

ANDROID CLOUD INTEGRATION &amp; DEPLOYMENT

# *Autenticação* COM FIREBASE E ANDROID

HEIDER PINHOLI LOPES



2

**LISTA DE FIGURAS**

Figura 2.1 - Entrada do iFood utilizando provedores externos .....	6
Figura 2.2 - Protótipo do aplicativo a ser desenvolvido integrado com o Firebase....	7
Figura 2.3 - Tela de início do Android Studio .....	8
Figura 2.4 - Seleção de Template do Projeto .....	9
Figura 2.5 - Configuração do seu projeto .....	10
Figura 2.6 - Criação de um novo package.....	15
Figura 2.7 - Nomeando o novo pacote para ui .....	15
Figura 2.8 - Pacote de UI .....	15
Figura 2.9 - Criando um pacote.....	16
Figura 2.10 - Criando um pacote signup .....	16
Figura 2.11 - Criando um pacote login .....	16
Figura 2.12 - Criando um pacote betterfuel.....	16
Figura 2.13 - Estrutura dos pacotes criados.....	17
Figura 2.14 - Criando o LoginFragment.....	17
Figura 2.15 - Definindo o nome do LoginFragment .....	18
Figura 2.16 - Criando o SignUp.....	21
Figura 2.17 - Definindo o nome do SignUpFragment .....	21
Figura 2.18 - Criando o Melhor Combustível.....	24
Figura 2.19 - Definindo o nome do BetterFuelFragment .....	25
Figura 2.20 - Criação do arquivo loading.....	29
Figura 2.21 - Definição do nome do arquivo include_loading.xml .....	30
Figura 2.22 - Android JetPack .....	31
Figura 2.23 - Navigation .....	32
Figura 2.24 - Criando um Android Resource Directory.....	33
Figura 2.25 - Criando o diretório navigation .....	34
Figura 2.26 - Criando o arquivo de navegação .....	34
Figura 2.27 - Definindo o nome do arquivo main_nav_graph.....	34
Figura 2.28 - New Destination e os fragments criados.....	35
Figura 2.29 - Telas no Navigation .....	35
Figura 2.30 - main_navigation_graph.xml com preview dos layouts .....	37
Figura 2.31 - Menu Firebase .....	39
Figura 2.32 - Assistente do Firebase no Android Studio .....	40
Figura 2.33 - Selecionando o Authentication para integração .....	41
Figura 2.34 - Integrando com o Firebase .....	41
Figura 2.35 - Permissão de acesso 1 .....	42
Figura 2.36 - Permissão de acesso 2 .....	42
Figura 2.37 - Arquivo de configuração do Firebase.....	42
Figura 2.38 - Adicionando a dependência via assistente .....	43
Figura 2.39 - Preview das mudanças que serão realizadas para adicionar o Auth ...	43
Figura 2.40 - Selecionando o Authentication no console do Firebase.....	44
Figura 2.41 - Ativando login por e-mail/senha .....	45
Figura 2.42 - Criação de um usuário para teste .....	45
Figura 2.43 - Criação de um novo pacote .....	45
Figura 2.44 - Definição do nome do pacote base .....	46
Figura 2.45 - Criação de uma nova classe .....	46
Figura 2.46 - Criação da BaseFragment .....	46
Figura 2.47 - Criação do pacote models.....	48

Figura 2.48 - Criação de uma nova classe 1 .....	48
Figura 2.49 - Criação de uma nova classe 2 .....	48
Figura 2.50 - Criação de um novo pacote .....	49
Figura 2.51 - Pacote de autenticação.....	49
Figura 2.52 - Pacote de autenticação.....	49
Figura 2.53 - Pacote de autenticação.....	49
Figura 2.54 - Aplicativo solicitando o login .....	52

EXEMPLO

**LISTA DE CÓDIGOS-FONTE**

Código-fonte 2.1 – Arquivo styles.xml .....	12
Código-fonte 2.2 – Arquivo colors.xml.....	13
Código-fonte 2.3 – Arquivo strings.xml.....	14
Código-fonte 2.4 – Arquivo dimens.xml.....	14
Código-fonte 2.5 – Layout xml da tela de Login .....	20
Código-fonte 2.6 – Layout xml da tela de SignUp .....	24
Código-fonte 2.7 – Layout xml da tela de BetterFuelFragment .....	29
Código-fonte 2.8 – Import da biblioteca do Airbnb Lottie.....	29
Código-fonte 2.9 – Definição do nome do arquivo include_loading.xml .....	31
Código-fonte 2.10 – Adicionando a dependência do navigation no projeto.....	33
Código-fonte 2.11 – main_navigation_graph.xml com os layouts das telas .....	36
Código-fonte 2.12 – layout da activity_main.....	38
Código-fonte 2.13 – Método para activity ficar em FullScreen .....	38
Código-fonte 2.14 – Atualização da lib de Autenticação .....	44
Código-fonte 2.15 – Criação de uma nova classe 1 .....	47
Código-fonte 2.16 – Criação de uma nova classe 2 .....	49
Código-fonte 2.17 – Código da classe BaseAuthViewModel 1.....	50
Código-fonte 2.18 – Código da classe BaseAuthViewModel 2.....	51
Código-fonte 2.19 – Código da classe BaseAuthViewModel 3.....	51
Código-fonte 2.20 – 2.59 – Código do LoginViewModel.....	53
Código-fonte 2.21 – Código de resetar a senha 1 .....	55
Código-fonte 2.22 – Código de resetar a senha 2 .....	56
Código-fonte 2.23 – Código para realizar o logout .....	56
Código-fonte 2.24 – Código para realizar o logout e setup das views.....	58

## SUMÁRIO

2 AUTENTICAÇÃO COM FIREBASE E ANDROID .....	6
2.1 Introdução .....	6
2.2 O projeto Calcula Flex .....	7
2.3 Desenvolvimento do projeto com autenticação .....	8
2.3.1 Estrutura Básica do Projeto .....	8
2.3.2 Criando Estruturas de UI .....	14
2.3.3 Criando a navegação do projeto .....	31
2.4 Navigation Component .....	31
2.4.1 Integrando o Firebase com o Android .....	38
2.4.2 Firebase Authentication .....	40
2.4.3 Reset de senha .....	55
2.4.4 Logout .....	56
CONCLUSÃO .....	59
REFERÊNCIAS .....	60

## 2 AUTENTICAÇÃO COM FIREBASE E ANDROID

### 2.1 Introdução

O processo de autenticação garante que usuários acessem suas contas mediante o uso de usuário e senha. Esse processo pode ser realizado também por meio de autenticadores externos, como Google e Facebook, por meio da padronização OAuth.

O padrão OAuth 2.0 permite que sistemas possam solicitar a autorização de outras aplicações sem necessitar conhecer a senha, o que transfere toda a segurança para esses outros provedores.



Figura 2.1 - Entrada do iFood utilizando provedores externos  
Fonte: ifood.com (2020)

Provedores de acesso como Google, Facebook e Microsoft possuem equipes e condições tecnológicas de manter um alto nível de confiabilidade em seus processos de segurança, por tal razão, faz sentido utilizar a autorização externa para usufruir dessa condição, ao mesmo tempo que passa maior segurança aos usuários.

O Firebase Authenticator conta com os seguintes provedores de autorização, além de fornecer também uma autenticação própria:

- Google

- Facebook
- GitHub
- Twitter
- Yahoo
- Microsoft
- Apple
- Play Games

Também é possível acessar de forma anônima ou ainda utilizar o próprio smartphone e SMS como formas de autorização.

## 2.2 O projeto Calcula Flex

Vamos criar um projeto para demonstrar como implementar o Firebase em um aplicativo para a plataforma Android. Nessa aplicação, vamos utilizar o Firebase como autorizador e também para armazenar os usuários da aplicação, assim, exploraremos melhor os recursos oferecidos pela plataforma.

O protótipo do aplicativo Calcula Flex é dividido em três telas principais:

- Tela de Login:** Apresenta o logo "Calcula Flex" e o texto "Faça o login para continuar". Possui campos para "E-MAIL" (preenchido com "heiderlopes@apps.com.br") e "PASSWORD" (com caracteres ocultos por pontos). Um botão "Conectar" está na base.
- Tela de Registro:** Também com o logo "Calcula Flex", esta tela contém campos para "USERNAME" (preenchido com "Heider Lopes"), "E-MAIL" (preenchido com "heiderlopes@apps.com.br"), "TELEFONE" (preenchido com "(11)99999-9999") e "SENHA" (com caracteres ocultos por pontos). Um botão "Criar Conta" está na base.
- Tela de Cálculo:** Exibe o logo "Calcula Flex" e um ícone de Facebook no canto superior direito. O formulário inclui:
  - Um campo "VEÍCULO" com o texto "Cruze Sport LTZ".
  - Uma seção "Consumo Médio" com dois campos: "KM/L GASOLINA" (valor 10,6) e "KM/L ÁLCOOL" (valor 6,1).
  - Uma seção "Posto de Gasolina" com dois campos: "PREÇO DA GASOLINA" (valor 3,57) e "PREÇO DO ÁLCOOL" (valor 2,09).
  - Um botão "Descobrir melhor combustível" em destaque.
  - Um link "Limpar dados" na base.

Figura 2.2 - Protótipo do aplicativo a ser desenvolvido integrado com o Firebase  
Fonte: Elaborado pelo autor (2020)

## 2.3 Desenvolvimento do projeto com autenticação

### 2.3.1 Estrutura Básica do Projeto

Começamos abrindo nosso ambiente de desenvolvimento, o Android Studio. Para criar um projeto, clique em “Start a new Android Studio Project”:

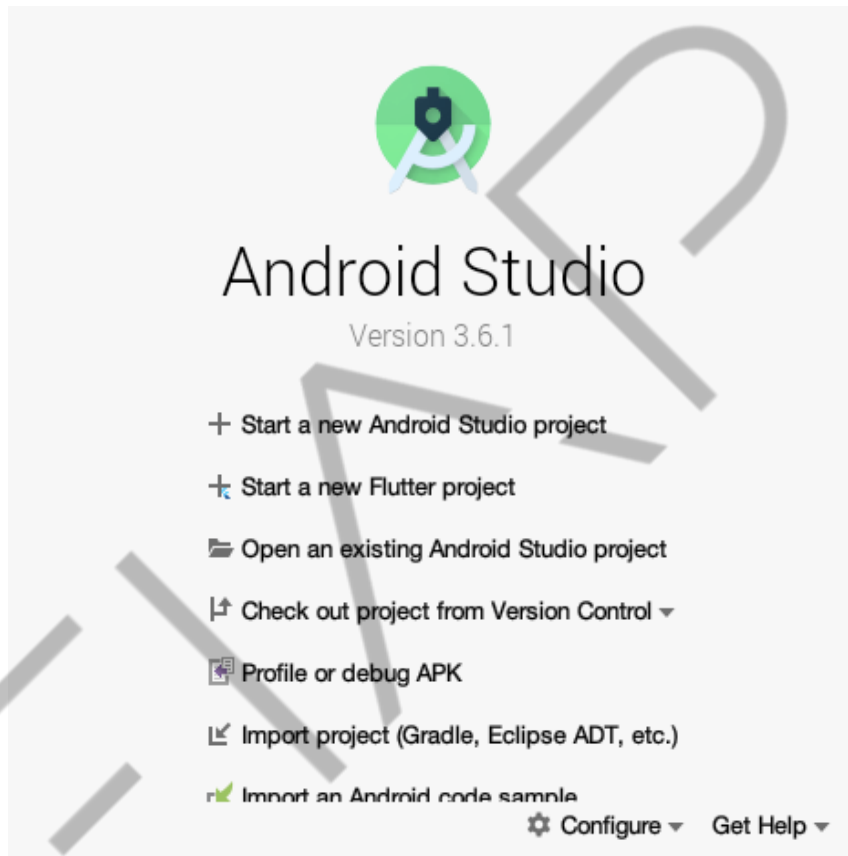


Figura 2.3 - Tela de início do Android Studio  
Fonte: Elaborado pelo autor (2020)

Em seguida, selecione “Empty Activity” e clique em “Next”.



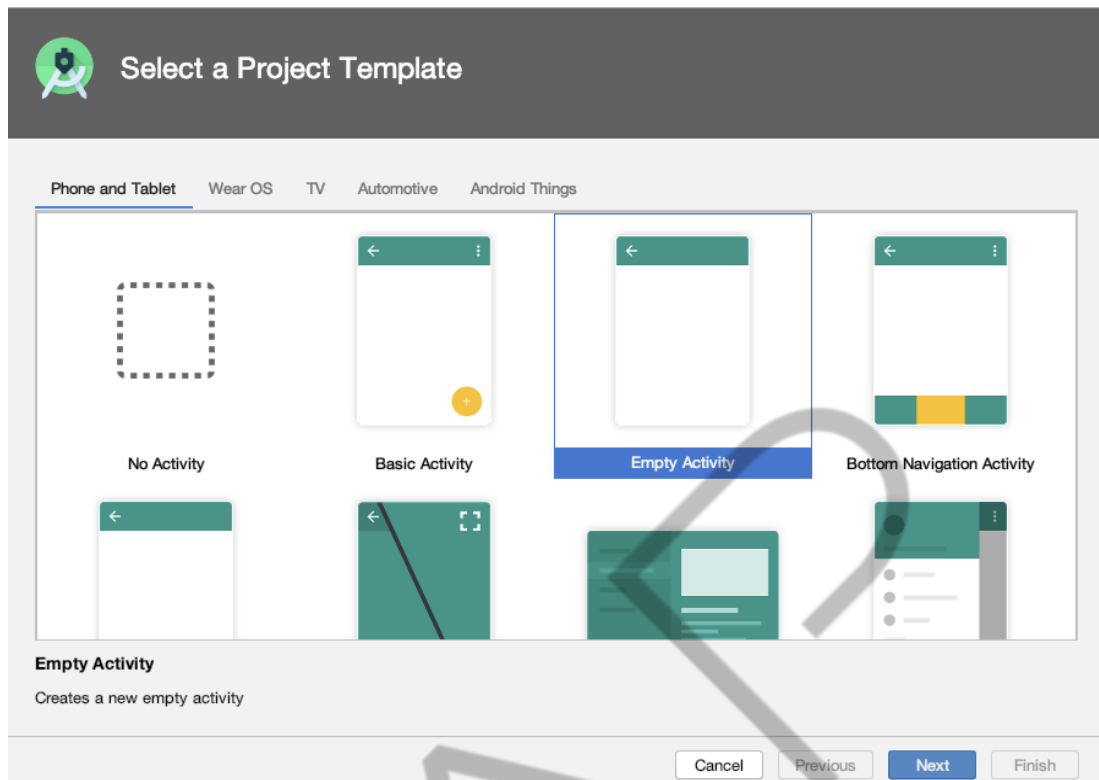


Figura 2.4 - Seleção de Template do Projeto  
Fonte: Elaborado pelo autor (2020)

Configure seu projeto. Neste ponto, é **importante você definir o pacote da sua aplicação de forma única**, para não ter problema de conflito no momento de configurarmos o app no Firebase. Por exemplo: `br.com.seunomecompleto.calculaflex`.

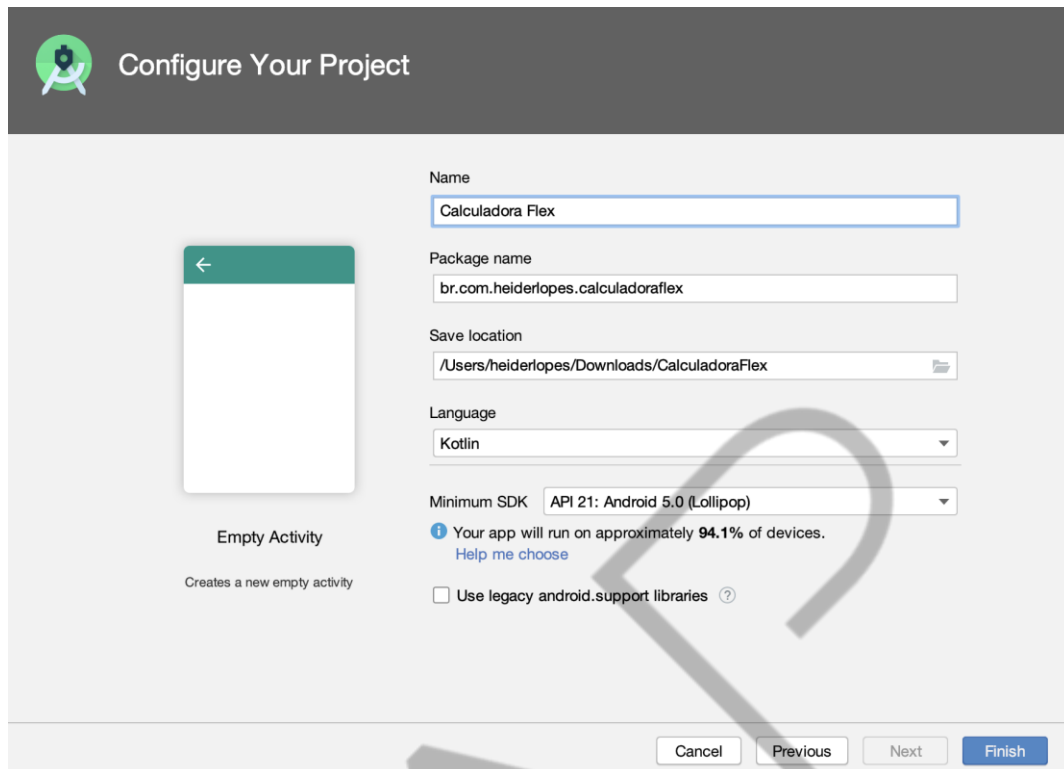


Figura 2.5 - Configuração do seu projeto  
Fonte: Elaborado pelo autor (2020)

Precisaremos incorporar alguns recursos visuais, por isso, baixe os arquivos disponíveis em <https://github.com/FIAPON/CalculaFlexRecursos>.

Agora, vamos configurar os estilos da aplicação. Abra o arquivo styles.xml e adicione o seguinte código:

```
<resources>

<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:textColor">@color/primaryTextColor</item>
    <item name="android:fontFamily">@font/helvetica_neue</item>

    <item name="android:editTextColor">@color/primaryTextColor</item>

    <item name="colorControlNormal">@color/backgroundComponentColor</item>
    <item name="colorControlActivated">#96ADC3</item>

</style>

<style name="splash_text">
    <item name="android:layout_width">wrap_content</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_marginTop">36dp</item>
```

```
<item name="android:fontFamily">@font/helvetica_neue</item>
<item name="android:text">@string/app_name</item>
<item name="android:textColor">@color/primaryTextColor</item>
<item name="android:textSize">@dimen/splash_text_size</item>
<item name="android:textStyle">bold</item>
</style>

<style name="link">
  <item name="android:layout_width">wrap_content</item>
  <item name="android:layout_height">wrap_content</item>
  <item name="android:layout_marginTop">36dp</item>
  <item name="android:fontFamily">@font/helvetica_neue</item>
  <item name="android:text">@string/app_name</item>
  <item name="android:textColor">@drawable/link_selector</item>
  <item name="android:textSize">@dimen/link_font_size</item>
  <item name="android:textStyle">bold</item>
  <item name="android:clickable">true</item>
</style>

<style name="container_edit_text">

  <item name="android:layout_width">match_parent</item>
  <item name="android:layout_height">wrap_content</item>
  <item name="android:background">@drawable/shape_edit_text</item>
  <item name="android:orientation">vertical</item>
  <item name="android:layout_marginBottom">16dp</item>
</style>

<style name="container_edit_text_inline">
  <item name="android:layout_width">match_parent</item>
  <item name="android:layout_height">wrap_content</item>
  <item name="android:orientation">horizontal</item>
</style>

<style name="container_edit_text_inline_left" parent="container_edit_text">
  <item name="android:layout_marginEnd">8dp</item>
  <item name="android:layout_weight">0.5</item>
</style>

<style name="container_edit_text_inline_right" parent="container_edit_text">
  <item name="android:layout_marginStart">8dp</item>
  <item name="android:layout_weight">0.5</item>
</style>

<style name="label_edit_text">

  <item name="android:layout_width">match_parent</item>
  <item name="android:layout_height">wrap_content</item>
  <item name="android:textColor">@color/secondaryTextColor</item>
  <item name="android:textStyle">bold</item>

</style>

<style name="field_edit_text">
  <item name="android:layout_width">match_parent</item>
```

```
<item name="android:layout_height">wrap_content</item>
<item name="android:textStyle">bold</item>
<item name="android:maxLines">1</item>
</style>

<style name="field_edit_text_number" parent="field_edit_text">
  <item name="android:maxLength">5</item>
  <item name="android:inputType">number</item>
</style>

<style name="header_text">
  <item name="android:textColor">@color/headerTextColor</item>
  <item name="android:textSize">@dimen/header_font_size</item>
  <item name="android:textStyle">bold</item>
</style>

<style name="button">
  <item name="android:layout_width">match_parent</item>
  <item name="android:layout_height">wrap_content</item>
  <item name="android:textAllCaps">false</item>
  <item name="android:textColor">#FFF</item>
  <item name="android:textSize">20sp</item>
  <item name="android:textStyle">bold</item>
  <item name="android:background">@drawable/button_selector</item>
</style>

<style name="section_title">
  <item name="android:layout_width">match_parent</item>
  <item name="android:layout_height">wrap_content</item>
  <item name="android:textAllCaps">false</item>
  <item name="android:textColor">@color/primaryTextColor</item>
  <item name="android:textSize">18sp</item>
  <item name="android:textStyle">bold</item>
  <item name="android:layout_marginStart">16dp</item>
  <item name="android:layout_gravity">center_vertical</item>
</style>

<style name="container_section">
  <item name="android:layout_width">match_parent</item>
  <item name="android:layout_height">wrap_content</item>
  <item name="android:layout_marginTop">8dp</item>
  <item name="android:layout_marginBottom">16dp</item>
</style>

<style name="icon_section">
  <item name="android:layout_width">32dp</item>
  <item name="android:layout_height">32dp</item>
  <item name="android:layout_marginStart">8dp</item>
</style>
</resources>
```

Código-fonte 2.1 – Arquivo styles.xml  
Fonte: Elaborado pelo autor (2020)

Nosso próximo passo é definir as cores que iremos utilizar no projeto. Abra o arquivo colors.xml e adicione o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>

    <color name="secondaryBackground">#214966</color>
    <color name="primaryTextColor">#214966</color>
    <color name="secondaryTextColor">#9EB0BC</color>
    <color name="backgroundComponentColor">#F2F4F6</color>

    <color name="backgroundButtonDisabled">#F2F4F6</color>
    <color name="backgroundButton">@color/secondaryBackground</color>
    <color name="backgroundButtonPressed">#6698BC</color>

    <color name="headerTextColor">#FFF</color>

    <color name="home_navigation_background">#214966</color>
    <color name="home_navigation_item">#FFF</color>

    <color name="linkTextColor">@color/primaryTextColor</color>
    <color name="linkTextColorClicked">#E49C0C</color>
    <color name="linkTextColorDisabled">#666</color>
</resources>
```

Código-fonte 2.2 – Arquivo colors.xml  
Fonte: Elaborado pelo autor (2020)

Precisamos definir os Strings (labels) do projeto que servirão para incorporarmos em algumas telas e botões. Abra o arquivo strings.xml e adicione o seguinte código:

```
<resources>
    <string name="app_name">Calcula Flex</string>

    <string name="menu_home_home">Home</string>
    <string name="menu_home_profile">Perfil</string>

    <string name="home_select_options">Selecione o que deseja calcular</string>

    <string name="label_username">SEU NOME</string>
    <string name="label_phone">TELEFONE</string>
    <string name="label_email">E-MAIL</string>
    <string name="label_password">SENHA</string>
    <string name="label_accept_terms">Li e aceitos os termos</string>
    <string name="label_terms">Termos de uso</string>

    <string name="label_create_account">Criar minha conta</string>
    <string name="label_reset_password">Esqueci minha senha</string>
```

```
<string name="logo_app_content_desc">Logo app</string>
<string name="button_login">Conectar</string>
<string name="button_create_account">Criar conta</string>
<string name="button_i_have_account">Já tenho conta</string>

<string name="login_subtitle">Faça o login para continuar</string>

<string name="signup_subtitle">Preencha os dados da sua conta</string>

<!-- TODO: Remove or change this placeholder text -->
<string name="hello_blank_fragment">Hello blank fragment</string>

<string name="label_better_fuel_subtitle">Descubra o melhor custo/benefício</string>
<string name="label_car">VEÍCULO</string>
<string name="label_km_gasoline">KM/L GASOLINA</string>
<string name="label_km_ethanol">KM/L ETANOL</string>
<string name="label_price_gasoline">Preço Gasolina</string>
<string name="label_price_ethanol">Preço Álcool</string>
<string name="label_gas_station">Posto de Gasolina</string>
<string name="label_average_consumption">Consumo médio</string>
<string name="label_calculate">Descobrir melhor combustível</string>
<string name="label_clear">Limpar dados</string>

<string name="loading_message_processing">Carregando</string>

</resources>
```

Código-fonte 2.3 – Arquivo strings.xml  
Fonte: Elaborado pelo autor (2020)

Nosso último ajuste é relacionado às dimensões de telas e tamanhos de fonte. Para isso, abra o arquivo `dimens.xml` e adicione o seguinte código:

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
    <dimen name="splash_text_size">32sp</dimen>
    <dimen name="link_font_size">16sp</dimen>
    <dimen name="header_font_size">16sp</dimen>
</resources>
```

Código-fonte 2.4 – Arquivo `dimens.xml`  
Fonte: Elaborado pelo autor (2020)

### 2.3.2 Criando Estruturas de UI

Para uma melhor organização do projeto, iremos adicionar os *fragments* e *activity* dentro do pacote de UI. Para isso, no pacote principal do aplicativo, clique com o botão direito do mouse em **new** ⇒ **package**

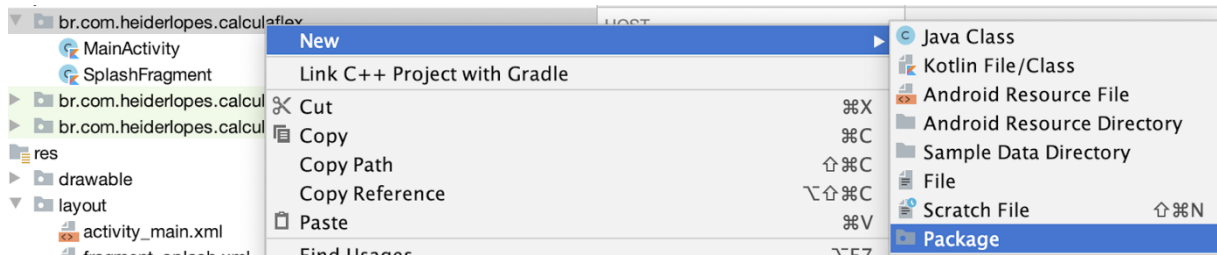


Figura 2.6 - Criação de um novo package  
Fonte: Elaborado pelo autor (2020)

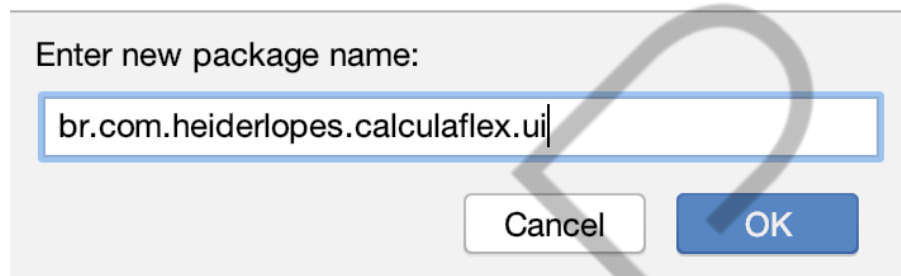


Figura 2.7 - Nomeando o novo pacote para ui  
Fonte: Elaborado pelo autor (2020)

Agora, serão criadas as telas do aplicativo. Dentro do package **ui** criado anteriormente, serão adicionados os **packages** para as telas do aplicativo (**SignIn**, **Login** e **BetterFuel**).

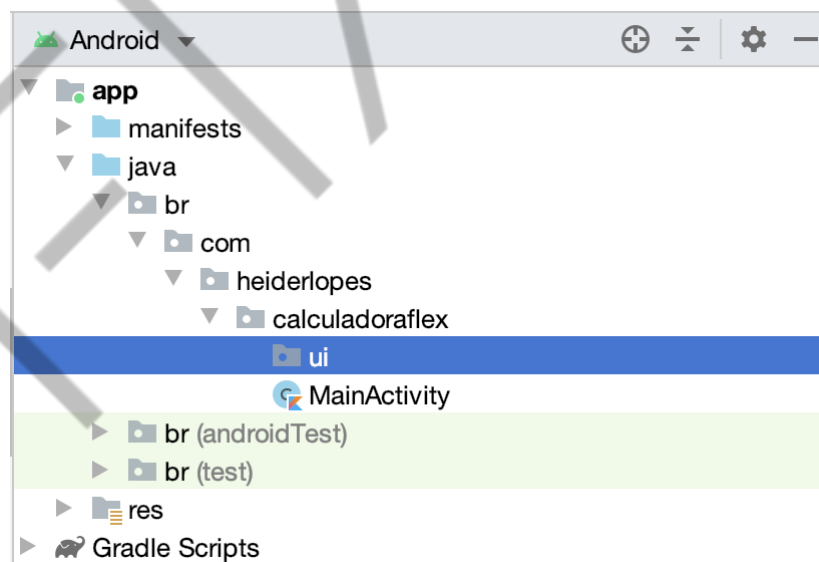


Figura 2.8 - Pacote de UI  
Fonte: Elaborado pelo autor (2020)

Clique com o botão direito do mouse em New → Package e, sobre o pacote **ui**, crie os pacotes **login**, **signup**, **betterfuel**.

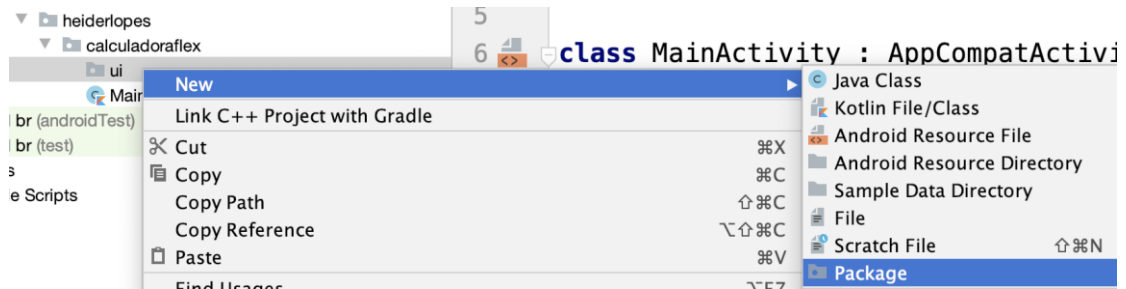


Figura 2.9 - Criando um pacote  
Fonte: Elaborado pelo autor (2020)

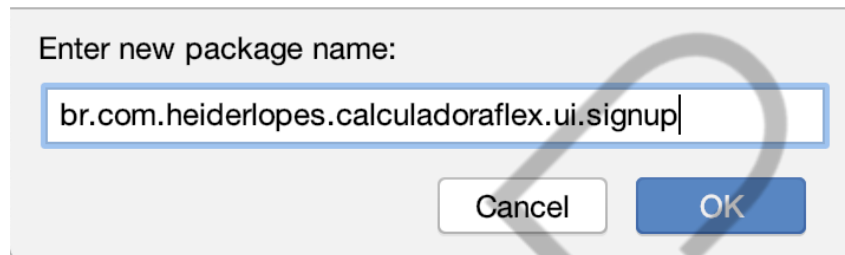


Figura 2.10 - Criando um pacote signup  
Fonte: Elaborado pelo autor (2020)

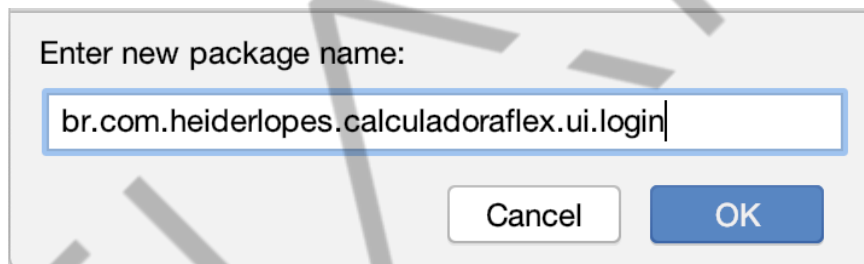


Figura 2.11 - Criando um pacote login  
Fonte: Elaborado pelo autor (2020)

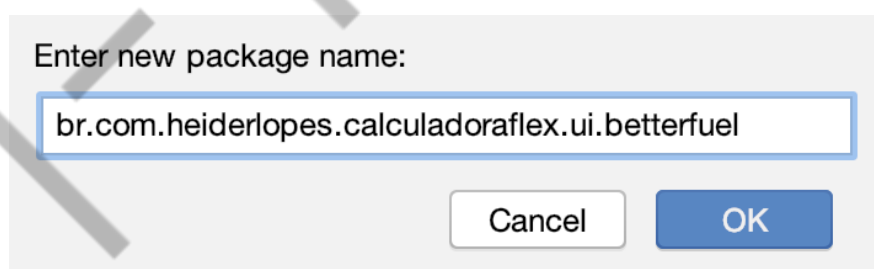


Figura 2.12 - Criando um pacote betterfuel  
Fonte: Elaborado pelo autor (2020)



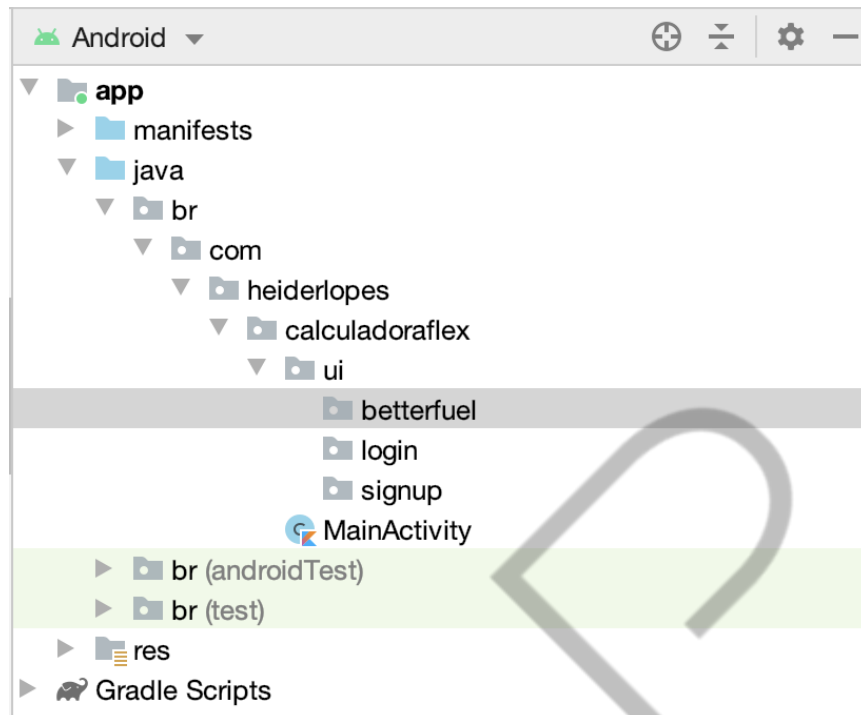


Figura 2.13 - Estrutura dos pacotes criados  
Fonte: Elaborado pelo autor (2020)

O próximo passo a ser seguido é a criação das telas do aplicativo. Para esse fim, será utilizado o fragments.

Começamos com a tela de login. Clique com o botão direito do mouse sobre o package login → New → Fragment → Fragment (Blank).

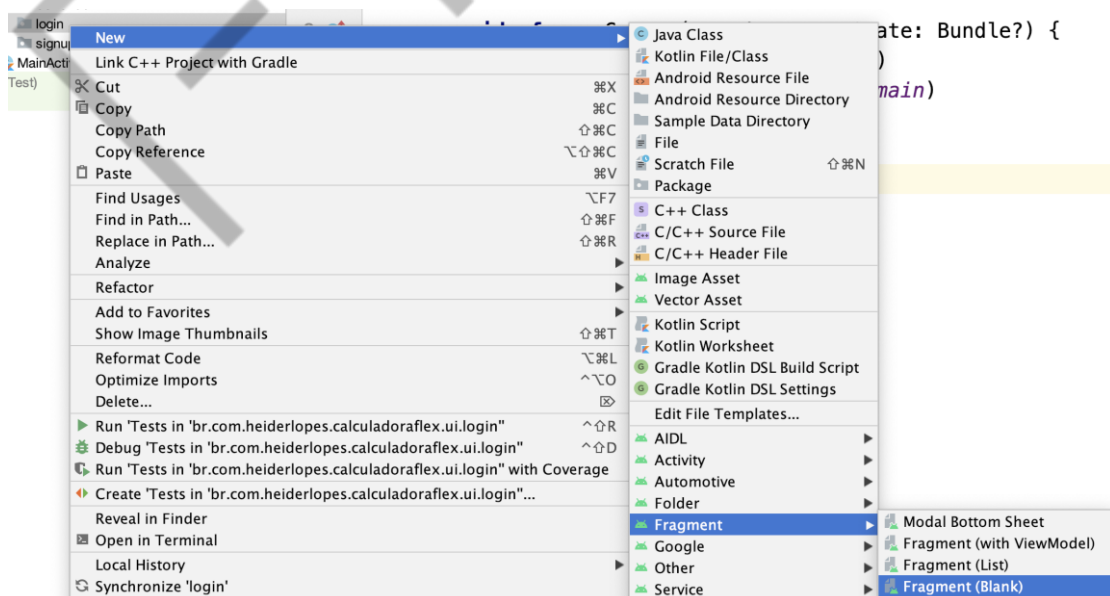


Figura 2.14 - Criando o LoginFragment  
Fonte: Elaborado pelo autor (2020)

Dê o nome de LoginFragment.

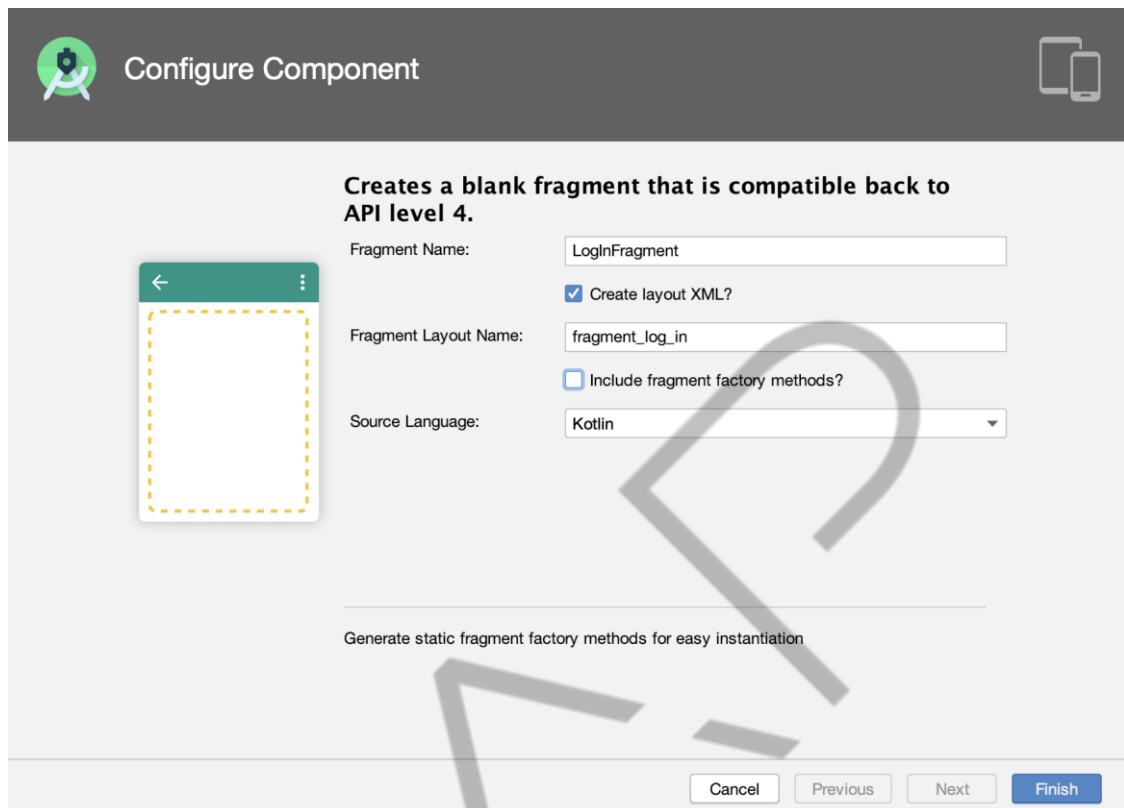


Figura 2.15 - Definindo o nome do LoginFragment  
Fonte: Elaborado pelo autor (2020)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/ivLogoApp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginTop="32dp"
        android:src="@drawable/ic_logo_login"
        android:contentDescription="@string/logo_app_content_desc"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvAppName"
        android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:layout_marginStart="42dp"
android:layout_marginTop="48dp"
android:fontFamily="@font/helvetica_neue"
android:text="@string/app_name"
android:textColor="@color/primaryTextColor"
android:textSize="32sp"
android:textStyle="bold"
app:layout_constraintStart_toStartOf="@+id/ivLogoApp"
app:layout_constraintTop_toTopOf="@+id/ivLogoApp" />

<LinearLayout
    android:id="@+id/containerLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"
    android:orientation="vertical"
    app:layout_constraintBottom_toTopOf="@+id/tvResetPassword"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tvSubTitleLogin">

    <LinearLayout style="@style/container_edit_text">

        <TextView
            style="@style/label_edit_text"
            android:text="@string/label_email" />

        <EditText
            android:id="@+id/etEmailLogin"
            style="@style/field_edit_text"
            tools:text="heiderlopes@apps.com.br" />

    </LinearLayout>

    <LinearLayout style="@style/container_edit_text">

        <TextView
            style="@style/label_edit_text"
            android:text="@string/label_password" />

        <EditText
            android:id="@+id/etPasswordLogin"
            style="@style/field_edit_text"
            android:inputType="textPassword"
            tools:text="heiderlopes@apps.com.br" />

    </LinearLayout>

    <Button
        android:id="@+id/btLogin"
        android:text="@string/button_login"
        style="@style/button"/>
```

```
</LinearLayout>

<TextView
    android:id="@+id/tvSubTitleLogin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/login_subtitle"
    android:textColor="#9EB0BC"
    android:textSize="16sp"
    android:textStyle="bold"
    app:layout_constraintStart_toStartOf="@+id/tvAppName"
    app:layout_constraintTop_toBottomOf="@+id/tvAppName" />

<TextView
    android:id="@+id/tvResetPassword"
    style="@style/link"
    android:layout_width="0dp"
    android:layout_marginStart="24dp"
    android:layout_marginBottom="24dp"
    android:text="@string/label_reset_password"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/guideline"
    app:layout_constraintStart_toStartOf="parent" />

<TextView
    android:id="@+id/tvNewAccount"
    style="@style/link"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="24dp"
    android:layout_marginBottom="24dp"
    android:gravity="end"
    android:text="@string/label_create_account"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="@+id/guideline" />

<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.5" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Código-fonte 2.5 – Layout xml da tela de Login  
Fonte: Elaborado pelo autor (2020)

Com a tela Login (Sign In) pronta, vamos agora construir a tela de cadastro ou Sign Up. Clique com o botão direito do mouse sobre o package signup → New → Fragment → Fragment (Blank):

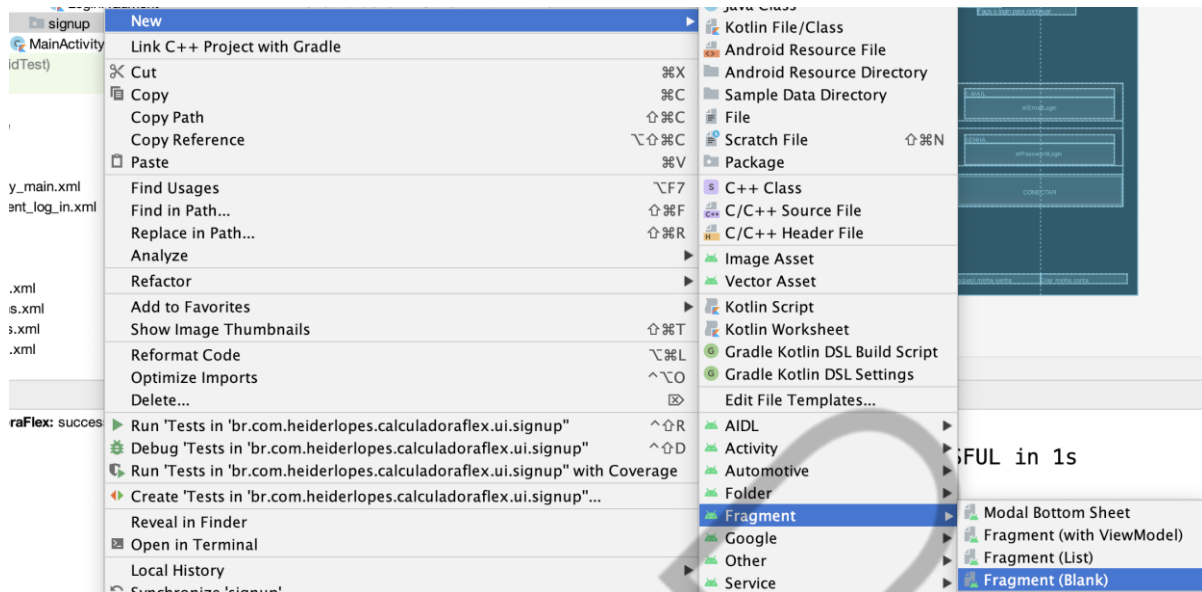


Figura 2.16 - Criando o SignUp  
Fonte: Elaborado pelo autor (2020)

Dê o nome de **SignUpFragment**.

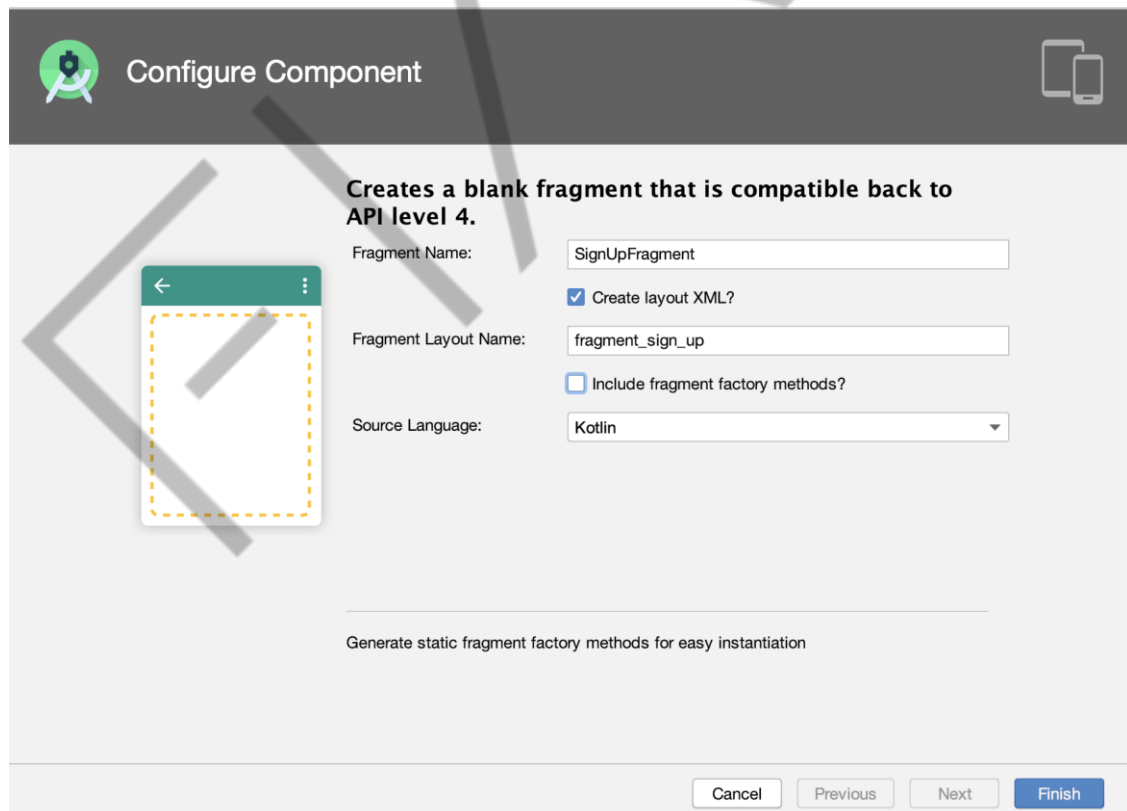


Figura 2.17 - Definindo o nome do SignUpFragment  
Fonte: Elaborado pelo autor (2020)

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/tvSubTitleSignUp">

<LinearLayout
    android:padding="32dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/ivLogoApp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_logo_login"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/tvAppName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="42dp"
            android:layout_marginTop="48dp"
            android:fontFamily="@font/helvetica_neue"
            android:text="@string/app_name"
            android:textColor="@color/primaryTextColor"
            android:textSize="32sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="@+id/ivLogoApp"
            app:layout_constraintTop_toTopOf="@+id/ivLogoApp" />

        <TextView
            android:id="@+id/tvSubTitleSignUp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/signup_subtitle"
            android:textColor="#9EB0BC"
            android:textSize="16sp"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="@+id/tvAppName"
            app:layout_constraintTop_toBottomOf="@+id/tvAppName" />
    </androidx.constraintlayout.widget.ConstraintLayout>

</LinearLayout
```

```
android:layout_marginTop="32dp"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical">

<LinearLayout style="@style/container_edit_text">

    <TextView
        style="@style/label_edit_text"
        android:text="@string/label_username" />

    <EditText
        android:id="@+id/etUserNameSignUp"
        style="@style/field_edit_text"
        tools:text="Heider Lopes" />

</LinearLayout>

<LinearLayout style="@style/container_edit_text">

    <TextView
        style="@style/label_edit_text"
        android:text="@string/label_email" />

    <EditText
        android:id="@+id/etEmailSignUp"
        style="@style/field_edit_text"
        tools:text="heiderlopes@apps.com.br" />

</LinearLayout>

<LinearLayout style="@style/container_edit_text">

    <TextView
        style="@style/label_edit_text"
        android:text="@string/label_phone" />

    <EditText
        android:id="@+id/etPhoneSignUp"
        style="@style/field_edit_text"
        tools:text="(11) 9999999-999" />

</LinearLayout>

<LinearLayout style="@style/container_edit_text">

    <TextView
        style="@style/label_edit_text"
        android:text="@string/label_password" />

    <EditText
        android:id="@+id/etPasswordSignUp"
        style="@style/field_edit_text"
        android:inputType="textPassword"
        tools:text="xxxxxx" />
```

```

</LinearLayout>

<Button
    android:id="@+id/btCreateAccount"
    android:text="@string/button_create_account"
    style="@style/button"/>

</LinearLayout>

</LinearLayout>
</ScrollView>

```

Código-fonte 2.6 – Layout xml da tela de SignUp  
Fonte: Elaborado pelo autor (2020)

Nesta etapa, vamos construir a tela para o cálculo do melhor combustível. Clique com o botão direito do mouse sobre o package betterfuel → New → Fragment → Fragment (Blank).

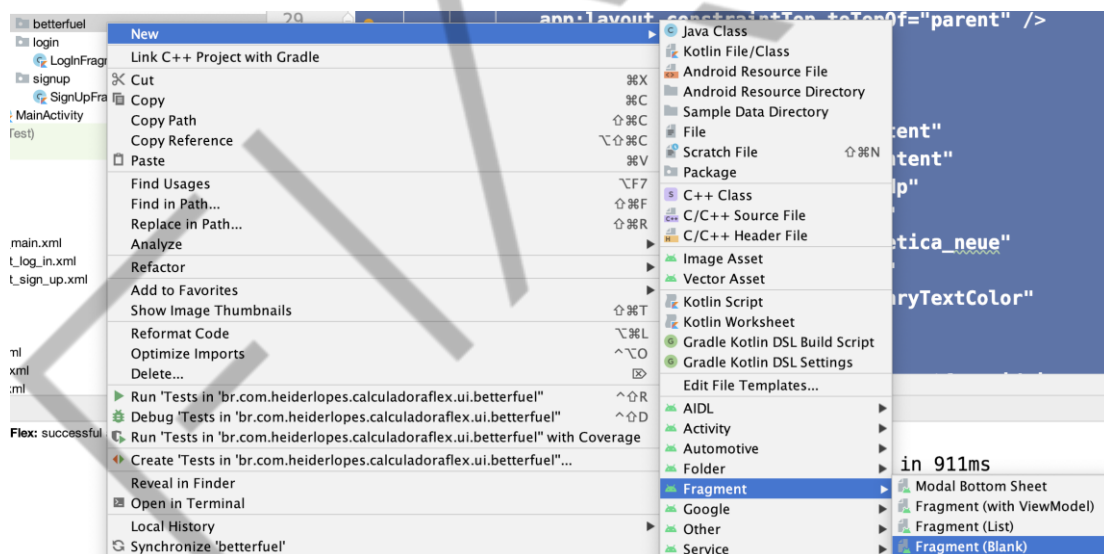


Figura 2.18 - Criando o Melhor Combustível  
Fonte: Elaborado pelo autor (2020)

Dê o nome de **BetterFuelFragment**.



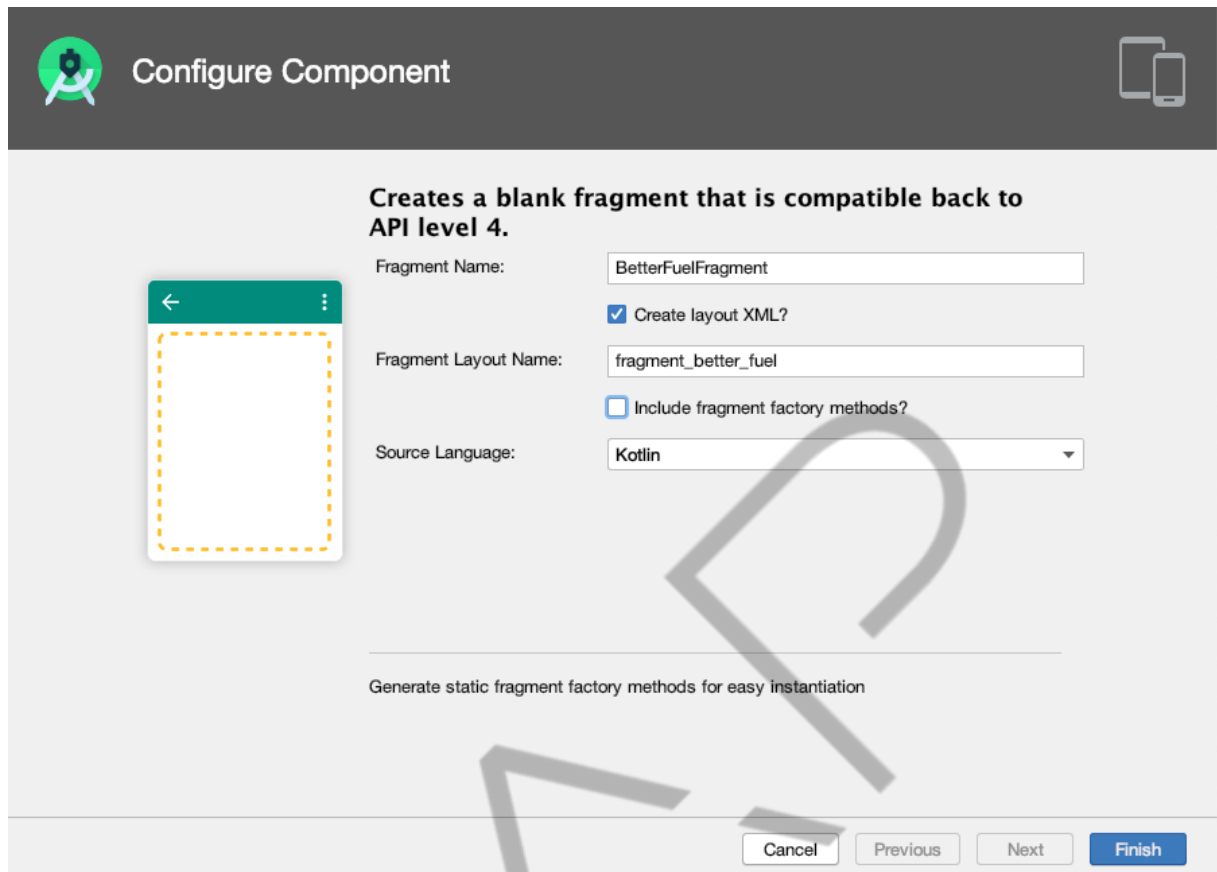


Figura 2.19 - Definindo o nome do BetterFuelFragment  
Fonte: Elaborado pelo autor (2020)

Podemos modularizar um aplicativo, basicamente, de duas maneiras: feature (por recurso) ou por layer (camada).

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tvSubTitleSignUp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingStart="16dp"
        android:paddingEnd="16dp"
        android:paddingBottom="16dp">
```

```
<ImageView
    android:id="@+id/btSignOut"
    android:layout_width="64dp"
    android:layout_height="64dp"
    android:layout_gravity="end"
    android:paddingStart="24dp"
    android:paddingTop="12dp"
    android:src="@drawable/ic_signout" />

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/ivLogoApp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_logo_login"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvAppName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="42dp"
        android:layout_marginTop="48dp"
        android:fontFamily="@font/helvetica_neue"
        android:text="@string/app_name"
        android:textColor="@color/primaryTextColor"
        android:textSize="32sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="@+id/ivLogoApp"
        app:layout_constraintTop_toTopOf="@+id/ivLogoApp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/label_better_fuel_subtitle"
        android:textColor="#9EB0BC"
        android:textSize="16sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="@+id/tvAppName"
        app:layout_constraintTop_toBottomOf="@+id/tvAppName" />
</androidx.constraintlayout.widget.ConstraintLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
android:layout_marginTop="32dp"
android:orientation="vertical">

<LinearLayout style="@style/container_edit_text">

    <TextView
        style="@style/label_edit_text"
        android:text="@string/label_car" />

    <EditText
        android:id="@+id/etCar"
        style="@style/field_edit_text"
        tools:text="Cruze Sport LTZ" />

</LinearLayout>

<LinearLayout style="@style/container_section">

    <ImageView
        style="@style/icon_section"
        android:src="@drawable/ic_average_consumption" />

    <TextView
        style="@style/section_title"
        android:text="@string/label_average_consumption" />
</LinearLayout>

<LinearLayout style="@style/container_edit_text_inline">

    <LinearLayout style="@style/container_edit_text_inline_left">

        <TextView
            style="@style/label_edit_text"
            android:text="@string/label_km_gasoline" />

        <EditText
            android:id="@+id/etKmGasoline"
            style="@style/field_edit_text_number"
            tools:text="10.6" />

    </LinearLayout>

    <LinearLayout style="@style/container_edit_text_inline_right">

        <TextView
            style="@style/label_edit_text"
            android:text="@string/label_km_ethanol" />

        <EditText
```

```
        android:id="@+id/etKmEthanol"
        style="@style/field_edit_text_number"
        tools:text="6.1" />

    </LinearLayout>
</LinearLayout>

<LinearLayout style="@style/container_section">

    <ImageView
        style="@style/icon_section"
        android:src="@drawable/ic_gas_station" />

    <TextView
        style="@style/section_title"
        android:text="@string/label_gas_station" />
</LinearLayout>

<LinearLayout style="@style/container_edit_text_inline">

    <LinearLayout style="@style/container_edit_text_inline_left">

        <TextView
            style="@style/label_edit_text"
            android:text="@string/label_price_gasoline" />

        <EditText
            android:id="@+id/etPriceGasoline"
            style="@style/field_edit_text_number"
            tools:text="10.6" />

    </LinearLayout>

    <LinearLayout style="@style/container_edit_text_inline_right">

        <TextView
            style="@style/label_edit_text"
            android:text="@string/label_price_ethanol" />

        <EditText
            android:id="@+id/etPriceEthanol"
            style="@style/field_edit_text_number"
            tools:text="6.1" />

    </LinearLayout>
</LinearLayout>

<Button
    android:layout_marginTop="24dp"
    android:id="@+id/btCalculate"
```

```

        style="@style/button"
        android:text="@string/label_calculate" />

        <TextView
            android:id="@+id/btClear"
            style="@style/link"
            android:layout_gravity="center"
            android:layout_marginStart="8dp"
            android:text="@string/label_clear"
            android:paddingBottom="16dp"/>

    </LinearLayout>

</LinearLayout>
</ScrollView>

```

Código-fonte 2.7 – Layout xml da tela de BetterFuelFragment  
Fonte: Elaborado pelo autor (2020)

Para que o app dê um feedback para o usuário enquanto estiver fazendo loading, deve-se criar um layout que será reaproveitado pela aplicação.

Para essa tela, será utilizada uma biblioteca do Airbnb para animações chamada Lottie. Abra o arquivo build.gradle (app) e adicione a seguinte dependência:

```
implementation 'com.airbnb.android:lottie:3.4.1'
```

Código-fonte 2.8 – Import da biblioteca do Airbnb Lottie  
Fonte: Elaborado pelo autor (2020)

Crie um arquivo xml dentro da pasta layout chamado **include\_loading.xml**.

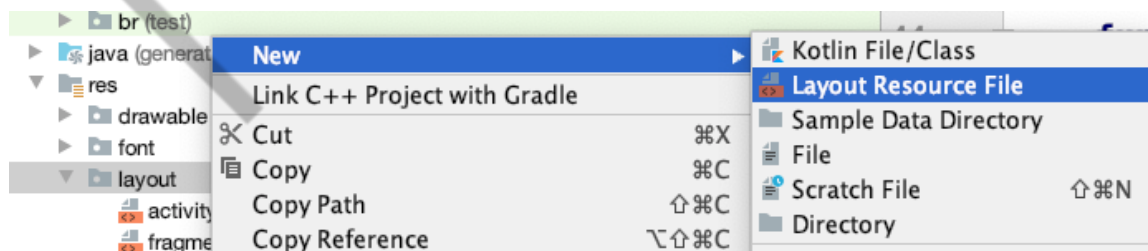


Figura 2.20 - Criação do arquivo loading  
Fonte: Elaborado pelo autor (2020)

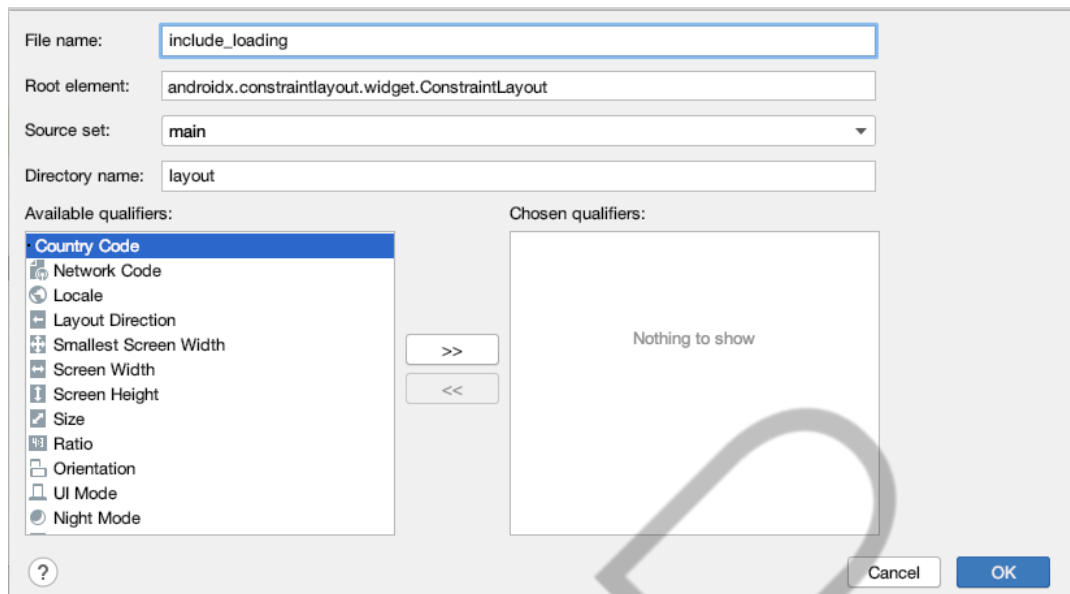


Figura 2.21 - Definição do nome do arquivo include\_loading.xml  
Fonte: Elaborado pelo autor (2020)

Adicione o seguinte código para tela de carregando:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/containerLoading"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/white"
    android:clickable="true"
    android:focusable="true"
    android:gravity="center"
    android:orientation="vertical"
    tools:visibility="gone"
    android:visibility="gone">

    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/ivLoading"
        android:layout_width="128dp"
        android:layout_height="128dp"
        app:lottie_autoPlay="true"
        app:lottie_rawRes="@raw/car_load"
        app:lottie_loop="true" />

    <TextView
        android:id="@+id/tvLoading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/loading_message_processing"
        android:textColor="#000000" />

</LinearLayout>
```

Código-fonte 2.9 – Definição do nome do arquivo include\_loading.xml  
Fonte: Elaborado pelo autor (2020)

### 2.3.3 Criando a navegação do projeto

A navegação do nosso projeto será realizada utilizando o componente navigation do Android JetPack.

O Android JetPack é um conjunto de bibliotecas e ferramentas criado com o objetivo de simplificar e melhorar a qualidade do processo de desenvolvimento de aplicativos Android. Tais bibliotecas facilitam o desenvolvimento de apps com qualidade, previsibilidade e simplicidade.

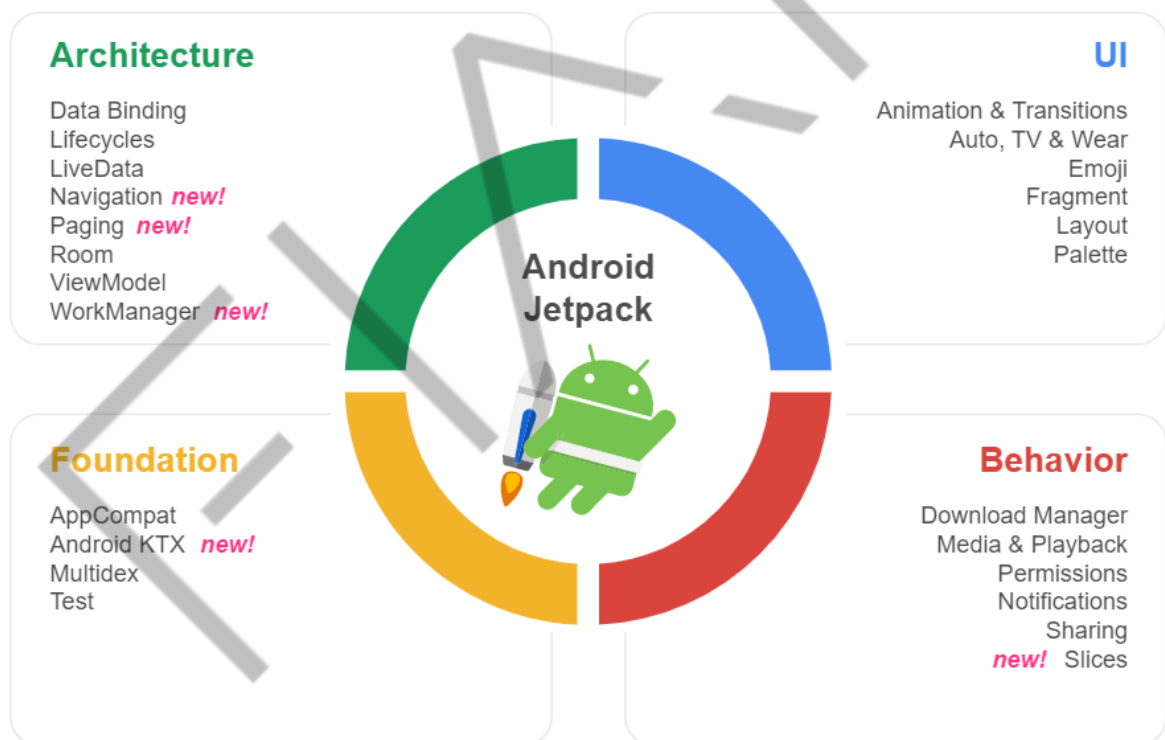


Figura 2.22 - Android JetPack  
Fonte: [developer.android.com/jetpack](https://developer.android.com/jetpack) (s.d.)

## 2.4 Navigation Component

O principal objetivo de utilizar este novo componente é reduzir a probabilidade de erro ao realizar alguma transação com fragments e melhorar a capacidade de testar a UI de forma isolada.

Ao utilizar o Navigation Component, você delegará ao componente os seguintes conceitos de navegação:

- Automação nas transações de fragments.
- Implementação dos princípios de navegação.

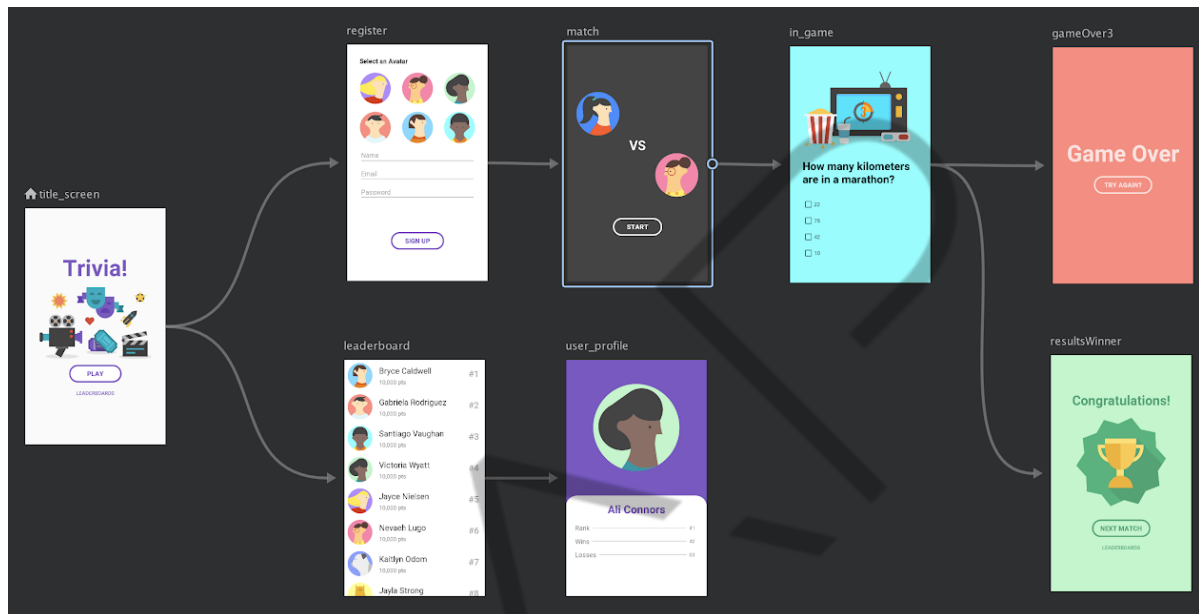


Figura 2.23 - Navigation

Fonte: [developer.android.com/jetpack](https://developer.android.com/jetpack) (2020)

Para adicionar o Navigation ao projeto, abra o arquivo build.gradle (app) e adicione as linhas em negrito:

```
android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"

    defaultConfig {
        applicationId "br.com.heiderlopes.calculadoraflex"
        minSdkVersion 21
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}
```



```

kotlinOptions {
    jvmTarget = "1.8"
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.core:core-ktx:1.3.1'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'

    implementation "androidx.navigation:navigation-fragment-ktx:2.3.0"
    implementation "androidx.navigation:navigation-ui-ktx:2.3.0"
}

```

Código-fonte 2.10 – Adicionando a dependência do navigation no projeto  
 Fonte: Elaborado pelo autor (2020)

Agora, clique com o botão direito do mouse sobre a pasta res → New → Android Resource Directory para criarmos o diretório navigation. Nessa pasta, ficarão os arquivos de navegação do aplicativo.

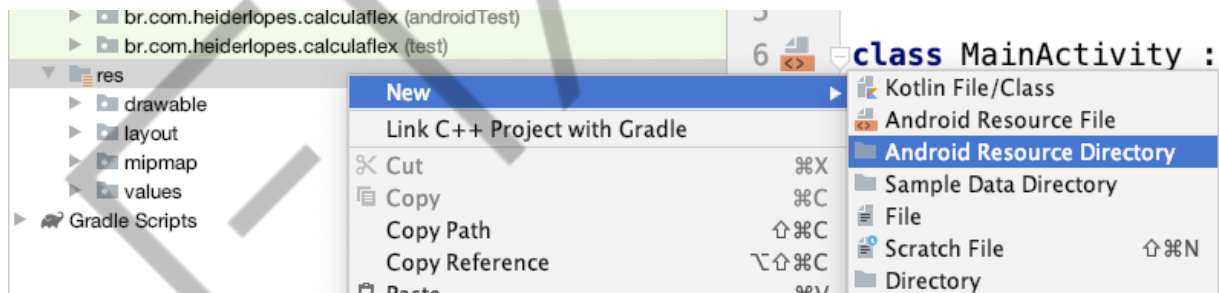
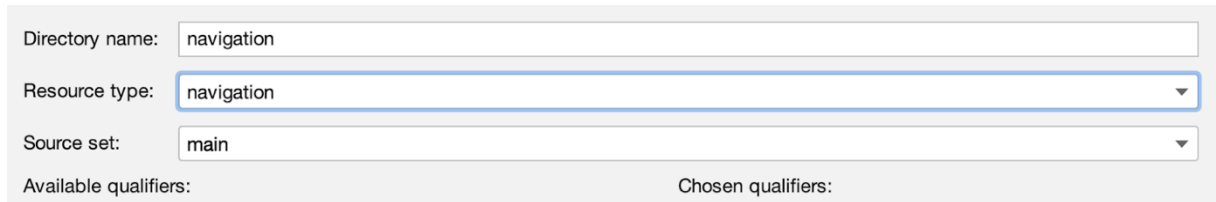


Figura 2.24 - Criando um Android Resource Directory  
 Fonte: Elaborado pelo autor (2020)

Escolha o **Directory Name** e o **Resource Type** como: **navigation**.

**Directory Name:** Nome do diretório. Por padrão, o diretório para os arquivos de navegação é chamado de navigation.

**Resource Type:** tipo de recurso a ser criado.



Directory name: navigation

Resource type: navigation

Source set: main

Available qualifiers: Chosen qualifiers:

Figura 2.25 - Criando o diretório navigation  
Fonte: Elaborado pelo autor (2020)

Agora, crie um arquivo chamado **main\_nav\_graph**. Nele, colocaremos a navegação do aplicativo. Para isso, clique com o botão direito sobre a pasta **navigation** → **New** → **Navigation Resource File**.

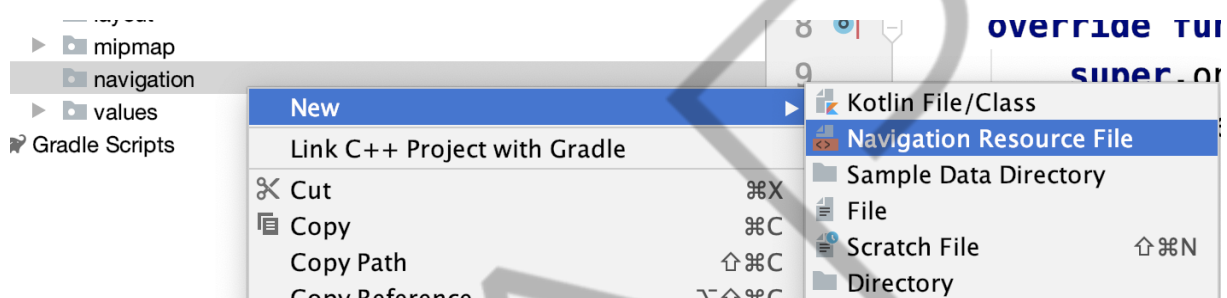
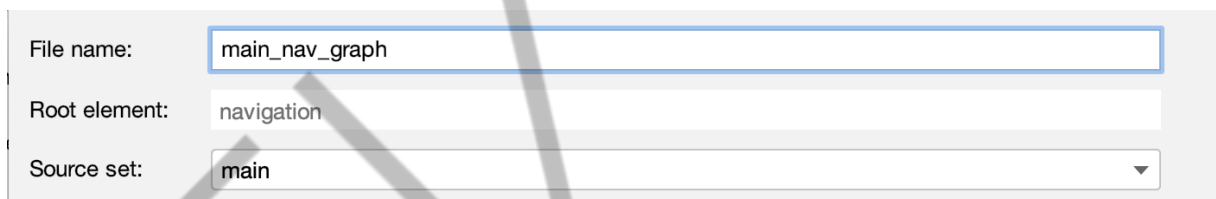


Figura 2.26 - Criando o arquivo de navegação  
Fonte: Elaborado pelo autor (2020)



File name: main\_nav\_graph

Root element: navigation

Source set: main

Figura 2.27 - Definindo o nome do arquivo main\_nav\_graph  
Fonte: Elaborado pelo autor (2020)

O próximo passo é adicionar os fragments (telas) criados anteriormente no **main\_nav\_graph**. Com o **arquivo main\_nav\_graph.xml** aberto, clique sobre o **New Destination**.

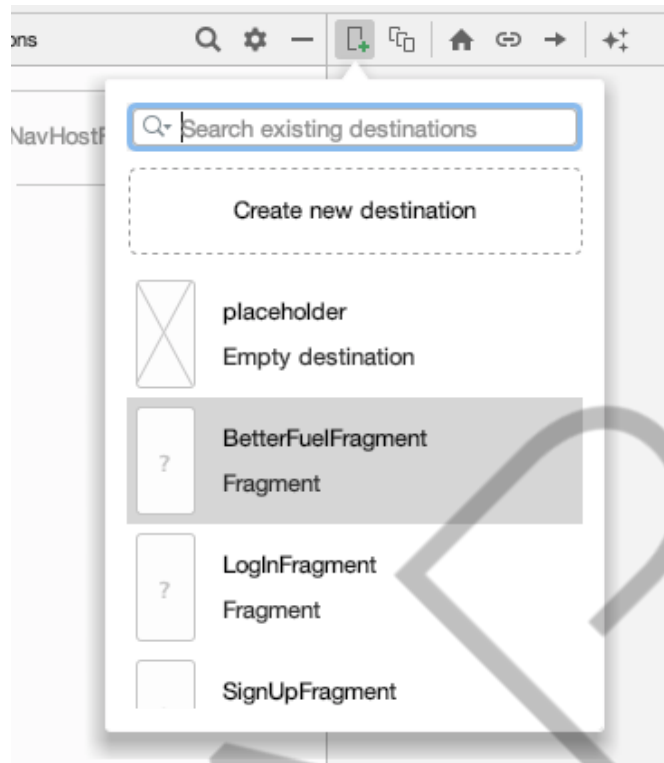


Figura 2.28 - New Destination e os fragments criados  
Fonte: Elaborado pelo autor (2020)

Adicione, nesta ordem, os **Fragments: BetterFuelFragment, LoginFragment, SignUpFragment.**

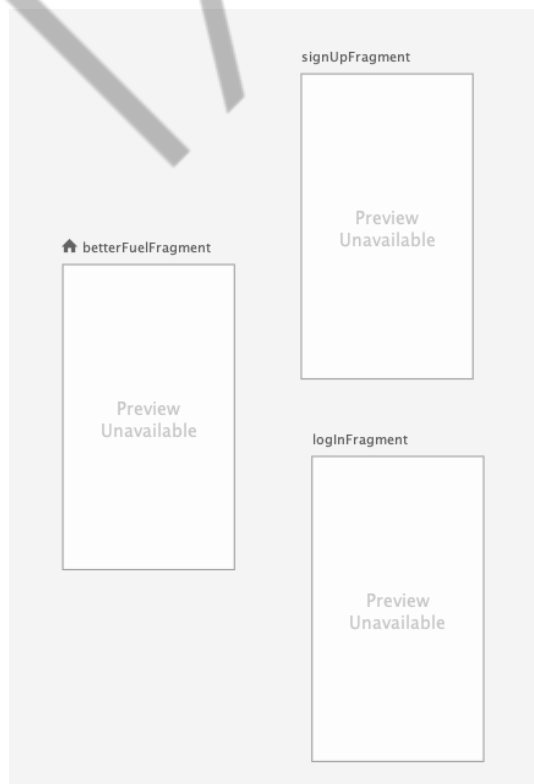


Figura 2.29 - Telas no Navigation  
Fonte: Elaborado pelo autor (2020)

Caso o navigation não exiba o layout das suas telas, você pode entrar no modo code, adicionar a tag `tools:layout` no fragment e escolher o referente àquela tela.



Figura 2.30 – Alteração da visualização do navigation para o modo code  
Fonte: Elaborado pelo autor (2020)

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main_nav_graph"
    app:startDestination="@id/betterFuelFragment">
    <fragment
        android:id="@+id/betterFuelFragment"
        android:name="br.com.heiderlopes.calculadoraflex.ui.betterfuel.BetterFuelFragment"
        android:label="BetterFuelFragment"
        tools:layout="@layout/fragment_better_fuel" />
    <fragment
        android:id="@+id/signUpFragment"
        android:name="br.com.heiderlopes.calculadoraflex.ui.signup.SignUpFragment"
        android:label="SignUpFragment"
        tools:layout="@layout/fragment_sign_up" />
    <fragment
        android:id="@+id/loginFragment"
        android:name="br.com.heiderlopes.calculadoraflex.ui.login.LoginFragment"
        android:label="LoginFragment"
        tools:layout="@layout/fragment_log_in" />
</navigation>
```

Código-fonte 2.11 – main\_navigation\_graph.xml com os layouts das telas  
Fonte: Elaborado pelo autor (2020)

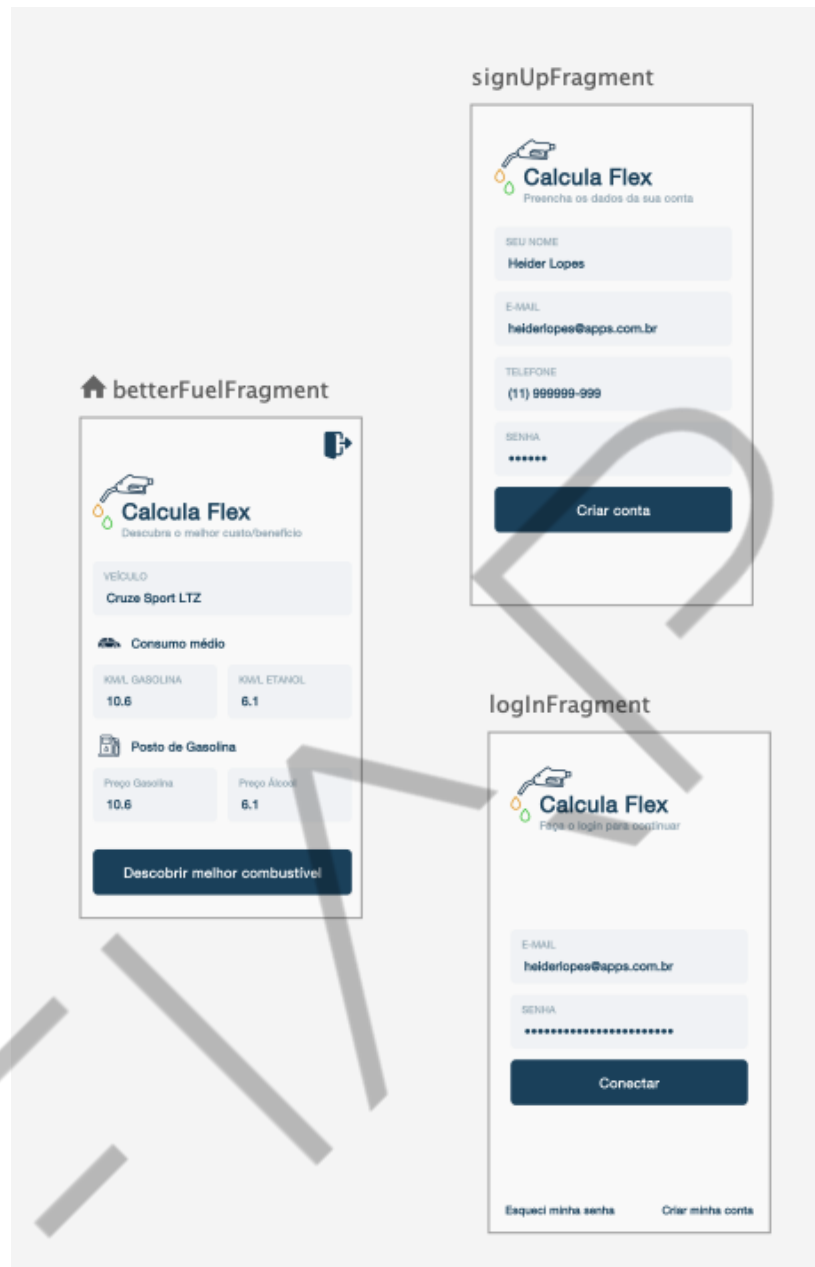


Figura 2.30 - main\_navigation\_graph.xml com preview dos layouts  
 Fonte: Elaborado pelo autor (2020)

Agora é hora de adicionarmos o navigation a nossa MainActivity.kt. Para isso, abra o arquivo **activity\_main.xml** e adicione o seguinte layout:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/navHostFragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
app:defaultNavHost="true"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:navGraph="@navigation/main_nav_graph" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Código-fonte 2.12 – layout da activity\_main  
Fonte: Elaborado pelo autor (2020)

Na propriedade `app:navGraph`, definimos qual navigation será utilizada. Como no projeto existe apenas o `main_nav_grap`, é ele que iremos utilizar.

Dentro da **MainActivity.kt** adicione o método para que o aplicativo seja executado no modo fullscreen:

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        fullscreen()
        setContentView(R.layout.activity_main)
    }

    private fun fullscreen() {
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);
        supportActionBar?.hide()
        this.window.setFlags(
            WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN
        );
    }
}
```

Código-fonte 2.13 – Método para activity ficar em FullScreen  
Fonte: Elaborado pelo autor (2020)

## 2.4.1 Integrando o Firebase com o Android

Para utilizar o Firebase, você precisa ter uma conta Google criada. Após isso, será possível realizar a integração com o app criado.

O Android Studio fornece suporte para facilitar a integração do app com o Firebase. essa integração será realizada por meio da IDE.

No Android Studio, clique no Menu Tools → Firebase.

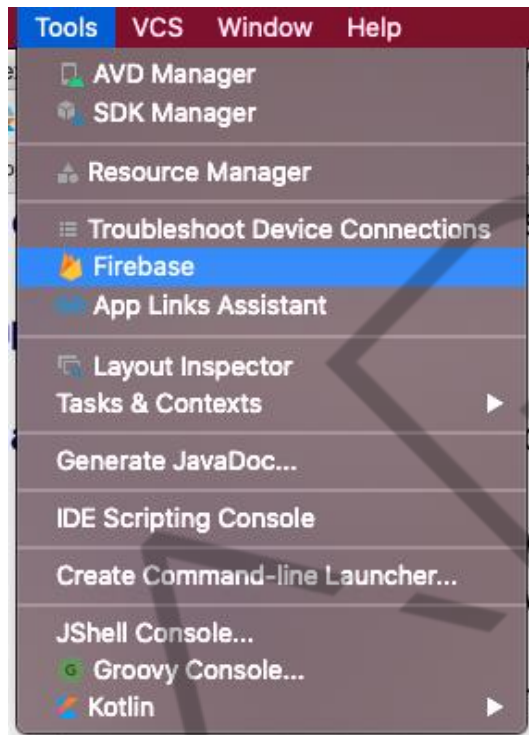


Figura 2.31 - Menu Firebase  
Fonte: Elaborado pelo autor (2020)

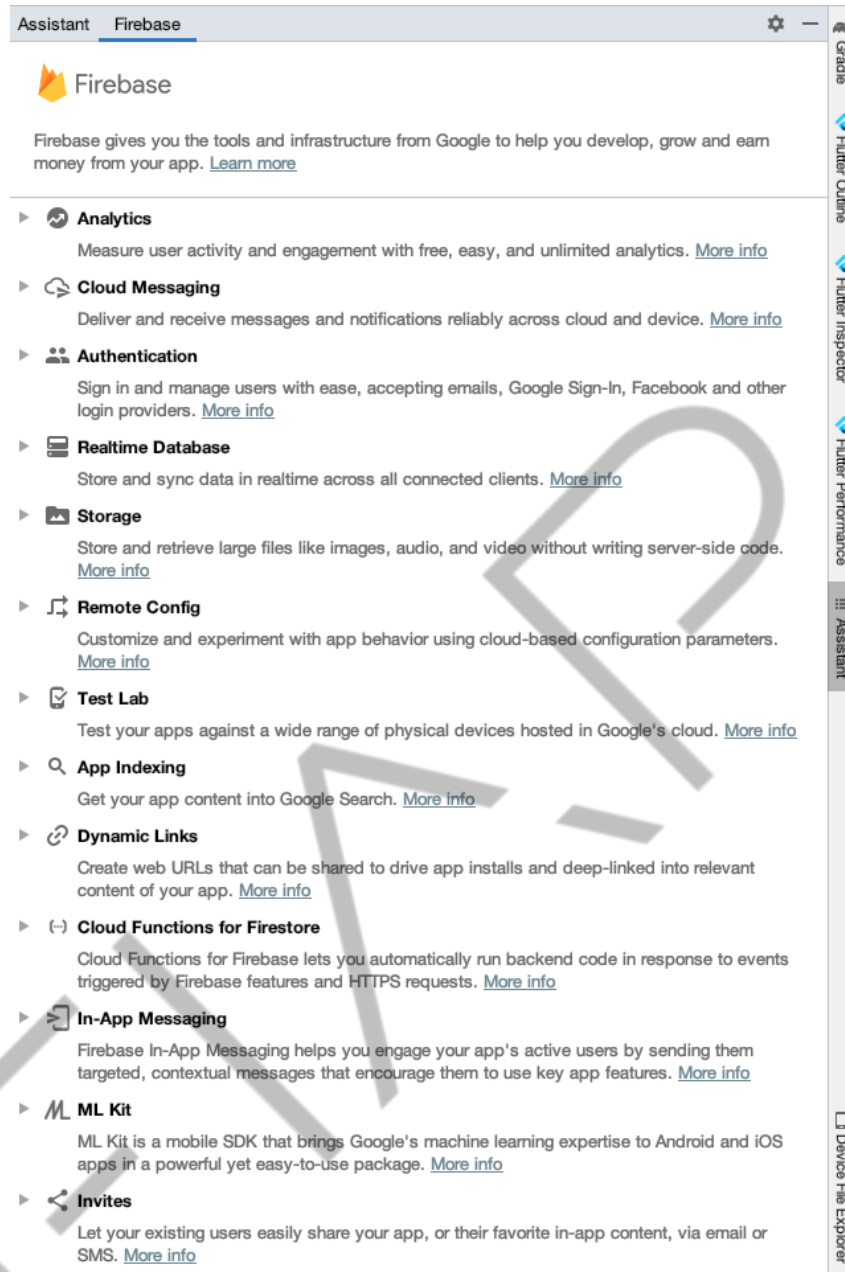


Figura 2.32 - Assistente do Firebase no Android Studio  
Fonte: Elaborado pelo autor (2020)

## 2.4.2 Firebase Authentication

O primeiro serviço a ser adicionado ao nosso aplicativo será o **Authentication**. Para isso, no assistente do Firebase aberto no passo anterior, selecione a opção **Authentication**, em seguida, **Email and password authentication**.



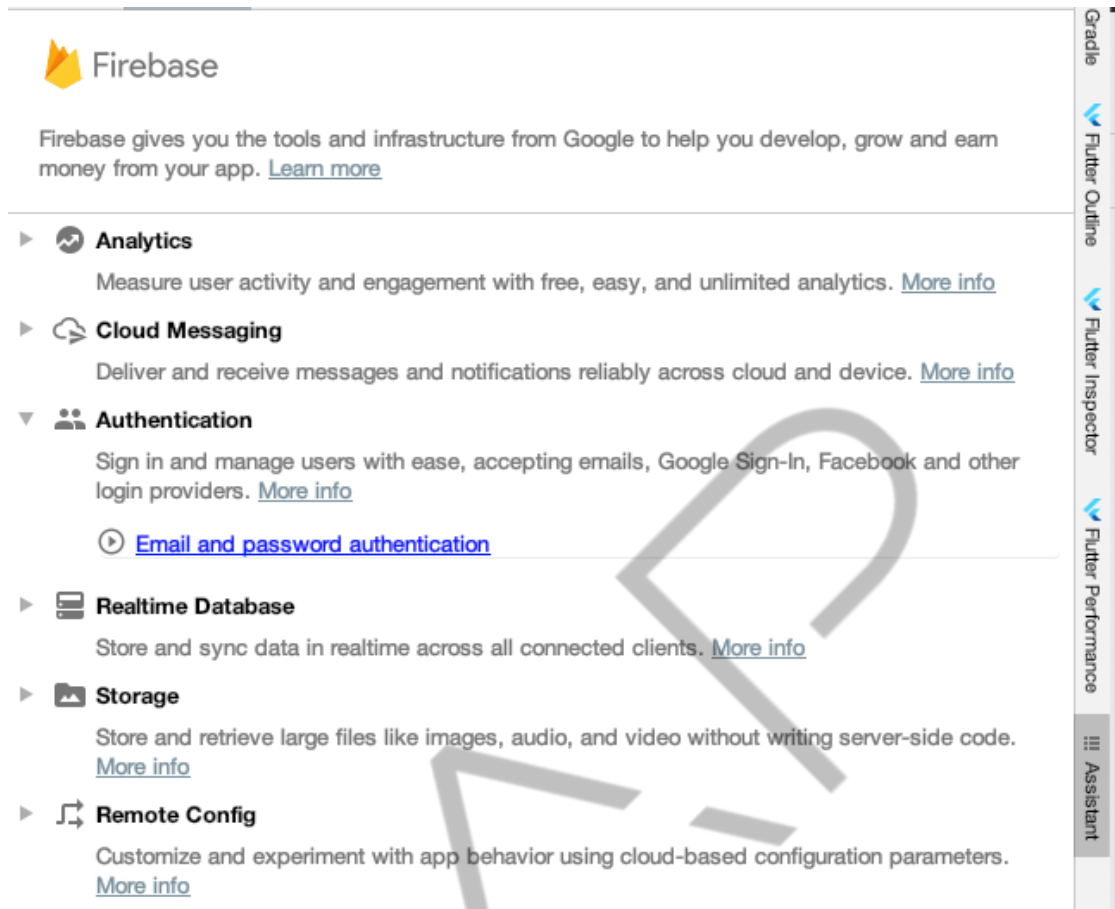


Figura 2.33 - Selecionando o Authentication para integração  
Fonte: Elaborado pelo autor (2020)

Clique sobre o botão **Connect to Firebase**.

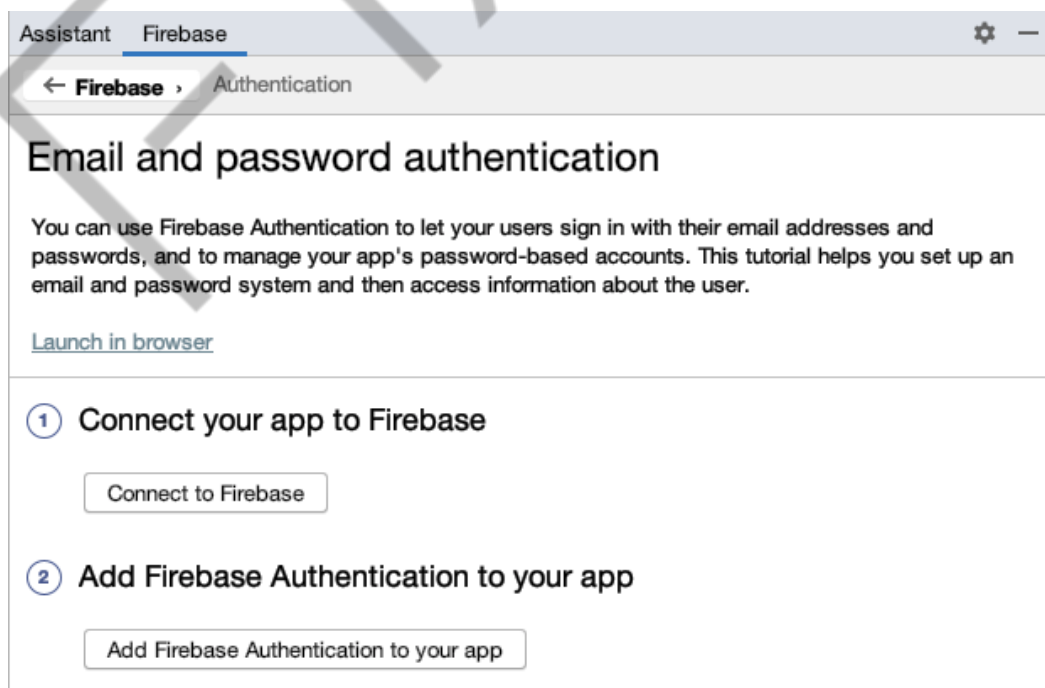


Figura 2.34 - Integrando com o Firebase  
Fonte: Elaborado pelo autor (2020)

Selecione a conta Google que irá realizar a integração com o Firebase.

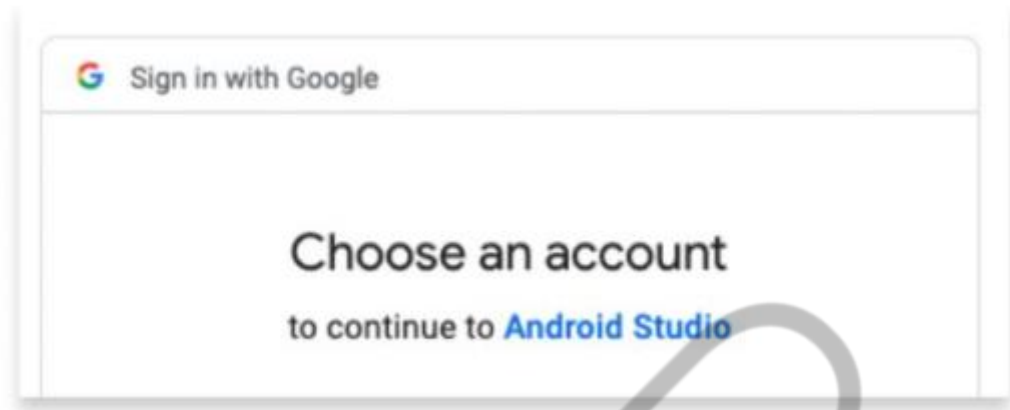


Figura 2.35 - Permissão de acesso 1  
Fonte: Elaborado pelo autor (2020)

Crie projeto com o nome do aplicativo ou utilize um existente.



Figura 2.36 - Permissão de acesso 2  
Fonte: Elaborado pelo autor (2020)

Ao realizar essa integração, o próprio Android Studio se encarregou de baixar e adicionar o arquivo google-services.json. Nesse arquivo, encontram-se as configurações de integração do seu aplicativo com o Firebase.

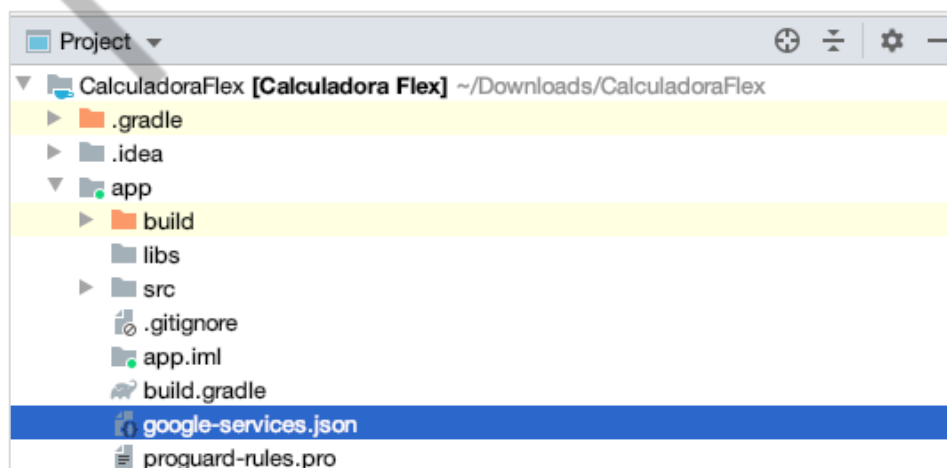


Figura 2.37 - Arquivo de configuração do Firebase  
Fonte: Elaborado pelo autor (2020)

Agora, adicione a biblioteca do Authentication no projeto. Para isso, clique no botão **Add Firebase Authentication to your app**.

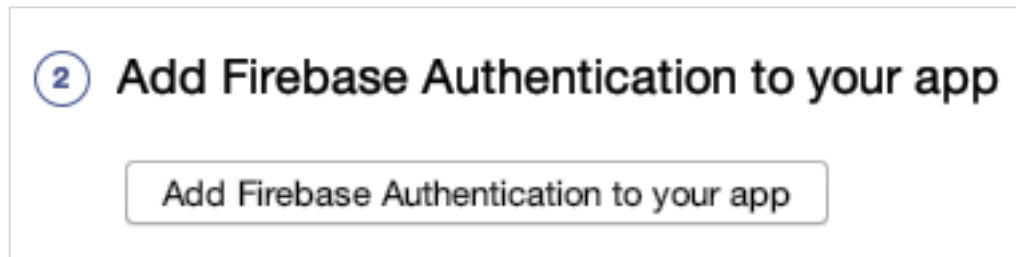


Figura 2.38 - Adicionando a dependência via assistente  
Fonte: Elaborado pelo autor (2020)



Figura 2.39 - Preview das mudanças que serão realizadas para adicionar o Auth  
Fonte: Elaborado pelo autor (2020)

Nem sempre o assistente do Android Studio adiciona a versão mais atualizada da biblioteca. Para garantir que utilizaremos a última versão, abra o arquivo **build.gradle (app)** e atualize a lib.

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"  
    implementation 'androidx.appcompat:appcompat:1.2.0'  
    implementation 'androidx.core:core-ktx:1.3.1'  
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
```

```
implementation 'androidx.legacy:legacy-support-v4:1.0.0'  
  
implementation 'com.google.firebase:firebase-auth:19.3.2'  
  
testImplementation 'junit:junit:4.12'  
androidTestImplementation 'androidx.test.ext:junit:1.1.1'  
androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'  
  
implementation "androidx.navigation:navigation-fragment-ktx:2.3.0"  
implementation "androidx.navigation:navigation-ui-ktx:2.3.0"  
}
```

Código-fonte 2.14 – Atualização da lib de Autenticação  
Fonte: Próprio autor (2020)

Agora, é necessário habilitar o provedor de autenticação que será utilizado no aplicativo. Neste projeto, o método utilizado será E-mail e senha.

Acesse o painel do Firebase através do endereço **console.firebase.google.com**. Selecione o projeto criado.

No painel à esquerda, selecione **Authentication** → **SignIn method** → **Email/senha** → **Ativar** → **Salvar**.

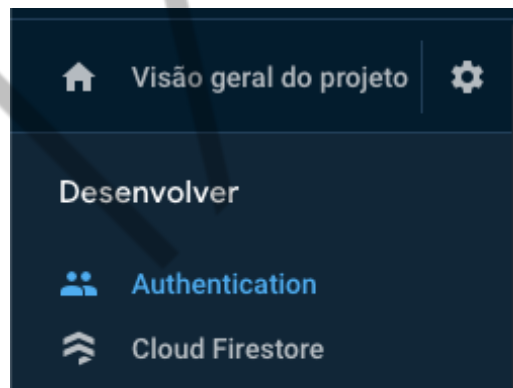


Figura 2.40 - Selecionando o Authentication no console do Firebase  
Fonte: Elaborado pelo autor (2020)

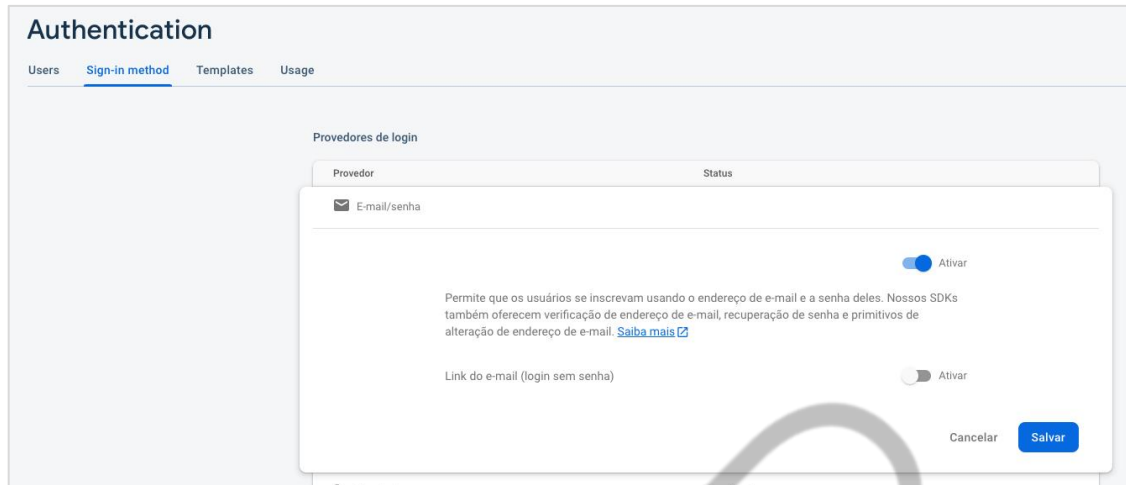


Figura 2.41 - Ativando login por e-mail/senha  
Fonte: Elaborado pelo autor (2020)

Para realizar um teste, clique sobre **Users** → **Adicionar usuário**. Então, crie um usuário que será utilizado para teste no aplicativo.

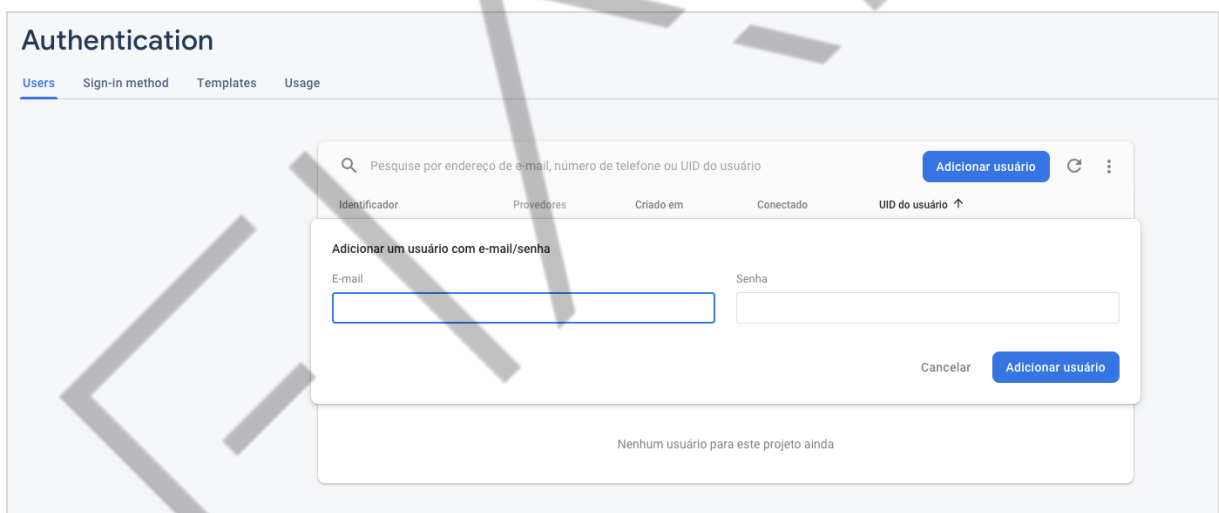


Figura 2.42 - Criação de um usuário para teste  
Fonte: Elaborado pelo autor (2020)

Crie um pacote dentro de **ui** chamado de **base**.

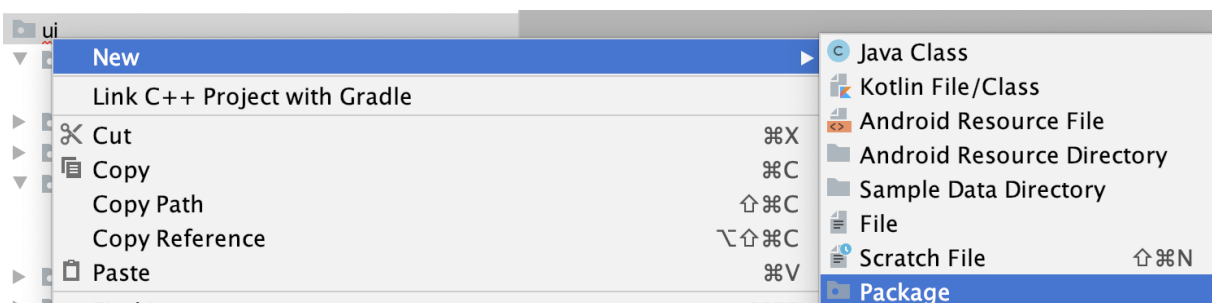


Figura 2.43 - Criação de um novo pacote  
Fonte: Elaborado pelo autor (2020)

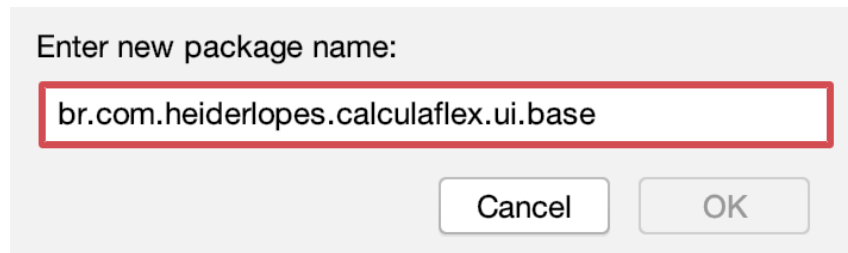


Figura 2.44 - Definição do nome do pacote base  
Fonte: Elaborado pelo autor (2020)

Dentro desse pacote, crie uma classe Kotlin chamada **BaseFragment.kt**. Nessa classe, serão adicionados métodos comuns a serem utilizados pelos fragments.

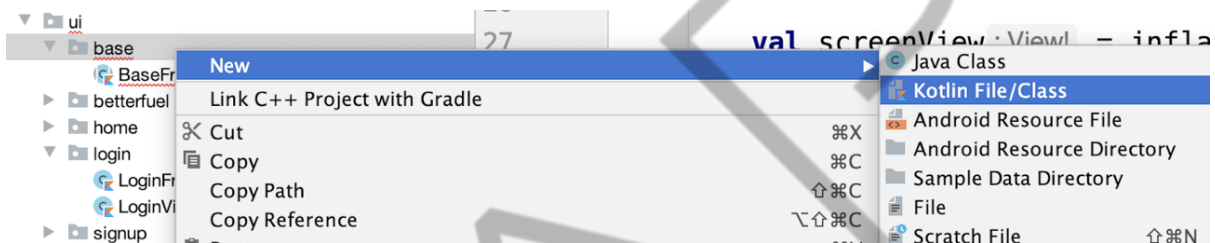


Figura 2.45 - Criação de uma nova classe  
Fonte: Elaborado pelo autor (2020)

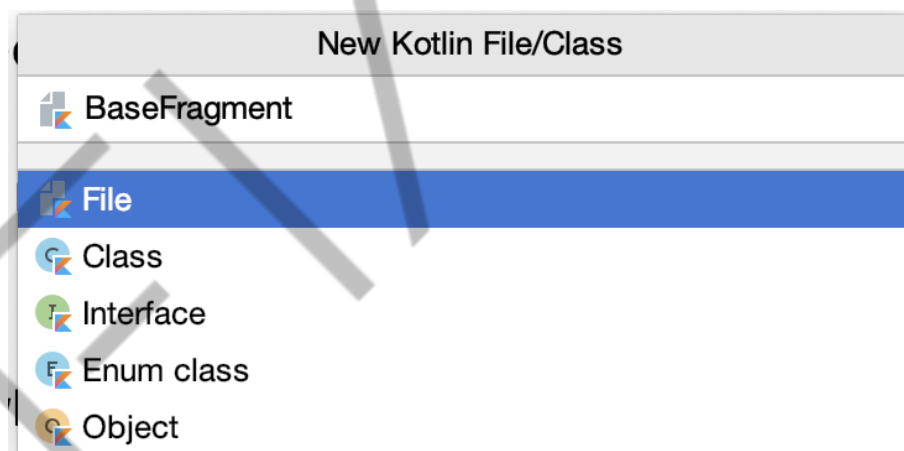


Figura 2.46 - Criação da BaseFragment  
Fonte: Elaborado pelo autor (2020)

Adicione o seguinte código:

```
abstract class BaseFragment : Fragment() {

    abstract val layout: Int

    private lateinit var loadingView: View

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
```

```
val rootView = FrameLayout(requireContext())

val screenView = inflater.inflate(layout, container, false)

loadingView = inflater.inflate(R.layout.include_loading, container, false)

rootView.addView(screenView)
rootView.addView(loadingView)

return rootView
}
fun showLoading(message: String = "Processando a requisição") {
    loadingView.visibility = View.VISIBLE

    if (message.isNotEmpty())
        loadingView.findViewById<TextView>(R.id.tvLoading).text = message
    }

fun hideLoading() {
    loadingView.visibility = View.GONE
}

fun showMessage(message: String?) {
    Toast.makeText(requireContext(), message, Toast.LENGTH_SHORT).show()
}
}
```

Código-fonte 2.15 – Criação de uma nova classe 1  
Fonte: Elaborado pelo autor (2020)

Antes da implementação das classes base de autenticação, é importante saber que, no contexto de uma aplicação MVVM no Android, é muito comum representar operações assíncronas em três possíveis estados: carregando, sucesso e erro.

É possível realizar uma abstração deste conceito utilizando Sealed Class. Neste projeto, será criada uma sealed class chamada Request State que será criada para restringir os três estados possíveis de uma operação. Essa classe é “tipada” para definir o dado que será retornado em caso de sucesso. A palavra **out** é usada para indicar que o dado do tipo T será apenas de saída

O **Loading** é um object, pois ele não traz nenhuma informação adicional.

Em caso de sucesso da requisição, um objeto da classe **Success** será instanciado e o dado obtido deve ser passado como parâmetro.

Quando ocorrer um erro, um **Error** será criado e receberá a exceção/erro como parâmetro.

Para criar a sealed class. Clique com o botão direito do mouse sobre o pacote principal da aplicação → New → Package e adicione um novo, chamado models.

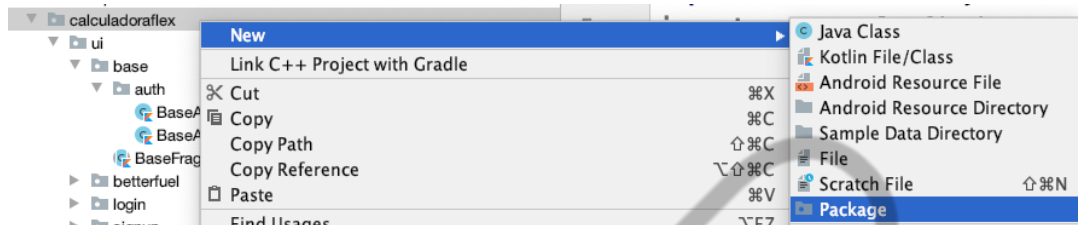


Figura 2.48 – Criação de um novo pacote  
Fonte: Elaborado pelo autor (2020)

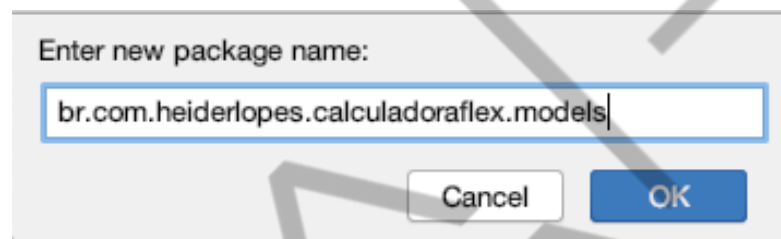


Figura 2.47 - Criação do pacote models  
Fonte: Elaborado pelo autor (2020)

Agora, clique com o botão direito do mouse sobre o pacote models → New → Kotlin File/Class.

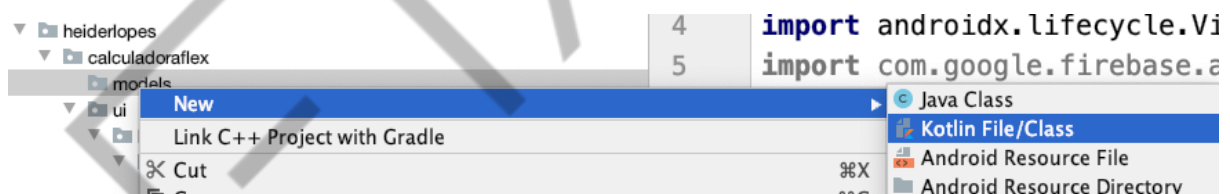


Figura 2.48 - Criação de uma nova classe 1  
Fonte: Elaborado pelo autor (2020)

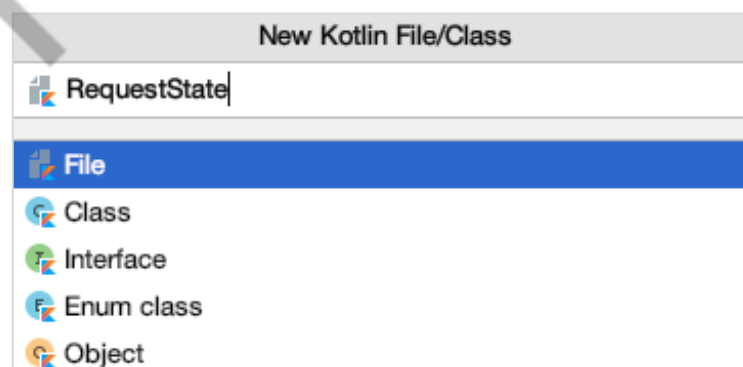


Figura 2.49 - Criação de uma nova classe 2  
Fonte: Elaborado pelo autor (2020)

Adicione o seguinte código conforme, explicado anteriormente:



```
sealed class RequestState<out T> {
    object Loading : RequestState<Nothing>()
    data class Success<T>(val data: T) : RequestState<T>()
    data class Error(val throwable: Throwable) : RequestState<Nothing>()
}
```

Código-fonte 2.16 – Criação de uma nova classe 2  
Fonte: Elaborado pelo autor (2020)

Dentro desse pacote base, crie um package chamado de **auth**. Dentro dele, crie duas classes Kotlin chamadas **BaseAuthFragment.kt** e **BaseAuthViewModel**.

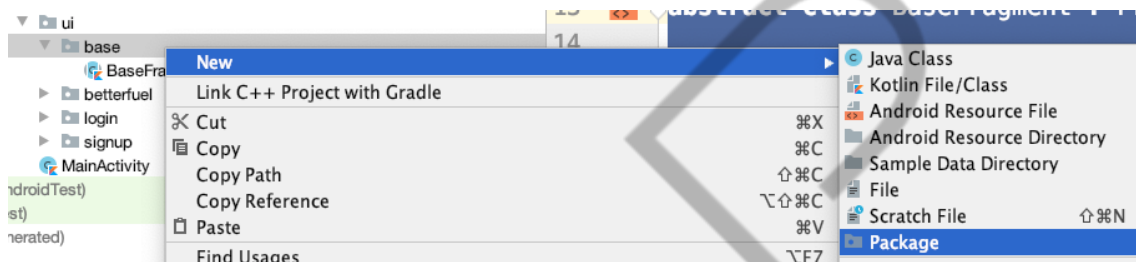


Figura 2.50 - Criação de um novo pacote  
Fonte: Elaborado pelo autor (2020)

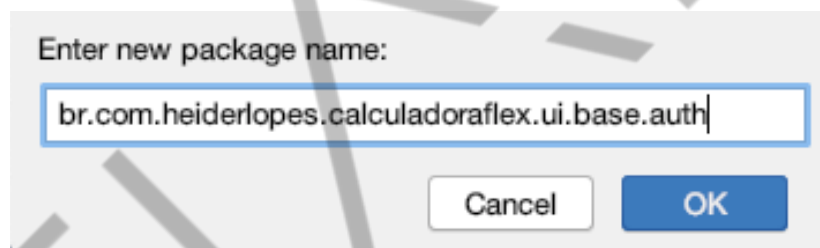


Figura 2.51 - Pacote de autenticação  
Fonte: Elaborado pelo autor (2020)

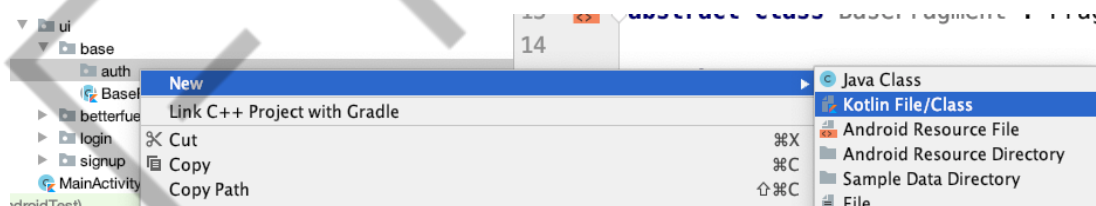


Figura 2.52 - Pacote de autenticação  
Fonte: Elaborado pelo autor (2020)

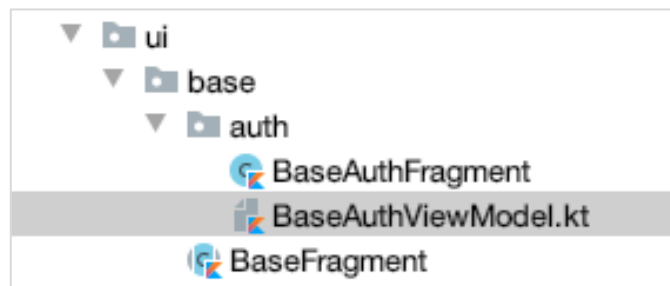


Figura 2.53 - Pacote de autenticação  
Fonte: Elaborado pelo autor (2020)

Na **BaseAuthViewModel**, adicionaremos o seguinte código:

```
class BaseAuthViewModel : ViewModel() {  
  
    private var mAuth: FirebaseAuth = FirebaseAuth.getInstance()  
  
    val loggedState = MutableLiveData<RequestState<FirebaseUser>>()  
  
    fun isLoggedIn() {  
        mAuth.currentUser?.reload()  
  
        val user = mAuth.currentUser  
        loggedState.value = RequestState.Loading  
        if (user == null) {  
            loggedState.value = RequestState.Error(Throwable("Usuário deslogado"))  
        } else {  
            loggedState.value = RequestState.Success(user)  
        }  
    }  
}
```

Código-fonte 2.17 – Código da classe BaseAuthViewModel 1  
Fonte: Elaborado pelo autor (2020)

Nesse código, observe que está sendo utilizada a instância do **FirebaseAuth**. O método **isLoggedIn**, quando chamado, realiza o **reload** para garantir que os dados do usuário estão atualizados (caso ele seja removido ou perca permissão no servidor, precisamos fazer essa atualização). Em seguida, verificamos se o usuário está logado ou não. Por meio do **MutableLiveData**, a aplicação será avisada e, caso necessário, o usuário poderá receber a tela para pedir novamente o usuário e a senha.

O próximo passo é implementar o BaseAuthFragment. Por meio dele, todas as telas que precisarem de usuário autenticado poderão herdar essa implementação, ou seja, caso o usuário não esteja logado, serão solicitadas as suas credenciais. Observe, no código abaixo que, caso a requisição de login dê algum erro, o usuário será redirecionado para a tela de login por meio do `findNavController().navigate`.

```
const val NAVIGATION_KEY = "NAV_KEY"  
  
abstract class BaseAuthFragment : BaseFragment() {  
  
    private val baseAuthViewModel: BaseAuthViewModel by viewModels()  
  
    override fun onCreateView(  

```

```

inflater: LayoutInflater,
container: ViewGroup?,
savedInstanceState: Bundle?
): View? {

    registerObserver()
    baseAuthViewModel.isLoggedIn()

    return super.onCreateView(inflater, container, savedInstanceState)
}

private fun registerObserver() {
    baseAuthViewModel.loggedState.observe(viewLifecycleOwner, Observer {
        when (it) {
            is RequestState.Loading -> showLoading()
            is RequestState.Success -> {
                hideLoading()
            }
            is RequestState.Error -> {
                hideLoading()
            }
        }
        findNavController().navigate(
            R.id.logInFragment, bundleOf(
                NAVIGATION_KEY to findNavController().currentDestination?.id
            )
        )
    })
}
}
}
}
}
}
}

```

Código-fonte 2.18 – Código da classe BaseAuthViewModel 2  
 Fonte: Elaborado pelo autor (2020)

Abra o arquivo BetterFuelFragment e herde da classe BaseAuthFragment.

```

class BetterFuelFragment : BaseAuthFragment() {

    override val layout: Int
    get() = R.layout.fragment_better_fuel

}

```

Código-fonte 2.19 – Código da classe BaseAuthViewModel 3  
 Fonte: Elaborado pelo autor (2020)

Rode o aplicativo e observe que solicitará o login.



Figura 2.54 - Aplicativo solicitando o login  
Fonte: Elaborado pelo autor (2020)

Dentro do pacote de login, crie uma classe chamada `LoginViewModel` e adicione o seguinte código-fonte: `LoginViewModel`. Observe que o código de `signIn` receberá o e-mail e a senha. Ele executará o método para validar os campos, caso algum possua erro, o `loginState` será atualizado e a tela de login, que está observando esse objeto, reagirá exibindo a mensagem para o usuário.

```
class LoginViewModel : ViewModel() {  
  
    private var mAuth = FirebaseAuth.getInstance()  
  
    val loginState = MutableLiveData<RequestState<FirebaseUser>>()  
    val resetPasswordState = MutableLiveData<RequestState<String>>()  
  
    fun signIn(email: String, password: String) {  
  
        loginState.value = RequestState.Loading  
  
        if (validateFields(email, password)) {  
            mAuth.signInWithEmailAndPassword(email, password)  
                .addOnCompleteListener {  
                if (it.isSuccessful) {  
                    loginState.value = RequestState.Success(mAuth.currentUser!!)  
                } else {  
                    loginState.value = RequestState.Error(  
                        Throwable(  

```

```

        it.exception?.message ?: "Não foi possível realizar a requisição"
    )
    )
    }
}

private fun validateFields(email: String, password: String): Boolean {

    if(email.isEmpty()) {
        loginState.value = RequestState.Error(Throwable("E-mail não pode ser vazio"))
        return false
    }

    if(password.isEmpty()) {
        loginState.value = RequestState.Error(Throwable("Senha não pode ser vazia"))
        return false
    }

    if(password.length < 6) {
        loginState.value = RequestState.Error(Throwable("Senha tem que ter pelo
menos 6 caracteres"))
        return false
    }

    return true
}
}

```

Código-fonte 2.20 – 2.59 – Código do LoginViewModel  
 Fonte: Elaborado pelo autor (2020)

Agora, será implementado o código referente à tela de login.

```

class LoginFragment : BaseFragment() {

    override val layout = R.layout.fragment_log_in

    private lateinit var tvSubTitleSignUp: TextView
    private lateinit var containerLogin: LinearLayout
    private lateinit var tvResetPassword: TextView
    private lateinit var tvNewAccount: TextView

    private lateinit var btLogin: Button
    private lateinit var etEmailLogin: EditText
    private lateinit var etPasswordLogin: EditText

    private val loginViewModel: LoginViewModel by viewModels()

    override fun onCreateView(view: View, savedInstanceState: Bundle?) {
        super.onCreateView(view, savedInstanceState)

        setUpView(view)
    }
}

```

```
registerObserver()
registerBackPressedAction()
}

private fun registerBackPressedAction() {
    val callback = object : OnBackPressedCallback(true) {
        override fun handleOnBackPressed() {
            activity?.finish()
        }
    }
    requireActivity().onBackPressedDispatcher.addCallback(callback)
}

private fun registerObserver() {
    loginViewModel.loginState.observe(viewLifecycleOwner, Observer {
        when (it) {
            is RequestState.Success -> showSuccess()
            is RequestState.Error -> showError(it.throwable)
            is RequestState.Loading -> showLoading("Realizando a autenticação")
        }
    })

    loginViewModel.resetPasswordState.observe(viewLifecycleOwner, Observer {
        when (it) {
            is RequestState.Success -> {
                hideLoading()
                showMessage(it.data)
            }
            is RequestState.Error -> showError(it.throwable)
            is RequestState.Loading -> showLoading("Reenviando o e-mail para troca de
senha")
        }
    })
}

private fun showSuccess() {
    hideLoading()
    val navIdForArguments = arguments?.getInt(NAVIGATION_KEY)
    if (navIdForArguments == null) {
        findNavController().navigate(R.id.main_nav_graph)
    } else {
        findNavController().popBackStack(navIdForArguments, false)
    }
}

private fun showError(throwable: Throwable) {
    hideLoading()
    showMessage(throwable.message)
}

private fun setUpView(view: View) {
    tvSubTitleSignUp = view.findViewById(R.id.tvSubTitleLogin)
    containerLogin = view.findViewById(R.id.containerLogin)
    tvResetPassword = view.findViewById(R.id.tvResetPassword)
}
```

```

tvNewAccount = view.findViewById(R.id.tvNewAccount)

btLogin = view.findViewById(R.id.btLogin)
etEmailLogin = view.findViewById(R.id.etEmailLogin)
etPasswordLogin = view.findViewById(R.id.etPasswordLogin)

btLogin.setOnClickListener {
    loginViewModel.signIn(
        etEmailLogin.text.toString(),
        etPasswordLogin.text.toString()
    )
}
}
}

```

Código-fonte - 2.60 – Código do LoginFragment  
 Fonte: Elaborado pelo autor (2020)

### 2.4.3 Reset de senha

Para implementar o reset de senha, o usuário irá digitar seu e-mail e clicar no botão Esqueci minha senha. Com isso, o Firebase irá disparar um e-mail com o link para o usuário redefini-la.

A primeira parte da implementação deverá ser realizada dentro do arquivo LoginViewModel. Dentro desse arquivo adicione o seguinte código:

```

fun resetPassword(email: String) {
    resetPasswordState.value = RequestState.Loading
    if(email.isNotEmpty()) {
        FirebaseAuth.getInstance().sendPasswordResetEmail(email)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                resetPasswordState.value = RequestState.Success("Verifique sua caixa
de e-mail")
            } else {
                resetPasswordState.value = RequestState.Error(
                    Throwable(
                        task.exception?.message ?: "Não foi possível realizar a requisição"
                    )
                )
            }
        }
    } else {
        resetPasswordState.value = RequestState.Error(Throwable("Não foi possível
resetar a senha"))
    }
}

```

Código-fonte 2.21 – Código de resetar a senha 1

Fonte: Elaborado pelo autor (2020)

Agora, para executar o método ao clicar no Esqueci minha senha, abra o arquivo LoginFragment e adicione o seguinte código dentro do método **setUpView**.

```
tvResetPassword.setOnClickListener {  
    loginViewModel.resetPassword(  
        etEmailLogin.text.toString()  
    )  
}
```

Código-fonte 2.22 – Código de resetar a senha 2  
Fonte: Elaborado pelo autor (2020)

#### 2.4.4 Logout

Crie uma classe dentro do pacote **betterfuel** chamada **BetterFuelViewModel** e adicione o método “realizar o **logout** do aplicativo”.

```
class BetterFuelViewModel : ViewModel() {  
  
    private var mAuth: FirebaseAuth = FirebaseAuth.getInstance()  
  
    val loggedState = MutableLiveData<RequestState<Boolean>>()  
  
    fun logout() {  
        loggedState.value = RequestState.Loading  
        mAuth.signOut()  
        loggedState.value = RequestState.Success(true)  
    }  
}
```

Código-fonte 2.23 – Código para realizar o logout  
Fonte: Elaborado pelo autor (2020)

Abra o arquivo **BetterFuelFragment** e adicione o seguinte código, em que:

**setUpView** → Aqui, é realizado o bind com o layout xml.

**betterFuelViewModel** → Declaração do objeto que representa a nossa ViewModel. Aqui, foi declarado o método de logout.

No registerObserver será declarado o observer que direcionará o usuário para tela de login quando ele for deslogado. Veja o código no código-fonte “Código para realizar o logout e setup das views”.

```
class BetterFuelFragment : BaseAuthFragment() {
```



```
private val betterFuelViewModel: BetterFuelViewModel by viewModels()

override val layout: Int
    get() = R.layout.fragment_better_fuel

private lateinit var etCar: EditText
private lateinit var etKmGasoline: EditText
private lateinit var etKmEthanol: EditText
private lateinit var etPriceGasoline: EditText
private lateinit var etPriceEthanol: EditText

private lateinit var btSignOut: AppCompatActivity
private lateinit var btCalculate: Button
private lateinit var btClear: TextView

override fun onCreateView(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    setUpView(view)

    registerObserver()
}

private fun setUpView(view: View) {
    etCar = view.findViewById(R.id.etCar)
    etKmGasoline = view.findViewById(R.id.etKmGasoline)
    etKmEthanol = view.findViewById(R.id.etKmEthanol)
    etPriceGasoline = view.findViewById(R.id.etPriceGasoline)
    etPriceEthanol = view.findViewById(R.id.etPriceEthanol)

    btSignOut = view.findViewById(R.id.btSignOut)
    btCalculate = view.findViewById(R.id.btCalculate)
    btClear = view.findViewById(R.id.btClear)

    btCalculate.setOnClickListener {

    }

    btClear.setOnClickListener {

    }

    btSignOut.setOnClickListener {
        betterFuelViewModel.logout()
    }
}

private fun registerObserver() {
    this.betterFuelViewModel.loggedState.observe(viewLifecycleOwner,
```

```
Observer {  
    when (it) {  
        is RequestState.Success -> {  
            hideLoading()  
            NavHostFragment.findNavController(this)  
                .navigate(R.id.loginFragment)  
        }  
        is RequestState.Error -> {  
            hideLoading()  
            showMessage(it.throwable.message)  
        }  
        is RequestState.Loading -> showLoading("Calculando o melhor  
combustível")  
    }  
}  
})  
}
```

Código-fonte 2.24 – Código para realizar o logout e setup das views  
Fonte: Elaborado pelo autor (2020)

O projeto completo pode ser baixado neste repositório:  
<https://github.com/FIAPON/CalculaFlexAndroid>.

## CONCLUSÃO

Neste capítulo, conhecemos como um processo de autenticação funciona e como integrá-lo ao Firebase utilizando o serviço próprio chamado Authentication. Com isso, os usuários já podem ser autorizados antes de entrar na aplicação. Nosso próximo passo é armazenar os dados de usuário utilizando uma estrutura de persistência em nuvem no Firebase.

EXEMPLO

## REFERÊNCIAS

AIRBNB. **Lottie**. EUA. 2020. Disponível em: <<https://airbnb.io/lottie/#/>>. Acesso em: 10 set. 2020.

GOOGLE. **Documentação do Firebase**. EUA. 2020. Disponível em: <<https://firebase.google.com/docs/auth?hl=pt-br>>. Acesso em: 10 set. 2020.

OAUT. **Documentação do OAuth 2.0**. EUA. 2020. Disponível em: <<https://oauth.net/2/>>. Acesso em: 10 set. 2020.

EXEMPLO