
Ame
FastForward

O que veremos hoje?

Agenda de hoje

- **Turbinando a aplicação com o Firebase**
  - Analytics
  - Crashlytics
  - Cloud Messaging
  - Remote Config
  - Storage
  - Cloud Functions
  - Realtime Database

Conhecendo nosso projeto

APP: Calculadora Flex

O cálculo do rendimento do carro é importante, pois o motorista poderá verificar também qual combustível é mais econômico em função do preço na bomba.

Existe a convenção de que o etanol é mais econômico se custar **até 70% do preço** da gasolina ou **30% mais barato** (baseado no teste do Inmetro).

Mas testes atuais realizado pelo Instituto Mauá o motorista poderá verificar que seu carro rende muito mais e então economizará usando etanol mesmo se o percentual estiver acima de 70%, podendo chegar até 75%!

APP: Calculadora Flex - Como o cálculo é feito?

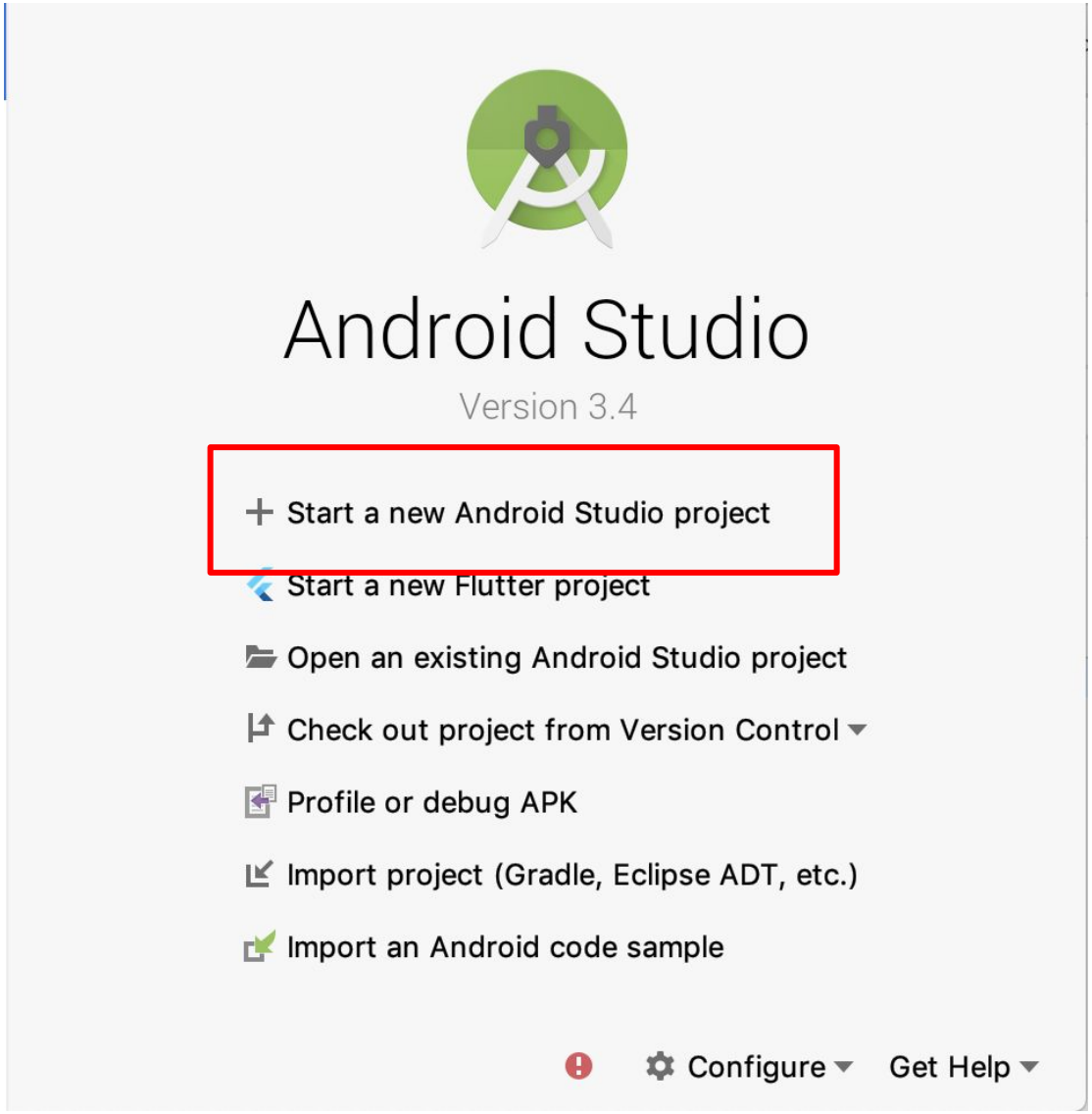
1 - Divida o desempenho do etanol pelo desempenho da gasolina (se seu carro faz 7,3 km/litro com etanol e 10 km/l com gasolina, você deve dividir 7,3 por 10 = a 0,73 ou 73%. Pronto você achou o rendimento do carro com etanol!

2- Faça agora o cálculo da relação de preço etanol/gasolina na bomba: divida o preço do etanol pelo preço da gasolina (ex: atualmente o litro do etanol está em R$ 2,74 e da gasolina R$ 4,64 = relação, então, de 0,59 ou 59%).

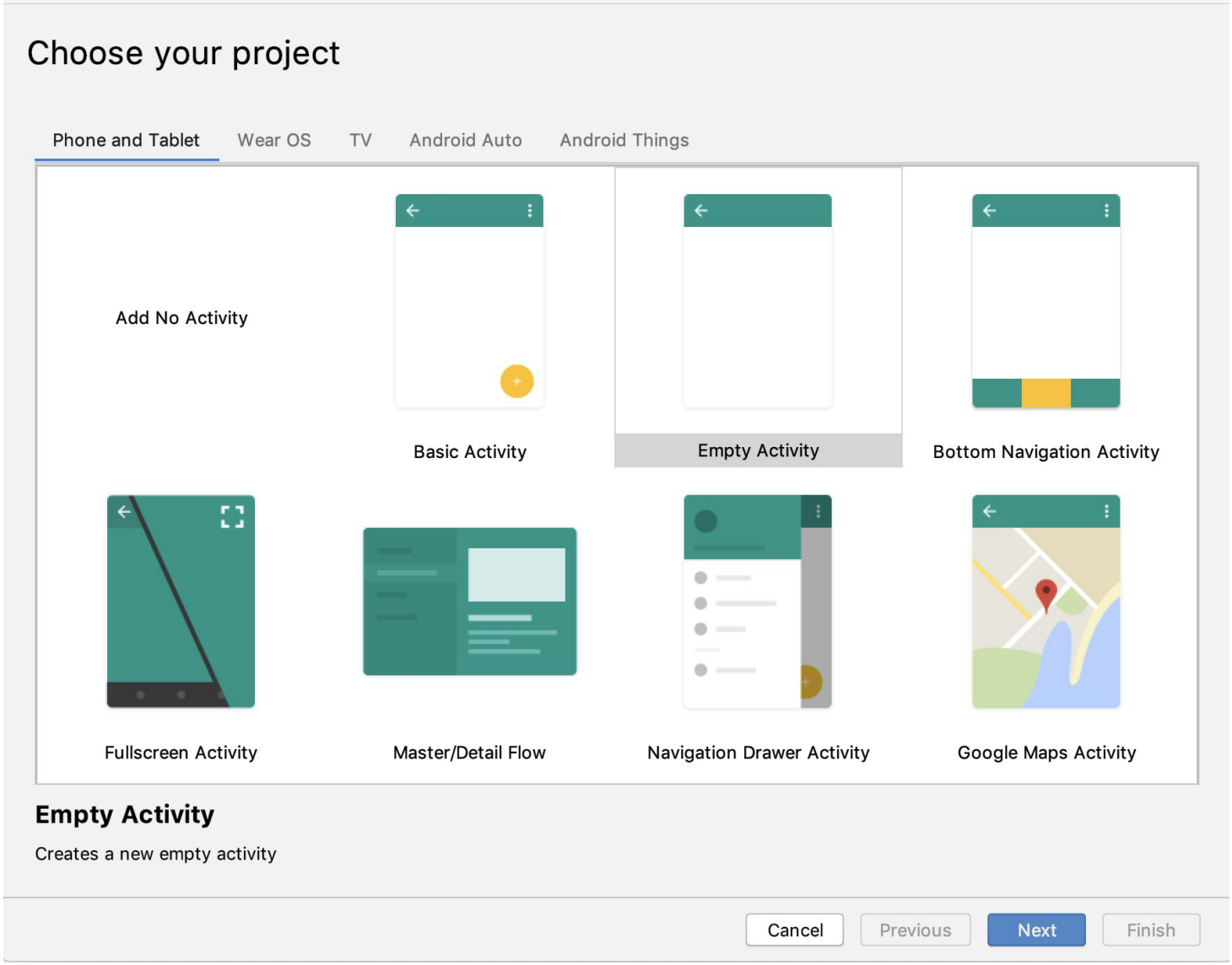
3 - A relação atual de preço na bomba (59%) dá uma enorme economia ao consumidor se abastecer com etanol e quando este cálculo estiver em 73%, pelo rendimento do carro neste exemplo, também estará economizando se usar etanol.

Criando nosso projeto

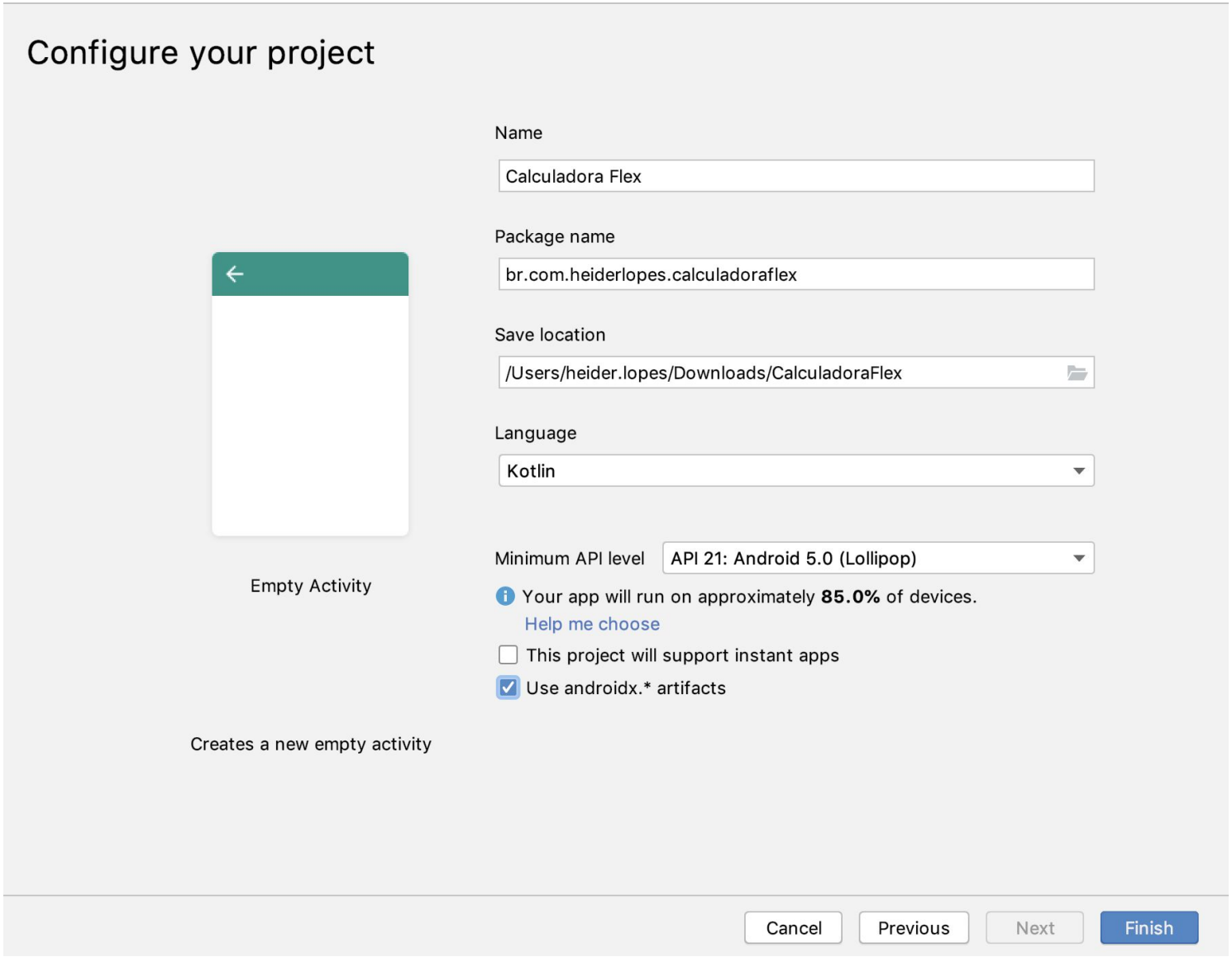
APP: Calculadora Flex - Criando o projeto



APP: Calculadora Flex - Criando o projeto



APP: Calculadora Flex - Criando o projeto

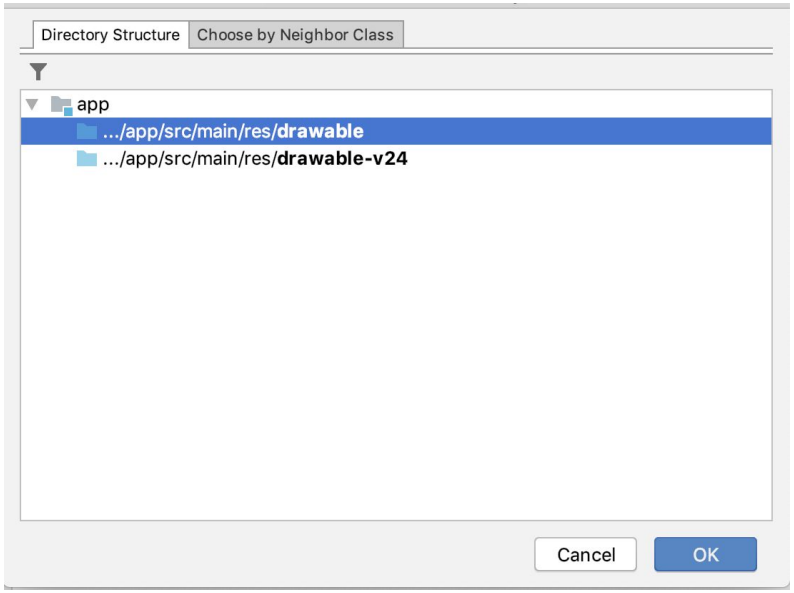


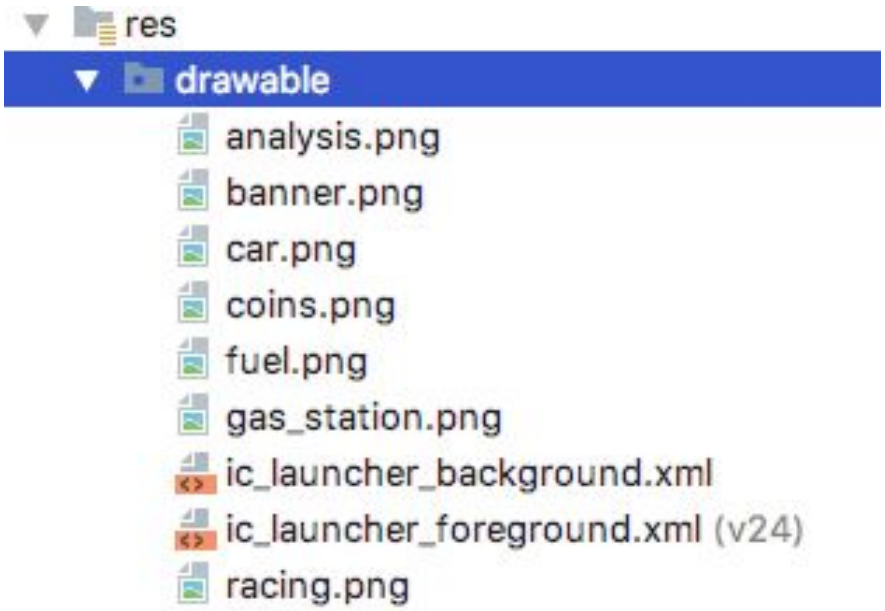
APP: Calculadora Flex - Drawables

Baixar as imagens em:

https://github.com/heiderlopes/calculadora_flex_res/tree/master/res/drawable

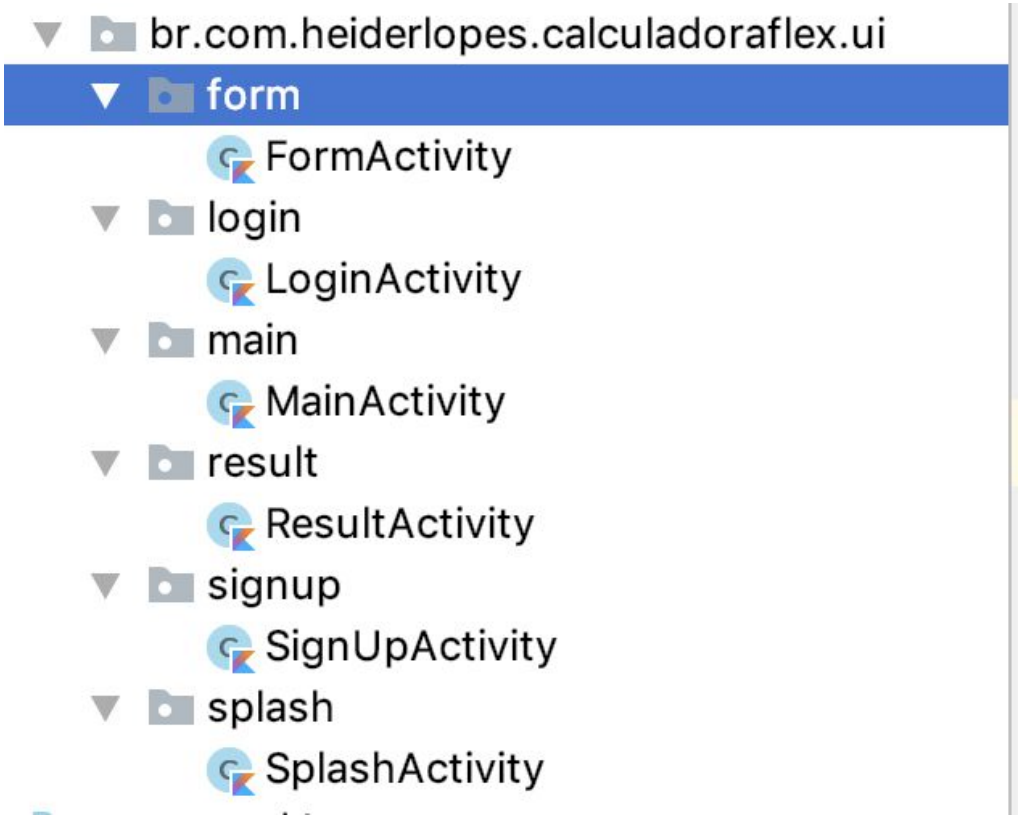
Adicionar as imagens na pasta **drawable**





APP: Calculadora Flex - UI

Crie as seguintes Activities (**Empty Activity**) com os seguintes nomes e dentro dos seus respectivos packages:



Os layouts e resources já estão disponíveis em:

https://github.com/heiderlopes/calculadora_flex_res

Hora do desafio

APP: Calculadora Flex - UI

Hora de praticar: Desenvolva o layout das telas abaixo:

E-mail

Senha

ENTRAR

Ainda não tem conta? Crie uma

Novo Usuário

Nome

E-mail

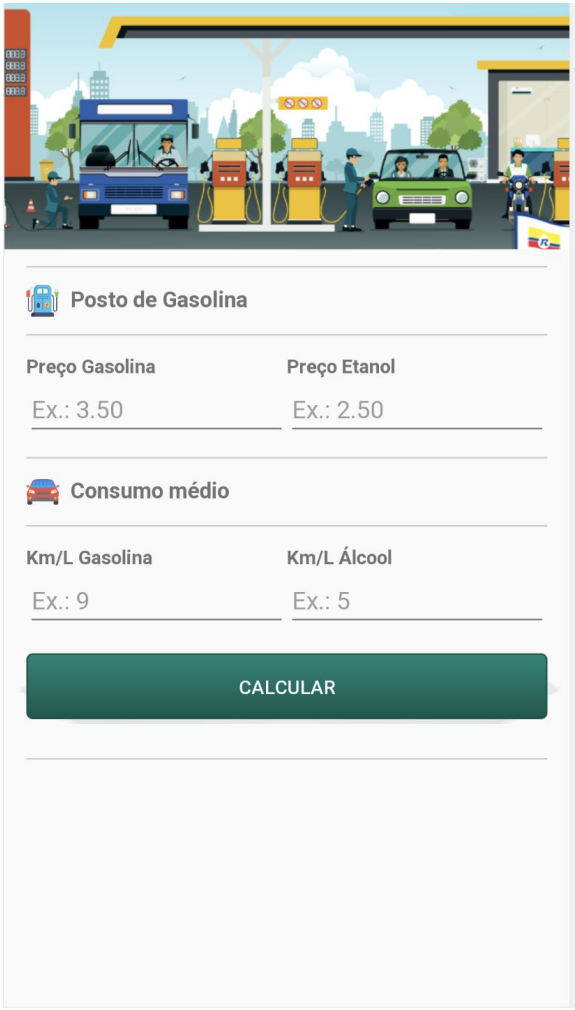
Senha

Telefone

AINDA NÃO TEM CONTA? CRIE UMA

APP: Calculadora Flex - UI

Hora de praticar: Desenvolva o layout das telas abaixo:



Criando nossa SplashScreen

APP: Calculadora Flex - SplashScreen

Abrir o arquivo **styles.xml**

```
<resources>

    // Nao é necessário apagar o código existente

      <!-- FullScreen theme. -->
      <style name="FullScreen" parent="AppTheme">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
        <item name="android:windowFullscreen">true</item>
      </style>


</resources>
```

APP: Calculadora Flex - AndroidManifest.xml

Configure seu AndroidManifest.xml da seguinte forma:

```
<activity
      android:name=".MainActivity"
      android:label="@string/app_name"
      android:theme="@style/AppTheme.NoActionBar">

</activity>
<activity android:name=".SplashActivity"
       android:theme="@style/FullScreen">
  <intent-filter>
      <action android:name="android.intent.action.MAIN"/>

      <category android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>
</activity>
```

APP: Calculadora Flex - AndroidManifest.xml

Na linha marcada abaixo definimos que a SplashScreen irá utilizar o tema FullScreen criado anteriormente

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name"
    android:theme="@style/AppTheme.NoActionBar">

</activity>
<activity android:name=".SplashActivity"
    android:theme="@style/FullScreen">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

APP: Calculadora Flex - AndroidManifest.xml

Na linha marcada abaixo definimos que a SplashScreen deverá ser aberta ao iniciar o aplicativo.

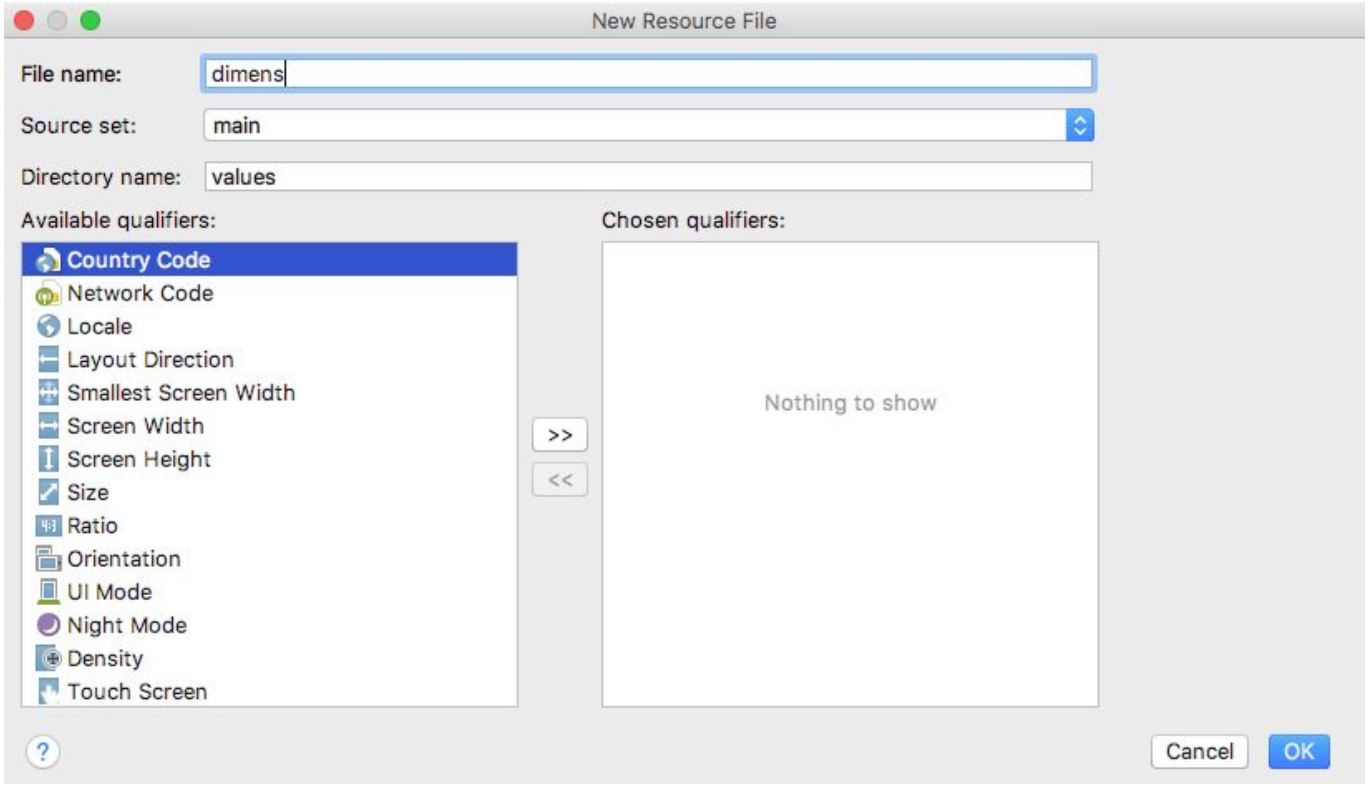
```
<activity
      android:name=".MainActivity"
      android:label="@string/app_name"
      android:theme="@style/AppTheme.NoActionBar">

</activity>
<activity android:name=".SplashActivity"
       android:theme="@style/FullScreen">
  <intent-filter>
     <action android:name="android.intent.action.MAIN"/>

     <category android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>
</activity>
```

APP: Calculadora Flex - dimens.xml

Crie um arquivo para na pasta values.

Botão direito sobre a pasta **values,** em seguida, **new ⇒ Values resource file.**

Chame o arquivo como **dimens.xml**



APP: Calculadora Flex - dimens.xml

Configure o arquivo **dimens.xml** conforme abaixo:

```
<resources>
    <dimen name="fab_margin">16dp</dimen>
    <dimen name="icon_header_size">24dp</dimen>
    <dimen name="icon_header_margin">8dp</dimen>
</resources>
```

APP: Calculadora Flex - colors.xml

Abra o arquivo **colors.xml** e adicione a nova linha em negrito

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="colorPrimary">#008577</color>
  <color name="colorPrimaryDark">#00574B</color>
  <color name="colorAccent">#4CAF50</color>
  <color name="background_splash">#FFFFFF</color>
</resources>
```

Você pode utilizar o site **https://www.materialpalette.com/** para ajudar na combinação de cores para o aplicativo.

APP: Calculadora Flex - colors.xml

Abra o arquivo **strings.xml** e adicione a nova linha em negrito

```
<resources>
    <string name="app_name">Calculadora Flex</string>
    <string name="header_gas_station">Posto de Gasolina</string>
    <string name="header_gasoline_price">Preço Gasolina</string>
    <string name="header_ethanol_price">Preço Etanol</string>
    <string name="header_average">Consumo médio</string>
    <string name="header_average_gasoline">Km/L Gasolina</string>
    <string name="header_average_ethanol">Km/L Álcool</string>
    <string name="hint_gasoline">Ex.: 3.50</string>
    <string name="hint_ethanol">Ex.: 2.50</string>
    <string name="hint_average_gasoline">Ex.: 9</string>
    <string name="hint_average_ethanol">Ex.: 5</string>
    <string name="calculate">Calcular</string>
    <string name="header_suggestion">Sugestão de Abastecimento</string>
    <string name="header_spent">Gasto R$/Km</string>
    <string name="gasoline">Gasolina</string>
```

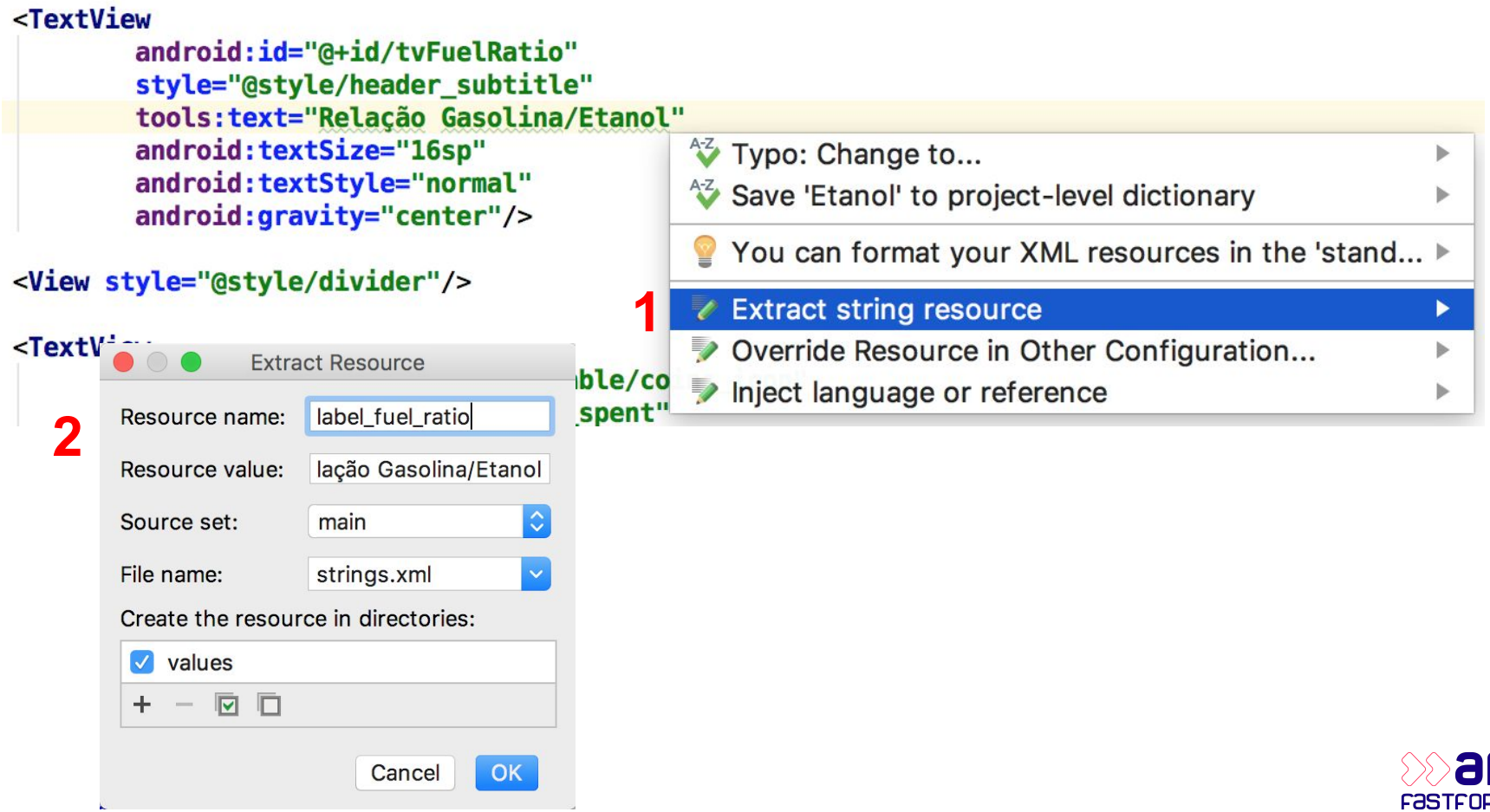
APP: Calculadora Flex - colors.xml

Abra o arquivo **strings.xml** e adicione a nova linha em negrito

```
    <string name="ethanol">Etanol</string>
    <string name="label_fuel_ratio">Relação Gasolina/Etanol: %1$s</string>
    <string name="label_email">E-mail</string>
    <string name="label_password">Senha</string>
    <string name="button_login">Entrar</string>
    <string name="label_new_account">Ainda não tem conta? Crie uma</string>
    <string name="button_new_account">Criar conta</string>
    <string name="label_name">Nome</string>
    <string name="label_phone">Telefone</string>
    <string name="header_new_user">Novo Usuário</string>
</resources>
```

DICA: strings.xml

Para enviar a string para o arquivo **strings.xml** podemos posicionar o cursor na frente do texto e segurar **alt** e apertar o **Enter.** Em seguida, selecione **Extract string resource.**



DICA: strings.xml

Abra o arquivo **strings.xml** e altere a string **fuel_ratio** conforme abaixo:

```
<resources>
    <string name="label_fuel_ratio">Relação Gasolina/Etanol: %1$f</string>
</resources>
```

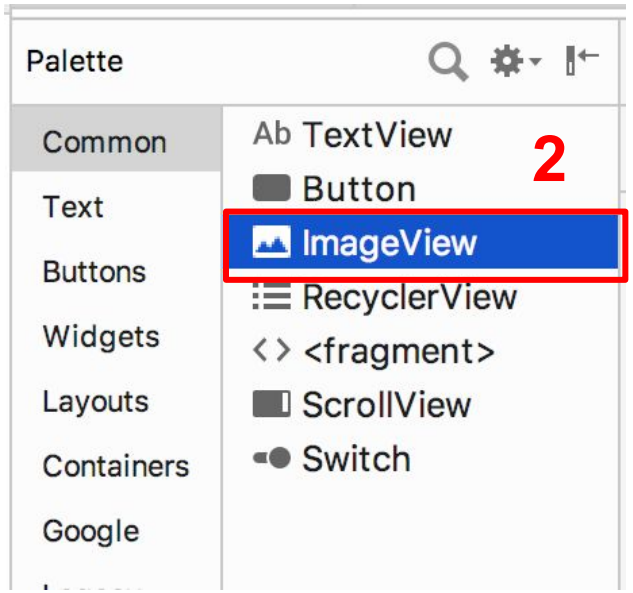
Onde **%1** é referente em que posição será enviado o valor e o **$f** é o tipo do dado que será enviado. Basicamente temos:

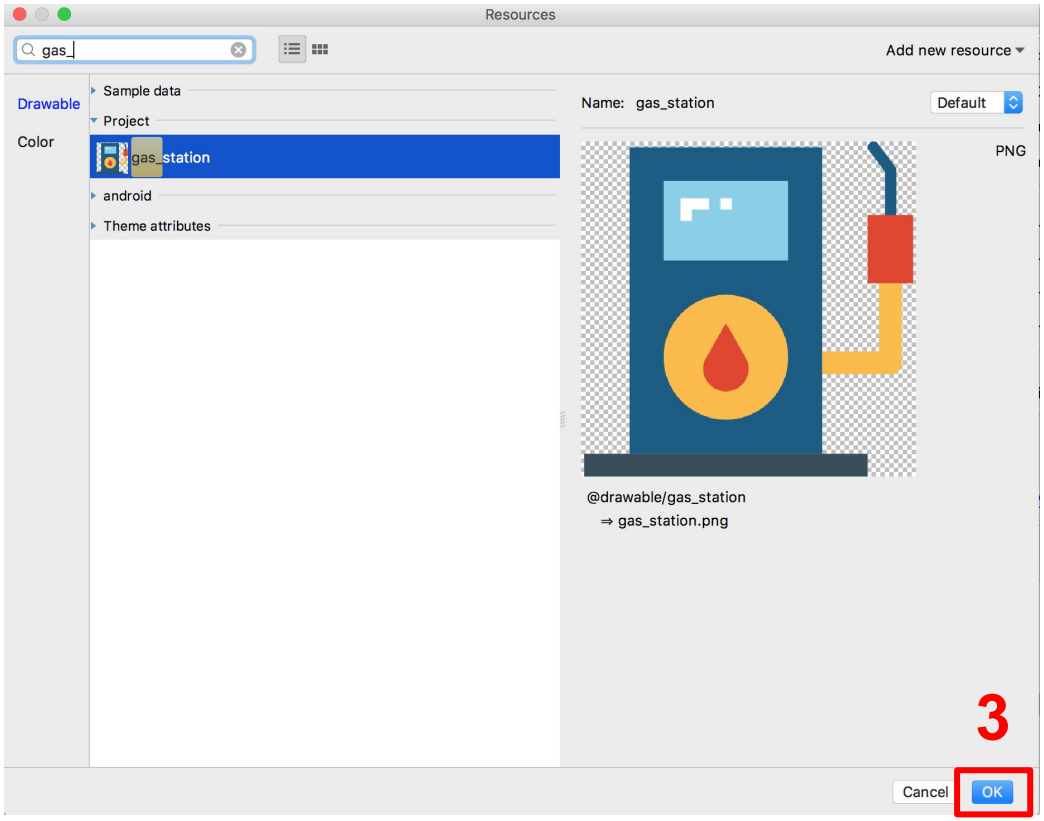
f-> Para valores com ponto flutuante. Por exemplo: double
d ⇒ Para valores inteiros
s⇒ Para string

Para mais informações acesse a documentação oficial:
https://developer.android.com/guide/topics/resources/string-resource?hl=pt-br#FormattingAndStyling

APP: Calculadora Flex - SplashScreen

Abra o arquivo **activity_splash.xml**





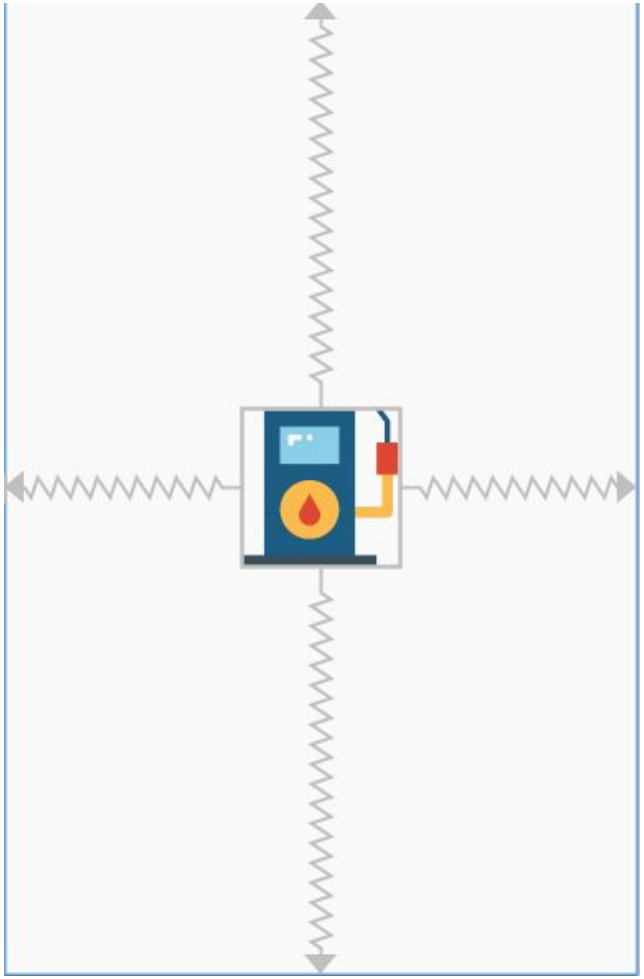
APP: Calculadora Flex - SplashScreen

Abra o arquivo **activity_splash.xml**

```
<?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/ivLogo"
      android:layout_width="96dp"
      android:layout_height="96dp"
      android:src="@drawable/gas_station"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

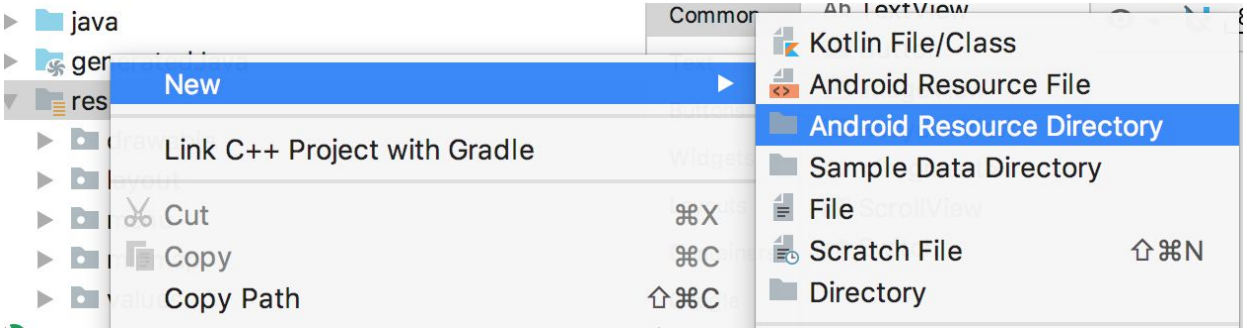
APP: Calculadora Flex - SplashScreen

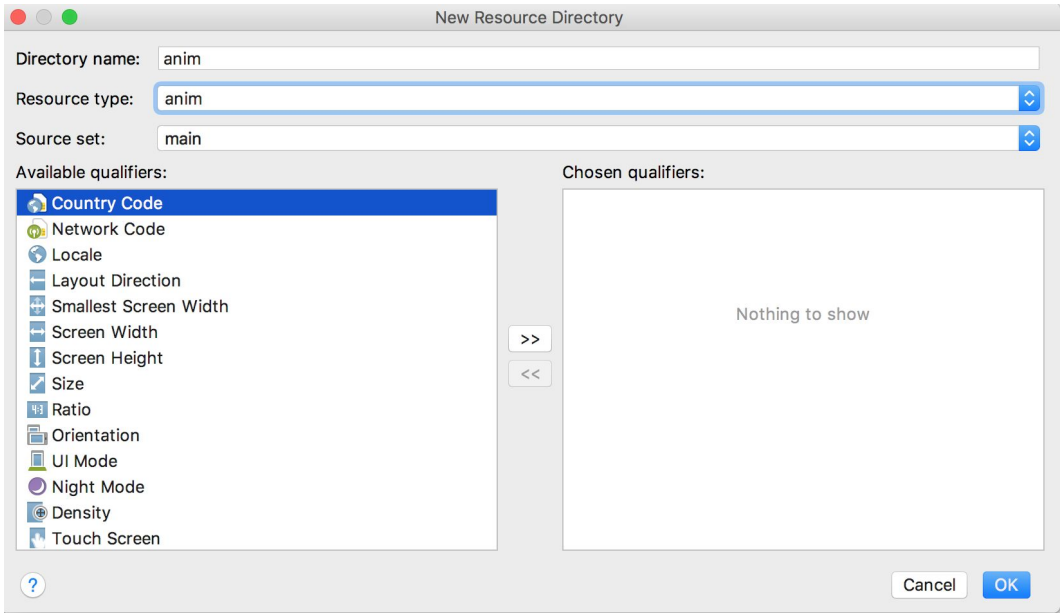
Resultado esperado



APP: Calculadora Flex - SplashScreen - Animação

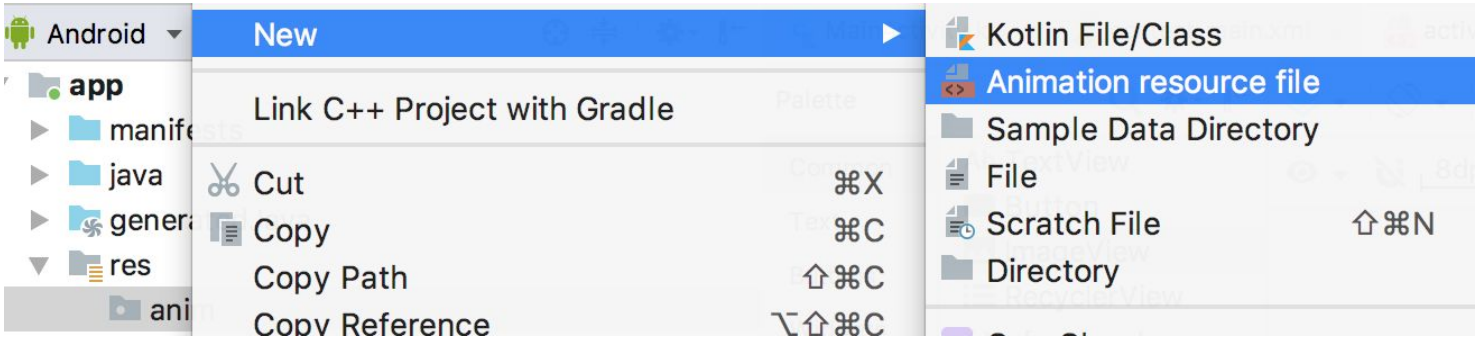
Criar a pasta para armazenar as animações. Botão direito sobre a pasta **res ⇒ New ⇒ Android Resource Directory.** Em seguida, nomeie para **anim.** Altera o **resource type** para **anim**

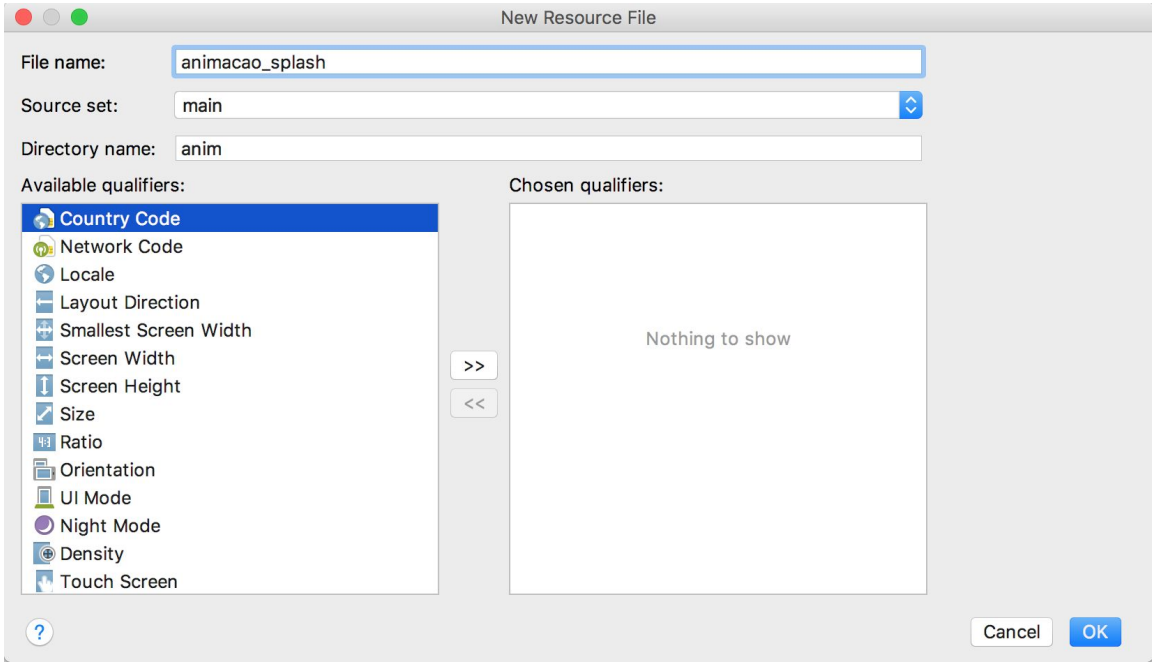




APP: Calculadora Flex - SplashScreen - Animação

Criar o arquivo da animação. Botão direito **new ⇒ Animation resource file**





APP: Calculadora Flex - SplashScreen - Animação

No arquivo **animacao_splash.xml** digite o seguinte código. Nesse exemplo, a animação criada será realizar um **fade-in** na nossa imagem.

```
<?xml version="1.0" encoding="utf-8"?>
<alpha
xmlns:android="http://schemas.android.com/apk/res/android"
        android:fromAlpha="0.0"
        android:toAlpha="1.0"
       android:duration="3000" />
```

APP: Calculadora Flex - SplashActivity.kt

```
class SplashActivity : AppCompatActivity() {

  private val TEMPO_AGUARDO_SPLASHSCREEN = 3500L

      override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_splash)
            carregar()
      }
      private fun carregar() {
            //Carrega a animacao
            val anim = AnimationUtils.loadAnimation(this, R.anim.animacao_splash)
            anim.reset()
            ivLogo.clearAnimation()
            //Roda a animacao
            ivLogo.startAnimation(anim)
            //Chama a próxima tela após 3,5 segundos definido na SPLASH_DISPLAY_LENGTH
            Handler().postDelayed({
                  val proximaTela = Intent(this@SplashActivity, FormActivity::class.java)
                  startActivity(proximaTela)
                  finish()
             }, TEMPO_AGUARDO_SPLASHSCREEN)
       }
}
```

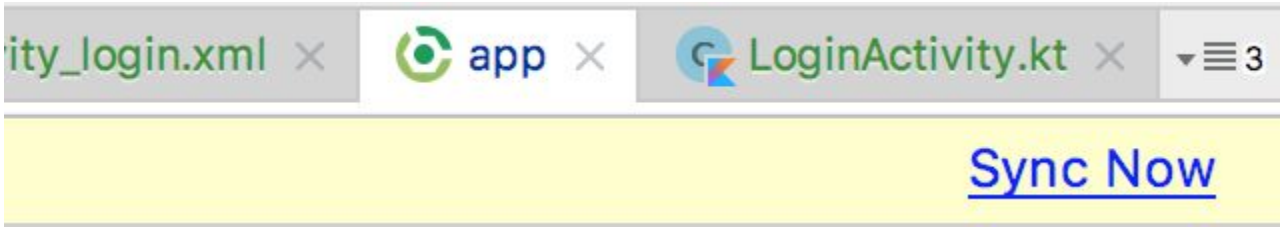
Criando nosso Login

APP: Calculadora Flex - Login

Adicionaremos a biblioteca do material para utilizar outros componentes disponíveis para o Android, por exemplo, o **TextInputLayout.** Abra o arquivo **build.gradle** do módulo **app,** e adicione a seguinte linha

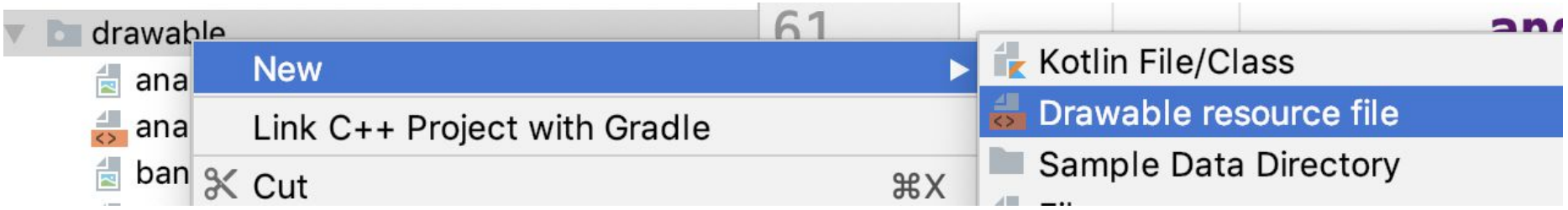
```
dependencies {
        ...
        implementation 'com.google.android.material:material:1.0.0'
        ...
}
```

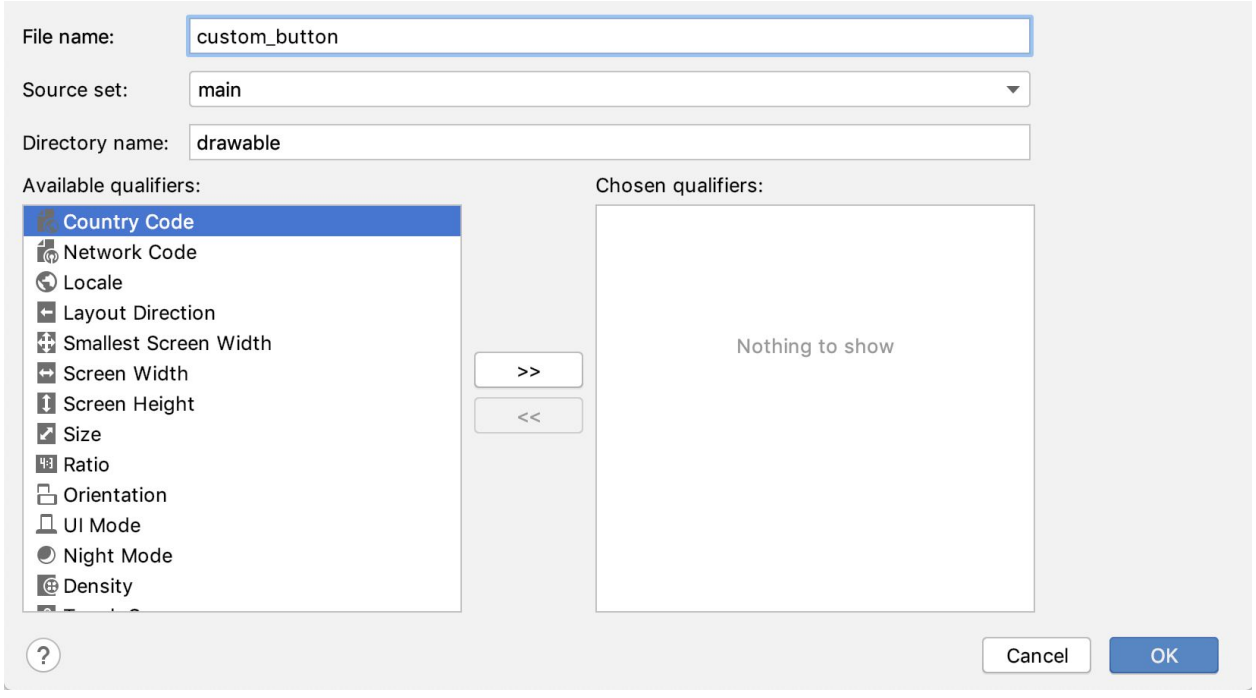
Após adicionar a biblioteca não esqueça de clicar no botão **Sync Now**



APP: Calculadora Flex - Login

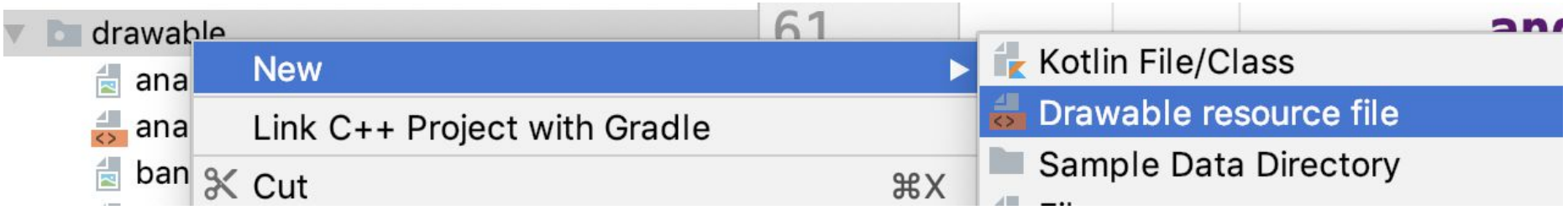
Clique com o botão direito sobre a pasta **drawable** e crie um arquivo chamado **custom_button.xml**

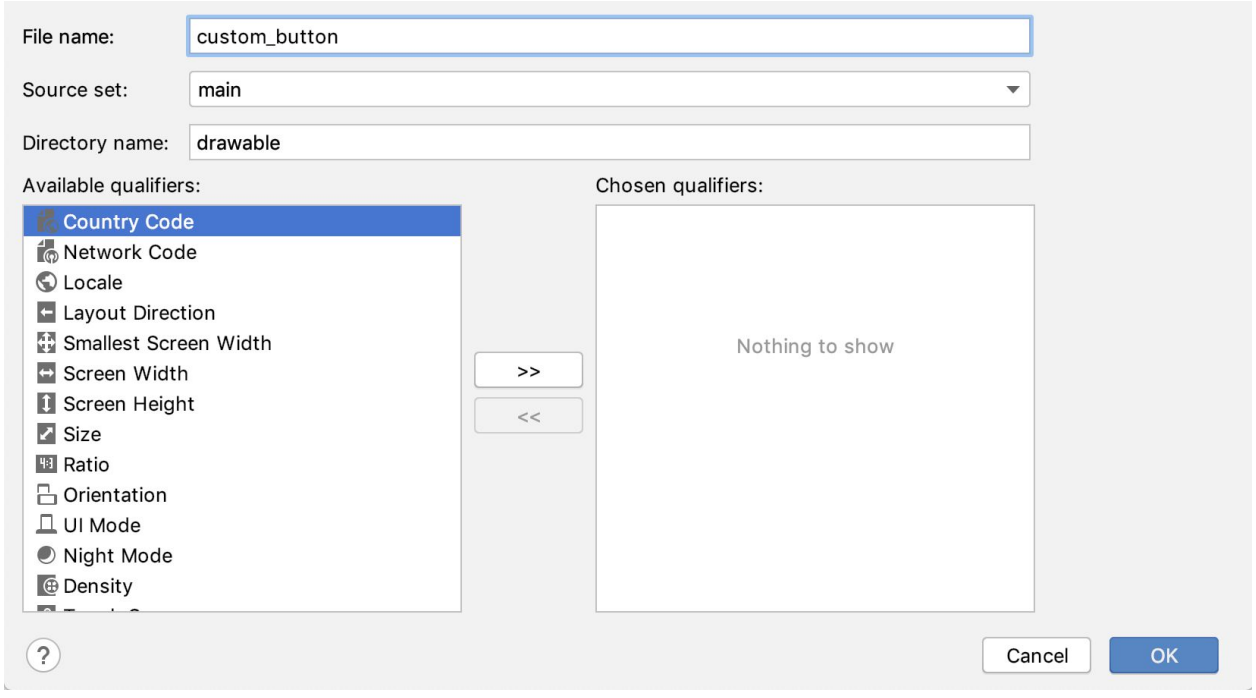




APP: Calculadora Flex - Login

Clique com o botão direito sobre a pasta **drawable** e crie um arquivo chamado **custom_button.xml**





APP: Calculadora Flex - custom_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:state_pressed="true" >
    <shape>
      <solid
          android:color="@color/colorPrimary" />
      <stroke
          android:width="1dp"
          android:color="@color/colorPrimaryDark" />
      <corners
          android:radius="3dp" />
      <padding
          android:left="10dp"
          android:top="10dp"
          android:right="10dp"
          android:bottom="10dp" />
    </shape>
  </item>
```

APP: Calculadora Flex - custom_button.xml

```
    <item>
        <shape>
            <gradient
                    android:startColor="@color/colorPrimary"
                    android:endColor="@color/colorPrimaryDark"
                    android:angle="270" />
            <stroke
                    android:width="1dp"
                    android:color="@color/colorPrimaryDark" />
            <corners
                    android:radius="4dp" />
            <padding
                    android:left="10dp"
                    android:top="10dp"
                    android:right="10dp"
                    android:bottom="10dp" />
        </shape>
    </item>
</selector>
```

APP: Calculadora Flex - Login

Abrir o arquivo **styles.xml**

```
<resources>
    // Nao é necessário apagar o código existente
     <style name="custom_button">
      <item name="android:layout_width">match_parent</item>
      <item name="android:layout_height">wrap_content</item>
      <item name="android:textColor">#FFF</item>
      <item name="android:background">@drawable/custom_button</item>
    </style>
</resources>
```

APP: Calculadora Flex - Login

No arquivo **activity_login.xml,** crie o seguinte layout.

APP: Calculadora Flex - Login

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true">

  <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="56dp"
        android:paddingLeft="24dp"
        android:paddingRight="24dp">

    <ImageView android:src="@drawable/gas_station_icon"
            android:layout_width="96dp"
            android:layout_height="96dp"
            android:layout_marginBottom="24dp"
            android:layout_gravity="center_horizontal"/>

    <com.google.android.material.textfield.TextInputLayout
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:layout_marginTop="8dp"
          android:layout_marginBottom="8dp">
      <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/inputLoginEmail"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"
            android:hint="@string/label_email"/>
    </com.google.android.material.textfield.TextInputLayout>
```

APP: Calculadora Flex - Login

```
    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp">
      <com.google.android.material.textfield.TextInputEditText
          android:id="@+id/inputLoginPassword"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:inputType="textPassword"
          android:hint="@string/label_password"/>
    </com.google.android.material.textfield.TextInputLayout>

<androidx.appcompat.widget.AppCompatButton
        android:id="@+id/btLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:layout_marginBottom="24dp"
        android:padding="12dp"
        style="@style/custom_button"
        android:text="@string/button_login"/>

    <TextView android:id="@+id/btSignup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        android:text="@string/label_new_account"
        android:gravity="center"
        android:textSize="16sp"/>

  </LinearLayout>
</ScrollView>
```

Melhorando a experiência do usuário

Imagine se todas as vezes que o usuário abrir a App, tiver que esperar um determinado tempo para a App mostrar o conteúdo que ele vai interagir, sendo que ele já sabe o que é o App e o que ela faz.

Para evitar o aborrecimento do usuário, podemos armazenar uma informação indicando que o usuário já iniciou o App uma vez.

Podemos optar por criar um banco de dados no SQLite, e então, criamos uma tabela para armazenar essas informações. Entretanto, precisamos apenas armazenar uma informação booleana (verdadeiro ou falso), ou seja, um banco de dados somente para isso não é necessário

Nesse caso, podemos utilizar o **SharedPreferences.**

SharedPreferences

No Android, existem diversas opções de armazenamento, conhecidas como **Storage Options**, dentre essas opções, temos a **SharedPreferences** que nos permite armazenar valores primitivos, como por exemplo:

- boolean
- float
- int
- long
- String

Tudo baseado em uma chave, por exemplo:

- **chave:** open_first, **valor**: true

SharedPreferences

Primeiro chamamos o método **getSharedPreference()** enviando 2 parâmetros:

getSharedPreferences("user_preferences", MODE_PRIVATE)

Onde:

1º parâmetro: nome da shared preferences que queremos buscar.
2º parâmetro: modo de acesso (Por default utilizamos o **MODE_PRIVATE** pois ele restringe o acesso apenas para a App que está chamando ou para todas as Apps que tiverem o mesmo user ID)

Esse método nos devolve uma instância da classe **SharedPreferences**:

A partir do objeto preferences podemos verificar se existe uma chave com o método contains(). Portanto, antes de chamar o carregar(), vamos verificar se existe a chave "**open_first**".

APP: Calculadora Flex - SplashScreen - 1/3

```
class SplashActivity : AppCompatActivity() {
    private val TEMPO_AGUARDO_SPLASHSCREEN = 3500L
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)
        val preferences = getSharedPreferences("user_preferences",
Context.MODE_PRIVATE)
        val isFirstOpen = preferences.getBoolean("open_first", true)
        if (isFirstOpen) {
            showLogin()
        } else {
            markAppAlreadyOpen(preferences)
            showSplash()
        }
    }
```

APP: Calculadora Flex - SplashScreen - 2/3

```
private fun markAppAlreadyOpen(preferences: SharedPreferences) {
    val editor = preferences.edit()
    editor.putBoolean("open_first", false)
    editor.apply()
}

private fun showLogin() {
    val nextScreen = Intent(this@SplashActivity, LoginActivity::class.java)
    startActivity(nextScreen)
    finish()
}
```

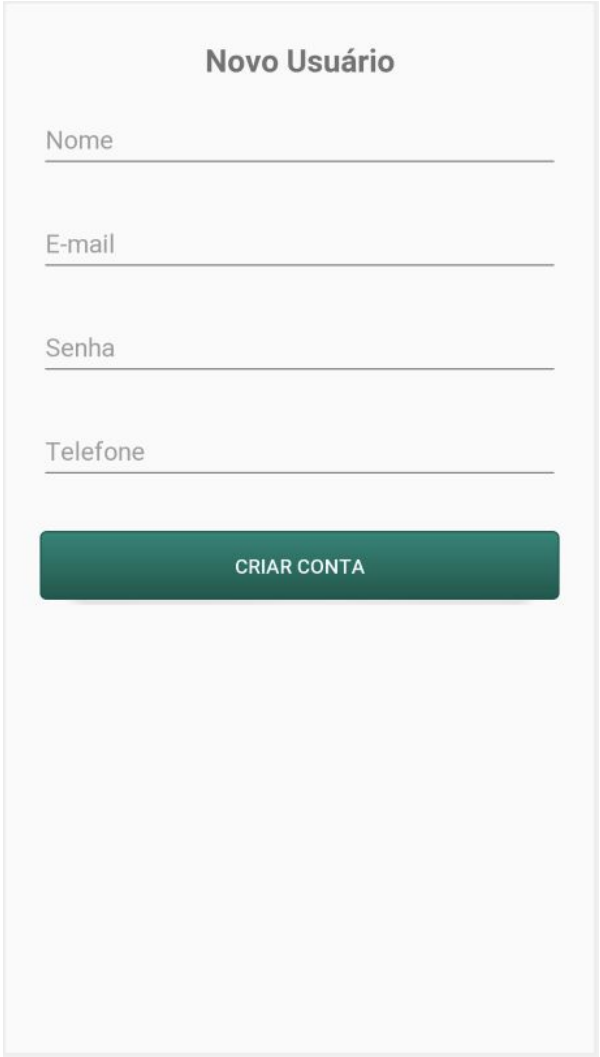
APP: Calculadora Flex - SplashScreen - 3/3

```
private fun showSplash() {
        val anim = AnimationUtils.loadAnimation(this, R.anim.animacao_splash)
        anim.reset()
        ivLogo.clearAnimation()
        ivLogo.startAnimation(anim)

        Handler().postDelayed({
            val nextScreen = Intent(this@SplashActivity, FormActivity::class.java)
            startActivity(nextScreen)
            finish()
        }, TEMPO_AGUARDO_SPLASHSCREEN)
    }
}
```

Criando nossa tela de Criação de Conta

APP: Calculadora Flex - Criação de Conta

Segue abaixo o desenho da tela a ser desenvolvida:



APP: Calculadora Flex - Criação de Conta

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true">
  <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="24dp"
        android:paddingLeft="24dp"
        android:paddingRight="24dp">
    <TextView android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/header_new_user"
            android:textSize="22sp"
            android:textStyle="bold"
            android:gravity="center"/>
    <com.google.android.material.textfield.TextInputLayout
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:layout_marginTop="8dp"
          android:layout_marginBottom="8dp">
```

APP: Calculadora Flex - Criação de Conta

```
    <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/inputName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"
            android:hint="@string/label_name"/>
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp">
    <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/inputEmail"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"
            android:hint="@string/label_email"/>
</com.google.android.material.textfield.TextInputLayout>
```

APP: Calculadora Flex - Criação de Conta

```
<com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp">
    <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/inputPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:hint="@string/label_password"/>
</com.google.android.material.textfield.TextInputLayout>
<com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp">
    <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/inputPhone"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"
            android:hint="@string/label_phone"/>
</com.google.android.material.textfield.TextInputLayout>
```

APP: Calculadora Flex - Criação de Conta

```
        <androidx.appcompat.widget.AppCompatButton
                style="@style/custom_button"
                android:id="@+id/btCreate"
                android:layout_marginTop="24dp"
                android:layout_marginBottom="24dp"
                android:padding="12dp"
                android:text="@string/button_new_account"/>

    </LinearLayout>
</ScrollView>
```

Conectando nosso app com o Firebase

O que é o Firebase?

- Um BaaS (ou MBaaS)

- Disponível on-line (você não precisa baixar)
  - Acessível via firebase.google.com

- Fornece

  - Autenticação
  - Real-time Database
  - Storage
  - Notifications
  - Hostings
  - e muito mais

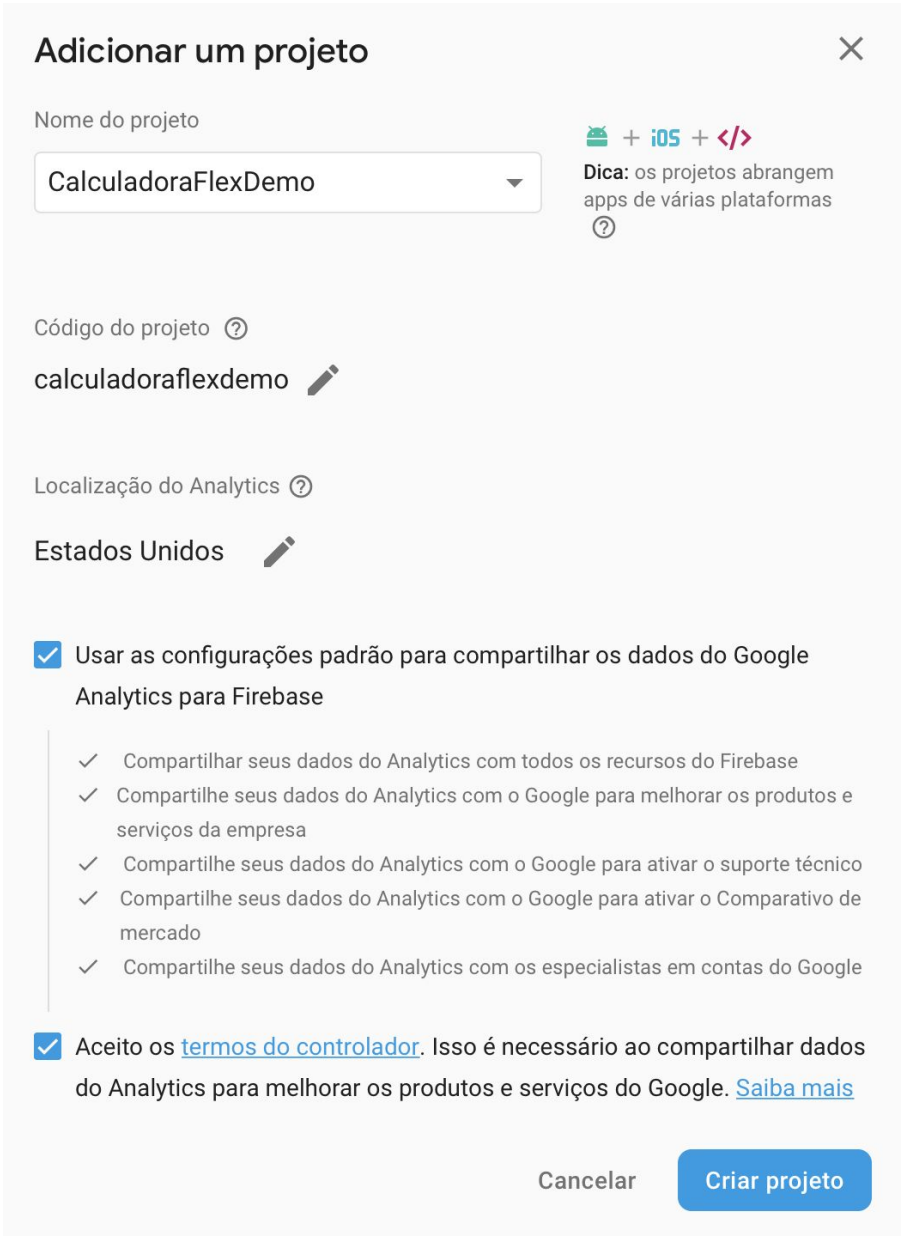
Criando uma conta

- Acessar o site
  - **console.firebase.google.com**

- Necessário ter uma conta Google

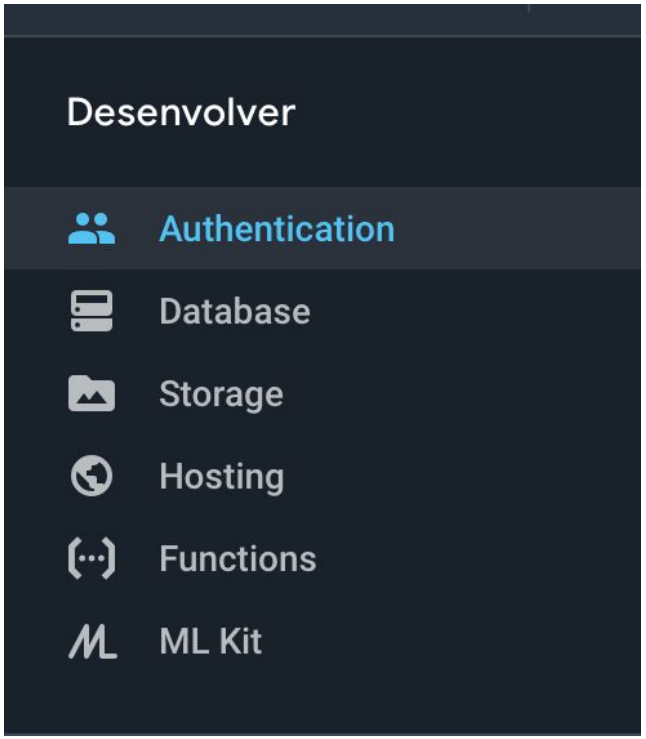
Criando uma conta

Clique em **Adicionar projeto** e realize a configuração do projeto. Depois disso clique em em **Criar projeto**



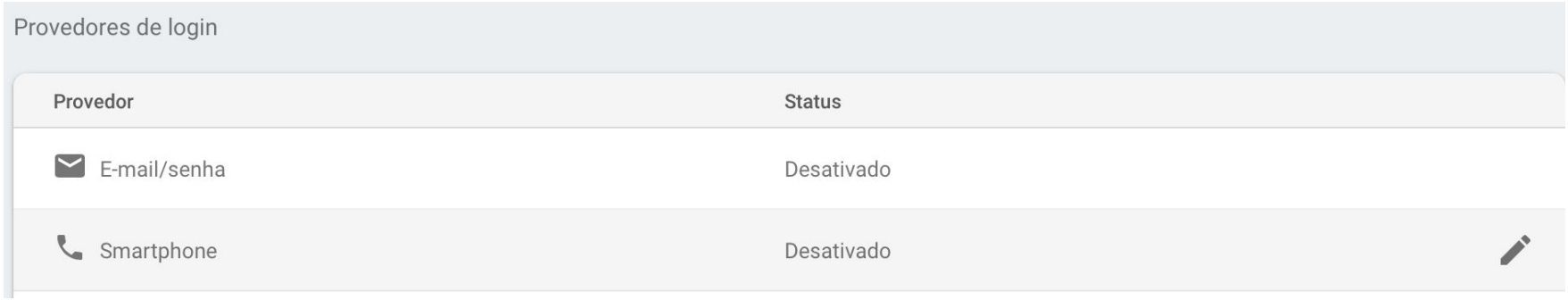
Firebase: Habilitando a autenticação por e-mail

Clique em **Authentication**, em seguida, **Configurar método de login**

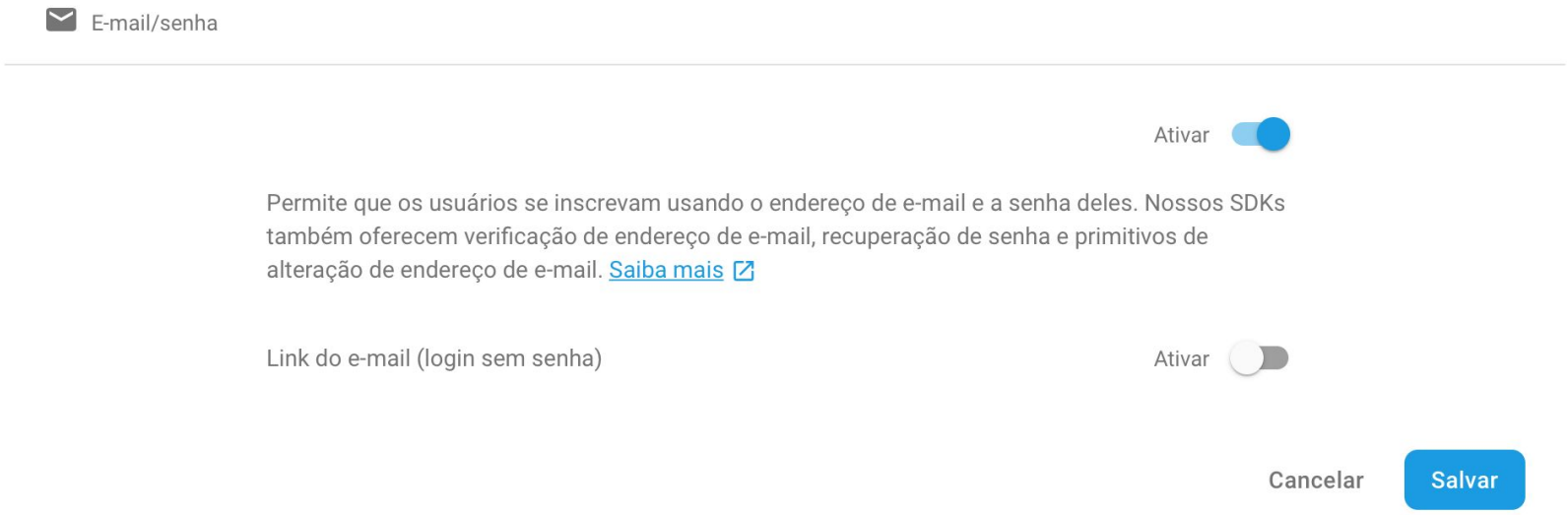


Firebase: Habilitando a autenticação por e-mail

Clique sobre E-mail/Senha

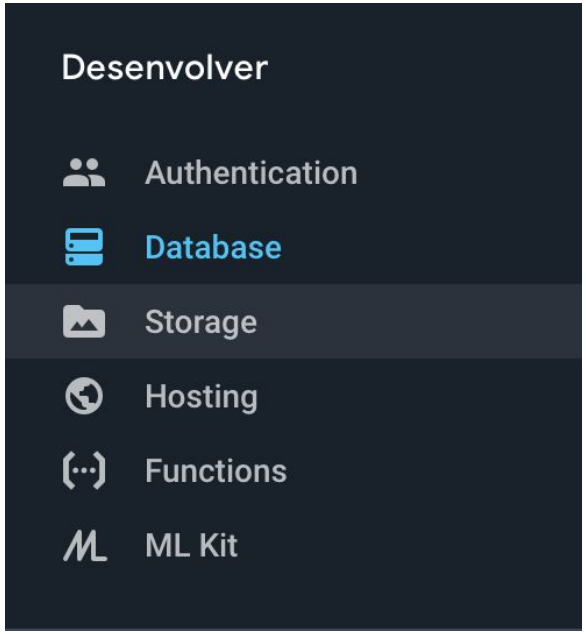


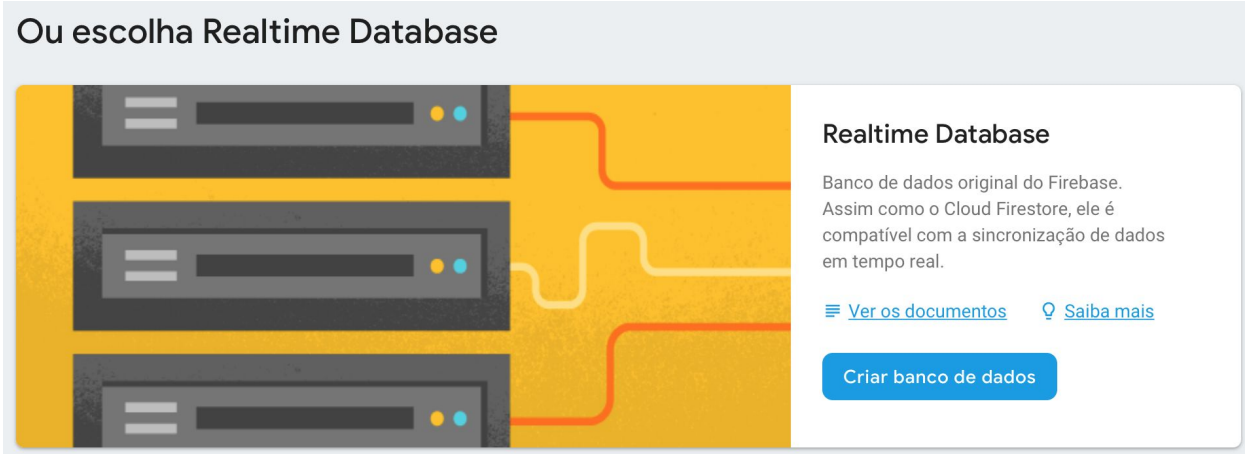
Ative esse modo de Autenticação e clique em **Salvar**



Firebase: Habilitando o RealTime Database

- No console do **Firebase** clique sobre **Database,** em seguida, clique em **Criar banco de dados**





Firebase: Habilitando o RealTime Database

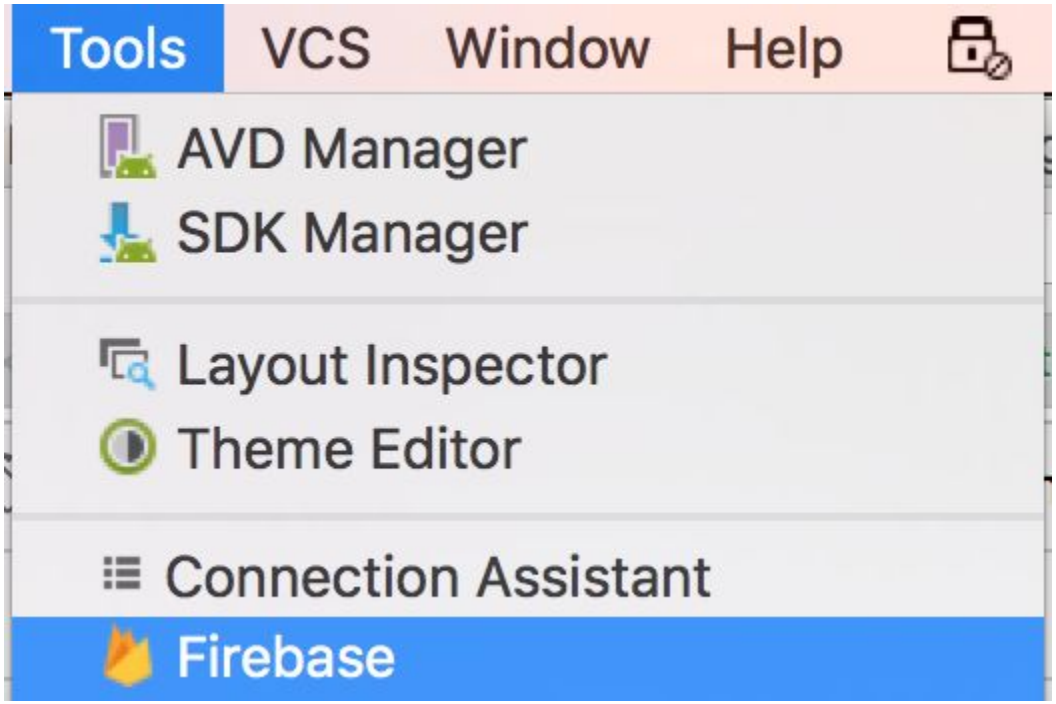
Em seguida, marque a opção **modo de teste** e clique em **Ativar**

Firebase: Habilitando o RealTime Database

Nesse primeiro momento iremos utilizar duas funcionalidades do **Firebase**, a primeira será o **Real Time Database** para armazenarmos as informações do nosso usuário e o **Authentication** para segurança do nosso app, através dele somente usuário autenticado terá acesso ao app.

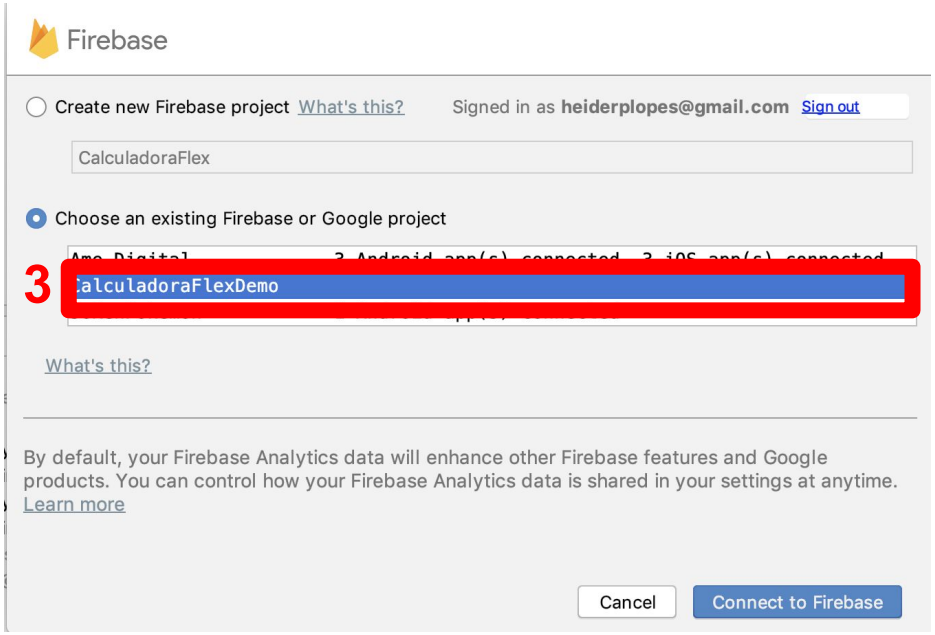
Integrando nosso app com o Firebase

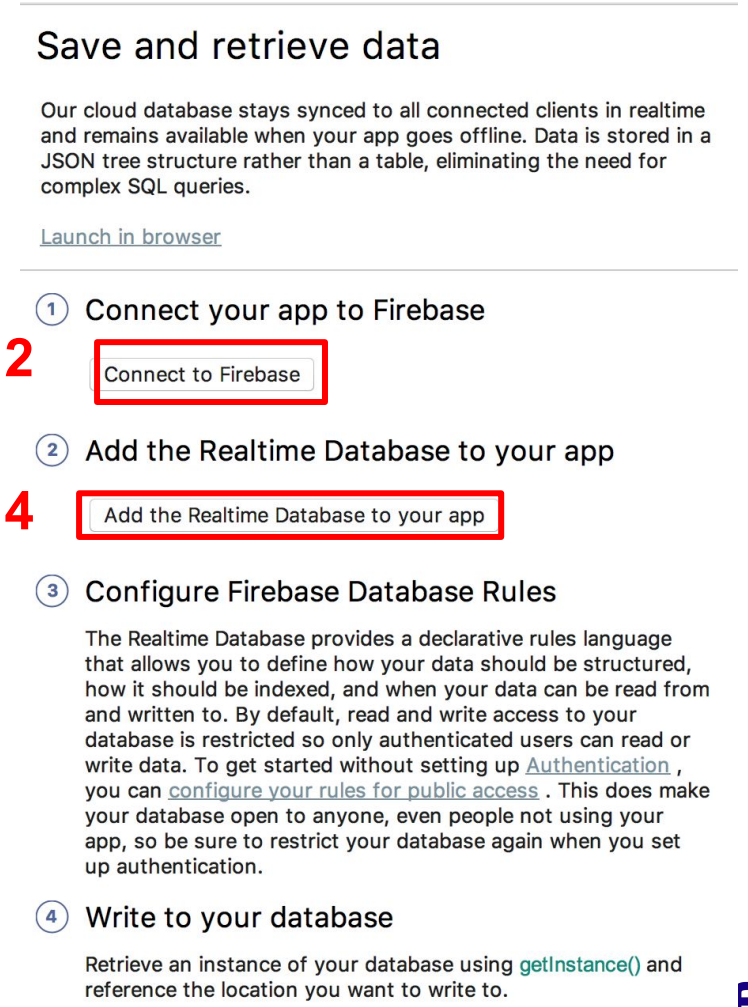
No **Android Studio** clicar no menu **Tools ⇒ Firebase**



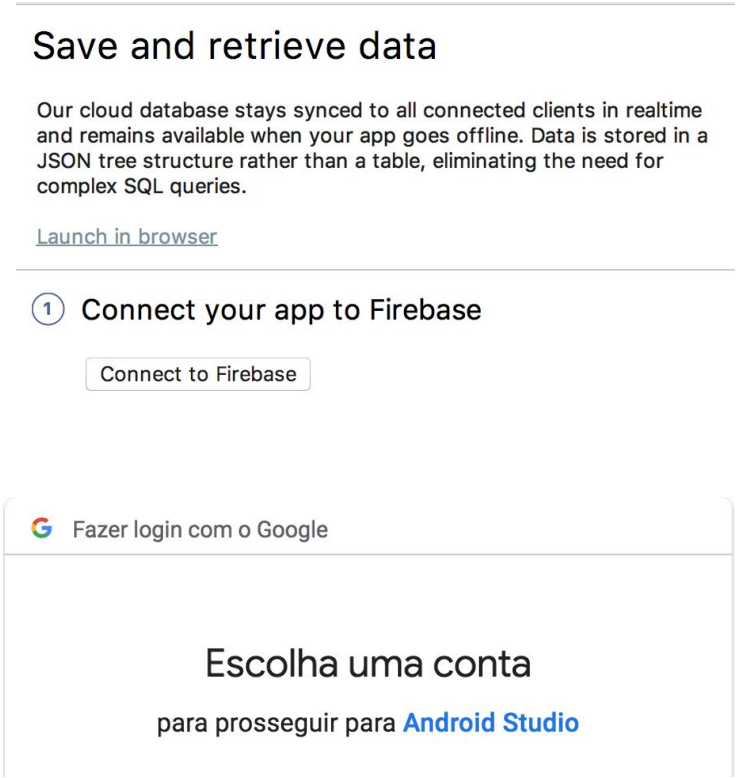
Integrando nosso app com o Firebase: RealTime Database

No **Android Studio** clicar no menu **Tools ⇒ Firebase**





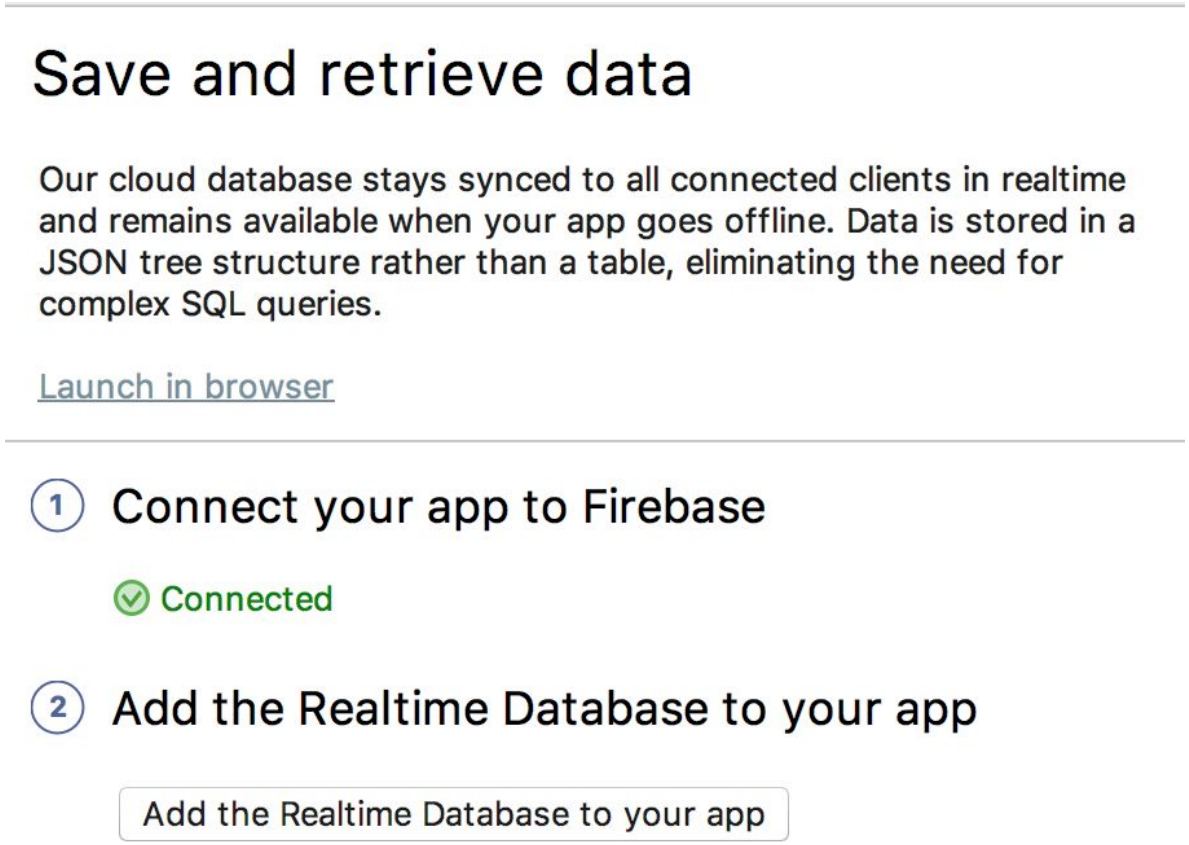
Integrando nosso app com o Firebase: RealTime Database



Ao clicar no botão **Connect to Firebase,** você será redirecionado para o browser para realizar a autenticação na sua conta **Google** vinculada ao **Firebase.**

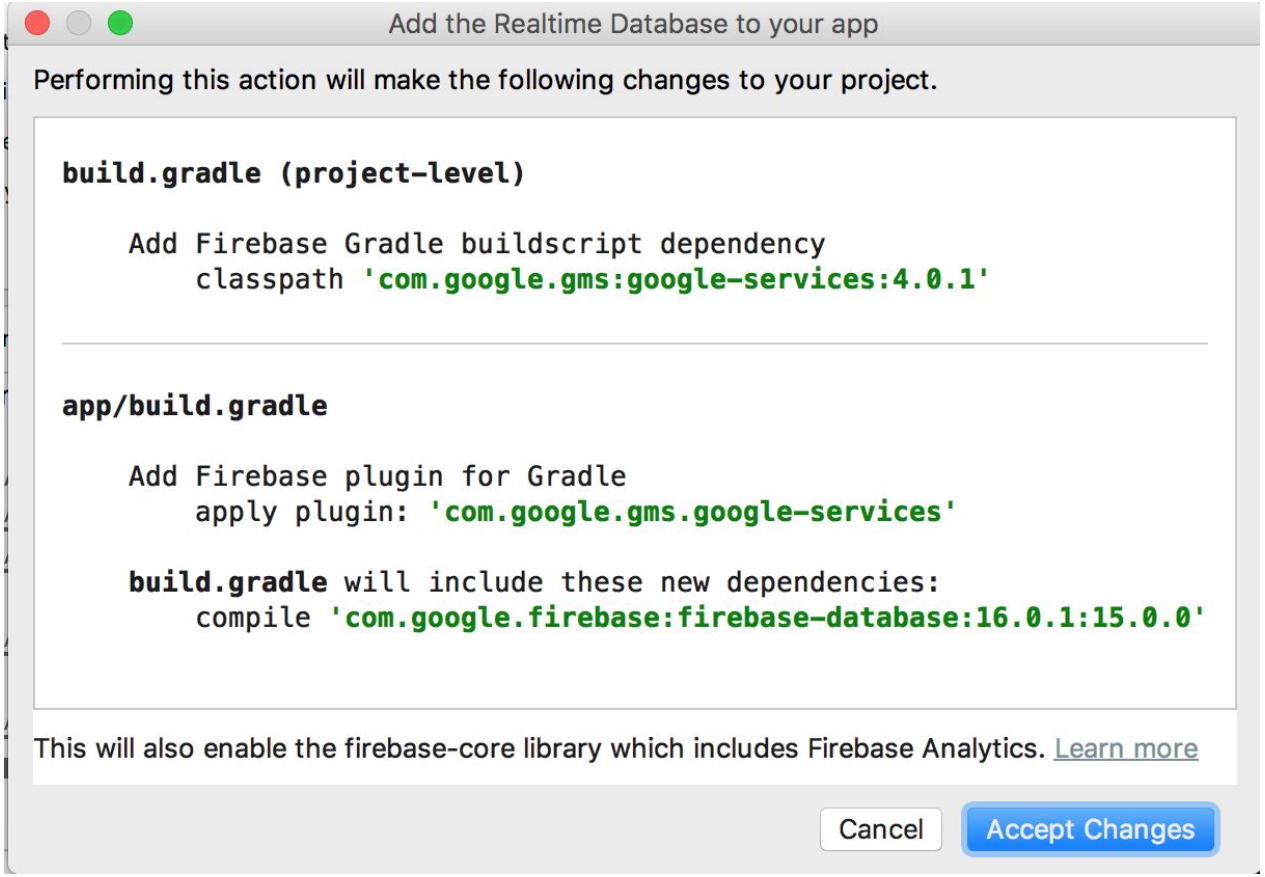
Você precisará dar as permissões solicitadas para que seja realizada a ligação do **Firebase** com seu **App**

Integrando nosso app com o Firebase: RealTime Database



Você verá a opção **Connected** quando a conexão estiver realizada. Após isso, precisamos adicionar as dependências necessárias para conversar como o **Firebase** utilizando o sua biblioteca. Clique em **Add the Realtime Database to your app**

Integrando nosso app com o Firebase: RealTime Database



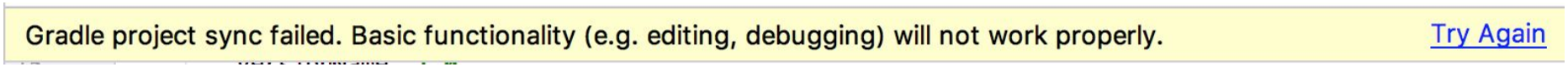
Clique em **Accept Changes**

Integrando nosso app com o Firebase: RealTime Database

Abra o arquivo **build.gradle** do módulo **app** e altere a implementação do Realtime Dabase conforme a linha abaixo:

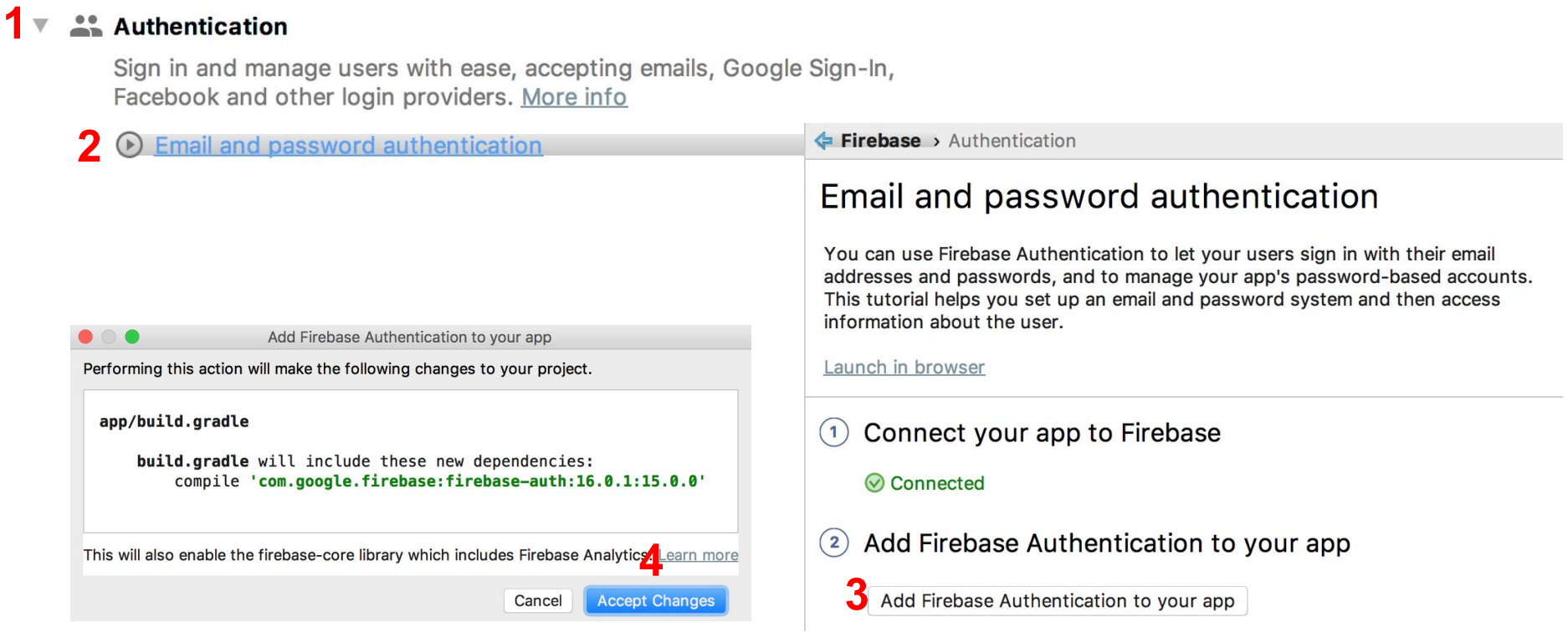
```
dependencies {
    implementation 'com.google.firebase:firebase-database:18.0.0'
}
```

Em seguida, clique em **Try Again**



Integrando nosso app com o Firebase: Authentication

Clique agora no **Authentication,** em seguida, **Email and password authentication.** Como já havíamos realizado a ligação com o **Firebase** já veremos a opção **Connected.** Após isso, devemos adicionar a dependência do **Authentication** através do Add Firebase to your app

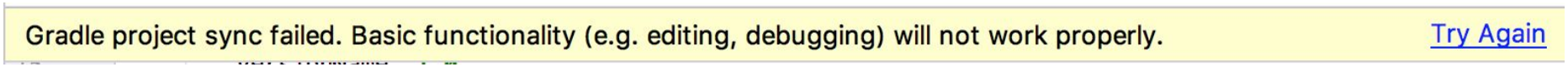


Integrando nosso app com o Firebase: Authentication

Abra o arquivo **build.gradle** do módulo **app** e altere a implementação do Authentication conforme a linha abaixo:

```
dependencies {
    implementation 'com.google.firebase:firebase-auth:18.1.0'
}
```

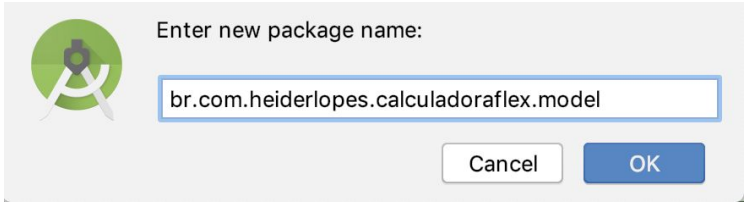
Em seguida, clique em **Try Again**

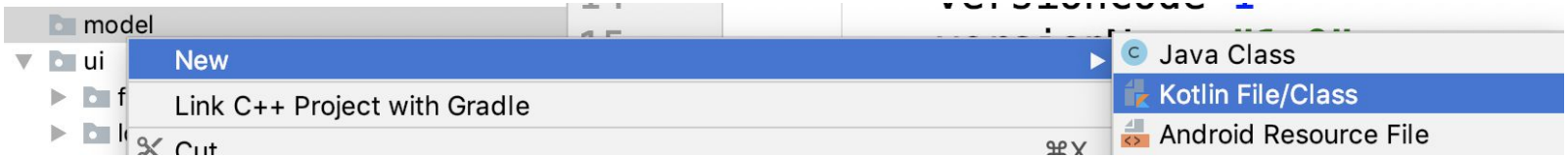


Criando a conta do usuário

Criando a nova conta

Para criar uma nova conta precisamos mapear o modelo referente ao nosso usuário. Para isso, crie um novo package chamado **model.** Dentro dele, crie uma classe Kotlin chamada **User.kt.** Dentro desse arquivo adicione o seguinte código:






```
data class User(
    val nome: String = "",
    val email: String = "",
    val fone: String = ""
)
```

APP: Calculadora Flex - Criação de Conta

```
class SignUpActivity : AppCompatActivity() {
    private lateinit var mAuth: FirebaseAuth
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sign_up)

        mAuth = FirebaseAuth.getInstance()

        btCreate.setOnClickListener {
            mAuth.createUserWithEmailAndPassword(
                inputEmail.text.toString(),
                inputPassword.text.toString()
            ).addOnCompleteListener {
                if (it.isSuccessful) {
                    saveInRealTimeDatabase()
                } else {
                    Toast.makeText(this@SignUpActivity, it.exception?.message,
Toast.LENGTH_SHORT).show()
                }
            }
        }
    }
```

APP: Calculadora Flex - Criação de Conta

```
private fun saveInRealTimeDatabase() {
        val user = User(inputName.text.toString(), inputEmail.text.toString(),
inputPhone.text.toString())
        FirebaseDatabase.getInstance().getReference("Users")
            .child(FirebaseAuth.getInstance().currentUser!!.uid)
            .setValue(user)
            .addOnCompleteListener {
                if (it.isSuccessful) {
                    Toast.makeText(this, "Usuário criado com sucesso",
Toast.LENGTH_SHORT).show()
                    val returnIntent = Intent()
                    returnIntent.putExtra("email", inputEmail.text.toString())
                    setResult(RESULT_OK, returnIntent)
                    finish()
                    finish()
                } else {
                    Toast.makeText(this, "Erro ao criar o usuário", Toast.LENGTH_SHORT).show()
                }
            }
    }
}
```

Programando nossa tela de Login

APP: Calculadora Flex - LoginActivity.kt

```
class LoginActivity : AppCompatActivity() {
    private lateinit var mAuth: FirebaseAuth
    private val newUserRequestCode = 1
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)
        mAuth = FirebaseAuth.getInstance()
        if (mAuth.currentUser != null) {
            goToHome()
        }
        btLogin.setOnClickListener {
            mAuth.signInWithEmailAndPassword(
                inputLoginEmail.text.toString(),
                inputLoginPassword.text.toString()
            ).addOnCompleteListener {
                if (it.isSuccessful) {
                    goToHome()
                } else {
                    Toast.makeText(this@LoginActivity, it.exception?.message,
Toast.LENGTH_SHORT).show()
                }
            }
        }
```

APP: Calculadora Flex - LoginActivity.kt

```
btSignup.setOnClickListener {
        startActivityForResult(Intent(this, SignUpActivity::class.java),
newUserRequestCode)
    }
  }

  private fun goToHome() {
    val intent = Intent(this, FormActivity::class.java)
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
    startActivity(intent)
    finish()
  }


  override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == newUserRequestCode && resultCode == Activity.RESULT_OK) {
      inputLoginEmail.setText(data?.getStringExtra("email"))
    }
  }
}
```

Criando nossa tela da calculadora

APP: Calculadora Flex - FormActivity

Posto de Gasolina

Preço Gasolina

Ex.: 3.50

Preço Etanol

Ex.: 2.50

Consumo médio

Km/L Gasolina

Ex.: 9

Km/L Álcool

Ex.: 5

CALCULAR

APP: Calculadora Flex - FormActivity

Crie um arquivo chamado **gas_station_icon.xml** na pasta **drawable.**
Botão direito sobre a pasta **drawable ⇒ new ⇒ Drawable resource file**

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

  <item
        android:drawable="@drawable/fuel"
        android:width="@dimen/icon_header_size"
        android:height="@dimen/icon_header_size"
        android:right="@dimen/icon_header_margin"/>

</layer-list>
```

APP: Calculadora Flex - FormActivity

Crie um arquivo chamado **car_icon.xml** na pasta **drawable.** Botão direito sobre a pasta **drawable ⇒ new ⇒ Drawable resource file**

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

  <item
        android:drawable="@drawable/car"
        android:width="@dimen/icon_header_size"
        android:height="@dimen/icon_header_size"
        android:right="@dimen/icon_header_margin"/>

</layer-list>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **styles.xml** e adicione os seguintes styles.

```
<style name="container_master">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">match_parent</item>
    <item name="android:orientation">vertical</item>
</style>

<style name="header_title" parent="margin_default_row">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textSize">16sp</item>
    <item name="android:textStyle">bold</item>
</style>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **styles.xml** e adicione os seguintes styles.

```
<style name="header_subtitle">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_weight">0.5</item>
    <item name="android:textStyle">bold</item>
</style>

<style name="input_number_decimal">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_weight">0.5</item>
    <item name="android:inputType">numberDecimal</item>
</style>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **styles.xml** e adicione os seguintes styles.

```
<style name="margin_default_row">
    <item name="android:layout_marginLeft">16dp</item>
    <item name="android:layout_marginRight">16dp</item>
</style>

<style name="divider" parent="margin_default_row">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">1dp</item>
    <item name="android:background">#CCC</item>
    <item name="android:layout_marginTop">12dp</item>
    <item name="android:layout_marginBottom">12dp</item>
</style>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **styles.xml** e adicione os seguintes styles.

```
<style name="container_row" parent="margin_default_row">
    <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="banner">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">180dp</item>
    <item name="android:scaleType">centerCrop</item>
    <item name="android:src">@drawable/banner</item>
</style>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **activity_form.xml** e adicione:

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

         style="@style/container_master">

      <ImageView
            style="@style/banner"/>

      <View style="@style/divider"/>

      <TextView
            style="@style/header_title"
            android:drawableStart="@drawable/gas_station_icon"
            android:text="@string/header_gas_station"/>

      <View style="@style/divider"/>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **activity_form.xml** e adicione:

```
<LinearLayout style="@style/container_row">

  <TextView
        android:text="@string/header_gasoline_price"
        style="@style/header_subtitle"/>

  <TextView

        android:text="@string/header_ethanol_price"
        style="@style/header_subtitle"/>
</LinearLayout>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **activity_form.xml** e adicione:

```
<LinearLayout style="@style/container_row">

  <EditText
        android:hint="@string/hint_gasoline"
        android:nextFocusDown="@+id/etEthanolPrice"
        style="@style/input_number_decimal"
        android:id="@+id/etGasPrice"/>

  <EditText
        style="@style/input_number_decimal"
        android:nextFocusDown="@+id/etGasAverage"
        android:hint="@string/hint_ethanol"
        android:id="@+id/etEthanolPrice"/>
</LinearLayout>

<View style="@style/divider"/>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **activity_form.xml** e adicione:

```
<TextView
      android:drawableStart="@drawable/car_icon"
      android:text="@string/header_average"
      style="@style/header_title"/>

<View style="@style/divider"/>

<LinearLayout style="@style/container_row">

  <TextView
        android:text="@string/header_average_gasoline"
        style="@style/header_subtitle"/>

  <TextView
        android:text="@string/header_average_ethanol"
        style="@style/header_subtitle"/>
</LinearLayout>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **activity_form.xml** e adicione:

```
<LinearLayout style="@style/container_row">

  <EditText
        style="@style/input_number_decimal"
        android:nextFocusDown="@+id/etEthanolAverage"
        android:hint="@string/hint_average_gasoline"
        android:id="@+id/etGasAverage"/>

  <EditText
        android:autofillHints=""
        style="@style/input_number_decimal"
        android:hint="@string/hint_average_ethanol"
        android:id="@+id/etEthanolAverage"/>

</LinearLayout>
```

APP: Calculadora Flex - FormActivity

Abra o arquivo **activity_form.xml** e adicione:

```
<Button
            android:layout_margin="16dp"
            style="@style/custom_button"
            android:text="@string/calculate"
            android:id="@+id/btCalculate"/>

        <View style="@style/divider"/>
    </LinearLayout>
</ScrollView>
```

Criando nossa tela de resultado

APP: Calculadora Flex - ResultActivity

APP: Calculadora Flex - ResultActivity

Crie um arquivo chamado **coins_icon.xml** na pasta **drawable.** Botão direito sobre a pasta **drawable ⇒ new ⇒ Drawable resource file**

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

  <item
        android:drawable="@drawable/coins"
        android:width="@dimen/icon_header_size"
        android:height="@dimen/icon_header_size"
        android:right="@dimen/icon_header_margin"/>

</layer-list>
```

APP: Calculadora Flex - ResultActivity

Crie um arquivo chamado **analysis_icon.xml** na pasta **drawable.** Botão direito sobre a pasta **drawable ⇒ new ⇒ Drawable resource file**

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

  <item
        android:drawable="@drawable/analysis"
        android:width="@dimen/icon_header_size"
        android:height="@dimen/icon_header_size"
        android:right="@dimen/icon_header_margin"/>

</layer-list>
```

APP: Calculadora Flex - ResultActivity

Abra o arquivo **styles.xml** e adicione os seguintes estilos:

```
<style name="label_spent_footer">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_weight">0.5</item>
    <item name="android:gravity">center</item>
</style>

<style name="value_spent_footer">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textSize">48sp</item>
    <item name="android:gravity">center</item>
    <item name="android:layout_weight">0.5</item>
</style>
```

APP: Calculadora Flex - ResultActivity

Abra o arquivo **activty_result.xml** e adicione:

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

  <LinearLayout
      style="@style/container_master">


    <ImageView
        style="@style/banner"/>

    <View style="@style/divider"/>

    <TextView
        style="@style/header_title"
        android:drawableStart="@drawable/analysis_icon"
        android:text="@string/header_suggestion"/>

    <View style="@style/divider"/>
```

APP: Calculadora Flex - ResultActivity

Abra o arquivo **activty_result.xml** e adicione:

```
<TextView
    style="@style/header_title"
    tools:text="Etanol"
    android:textSize="48sp"
    android:gravity="center"
    android:id="@+id/tvSuggestion"/>


<TextView
    android:id="@+id/tvFuelRatio"
    style="@style/header_subtitle"
    tools:text="@string/label_fuel_ratio"
    android:textSize="16sp"
    android:textStyle="normal"
    android:gravity="center"/>

<View style="@style/divider"/>

<TextView
    android:drawableStart="@drawable/coins_icon"
    android:text="@string/header_spent"
    style="@style/header_title"/>
```

APP: Calculadora Flex - ResultActivity

Abra o arquivo **activty_result.xml** e adicione:

```
<View style="@style/divider"/>

<LinearLayout style="@style/container_row">

  <TextView
        style="@style/value_spent_footer"
        tools:text="0.30"
        android:id="@+id/tvGasAverageResult"/>

  <TextView
        style="@style/value_spent_footer"
        tools:text="0.30"
        android:id="@+id/tvEthanolAverageResult"/>
</LinearLayout>
```

APP: Calculadora Flex - ResultActivity

Abra o arquivo **activty_result.xml** e adicione:

```
<LinearLayout style="@style/container_row">

        <TextView
            style="@style/label_spent_footer"
            android:text="@string/gasoline"
        />

        <TextView
            style="@style/label_spent_footer"
            android:text="@string/ethanol"/>
    </LinearLayout>

  </LinearLayout>
</ScrollView>
```

Hora de calcular

APP: Calculadora Flex - Formatando o input

Crie uma classe kotlin chamada **DecimalTextWatcher.kt** dentro do package **watchers,** e adicione o código dos próximos slides:

```
class DecimalTextWatcher(editText: EditText, val totalDecimalNumber: Int = 2) :
TextWatcher {
  private val editTextWeakReference: WeakReference<EditText> =
WeakReference(editText)
  init {
    formatNumber(editTextWeakReference.get()!!.text)
  }
  override fun beforeTextChanged(s: CharSequence, start: Int, count: Int, after: Int) {}
  override fun onTextChanged(s: CharSequence, start: Int, before: Int, count: Int) {}
  override fun afterTextChanged(editable: Editable) {
    formatNumber(editable)
  }
```

APP: Calculadora Flex - Formatando o input

```
private fun getTotalDecimalNumber(): String {
    val decimalNumber = StringBuilder()
    for (i in 1..totalDecimalNumber) {
        decimalNumber.append("0")
    }
    return decimalNumber.toString()
}
```

APP: Calculadora Flex - Formatando o input

```
private fun formatNumber(editable: Editable) {
        val editText = editTextWeakReference.get() ?: return
        val cleanString = editable.toString().trim().replace(" ", "")
        editText.removeTextChangedListener(this)
        val number = Math.pow(10.toDouble(), totalDecimalNumber.toDouble())
        val parsed = when (cleanString) {
            null -> BigDecimal(0)
            "" -> BigDecimal(0)
            "null" -> BigDecimal(0)
            else -> BigDecimal(cleanString.replace("\\D+".toRegex(), ""))
                .setScale(totalDecimalNumber, BigDecimal.ROUND_FLOOR)
                .divide(
                    BigDecimal(number.toInt()),
                    BigDecimal.ROUND_FLOOR
                )
        }
        val dfnd = DecimalFormat("#,##0.${getTotalDecimalNumber()}")
        val formatted = dfnd.format(parsed)
        editText.setText(formatted.replace(',', '.'))
        editText.setSelection(formatted.length)
        editText.addTextChangedListener(this)
    }
}
```

APP: Calculadora Flex - FormActivity - Enviando dados

```
class FormActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_form)

        etGasPrice.addTextChangedListener(DecimalTextWatcher(etGasPrice))
        etEthanolPrice.addTextChangedListener(DecimalTextWatcher(etEthanolPrice))
        etGasAverage.addTextChangedListener(DecimalTextWatcher(etGasAverage, 1))
        etEthanolAverage.addTextChangedListener(DecimalTextWatcher(etEthanolAverage, 1))

        btCalculate.setOnClickListener {
            val proximatela = Intent(this@FormActivity, ResultActivity::class.java)
            proximatela.putExtra("GAS_PRICE", etGasPrice.text.toString().toDouble())
            proximatela.putExtra("ETHANOL_PRICE", etEthanolPrice.text.toString().toDouble())
            proximatela.putExtra("GAS_AVERAGE", etGasAverage.text.toString().toDouble())
            proximatela.putExtra("ETHANOL_AVERAGE", etEthanolAverage.text.toString().toDouble())
            startActivity(proximatela)
        }

    }
}
```

APP: Calculadora Flex - ResultActivity - Recebendo dados

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_result)

        if (intent.extras == null) {
                Toast.makeText(this, "Não foi possível realizar a operação",
        Toast.LENGTH_SHORT).show()
        } else {
                calculate()
        }
}
```

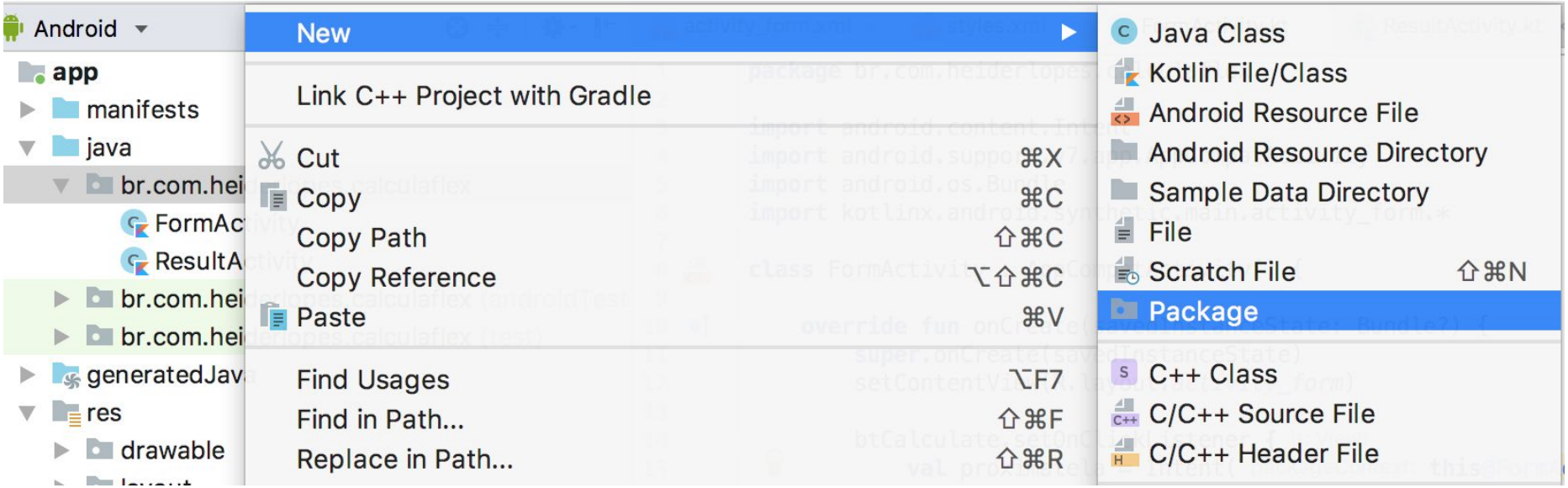
APP: Calculadora Flex - ResultActivity - Recebendo dados

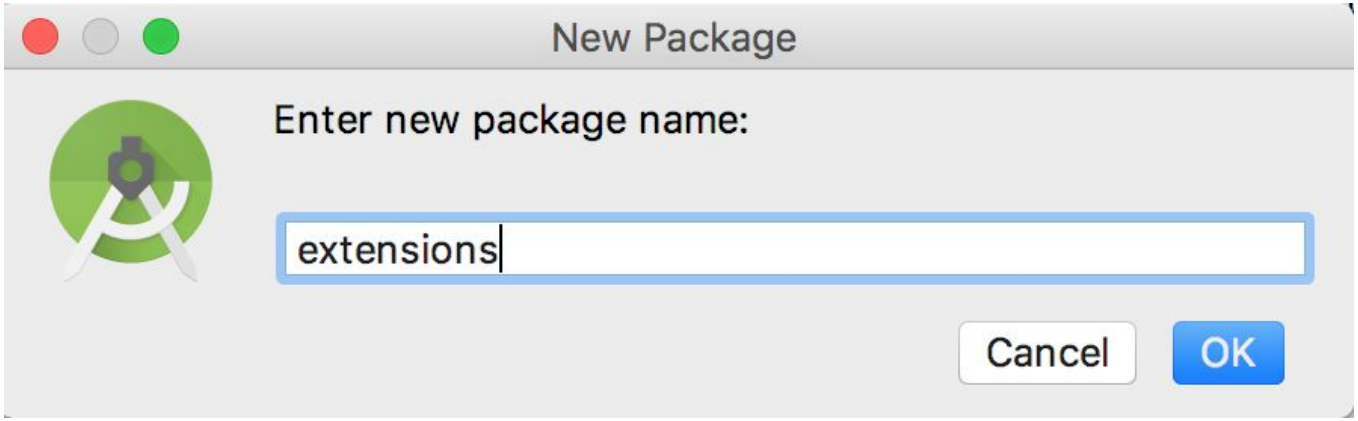
Abra o arquivo **ResultActivity.kt** e adicione a linha em negrito

```
private fun calculate() {
    val gasPrice = intent.extras.getDouble("GAS_PRICE", 0.0)
    val ethanolPrice = intent.extras.getDouble("ETHANOL_PRICE", 0.0)
    val gasAverage = intent.extras.getDouble("GAS_AVERAGE", 0.0)
    val ethanolAverage = intent.extras.getDouble("ETHANOL_AVERAGE", 0.0)
    val performanceOfMyCar = ethanolAverage / gasAverage
    val priceOfFuelIndice = ethanolPrice / gasPrice

    if (priceOfFuelIndice <= performanceOfMyCar) {
        tvSuggestion.text = getString(R.string.ethanol)
    } else {
        tvSuggestion.text = getString(R.string.gasoline)
    }
    tvEthanolAverageResult.text = (ethanolPrice / ethanolAverage).toString()
    tvGasAverageResult.text = (ethanolPrice / ethanolAverage).toString()
}
```

APP: Calculadora Flex - Extensions

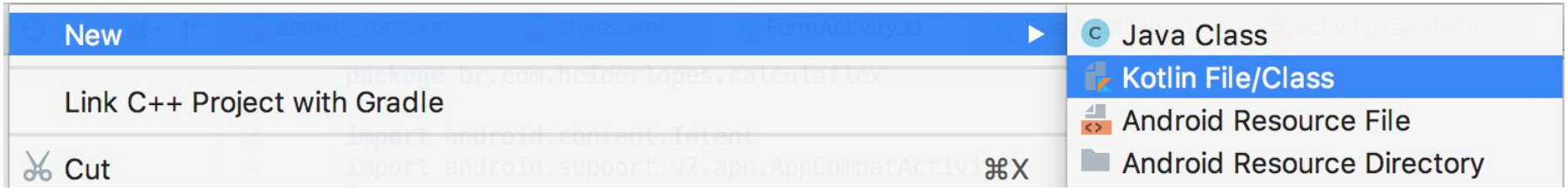
Crie um pacote chamado de **extensions**

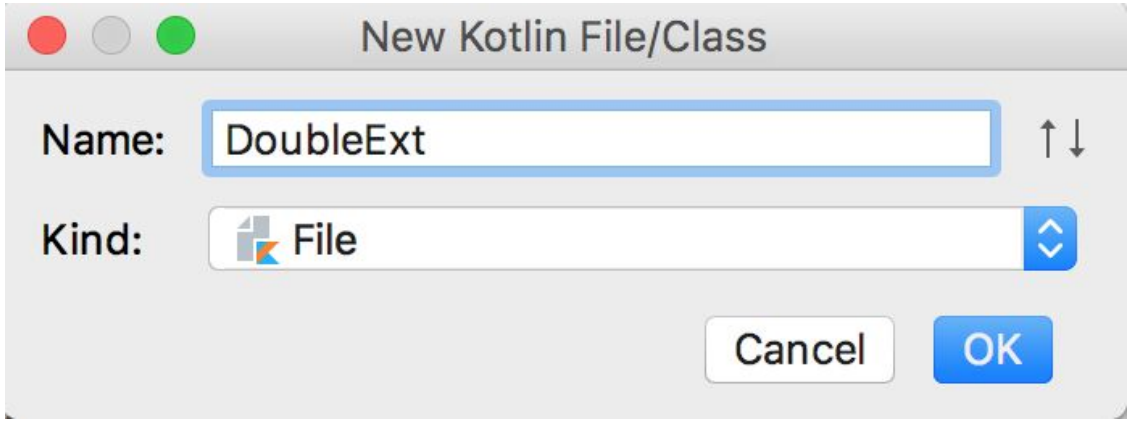




APP: Calculadora Flex - Extensions

Dentro desse pacote crie uma classe chamada **DoubleExt.kt**





APP: Calculadora Flex - Extensions

Abra o arquivo **DoubleExt.kt** e adicione a linha abaixo:

```
fun Double.format(digits: Int) =
java.lang.String.format("%.${digits}f", this)
```

APP: Calculadora Flex - Extensions

Abra o arquivo **ResultActivity.kt** e altere conforme abaixo:

```
private fun calculate() {
        ....
        tvEthanolAverageResult.text = (ethanolPrice / ethanolAverage).format(2)
        tvGasAverageResult.text = (ethanolPrice / ethanolAverage).format(2)
        tvFuelRatio.text =
        getString(R.string.label_fuel_ratio,performanceOfMyCar.format(2))
}
```

APP: Calculadora Flex - Adicionando botão de voltar na ResultActivity

No arquivo **ResultActivity.kt** e adicione as linhas abaixo:

```
override fun onCreate(savedInstanceState: Bundle?) {
    ...
    supportActionBar?.setDisplayHomeAsUpEnabled(true);
    supportActionBar?.setDisplayShowHomeEnabled(true);
    ...
}

override fun onSupportNavigateUp(): Boolean {
    onBackPressed()
    return true
}
```

Gravando e Lendo as calculados pelo usuário

APP: Calculadora Flex

Dentro package **model**, crie uma classe Kotlin chamada **DataCar.kt.**
Dentro desse arquivo adicione o seguinte código:

```
data class CarData(
        val gasPrice: Double = 0.0,
        val ethanolPrice: Double = 0.0,
        val gasAverage: Double = 0.0,
        val ethanolAverage: Double = 0.0
)
```

APP: Calculadora Flex

```
class FormActivity : AppCompatActivity() {
    private lateinit var userId: String
    private lateinit var mAuth: FirebaseAuth
    private val firebaseReferenceNode = "CarData"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity form)
        mAuth = FirebaseAuth.getInstance()
        userId = FirebaseAuth.getInstance().currentUser?.uid ?: ""
        listenerFirebaseRealtime()

        etGasPrice.addTextChangedListener(DecimalTextWatcher(etGasPrice))
        etEthanolPrice.addTextChangedListener(DecimalTextWatcher(etEthanolPrice))
        etGasAverage.addTextChangedListener(DecimalTextWatcher(etGasAverage, 1))
        etEthanolAverage.addTextChangedListener(DecimalTextWatcher(etEthanolAverage, 1))

        btCalculate.setOnClickListener {
            saveCarData()
            val proximatela = Intent(this@FormActivity, ResultActivity::class.java)
            proximatela.putExtra("GAS PRICE", etGasPrice.text.toString().toDouble())
            proximatela.putExtra("ETHANOL PRICE", etEthanolPrice.text.toString().toDouble())
            proximatela.putExtra("GAS AVERAGE", etGasAverage.text.toString().toDouble())
            proximatela.putExtra("ETHANOL AVERAGE",
etEthanolAverage.text.toString().toDouble())
            startActivity(proximatela)
        }
    }
```

APP: Calculadora Flex

```
    private fun saveCarData() {
        val carData = CarData(
            etGasPrice.text.toString().toDouble(),
            etEthanolPrice.text.toString().toDouble(),
            etGasAverage.text.toString().toDouble(),
            etEthanolAverage.text.toString().toDouble()
        )
        FirebaseDatabase.getInstance().getReference(firebaseReferenceNode)
            .child(userId)
            .setValue(carData)
    }
    private fun listenerFirebaseRealtime() {
        val database = FirebaseDatabase.getInstance()
        //Define para usar dados off-line
        database.setPersistenceEnabled(true)
        database
            .getReference(firebaseReferenceNode)
            .child(userId)
            .addValueEventListener(object : ValueEventListener {
                override fun onDataChange(dataSnapshot: DataSnapshot) {
                    val carData = dataSnapshot.getValue(CarData::class.java)
                    etGasPrice.setText(carData?.gasPrice.toString())
                    etEthanolPrice.setText(carData?.ethanolPrice.toString())
                    etGasAverage.setText(carData?.gasAverage.toString())
                    etEthanolAverage.setText(carData?.ethanolAverage.toString())
                }


                override fun onCancelled(error: DatabaseError) {
                }
            })
    }
}
```

APP: Calculadora Flex - Persistindo os dados off-line

Os apps do Firebase funcionam mesmo se o app perder temporariamente a conexão de rede. Além disso, o Firebase oferece ferramentas para a permanência local de dados, gerenciamento de presença e tratamento de latência.

APP: Calculadora Flex - Permanência em disco

Os apps do Firebase administram automaticamente interrupções temporárias de rede. Os dados em cache estão disponíveis off-line, e o Firebase reenvia quaisquer gravações quando a conectividade de rede é restaurada.

Quando você ativa a permanência em disco, o app grava os dados localmente no dispositivo para que possa manter o estado enquanto estiver off-line, mesmo que o usuário ou o sistema operacional o reinicie.

É possível ativar a persistência em disco com apenas uma linha de código.

```
FirebaseDatabase.getInstance().setPersistenceEnabled(true)
```

Para mais informações acesse:
https://firebase.google.com/docs/database/android/offline-capabilities?hl=pt-br

APP: Calculadora Flex - Refatorando nosso código

Crie uma nova classe chamada **DatabaseUtil** dentro do package **utils.** Adicione o seguinte código:

```
import com.google.firebase.database.FirebaseDatabase

class DatabaseUtil {
        companion object {
                private val firebaseDatabase: FirebaseDatabase =
        FirebaseDatabase.getInstance()

                init {
                        firebaseDatabase.setPersistenceEnabled(true)
                }

                fun getDatabase() : FirebaseDatabase {
                        return firebaseDatabase
                }
        }
}
```

APP: Calculadora Flex - Refatorando nosso código

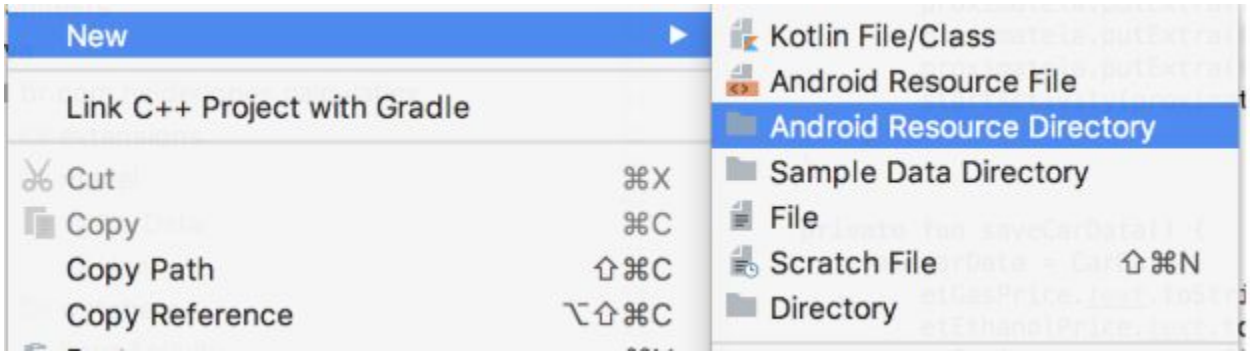
Na classe **FormActivity,** altere configure seu método conforme código abaixo:

```
private fun listenerFirebaseRealtime() {
        DatabaseUtil.getDatabase()
            .getReference(firebaseReferenceNode)
            .child(userId)
            .addValueEventListener(object : ValueEventListener {
                override fun onDataChange(dataSnapshot: DataSnapshot) {
                    val carData =
dataSnapshot.getValue(CarData::class.java)
                    etGasPrice.setText(carData?.gasPrice.toString())

etEthanolPrice.setText(carData?.ethanolPrice.toString())
                    etGasAverage.setText(carData?.gasAverage.toString())

etEthanolAverage.setText(carData?.ethanolAverage.toString())
                }


                override fun onCancelled(error: DatabaseError) {}
            })
    }
```

Limpando os dados do formulário

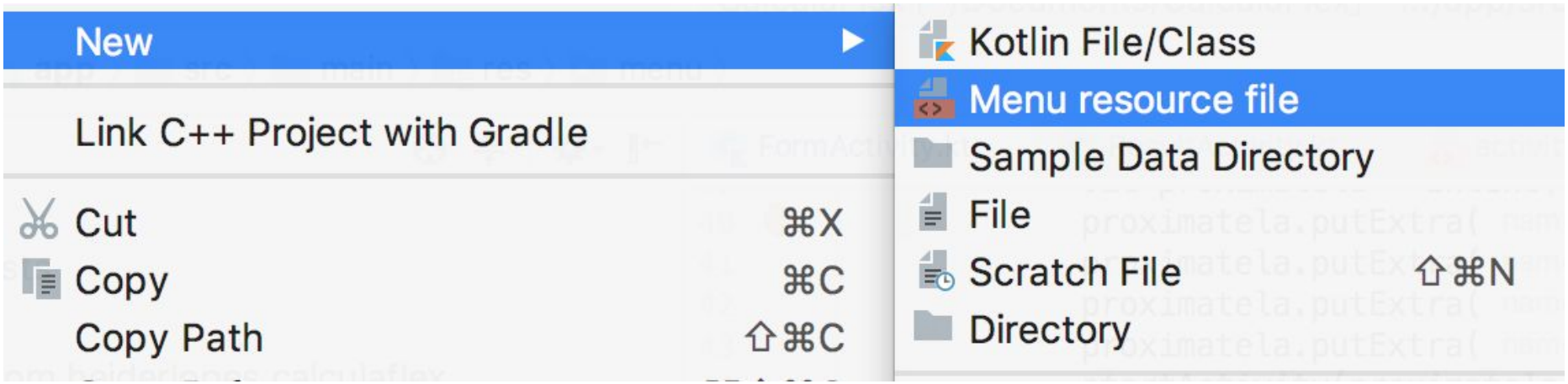
APP: Calculadora Flex - Criando menu

Crie um arquivo chamado **form_menu.xml** dentro da pasta **menu** localizada em **res.** Caso não tenha a pasta execute os passos abaixo para criá-la.



APP: Calculadora Flex - Criando menu

Crie um arquivo chamado **form_menu.xml** dentro da pasta **menu** localizada em **res.** Caso não tenha a pasta execute os passos abaixo para criá-la.



APP: Calculadora Flex - Criando menu

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:app="http://schemas.android.com/apk/res-auto">
            <!--<item-->
            <!--android:id="@+id/action_add"-->
            <!--android:icon="@drawable/ic_add"-->
            <!--app:showAsAction="ifRoom"-->
            <!--android:title="Add">-->
            <!--</item>-->
    <item
        android:id="@+id/action_clear"
        app:showAsAction="never"
        android:title="Limpar dados">
    </item>
    <item
        android:id="@+id/action_logout"
        app:showAsAction="never"
        android:title="Sair">
    </item>
</menu>
```

APP: Calculadora Flex - Criando menu - FormActivity

```
private val defaultClearValueText = "0.0"


override fun onCreateOptionsMenu(menu: Menu): Boolean {
    val inflater = menuInflater
    inflater.inflate(R.menu.menu_form, menu)
    return true
}


override fun onOptionsItemSelected(item: MenuItem): Boolean {
    when (item.getItemId()) {
        R.id.action_clear -> {
            clearData()
            return true
        }
        R.id.action_logout -> {
            logout()
            return true
        }
        else -> return super.onOptionsItemSelected(item)
    }
}
```

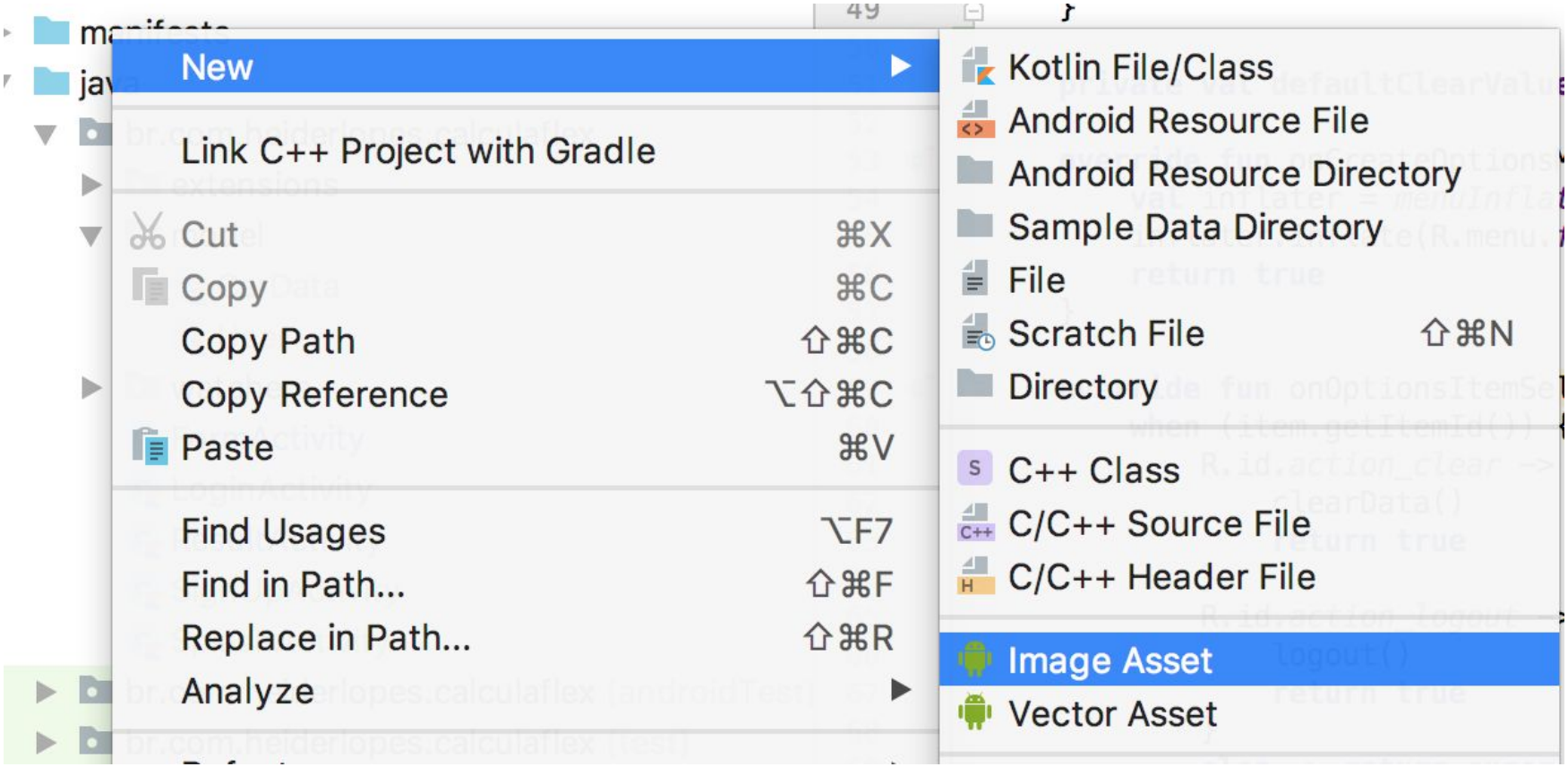
APP: Calculadora Flex - Criando menu - FormActivity

```
private fun logout() {
    mAuth.signOut()
    startActivity(Intent(this, LoginActivity::class.java))
    finish()
}


private fun clearData() {
    etGasPrice.setText(defaultClearValueText)
    etEthanolPrice.setText(defaultClearValueText)
    etGasAverage.setText(defaultClearValueText)
    etEthanolAverage.setText(defaultClearValueText)
}
```

Alterando o ícone do aplicativo

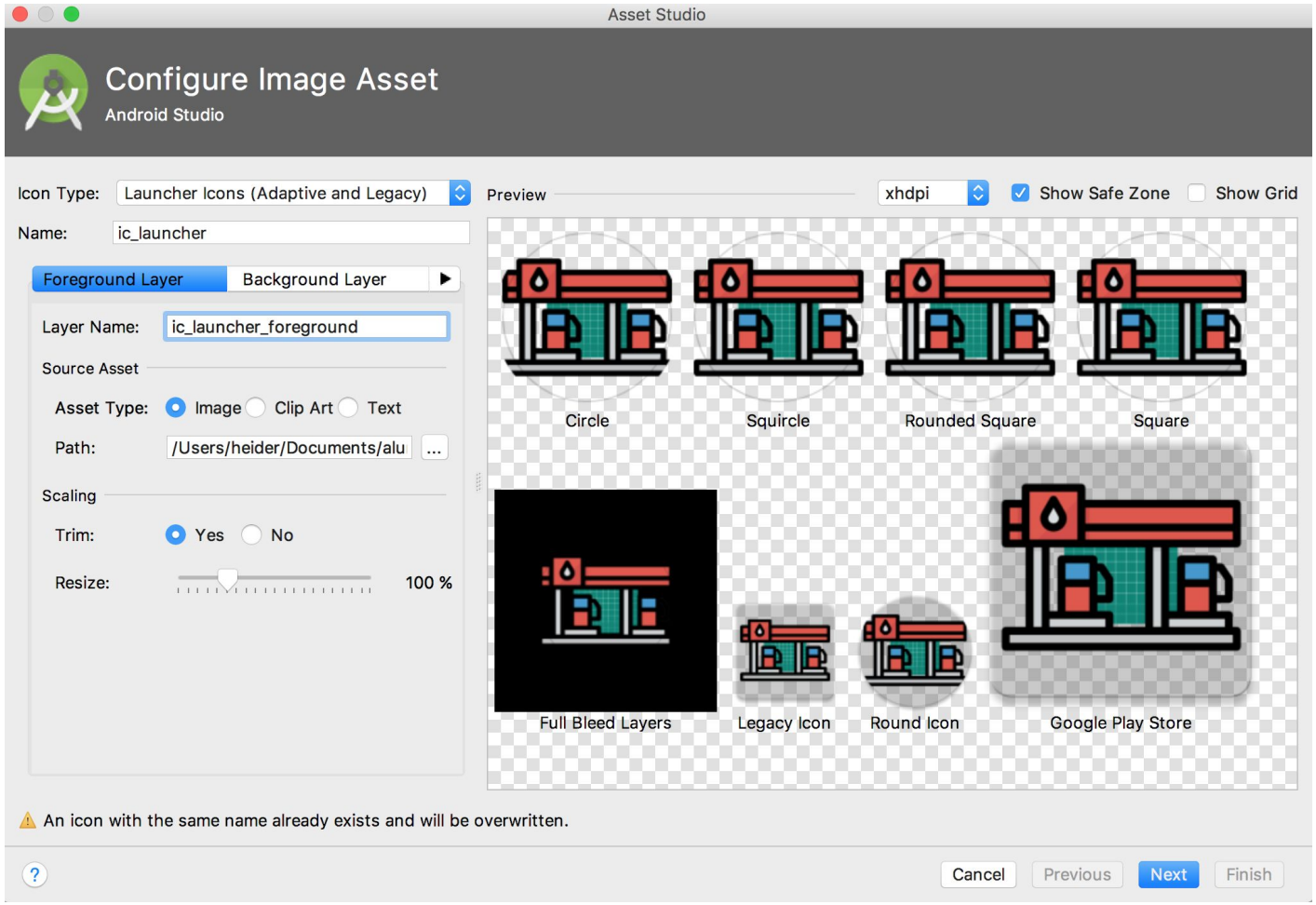
APP: Calculadora Flex - Alterando o ícone do aplicativo

Clique com o botão direito sobre a pasta **res.** Em seguida, clique **New ⇒ Image Asset**.



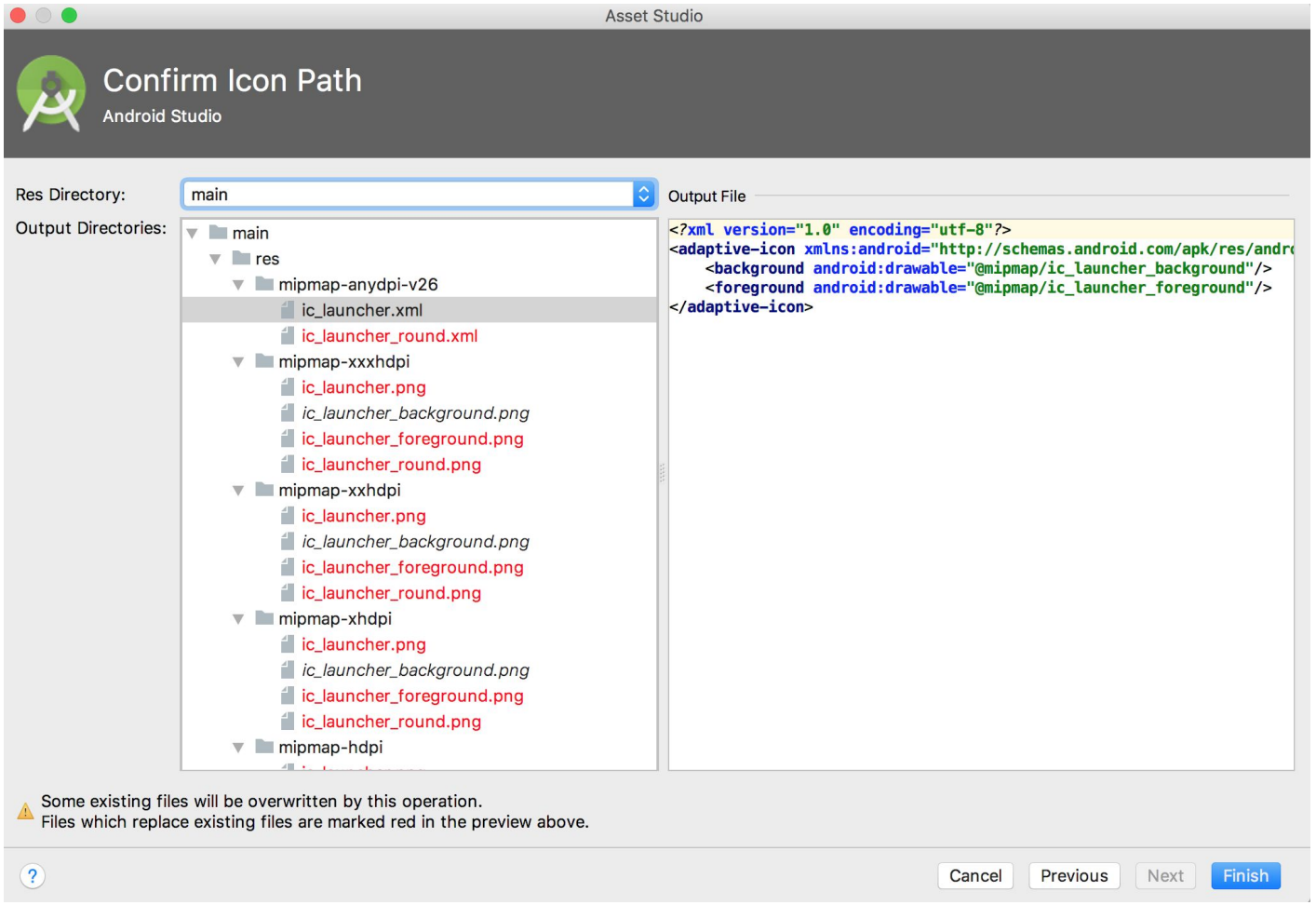
APP: Calculadora Flex - Alterando o ícone do aplicativo

Clique com o botão direito sobre a pasta **res.** Em seguida, clique **New ⇒ Image Asset**.



APP: Calculadora Flex - Alterando o ícone do aplicativo

Clique com o botão direito sobre a pasta **res.** Em seguida, clique **New ⇒ Image Asset**.



Integrando o Firebase Cloud Messaging (FCM)

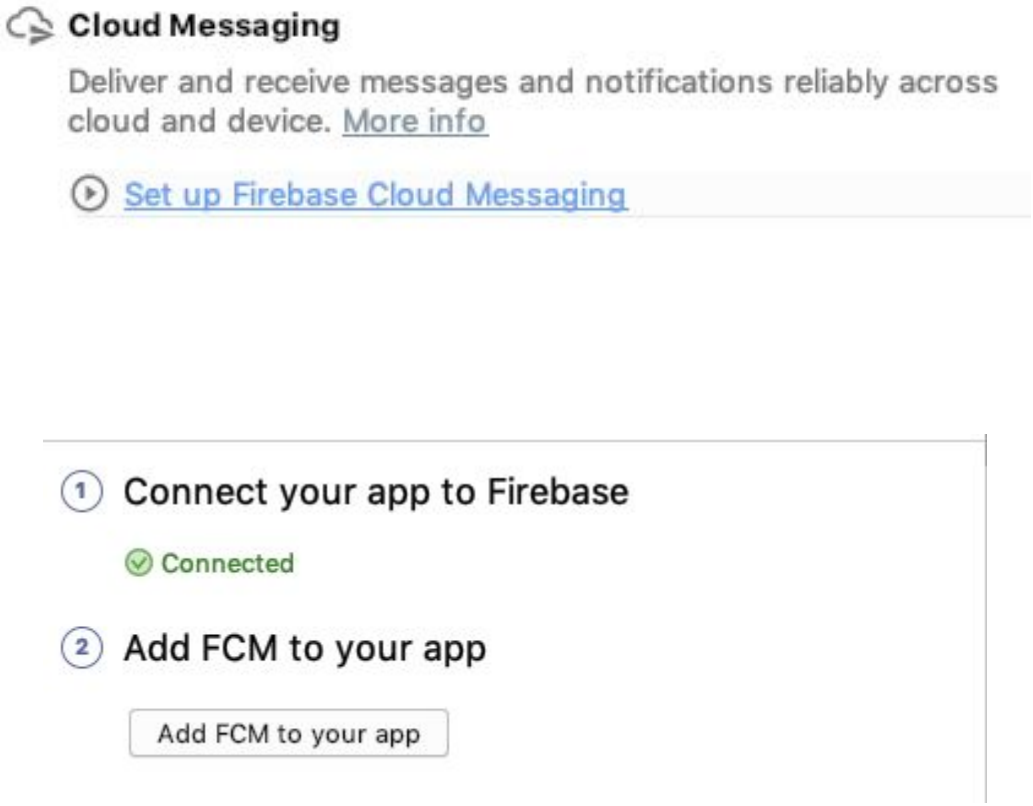
Firebase Cloud Messaging - FCM

É uma solução de mensagens entre plataformas que permite a entrega confiável de mensagens sem custo.

Usando o FCM, você pode notificar um app cliente de que novos e-mails ou outros dados estão disponíveis para sincronização. Você pode enviar messages de notificação para promover novas interações e a retenção de usuários. Para casos de uso como messages instantâneas, uma message pode transferir uma payload de até 4 KB para um app cliente.

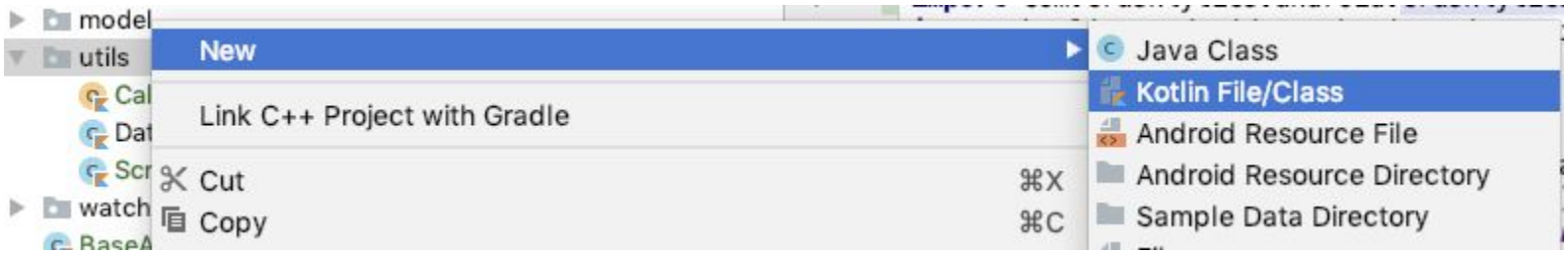
Firebase Cloud Messaging - FCM

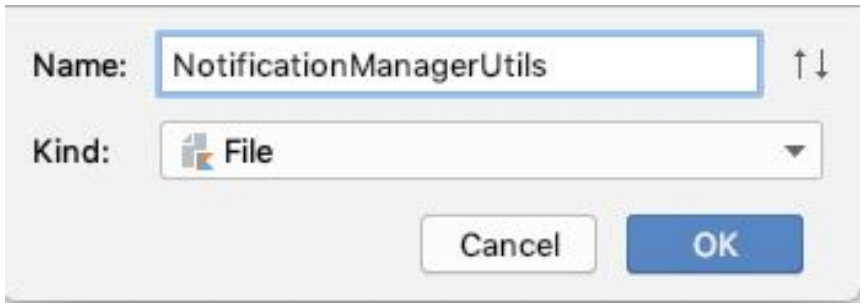
Clica no menu **Tools ⇒ Firebase**



Firebase Cloud Messaging - FCM

Crie um arquivo chamado **NotificationManagerUtils.kt** dentro da pasta **utils**



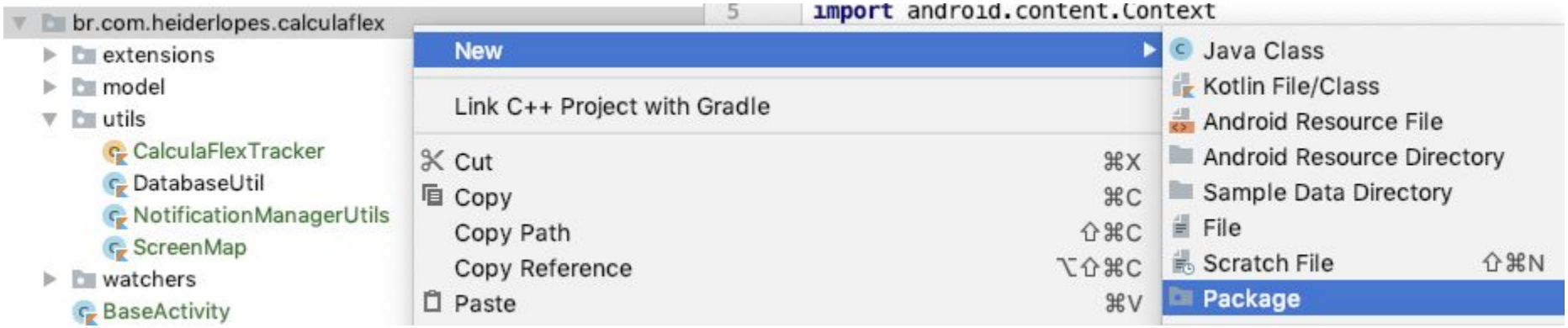


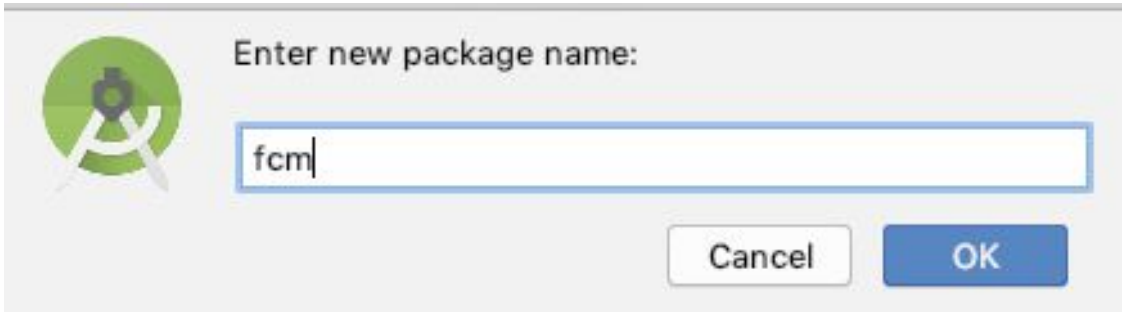
Firebase Cloud Messaging - FCM

```
class NotificationManagerUtils(private val context: Context) {

    companion object {
        private val CHANNEL_ID = "YOUR_CHANNEL_ID"
        private val CHANNEL_NAME = "Miscellaneous"
        private val CHANNEL_DESCRIPTION = "description"
    }

    @RequiresApi(Build.VERSION_CODES.O)
    fun getMainNotificationId(): String {
        return CHANNEL_ID
    }

    @RequiresApi(Build.VERSION_CODES.O)
    fun createMainNotificationChannel() {
        val id = CHANNEL_ID
        val name = CHANNEL_NAME
        val description = CHANNEL_DESCRIPTION
        val importance = android.app.NotificationManager.IMPORTANCE_HIGH
        val mChannel = NotificationChannel(id, name, importance)
        mChannel.description = description
        mChannel.enableLights(true)
        mChannel.lockscreenVisibility = Notification.VISIBILITY_PUBLIC
        mChannel.lightColor = Color.RED
        mChannel.enableVibration(true)
        val mNotificationManager = context.getSystemService(Context.NOTIFICATION_SERVICE) as android.app.NotificationManager
        mNotificationManager.createNotificationChannel(mChannel)
    }
}
```

Firebase Cloud Messaging - FCM

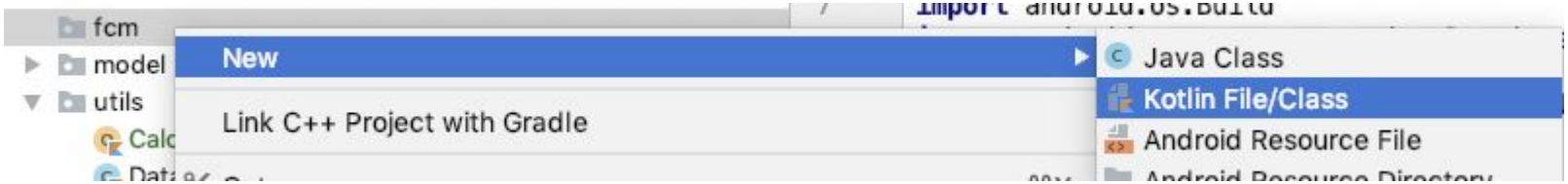
Crie um pacote chamado **fcm** no package default da aplicação

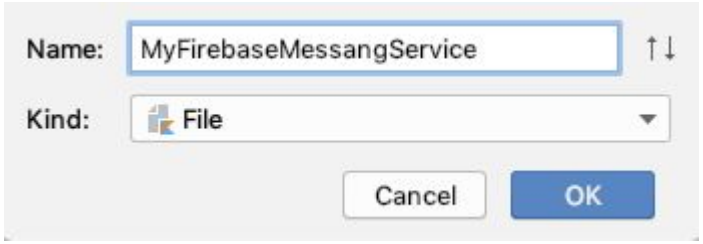




Firebase Cloud Messaging - FCM

Dentro do pacote **fcm** crie uma classe chamada **MyFirebaseMessagingService**





Abra o arquivo **DatabaseUtil** e adicione o seguinte método:

```
fun saveToken(token: String?) {
    val user = FirebaseAuth.getInstance().currentUser?.uid ?: ""
    if (user != "") {
        FirebaseDatabase.getInstance().getReference("UsersTokens")
            .child(FirebaseAuth.getInstance().currentUser?.uid ?: "")
            .setValue(token)

    }
}
```

Firebase Cloud Messaging - FCM

```
class MyFirebaseMessagingService : FirebaseMessagingService() {
    companion object {
      private const val notificationID = 1000
    }

    override fun onNewToken(s: String?) {
      super.onNewToken(s)
      DatabaseUtil.saveToken(s)
    }


    override fun onMessageReceived(remoteMessage: RemoteMessage) {
      super.onMessageReceived(remoteMessage)
      configureNotification(remoteMessage, getLastScreen())
    }

    //..
}
```

Firebase Cloud Messaging - FCM

```
class MyFirebaseMessagingService : FirebaseMessagingService() {

    //..
    private fun getLastScreen(): PendingIntent {
      val nIntent = packageManager.getLaunchIntentForPackage(packageName)
      return PendingIntent.getActivity(
          this, 0, nIntent,
          PendingIntent.FLAG_UPDATE_CURRENT
      )
    }

    private fun configureNotification(remoteMessage: RemoteMessage, pendingIntent:
    PendingIntent?) {
      val title = remoteMessage.notification?.title ?: getString(R.string.app_name)
      val message = remoteMessage.notification?.body ?: ""
      showNotification(pendingIntent, title, message)
    }
}
```

Firebase Cloud Messaging - FCM

```
class MyFirebaseMessagingService : FirebaseMessagingService() {

private fun createNotificationCompatBuilder(context: Context): NotificationCompat.Builder {
        return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationCompat.Builder(context,
      NotificationManagerUtils(context).getMainNotificationId())
        } else {
            NotificationCompat.Builder(context)
        }
    }
}
```

Firebase Cloud Messaging - FCM

```
class MyFirebaseMessagingService : FirebaseMessagingService() {
    private fun showNotification(pendingIntent: PendingIntent?, title: String, message: String) {
      val defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION)

      val mBuilder =
          createNotificationCompatBuilder(applicationContext)
              .setContentTitle(title)
              .setSound(defaultSoundUri)
              .setStyle(NotificationCompat.BigTextStyle())
              .setVisibility(NotificationCompat.VISIBILITY_PUBLIC)
              .setSmallIcon(R.mipmap.ic_launcher)
              .setAutoCancel(true)
              .setContentIntent(pendingIntent)
              .setContentText(message)
              .setPriority(NotificationCompat.PRIORITY_MAX)
              .setDefaults(NotificationCompat.DEFAULT_VIBRATE)

      val mNotificationManager =
          applicationContext.getSystemService(Context.NOTIFICATION_SERVICE) as
    NotificationManager
      mNotificationManager.notify(notificationID, mBuilder.build())
    }
}
```

Abra o arquivo **AndroidManifest.xml** e declare o serviço criado da seguinte forma:

```
<service
      android:name=".fcm.MyFirebaseMessagingService"
      android:exported="false">
   <intent-filter>
      <action
android:name="com.google.firebase.MESSAGING_EVENT" />
   </intent-filter>
</service>
```

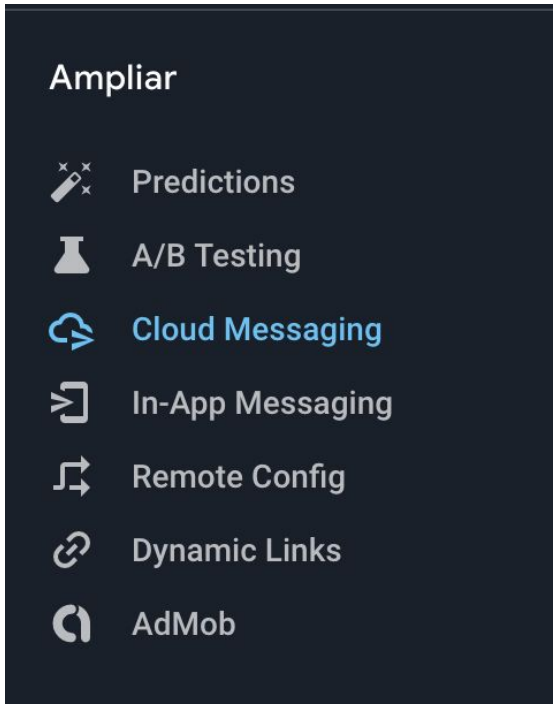
Firebase Cloud Messaging - FCM

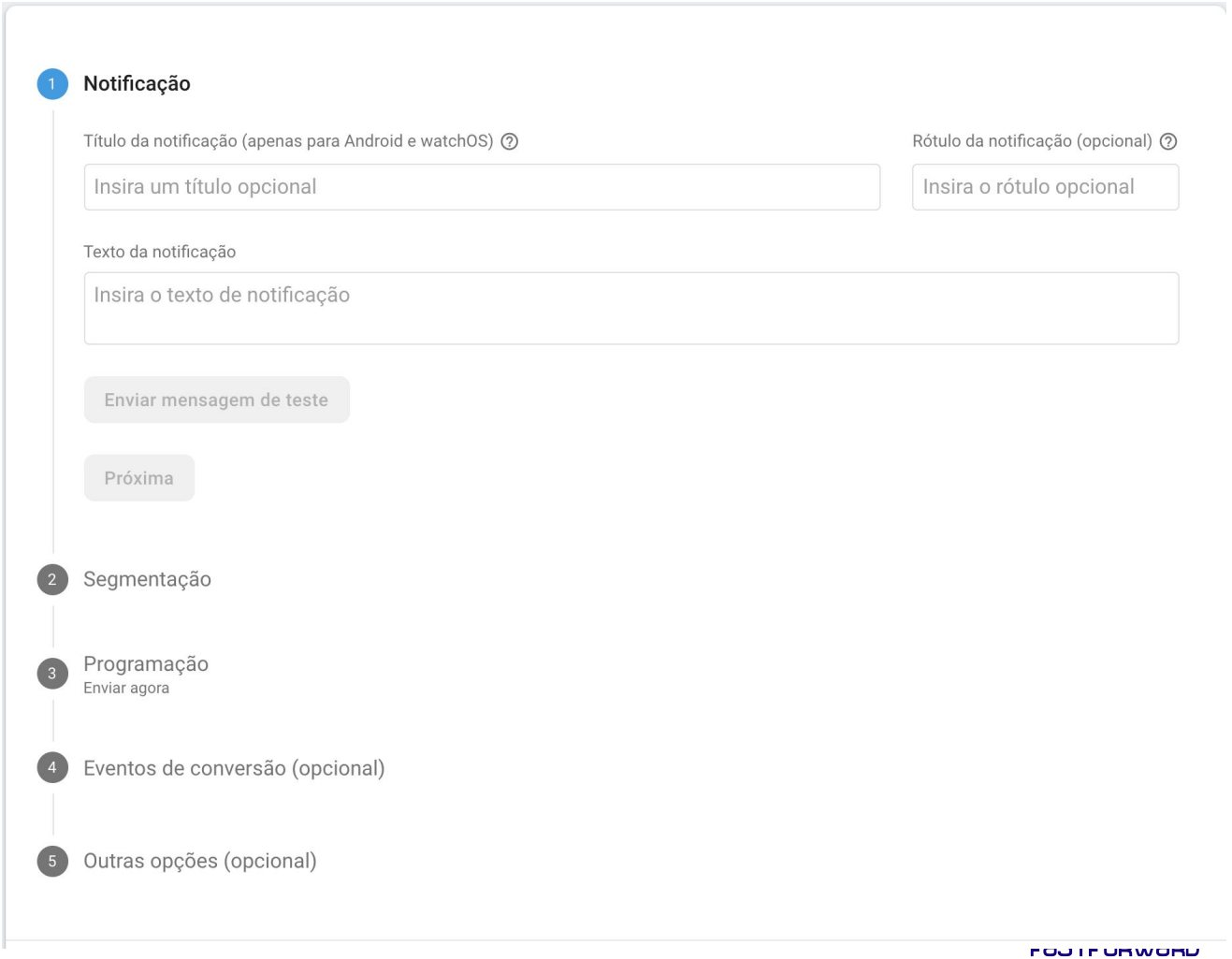
Abra o arquivo **LoginActivity.kt** e adicione a linha em negrito abaixo:

```
private fun goToHome() {
   FirebaseInstanceId.getInstance().instanceId.addOnSuccessListener(this) {
instanceIdResult ->
       val newToken = instanceIdResult.token
       DatabaseUtil.saveToken(newToken)
   }

  //……
}
```

Firebase Cloud Messaging - FCM

Abra o Cloud Messaging no Console do Firebase e crie sua notificação





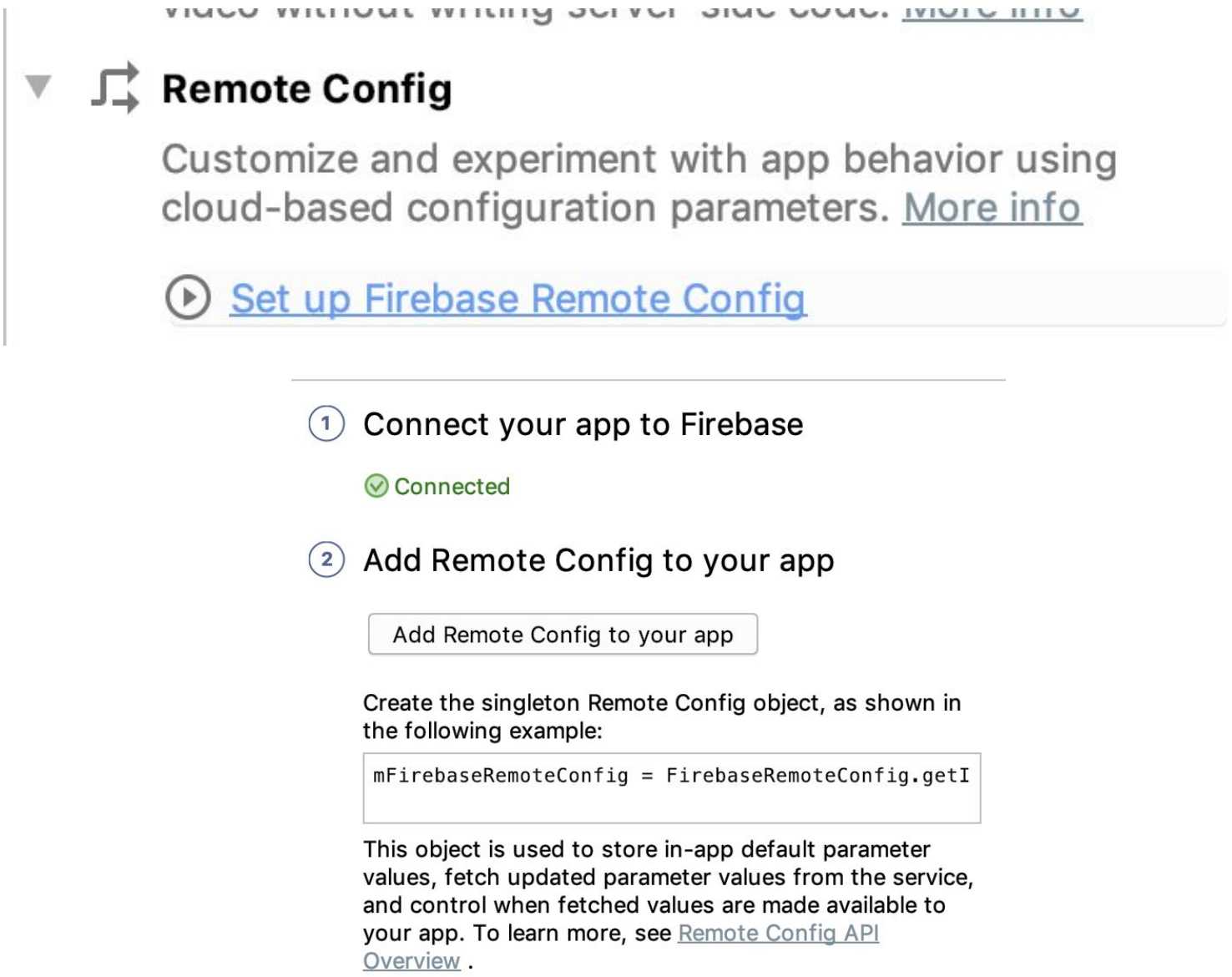
Firebase Remote Config

Firebase Remote Config

É um serviço em nuvem que permite alterar o comportamento e a aparência do aplicativo sem exigir que os usuários baixem uma atualização de aplicativo. Ao usar o Remote config, você cria valores padrão no aplicativo que controlam o comportamento e a aparência do seu aplicativo.

Em seguida, você poderá usar posteriormente o console do Firebase ou a API REST de configuração remota para substituir os valores padrão no aplicativo de todos os usuários do aplicativo ou de segmentos da sua base de usuários.

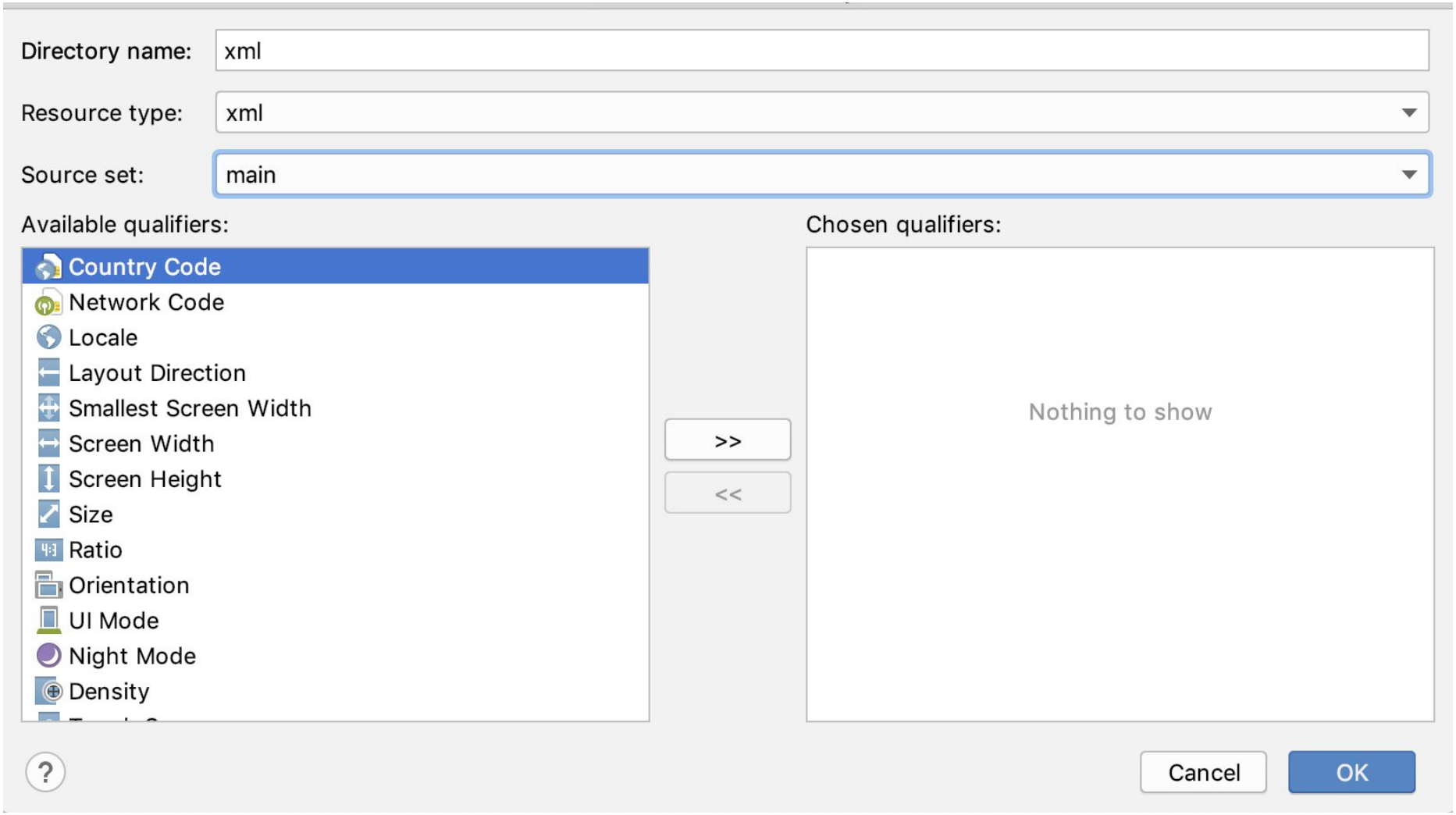
Firebase Remote Config



Firebase Remote Config

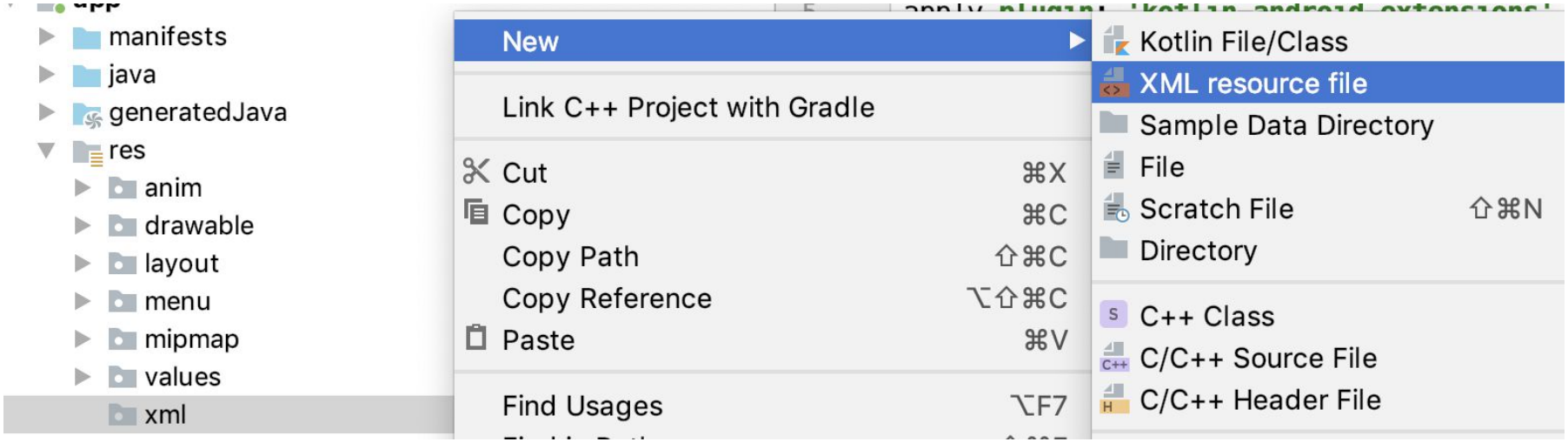
Agora é hora de implementar a configuração do Firebase Remote em nosso aplicativo. Primeiro precisamos criar um arquivo .xml na pasta **XML (dentro de resources - caso não exista basta criá-la)**.

Firebase Remote Config



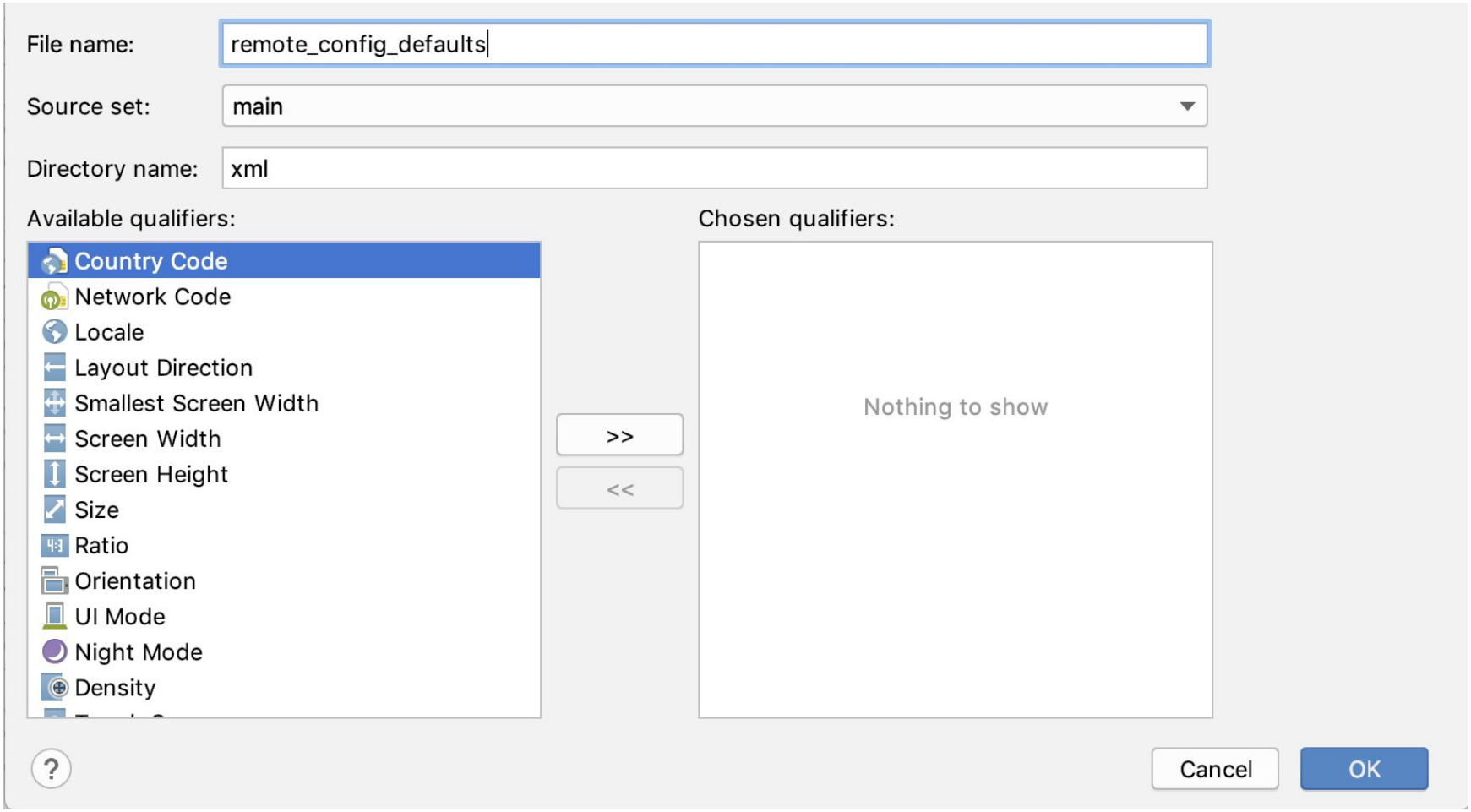
Firebase Remote Config

Dentro dessa pasta crie o arquivo **remote_config_defaults.xml**



Firebase Remote Config

Dentro dessa pasta crie o arquivo **remote_config_defaults.xml**



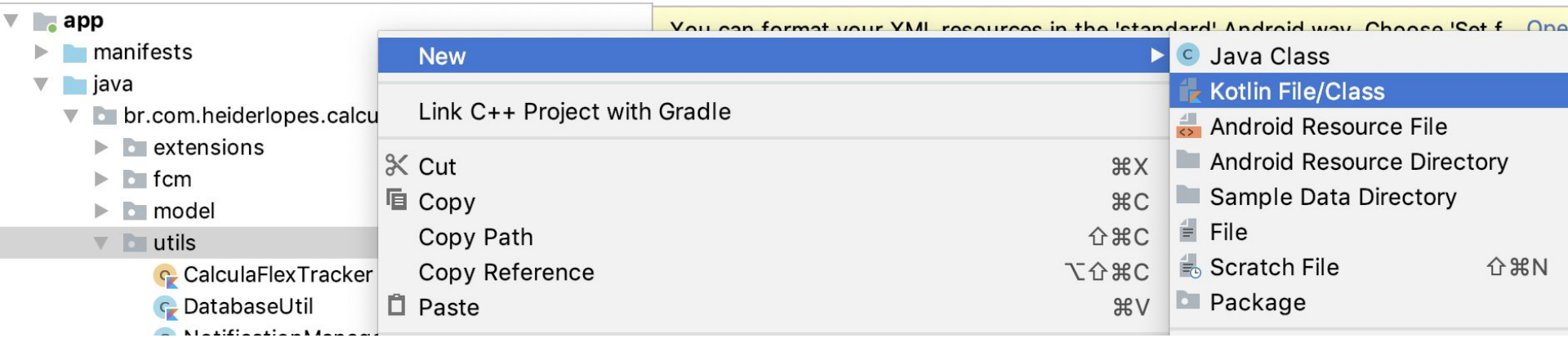
Firebase Remote Config

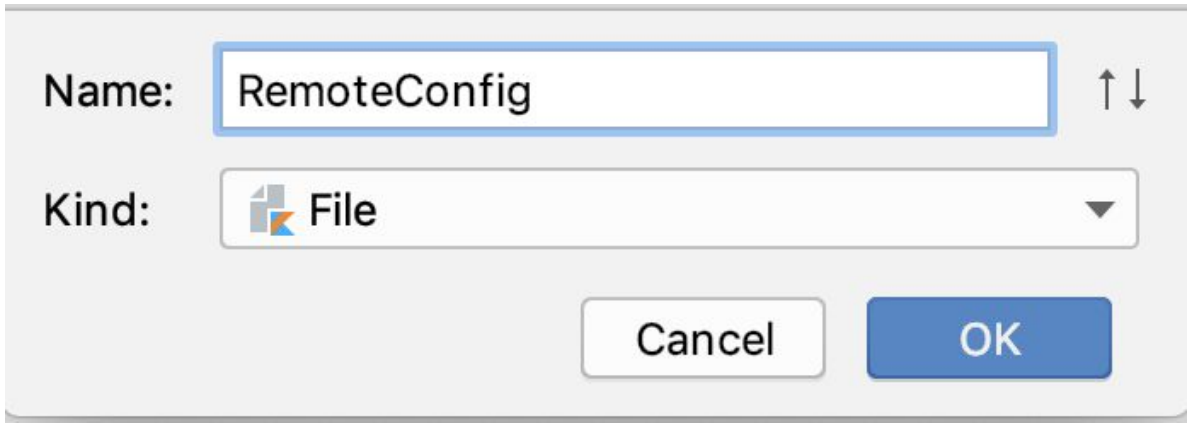
Adicione o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<defaultsMap>
  <entry>
    <key>min_version_app</key>
    <value>1</value>
  </entry>
</defaultsMap>
```

Firebase Remote Config

Crie uma nova classe Kotlin chamada **RemoteConfig.kt**

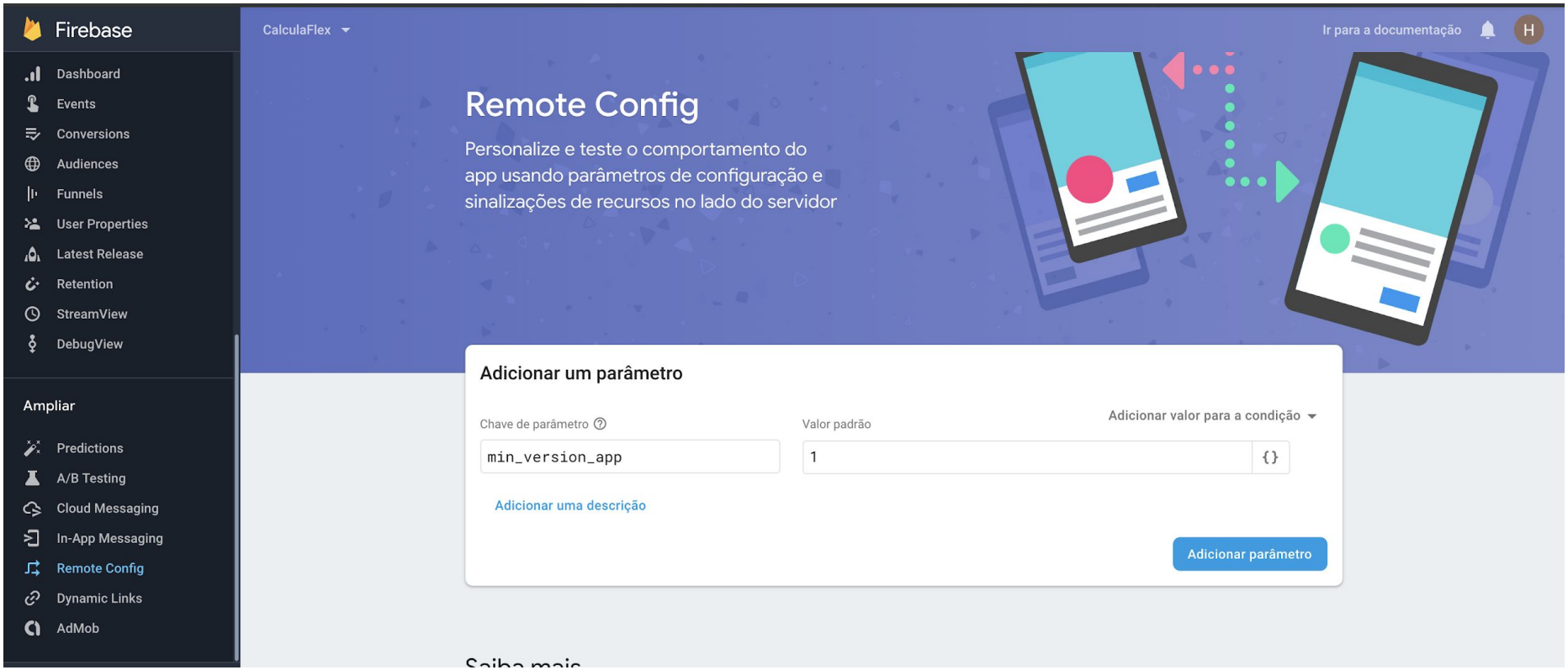




Firebase Remote Config

```
object RemoteConfig {

        fun getFirebaseRemoteConfig(): FirebaseRemoteConfig {
         val mFirebaseRemoteConfig: FirebaseRemoteConfig? = FirebaseRemoteConfig.getInstance()
         val configSettings = FirebaseRemoteConfigSettings.Builder()
                .setDeveloperModeEnabled(BuildConfig.DEBUG)
                .build()
         mFirebaseRemoteConfig?.setConfigSettings(configSettings)
         mFirebaseRemoteConfig?.setDefaults(R.xml.remote_config_defaults)
         return mFirebaseRemoteConfig!!
        }

        fun remoteConfigFetch(): Task<Void> {
         //var cacheExpiration: Long = 3600
         var cacheExpiration: Long = 720
         if (getFirebaseRemoteConfig().info.configSettings.isDeveloperModeEnabled) {
            cacheExpiration = 0
         }
         return getFirebaseRemoteConfig().fetch(cacheExpiration)
        }
}
```

Firebase Remote Config



Firebase Remote Config

Inclua o método abaixo na **SplashScreen.kt:**

```
private fun showAlertMinVersion() {
  AlertDialog.Builder(this)
      .setTitle("Ops")
      .setMessage("Você esta utilizando uma versão muito antiga do aplicativo. Deseja atualizar?")

      .setPositiveButton("Sim") { dialog, which ->
          var intent: Intent
          try {
              intent = Intent(Intent.ACTION_VIEW, Uri.parse("market://details?id=$packageName"))
              startActivity(intent)
          } catch (e: android.content.ActivityNotFoundException) {
              intent = Intent(
                  Intent.ACTION_VIEW,
                  Uri.parse("https://play.google.com/store/apps/details?id=$packageName")
              )
              startActivity(intent)
          }
      }

      .setNegativeButton("Não") { dialog, which ->
          finish()
      }
      .setIcon(android.R.drawable.ic_dialog_alert)
      .show()
}
```

Firebase Remote Config

Inclua o método abaixo na **SplashScreen.kt:**

```
private fun continueApp() {
    val preferences = getSharedPreferences("user_preferences", Context.MODE_PRIVATE)
    val isFirstOpen = preferences.getBoolean("open_first", true)

    if (isFirstOpen) {
        showLogin()
    } else {
        markAppAlreadyOpen(preferences)
        showSplash()
    }
}
```

Firebase Remote Config

Altere o método **onCreate** igual abaixo:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_splash)

    RemoteConfig.remoteConfigFetch()
    .addOnCompleteListener(this) { task ->
        if (task.isSuccessful) {
            RemoteConfig.getFirebaseRemoteConfig().activateFetched()
            val minVersionApp = RemoteConfig.getFirebaseRemoteConfig()
                .getLong("min_version_app")
                .toInt()
            if (minVersionApp <= BuildConfig.VERSION_CODE)
                continueApp()
            else
                showAlertMinVersion()
        } else
            continueApp()
    }

}
```

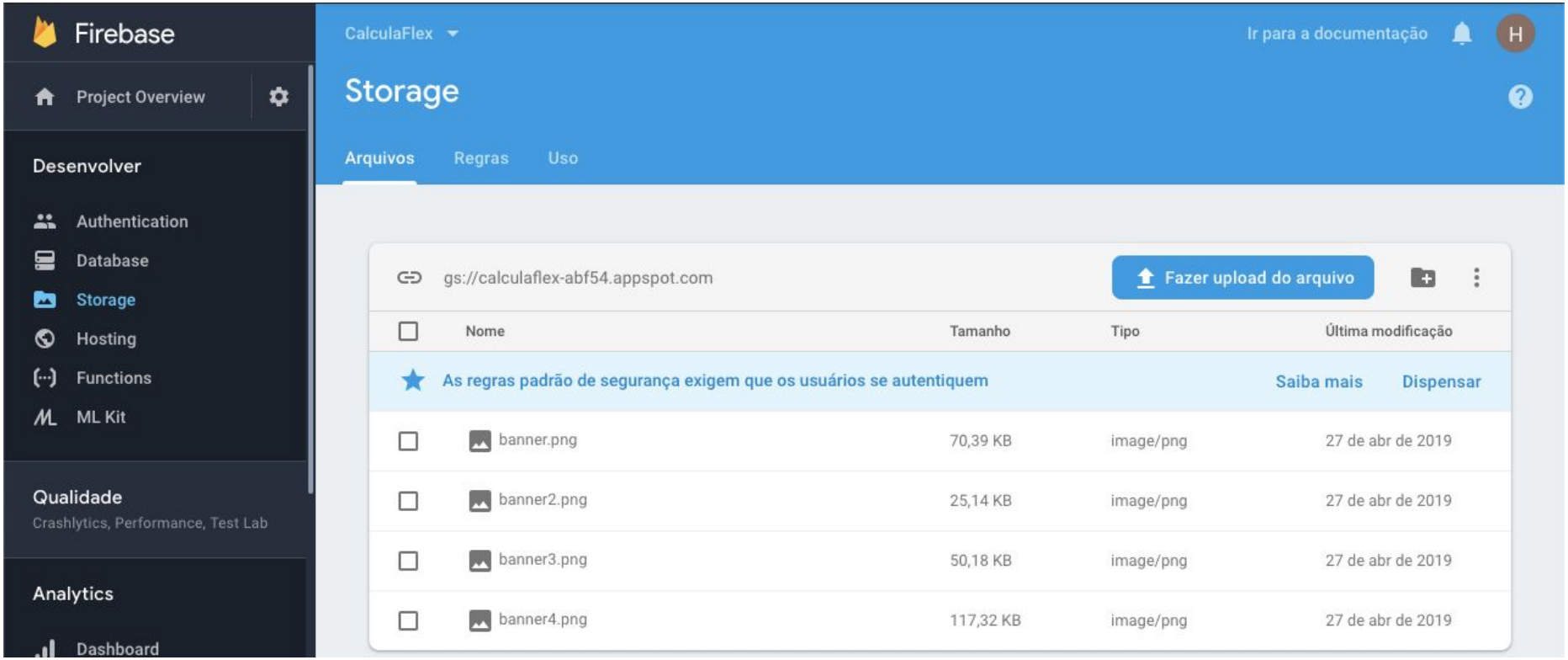
Firebase Remote Config

Firebase Storage

O Cloud Storage para Firebase é um serviço de armazenamento de objetos poderoso, simples e econômico criado para a escala do Google.

Com os SDKs do Firebase para Cloud Storage, você usa a segurança do Google para fazer o upload e o download de arquivos nos aplicativos Firebase, independentemente da qualidade da rede.

Firebase Storage

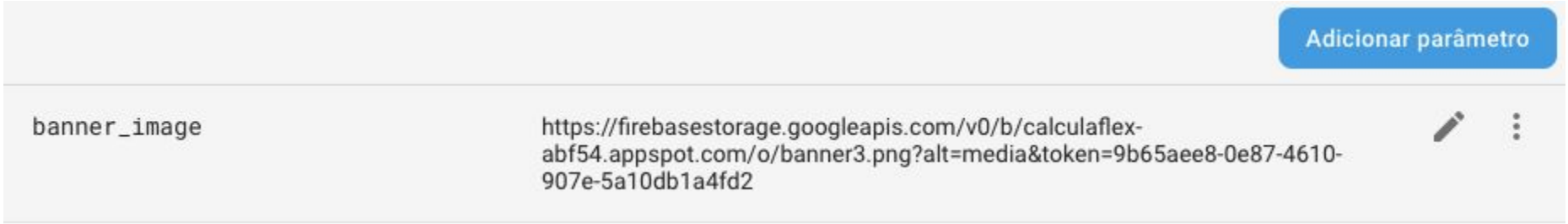


Imagens disponíveis em:

https://github.com/heiderlopes/calcula-flex-banners

Firebase Storage

No Remote Config adicione uma nova configuração:



Firebase Storage

Consumindo uma imagem da web.

Abra o arquivo **build.gradle** do módulo e adicione a seguinte biblioteca:

```
implementation 'com.squareup.picasso:picasso:2.71828'
```

Em seguida realize o **Sync Now.**

Abra o arquivo **FormActivity.kt** e adicione o seguinte código:

```
private fun loadBanner() {
    val loginBanner = RemoteConfig.getFirebaseRemoteConfig()
        .getString("banner_image")

    Picasso.get().load(loginBanner).into(ivBanner)
}
```

Chame esse método dentro do **onCreate** da **FormActivity:**

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_form)

    //…..
    loadBanner()

    //…..
}
```

Integrando o Firebase Crashlytics

Firebase Crashlytics

Para adicionar a dependência do projeto siga os passos do link:

https://firebase.google.com/docs/crashlytics/get-started?authuser=0&platform=android#unity

Firebase Crashlytics

Abra o arquivo **build.gradle** do projeto e configure conforme imagem abaixo:

```
buildscript {
    repositories {
        // ...

        // Add repository
        maven {
           url 'https://maven.fabric.io/public'
        }
    }
    dependencies {
        // ...

        // Check for v3.1.2 or higher
        classpath 'com.google.gms.google-services:4.2.0'

        // Add dependency
        classpath 'io.fabric.tools:gradle:1.26.1'
    }
}

allprojects {
    // ...
    repositories {
       // ...

       // Check that you have the following line (if not, add it):
       google()  // Google's Maven repository
       // ...
    }
}
```

Firebase Crashlytics

Abra o arquivo **build.gradle** do módulo e configure conforme imagem abaixo:

```
apply plugin: 'com.android.application'
apply plugin: 'io.fabric'

dependencies {
    // ...

    // Check for v11.4.2 or higher
    implementation 'com.google.firebase:firebase-core:16.0.8'

    // Add dependency
    implementation 'com.crashlytics.sdk.android:crashlytics:2.9.9'
}
```

Firebase Crashlytics

Forçando um crash

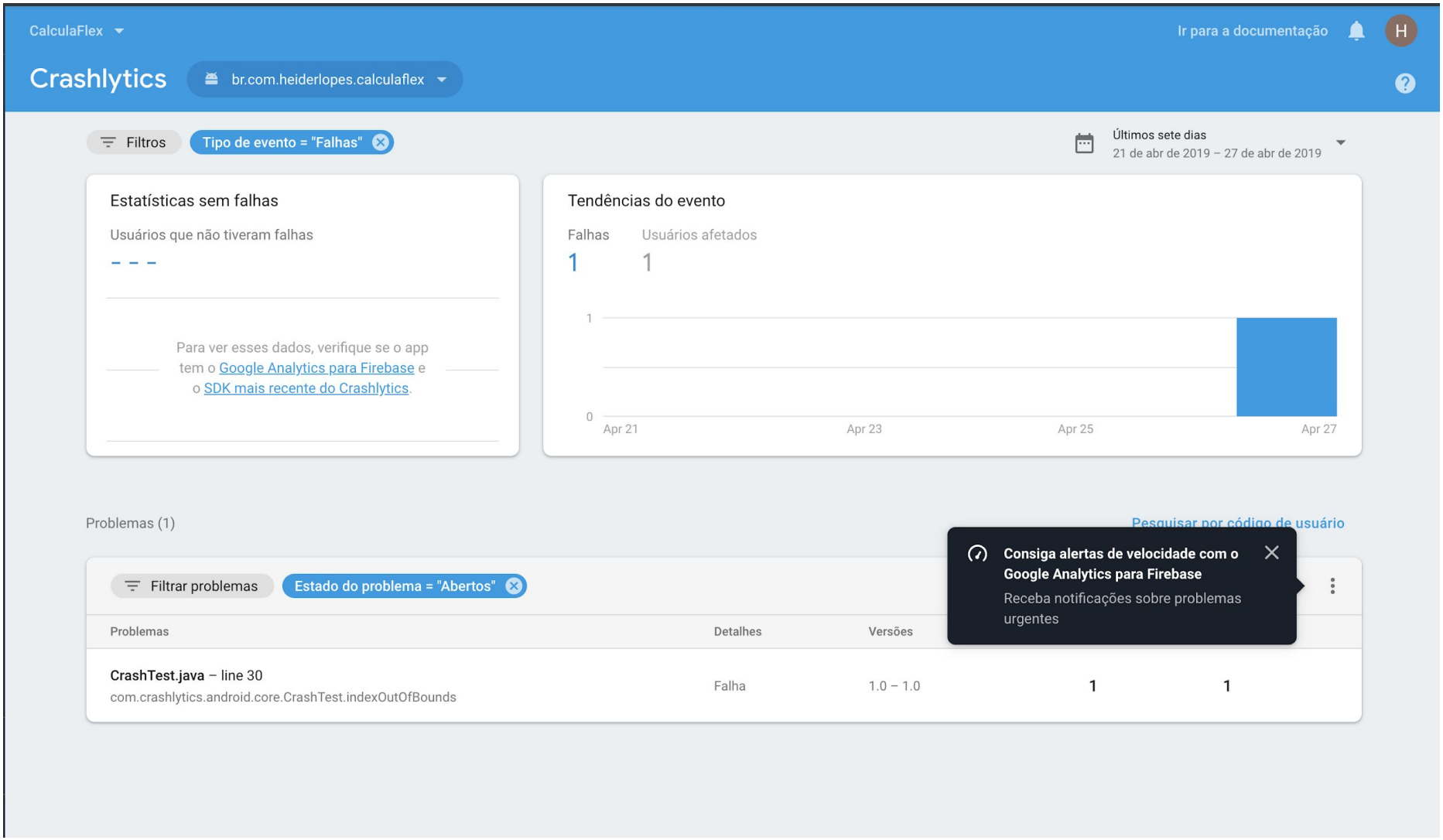
```
Crashlytics.getInstance().crash()
```

Enviando um log

```
Crashlytics.log("Exemplo de mensagem de erro.")
```

Rode o projeto

Firebase Crashlytics



Integrando o Firebase Analytics

Firebase Analytics

O Analytics integra-se a todos os recursos do Firebase e fornece relatórios ilimitados para até 500 eventos distintos que você pode definir usando o SDK do Firebase.

Os relatórios do **Firebase Analytics** ajudam a entender claramente como os usuários se comportam, o que permite tomarmos decisões fundamentadas sobre marketing de aplicativos e otimizações de desempenho.

O Firebase Analytics ajuda você a entender como as pessoas usam seu aplicativo.

Firebase Analytics

O **SDK do Firebase** captura automaticamente vários eventos e propriedades do usuário e também permite que você defina seus próprios eventos personalizados para avaliar as coisas que são importantes para sua empresa.

Depois que os dados são capturados, eles ficam disponíveis em um painel através do console do Firebase. Esse painel fornece informações detalhadas sobre seus dados, desde dados resumidos, como usuários ativos e dados demográficos, até dados mais detalhados, como a identificação dos itens mais comprados.

O Firebase Analytics também registra automaticamente os eventos que correspondem às suas notificações do Firebase e fornece relatórios sobre o impacto de cada campanha.

Firebase Analytics

Nome do Evento	Trigger
first_open	Esse evento não é acionado quando um usuário faz o download do aplicativo em um dispositivo, mas quando o usa pela primeira vez.
in_app_purchase	quando um usuário conclui uma compra no aplicativo que é processada pela App Store no iTunes ou no Google Play. O ID do produto, o nome do produto, a moeda e a quantidade são transmitidos como parâmetros. Esse evento é acionado apenas pelas versões do seu aplicativo que incluem o SDK do Firebase. Além disso, a receita da assinatura, a receita paga de compra do aplicativo e os reembolsos não são acompanhados automaticamente e, portanto, a receita informada pode diferir dos valores que você vê no Console do desenvolvedor do Google Play.
user_engagement	Periodicamente, enquanto o aplicativo está em primeiro plano.
app_update	Quando o aplicativo é atualizado para uma nova versão e aberto novamente.
app_remove	Quando o aplicativo é removido ou "desinstalado" de um dispositivo Android.
os_update	Quando o sistema operacional do dispositivo é atualizado para uma nova versão.

Firebase Analytics

Nome do Evento	Trigger
app_clear_data	quando o usuário redefine / limpa os dados do aplicativo, removendo todas as configurações e dados de login.
app_exception	quando o aplicativo falha ou lança uma exceção.
notification_foreground	quando uma notificação enviada pelo Firebase Cloud Messaging é recebida enquanto o aplicativo está em primeiro plano.
notification_receive	quando uma notificação enviada pelo Firebase Cloud Messaging é recebida por um dispositivo quando o aplicativo está em segundo plano. Apenas aplicativos para Android.
notification_open	quando um usuário abre uma notificação enviada pelo Firebase Cloud Messaging.
notification_dismiss	quando um usuário descarta uma notificação enviada pelo Firebase Cloud Messaging. Apenas aplicativos para Android.

Firebase Analytics

O Firebase também coleta automaticamente as propriedades do usuário, como **versão do aplicativo, modelo do dispositivo, sexo, idade, interesses, versão do sistema operacional, novo / estabelecido**.

Estas informações podem ser vista diretamente no painel sem escrever nenhum código adicional.

Firebase Analytics

Log an Analytics event

Add Analytics to your app to log events and define audiences with user properties.

Launch in browser

1 Connect your app to Firebase

Connected

2 Add Analytics to your app

Add Analytics to your app

Declare the `com.google.firebase.analytics.FirebaseAnalytics` object at the top of your activity

```
private FirebaseAnalytics mFirebaseAnalytics;
```

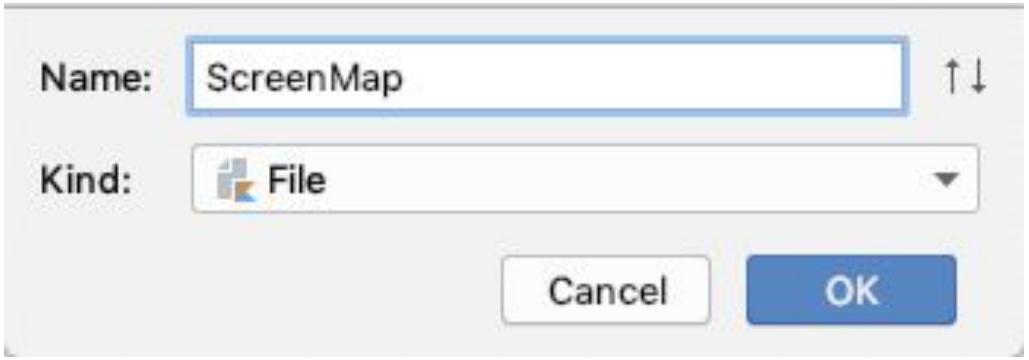
Then initialize it in the `onCreate()` method

```
// Obtain the FirebaseAnalytics instance.
mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);
```

3 Log events

Firebase Analytics

Crie uma nova classe chamada **CalculaFlexTracker.kt** dentro do package **utils**



Firebase Analytics

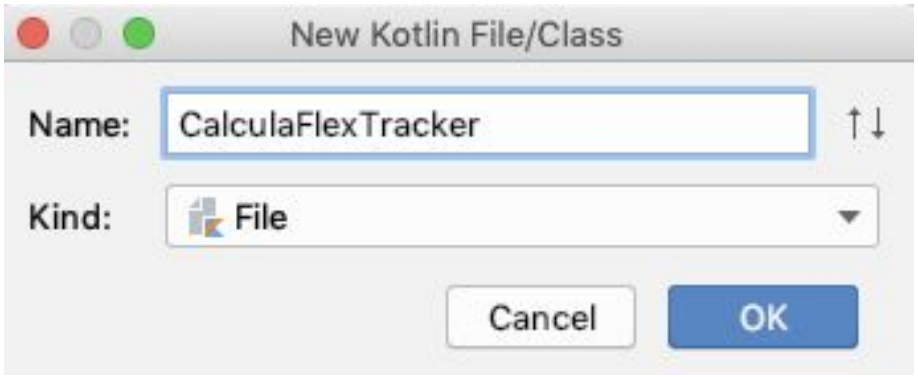
```
class ScreenMap {

    companion object {
        val SCREEN_NOT_TRACKING = "SCREEN_NOT_TRACKING"

        private fun getScreenNameBy(className: String): String {
            val screensNames = getScreenNames()
            return if (screensNames[className] == null) SCREEN_NOT_TRACKING else screensNames[className]!!
        }

        fun getScreenNameBy(activty: Activity): String {
            return getScreenNameBy(activty::class.java.simpleName)
        }

        fun getClassName(screenName: String): String {
            val screenNames = getScreenNames()
            for (o in screenNames.keys) {
                if (screenNames[o] == screenName) {
                    return o
                }
            }
            return ""
        }

        // TRECHO DO CÓDIGO DO PRÓXIMO SLIDE DEVERÁ SER INSERIDO AQUI
    }
}
```

Firebase Analytics

```
private fun getScreenNames(): Map<String, String> {
        return mapOf(
            //Login
            Pair(FormActivity::class.java.simpleName, "Cadastro_Formulario"),
            Pair(LoginActivity::class.java.simpleName, "Login_Usuario"),
            Pair(ResultActivity::class.java.simpleName, "Cálculo_Resultado"),
            Pair(SignUpActivity::class.java.simpleName, "Criacao_Usuario"),
            Pair(SplashActivity::class.java.simpleName, "Splash")
        )
    }
```

Firebase Analytics

Crie uma nova classe chamada **CalculaFlexTracker.kt** dentro do package **utils**

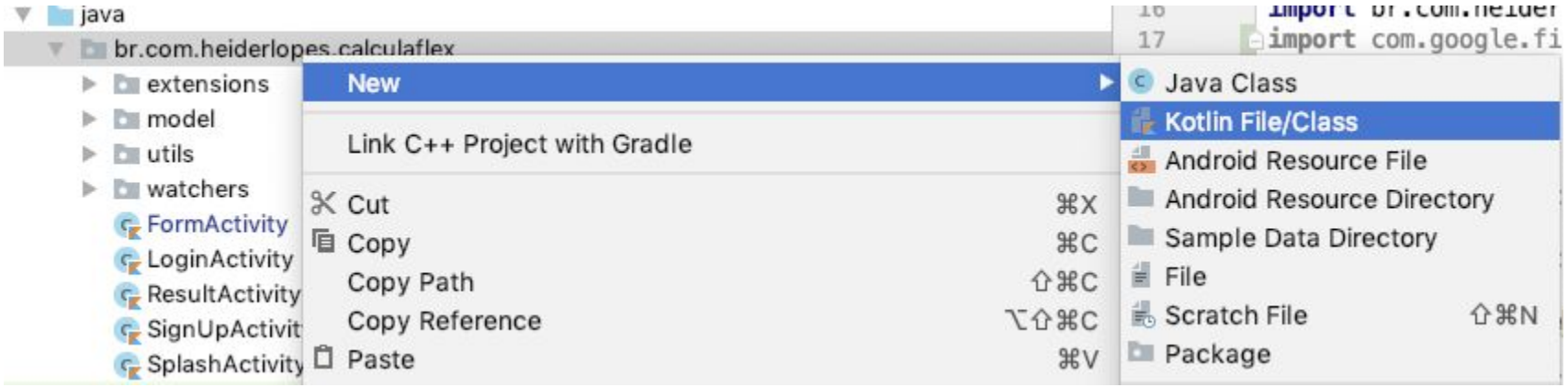


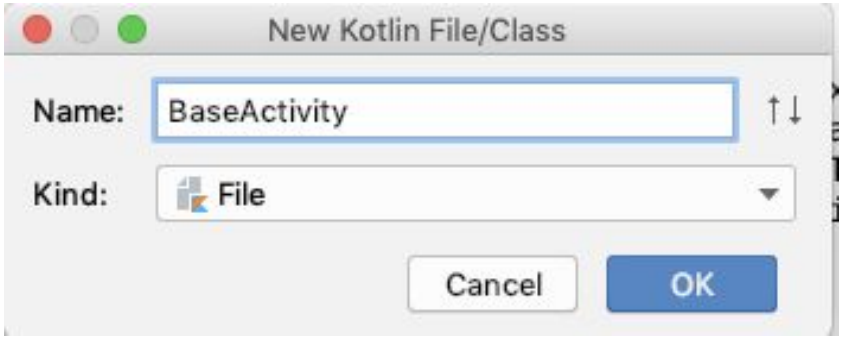
Firebase Analytics

```
import android.app.Activity
import android.os.Bundle
import com.google.firebase.analytics.FirebaseAnalytics

object CalculaFlexTracker {

    fun trackScreen(activity: Activity, screenName: String) {
        if (screenName != ScreenMap.SCREEN_NOT_TRACKING) {
            //Log.i("ANALYTICS", screenName)
            val mFirebaseAnalytics = FirebaseAnalytics.getInstance(activity)
            mFirebaseAnalytics.setCurrentScreen(activity, screenName, null)
        }
    }

    fun trackEvent(activity: Activity, bundle: Bundle) {
        val mFirebaseAnalytics = FirebaseAnalytics.getInstance(activity)
        mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);
    }
}
```

Firebase Analytics

Para facilitar o uso no projeto, iremos criar uma nova classe chamada **BaseActivity**





Firebase Analytics

```
import android.support.v7.app.AppCompatActivity
import br.com.heiderlopes.calculaflex.utils.CalculaFlexTracker
import br.com.heiderlopes.calculaflex.utils.ScreenMap

open class BaseActivity : AppCompatActivity() {

    open fun getScreenName(): String {
        return ScreenMap.getScreenNameBy(this)
    }

    override fun onStart() {
        super.onStart()
        CalculaFlexTracker.trackScreen(this, getScreenName())
    }
}
```

Firebase Analytics

Altere de **AppCompatActivity** para **BaseActivity** conforme exemplo abaixo:

```
class FormActivity : AppCompatActivity() {
}
```

Para:

```
class FormActivity : BaseActivity() {
}
```

Faça o mesmo procedimento para as demais activities do projeto.

Firebase Analytics

Para visualizar os eventos em modo de debug acesse o terminal e digite o seguinte comando

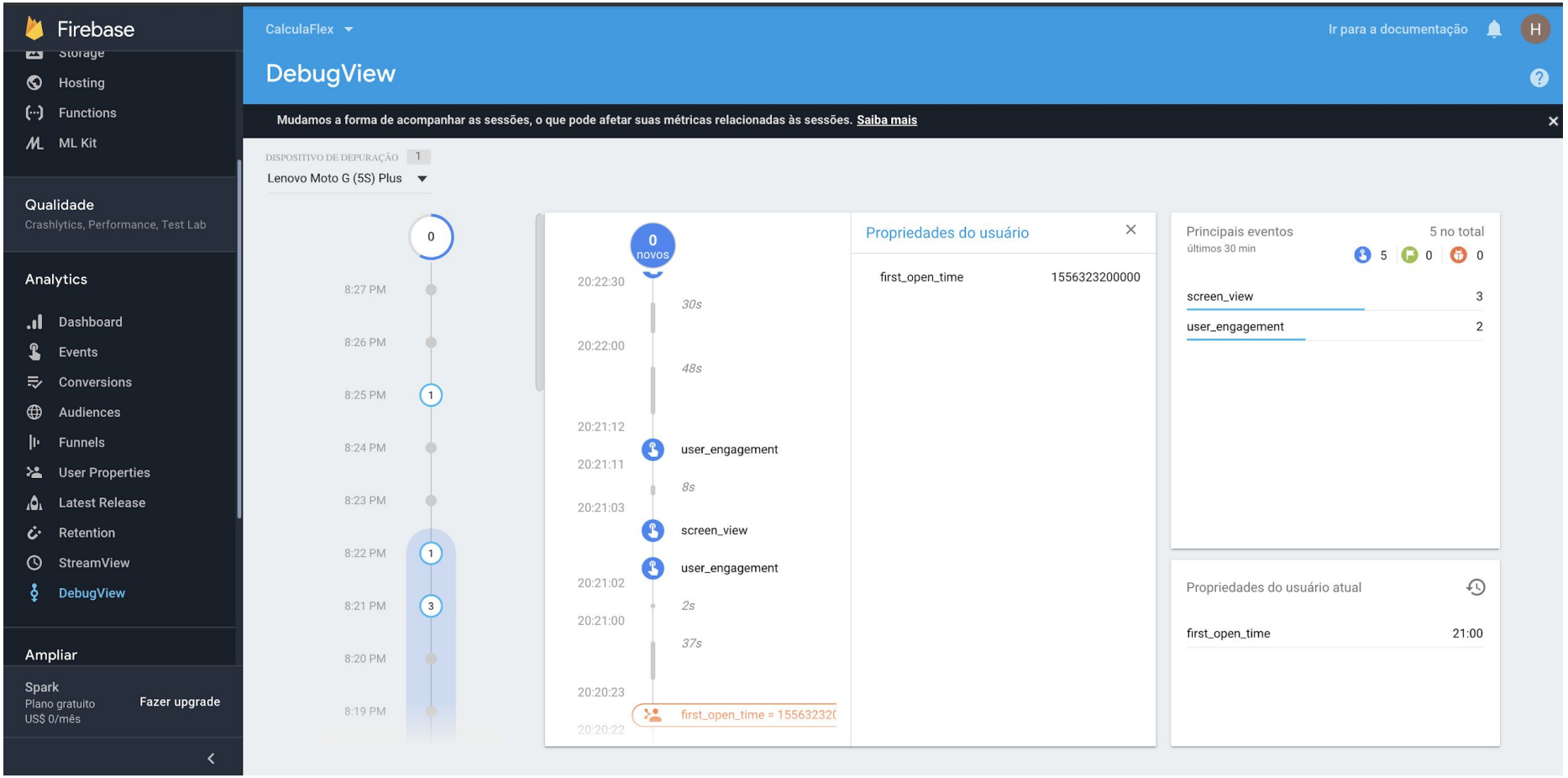
```
adb shell setprop debug.firebase.analytics.app [your_app_package_name]
```

Onde [your_app_package_name] é o pacote do seu projeto. Você pode ver essa informação no arquivo AndroidManifest.xml

Para parar o debug view digite a seguinte linha:

```
adb shell setprop debug.firebase.analytics.app .none.
```

Firebase Analytics



Firebase Analytics

Abra o arquivo **FormActivity.kt** e crie o seguinte método:

```
private fun sendDataToAnalytics() {
   val bundle = Bundle()
   bundle.putString("EVENT_NAME", "CALCULATION")
   bundle.putDouble("GAS_PRICE", etGasPrice.text.toString().toDouble());
   bundle.putDouble("ETHANOL_PRICE", etEthanolPrice.text.toString().toDouble());
   bundle.putDouble("GAS_AVERAGE", etGasAverage.text.toString().toDouble());
   bundle.putDouble("ETHANOL_AVERAGE", etEthanolAverage.text.toString().toDouble());

   CalculaFlexTracker.trackEvent(this, bundle)
}
```

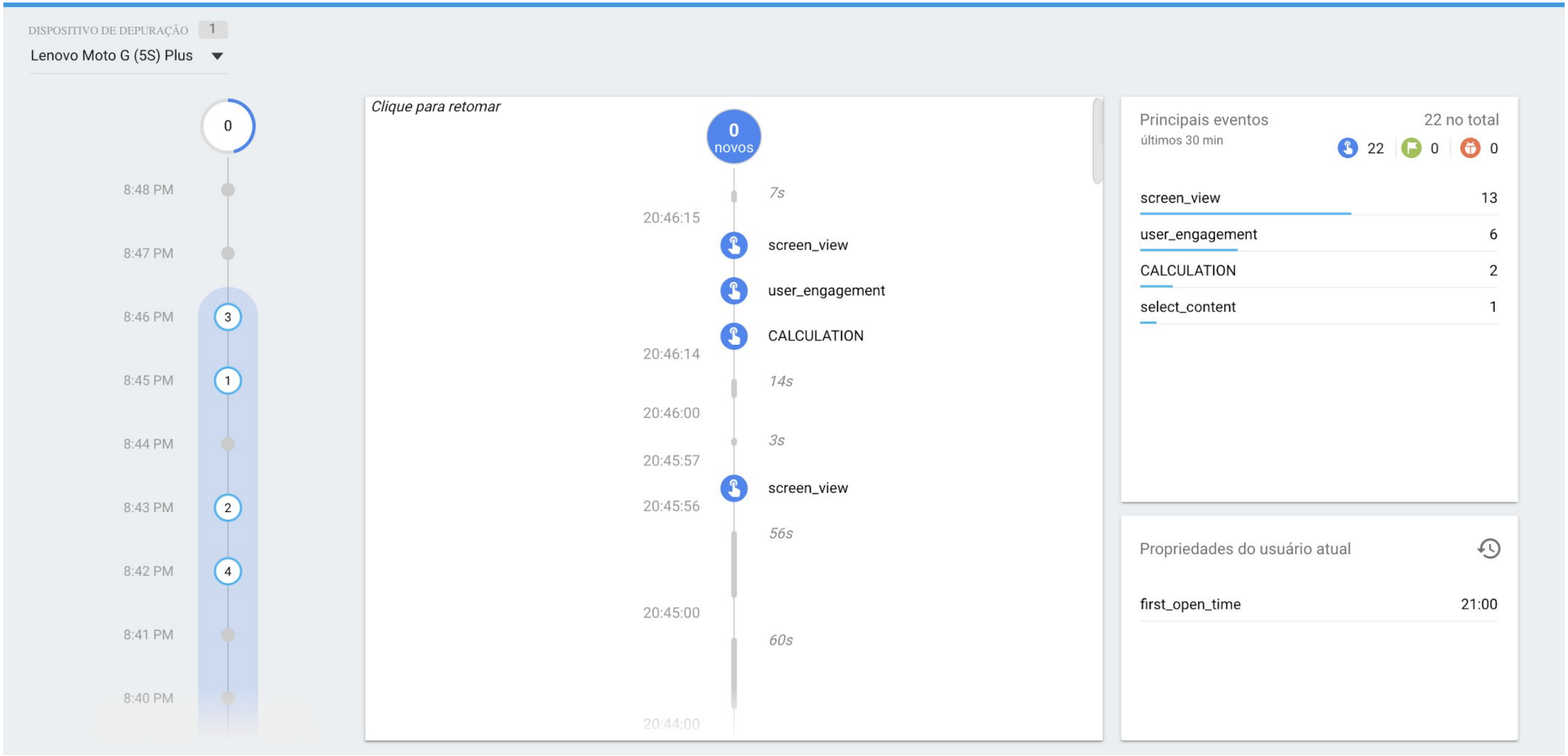
Firebase Analytics

Chame esse método dentro do listener do botão calcular (btCalculate)

```
btCalculate.setOnClickListener {
   saveCarData()
   val proximatela = Intent(this@FormActivity, ResultActivity::class.java)
   proximatela.putExtra("GAS_PRICE", etGasPrice.text.toString().toDouble())
   proximatela.putExtra("ETHANOL_PRICE", etEthanolPrice.text.toString().toDouble())
   proximatela.putExtra("GAS_AVERAGE", etGasAverage.text.toString().toDouble())
   proximatela.putExtra("ETHANOL_AVERAGE",
etEthanolAverage.text.toString().toDouble())

   sendDataToAnalytics()

   startActivity(proximatela)
}
```

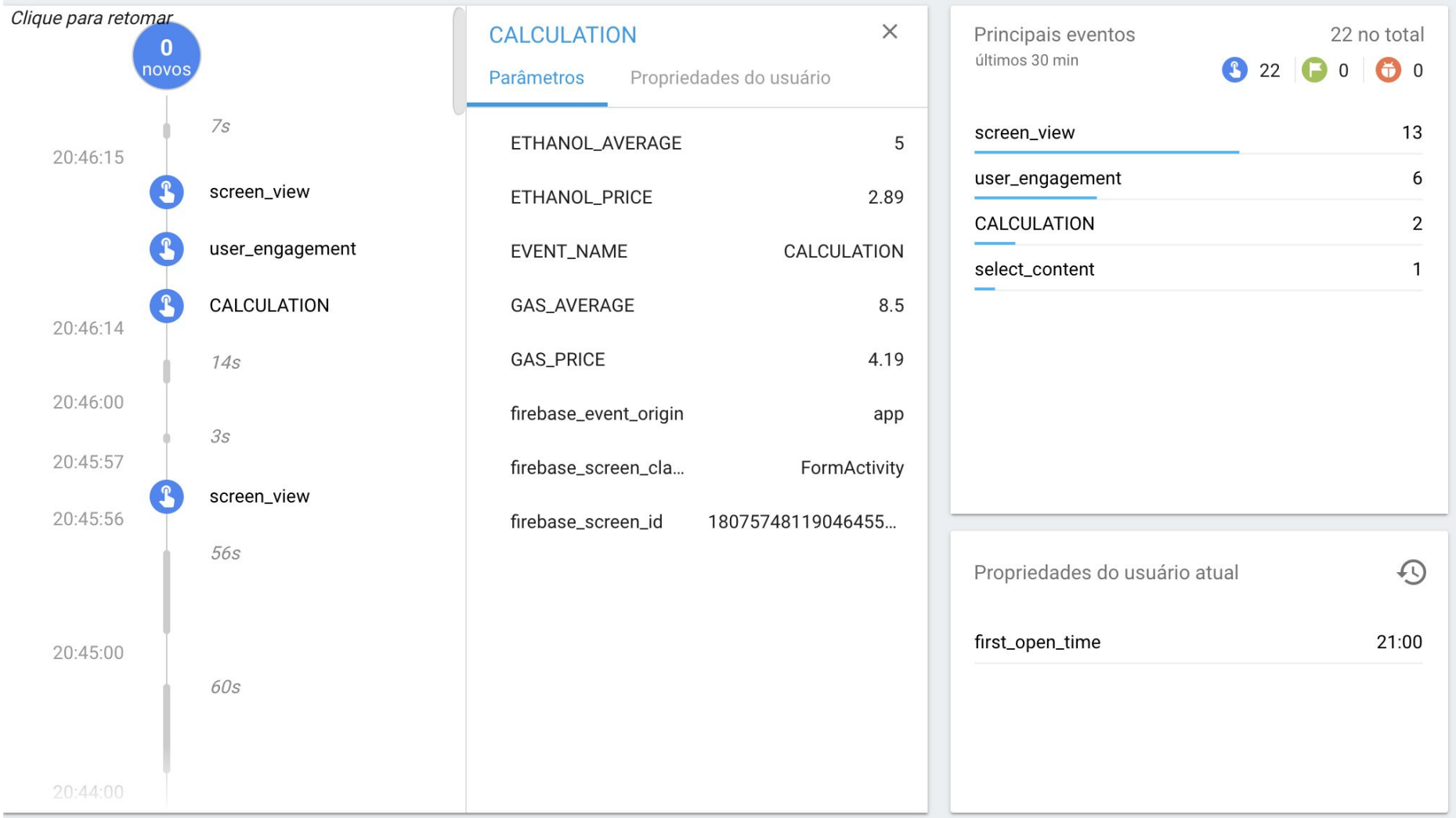
Firebase Analytics - DebugView

Os valores sendo enviados para o painel:

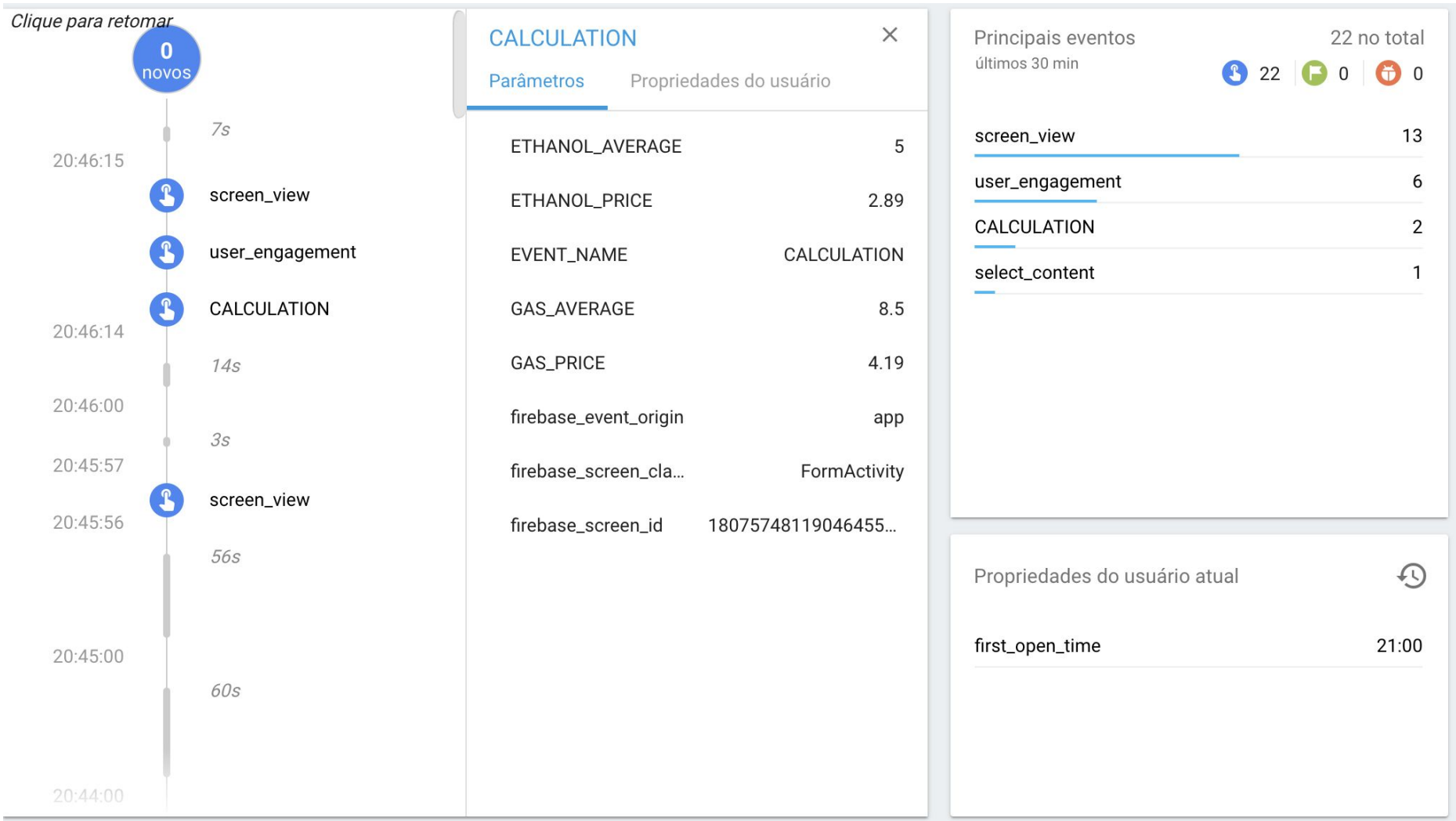


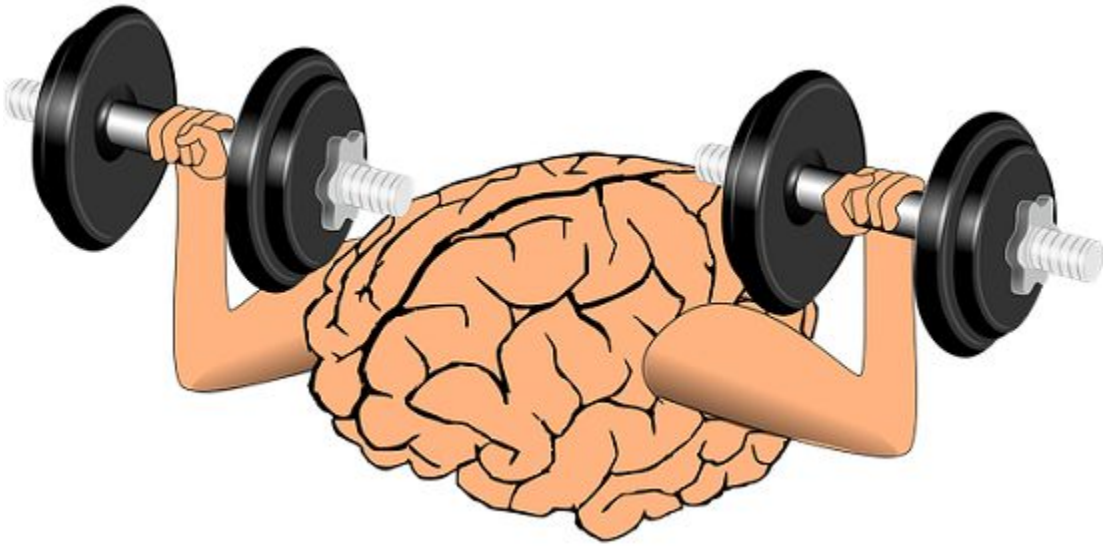
Firebase Analytics - DebugView

Os valores sendo enviados para o painel:



Firebase Analytics - DebugView





Exercitando a mente

Firebase Analytics - DebugView

Criar evento de **Analytics** para o **login**.

Criar uma nova configuração no **Remote Config** e utilizá-la no aplicativo.