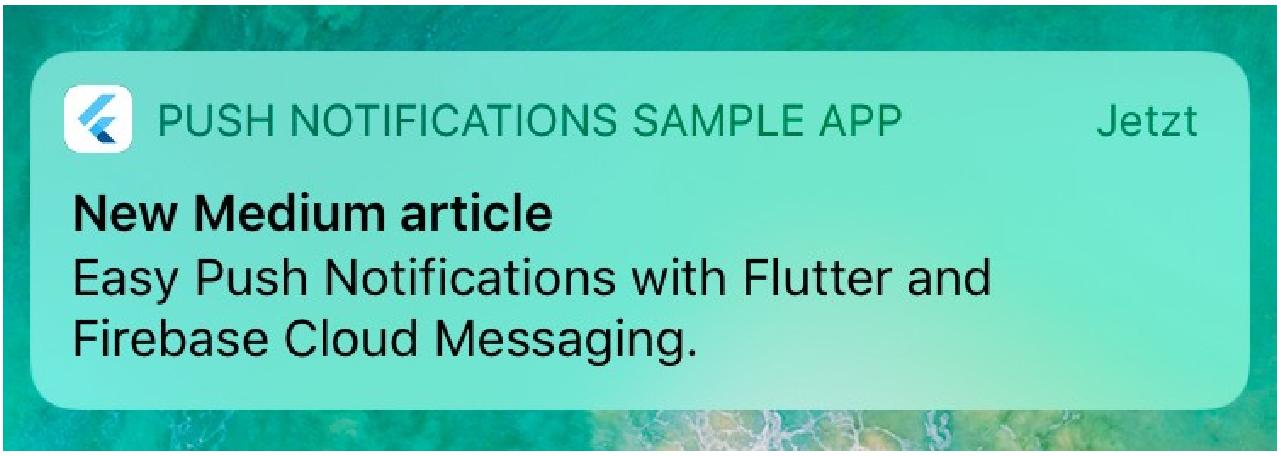
Easy Push Notifications with Flutter and Firebase Cloud Messaging

medium.com/@SebastianEngel/easy-push-notifications-with-flutter-and-firebase-cloud-messaging-d96084f5954f

August 11, 2019


PUSH NOTIFICATIONS SAMPLE APP
Jetzt
New Medium article
Easy Push Notifications with Flutter and Firebase Cloud Messaging.

Many apps are forgotten by its users very shortly after they got installed. Push notifications are a great possibility to bring your app back into your users mind and to increase the retention rate.

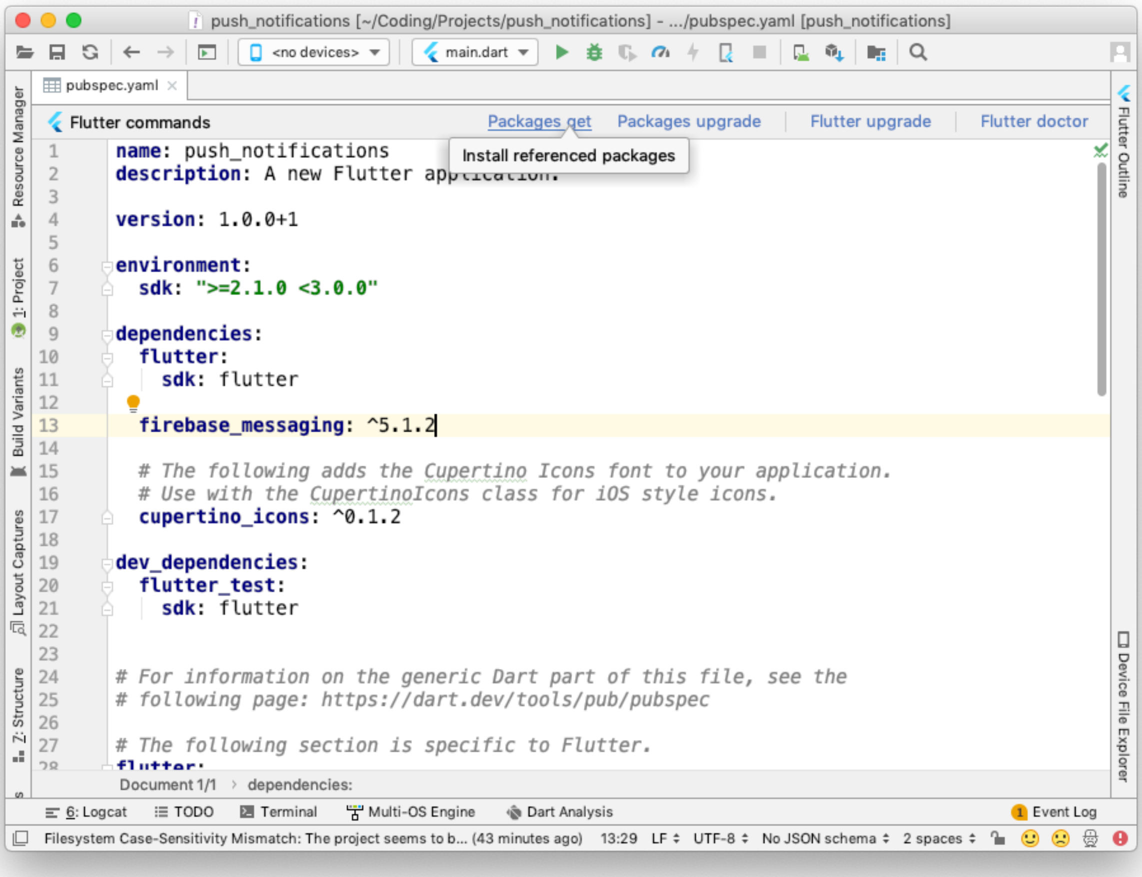
In this article I'll show you how to integrate push notifications easily into your Flutter app and how to send simple notifications from the Firebase console. The article assumes that you have an existing Flutter project or that a new one was created, for example with the Flutter standard project template.

1. Creating a Firebase project

Sending push notifications with Firebase Cloud Messaging requires a Firebase project. If you do not have one yet, create your's at firebase.google.com.

2. Integrating the firebase_messaging package

For the integration of *Firebase Cloud Messaging* Google's Flutter team provided the "*firebase_messaging*" package. Add the package dependency to your project's *"pubspec.yaml"* and load it by running `flutter pub get`.


push_notifications [~/Coding/Projects/push_notifications] - .../pubspec.yaml [push_notifications]
<no devices>
main.dart
pubspec.yaml
Flutter commands
Packages get
Packages upgrade
Flutter upgrade
Flutter doctor
Install referenced packages
Document 1/1 › dependencies:
6: Logcat
TODO
Terminal
Multi-OS Engine
Dart Analysis
Event Log
Filesystem Case-Sensitivity Mismatch: The project seems to b... (43 minutes ago) 13:29 LF UTF-8 No JSON schema 2 spaces

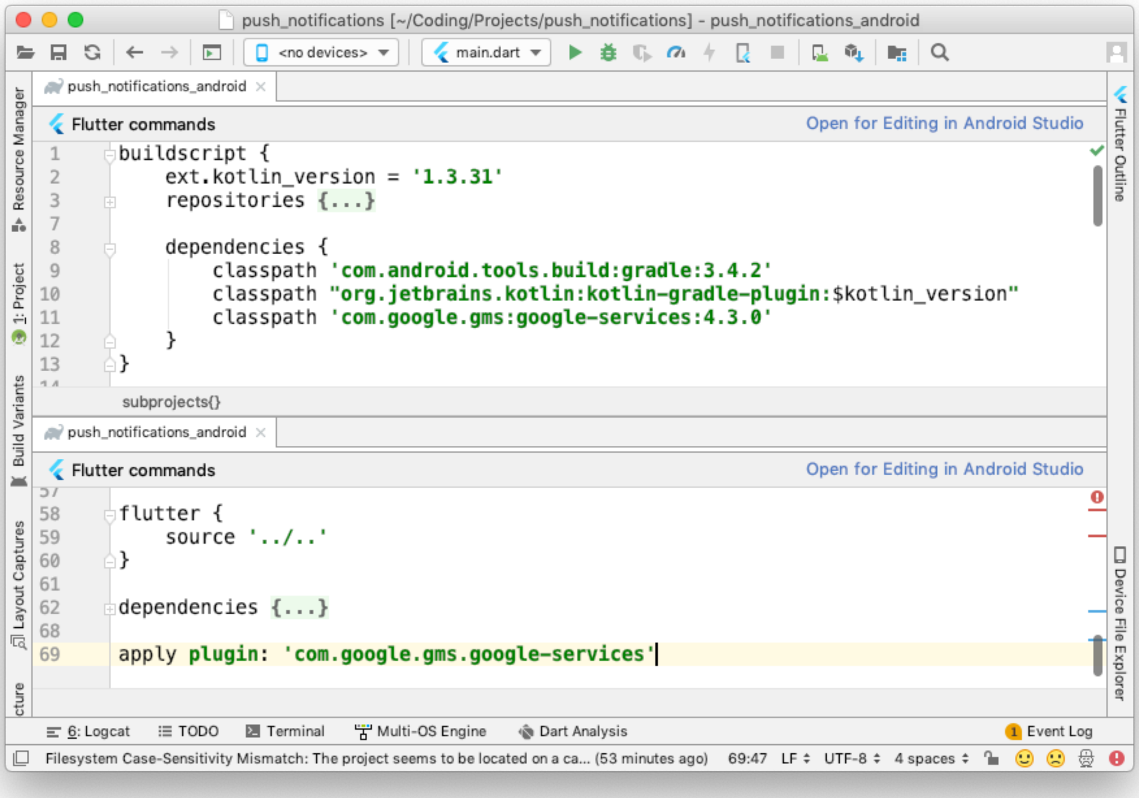
```yaml
name: push_notifications
description: A new Flutter application.

version: 1.0.0+1

environment:
  sdk: ">=2.1.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter

  firebase_messaging: ^5.1.2

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.2

dev_dependencies:
  flutter_test:
    sdk: flutter


# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.
flutter:
```

Integration of the firebase_messaging dependency

2.1 Android-specific configurations

On the Android side the *Google Services Gradle Plugin* needs to be included in the Gradle configuration. Simply add the following line to the *"dependencies"* section of your *"android/build.gradle"* file:
`classpath 'com.google.gms:google-services:x.x.x'`

To activate the plugin add the line following line to the end of your *"android/app/build.gradle"*:
`apply plugin: 'com.google.gms:google-services'` .

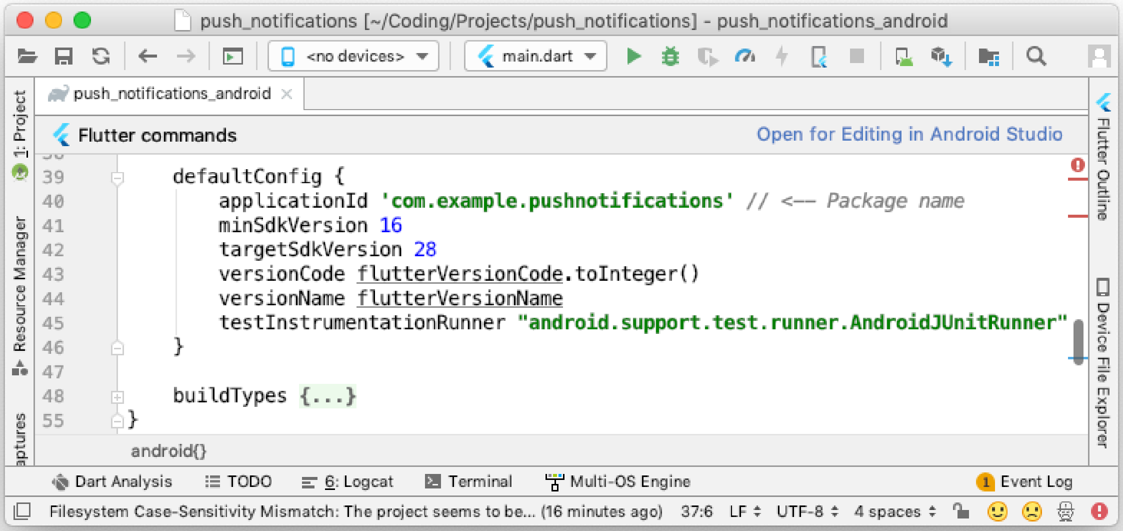

push_notifications [~/Coding/Projects/push_notifications] - push_notifications_android
push_notifications_android
Flutter commands
Open for Editing in Android Studio

```
buildscript {
    ext.kotlin_version = '1.3.31'
    repositories {...}

    dependencies {
        classpath 'com.android.tools.build:gradle:3.4.2'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
        classpath 'com.google.gms:google-services:4.3.0'
    }
}
```

```
flutter {
    source '../..'
}

dependencies {...}

apply plugin: 'com.google.gms.google-services'
```

Filesystem Case-Sensitivity Mismatch: The project seems to be located on a ca... (53 minutes ago) 69:47 LF UTF-8 4 spaces

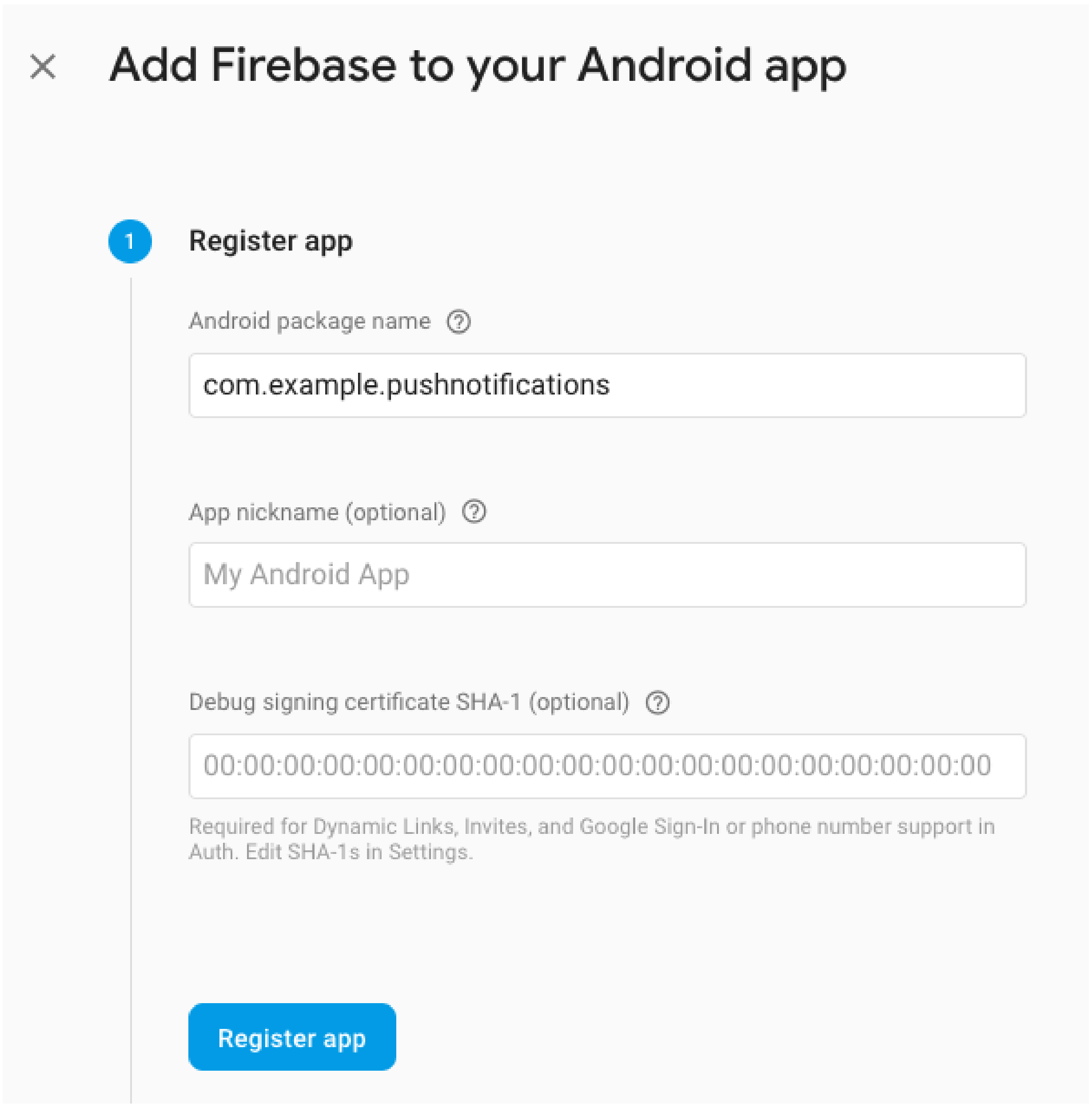
Next, create an Android-App in your Firebase-Project's "*Project Overview*" section. The package name can be found and defined in your *"app/build.gradle"* file.


push_notifications [~/Coding/Projects/push_notifications] - push_notifications_android
push_notifications_android
Flutter commands
Open for Editing in Android Studio

```
    defaultConfig {
        applicationId 'com.example.pushnotifications' // <-- Package name
        minSdkVersion 16
        targetSdkVersion 28
        versionCode flutterVersionCode.toInteger()
        versionName flutterVersionName
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }

    buildTypes {...}
}
```

Filesystem Case-Sensitivity Mismatch: The project seems to be... (16 minutes ago) 37:6 LF UTF-8 4 spaces

Package name in "app/build.gradle"


Add Firebase to your Android app
1
Register app
Android package name
com.example.pushnotifications
App nickname (optional)
My Android App
Debug signing certificate SHA-1 (optional)
00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
Required for Dynamic Links, Invites, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.
Register app

Creating the Android app in the Firebase console

After registering the app download the generated *google-services.json* and put it into your project's *"anroid/app"* folder.

If the app should open when the user taps on a notification, add the following intent filter inside of the `activity` element of your AndroidManifest.

```
<intent-filter>
  <action android:name="FLUTTER_NOTIFICATION_CLICK" />
  <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

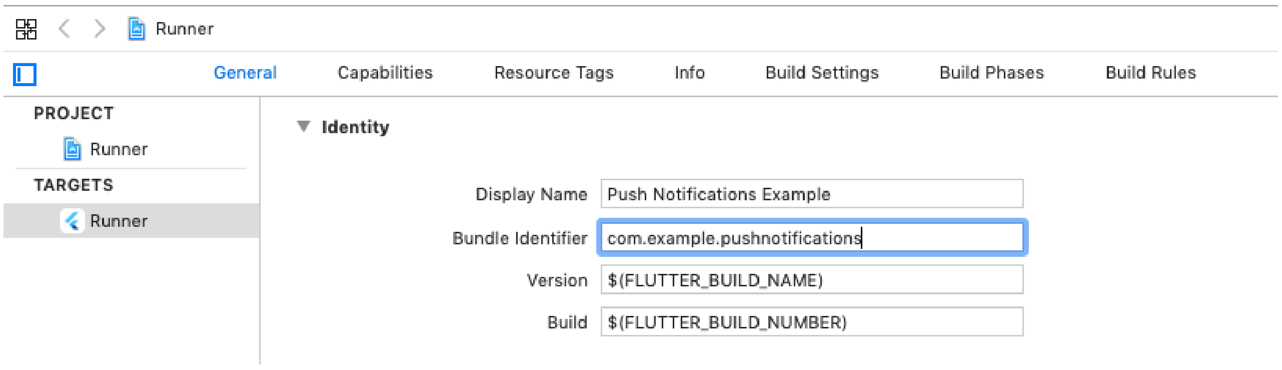
In order to receive these kind if intents, it is important to add the value "FLUTTER_NOTIFICATION_CLICK" with the key "click_action" to the "custom data" section when sending messages from the Firebase console.

2.2 iOS-specific configuration

To send messages to iOS devices, Firebase Cloud Messaging uses the Apple Push Notification Service (APNs). Using this services requires an *Apple Developer Account* and an *Apple Push Notification Authentication Key*. Both can be created in the Apple Developer Member Center. Follow the instructions of the FCM documentation, ignoring the section called "Create the Provisioning Profile". After successful creation download the authentication key and store in a secure place.

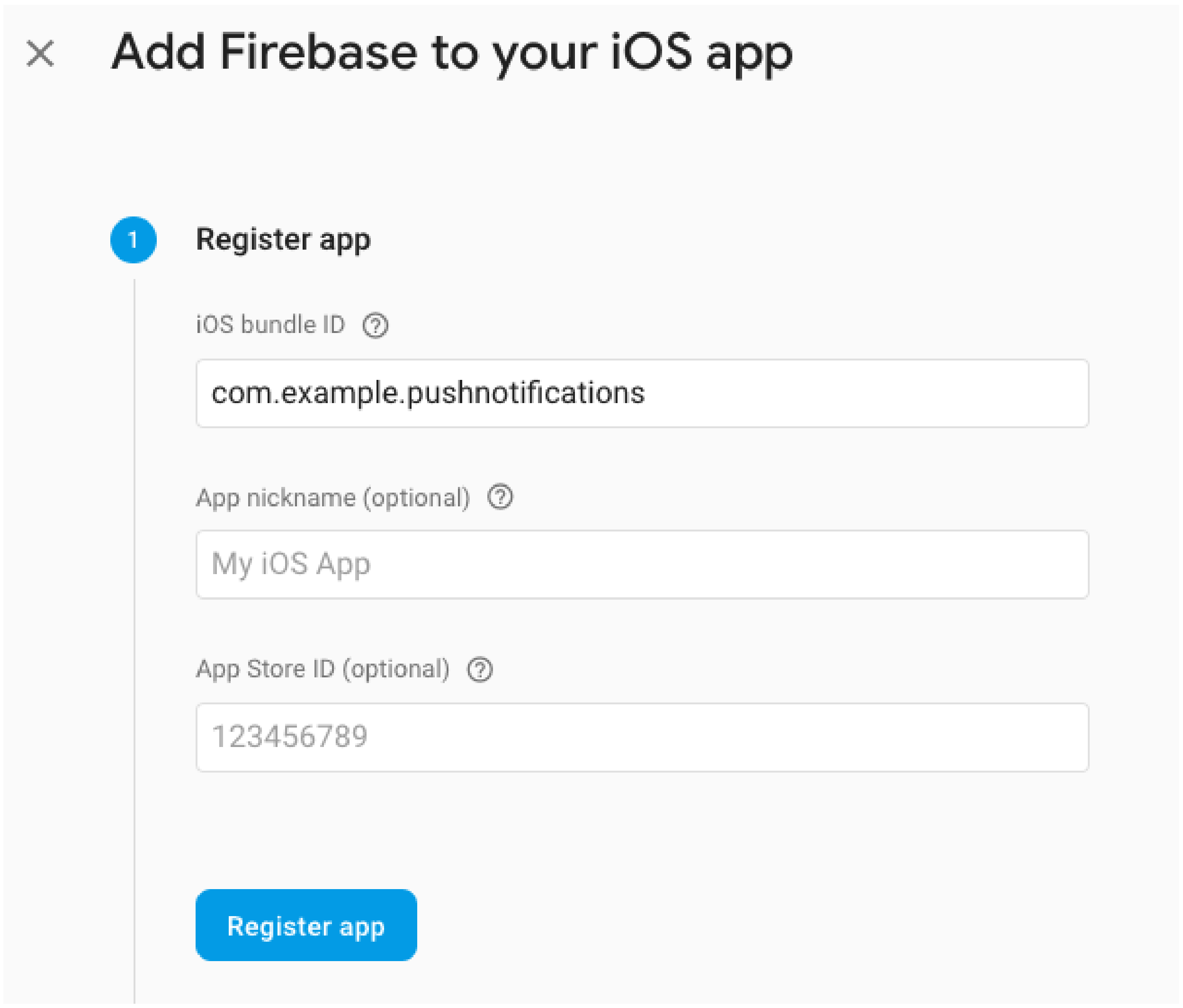
> Important note: To test push notifications on iOS an iOS device is required. Apple's push notifications cannot be sent to simulators.

As with Android, we now need to create an iOS project in the Firebase console. The requested iOS -Bundle-ID can be found in XCode under
"Runner > Target (Runner) > General".



Display Name	Push Notifications Example
Bundle Identifier	com.example.pushnotifications
Version	$(FLUTTER_BUILD_NAME)
Build	$(FLUTTER_BUILD_NUMBER)

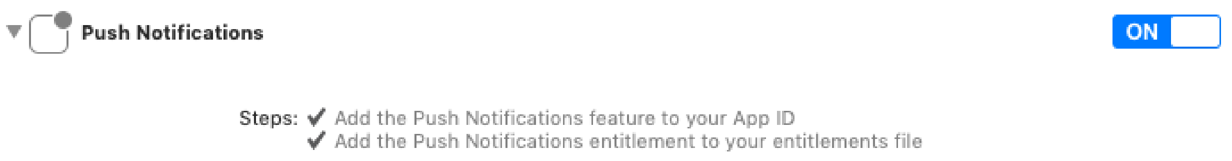
iOS-Bundle-ID in XCode



Creating the iOS app in the Firebase console

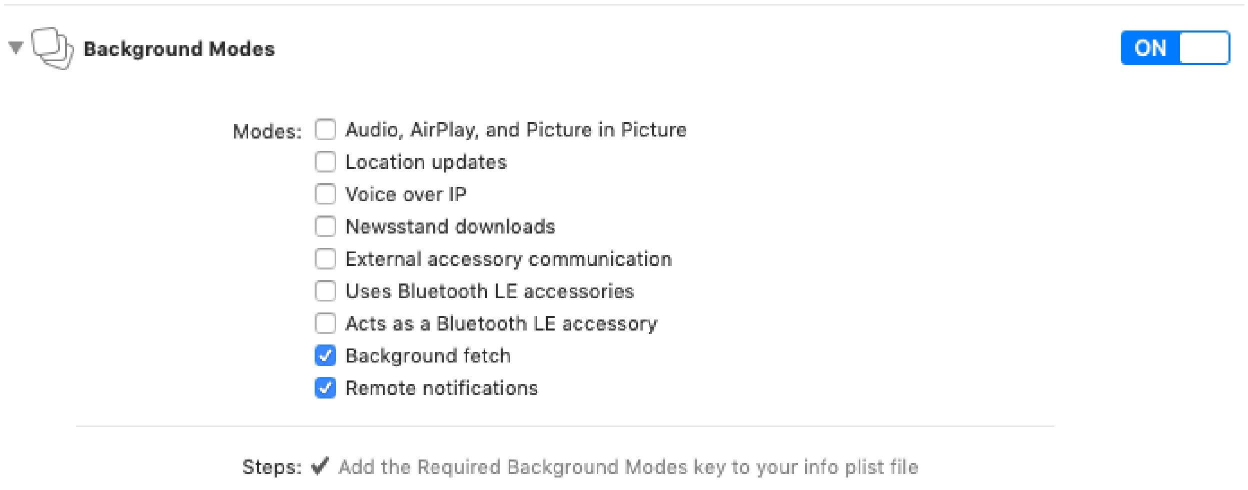
Next, the previously downloaded *Authentication Key* needs to be uploaded in your Firebase project's "*Cloud Messaging*" settings. You can find these settings via the gear icon besides the menu item called *"Project Overview"*.
Find your iOS app in the section called *"iOS app configuration"* and upload the APN key.

Now open the app's iOS module in XCode and activate *Push Notifications* and *Background Modes* under *"Project > Capabilities"*.


Push Notifications ON

Steps: ✓ Add the Push Notifications feature to your App ID
✓ Add the Push Notifications entitlement to your entitlements file

Activating Push Notifications


Background Modes ON

Modes:
- [ ] Audio, AirPlay, and Picture in Picture
- [ ] Location updates
- [ ] Voice over IP
- [ ] Newsstand downloads
- [ ] External accessory communication
- [ ] Uses Bluetooth LE accessories
- [ ] Acts as a Bluetooth LE accessory
- [x] Background fetch
- [x] Remote notifications

Steps: ✓ Add the Required Background Modes key to your info plist file

Aktivierung Background Modes

2.3 Initializing the packages in Dart-Code

All further configuration and handling of push notifications happens in Dart code.

To encapsulate the push notification related logic create a new file called *"push_nofitications.dart"* in you project's *"lib"* folder and define a class called *"PushNotificationManager"* in it. In order to prevent multiple initializations add a factory constructor which returns a singleton instance of the class.

Then add an *init()* method which initializes the *"FirebaseMessaging"* instance. In its simplest form the initialization only consists of requesting the user's permission on iOS and invoking the configuration of the *"FirebaseMessaging"* instance. In your existing app code create an instance of the *"PushNotificationManager"* and call its *init()* method.

From this moment on receiving push notifications is already possible. For sending test messages from the Firebase console to specific devices you might also want to print the Firebase Messaging token to the console.

When receiving messages the further behavior depends on whether the app is running in the foreground, the background or not at all.

- **App in foreground:**
  The notification will **not** be shown by the system automatically.
- **App in background:**
  The notification will be shown as a system notification automatically. Tapping it brings the app to the foreground.
- **App not running:**
  The notification will be shown as a system notification automatically. Tapping it launches the app.

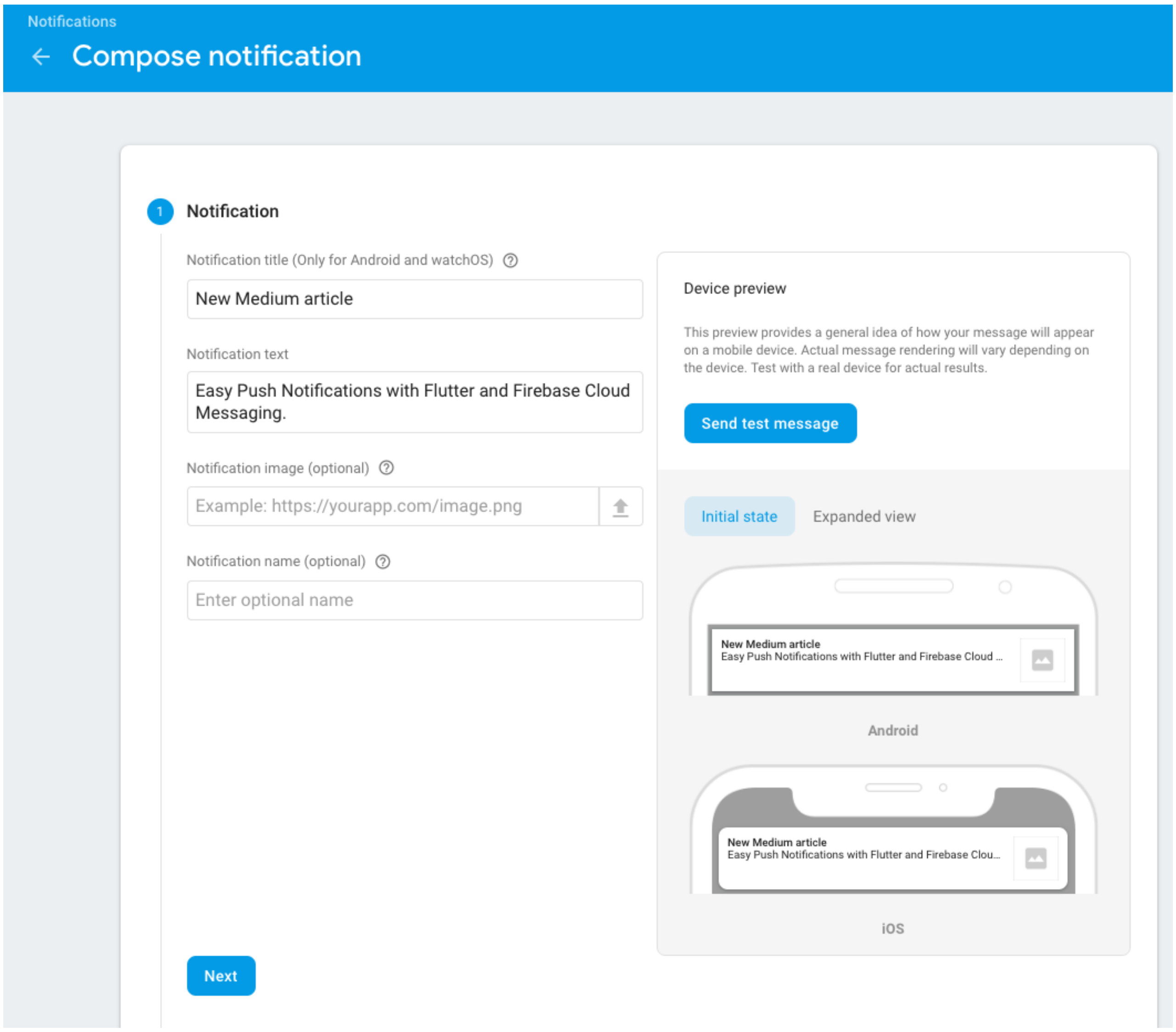
Letting the system handle and display the notifications and bringing the app to the foreground when the user taps on them might be sufficient for simple messages. But,

when the app is in foreground while the notification comes in, the user will never see it, nor know about it.
Also, if you send additional data besides title and message, this data is lost when the system handles the notifications automatically.
To solve these cases Firebase Messaging allows you to hand in *"MessageHandler"* functions to its `configure()` method.

In a follow-up article I will show how apps can react to notifications when in foreground and how to open a specific screen when a push notification comes in.

3. Sending Notifications with the Firebase console

After finishing the implementation push notifications can now be send from the Firebase console. Select *"Cloud Messaging"* in the console's navigation. Now click on the *"Send your first message"* button to open the form. Here, push notifications are created as campaigns. For a simple test enter a notification title and text.


Notifications
Compose notification

1 Notification

Notification title (Only for Android and watchOS): New Medium article

Notification text: Easy Push Notifications with Flutter and Firebase Cloud Messaging.

Notification image (optional): Example: https://yourapp.com/image.png

Notification name (optional): Enter optional name

Device preview

This preview provides a general idea of how your message will appear on a mobile device. Actual message rendering will vary depending on the device. Test with a real device for actual results.

Send test message

Initial state | Expanded view

New Medium article
Easy Push Notifications with Flutter and Firebase Cloud ...
Android

New Medium article
Easy Push Notifications with Flutter and Firebase Clou...
iOS

Next

Send test message

A click on the "Send test message" button opens a dialog that let's you select specific devices to send the test notifications to. Enter the Firebase Messaging token here that

was printed to the console as part of the initialization.
Finally, the *“Test”* button sends your message. Your test devices should ring now :)

Have fun with Flutter and push notifications!