Google Cloud Messaging

Nesta aula, apresentaremos o Google Cloud Messaging (GCM), uma solução do Google para envio de mensagens push para dispositivos Android, iOS e web. Discutiremos os benefícios do uso de GCM, como ele funciona e os conceitos básicos que você precisa entender para integrá-lo em seus aplicativos Flutter.

Introdução ao Google Cloud Messaging

O que é Google Cloud Messaging?

Google Cloud Messaging (GCM) é um serviço que permite enviar dados de até 4 KB para dispositivos clientes através de servidores de terceiros. Este serviço é útil para enviar notificações push, alertas, mensagens de bate-papo e outras atualizações.

Como Funciona o GCM?

1. Servidor GCM: O servidor GCM gerencia a entrega de mensagens para dispositivos clientes.

2. Aplicativo Servidor: Seu servidor envia mensagens para o servidor GCM, que então as entrega aos dispositivos.

3. Aplicativo Cliente: O aplicativo no dispositivo recebe e processa as mensagens.

Benefícios do GCM

- Entrega Fiável: GCM garante a entrega de mensagens mesmo quando o aplicativo está em segundo plano.

- Escalabilidade: Pode enviar mensagens para milhões de dispositivos.

- Eficiência de Recursos: Minimiza o uso de bateria e dados.

Nesta aula, vamos configurar o GCM no seu aplicativo Flutter. Abordaremos os passos necessários para registrar seu aplicativo no Firebase (substituto do GCM), integrar o SDK do Firebase no seu projeto Flutter, e configurar os arquivos de configuração necessários. Ao final desta aula, seu aplicativo estará pronto para receber mensagens push.

Configurando o GCM no seu aplicativo

Passos para Configurar GCM

1. Registrar seu Projeto no Firebase

- Acesse o Firebase Console.

- Crie um novo projeto ou use um existente.

- Adicione seu aplicativo Android e iOS ao projeto.

2. Adicionar Dependências ao pubspec.yaml

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: latest_version
  firebase_messaging: latest_version
```

3. Configurar Android

- Adicione o arquivo google-services.json ao diretório android/app.

- Atualize android/build.gradle:

```
buildscript {
  dependencies {
    classpath ‘com.google.gms:google-services:4.3.3’
  }
}
```

- Atualize android/app/build.gradle:

```
apply plugin: ‘com.google.gms.google-services’
```

4. Configurar iOS

- Adicione o arquivo GoogleService-Info.plist ao diretório ios/Runner.

- Abra o arquivo ios/Runner/Info.plist e adicione:

```
<key>FirebaseAppDelegateProxyEnabled</key>
<false/>
```

5. Inicializar Firebase no Flutter

```
import ‘package:flutter/material.dart’;
import ‘package:firebase_core/firebase_core.dart’;
import ‘package:firebase_messaging/firebase_messaging.dart’;
```

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomeScreen(),
    );
  }
}
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}
class _HomeScreenState extends State<HomeScreen> {
  @override
```

```
  void initState() {
    super.initState();
    FirebaseMessaging.instance.getToken().then((token) {
      print("FCM Token: $token");
    });
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Configuração GCM'),
      ),
      body: Center(
        child: Text('Firebase Messaging Configurado!'),
      ),
    );
  }
}
```

Explicação: Este exemplo mostra como inicializar o Firebase no Flutter e obter um token FCM, que é necessário para receber mensagens push.

Nesta aula, abordaremos como enviar mensagens push para o seu aplicativo Flutter usando o Firebase Cloud Messaging (FCM). Veremos como configurar o servidor para enviar mensagens, como enviar notificações através do Firebase Console e como usar as APIs do FCM para enviar mensagens programaticamente.

Enviando Mensagens Push

Enviando Mensagens do Firebase Console

1. Acessar o Firebase Console

- Navegue até o projeto no Firebase Console.

- Selecione Cloud Messaging no menu à esquerda.

2. Criar e Enviar Mensagem

- Clique em “Send your first message”.

- Preencha o título e o corpo da mensagem.

- Selecione o aplicativo alvo.

- Clique em “Next” e “Review” e depois “Send”.

Enviando Mensagens Programaticamente

1. Configurar Servidor para Enviar Mensagens

- Use a biblioteca firebase-admin para enviar mensagens.

- Exemplo em Node.js:

```
const admin = require('firebase-admin');

admin.initializeApp({
  credential: admin.credential.cert('path/to/serviceAccountKey.json'),
});

const message = {
  notification: {
    title: 'Hello!',
    body: 'This is a push notification',
  },
  token: 'user_device_token',
};

admin.messaging().send(message)
  .then((response) => {
    console.log('Successfully sent message:', response);
  })
  .catch((error) => {
    console.log('Error sending message:', error);
  });
```

2. Exemplo de Código Completo em Dart

```
import ‘package:flutter/material.dart’;

import ‘package:firebase_core/firebase_core.dart’;

import ‘package:firebase_messaging/firebase_messaging.dart’;

void main() async {

  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp();

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: HomeScreen(),

    );

  }

}

class HomeScreen extends StatefulWidget {

  @override

  _HomeScreenState createState() => _HomeScreenState();
```

```
}

class _HomeScreenState extends State<HomeScreen> {

 @override

 void initState() {

  super.initState();

  FirebaseMessaging.instance.getToken().then((token) {

   print("FCM Token: $token");

  });

  FirebaseMessaging.onMessage.listen((RemoteMessage message) {

   print('Message data: ${message.data}');

   if (message.notification != null) {

    print('Message also contained a notification: ${message.notification}');

   }

  });

}

@override

Widget build(BuildContext context) {

 return Scaffold(

  appBar: AppBar(
```

```
        title: Text('Enviando Mensagens Push'),
      ),
      body: Center(
        child: Text('Aguardando Mensagens Push...'),
      ),
    );
  }
}
```

Explicação: Este código inicializa o Firebase, obtém o token FCM do dispositivo e escuta mensagens recebidas enquanto o aplicativo está em primeiro plano.

Nesta aula, aprenderemos como configurar o aplicativo para receber e responder a mensagens push enviadas pelo FCM. Discutiremos como lidar com mensagens enquanto o aplicativo está em primeiro plano, em segundo plano e fechado. Também veremos como personalizar a exibição de notificações e ações associadas a elas.

Recebendo e Respondendo a Mensagens Push

Lidando com Mensagens em Diferentes Estados do Aplicativo

1. Mensagens em Primeiro Plano

- Utilize FirebaseMessaging.onMessage para lidar com mensagens recebidas enquanto o aplicativo está aberto.

```
FirebaseMessaging.onMessage.listen((RemoteMessage message) {
```

```
  print('Received message in foreground: ${message.messageId}');

});
```

2. Mensagens em Segundo Plano e Fechado

- Utilize FirebaseMessaging.onMessageOpenedApp para lidar com mensagens que abriram o aplicativo a partir de um estado fechado ou em segundo plano.

```
FirebaseMessaging.onMessageOpenedApp.listen((RemoteMessage
message) {

  print('Message opened the app: ${message.messageId}');

});
```

3. Configurar Notificações na Inicialização do App

- Utilize FirebaseMessaging.instance.getInitialMessage() para lidar com mensagens que abriram o aplicativo quando estava fechado.

```
FirebaseMessaging.instance.getInitialMessage().then((RemoteMessage?
message) {

  if (message != null) {

    print('Message caused app to open: ${message.messageId}');

  }

});
```

Personalizando Notificações

1. Exibir Notificações Personalizadas

- Utilize pacotes como flutter_local_notifications para personalizar a exibição de notificações.

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

final FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin =
    FlutterLocalNotificationsPlugin();

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  const AndroidInitializationSettings initializationSettingsAndroid =
      AndroidInitializationSettings('@mipmap/ic_launcher');
  final InitializationSettings initializationSettings = InitializationSettings(
    android: initializationSettingsAndroid,
  );
  await flutterLocalNotificationsPlugin.initialize(initializationSettings);
  FirebaseMessaging.onMessage.listen((RemoteMessage message) {
    RemoteNotification? notification = message.notification;
    AndroidNotification? android = message.notification?.android;
    if (notification != null && android != null) {
      flutterLocalNotificationsPlugin.show(
```

```
        notification.hashCode,
        notification.title,
        notification.body,
        NotificationDetails(
          android: AndroidNotificationDetails(
            'your_channel_id',
            'your_channel_name',
            'your_channel_description',
            importance: Importance.max,
            priority: Priority.high,
            showWhen: false,
          ),
        ),
      );
    }
  });
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomeScreen(),
    );
  }
}
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}
class _HomeScreenState extends State<HomeScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Recebendo Mensagens Push'),
      ),
      body: Center(
```

```
        child: Text('Aguardando Mensagens Push...'),
      ),
    );
  }
}
```

Explicação: Este exemplo utiliza flutter_local_notifications para exibir notificações personalizadas quando o aplicativo recebe mensagens enquanto está em primeiro plano.

Esses exemplos e explicações detalhadas fornecem uma base sólida para iniciantes em Flutter aprenderem a implementar o Google Cloud Messaging em seus aplicativos. Para mais detalhes, consulte a documentação oficial do Flutter: Flutter Documentation.

Materiais Extras

Você pode realizar o download do arquivo contendo os materiais extras utilizados ao longo das aulas por meio do seguinte link: https://drive.google.com/file/d/1mg7lqMl8Pt2zl0rHIsFS0Qew00YN-sEX/view?usp=sharing.

Conteúdo Bônus

Para aprofundar seus conhecimentos em Google Cloud Messaging (GCM) e na migração para o Firebase Cloud Messaging (FCM), recomendamos o seguinte material:

Nome da obra: “Migrando aplicativos de GCM para FCM usando a API REST”

Autor: Microsoft Docs

Plataforma: Microsoft Learn

Este conteúdo gratuito oferece orientações detalhadas sobre como migrar seus aplicativos de GCM para FCM utilizando a API REST, além de integrar com o Azure Notification Hubs para melhorar o envio de notificações push em seus projetos.

Referências Bibliográficas

BOYLESTAD, R. L.; NASHELSKY, L. Dispositivos Eletrônicos e Teoria de Circuitos. 11. ed. Pearson, 2013.

DEITEL, P. J.; DEITEL, H. M. Ajax, Rich Internet Applications e Desenvolvimento Web para Programadores. Pearson, 2008.

DUARTE, W. Delphi para Android e iOS: Desenvolvendo Aplicativos Móveis. Brasport, 2015.

FELIX, R.; SILVA, E. L. da. Arquitetura para Computação Móvel. 2. ed. Pearson, 2019.

LEE, V.; SCHNEIDER, H.; SCHELL, R. Aplicações Móveis: Arquitetura, Projeto e Desenvolvimento. Pearson, 2005.

MARINHO, A. L.; CRUZ, J. L. da. Desenvolvimento de Aplicações para Internet. 2. ed. Pearson, 2019.

MOLETTA, A. Você na Tela: Criação Audiovisual para a Internet. Summus, 2019.

SILVA, D. (Org.) Desenvolvimento para dispositivos móveis. Pearson, 2017.

Ir para exercício