

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

—o0o—



LẬP TRÌNH ANDROID

ĐỒ ÁN II

Chuyên ngành: TOÁN TIN

Chuyên sâu: Lập trình Android với React Native

Giảng viên hướng dẫn : **TS. Nguyễn Huy Trường**

Nhóm sinh viên thực hiện : **Nguyễn Bùi Nam Trường**

MSSV : **20185418**

HÀ NỘI, tháng ../2022

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**1. Mục đích và nội dung của đồ án:**

-
-
-

2. Kết quả đạt được

-
-
-

3. Ý thức làm việc của sinh viên:

-
-
-

Hà Nội, ngày .. tháng .. năm 2022

Giảng viên

TS. Nguyễn Huy Trường

Lời nói đầu

Ngày nay, cùng với sự phát triển mạnh mẽ của lĩnh vực Công nghệ thông tin, có rất nhiều công ty công nghệ lớn nhỏ được sinh ra. Các công ty đó vẫn liên tục phát triển và cải tiến để phục vụ cuộc sống con người và xã hội. Một trong những sản phẩm phổ biến và dần trở nên không thể thiếu đối với người dùng là điện thoại di động. Tính đến tháng 6/2020, trong thế giới rộng lớn của điện thoại di động, hệ điều hành Android đã chiếm 74.4% thị phần toàn cầu và 60.8% thị phần Việt Nam. Với sức bao phủ và ảnh hưởng lớn như vậy, việc tìm hiểu về hệ điều hành này là qua trọng đối với những người học tập và làm việc trong lĩnh vực công nghệ. Cũng như sự phổ biến của Android trong thế giới điện thoại di động, các phần mềm được xây dựng dựa trên công cụ React Native cũng dần trở nên phổ biến và được ưa chuộng đối với các nhà phát triển ứng dụng. Sau một thời gian học tập tại trường Đại học Bách Khoa Hà Nội, viện Toán ứng dụng và Tin học, em đã tích lũy được vốn kiến thức nhất định và đang thực hiện học phần Đồ Án II. Được sự đồng ý của nhà trường và các thầy cô trong viện, em được giao đề tài cho đồ án: *"Tìm hiểu về Lập trình Android với React Native"*. Bài Đồ án này gồm ba chương:

Chương 1: Giới thiệu về Android.

Chương 2: Giới thiệu về React Native.

Chương 2: Xây dựng ứng dụng Android với React Native.

Bằng sự cố gắng nỗ lực của bản thân và đặc biệt là sự giúp đỡ tận tình, chu đáo của thầy Nguyễn Huy Trường, em đã hoàn thành đồ án đúng thời hạn. Do thời gian làm đồ án có hạn và trình độ còn nhiều hạn chế nên không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến của các thầy cô cũng như là của các bạn sinh viên để bài đồ án này hoàn thiện hơn nữa. Em xin chân thành cảm ơn thầy Nguyễn Huy Trường, các thầy cô trong viện Toán ứng dụng và Tin học trường Đại

học Bách Khoa Hà Nội đã tạo điều kiện giúp đỡ em trong thời gian qua.

Hà Nội, ngày .. tháng .. năm 2022

Sinh viên thực hiện

Nguyễn Bùi Nam Trường

Mục lục

Lời nói đầu	2
1 Giới thiệu về Android	5
1.1 Khái niệm về Android	5
1.2 Kiến trúc của Android	6
2 Giới thiệu về React Native	7
2.1 Giới thiệu	7
2.2 Lịch sử phát triển	8
2.3 Lý do React Native được ưa chuộng	8
2.4 Nguyên lý hoạt động	9
2.5 Một số ứng dụng sử dụng react native	10
3 Xây dựng ứng dụng Android với React Native	10
3.1 Thiết lập môi trường	10
3.1.1 NodeJS	11
3.1.2 JDK	12
3.1.3 Android Studio	13
3.2 Xây dựng chương trình đầu tiên	14
3.2.1 Chạy trên thiết bị vật lý	17
3.2.2 Chạy trên máy ảo	18
3.3 Các khái niệm, thành phần	19
3.3.1 Components	19
3.3.2 JSX	23

3.3.3	Props	25
3.3.4	State	26
3.3.5	Style	27
3.3.6	Vòng đời của Component	27
3.4	Một số thư viện phổ biến	29
3.4.1	React và ReactNative	29
3.4.2	PropTypes	32
3.4.3	Thư viện chứa Basic Component theo chuẩn thiết kế	32
3.4.4	Axios	33
3.5	Phát hành ứng dụng	33
Kết luận		35
Tài liệu tham khảo		37

1 Giới thiệu về Android

1.1 Khái niệm về Android

Trước hết, Hệ điều hành (Operating System - OS) là phần mềm hệ thống quản lý phần cứng máy tính, phần mềm và cung cấp các dịch vụ chung cho các chương trình máy tính. Android là hệ điều hành dựa trên mã nguồn mở Linux OS cho các thiết bị di động và các thiết bị thông minh như máy tính bảng, laptop, netbook, smartbook, TV thông minh(Google TV),... Hiện nay, một số hệ điều hành cho thiết bị di động: Android, OS X (iPhone), Windows Mobile, Symbian,... Tuy nhiên, hệ điều hành được sử dụng nhiều nhất là Android và OS X.

Ngay từ khi ra mắt, Android đã gây ấn tượng mạnh khi đây là đứa con của Google sử dụng giấy phép mã nguồn mở. Android là sản phẩm kết tinh từ ý tưởng của Khối Liên minh thiết bị cầm tay mở do Google dẫn đầu, gồm 34 thành viên với các công ty hàng đầu về công nghệ và di động toàn cầu như Qualcomm, Intel, Motorola, Texas Instruments và LG Electronics, các nhà mạng như T-Mobile, Sprint Nextel, NTT DoCoMo và China Mobile.

Với đặc tính "mở", Android cho phép các nhà phát triển có thể sử dụng miễn phí bộ Kit Android Software Development để xây dựng các ứng dụng của mình. Ví dụ, một ứng dụng có thể gọi bất kỳ chức năng lõi của điện thoại như thực hiện gọi, gửi tin nhắn, sử dụng máy ảnh,... Điều này thực sự thu hút các nhà phát triển sáng tạo ra các phần mềm hấp dẫn, từ đó tạo ra một cộng đồng phát triển lớn - là nguyên nhân đến sự phổ biến của Android như hiện nay.

1.2 Kiến trúc của Android

Để bắt đầu lập trình với Android, chúng ta cần nghiên cứu qua bản thân HĐH Android, chúng ta không cần hiểu quá chi tiết, nhưng trước hết là cần có cái nhìn chung và toàn diện nhất về Android.

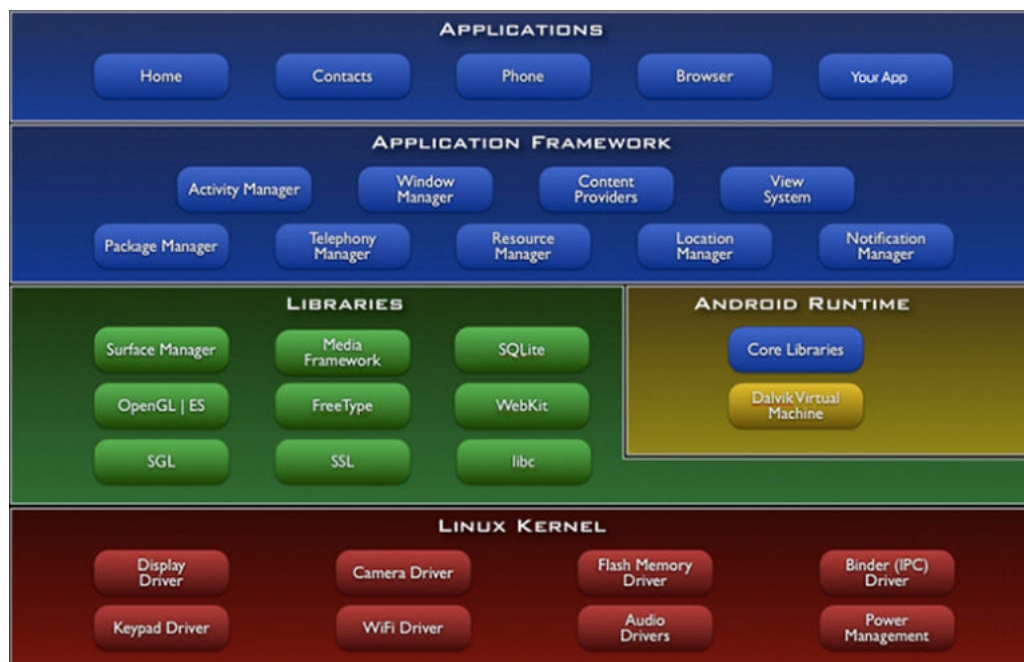
Android Platform bao gồm đầy đủ các tính năng HĐH Android, các ứng dụng và các tầng trung gian để nhà phát triển có thể mở rộng, tùy chỉnh thêm theo nhu cầu của họ. Có 4 tầng cơ bản trong HĐH Android: Linux Kernel, Native Libraries, Android Runtime, Application Framework,... Các tầng làm việc có sự liên kết với nhau.

1. *Tầng Linux Kernel*: Đây là tầng nhân của HĐH Android, Linux Kernel giúp hệ điều hành có thể giao tiếp với phần cứng của thiết bị như: camera, USB, Wifi, Bluetooth, Display, Power Management,... Linux Kernel chịu trách nhiệm cho các trình điều khiển thiết bị, quản lý nguồn điện, quản lý bộ nhớ, quản lý thiết bị và truy cập tài nguyên. Kernel hoạt động như một lớp trừu tượng giữa phần cứng và phần mềm còn lại của hệ thống.
2. *Native Libraries*: Native Libraries là tập hợp của nhiều thư viện như WebKit, OpenGL, FreeType, SQLite, Media, SSL,...
3. *Tầng Android Runtime*: Android Runtime cung cấp một thành phần quan trọng được gọi là DVM (Dalvik Virtual Machine - máy ảo) có trách nhiệm chạy ứng dụng android. DVM thực thi các file có định dạng .dex (Dalvik Executable), định dạng này là định dạng được tối ưu hóa về bộ nhớ.

DVM có phần tương tự như JVM (Java Virtual Machine) nhưng được tối ưu hóa cho các thiết bị di động như tiêu thụ ít bộ nhớ hơn và tăng hiệu suất hoạt động tốt hơn.
4. *Tầng Application Framework*: Application Framework bao gồm tập hợp những

API cho phép các nhà phát triển ứng dụng được phép sử dụng các dịch vụ này trong các ứng dụng của họ.

Ở góc độ người dùng, ta có thêm tầng Application - là các ứng dụng do nhà phát triển ứng dụng viết. Dưới đây là sơ đồ tương quan liên kết giữa các tầng:



Hình 1: Sơ đồ tương quan liên kết giữa các tầng

2 Giới thiệu về React Native

2.1 Giới thiệu

React Native là một Framework, do công ty Công nghệ Meta (trước đây là Facebook) phát triển, nhằm giải quyết vấn đề về hiệu năng và việc phải sử dụng nhiều ngôn ngữ native trên nền tảng di động.

React Native cho phép xây dựng và phát triển ứng dụng native đa nền tảng một

cách dễ dàng, khác với HTML5 App, Mobile Web App, Hybrid App. Mục đích khi tạo ra React Native là khắc phục các điểm yếu của ứng dụng web, giúp cho nhà lập trình tiết kiệm thời gian, công sức bởi sự hỗ trợ đắc lực từ JavaScript.

React Native là một trong những framework sử dụng cấu hình thiết kế tương tự như React. Có thể nói, biết về React là biết được 80% về React Native. Tuy nhiên, đồ án này sẽ trình bày đủ các thông tin cả về React.

2.2 Lịch sử phát triển

Năm 2012 Mark Zuckerberg đã phát biểu: "Sai lầm lớn nhất của chúng tôi khi làm công ty là dựa trên quá nhiều HTML hơn là môi trường phát triển gốc". Ông hứa rằng Facebook sẽ sớm cung cấp trải nghiệm di động tốt hơn.

Kỹ sư Jordan Walke tại Facebook đã tìm ra cách xây dựng các thành phần UI cho iOS bằng một luồng JavaScript. Họ quyết định tổ chức cuộc thi Hackathon để hoàn thiện nguyên mẫu hệ thống để có thể xây dựng các ứng dụng di động gốc (native app) bằng công nghệ này.

Sau nhiều tháng phát triển, Facebook đã phát hành phiên bản đầu tiên cho React Native vào năm 2015. Trong một cuộc hội thảo công nghệ, Christopher Chedeau cho biết Facebook đã sử dụng React Native trong phát triển ứng dụng nhóm và ứng dụng quản lý quảng cáo của họ.

Với cộng đồng phát triển ứng dụng lớn, ngày nay React Native trở thành framework được ưa chuộng trong việc xây dựng ứng dụng native.

2.3 Lý do React Native được ưa chuộng

Để biết tại sao React Native lại được ưa chuộng, ta đề cập đến các ứng dụng phổ biến trước đây là Hybrid Apps. Hybrid App được hiểu là ứng dụng được xây dựng

dựa trên các công nghệ web phổ biến là CSS, Javascript, HTML. Như vậy, ứng dụng được xây dựng lớn, cần phát triển lâu dài thì sẽ không đảm bảo được hiệu năng.

Trong khi đó, React Native lại có những ưu điểm sau:

1. Tiết kiệm thời gian học: Việc học từng loại ngôn ngữ cho từng nền tảng thường rất khó và mất nhiều thời gian. Tuy nhiên với React Native, lập trình viên chỉ cần học duy nhất một bộ công cụ.
2. Tái sử dụng code: Trong lập trình phần mềm, React Native là công cụ tái sử dụng code hiệu quả nhất mang lại các lợi thế như duy trì ít code, tận dụng tốt nguồn nhân lực,...
3. Hot reloading: Khi phát triển ứng dụng, nhà phát triển không tốn quá nhiều thời gian để tổng hợp app mỗi khi có sự thay đổi mà chỉ cần làm mới app trong thiết bị hoặc giả lập

Chính vì lý do trên mà hiện nay, React Native đang dần trở thành lựa chọn số một cho công việc xây dựng app của hầu hết các công ty lớn. Cũng từ việc được ưa chuộng khiến cho cộng đồng phát triển lớn, từ đó tạo thành vòng lặp khiến các nhà phát triển mới tiếp cận đều chọn React Native.

2.4 Nguyên lý hoạt động

Về cơ bản, React Native hoạt động bằng cách tích hợp cho ứng dụng đi động 2 thread là JS thread và Main thread.

1. *Main thread*: giữ vai trò cập nhật giao diện người dùng UI và xử lý các tương tác của người dùng ngay sau đó.
2. *JS thread*: thực thi và xử lý các code JavaScript.

Hai thread này hoạt động độc lập với nhau và giao tiếp qua một cầu nối trung gian.

2.5 Một số ứng dụng sử dụng react native

Với việc được ưa chuộng và có cộng đồng phát triển lớn, thế giới càng ngày càng có nhiều ứng dụng sử dụng React Native ra đời. Một số ứng dụng có thể kể đến như sau:

1. *Facebook*: là công ty phát triển React Native, sau khi phát triển xong framework này, Meta đã chuyển đổi tính năng Event Dashboard cho iOS sang React Native để kiểm tra hiệu suất ứng dụng, từ đó cắt giảm thời gian tìm hiểu thị trường đi một nửa
2. *Facebook Ads*: Đến thời điểm hiện tại, tất cả ứng dụng quảng cáo trên Facebook đều được sử dụng React Native.
3. *Instagram*: Sau khi được Meta mua lại, ứng dụng này cũng được chuyển đổi sử dụng React Native, cụ thể chế độ Push Notifications đã được triển khai dưới dạng WebView và không yêu cầu xây dựng cơ sở hạ tầng Navigation vì UI khá đơn giản.

3 Xây dựng ứng dụng Android với React Native

3.1 Thiết lập môi trường

Để bắt đầu xây dựng một ứng dụng React Native, ta cần cài đặt môi trường lập trình trên thiết bị của mình. Cụ thể cần: NodeJS, JDK, Android Studio.

3.1.1 NodeJS

a. Tổng quan

NodeJS là một nền tảng (platform) cung cấp môi trường runtime chạy JavaScript, được sử dụng để chạy các ứng dụng web bên ngoài client. Nền tảng này được phát triển bởi Ryan Dahl vào năm 2009, được xem là một giải pháp hoàn hảo cho các ứng dụng sử dụng nhiều dữ liệu nhờ vào mô hình hướng sự kiện (event-driven) không đồng bộ. NodeJS cung cấp các package để xây dựng ứng dụng React Native.

b. Cài đặt

Link cài đặt: <https://nodejs.org/en/download/>

Có thể cài bằng các dòng lệnh thông qua terminal (phụ thuộc vào hệ điều hành của thiết bị):

- Windows: `choco install -y nodejs-lts`
- MacOS: `brew install node` | `brew install watchman`
- Linux: `apk add nodejs npm`

Sau khi cài đặt xong, có thể kiểm tra lại bằng cách chạy dòng lệnh trong terminal:

```
C:\Users\admin>node -v
v16.18.0

C:\Users\admin>npm -v
8.19.2

C:\Users\admin>npx -v
8.19.2
```

Hình 2: Cài đặt NodeJs

3.1.2 JDK

a. Tổng quan

JDK (Java Development Kit) là một trong ba gói công nghệ cốt lõi được sử dụng trong lập trình Java: JVM (Máy ảo Java - Java Virtual Machine), JRE (Java Runtime Enviroment - Môi trường Java Runtime), JDK.

Có thể định nghĩa JDK theo 2 cách sau:

- *Định nghĩa chuyển ngành*: JDK là một hệ tiêu chuẩn trong việc triển khai nền tảng Java, bao gồm các trình thông dịch và thư viện lớp.
- *Định nghĩa thông thường*: JDK là gói phần mềm bạn tải xuống để tạo các ứng dụng dựa trên Java.

Như đã trình bày ở phần kiến trúc Android, tầng Android Runtime cần trình biên dịch Java để compile ra các lớp.

b. Cài đặt

Link cài đặt: <https://www.oracle.com/eg/java/technologies/downloads/>

Có thể cài bằng các dòng lệnh thông qua terminal (phụ thuộc vào hệ điều hành của thiết bị):

- Windows: `choco install -y microsoft-openjdk11`
- MacOS: `brew install java`
- Linux: `sudo apt install default-jdk`

Sau khi cài đặt xong, có thể kiểm tra lại bằng cách chạy dòng lệnh trong terminal:

```
C:\Users\admin>java --version
openjdk 11.0.17 2022-10-18
OpenJDK Runtime Environment OpenLogic-OpenJDK (build 11.0.17+8-adhoc..jdk11u)
OpenJDK 64-Bit Server VM OpenLogic-OpenJDK (build 11.0.17+8-adhoc..jdk11u, mixed mode)
```

Hình 3: Cài đặt JDK

3.1.3 Android Studio

a. Tổng quan

Android Studio là Môi trường phát triển tích hợp (IDE) chính thức để phát triển ứng dụng Android.

Nhờ các công cụ cho nhà phát triển tích hợp trình soạn thảo mạnh mẽ của IntelliJ IDEA, Android Studio cung cấp các tính năng giúp hỗ trợ và nâng cao năng suất trong quá trình phát triển ứng dụng Android:

- Hệ thống xây dựng linh hoạt dựa trên Gradle.
- Trình mô phỏng nhanh và có nhiều tính năng.
- Môi trường hợp nhất để phát triển các thiết bị Android.
- Tự cập nhật thay đổi ứng dụng mà đang chạy mà không cần khởi động lại.
- Hỗ trợ tự động chạy code template và code snippet.
- Tích hợp GitHub để tạo môi trường làm việc hiệu quả.

b. Cài đặt

Link cài đặt: <https://developer.android.com/studio>

Trong quá trình cài đặt Android Studio cần đảm bảo có các mục được chọn:

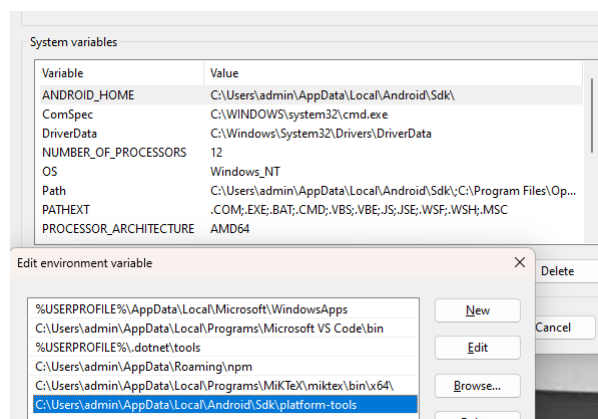
- Android SDK.
- Android SDK Platform.

- Android Virtual Device.

Sau khi cài đặt Android Studio, cần cấu hình Android SDK phù hợp để chạy React Native (theo yêu cầu được cung cấp trong tài liệu). Cụ thể, trong Android Studio, chọn "Appearance & Behavior" -> "System setting" -> "Android SDK". Khi đó cần cài các gói:

- SDK Platform: Android SDK Platform 33, Intel x86 Atom_64 System Image.
- SDK Tools: Android SDK Build-Tools 33.0.0.

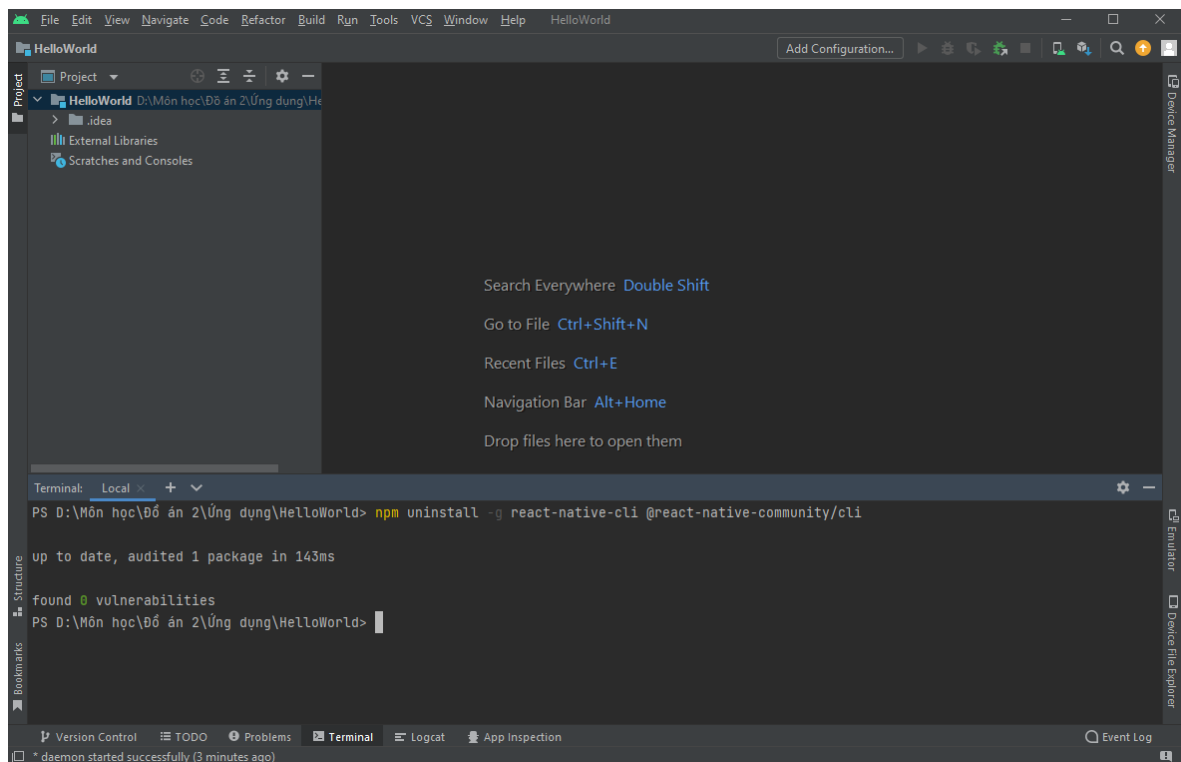
Một yêu cầu khác, các công cụ React Native yêu cầu biến môi trường được thiết lập để code. Cụ thể, cần cấu hình biến ANDROID_HOME cho đường dẫn SDK Android và thêm đường dẫn SDK Android Platform vừa cài đặt trên (cụ thể phụ thuộc vào hệ điều hành của thiết bị).



Hình 4: Cài đặt biến môi trường ANDROID_HOME

3.2 Xây dựng chương trình đầu tiên

Nếu trước đó thiết bị đã cài đặt gói react-native-cli toàn cầu, chúng ta cần xóa gói này vì gói đó có thể gây ra các sự cố không mong muốn:



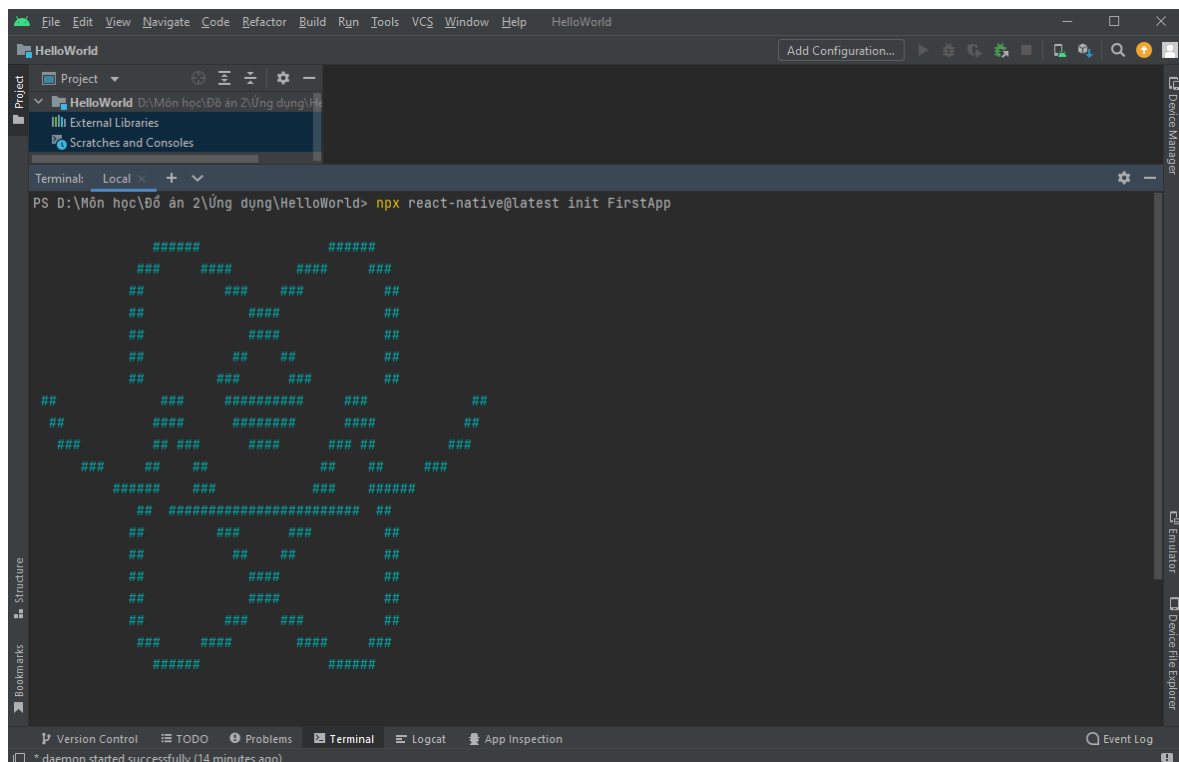
Hình 5: Xoá gói React Native CLI

Để bắt đầu xây dựng chương trình, ta tạo thư mục cho project này. Cụ thể, tôi đặt "HelloWorld". Sau đó mở thư mục đó bằng Android Studio, chạy dòng lệnh sau trên terminal:

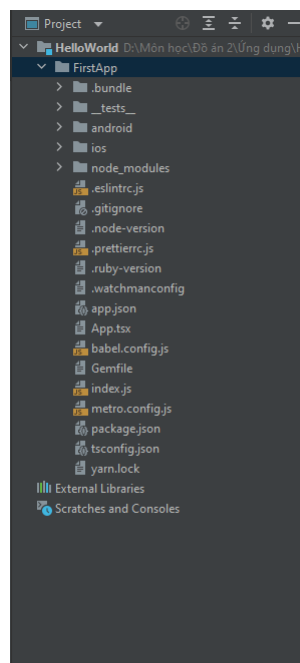
Cụ thể, "npx" là công cụ CLI do NodeJs cung cấp để cài đặt và quản lý các thành phần phụ thuộc npm. Ta sử dụng gói react-native để khởi tạo ứng dụng FirstApp. Sau khi chạy dòng lệnh trên, quan sát project, ta thấy thư mục FirstApp và các file trong đó:

Công việc tiếp theo là chạy ứng dụng FirstApp đã khởi tạo trên. Có 2 cách để build và chạy ứng dụng react native trong quá trình xây dựng:

- Chạy trên thiết bị vật lý
- Chạy trên máy ảo



Hình 6: Khởi tạo ứng dụng React Native



Hình 7: Dữ liệu FirstApp

3.2.1 Chạy trên thiết bị vật lý

a. *Bật gỡ lỗi trên thiết bị vật lý*

Để bật gỡ lỗi trên thiết bị Android của bạn, ta cần thực hiện các bước:

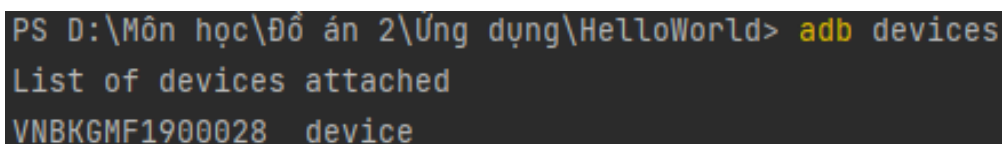
- Bật chế độ nhà phát triển
- Bật gỡ lỗi USB

Cụ thể cách bật chế độ nhà phát triển sẽ khác nhau tùy thuộc vào thiết bị, có thể tìm kiếm cách bật trên trình duyệt. Tuy nhiên để bật chế độ này có thể tổng quát:

Cài đặt → **Giới thiệu về điện thoại** → **Thông tin phần mềm** rồi nhấn Build number bảy lần vào hàng ở dưới cùng; sau đó, quay lại **Cài đặt** → **Tùy chọn nhà phát triển** để bật "Gỡ lỗi USB".

b. *Kết nối với thiết bị*

Tiếp theo cần kết nối thiết bị Android với thiết bị phát triển ứng dụng qua USB. Sau đó, kiểm tra lại thiết bị đã bật gỡ lỗi USB chưa bằng cách chạy dòng lệnh "adb devices".



```
PS D:\Môn học\Đồ án 2\Ứng dụng\HelloWorld> adb devices
List of devices attached
VNBKGMF1900028    device
```

Hình 8: Kiểm tra kết nối thiết bị

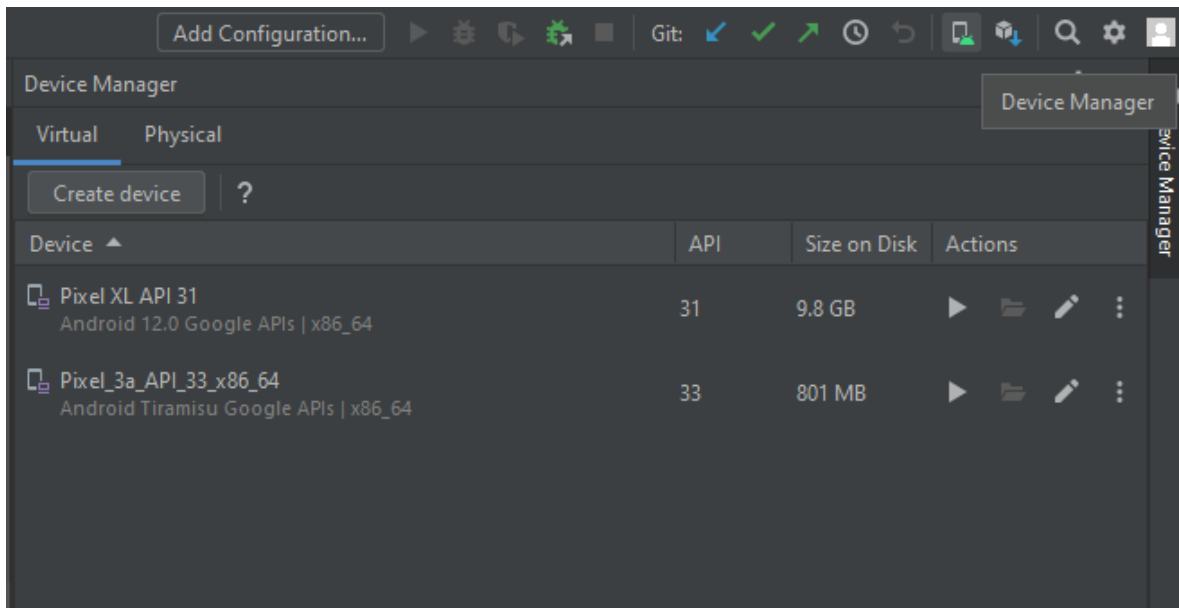
c. *Chạy ứng dụng trên thiết bị*

Để chạy ứng dụng, ta di chuyển đến FirstApp và chạy dòng lệnh sau: npx react-native run-android.

3.2.2 Chạy trên máy ảo

a. Tạo AVD mới

Trước hết ta cần tạo AVD mới bằng Android Studio. Ta vào mục **Device Manager** → **Create device** Sau đó, Android Studio hiển thị giao diện cấu hình thiết



Hình 9: Tạo thiết bị ảo trên Android Studio

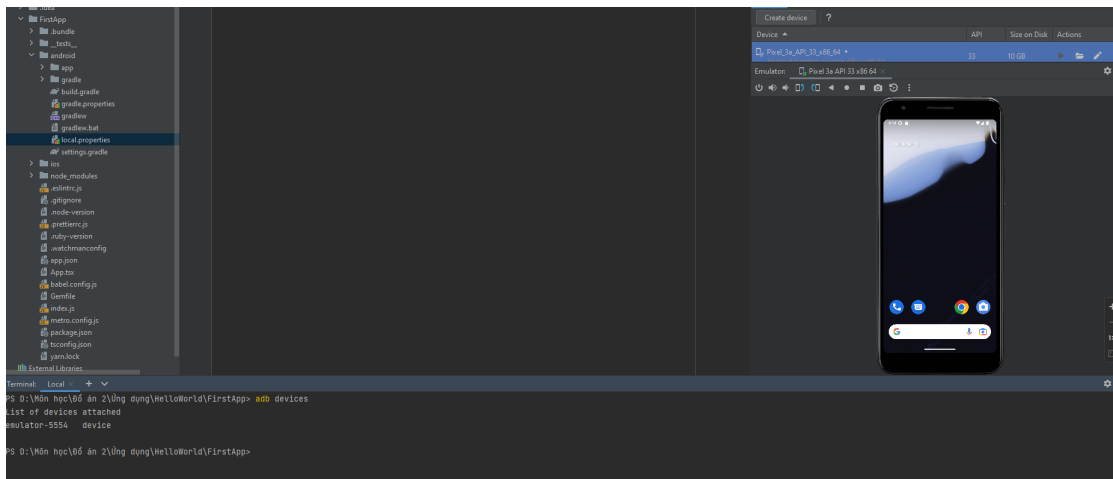
bị, ta chọn bất kỳ Điện thoại nào từ danh sách và nhấp vào **Next**, sau đó chọn hình ảnh **Tiramisu** API Cấp 33. Tiếp theo chọn **Finish** để tạo thiết bị ảo.

b. Kiểm tra kết nối

Sau khi khởi tạo, ta mở thiết bị ảo trên Android Studio và kiểm tra (giống chạy trên thiết bị vật lý):

c. Chạy ứng dụng trên AVD

Thực hiện chạy ứng dụng trên thiết bị ảo giống trên thiết bị vật lý.

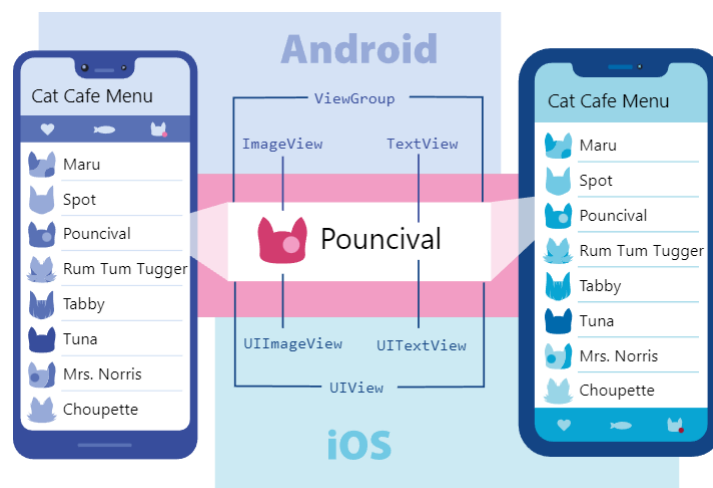


Hình 10: Kiểm tra kết nối thiết bị ảo trên Android Studio

3.3 Các khái niệm, thành phần

3.3.1 Components

Trong quá trình phát triển Android, người ta thường phân giao diện thành các khối hình chữ nhật để thực hiện các chức năng riêng như: hiển thị văn bản, hình ảnh, phản hồi của người dùng, ... Mỗi một khối như vậy được gọi là View.

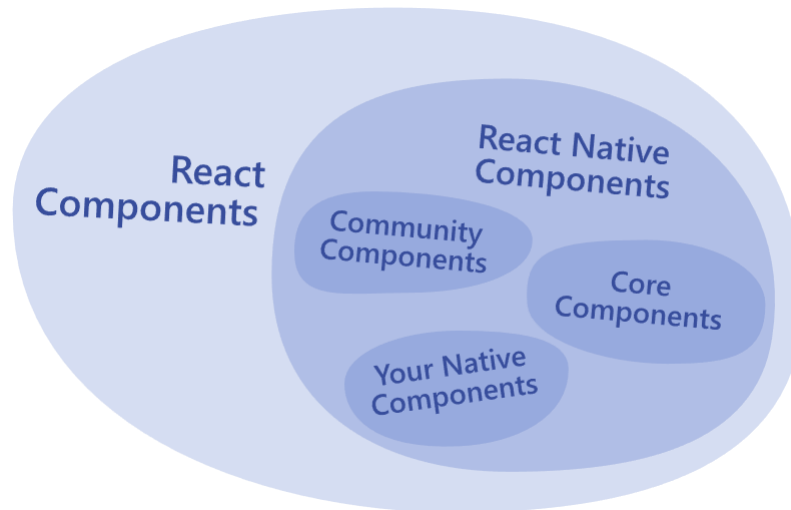


Hình 11: View trong giao diện Native

Đối với React Native, các View trên được gọi là *Component*. Một cách dễ hiểu, để

xây dựng ứng dụng React Native, ta cần tập chung mô tả giao diện mong muốn thành các component như trên.

Ta có thể phân loại các components như hình vẽ sau (chứa cả React Component vì React Native được xây dựng dựa trên ReactJS)



Hình 12: Phân loại các component

a. Core Components

Core Components là các component cốt lõi để xây dựng ra các component khác. Các component này điều khiển các hành động cơ bản về giao diện như: hiển thị văn bản, hình ảnh, cho phép người dùng nhập, ấn,...

Có thể phân loại Core Component như sau:

- *Basic Components*: bao gồm các component trong bảng 13.
- *User Interface*: Gồm các component điều khiển các hành động của người dùng:
 - + Button: Xử lý thao tác chạm của người dùng.
 - + Switch: Xử lý bật/tắt cho một chức năng.

REACTNATIVE COMPONENT	ANDROID VIEW	IOS VIEW	WEB VIEW TƯƠNG ỨNG	MÔ TẢ
<code><View></code>	<code><ViewGroup></code>	<code><UIView></code>	Không cuộn <code><div></code>	Vùng chứa hỗ trợ bố cục với flexbox, kiểu, một số xử lý cảm ứng và điều khiển trợ năng
<code><Text></code>	<code><TextView></code>	<code><UITextView></code>	<code><p></code>	Hiển thị, tạo kiểu và lồng các chuỗi văn bản và thậm chí xử lý các sự kiện chạm
<code><Image></code>	<code><ImageView></code>	<code><UIImageView></code>	<code></code>	Hiển thị các loại hình ảnh khác nhau
<code><ScrollView></code>	<code><ScrollView></code>	<code><UIScrollView></code>	<code><div></code>	Một vùng chứa cuộn chung có thể chứa nhiều thành phần và dạng xem
<code><TextInput></code>	<code><EditText></code>	<code><UITextField></code>	<code><input type="text"></code>	Cho phép người dùng nhập văn bản

Hình 13: Mô tả một số Core Components

- *List View*: Không giống như ScrolView, các component này hiển thị nội dung dưới dạng các phần tử. Điều này làm cho chúng trở thành lựa chọn hiệu quả để hiển thị danh sách dữ liệu dài. Bao gồm:
 - + FlatList: Hiển thị một danh sách có thể cuộn.
 - + SectionList: Giống FlatList nhưng đối với danh sách được phân đoạn.
- *Android-specific*: là các component cung cấp các dịch vụ đặc trưng của Android. Bao gồm:
 - + BackHandler: Phát hiện các nút phần cứng điều hướng quay lại. (Ví dụ: nút back ảo, vuốt cạnh bên,...)
 - + DrawerLayoutAndroid: Tạo một giao diện dạng ngăn kéo trên Android.
 - + PermissionsAndroid: Cung cấp quyền truy cập đối với Android
 - + ToastAndroid: Tạo một cảnh báo trên Android
- *iOS-specific*: là các component cung cấp các dịch vụ đặc trưng của UIKit. (Đồ án này chỉ tìm hiểu về react native và android nên không đi sâu vào

IOS, chỉ nêu ra ở đây)

- *Other*: các component thường dùng khác
 - + `ActivityIndicator`: Hiển thị giao diện loading xoay tròn
 - + `Alert`: Hiển thị hộp thoại thông báo với tiêu đề và nội dung
 - + `Animated`: Một thư viện để tạo các hoạt ảnh linh hoạt, mạnh mẽ, dễ xây dựng và bảo trì.
 - + `Dimensions`: Cung cấp giao diện để nhận kích thước thiết bị.
 - + `KeyboardAvoidingView`: Cung cấp giao diện điều khiển hiển thị bàn phím.
 - + `Linking`: Xử lý hành động tương tác với liên kết.
 - + `Modal`: Một giao diện nổi phía trên giao diện gốc.
 - + `PixelRatio`: Cung cấp quyền truy cập vào mật độ pixel của thiết bị.
 - + `RefreshControl`: Thành phần này được sử dụng bên trong `ScrollView` chức năng thêm kéo để làm mới.
 - + `StatusBar`: Thành phần để kiểm soát thanh trạng thái ứng dụng.

Các core component được cung cấp trong package "react-native", vì thế để sử dụng chúng, ta chỉ cần import từ package đó:

b. *Community Components*

Như đã trình bày, cộng đồng phát triển React Native rất lớn, vì thế để phục vụ cho phát triển ứng dụng nhanh chóng, một số nhóm, công ty xây dựng ra các gói, thư viện chứa các component xây dựng sẵn như: ant-design, axios, calendar, react-native-webview,... Cụ thể các thư viện này được trình bày tại mục "Một số thư viện phổ biến".

Để sử dụng được các component này, ta cần cài đặt các thư viện bằng dòng lệnh:


```
2  import { View, Text } from 'react-native';
3
4  const MyComponent = () => {
5    return (
6      <View>
7        <Text>my component</Text>
8      </View>
9    )
10 }
11
12 export default MyComponent;
```

Hình 14: Ví dụ sử dụng Core Component

```
PS C:\Món học\08 an 2\Ứng dụng\HelloWorld\firstApp> npm install @ant-design/react-native
npm WARN deprecated core-js@2.6.12: core-js@3.23.3 is no longer maintained and not recommended for usage due to the number of issues. Because of the V8 engine whines, feature detection in old core-js versions could cause a slowdown up to 100x even if nothing is polyfilled. Some versions have web compatibility issues. Please, upgrade your dependencies to the actual version of core-js.
added 39 packages, changed 50 packages, and audited 1003 packages in 20s
121 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
```

Hình 15: Cài đặt package - AntDesign

npm install <package-name> Sau khi cài đặt gói thành công, các component này sử dụng như core component, chỉ cần import và sử dụng.

c. Your Native Components

Đây là các component do chính người phát triển ứng dụng xây dựng.

Việc phân chia, xây dựng các component một cách chính xác giúp cho ứng dụng được tối ưu, tái sử dụng code một cách hiệu quả

3.3.2 JSX

React và React Native sử dụng JSX - một cú pháp cho phép bạn viết các phần tử bên trong JavaScript như sau: <Text>Hello HUST</Text>.

Vì JSX là JavaScript nên hoàn toàn có thể sử dụng biến hay gọi hàm bên trong nó

như hình sau:

```
import React from 'react';
import {Text} from 'react-native';

const MyComponent = () => {
  const university = 'HUST';
  const getMyName = () => "Nguyễn Bùi Nam Trường";
  return <Text>Hello, {university}, I am {getMyName()}</Text>;
};

export default MyComponent;
```

Hình 16: Ví dụ về JSX

Vì JSX được bao gồm trong thư viện React, nên để sử dụng JSX trong việc xây dựng các component thì ta cần phải khai báo sử dụng thư viện React trong file của mình (trong ảnh là dòng "import React from 'react';").

Sau khi có khái niệm cơ bản về Component và JSX, ta hoàn toàn có thể xây dựng một component từ các Core Component như ví dụ sau:

```
1  import React from 'react';
2  import {View, Text, TextInput} from 'react-native';
3
4  const CustomComponent = () => {
5    <View>
6      <Text>Hello, I am...</Text>
7      <TextInput/>
8    </View>
9  };
10
11  export default CustomComponent;
```

Hình 17: Ví dụ về JSX

Không chỉ đối với component, ta cũng có thể tái sử dụng *CustomComponent* vào

các component khác trong ứng dụng.

3.3.3 Props

Props - viết tắt của Properties, đây là một khái niệm cơ bản trong lập trình hướng đối tượng. Đây là những thuộc tính cho phép tùy chỉnh các component.

```
1  import React from 'react';
2  import {View, Text} from 'react-native';
3
4  const People = (props) => {
5    const { name } = props;
6    return <Text>{name}</Text>;
7  };
8
9  const CustomComponent = () => {
10    return (
11      <View>
12        <People name={"Nguyễn Bùi Nam Trường"} />
13        <People name={"HUST"} />
14      </View>
15    );
16  }
17  export default CustomComponent;
```

Hình 18: Ví dụ Props trong Component

Sau khi có khái niệm về Props, ta có thể thử với các Core Component để hiểu chi tiết về cách sử dụng của chúng.

Chú ý: Props là các properties nhưng không có setter, nó không thay đổi giá trị trong component mà chỉ nhận giá trị truyền vào khi gọi đến component. Cần ghi nhớ điều này vì đây là điều khác biệt lớn nhất khi so sánh Props với State.

3.3.4 State

Có thể coi props như là đối số để cấu hình cách hiển thị các thành phần, khi đó state giống như bộ lưu trữ dữ liệu của các thành phần. Khi State thay đổi cũng làm thay đổi tùy chỉnh của component (thay đổi này có thể theo thời gian hoặc dựa trên tương tác của người dùng). Như đã trình bày khi nói về Props, State là các Properties có đầy đủ cả setter và getter.

Trong ví dụ sau, component Counter có sử dụng state Count (setter: setCount, getter: count) bằng cách sử dụng React Hook là useState (cụ thể trình bày trong vòng đời component và các thư viện). State Count theo dõi mỗi khi người dùng nhấn vào button, count tăng lên 1 đơn vị, khi đó giá trị count hiển thị thay đổi trên giao diện người dùng.

```
1  import React, { useState } from 'react';
2  import {View, Text} from 'react-native';
3
4  const Counter = () => {
5    const [count, setCount] = useState(0);
6
7    const handleClick = (e) => {
8      setCount(count + 1);
9    }
10
11    return (
12      <View>
13        <Text>Click times: {count} </Text>
14        <Button onPress={handleClick}>Click me!</Button>
15      </View>
16    );
17  }
18  export default Counter;
```

Hình 19: Ví dụ State trong Component

3.3.5 Style

Đối với mỗi phần tử hiển thị trên giao diện người dùng, ta cần điều chỉnh hình thức hiển thị như: có cần đường viền; hiển thị theo chiều dọc hay ngang; chiều dài, chiều rộng bao nhiêu;... Để thực hiện nhu cầu này, ta có thể thêm Props là Style cho các component của mình. Style cũng là prop của các Core Component.

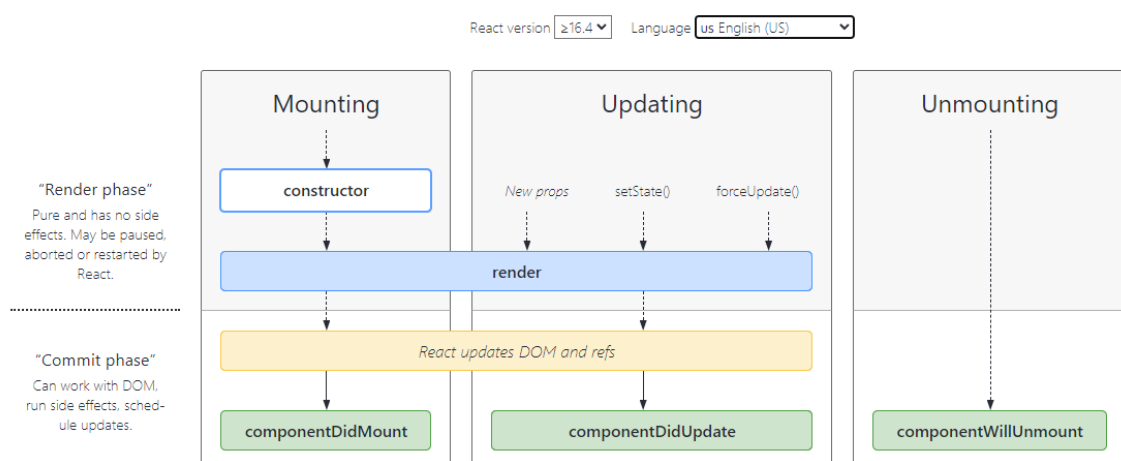
```
1 import React from 'react';
2 import {TextInput} from 'react-native';
3
4 const CustomInput = () => {
5   return <TextInput style={{ width: 50, height: 200, borderRadius: 4, borderWidth: 1, borderColor: '#f00' }}/>;
6 }
7 export default CustomInput;
```

Hình 20: Ví dụ Style trong Component

Chú ý: Các thuộc tính của style trong ReactNative thông thường được viết giống các thuộc tính CSS nhưng dưới dạng camelcase.

3.3.6 Vòng đời của Component

Ta có thể tóm tắt vòng qua sơ đồ dưới đây:



Hình 21: Sơ đồ vòng đời trong component

Cụ thể, vòng đời của một component gồm 3 giai đoạn:

- a. *Mounting*: Là quá trình bắt đầu từ khi gọi đến component, nhận giá trị JSX trả về với props và states khởi tạo đến khi cập nhật giá trị đó lên DOM và thực thi một số yêu cầu được quy định trước.
- b. *Updating*: Sau khi thực thi một số yêu cầu hoặc xử lý tương tác của người dùng khiến giá trị props hoặc state thay đổi, quá trình Updating bắt đầu. Từ các states, props mới, component trả về giá trị JSX mới theo chúng. Sau đó, JSX mới lại tiếp tục được cập nhật vào DOM và đợi cập nhật tiếp theo.
- c. *Unmounting*: Các quá trình cập nhật sẽ tiếp tục diễn ra cho đến khi có cập nhật yêu cầu xóa component này trong DOM, khi đó quá trình Unmounting bắt đầu. Quá trình này thực thi một số yêu cầu của người lập trình đã quy định trước.

Để quy định các yêu cầu thực thi trong các quá trình trên, ta cần khai báo hàm với đúng tên gọi như trong sơ đồ:

- *Constructor*: Khai báo Props, khởi tạo các giá trị ban đầu của State
- *render*: trả về giá trị JSX phụ thuộc vào state, props.
- *componentDidMount*: các yêu cầu thực thi sau khi gán giá trị đầu tiên của component vào DOM.
- *componentDidUpdate*: các yêu cầu thực thi sau khi có giá trị mới được cập nhật vào DOM.
- *componentWillUnmount*: các yêu cầu thực thi sau khi component được xóa khỏi DOM.

Các hàm trên được gọi trong Class Component - đây là cách định nghĩa component duy nhất từ ES5 trở về trước (ES - viết tắt của ECMAScript là tiêu chuẩn cho ngôn ngữ lập trình JavaScript). Từ ES6 (ở thời điểm tôi viết đề án này là phiên bản mới nhất của ECMAScript), ta có thể định nghĩa component bằng Function Component. Khi đó, thay vì sử dụng các hàm vòng đời trên, ta có thể sử dụng các hàm React Hook được cung cấp bởi react (chi tiết được trình bày trong mục các thư viện phổ biến).

3.4 Một số thư viện phổ biến

Ở mục trước, tôi đã nhắc đến Core Component và đưa ra thông tin về các Core Component. Ở mục này, tôi sẽ giới thiệu về Community Components. Như đã nói, các Community Components là các component được cung cấp bởi các gói, các thư viện. Có rất nhiều thư viện như vậy, một số thư viện sau được sử dụng phổ biến trong các phần mềm hiện nay, điều này dựa trên quan sát và kinh nghiệm thực tế của tôi.

3.4.1 React và ReactNative

Chắc chắn rồi, hai thư viện này tồn tại ngay khi khởi tạo một ứng dụng React Native qua dòng lệnh: `npx react-native@latest init <project-name>`.

Ngoài các Core Component mà thư viện này cung cấp, ta còn cần sử dụng các hàm React Hook. Khi ES6 ra đời cùng với khái niệm về Function Component, việc phải sử dụng chúng trong các ứng dụng sử dụng ngôn ngữ JavaScript là rất cần thiết bởi đây sẽ là tiêu chuẩn của các ứng dụng sau này.

Trước hết, về định nghĩa, Hook là một thành phần trong component không sử dụng để hiển thị (khi thay đổi giá trị không gây ra quá trình updating). Đối với các Hook, cần có yêu cầu đều phải bắt đầu bằng "use" và được viết ở đầu trong component. Hook được phân loại bao gồm:

- *State Hook*: để đáp ứng mục đích định nghĩa các state trong component. Bao gồm:
 - + *useState*: Khai báo một state với getter, setter từ đó có thể trực tiếp cập nhật giá trị.
 - + *useReducer*: Khai báo một state với logic cập nhật được định nghĩa trước.
- *Context Hook*: để đáp ứng nhu cầu cập nhật dựa trên tương tác từ component khác. (Ví dụ khi người dùng thực hiện tương tác với 1 component trên giao diện với nhu cầu xử lý thay đổi một component khác). React hiện tại chỉ cung cấp 1 loại Context Hook là *useContext*.
- *Ref Hook*: Bên cạnh State, Props, trong quá trình xây dựng component cần đến các giá trị lưu trữ nhưng khi thay đổi không gây ra quá trình updating. Các Hook loại này đáp ứng nhu cầu đó, bao gồm:
 - + *useRef*: khai báo một giá trị, tuy nhiên nó thường để lưu trữ giá trị của một phần tử DOM.
 - + *useImperativeHandle*: giống *useRef* nhưng có thể tùy chỉnh xử lý khi thay đổi giá trị
- *Effect Hook*: chứa các yêu cầu thực thi trong vòng đời của component (tương đương với *componentDidMount* + *componentDidUpdate* + *componentWillUnmount* trong Class Component). Bao gồm:
 - + *useEffect*: thường sử dụng để kết nối component với sự kiện bên ngoài.
 - + *useLayoutEffect*: được gọi đến trước khi cập nhật lại DOM, thường sử dụng để tính toán và xử lý bố cục.

- + `useInsertionEffect`: được gọi đến trước khi React tính giá trị JSX mới cho component, thường sử dụng để chèn CSS động tại đây.
- *Performance Hook*: để đáp ứng nhu cầu tối ưu hóa hiệu suất trong việc định nghĩa lại các hàm, các giá trị sau mỗi lần rendering. Cụ thể lưu trữ lại các giá trị ở bộ nhớ cache và bỏ đi quá trình kết xuất dữ liệu nếu dữ liệu không thay đổi so với lần kết xuất trước. Bao gồm:
 - + `useMemo`: cho phép lưu trữ kết quả của một yêu cầu thực thi gây tốn hiệu năng.
 - + `useCallback`: cho phép lưu trữ định nghĩa các hàm.
 - + `useTransition`: cho phép đánh dấu một quá trình chuyển đổi trạng thái là không bị chặn và cho phép các bản cập nhật khác làm gián đoạn quá trình đó.
 - + `useDeferredValue`: cho phép hoãn cập nhật một phần không quan trọng của giao diện người dùng và để các phần khác cập nhật trước.
- *Other Hook*: Ngoài các Hook trên, React còn cung cấp một số Hook phục vụ quá trình xây dựng ứng dụng, bao gồm
 - + `useDebugValue`: cho phép tùy chỉnh hiển thị trên React DevTools.
 - + `useId`: cho phép một thành phần liên kết một ID duy nhất với chính nó. Thường được sử dụng với các API trợ năng.
 - + `useSyncExternalStore`: cho phép một thành phần đăng ký vào một cửa hàng bên ngoài.
- *Your Hook*: Ngoài các Hook mà React cung cấp, ta hoàn toàn có thể định nghĩa các Hook cá nhân dưới dạng các hàm JavaScript. Lưu ý: vẫn phải đáp ứng yêu cầu của Hook: bắt đầu bằng "use" và được viết ở đầu trong component.

3.4.2 PropTypes

Trong quá trình phát triển ứng dụng, ta có thể xảy ra một số lỗi về lỗi đánh máy, sai kiểu dữ liệu,... Điều này thường xảy ra do JavaScript là ngôn ngữ không có yêu cầu cao về tính hướng đối tượng.

Để khắc phục tình trạng này, tôi khuyên dùng sử dụng tiện ích PropTypes. Tiện ích này giúp định nghĩa đúng kiểu dữ liệu cho các props của component, khi sử dụng truyền sai kiểu dữ liệu, nhà phát triển sẽ nhận biết thông qua thông báo trên trình soạn thảo.

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import {Text} from 'react-native';
4
5  const Hello = (props) => {
6    const { name } = props;
7    return <Text>Hello {name}!</Text>;
8  }
9
10 Hello.propTypes = {
11   name: PropTypes.string;
12 }
13
14 export default CustomInput;
```

Hình 22: Ví dụ sử dụng PropTypes

PropTypes ngừng cung cấp bởi React từ v15.5, vì vậy để sử dụng cần tải thư viện "prop-types" thay thế. Để cài đặt thư viện này, ta cần chạy dòng lệnh: "npm i prop-types".

3.4.3 Thư viện chứa Basic Component theo chuẩn thiết kế

Để xây dựng ứng dụng có thiết kế mang tính tổng quát, đem lại trải nghiệm tốt cho người dùng, ta cần đến các component mẫu chuẩn như Layout, Button, Icon,

DatePicker, ... và sử dụng chúng thống nhất trong cả ứng dụng.

Đối với các ứng dụng có quy mô lớn, ta hoàn toàn có thể tự xây dựng các component mẫu trên. Tuy nhiên đối với các ứng dụng nhỏ, quá trình trên gây tốn rất nhiều thời gian, vì thế việc sử dụng các thư viện hỗ trợ là thiết thực nhất. Hiện nay, có một số thư viện nổi tiếng đáp ứng nhu cầu trên đối với React Native: Ant Design, Material Design.

3.4.4 Axios

Đối với các ứng dụng native, ngoài việc xử lý tương tác người dùng, đôi khi cần xử lý dữ liệu từ server. Axios giúp giải quyết nhu cầu này. Cụ thể, Axios là một thư viện HTTP Client dựa trên Promise dành cho node.js và trình duyệt. Nó có tính đẳng hình (tức là cùng codebase có thể chạy trong cả trình duyệt và node.js). Đối với native, Axios sử dụng module http trong node.js

3.5 Phát hành ứng dụng

Sau khi xây dựng xong ứng dụng của mình, các nhà phát triển đưa sản phẩm của mình tới khách hàng. Đối với Android, nhà phát triển cần tạo file .apk (Android Package Kit).

Trong quá trình cài đặt Android Studio, ta đã cài đầy đủ môi trường. Vì vậy, ta chỉ cần thực hiện các bước sau:

Bước 1 : Tạo một keyStore

Để tạo tệp nhị phân thực thi React Native cho Android, ta cần khóa do Java tạo. Để tạo khóa này, ta cần chạy dòng lệnh trong terminal: `keytool -genkey -v -keystore <your_key_name>.keystore -alias <your_key_alias> -keyalg RSA -keysize 2048 -validity 10000`.

Sau khi chạy dòng lệnh, kết quả yêu cầu nhập các thông tin về dự án, đây là những thông tin yêu cầu để Java tạo key.

Sau khi điền đầy đủ thông tin này, trên thư mục của dự án, ta thấy file <your_key_name>.key

```
PS D:\Môn học\Đồ án 2\Ứng dụng\HelloWorld> keytool -genkey -ke my_keystore -alias your_key_alias -keyalg RSA -keysize 2048 -validity 10000
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Nguyen Bui Nam Truong
What is the name of your organizational unit?
  [Unknown]:  HUST
What is the name of your organization?
  [Unknown]:  MI1-01-K63
What is the name of your City or Locality?
  [Unknown]:  HaNoi
What is the name of your State or Province?
  [Unknown]:  HaNoi
What is the two-letter country code for this unit?
  [Unknown]:  VI
Is CN=Nguyen Bui Nam Truong, OU=HUST, O=MI1-01-K63, L=HaNoi, ST=HaNoi, C=VI correct?
[no]:  yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10,000 days
for: CN=Nguyen Bui Nam Truong, OU=HUST, O=MI1-01-K63, L=HaNoi, ST=HaNoi, C=VI
[Storing my_key_name.keystore]
PS D:\Môn học\Đồ án 2\Ứng dụng\HelloWorld>
```

Hình 23: Tạo KeyStore

Bước 2 : Thêm keyStore vào dự án

Copy file key có được từ bước 1, dán vào thư mục android/app, hoặc chạy dòng lệnh: `mv my-release-key.keystore /android/app`.

Sau đó cấu hình file build.gradle như sau:

Bước 3 : Tạo file APK

Từ thư mục ban đầu của dự án, chạy các dòng lệnh sau:

- `cd android`
- `gradlew assembleRelease`
- `./gradlew assembleRelease`

Sau khi hoàn thành, nếu thành công, ta thấy file apk tại đường dẫn android/app/build/outputs/release.apk. Đây là file giúp cài đặt ứng dụng trên thiết bị android.

```
1.  android {  
2.      ....  
3.      signingConfigs {  
4.          release {  
5.              storeFile file('your_key_name.keystore')  
6.              storePassword 'your_key_store_password'  
7.              keyAlias 'your_key_alias'  
8.              keyPassword 'your_key_file_alias_password'  
9.          }  
10.     }  
11.  
12.     buildTypes {  
13.         release {  
14.             ....  
15.             signingConfig signingConfigs.release  
16.         }  
17.     }  
18. }  
19. }
```

Hình 24: Tạo KeyStore

Kết luận

Sau quá trình nghiên cứu và tìm hiểu về chủ đề Lập trình Android với React Native, em đã có thêm kiến thức nhất định về lập trình Android nói riêng và ứng dụng Native nói chung, cùng với đó là những kỹ năng, kinh nghiệm trong việc xây dựng và phát triển phần mềm.

Mặc dù đã rất cố gắng trong việc nghiên cứu và thực hiện đồ án, nhưng do thời gian hạn cũng như sự hiểu biết của bản thân em còn hạn chế nên đồ án chỉ dừng lại ở việc tìm hiểu về Android, React Native mà chưa đi sâu vào phát triển ứng dụng hoàn chỉnh, giải quyết bài toán thực trong cuộc sống. Bài đồ án này cũng giúp em mở ra

định hướng nghiên cứu thêm cách phát triển ứng dụng React Native lớn với quy mô nhóm: code convention, design partent đối với React Native,... Nếu có cơ hội, em sẽ tìm hiểu và trình bày ở đề án sau.

Em xin chân thành cảm ơn!

Tài liệu tham khảo

Tài liệu do nhà phát hành cung cấp

1. <https://reactnative.dev/docs>.
2. <https://reactjs.org/docs>.

Tài liệu từ các thông tin tổng hợp trên trình duyệt

1. <https://teamvietdev.com/thanh-phan-kien-truc-android/>.
2. https://wikipedia.org/wiki/React_Native.
3. <https://vietnix.vn/nodejs-la-gi/>.
4. <https://vietnix.vn/nodejs-la-gi/>.