**Actividad N. 3**

**Aplicación móvil en Android Studio**

Presentado por:

**LUIS FELIPE DÍAZ CABANA**

**LUIGGI MAR SALEMI SOTO**

**Universidad de Cartagena**

**Ingeniería de software**

**Desarrollo de apps**

**2024**

**Tabla de Contenidos**

[Introducción 3](#_Toc148016971)

[Objetivo general 4](#_Toc148016972)

[Objetivos específicos 4](#_Toc148016973)

[Justificación 5](#_Toc148016974)

[Desarrollo 6](#_Toc148016975)

[Síntesis 58](#_Toc148016976)

[Bibliografía 59](#_Toc148016977)

# 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 el servicio base de datos en la nube de firebase para el almacenamiento de los datos.

# 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 en la base de datos firestore del servicio en nube firebase.

# Objetivos específicos

* Implementar funcionalidades como registro, loguin, información de la aplicación, recuperar contraseña para el acceso y recuperación de datos de los usuarios.
* Un menú principal con las diferentes opciones para gestionar información que permita al usuario guardar, buscar, editar y eliminar datos específicos y un botón para cerrar la sesión del usuario.
* Almacenar los datos en la base de datos firestore del servicio en nube firebase.
* Diseñar la interfaz gráfica para el usuario final de cada una las actividades.

# Justificación

Esta actividad permite adquirir habilidades prácticas en el desarrollo de aplicaciones móviles y servicios en la nube, comprender la arquitectura de aplicaciones full-stack, y aplicar 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.

# 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:**



**Firestore:** es una base de datos NoSQL en tiempo real ofrecida por Firebase, que permite almacenar, sincronizar y consultar datos fácilmente a través de aplicaciones web y móviles. Proporciona capacidades avanzadas como consultas complejas, escalabilidad automática y sincronización en tiempo real, facilitando el desarrollo de aplicaciones interactivas y dinámicas.

**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.

**Frameworks:**

**Firebase Auth**

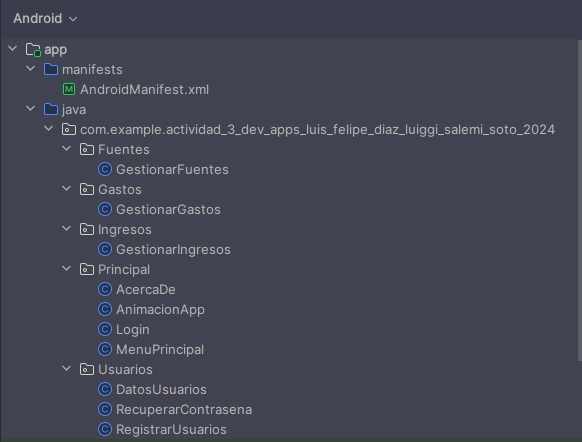
Proporciona servicios de autenticación para permitir a los usuarios iniciar sesión utilizando varios métodos como correo electrónico y redes sociales.

**Firebase firestore**

Ofrece almacenamiento y sincronización de datos en tiempo real, permitiendo manejar datos dinámicos y actualizaciones instantáneas.

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

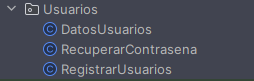
**Paquetes Java**



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.

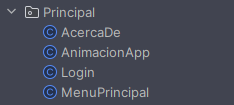
**Contenido de los paquetes:**

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



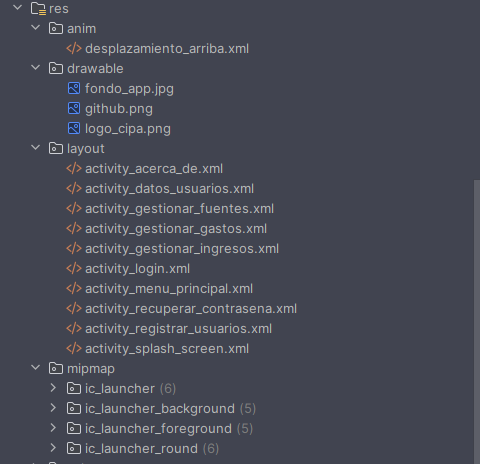
Contiene la logica para gestionar los ingresos de los usuarios ofreciendo las funcinalidades de buscar, editar, eliminar y guardar ingresos.

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**



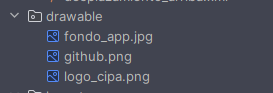
Esta es la carpeta principal de recursos y contiene todos los subdirectorios de recursos de la aplicación, como drawable, layout, values, etc.

Carpeta anim



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

Drawable



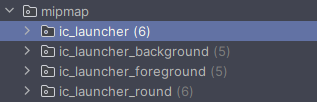
Contiene los recursos gráficos utilizados para la aplicación, En este caso contiene imágenes.

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 campoCorreo, campoContraseña;

TextView recuperarContraseña, acercaDe;

private FirebaseAuth mAuth;

private ProgressDialog progressDialog;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_login);

campoCorreo = findViewById(R.id.campoCorreoLogin);

campoContraseña = findViewById(R.id.campoContraseñaLogin);

recuperarContraseña = findViewById(R.id.recuperarContraseña);

mAuth = FirebaseAuth.getInstance();

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);

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;

});

//Evento del TextView o Label "Registrarse" para que abra la actividad o pantalla de registrar usuarios

registrarse.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

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

startActivity(intent);

}

});

//Evento del TextView o Label "Acerca de" para que abra la actividad o pantalla de acerca de la app

acercaDe.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

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

startActivity(intent); }

});

//Evento del botón "Iniciar Sesión"

BotonLogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

validarCamposLogin();

}

});

//Evento del TextView o Label "Recuperar contraseña" para que abra la actividad o pantalla de recuperar contraseña

recuperarContraseña.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

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

startActivity(intent);

finish();

}

});

}

//Funcion para validar los datos ingresados en los campos de inicio de sesion

public void validarCamposLogin() {

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

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

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

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

return;

} else {

campoCorreo.setError(null);

}

if (password.isEmpty() || password.length() < 12) {

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(email, password);

}

//Funcion para iniciar sesion

public void LogicaInciarSesion(String correo, String contraseña) {

progressDialog.show();

mAuth.signInWithEmailAndPassword(correo, contraseña)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

Toast.makeText(Login.this, "Inicio de sesión exitoso", Toast.LENGTH\_SHORT).show();

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

startActivity(intent);

finish();

} else {

Toast.makeText(Login.this, "Error al iniciar sesión, credenciales incorrectas", Toast.LENGTH\_SHORT).show();

}

}

});

}

}

**EXPLICACION:**

**Declaración de Variables:**

* Se declaran las variables campoCorreo, campoContraseña, recuperarContraseña, acercaDe para los campos de texto de correo electrónico, contraseña y dos textos, respectivamente.
* Se declara la variable mAuth de tipo FirebaseAuth para manejar la autenticación con Firebase.
* Se declara la variable progressDialog de tipo ProgressDialog para mostrar un diálogo de progreso durante el inicio de sesión.

**Método onCreate():**

* Se llama al método onCreate() cuando la actividad se crea por primera vez.
* Se configura el diseño de la actividad utilizando setContentView().
* Se inicializan las variables de los campos de texto, botones y se configuran los oyentes para los eventos de clic en los botones y textos.

**Método validarCamposLogin():**

* Se valida si los campos de correo electrónico y contraseña están vacíos o no cumplen con ciertos criterios (como un formato de correo electrónico válido y una contraseña con al menos 12 caracteres y al menos un número).
* Si los campos no son válidos, se muestra un mensaje de error en los campos correspondientes.
* Si los campos son válidos, se llama al método LogicaInciarSesion() para iniciar sesión.

**Método LogicaInciarSesion():**

* Se muestra un diálogo de progreso para indicar que se está iniciando sesión.
* Se utiliza mAuth.signInWithEmailAndPassword() para intentar iniciar sesión con el correo electrónico y la contraseña proporcionados.
* Se agrega un OnCompleteListener para manejar el resultado de la autenticación.
* Si el inicio de sesión es exitoso, se muestra un mensaje de éxito y se inicia la actividad del menú principal.
* Si el inicio de sesión falla, se muestra un mensaje de error.

**MenuPrincipal**

public class MenuPrincipal extends AppCompatActivity {

TextView vistaNombreUsuario;

private FirebaseAuth mAuth;

Button datosUsuarios, gestionarIngresos, gestionarFuentes, gestionarGastos, Logout;

ProgressDialog progressDialog;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

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

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;

});

// Inicialización de FirebaseAuth

mAuth = FirebaseAuth.getInstance();

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);

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

FirebaseFirestore db = FirebaseFirestore.getInstance();

db.collection("usuarios").document(userId).get()

.addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {

@Override

public void onSuccess(DocumentSnapshot documentSnapshot) {

if (documentSnapshot.exists()) {

String primerNombre = documentSnapshot.getString("primerNombre");

vistaNombreUsuario.setText(primerNombre);

}

}

});

}

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 cerrarSesion() {

progressDialog.show();

mAuth.signOut();

// 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:**

* Se declaran las variables vistaNombreUsuario, mAuth, datosUsuarios, gestionarIngresos, gestionarFuentes, gestionarGastos, Logout y progressDialog.
* vistaNombreUsuario es un TextView para mostrar el nombre del usuario.
* mAuth es una instancia de FirebaseAuth para manejar la autenticación con Firebase.
* datosUsuarios, gestionarIngresos, gestionarFuentes, gestionarGastos, Logout son botones para diferentes funciones del menú.
* progressDialog es un ProgressDialog para mostrar un diálogo de progreso.

**Método onCreate():**

* Se llama al método onCreate() cuando la actividad se crea por primera vez.
* Se configura el diseño de la actividad utilizando setContentView().
* Se inicializan las variables y se configuran los oyentes para los eventos de clic en los botones.
* Se utiliza FirebaseAuth.getInstance() para obtener una instancia de FirebaseAuth.
* Se obtiene el usuario actualmente autenticado y se muestra su nombre de usuario si está disponible.

**Método cerrarSesion():**

* Muestra un diálogo de progreso para indicar que se está cerrando la sesión.
* Utiliza mAuth.signOut() para cerrar la sesión del usuario actual.
* Utiliza un Handler para retrasar el inicio de la siguiente actividad y mostrar un mensaje de éxito.
* Finalmente, inicia la actividad de inicio de sesión (Login) y finaliza la actividad actual.

**DatosUsuarios**

public class DatosUsuarios extends AppCompatActivity {

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

Button guardar, eliminarUser;

private FirebaseAuth mAuth;

private FirebaseFirestore db;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

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

// 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;

});

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);

currentPasswordInput = findViewById(R.id.currentPasswordInput);

guardar = findViewById(R.id.actualizarDatos);

eliminarUser = findViewById(R.id.eliminarUsuario);

mAuth = FirebaseAuth.getInstance();

db = FirebaseFirestore.getInstance();

cargarDatosUsuario();

guardar.setOnClickListener(v -> {

validarCampos();

});

eliminarUser.setOnClickListener(v -> {

eliminarCuentaUsuario();

});

}

private void cargarDatosUsuario() {

FirebaseUser user = mAuth.getCurrentUser();

if (user == null) {

Toast.makeText(this, "No hay usuario autenticado.", Toast.LENGTH\_SHORT).show();

return;

}

String userId = user.getUid();

DocumentReference userRef = db.collection("usuarios").document(userId);

userRef.get()

.addOnSuccessListener(documentSnapshot -> {

if (documentSnapshot.exists()) {

telefono.setText(documentSnapshot.getString("telefono"));

email.setText(user.getEmail());

pregunta.setText(documentSnapshot.getString("pregunta"));

respuesta.setText(documentSnapshot.getString("respuesta"));

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

} else {

Toast.makeText(this, "No se encontraron datos del usuario.", Toast.LENGTH\_SHORT).show();

}

})

.addOnFailureListener(e -> {

Toast.makeText(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();

String currentPassword = currentPasswordInput.getText().toString().trim();

if (contraseña.isEmpty() || password.length() < 12) {

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

return;

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

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

return;

} else {

password.setError(null);

}

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);

}

if (currentPassword.isEmpty()) {

currentPasswordInput.setError("Ingrese su contraseña actual");

return;

} else {

currentPasswordInput.setError(null);

}

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

}

public void actualizarDatosUsuario(String telefono, String email, String password, String pregunta, String respuesta, String pais, String currentPassword) {

FirebaseUser user = mAuth.getCurrentUser();

if (user == null) {

Toast.makeText(this, "No hay usuario autenticado.", Toast.LENGTH\_SHORT).show();

return;

}

String userId = user.getUid();

DocumentReference userRef = db.collection("usuarios").document(userId);

// Datos a actualizar en Firestore

Map<String, Object> updates = new HashMap<>();

updates.put("telefono", telefono);

updates.put("email", email);

updates.put("pregunta", pregunta);

updates.put("respuesta", respuesta);

updates.put("pais", pais);

// Reautenticar al usuario antes de actualizar su email y contraseña

AuthCredential credential = EmailAuthProvider.getCredential(user.getEmail(), currentPassword);

user.reauthenticate(credential)

.addOnCompleteListener(task -> {

if (task.isSuccessful()) {

// Actualizar datos en Firestore

userRef.update(updates)

.addOnSuccessListener(aVoid -> {

// Actualizar el correo electrónico

user.updateEmail(email)

.addOnCompleteListener(task1 -> {

if (task1.isSuccessful()) {

// Actualizar la contraseña

user.updatePassword(password)

.addOnCompleteListener(task2 -> {

if (task2.isSuccessful()) {

Toast.makeText(this, "Datos actualizados exitosamente.", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(this, "Error al actualizar la contraseña.", Toast.LENGTH\_SHORT).show();

}

});

} else {

Toast.makeText(this, "Error al actualizar el correo.", Toast.LENGTH\_SHORT).show();

}

});

})

.addOnFailureListener(e -> {

Toast.makeText(this, "Error al actualizar los datos en Firestore.", Toast.LENGTH\_SHORT).show();

});

} else {

Toast.makeText(this, "Error en la reautenticación. Por favor, verifica tu contraseña actual.", Toast.LENGTH\_SHORT).show();

}

});

}

private void eliminarCuentaUsuario() {

FirebaseUser user = mAuth.getCurrentUser();

if (user == null) {

Toast.makeText(this, "No hay usuario autenticado.", Toast.LENGTH\_SHORT).show();

return;

}

String currentPassword = currentPasswordInput.getText().toString().trim();

if (currentPassword.isEmpty()) {

Toast.makeText(this, "Por favor, ingrese su contraseña actual.", Toast.LENGTH\_SHORT).show();

return;

}

String userId = user.getUid();

DocumentReference userRef = db.collection("usuarios").document(userId);

// Eliminar el usuario de Firebase Authentication

AuthCredential credential = EmailAuthProvider.getCredential(user.getEmail(), currentPassword);

user.reauthenticate(credential)

.addOnSuccessListener(aVoid -> {

user.delete()

.addOnSuccessListener(aVoid1 -> {

// Eliminar los datos del usuario de Firestore

userRef.delete()

.addOnSuccessListener(aVoid2 -> {

Toast.makeText(this, "Cuenta y datos eliminados exitosamente.", Toast.LENGTH\_SHORT).show();

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

finish();

})

.addOnFailureListener(e -> {

Toast.makeText(this, "Error al eliminar los datos del usuario de Firestore: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

});

})

.addOnFailureListener(e -> {

Toast.makeText(this, "Error al eliminar la cuenta de autenticación: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

});

})

.addOnFailureListener(e -> {

Toast.makeText(this, "Error al reautenticar al usuario: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

});

}

public void onBackPressed() {

super.onBackPressed();

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

startActivity(intent);

finish();

}

}

**EXPLICACION:**

Declaración de Variables:

* Se declaran las variables password, email, pregunta, respuesta, telefono, pais, currentPasswordInput, guardar y eliminarUser.
* Estas variables representan campos de texto, botones y objetos de Firebase necesarios para la funcionalidad de esta actividad.

Método onCreate():

* Se llama al método onCreate() cuando la actividad se crea por primera vez.
* Se configura el diseño de la actividad utilizando setContentView().
* Se inicializan las variables y se configuran los oyentes para los eventos de clic en los botones.
* Se obtiene una instancia de FirebaseAuth y FirebaseFirestore.

Método cargarDatosUsuario():

* Obtiene el usuario actualmente autenticado y carga sus datos almacenados en Firestore, como correo electrónico, teléfono, pregunta de seguridad, etc.
* Muestra los datos del usuario en los campos correspondientes de la interfaz de usuario.

Método validarCampos():

* Valida los campos de contraseña, correo electrónico, pregunta de seguridad, respuesta de seguridad, teléfono y país.
* Muestra mensajes de error si los campos no son válidos.

Método actualizarDatosUsuario():

* Actualiza los datos del usuario en Firestore, incluido el correo electrónico y la contraseña.
* Reautentica al usuario antes de actualizar los datos sensibles.
* Utiliza transacciones para garantizar que los datos se actualicen correctamente en Firestore y Firebase Authentication.

Método eliminarCuentaUsuario():

* Elimina la cuenta del usuario de Firebase Authentication y sus datos asociados en Firestore.
* Reautentica al usuario antes de eliminar la cuenta para garantizar la autenticidad.
* Muestra mensajes de error si ocurren problemas durante el proceso de eliminación.

Método onBackPressed():

Sobrescribe el método onBackPressed() para permitir que el usuario regrese al menú principal cuando presiona el botón de retroceso del dispositivo.

**RegistrarUsuarios**

public class RegistrarUsuarios extends AppCompatActivity {

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

Button registrarse;

TextView errorID;

Spinner tipoIdentidad;

private FirebaseAuth mAuth;

@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);

tipoIdentidad = findViewById(R.id.tipoDocumento);

errorID = findViewById(R.id.errorTipoID);

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();

}

});

mAuth = FirebaseAuth.getInstance();

// Este es el código para el comportamiento del componente spinner para que el usuario pueda elegir una opción

String[] opcionesTipoIdentificacion = {"CC", "TI", "CE", "LC", "PPT"};

ArrayAdapter<String> adapter = new ArrayAdapter<>(RegistrarUsuarios.this, android.R.layout.simple\_spinner\_item, opcionesTipoIdentificacion);

tipoIdentidad.setAdapter(adapter);

}

//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 tipoDocumentoText = tipoIdentidad.getSelectedItem().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 (tipoDocumentoText.isEmpty()) {

errorID.setText("Seleccione su tipo de documento");

return;

} else {

errorID.setText("");

}

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() < 12) {

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(email, password);

}

//Este es el código para registrar un usuario en Firestore y Firebase Authentication una ves que le hayamos pasado los datos del metodo guardarInformacionUsuario

public void registrarUsuario(String email, String password) {

mAuth.createUserWithEmailAndPassword(email, password)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

// Registro exitoso

FirebaseUser user = mAuth.getCurrentUser();

guardarInformacionUsuario(user.getUid(), primerNombre.getText().toString(), segundoNombre.getText().toString(), tipoIdentidad.getSelectedItem().toString().trim(), primerApellido.getText().toString(), segundoApellido.getText().toString(), preguntaContraseña.getText().toString(), respuestaContraseña.getText().toString(), cedula.getText().toString(), telefono.getText().toString(), pais.getText().toString());

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

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

startActivity(intent);

finish();

} else {

// Error en el registro

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

}

}

});

}

//Este es el código para guardar la información adicional al correo y contraseña del usuario en Firestore antes de poder registrarlo en la base de datos de Firestore

private void guardarInformacionUsuario(String userId, String primerNombre, String segundoNombre, String tipoId, String primerApellido, String segundoApellido, String pregunta, String respuesta, String cedula, String tel, String pais) {

// Referencia a la colección de usuarios en Firestore

FirebaseFirestore db = FirebaseFirestore.getInstance();

DocumentReference userRef = db.collection("usuarios").document(userId);

// Crear un objeto con la información del usuario

Map<String, Object> usuario = new HashMap<>();

usuario.put("primerNombre", primerNombre);

usuario.put("segundoNombre", segundoNombre);

usuario.put("primerApellido", primerApellido);

usuario.put("segundoApellido", segundoApellido);

usuario.put("pregunta", pregunta);

usuario.put("respuesta", respuesta);

usuario.put("cedula", cedula);

usuario.put("tipoIdentidad", tipoId);

usuario.put("telefono", tel);

usuario.put("pais", pais);

// Guardar la información del usuario en Firestore

userRef.set(usuario)

.addOnSuccessListener(new OnSuccessListener<Void>() {

@Override

public void onSuccess(Void aVoid) {

// Informar al usuario que se ha registrado correctamente

Toast.makeText(RegistrarUsuarios.this, "Usuario registrado exitosamente.", Toast.LENGTH\_SHORT).show();

}

})

.addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

// Manejar el error al guardar la información del usuario en Firestore

Toast.makeText(RegistrarUsuarios.this, "Error al guardar la información del usuario.", Toast.LENGTH\_SHORT).show();

}

});

}

}

**EXPLICACION**

**Declaración de Variables:**

* Se declaran las variables contraseñaRegistro, correoElectronico, primerNombre, segundoNombre, primerApellido, segundoApellido, preguntaContraseña, respuestaContraseña, cedula, telefono, pais, registrarse, errorID y tipoIdentidad.
* Estas variables representan campos de texto, botones, textos y un spinner necesarios para el proceso de registro.

**Método onCreate():**

* Se llama al método onCreate() cuando la actividad se crea por primera vez.
* Se configura el diseño de la actividad utilizando setContentView().
* Se inicializan las variables y se configuran los oyentes para los eventos de clic en los botones.
* Se crea un ArrayAdapter para configurar las opciones del Spinner (tipoIdentidad).

**Método validarCampos():**

* Valida los campos del formulario de registro, como nombre, apellido, tipo de documento, correo electrónico, contraseña, etc.
* Muestra mensajes de error si los campos no son válidos.

**Método registrarUsuario():**

* Utiliza FirebaseAuth para crear un nuevo usuario con correo electrónico y contraseña.
* Si el registro es exitoso, guarda la información adicional del usuario en Firestore utilizando el método guardarInformacionUsuario() y redirige al usuario a la pantalla de inicio de sesión.
* Muestra un mensaje de error si ocurre algún problema durante el registro.

**Método guardarInformacionUsuario():**

* Guarda la información adicional del usuario en Firestore (como nombre, apellido, tipo de documento, teléfono, etc.) después de que el usuario se haya registrado correctamente en Firebase Authentication.
* Utiliza un objeto Map para almacenar los datos del usuario y luego los guarda en Firestore.

**GestionIngresos**

package com.example.actividad\_3\_dev\_apps\_luis\_felipe\_diaz\_luiggi\_salemi\_soto\_2024.Ingresos;

import static androidx.constraintlayout.widget.Constraints.TAG;

import android.app.DatePickerDialog;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.DatePicker;

import android.widget.EditText;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import com.example.actividad\_3\_dev\_apps\_luis\_felipe\_diaz\_luiggi\_salemi\_soto\_2024.Principal.Login;

import com.example.actividad\_3\_dev\_apps\_luis\_felipe\_diaz\_luiggi\_salemi\_soto\_2024.Principal.MenuPrincipal;

import com.example.actividad\_3\_dev\_apps\_luis\_felipe\_diaz\_luiggi\_salemi\_soto\_2024.R;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.firestore.DocumentSnapshot;

import com.google.firebase.firestore.FirebaseFirestore;

import java.util.Calendar;

import java.util.HashMap;

import java.util.Map;

public class GestionarIngresos extends AppCompatActivity {

private EditText idIngreso, fechaIngreso, valorIngreso, descripcionIngreso;

private Button botonGuardar, botonBuscar, botonEditar, botonEliminar;

private FirebaseFirestore db;

private FirebaseAuth mAuth;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

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

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;

});

// Inicializar Firestore y Auth

db = FirebaseFirestore.getInstance();

mAuth = FirebaseAuth.getInstance();

// Inicializar vistas

idIngreso = findViewById(R.id.idIngreso);

fechaIngreso = findViewById(R.id.fechaIngreso);

valorIngreso = findViewById(R.id.valorIngreso);

descripcionIngreso = findViewById(R.id.descripcionIngreso);

botonGuardar = findViewById(R.id.botonGuardar);

botonBuscar = findViewById(R.id.botonBuscar);

botonEditar = findViewById(R.id.botonEditar);

botonEliminar = findViewById(R.id.botonEliminar);

fechaIngreso.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

showDatePickerDialog();

}

});

// Acción botón Buscar

botonBuscar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

buscarIngreso();

}

});

// Acción botón Editar

botonEditar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

editarIngreso();

}

});

// Acción botón Eliminar

botonEliminar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

eliminarIngreso();

}

});

// Acción botón Guardar

botonGuardar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

guardarIngreso();

}

});

}

private void showDatePickerDialog() {

Calendar calendar = Calendar.getInstance();

int year = calendar.get(Calendar.YEAR);

int month = calendar.get(Calendar.MONTH);

int day = calendar.get(Calendar.DAY\_OF\_MONTH);

DatePickerDialog datePickerDialog = new DatePickerDialog(this, new DatePickerDialog.OnDateSetListener() {

@Override

public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {

// Mes en DatePicker empieza en 0, por eso sumamos 1

String selectedDate = year + "-" + (monthOfYear + 1) + "-" + dayOfMonth;

fechaIngreso.setText(selectedDate);

}

}, year, month, day);

datePickerDialog.show();

}

private void buscarIngreso() {

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

if (id.isEmpty()) {

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

return;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

db.collection("usuarios").document(userId).collection("ingresos").document(id)

.get()

.addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {

@Override

public void onComplete(@NonNull Task<DocumentSnapshot> task) {

if (task.isSuccessful()) {

DocumentSnapshot document = task.getResult();

if (document.exists()) {

fechaIngreso.setText(document.getString("fechaIngreso"));

valorIngreso.setText(document.getString("valorIngreso"));

descripcionIngreso.setText(document.getString("descripcionIngreso"));

} else {

Toast.makeText(GestionarIngresos.this, "No se encontró el ingreso", Toast.LENGTH\_SHORT).show();

}

} else {

Log.d(TAG, "Error al obtener el documento: ", task.getException());

}

}

});

}

}

private void editarIngreso() {

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

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

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

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

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

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

return;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

Map<String, Object> ingreso = new HashMap<>();

ingreso.put("fechaIngreso", fecha);

ingreso.put("valorIngreso", valor);

ingreso.put("descripcionIngreso", descripcion);

db.collection("usuarios").document(userId).collection("ingresos").document(id)

.update(ingreso)

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

Toast.makeText(GestionarIngresos.this, "Ingreso actualizado correctamente", Toast.LENGTH\_SHORT).show();

} else {

Log.d(TAG, "Error al actualizar el documento: ", task.getException());

}

}

});

}

}

private void eliminarIngreso() {

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

if (id.isEmpty()) {

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

return;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

db.collection("usuarios").document(userId).collection("ingresos").document(id)

.delete()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

Toast.makeText(GestionarIngresos.this, "Ingreso eliminado correctamente", Toast.LENGTH\_SHORT).show();

limpiarCampos();

} else {

Log.d(TAG, "Error al eliminar el documento: ", task.getException());

}

}

});

}

}

private void guardarIngreso() {

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

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

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

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

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

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

return;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

Map<String, Object> ingreso = new HashMap<>();

ingreso.put("idIngreso", id);

ingreso.put("fechaIngreso", fecha);

ingreso.put("valorIngreso", valor);

ingreso.put("descripcionIngreso", descripcion);

db.collection("usuarios").document(userId).collection("ingresos").document(id)

.set(ingreso)

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

Toast.makeText(GestionarIngresos.this, "Ingreso guardado correctamente", Toast.LENGTH\_SHORT).show();

limpiarCampos();

} else {

Log.d(TAG, "Error al guardar el documento: ", task.getException());

}

}

});

}

}

private void limpiarCampos() {

idIngreso.setText("");

fechaIngreso.setText("");

valorIngreso.setText("");

descripcionIngreso.setText("");

}

@Override

public void onBackPressed() {

super.onBackPressed();

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

startActivity(intent);

finish();

}

}

**EXPLICACION**

**Declaración de Variables:**

* Se declaran las variables idIngreso, fechaIngreso, valorIngreso, descripcionIngreso, botonGuardar, botonBuscar, botonEditar y botonEliminar.
* Estas variables representan campos de texto y botones necesarios para gestionar los ingresos.

**Método onCreate():**

* Se llama al método onCreate() cuando la actividad se crea por primera vez.
* Se configura el diseño de la actividad utilizando setContentView().
* Se inicializan Firestore y FirebaseAuth.
* Se inicializan las vistas y se configuran los oyentes para los eventos de clic en los botones.

**Método showDatePickerDialog():**

Muestra un cuadro de diálogo para seleccionar una fecha y establece la fecha seleccionada en el campo fechaIngreso.

**Método buscarIngreso():**

* Busca un ingreso en Firestore utilizando el ID ingresado en idIngreso.
* Si el ingreso se encuentra, llena los campos fechaIngreso, valorIngreso y descripcionIngreso con los datos obtenidos de Firestore.
* Muestra un mensaje si no se encuentra el ingreso o si ocurre un error durante la búsqueda.

**Método editarIngreso():**

* Edita un ingreso en Firestore utilizando el ID ingresado en idIngreso.
* Actualiza los datos del ingreso con los valores ingresados en fechaIngreso, valorIngreso y descripcionIngreso.
* Muestra un mensaje si la actualización es exitosa o si ocurre un error durante la actualización.

**Método eliminarIngreso():**

* Elimina un ingreso en Firestore utilizando el ID ingresado en idIngreso.
* Muestra un mensaje si la eliminación es exitosa o si ocurre un error durante la eliminación.
* Limpia los campos después de la eliminación.

**Método guardarIngreso():**

* Guarda un nuevo ingreso en Firestore utilizando los valores ingresados en idIngreso, fechaIngreso, valorIngreso y descripcionIngreso.
* Muestra un mensaje si el guardado es exitoso o si ocurre un error durante el guardado.
* Limpia los campos después de guardar.

**Método limpiarCampos():**

Limpia los campos idIngreso, fechaIngreso, valorIngreso y descripcionIngreso.

**Método onBackPressed():**

* Maneja el evento de retroceso del dispositivo.
* Redirige al usuario a la actividad MenuPrincipal cuando se presiona el botón de retroceso.

**GestionarGastos**

public class GestionarGastos extends AppCompatActivity {

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

private Button botonGuardar, botonBuscar, botonEditar, botonEliminar;

private FirebaseFirestore db;

private FirebaseAuth mAuth;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

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

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;

});

db = FirebaseFirestore.getInstance();

mAuth = FirebaseAuth.getInstance();

// 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(new View.OnClickListener() {

@Override

public void onClick(View v) {

buscarGasto();

}

});

botonEditar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

editarGasto();

}

});

botonEliminar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

eliminarGasto();

}

});

botonGuardar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

guardarGasto();

}

});

}

//funciones para los botones de la interfaz de usuario

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;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

db.collection("usuarios").document(userId).collection("gastos").document(id)

.get()

.addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {

@Override

public void onComplete(@NonNull Task<DocumentSnapshot> task) {

if (task.isSuccessful()) {

DocumentSnapshot document = task.getResult();

if (document.exists()) {

fechaIngreso.setText(document.getString("fechaIngreso"));

nombreGasto.setText(document.getString("nombreGasto"));

valorGasto.setText(document.getString("valorGasto"));

categoriaGasto.setText(document.getString("categoriaGasto"));

descripcionGasto.setText(document.getString("descripcionGasto"));

} else {

Toast.makeText(GestionarGastos.this, "No se encontró el gasto", Toast.LENGTH\_SHORT).show();

}

} else {

Log.d(TAG, "Error al obtener el documento: ", task.getException());

}

}

});

}

}

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;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

Map<String, Object> gasto = new HashMap<>();

gasto.put("fechaIngreso", fecha);

gasto.put("nombreGasto", nombre);

gasto.put("valorGasto", valor);

gasto.put("categoriaGasto", categoria);

gasto.put("descripcionGasto", descripcion);

db.collection("usuarios").document(userId).collection("gastos").document(id)

.update(gasto)

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

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

} else {

Log.d(TAG, "Error al actualizar el documento: ", task.getException());

}

}

});

}

}

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;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

db.collection("usuarios").document(userId).collection("gastos").document(id)

.delete()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

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

limpiarCampos();

} else {

Log.d(TAG, "Error al eliminar el documento: ", task.getException());

}

}

});

}

}

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;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

Map<String, Object> gasto = new HashMap<>();

gasto.put("idGasto", id);

gasto.put("fechaIngreso", fecha);

gasto.put("nombreGasto", nombre);

gasto.put("valorGasto", valor);

gasto.put("categoriaGasto", categoria);

gasto.put("descripcionGasto", descripcion);

db.collection("usuarios").document(userId).collection("gastos").document(id)

.set(gasto)

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

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

limpiarCampos();

} else {

Log.d(TAG, "Error al guardar el documento: ", task.getException());

}

}

});

}

}

private void limpiarCampos() {

idGastos.setText("");

fechaIngreso.setText("");

nombreGasto.setText("");

valorGasto.setText("");

categoriaGasto.setText("");

descripcionGasto.setText("");

}

@Override

public void onBackPressed() {

super.onBackPressed();

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

startActivity(intent);

finish();

}

}

**EXPLICACION**

**Declaración de Variables:**

* Se declaran las variables idGastos, fechaIngreso, nombreGasto, valorGasto, categoriaGasto, descripcionGasto, botonGuardar, botonBuscar, botonEditar y botonEliminar.
* Estas variables representan campos de texto y botones necesarios para gestionar los gastos.

**Método onCreate():**

* Se llama al método onCreate() cuando la actividad se crea por primera vez.
* Se configura el diseño de la actividad utilizando setContentView().
* Se inicializan Firestore y FirebaseAuth.
* Se inicializan las vistas y se configuran los oyentes para los eventos de clic en los botones.

**Método buscarGasto():**

* Busca un gasto en Firestore utilizando el ID ingresado en idGastos.
* Si el gasto se encuentra, llena los campos fechaIngreso, nombreGasto, valorGasto, categoriaGasto y descripcionGasto con los datos obtenidos de Firestore.
* Muestra un mensaje si no se encuentra el gasto o si ocurre un error durante la búsqueda.

**Método editarGasto():**

* Edita un gasto en Firestore utilizando el ID ingresado en idGastos.
* Actualiza los datos del gasto con los valores ingresados en fechaIngreso, nombreGasto, valorGasto, categoriaGasto y descripcionGasto.
* Muestra un mensaje si la actualización es exitosa o si ocurre un error durante la actualización.

**Método eliminarGasto():**

* Elimina un gasto en Firestore utilizando el ID ingresado en idGastos.
* Muestra un mensaje si la eliminación es exitosa o si ocurre un error durante la eliminación.
* Limpia los campos después de la eliminación.

**Método guardarGasto():**

* Guarda un nuevo gasto en Firestore utilizando los valores ingresados en idGastos, fechaIngreso, nombreGasto, valorGasto, categoriaGasto y descripcionGasto.
* Muestra un mensaje si el guardado es exitoso o si ocurre un error durante el guardado.
* Limpia los campos después de guardar.

**Método limpiarCampos():**

Limpia los campos idGastos, fechaIngreso, nombreGasto, valorGasto, categoriaGasto y descripcionGasto.

**Método onBackPressed():**

* Maneja el evento de retroceso del dispositivo.
* Redirige al usuario a la actividad MenuPrincipal cuando se presiona el botón de retroceso.

**GestionarFuentes**

public class GestionarFuentes extends AppCompatActivity {

private static final String TAG = "GestionarFuentes";

private EditText idFuente, fechaIngreso, nombreFuente, descripcionFuente;

private Button botonGuardar, botonBuscar, botonEditar, botonEliminar;

private FirebaseFirestore db;

private FirebaseAuth mAuth;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

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

// Inicialización de FirebaseAuth y FirebaseFirestore

mAuth = FirebaseAuth.getInstance();

db = FirebaseFirestore.getInstance();

// Inicialización de vistas

idFuente = findViewById(R.id.idFuente);

fechaIngreso = findViewById(R.id.fechaIngreso);

nombreFuente = findViewById(R.id.nombreFuente);

descripcionFuente = findViewById(R.id.descripcionFuente);

botonGuardar = findViewById(R.id.botonGuardar);

botonBuscar = findViewById(R.id.botonBuscar);

botonEditar = findViewById(R.id.botonEditar);

botonEliminar = findViewById(R.id.botonEliminar);

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;

});

// Configurar DatePicker para fechaIngreso

fechaIngreso.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

showDatePickerDialog();

}

});

// Acción botón Buscar

botonBuscar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

buscarFuente();

}

});

// Acción botón Editar

botonEditar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

editarFuente();

}

});

// Acción botón Eliminar

botonEliminar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

eliminarFuente();

}

});

// Acción botón Guardar

botonGuardar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

guardarFuente();

}

});

}

private void showDatePickerDialog() {

Calendar calendar = Calendar.getInstance();

int year = calendar.get(Calendar.YEAR);

int month = calendar.get(Calendar.MONTH);

int day = calendar.get(Calendar.DAY\_OF\_MONTH);

DatePickerDialog datePickerDialog = new DatePickerDialog(this, new DatePickerDialog.OnDateSetListener() {

@Override

public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {

// Mes en DatePicker empieza en 0, por eso sumamos 1

String selectedDate = year + "-" + (monthOfYear + 1) + "-" + dayOfMonth;

fechaIngreso.setText(selectedDate);

}

}, year, month, day);

datePickerDialog.show();

}

private void buscarFuente() {

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

if (id.isEmpty()) {

Toast.makeText(this, "Por favor ingrese el ID de la fuente", Toast.LENGTH\_SHORT).show();

return;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

db.collection("usuarios").document(userId).collection("fuentes").document(id)

.get()

.addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {

@Override

public void onComplete(@NonNull Task<DocumentSnapshot> task) {

if (task.isSuccessful()) {

DocumentSnapshot document = task.getResult();

if (document.exists()) {

fechaIngreso.setText(document.getString("fechaIngreso"));

nombreFuente.setText(document.getString("nombreFuente"));

descripcionFuente.setText(document.getString("descripcionFuente"));

} else {

Toast.makeText(GestionarFuentes.this, "No se encontró la fuente", Toast.LENGTH\_SHORT).show();

}

} else {

Log.d(TAG, "Error al obtener el documento: ", task.getException());

}

}

});

}

}

private void editarFuente() {

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

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

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

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

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

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

return;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

Map<String, Object> fuente = new HashMap<>();

fuente.put("fechaIngreso", fecha);

fuente.put("nombreFuente", nombre);

fuente.put("descripcionFuente", descripcion);

db.collection("usuarios").document(userId).collection("fuentes").document(id)

.update(fuente)

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

Toast.makeText(GestionarFuentes.this, "Fuente actualizada correctamente", Toast.LENGTH\_SHORT).show();

} else {

Log.d(TAG, "Error al actualizar el documento: ", task.getException());

}

}

});

}

}

private void eliminarFuente() {

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

if (id.isEmpty()) {

Toast.makeText(this, "Por favor ingrese el ID de la fuente", Toast.LENGTH\_SHORT).show();

return;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

db.collection("usuarios").document(userId).collection("fuentes").document(id)

.delete()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

Toast.makeText(GestionarFuentes.this, "Fuente eliminada correctamente", Toast.LENGTH\_SHORT).show();

limpiarCampos();

} else {

Log.d(TAG, "Error al eliminar el documento: ", task.getException());

}

}

});

}

}

private void guardarFuente() {

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

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

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

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

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

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

return;

}

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

String userId = user.getUid();

Map<String, Object> fuente = new HashMap<>();

fuente.put("idFuente", id);

fuente.put("fechaIngreso", fecha);

fuente.put("nombreFuente", nombre);

fuente.put("descripcionFuente", descripcion);

db.collection("usuarios").document(userId).collection("fuentes").document(id)

.set(fuente)

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if (task.isSuccessful()) {

Toast.makeText(GestionarFuentes.this, "Fuente guardada correctamente", Toast.LENGTH\_SHORT).show();

limpiarCampos();

} else {

Log.d(TAG, "Error al guardar el documento: ", task.getException());

}

}

});

}

}

private void limpiarCampos() {

idFuente.setText("");

fechaIngreso.setText("");

nombreFuente.setText("");

descripcionFuente.setText("");

}

@Override

public void onBackPressed() {

super.onBackPressed();

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

startActivity(intent);

finish();

}

}

**EXPLICACION**

**Declaración de Variables:**

* Se declaran las variables idFuente, fechaIngreso, nombreFuente, descripcionFuente, botonGuardar, botonBuscar, botonEditar y botonEliminar.
* Estas variables representan campos de texto y botones necesarios para gestionar las fuentes.

**Método onCreate():**

* Se llama al método onCreate() cuando la actividad se crea por primera vez.
* Se configura el diseño de la actividad utilizando setContentView().
* Se inicializan Firestore y FirebaseAuth.
* Se inicializan las vistas y se configuran los oyentes para los eventos de clic en los botones.

**Método showDatePickerDialog():**

Muestra un cuadro de diálogo para seleccionar una fecha y establece la fecha seleccionada en el campo fechaIngreso.

**Método buscarFuente():**

* Busca una fuente en Firestore utilizando el ID ingresado en idFuente.
* Si la fuente se encuentra, llena los campos fechaIngreso, nombreFuente y descripcionFuente con los datos obtenidos de Firestore.
* Muestra un mensaje si no se encuentra la fuente o si ocurre un error durante la búsqueda.

**Método editarFuente():**

* Edita una fuente en Firestore utilizando el ID ingresado en idFuente.
* Actualiza los datos de la fuente con los valores ingresados en fechaIngreso, nombreFuente y descripcionFuente.
* Muestra un mensaje si la actualización es exitosa o si ocurre un error durante la actualización.

**Método eliminarFuente():**

* Elimina una fuente en Firestore utilizando el ID ingresado en idFuente.
* Muestra un mensaje si la eliminación es exitosa o si ocurre un error durante la eliminación.
* Limpia los campos después de la eliminación.

**Método guardarFuente():**

* Guarda una nueva fuente en Firestore utilizando los valores ingresados en idFuente, fechaIngreso, nombreFuente y descripcionFuente.
* Muestra un mensaje si el guardado es exitoso o si ocurre un error durante el guardado.
* Limpia los campos después de guardar.

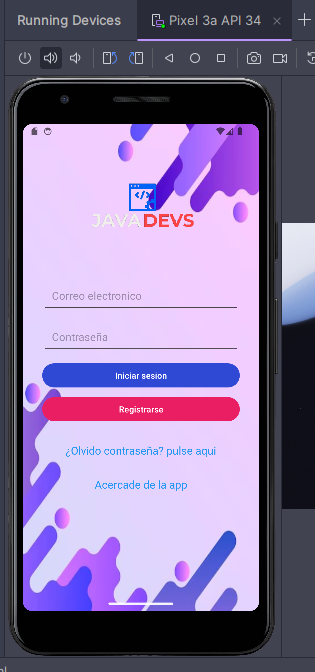
**Método limpiarCampos():**

Limpia los campos idFuente, fechaIngreso, nombreFuente y descripcionFuente.

**Método onBackPressed():**

* Maneja el evento de retroceso del dispositivo.
* Redirige al usuario a la actividad MenuPrincipal cuando se presiona el botón de retroceso.

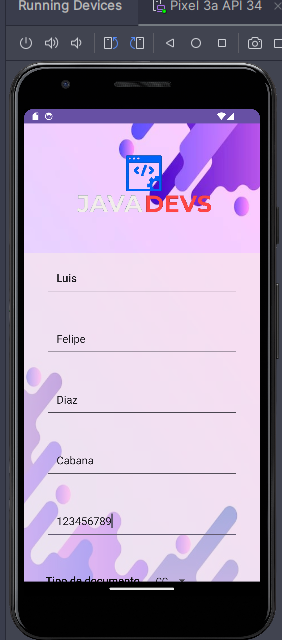
**Funcionamiento de la aplicación**



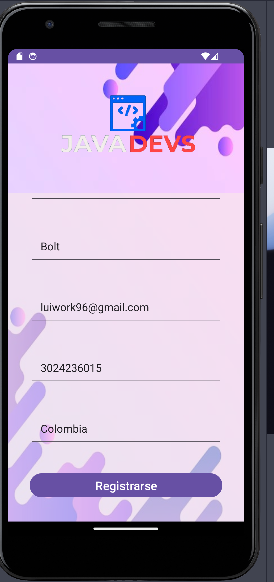
Presionamos en el boton de registrarse

Llenamos los campos con los datos que indican las pistas





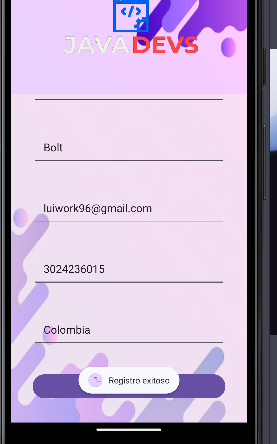
Una ves digitado todos los campos procedemos a presionar el boton de registrarse.



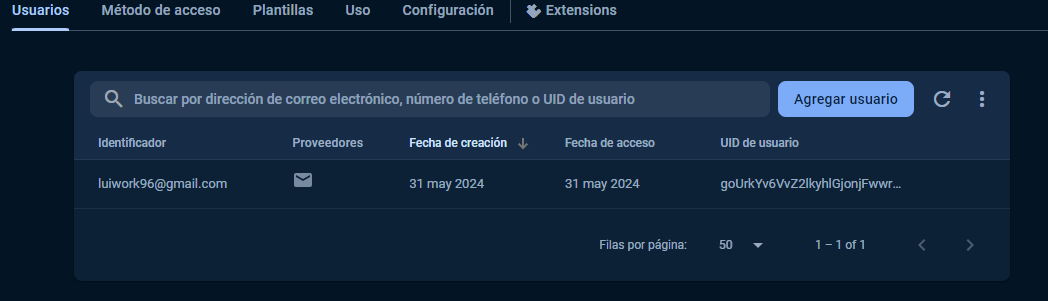
Si al presionar el boton de registrarse falta un campo por llenar, la aplicación lo notifica



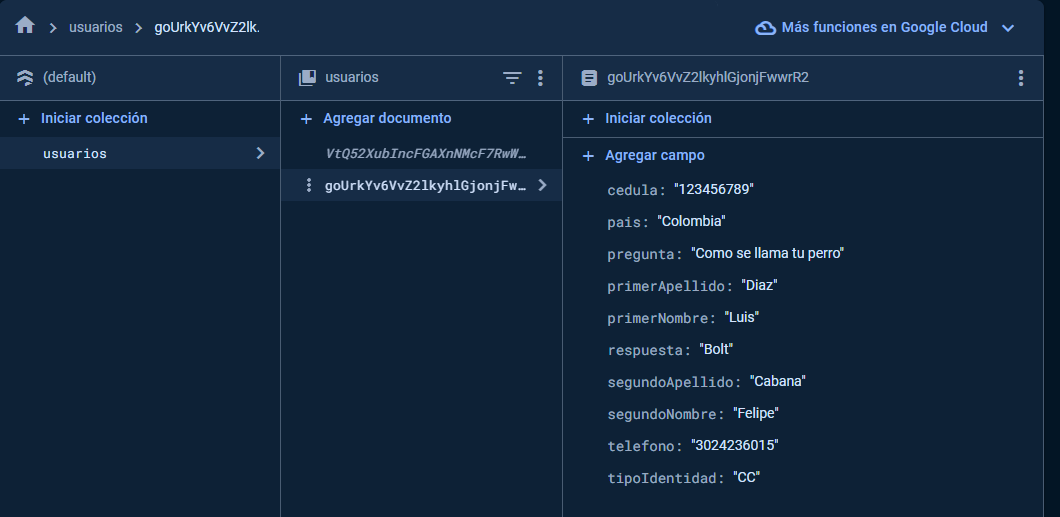
Una ves presionado el boton, la aplicación informa que el registro fue exitoso y lo lleva a la pantalla de login.



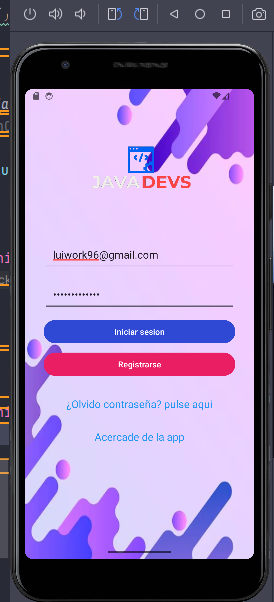
Se crea el usuario en el servicio de Authentication de firebase

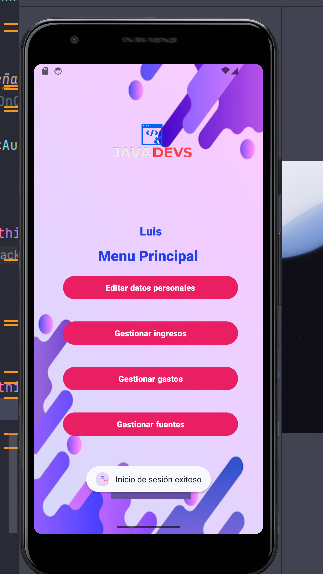


Y sus datos en la base de datos firestore

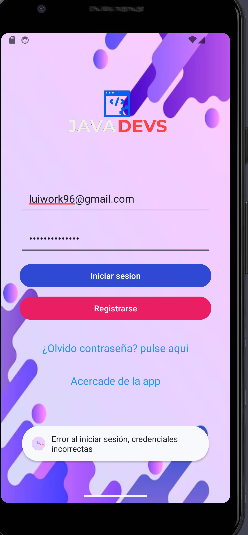


Ahora procederemos a iniciar sesion utilizando solamente el correo y la contraseña registrados previamente. Si es correcto los datos ingresados, nos llevara al menu principal de la aplicación y notificara que el inicio de sesion fue correcto.

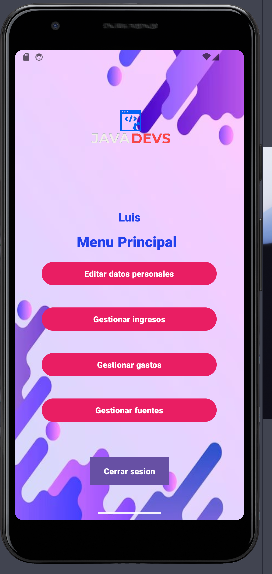




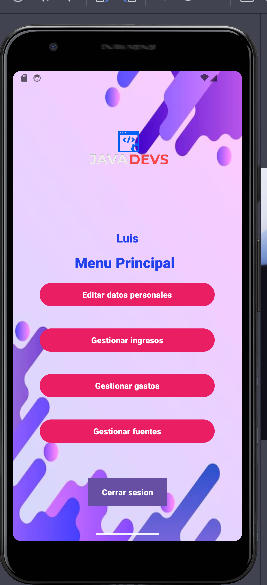
Si colocamos erronamente el correo o la contraseña, la app lo notificara y no permitira el ingreso.



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.

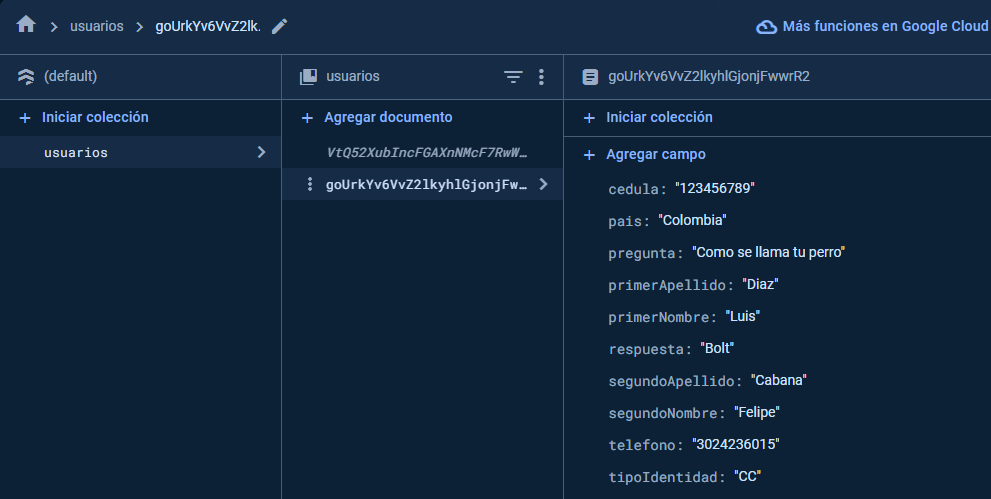


Una ves que hayamos modificado algun campo y la contraseña actual, solo presionamos el boton para actualizar y se actualizaran los datos.

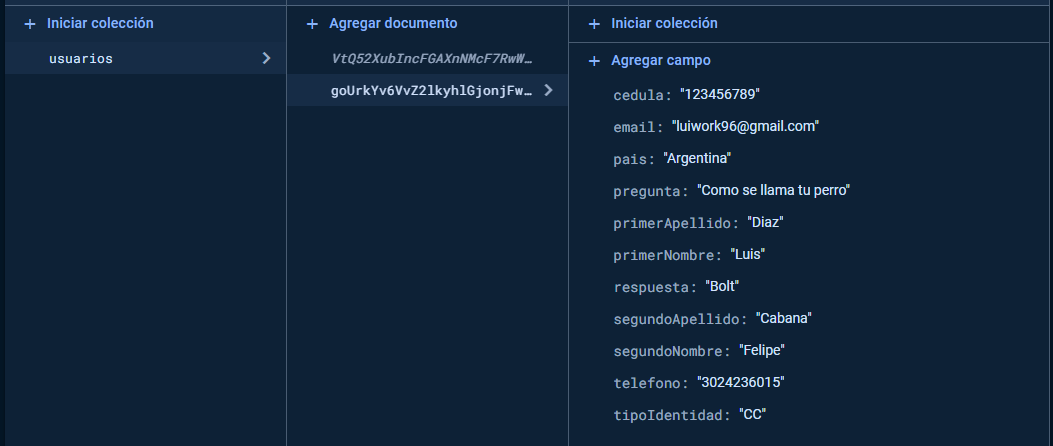




Antes de actualizar el campo



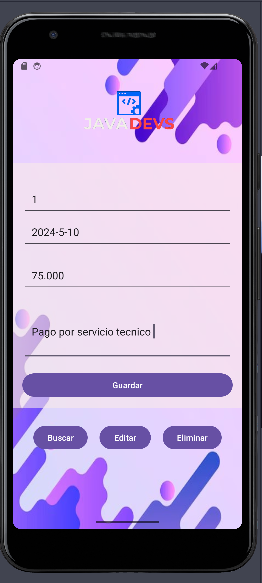
Despues de actualizar el campo y presionar el boton de actualizar



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

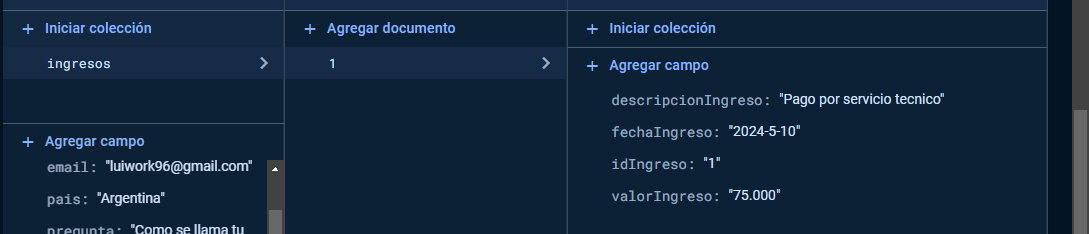


Llenamos los campos



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





Ahora busquemos el ingreso 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.



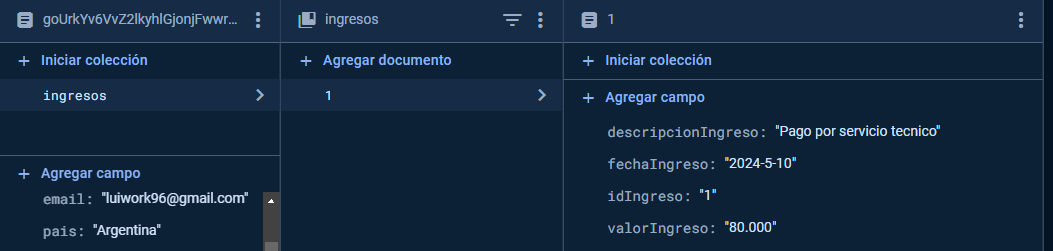
El resultado es



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







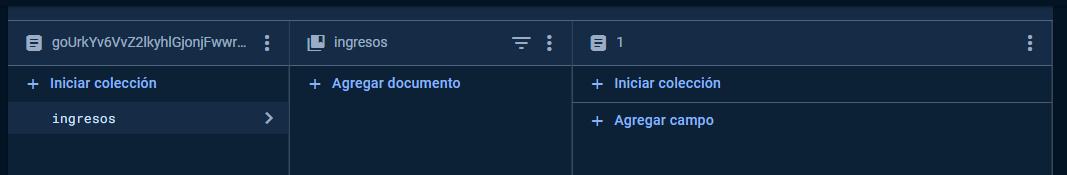
Y por ultimo probemos eliminando el ingreso



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>

Codelia [@codelia\_tutorials]. (2020, julio 22). Android Studio - Integrar Firebase Authentication / Crear usuarios Pt. 4 | Diseño Material 2020. Youtube. <https://www.youtube.com/watch?v=CTKGI-Cf_NA&list=PLaFe1-DmNGoLuf0UKq32vnYkSpJkrEgN1&index=5>

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>