

# How to handle all kind of notifications?

**What are the different types of notifications?**



# The different notification providers

- Firebase Cloud Messaging
- APNS
- OneSignal
- Airship
- ...

A lot of them are based on Firebase Cloud Messaging directly, often providing additional features

# Differences between device notifications and server notifications?

## Device notifications

- Scheduled from the user
- Doesn't need tracking from the server
- Triggered depending on user's location

## Server notifications

- Notifications that depend on another person's action
- Notifications that depend on a specific time or event



# Examples

# Setup notification your Flutter App using FCM

- Each platform needs a specific setup
- You will need platform configuration files `google-services.json` and `GoogleInfo.plist`
- You should confirm each platform as you go when building notifications



# Setup notification your Flutter App using FCM

## Requesting permissions

- Every platform now requires permission to display a notification
- **Good practices:**
  - Not requesting permissions as soon as your app is open
  - You should explain what will be sent
  - Users should be able to unsubscribe from each notification type individually

# Setup notification your Flutter App using FCM

## Requesting permissions

```
FirebaseMessaging messaging = FirebaseMessaging.instance;  
  
NotificationSettings settings = await messaging.requestPermission(  
  alert: true,  
  announcement: false,  
  badge: true,  
  carPlay: false,  
  criticalAlert: false,  
  provisional: false,  
  sound: true,  
);
```



# Differences between foreground and background

## Foreground notifications

- Receiving a notification when the app is already opened
- A device notification should probably not be displayed
- Adding the notification to the notification page

## Background notifications

- App can be opened or not
- If app is killed, it will be started to handle the notification
- You should handle the notification quickly enough

# Setup notification your Flutter App using FCM

## Listen to notifications in Foreground

```
FlutterMessaging.onMessage.listen((RemoteMessage message) {  
  print('Got a message whilst in the foreground!');  
  print('Message data: ${message.data}');  
  
  if (message.notification != null) {  
    print('Message also contained a notification: ${message.notification}');  
  }  
});
```



# Setup notification your Flutter App using FCM

## Listen to notifications in Background

```

@pragma('vm:entry-point')
Future<void> _firebaseMessagingBackgroundHandler(RemoteMessage message) async {
  // If you're going to use other Firebase services in the background, such as Firestore,
  // make sure you call `initializeApp` before using other Firebase services.
  await Firebase.initializeApp();

  print("Handling a background message: ${message.messageId}");
}

void main() {
  FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgroundHandler);
  runApp(MyApp());
}

```

# Send notification from a server using TS

- You can send a notification using Node.js, Java, Python, Go or C#
- No Dart SDK currently, you can use the API directly
- Using `firebase init functions` is the easiest way of initializing a TypeScript Cloud Function to send a notification



# Send notification from a server using TS

```
async function sendFcmNotification(fcmTokens: string[], historyId: string) {
  const payload: MessagingPayload = {
    notification: {
      title: "New email",
      body: "You have a new email",
    }
  };

  functions.logger.debug("HistoryId:", historyId);

  try {
    const result = await admin
      .messaging()
      .sendToDevice(fcmTokens, payload);
    functions.logger.debug("Successfully sent message:", result);
  } catch (error) {
    functions.logger.error("Error sending message:", error);
  }
}
```

# The data-only notifications

## Trigger actions in your app

- Data-only notifications are used to trigger actions in your App
- Could be logging something or changing a setting
- Your app has to answer this notification fast enough; otherwise it might get rate limited
- Your app will have to handle the notification on its own



# The data-only notifications

```
async function sendFcmNotification(fcmTokens: string[], historyId: string) {
  const payload: MessagingPayload = {
    data: {
      type: "new-email",
      historyId: historyId.toString(),
    },
  };

  const options: MessagingOptions = {
    priority: "high",
    contentAvailable: true,
  };

  functions.logger.debug("HistoryId:", historyId);

  try {
    const result = await admin
      .messaging()
      .sendToDevice(fcmTokens, payload, options);
    functions.logger.debug("Successfully sent message:", result);
  } catch (error) {
    functions.logger.error("Error sending message:", error);
  }
}
```

# Bind a notification to GoRouter

- You can get the global instance of the router and use it

```
final router = GoRouter(  
  navigatorKey: _rootNavigatorKey,  
  ...  
  
  @pragma('vm:entry-point')  
  void onDidReceiveNotificationResponse(  
    NotificationResponse notificationResponse,  
  ) async {  
    final String? payload = notificationResponse.payload;  
    final payloadParsed =  
      jsonDecode(payload?.isNotEmpty == true ? payload! : '{}');  
  
    final threadId = payloadParsed['threadId'];  
    if (threadId != null) {  
      log('Going to thread $threadId');  
      router.go('/home/thread/$threadId');  
    }  
  }  
}
```



# Access Riverpod in the background thread

- Allows to reuse the chain of dependency from Riverpod outside of the Widget Tree
- You should still access the ProviderContainer from Consumer widget when you can

```
@pragma('vm:entry-point')
Future<void> _firebaseMessagingBackgroundHandler(RemoteMessage message) async {
  late NotificationService notificationService;

  ProviderContainer? container = ProviderContainer();

  notificationService = await container.read(notificationServiceProvider.future);
  await notificationService.handleNewMail(message);

  container.dispose();
}
```

# When to use Flutter Local Notification

- FLN is a powerful Flutter plugins that allow to display of notification locally
- It supports features like displaying grouped notification
- Can be used to customise notification for each platforms
- Can be used with data-only notifications



# Notification Id

- You should ensure that your notification is always bound to an Id
- Helps to clear the notification if needed
- Prevent displaying the notification twice if the servers sends it too many time

```
await flutterLocalNotificationsPlugin.cancel(0);
```

# Threads

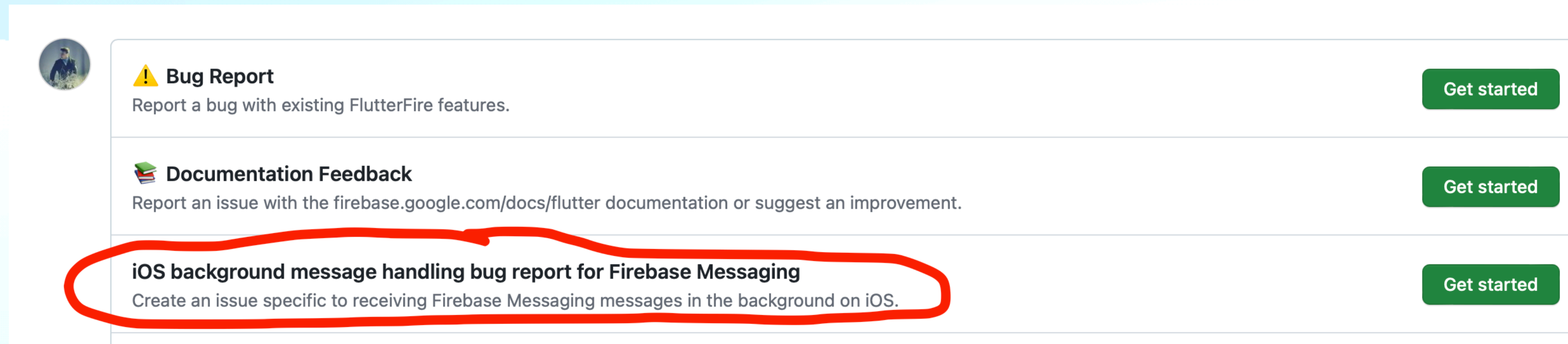
- Helps user identifies notifications from the same conversation
- Easy with Flutter Local Notification


```
await flutterLocalNotificationsPlugin.show(  
  mail.id,  
  mail.sender,  
  mail.snippet,  
  NotificationDetails(  
    iOS: DarwinNotificationDetails(  
      presentAlert: true,  
      presentBadge: true,  
      presentSound: true,  
      subtitle: mail.subject,  
      threadIdentifier: mail.threadId,  
    ),  
    android: AndroidNotificationDetails(  
      "channelId",  
      "New mails",  
      groupKey: mail.threadId,  
    ),  
  ),  
  payload: jsonEncode(  
    'threadId': thread?.id.toString(),  
  ),  
);
```



# Debugging notifications

- When your notification cannot be received on iOS, there is a special issue to help you troubleshoot and get all the logs required



	<b>⚠ Bug Report</b> Report a bug with existing FlutterFire features.	<a href="#">Get started</a>
	<b>📖 Documentation Feedback</b> Report an issue with the <a href="https://firebase.google.com/docs/flutter">firebase.google.com/docs/flutter</a> documentation or suggest an improvement.	<a href="#">Get started</a>
	<b>iOS background message handling bug report for Firebase Messaging</b> Create an issue specific to receiving Firebase Messaging messages in the background on iOS.	<a href="#">Get started</a>

Questions? 😄