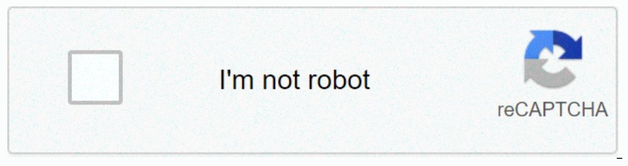

Continue

Amazon sns push notification android tutorial

-- This is a basic example of hooking up to Amazon SNS to register and broadcast push notifications to iOS and Android. The demo is a simple JVM server written in Cotlin. The code is kept very simple to focus on amazon SNS integration. You can expect an understanding of the mobile app so that you can digest it. Integrating push notifications for Firebase for Android and Apple iOS into the mobile application itself is well documented on Google Firebase and apple notification developer sites, although users are left to research. -- Create an iOS application with Prerequisites push notification enabled and an APNS certificate created. This certificate is sent to AWS SNS so that you can communicate with Apple APNS on your behalf. Integrate apns notification lifecycle code into iOS applications. Create an Android application, register with Firebase as a new project, and get a server key that is sent to AWS SNS so that you can communicate with Google GCM/Firebase on your behalf. Integrate the Firebase SDK and configuration into your Android application. Create an AWS account and create iOS platform applications and Android platform applications within the SNS service area. Attach the Apple APNS certificate to the iOS platform application credentials in the AWS SNS console. In the AWS SNS console, attach the Firebase server key to the credentials of the Android platform application. Create a recommended method for credentials to communicate with AWS. For basic information on how to do this, see the repository's 'Amazon.kt' source file. -- Download the clone or repository that runs the server. The demo server is Gradle-based, but can be opened for editing using IntelliJ. If you want to run the server through the terminal, use the following Gradle command: ./gradlew appRun Note: Running or opening a project for the first time can take some time to download all the dependencies you need. After you start successfully -- to open the project in IntelliJ, do the following: Navigate and open the cloned repository. Choose a setting like this: After opening the project in IntelliJ, install the IntelliJ Jetty Runner plug-in and create a new execution profile that looks like this: You can also run the Gradle task appRun from the Gretty task group to launch the server if you want: Opinion: The Pier Runner plug-in is much better to use in IntelliJ! Register content type POST: Application/json iOS client calls this {platform: ios,userId: AppleUser,token: APNS-push token provided iOS operating system} } The Android client calls this {platform.userId: AndroidUser, Token: Firebase Registration-id - By Provided Operating System } Content Type Notification POST: Application/json { userId: UserIdToTarget, Notification Title: Title NotificationBody: Body here!} -- There are a lot of articles on the net about how to integrate iOS notifications. Here are some super basic examples of common lifecycle callbacks for main application delegates for handling notifications (by no means exhaustive): UIKit import user notifications@UIApplicationMain class AppDeregate: UIResponder, UIApplicationDergate { var window: UIWindow? func Application (_Application: UIApplication: UIApplication, didFinishLaunchWithOptions Launch Options: UIApplicationLaunchOptionsKey: Optional)] -> Boolean { #available場合 (iOS 10.0, *) If optional: UNAuthoizationOptions = [.alert, .badge, .sound] UNUserNotifizationCenter.current(.Request Approval (Optional: Authentication Options, Completed Handlers: { Notification Settings = UIUserNotizationSettings(Type: [.alert, .badge, .sound,category: nil) application.registerUser notification settings (settings)} application.registerForRemoteNotifictions() // Notification data may also be delivered in this method. \(userInfo)) } func application (_ Application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken Device Token: Data) { Let Token Part = deviceToken.map { Data -> String returning string (format: %02.2hhx, data) } token = %02.2hhx, data) didFailToReggiterForRemoteNotifictionsWithError error: error) { print (registration failed: \(error)} } } -- Android Notification Integration Firebase documentation Run in good detail how to set up an Android application: page 2 -- is a basic example of hooking up to Amazon SNS for registering and broadcasting push notifications to iOS and Android. The demo is a simple JVM server written in Cotlin. The code is kept very simple to focus on amazon SNS integration. You can expect an understanding of the mobile app so that you can digest it. Integrating push notifications for Firebase for Android and Apple iOS into the mobile application itself is well documented on Google Firebase and apple notification developer sites, although users are left to research. -- Create an iOS application with Prerequisites push notification enabled and an APNS certificate created. This certificate is sent to AWS SNS so that you can communicate with Apple APNS on your behalf.Add the APNS notification lifecycle code to the iOS application. Create an Android application, register with Firebase as a new project, and get a server key that is sent to AWS SNS so that you can communicate with Google GCM/Firebase on your behalf. Integrate the Firebase SDK and configuration into your Android application. Create an AWS account and create iOS platform applications and Android platform applications within the SNS service area. Attach the Apple APNS certificate to the iOS platform application credentials in the AWS SNS console. In the AWS SNS console, attach the Firebase server key to the credentials of the Android platform application. Create a recommended method for credentials to communicate with AWS. For basic information on how to do this, see the repository's 'Amazon.kt' source file. -- Download the clone or repository that runs the server. The demo server is Gradle-based, but can be opened for editing using IntelliJ. If you want to run the server through the terminal, use the following Gradle command: ./gradlew appRun Note: Running or opening a project for the first time can take some time to download all the dependencies you need. After you start successfully -- to open the project in IntelliJ, do the following: Navigate and open the cloned repository. Choose a setting like this: After opening the project in IntelliJ, install the IntelliJ Jetty Runner plug-in and create a new execution profile that looks like this: You can also run the Gradle task appRun from the Gretty task group to launch the server if you want: Opinion: The Pier Runner plug-in is much better to use in IntelliJ! Register content type POST: Application/json iOS client calls this {platform:ios, userId:AppleUser, token:APNS push token offer iOS operating system}Android client calls this {platform:Android, this platform. userId: AndroidUser,Token:Firebase Registration-id-By Provided Operating System } post Content Type: Application/json { userId: UserIdToTarget,NotificationTitle: Title, NotificationBody: Body here!} -- There are a lot of articles on the net about how to integrate iOS notifications. Here are some super basic examples of common lifecycle callbacks for main application delegates for handling notifications (never exhaustive): import UIKit import user notifications@UIApplicationMain class AppDergate: UIResponder, UIApplicationDergate {varwindow? func applications (_ applications: UIApplication, didLaunchWithOptions launch options: UIApplicationLaunchOptions: optional]>)10.0, *) { Allow authentication options: Authorization options = [.alert, .badge, .sound] UNUserNotifizationCenter.current().Request approval (optional: authentication option, completion handler: { _, _) } Allow other settings: UIUser notification settings = UIUser notification settings (type: [.alert, .badge, .sound], category: nil) application.registerUser notification settings (settings) }true } return application (_ application: UI application, didReceiveRemoteNotication user information: [AnyHashable: optional]) { // Notifications can be processed here I received a notification!\(userInfo)) } func application (_ Application: UIApplication, didReggisterForRemoteNotificationsWithDeviceToken Device Token: Data) { Let Token Parts = Device Token.Map { Data -> String returning string (format: %02.2hhx, data) } Allow tokens = tokenParts.joined() // LOOK: Tokens can now be sent to the registration application server .