android, incluyendo un editor de código avanzado, un emulador de Android, herramientas de depuración, y una integración completa con el sistema de compilación Gradle. Android Studio está basado en el IDE IntelliJ IDEA y está disponible para Windows, macOS y Linux. Este trabajo se centra en el desarrollo de una aplicación para Android diseñada para facilitar la gestión de la información financiera del usuario, utilizando Android Studio como entorno de desarrollo y la integración de REST API para la comunicación con servicios web. A través de REST API, la aplicación puede interactuar con servidores remotos para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los datos financieros del usuario. Esta arquitectura permite un almacenamiento y manejo de datos más flexible y escalable en comparación con las soluciones locales, además de facilitar la integración con otros servicios y plataformas.

# Objetivo general

Desarrollar una aplicación móvil para Android utilizando el entorno de desarrollo Android Studio que permita al usuario gestionar su información financiera y almacenar los datos mediante el uso de REST API para interactuar con una base de datos remota.

# Objetivos específicos

* Implementar funcionalidades como registro, inicio de sesión, información de la aplicación, y recuperación de contraseña para el acceso y recuperación de datos de los usuarios mediante llamadas a REST API.
* Diseñar un menú principal con diferentes opciones para gestionar información que permita al usuario guardar, buscar, editar y eliminar datos específicos a través de la comunicación con una REST API, y un botón para cerrar la sesión del usuario.
* Utilizar REST API para almacenar y gestionar los datos financieros del usuario en un servidor remoto.
* Diseñar la interfaz gráfica para el usuario final de cada una de las actividades, asegurando una interacción intuitiva con las funcionalidades provistas por las API.

# Justificación

Esta actividad permite adquirir habilidades prácticas en el desarrollo de aplicaciones móviles y en la implementación de servicios web mediante REST API. Facilita la comprensión de la arquitectura de aplicaciones full-stack, integrando el frontend de Android con servicios backend, y la aplicación de conceptos teóricos en un proyecto real. Además, proporciona una experiencia valiosa para el portafolio profesional, mejora la competitividad en el mercado laboral y ofrece la oportunidad de resolver problemas del mundo real utilizando técnicas modernas de desarrollo y comunicación entre aplicaciones y servidores.

# Desarrollo

**Tecnologias utilizadas**

**IDE:**

**Android Studio:** es el entorno de desarrollo integrado (IDE) oficial de Google para desarrollar aplicaciones en Android. Posee herramientas y recursos para crear, probar y depurar aplicaciones Android, incluyendo un editor de código avanzado, un emulador de Android, herramientas de depuración, y una integración completa con el sistema de compilación Gradle.

**Base de datos:**

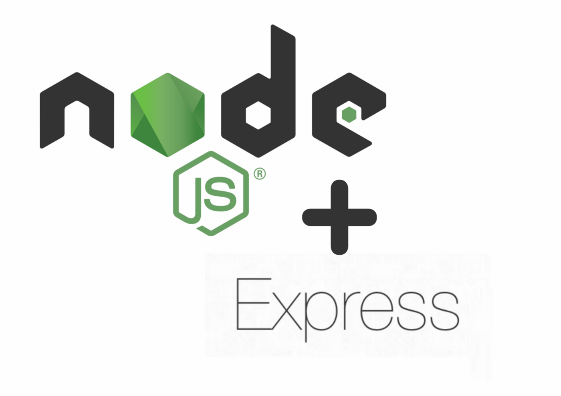
MySQL: es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS, por sus siglas en inglés) con un modelo cliente-servidor.

**Entorno servidor(Api)**



**Node.js:** Node.js es un entorno en tiempo de ejecución multiplataforma para la capa del servidor basado en JavaScript. Se usa para crear aplicaciones escalables, sin subprocesos, con bucle de eventos y velocidad.

**Framework:**



**Express:** es un framework de backend Node.js minimalista, rápido y similar a Sinatra, que proporciona características y herramientas robustas para desarrollar aplicaciones de backend escalables.

**Lenguaje programacion:**



**Java:** es el lenguaje principal utilizado en el desarrollo de aplicaciones Android. Es ampliamente adoptado, fácil de aprender, compatible con la plataforma Android, y cuenta con un sólido soporte de la comunidad. Ofrece facilidad de desarrollo, rapidez y acceso a una amplia gama de características de Android.

**Librerias:**

**Appcompat**

Proporciona soporte para características de UI en versiones antiguas de Android, garantizando compatibilidad con versiones anteriores.

**Material**

Incluye componentes de Material Design para mejorar la estética y usabilidad de la aplicación con botones, tarjetas y otros elementos visuales.

**Activity**

Facilita la gestión del ciclo de vida de las actividades, asegurando transiciones suaves y manejo adecuado de cambios de configuración.

**Constraintlayout**

Permite crear interfaces de usuario complejas y receptivas mediante restricciones, mejorando el diseño flexible y eficiente de la UI.

**Junit**

Facilita la escritura y ejecución de pruebas unitarias para verificar el correcto funcionamiento del código de manera aislada.

**Ext junit**

Extiende JUnit para pruebas instrumentadas en Android, asegurando el correcto funcionamiento de la aplicación en dispositivos reales o emuladores.

**Espresso core**

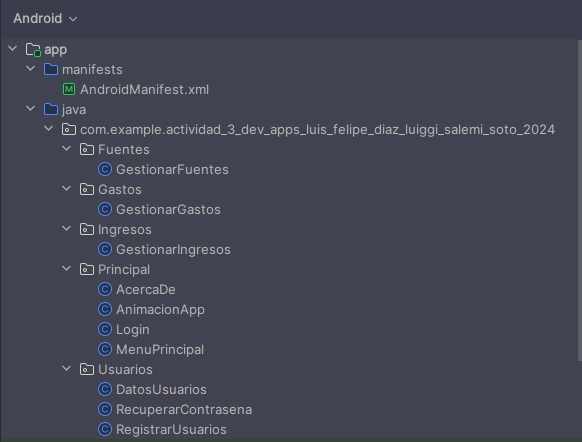
Permite escribir pruebas automatizadas de la UI para verificar la interacción del usuario con la aplicación y asegurar el comportamiento esperado de la interfaz.

**OkHttp**

Es una librería cliente HTTP para Android y Java que facilita la realización de solicitudes HTTP.

**Vista de la estructura del proyecto desde Android Studio**

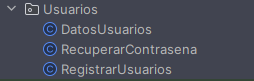
**Paquetes Java**



Se puede observar el proyecto esta modulado, de manera que cada paquete Android contiene una o varias clases Java que poseen la logica de las activitys.

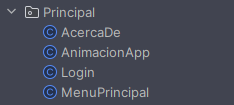
**Contenido de los paquetes:**

Paquete Usuarios



Contiene la logica para registrar los usuarios, recuperar la contraseña y consultar los datos de los usuarios.

Paquete Principal



Contiene la logica para loguearse, menu principal, animacion al abrir y la informacion de la app.

Paquete Fuentes



Contiene la logica para gestionar la fuente de ingresos de los usuarios ofreciendo las funcinalidades de buscar, editar, eliminar y guardar fuentes de ingreso.

Paquete Ingresos



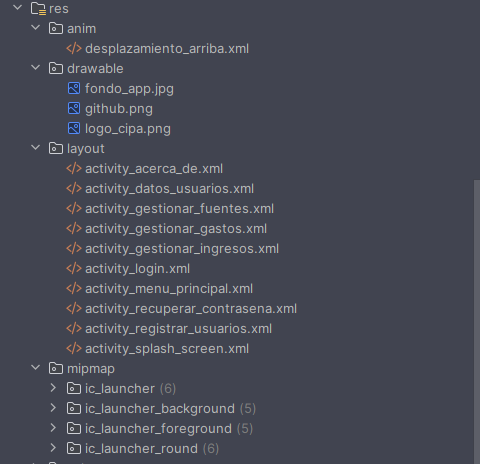
Contiene la logica para gestionar los ingresos de los usuarios ofreciendo las funcinalidades de buscar, editar, eliminar y guardar ingresos.

Paquete Gastos



Contiene la logica para gestionar los gastos de los usuarios ofreciendo las funcinalidades de buscar, editar, eliminar y guardar los gastos.

**Carpeta res**



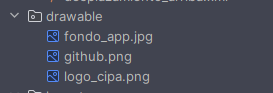
Esta es la carpeta principal de recursos y contiene todos los subdirectorios de recursos de la aplicación, como drawable, layout, values, etc.

Carpeta anim



Contiene el archivo XML de la animacion usada para cuando se ejecute la aplicación.

Drawable



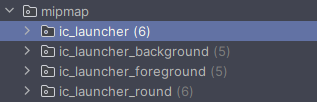
Contiene los recursos gráficos utilizados para la aplicación, En este caso contiene imágenes.

Layout



Contiene archivos XML que definen la disposición (layout) de la interfaz de usuario de la aplicación, en este caso de la diferentes activitys de los paquetes Java que contienen la respectiva logica.

Mimap



Contiene los recursos de imágenes optimizados para diferentes densidades de pantalla. En este caso contiene los íconos de la aplicación en varias resoluciones.

**Codigos relevantes del proyecto**

**Login**

public class Login extends AppCompatActivity {

TextInputEditText campoCedula, campoContraseña;

TextView recuperarContraseña, acercaDe;

private ProgressDialog progressDialog;

private OkHttpClient client;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_login);

campoCedula = findViewById(R.id.campoCedulaLogin);

campoContraseña = findViewById(R.id.campoContraseñaLogin);

recuperarContraseña = findViewById(R.id.recuperarContraseña);

Button registrarse = findViewById(R.id.BotonRegistrarse);

Button BotonLogin = findViewById(R.id.BotonLogin);

acercaDe = findViewById(R.id.acercaApp);

progressDialog = new ProgressDialog(this);

progressDialog.setMessage("Iniciando sesión...");

progressDialog.setCancelable(false);

client = new OkHttpClient();

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

registrarse.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(Login.this, RegistrarUsuarios.class);

startActivity(intent);

}

});

acercaDe.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(Login.this, AcercaDe.class);

startActivity(intent);

}

});

BotonLogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

validarCamposLogin();

}

});

recuperarContraseña.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent(Login.this, RecuperarContrasena.class);

startActivity(intent);

finish();

}

});

}

public void validarCamposLogin() {

String cedula = campoCedula.getText().toString().trim();

String password = campoContraseña.getText().toString().trim();

if (cedula.isEmpty()) {

campoCedula.setError("Ingrese una cédula válida");

return;

} else {

campoCedula.setError(null);

}

if (password.isEmpty() || password.length() < 9) {

campoContraseña.setError("La contraseña debe contener al menos 12 caracteres");

return;

} else if (!Pattern.compile("[0-9]").matcher(password).find()) {

campoContraseña.setError("La contraseña debe contener al menos un número");

return;

} else {

campoContraseña.setError(null);

}

LogicaInciarSesion(cedula, password);

}

public void LogicaInciarSesion(String cedula, String contraseña) {

progressDialog.show();

String url = "http://192.168.1.4:5000/api/usuarios/login";

MediaType JSON = MediaType.get("application/json; charset=utf-8");

String json = "{\"cedula\":\"" + cedula + "\", \"contrasena\":\"" + contraseña + "\"}";

RequestBody body = RequestBody.create(json, JSON);

Request request = new Request.Builder()

.url(url)

.post(body)

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(@NonNull Call call, @NonNull IOException e) {

e.printStackTrace();

runOnUiThread(() -> {

progressDialog.dismiss();

Toast.makeText(Login.this, "Error en la conexión", Toast.LENGTH\_SHORT).show();

});

}

@Override

public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {

runOnUiThread(() -> {

progressDialog.dismiss();

});

if (response.isSuccessful()) {

try {

String responseData = response.body().string();

JSONObject jsonResponse = new JSONObject(responseData);

String message = jsonResponse.getString("message");

String primerNombre = jsonResponse.getString("primer\_nombre");

SharedPreferences sharedPreferences = PreferenceManager.getDefaultSharedPreferences(Login.this);

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.putString("CEDULA", cedula);

editor.apply();

runOnUiThread(() -> {

Toast.makeText(Login.this, message, Toast.LENGTH\_SHORT).show();

Intent intent = new Intent(Login.this, MenuPrincipal.class);

intent.putExtra("PRIMER\_NOMBRE", primerNombre);

startActivity(intent);

finish();

});

} catch (Exception e) {

e.printStackTrace();

runOnUiThread(() -> {

Toast.makeText(Login.this, "Error al procesar la respuesta", Toast.LENGTH\_SHORT).show();

});

}

} else {

try {

String responseData = response.body().string();

JSONObject jsonResponse = new JSONObject(responseData);

String message = jsonResponse.getString("message");

runOnUiThread(() -> {

Toast.makeText(Login.this, message, Toast.LENGTH\_SHORT).show();

});

} catch (Exception e) {

e.printStackTrace();

runOnUiThread(() -> {

Toast.makeText(Login.this, "Error en el servidor", Toast.LENGTH\_SHORT).show();

});

}

}

}

});

}

}

**EXPLICACION:**

**Declaración de variables y objetos:** Se definen las variables para los elementos de interfaz de usuario como el campo de cédula, campo de contraseña, botones y texto de recuperación de contraseña. También se inicializan un objeto ProgressDialog y un cliente OkHttpClient para manejar las solicitudes HTTP.

**Método onCreate():** Este método es llamado cuando la actividad se crea por primera vez. Aquí se establece el diseño de la actividad y se asignan los elementos de interfaz de usuario a las variables correspondientes. También se configuran los listeners para los botones y el texto de recuperación de contraseña.

**Listeners para eventos de clic:** Se configuran los listeners para los botones de registro, inicio de sesión y recuperación de contraseña, así como para el texto de información acerca de la aplicación. Cada listener define la acción a realizar cuando el usuario hace clic en el elemento correspondiente.

**Método validarCamposLogin():** Este método se encarga de validar los campos de entrada del usuario, como la cédula y la contraseña. Se verifica si los campos están vacíos o si la contraseña no cumple con ciertos criterios, como una longitud mínima y la inclusión de al menos un número.

**Método LogicaInciarSesion():** Este método implementa la lógica para iniciar sesión en la aplicación. Se muestra un ProgressDialog mientras se realiza la solicitud HTTP al servidor. Se construye la solicitud HTTP POST con los datos de inicio de sesión proporcionados por el usuario (cédula y contraseña). Se manejan las respuestas del servidor, mostrando mensajes de error o redireccionando al usuario a la actividad principal si el inicio de sesión es exitoso.

**Registro de usuarios**

ublic class RegistrarUsuarios extends AppCompatActivity {

TextInputEditText contraseñaRegistro, correoElectronico, primerNombre, segundoNombre, primerApellido, segundoApellido, preguntaContraseña, respuestaContraseña, cedula, telefono, pais;

Button registrarse;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_registrar\_usuarios);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

// Obtener los datos ingresados en los text input edit text

// y demas elementos del formulario

correoElectronico = findViewById(R.id.correoElectronico);

contraseñaRegistro = findViewById(R.id.contraseñaRegistro);

primerNombre = findViewById(R.id.primerNombre);

segundoNombre = findViewById(R.id.segundoNombre);

primerApellido = findViewById(R.id.primerApellido);

segundoApellido = findViewById(R.id.segundoApellido);

cedula = findViewById(R.id.registroCedula);

preguntaContraseña = findViewById(R.id.preguntaContra);

respuestaContraseña = findViewById(R.id.respuestaContra);

registrarse = findViewById(R.id.registrarse);

telefono = findViewById(R.id.telefonoUsuario);

pais = findViewById(R.id.paisUsuario);

registrarse.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

validarCampos();

}

});

}

//Este codigo es para validar los datos ingresados en el formulario

public void validarCampos() {

String email = correoElectronico.getText().toString().trim();

String password = contraseñaRegistro.getText().toString().trim();

String primerNombreText = primerNombre.getText().toString().trim();

String segundoNombreText = segundoNombre.getText().toString().trim();

String primerApellidoText = primerApellido.getText().toString().trim();

String segundoApellidoText = segundoApellido.getText().toString().trim();

String preguntaContraseñaText = preguntaContraseña.getText().toString().trim();

String respuestaContraseñaText = respuestaContraseña.getText().toString().trim();

String cedulaText = cedula.getText().toString().trim();

String telefonoText = telefono.getText().toString().trim();

String paisText = pais.getText().toString().trim();

if (primerNombreText.isEmpty()) {

primerNombre.setError("Ingrese su primer nombre");

return;

} else {

primerNombre.setError(null);

}

if (segundoNombreText.isEmpty()) {

segundoNombre.setError("Ingrese su segundo nombre");

return;

} else {

segundoNombre.setError(null);

}

if (primerApellidoText.isEmpty()) {

primerApellido.setError("Ingrese su primer apellido");

return;

} else {

primerApellido.setError(null);

}

if (segundoApellidoText.isEmpty()) {

segundoApellido.setError("Ingrese su segundo apellido");

return;

} else {

segundoApellido.setError(null);

}

if (cedulaText.isEmpty()) {

cedula.setError("Ingrese su número de cédula");

return;

} else {

cedula.setError(null);

}

if (preguntaContraseñaText.isEmpty()) {

preguntaContraseña.setError("Ingrese su pregunta de seguridad");

return;

} else {

preguntaContraseña.setError(null);

}

if (respuestaContraseñaText.isEmpty()) {

respuestaContraseña.setError("Ingrese su respuesta de seguridad");

return;

} else {

respuestaContraseña.setError(null);

}

if (telefonoText.isEmpty()) {

telefono.setError("Ingrese su número de teléfono");

return;

} else {

telefono.setError(null);

}

if (paisText.isEmpty()) {

pais.setError("Ingrese su país");

return;

} else {

pais.setError(null);

}

if (email.isEmpty() || !Patterns.EMAIL\_ADDRESS.matcher(email).matches()) {

correoElectronico.setError("Ingrese un correo electrónico válido");

return;

} else {

correoElectronico.setError(null);

}

if (password.isEmpty() || password.length() < 9) {

contraseñaRegistro.setError("La contraseña debe contener al menos 12 caracteres");

return;

} else if (!Pattern.compile("[0-9]").matcher(password).find()) {

contraseñaRegistro.setError("La contraseña debe contener al menos un número");

return;

} else {

contraseñaRegistro.setError(null);

}

registrarUsuario();

}

private final OkHttpClient client = new OkHttpClient();

public void registrarUsuario() {

String email = correoElectronico.getText().toString().trim();

String password = contraseñaRegistro.getText().toString().trim();

JSONObject userJson = new JSONObject();

try {

userJson.put("cedula", cedula.getText().toString().trim());

userJson.put("primer\_nombre", primerNombre.getText().toString().trim());

userJson.put("segundo\_nombre", segundoNombre.getText().toString().trim());

userJson.put("primer\_apellido", primerApellido.getText().toString().trim());

userJson.put("segundo\_apellido", segundoApellido.getText().toString().trim());

userJson.put("correo", email);

userJson.put("contrasena", password);

userJson.put("pregunta\_contra", preguntaContraseña.getText().toString().trim());

userJson.put("respuesta\_contra", respuestaContraseña.getText().toString().trim());

userJson.put("numero\_tel", telefono.getText().toString().trim());

userJson.put("pais", pais.getText().toString().trim());

} catch (Exception e) {

e.printStackTrace();

}

String url = "http://192.168.1.4:5000/api/usuarios/registro";

MediaType JSON = MediaType.get("application/json; charset=utf-8");

RequestBody body = RequestBody.create(userJson.toString(), JSON);

Request request = new Request.Builder()

.url(url)

.post(body)

.build();

OkHttpClient client = new OkHttpClient();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(@NonNull Call call, @NonNull IOException e) {

runOnUiThread(() -> {

Toast.makeText(RegistrarUsuarios.this, "Error en la conexión", Toast.LENGTH\_SHORT).show();

});

}

@Override

public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {

runOnUiThread(() -> {

if (response.isSuccessful()) {

Toast.makeText(RegistrarUsuarios.this, "Registro exitoso", Toast.LENGTH\_SHORT).show();

Intent intent = new Intent(RegistrarUsuarios.this, Login.class);

startActivity(intent);

finish();

} else {

Toast.makeText(RegistrarUsuarios.this, "Error al registrar el usuario.", Toast.LENGTH\_SHORT).show();

}

});

}

});

}

}

**EXPLICACION:**

**Declaración de variables y objetos:** Se declaran variables para los elementos de la interfaz de usuario, como TextInputEditText para campos como correo electrónico, contraseña, nombres, apellidos, etc. También se declara un botón para el proceso de registro.

**Método onCreate():** En este método se configura la vista de la actividad y se asignan los elementos de la interfaz de usuario a las variables correspondientes. También se establece un listener para el botón de registro.

**Listener para evento de clic:** Se establece un listener para el botón de registro. Cuando se hace clic en este botón, se llama al método validarCampos().

**Método validarCampos():** Este método se encarga de validar los datos ingresados por el usuario en el formulario de registro. Se verifica que todos los campos obligatorios estén llenos y que el formato de correo electrónico sea válido. Además, se verifica que la contraseña cumpla con ciertos criterios de seguridad, como una longitud mínima y la inclusión de al menos un número.

**Método registrarUsuario():** Este método realiza la solicitud HTTP al servidor para registrar al usuario. Se construye un objeto JSON con los datos del usuario proporcionados por el formulario. Luego, se realiza una solicitud HTTP POST al servidor con estos datos. Dependiendo de la respuesta del servidor, se muestra un mensaje de éxito o error al usuario.

**Recuperar contraseña**

public class RecuperarContrasena extends AppCompatActivity {

TextInputEditText campoCedula, campoRespuesta;

Button botonBuscarPregunta, botonValidarRespuesta;

TextView preguntaRecuperacion;

OkHttpClient client = new OkHttpClient();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_recuperar\_contrasena);

campoCedula = findViewById(R.id.campoCedula);

campoRespuesta = findViewById(R.id.respuestaRecuperacion);

botonBuscarPregunta = findViewById(R.id.BotonBuscarPregunta);

botonValidarRespuesta = findViewById(R.id.botonValidarRespuesta);

preguntaRecuperacion = findViewById(R.id.preguntaRecuperacion);

botonBuscarPregunta.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

buscarPreguntaRecuperacion();

}

});

botonValidarRespuesta.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

validarRespuesta();

}

});

}

private void buscarPreguntaRecuperacion() {

String cedula = campoCedula.getText().toString().trim();

if (cedula.isEmpty()) {

campoCedula.setError("Ingrese su cédula");

return;

}

Request request = new Request.Builder()

.url("http://192.168.1.4:5000/api/usuarios/recuperar/" + cedula)

.get()

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(@NonNull Call call, @NonNull IOException e) {

runOnUiThread(() -> Toast.makeText(RecuperarContrasena.this, "Error en la solicitud", Toast.LENGTH\_SHORT).show());

}

@Override

public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {

if (response.isSuccessful()) {

try {

JSONObject jsonResponse = new JSONObject(response.body().string());

String pregunta = jsonResponse.getString("pregunta");

runOnUiThread(() -> {

preguntaRecuperacion.setText(pregunta);

preguntaRecuperacion.setVisibility(View.VISIBLE);

campoRespuesta.setVisibility(View.VISIBLE);

botonValidarRespuesta.setVisibility(View.VISIBLE);

});

} catch (JSONException e) {

e.printStackTrace();

}

} else {

runOnUiThread(() -> Toast.makeText(RecuperarContrasena.this, "Usuario no encontrado", Toast.LENGTH\_SHORT).show());

}

}

});

}

private void validarRespuesta() {

String cedula = campoCedula.getText().toString().trim();

String respuesta = campoRespuesta.getText().toString().trim();

if (respuesta.isEmpty()) {

campoRespuesta.setError("Ingrese su respuesta");

return;

}

JSONObject json = new JSONObject();

try {

json.put("respuesta", respuesta);

} catch (JSONException e) {

e.printStackTrace();

}

RequestBody body = RequestBody.create(json.toString(), MediaType.get("application/json; charset=utf-8"));

Request request = new Request.Builder()

.url("http://192.168.1.4:5000/api/usuarios/recuperar/validar/" + cedula)

.post(body)

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(@NonNull Call call, @NonNull IOException e) {

runOnUiThread(() -> Toast.makeText(RecuperarContrasena.this, "Error en la solicitud", Toast.LENGTH\_SHORT).show());

}

@Override

public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {

if (response.isSuccessful()) {

try {

JSONObject jsonResponse = new JSONObject(response.body().string());

String mensaje = jsonResponse.getString("message");

runOnUiThread(() -> mostrarContrasena(mensaje));

} catch (JSONException e) {

e.printStackTrace();

}

} else {

runOnUiThread(() -> Toast.makeText(RecuperarContrasena.this, "Respuesta incorrecta", Toast.LENGTH\_SHORT).show());

}

}

});

}

private void mostrarContrasena(String contrasena) {

// Mostrar un diálogo con la contraseña

AlertDialog.Builder builder = new AlertDialog.Builder(RecuperarContrasena.this);

builder.setTitle("Contraseña recuperada");

builder.setMessage(contrasena);

builder.setPositiveButton("OK", (dialog, which) -> {

Intent intent = new Intent(RecuperarContrasena.this, Login.class);

startActivity(intent);

finish();

});

builder.show();

}

//codigo para regresar al login con el boton de atras del movil

@Override

public void onBackPressed() {

super.onBackPressed();

Intent intent = new Intent(this, Login.class);

startActivity(intent);

finish();

}

}

**EXPLICACION:**

**Método onCreate():** En este método se configura la vista de la actividad y se asignan los elementos de la interfaz de usuario a las variables correspondientes. También se establecen listeners para los botones de buscar pregunta y validar respuesta.

**Listeners para eventos de clic:** Se establecen listeners para los botones de buscar pregunta y validar respuesta. Cuando se hace clic en estos botones, se llaman a los métodos correspondientes para buscar la pregunta de seguridad y validar la respuesta del usuario.

**Método buscarPreguntaRecuperacion():** Este método realiza una solicitud HTTP GET al servidor para obtener la pregunta de seguridad asociada a la cédula proporcionada por el usuario. Si la solicitud es exitosa, la pregunta se muestra en la interfaz de usuario.

**Método validarRespuesta():** Este método valida la respuesta proporcionada por el usuario comparándola con la respuesta almacenada en el servidor. Se realiza una solicitud HTTP POST con la respuesta proporcionada por el usuario y se compara con la respuesta almacenada en el servidor. Dependiendo de la respuesta del servidor, se muestra un mensaje de éxito o error al usuario.

**Datos del usuario**

public class DatosUsuarios extends AppCompatActivity {

TextInputEditText password, email, pregunta, respuesta, telefono, pais, currentPasswordInput;

Button guardar, eliminarUser;

String cedulaUsuario;

private OkHttpClient client;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_datos\_usuarios);

SharedPreferences sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);

cedulaUsuario = sharedPreferences.getString("CEDULA", "");

// Set up the system bars

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {

Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);

return insets;

});

client = new OkHttpClient();

password = findViewById(R.id.updateContraseña);

email = findViewById(R.id.updateCorreo);

pregunta = findViewById(R.id.updatePregunta);

respuesta = findViewById(R.id.updateRespuesta);

telefono = findViewById(R.id.updateTelefono);

pais = findViewById(R.id.updatePais);

guardar = findViewById(R.id.actualizarDatos);

eliminarUser = findViewById(R.id.eliminarUsuario);

cargarDatosUsuario();

guardar.setOnClickListener(v -> {

validarCampos();

});

eliminarUser.setOnClickListener(v -> {

eliminarCuentaUsuario();

});

}

private void cargarDatosUsuario() {

String url = "http://192.168.1.4:5000/api/usuarios/" + cedulaUsuario;

Request request = new Request.Builder()

.url(url)

.get()

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

e.printStackTrace();

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Error al cargar los datos del usuario", Toast.LENGTH\_SHORT).show());

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

String responseData = response.body().string();

try {

JSONObject jsonResponse = new JSONObject(responseData);

if (jsonResponse.has("usuario")) {

JSONObject usuarioData = jsonResponse.getJSONObject("usuario");

runOnUiThread(() -> {

try {

telefono.setText(usuarioData.getString("numero\_tel"));

email.setText(usuarioData.getString("correo"));

pregunta.setText(usuarioData.getString("pregunta\_contra"));

respuesta.setText(usuarioData.getString("respuesta\_contra"));

pais.setText(usuarioData.getString("pais"));

} catch (JSONException e) {

e.printStackTrace();

Toast.makeText(DatosUsuarios.this, "Error al obtener datos del usuario", Toast.LENGTH\_SHORT).show();

}

});

} else {

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Usuario no encontrado", Toast.LENGTH\_SHORT).show());

}

} catch (JSONException e) {

e.printStackTrace();

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Error al procesar la respuesta del servidor", Toast.LENGTH\_SHORT).show());

}

} else {

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Error al cargar los datos del usuario", Toast.LENGTH\_SHORT).show());

}

}

});

}

public void validarCampos() {

String contraseña = password.getText().toString().trim();

String correo = email.getText().toString().trim();

String preguntaContraseña = pregunta.getText().toString().trim();

String respuestaContraseña = respuesta.getText().toString().trim();

String telefonoUsuario = telefono.getText().toString().trim();

String paisUsuario = pais.getText().toString().trim();

if (correo.isEmpty() || !Patterns.EMAIL\_ADDRESS.matcher(correo).matches()) {

email.setError("Ingrese un correo electrónico válido");

return;

} else {

email.setError(null);

}

if (preguntaContraseña.isEmpty()) {

pregunta.setError("Ingrese su pregunta de seguridad");

return;

} else {

pregunta.setError(null);

}

if (respuestaContraseña.isEmpty()) {

respuesta.setError("Ingrese su respuesta de seguridad");

return;

} else {

respuesta.setError(null);

}

if (telefonoUsuario.isEmpty()) {

telefono.setError("Ingrese su teléfono");

return;

} else {

telefono.setError(null);

}

if (paisUsuario.isEmpty()) {

pais.setError("Ingrese su país");

return;

} else {

pais.setError(null);

}

actualizarDatosUsuario(telefonoUsuario, correo, preguntaContraseña, respuestaContraseña, paisUsuario, contraseña);

}

private void actualizarDatosUsuario(String telefono, String correo, String pregunta, String respuesta, String pais, String contraseña) {

// Construir el cuerpo de la solicitud JSON

JSONObject jsonBody = new JSONObject();

try {

jsonBody.put("telefono", telefono);

jsonBody.put("correo", correo);

jsonBody.put("pregunta\_contra", pregunta);

jsonBody.put("respuesta\_contra", respuesta);

jsonBody.put("pais", pais);

jsonBody.put("contraseña", contraseña);

} catch (JSONException e) {

e.printStackTrace();

Toast.makeText(this, "Error al construir la solicitud JSON", Toast.LENGTH\_SHORT).show();

return;

}

// Crear una solicitud HTTP utilizando OkHttp

RequestBody requestBody = RequestBody.create(MediaType.parse("application/json"), jsonBody.toString());

Request request = new Request.Builder()

.url("http://192.168.1.4:5000/api/usuarios/actualizar/" + cedulaUsuario)

.put(requestBody)

.build();

// Realizar la solicitud HTTP de forma asíncrona

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

e.printStackTrace();

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Error al actualizar los datos del usuario", Toast.LENGTH\_SHORT).show());

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

// Manejar la respuesta exitosa de la API

String responseData = response.body().string();

try {

JSONObject jsonResponse = new JSONObject(responseData);

String message = jsonResponse.getString("message");

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, message, Toast.LENGTH\_SHORT).show());

} catch (JSONException e) {

e.printStackTrace();

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Error al procesar la respuesta del servidor", Toast.LENGTH\_SHORT).show());

}

} else {

// Manejar errores de la solicitud HTTP

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Error al actualizar los datos del usuario", Toast.LENGTH\_SHORT).show());

}

}

});

}

private void eliminarCuentaUsuario() {

// Construir la URL para la solicitud HTTP DELETE

String url = "http://192.168.1.4:5000/api/usuarios/" + cedulaUsuario;

// Crear una solicitud HTTP utilizando OkHttp

Request request = new Request.Builder()

.url(url)

.delete()

.build();

// Realizar la solicitud HTTP de forma asíncrona

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

e.printStackTrace();

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Error al eliminar la cuenta del usuario", Toast.LENGTH\_SHORT).show());

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

// Manejar la respuesta exitosa de la API

String responseData = response.body().string();

try {

JSONObject jsonResponse = new JSONObject(responseData);

String message = jsonResponse.getString("message");

runOnUiThread(() -> {

Toast.makeText(DatosUsuarios.this, message, Toast.LENGTH\_SHORT).show();

// Redirigir a la pantalla de inicio de sesión

startActivity(new Intent(DatosUsuarios.this, Login.class));

finish();

});

} catch (JSONException e) {

e.printStackTrace();

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Error al procesar la respuesta del servidor", Toast.LENGTH\_SHORT).show());

}

} else {

// Manejar errores de la solicitud HTTP

runOnUiThread(() -> Toast.makeText(DatosUsuarios.this, "Error al eliminar la cuenta del usuario", Toast.LENGTH\_SHORT).show());

}

}

});

}

public void onBackPressed() {

super.onBackPressed();

finish();

}

}

**EXPLICACION:**

**Declaración de variables y objetos:** Se declaran variables para los elementos de la interfaz de usuario, como TextInputEditText para los campos de contraseña, correo electrónico, pregunta y respuesta de seguridad, teléfono y país, así como botones para guardar y eliminar datos de usuario. También se declara una variable para almacenar la cédula del usuario y un objeto OkHttpClient para realizar solicitudes HTTP.

**Método onCreate():** En este método se configura la vista de la actividad, se obtienen los datos del usuario desde SharedPreferences, y se asignan los elementos de la interfaz de usuario a las variables correspondientes. También se establecen listeners para los botones de guardar y eliminar, que llaman a los métodos correspondientes para validar campos y eliminar la cuenta del usuario.

**Método cargarDatosUsuario():** Este método realiza una solicitud HTTP GET al servidor para obtener los datos del usuario asociados a la cédula proporcionada. Si la solicitud es exitosa, los datos se muestran en la interfaz de usuario.

**Método validarCampos():** Este método valida los datos proporcionados por el usuario, asegurándose de que no estén vacíos y cumplan con ciertos criterios (como formato de correo electrónico válido). Si los datos son válidos, se llama al método actualizarDatosUsuario().

**Método actualizarDatosUsuario():** Este método construye un objeto JSON con los datos del usuario y realiza una solicitud HTTP PUT para actualizar estos datos en el servidor. Si la solicitud es exitosa, se muestra un mensaje de éxito al usuario.

**Método eliminarCuentaUsuario():** Este método construye una URL para la solicitud HTTP DELETE y realiza esta solicitud para eliminar la cuenta del usuario. Si la solicitud es exitosa, se muestra un mensaje de éxito y se redirige al usuario a la pantalla de inicio de sesión.

**Método onBackPressed():** Este método permite que el usuario regrese a la pantalla anterior al presionar el botón de retroceso en el dispositivo móvil.

**Menu principal de la aplicacion**

public class MenuPrincipal extends AppCompatActivity {

TextView vistaNombreUsuario;

Button datosUsuarios, gestionarIngresos, gestionarFuentes, gestionarGastos, Logout;

ProgressDialog progressDialog;

OkHttpClient client = new OkHttpClient();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_menu\_principal);

vistaNombreUsuario = findViewById(R.id.nombreUsuario);

Logout = findViewById(R.id.botonCerrarSesion);

progressDialog = new ProgressDialog(this);

progressDialog.setMessage("Cerrando sesión...");

progressDialog.setCancelable(false);

datosUsuarios = findViewById(R.id.datosPersonales);

gestionarIngresos = findViewById(R.id.gestionarIngreso);

gestionarFuentes = findViewById(R.id.GestionarFuentes);

gestionarGastos = findViewById(R.id.GestionarGastos);

SharedPreferences sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);

String cedula = sharedPreferences.getString("CEDULA", "");

if (cedula != null) {

obtenerDatosUsuario(cedula);

}

Logout.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

cerrarSesion();

}

});

datosUsuarios.setOnClickListener(v -> {

Intent intent = new Intent(MenuPrincipal.this, DatosUsuarios.class);

startActivity(intent);

});

gestionarIngresos.setOnClickListener(v -> {

Intent intent = new Intent(MenuPrincipal.this, GestionarIngresos.class);

startActivity(intent);

});

gestionarFuentes.setOnClickListener(v -> {

Intent intent = new Intent(MenuPrincipal.this, GestionarFuentes.class);

startActivity(intent);

});

gestionarGastos.setOnClickListener(v -> {

Intent intent = new Intent(MenuPrincipal.this, GestionarGastos.class);

startActivity(intent);

});

}

private void obtenerDatosUsuario(String cedula) {

String url = "http://192.168.1.4:5000/api/usuarios/" + cedula;

Request request = new Request.Builder()

.url(url)

.get()

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(@NonNull Call call, @NonNull IOException e) {

runOnUiThread(() -> Toast.makeText(MenuPrincipal.this, "Error en la solicitud", Toast.LENGTH\_SHORT).show());

}

@Override

public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {

if (response.isSuccessful()) {

try {

JSONObject jsonResponse = new JSONObject(response.body().string());

JSONObject usuario = jsonResponse.getJSONObject("usuario");

String primerNombre = usuario.getString("primer\_nombre");

String primerApellido = usuario.getString("primer\_apellido");

runOnUiThread(() -> vistaNombreUsuario.setText(primerNombre + " " + primerApellido));

} catch (JSONException e) {

e.printStackTrace();

}

} else {

runOnUiThread(() -> Toast.makeText(MenuPrincipal.this, "Error en la respuesta del servidor", Toast.LENGTH\_SHORT).show());

}

}

});

}

private void cerrarSesion() {

progressDialog.show();

SharedPreferences sharedPreferences = PreferenceManager.getDefaultSharedPreferences(MenuPrincipal.this);

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.clear();

editor.apply();

// Retrasar el inicio de la siguiente actividad

new Handler().postDelayed(new Runnable() {

@Override

public void run() {

progressDialog.dismiss();

Toast.makeText(MenuPrincipal.this, "Sesión cerrada exitosamente", Toast.LENGTH\_SHORT).show();

Intent intent = new Intent(MenuPrincipal.this, Login.class);

startActivity(intent);

finish();

}

}, 2000); // 2 segundos de retraso

}

}

**EXPLICACION:**

**Declaración de variables y objetos**

Se declaran variables para los elementos de la interfaz de usuario, como TextView para mostrar el nombre del usuario y Button para las diferentes opciones del menú y cerrar sesión. También se declara un ProgressDialog para mostrar un mensaje mientras se cierra la sesión y un objeto OkHttpClient para realizar solicitudes HTTP.

**Método onCreate()**

En este método se configura la vista de la actividad, se obtienen los datos del usuario desde SharedPreferences, y se asignan los elementos de la interfaz de usuario a las variables correspondientes. También se establecen listeners para los botones del menú y el botón de cerrar sesión, que llaman a los métodos correspondientes para realizar las acciones deseadas.

**Método obtenerDatosUsuario(String cedula)**

Este método realiza una solicitud HTTP GET al servidor para obtener los datos del usuario asociados a la cédula proporcionada. Si la solicitud es exitosa, los datos se extraen del JSON de respuesta y se muestran en la interfaz de usuario (vistaNombreUsuario).

**Método cerrarSesion()**

Este método muestra un ProgressDialog mientras se cierra la sesión, borra las preferencias compartidas y, después de un retraso de 2 segundos, cierra el diálogo de progreso, muestra un mensaje de éxito y redirige al usuario a la pantalla de inicio de sesión.

**Gestionar gastos del usuario**

public class GestionarGastos extends AppCompatActivity {

private EditText idGastos, fechaIngreso, nombreGasto, valorGasto, categoriaGasto, descripcionGasto;

private Button botonGuardar, botonBuscar, botonEditar, botonEliminar;

private OkHttpClient client;

private String cedulaUsuario;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_gestionar\_gastos);

client = new OkHttpClient();

// Obtener cédula del usuario desde SharedPreferences

cedulaUsuario = getSharedPreferences("MyPrefs", MODE\_PRIVATE).getString("cedula", null);

// Inicializar elementos de la interfaz

idGastos = findViewById(R.id.idGastos);

fechaIngreso = findViewById(R.id.fechaIngreso);

nombreGasto = findViewById(R.id.nombreGasto);

valorGasto = findViewById(R.id.valorGasto);

categoriaGasto = findViewById(R.id.categoriaGasto);

descripcionGasto = findViewById(R.id.descripcionGasto);

botonGuardar = findViewById(R.id.botonGuardar);

botonBuscar = findViewById(R.id.botonBuscar);

botonEditar = findViewById(R.id.botonEditar);

botonEliminar = findViewById(R.id.botonEliminar);

//eventos para los botones de la interfaz de usuario

botonBuscar.setOnClickListener(v -> buscarGasto());

botonEditar.setOnClickListener(v -> editarGasto());

botonEliminar.setOnClickListener(v -> eliminarGasto());

botonGuardar.setOnClickListener(v -> guardarGasto());

}

private void buscarGasto() {

String id = idGastos.getText().toString().trim();

if (id.isEmpty()) {

Toast.makeText(this, "Por favor ingrese el ID del gasto", Toast.LENGTH\_SHORT).show();

return;

}

Request request = new Request.Builder()

.url("http://192.168.1.4:5000/api/gastos/buscar/" + id)

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

runOnUiThread(() -> Toast.makeText(GestionarGastos.this, "Error al buscar el gasto", Toast.LENGTH\_SHORT).show());

e.printStackTrace();

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

String responseData = response.body().string();

runOnUiThread(() -> {

try {

JSONObject jsonResponse = new JSONObject(responseData);

fechaIngreso.setText(jsonResponse.getString("fecha\_gasto"));

nombreGasto.setText(jsonResponse.getString("nombre\_gasto"));

valorGasto.setText(jsonResponse.getString("valor\_gasto"));

categoriaGasto.setText(jsonResponse.getString("categoria\_gasto"));

descripcionGasto.setText(jsonResponse.getString("descripcion\_gasto"));

} catch (JSONException e) {

e.printStackTrace();

Toast.makeText(GestionarGastos.this, "Error al procesar los datos del gasto", Toast.LENGTH\_SHORT).show();

}

});

} else {

runOnUiThread(() -> Toast.makeText(GestionarGastos.this, "Gasto no encontrado", Toast.LENGTH\_SHORT).show());

}

}

});

}

private void editarGasto() {

String id = idGastos.getText().toString().trim();

String fecha = fechaIngreso.getText().toString().trim();

String nombre = nombreGasto.getText().toString().trim();

String valor = valorGasto.getText().toString().trim();

String categoria = categoriaGasto.getText().toString().trim();

String descripcion = descripcionGasto.getText().toString().trim();

if (id.isEmpty() || fecha.isEmpty() || nombre.isEmpty() || valor.isEmpty() || categoria.isEmpty() || descripcion.isEmpty()) {

Toast.makeText(this, "Por favor complete todos los campos", Toast.LENGTH\_SHORT).show();

return;

}

JSONObject jsonBody = new JSONObject();

try {

jsonBody.put("id\_usuario", cedulaUsuario);

jsonBody.put("fecha\_gasto", fecha);

jsonBody.put("nombre\_gasto", nombre);

jsonBody.put("valor\_gasto", valor);

jsonBody.put("categoria\_gasto", categoria);

jsonBody.put("descripcion\_gasto", descripcion);

} catch (JSONException e) {

e.printStackTrace();

Toast.makeText(this, "Error al construir la solicitud JSON", Toast.LENGTH\_SHORT).show();

return;

}

RequestBody requestBody = RequestBody.create(MediaType.parse("application/json"), jsonBody.toString());

Request request = new Request.Builder()

.url("http://192.168.1.4:5000/api/gastos/actualizar/" + id)

.put(requestBody)

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

runOnUiThread(() -> Toast.makeText(GestionarGastos.this, "Error al actualizar el gasto", Toast.LENGTH\_SHORT).show());

e.printStackTrace();

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

runOnUiThread(() -> Toast.makeText(GestionarGastos.this, "Gasto actualizado correctamente", Toast.LENGTH\_SHORT).show());

} else {

runOnUiThread(() -> Toast.makeText(GestionarGastos.this, "Error al actualizar el gasto", Toast.LENGTH\_SHORT).show());

}

}

});

}

private void eliminarGasto() {

String id = idGastos.getText().toString().trim();

if (id.isEmpty()) {

Toast.makeText(this, "Por favor ingrese el ID del gasto", Toast.LENGTH\_SHORT).show();

return;

}

Request request = new Request.Builder()

.url("http://192.168.1.4:5000/api/gastos/eliminar/" + id)

.delete()

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

runOnUiThread(() -> Toast.makeText(GestionarGastos.this, "Error al eliminar el gasto", Toast.LENGTH\_SHORT).show());

e.printStackTrace();

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

runOnUiThread(() -> {

Toast.makeText(GestionarGastos.this, "Gasto eliminado correctamente", Toast.LENGTH\_SHORT).show();

limpiarCampos();

});

} else {

runOnUiThread(() -> Toast.makeText(GestionarGastos.this, "Error al eliminar el gasto", Toast.LENGTH\_SHORT).show());

}

}

});

}

private void guardarGasto() {

String id = idGastos.getText().toString().trim();

String fecha = fechaIngreso.getText().toString().trim();

String nombre = nombreGasto.getText().toString().trim();

String valor = valorGasto.getText().toString().trim();

String categoria = categoriaGasto.getText().toString().trim();

String descripcion = descripcionGasto.getText().toString().trim();

if (id.isEmpty() || fecha.isEmpty() || nombre.isEmpty() || valor.isEmpty() || categoria.isEmpty() || descripcion.isEmpty()) {

Toast.makeText(this, "Por favor complete todos los campos", Toast.LENGTH\_SHORT).show();

return;

}

// Eliminar separadores de miles

valor = valor.replace(".", "").replace(",", ".");

double cantidad;

try {

cantidad = Double.parseDouble(valor);

} catch (NumberFormatException e) {

Toast.makeText(this, "Valor de gasto inválido", Toast.LENGTH\_SHORT).show();

return;

}

JSONObject jsonBody = new JSONObject();

try {

jsonBody.put("id\_usuario", cedulaUsuario);

jsonBody.put("fecha\_gasto", fecha);

jsonBody.put("nombre\_gasto", nombre);

jsonBody.put("valor\_gasto", valor);

jsonBody.put("categoria\_gasto", categoria);

jsonBody.put("descripcion\_gasto", descripcion);

} catch (JSONException e) {

e.printStackTrace();

Toast.makeText(this, "Error al construir la solicitud JSON", Toast.LENGTH\_SHORT).show();

return;

}

RequestBody requestBody = RequestBody.create(MediaType.parse("application/json"), jsonBody.toString());

Request request = new Request.Builder()

.url("http://192.168.1.4:5000/api/gastos/registrar/")

.post(requestBody)

.build();

client.newCall(request).enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

runOnUiThread(() -> Toast.makeText(GestionarGastos.this, "Error al guardar el gasto", Toast.LENGTH\_SHORT).show());

e.printStackTrace();

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

runOnUiThread(() -> {

Toast.makeText(GestionarGastos.this, "Gasto guardado correctamente", Toast.LENGTH\_SHORT).show();

limpiarCampos();

});

} else {

runOnUiThread(() -> Toast.makeText(GestionarGastos.this, "Error al guardar el gasto", Toast.LENGTH\_SHORT).show());

}

}

});

}

private void limpiarCampos() {

idGastos.setText("");

fechaIngreso.setText("");

nombreGasto.setText("");

valorGasto.setText("");

categoriaGasto.setText("");

descripcionGasto.setText("");

}

@Override

public void onBackPressed() {

super.onBackPressed();

finish();

}

}

**EXPLICACION:**

**Declaración de Variables y Objetos**

Se declaran variables para los elementos de la interfaz de usuario, como EditText para los campos de datos del gasto y Button para las acciones de guardar, buscar, editar y eliminar. También se declara un objeto OkHttpClient para realizar solicitudes HTTP y una variable para almacenar la cédula del usuario.

**Método onCreate()**

En este método se configura la vista de la actividad, se obtienen los datos del usuario desde SharedPreferences, y se asignan los elementos de la interfaz de usuario a las variables correspondientes. También se establecen listeners para los botones que llaman a los métodos correspondientes para las acciones de buscar, editar, eliminar y guardar gastos.

**Método buscarGasto()**

Este método realiza una solicitud HTTP GET al servidor para obtener los datos del gasto asociado al ID proporcionado. Si la solicitud es exitosa, los datos se extraen del JSON de respuesta y se muestran en la interfaz de usuario.

**Método editarGasto()**

Este método construye un objeto JSON con los datos del gasto y realiza una solicitud HTTP PUT para actualizar estos datos en el servidor. Si la solicitud es exitosa, se muestra un mensaje de éxito al usuario.

**Método eliminarGasto()**

Este método realiza una solicitud HTTP DELETE para eliminar un gasto especificado por su ID. Si la solicitud es exitosa, se muestra un mensaje de éxito al usuario y se limpian los campos del formulario.

**Método guardarGasto()**

Este método construye un objeto JSON con los datos del gasto y realiza una solicitud HTTP POST para guardar estos datos en el servidor. Antes de realizar la solicitud, se validan los campos del formulario y se convierte el valor del gasto a un formato numérico adecuado.

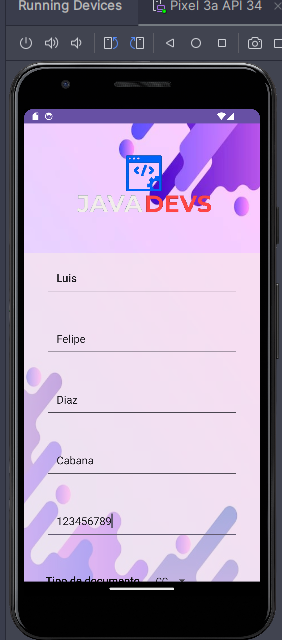
**Funcionamiento de la aplicación**



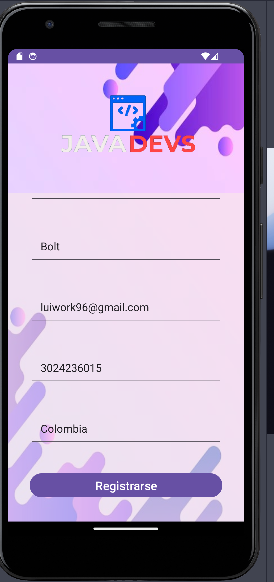
Presionamos en el boton de registrarse

Llenamos los campos con los datos que indican las pistas





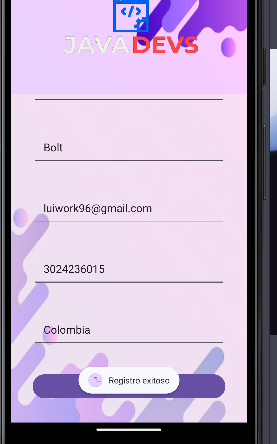
Una ves digitado todos los campos procedemos a presionar el boton de registrarse.



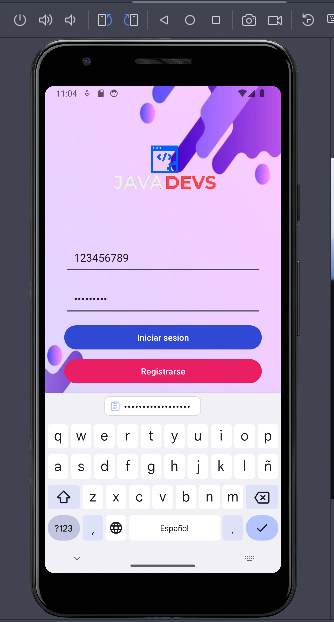
Si al presionar el boton de registrarse falta un campo por llenar, la aplicación lo notifica

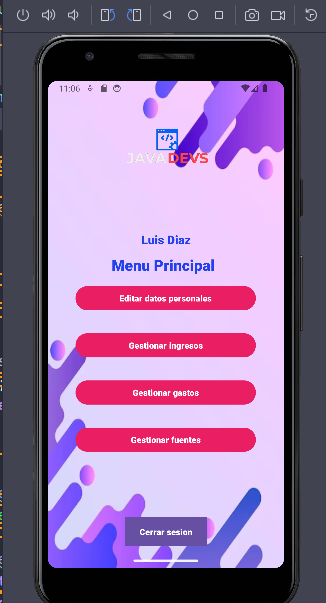


Una ves presionado el boton, la aplicación informa que el registro fue exitoso y lo lleva a la pantalla de login.



Ahora procederemos a iniciar sesion utilizando la cedula y contraseña registrados previamente. Si es correcto los datos ingresados, nos llevara al menu principal de la aplicación y notificara que el inicio de sesion fue correcto.

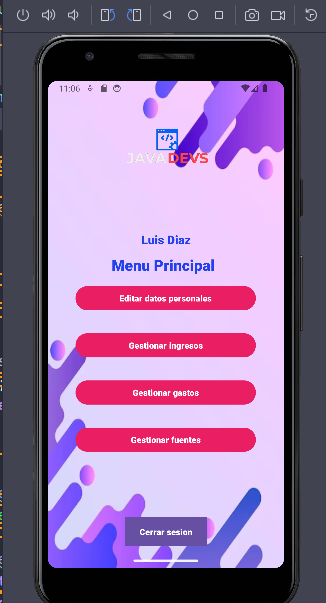




Si colocamos erronamente cedula o la contraseña, la app lo notificara y no permitira el ingreso.



Ahora miremos el menu principal de la aplicación. Podemos observar que en la parte superior se muestra el primer nombre del usuario en sesion y en la parte inferior existe un boton que nos permite cerrar la sesion. El menu cuenta con 4 opciones que nos llevara a ciertas pantalla especificas.



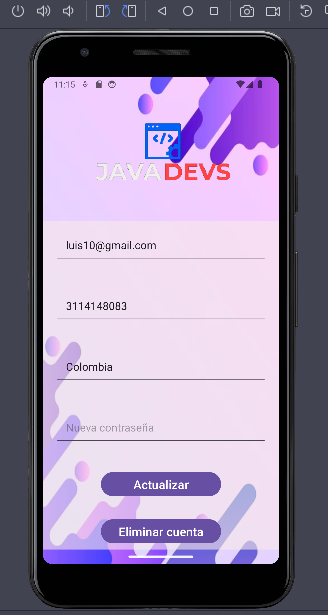
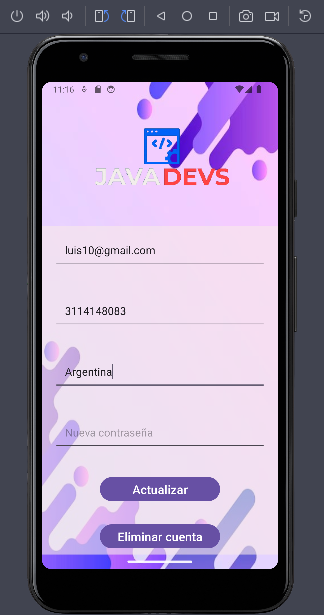
Si presionamos la primera opcion nos llevara a la pantalla para editar datos personales del usuario.



Como se puede apreciar, se cargan los campos con los datos del usuario que se pueden modificar, permitiendo asi la actualizacion de los mismos.



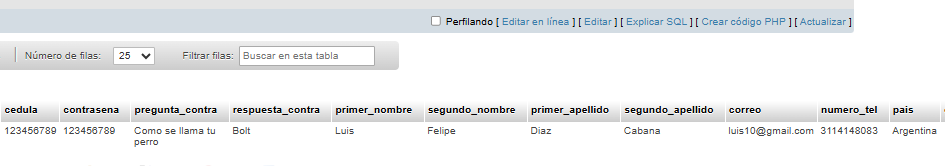
Una ves que hayamos modificado algun campo, solo presionamos el boton para actualizar y se actualizaran los datos.



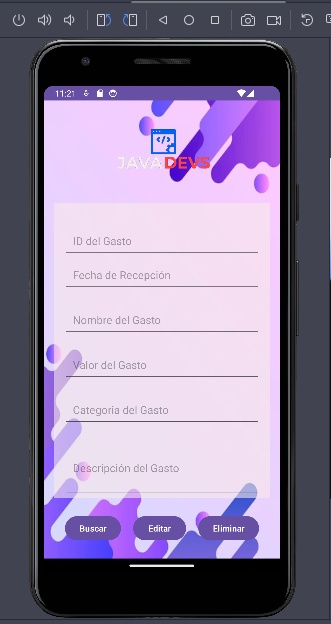
Antes de actualizar el campo



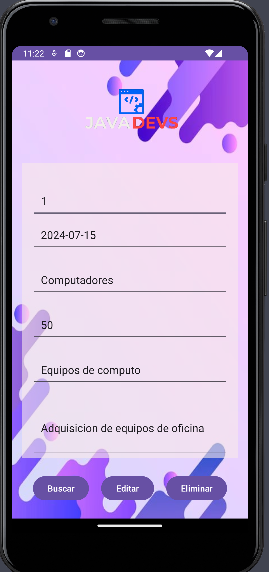
Despues de actualizar el campo y presionar el boton de actualizar



Ahora vamos a la siguiente opcion que gestionar gastos. Cuya funcionalidad es la misma para las otras tres opciones restantes.



Llenamos los campos



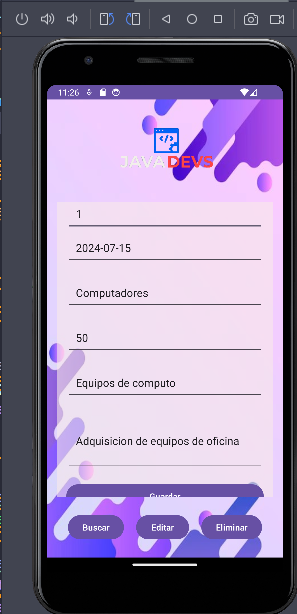
Guardamos la informacion del ingreso nuevo y luego de esto la aplicación notificara con un mensaje exitoso y limpiando los campos.



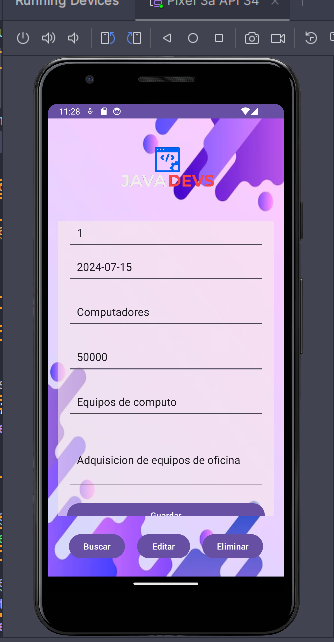
Ahora busquemos el gasto recien creado. Para ello primero colocamos el id con el que lo registramos y luego presionamos el boton de buscar, deberan cargarse los campos asociados a esa ID.



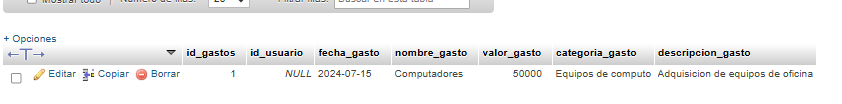
El resultado es



Ahora actualizemos algun campo, en este caso sera el valor del gasto y luego presinamos el boton editar, esto actualizara la informacion.







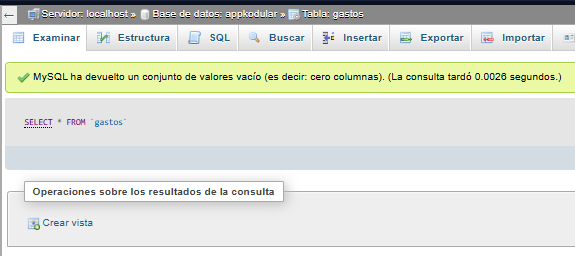
Y por ultimo probemos eliminando el gasto



Debera borrarse de la base de datos y mostrar un mensaje exitoso



Como podemos apreciar el ingreso se elimino correctamente.



# Síntesis

Desarrollar la aplicación de gestión financiera en Android Studio con Java ha sido una experiencia enriquecedora. Nuestra aplicación permite a los usuarios gestionar ingresos, fuentes de ingresos y gastos, ofreciendo una herramienta práctica para el control financiero personal. Enfrentamos y superamos retos técnicos, mejorando nuestras habilidades y reforzando nuestra capacidad de trabajo en equipo. Este proyecto no solo nos permitió crear una herramienta útil, sino que también nos brindó una valiosa oportunidad de aprendizaje y crecimiento profesional.

# Bibliografía

Document. (s/f). Android Developers. Recuperado el 1 de junio de 2024, de <https://developer.android.com/reference/org/w3c/dom/Document>

Aleman, E. S. [@enriquesolisaleman5270]. (2021, May 19). Rest API NodeJs+Android|Kotlin. Youtube. <https://www.youtube.com/watch?v=AZbqFfWTx20>

Moure, M. by B. [@mouredev]. (2020a, enero 17). ANDROID STUDIO: COMO Crear una APP (para Principiantes) 📲 [Tutorial]. Youtube. <https://www.youtube.com/watch?v=BQaxPwZWboA&list=PLNdFk2_brsRdYF0FXDtSaGvluzBNHRbNe>

Codelia [@codelia\_tutorials]. (2021, May 9). Node.js - REST API | Como crear y probar tu API de manera sencilla! Youtube. <https://www.youtube.com/watch?v=AWcm56_eNZg>