

Firestore Cloud Messaging (FCM)



Firestore Cloud Messaging

- O Firestore Cloud Messaging (FCM) é o serviço de notificação do Firestore.
- Com ele é possível enviar notificações para os usuários do nosso app.
- É a forma oficial de enviar uma **Notificação Push** (é um tipo de mensagem enviada para aplicativos de celular ou aplicativos gerais) para o Android.
- No iOS a forma oficial é o APNs (Apple Push Notification Service).
- É possível enviar Notificação Push para iOS usando o Firestore Cloud Messaging, mas apenas para um dispositivo real, ou seja, não funciona no simulador.



Instalação do Plugin

- Para trabalhar com o FCM precisaremos instalar o plugin **firebase_messaging**.
- A configuração básica para instalação do plugin se encontra em : https://pub.dev/packages/firebase_messaging#-installing-tab-
- Entretanto é necessário fazer algumas configurações extras tanto no Android quanto no iOS (infelizmente não pude testar esse serviço no iOS).
- Essas configurações extras se encontram na aba Readme: https://pub.dev/packages/firebase_messaging#-readme-tab-



Configuração no Android

- Verifique se as dependências foram acrescentadas nos arquivos **build.gradle** conforme descrito na aba **Readme** (nos meus projetos Flutter essas linhas eram automaticamente incluídas nos arquivos build.gradle). Ou seja, é muito provável que não precisemos colocá-las pois já estarão lá (se não tiverem, acrescente-as).
- Após essas configurações iniciais a aba **Readme** sugere a colocação da seguinte configuração no AndroidManifest.xml (nós precisaremos dela para capturar a notificação com o app aberto):

`<intent-filter>`

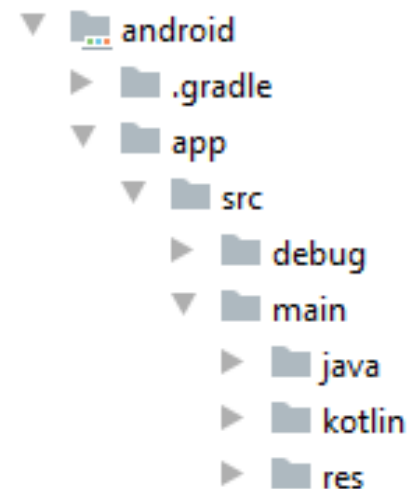
`<action android:name="FLUTTER_NOTIFICATION_CLICK" />`

`<category android:name="android.intent.category.DEFAULT" />`

`</intent-filter>`



Configuração no Android



```
<activity
  android:name=".MainActivity"
  android:launchMode="singleTop"
  android:theme="@style/LaunchTheme"
  android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreenSize|
  android:hardwareAccelerated="true"
  android:windowSoftInputMode="adjustResize">
  <intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>
  <intent-filter>
    <action android:name="FLUTTER_NOTIFICATION_CLICK" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```



Flutter 2.0

Algumas configurações nos arquivos build.gradle são necessárias. Basta inserir os códigos apresentados em:

<https://firebase.google.com/docs/cloud-messaging/android/client>

```
<activity
  android:name=".MainActivity"
  android:launchMode="singleTop"
  android:theme="@style/LaunchTheme"
  android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreenSize|
  android:hardwareAccelerated="true"
  android:windowSoftInputMode="adjustResize">
  <intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>
  <intent-filter>
    <action android:name="FLUTTER_NOTIFICATION_CLICK" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```

Não é mais necessário



TratadorNotificacao

```
import 'package:firebase_messaging/firebase_messaging.dart';

class TratadorNotificacao{
  static BuildContext _context;
  static FirebaseMessaging _fcm;

  static void inicializarFCM(BuildContext context) {
    _context = context;

    if (_fcm == null) {
      _fcm = FirebaseMessaging();
    }
  }
}
```

- Basicamente fazemos a importação do pacote necessário e inicializamos o objeto de FirebaseMessaging.



TratadorNotificacao

```
_fcm.configure(  
  // Quando a aplicação está ativa  
  onMessage: (Map<String, dynamic> message) async {  
    // 'mensagem' deve ser chave um Dado Personalizado do item  
    // Outras opções (opcional)  
    // Também deve ser colocado o dado personalizado com chave  
    // 'click_action' com o valor 'FLUTTER_NOTIFICATION_CLICK'  
    _showDialog(message['data']['mensagem']);  
  },  
  // Quando a aplicação está voltando ao foco  
  onResume: (Map<String, dynamic> message) async {  
    // _showDialog(message['data']['mensagem']);  
  },  
  // Quando a aplicação está sendo recarregada  
  // (havia saído pelo botão voltar)  
  onLaunch: (Map<String, dynamic> message) async {  
    // _showDialog(message['data']['mensagem']);  
  },  
);
```

- Aqui temos o método que configura os 3 possíveis eventos.
- **onMessage**: ocorre quando a aplicação está ativa e a mensagem chega.
- **onResume**: ocorre quando a aplicação está voltando ao foco e a mensagem chega.
- **onLaunch**: ocorre quando a aplicação está sendo recarregada e mensagem chega.
- E quando a mensagem chega e a aplicação está fechada?

Ocorrerá uma notificação na barra de notificações.



Flutter 2.0

```
_fcm.configure(  
  // Quando a aplicação está ativa  
  onMessage: (Map<String, dynamic> message) async {  
  
    // 'mensagem' deve ser chave um Dado Personalizado do item  
    // Outras opções (opcional)  
    // Também deve ser colocado o dado personalizado com chave  
    // 'click_action' com o valor 'FLUTTER_NOTIFICATION_CLICK'  
    _showDialog(message['data']['mensagem']);  
  },  
  // Quando a aplicação está voltando ao foco  
  onResume: (Map<String, dynamic> message) async {  
    // _showDialog(message['data']['mensagem']);  
  },  
  // Quando a aplicação está sendo recarregada  
  // (havia saído pelo botão voltar)  
  onLaunch: (Map<String, dynamic> message) async {  
    // _showDialog(message['data']['mensagem']);  
  },  
);
```

```
class TratadorNotificacao{  
  static BuildContext? _context;  
  static late FirebaseMessaging _fcm;  
  
  static Future<void> inicializarFCM(BuildContext? context) async {  
    _context = context;  
  
    _fcm = FirebaseMessaging.instance;  
  
    _fcm.requestPermission(alert: true, badge: true, sound: true);  
  
    // Quando a aplicação está ativa  
    FirebaseMessaging.onMessage.listen((RemoteMessage message) {  
      print("onMessage chamado");  
      _showDialog(message.data['mensagem']);  
    });  
  
    // Quando a aplicação está voltando ao foco ou está sendo recarregada  
    FirebaseMessaging.onMessageOpenedApp.listen((RemoteMessage message) {  
      print("onMessageOpenedApp chamado");  
    });  
  }  
}
```

- On iOS, a dialog is shown requesting the users permission.
- On macOS, a notification will appear asking to grant permission.
- On Android, is it not required to call this method. If called however, a [NotificationSettings] class will be returned with [NotificationSettings.authorizationStatus] returning [AuthorizationStatus.authorized].
- On Web, a popup requesting the users permission is shown using the native browser API.

<https://firebase.flutter.dev/docs/messaging/usage/>

Solicitando permissões de uso ao usuário (para Android não é necessário)



TratadorNotificacao

- Ao lado vemos o método chamado para exibir um alerta com a mensagem enviada via FCM.

```
static void _showDialog(String mensagem) {  
  showDialog(  
    context: _context,  
    builder: (BuildContext context) {  
      // retorna um objeto do tipo Dialog  
      return AlertDialog(  
        title: new Text("Notificação"),  
        content:  
        new Text(mensagem),  
        actions: <Widget>[  
          // define os botões na base do dialogo  
          FlatButton(  
            child: new Text("Fechar"),  
            onPressed: () async {  
              pop(context);  
            },  
          ), // FlatButton  
        ], // <Widget>[]  
      ); // AlertDialog  
    },  
  );  
}
```



Flutter 2.0

```
static void _showDialog(String mensagem) {  
  showDialog(  
    context: _context,  
    builder: (BuildContext context) {  
      // retorna um objeto do tipo Dialog  
      return AlertDialog(  
        title: new Text("Notificação"),  
        content:  
        new Text(mensagem),  
        actions: <Widget>[  
          // define os botões na base do dialogo  
          FlatButton(  
            child: new Text("Fechar"),  
            onPressed: () async {  
              pop(context);  
            },  
          ), // FlatButton  
        ], // <Widget>[]  
      ); // AlertDialog  
    },  
  );  
}
```

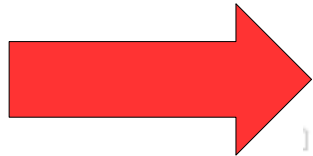
```
static void _showDialog(String mensagem) {  
  showDialog(  
    context: _context!,  
    builder: (BuildContext context) {  
      // retorna um objeto do tipo Dialog  
      return AlertDialog(  
        title: const Text("Notificação"),  
        content: Text(mensagem),  
        actions: <Widget>[  
          // define os botões na base do dialogo  
          TextButton(  
            child: const Text("Fechar"),  
            onPressed: () async {  
              pop(context);  
            },  
          ), // TextButton  
        ], // <Widget>[]  
      ); // AlertDialog  
    },  
  );  
}
```



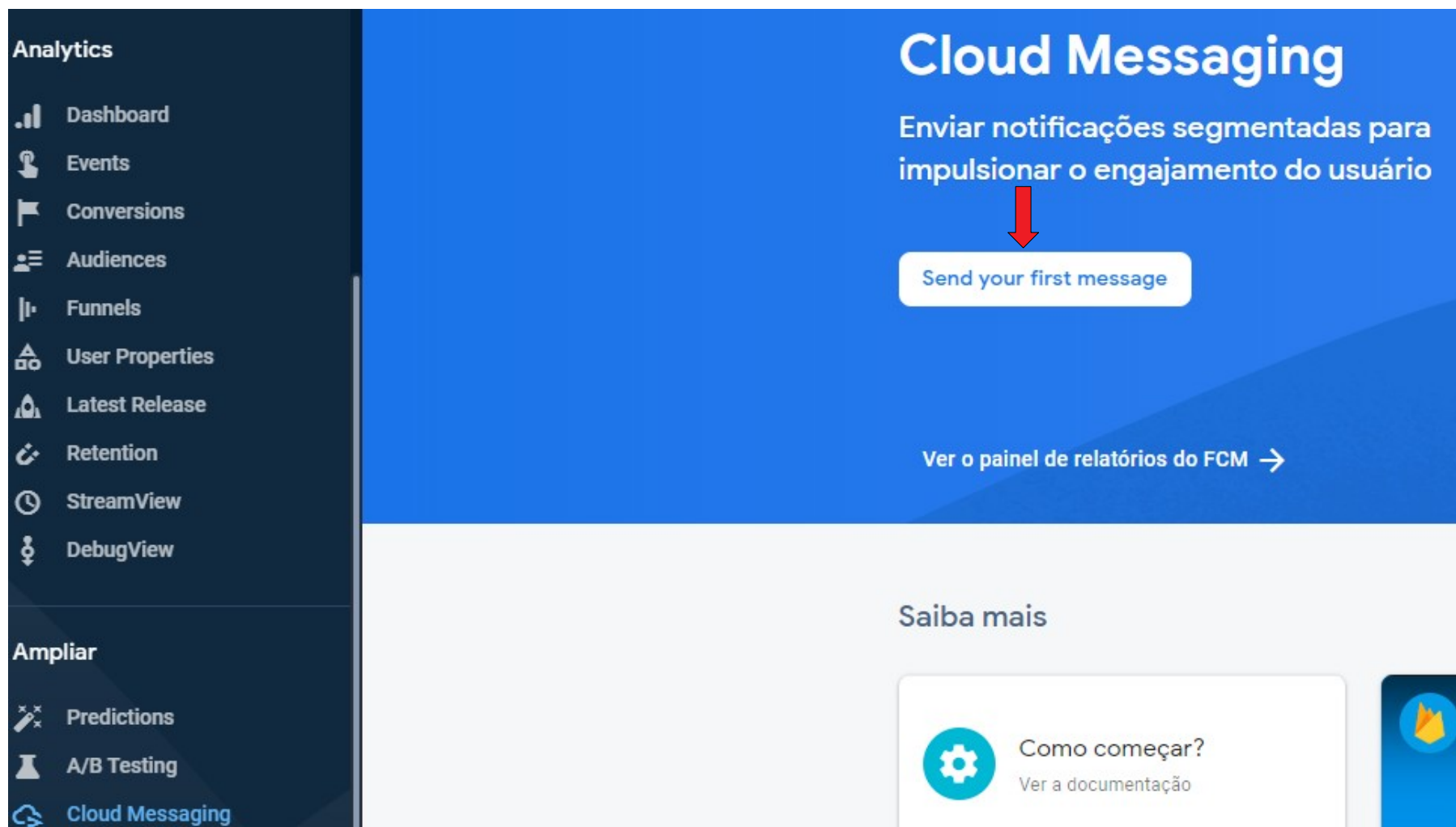
Usando o TratadorNotificacao

- O **TratadorNotificacao** pode ser acionado em qualquer lugar. No Flls Plan ele é inicializado na **TelaPrincipal** da aplicação, ou seja, a partir desse momento ele estará ativo.

```
class _TelaPrincipalState extends State<TelaPrincipal> {  
  ControleTelaPrincipal _controle;  
  
  @override  
  void initState() {  
    // TODO: implement initState  
    super.initState();  
    _controle = ControleTelaPrincipal(widget.usuario);  
    _controle.buscarPatrimonios();  
  
    TratadorNotificacao.inicializarFCM(context);  
  }  
}
```

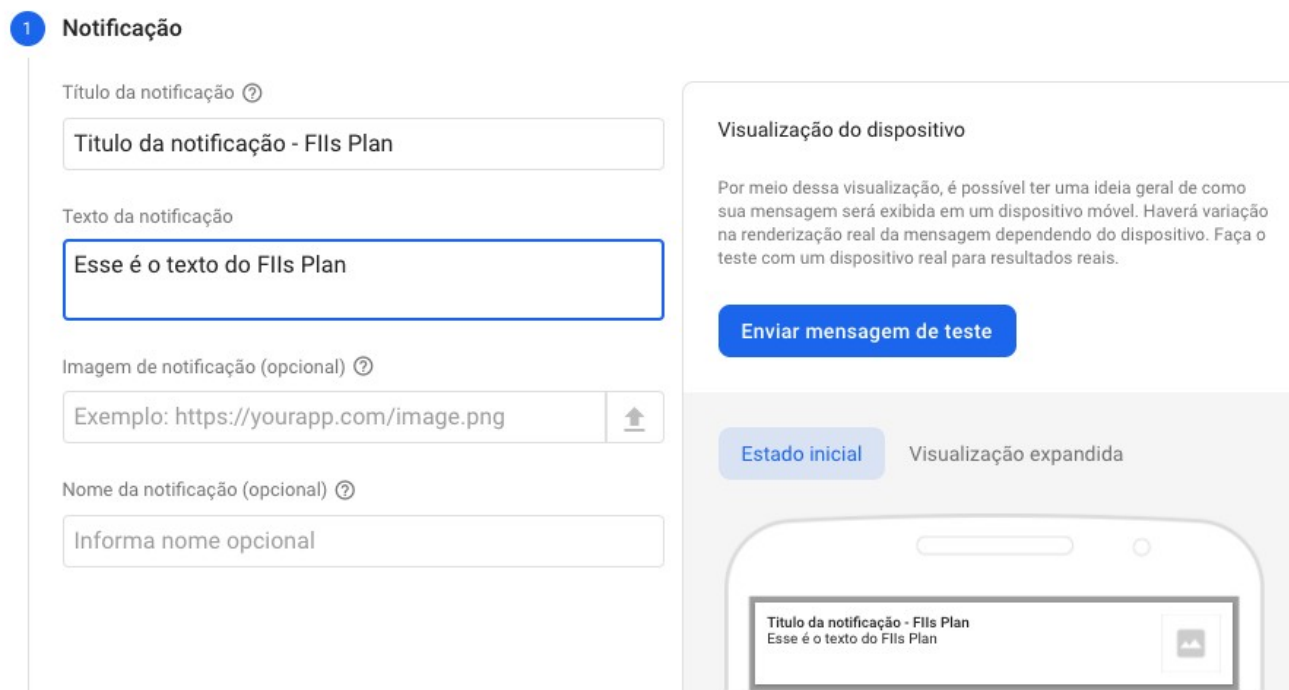


Console do Firebase



Console do Firebase

- Começamos informando o título e texto da mensagem (note que é possível acrescentar um link para uma imagem que apareceria na notificação):



1 Notificação


Título da notificação ⓘ

Titulo da notificação - Filis Plan

Texto da notificação

Esse é o texto do Filis Plan

Imagem de notificação (opcional) ⓘ

Exemplo: <https://yourapp.com/image.png> 

Nome da notificação (opcional) ⓘ


Informa nome opcional

Enviar mensagem de teste

Visualização do dispositivo

Por meio dessa visualização, é possível ter uma ideia geral de como sua mensagem será exibida em um dispositivo móvel. Haverá variação na renderização real da mensagem dependendo do dispositivo. Faça o teste com um dispositivo real para resultados reais.

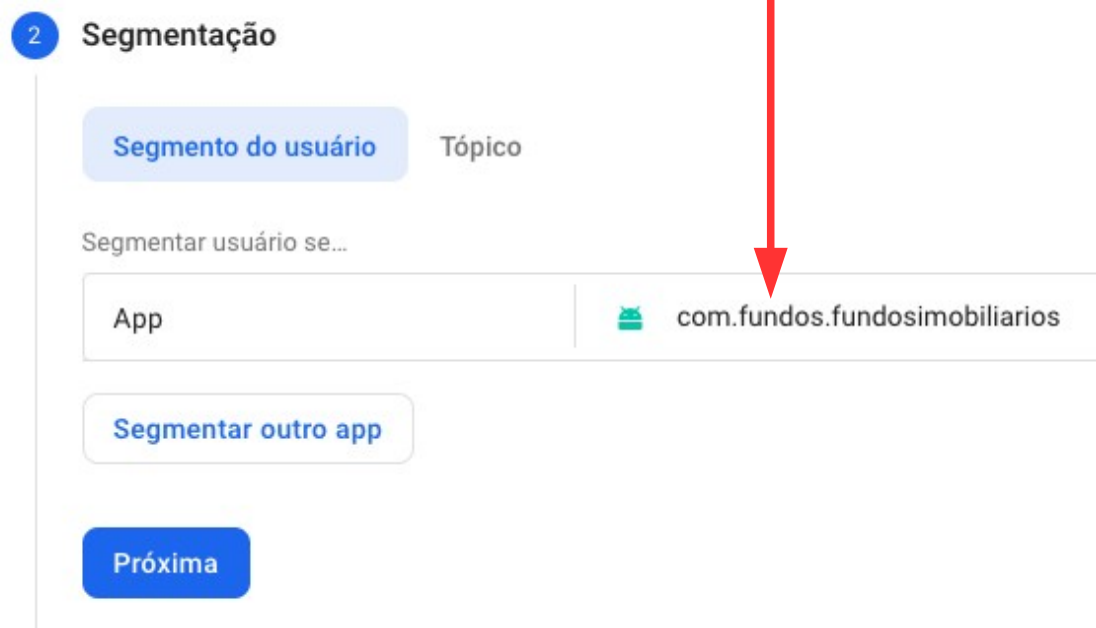
Estado inicial Visualização expandida

Titulo da notificação - Filis Plan
Esse é o texto do Filis Plan 



Console do Firebase


- Vamos agora segmentar a quem enviar. Nesse caso, estamos enviando para quem tiver esse aplicativo na versão Android:



2 Segmentação

Segmento do usuário Tópico

Segmentar usuário se...

App	
	 com.fundos.fundosimobiliarios

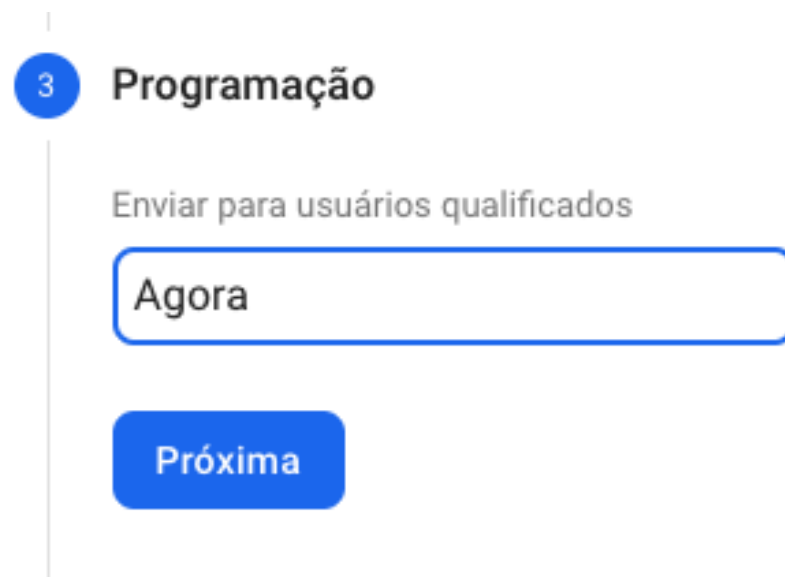
Segmentar outro app

Próxima



Console do Firebase

- Podemos dizer quando enviar:



The screenshot shows the 'Programação' (Scheduling) step in the Firebase console. It features a vertical progress bar on the left with a blue circle containing the number '3'. To the right of the bar, the text 'Programação' is displayed. Below this, the instruction 'Enviar para usuários qualificados' (Send to qualified users) is shown. A text input field with a blue border contains the word 'Agora' (Now). At the bottom of the form is a blue button with the text 'Próxima' (Next).

- No passo 4 apenas avance (Próxima).



Console do Firebase

- Aqui temos o uso do `FLUTTER_NOTIFICATION_CLICK` configurado anteriormente. Essa configuração é fundamental para conseguirmos capturar **mensagem** quando a aplicação estiver aberta.

5 Outras opções (opcional)

Todos os campos são opcionais

Canal de notificações do Android ?

Dados personalizados ?

<input type="text" value="mensagem"/>	<input type="text" value="Esse é o conteúdo da minha mensagem"/>
<input type="text" value="click_action"/>	<input type="text" value="FLUTTER_NOTIFICATION_CLICK"/>
<input type="text" value="Chave"/>	<input type="text" value="Valor"/>



Notificação com o app fechado

- Com o app fechado a notificação aparece na barra de notificações.
- Basicamente ele mostrará o título e a texto da mensagem.
- Por padrão o ícone da notificação é o ícone do aplicativo.

1 Notificação

Título da notificação ⓘ

Título da notificação - Fils Plan

Texto da notificação

Esse é o texto do Fils Plan

Imagem de notificação (opcional) ⓘ

Exemplo: <https://yourapp.com/image.png>

Nome da notificação (opcional) ⓘ

Informa nome opcional

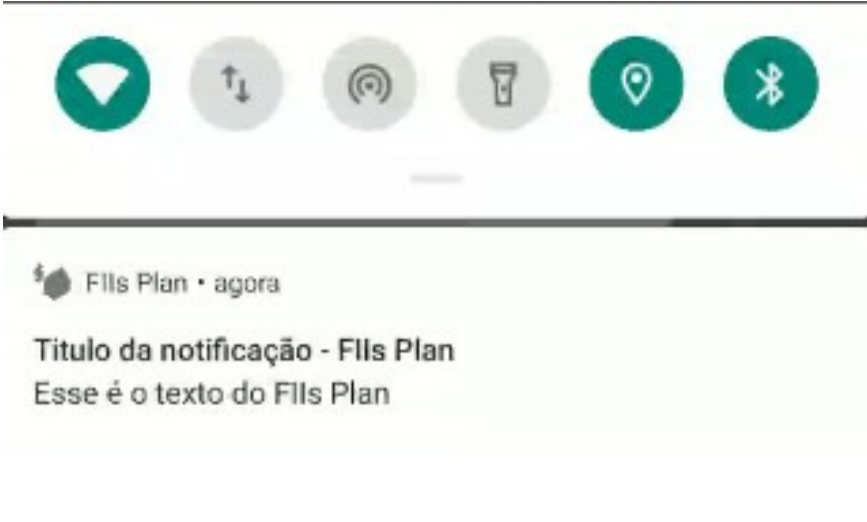
Visualização do dispositivo

Por meio dessa visualização, é possível ter uma ideia geral de como sua mensagem será exibida em um dispositivo móvel. Haverá variação na renderização real da mensagem dependendo do dispositivo. Faça o teste com um dispositivo real para resultados reais.

Enviar mensagem de teste

Estado inicial Visualização expandida

Título da notificação - Fils Plan
Esse é o texto do Fils Plan



The image shows a notification bar on a mobile device. At the top, there is a row of six circular icons: a green Wi-Fi icon, a grey cellular signal icon, a grey Bluetooth icon, a grey location pin icon, a green location pin icon, and a green Bluetooth icon. Below the icons, the notification bar is displayed. It starts with the app icon and name 'Fils Plan • agora'. Below that, the title 'Título da notificação - Fils Plan' and the text 'Esse é o texto do Fils Plan' are shown.



Notificação com o app aberto

- Por padrão, com o app aberto nada aconteceria. Ou seja, o usuário do app não iria saber que chegou uma notificação.
- Entretanto, como foi implementado comportamento no **onMessage**, esse comportamento será acionado.

```
_fcm.configure(  
  // Quando a aplicação está ativa  
  onMessage: (Map<String, dynamic> message) async {  
  
    // 'mensagem' deve ser chave um Dado Personalizado do item  
    // Outras opções (opcional)  
    // Também deve ser colocado o dado personalizado com chave  
    // 'click_action' com o valor 'FLUTTER_NOTIFICATION_CLICK'  
    _showDialog(message['data']['mensagem']);  
  },  
  // Quando a aplicação está voltando ao foco  
  onResume: (Map<String, dynamic> message) async {  
    // _showDialog(message['data']['mensagem']);  
  },  
  // Quando a aplicação está sendo recarregada  
  // (havia saído pelo botão voltar)  
  onLaunch: (Map<String, dynamic> message) async {  
    // _showDialog(message['data']['mensagem']);  
  },  
);
```

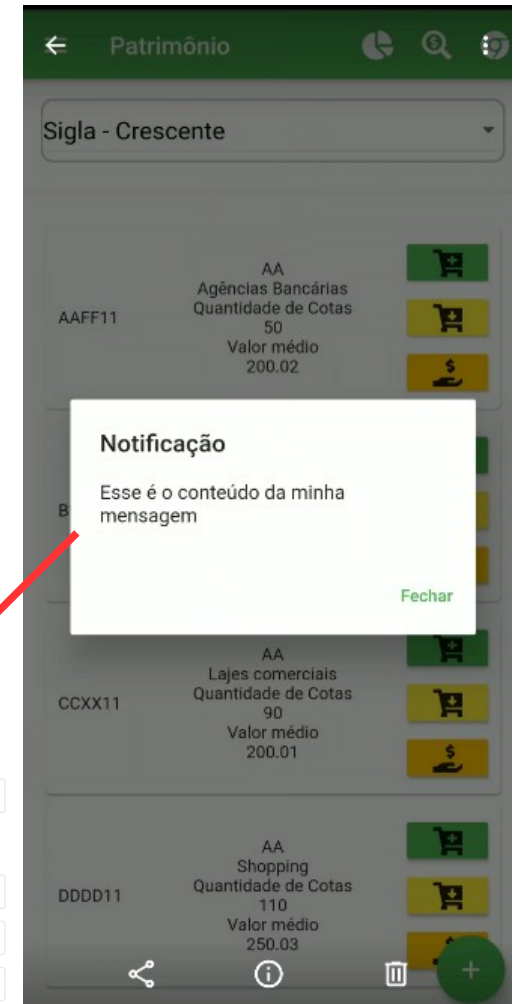
6 Outras opções (opcional)

Todos os campos são opcionais

Canal de notificações do Android ⓘ

Dados personalizados ⓘ

mensagem	Esse é o conteúdo da minha mensagem
click_action	FLUTTER_NOTIFICATION_CLICK
Chave	Valor



Dúvidas?

