TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC

KHOA CÔNG NGHỆ THÔNG TIN

--------------------------------

BÁO CÁO KẾT THÚC MÔN HỌC

Lập trình di động 3

**Note Project**

Giảng viên hướng dẫn: **Phan Bá Thái**

Sinh viên thực hiện:

1. **Lê Đình Cường**
2. **Trần Đức Phương**

Ngành: Công nghệ thông tin Lớp: Di Động 3 Khoá: 12

*Tp. Hồ Chí Minh*, ngày 4 tháng 12 năm 2018

**NHẬT KÝ HOẠT ĐỘNG NHÓM**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Stt** | **Họ và tên** | **Công việc đã thực hiện** | **Tự đánh giá** | **Nhóm đánh giá** | **Chữ ký** |
| 1 | Lê Đình Cường | 1. Tìm hiểu về react native. 2. Tìm hiểu về cách xây dựng project react native. 3. Tìm hiểu về class, arrow function, thiết kế minh họa đồ án. 4. Tìm hiểu về thuộc tính width, flex, height và các loại flex box như: flex direction, justify content, align items. 5. Tìm hiểu về button, Text, Text Input, ListView, Navigator. 6. Xây dựng giao diện màn hình home, notescreen, addnote, editnote, detailnote. 7. Thực hiện hiển thị, thêm, xóa, sửa ghi chú, xây dựng webservice. |  |  |  |
| 2 | Trần Đức Phương | 1. Tạo github, tìm hiểu về Object, state. 2. Tìm hiểu về checkbok và radiobutton. 3. Xây dựng giao diện hiển thị các task, thêm task, sửa task và hiển thị calendar. 4. Thực hiện hiển thị, thêm, xóa, sửa các task và xây dựng webservice. |  |  |  |
| 3 | Bùi Đưc Cơ | 1. Tìm hiểu về biến var, let. 2. Tìm hiểu về collection Map, và các thuộc tính của style. 3. Tìm hiểu về webservice. |  |  |  |
| 4 | Đoàn Thị Yến Nhi | 1. Tìm hiểu về biến var và biến let 2. Tìm hiểu về collection Set. 3. Tìm hiểu về props. 4. Xây dựng giao diện phần thùng rác trong project. 5. Thực hiện chức năng hiển thị các ghi chú đã xóa, và phục hồi ghi chú đã xóa. |  |  |  |

**Muc Lục**

[I. TỔNG QUAN VỀ REACT NATIVE 1](#_Toc532038853)

[1. Giới thiệu tổng quan về React native 1](#_Toc532038854)

[1.1 Giới thiệu tổng quan về react native: 1](#_Toc532038855)

[1.2 Cách cài đặt react native (Đình Cường) 1](#_Toc532038856)

[1.3 Cách tạo một project react native (Đình Cường) 5](#_Toc532038857)

[1.4 Cách chạy ứng dụng react native (Đình Cường) 6](#_Toc532038858)

[2. Kiến thức cơ bản về ES6 trong React native 8](#_Toc532038859)

[2.1 Giới thiệu 8](#_Toc532038860)

[2.2 Block-Scoped 9](#_Toc532038861)

[2.3 Function (Đình Cường) 9](#_Toc532038862)

[2.4 Tìm hiểu về biến var, let, const 10](#_Toc532038863)

[2.5 Mảng trong Java Script (Đình Cường) 11](#_Toc532038864)

[2.6 Collection Set 12](#_Toc532038865)

[2.7 Collection Map 14](#_Toc532038866)

[2.8 Class (Đình Cường) 17](#_Toc532038867)

[2.9 Object (Đức Phương) 20](#_Toc532038868)

[3. Component trong React Native 21](#_Toc532038869)

[3.1 State (Đức Phương) 21](#_Toc532038870)

[3.2 Props 23](#_Toc532038871)

[3.3 Style 24](#_Toc532038872)

[3.4 Thuộc tính Flex Dimensions 25](#_Toc532038873)

[3.5 Tìm hiểu Flexbox (Đình Cường) 26](#_Toc532038874)

[3.5.1 Flex Direction 27](#_Toc532038875)

[3.5.2 Justify Content 27](#_Toc532038876)

[3.5.3 Align Items 28](#_Toc532038877)

[3.6 TextInput (Đình Cường) 29](#_Toc532038878)

[3.7 Button 32](#_Toc532038879)

[3.8 Text (Đình Cường) 34](#_Toc532038880)

[3.9 Image 37](#_Toc532038881)

[3.10 TouchableOpacity (Đình Cường) 38](#_Toc532038882)

[3.11 CheckBox (Đưc Phương) 38](#_Toc532038883)

[3.12 Radio Button (Đức Phương) 38](#_Toc532038884)

[3.13 List View (Đình Cường) 39](#_Toc532038885)

[3.14 Navigation trong React 41](#_Toc532038886)

[II. Phân Tích Thiết Kế Hệ Thống 42](#_Toc532038887)

[1 Phân tích hệ thống 43](#_Toc532038888)

[1.1 Trang chủ (Đình Cường) 43](#_Toc532038889)

[1.1.1 Giao diện người dùng 44](#_Toc532038890)

[1.1.2 Mô tả 44](#_Toc532038891)

[1.1.3 Phân tích các thành phần 45](#_Toc532038892)

[1.1.4 Các xử lý 48](#_Toc532038893)

[1.2 Hiển thị danh sách ghi chú NoteScreen (Đình Cường) 52](#_Toc532038894)

[1.2.1 Giao diện người dùng 52](#_Toc532038895)

[1.2.2 Mô tả 52](#_Toc532038896)

[1.2.3 Phân tích các thành phần 52](#_Toc532038897)

[1.2.4 Các xử lý 55](#_Toc532038898)

[1.3 Thêm ghi chú (Đình Cường) 58](#_Toc532038899)

[1.3.1 Giao diện người dùng 58](#_Toc532038900)

[1.3.2 Mô tả 58](#_Toc532038901)

[1.3.3 Phân tích các thành phần 58](#_Toc532038902)

[1.3.4 Các xử lý trong màn hình 61](#_Toc532038903)

[1.4 Sửa ghi chú 63](#_Toc532038904)

[1.4.1 Giao diện người dùng 63](#_Toc532038905)

[1.4.2 Mô tả 63](#_Toc532038906)

[1.4.3 Phân tích các thành phần 63](#_Toc532038907)

[1.4.4 Các xử lý tại màn hình 65](#_Toc532038908)

[1.5 Chi tiết ghi chú 67](#_Toc532038909)

[1.5.1 Giao diện người dùng 67](#_Toc532038910)

[1.5.2 Mô tả 67](#_Toc532038911)

[1.5.3 Phân tích các thành phần 67](#_Toc532038912)

[1.5.4 Các xử lý trên màn hình 69](#_Toc532038913)

[1.6 Hiển thị các nhiệm vụ (Đức Phương) 70](#_Toc532038914)

[1.6.1 Giao diện người dùng 70](#_Toc532038915)

[1.6.2 Mô tả 70](#_Toc532038916)

[1.6.3 Phân tích các thành phần 70](#_Toc532038917)

[1.6.4 Các xử lý trên màn hình 71](#_Toc532038918)

[1.7 Thêm nhiệm vụ (Đức Phương) 71](#_Toc532038919)

[1.7.1 Giao diện người dùng 71](#_Toc532038920)

[1.7.2 Mô tả 71](#_Toc532038921)

[1.7.3 Phân tích các thành phần 71](#_Toc532038922)

[1.7.4 Các xử lý trên màn hình 71](#_Toc532038923)

[1.8 Sửa nhiệm vụ (Đức Phương) 71](#_Toc532038924)

[1.8.1 Giao diện người dùng 71](#_Toc532038925)

[1.8.2 Mô tả 71](#_Toc532038926)

[1.8.3 Phân tích các thành phần 71](#_Toc532038927)

[1.8.4 Các xử lý trên màn hình 71](#_Toc532038928)

[1.9 Chi tiết nhiệm vụ (Đức Phương) 72](#_Toc532038929)

[1.9.1 Giao diện người dùng 72](#_Toc532038930)

[1.9.2 Mô tả 72](#_Toc532038931)

[1.9.3 Phân tích các thành phần 72](#_Toc532038932)

[1.9.4 Các xử lý trên màn hình 72](#_Toc532038933)

[III. Kết quả đạt được 72](#_Toc532038934)

[1 Kết quả đạt được 72](#_Toc532038935)

[1.1 Kết quả chạy của tính chức năng ghi chú 72](#_Toc532038936)

[1.2 Chức năng thêm nhiệm vụ 78](#_Toc532038937)

[2 Các kết luận và kiến nghị 78](#_Toc532038938)

[2.1 Những điểm đã làm được 78](#_Toc532038939)

[2.2 Những điểm chưa làm được 79](#_Toc532038940)

[2.3 Các chức năng trong tương lai 79](#_Toc532038941)

[VI. Tài liệu tham khảo 79](#_Toc532038942)

# I. TỔNG QUAN VỀ REACT NATIVE

## 1. Giới thiệu tổng quan về React native

### 1.1 Giới thiệu tổng quan về react native:

- Đây là công nghệ được tạo ra bởi Facebook và có sự hỗ trợ cùng phát triển nền tảng bởi cộng đồng lập trình viên. Sử dụng chính sdk của android và ios để phát triển ứng dụng cho hai nền tảng này.

- Sử dụng Javascript theo tiêu chuẩn es6 để viết và cho phép thiết kế và tái sử dụng giao diện từ các component và các thành phần đóng góp từ người dùng.

- Có thể sủ dụng react native để phát triển ứng dụng android và ios.

- Nếu muốn phát triển mà không yêu cầu khắt khe về mặt hiệu năng thì có thể sử dụng react native.

**Cách thức hoạt động của react native:**

Ứng dụng react native được chia thành hai phần: Phần view để hiện thị và phần xử lý. Phần hiển thị được biện dịch từ javascript sẽ map với những component của hệ thống. Phần xử lý vẫn được thực hiện trực tiếp bằng ngôn ngữ javascript.

**Ưu điểm của react native:**

- Có thể viết ứng dụng cho nền web, nền tảng ios và android mà không mất nhiều thời gian.

- Xây dựng ứng dụng nhanh chóng.

**Nhược điểm:**

- Còn thiếu nhiều thành phần đặc trưng của nền tảng android và ios.

- Cấu trúc thay đổi liên tục gây khó nắm bắt.

- Không xây dựng được ứng dụng quá phức tạp và mang quá nhiều đặc trưng của nền tảng android hoặc ios.

- Không có IDE đặc trưng để phát triển react native.

- Không thể phát triển ứng dụng ứng dụng ios trên nền tảng windows và linux.

- Sử dụng cấu trúc es6 khá phức tạp.

### 1.2 Cách cài đặt react native (Đình Cường)

Để cài đặt react native cần có Node, Python, JDK, Android studio.

**Bước 1: Cài đặt các thành phần**

- Cài đặt Node, Python và JDK ở đây chúng ta sẽ cài chúng thông qua Chocolatey (Đây là một gói quản lý phổ biến trên windows).

+ Cài đặt Chocolatey bằng cmd hoặc powershell với quyền quản trị administrator

CMD

@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

PowerShell

Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

- Sau khi cài đặt xong chocolatey ta đóng cửa sổ vừa chạy lệnh lại, sau đó mở CMD và chạy với quyền quản trị để tiến hành cài tải và cài đặt node, pyhon và jdk thông qua choco với câu lệnh

choco install -y nodejs.install python2 jdk8

**Bước 2: Sau khi cài đặt đủ các thành phần thì chúng ta bắt đầu cài React Native thông qua CMD hoặc Shell với câu lệnh:**

npm install -g react-native-cli

**Bước 3: Thiết lập môi trường phát triển android**

- Tải và cài đặt android studio và phải chọn các thành phần sau

+ Android sdk

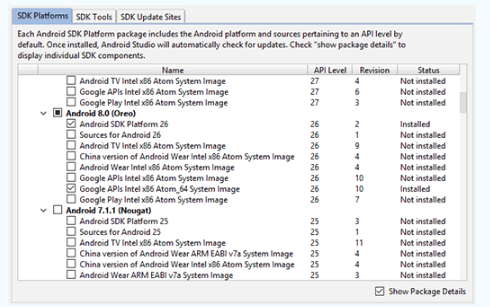
+ Android SDK Platform

+ Performance

+ Android Virtual Device

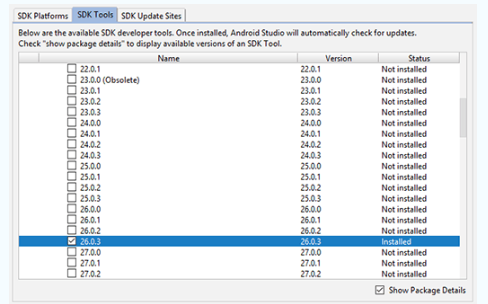
- Sau khi cài xong thì ta tiến hành cài android sdk theo mặc định android studio sẽ cài android sdk mới nhất. Nhưng để xây dựng ứng dụng với native code với mã nguồn gốc yêu cầu sdk android 8.0.

+ Appearance & Behavior 🡪 System Settings -> Android SDK và chọn sdk cần cài



+ Ta cũng phải cài luôn Google Apis Intel x86 Atom\_64 System Image

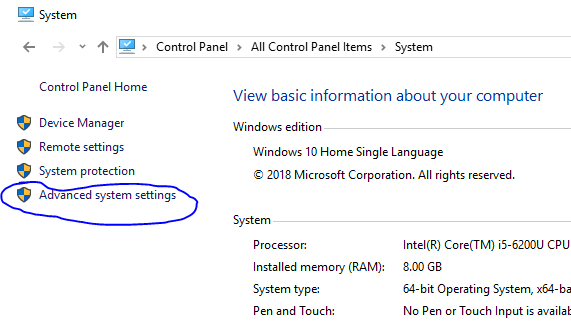
+ Tại thẻ SDK Tools ta nhấn còa check box “Show package Details” để hiện lên thêm thông tin và chọn Android SDK Build-Tools 26.0.3 được chọn



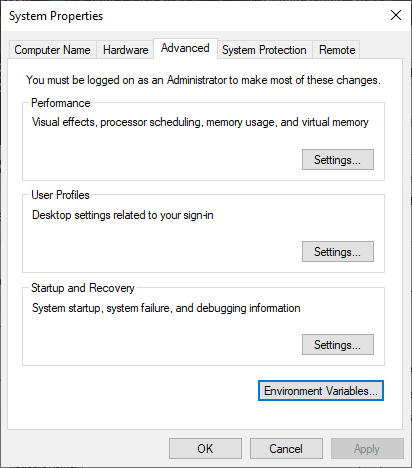
Sau khi chọn xong ta nhấn áp dụng để android studio bắt đầu download về

**Bước 5: Thiết lập biến môi trường**

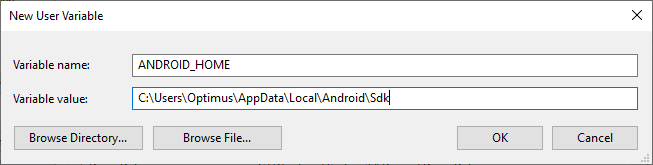
Tại bước này ta sẽ thêm biến môi trường là ANDROID\_HOME với giá trị là đường dẫn đến thư mục chứa Android SDK



Sau đó



Sau đó ta thêm mới biến môi trường cho cả hệ thông luôn hoặc của user cũng được



// Đường dẫn SDK thường

c:\Users\YOUR\_USERNAME\AppData\Local\Android\Sdk

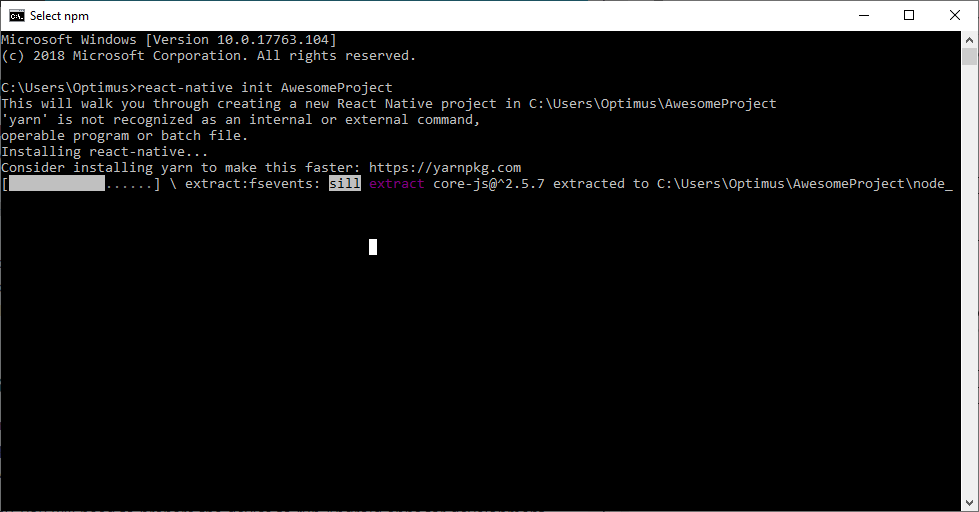
### 1.3 Cách tạo một project react native (Đình Cường)

Chạy câu lệnh bên dưới trong cmd

react-native init AwesomeProject

// AwesomeProject là tên của đồ án đặt tên gì cũng được

// CMD đang trỏ đến dường dẫn nào thì project native sẽ được tạo tại đường dẫn đó



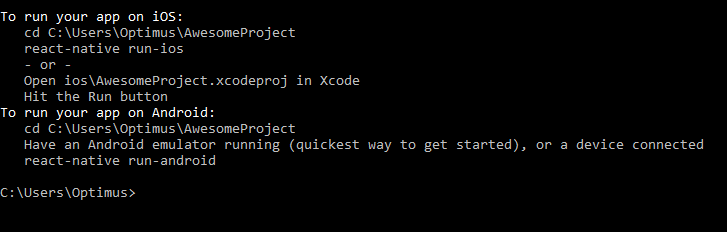
// Phải mất một lúc thì hệ thống mới tạo xong một project native mới

// Ta nên thay đổi đường dẫn mà cmd đang trỏ để dễ dàng hơn truy cập hơn

### 1.4 Cách chạy ứng dụng react native (Đình Cường)

Ta phải chuẩn bị một máy thật hoặc một máy ảo có thể khởi đổng máy ảo trong android studio hoặc là máy ảo genyration lên trước để ta thấy được kết quả chạy ứng dung.

Sau khi khởi động máy ao xong ta tiến hành chạy ứng dụng theo hướng dẫn sau khi ta chạy xong project



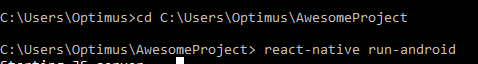
Tổng thể như sau chạy lệnh

cd AwesomeProject

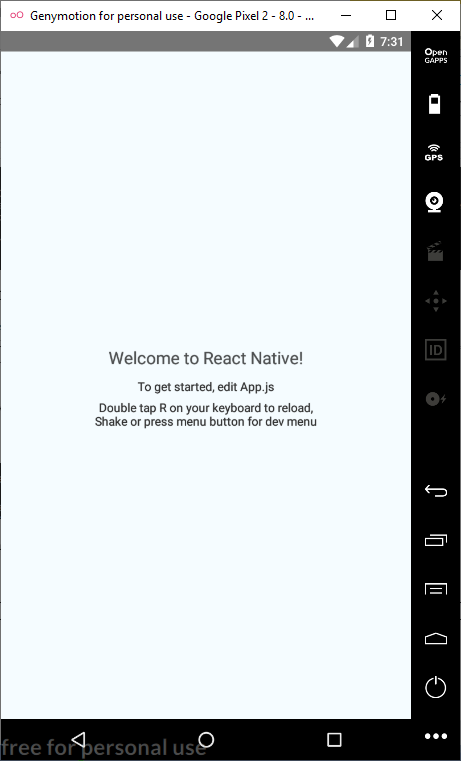
// Để xác định project cần chạy hay nói là trỏ đường dẫn cmd đến thư mục project giống như laravel đấy (Lệnh cd + với đường dẫn sẽ trỏ cmd đến đó)

react-native run-android

// Chạy ứng dụng

VD  


Sau khi chạy thành công trên máy ảo sẽ xuất hiện như sau



## 2. Kiến thức cơ bản về ES6 trong React native

### 2.1 Giới thiệu

ES6 là chữ viết tắt của ECMAScript 6 đây được gọi là một tập hợp các kỹ thuật nâng cao của javascipt và là phiên bản mới nhất của chẩn ECMAScript.

ECMAScript 6 (Còn được gọi là ES6, ES2015) là phiên bản mới nhất của chuẩn ECMAScript. được phê duyệt vào thang 6/2015. Nó là một bản nâng cấp quan trọng cho ES kể từ ES5 được chuẩn hóa vào năm 2009.

ES6 là một tập hợp kỹ thuật nâng cao của javascript và nó là một chuẩn mực để làm theo.

Những cái không nhắc tới trong es6 thì nó cũng tương tự java scrip thôi và bổ sung thêm cái chức năng mới thôi chúng ta cũng có thể sử dụng một số cú pháp cũ nếu react native cho phép.

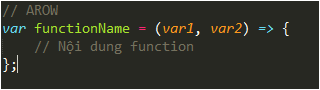
### 2.2 Block-Scoped

Đây là phạm vi của một khối nghĩa là chỉ hoạt động trong phạm vi được khai báo bởi căp dấu {}

### 2.3 Function (Đình Cường)

Ở trong javascript chúng ta có rất nhiều cách khai báo hàm đặc biệt chúng ta có thể gán 1 biến bằng một hàm ẩn danh giống ví dụ ở phía dưới và hàm trong java script có thể không nằm trong class cũng được.

**Cách tạo một arrow function**



**Gọi một function**

functionName();

**Cách khai báo một funtion vô danh**

(param1, param2,...,paramN) => {

*// Nội dung của hàm*

}

**Khai báo một arrow funtion không có tham số truyền vào**

var **loiChao** = () => {

return 'Hello';

}

**console**.**log**(**loiChao**()); *// Hello*

**Khai báo một arrow function có tham số truyền vào**

var **loiChao** = (name) => {

return name;

}

**console**.**log**(**loiChao**('Xin chao')); *// Xin chao*

**Xây dựng function trong Class**

class NameClass {

functionName() {

return “Hello”;

}

}

// Khởi tạo class và gọi function

var project = new NameClass();

project.start(); // "Hello"

### 2.4 Tìm hiểu về biến var, let, const

Biến var có thể hoạt động như một biến toàn cục và đây là cấu trúc khai báo biến củ ở trong java script.

Biến với từ khóa var khác với biến let ở chổ nếu khai báo biến trong hàm thì nó là biến cục bộ còn bên ngoài hàm thì là biến toàn cục. Còn biến với từ khóa let thì phạm vi hoạt động chỉ tồn tại trong khối đang khai báo và ta gọi đây là phạm vi block scoped (Trong phạm vi { }) .

**Cú pháp**

let tenBien = Giá trị;

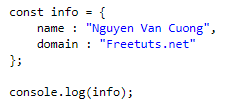
var tenBien = Giá trị;

// Kiểu của biến phụ thuộc vào giá trị mà nó chứa

**Hằng số trong ES6**

Sử dụng biến hằng đối với các đối tượng không thay đổi giá trị. Ở trong java scipt thì biến const có thể chứa bất kỳ giá trị gì và chỉ sử dụng trong phạm vi {} chứa nó mà thôi.

Khai báo: const tenBien = Giá Trị;



### 2.5 Mảng trong Java Script (Đình Cường)

**Khai báo:**

var tenBien = new Array();

// Hoặc khởi tạo có truyền giá trị

var tenBien = new Array(1, 2, 3);

// Cách khởi tạo khác

var name\_array = [];

var name\_array = [1,2,3];

**Truy xuất giá trị**

var t = new **Array**(1,2,3);

**alert**(t[0]); *// kết quả là 1*

**alert**(t[1]); *// kết quả là 2*

**alert**(t[2]); *// kết quả là 3*

**Sử dụng vòng lặp để duyệt mảng**

var name\_array = [1,2,3];

for (var i = 0; i < name\_array.length; i++){

document.**write**(name\_array[i]);

}

### 2.6 Collection Set

Đây là kiểu dữ liệu tập hợp Set với giá trị truyền vào tùy ý theo tốc độ xử lý nhanh chóng.

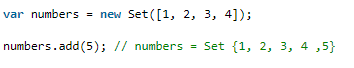
**Khởi tạo**

let set = new Set();



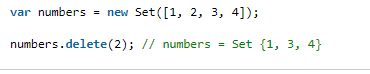
**Thêm phần tử**

set.add(Giá trị);



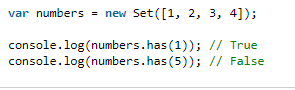
**Xóa phần tử**

set.delete(giá trị);



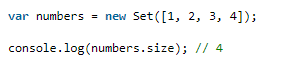
**Kiểm tra tồn tại của giá trị**

set.has(value);



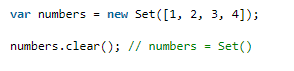
**Đếm tổng số phần tử**

set.size;

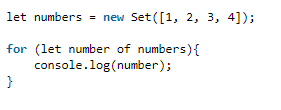


**Xóa toàn bộ phần tử**

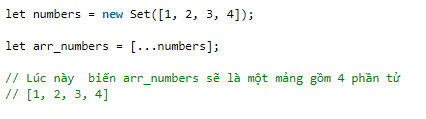
set.clear();



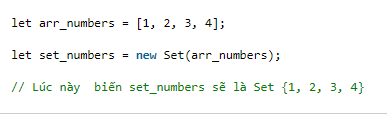
**Xử dụng vòng lăp**



**Chuyển đổi set sang array**



**Chuyển đổi array sang set**



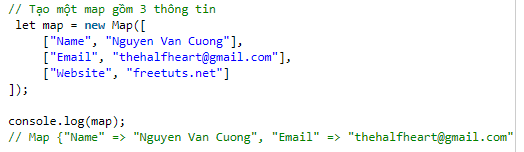
### 2.7 Collection Map

Map là một kiểu dữ liệu tương tự set, tuy nhiên map thì có cấu trúc dạng key=>value còn với set chỉ có value.

**Khởi tạo**

let map = new Map();

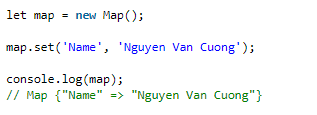
VD



**Thêm phần tử**

map.set(‘Key’, ‘Value’);

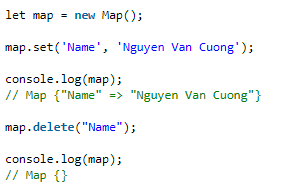
VD



**Xóa phần tử**

map.delete(‘Name’);

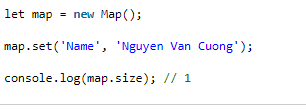
VD



**Đếm tổng số phần tử**

map.size

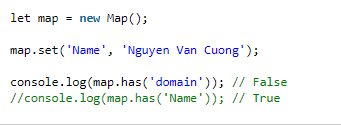
VD



**Kiểm tra tồn tại**

map.has(‘Name’);

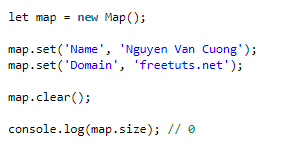
VD



**Xóa toàn bộ phần tử**

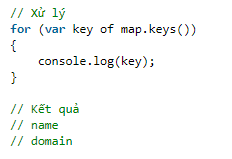
map.clear();

VD

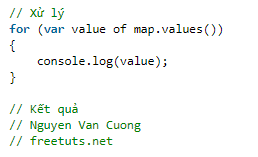


**Các hàm hỗ trợ trong map**

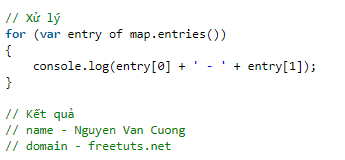
- Hàm keys lấy danh sách các keys



- Hàm values() lấy danh sách values

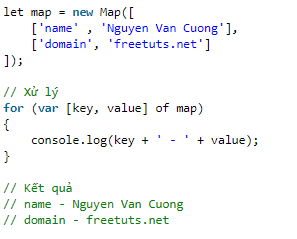


- Hàm entries() lấy cả key và values

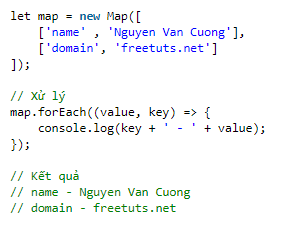


**Các vòng lặp dùng trong map**

- Dùng vòng lặp for



- Sử dụng hàm forEach



### 2.8 Class (Đình Cường)

Lúc trước thì javascript tạo class bằng function nhưng đến es6 thì java script mới có class đúng nghĩa .

**Cách tạo một class**

class **TenClass** {

}

**Khởi tạo một class**

new TenClass();

**Phương thức khởi tạo trong class**

Để xây dựng phương thức khởi tạo ta sử dụng từ khóa constructor

VD

class **Employee** {

*// Phuong thuc khoi tao*

constructor (name, age) {

this.name = name;

this.age = age;

}

};

*// Khởi tạo với tham số trong phương thức khởi tạo*

new **Employee**("Lê Đình Cường", 20);

**Kế thừa trong class**

Để kế thừa trong lúc khai báo tên class của lớp con ta sử dụng từ khóa extends sau đó là tên class B cần kế thừa.

class **A** extends **B** {

}

**Gọi phương thức lớp cha trong lớp con**

Trong es6 để thực hiện gọi các phương thức trong lớp cha khi đang ở lớp con mà phương thức đó đã rewrite trong lớp con thì chúng ta sử dụng keyword với từ khóa super

supert.tenPhuongThucCuaLopCha();

**Sử phương thức tỉnh trong Class**

Trong es6 cũng hỗi trợ chúng ta khai báo các thành phần tĩnh cho đối tượng bằng cách sử dụng từ khóa static trước tên phương thức.

class **Employee** {

static **defaultEmployee** () {

return "Đây là phương thức tĩnh";

}

};

// Gọi phương thức tỉnh đó

className.**staticMethodName**();

**Xây dựng phương thức setter và getter**

Để xây dựng phương thức getter và setter thì trước tên hàm đó ta bổ sung từ khóa set hoặc get tương ứng.

class **Employee** {

// Khởi tạo

constructor (name, age) {

this.name = name;

this.age = age;

}

// Phương thức setter

set **employeeName** (name) {

this.name = name;

}

// Phương thức getter

get **employeeName** () {

return this.name;

}

};

*//khởi tạo đối tượng*

var employee = new **Employee**();

*//setter*

employee.employeeName = "Lê Đình Cường"

*//getter*

employee.employeeName;

*//kết quả: Lê Đình Cường*

### 2.9 Object (Đức Phương)

Object trong java script có thể được tạo theo một số cách khác nhau. Object literal là kiểu cú pháp tạo object sử dụng cặp dấu ngoặ {} và bên trong đó là danh sách các property của object.

**Cú pháp xây dựng một củ:**

VD

var name = “John”;

var myObj = {name: name};

**var** myObj = { name: name,

sayHi: **function** () {

console.log("Hi, my nam is " + **this**.name)

}

};

**Cú pháp mới trong es6**

// Đối với các thuộc tính có tên trùng với giá trị thì chúng ta có thể không thay thế tên là giá trị luôn.

**let** name = "John";

**let** myObj = { name };

// Ví dụ

**let** name = "John";

**let** myObj = { name, sayHi() {

console.log("Hi, my nam is " + **this**.name) }

};

// Ví dụ tạo một object như sau

var student = {

name: "Lê Đình Cường",

age: 20,

**getName**() {

return this.name;

},

**getAge**() {

return this.age;

}

};

**console**.**log**(student.**getName**());

*// Lê Đình Cường*

**console**.**log**(student.**getAge**());

*// 20*

**Cách gọi giá trị của một object**

tenObject.tenThuocTinh/tenHam();

## 3. Component trong React Native

### 3.1 State (Đức Phương)

Đây là thành phần nội bộ của một Component trong react native sử dụng state với các component với dữ liệu sẽ thay đổi. Thường chúng ta khởi tạo các biến state trong hàm khởi tạo. Và sử dụng phương thức setState để cập nhật lại giá trị.

**Để tạo ra các đối tượng trong state ta sử dụng cú pháp sau ở hàm khởi tạo của class.**

this.state = {

tenBien: ‘Giá trị’,

}

Các state là private chỉ có thể thay đổi bên trong bản thân của componnet đó.

Thay đổi giá trị của biến được khai báo trong state thì chúng ta gọi phương thức this.setState({tenBien: giá trị đặt lại}); . Khi chúng ta dùng phương thức setState nó sẽ cũng render component và tất cả component con nằm trong nó.

VD

class **Form** extends **React**.**Component** {

constructor (props) {

super(props)

*// Thiet lap state*

this.state = {

input: ''

}

}

*// Phương thức set State*

**handleChangeInput** = (text) => {

this.**setState**({ input: text })

}

**render** () {

const { input } = this.state

return (

<**View**>

<**TextInput** *style*={{height: 40, borderColor: 'gray', borderWidth: 1}}

*onChangeText*={this.handleChangeInput}

*value*={input}

/>

</**View**>

)

}

}

### 3.2 Props

Props là viết tắt của properties. Ta có thay đổi của một component bằng cách truyền dữ liệu từ bên ngoài vào. Các giá trị của props có thể là 1 object, string, number, function …

Chúng ta thường sử dụng props để truyền giá trị lấy được vào cho thành phần con. Nói dễ hiểu hơn các thành phần con sẽ nhận giá trị được gửi qua các thuộc tính cho thành phần con dữ liệu sẽ đi theo một chiều từ component cha đến component con.

VD

export default class **SanPham** extends **Component** {

constructor(props) {

super(props);

this.state = {

soluong: 0

}

}

**render**() {

return (

<**View** *style*={styles.sp} >

<**Text** >Mã sản phẩm: {this.props.masp}</**Text**>

<**Text**>Tên sản phẩm: {this.props.tensp}</**Text**>

</**View**>

);

}

}

export default class **DanhSachSP** extends **Component** {

**render**() {

return (

<**View** >

<**SanPham** *masp*="SP001" *tensp*="Áo"/>

<**SanPham** *masp*="SP002" *tensp*="Quần"/>

</**View**>

);

}

}

### 3.3 Style

Với react native chúng ta không sử dụng ngôn ngữ đặc biệt hoặc một cú pháp đặc biệt để định nghĩa styles. style của ứng dụng sử dụng bằng JavaScript. Các giá trị để thiết lập style có giá trị giống với CSS tương tự với web. Nhưng các thuộc tính trong CSS lại viết theo kiểm đó là camelCasing. VD backgroundColor.

Chúng ta có thể sử dụng style như là một mảng.

Chúng ta có thể định nghĩa style tại một nơi khác thường đặt là StyleSheet.create

VD:

Sử dụng style

import React, { Component } from 'react';

import { AppRegistry, StyleSheet, Text, View } from 'react-native';

export default class **LotsOfStyles** extends **Component** {

**render**() {

return (

<**View**>

<**Text** *style*={styles.red}>just red</**Text**>

<**Text** *style*={styles.bigblue}>just bigblue</**Text**>

<**Text** *style*={[styles.bigblue, styles.red]}>bigblue, then red</**Text**>

<**Text** *style*={[styles.red, styles.bigblue]}>red, then bigblue</**Text**>

</**View**>

);

}

}

const styles = **StyleSheet**.**create**({

bigblue: {

color: 'blue',

fontWeight: 'bold',

fontSize: 30,

},

red: {

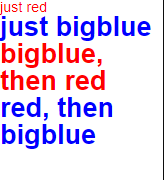
color: 'red',

},

});

*// skip this line if using Create React Native App*

AppRegistry.**registerComponent**('AwesomeProject', () => LotsOfStyles);



### 3.4 Thuộc tính Flex Dimensions

Sử dụng flex trong style của component để mở mở rộng và thu hẹp tự động dựa trên không gian có sẵn. Nó giúp chia tỉ lệ giữa các thành phần.

Thông thường chúng ta sử dụng flex: 1 nó sẽ giúp cho thành phần fill tất cả không gian còn lại với điều kiện các thành phần khác chúng ta không chia flex.

Nếu có nhiều flex trong một view thì thằng nào có flex lớn hơn thì sẽ chiếm nhiều không gian hơn thành phần khác.

VD

import React, { Component } from 'react';

import { AppRegistry, View } from 'react-native';

export default class **FlexDimensionsBasics** extends **Component** {

**render**() {

return (

*// Try removing the `flex: 1` on the parent View.*

*// The parent will not have dimensions, so the children can't expand.*

*// What if you add `height: 300` instead of `flex: 1`?*

<**View** *style*={{flex: 1}}>

<**View** *style*={{flex: 1, backgroundColor: 'powderblue'}} />

<**View** *style*={{flex: 2, backgroundColor: 'skyblue'}} />

<**View** *style*={{flex: 3, backgroundColor: 'steelblue'}} />

</**View**>

);

}

}



### 3.5 Tìm hiểu Flexbox (Đình Cường)

Đây là một component có thể xác định và bố trí các con của nó bằng cách sử dụng thuật toán flexbox. Flexbox được thiết kế để cung cấp bố cục nhất quán trên các kích thước màn hình khác nhau.

Chúng ta thường sử dụng các flexbox sau: flexDirection, alignItems, justtifyContent để đạt được bố cục đúng.

Flexbox làm việc cùng một cách trong React Native hoạt động trong giống css on web.

#### 3.5.1 Flex Direction

Thuộc tính Flex Direction là một thành phần trong thẻ style có hai giá trị định nghĩa hai hướng đó là horizontally với giá trị row và vertically với giá trị column . Sử dụng thuộc tính style này cho thẻ cha để định hướng chiều hiển thị cho các thẻ con bên trong nó.

export default class **FlexDirectionBasics** extends **Component** {

**render**() {

return (

*// Try setting `flexDirection` to `column`.*

<**View** *style*={{flex: 1, flexDirection: 'row'}}>

<**View** *style*={{width: 50, height: 50, backgroundColor: 'powderblue'}} />

<**View** *style*={{width: 50, height: 50, backgroundColor: 'skyblue'}} />

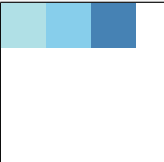
<**View** *style*={{width: 50, height: 50, backgroundColor: 'steelblue'}} />

</**View**>

);

}

};



#### 3.5.2 Justify Content

Sử dụng JustifyContent vào component cha để xác định sử phân bố của các thẻ con dọc theo trục chính. Các thẻ con có thể gắn ở đầu, ở giữa và phía cuối hoặc một khoảng cách đều nhau.

Các giá trị của thuộc tính này là : flex-start, center, flex-end, space-around, space-between, space-evenly.

VD

import React, { Component } from 'react';

import { AppRegistry, View } from 'react-native';

export default class **JustifyContentBasics** extends **Component** {

**render**() {

return (

*// Try setting `justifyContent` to `center`.*

*// Try setting `flexDirection` to `row`.*

<**View** *style*={{

flex: 1,

flexDirection: 'column',

justifyContent: 'space-between',

}}>

<**View** *style*={{width: 50, height: 50, backgroundColor: 'powderblue'}} />

<**View** *style*={{width: 50, height: 50, backgroundColor: 'skyblue'}} />

<**View** *style*={{width: 50, height: 50, backgroundColor: 'steelblue'}} />

</**View**>

);

}

};

*// skip this line if using Create React Native App*

AppRegistry.**registerComponent**('AwesomeProject', () => JustifyContentBasics);



#### 3.5.3 Align Items

Thêm alignItems để xác định kiểu của một thành phần căn chình các thẻ con dọc theo trục thứ cấp (Tức là hai giá trị row và column).

Các giá trị của thuộc tính này là: flex-start, center, flex-end, stretch.

VD

import React, { Component } from 'react';

import { AppRegistry, View } from 'react-native';

export default class **AlignItemsBasics** extends **Component** {

**render**() {

return (

*// Try setting `alignItems` to 'flex-start'*

*// Try setting `justifyContent` to `flex-end`.*

*// Try setting `flexDirection` to `row`.*

<**View** *style*={{

flex: 1,

flexDirection: 'column',

justifyContent: 'center',

alignItems: 'stretch',

}}>

<**View** *style*={{width: 50, height: 50, backgroundColor: 'powderblue'}} />

<**View** *style*={{height: 50, backgroundColor: 'skyblue'}} />

<**View** *style*={{height: 100, backgroundColor: 'steelblue'}} />

</**View**>

);

}

};

*// skip this line if using Create React Native App*

AppRegistry.**registerComponent**('AwesomeProject', () => AlignItemsBasics);

### 3.6 TextInput (Đình Cường)

TextInput là một component cơ bản cho phép người dùng nhập văn bản nó tương tự như TextBox mà chúng ta đã từng học ở trong ngôn ngữ khác. Nó có hai phương thức onChangeText có chức năng gọi môi khi văn bản thay đổi và một prop onSubmitEditing có chức năng gọi khi văn bản được gửi.

VD

constructor(props) {

super(props);

this.state = {text: ''};

}

<**TextInput**

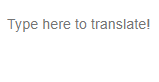
*style*={{height: 40}}

*placeholder*="Type here to translate!"

// onChangeText nó gọi phương thức arrow đó.

*onChangeText*={(text) => this.**setState**({text})}

/>



**Các Props của TextInput**

- allowFontScaling (bool)

// Chỉ định xem font chữ có nên mở rộng hỗ trợ cài đặt trợ năng hay không. mặc định là true

- autoCapitalize

// Có thể yêu cầu text input tự động viết hoa các ký tự nhất định.

- autoCorrect (bool)

// Mặc định đã được bật nó giúp sửa lỗi chính tả.

- autoFocus (bool)

// Tự động trỏ vào khi TextInput này xuất hiện

- blurOnSubmit (bool)

- clearTextOnFocus (bool) (ios)

- TYPE (IOS)

- defautlValue (String)

- disableFullscreenUI (bool)

- keyboardType

// Quy định kiểu hiện thị của bàn phím

+ default

+ nuber-pad

+ decimal-pad

+ numeric

+ email-address

+ phone-pad

- maxLength (số)

- multiline (bool)

- numberOfLines

- onBlur (function)

- onChange (function)

// Gọi phương thức khi text bắt đầu thay đổi

- onChangeText (Function)

// Gọi lại khi văn bản đầu thay đổi

- onEndEditing (function)

// Dược gọi khi nhập xong

- onFocus (function)

// Gọi sự kiện chọn input

- onKeyPress (function)

- onScroll (Function)

- onSubmitEditing (Function)

- placeholder (String)

- placeholderTextColor (color)

- value (String)

- inlineImageLeft

Dùng để hiển thị hình ảnh bên trái text box

<TextInput

inlineImageLeft='search\_icon'

/>

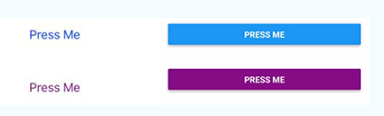
**Các methods**

- clear();

- isFocused();

### 3.7 Button

Tùy vào hệ điều hành dùng button này sẽ hiển thị với giao diện khác nhau. Các button là các nút được định nghĩa mặc định trong các hệ điều hành.



**Tạo một button**

<**Button**

*onPress*={() => {

Alert.**alert**('You tapped the button!');

}}

*title*="Press Me"

/>

VD

import React, { Component } from 'react';

import { Alert, AppRegistry, Button, StyleSheet, View } from 'react-native';

export default class **ButtonBasics** extends **Component** {

// Cái này ta sử dụng function của class

**\_onPressButton**() {

Alert.**alert**('You tapped the button!')

}

**render**() {

return (

<**View** *style*={styles.container}>

<**View** *style*={styles.buttonContainer}>

<**Button**

*onPress*={this.\_onPressButton}

*title*="Press Me"

/>

</**View**>

<**View** *style*={styles.buttonContainer}>

<**Button**

*onPress*={this.\_onPressButton}

*title*="Press Me"

*color*="#841584"

/>

</**View**>

<**View** *style*={styles.alternativeLayoutButtonContainer}>

<**Button**

*onPress*={this.\_onPressButton}

*title*="This looks great!"

/>

<**Button**

*onPress*={this.\_onPressButton}

*title*="OK!"

*color*="#841584"

/>

</**View**>

</**View**>

);

}

}

const styles = **StyleSheet**.**create**({

container: {

flex: 1,

justifyContent: 'center',

},

buttonContainer: {

margin: 20

},

alternativeLayoutButtonContainer: {

margin: 20,

flexDirection: 'row',

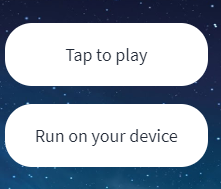
justifyContent: 'space-between'

}

});

*// skip this line if using Create React Native App*

AppRegistry.**registerComponent**('AwesomeProject', () => ButtonBasics);



Các Props quan trọng của Button

- onPress // đây là một function khi ta chạm vào nút

- title // Đây là nhãn của nút.

- color: Màu của nút tùy vào hệ điều hành button sẽ đặt background cho button trong android còn đối ios thì nó sẽ đặt màu chữ.

- disabled: Có hai giá trị là true hoặc false cho phép người dùng tương tác với nút hay không.

### 3.8 Text (Đình Cường)

Là một thành phần để hiện thị text hay nói dể hiểu hơn nó dùng để hiển thị các label với các giá trị mà người dùng không thể sửa trực tiếp.

VD

export default class **TextInANest** extends **Component** {

constructor(props) {

super(props);

this.state = {

titleText: "Bird's Nest",

bodyText: 'This is not really a bird nest.'

};

}

**render**() {

return (

<**Text** *style*={styles.baseText}>

<**Text** *style*={styles.titleText} *onPress*={this.onPressTitle}>

{this.state.titleText}{'\n'}{'\n'}

</**Text**>

<**Text** *numberOfLines*={5}>

{this.state.bodyText}

</**Text**>

</**Text**>

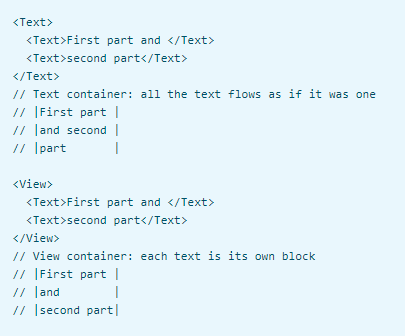
);

}

}



**Các thành phần trong Text**



**Các Props (Thuộc tính đó)**

- selectable

// Để cho phép người dùng chọn văn bản, để sử dụng chức năng sao chép và dán. (Không yêu cầu) giá trị là kiểu bool

- accessibilityHint (String)

// Gợi ý trợ năng giúp người dùng hiểu điều gỉ xảy ra khi họ thực hiện hành động trên phần tử (Không yêu cầu)

- accessibilityLabel (node)

// Ghi dè văn bản được đọc bởi trình đọc màn hình khi người dùng tương tác với phần tử. (Không yêu cầu)

- accessible (Bool)

Khi được đặt thành true cho biết chế độ xem là yếu tố trợ năng. Giá trị mặc định cho phần tử văn bản là đúng (Không yêu cầu)

- ellipsizeMode

- TYPE

- nativeID (String)

Được sử dụng để định vị chế độ xem này từ mã gốc (Không yeu cầu)

- numberOfLines (Số)

- onLayout

{nativeEvent: {layout: {x, y, width, height}}}

- onLongPress

// Gọi sự kiện khi nhấn giữ gọi một phương thức trong hàm hoặc một arrow function

- onPress

// Gọi sự kiện khi nhấn

VD

onPress={() => console.log('1st')}

- pressRetentionOffset

- allowFontScaling (bool)

- style (style)

// Định dạng các css trong đây đó

+ color

+fontStyle  enum('normal', 'italic')

+ fontWeight enum('normal', 'bold', '100', '200', '300', '400', '500', '600', '700', '800', '900')

+ lineHeight

+ textAlign enum('auto', 'left', 'right', 'center', 'justify')

+ textDecorationLine enum('none', 'underline', 'line-through', 'underline line-through')

+ fontFamily

+ letterSpacing (Số)

- testID (String)

// Sử dụng cho cục bộ muốn gọi view này

- disabled (Bool)

- selectionColor (color)

// Màu highlight của text

### 3.9 Image

Đây là một thành phần để hiển thị các loại hình ảnh khác nhau như ảnh mảng, ảnh tài nguyên, ảnh từ bộ nhớ, hoặc ảnh camera.

*// Lấy image từ dữ liệu cục bộ*

<**Image**

*source*={**require**('/react-native/img/favicon.png')}

/>

// Lấy image từ một trang web

<**Image**

*style*={{width: 50, height: 50}}

*source*={{uri: 'https://facebook.github.io/react-native/docs/assets/favicon.png'}}

/>

Các Props thường dùng cho Image

- Style : Dùng để định kiểu cho thẻ image này tương tự như cách viết css.

- boderColor: color

- boderRadius: number // Bo góc hình

- opacity: number // Độ mờ của hình ảnh

- onLoad (function) // Khi ảnh load thành công.

- onLoadEnd (function) // Khi ảnh đã load thành công hoặc thất bại.

- onLoadStart(function) // Khi ảnh bắt đầu load

- resizeMode: cover | contain | stretch | repeat | center

- source // Đường dẫn nguồn của hình ảnh

### 3.10 TouchableOpacity (Đình Cường)

Đây là thành phần bao bọc lấy một giao diện để biến thành phần này thành thành phần có thể thao tác. Khi nhấn vào thành phần này sẽ có hiệu ứng giảm mờ đi. Nó cũng có các thuộc tính khá giông button.

<**TouchableOpacity** *onPress*={this.\_onPressButton}>

<**Image**

*style*={styles.button}

*source*={**require**('./myButton.png')}

/>

</**TouchableOpacity**>

### 3.11 CheckBox (Đưc Phương)

Đây là thành phần không có sẵn trong thư viện mặc định khi tạo ứng dụng mà ta phải import từ bên ngoài

import { CheckBox } from 'react-native-elements'

<CheckBox

title='Click Here'

checked={this.state.checked}

/>

<https://react-native-training.github.io/react-native-elements/docs/checkbox.html>

**Các thuộc tính quan trọng:**

- checked (bool) // Giá trị true thì checkbox này đã được check sẵn.

- size: number // Kích thước của check box.

- iconRight: bool // Di chuyển icon sang bên phải

- center: bool // Cắn giữa

- title: string // Tiêu đề của check box

- onPress // Sự kiện khi nhấn

- onLongPress // Sự kiện nhấn giữ

- onLongIconPress // Sự kiện khi nhấn vào icon check box.

### 3.12 Radio Button (Đức Phương)



Đây cũng là thành phần mà ta phải cài đặt từ bên ngoài

**Để thêm radio button ta phải chạy câu lệnh sau**

npm i radio-button-react-native --save

Sau đó import nó vào đồ án

**import** RadioButton **from** 'radio-button-react-native';

**Cơ bản của một Radio Button**

**<**RadioForm

  radio\_props**=**{radio\_props}

  initial**=**{0}

  onPress**=**{(value) => {this.setState({value**:**value})}}

**/>**

### 3.13 List View (Đình Cường)

ListView là một thành phần cốt lõi được thiết kế để hiện thị hiệu quả đối với dữ liệu cần hiển thị dạng danh sách các dữ liệu sẽ được hiển thị theo chiều dọc.

Có hai thuộc tính quan trong list view đó là dataSource và renderRow. Datasource là dữ liệu sẽ được hiển thị lên. RenderRow sẽ trả về giao diện giữa các list item.

VD

class **MyComponent** extends **Component** {

constructor() {

super();

const ds = new **ListView**.**DataSource**({**rowHasChanged**: (r1, r2) => r1 !== r2});

this.state = {

*// Tạo một đối tượng data source với dữ liệu là một mảng*

dataSource: ds.**cloneWithRows**(['row 1', 'row 2']),

};

}

**render**() {

return (

<**ListView**

*// Set nguồn dữ liệu của list view*

*dataSource*={this.state.dataSource}

*// Lấy từng dữ liệu từ mảng hiển thị lên mỗi dòng sẽ là một thẻ text*

*renderRow*={(rowData) => <**Text**>{rowData}</**Text**>}

/>

);

}

}

**Các props (Các thuộc tính trong thẻ list view)**

- dataSource

Đây là thuộc tính bắt buộc với giá trị là ListViewDataSource

var ds = new ListView.DataSource({rowHasChanged: this.\_rowHasChanged});

- initialListSize (Số)

Quy định số hàng hiển thị trên màn hình lúc đầu.

- onEndReachedThreshold (Số)

Quy định số pixel hiển thị trên thiết bị vật lý

- pageSize (Số)

Số dòng sẽ được render cho mỗi lần loop (Mỗi lần kéo). Đây là thuộc tính yêu cầu

- renderRow (Function)

(rowData, sectionID, rowID, highlightRow) => renderable

Lấy một mục dữ liệu từ nguồn và các id của nó và sẽ trả về một thành phần có thể hiển thị dưới dạng hàng. Theo mặc định dữ liệu chính là những gì được đưa vào nguồn dữ liệu. List view có thể đánh dấu hàng được chọn bằng highlighRow(sectionID, rowID).

Những thiết lập giá trị boolean này của adjacentRowHighlighted in renderSeparator cho phép bạn điều khiển các dòng riêng biệt trên và dưới highlightedrow.

- renderScrollComponent (function)

(props)=> renderable

Một phương thức trả về thành phần được cuộn trong danh sách các hàng đã được render. Mặc định để để trả về một ScrollView với giá trị được gắn vô props.

- scrollRenderAheadDistance (Số)

Hiển thị sớm các hàng trên màn hình trước khi nó sẵn sàng hiện dữ liệu

- stickyHeaderIndices

**Các methods**

- getMetrics()

Mở rộng một vài dữ liệu.

- scrollTo()

scrollTo(...args: Array)

Cuộn tới x, y với hiệu ứng

- scrollToEnd()

scrollToEnd(([options]: object));

- flashScrollIndicators()

flashScrollIndicators();

### 3.14 Navigation trong React

Naviagation đây là thành phần giúp chúng ta chuyển đổi giữa các màn hình cũng là thành phần giúp chúng ta chuyển đổi dữ liệu giữa các màn hình với nhau.

**Để sử dụng navigation chúng ta phải trải qua các bước sau:**

- Bước 1: Cài gói navigation vào project thông qua lệnh cmd trỏ vào project

npm install --save react-navigation

npm install --save react-native-gesture-handler

react-native link react-native-gesture-handler

Bước 2: Tiến hành tạo stack navigator có bao nhiêu màn hình thì chúng ta đều khai báo tại đây

<https://reactnavigation.org/docs/en/hello-react-navigation.html>

Bước 3: Sau khi khai báo đây là cách di chuyển giữa các màn hình

- Để di chuyển sang màn hình khác tại sự kiện của component nào đó ta gọi cú pháp như sau:

this.props.navigation.**navigate**('TenManHinhDaDinhNghia')

// Nếu muốn load lại màn hình sau khi đã chuyển đến màn hình đó rồi thì theo cú pháp sau:

his.props.navigation.**push**('ManHinhMuonKhoiTaoLai')

// Trường hợp muốn back lại trường hợp trước đó

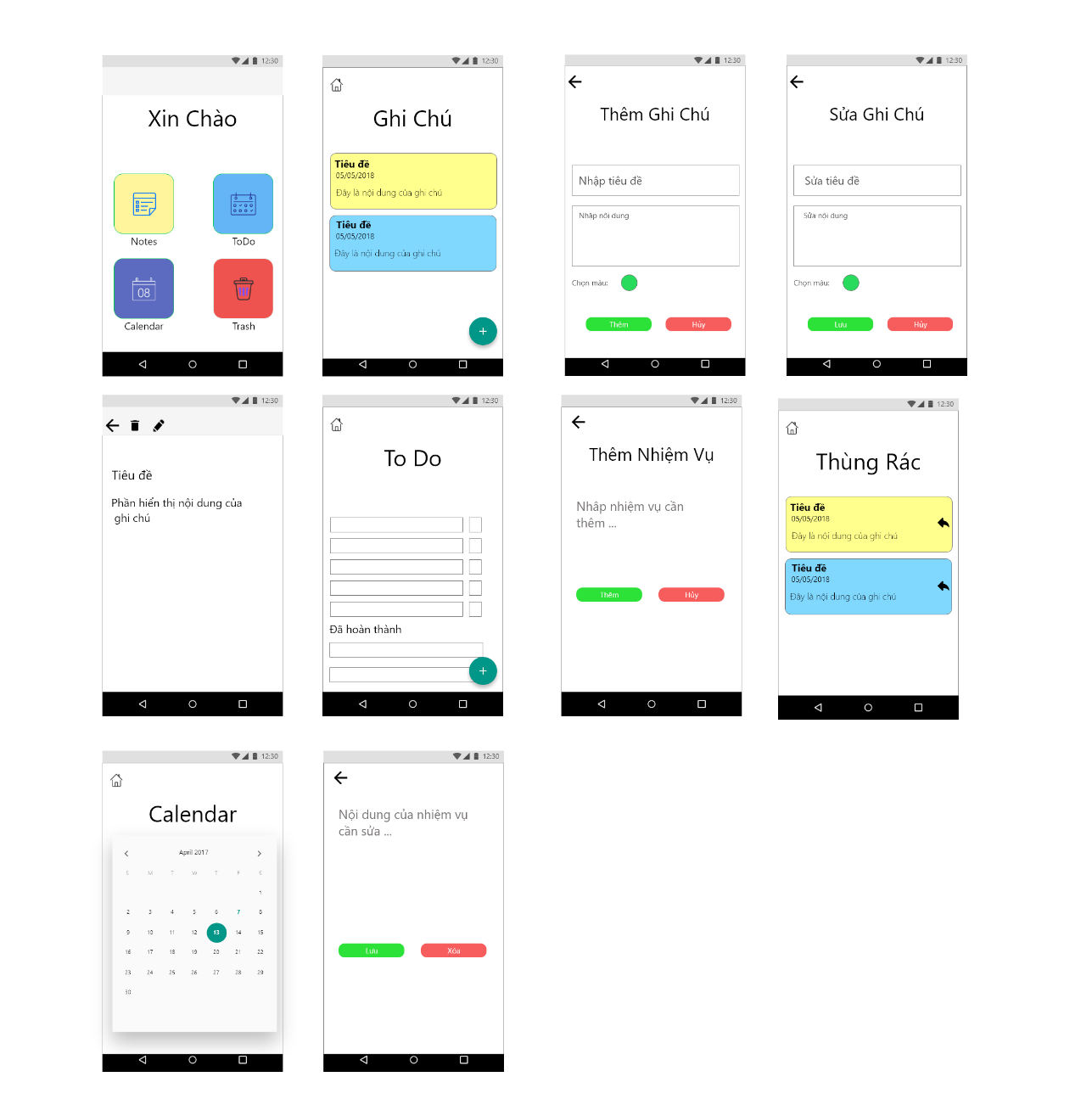
this.props.navigation.**goBack**()

// Trường hợp muốn chuyển dữ liệu giữa các màn hình

this.props.navigation.**navigate**('TenManHinhDaDinhNghia', {key: value})

Đây chỉ là một phần về navigation để nắm rõ kiến thức này cần tham khảo nhiều tài liệu bên ngoài

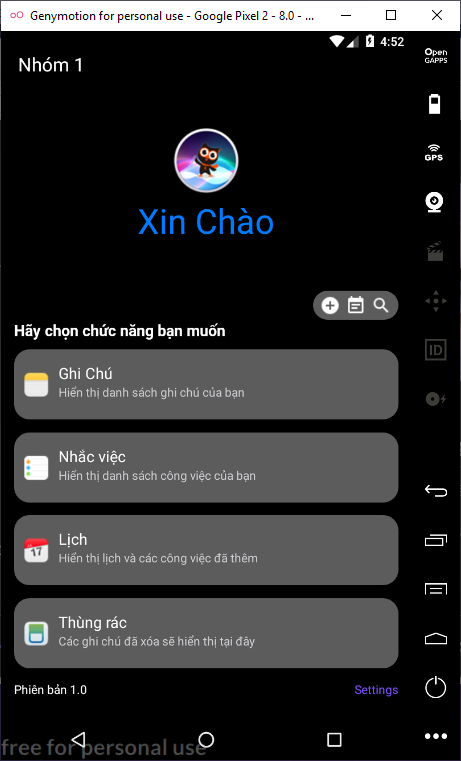
# II. Phân Tích Thiết Kế Hệ Thống



## 1 Phân tích hệ thống

### Trang chủ (Đình Cường)

#### Giao diện người dùng



#### Mô tả

Luồng truy cập:

**Home**

**Mô Tả:**

Giao diện này sẽ khởi chạy đầu tiên khi khởi chạy ứng dụng. Mục đích chuyển đổi giữa các chức năng của ứng dụng. Đây là cũng là trang điều hướng cho toàn ứng dụng.

#### Phân tích các thành phần

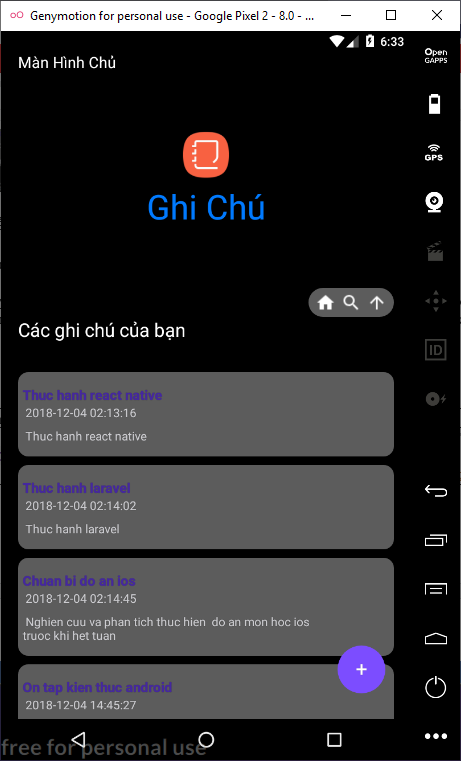
|  |  |  |  |
| --- | --- | --- | --- |
| **Thành phần** | **Mô tả** | **Hành động** | **Yêu cầu** |
| Text | Hiển thị label với giá trị “Nhóm 1” |  | color: #fff |
| Text | Hiển thị tiêu đề của trang |  | color: '#007aff'  fontSize: 40 |
| Text | Chỉ dẫn người dùng chọn các chức năng của ứng dụng. |  | color: '#fff'  marginBottom: 10,  fontSize: 18  fontWeight: 'bold' |
| TouchableOpacity | Thành phần này bao bọc lấy một image để biến nó một button | Chuyển đến màn hình thêm ghi chú | addnoteicon.png |
| TouchableOpacity | Thành phần này bao bọc lấy một image làm button image | Chuyển đến màn hình thêm nhiệm vụ | addtaskicon.png |
| TouchableOpacity | Thành phần này bao bọc lấy một image để làm button image | Chuyển đến màn hình tìm kiếm ghi chú | searchicon.png |
| Image | Hiển thị icon của các chức năng chính của ứng dụng. |  | note.png  reminder.png  calendar.png  trash.png |
| Text | Hiển thị tên của chức năng |  | color: '#fff'  fontSize: 18 |
| Text | Hiển thị mô tả nhanh cho chức năng. |  | color: '#ceced2' |
| TouchableOpacity |  | Điều hướng đến màn hình hiển thị danh sách ghi chú | flexDirection: 'row'  paddingTop: 15  paddingBottom: 15  borderRadius: 18  borderColor: '#fff' backgroundColor: '#5C5C5C'  marginBottom: 15 |
| TouchableOpacity |  | Điều hướng đến màn hình hiển thị danh sách các nhiệm vụ. | flexDirection: 'row'  paddingTop: 15  paddingBottom: 15  borderRadius: 18  borderColor: '#fff' backgroundColor: '#5C5C5C'  marginBottom: 15 |
| TouchableOpacity |  | Điều hướng màn hình hiển thị lịch | flexDirection: 'row'  paddingTop: 15  paddingBottom: 15  borderRadius: 18  borderColor: '#fff' backgroundColor: '#5C5C5C'  marginBottom: 15 |
| TouchableOpacity |  | Điều hướng đến màn hình hiển thị các ghi chú đã bị xóa. | flexDirection: 'row'  paddingTop: 15  paddingBottom: 15  borderRadius: 18  borderColor: '#fff' backgroundColor: '#5C5C5C'  marginBottom: 15 |
|  | Đây là logo của ứng dụng |  | Hình ảnh: logoapp.png |
| TouchableOpacity | Dùng làm button chuyển đến màn hình cài đặt của ứng dụng | Chuyển đến màn hình cài đặt của ứng dụng. |  |
| Text | Hiển thị phiên bản hiện tại của ứng dụng |  | color: #fff |

#### 1.1.4 Các xử lý

|  |  |
| --- | --- |
| Chuyển sang màn hình NoteScreen |  |
| Chuyển sang màn hình Nhiệm vụ |  |
| Chuyển sang màn hình hiển thị lịch |  |
| Chuyển sang màn hình thùng rác chứa các ghi chú đã xóa |  |
| Chuyển sang màn hình cài đặt |  |
| Hành động nhấn nút search |  |
| Hành động chuyển sang màn hình thêm nhanh ghi chú |  |

### Hiển thị danh sách ghi chú NoteScreen (Đình Cường)

#### 1.2.1 Giao diện người dùng



#### 1.2.2 Mô tả

**Màn hình home 🡪 Hiển thị danh sách ghi chú**

Màn hình này sẽ hiển thị dữ liệu các ghi chú được lấy từ cơ sở dữ liệu và hiển thị nội dung của các ghi chú này dưới dạng list. Tại màn hình này có thể chuyển về màn hình home hoặc chuyển tới màn hình thêm ghi chú và màn hình chi tiết ghi chú.

#### 1.2.3 Phân tích các thành phần

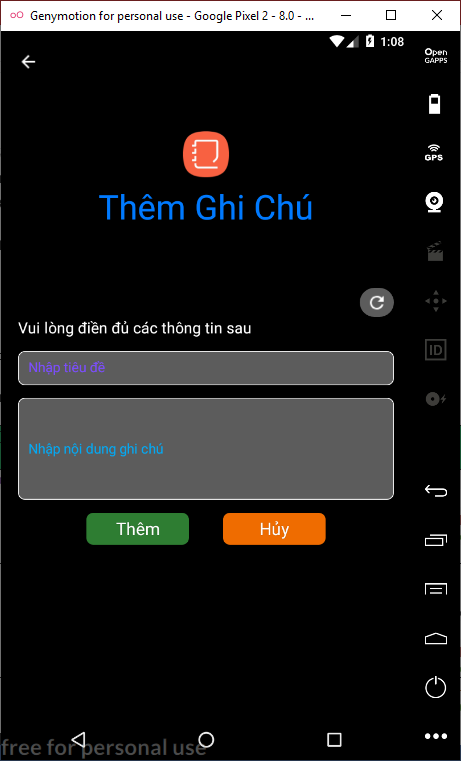
|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Mô tả | Hành động | Yêu cầu |
| TouchableOpacity |  | Điều hướng về màn hình chủ | color: "#fff"  fontSize: 18  marginLeft: 20 |
| Image | Hiển thị icon của màn hình |  | width: 56  height: 56  *source*={**require** ('./notei.png') |
| Text | Hiển thị tiêu đề đại diện cho chức năng của màn hình |  | fontSize: 40  color: '#007aff' |
| Text | Được dùng làm nhãn cho danh sách ghi chú phía bên dưới |  | color: '#fff'  fontSize: 22  marginBottom: 15 |
| TouchableOpacity | Giao diện hiển thị của một ghi chú được lấy từ từ cơ sở dữ liệu. | Điều hướng đến màn hình hiển thị chi tiết ghi chú | backgroundColor: '#5C5C5C'  paddingTop: 15  paddingBottom: 15  paddingLeft: 5  borderRadius: 12,  marginBottom: 10 |
| **ActionButton** |  | Điều hướng đến màn hình thêm ghi chú | *buttonColor*="#7c4dff" |
| Text | Thành phần hiển thị tiêu đề của ghi chú |  | color: "#4527a0"  fontWeight: "bold"  fontSize: 16, |
| Text | Thành phần hiển thị ngày tạo của ghi chú đó |  | marginBottom: 8,  color: '#ceced2' |
| Text | Hiển thị nội dung của ghi chú |  | marginBottom: 8  color: '#ceced2' |
| TouchableOpacity |  | ImageHome: Điều hướng về màn hình chủ  IconSeach: Điều hướng màn hình tìm kiếm ghi chú  Icon arrow up: Trở về đầu danh sách | *source*={**require** ('./homeicon.png')  *source*={**require** ('./searchicon.png')  *source*={**require** ('./upbuttonicon.png') |
| ListView | Hiển thị nội dung các ghi chú lấy từ cơ sở dữ liệu |  |  |

#### 1.2.4 Các xử lý

|  |  |
| --- | --- |
| Hiển thị danh sách các ghi chú |  |
| Khi một ghi chú được nhấn trong danh sách |  |
| Chuyển màn hình NoteAddScreen |  |
| Khi nhấn nút home hoặc nút màn hình chủ |  |
| Khi nhấn nút up |  |

### Thêm ghi chú (Đình Cường)

#### 1.3.1 Giao diện người dùng



#### 1.3.2 Mô tả

Home 🡪 NoteScreen 🡪 NoteAddScreen

Giao diện Thêm ghí chú (NoteAddScreen) xuất hiện khi người dùng nhấn nút cộng từ màn hình home hoặc màn hình xem danh sách các ghi chú (NoteScreen). Tại màn hình này người dùng nhập tiêu đề và nội dung sau đó nhấn nút thêm để ghi chú mới được thêm vào cơ sở dữ liệu của chương trình.

#### 1.3.3 Phân tích các thành phần

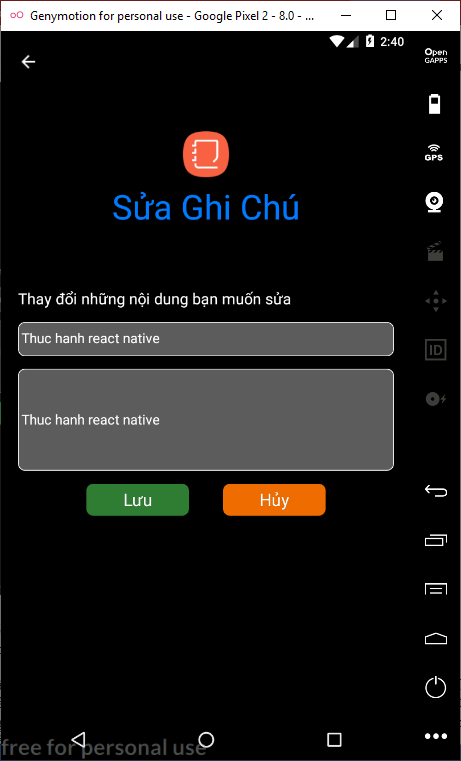
|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Mô tả | Hành động | Yêu cầu |
| TouchableOpacity |  | Điều hướng về màn hình trước đó. | marginLeft: 20  *source*={**require** ('./backicon.png') |
| Image | Hiển thị icon của màn hình |  | width: 56  height: 56 marginBottom: 5  *source*={**require** ('./notei.png') |
| Text | Thành phần dùng để hiển thị tiêu đề của trang |  | fontSize: 40,  color: '#007aff' |
| Text | Nhãn nhắc nhở người dùng điền các trường bên dưới. |  | color: "#fff" fontSize: 18  marginBottom: 15 |
| TextInput | TextBox nhập tiêu đề của ghi chú |  | *placeholder*=" Nhập tiêu đề"  *placeholderTextColor*= "#7c4dff"  *maxLength*={40} |
| TextInput | TextBox nhập nội dung của ghi chú |  | *placeholder*=" Nhập nội dung ghi chú"  *numberOfLines*={5}  *multiline*={true}  *editable*={true}  *maxLength*={75}  *placeholderTextColor*= "#03a9f4" |
| TouchableOpacity |  | Lấy dữ liệu từ các text input để thêm vào cơ sở dữ liệu sau đó điều hướng trở về màn hình hiển thị danh sách các ghi chú | backgroundColor: '#4cd964'  width: 120  paddingTop: 5  paddingBottom: 5  textAlign: 'center',  color: '#fff'  borderRadius: 8  fontSize: 20  marginRight: 40 |
| TouchableOpacity |  | Điều hướng trở về màn hình trước đó | backgroundColor: '#ff9100'  width: 120  paddingTop: 5,  paddingBottom: 5  textAlign: 'center'  color: '#fff'  borderRadius: 8  fontSize: 20 |
| TouchableOpacity | Thẻ này bao bọc lấy một thẻ image để làm một button | Xóa dữ liệu cho TextInput Tiêu đề và Text Input nội dung. | Image  refreshicon.png |

#### 1.3.4 Các xử lý trong màn hình

|  |  |
| --- | --- |
| Thêm ghi chú |  |
| Khi nhấn nút hủy |  |
| Khi nhấn nút back |  |
| Nhấn nút refresh |  |

### Sửa ghi chú

#### 1.4.1 Giao diện người dùng



#### 1.4.2 Mô tả

Home 🡪 NoteScreen 🡪 DetailNote 🡪 EditNoteScreen.

Màn hình này sẽ nhận giá trị của ghi chú được người dùng chọn từ màn hình DetailNote sau đó các giá trị này sẽ được hiển thị lên các TextInput tương ứng sau khi thực hiện thay đổi các trường tương ứng nhấn nút Lưu các giá trị thay đổi sẽ được cập nhật lại vào trong cơ sở dữ liệu sau đó trở về màn hình NoteScreen. Nhấn nút hủy để trở về màn hình trước đó cụ thể là màn hình DetailNote.

#### 1.4.3 Phân tích các thành phần

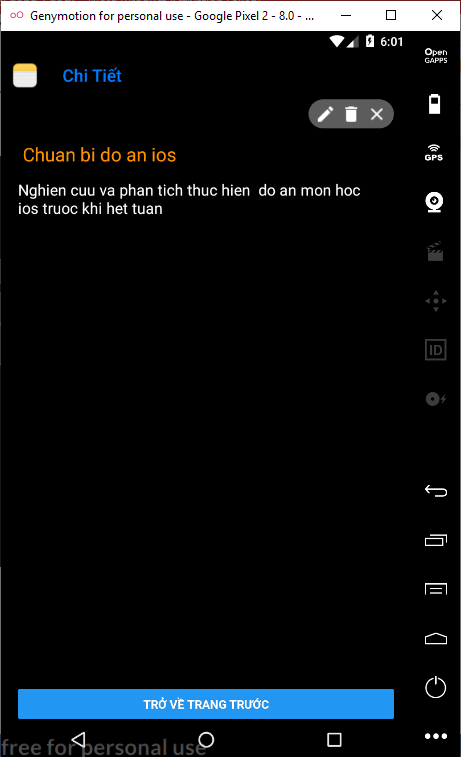
|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Mô tả | Hành động | Yêu cầu |
| TouchableOpacity |  | Điều hướng về màn hình trước đó. | marginLeft: 20  *source*={**require** ('./backicon.png') |
| Image | Hiển thị icon của màn hình |  | width: 56  height: 56 marginBottom: 5  *source*={**require** ('./notei.png') |
| Text | Thành phần dùng để hiển thị tiêu đề của trang |  | fontSize: 40,  color: '#007aff' |
| Text | Nhãn nhắc nhở người dùng điền các trường bên dưới. |  | color: "#fff" fontSize: 18  marginBottom: 15 |
| TextInput | TextBox sửa tiêu đề của ghi chú |  | *placeholder*=" Nhập tiêu đề"  *value*={this.state.titlenote}  *maxLength*={40}  *placeholderTextColor*= "#7c4dff" |
| TextInput | TextBox sửa nội dung của ghi chú |  | *placeholder*=" Nhập nội dung ghi chú"  *numberOfLines*={5}  *multiline*={true} *editable*={true}  *maxLength*={75}  *value*={this.state.contentnote}  *placeholderTextColor*= "#03a9f4" |
| TouchableOpacity |  | Lưu dữ liệu lấy từ các text input vào cơ sở dữ liệu sau đó điều hướng về trang hiển thị danh sách các ghi chú | background = ‘green’  color = ‘#fff’ |
| TouchableOpacity |  | Điều hướng trở về màn hình trước đó | background = ‘Yellow’  color = ‘#fff’ |

#### 1.4.4 Các xử lý tại màn hình

|  |  |
| --- | --- |
| Sửa ghi chú |  |
| Nhận dữ liệu từ màn hình DetailNote |  |
| Nhấn nút hủy |  |
| Nhấn nút back |  |

### Chi tiết ghi chú

#### 1.5.1 Giao diện người dùng



#### 1.5.2 Mô tả

**Home 🡪 NoteScreen 🡪 DetailNote**

Màn hình này sẽ hiện thị tiêu đề và nội dung của ghi chú đó được chọn từ danh sách. Tại màn hình này chúng ta có thể điều hướng đến màn hình EditNoteScreen, hoặc chuyển về màn hình trước đó cụ thể đây là màn hình NoteScreen. Tại màn hình này ta có thể xóa ghi chú được chọn sau khi xóa xong màn hình sẽ điều hướng về màn hình NoteScreen.

#### 1.5.3 Phân tích các thành phần

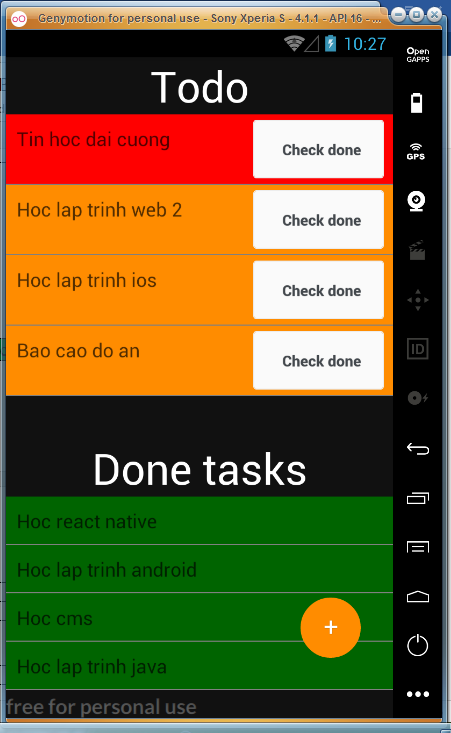
|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Mô tả | Hành động | Yêu cầu |
| **ToolbarAndroid** | Hiển thị title của trang và icon của trang |  | *title*="Chi Tiết"  *navIcon*={**require** ('./note.png')}  *titleColor*={'#007aff' |
| Text | Hiển thị tiêu đề của ghi chú được chọn |  | fontSize: 22  marginBottom: 15  marginTop: 15  color: '#ff9100' |
| Text | Hiển thị nội dung của ghi chú được chọn |  | fontSize: 18,  color: '#fff' |
| Button |  | Điều hướng về màn hình trước đó | *title*="Trở về trang trước" |
| TouchableOpacity |  | Điều hướng đến màn hình sửa ghi chú | *source*={**require** ('./editicon.png') |
| TouchableOpacity |  | Thực hiện xóa ghi chú được chọn sau đó chuyển về màn hình hiển thị danh sách các ghi chú | *source*={**require** ('./deleteicon.png') |
| TouchableOpacity |  | Đóng màn hình hiển thị chi tiết chuyển về màn hình hiển thị các ghi chú | *source*={**require** ('./close.png') |
|  |  |  |  |

#### 1.5.4 Các xử lý trên màn hình

|  |  |
| --- | --- |
| Xóa ghi chú |  |
| Nhận dữ liệu từ màn hình NoteScreen |  |
| Khi nhấn nút edit |  |
| Khi nhấn nút close hoặc nút trở về |  |
|  |  |

### Hiển thị các nhiệm vụ (Đức Phương)

#### 1.6.1 Giao diện người dùng



#### 1.6.2 Mô tả

#### 1.6.3 Phân tích các thành phần

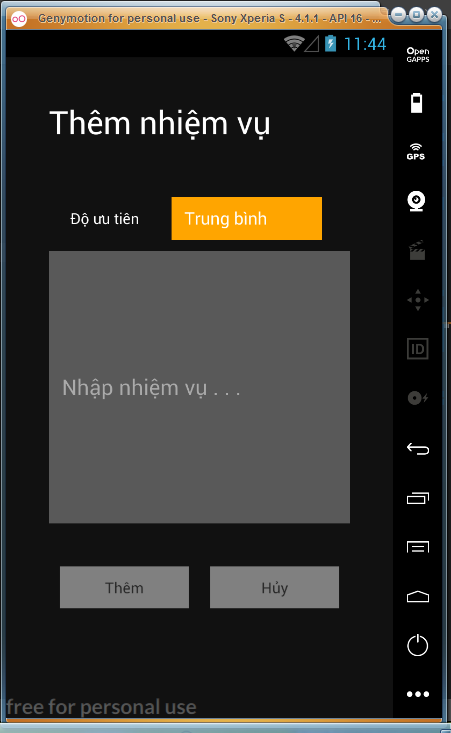
|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Mô tả | Hành động | Yêu cầu |
|  | Tên tác vụ cần thực hiện | Chuyển sang màn hình chỉnh sửa |  |
|  | Nút hoàn tất | Đánh dấu tác vụ đã hoàn tất |  |
|  | Tên tác vụ đã hoàn tất |  |  |
|  | Nút thêm tác vụ mới | Chuyển sang màn hình thêm tác vụ |  |

#### 1.6.4 Các xử lý trên màn hình

|  |  |
| --- | --- |
| **Sự kiện** | **Workflow** |
| Tải trang | 1. Nhận dữ liệu từ serve thông qua file getTodo.php query “select” 2. Gán dữ liệu vào 2 mảng: mảng todoTasks và doneTasks 3. Bố trí dữ liệu |
| Nhấp vào tên tác vụ cần thực thiện | Đóng gói dữ liệu và gửi qua màn hình chỉnh sửa |
| Nhập vào nút hoàn tất | 1. Xác định id của tác vụ 2. Đặt lại trạng thái của tác vụ thành hoàn tất (state: true) 3. Gửi id và state của tác vụ lên serve thông qua câu lệnh sql update 4. Load lại trang, tải lại dữ liệu |
| Nhấp vào nút thêm mới | Chuyển sang màn hình thêm tác vụ |

### Thêm nhiệm vụ (Đức Phương)

#### 1.7.1 Giao diện người dùng



#### 1.7.2 Mô tả

#### 1.7.3 Phân tích các thành phần

|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Mô tả | Hành động | Yêu cầu |
|  | Độ ưu tiên | Chọn độ ưu tiên |  |
|  | Khung nhập nhiệm vụ | Focus |  |
|  | Nút thêm | Thêm tác vụ vào danh sách |  |
|  | Nút hủy | Quay về trang trước |  |

#### 1.7.4 Các xử lý trên màn hình

|  |  |
| --- | --- |
| **Sự kiện** | **Workflow** |
| Tải trang | Đặt dữ liệu mặc định của độ ưu tiên là 0 (trung bình), của content là rỗng. |
| Nhấp vào nút thêm | 1. Đóng gói dữ liệu thành json gồm có các key: content, priority, date 2. Gửi dữ liệu lên server thông qua query “insert” 3. Quay lại trang danh sách tác vụ và tải lại trang |
| Nhấp vào nút hủy | Quay về trang trước |

### Sửa nhiệm vụ (Đức Phương)

#### 1.8.1 Giao diện người dùng

#### 

#### 1.8.2 Mô tả

#### 1.8.3 Phân tích các thành phần

|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Mô tả | Hành động | Yêu cầu |
|  | Nút hoàn tất. Nhấp chuột chuyển thành màu xanh.  Sẽ bị vô hiệu hóa sau click. | Đồi trạng thái của tác vụ sang “hoàn tất”. |  |
|  | Độ ưu tiên Màu nền: ● cao: màu đỏ ● trung bình: màu cam | Chuyển đổi độ ưu tiên của tác vụ |  |
|  | Tên tác vụ  Có thể thay đổi |  |  |
|  | Lưu thay đổi | Lưu những thay đổi của thuộc tính của tác vụ |  |
|  | Hủy thay đổi | Quay về trang trước |  |

#### 1.8.4 Các xử lý trên màn hình

|  |  |
| --- | --- |
| **Sự kiện** | **Workflow** |
| Tải trang | 1. Nhận bộ dữ liệu từ trang danh sách gồm có các key: id, content, state, priority 2. Bố trí dữ liệu |
| Nhấp vào nút Mark this as done | Cập nhật thuộc tính state từ 0 thành 1 |
| Nhấp vào nút Priority | Xổ danh sách độ ưu tiên |
| Nhấp vào nút Save | Cập nhật bộ dữ liệu và gửi lên server thông qua file editTodo query “update” |
| Nhấp vào nút hủy | Quay lại màn hình trước |

### 1.9 Chi tiết nhiệm vụ (Đức Phương)

#### 1.9.1 Giao diện người dùng

#### 1.9.2 Mô tả

#### 1.9.3 Phân tích các thành phần

|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Mô tả | Hành động | Yêu cầu |
|  |  |  |  |
|  |  |  |  |

#### 1.9.4 Các xử lý trên màn hình

|  |  |
| --- | --- |
|  |  |
|  |  |

# III. Kết quả đạt được

## 1 Kết quả đạt được

### 1.1 Kết quả chạy của tính chức năng ghi chú

|  |  |
| --- | --- |
| Xem danh sách ghi chú | Tại màn hình home chọn chức năng ghi chú    Sau khi chọn màn hình hiển thị danh sách các ghi chú hiện lên |
| Thêm ghi chú | Để thêm ghi chú tại màn hình xem danh sách các ghi chú nhấn nút cộng ở phía dưới màn hình ở góc phải    Màn hình thêm ghi chú hiện lên và chúng ta nhập tiêu đề của ghi chú là “Tiêu đề ghi chú mới” và nội dung là “Demo tính năng thêm ghi chú” sau đó nhấn nút thêm lúc này chương trình sẽ chuyển về màn hình notescreen hiển thị danh sách các ghi chú và ta sẽ thấy kết quả tại đây. |
| Xem chi tiết một ghi chú | Tại màn hình xem danh sách ghi chú chọn một ghi chú muốn xem chi tiết    Lúc này chúng ta sẽ được chuyển sang màn hình hiển thị chi tiết ghi chú |
| Sửa ghi chú | Sau khi chọn ghi chú và đang ở màn hình hiển thị chi tiết ghi chú ta nhấn vào biểu tượng cây bút chì để chuyển sang màn hình sửa ghi chú    Màn hình sửa ghi chú sẽ hiện lên với các text input đã được điền nội dung của ghi chú mà ta muốn sửa.    Ở ví dụ này tôi thêm 111 và 222 vào tiêu đề và nội dung của ghi chú muốn sửa sau đó nhấn nút lưu. Chúng ta sẽ được chuyển về màn hình hiển thị danh sách ghi chú để thấy kết quả |
| Xóa ghi chú | Tại màn hình hiển thị danh sách các ghi chú nhấn vào ghi chú muốn xóa để chuyển sang màn hình hiển thị chi tiết    Sau khi chuyển sang màn hình hiển thị chi tiết nhấn vào biểu tượng thùng rác để xóa ghi chú đó khỏi cơ sở dữ liệu của chương trình    Sau khi ghi chú đã được xóa chương trình sẽ chuyển bạn về màn hình hiển thị danh sách các ghi chú |

### Chức năng thêm nhiệm vụ

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

## 2 Các kết luận và kiến nghị

### 2.1 Những điểm đã làm được

- Biết sử dụng các component để xây dựng giao diện cho app.

- Biết sử dụng tài liệu trên mạng để xây dựng hoàn chỉnh chức năng cho chương trình.

- Biết cách xử lý sự kiện cho các component.

- Xây dựng hoàn chỉnh các chức năng của project như đã đề ra lúc đầu.

### 2.2 Những điểm chưa làm được

- Vì thời gian thực hiện đồ án quá ít và thiếu vắng một số thành viên ban đầu nên một số chức năng đề ra lúc ban đầu không được thực hiện.

- Các chức năng đã được xây dựng chưa được xử lý triệt để các tình huống có thể xảy ra khi chạy chương trình.

- Trong quá trình thực hiện xuất hiện vấn đề bất cập trong thiết kế nên giao diện chương trình khác so với thiết kế ban đầu và các thành viên thực hiện chức năng khác nhau nên giao diện các chức năng lớn sẽ không giống nhau.

### 2.3 Các chức năng trong tương lai

- Hoàn thiện các chức năng được đề ra lúc ban đầu.

- Bổ sung thêm chức năng tìm kiếm cho tính năng ghi chú và nhắc việc.

- Thống nhất giao diện các chức năng trong chương trình.

- Xử lý các tình huống chương trình ngoài ý muốn.

# VI. Tài liệu tham khảo

-Tài liệu tham khảo chính

<https://facebook.github.io/react-native/docs/getting-started>

- Navigation

<https://reactnavigation.org/docs/en/navigating.html>

- Life Component

<https://viblo.asia/p/lifecycle-component-trong-reactjs-gGJ59jzxKX2>

<https://5minuteshack.blogspot.com/2018/03/tim-hieu-ve-component-life-cycle-trong-reactjs.html>

- Refs

<https://viblo.asia/p/refs-in-react-tat-ca-nhung-gi-ban-can-biet-V3m5WOgQ5O7>

- State và props

<https://viblo.asia/p/component-state-va-props-trong-react-native-Eb85oM34Z2G>

- Webservice

<https://www.youtube.com/watch?v=MwKrqmvGZws>

<https://www.youtube.com/watch?v=kRuRXAEKKik>

<https://www.youtube.com/watch?v=C2lNYTbbwAA>

<https://www.youtube.com/watch?v=n2vKAJOrmUw&t=14s>

Và nhiều trang cộng đồng hỗ trợ khác…