モバイル端末を利用した加速器運転情報アプリケーションの開発

DEVELOPMENT OF ACCELERATOR STATUS APPLICATION FOR MOBILE DEVICES

草野史郎 *A)、佐藤政則 B)、佐武いつか B)
Shiro Kusano* A), Masanori Satoh B), Itsuka Satake B)
A)Mitsubishi Electric System & Service Co., Ltd.
B)High Energy Accelerator Research Organization (KEK)

Abstract

The KEK injector linac provides electron/positron beams to five different rings (SuperKEKB LER 4 GeV, HER 7 GeV, positron damping ring 1.1 GeV, PF 2.5 GeV PF-AR 5 GeV). For the long-term stable beam injection, it is very important to monitor the accelerator operation status. In recent years, the evolution of mobile devices has been remarkable. By using these devices, it is easily possible to grasp the status of accelerators anytime and anywhere. We developed an application displaying the accelerator operation log for both of iOS and Android OS. The development environment and language of the application are Xcode/Swift and Android Studio/Kotlin for iOS and Android OS, respectively. This paper reports the details of this application and future perspectives.

1．はじめに

KEK 電子陽電子入射器（以下入射器）では、エネルギーの異なる 5 つのリング（PF、PF-AR、SuperKEKB HER/LER、および陽電子ビーム用ダンピングリング）にビームを供給している。長期間、安定したビームを供給するには、加速器運転状況を把握することは非常に重要である。近年、モバイル端末の進化は著しく、これらの端末を利用することでいつでもどこからでも加速器の状況を把握することができる。今回、iOS および Android OS で動作する加速器運転ログ表示用アプリケーション（以下アプリ）を開発した。本アプリの開発環境および使用言語は、それぞれ Xcode/Swift [1]、Android Studio/kotlin [2] である。本稿では、本アプリケーションについて、詳細を報告する。

2．アプリケーション概要

2.1 開発環境

iOS アプリ開発用のコンパイラは、macOS 用のみ提供されていた。Android OS アプリの開発は、macOS 用、Windows 用ともに利用可能であったため、開発環境として MacBook Pro を採用した。本アプリの開発環境を Table 1 に示す。

Table 1: Development Environment

PC	MacBook Pro
CPU	R2.3 GHz クアッドコア Intel Core i5
Memory	16 GB
OS	macOS Big Sur

* skusa@post.kek.jp

2.1.1 iOS 用開発環境

iOS アプリの開発は、統合開発ツールである Xcode でおこなった。Xcode は、Apple 社が提供する iOS 向けアプリ開発用の統合開発環境（IDE）である。ソースコードの編集のみならず、デバッグ機能や iOS シュミレータを利用した端末での動作確認などがおこなえる。Xcode で使用できる OS は、macOS のみとなっている。Xcode が対応している言語には、Swift、Objective-C、C 言語、C++、Java および AppleScript がある。Figure 1 に Xcode の表示画面を示す。

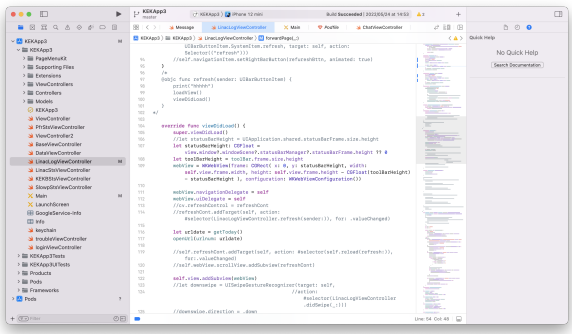


Figure 1: Screen shot image of Xcode.

2.1.2 iOS 開発言語

iOS アプリの開発言語には、Swift を使用した。Swift は、Apple 社が開発したオープンソースのプログラミング言語で、主に iOS、macOS 向けのアプリケーション開発のために開発された言語である。Swift の特徴は、以下の通りである。

- モダンなプログラミング言語で書きやすく・可読性が高い。
- 軽量な動作であるため、Objective-C より高速処理が可能である。

iOS や macOS のアプリ開発以外にも、さまざまな企業で Swift を導入する動きが高まっており、Google

や Facebook でも Swift に対応したソフトウェア開発キットを提供している。このように多くの企業で採用されているため、将来性が高いと判断して開発言語として採用した。

2.1.3 Android 用開発環境

Android アプリの開発は、Android Studio でおこなった。Android Studio は、Google が提供する Android OS 向けアプリ開発用の IDE である。Android Studio は、Android 開発者向けのサイト Android Develpers から無償でダウンロードすることができる。Android Studio は開発するための OS に制限がなく、Windows、macOS、Linux および Google Chrome OS に対応している。Figure 2 に Android Studio の表示画面を示す。

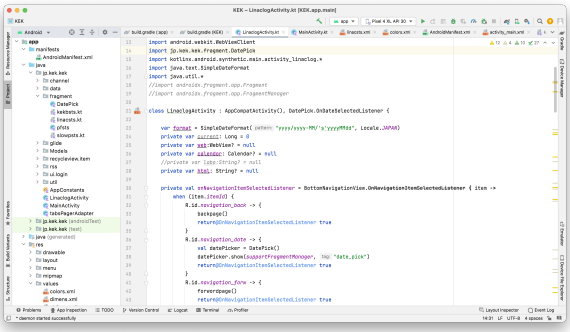


Figure 2: Screen shot image of Android Studio.

2.1.4 Android 開発言語

Android アプリの開発言語には、Java、Kotlin、C#など複数の言語があるが、本アプリの開発では、kotlin を使用した。Kotlin の特徴は、以下の通りである。

- Java と 100% の互換性を保ちつつ、より簡単に書くことができる。
- Google が正式言語として採用している。
- NullPointerException によるエラーが発生しない

kotlin は、Swift と同様に多くの企業で採用されているため、本アプリの開発言語として採用した。

2.2 mobile Backend as a Service（mBaaS）

本アプリでは、加速器の運転状況などを表示しているため、関係者のみが閲覧できることが望ましい。そのため、ユーザー認証サーバーや加速器の情報を保管するサーバーを用意する必要がある。しかしながら、モバイルアプリと研究所内サーバー間でデータ交換やユーザー認証の構築が容易でないため、Google 社が提供する mBaaS である Firebase を利用した。Firebase は、クラウドサービスの一種であり、主な機能として以下のものがあげられる。

- ユーザー認証機能の Firebase Authentication
- 端末へのプッシュ通知機能の Firebase Cloud Messaging
- クラウド上の外部ストレージの Cloud Storage for Firebase
- NoSQL データベースの Cloud Firestore
- リアルタイムデータベースの Firebase Realtime Database

2.2.1 Firebase Authentication

Firebase Authentication は、ユーザー認証機能を提供し、ユーザー情報をクラウド上で管理する。認証方法は、メールアドレスとパスワードによる認証、主要なプロバイダーアカウントによる認証、匿名認証、カスタム認証および電話番号認証がある。本アプリでは、メールアドレスとパスワード認証を利用している。本アプリインストール前に管理者がメールアドレス、パスワードを Firebase に登録し、ユーザーがアプリ使用時に登録された情報で認証をおこなうことでアプリを使用可能としている。

2.2.2 Firebase Realtime Database

Firebase Realtime Database は、クラウドホスト型の NoSQL データベースであり、ユーザー同士でデータをリアルタイムに保存・同期ができる。本アプリでは加速器のスケジュール、運転情報および電荷量などのリアルタイムに変化する情報を表示するものに使用している。Firebase Realtime Database は、http 経由でデータの同期が可能であり、python モジュールも用意されている。入射器では、EPICS CA モジュールと組み合わせた python スクリプトを用いて、加速器の電荷量、ビームパワーなどの情報を取得し、http、proxy 経由で Firebase Realtime Database への書き込みをおこなっている。また、Firebase Realtime Database と Firebase Authentication を組み合わせることで他のユーザーがデータベースにアクセスできないようにしている。

2.3 アプリの配布方法

2.3.1 iOS アプリ

iOS アプリの配布は、実機デバッグ、Web サイトでの配布（AdHoc）、AppStore での配布（TestFlight）、AppStore（一般公開）の方法がある。実機デバッグとは、開発中のアプリを Mac と iOS デバイスを直接 USB で繋いで Xcode 経由でインストールする方法である。この方法は、有料ライセンスが不要であるがデバッグ期間が 7 日間と短い。AdHoc は、有償の Apple Developper Program と呼ばれるライセンスが必要（年間 99 ドル）で、Xcode で作成した ipa ファイルを任意の Web サーバーから配布することが出来る。ただし、UDID（アプリを使用する端末に ID 番号）やプロビジョニングプロファイルの管理が必要となる。TestFlight での配布は、まずアプリ利用者がアプリ作成者の招待状により AppStore から TestFlight アプリのインストールをおこなう。TestFlight アプリを通じてアプリのインストールが可能となる。TestFlight のメリットは、UUID の管理が不要で、アプリを更新した際、TestFlight アプリよりアップグレード通知のサービスを受けることが出来る。

2.3.2 Android アプリ

Android アプリの配布は、Google Play を通じた配

布、アプリをメール添付しての配布、個人または企業の Web Site する方法がある。任意の Web サイトから配布する場合、アプリをインストールするユーザーは、端末本体の設定で提供元不明のアプリのインストールを許可する必要がある。メールの添付で配布する場合、メールを受けとったユーザーが他人に転送すると誰でもアプリをインストールできるため、プライバシーの保護や不正な配布に対処する手段がない。Google Play を用いた配布する場合、アプリ内課金やアップグレード通知などのサービスが利用できる。

3. 加速器運転情報アプリ

Figure 3 に iOS と Android アプリのトップ画面を示す。(a) が iOS 用、(b) が Android 用アプリのトップ画面である。各トップ画面に表示されているタブは、入射器、SuperKEKB、PF、低速陽電子の加速器情報を表示している画面である。入射器タブにおいては、シフトリーダーとオペレーターの氏名、加速器施設の概況および予定を表示している。運転日誌、トラブル、メッセージは、以下に記述する各パネルへのリンクボタンとなっている。SuperKEKB、PF、低速陽電子タブは、現在開発中である。

(a) iOS Top Page (b) Android Top Page

Figure 3: Top page image of iOS and Android Applications.

3.1 運転日誌パネル

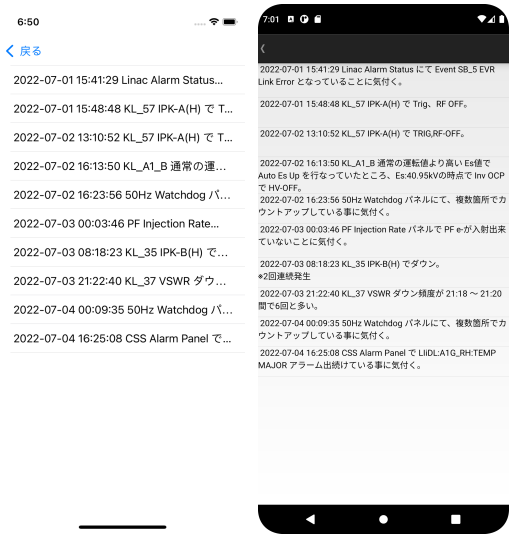
Figure 4 に運転日誌パネルを示す。運転日誌パネルは、すでに所外向けに提供している電子ログブックシステムの Web ページを利用している [3]。画面下部に表示されている前ボタンは前シフト、後ボタンは次シフト、日付は任意の日付に遷移する。電子ログブックシステムの Web ページでは、関係者以外が閲覧できないようにユーザー認証で管理されている。本アプリは、接続する度パスワードを入力する手間を省くため、iOS では keychain、Android では keystore モジュールを用いて端末本体にユーザーおよびパスワードを暗号化して管理している。

(a) iOS Operation Log (b) Android Operation Log

Figure 4: iOS and Android Operation Log.

3.2 トラブルパネル

Figure 5 にトラブルパネルの表示画面を示す。トラブルの表示は、入射器で提供している RSS フィードの Web ページを利用して表示している。パネルでは、直近のトラブル件数 10 件をリスト表示している。各リストをタップするとトラブルの詳細が表示される。



(a) iOS Trouble Report (b) Android Trouble Report

Figure 5: iOS and Android Trouble Report.

4． 問題点

4.1 https Web Server

運転日誌パネル表示は、iOS、Android ともに WebView クラスを利用している。Webview クラスを利用することで、Web ブラウザの機能を持ったアプリを簡単に作成することができる。しかしながら、Webview クラスは https での接続を推奨している。従来の入射器 Web サーバーは、http 接続のみ使用しており、Webview クラスから接続ができない。そのため、入射器 Web サーバーへの https 接続を可能するための認証システムを構築した。

4.2 AppStore によるアプリの配布

本アプリの本格運用を目指すにあたり、AppStore からのアプリ配布を検討した。研究所関係者のみの利用を想定しているため、アプリ内の情報や運転日誌閲覧は、ユーザー認証による管理をおこなっている。一般ユーザーでも利用できるアプリとならないため、AppStore からの配布ができないことがわかった。

4.3 共通開発環境

現在、本アプリの開発には、2 つの言語を使用している。開発時間が 2 倍かかることから、共通開発環境による開発も検討した。共通開発環境には、Xamarin [4]、Flutter [5]、React Native [6] など複数存在する。しかしながら、共通開発環境の多くは、独自の UI による処理の遅さ、利用できない端末の機能（GPS、プッシュ通知など）などがあることから共通開発環境による開発を断念した。

5． まとめと今後の課題

入射器の加速器運転情報アプリケーションの開発をおこなった。現在、2~3 名のユーザーで試験運用をおこなっている。本アプリは加速器の情報を得るツールとして有用であることが確認できた。今後は、さらに多くの関係者が利用できるようなアプリの配布方法を検討し、本格的な運用を目指す予定である。

参考文献

[1] https://developer.apple.com/jp/xcode/
[2] https://developer.android.com/studio
[3] T. Kudou *et al.*, “入射器における電子ログブックシステムの更新”, Proceedings of the 8th Annual Meeting of Particle Accelerator Society of Japan, Tsukuba, Japan, August 1-3, 2011.
[4] https://docs.microsoft.com/ja-jp/xamarin/
[5] https://flutter.dev/
[6] https://reactnative.dev/