補充教材

程式總是學不完的

iOS 有成千上萬個 SDK, 是不可能教完和學完的, 在最後的時間, 會選擇學員工作上最需要的, 或最想學的來實作。

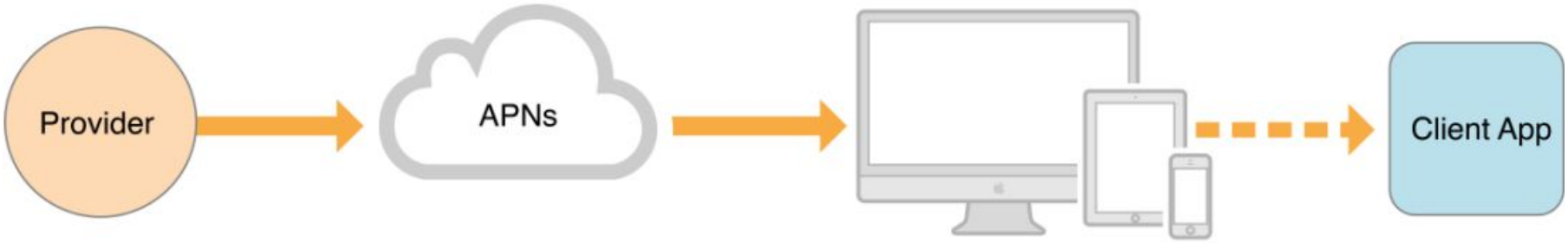
語音說話程式
Siri 整合
UIAnimation 動畫
Facebook 發文
字串的那些事
文字轉拼音
陣列進階應用
有顏色的字
影片, 音樂播放
APNS ＋Firebase 推播
各種跨平台方案簡介
　如 Flutter, Xamarin

加速器, 電子羅盤, 計步器
Firebase Realtime DB 進階
Firebase Storage
手勢放大縮小旋轉
CoreData
檔案 APP
iColud
檔案存取
照片與相機
macOS 程式
Apple Map
其他

APNS + Firebase（FCM）實作遠端推播

遠端推播基本原理

在每一個平台上的推播方案，雖都有完整的解決方案，如Apple的 APNS，Google 的 GCM，或 Microsoft 的 WPNS等，但只有 PNS（Push Notification Service）是不以完成安全與完整的推播的。最小的狀況下，需要有一個發送器的後台，來執行推播的發送。



推播流程1~登錄可被推播設備

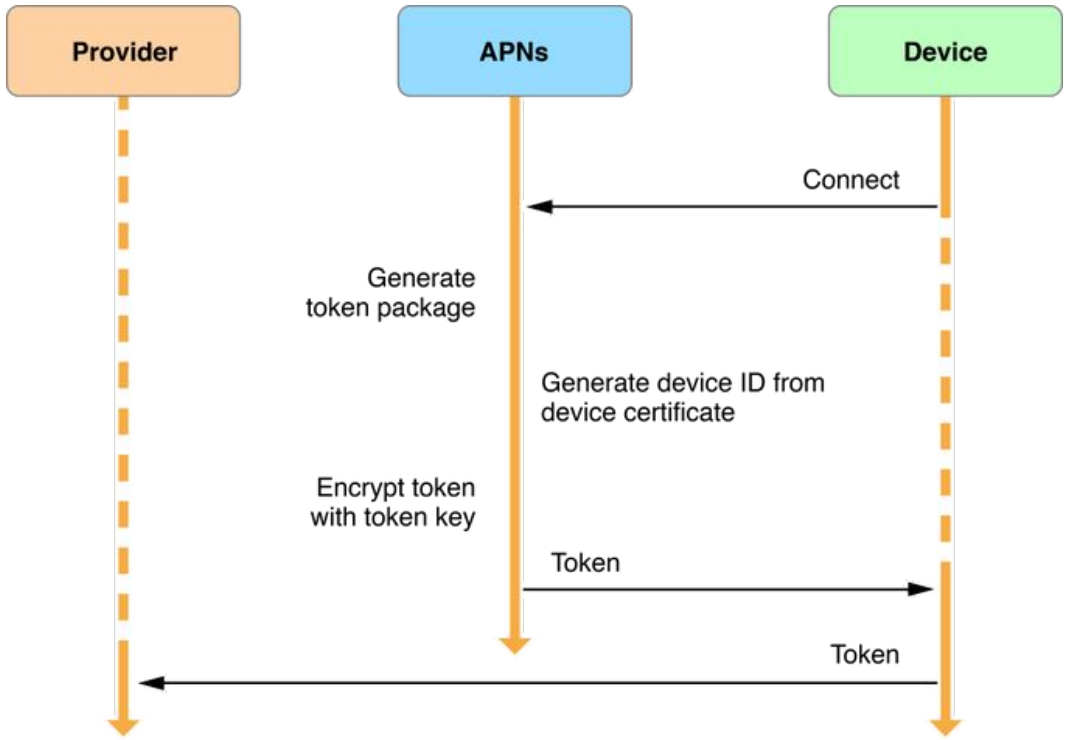
首先, 你必需要有一個提供推播者(Provider)來通知 APNS, 這個提供推播者至少要提供三個東西, APNS才會執行通知

1. 安全憑證

2. 目標機器

3. 推播內容

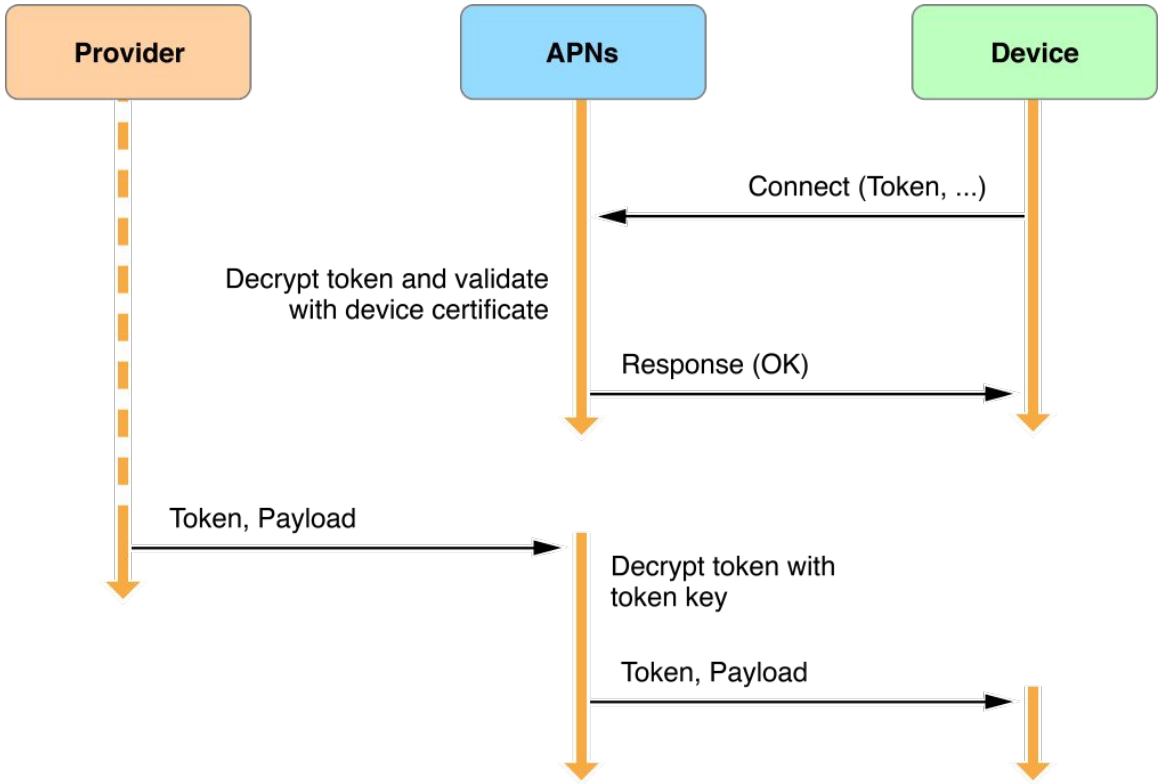
安全憑證都是在建立應用時, 各平台依據各自的 ID 所產生的, 如 Android 的 Package Name 或 iOS 的 Bundle ID 經過管理平台加密後 產生, 解密的關鍵也只有平台有, 所以平台就能知道該送給那一支 APP, 而目標機器是在手機或平板在開啟時與系統加密 產生再回送給 Provider 一個 Toke



推播流程 2 ~ 發送訊息至推播設備

在取得安全憑證, 知道 Device Token 之後, 就可以把要發送的 內容與相關參數等資料, 包成 Payload 送給 APNS , APNS 就會在 Device 上線時 , 發送給它。

Provider 原則上就是自家的後台, 但實際上建構 Provider 是很麻煩的, 通常需要了解 iOS / Android 甚至其他系統的推播, 分別去處理, 所以類似 Firebase 等系統就協助處理了大部份的工作, 甚至不一定需要自架 Provider



推播本身應有的概念

遠端推播在 iOS 上是基於作業系統的推播, 所以收到之後的動作, 作業系統有絕對主導權, 使用者的各項設定, 作業系統的版本, 都有可能影響到收到推播時的處理

推播本身是不保證送達, 不主動回應的機制。

推播透過網路送達, 所以網路狀態, 路由器, 防火牆等, 都會影響收發。

推播不是簡訊, 沒有 APP 只有電話號碼是無法執行推播的

iOS 只能在付費帳號, 實體機才能正確運作

Firebase 在推播系統的角色

整合 iOS / Android 等各種 PNS 與版本整合

不同系統之間, 推播的設定不儘相同

提供群推時的效能

提供分群/預約等進階功能

iOS + Firebase 建立推播 測試機制

1. 取得 APNS push notification 的 key, 上傳至 FireBase 專案設定-> 雲端通訊 -> 你的 APP 中, 注意 Team ID 與 Key ID
2. 建立 Xcode 的 iOS 專案, 並在 Target 的 Signing & Capabilityies 中, 加入 Push Notification
3. 將 iOS 專案加入 Firebase 對應專案
4. 在 iOS 專案加入 Firebase CloudMessage SDK, GoogleInfo.plist 等 Firebase 基礎設定
5. 在 Appdelegate 中加入
   a. 登入要求使用 APNS 相關程式碼
   b. 在取得 Token 時, 回傳給 Firebase 的相關程式碼
6. 依 Firebase 文件建立相關程式碼
7. 至 Firebase 後台試送推播
8. 確認有網路, 未開靜音, APP 準許推播的條件下在實體機執行 APP, 確認有收到 Token 後, 關閉 APP

```
//  AppDelegate.swift

import UIKit
import Firebase
import FirebaseMessaging

@main
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate,
MessagingDelegate {
    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.

        UNUserNotificationCenter.current().delegate = self
        FirebaseApp.configure()
        Messaging.messaging().delegate = self
        let authOptions: UNAuthorizationOptions = [.alert, .badge, .sound]
        UNUserNotificationCenter.current().requestAuthorization(
          options: authOptions,
          completionHandler: { _, _ in }
        )
        application.registerForRemoteNotifications()
        return true
    }

……..
```

```
  func application(_ application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: Data) {
      let tokenString = deviceToken.map { String(format: "%02x", $0) }.joined()
      print("Device Token: \(tokenString)")
   }


   func messaging(_ messaging: Messaging, didReceiveRegistrationToken fcmToken: String?) {
      Messaging.messaging().token { token, error in
        if let error = error {
          print("Error fetching FCM registration token: \(error)")
        } else if let token = token {
          print("FCM registration token: \(token)")
        }
      }
      let dataDict: [String: String] = ["token": fcmToken ?? ""]
      NotificationCenter.default.post(
        name: Notification.Name("FCMToken"),
        object: nil,
        userInfo: dataDict
      )
```

P8 與 P12 的差別

用途:

P8:用於 APNs 的 Token 驗證方式。它是一種更現代化的身份驗證方法,適合多個應用程序使用,並且不會過期。

P12:用於基於證書的身份驗證方式。每個應用需要單獨配置兩個證書(開發和生產環境),並且需要定期更新。

內容:

P8:僅包含私鑰,並以 Base64 編碼的文本格式存儲。

P12:包含私鑰、公鑰、證書鏈以及其他元數據,並以二進制格式存儲。

P8 與 P12 的差別~ 續

安全性:

P8:因為只包含私鑰, 安全風險相對較低。

P12:包含私鑰和公鑰, 若文件洩露, 可能會導致更高的安全風險。

管理便利性:

P8:一個密鑰可以用於多個應用, 並且生成後只允許下載一次, 需妥善保存。

P12:每個應用需要單獨配置, 並且證書有有效期, 過期後需要重新生成。

目前 P12 通常只用於客戶自己沒有開發者帳號, 代理發行, 但客戶要自己發送後台時。不過強烈建議客戶自己維護開發者帳號 P12 生成方法, 建議搜尋或問 AI