

UNIVERSIDADE ANHEMBI MORUMBI
COMPUTAÇÃO MÓVEL – PROFESSOR RICARDO DE SOUZA JACOMINI

RAFAEL PEREIRA DE ALMEIDA FILHO - RA: 20324573

RAFAEL STIVANELLI - RA: 20608546

RODRIGO LUIS PIRES – RA: 20604496

VITOR GIAMMELLA RA: 20614467

TIMÓTEO KIWON KANG - RA:20602849

PROJETO– APLICATIVO PARA RECEITAS GASTRONOMICAS

São Paulo

2017

1. EXPLICAÇÃO DAS FUNCIONALIDADES

As funcionalidades desenvolvidas para este projeto de aplicativos são:

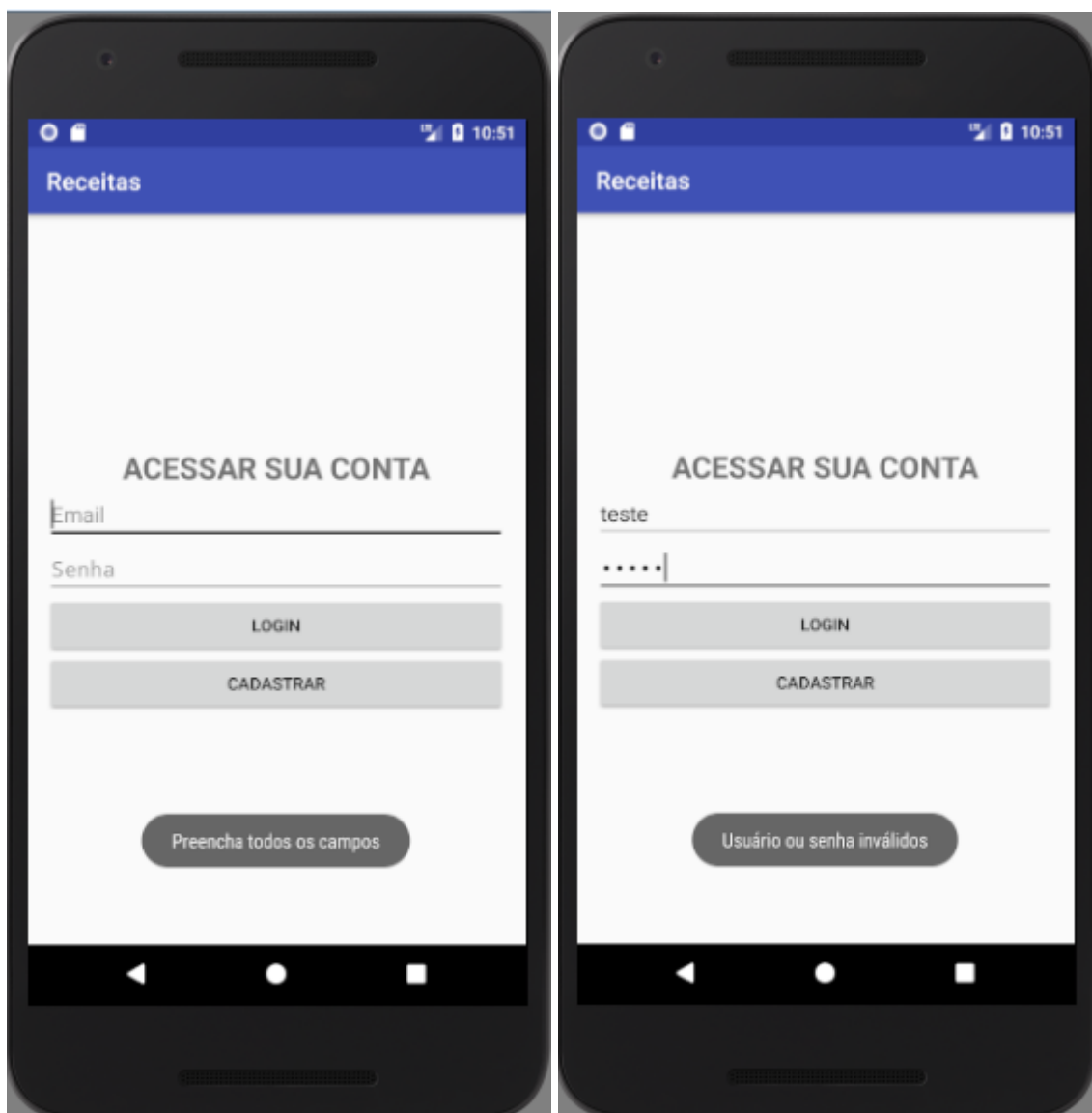
- a) Acesso ao aplicativo através de chaves de segurança: Email e Senha:
Nessa funcionalidade, teremos a activity inicial do projeto, onde partira de uma tela de requisição de preenchimento do usuário com o seu Email e Senha registrado no aplicativo.
E cadastro de novos clientes: Nessa funcionalidade, teremos a activity de registro, onde um novo usuário deverá preencher alguns de seus dados exigidos pelo aplicativo para a criação de uma conta.
- b) Interação e apresentação de opções selecionáveis (Drawer) do aplicativo:
Nessa funcionalidade, teremos a activity do drawer, onde permitirá o usuário a visualizar uma scroll de opções para interação com o aplicativo encaminhando o usuário para as próximas funcionalidades do aplicativo.
- c) Listas de receitas - Comida, Sobremesa e Bebidas (batidas):
Teremos a divisão de 3 categorias de receitas, onde ao selecionar uma das opções, mostrara a lista de receitas que estão gravadas no aplicativo para a visualização das receitas.
- d) Mural– Uma funcionalidade que possibilita os usuários a lerem mensagens postadas por outros usuários, onde solicitam por receitas novas que não se encontram no aplicativo ainda.
- e) Send – Essa funcionalidade permitirá que o usuário poste mensagens no mural para solicitar receitas que ainda não existem no aplicativo.
- f) Receitas ilustradas – Teremos a chamada de imagens para ilustrar cada receita deste aplicativo.
- g) Ranque receitas – Teremos a funcionalidade que permite cada usuário a visualizar as pontuações das receitas, assim, permitindo os demais usuários a correr atrás de receitas com mais estrelas
- h) Algumas mensagem que surgem durante a utilização do sistema (Toast).

1.1 APRESENTAÇÃO DO APLICATIVO

Ao decorrer dos tempos, nos deparamos com situações em que as refeições preparadas em nossas residências ou restaurantes se tornam algo muito enjoativo e a procura por novas opções não são simples como muitos imaginam. Para resolver essa dificuldade, decidimos desenvolver uma solução para esse tipo de problema.

Um aplicativo de Receitas com o intuito de fornecer uma maneira simples, rápida e portátil de se obter um livro de Receitas Completo para preparar as melhores refeições em qualquer local pelo através do smartphone, oferecendo opções para as difíceis escolhas de um prato, sobremesa e bebidas de cada dia.

O aplicativo se inicia com a solicitação de acesso através de um Email e Senha registrados, sua MainActivity (PrincipalActivity) possui a funcionalidade de permitir o acesso com segurança ao aplicativo, contém as ações de Login (acessar) e Cadastrar. Quaisquer tentativas de acessar sem um cadastro, resultará no alerta de cadastro inválido.



Em situações de ausência de uma conta, selecionamos a opção “Cadastrar”, ela encaminhará o usuário para a Activity de cadastro (CadastroDeUsuariosActivity), onde oferecem os campos preenchíveis como: Apelido, Email, Senha e repetir a senha.



The image displays two side-by-side screenshots of a mobile application interface for user registration. Both screens have a blue header bar with the text "Receitas". The main title "CADASTRO DE USUARIO" is centered above four input fields labeled "Apelido", "Email", "Senha", and "Repita a senha". Below the fields are two buttons: "CADASTRAR" and "CANCELAR". In the left screenshot, the fields are empty and the buttons are visible. In the right screenshot, a rounded rectangular button with the text "Preencha todos os campos" is positioned below the "CANCELAR" button, indicating an error state where all fields must be filled.

Caso pressionado o botão “Cadastrar”, sem preencher os campos, retornara com mensagem de erro: “Preencha todos os campos”.
E caso o usuário pressione o botão: “Cancelar”, retornará para a página inicial de Login.

Para uma segurança maior com os campos chaves como o Email e Senha, o aplicativo possui padrões que indicam erros ou informações incorretos durante o preenchimento, como:

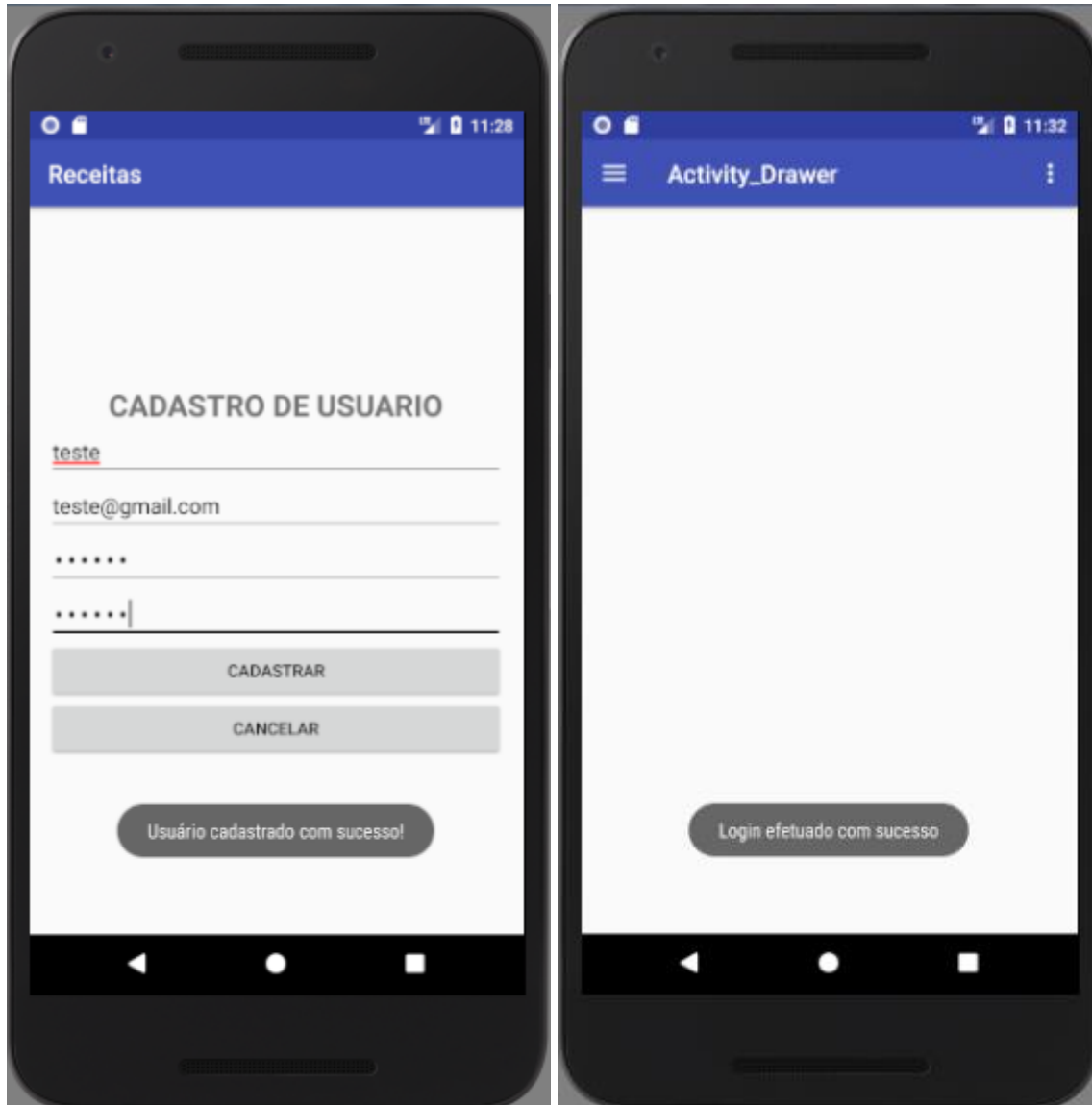
- erros de registrar Email pela ausência de “@” e “.com”;
- erros quando os campos “senha” e “repita a senha” estiverem diferentes;
- erros quando a senha possuir menos que 6 dígitos (não ilustrado a baixo).



Após preencher dados corretamente, pressionamos o botão “Cadastrar”, logo uma mensagem alertará o cadastro com sucesso. Assim que cadastrado pressione o botão de voltar do celular para retornar para a tela de Login novamente e preencha a sua conta cadastrada.

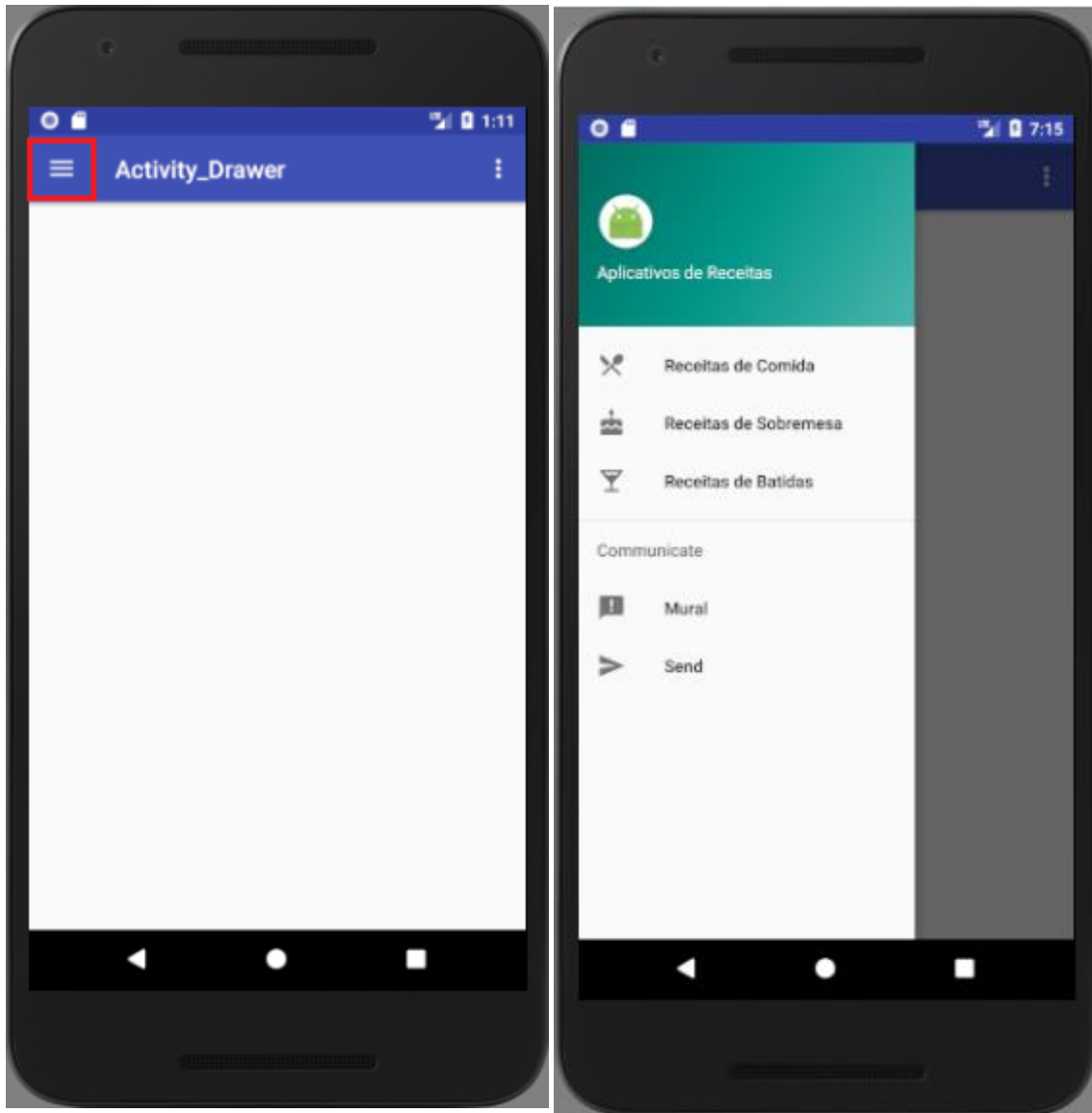
Ao inserir sua conta corretamente e pressionar o botão “Login”, será encaminhado para dentro do aplicativo (Activity_Drawer) com uma mensagem de login efetuado com sucesso.

Assim, iniciamos a interação com o aplicativo de Receitas.



Dentro do Drawer do aplicativo, selecionamos o botão de 3 linhas para que o aplicativo nos mostre as opções de interações disponíveis do aplicativo, que são:

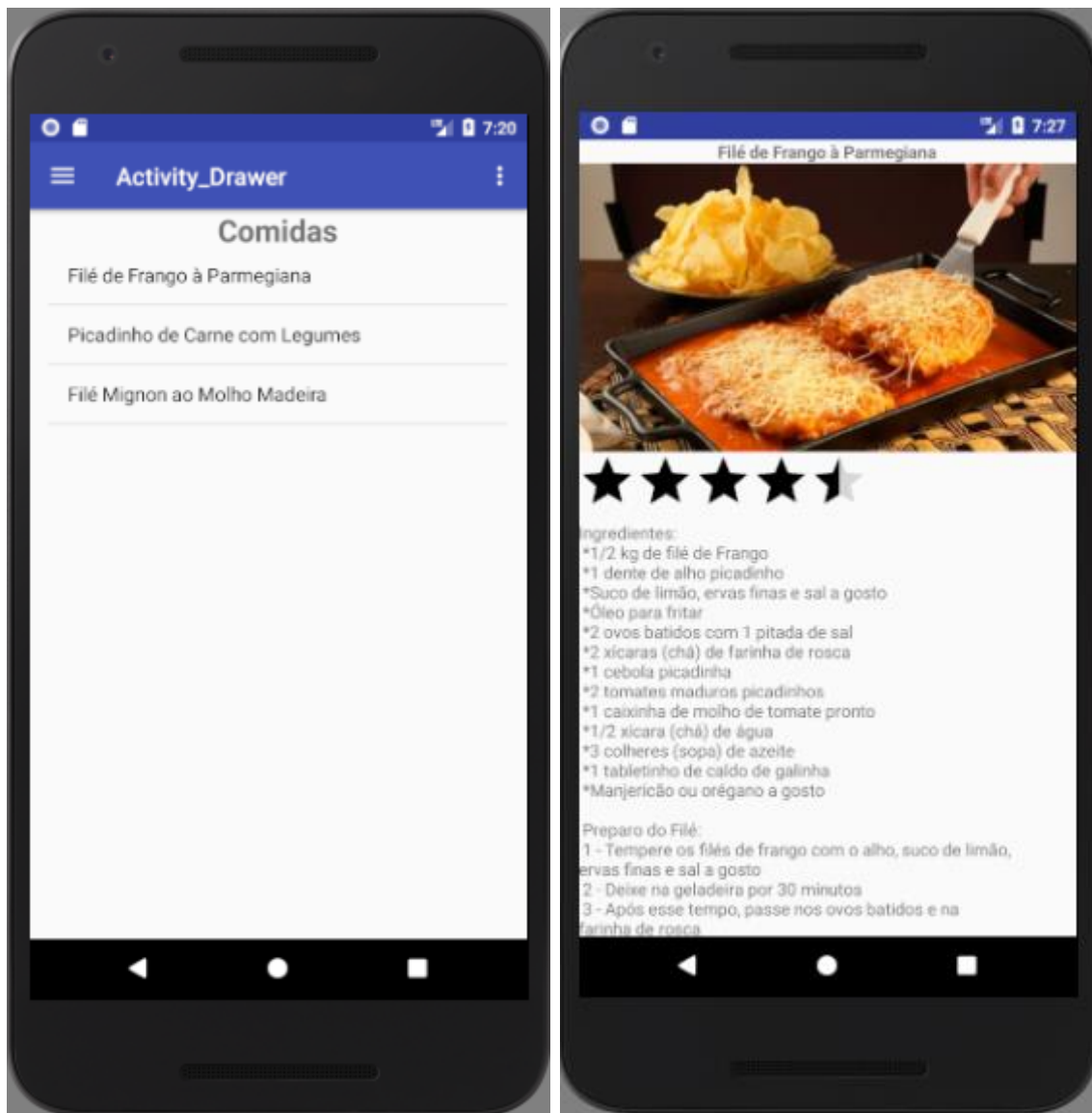
- Receitas de Comida;
- Receitas de Sobremesa;
- Receitas de Batidas (Drinks);
- Mural;
- Send.



*Informação técnica: Todas as opções selecionáveis do drawer são uma Fragment que se sobrepõem na “content_activity__drawer.xml”, que seria o espaço em branco que se encontra no fundo desta tela. *

Simulemos que o usuário seleciona a opção “Receitas de Comida”, o sistema fornecerá uma “ListView” com os pratos disponíveis pelo aplicativo.

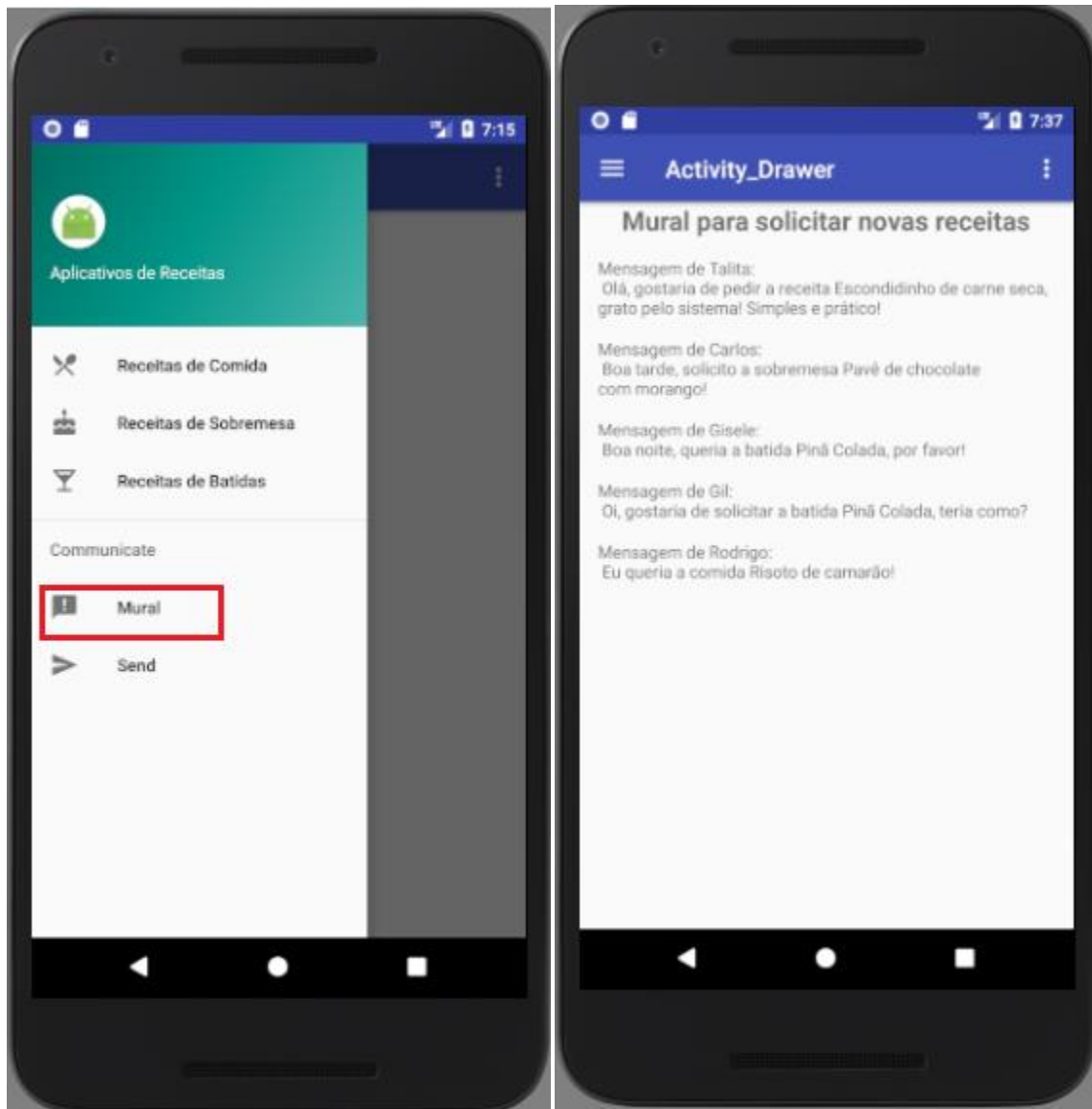
Nesta lista, foi selecionado a primeira opção, “Filé de Frango à Parmegiana”, assim o sistema sai de uma fragmente e faz a chamada e encaminhamento para uma “Activity” chamada: “ActivityReceitaParmegiana” e sua layout “Activity_receita_parmegiana.xml”, onde se encontram uma imagem, estrelas de pontuação e sua receita digitada logo a baixo.



Após a utilização, o usuário poderá pressionar o botão do próprio celular para voltar para o Drawer e suas opções de escolha.

*OBS: Como as próximas opções da drawer, como: Receitas de Sobremesa e Batidas, possuem a mesma estrutura de programação, não citaremos elas neste documento para evitar repetições do mesmo conteúdo. *

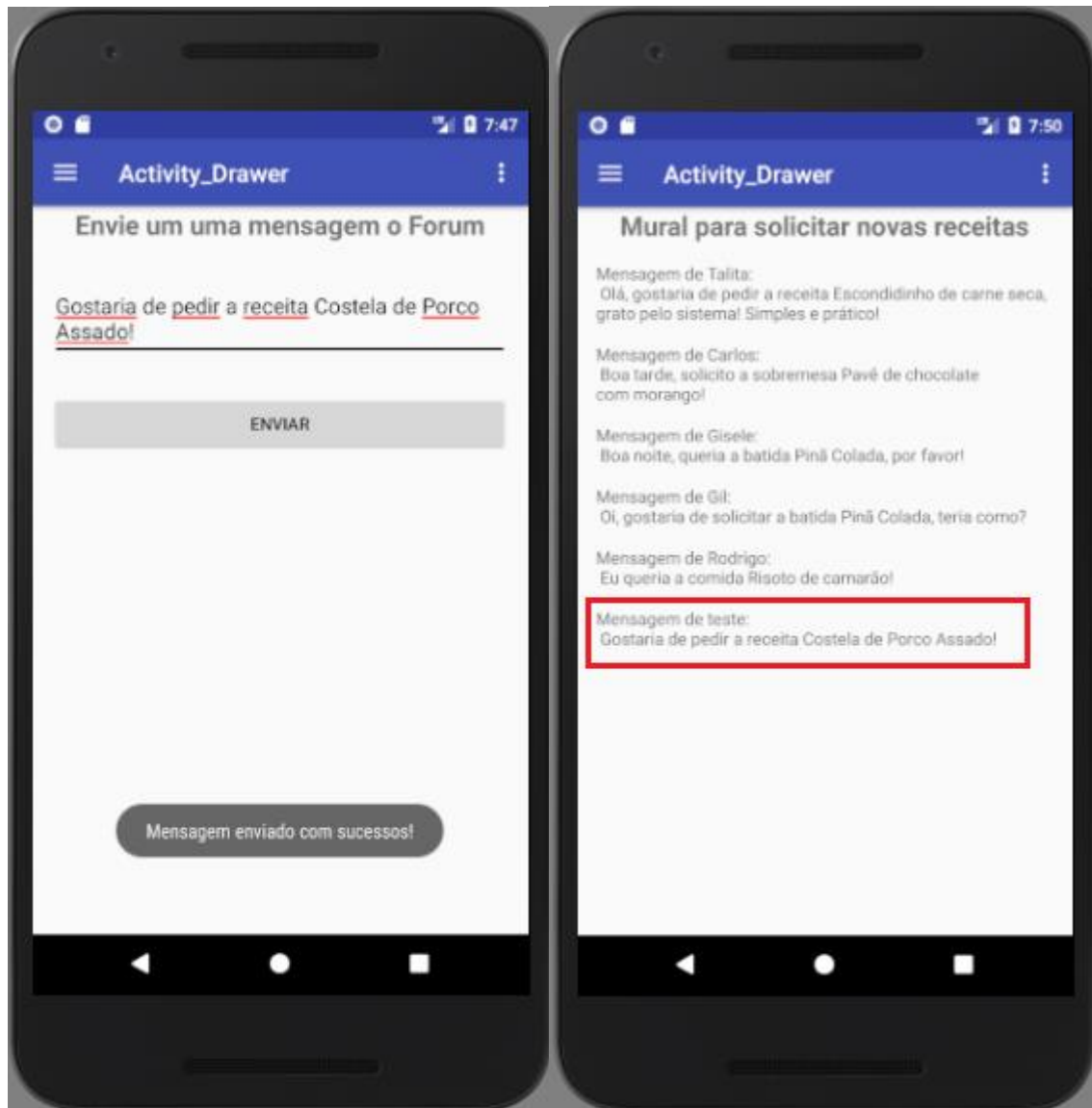
Selecione a opção Mural do Drawer, nela poderemos visualizar mensagens deixadas por alguns usuários que procuram por receitas que não constam no aplicativo para que o administrador do aplicativo possa acrescentar mais receitas recomendadas pelos usuários do aplicativo.



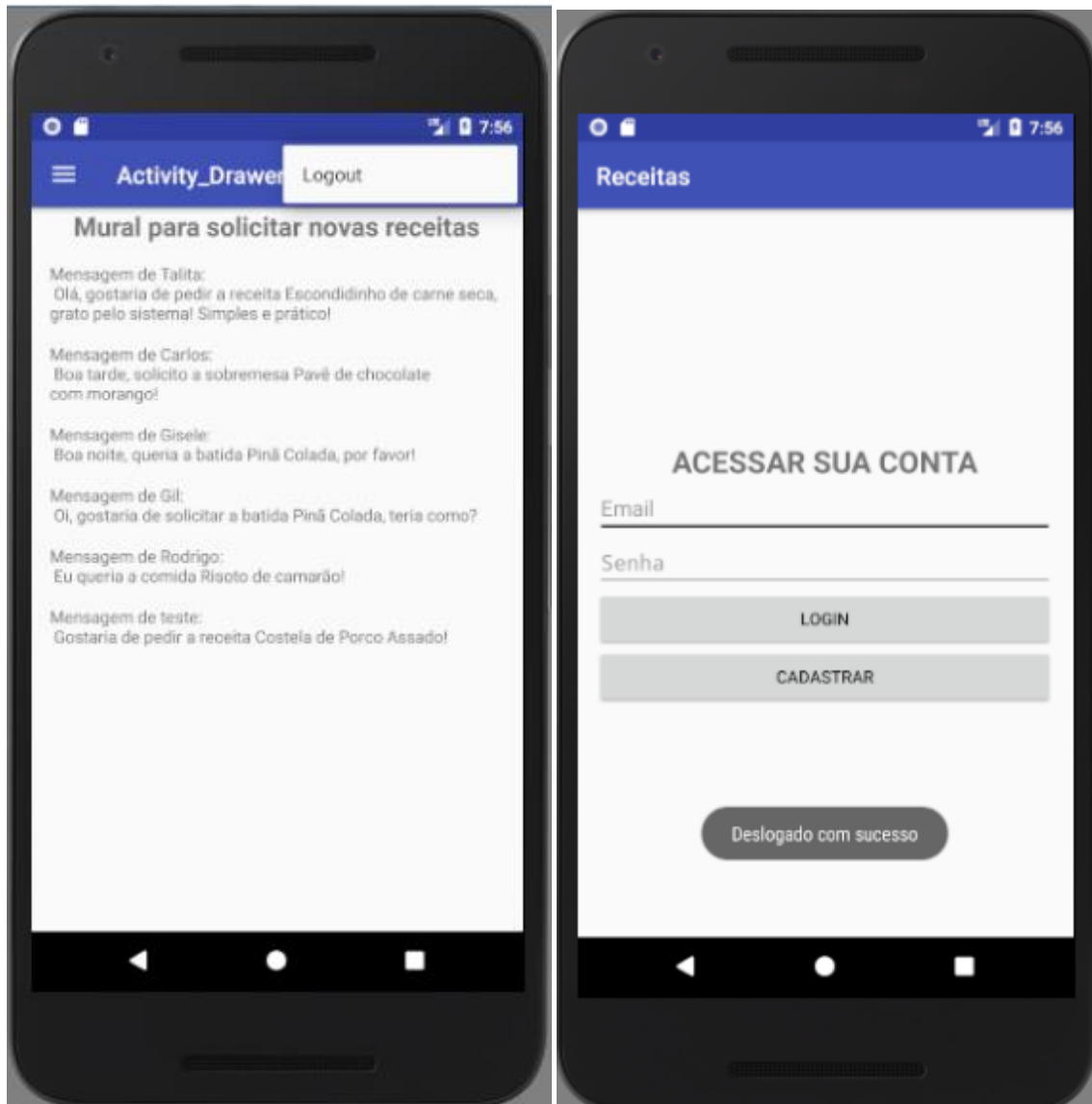
E assim, iremos para a última opção do aplicativo e do drawer, onde o usuário selecionará a opção “Send”, onde será possível que o usuário digite mensagens de solicitação de uma nova receita que não se encontra nas opções selecionáveis do usuário.

Após digitado a mensagem, o usuário precionará o botão de “Enviar” para que a mensagem seja enviada ao Mural do aplicativo, uma mensagem Toast surgirá para indicar que a mensagem foi enviada com sucesso.

Ao voltarmos para a opção Mural, encontraremos a mensagem nova salva pela conta teste que criamos no começo da apresentação do aplicativo.



Por fim, o usuário usará o “Logout” para sair do aplicativo e encerrar seu acesso ao sistema. Clicando na opção de “três bolinhas” no canto superior direito e o “Logout”, pressionado, o usuário será encaminhado para a tela inicial de “Login e Cadastro” e uma mensagem em “Toast” será exibida.



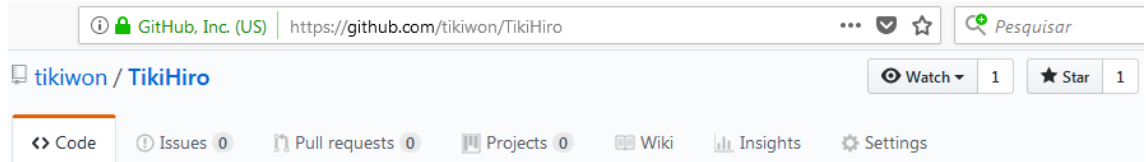
*Assim, concluímos a apresentação e utilização do aplicativo de “Livro de Receitas”.

2. USO E DOCUMENTAÇÃO DO CONTROLE DE VERSÃO (GITHUB)

Algumas das versões mais cruciais aplicativo foram enviados e postados ao GITHUB:

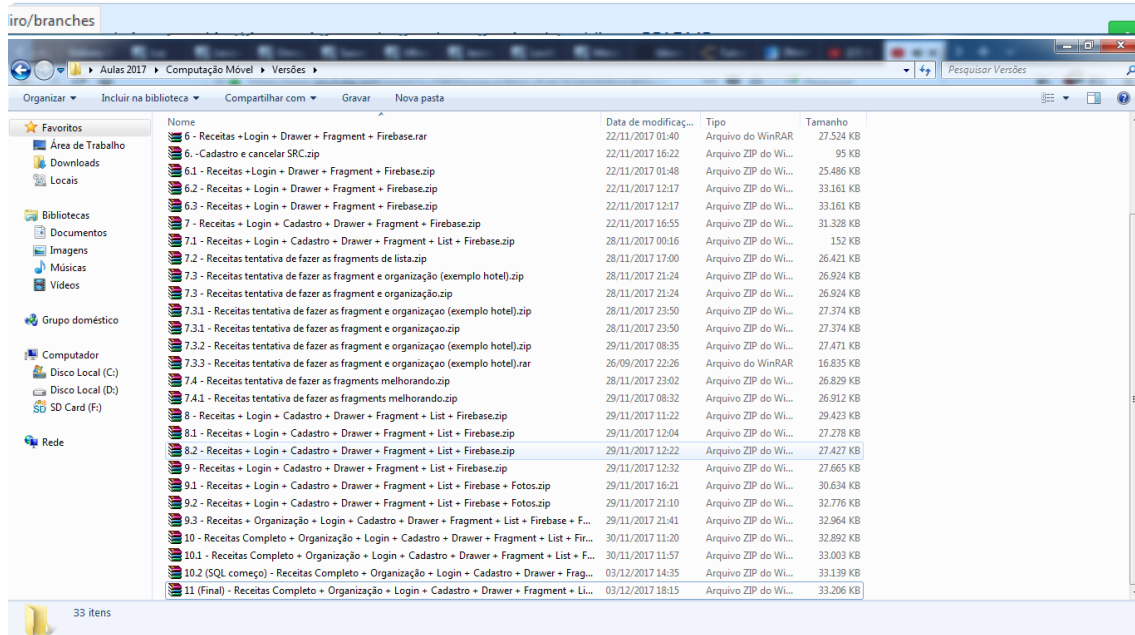
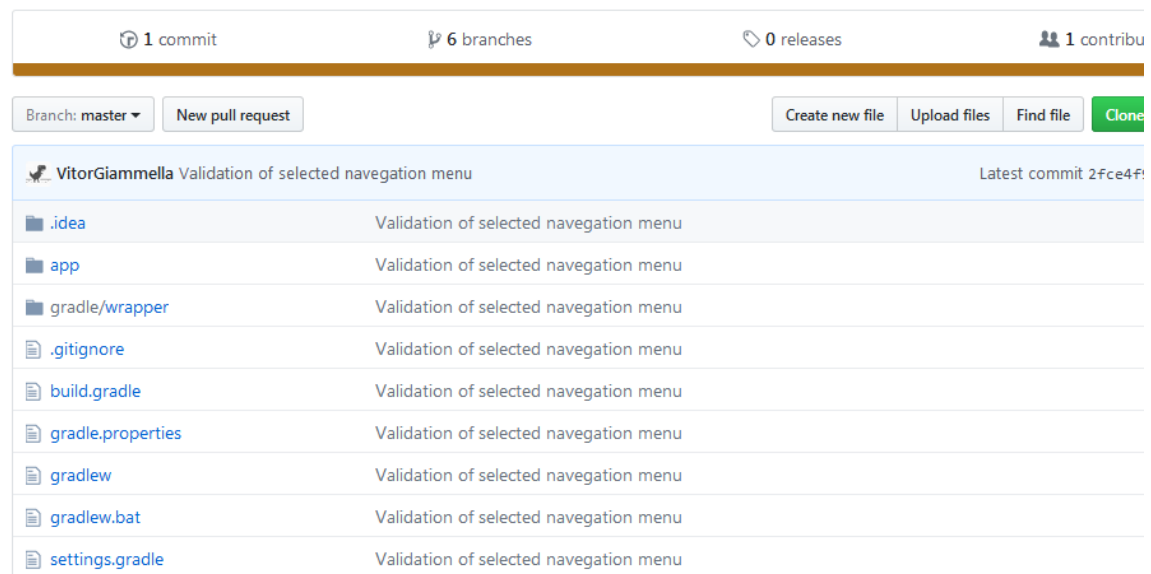
<https://github.com/tikiwon/TikiHiro>

Para a finalização do projeto, foram criados 33 arquivos “.zip” e “.rar” para a segurança e preservação do aplicativo desenvolvido neste projeto.



No description, website, or topics provided.

[Add topics](#)



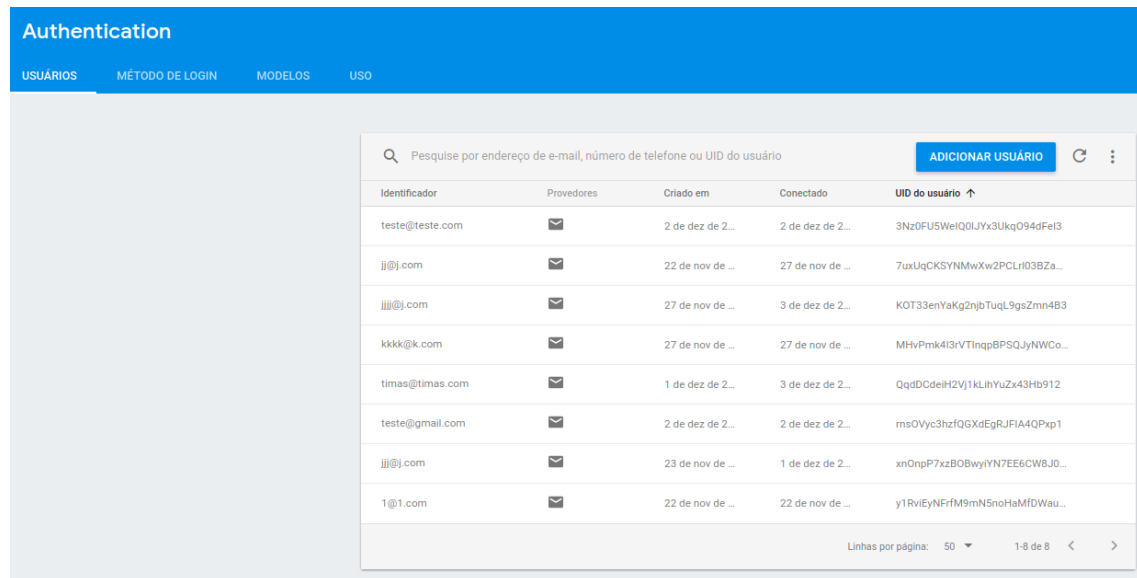
3. USO E DOCUMENTAÇÃO DE BANCO DE DADOS (PERSISTÊNCIA)

Para a utilização do banco de dados, foi desenvolvido um **FIREBASE**.

Possui padrões de receber novos usuários e autenticar usuários existentes.

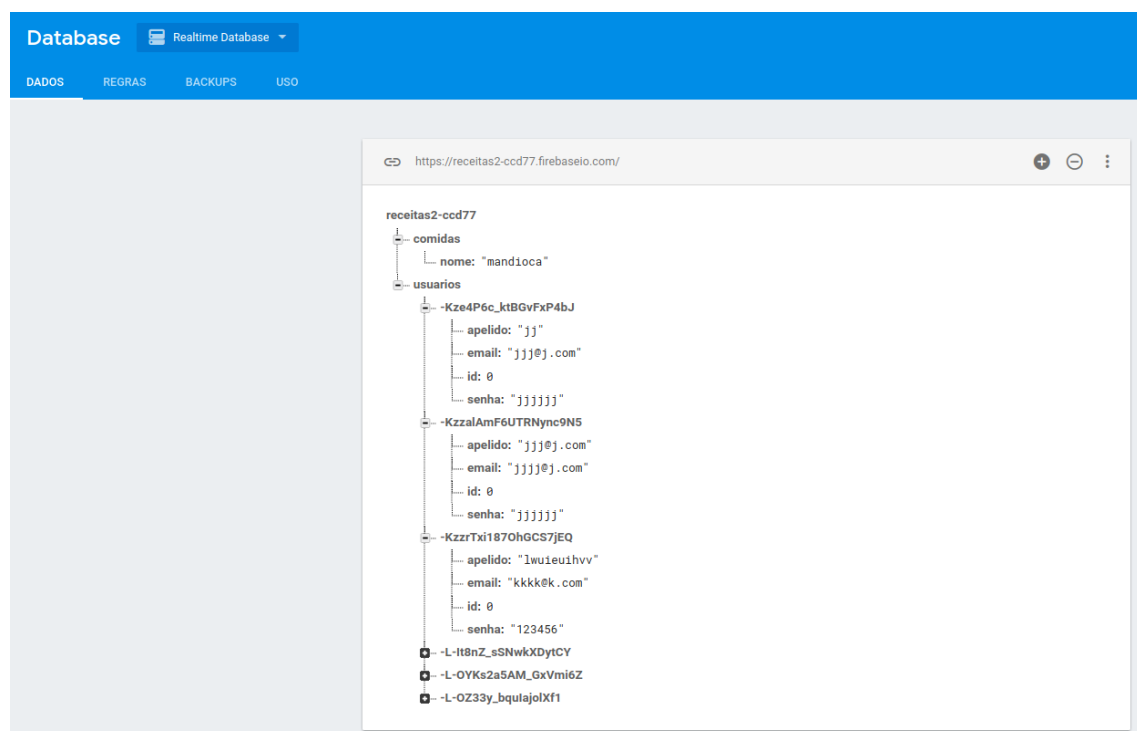
Configurado no endereço:

<https://console.firebase.google.com/project/receitas2-ccd77/database>



The screenshot shows the 'Authentication' section of the Firebase console. It features a search bar at the top with the text 'Pesquise por endereço de e-mail, número de telefone ou UID do usuário'. Below the search bar is a table with the following columns: 'Identificador', 'Provedores', 'Criado em', 'Conectado', and 'UID do usuário'. The table contains 10 rows of user data. At the bottom right, there is a pagination control showing 'Linhas por página: 50' and '1-8 de 8'.

Identificador	Provedores	Criado em	Conectado	UID do usuário
teste@teste.com		2 de dez de 2...	2 de dez de 2...	3Nz0FU5WeiQ0IJYx3Ukq094dFeI3
jj@j.com		22 de nov de ...	27 de nov de ...	7uxUqCKSYNMwXw2PCLr03BZa...
jjj@j.com		27 de nov de ...	3 de dez de 2...	KOT33enYaKg2nbTuqL9gsZmn4B3
kkkk@k.com		27 de nov de ...	27 de nov de ...	MHvPmk4I3rVTInqpBPSQJyNwCo...
timas@timas.com		1 de dez de 2...	3 de dez de 2...	QqdDCdeIH2Vj1kLhYuZx43Hb912
teste@gmail.com		2 de dez de 2...	2 de dez de 2...	rmsOVyc3hzIQGXdeEgRJFIA4QPp1
jjj@j.com		23 de nov de ...	1 de dez de 2...	xnOnpP7xzBOBwyiYN7EE6CWBJ0...
1@1.com		22 de nov de ...	22 de nov de ...	y1RviEyNfrfM9mN5noHaMFDWau...

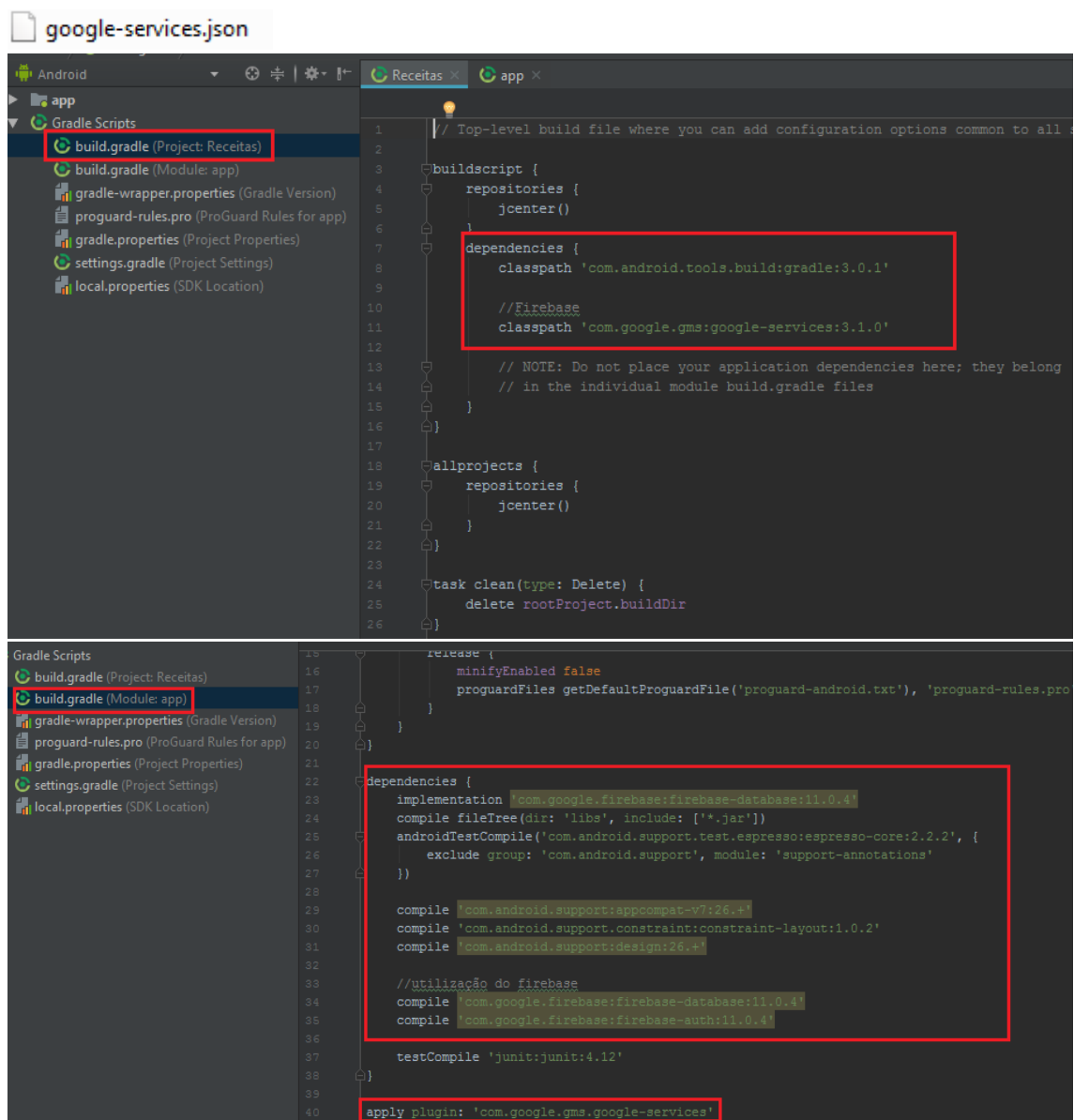


Será mostrado em sala.

4. DOCUMENTAÇÃO DOS CÓDIGOS (PARÂMETROS DE ENTRADA E SAÍDA, E DESCRIÇÃO DA FUNÇÃO)

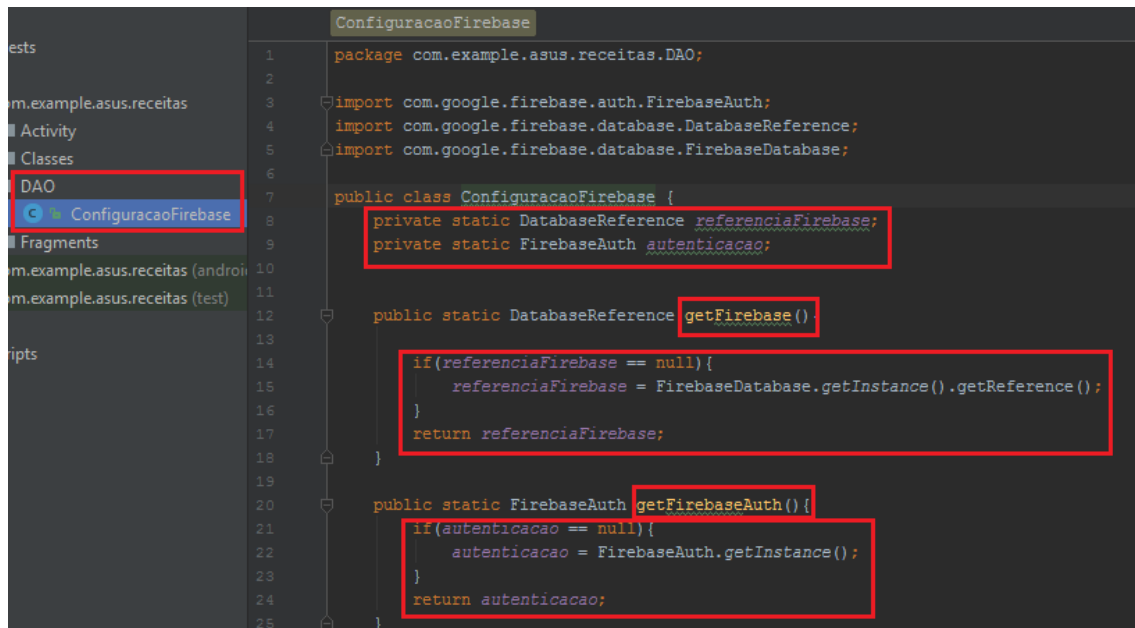
1 – CHAMADA DO FIREBASE, CONSULTA DE EMAIL E SENHA E CADASTRO DE CONTA.

CONFIGURANDO FIREBASE: Para a ligação do banco de dados Firebase com o projeto, foi necessário gerar o arquivo: “google-services.json” (que surge ao configurar o Firebase da google) para podermos configurar as dependências do “Build.gradle” (Project: Receitas e Module: app):



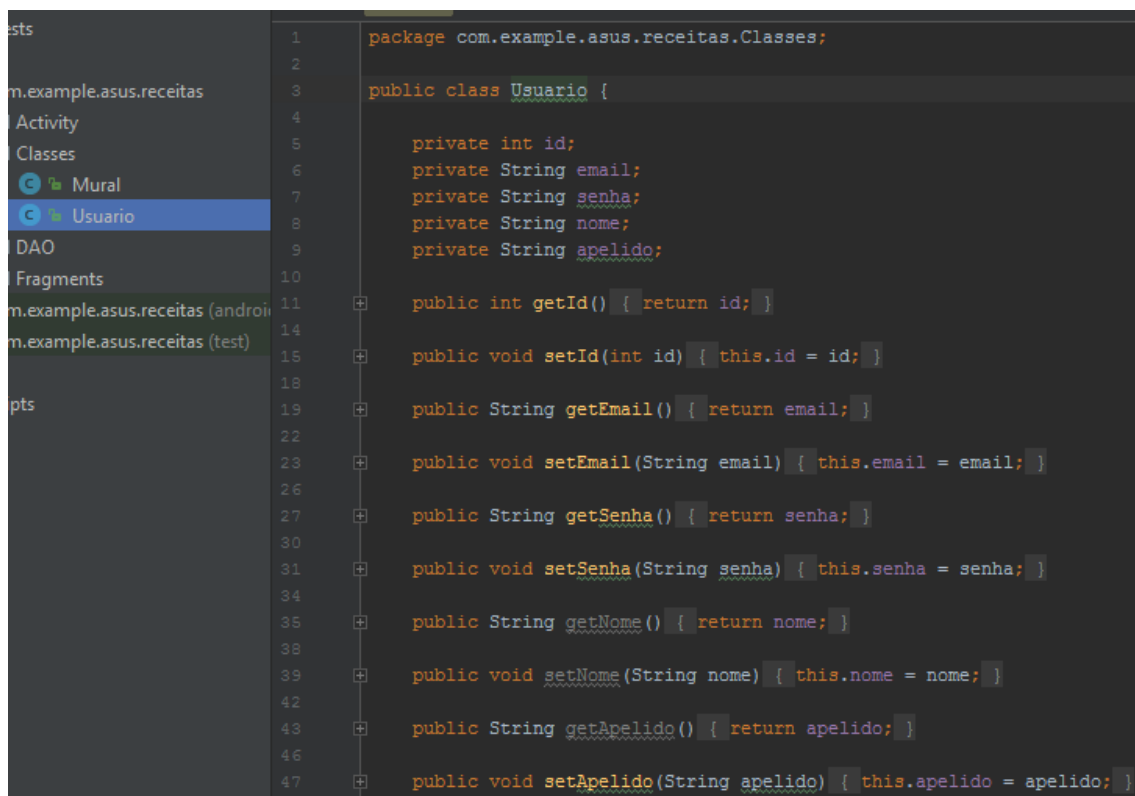
Assim que configurados corretamente, fazemos um teste na comunicação. Tendo resultados positivos, criamos mais 2 classes que utiliza as dependências:

“ConfiguracaoFirebase.class” no Package “DAO”, com os parâmetros “getFirebase()” e “getFirebaseAuth” e seus “returns” programados.



```
1 package com.example.asus.receitas.DAO;
2
3 import com.google.firebase.auth.FirebaseAuth;
4 import com.google.firebase.database.DatabaseReference;
5 import com.google.firebase.database.FirebaseDatabase;
6
7 public class ConfiguracaoFirebase {
8     private static DatabaseReference referenciaFirebase;
9     private static FirebaseAuth autenticacao;
10
11     public static DatabaseReference getFirebase() {
12         if (referenciaFirebase == null) {
13             referenciaFirebase = FirebaseDatabase.getInstance().getReference();
14         }
15         return referenciaFirebase;
16     }
17
18     public static FirebaseAuth getFirebaseAuth() {
19         if (autenticacao == null) {
20             autenticacao = FirebaseAuth.getInstance();
21         }
22         return autenticacao;
23     }
24 }
```

E a Classe: “Usuario.class” na Package Classes, permite que as funcionalidade de login e cadastro tenham seus parâmetros de entrada e saída, ou seja, os getters e setters que serão utilizados nas activitys: PrincipalActivity (onde o usuário preenche seu email e senha registrado no banco de dados, então o sistema busca pelas cadastro em seu banco) e o CadastroUsuarioActivity (onde será feito um novo cadastro).



```
1 package com.example.asus.receitas.Classes;
2
3 public class Usuario {
4
5     private int id;
6     private String email;
7     private String senha;
8     private String nome;
9     private String apelido;
10
11     public int getId() { return id; }
12
13     public void setId(int id) { this.id = id; }
14
15     public String getEmail() { return email; }
16
17     public void setEmail(String email) { this.email = email; }
18
19     public String getSenha() { return senha; }
20
21     public void setSenha(String senha) { this.senha = senha; }
22
23     public String getNome() { return nome; }
24
25     public void setNome(String nome) { this.nome = nome; }
26
27     public String getApelido() { return apelido; }
28
29     public void setApelido(String apelido) { this.apelido = apelido; }
30 }
```

LOGIN: Na nossa MainActivity (nomeado como PrincipalActivity), encontramos as chamadas de cada “EditText” e “Button” nessa Activity.

```

public class PrincipalActivity extends AppCompatActivity {

    private FirebaseAuth autenticacao;
    private EditText edtEmailLogin;
    private EditText edtSenhaLogin;
    private Button btnLogin;
    private Button btnCadastro;
    private Usuario usuario;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);

        edtEmailLogin = (EditText) findViewById(R.id.edtEmail);
        edtSenhaLogin = (EditText) findViewById(R.id.edtSenha);
        btnLogin = (Button) findViewById(R.id.btnLogin);
        btnCadastro = (Button) findViewById(R.id.btnCadastro);
    }
}

```

Logo abaixo, foi dada ao botão: “Login” a função “setOnClickListener”, onde coletará os campos preenchidos “edtEmailLogin” e “edtSenhaLogin”, se estiverem preenchidos, ela iniciará o passo seguinte: “validarLogin”. E se os campos estiverem vazios, o sistema retornará com uma Toast relatando para que preencha os campos vazios

```

btnLogin.setOnClickListener((view) -> {
    if(!edtEmailLogin.getText().toString().equals("") && !edtSenhaLogin.getText().toString().equals("")){
        usuario = new Usuario();
        usuario.setEmail(edtEmailLogin.getText().toString());
        usuario.setSenha(edtSenhaLogin.getText().toString());

        validarLogin();
    }else{
        Toast.makeText(context: PrincipalActivity.this, text: "Preencha todos os campos", Toast.LENGTH_SHORT).show();
    }
});

btnCadastro.setOnClickListener((v) -> { abrirTelaCadastro(); });

```

Ao entrar em “validarLogin()”, autenticação do banco de dados entra em ação a procura pelo email e senha registrado no banco de dados, caso encontrados, o sistema encaminhará para a “abrirTelaDrawer” com uma mensagem: “Login efetuado com sucesso”. Se caso usuário não for encontrado, indicará com uma mensagem de erro: “Usuário ou senha inválidos”.

```

private void validarLogin() {
    autenticacao = ConfiguracaoFirebase.getFirebaseAuth();
    autenticacao.signInWithEmailAndPassword(usuario.getEmail().toString(), usuario.getSenha().toString()).addOnCompleteListener((task) -> {
        if(task.isSuccessful()){
            abrirTelaDrawer();

            Toast.makeText(context: PrincipalActivity.this, text: "Login efetuado com sucesso", Toast.LENGTH_SHORT).show();
        }else{
            Toast.makeText(context: PrincipalActivity.this, text: "Usuário ou senha inválidos", Toast.LENGTH_SHORT).show();
        }
    });
}

```

Caso a validação da conta foi com sucesso, o sistema utiliza o “Intent” para sair do “PrincipalActivity” e encaminha para a próxima activity: “Activity_Drawer” ou se o cliente não possui uma conta, ao clicar no Button “Registrar”. Ele é encaminhado para a activity: “CadastroUsuarioActivity”.


```

private void abrirTelaDrawer(){
    Intent intent = new Intent( packageContext: PrincipalActivity.this, Activity_Drawer.class);
    startActivity(intent);
}

private void abrirTelaCadastro(){
    Intent intent = new Intent( packageContext: PrincipalActivity.this, CadastroUsuarioActivity.class);
    startActivity(intent);
}

```

CADASTRO DE USUÁRIO: Em “CadastroUsuarioActivity”, encontramos o cadastro que irá para o nosso Banco de Dados e algumas exceções de erro que possam surgir ao longo do cadastro de um usuário.

Nesta Activity de cadastro, o campo chave (crucial) é a SENHA.

Sendo os padrões de exceções:

- Primeiro “if”, os campos senha1 e senha2 deve ser iguais;
- Segundo “if”, os dois campos precisam ser maiores ou iguais a 6 dígitos.

E outra exceção para quando os campos não estiverem todos preenchidos.

Vale lembrar que todas as exceções possuem uma mensagem (Toast) ao enquadrarem em uma das exceções.

```

//botao cadastrar
btnCadastrar.setOnClickListener((view) -> {

    //valida se todos os campos estão preenchidos
    if(!email.getText().toString().equals(""))
        && !senha1.getText().toString().equals("")
        && !senha2.getText().toString().equals("")
        && !apelido.getText().toString().equals("")) {

        //se tudo estiver preenchido
        if (senha1.getText().toString().equals( senha2.getText().toString() )) {
            if (senha1.getText().toString().length() >=6
                && senha2.getText().toString().length() >= 6) {

                usuario = new Usuario();

                usuario.setEmail( email.getText().toString() );
                usuario.setSenha( senha1.getText().toString() );
                usuario.setApelido( apelido.getText().toString() );

                cadastrarUsuario();

            } else {
                Toast.makeText( context: CadastroUsuarioActivity.this, text: "A senha precisa ter pelo menos 6 caracteres", Toast.LENGTH_LONG ).show();
            }
        } else {
            Toast.makeText( context: CadastroUsuarioActivity.this, text: "Senhas divergentes", Toast.LENGTH_LONG ).show();
        }
    } else{
        Toast.makeText( context: CadastroUsuarioActivity.this, text: "Preencha todos os campos", Toast.LENGTH_LONG ).show();
    }
}

```

E o segundo campo chave (crucial) é o EMAIL.

Sendo os padrões de exceções gravadas no Banco de Dados, utilizando: try, throw e catch:

- O Email deve possuir a String “@” e a String “.com”;
- O Email não poderá existir 2 iguais;
- O Email não pode possuir código vencido ou inválido;
- O Email são sujeitos a erros.

Assim como as Senhas, o Email também possuem mensagens (Toast) de erros ao se enquadrarem alguma das exceções estabelecidas.

```

private void cadastrarUsuario() {

    autenticacao = ConfiguracaoFirebase.getFirebaseAuth();
    autenticacao.createUserWithEmailAndPassword(
        usuario.getEmail(),
        usuario.getSenha()
    ).addOnCompleteListener( activity: CadastroUsuarioActivity.this, (task) - {

        if (task.isSuccessful()) {

            insereUsuario(usuario);

        } else {

            String erroExcecao = "";

            try {
                throw task.getException();
            } catch (FirebaseAuthInvalidCredentialsException e) {
                erroExcecao = "O e-mail digitado é inválido, digite um novo e-mail";
            } catch (FirebaseAuthUserCollisionException e) {
                erroExcecao = "Esse e-mail já está cadastrado!";
            } catch (FirebaseAuthActionCodeException e) {
                erroExcecao = "Authcodeerror"+e;
            } catch (Exception e) {
                erroExcecao = "Erro ao efetuar o cadastro!" + e;
                e.printStackTrace();
            }

            Toast.makeText( context: CadastroUsuarioActivity.this, text: "Erro: " + erroExcecao, Toast.LENGTH_LONG).show();
        }
    });
}

```

Após os campos preenchidos corretamente o Button “Cadastrar” será pressionado, a classe filho “usuários” é executado, para que a gravação da conta seja efetuada no banco de dados. Assim que sucedido uma mensagem (Toast) será exibido, caso contrário, mensagem de erro surgirá.

```

private boolean insereUsuario(Usuario usuario) {

    try {

        reference = ConfiguracaoFirebase.getFirebase().child("usuarios");
        reference.push().setValue(usuario);
        Toast.makeText( context: CadastroUsuarioActivity.this, text: "Usuário cadastrado com sucesso!", Toast.LENGTH_LONG).show();
        return true;

    } catch (Exception e) {
        Toast.makeText( context: CadastroUsuarioActivity.this, text: "Erro ao gravar o usuário!", Toast.LENGTH_LONG).show();
        e.printStackTrace();
        return false;
    }
}

```

2– Drawer e suas ligações entre Fragments.

Drawer: Para o projeto, o drawer foi desenvolvido usando como base o Drawer do próprio Android Studio, implementado sobre ela somente as transições entre as janelas (fragments) e pequenas funcionalidades como a opção: “Logout”.

FRAGMENTS DA DRAWER: Para as transições ao selecionados uma das opções da aba do drawer, fizemos as ligações entre elas através de uma “boolean onNavigationItemSelectedListener” onde os comandos principais com “if” e “else” para alcançar cada Fragments: “getItemId()”, “getSupportFragmentManager()”, “.beginTransaction()”, “.replace” e “.commit()”.

Além disso, todas as fragments são sobrepostas sobre uma “RelativeLayout” com o ID: “mainLayout” que se encontra com uma “.xml” no diretório “res/layout” com o nome: “content_activity__drawer.xml”.

```

public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();

    if (id == R.id.nav_comida) {
        ComidaFragment comidaFragment = new ComidaFragment();
        FragmentManager manager = getSupportFragmentManager();
        manager.beginTransaction()
            .replace(R.id.mainLayout, comidaFragment)
            .commit();

    } else if (id == R.id.nav_sobremesa) {
        SobremesaFragment sobremesaFragment = new SobremesaFragment();
        FragmentManager manager = getSupportFragmentManager();
        manager.beginTransaction()
            .replace(R.id.mainLayout, sobremesaFragment)
            .commit();

    } else if (id == R.id.nav_batidas) {
        BatidaFragment batidaFragment = new BatidaFragment();
        FragmentManager manager = getSupportFragmentManager();
        manager.beginTransaction()
            .replace(R.id.mainLayout, batidaFragment)
            .commit();
    }
}

```

```

<RelativeLayout
    android:id="@+id/mainLayout"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

</RelativeLayout>

```

Vale ressaltar que cada uma das opções (fragments) apresentadas no drawer, são transitadas entre elas com os mesmos métodos citados acima.

3 – Lista de receitas:

LISTA (LISTVIEW) E ARRAYADAPTER: Após selecionado uma das Receitas (por exemplo: Receitas de comida), o aplicativo buscar pela “.xml” e sua “fragmente” correspondente: “fragmente_comida.xml” e “ComidaFragment”.

Explicando primeiramente a composição da “.xml”, encontramos a ListView, onde as receitas entrarão listadas.

```

<ListView
    android:id="@+id/menuComidas"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

```

Para que a ListView obtenham as informações a serem apresentadas é necessário os métodos para isso na Frament: "ComidaFrament".

Utilizamos uma "onCreateView" com uma "inflater.inflate", colocamos os nomes de cada receitas diretamente na fragment e assim, criamos a função importante para esse processo: "ArrayAdapter<string>" com suas "getActivity()", "simple_list_item_1" do "menuComidas", por fim a lista é apresentada como: "listView.setAdapter(listViewAdapter)".

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_comida, container, attachToRoot: false);

    String[] menuComidas = {"Filé de Frango à Parmegiana",
        "Picadinho de Carne com Legumes",
        "Filé Mignon ao Molho Madeira"};

    ListView listView = (ListView) view.findViewById(R.id.menuComidas);

    ArrayAdapter<String> listViewAdapter = new ArrayAdapter<~>(
        getActivity(),
        android.R.layout.simple_list_item_1,
        menuComidas
    );

    listView.setAdapter(listViewAdapter);
}
```

Após configurar a lista, fizemos com que as listas sejam clicáveis, para que assim, faça a chamada de uma outra Fragment ou Activity assim que clicados.

Utilizamos os métodos: "onItemClickListener" na AdapterView e montamos casos de "if" e "else" para a interação.

Dentro deles, colocamos um contador para as Array que parte do "i == 0" e as "Intent" para cada seleção, onde o a fragment passará para uma Activity.class e sua .XML, através do: "getActivity" a activity.class destinada e que é inicializada através da "startActivity(intent)".

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        if (i == 0) {
            Intent intent = new Intent(getActivity(), ActivityReceitaParmegiana.class);
            startActivity(intent);
        } else if (i == 1) {
            Intent intent = new Intent(getActivity(), ActivityReceitaPicadinho.class);
            startActivity(intent);
        } else if (i == 2) {
            Intent intent = new Intent(getActivity(), ActivityReceitaMadeira.class);
            startActivity(intent);
        }
    }
});

// Inflate the layout for this fragment
return view;
}
```

4 – Activities de receitas e XML com imagem, raqueamento e instruções de receita.

Digamos que o usuário selecionou a opção 3 (Filé Mignon ao Molho Madeira) em receitas de comida. Nesta XML, encontramos configurações:

- “ScrollView” para a possibilidade a visualização de receitas cumpridas;

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

- “ImageView” para a ilustração das comidas, feito como “android:src”;

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="239dp"
    android:scaleType="centerCrop"
    android:src="@drawable/madeira"
    tools:layout_editor_absoluteX="0dp" />
```

- “RatingBar” para a visualização de estrelas de cada prato;

```
<RatingBar
    android:id="@+id/estrelas3"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
```

- “TextView” para a visualização da receita.

```
<TextView
    android:id="@+id/receita3"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

*OBS: Esses métodos são implementados em todos os pratos existentes nesse sistema, por exato motivo, não citaremos outras receitas. *

5 – MURAL DE SUJESTÕES DE NOVOS PRATOS.

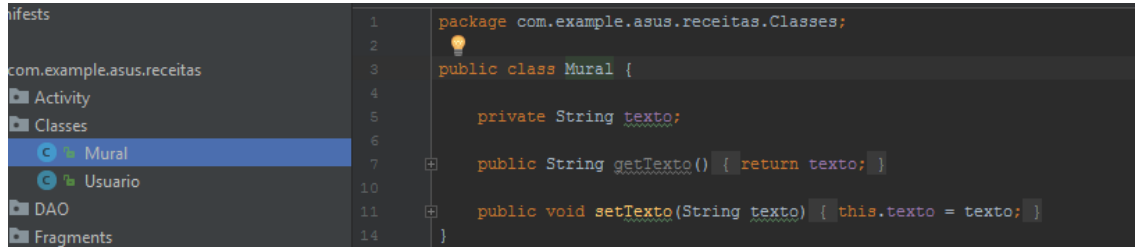
Com relação ao mural, temos a apresentação das mensagens dos usuários que postaram no aplicativo.

```
<TextView
    android:layout_marginTop="15dp"
    android:id="@+id/notification1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

6 – Envio de mensagens para o Banco de dados.

Nesta ultima fragment, possuímos os parâmetros que ligam as informações ao banco de dados através de uma classe criada separadamente, chamada “Mural”, localizada na package

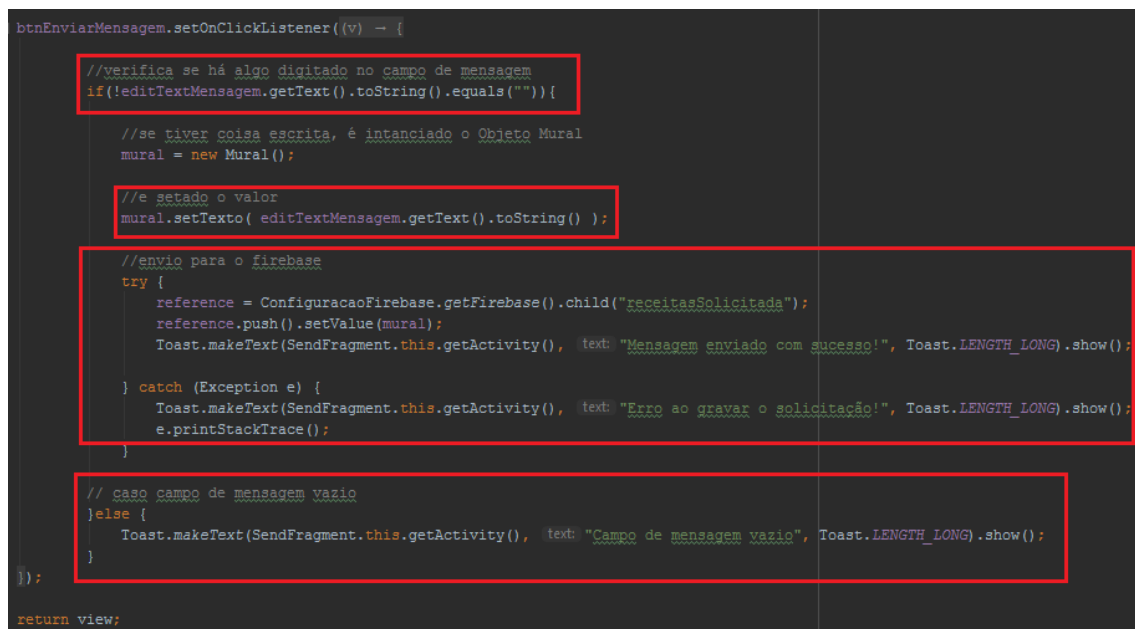
“Classes”, onde se encontramos o “get” e o “set” dos textos desta fragmente.



```
1 package com.example.asus.receitas.Classes;
2
3 public class Mural {
4
5     private String texto;
6
7     public String getText() { return texto; }
8
9     public void setTexto(String texto) { this.texto = texto; }
10
11 }
12
13
14
```

Em sua fragment, utilizamos o “if”, para verificar se o campo “editTextMenagem” está vazia, caso possua algo escrito nos campos, ela prossegue para os campos de “try” e “catch”, onde o envio da mensagem é feita ao banco de dados, nesse processo mensagens “Toast” são mostradas para cada situação, seja se a mensagem foi enviada com sucesso ou erro ao gravar a solicitação.

Por fim, o “if” que analisa se o campo “editTextMesagem” está vazio ou não, fecha com o “else”, para casos em que o campo estiver vazio, relatando uma mensagem “Toast”, solicitando que preencha o campo vazio.



```
btnEnviarMensagem.setOnClickListener((v) - {
    //verifica se há algo digitado no campo de mensagem
    if(!editTextMensagem.getText().toString().equals("")){
        //se tiver coisa escrita, é instanciado o Objeto Mural
        mural = new Mural();
        //e setado o valor
        mural.setTexto( editTextMensagem.getText().toString() );
        //envio para o firebase
        try {
            reference = ConfiguracaoFirebase.getFirebase().child("receitasSolicitada");
            reference.push().setValue(mural);
            Toast.makeText(SendFragment.this.getActivity(), text: "Mensagem enviado com sucesso!", Toast.LENGTH_LONG).show();
        } catch (Exception e) {
            Toast.makeText(SendFragment.this.getActivity(), text: "Erro ao gravar o solicitação!", Toast.LENGTH_LONG).show();
            e.printStackTrace();
        }
    }
    // caso campo de mensagem vazio
    else {
        Toast.makeText(SendFragment.this.getActivity(), text: "Campo de mensagem vazio", Toast.LENGTH_LONG).show();
    }
});
return view;
```

A composição da sua “fragmente_send.xml” desta fragment, existe uma “textEdit” e uma “Button”.



```
<EditText
    android:layout_marginTop="30dp"
    android:id="@+id/editTextMensagem"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:baselineAligned="false"
    android:ems="100"
    android:hint="Mensagem"
    android:inputType="textMultiLine" />
```

```
<Button
    android:id="@+id/btnEnviarMensagem"
    android:layout_marginTop="30dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Enviar" />
```

5. DIAGRAMA DE SEQUÊNCIA E CLASSES, completo do APP.