**IrrigacaoService**

import android.app.Notification;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.content.Context;  
import android.graphics.Color;  
import androidx.core.app.NotificationCompat;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
public class IrrigacaoService extends android.app.Service {  
  
 private DatabaseReference mDatabase; // Referência ao banco de dados do Firebase  
  
 // Método chamado quando o serviço é criado  
 @Override  
 public void onCreate() {  
 super.onCreate();  
  
 // Inicializa a referência para o nó 'status' no Firebase Realtime Database  
 mDatabase = FirebaseDatabase.*getInstance*("https://situt-258258-default-rtdb.firebaseio.com/")  
 .getReference("status"); // Caminho para o nó 'status'  
  
 // Adiciona um listener para monitorar as mudanças na variável "Irrigando"  
 mDatabase.child("Irrigando").addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
 // Obtém o valor da variável "Irrigando" como String  
 String irrigandoStatus = dataSnapshot.getValue(String.class);  
  
 // Se o valor de "Irrigando" for "Sim", exibe uma notificação  
 if ("Sim".equalsIgnoreCase(irrigandoStatus)) {  
 mostrarNotificacao(); // Chama o método para exibir a notificação  
 }  
 }  
  
 @Override  
 public void onCancelled(DatabaseError databaseError) {  
 // Em caso de erro na leitura do Firebase, não faz nada  
 }  
 });  
 }  
  
 // Método para exibir a notificação  
 private void mostrarNotificacao() {  
 String channelId = "irrigacao\_channel"; // ID do canal de notificação  
 NotificationManager notificationManager = (NotificationManager) getSystemService(Context.*NOTIFICATION\_SERVICE*);  
  
 // Cria um canal de notificação para Android 8.0 ou superior (necessário)  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*O*) {  
 CharSequence name = "Canal Irrigação"; // Nome do canal  
 String description = "Notificações sobre o status da irrigação"; // Descrição do canal  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*; // Importância da notificação  
 NotificationChannel channel = new NotificationChannel(channelId, name, importance);  
 channel.setDescription(description);  
  
 // Define a cor do canal (Cobalt Blue)  
 channel.setLightColor(Color.*parseColor*("#0047AB"));  
  
 // Cria o canal de notificação no sistema  
 notificationManager.createNotificationChannel(channel);  
 }  
  
 // Criação da notificação com a cor personalizada  
 Notification notification = new NotificationCompat.Builder(this, channelId)  
 .setContentTitle("ATENÇÃO!!!") // Título da notificação  
 .setContentText("Sua planta está recebendo irrigação no momento.") // Texto da notificação  
 .setSmallIcon(R.drawable.*logoclara*) // Ícone da notificação  
 .setColor(Color.*parseColor*("#0047AB")) // Cor da notificação (Cobalt Blue)  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*) // Prioridade da notificação  
 .build();  
  
 // Exibe a notificação  
 notificationManager.notify(1, notification); // Exibe a notificação com ID 1  
 }  
  
 // Método chamado quando o serviço é iniciado  
 @Override  
 public int onStartCommand(android.content.Intent intent, int flags, int startId) {  
 return *START\_STICKY*; // Mantém o serviço em execução (recomeça automaticamente se o serviço for interrompido)  
 }  
  
 // Método necessário para serviços, mas não utilizado neste caso  
 @Override  
 public android.os.IBinder onBind(android.content.Intent intent) {  
 return null; // Este serviço não permite vinculação  
 }  
}

**IrrigacaoWorker**

import android.app.Notification;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.content.Context;  
import android.graphics.Color;  
import androidx.core.app.NotificationCompat;  
import androidx.work.Worker;  
import androidx.work.WorkerParameters;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
  
public class IrrigacaoWorker extends Worker {  
  
 // Construtor da classe, chamado quando o worker é instanciado  
 public IrrigacaoWorker(Context context, WorkerParameters workerParams) {  
 super(context, workerParams);  
 }  
  
 // Método principal do worker, que será executado em segundo plano  
 @Override  
 public Result doWork() {  
 // Inicializa a referência do Firebase para o nó "status"  
 DatabaseReference mDatabase = FirebaseDatabase.*getInstance*("https://situt-258258-default-rtdb.firebaseio.com/")  
 .getReference("status"); // Caminho para o nó 'status'  
  
 // Adiciona um listener para monitorar a variável "Irrigando" no Firebase  
 mDatabase.child("Irrigando").addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
 // Obtém o valor da variável "Irrigando" como String  
 String irrigandoStatus = dataSnapshot.getValue(String.class);  
  
 // Se o status for "Sim", exibe a notificação  
 if ("Sim".equalsIgnoreCase(irrigandoStatus)) {  
 mostrarNotificacao(); // Chama o método para mostrar a notificação  
 }  
 }  
  
 @Override  
 public void onCancelled(DatabaseError databaseError) {  
 // Se ocorrer um erro na leitura do Firebase, não faz nada aqui  
 }  
 });  
  
 // Retorna sucesso após a execução da tarefa  
 return Result.*success*();  
 }  
  
 // Método para exibir a notificação quando a irrigação está em andamento  
 private void mostrarNotificacao() {  
 String channelId = "irrigacao\_channel"; // ID do canal de notificação  
 NotificationManager notificationManager = (NotificationManager) getApplicationContext().getSystemService(Context.*NOTIFICATION\_SERVICE*);  
  
 // Cria um canal de notificação, necessário para Android 8.0 ou superior  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*O*) {  
 CharSequence name = "Canal Irrigação"; // Nome do canal  
 String description = "Notificações sobre o status da irrigação"; // Descrição do canal  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*; // Importância da notificação  
 NotificationChannel channel = new NotificationChannel(channelId, name, importance);  
 channel.setDescription(description);  
  
 // Define a cor do canal (utiliza a cor Cobalt Blue)  
 channel.setLightColor(Color.*parseColor*("#0047AB")); // Cobalt Blue  
  
 // Cria o canal de notificação no sistema  
 notificationManager.createNotificationChannel(channel);  
 }  
  
 // Criação da notificação com a cor personalizada  
 Notification notification = new NotificationCompat.Builder(getApplicationContext(), channelId)  
 .setContentTitle("ATENÇÃO!!!") // Título da notificação  
 .setContentText("Sua planta está recebendo irrigação no momento.") // Texto da notificação  
 .setSmallIcon(R.drawable.*logoclara*) // Ícone da notificação  
 .setColor(Color.*parseColor*("#0047AB")) // Cor da notificação (Cobalt Blue)  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*) // Prioridade da notificação  
 .build();  
  
 // Exibe a notificação  
 notificationManager.notify(1, notification); // Notificação com ID 1  
 }  
}

**NotificaMain**

import android.app.Notification;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.content.Context;  
import android.graphics.Color;  
import androidx.core.app.NotificationCompat;  
import com.google.firebase.messaging.FirebaseMessagingService;  
import com.google.firebase.messaging.RemoteMessage;  
  
public class MyFirebaseMessagingService extends FirebaseMessagingService {  
  
 @Override  
 public void onMessageReceived(RemoteMessage remoteMessage) {  
 super.onMessageReceived(remoteMessage);  
  
 // Verifica se a mensagem recebida contém dados  
 if (remoteMessage.getData().size() > 0) {  
 // Extrai o valor da chave "Irrigando" dos dados da mensagem  
 String irrigandoStatus = remoteMessage.getData().get("Irrigando");  
  
 // Se o valor de "Irrigando" for "Sim", chama o método para mostrar a notificação  
 if ("Sim".equalsIgnoreCase(irrigandoStatus)) {  
 mostrarNotificacao(); // Chama o método que exibe a notificação  
 }  
 }  
 }  
  
 // Método para exibir a notificação  
 private void mostrarNotificacao() {  
 String channelId = "irrigacao\_channel"; // ID único do canal de notificação  
 NotificationManager notificationManager = (NotificationManager) getSystemService(Context.*NOTIFICATION\_SERVICE*); // Gerenciador de notificações  
  
 // Criação do canal de notificação, necessário para Android 8.0 (API nível 26) ou superior  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*O*) {  
 CharSequence name = "Canal Irrigação"; // Nome do canal  
 String description = "Notificações sobre o status da irrigação"; // Descrição do canal  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*; // Importância da notificação  
 NotificationChannel channel = new NotificationChannel(channelId, name, importance); // Cria o canal de notificação  
 channel.setDescription(description); // Define a descrição do canal  
  
 // Definindo a cor do canal de notificação (para efeitos de luz no dispositivo)  
 channel.setLightColor(Color.*parseColor*("#0047AB")); // Cor personalizada (Cobalt Blue)  
  
 notificationManager.createNotificationChannel(channel); // Cria o canal de notificações  
 }  
  
 // Criação da notificação com uma cor personalizada e ícone  
 Notification notification = new NotificationCompat.Builder(this, channelId)  
 .setContentTitle("ATENÇÃO!!!") // Título da notificação  
 .setContentText("Sua planta está recebendo irrigação no momento.") // Texto da notificação  
 .setSmallIcon(R.drawable.*logoclara*) // Ícone da notificação (personalizado)  
 .setColor(Color.*parseColor*("#0047AB")) // Cor personalizada (Cobalt Blue) para a notificação  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*) // Prioridade padrão da notificação  
 .build(); // Constrói a notificação  
  
 // Exibe a notificação  
 notificationManager.notify(1, notification); // ID único para exibir a notificação  
 }

**MyFirebaseMessagingService**  
  
import android.app.Notification;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.content.Context;  
import android.graphics.Color;  
import androidx.core.app.NotificationCompat;  
import com.google.firebase.messaging.FirebaseMessagingService;  
import com.google.firebase.messaging.RemoteMessage;  
  
public class MyFirebaseMessagingService extends FirebaseMessagingService {  
  
 @Override  
 public void onMessageReceived(RemoteMessage remoteMessage) {  
 super.onMessageReceived(remoteMessage);  
  
 // Verifica se a mensagem recebida contém dados  
 if (remoteMessage.getData().size() > 0) {  
 // Extrai o valor da chave "Irrigando" dos dados da mensagem  
 String irrigandoStatus = remoteMessage.getData().get("Irrigando");  
  
 // Se o valor de "Irrigando" for "Sim", chama o método para mostrar a notificação  
 if ("Sim".equalsIgnoreCase(irrigandoStatus)) {  
 mostrarNotificacao(); // Chama o método que exibe a notificação  
 }  
 }  
 }  
  
 // Método para exibir a notificação  
 private void mostrarNotificacao() {  
 String channelId = "irrigacao\_channel"; // ID único do canal de notificação  
 NotificationManager notificationManager = (NotificationManager) getSystemService(Context.*NOTIFICATION\_SERVICE*); // Gerenciador de notificações  
  
 // Criação do canal de notificação, necessário para Android 8.0 (API nível 26) ou superior  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*O*) {  
 CharSequence name = "Canal Irrigação"; // Nome do canal  
 String description = "Notificações sobre o status da irrigação"; // Descrição do canal  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*; // Importância da notificação  
 NotificationChannel channel = new NotificationChannel(channelId, name, importance); // Cria o canal de notificação  
 channel.setDescription(description); // Define a descrição do canal  
  
 // Definindo a cor do canal de notificação (para efeitos de luz no dispositivo)  
 channel.setLightColor(Color.*parseColor*("#0047AB")); // Cor personalizada (Cobalt Blue)  
  
 notificationManager.createNotificationChannel(channel); // Cria o canal de notificações  
 }  
  
 // Criação da notificação com uma cor personalizada e ícone  
 Notification notification = new NotificationCompat.Builder(this, channelId)  
 .setContentTitle("ATENÇÃO!!!") // Título da notificação  
 .setContentText("Sua planta está recebendo irrigação no momento.") // Texto da notificação  
 .setSmallIcon(R.drawable.*logoclara*) // Ícone da notificação (personalizado)  
 .setColor(Color.*parseColor*("#0047AB")) // Cor personalizada (Cobalt Blue) para a notificação  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*) // Prioridade padrão da notificação  
 .build(); // Constrói a notificação  
  
 // Exibe a notificação  
 notificationManager.notify(1, notification); // ID único para exibir a notificação  
 }  
}

**SalvarDadosWorker**

import android.content.Context;  
import android.util.Log;  
import androidx.annotation.NonNull;  
import androidx.work.Worker;  
import androidx.work.WorkerParameters;  
import com.google.firebase.FirebaseApp;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Map;

public class SalvarDadosWorker extends Worker {

// Tag para uso nos logs de depuração

private static final String TAG = "SalvarDadosWorker";

// Construtor que recebe o contexto e os parâmetros do worker

public SalvarDadosWorker(@NonNull Context context, @NonNull WorkerParameters workerParams) {

super(context, workerParams);

}

// Método principal que executa o trabalho do worker

@NonNull

@Override

public Result doWork() {

// Inicializa o Firebase no contexto atual

FirebaseApp.initializeApp(getApplicationContext());

// Referências para as pastas do Firebase Realtime Database: 'sensores' e 'Historico'

FirebaseDatabase database = FirebaseDatabase.getInstance();

DatabaseReference ref = database.getReference("sensores");

DatabaseReference historicoRef = ref.getParent().child("Historico");

// Lê os dados da pasta 'sensores' no Firebase

ref.addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot snapshot) {

if (snapshot.exists()) {

// Cria um mapa para armazenar os dados do histórico

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

for (DataSnapshot data : snapshot.getChildren()) {

// Coloca cada item lido na estrutura de dados 'historicoData'

historicoData.put(data.getKey(), data.getValue());

}

// Adiciona a data e hora atuais ao histórico como "dataExpedicao"

String currentDate = getCurrentDate();

historicoData.put("dataExpedicao", currentDate);

// Verifica quantos dados existem na pasta 'Historico'

historicoRef.addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot historicoSnapshot) {

int size = (int) historicoSnapshot.getChildrenCount(); // Conta o total de registros

if (size >= 10) {

// Se houver 10 ou mais registros, remove o mais antigo

String keyToRemove = null;

for (DataSnapshot data : historicoSnapshot.getChildren()) {

keyToRemove = data.getKey(); // Pega a chave do primeiro item

break; // Sai após pegar o primeiro

}

if (keyToRemove != null) {

// Remove o dado mais antigo

historicoRef.child(keyToRemove).removeValue()

.addOnSuccessListener(aVoid -> Log.d(TAG, "Dado mais antigo removido com sucesso!"))

.addOnFailureListener(e -> Log.e(TAG, "Erro ao remover dado: " + e.getMessage()));

}

}

// Salva os novos dados na pasta 'Historico'

historicoRef.push().setValue(historicoData)

.addOnSuccessListener(aVoid -> Log.d(TAG, "Dados salvos com sucesso!"))

.addOnFailureListener(e -> Log.e(TAG, "Erro ao salvar os dados: " + e.getMessage()));

}

@Override

public void onCancelled(DatabaseError error) {

// Loga erro se ocorrer falha na leitura dos dados de histórico

Log.e(TAG, "Erro ao ler os dados do histórico: " + error.getMessage());

}

});

} else {

// Loga erro se não existir dados na pasta 'sensores'

Log.e(TAG, "Dados não encontrados na pasta sensores");

}

}

@Override

public void onCancelled(DatabaseError error) {

// Loga erro caso ocorra falha na leitura dos dados da pasta 'sensores'

Log.e(TAG, "Erro ao ler os dados: " + error.getMessage());

}

});

// Retorna um resultado de sucesso para o Worker

return Result.success();

}

// Método para obter a data e hora atual formatada

private String getCurrentDate() {

// Define o formato da data e hora

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

// Retorna a data no formato "ano-mês-dia hora:minuto:segundo"

return sdf.format(new Date());

}

}

**SalvarDadosActivity**

import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import androidx.work.PeriodicWorkRequest;  
import androidx.work.WorkManager;  
import androidx.work.WorkRequest;  
import java.util.concurrent.TimeUnit;

public class SalvarDadosActivity extends AppCompatActivity {

// Tag para logs de depuração da atividade

private static final String TAG = "SalvarDadosActivity";

// Método chamado ao criar a atividade

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// Define o layout da atividade para 'activity\_form\_home'

setContentView(R.layout.activity\_form\_home);

// Agendar o trabalho para salvar os dados a cada 5 minutos

agendarSalvarDados();

}

// Método para agendar a tarefa de salvar os dados periodicamente

private void agendarSalvarDados() {

// Cria uma solicitação periódica para salvar os dados

WorkRequest salvarDadosRequest = new PeriodicWorkRequest.Builder(

SalvarDadosWorker.class, // Classe Worker que irá realizar o trabalho

5, TimeUnit.MINUTES) // Define a periodicidade: executa a cada 5 minutos

.setInitialDelay(1, TimeUnit.MINUTES) // Espera 1 minuto antes da primeira execução

.build();

// Agenda o trabalho usando o WorkManager

WorkManager.getInstance(this).enqueue(salvarDadosRequest);

// Log para informar que a tarefa foi agendada

Log.d(TAG, "Tarefa agendada para salvar os dados periodicamente.");

}

}

**FormLogin**

import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageButton;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.appcompat.app.AppCompatActivity;  
import com.google.android.gms.auth.api.signin.GoogleSignIn;  
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;  
import com.google.android.gms.auth.api.signin.GoogleSignInClient;  
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;  
import com.google.android.gms.tasks.Task;  
import com.google.firebase.auth.AuthCredential;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.auth.GoogleAuthProvider;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import java.util.HashMap;  
import java.util.Map;

public class FormLogin extends AppCompatActivity {

private TextView cadastrar; // Elemento de texto para a opção de cadastro

private Button btnEntrar; // Botão para login com e-mail e senha

private ImageButton btnLoginGoogle; // Botão para login com Google

private EditText editEmail, editSenha; // Campos de entrada de e-mail e senha

private FirebaseAuth mAuth; // Instância do FirebaseAuth para autenticação

private GoogleSignInClient mGoogleSignInClient; // Cliente para o login com Google

private static final int RC\_SIGN\_IN = 9001; // Código de requisição para login com Google

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_logar\_google); // Carrega o layout da activity

// Inicializa o FirebaseAuth

mAuth = FirebaseAuth.getInstance();

// Configura o Google Sign-In

GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT\_SIGN\_IN)

.requestIdToken(getString(R.string.default\_web\_client\_id)) // Solicita o ID do cliente do Firebase

.requestEmail() // Solicita o e-mail

.build();

mGoogleSignInClient = GoogleSignIn.getClient(this, gso); // Inicializa o cliente de login Google

// Associa as variáveis aos campos do layout XML

editEmail = findViewById(R.id.editEmail);

editSenha = findViewById(R.id.editSenha);

cadastrar = findViewById(R.id.Cadastro);

// Configura o botão de login com e-mail e senha

btnEntrar = findViewById(R.id.btnEntrar);

btnEntrar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

loginWithEmail(); // Chama o método para login com e-mail e senha

}

});

// Configura o listener para abrir a tela de cadastro

cadastrar.setOnClickListener(view -> {

Intent intent = new Intent(FormLogin.this, FormCadastro.class);

startActivity(intent); // Inicia a tela de cadastro

});

// Configura o botão de login com Google

btnLoginGoogle = findViewById(R.id.btnLoginGoogle);

btnLoginGoogle.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

signInWithGoogle(); // Chama o método para login com Google

}

});

}

// Método para realizar login com e-mail e senha

private void loginWithEmail() {

String email = editEmail.getText().toString().trim(); // Obtém o e-mail digitado

String senha = editSenha.getText().toString().trim(); // Obtém a senha digitada

// Verifica se os campos de e-mail e senha não estão vazios

if (email.isEmpty()) {

editEmail.setError("Email é obrigatório!"); // Exibe um erro se o e-mail estiver vazio

return;

}

if (senha.isEmpty()) {

editSenha.setError("Senha é obrigatória!"); // Exibe um erro se a senha estiver vazia

return;

}

// Verifica se o e-mail termina com o domínio permitido

String allowedDomain = "@gmail.com"; // Domínio permitido para login

if (!email.endsWith(allowedDomain)) {

editEmail.setError("Apenas e-mails " + allowedDomain + " são permitidos!"); // Exibe um erro se o domínio não for permitido

return;

}

// Realiza a autenticação com Firebase usando e-mail e senha

mAuth.signInWithEmailAndPassword(email, senha)

.addOnCompleteListener(this, task -> {

if (task.isSuccessful()) {

// Se o login for bem-sucedido

FirebaseUser user = mAuth.getCurrentUser();

Toast.makeText(FormLogin.this, "Login bem-sucedido!", Toast.LENGTH\_SHORT).show();

// Redireciona para a tela de primeiro acesso

startActivity(new Intent(FormLogin.this, primeiro\_acesso.class));

// Finaliza a FormLogin para evitar voltar à tela de login

finish();

} else {

// Se o login falhar, exibe o erro

String errorMessage = task.getException() != null ? task.getException().getMessage() : "Erro desconhecido";

Toast.makeText(FormLogin.this, "Falha ao fazer login: " + errorMessage, Toast.LENGTH\_SHORT).show();

}

});

}

@Override

protected void onStart() {

super.onStart();

// Verifica se o usuário já está logado

FirebaseUser usuarioAtual = FirebaseAuth.getInstance().getCurrentUser();

if(usuarioAtual != null) {

TelaPrincipal(); // Se logado, redireciona para a tela principal

}

}

// Método para redirecionar para a tela principal

private void TelaPrincipal(){

Intent intent = new Intent(FormLogin.this, Form\_home.class);

startActivity(intent); // Inicia a tela principal

}

// Método para login com Google

private void signInWithGoogle() {

Intent signInIntent = mGoogleSignInClient.getSignInIntent();

startActivityForResult(signInIntent, RC\_SIGN\_IN); // Inicia o fluxo de login com Google

}

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

// Verifica se o requestCode é do login do Google

if (requestCode == RC\_SIGN\_IN) {

Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);

handleSignInResult(task); // Trata o resultado do login

}

}

// Método para lidar com o resultado do login do Google

private void handleSignInResult(Task<GoogleSignInAccount> completedTask) {

try {

// Se o login do Google foi bem-sucedido, autentica com Firebase

GoogleSignInAccount account = completedTask.getResult();

firebaseAuthWithGoogle(account);

} catch (Exception e) {

// Se falhar, exibe uma mensagem de erro

String errorMessage = e.getMessage() != null ? e.getMessage() : "Erro desconhecido";

Toast.makeText(this, "Erro ao autenticar com Google: " + errorMessage, Toast.LENGTH\_SHORT).show();

}

}

// Método para autenticar no Firebase usando a conta do Google

private void firebaseAuthWithGoogle(GoogleSignInAccount acct) {

AuthCredential credential = GoogleAuthProvider.getCredential(acct.getIdToken(), null);

// Realiza o login no Firebase

mAuth.signInWithCredential(credential)

.addOnCompleteListener(this, task -> {

if (task.isSuccessful()) {

// Se o login for bem-sucedido

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

// Salva nome e e-mail no Firebase Database

salvarDadosUsuario(user.getDisplayName(), user.getEmail());

}

Toast.makeText(FormLogin.this, "Login com Google bem-sucedido!", Toast.LENGTH\_SHORT).show();

// Redireciona para a tela de primeiro acesso

startActivity(new Intent(FormLogin.this, primeiro\_acesso.class));

finish(); // Finaliza a tela de login

} else {

// Se falhar, exibe mensagem de erro

String errorMessage = task.getException() != null ? task.getException().getMessage() : "Erro desconhecido";

Toast.makeText(FormLogin.this, "Falha ao fazer login com Google: " + errorMessage, Toast.LENGTH\_SHORT).show();

}

});

}

// Método para salvar o nome e e-mail do usuário no Firebase Realtime Database

private void salvarDadosUsuario(String nome, String email) {

DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference("Users");

String userId = mAuth.getCurrentUser().getUid(); // Obtém o UID do usuário

// Cria um mapa para salvar nome e e-mail do usuário

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

dadosUsuario.put("nome", nome);

dadosUsuario.put("email", email);

// Salva os dados no Firebase Database

mDatabase.child(userId).setValue(dadosUsuario)

.addOnSuccessListener(aVoid -> Toast.makeText(FormLogin.this, "Dados do usuário salvos!", Toast.LENGTH\_SHORT).show())

.addOnFailureListener(e -> Toast.makeText(FormLogin.this, "Erro ao salvar dados do usuário.", Toast.LENGTH\_SHORT).show());

}

}

**FormCadastro**

import android.content.Intent;  
import android.graphics.Color;  
import android.os.Bundle;  
import android.os.Handler;  
import android.os.Looper;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
import androidx.appcompat.app.AppCompatActivity;  
import com.google.android.material.snackbar.Snackbar;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseAuthInvalidCredentialsException;  
import com.google.firebase.auth.FirebaseAuthUserCollisionException;  
import com.google.firebase.auth.FirebaseAuthWeakPasswordException;  
import com.google.firebase.firestore.FirebaseFirestore;  
import java.util.HashMap;  
import java.util.Map;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.auth.UserProfileChangeRequest;

public class FormCadastro extends AppCompatActivity {

private EditText edit\_nome, edit\_email, edit\_senha; // Campos de entrada de nome, email e senha

private Button btn\_cad; // Botão de cadastro

private FirebaseAuth mAuth; // Autenticação Firebase

String[] mensagem = {"Preencha Todos os Campos", "Cadastro Realizado com Sucesso", "Verifique seu Email"}; // Mensagens para exibir ao usuário

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_form\_cadastro); // Define o layout para a activity de cadastro

mAuth = FirebaseAuth.getInstance(); // Inicializa a autenticação Firebase

IniciarComponentes(); // Método que associa os componentes XML aos objetos da classe

// Configura o evento de clique para o botão de cadastro

btn\_cad.setOnClickListener(v -> {

String nome = edit\_nome.getText().toString(); // Obtém o texto do campo de nome

String email = edit\_email.getText().toString(); // Obtém o texto do campo de email

String senha = edit\_senha.getText().toString(); // Obtém o texto do campo de senha

// Verifica se os campos estão vazios

if (nome.isEmpty() || email.isEmpty() || senha.isEmpty()) {

// Exibe uma mensagem de erro caso algum campo esteja vazio

Snackbar snackbar = Snackbar.make(v, mensagem[0], Snackbar.LENGTH\_SHORT);

snackbar.setBackgroundTint(Color.WHITE); // Cor de fundo branca

snackbar.setTextColor(Color.BLACK); // Cor do texto preta

snackbar.show();

} else {

// Aguarda 2 segundos e chama o método para cadastrar o usuário

new Handler(Looper.getMainLooper()).postDelayed(() -> {

CadastrarUsuario(v);

startActivity(new Intent(FormCadastro.this, FormLogin.class)); // Navega para a tela de login

finish(); // Finaliza a tela de cadastro para que o usuário não volte a ela

}, 2000);

}

});

// Configura o evento de clique para recuperação de senha

findViewById(R.id.esqueceu\_senha).setOnClickListener(v -> {

String email = edit\_email.getText().toString().trim(); // Obtém o email para recuperação

if (email.isEmpty()) {

// Exibe mensagem caso o campo de email esteja vazio

Snackbar snackbar = Snackbar.make(v, "Informe seu e-mail para recuperação de senha.", Snackbar.LENGTH\_SHORT);

snackbar.setBackgroundTint(Color.WHITE);

snackbar.setTextColor(Color.BLACK);

snackbar.show();

} else {

// Chama o método para enviar email de recuperação de senha

enviarEmailRecuperacao(email);

}

});

}

// Método para cadastrar o usuário

private void CadastrarUsuario(View v) {

String nome = edit\_nome.getText().toString(); // Obtém o nome

String email = edit\_email.getText().toString(); // Obtém o email

String senha = edit\_senha.getText().toString(); // Obtém a senha

String allowedDomain = "@gmail.com"; // Domínio permitido para cadastro

// Verifica se o email possui o domínio permitido

if (!email.endsWith(allowedDomain)) {

// Exibe erro caso o domínio do email seja diferente do permitido

Snackbar.make(v, "Apenas e-mails " + allowedDomain + " são permitidos!", Snackbar.LENGTH\_SHORT)

.setBackgroundTint(Color.WHITE)

.setTextColor(Color.BLACK).show();

return;

}

// Cria o usuário com o email e senha no Firebase

mAuth.createUserWithEmailAndPassword(email, senha)

.addOnCompleteListener(task -> {

if (task.isSuccessful()) {

FirebaseUser user = mAuth.getCurrentUser(); // Obtém o usuário atual

if (user != null) {

// Atualiza o perfil do usuário com o nome fornecido

UserProfileChangeRequest profileUpdates = new UserProfileChangeRequest.Builder()

.setDisplayName(nome)

.build();

user.updateProfile(profileUpdates)

.addOnCompleteListener(profileTask -> {

if (profileTask.isSuccessful()) {

Log.d("Firebase", "Nome de exibição atualizado.");

}

});

}

// Envia email de verificação para o usuário

mAuth.getCurrentUser().sendEmailVerification().addOnCompleteListener(verifyTask -> {

if (verifyTask.isSuccessful()) {

// Salva os dados do usuário no Firebase Database

SalvarDadosUsuario();

// Exibe mensagem de sucesso

Snackbar.make(v, mensagem[1], Snackbar.LENGTH\_SHORT)

.setBackgroundTint(Color.WHITE)

.setTextColor(Color.BLACK).show();

} else {

// Exibe mensagem para verificar o email

Snackbar.make(v, mensagem[2], Snackbar.LENGTH\_SHORT)

.setBackgroundTint(Color.WHITE)

.setTextColor(Color.BLACK).show();

}

});

} else {

// Trata erros no cadastro, como senha fraca ou email duplicado

tratarErroCadastro(task.getException(), v);

}

});

}

// Método para enviar email de recuperação de senha

private void enviarEmailRecuperacao(String email) {

mAuth.sendPasswordResetEmail(email)

.addOnCompleteListener(task -> {

if (task.isSuccessful()) {

// Exibe confirmação de envio de email

Toast.makeText(FormCadastro.this, "E-mail de recuperação enviado!", Toast.LENGTH\_SHORT).show();

} else {

// Exibe mensagem de erro em caso de falha

Toast.makeText(FormCadastro.this, "Erro ao enviar e-mail de recuperação: " + task.getException().getMessage(), Toast.LENGTH\_LONG).show();

}

});

}

// Método para salvar os dados do usuário no Firestore

private void SalvarDadosUsuario() {

String nome = edit\_nome.getText().toString(); // Obtém o nome do usuário

FirebaseFirestore db = FirebaseFirestore.getInstance(); // Instância do Firestore

Map<String, Object> usuarios = new HashMap<>(); // Mapa para armazenar dados do usuário

usuarios.put("nome", nome); // Adiciona o nome ao mapa

String usuarioID = mAuth.getCurrentUser().getUid(); // Obtém o UID do usuário

db.collection("Usuarios").document(usuarioID).set(usuarios) // Salva no Firestore

.addOnSuccessListener(unused -> Log.d("db", "Sucesso ao Salvar os Dados"))

.addOnFailureListener(e -> Log.d("db\_error", "Erro ao Salvar os Dados" + e.toString()));

}

// Método para tratar erros no cadastro

private void tratarErroCadastro(Exception exception, View v) {

String erro;

try {

throw exception;

} catch (FirebaseAuthWeakPasswordException e) {

erro = "A senha deve ter no mínimo 6 caracteres";

} catch (FirebaseAuthUserCollisionException e) {

erro = "Email já Cadastrado";

} catch (FirebaseAuthInvalidCredentialsException e) {

erro = "Email Inválido";

} catch (Exception e) {

erro = "Erro ao Cadastrar Usuário";

}

Snackbar.make(v, erro, Snackbar.LENGTH\_SHORT)

.setBackgroundTint(Color.WHITE)

.setTextColor(Color.BLACK).show(); // Exibe erro específico

}

// Método para associar os componentes do layout aos objetos da classe

private void IniciarComponentes() {

edit\_nome = findViewById(R.id.editName);

edit\_email = findViewById(R.id.editEmail);

edit\_senha = findViewById(R.id.editSenha);

btn\_cad = findViewById(R.id.btnCadastrar);

}

}

**Primeiro\_acesso**

import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.graphics.Insets;  
import androidx.core.view.ViewCompat;  
import androidx.core.view.WindowInsetsCompat;

public class primeiro\_acesso extends AppCompatActivity {  
  
 private Button btn\_umavez, btn\_outra; // Declaração dos botões  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_primeiro\_acesso*); // Define o layout da activity  
  
 // Configura o comportamento de ajuste de padding quando as barras de status e navegação são detectadas  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (v, insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*()); // Obtém os insets das barras do sistema  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom); // Aplica os insets como padding  
 return insets; // Retorna os insets aplicados  
 });  
  
 // Inicializa os componentes da tela  
 IniciarComponentes();  
  
 // Configura o listener de clique para o botão "Primeira vez"  
 btn\_umavez.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 // Cria uma intent para abrir a atividade "IntroActivity"  
 Intent intent = new Intent(primeiro\_acesso.this, IntroActivity.class);  
 startActivity(intent); // Inicia a nova activity  
 }  
 });  
  
 // Configura o listener de clique para o botão "Outra vez"  
 btn\_outra.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 // Cria uma intent para abrir a atividade "Form\_home"  
 Intent intent = new Intent(primeiro\_acesso.this, Form\_home.class);  
 startActivity(intent); // Inicia a nova activity  
 }  
 });  
 }  
  
 // Método para inicializar os componentes da tela  
 private void IniciarComponentes() {  
 btn\_umavez = findViewById(R.id.*btnPrimeiravez*); // Inicializa o botão "Primeira vez"  
 btn\_outra = findViewById(R.id.*btnOutravez*); // Inicializa o botão "Outra vez"  
 }  
}

**Intro\_Activity**

import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.viewpager.widget.ViewPager;

public class IntroActivity extends AppCompatActivity {

// Declaração dos componentes ViewPager, adaptador, layouts e botão de avançar

private ViewPager viewPager;

private IntroPagerAdapter adapter;

private int[] layouts;

private Button btn\_passar;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_intro);

// Inicializando o array de layouts para cada slide no ViewPager

layouts = new int[]{

R.layout.intro\_slide\_0,

R.layout.intro\_slide\_1,

R.layout.intro\_slide\_2,

R.layout.intro\_slide\_3,

R.layout.intro\_slide\_4,

R.layout.intro\_slide\_5,

R.layout.intro\_slide\_6,

};

// Configurando o ViewPager e associando o adaptador personalizado a ele

viewPager = findViewById(R.id.viewPager);

adapter = new IntroPagerAdapter(this, layouts);

viewPager.setAdapter(adapter);

// Definindo uma animação de transição personalizada para as páginas do ViewPager

viewPager.setPageTransformer(true, new ZoomOutPageTransformer());

// Adiciona um listener para monitorar as mudanças de página no ViewPager

viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {

@Override

public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {

// Sem ação necessária ao deslizar entre páginas

}

@Override

public void onPageSelected(int position) {

// Atualiza os dots na interface conforme a posição da página atual

updateDots(position);

}

@Override

public void onPageScrollStateChanged(int state) {

// Sem ação necessária ao mudar o estado de rolagem

}

});

// Chama a função para atualizar os dots na primeira página ao iniciar

updateDots(0);

}

private void updateDots(int position) {

// Atualiza a cor dos dots (branco/preto) dependendo da página atual (posição)

findViewById(R.id.dot1).setBackgroundResource(position == 0 ? R.drawable.branco : R.drawable.preta);

findViewById(R.id.dot2).setBackgroundResource(position == 1 ? R.drawable.branco : R.drawable.preta);

findViewById(R.id.dot3).setBackgroundResource(position == 2 ? R.drawable.branco : R.drawable.preta);

findViewById(R.id.dot4).setBackgroundResource(position == 3 ? R.drawable.branco : R.drawable.preta);

findViewById(R.id.dot5).setBackgroundResource(position == 4 ? R.drawable.branco : R.drawable.preta);

findViewById(R.id.dot6).setBackgroundResource(position == 5 ? R.drawable.branco : R.drawable.preta);

findViewById(R.id.dot7).setBackgroundResource(position == 6 ? R.drawable.branco : R.drawable.preta);

// Verifica se estamos na última página (posição 6)

if (position == 6) {

btn\_passar = findViewById(R.id.btn\_avancar); // Obtém referência ao botão de avançar

if (btn\_passar != null) { // Confirma se o botão existe na interface

// Define o clique do botão para iniciar a Form\_registro\_sistema

btn\_passar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent(IntroActivity.this, Form\_registro\_sistema.class);

startActivity(intent);

}

});

}

}

}

}

**IntroPagerAdapter**

import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
import androidx.viewpager.widget.PagerAdapter;  
import com.bumptech.glide.Glide;  
  
public class IntroPagerAdapter extends PagerAdapter {  
  
 private Context context;  
 private int[] layouts; // Array de layouts (IDs de recursos) para os slides  
  
 // Construtor do adaptador que recebe o contexto e os layouts dos slides  
 public IntroPagerAdapter(Context context, int[] layouts) {  
 this.context = context;  
 this.layouts = layouts;  
 }  
  
 @Override  
 public Object instantiateItem(ViewGroup container, int position) {  
 // Infla o layout do slide usando o LayoutInflater  
 LayoutInflater layoutInflater = (LayoutInflater) context.getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);  
 View view = layoutInflater.inflate(layouts[position], container, false);  
  
 // Carrega o GIF correspondente para cada slide  
 if (position == 0) {  
 // Slide 1: Carregar o GIF para o primeiro slide  
 ImageView imageView = view.findViewById(R.id.*imagem\_situt2*); // Referência ao ImageView no layout  
 Glide.*with*(context)  
 .asGif() // Especifica que o tipo de recurso a ser carregado é um GIF  
 .load(R.drawable.*situtw*) // Substitua pelo nome do seu primeiro GIF  
 .into(imageView); // Carrega o GIF na ImageView  
 } else if (position == 1) {  
 // Slide 2: Carregar o GIF para o segundo slide  
 ImageView imageView = view.findViewById(R.id.*imagem\_plantairrigada*); // ID do ImageView no slide 2  
 Glide.*with*(context)  
 .asGif()  
 .load(R.drawable.*planta*) // Substitua pelo nome do seu segundo GIF  
 .into(imageView);  
 } else if (position == 2) {  
 // Slide 3: Carregar o GIF para o terceiro slide  
 ImageView imageView = view.findViewById(R.id.*imagem\_clima*); // ID do ImageView no slide 3  
 Glide.*with*(context)  
 .asGif()  
 .load(R.drawable.*climagif*) // Substitua pelo nome do seu terceiro GIF  
 .into(imageView);  
 } else if (position == 3) {  
 // Slide 4: Carregar o GIF para o quarto slide  
 ImageView imageView = view.findViewById(R.id.*imagem\_sensor*); // ID do ImageView no slide 4  
 Glide.*with*(context)  
 .asGif()  
 .load(R.drawable.*sensorplanta*) // Substitua pelo nome do seu quarto GIF  
 .into(imageView);  
 } else if (position == 4) {  
 // Slide 5: Carregar o GIF para o quinto slide  
 ImageView imageView = view.findViewById(R.id.*imagem\_grafico*); // ID do ImageView no slide 5  
 Glide.*with*(context)  
 .asGif()  
 .load(R.drawable.*grafico*) // Substitua pelo nome do seu quinto GIF  
 .into(imageView);  
 } else if (position == 5) {  
 // Slide 6: Carregar o GIF para o sexto slide  
 ImageView imageView = view.findViewById(R.id.*imagem\_home*); // ID do ImageView no slide 6  
 Glide.*with*(context)  
 .asGif()  
 .load(R.drawable.*homegif*) // Substitua pelo nome do seu sexto GIF  
 .into(imageView);  
 } else if (position == 6) {  
 // Slide 7: Carregar o GIF para o sétimo slide  
 ImageView imageView = view.findViewById(R.id.*imagem\_situt*); // ID do ImageView no slide 7  
 Glide.*with*(context)  
 .asGif()  
 .load(R.drawable.*situtgif*) // Substitua pelo nome do seu sétimo GIF  
 .into(imageView);  
 }  
  
 // Adiciona o slide ao container do ViewPager  
 container.addView(view);  
 return view; // Retorna a view inflada para o ViewPager  
 }  
  
 @Override  
 public int getCount() {  
 return layouts.length; // Retorna o número total de slides  
 }  
  
 @Override  
 public boolean isViewFromObject(View view, Object object) {  
 return view == object; // Verifica se a view passada é a mesma do objeto  
 }  
  
 @Override  
 public void destroyItem(ViewGroup container, int position, Object object) {  
 // Remove a view do ViewPager  
 View view = (View) object;  
 container.removeView(view);  
 }  
}

**ZoomOutPageAdapter**

import android.view.View;  
import androidx.viewpager.widget.ViewPager;  
  
public class ZoomOutPageTransformer implements ViewPager.PageTransformer {  
  
 // Define os limites para o efeito de zoom e opacidade.  
 private static final float *MIN\_SCALE* = 0.85f; // Escala mínima das páginas durante a transição.  
 private static final float *MIN\_ALPHA* = 0.5f; // Opacidade mínima das páginas durante a transição.  
  
 @Override  
 public void transformPage(View view, float position) {  
 // Obtém a largura e altura da página  
 int pageWidth = view.getWidth();  
 int pageHeight = view.getHeight();  
  
 // Se a posição da página for menos que -1 (fora da tela à esquerda)  
 if (position < -1) {  
 view.setAlpha(0f); // A página é invisível (completamente opaca).  
  
 // Se a posição estiver dentro da faixa de visibilidade (-1 a 1)  
 } else if (position <= 1) {  
 // Calcula o fator de escala com base na posição (quanto mais perto de 0, maior a escala)  
 float scaleFactor = Math.*max*(*MIN\_SCALE*, 1 - Math.*abs*(position));  
  
 // Margens verticais e horizontais para aplicar o efeito de zoom  
 float vertMargin = pageHeight \* (1 - scaleFactor) / 2;  
 float horzMargin = pageWidth \* (1 - scaleFactor) / 2;  
  
 // Se a página está à esquerda (posição negativa), aplica a tradução à esquerda  
 if (position < 0) {  
 view.setTranslationX(horzMargin - vertMargin / 2);  
 // Se a página está à direita (posição positiva), aplica a tradução à direita  
 } else {  
 view.setTranslationX(-horzMargin + vertMargin / 2);  
 }  
  
 // Aplica o efeito de escala na página  
 view.setScaleX(scaleFactor);  
 view.setScaleY(scaleFactor);  
  
 // Aplica a opacidade, dependendo do fator de escala  
 view.setAlpha(*MIN\_ALPHA* + (scaleFactor - *MIN\_SCALE*) / (1 - *MIN\_SCALE*) \* (1 - *MIN\_ALPHA*));  
  
 // Se a posição da página estiver fora da tela à direita (+1)  
 } else {  
 view.setAlpha(0f); // A página é invisível (completamente opaca).  
 }  
 }  
}

**Form\_registro\_sistema**

import android.app.DatePickerDialog;  
import android.content.Intent;  
import android.os.Bundle;  
import android.widget.Toast;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.AppCompatButton;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.android.material.textfield.TextInputEditText;  
import com.google.android.material.textfield.TextInputLayout;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Locale;  
import android.text.Editable;  
import android.text.TextWatcher;  
  
public class Form\_registro\_sistema extends AppCompatActivity {  
  
 private TextInputEditText editName, editData;  
 private TextInputLayout txtInputLayoutName, txtInputLayoutData;  
 private DatabaseReference mDatabase;  
 private FirebaseAuth mAuth;  
 private AppCompatButton btn\_jacriei;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_form\_registro\_sistema*);  
  
 // Inicialização dos componentes  
 editName = findViewById(R.id.*editName*);  
 editData = findViewById(R.id.*editData*);  
 txtInputLayoutName = findViewById(R.id.*txtInputLayoutName*);  
 txtInputLayoutData = findViewById(R.id.*txtInputLayoutData*);  
 btn\_jacriei = findViewById(R.id.*btnjacriei*);  
  
 // Inicializar o Firebase Authentication e o Firebase Realtime Database  
 mAuth = FirebaseAuth.*getInstance*();  
 mDatabase = FirebaseDatabase.*getInstance*().getReference("Sistemas");  
  
 btn\_jacriei.setOnClickListener(v -> {  
 Intent intent = new Intent(Form\_registro\_sistema.this, Form\_home.class);  
 startActivity(intent);  
 finish();  
 });  
  
 // Configurar o botão de cadastro  
 findViewById(R.id.*btnCadastrar*).setOnClickListener(v -> {  
 // Verificar se os campos estão preenchidos  
 String nome = editName.getText().toString().trim();  
 String data = editData.getText().toString().trim();  
  
 if (nome.isEmpty()) {  
 txtInputLayoutName.setError("Nome é obrigatório");  
 return;  
 }  
  
 if (data.isEmpty()) {  
 txtInputLayoutData.setError("Data é obrigatória");  
 return;  
 }  
  
 // Se os campos estiverem válidos, salvar os dados no Firebase  
 salvarSistema(nome, data);  
 });  
  
 // Configurar o campo de data para abrir o DatePicker  
 editData.setOnClickListener(v -> {  
 // Obter a data atual  
 Calendar calendar = Calendar.*getInstance*();  
 int year = calendar.get(Calendar.*YEAR*);  
 int month = calendar.get(Calendar.*MONTH*);  
 int day = calendar.get(Calendar.*DAY\_OF\_MONTH*);  
  
 // Criar o DatePicker  
 DatePickerDialog datePickerDialog = new DatePickerDialog(  
 Form\_registro\_sistema.this,  
 (view, year1, monthOfYear, dayOfMonth) -> {  
 // Formatar a data no formato dd/MM/yyyy  
 Calendar selectedDate = Calendar.*getInstance*();  
 selectedDate.set(year1, monthOfYear, dayOfMonth);  
 SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy", Locale.*getDefault*());  
 String formattedDate = sdf.format(selectedDate.getTime());  
 // Preencher o campo com a data selecionada  
 editData.setText(formattedDate);  
 },  
 year,  
 month,  
 day  
 );  
 datePickerDialog.show();  
 });  
  
 // Adicionar um TextWatcher para formatar automaticamente a data enquanto o usuário digita  
 editData.addTextChangedListener(new TextWatcher() {  
 @Override  
 public void beforeTextChanged(CharSequence charSequence, int start, int count, int after) {}  
  
 @Override  
 public void onTextChanged(CharSequence charSequence, int start, int before, int after) {  
 // Evitar formatação excessiva  
 if (charSequence.length() == 2 || charSequence.length() == 5) {  
 // Inserir a barra automaticamente após o dia e o mês  
 if (charSequence.length() == 2) {  
 editData.append("/");  
 } else if (charSequence.length() == 5) {  
 editData.append("/");  
 }  
 }  
 }  
  
 @Override  
 public void afterTextChanged(Editable editable) {  
 // Verificar se a entrada ultrapassou o limite de caracteres  
 if (editable.length() > 10) {  
 editData.setText(editable.subSequence(0, 10));  
 editData.setSelection(editable.length());  
 Toast.*makeText*(Form\_registro\_sistema.this, "Formato de data inválido", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
  
 private void salvarSistema(String nome, String data) {  
 // Obter o UID do usuário logado  
 String userId = mAuth.getCurrentUser().getUid(); // UID do usuário logado  
  
 // Criar um ID único para cada sistema ou usar o nome como chave  
 String sistemaId = mDatabase.push().getKey(); // Gera um ID único  
  
 // Criar um objeto Sistema com os dados  
 Sistema sistema = new Sistema(nome, data);  
  
 // Salvar os dados no Firebase Realtime Database sob o ID do usuário  
 if (sistemaId != null) {  
 // Salvando os dados sob o ID do usuário  
 mDatabase.child(userId).child(sistemaId).setValue(sistema)  
 .addOnSuccessListener(aVoid -> {  
 Toast.*makeText*(Form\_registro\_sistema.this, "Sistema cadastrado com sucesso!", Toast.*LENGTH\_SHORT*).show();  
 })  
 .addOnFailureListener(e -> {  
 Toast.*makeText*(Form\_registro\_sistema.this, "Erro ao cadastrar sistema", Toast.*LENGTH\_SHORT*).show();  
 });  
  
 // Redirecionar para a tela de exibição de dados  
 Intent intent = new Intent(Form\_registro\_sistema.this, Form\_home.class);  
 startActivity(intent);  
 finish(); // Fecha a tela de cadastro para não permitir voltar a ela  
 }  
 }  
  
 // Classe para representar um Sistema  
 public static class Sistema {  
 public String nome;  
 public String data;  
  
 public Sistema() {  
 // Constructor default necessário para o Firebase  
 }  
  
 public Sistema(String nome, String data) {  
 this.nome = nome;  
 this.data = data;  
 }  
 }  
}

**Form\_home**

import android.os.Bundle;  
import android.content.Intent;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.AppCompatButton;  
import androidx.core.graphics.Insets;  
import androidx.core.view.ViewCompat;  
import androidx.core.view.WindowInsetsCompat;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
import com.google.firebase.database.DataSnapshot;

public class Form\_home extends AppCompatActivity {

// Declaração dos botões e do TextView

private AppCompatButton btclima, btirriga, btstatus, btinfo, btnuser, btnConfig, btnBook;

private TextView txtNomeSistema;

// Declaração das instâncias do Firebase Authentication e Database

private FirebaseAuth mAuth;

private DatabaseReference mDatabase;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_form\_home);

// Configuração do Edge-to-Edge para layout fullscreen

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

// Obtém as margens do sistema e aplica ao layout para exibição fullscreen

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

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

return insets;

});

// Inicializa os componentes de UI e Firebase

IniciarComponentes();

mAuth = FirebaseAuth.getInstance(); // Obter instância do FirebaseAuth

mDatabase = FirebaseDatabase.getInstance().getReference("Sistemas"); // Referência do nó "Sistemas" no Firebase

// Obter o usuário autenticado e carregar o nome do sistema

FirebaseUser user = mAuth.getCurrentUser();

if (user != null) {

// Se o usuário estiver autenticado, obtém o ID e carrega o nome do sistema

String userId = user.getUid();

carregarNomeSistema(userId);

} else {

// Exibe mensagem caso não haja usuário autenticado

Toast.makeText(this, "Usuário não autenticado.", Toast.LENGTH\_SHORT).show();

}

// Configuração dos botões

configurarBotoes();

}

private void IniciarComponentes() {

// Associa as variáveis de botões e o TextView aos elementos do layout

btclima = findViewById(R.id.btn\_clima);

btirriga = findViewById(R.id.btn\_irrigacao);

btinfo = findViewById(R.id.btn\_info);

btstatus = findViewById(R.id.btn\_status);

btnuser = findViewById(R.id.btnuser);

btnConfig = findViewById(R.id.btnConfig);

btnBook = findViewById(R.id.btnbook);

txtNomeSistema = findViewById(R.id.txtNomeSistema);

}

private void configurarBotoes() {

// Define ação do botão "Clima" para abrir a MainActivity

btclima.setOnClickListener(view -> {

Intent intent = new Intent(Form\_home.this, MainActivity.class);

startActivity(intent);

});

// Define ação do botão "Irrigação" para abrir a Activity Form\_irrigacao

btirriga.setOnClickListener(view -> {

Intent intent = new Intent(Form\_home.this, Form\_irrigacao.class);

startActivity(intent);

});

// Define ação do botão "Status" para abrir a Activity Form\_status

btstatus.setOnClickListener(view -> {

Intent intent = new Intent(Form\_home.this, Form\_status.class);

startActivity(intent);

});

// Define ação do botão "Info" para abrir a Activity Form\_info

btinfo.setOnClickListener(view -> {

Intent intent = new Intent(Form\_home.this, Form\_info.class);

startActivity(intent);

});

// Define ação do botão "Book" para abrir a Activity IntroActivity

btnBook.setOnClickListener(view -> {

Intent intent = new Intent(Form\_home.this, IntroActivity.class);

startActivity(intent);

});

// Define ação do botão "Usuário" para abrir a Activity Form\_User

btnuser.setOnClickListener(view -> {

Intent intent = new Intent(Form\_home.this, Form\_User.class);

startActivity(intent);

});

// Define ação do botão "Configuração" para abrir a Activity FormSobrenos

btnConfig.setOnClickListener(view -> {

Intent intent = new Intent(Form\_home.this, FormSobrenos.class);

startActivity(intent);

});

}

private void carregarNomeSistema(String userId) {

// Buscar o nome do sistema do usuário no Firebase

mDatabase.child(userId).addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

if (dataSnapshot.exists()) {

// Itera sobre os filhos do snapshot para obter o nome do sistema

for (DataSnapshot sistemaSnapshot : dataSnapshot.getChildren()) {

Sistema sistema = sistemaSnapshot.getValue(Sistema.class); // Obtém o objeto Sistema

if (sistema != null) {

txtNomeSistema.setText(sistema.nome); // Define o nome do sistema no TextView

}

}

} else {

// Exibe mensagem se não houver sistema para o usuário

Toast.makeText(Form\_home.this, "Nenhum sistema encontrado para este usuário.", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

// Exibe mensagem de erro caso a leitura do Firebase seja cancelada

Toast.makeText(Form\_home.this, "Erro ao recuperar dados do sistema.", Toast.LENGTH\_SHORT).show();

}

});

}

// Classe interna para representar um Sistema

public static class Sistema {

public String nome;

public Sistema() {

// Construtor padrão necessário para o Firebase

}

public Sistema(String nome, String data) {

this.nome = nome;

}

}

}

**Form\_irrigacao**

import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.AppCompatButton;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.EditText;  
import android.widget.ToggleButton;  
import com.firebase.client.Firebase;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;

public class Form\_irrigacao extends AppCompatActivity { // Declara a classe Form\_irrigacao que herda de AppCompatActivity

Firebase meuFirebase; // Declara uma instância de Firebase para conexão com o banco de dados

DatabaseReference dados; // Referência para o banco de dados Firebase

int intervalo\_int; // Variável para armazenar o valor do intervalo

int duracao\_int; // Variável para armazenar o valor da duração

AppCompatButton btSalvar, btApagar; // Botões para salvar e apagar os dados

ToggleButton btI; // ToggleButton para alternar o estado de irrigação

private AppCompatButton btn\_voltar; // Botão para voltar a tela de Home

@Override

protected void onCreate(Bundle savedInstanceState) { // Método onCreate, chamado ao iniciar a Activity

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_form\_irrigacao); // Define o layout da Activity

// Inicializa o Firebase

dados = FirebaseDatabase.getInstance().getReference(); // Obtém a referência principal do Firebase Database

Firebase.setAndroidContext(this); // Define o contexto Android para o Firebase

// Encontrando os elementos da interface

btSalvar = findViewById(R.id.btn\_salvar); // Associa o botão salvar ao elemento do layout

btApagar = findViewById(R.id.btn\_descartar); // Associa o botão apagar ao elemento do layout

btI = findViewById(R.id.Irrigando); // Associa o ToggleButton ao elemento do layout

EditText edtI = findViewById(R.id.edit\_intervalo); // Associa o campo de texto do intervalo ao layout

EditText edt2 = findViewById(R.id.edit\_duracao); // Associa o campo de texto da duração ao layout

btn\_voltar = findViewById(R.id.btn\_voltar); // Associa o botão de voltar ao elemento do layout

btn\_voltar.setOnClickListener(v -> { // Define o clique do botão de voltar

Intent intent = new Intent(Form\_irrigacao.this, Form\_home.class); // Cria uma intenção para abrir Form\_home

startActivity(intent); // Inicia a Activity Form\_home

finish(); // Finaliza a Activity atual

});

// Referência para o Firebase

meuFirebase = new Firebase("https://situt-258258-default-rtdb.firebaseio.com/"); // Instancia o Firebase com a URL do banco de dados

// Botão Salvar - já presente no seu código

btSalvar.setOnClickListener(new View.OnClickListener() { // Define o clique do botão salvar

@Override

public void onClick(View v) {

intervalo\_int = Integer.parseInt(edtI.getText().toString()); // Converte o texto de intervalo para inteiro

duracao\_int = Integer.parseInt(edt2.getText().toString()); // Converte o texto de duração para inteiro

// Atualizando os valores no Firebase

Firebase info = meuFirebase.child("/Irriga/Intervalo"); // Referência para o caminho de intervalo no Firebase

info.setValue(intervalo\_int); // Define o valor do intervalo no banco de dados

Firebase info2 = meuFirebase.child("/Irriga/Duracao"); // Referência para o caminho de duração no Firebase

info2.setValue(duracao\_int); // Define o valor da duração no banco de dados

}

});

// Botão Apagar - Nova funcionalidade que zera os valores e envia para o Firebase

btApagar.setOnClickListener(new View.OnClickListener() { // Define o clique do botão apagar

@Override

public void onClick(View v) {

// Definindo intervalo e duração para 0

intervalo\_int = 0; // Zera o valor do intervalo

duracao\_int = 0; // Zera o valor da duração

// Enviando para o Firebase

Firebase info = meuFirebase.child("/Irriga/Intervalo"); // Referência para o caminho de intervalo no Firebase

info.setValue(intervalo\_int); // Atualiza o valor de intervalo no banco de dados para 0

Firebase info2 = meuFirebase.child("/Irriga/Duracao"); // Referência para o caminho de duração no Firebase

info2.setValue(duracao\_int); // Atualiza o valor de duração no banco de dados para 0

// Atualizando os campos EditText para 0

edtI.setText("0"); // Define o campo de intervalo para 0 na interface

edt2.setText("0"); // Define o campo de duração para 0 na interface

}

});

}

// Referência para a chave "Irriga\_a" no Firebase

DatabaseReference referenceIrriga\_a = FirebaseDatabase.getInstance("https://situt-258258-default-rtdb.firebaseio.com/").getReference("Irriga/Irriga\_a"); // Referência direta ao caminho Irriga/Irriga\_a no banco de dados

// Função chamada quando o ToggleButton é alterado

public void onToggleClicked(View view) { // Método executado quando o ToggleButton é clicado

if (((ToggleButton) view).isChecked()) { // Se o botão está marcado

referenceIrriga\_a.setValue(0); // Define o valor no Firebase como 0 (ligado)

} else { // Se o botão não está marcado

referenceIrriga\_a.setValue(1); // Define o valor no Firebase como 1 (desligado)

}

}

}

**Form\_info**

import android.Manifest;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.graphics.Color;  
import android.os.Bundle;  
import android.os.Environment;  
import android.util.Log;  
import android.widget.Toast;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.AppCompatButton;  
import androidx.core.content.ContextCompat;  
import com.github.mikephil.charting.charts.BarChart;  
import com.github.mikephil.charting.charts.PieChart;  
import com.github.mikephil.charting.data.BarData;  
import com.github.mikephil.charting.data.BarDataSet;  
import com.github.mikephil.charting.data.BarEntry;  
import com.github.mikephil.charting.data.PieData;  
import com.github.mikephil.charting.data.PieDataSet;  
import com.github.mikephil.charting.data.PieEntry;  
import com.google.firebase.FirebaseApp;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
import com.itextpdf.kernel.pdf.PdfDocument;  
import com.itextpdf.kernel.pdf.PdfWriter;  
import com.itextpdf.layout.Document;  
import com.itextpdf.layout.element.Paragraph;  
import java.io.IOException;  
import java.util.ArrayList;

public class Form\_info extends AppCompatActivity { // Declara a classe Form\_info que herda de AppCompatActivity

private DatabaseReference historicoRef, sensoresRef; // Declara referências para os dados de histórico e sensores no Firebase

private BarChart barChart; // Declara um gráfico de barras

private PieChart pieChart; // Declara um gráfico de pizza

private AppCompatButton btn\_voltar; // Declara um botão para voltar à Activity anterior

@Override

protected void onCreate(Bundle savedInstanceState) { // Método onCreate, chamado ao iniciar a Activity

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_form\_info); // Define o layout da Activity

// Inicializa o Firebase

FirebaseApp.initializeApp(this); // Inicializa o Firebase com o contexto da Activity

FirebaseDatabase database = FirebaseDatabase.getInstance(); // Obtém a instância do banco de dados Firebase

// Referências para as pastas no Firebase

historicoRef = database.getReference("Historico"); // Referência para a pasta "Historico" no Firebase

sensoresRef = database.getReference("sensores"); // Referência para a pasta "sensores" no Firebase

// Configura o BarChart

barChart = findViewById(R.id.barChart); // Associa o BarChart ao layout

carregarDadosEConstruirGrafico(); // Carrega dados e constrói o gráfico de barras

// Configura o PieChart

pieChart = findViewById(R.id.pieChart); // Associa o PieChart ao layout

carregarDadosEConstruirGraficoPizza(); // Carrega dados e constrói o gráfico de pizza

btn\_voltar = findViewById(R.id.btn\_voltar); // Associa o botão de voltar ao layout

btn\_voltar.setOnClickListener(v -> { // Define o clique no botão de voltar

Intent intent = new Intent(Form\_info.this, Form\_home.class); // Cria uma intenção para abrir Form\_home

startActivity(intent); // Inicia a Activity Form\_home

finish(); // Finaliza a Activity atual

});

// Configura o botão para gerar o PDF

AppCompatButton button = findViewById(R.id.btn\_download); // Associa o botão de download ao layout

button.setOnClickListener(v -> { // Define o clique para gerar o PDF

solicitarPermissao(); // Solicita permissão para gravar no armazenamento

gerarPdf(); // Chama o método para gerar o PDF

});

}

// Método para solicitar permissão de escrita no armazenamento externo

private void solicitarPermissao() {

if (checkSelfPermission(Manifest.permission.WRITE\_EXTERNAL\_STORAGE) != PackageManager.PERMISSION\_GRANTED) { // Verifica se a permissão não foi concedida

requestPermissions(new String[]{Manifest.permission.WRITE\_EXTERNAL\_STORAGE}, 1); // Solicita permissão de escrita no armazenamento

}

}

// Método para carregar dados e construir o gráfico de barras

private void carregarDadosEConstruirGrafico() {

sensoresRef.addListenerForSingleValueEvent(new ValueEventListener() { // Escuta uma vez os dados da pasta "sensores"

@Override

public void onDataChange(DataSnapshot snapshot) { // Executado quando os dados são recebidos

Log.d("Firebase", "Dados recebidos: " + snapshot.getValue()); // Loga os dados recebidos

if (snapshot.exists()) { // Verifica se os dados existem

ArrayList<BarEntry> chuvaEntries = new ArrayList<>(); // Lista para entradas do gráfico de chuva

ArrayList<BarEntry> umidadeEntries = new ArrayList<>(); // Lista para entradas do gráfico de umidade

ArrayList<BarEntry> uvEntries = new ArrayList<>(); // Lista para entradas do gráfico de UV

String chuvaStr = snapshot.child("chuva").getValue(String.class); // Obtém o valor de chuva como string

String umidadeStr = snapshot.child("umidade").getValue(String.class); // Obtém o valor de umidade como string

String uvStr = snapshot.child("uv").getValue(String.class); // Obtém o valor de UV como string

if (chuvaStr != null && umidadeStr != null && uvStr != null) { // Verifica se todos os valores existem

try {

float chuva = Float.parseFloat(chuvaStr); // Converte o valor de chuva para float

float umidade = Float.parseFloat(umidadeStr); // Converte o valor de umidade para float

float uv = Float.parseFloat(uvStr); // Converte o valor de UV para float

chuvaEntries.add(new BarEntry(0, chuva)); // Adiciona entrada de chuva no gráfico

umidadeEntries.add(new BarEntry(1, umidade)); // Adiciona entrada de umidade no gráfico

uvEntries.add(new BarEntry(2, uv)); // Adiciona entrada de UV no gráfico

} catch (NumberFormatException e) {

Log.e("Firebase", "Erro ao converter valores: " + e.getMessage()); // Loga erro de conversão

}

} else {

Log.e("Firebase", "Valores nulos para chuva, umidade ou UV."); // Loga valores nulos

}

if (!chuvaEntries.isEmpty() && !umidadeEntries.isEmpty() && !uvEntries.isEmpty()) { // Verifica se há entradas válidas

// Define um conjunto de dados e cor para cada variável

BarDataSet chuvaDataSet = new BarDataSet(chuvaEntries, "Chuva"); // Define o dataset para chuva

chuvaDataSet.setColor(Color.BLUE); // Define a cor do dataset

chuvaDataSet.setValueTextColor(Color.parseColor("#f5f5f5")); // Define a cor do texto

chuvaDataSet.setValueTextSize(15f); // Define o tamanho do texto

BarDataSet umidadeDataSet = new BarDataSet(umidadeEntries, "Umidade"); // Define o dataset para umidade

umidadeDataSet.setColor(Color.GREEN);

umidadeDataSet.setValueTextColor(Color.parseColor("#f5f5f5"));

umidadeDataSet.setValueTextSize(15f);

BarDataSet uvDataSet = new BarDataSet(uvEntries, "UV"); // Define o dataset para UV

uvDataSet.setColor(Color.RED);

uvDataSet.setValueTextColor(Color.parseColor("#f5f5f5"));

uvDataSet.setValueTextSize(15f);

BarData data = new BarData(chuvaDataSet, umidadeDataSet, uvDataSet); // Define os dados do gráfico

barChart.setData(data); // Configura os dados no BarChart

barChart.animateY(2000); // Anima o gráfico

// Configura as cores do texto dos eixos X e Y

barChart.getXAxis().setTextColor(Color.parseColor("#f5f5f5"));

barChart.getAxisLeft().setTextColor(Color.parseColor("#f5f5f5"));

barChart.getAxisRight().setTextColor(Color.parseColor("#f5f5f5"));

// Configura a cor do texto da legenda

barChart.getLegend().setTextColor(Color.parseColor("#f5f5f5"));

// Configura o texto e a cor da descrição

barChart.getDescription().setText("Valor dos sensores");

barChart.getDescription().setTextColor(Color.parseColor("#f5f5f5"));

barChart.invalidate(); // Redesenha o gráfico

} else {

Log.e("Firebase", "Nenhum dado válido encontrado."); // Loga ausência de dados válidos

}

} else {

Log.e("Firebase", "Nenhum dado encontrado na pasta Sensores"); // Loga ausência de dados na pasta

}

}

@Override

public void onCancelled(DatabaseError error) {

Log.e("Firebase", "Erro ao ler os dados: " + error.getMessage()); // Loga erro de leitura

}

});

}

// Método para carregar dados e construir o gráfico de pizza

private void carregarDadosEConstruirGraficoPizza() {

sensoresRef.addListenerForSingleValueEvent(new ValueEventListener() { // Escuta uma vez os dados da pasta "sensores"

@Override

public void onDataChange(DataSnapshot snapshot) { // Executado quando os dados são recebidos

Log.d("Firebase", "Dados recebidos: " + snapshot.getValue());

if (snapshot.exists()) { // Verifica se os dados existem

ArrayList<PieEntry> entries = new ArrayList<>(); // Lista para entradas do gráfico de pizza

String chuvaStr = snapshot.child("chuva").getValue(String.class);

String umidadeStr = snapshot.child("umidade").getValue(String.class);

String uvStr = snapshot.child("uv").getValue(String.class);

if (chuvaStr != null && umidadeStr != null && uvStr != null) { // Verifica se todos os valores existem

try {

float chuva = Float.parseFloat(chuvaStr); // Converte o valor de chuva para float

float umidade = Float.parseFloat(umidadeStr); // Converte o valor de umidade para float

float uv = Float.parseFloat(uvStr); // Converte o valor de UV para float

entries.add(new PieEntry(chuva, "Chuva")); // Adiciona entrada de chuva no gráfico de pizza

entries.add(new PieEntry(umidade, "Umidade")); // Adiciona entrada de umidade

entries.add(new PieEntry(uv, "UV")); // Adiciona entrada de UV

} catch (NumberFormatException e) {

Log.e("Firebase", "Erro ao converter valores: " + e.getMessage()); // Loga erro de conversão

}

} else {

Log.e("Firebase", "Valores nulos para chuva, umidade ou UV."); // Loga valores nulos

}

if (!entries.isEmpty()) { // Verifica se há entradas válidas

PieDataSet dataSet = new PieDataSet(entries, ""); // Define o dataset para o gráfico de pizza

dataSet.setColors(Color.BLUE, Color.GREEN, Color.RED); // Define as cores para cada entrada

dataSet.setValueTextColor(Color.parseColor("#f5f5f5")); // Define a cor do texto

dataSet.setValueTextSize(15f); // Define o tamanho do texto

PieData data = new PieData(dataSet); // Define os dados do gráfico de pizza

pieChart.setData(data); // Configura os dados no PieChart

pieChart.animateY(2000); // Anima o gráfico

pieChart.getLegend().setTextColor(Color.parseColor("#f5f5f5")); // Define a cor do texto da legenda

pieChart.getDescription().setText("Valor dos sensores"); // Define o texto da descrição

pieChart.getDescription().setTextColor(Color.parseColor("#f5f5f5")); // Define a cor da descrição

pieChart.invalidate(); // Redesenha o gráfico

} else {

Log.e("Firebase", "Nenhum dado válido encontrado."); // Loga ausência de dados válidos

}

} else {

Log.e("Firebase", "Nenhum dado encontrado na pasta Sensores"); // Loga ausência de dados na pasta

}

}

@Override

public void onCancelled(DatabaseError error) {

Log.e("Firebase", "Erro ao ler os dados: " + error.getMessage()); // Loga erro de leitura

}

});

}

// Método para gerar o PDF

private void gerarPdf() {

PdfDocument pdfDocument = new PdfDocument(); // Cria um novo documento PDF

Paint paint = new Paint(); // Define o estilo de pintura

PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder(400, 600, 1).create(); // Define o tamanho e número da página

PdfDocument.Page page = pdfDocument.startPage(pageInfo); // Inicia a página

Canvas canvas = page.getCanvas(); // Obtém o canvas da página

paint.setTextAlign(Paint.Align.CENTER); // Define o alinhamento do texto

paint.setTextSize(20f); // Define o tamanho do texto

canvas.drawText("Dados dos sensores", pageInfo.getPageWidth() / 2, 40, paint); // Adiciona texto ao PDF

paint.setTextSize(12f); // Define o tamanho do texto

paint.setTextAlign(Paint.Align.LEFT); // Define o alinhamento do texto

// Adiciona texto ao PDF com os dados dos sensores

canvas.drawText("Chuva: 85%", 20, 100, paint);

canvas.drawText("Umidade: 45%", 20, 120, paint);

canvas.drawText("UV: 3", 20, 140, paint);

pdfDocument.finishPage(page); // Finaliza a página

String targetPdf = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_DOWNLOADS) + "/DadosSensores.pdf"; // Define o caminho do PDF

File filePath = new File(targetPdf); // Cria um arquivo para salvar o PDF

try {

pdfDocument.writeTo(new FileOutputStream(filePath)); // Escreve o PDF no caminho especificado

Toast.makeText(this, "PDF gerado com sucesso!", Toast.LENGTH\_SHORT).show(); // Exibe uma mensagem de sucesso

} catch (IOException e) {

Log.e("PDF", "Erro ao escrever o arquivo PDF: " + e.getMessage()); // Loga erro de escrita

}

pdfDocument.close(); // Fecha o documento

}

}

**MainActivity**

import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.AppCompatButton;  
import android.content.Intent;  
import android.widget.ImageView;  
import android.widget.TextView;  
import com.example.tccpuxxadados.api.Forecast;  
import com.example.tccpuxxadados.api.Results;  
import com.example.tccpuxxadados.api.WeatherModel;  
import com.example.tccpuxxadados.api.WeatherService;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import java.text.DecimalFormat;  
import java.util.List;  
import retrofit2.Call;  
import retrofit2.Callback;  
import retrofit2.Response;  
import retrofit2.Retrofit;  
import retrofit2.converter.gson.GsonConverterFactory;  
import android.os.Bundle;  
  
public class MainActivity extends AppCompatActivity {  
  
 private Integer currentTemperature;  
 String currentCondition;  
 private AppCompatButton btn\_voltar;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_form\_clima*);  
  
  
 btn\_voltar = findViewById(R.id.*btn\_voltar*);  
 btn\_voltar.setOnClickListener(v -> {  
 Intent intent = new Intent(MainActivity.this, Form\_home.class);  
 startActivity(intent);  
 finish();  
 });  
  
 Retrofit retrofit = new Retrofit.Builder()  
 .baseUrl("https://api.hgbrasil.com/")  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
  
 WeatherService service = retrofit.create(WeatherService.class);  
 Call<WeatherModel> call = service.getWeather("12b81fef","São Bernardo do Campo, SP");  
  
 call.enqueue(new Callback<WeatherModel>() {  
 @Override  
 public void onResponse(Call<WeatherModel> call, Response<WeatherModel> response) {  
  
 assert response.body() != null;  
 Results results = response.body().getResults();  
  
 setDataToView(R.id.*temp\_hoje*, formatTempToC(results.getTemp()));  
 setDataToView(R.id.*condicao\_hoje*, results.getDescription());  
 setDataToView(R.id.*text\_dataHoje*,results.getDate());  
 Integer codeHoje = Integer.*valueOf*(results.getConditionCode());  
 getImageForCondition0(codeHoje);  
  
 List<Forecast> forecasts = results.getForecast();  
  
 String dayOfWeek1 = forecasts.get(1).getWeekday();  
 setDataToView(R.id.*next\_day1*, dayOfWeek1);  
 String description1 = forecasts.get(1).getDescription();  
 setDataToView(R.id.*cond\_nextDay1*, description1);  
 Integer minTemperature1 = forecasts.get(1).getMin();  
 setDataToView(R.id.*minTemp\_d1*, formatTempToC(minTemperature1));  
 Integer maxTemperature1 = forecasts.get(1).getMax();  
 setDataToView(R.id.*maxTemp\_d1*, formatTempToC(maxTemperature1));  
 String conditionCode1 = forecasts.get(1).getCondition();  
 getImageForCondition1(conditionCode1);  
 String dayOfWeek2 = forecasts.get(2).getWeekday();  
 setDataToView(R.id.*next\_day2*, dayOfWeek2);  
 String descripition2 = forecasts.get(2).getDescription();  
 setDataToView(R.id.*cond\_nextDay2*, descripition2);  
 Integer minTemperature2 = forecasts.get(2).getMin();  
 setDataToView(R.id.*minTemp\_d2*, formatTempToC(minTemperature2));  
 Integer maxTemperature2 = forecasts.get(2).getMax();  
 setDataToView(R.id.*maxTemp\_d2*, formatTempToC(maxTemperature2));  
 String conditionCode2 = forecasts.get(2).getCondition();  
 getImageForCondition(conditionCode2);  
  
 }  
  
 @Override  
 public void onFailure(Call<WeatherModel> call, Throwable t) {  
 t.printStackTrace();  
 }  
 }  
 );  
 }  
 DatabaseReference referenceClima= FirebaseDatabase.*getInstance*("https://situt-258258-default-rtdb.firebaseio.com/").getReference("Clima/Previsao");  
 private void getImageForCondition0(Integer condition) {  
  
 ImageView imageCondition0 = findViewById(R.id.*image\_today*);  
  
 if (condition <=18 || (condition>=37 && condition<=43) || (condition>=45) || ((condition==35))) { //chuva  
 imageCondition0.setImageResource(R.drawable.*chuvaimg*);  
 referenceClima.setValue(1);  
  
 } else if ( condition <=30) { //nublado  
 imageCondition0.setImageResource(R.drawable.*nubladoimg*);  
 referenceClima.setValue(0);  
 } else if (condition==33){ //Ensolarado  
 imageCondition0.setImageResource(R.drawable.*noite*);  
 referenceClima.setValue(0);  
 }else{  
 imageCondition0.setImageResource(R.drawable.*solimage*);  
 referenceClima.setValue(0);  
 }  
  
 }  
  
 private void getImageForCondition1(String condition) {  
  
 ImageView imageCondition1 = findViewById(R.id.*icon\_nextDay1*);  
  
 if (condition.equals("storm")||condition.equals("snow")||condition.equals("hail")||condition.equals("rain")) { //chuva  
 imageCondition1.setImageResource(R.drawable.*chuvaicon*);  
  
 } else if (condition.equals("fog")||condition.equals("cloud")||condition.equals("cloudly\_day")||condition.equals("cloudly\_night")||condition.equals("none\_night")) { //nublado  
 imageCondition1.setImageResource(R.drawable.*nubladoicon*);  
 } else { //Ensolarado  
 imageCondition1.setImageResource(R.drawable.*solicon*);  
 }  
  
 }  
 private void getImageForCondition(String condition) {  
  
 ImageView imageCondition2 = findViewById(R.id.*icon\_nextDay2*);  
  
 if (condition.equals("storm")||condition.equals("snow")||condition.equals("hail")||condition.equals("rain")) { //chuva  
 imageCondition2.setImageResource(R.drawable.*chuvaicon*);  
  
 } else if (condition.equals("fog")||condition.equals("cloud")||condition.equals("cloudly\_day")||condition.equals("cloudly\_night")||condition.equals("none\_night")) { //nublado  
 imageCondition2.setImageResource(R.drawable.*nubladoicon*);  
 } else { //Ensolarado  
 imageCondition2.setImageResource(R.drawable.*solicon*);  
 }  
  
 }  
  
 private void setDataToView(int resourceId, String value) {  
 TextView view = findViewById(resourceId);  
 view.setText(value);  
 }  
  
 private String formatTempToC(Integer temperature) {  
 return new DecimalFormat("#.#").format(temperature) + "\u2103";  
 }  
}

**Form\_status**

import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.AppCompatButton;  
import android.content.Intent;  
import android.os.Bundle;  
import android.os.Handler;  
import android.util.Log;  
import android.widget.TextView;  
import com.google.firebase.FirebaseApp;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;

public class Form\_status extends AppCompatActivity {

// Declaração das variáveis de TextViews para exibir os dados dos sensores

private TextView umidadeTextView;

private TextView chuvaTextView;

private TextView uvTextView;

// Declaração do botão de voltar para a Activity anterior

private AppCompatButton btn\_voltar;

// Declaração da referência ao banco de dados Firebase

private DatabaseReference ref;

// Declaração do Handler e Runnable para atualizar periodicamente os dados

private Handler handler;

private Runnable runnable;

// Intervalo de atualização em milissegundos (5 minutos)

private final long REFRESH\_INTERVAL = 5 \* 60 \* 1000;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_form\_status);

// Inicializa o Firebase

FirebaseApp.initializeApp(this);

// Inicializa os TextViews com os IDs dos elementos no layout XML

umidadeTextView = findViewById(R.id.umidadeTextView);

chuvaTextView = findViewById(R.id.chuvaTextView);

uvTextView = findViewById(R.id.uvTextView);

// Configura o botão "Voltar" para retornar à Activity anterior

btn\_voltar = findViewById(R.id.btn\_voltar);

btn\_voltar.setOnClickListener(v -> {

Intent intent = new Intent(Form\_status.this, Form\_home.class);

startActivity(intent);

finish();

});

// Configura a referência ao banco de dados do Firebase na localização "sensores"

FirebaseDatabase database = FirebaseDatabase.getInstance();

ref = database.getReference("sensores");

// Inicializa o Handler e Runnable para atualizar os dados periodicamente

handler = new Handler();

runnable = new Runnable() {

@Override

public void run() {

// Lê os dados do Firebase e agenda a próxima execução

readDataFromFirebase();

handler.postDelayed(this, REFRESH\_INTERVAL);

}

};

// Inicia a leitura dos dados do Firebase

handler.post(runnable);

}

@Override

protected void onDestroy() {

super.onDestroy();

// Remove o Runnable para evitar chamadas após a Activity ser destruída

handler.removeCallbacks(runnable);

}

// Método para ler os dados do Firebase

private void readDataFromFirebase() {

// Adiciona um ValueEventListener para o campo "umidade" no Firebase

ref.child("umidade").addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot snapshot) {

try {

Log.d("Firebase", "Dados da umidade: " + snapshot.getValue());

// Verifica se o snapshot contém dados

if (snapshot.exists()) {

String umidadeStr = snapshot.getValue(String.class);

if (umidadeStr != null) {

try {

// Converte o valor para Integer

Integer umidade = Integer.parseInt(umidadeStr.trim());

// Define o texto do TextView de acordo com o valor

umidadeTextView.setText(umidade > 60 ? "Úmido" : "Não úmido");

} catch (NumberFormatException e) {

umidadeTextView.setText("Erro ao converter a umidade");

Log.e("Firebase", "Erro ao converter a umidade: " + e.getMessage());

}

} else {

umidadeTextView.setText("Dados não disponíveis");

}

} else {

umidadeTextView.setText("Dados não disponíveis");

}

} catch (Exception e) {

Log.e("Firebase", "Erro ao processar a umidade: " + e.getMessage());

umidadeTextView.setText("Erro ao processar a umidade");

}

}

@Override

public void onCancelled(DatabaseError error) {

// Log de erro caso a leitura do Firebase seja cancelada

Log.e("Firebase", "Erro ao ler a umidade: " + error.getMessage());

umidadeTextView.setText("Erro ao ler a umidade");

}

});

// Adiciona um ValueEventListener para o campo "chuva" no Firebase

ref.child("chuva").addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot snapshot) {

try {

Log.d("Firebase", "Dados da chuva: " + snapshot.getValue());

// Verifica se o snapshot contém dados

if (snapshot.exists()) {

String chuvaStr = snapshot.getValue(String.class);

if (chuvaStr != null) {

// Converte o valor para Boolean e define o texto do TextView

Boolean chovendo = Boolean.parseBoolean(chuvaStr.trim());

chuvaTextView.setText(chovendo ? "Chovendo" : "Não chovendo");

} else {

chuvaTextView.setText("Dados não disponíveis");

}

} else {

chuvaTextView.setText("Dados não disponíveis");

}

} catch (Exception e) {

Log.e("Firebase", "Erro ao processar a chuva: " + e.getMessage());

chuvaTextView.setText("Erro ao processar a chuva");

}

}

@Override

public void onCancelled(DatabaseError error) {

// Log de erro caso a leitura do Firebase seja cancelada

Log.e("Firebase", "Erro ao ler a chuva: " + error.getMessage());

chuvaTextView.setText("Erro ao ler a chuva");

}

});

// Adiciona um ValueEventListener para o campo "uv" no Firebase

ref.child("uv").addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot snapshot) {

try {

Log.d("Firebase", "Dados do UV: " + snapshot.getValue());

// Verifica se o snapshot contém dados

if (snapshot.exists()) {

String uvStr = snapshot.getValue(String.class);

if (uvStr != null) {

try {

// Converte o valor para Integer e exibe no TextView

Integer uv = Integer.parseInt(uvStr.trim());

uvTextView.setText(uv.toString());

} catch (NumberFormatException e) {

uvTextView.setText("Erro ao converter o UV");

Log.e("Firebase", "Erro ao converter o UV: " + e.getMessage());

}

} else {

uvTextView.setText("Dados não disponíveis");

}

} else {

uvTextView.setText("Dados não disponíveis");

}

} catch (Exception e) {

Log.e("Firebase", "Erro ao processar o UV: " + e.getMessage());

uvTextView.setText("Erro ao processar o UV");

}

}

@Override

public void onCancelled(DatabaseError error) {

// Log de erro caso a leitura do Firebase seja cancelada

Log.e("Firebase", "Erro ao ler o UV: " + error.getMessage());

uvTextView.setText("Erro ao ler o UV");

}

});

}

}

**Form\_User**

import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.AppCompatButton;  
import androidx.core.graphics.Insets;  
import androidx.core.view.ViewCompat;  
import androidx.core.view.WindowInsetsCompat;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
import com.google.firebase.database.DataSnapshot;  
  
public class Form\_User extends AppCompatActivity {  
  
 private AppCompatButton btn\_sair, btn\_voltar;  
 private TextView txtEmail, txtDataSistema, txtNomeUsuario;  
 private FirebaseAuth mAuth;  
 private DatabaseReference mDatabase;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_form\_user*);  
  
 // Configuração de padding para áreas de sistema  
 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 Firebase  
 mAuth = FirebaseAuth.*getInstance*();  
 mDatabase = FirebaseDatabase.*getInstance*().getReference("Sistemas");  
  
 // Inicializar componentes de exibição  
 txtEmail = findViewById(R.id.*txtEmail*);  
 txtDataSistema = findViewById(R.id.*txtDataSistema*);  
 btn\_sair = findViewById(R.id.*btn\_sair*);  
 txtNomeUsuario = findViewById(R.id.*txtNomeUsuario*);  
 btn\_voltar = findViewById(R.id.*btn\_voltar*);  
  
 // Obter o usuário autenticado  
 FirebaseUser user = mAuth.getCurrentUser();  
 if (user != null) {  
 // Exibir o e-mail do usuário logado  
 String email = user.getEmail();  
 txtEmail.setText(" " + email);  
  
 // Exibir nome do usuário  
 String nomeUsuario = user.getDisplayName();  
 if (nomeUsuario != null) {  
 txtNomeUsuario.setText(" " + nomeUsuario);  
 } else {  
 txtNomeUsuario.setText(" Não disponível");  
 }  
  
 String userId = user.getUid();  
 recuperarSistema(userId);  
 } else {  
 Toast.*makeText*(this, "Usuário não autenticado.", Toast.*LENGTH\_SHORT*).show();  
 }  
  
  
 btn\_voltar.setOnClickListener(v -> {  
 Intent intent = new Intent(Form\_User.this, Form\_home.class);  
 startActivity(intent);  
 finish();  
 });  
 btn\_sair.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 FirebaseAuth.*getInstance*().signOut();  
 Intent intent = new Intent(Form\_User.this, FormLogin.class);  
 startActivity(intent);  
 }  
 });  
 }  
  
  
 private void recuperarSistema(String userId) {  
 // Buscar dados do sistema do usuário no Firebase  
 mDatabase.child(userId).addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
 if (dataSnapshot.exists()) {  
 for (DataSnapshot sistemaSnapshot : dataSnapshot.getChildren()) {  
 Sistema sistema = sistemaSnapshot.getValue(Sistema.class);  
 if (sistema != null) {  
 // Exibir data de criação do sistema  
 txtDataSistema.setText(" " + sistema.data);  
 }  
 }  
 } else {  
 // Caso não exista sistema para o usuário  
 Toast.*makeText*(Form\_User.this, "Nenhum sistema encontrado para este usuário.", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onCancelled(DatabaseError databaseError) {  
 Toast.*makeText*(Form\_User.this, "Erro ao recuperar dados do sistema.", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
  
 // Classe para representar um Sistema  
 public static class Sistema {  
 public String data;  
  
 public Sistema() {  
 // Construtor padrão necessário para o Firebase  
 }  
  
 public Sistema(String data) {  
 this.data = data;  
 }  
 }  
}

**FormSobreNos**

import android.content.Intent;  
import android.net.Uri;  
import android.os.Bundle;  
import android.view.View;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.AppCompatButton;  
import androidx.core.graphics.Insets;  
import androidx.core.view.ViewCompat;  
import androidx.core.view.WindowInsetsCompat;  
  
public class FormSobrenos extends AppCompatActivity {  
  
 private AppCompatButton btn\_voltar;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_form\_sobrenos*);  
  
 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;  
 });  
  
 btn\_voltar = findViewById(R.id.*btn\_voltar*);  
 btn\_voltar.setOnClickListener(v -> {  
 Intent intent = new Intent(FormSobrenos.this, Form\_home.class);  
 startActivity(intent);  
 finish();  
 });  
  
 // Configurando o clique do botão btnsaibamais  
 findViewById(R.id.*btn\_saibamais*).setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intent = new Intent();  
 intent.setAction(Intent.*ACTION\_VIEW*);  
 intent.addCategory(Intent.*CATEGORY\_BROWSABLE*);  
 intent.setData(Uri.*parse*("https://situt-tpg.web.app/sobre.html"));  
 startActivity(intent);  
 }  
 });  
 }  
}