Comparación de procesos de desarrollo móvil: Android e iOS

David André Díaz Rosario
Facultad de Ingeniería y Tecnología
Ingeniería en Sistemas Computacionales
Universidad de Montemorelos

El objetivo de este proyecto es comparar los procesos de desarrollo de aplicaciones móviles iOS y Android a través de varias facetas como el diseño, los lenguajes utilizados, el entorno de desarrollo y la programación. Esta comparación pretende servir de pauta al barajar las ventajas y desventajas de desarrollar para cada plataforma, así como explicar ciertos fundamentos teóricos del desarrollo. Se crearon dos aplicaciones, una para iOS y otra para Android, con características y contenido similares.

Keywords: Comparación, procesos, android, iOS, aplicaciones

Introducción

Hoy en día, los dispositivos móviles, y más específicamente, los smartphones son unos de los dispositivos electrónicos más usados. Los aparatos electrónicos más poderosos y que están al alcance de las personas comunes, como las computadoras personales y los teléfonos móviles, utilizan un conjunto de software llamado Sistema Operativo que se encarga de que el sistema y el resto de aplicaciones funcionen bien [1]. El caso es que cuando se trata de teléfonos móviles, hay dos sistemas operativos quen no pueden quedar fuera de la conversación: iOS y Android. Esto se debe a que son prácticamente los únicos utilizados; en México, la cantidad de dispositivos Android supera a la de iOS en una proporción de 5 a 1 y juntos acaparan más del 99% del mercado [2]. Los programas no existen en el vacío, ajenos a las minucias de la vida real, sino que forman parte de otros programas [3] e interactúan con diversos sistemas.

Debido a esto, crear una aplicación que funcione para ambos sistemas operativos no es una tarea trivial. Existen distintas opciones y términos relevantes para enfrentar este problema, como las aplicaciones híbridas y las nativas (estas últimas son las que usamos en este proyecto). Aprender a programar bien es un proceso lento y difícil [4] y a veces no es fácil encontrar recursos en español. El proceso de desarrollo no solo incluye la programación propiamente dicha sino otras actividades y cosas misceláneas como la elección de la base de datos, el diseño previo de la aplicación y las operaciones y conexiones a través de la red.

Marco Teórico

2.1.1 Aplicaciones

Los programas (informáticos) son patrones de reglas creados para dirigir los procesos computacionales que viven en los sistemas computacionales [7]. Las aplicaciones son programas que han sido diseñados expresamente para que el usuario final los use y manipule. Se llama aplicaciones móviles a las que funcionan en un teléfono inteligente, tableta u otro dispositivo móvil. Las aplicaciones pueden tener una miríada de usos y propósitos diferentes como entretener, mejorar el desempeño laboral, ayudar a aprender o configurar el sistema. En nuestro caso, el propósito de nuestra aplicación es coordinar la salida de una escuela. Podría decirse que la aplicación creada en este proyecto entra en «mejorar el desempeño laboral. Existen varios tipos de aplicaciones móviles: las nativas, las híbridas y las web.

Aplicación nativa

Las aplicaciones nativas son creadas para ser usadas en una plataforma o sistema específico. Por ejemplo, una aplicación nativa creada para el sistema operativo iOS no funcionará en un

dispositivo Android. Las aplicaciones nativas gozan de ciertas ventajas como una mejor integración con el sistema y sus API (Una interfaz de programación de aplicaciones así como un mejor rendimiento). En este proyecto se crearon dos aplicaciones nativas, una para iOS y otra para Android.

Aplicación web

Una aplicación web es una aplicación que no se instala en el dispositivo, sino que corre en un navegador como Google Chrome o Mozilla Firefox. Se crean con Javascript, HTML5 y CSS. Se ha encontrado que las aplicaciones web son menos usadas que aplicaciones instaladas en un contexto de dispositivos móviles [9].

Aplicación híbrida

A diferencia de las nativas, las aplicaciones híbridas pueden ser usadas en más de una plataforma (esto es, tanto en iOS, como en Android). A través de ciertas plataformas de desarrollo como Apache Cordova, es posible crear una sola aplicación que funcione en ambos sistemas operativos. Como solo hay que hacer una aplicación, el proceso de desarrollo es más rápido y sencillo. A pesar de esto, puede haber problemas en el desempeño (uso de memoria, velocidad de respuesta) de la aplicación e integración de la interfaz de usuario con el diseño estándar de la interfaz del sistema. Así como en el caso de las aplicaciones web, las aplicaciones híbridas se crean utilizando Javascript, HTML5 y CSS, pero a diferencia de ellas, sí se distribuyen a través de las tiendas de aplicaciones y pueden hacer uso de funciones del sistema operativo. A decir verdad, como la aplicación creada en este proyecto es relativamente simple, esto es, sin demasiadas partes, se podría haber creado una aplicación híbrida sin que las consecuencias negativas se dejaran ver.

2.1.2 Sistemas Operativos

Los programadores necesitan programar para sentirse vivos, por lo tanto, a los programas se les continúa añadiendo cosas y aparecen nuevas versiones de estos programas [10]. En el caso de los sistemas operativos que nos atañen, iOS y Android, este hecho implica la existencia, y uso en el mercado, de varias versiones del mismo sistema operativo a la vez. Es decir, algunas personas tienen instalada una versión actualizada de Android y otras tienen instalada otra versión anteriores y no tan recientes. Las versiones de un sistema operativo tienen diferencias de compatibilidad y de características. A la hora de elegir la versión de sistema operativo en la que nuestra aplicación funcionará, hay que tomar en cuenta varios factores.

Para empezar, los desarrolladores, o quienes se encarguen de la asistencia técnica, eventualmente dejarán de ofrecer esta ayuda a los usuarios de las versiones más antiguas así que desarrollar una aplicación apuntando a una versión antigua podría hacer que nuestra aplicación se volviera obsoleta prematuramente. Por otro lado, las versiones más nuevas de un sistema operativo tardan en ser adoptadas por los usuarios ya que cambiar de versión conlleva un costo de tiempo y memoria informática; así como el hecho de que, en dispositivos relativamente antiguos, no se puede siquiera instalar la última versión.

A todo esto, podemos añadir que las nuevas versiones suelen tener mejoras de seguridad, usabilidad y funciones y que, en última instancia, son el futuro de sus respectivas plataformas. La decisión que se tome deberá hacerse con el objetivo de la aplicación en mente, el cual suele tener que ver con los usuarios finales [11].

iOS

Las aplicaciones nativas para iOS se pueden crear usando los lenguajes de programación Objective-C o Swift, que son los lenguajes oficiales de Apple para iOS, así como otros lenguajes como Xojo[12], C# y Javascript. Las aplicaciones iOS son almacenadas en un archivo con extensión .ipa (iOS AppStore Package). Para realizar varias tareas y acceder a ciertos datos, por ejemplo, acceder a la localización del dispositivo, las aplicaciones tienen que pedir permiso al usuario. El nivel de opciones de permisos varía entre versiones de iOS.

Android

Las aplicaciones Android pueden ser escritas con Kotlin o Java, que son los lenguajes apoyados por Google, así como C# o C++. Las herramientas del toolkit de desarrollador Android (Android SKD) compilan el código, datos y archivos misceláneos y los convierten en un APK (Android package), esto es, un archivo con extensión .apk.

Los archivos APK contienen la aplicación Android y a través de ellos el sistema Android instala la aplicación [13].
Android cuenta con ciertas características de seguridad, por ejemplo: el sistema operativo es un sistema multiusuario y cada aplicación es un usuario diferente y las aplicaciones solo tienen acceso a los componentes que necesitan para funcionar. Para que una aplicación pueda acceder los servicios del sistema y compartir sus datos, tiene que pedir permiso al usuario. Algunos ejemplos de los tipos de datos que se podrían compartir son la localización y la conexión por Bluetooth. [13].

2.1.3 Hardware

Los teléfonos inteligentes son un tipo de teléfono móvil con mayores capacidades que otros teléfonos más básicos, en cuestión de software, acceso a internet y multimedia. Sus sistemas operativos son más avanzados y permiten la instalación de aplicaciones con diversas funciones. Tanto Android como iOS funcionan en otros dispositivos que no son teléfonos inteligentes como tabletas (que son como teléfonos grandes y, a veces, sin servicio de celular o llamadas), relojes inteligentes (básicamente un teléfono de pulsera) y muchos más[14].

2.1.4 Proceso de desarrollo

Al crear una aplicación hay muchas cosas que tener en cuenta como editores de texto, lenguajes de programación, procesos de diseño, bases de datos, división de responsabilidades, proceso de destrucción de errores, planeación, estándares a seguir, etc.

Ambientes de desarrollo

Los editores de texto son programas que sirven para modificar el contenido de un archivo de texto como puede ser un archivo con código fuente o con parámetros de configuración. Hay muchos editores de texto en uso actual. Algunos, como Emacs (1976) y Vim (1991) han existido por un largo tiempo; otros, como Visual Studio Code (2015), han sido lanzados por primera vez hace poco. Aunque hoy en día estos editores de texto tienen muchas funciones que permiten llevar a cabo en ellos, (prácticamente) todo el proceso de desarrollo [15], a veces se hace la distinción entre un IDE (Entorno de desarrollo integrado) y un editor de texto. Un IDE, además de permitir editar texto, suele tener herramientas de depuración y de construcción de programas. En programación móvil hay varios IDE disponibles para usar. Para Android hay varias opciones como Android Studio, e InteliJ IDEA. Para iOS, Xcode y Appcode. En este proyecto se usó Android Studio y Xcode, respectivamente.

Bases de datos

Existen bases de datos SQL y NoSQL. La diferencia principal es que las tablas son la estructura de las SQL. Firebase Real-time Database es una base de datos NoSQL con servicio en la nube que se integra fácilmente a una aplicación. Una misma instancia de base de datos se puede conectar con múltiples aplicaciones. Los datos son almacenados en una estructura JSON[16].

Procesos de diseño

Hay un estilo de diseño llamado bottom-up (de abajo arriba) en el que los programas son escritos como una sucesión de capas que funcionan como una especie de lenguaje para la capa de arriba [17]. También existen procesos de diseño como el ágil y de cascada.

2.2 Android

2.2.1 ¿Kotlin o Java?

Java es un lenguaje utilizado en muchos ámbitos de la industria computacional; es un lenguaje concurrente orientado a objetos, con una sintaxis parecida a C++ [18], bastante verboso, que funciona sobre una máquina virtual de Java (JVM). Así como Java, Kotlin es un lenguaje que funciona sobre una JVM; Kotlin es un lenguaje más moderno, menos verboso y más fácil de entender. Ambos lenguajes son apoyados oficialmente por Google para el desarrollo de aplicaciones Android. Algunos dicen que Kotlin es mejor que Java para desarrollar en Android [19]. Relativamente convencido por las explicaciones que varios autores dan, y dado que Swift se parece a Kotlin [20], decidí usar Kotlin.

2.3.2 Aplicaciones

Diseño

Para diseñar aplicaciones Android, hay un sistema de guías, herramientas y componentes llamado Material [21]. La naturaleza de Android

de estar disponibles para hardware diverso, dificulta encierta medida el proceso de diseño pues hay que tener en cuenta más diferencias entre dispositivos.

Estructura

Hay cuatro tipos de componentes que conforman las aplicaciones Android y por medio de los cuales se puede acceder a la aplicación [22]:
1. Actividades
2. Servicios
3. Receptores de emisión
4. Proveedores de contenido

Actividades

Una actividad provee la ventana en la que la aplicación dibuja la interfaz de usuario. Usualmente una actividad implementa una pantalla en la aplicación. La mayoría de aplicaciones tienen varias pantallas y, por consiguiente, contiene varias actividades. Por ejemplo, una aplicación puede tener una actividad para iniciar sesión, otra para mostrar una lista con información y otra para modificar esta información [23].

Servicios

Los servicios son componentes que ejecutan operaciones en segundo plano y no proveen una interfaz de usuario. Otros componentes pueden iniciar un servicio y este continuará en estado de ejecución, aunque el usuario cambie de aplicación. Hay tres tipos de servicios: de primer plano, que ejecutan operaciones visibles para el usuario; de segundo plano, cuyas operaciones, en cambio, no son notadas por el usuario; y de enlace, que están unidos a otros componentes usando bindService()[24].

Intent

Un intent es una descripción abstracta de una operación a realizar que sirve para activar actividades, servicios y receptores de emisión [25].

Archivo de manifiesto

El sistema debe conocer la existencia de un componente antes de activarlo. Todos los componentes de la aplicación deben ir declarados en el archivo de manifiesto: AndroidManifest.xml, que se encuentra en el directorio raíz10 del proyecto. Además, en este archivo se declaran los permisos que la aplicación pedirá al usuario, el nivel de API mínimo y otras cosas más [26].

2.3 iOS

2.3.1 ¿Objective-C o Swift?

Objective-C era el lenguaje oficial para desarrollar aplicaciones para dispositivos Apple. Swift es el sucesor de Objective-C y cuenta con numerosas mejoras y por lo tanto es el mejor lenguaje para comenzar a programar para iOS [27].

2.3.2 Aplicaciones

Estructura interna

La estructura interna de las aplicaciones iOS está basada en controladores de vista o View controllers. Cada controlador administra una parte de la interfaz de usuario así como las interacciones entre los datos y la interfaz [28].

Diseño

El diseño de aplicaciones para iOS sigue la guía establecida por Apple en sus Human Interface Guidelines. Esta guía menciona seis principios de diseño a seguir: integridad estética, esto es, la integración entre apariencia y función; consistencia: uso de elementos provistos por el sistema; manipulación directa (del contenido en pantalla); realimentación, es decir, reconocer las acciones del usuario y mostrar una respuesta; metáforas: usar metáforas de actividades cotidianas como pulsar un interruptor o arrastrar contenido; y control del usuario, que significa que lo que suceda en el uso de la aplicación sea lo que el usuario pretenda. [29]

UIApplication

Es una clase que maneja la elección inicial de rutas para los eventos del usuario. Utiliza un objeto delegate que va de acuerdo al protocolo UIApplicationDelegate para manejar las interacciones entre la aplicación y el sistema [30].

Objetivo general

El objetivo principal de este proyecto es presentar las diferencias y similitudes en el desarrollo de una aplicación igual para iOS y Android. Un objetivo adjunto es crear dos

aplicaciones cuyo diseño esté enfocado en ayudar a las escuelas a tener un proceso de cese de las actividades diarias más eficaz

Metodología

3.1 Desarrollo de las aplicaciones

El propósito de esta aplicación es ayudar a coordinar las salidas de los estudiantes de primaria. En la base de datos de Firebase se guardan los datos de los estudiantes como el grado, el grupo, el id del tutor y el estatus de salida o entrada (hay 4 opciones: clases, recoger, camino y listo). En esta versión, que es un prototipo, la autenticación por usuario no está implementada.

3.1.1 Preparación de recursos

Los recursos que se utilizaron en este proyecto fueron los dos entornos de desarrollo integrado (IDE), Android Studio y Xcode; la base de datos, Firebase Realtime Database; y el servicio de control de versiones, GitHub. Para crear la base de datos se crea un nuevo proyecto desde la consola de Firebase (habiendo creado una cuenta de usuario previamente).

Android

Al crear un proyecto en Android Studio[31] aparecen varias opciones de configuración; se puede elegir el tipo de proyecto para que Android Studio incluya automáticamente recursos y código relevantes. También se escoge la localización del archivo del proyecto, el lenguaje: Kotlin o Java, el nivel mínimo de API y la opción "This project will support instant apps". Una vez creado el proyecto, en la barra de herramienta encontramos las opciones para integrar la base de datos (Tools > Firebase . . . Realtime Database > Save and Retrieve Data) y el control de versiones (VCS > Import into Version Control > Share Project on GitHub) a nuestra aplicación.

Para correr una aplicación necesitamos un dispositivo Android o instalar un dispositivo virtual. Android Studio cuenta con un administrador de dispositivos virtuales (AVD Manager). En ocasiones, el uso de emuladores representa un problema pues la aplicación podría correr de manera no óptima, por lo que es preferible utilizar un dispositivo físico.

iOS

En el caso de iOS, también hay varias opciones de configuración al crear un proyecto. Para añadir dependencias al proyecto se instala CocoaPods. En Xcode, la conexión a Firebase y GitHub se hizo de manera manual. Desde el proyecto de Firebase en el navegador, se selecciona la opción de añadir una aplicación iOS al proyecto de Firebase. Así, descargamos un archivo GoogleService-Info.plist y lo añadimos en el directorio raíz de nuestro proyecto en Xcode. También se usa CocoaPods para administrar las dependencias de Firebase. En cuanto a GitHub, se crea un repositorio en GitHub y se clona desde Xcode. A diferencia de Android Studio, Xcode ya viene con cierta cantidad de dispositivos simulados (y se pueden añadir otros más a los que vienen).

3.1.2 Estructura del proyecto

Android

La aplicación Android tiene 4 archivos Kotlin:

- ☐ MainActivity
- ☐ TutorActivity
- ☐ Estudiante
- ☐ EstudianteAdapter

y 4 archivos de XML que contienen layouts:

- ☐ activity_main.xml
- ☐ estudiante_camino.xml
- ☐ estudiantes.xml
- ☐ tutor.xmlz

Todos los recursos se almacenan en una carpeta llamada res, y entre estos recursos están incluidos los archivos XML (Extensible Markup Language) que describen interfaces de usuario. La clase MainActivity es la pantalla principal de la aplicación; presenta la interfaz descrita en activity_main.xml. EstudianteAdapter sirve para mostrar varias filas, descritas en estudiantes.xml, en la lista (ListView) presente en MainActivity/activity_main. También presenta un dialogo de confirmación de cambio de estatus (estudiante_camino.xml) al hacer clic en el botón de la fila. La clase Estudiante describe la estructura de la instancia de un estudiante en la base de datos. La clase TutorActivity, a la que se accede por medio de un intent desde MainActivity, permite modificar el estatus de los estudiantes con base a cierto id de tutor; su interfaz es tutor.xml.

iOS

En los archivos Swift que ya vienen por defecto, se añadió la configuración de Firebase (en AppDelegate). A la aplicación se le añadió un archivo Swift llamado tableView-Controller donde se maneja la presentación/invocación de la tabla/lista de estudiantes.Los storyboards son tipos de archivos editables visualmente por medio del InterfaceBuilder que contienen escenas, controladores de vista y las relaciones entre ellos.[32][33]

En el archivo Main.storyboard, está descrita la interfaz de tres viewControllers que muestran:

- El formulario de añadir estudiante y acceso a los otros viewControllers.
- La tabla con los estudiantes.
- La modificación del estatus con base a cierto id de tutor.

También, en el storyboard mismo (y no en el código), se define la presentación de un viewController al hacer clic en un botón.

3.2 Publicación de aplicaciones

Hay un programa (no informático) llamado Apple Developer Program al que es necesario unirnos si queremos publicar nuestra aplicación en la App Store. Sin embargo, es necesario verificar nuestra identidad proveyendo nuestro nombre, número telefónico, dirección y fotografía de identificación personal. En el caso Android, las aplicaciones se publican a través de Google Play Console; uno de los pasos en el registro es pagar 25 dólares.[34] También existe la opción de publicar en la tienda de aplicaciones de Amazon.

Conclusion

Hay ciertas cosas que son más intuitivas al programar para iOS ya que pertenece al ecosistema Apple y este está bien integrado. Mientras que en Xcode el diseño de la interfaz fue relativamente sencillo, el resto del proceso fue más claro en Android.

En cuanto a los lenguajes, fue agradable usar tanto Swift como Kotlin ya que su sintaxis no es muy verbosa y es fácil de leer. Son lenguajes relativamente fáciles de aprender y más sencillos y modernos que sus contrapartes (Objective-C y Java).

El diseño de las aplicaciones iOS, que se basa en controladores de vista, se considera un poco más sencillo que el de Android, cuyo fundamento son las actividades.

El uso de Firebase es relativamente fácil de integrar con ambas plataformas y es
una buena opción al decidir qué base de datos usar Studio.

Con respecto a la publicación de las aplicaciones en las tiendas respectivas, es más sencillo publicar en la Play Store de Google que en la App Store de Apple. Además, las aplicaciones Android también se pueden publicar en la Amazon Appstore. Si queremos que nuestra aplicación esté disponible para el mayor número posible de personas, Android es la opción a elegir, pues tiene más usuarios. Sin embargo, la facilidad para publicar en la tienda conlleva que haya muchas aplicaciones y esto reduce la posibilidad de que nuestra aplicación sea vista. Por último, si se desea alcanzar a usuarios de ambas plataformas y ahorrar costos y esfuerzos (de desarrollo, no de mantenimiento), quizá sería conveniente desarrollar una aplicación híbrida. Aunque se perdería el uso de Swift/Kotlin, que son buenos lenguajes para utilizar.

Referencias

[1] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, Operating Systems: Three Easy
Pieces, 1st ed. Arpaci-Dusseau Books, August 2018.

[2] StatCounter, “Mobile operating system market share mexico,” Marzo 2020.
[Online]. Available: https://gs.statcounter.com/os-market-share/mobile/mexico

[3] A. J. Perlis, “Special feature: Epigrams on programming,” SIG-
PLAN Not., vol. 17, no. 9, p. 7–13, Sep. 1982. [Online]. Available:

https://doi.org/10.1145/947955.1083808

[4] P. Norvig, “Teach yourself programming in ten years,” Visitado en 2020.
[Online]. Available: https://norvig.com/21-days.html

[5] H. F. Silver, Compare & Contrast: Teaching Comparative Thinking to Strengthen

Student Learning. Alexandria, Va. : Association for Supervision and Curriculum Development, 2010.

[6] P. Norvig, "Python for lisp programmers," Mayo 2000. [Online]. Available: https://norvig.com/python-lisp.html

[7] H. Abelson and G. J. Sussman, Structure and Interpretation of Computer Programs, 2nd ed. Cambridge, MA, USA: MIT Press, 1996.

[8] L. I. Institute, "22 u.s. code § 8541.definitions," Abril 2020. [Online]. Available: https://www.law.cornell.edu/uscode/text/22/8541

[9] C. Tossell, P. Kortum, A. Rahmati, C. Shepard, and L. Zhong, "Characterizing web use on smartphones," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 2769–2778. [Online]. Available: https://doi.org/10.1145/2207676.2208676

[10] E. S. Raymond, "The jargon file." [Online]. Available: http://www.catb.org/jargon/html/C/creeping-featurism.html

[11] P. Graham, "Be good," Abril 2008. [Online]. Available: http://paulgraham.com/good.html

[12] I. Xojo, "Xojo, inc. announces xojo 2014 release 3; includes support for developing native ios applications," Diciembre 2014. [Online]. Available: https://www.xojo.com/company/news/2014r3.php

[13] A. O. S. Project, "Application fundamentals," Diciembre 2019. [Online]. Available: https://developer.android.com/guide/components/fundamentals

[14] Wikipedia contributors, "Smartphone," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Smartphone

[15] jcs, "https://irreal.org/blog/?p=1692," Febrero 2013. [Online]. Available: https://irreal.org/blog/?p=1692

[16] Google., "Firebase realtime database." [Online]. Available: https://firebase.google.com/docs/database

[17] P. Graham, On LISP: Advanced Techniques for Common LISP. USA: Prentice-Hall, Inc., 1993.

[18] P. Van Roy and S. Haridi, Concepts, Techniques, and Models of Computer Programming, 1st ed. The MIT Press, 2004.

[19] S. Yegge, "Why kotlin is better than whatever dumb language you're using," Mayo 2017. [Online]. Available: http://steve-yegge.blogspot.com/2017/05/why-kotlin-is-better-than-whatever-dumb.html

[20] G. Mechling, "Swift is like kotlin," Mayo 2017. [Online]. Available: http://nilhcem.com/swift-is-like-kotlin/

[21] Google. [Online]. Available: https://material.io/design/

[22] A. O. S. Project, "Application fundamentals; components," Diciembre 2019. [Online]. Available: https://developer.android.com/guide/components/fundamentals#Components

[23] ——, "Introduction to activities," Diciembre 2019. [Online]. Available: https://developer.android.com/guide/components/activities/intro-activities

[24] ——, "Services overview," Marzo 2020. [Online]. Available: https://developer.android.com/guide/components/services

[25] ——, "Intent," Abril 2020. [Online]. Available: https://developer.android.com/reference/android/content/Intent

[26] ——, "Application fundamentals; manifest," Diciembre 2019. [Online]. Available: https://developer.android.com/guide/components/fundamentals#Manifest

[27] A. Inc., "Swift." [Online]. Available: https://developer.apple.com/swift/

[28] ——, "The role of view controllers." [Online]. Available: https://developer.apple.com/library/archive/featuredarticles/ViewControllerPGforiPhoneOS/index.html

[29] ——, "ios design themes." [Online]. Avail-

able: https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/

[30] ——, “Uiapplication.” [Online]. Available: https://developer.apple.com/documentation/uikit/uiapplication

[31] A. O. S. Project, “Android studio.” [Online]. Available: https://developer.android.com/studio

[32] A. Inc., “About storyboards, scenes, and connections.” [Online]. Available: https://help.apple.com/xcode/mac/current/#/dev62c993289

[33] ——, “Interface builder built-in.” [Online]. Available: https://developer.apple.com/xcode/interface-builder/

[34] Google. [Online]. Available: https://play.google.com/apps/publish/signup/