TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC

KHOA CÔNG NGHỆ THÔNG TIN

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



BÁO CÁO KẾT THÚC MÔN HỌC

Lập Trình Di Động 3

**APP DỰ BÁO THỜI TIẾT**

Giảng viên hướng dẫn: **TRƯƠNG BÁ THÁI**

Sinh viên thực hiện:

1. NGUYỄN THANH PHÚC NGUYÊN
2. NGUYỄN VÕ THÁI SƠN

*Tp. Hồ Chí Minh*, ngày 15 tháng 10 năm 2019

**NHẬT KÝ HOẠT ĐỘNG NHÓM**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **Công việc thực hiện** |
| 1 | Nguyễn Thanh Phúc Nguyên | * Thiết kế giao diện mockup * Viết báo cáo chương 1 + 2 + 3 * Code chức năng tìm kiếm địa điểm * Code chức năng lấy dữ liệu thời tiết hôm nay từ API của Openweather |
| 2 | Nguyễn Võ Thái Sơn | * Code chức năng lấy dữ liệu thời tiết 5 ngày tiếp theo từ API của Openweather * Code chức năng cập nhật lại dữ liệu mỗi 3 giờ * Viết báo cáo chương 4 + 5 * Viết kiểm thử |

**MỤC LỤC**

[CHƯƠNG I – TỔNG QUÁT 10](#_Toc22798250)

1. [1. Giới thiệu đề tài đồ án 10](#_Toc22798251)
2. [2. Màn hình mockup 10](#_Toc22798253)

[CHƯƠNG II – TÌM HIỂU VỀ ES6 13](#_Toc22798258)

1. [1. Tìm hiểu 13](#_Toc22798259)

[a. Variable 13](#_Toc22798260)

[b. Array 14](#_Toc22798266)

[c. Arrow function 15](#_Toc22798267)

1. [2. Ví dụ 17](#_Toc22798268)

[a. Ví dụ về xử lý mảng sử dụng các phương thức như: map, push, pop, length. 17](#_Toc22798269)

[b. Ví dụ về tính năm nhuận 17](#_Toc22798270)

[CHƯƠNG III – COMPONENT 18](#_Toc22798271)

1. [1. Tìm hiểu 18](#_Toc22798272)
2. [2. Ví dụ 21](#_Toc22798281)

[CHƯƠNG IV – XÂY DỰNG ỨNG DỤNG 22](#_Toc22798284)

[4.1 Feature/Component #1: Màn hình dữ liệu chi tiết thời tiết hôm nay 22](#_Toc22798285)

[4.2 Feature/Component #2: Màn hình thông tin thời tiết trong 5 ngày tiếp theo 25](#_Toc22798286)

[4.3 Code đồ án 26](#_Toc22798287)

1. [4.4 Cài đặt và triển khai 36](#_Toc22798290)

[CHƯƠNG V – KẾT LUẬT VÀ KIẾN NGHỊ 38](#_Toc22798304)

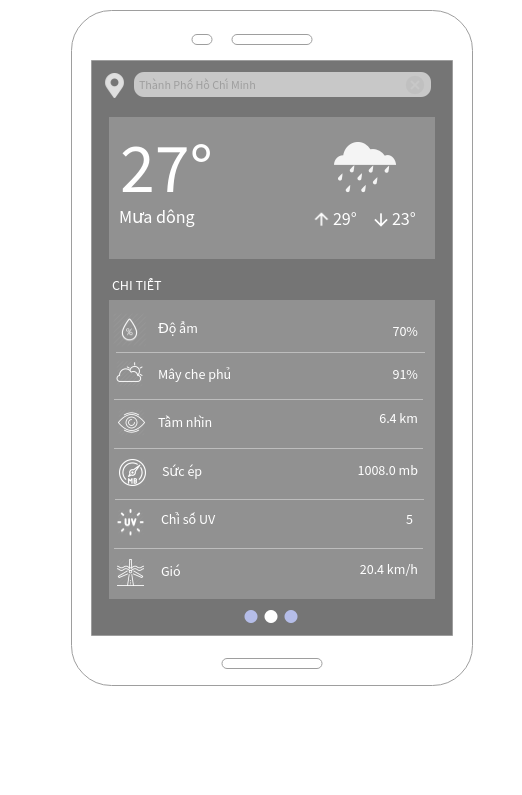
1. [1. Kết luận 38](#_Toc22798305)
2. [2 Kiến nghị 38](#_Toc22798309)
3. [3 Tài liệu tham khảo 38](#_Toc22798312)

CHƯƠNG I – TỔNG QUÁT

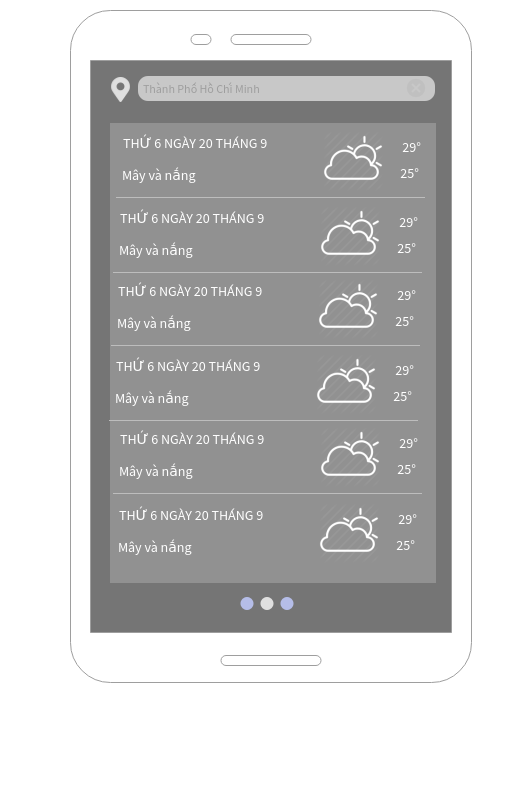
Giới thiệu đề tài đồ án

Vận dụng những kiến thức trong quá trình học, nhóm 6 lựa chọn đề tài app thời tiết, trong app có hai màn hình chính hiển thị thông tin thời tiết hiện tại và thông tin thời tiết cho năm ngày sau. Dữ liệu thông tin thời tiết được lấy từ API của trang openweathermap.org. Ngoài ra, để biết thông tin thời tiết ở những địa điểm khác, chúng ta có thể nhập tên thành phố trong khung tìm kiếm để hiển thị kết quả. Vì do sử dụng API dùng thử và miễn phí nên còn nhiều chức năng hạn chế mà nhóm em không khai thác hết được cũng như không tránh được những sai xót trong quá trình xây dựng đồ án. Mong thầy thông cảm và đóng góp ý kiến cho đồ án nhóm em.

Màn hình mockup



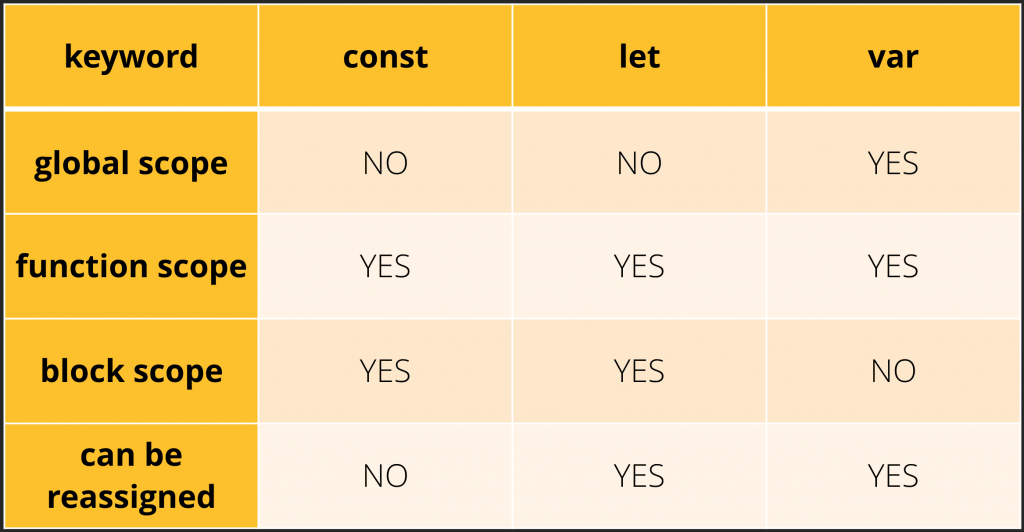
*Hình 1: màn hình mockup giao diện thời tiết hôm nay*



*Hình 2: màn hình mockup giao diện thời tiết 5 ngày tiếp theo*

CHƯƠNG II – TÌM HIỂU VỀ ES6

1. Tìm hiểu
   1. Variable



*Hình 3: bảng tóm tắt tính chất của 3 loại biến trong ES6*

* + var: với từ khóa *var* chúng ta có thể khai báo đa dạng các kiểu biến như number, string, boolean, etc. Biến var sẽ có scope là globally scoped trừ trường hợp được khai báo bên trong 1 function (khi đó biến var sẽ có scope là local). Đặc biệt, biến var còn có thêm tính chất hoisting: nghĩa là dù khai báo ở đâu thì biến đều sẽ được đem lên đầu scope trước khi code được thực hiện. Hạn chế của biến var là sự không rõ ràng, dễ nhầm lẫn cho lập trình viên vì thế với bản update ES6 ngôn ngữ Javascript đã được cộng đồng lập trình viên trên thế giới đánh giá cao vể nhiều mặt. Một trong những thay đổi đầu tiên và được hoan nghênh nhất trong phiên bản cập nhật ES6 là bổ sung thêm từ khóa let/const trong khai bao biến. Với let/const giờ đây các lập trình viên có thêm tùy chọn hợp lý hơn so với việc phải dùng var trong tất cả các trường hợp.
  + let:  là từ khóa định nghĩa 1 biến có phạm vi truy cập trong 1 block – khối code. Biến let có phạm vi trong dấu 1 cặp dấu {} bao quanh nó
  + const: là từ khóa định nghĩa 1 biến sẽ là hằng số. Biến const lưu trữ giá trị không thể thay đổi được trong suốt vòng đời của biến.
  1. Array
* Mảng là một tập hợp các phần tử lại và mỗi phần tử sẽ được đánh dấu một vị trí trong tập hợp đó. Trong javascript nếu mảng có 10 phần tử thì các phần tử sẽ được đánh dấu từ 0 -9. Array giúp chúng ta quản lý dữ liệu thống nhất, không rời rạc.
* Cấu trúc khai báo:
  + - *Khai báo theo kiểu new Array()*

var name\_array = new **Array**();

*// Hoặc*

var name\_array = new **Array**(1,2,3);

* + - *Khai báo với cặp dấu ngoặc vuông []*

var name\_array = [];

*// Hoặc*

var name\_array = [1,2,3];

* Truy cập phần tử trong mảng theo cú pháp:

var firstItem = name\_array[0];

var secondItem = name\_array[1];

* Duyệt mảng:

name\_array.**forEach**(function(item, index, array){

    console.**log**(item, index);

});

*//1 0*

*//2 1*

*//3 2*

*//hoặc nếu không xử lý mảng thì có thể duyệt*

*theo cách dưới*

console.**log**(`${name\_array}`); //1,2,3

* Thêm phần tử vào cuối:

var newLength = name\_array**.push**(4); // 1,2,3,4

* Thêm phần tử vào đầu:

 var newLength = name\_array**.unshift**(0); //0,1,2,3

* Xóa phần tử cuối:

 var newLength = name\_array**.pop**(); //1,2

* Xóa phần tử đầu:

 var newLength = name\_array**.shift**(); //2,3

* Array.map():

Phương thức map () tạo ra một mảng mới với kết quả gọi hàm được cung cấp trên mọi phần tử trong mảng gọi. Với cú pháp:

var new\_array = arr.**map**(function **callback**(currentValue[, index[, array]]) {

*// Return element for new\_array*

}

* 1. Arrow function

Arrow functon trong ES6 là kiểu cú pháp giúp đơn giản hoá việc việc định nghĩa hàm. Cách định nghĩa một hàm JavaScript theo cách thông thường trước đây:

**var** sumNumbers = **function** (a, b) {

**return** a + b;

}

Với tính năng arrow function trong ES6 chúng ta có thể viết lại đoạn code trên như sau:

**const** sumNumbers = (a, b) => {

**return** a + b;

}

* Từ khoá function được bỏ đi
* Thêm ký tự => (gần giống với mũi tên) đặt giữa dấu ngoặc kết thúc danh sách tham số ) và dấu ngoặc bắt đầu logic của hàm {

Ngoài ra chúng ta thay từ khoá var bằng const để quy định rằng giá trị của sumNumbers sẽ không thay đổi (tất nhiên bạn có thể dùng let nếu muốn).

Đối với các hàm được định nghĩa chỉ với một tham số thì bạn có thể bỏ qua cặp dấu ngoặc ():

**let** doubleNumber = a => {

**return** a \* 2;

}

Tuy nhiên nếu hàm không có đối số nào thì chúng ta vẫn phải sử dụng ():

**let** sayHello => () {

console.log("Hello everyone!");

}

Ví dụ

* 1. Ví dụ về xử lý mảng sử dụng các phương thức như: map, push, pop, length.

let  **xuLyMang** = num =>

{

*//nhân đôi các phần tử mảng*

  const myNewDoubleArray = num.**map**(x => x \* 2);

  console.**log**(`Mảng sau khi nhân đôi: ${myNewDoubleArray}`);

*//Mảng sau khi nhân đôi: 2,4,6,8,10,12,14,16,18,20*

  var lastItem = myNewDoubleArray.length - 1;*//20*

  if(myNewDoubleArray[lastItem] != 0)

  {

*//phan tử cuối khác không thì thêm số 0 vào cuối mảng*

    var newLength = myNewDoubleArray**.push**(0);

  }

  else

  {

*//nếu phần tử cuối bằng 0 thì xóa phần tử ở cuối mảng*

    var newLength = myNewDoubleArray**.pop**();

  }

  console.**log**(`Mảng sau khi xử lý: ${myNewDoubleArray}`);

*// Mảng sau khi xử lý: 2,4,6,8,10,12,14,16,18,20,0*

}

let arr = [1,2,3,4,5,6,7,8,9,10];

**xuLyMang**(arr);

* 1. Ví dụ về tính năm nhuận

*//Ham tinh nam nhuan am*

let **laNamNhuanAm** = n =>

{

    if(n % 19 == 0 || n % 19 == 3 || n % 19 == 6 ||

n % 19 == 9 || n % 19 == 11 || n % 19 == 14 ||  n % 19 == 17){

        return true;

    }

    return false;

}

*//In nam nhuan giua 2 moc thoi gian*

let **inNamNhuan** = (a,b)=>{

    console.**log**(`Cac nam nhuan am giua ${a} va ${b}  la: `);

    for(let i = a; i <= b; i++){

        if(**laNamNhuanAm**(i)){

            console.**log**(i);

        }

    }

}

*//Goi ham in nam nhuan*

*//inNamNhuan(1995,2031);*

if(**laNamNhuanAm**(2015)){

    console.**log**("la nam nhuan");

}else{

    console.**log**("khong la nam nhuan");

}

CHƯƠNG III – COMPONENT

1. Tìm hiểu

Các component nhóm em đã tìm hiểu và áp dụng:

* **View:**

Là thành phần cơ bản trong xây dựng giao diện, View là một khung chứa được tùy chỉnh bởi các thông số flexbox, style, xử lý cảm ứng và kiểm soát khả năng truy cập. Khi ứng dụng được chạy trên nền React-native trên thiết bị iOS hay Android, View sẽ được biên dịch tương ứng sao cho phù hợp với hệ điều hành trên thiết bị đang chạy.

View được thiết kế có thể lồng nhiều View khác hoặc nhiều Component khác.

* **Text:**

Một thành phần React để hiển thị văn bản.Văn bản hỗ trợ lồng, tạo kiểu và xử lý.

* TouchableHighlight:

Cung cấp các xử lý đa dạng hơn 1 Button bình thường như nó có thể lồng các Component khác như Image hoặc Text vào .Nền màn hình sẽ bị tối khi người dùng nhấn xuống nút.

* Image:

Component hiển thị hình ảnh và cung cấp 1 số xử lý tương tự như Text hoặc View

* TextInput:

Là UI nhận dữ liệu do người dùng nhập vào, textInput cũng cung cấp 1 số xử lý cơ bản. Trường hợp sử dụng đơn giản nhất là sự kiện onChangeText để đọc dữ liệu nhập của người dùng. Ngoài ra còn có các sự kiện khác, chẳng hạn như onSubmitEditing và onFocus.

* ImageBackground: giúp hiển thị nền giao diện là một hình ảnh từ local hoặc network.
* **Flatlist**: là một cách dễ dàng để tạo một list của data. Không chỉ hiệu quả mà còn có một API cực kỳ đơn giản để làm việc. Flatlist không còn phải định dạng dữ liệu – mà là truyền cho nó một array data và hiển thị lại giúp nâng cao hiệu xuất so với ScrollView thông thường. Flatlist hỗ trợ các tính năng tiện dụng nhất:
  + Đa nền tảng.
  + Chế độ ngang tùy chọn.
  + Cấu hình khả năng callback.
  + Hỗ trợ tiêu đề.
  + Hỗ trợ chân trang.
  + Hỗ trợ tách.
  + Kéo để làm mới.
  + Cuộn tải.
  + Hỗ trợ ScrollToIndex.

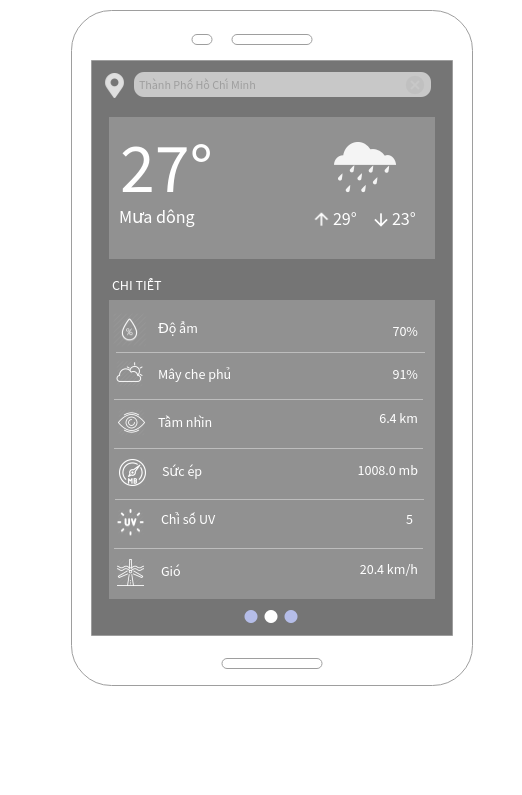
Ví dụ



*Hình 4 (Sơn) – 5 (Nguyên): sử dụng Component để thiết kế giao diện trong react-native*

CHƯƠNG IV – XÂY DỰNG ỨNG DỤNG

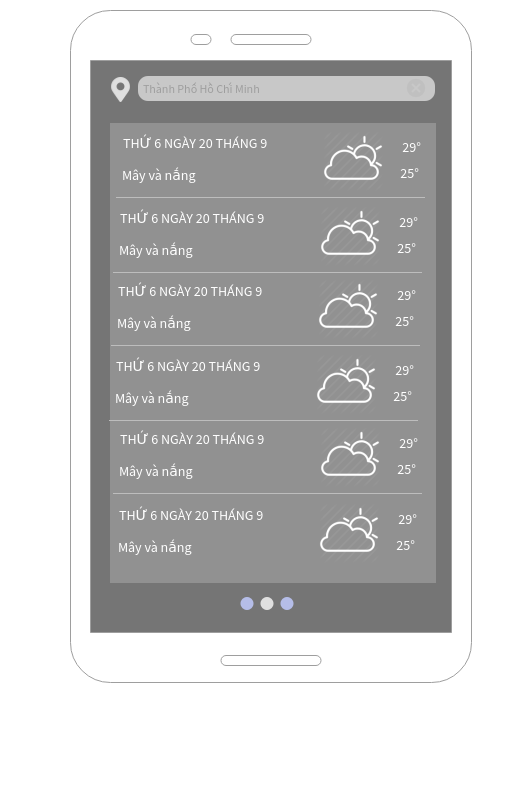
1. 1. Feature/Component #1: Màn hình dữ liệu chi tiết thời tiết hôm nay



*Hình 6: màn hình thiết kế hiển thị thông tin thời tiết trong ngày*

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Màn hình thông tin thời tiết chi tiết hôm nay** | **Màn hình thông tin thời tiết chi tiết hôm nay gồm có:**   * Khung nhập địa điểm cần biết thời tiết * Button icon dấu ‘x’ để xóa địa điểm hiện tại * Tab layout: Hôm nay và 5 ngày tiếp theo. Trong tab ‘Hôm nay’ gồm có:   + View đầu tiên:     - Nhiệt độ hiện tại     - Tình trạng thời tiết hiện tại   (biểu tượng)   * + - Tình trạng thời tiết hiện tại (bằng chữ)     - Nhiệt độ cao nhất trong ngày     - Nhiệt độ thấp nhất trong ngày   + List view bên dưới thể hiện các thông tin khác nhau mỗi dòng thông tin chứa biểu tượng, tiêu đề và chỉ số của thông tin đó, các dòng thông tin gồm có:     - Độ ẩm     - Mây che phủ     - Tầm nhìn     - Điểm sương     - Sức ép     - Chỉ số UV     - Sức gió | N/A | N/A |
| **Khung nhập địa điểm cần biết thời tiết** | Nhập địa điểm khác muốn biết thông tin thời tiết. Ví dụ như: Hà Nội hoặc Đống Đa, nhập địa điểm một Quận hoặc Thành Phố trong đất nước Việt Nam. | Nhập chữ vào khung tìm kiếm sau đó nhấn Enter trên bàn phím | Hiển thị thông tin thời tiết tại địa điểm vừa nhập |
| **Button icon dấu ‘x’ để xóa địa điểm hiện tại** | Xóa nội dung hiện tại trong khung nhập để nhập lại nội dung mới hoặc hủy tìm kiếm | Bấm vào Button | Xóa nội dung hiện tại trong khung nhập |
| **Tab layout thời tiết hôm nay** | Tab ‘Hôm nay’ là tab layout hiển thị mặc định khi truy cập vào ứng dụng. Ở tab này, hiển thị thông tin chi tiết thời tiết hôm nay gồm có view đầu tiên hiển thị thông tin cơ bản như nhiệt độ hiện tại, nhiệt độ cao/thấp nhất, tình trạng thời tiết hiện tại, thông tin này được cập nhật lại mỗi 3 giờ đồng hồ và độ chính xác ở mức tương đối, tất nhiên vẫn sẽ có sự chênh lệnh trong khoảng 1 giờ đồng hồ. Ví dụ, thông tin dự báo thời tiết sẽ mưa lúc 2 giờ chiều nhưng trên thực tế lại mưa vào khoảng 3 giờ chiều, chênh lệch 1 giờ so với thông tin dự báo.  Phía dưới view thông tin cơ bản sẽ có danh sách thông tin chi tiết khác và các chỉ số đi kèm. | Bấm vào tab ‘Hôm nay’ nếu đang ở tab ‘Hàng ngày’. | Hiển thị thông tin cơ bản và chi tiết về thời tiết trong ngày hôm nay |

* 1. Feature/Component #2: Màn hình thông tin thời tiết trong 5 ngày tiếp theo



*Hình 7: màn hình hiển thị thông tin thời tiết cơ bản trong 5 ngày tiếp theo*

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Màn hình thông tin thời tiết chi tiết 5 ngày tiếp theo** | **Màn hình thông tin thời tiết chi tiết 5 ngày tiếp theo gồm có:**   * Tab layout: Hôm nay và 5 ngày tiếp theo. Trong tab ‘5 ngày tiếp theo’ gồm có:   + List view bên dưới thể hiện các thông tin cơ bản của mỗi ngày theo dạng dòng, trong 1 dòng (tương ứng với mỗi ngày) gồm có những thông tin như:     - Thứ/ ngày/ tháng     - Tình trạng thời tiết chung của ngày đó     - Biểu tượng tình trạng thời tiết của ngày đó     - Nhiệt độ cao nhất và nhiệt độ thấp nhất của ngày đó | N/A | N/A |

* 1. Code đồ án
* **Weather.js (Nguyên)**

import React, { Component } from 'react';

import {

  StyleSheet,

  Text,

  View,

  Image,

  ActivityIndicator,

} from 'react-native';

*const* styles = StyleSheet.create({

  wrapper: {

  },

  slide1: {

    flex: 1,

    backgroundColor: '#9DD6EB',

  },

  slide2: {

    flex: 1,

    backgroundColor: '#9DD6EB',

  },

  input:

  {

    width: '90%',

    marginLeft: 15,

    color: 'white',

    backgroundColor: 'rgba(52, 52, 52, 0.4)',

    alignSelf: 'baseline',

    borderRadius: 30,

    paddingLeft: 20,

    marginVertical: 15,

  },

  info\_basic\_today:

  {

    paddingLeft: 20,

    marginLeft: 15,

    paddingRight: 20,

    width: '90%',

    alignSelf: "baseline",

    backgroundColor: 'rgba(52, 52, 52, 0.4)',

    marginVertical: 10

  },

  temp\_now:

  {

    color: 'white',

    fontSize: 60,

    paddingRight: 50

  },

  icon\_temp\_now:

  {

    width: 100,

    height: 90,

    flexDirection: "row-reverse",

  },

  rowbasic\_1:

  {

    marginVertical: 10,

    flexDirection: "row"

  },

  rowbasic\_2:

  {

    flexDirection: "row",

    marginVertical: 15

  },

  temp\_low\_high:

  {

    color: 'white',

    flexDirection: "row-reverse",

  },

  info\_temp:

  {

    color: 'white'

  },

  rowbasic\_2\_1:

  {

    flexDirection: "row",

    marginStart: 100,

    marginEnd: 20

  },

  info\_detail\_today:

  {

    paddingHorizontal: 20,

    marginLeft: 15,

    paddingRight: 20,

    width: '90%',

    backgroundColor: 'rgba(52, 52, 52, 0.4)',

    marginVertical: 10,

  },

  icon\_weather:

  {

    width: 27,

    height: 27,

  },

  detail\_title:

  {

    color: 'white',

    height: 25,

    width: 100,

    paddingLeft: 15

  },

  detail\_number:

  {

    color: 'white',

    height: 25,

    marginStart: '35%',

    marginEnd: 10

  },

  detail\_row:

  {

    flexDirection: "row",

    marginTop: 23,

    borderBottomWidth: 1,

    borderBottomColor: '#cecdcd69',

    paddingBottom: 10

  }

})

export default *class* Weather *extends* Component {

  constructor(props){

    super(props);

    this.state ={

      isLoading: true,

      city: this.props.navigation.getParam('city'),

    }

  }

  componentDidMount(){

*console*.log(this.state.city);

    return fetch(`http://api.openweathermap.org/data/2.5/forecast?q=${this.state.city}&lang=vi&units=metric&APPID=b5177eb82d0e5d0cbdbbf5a5d2cd19b1`)

      .then((response) *=>* response.json())

      .then((responseJson) *=>* {

        this.setState({

          isLoading: false,

          dataSource: responseJson.list,

        }, *function*(){

        });

      })

      .catch((error) *=>*{

*console*.error(error);

      });

  }

  render(){

    if(this.state.isLoading){

      return(

        <View style={{flex: 1, padding: 20}}>

          <ActivityIndicator/>

        </View>

      )

    }

*const* weather = this.state.dataSource;

    for(*let* i = 0; i < weather.length; i++){

      if(this.props.navigation.getParam('key') === weather[i].dt\_txt){

        return (

            <View style={styles.slide1}>

            <View style={styles.info\_basic\_today}>

              <View style={styles.rowbasic\_1}>

                <Text style={styles.temp\_now}>{parseInt(weather[i].main.temp, 10)}°C</Text>

                <Image source={{uri: "http://openweathermap.org/img/wn/"+ weather[i].weather[0].icon + "@2x.png"}} style={styles.icon\_temp\_now}></Image>

              </View>

              <View style={styles.rowbasic\_1}>

                <Text style={styles.detail\_title}>Ngày giờ</Text>

                <Text style={styles.detail\_number}>{weather[i].dt\_txt}</Text>

              </View>

              <View style={styles.rowbasic\_2}>

                <Text style={styles.info\_temp}>{weather[i].weather[0].description}</Text>

                <View style={styles.rowbasic\_2\_1}>

                  <Image source={require('./img/up-arrow.png')} style={{width: 10, height: 14, marginTop: 3}}></Image>

                  <Text style={styles.temp\_low\_high}>  {weather[i].main.temp\_max}°C     </Text>

                  <Image source={require('./img/down-arrow.png')} style={{width: 10, height: 14, marginTop: 3}}></Image>

                  <Text style={styles.temp\_low\_high}>  {weather[i].main.temp\_min}°C</Text>

                </View>

              </View>

            </View>

            <View style={styles.info\_detail\_today}>

              <View style={styles.detail\_row}>

                <Image source={require('./img/humidity.png')} style={styles.icon\_weather}></Image>

                <Text style={styles.detail\_title}>Độ ẩm</Text>

                <Text style={styles.detail\_number}>{weather[i].main.humidity}%</Text>

              </View>

              <View style={styles.detail\_row}>

                <Image source={require('./img/cloudcover.png')} style={styles.icon\_weather}></Image>

                <Text style={styles.detail\_title}>Mây che phủ</Text>

                <Text style={styles.detail\_number}>{weather[i].clouds.all}%</Text>

              </View>

              <View style={styles.detail\_row}>

              <Image source={require('./img/view.png')} style={styles.icon\_weather}></Image>

              <Text style={styles.detail\_title}>Tầm nhìn</Text>

              <Text style={styles.detail\_number}>6.4km</Text>

              </View>

              <View style={styles.detail\_row}>

                <Image source={require('./img/meter.png')} style={styles.icon\_weather}></Image>

                <Text style={styles.detail\_title}>Sức ép</Text>

                <Text style={styles.detail\_number}>{weather[i].main.pressure}hPa</Text>

              </View>

              <View style={styles.detail\_row}>

                <Image source={require('./img/windmill.png')} style={styles.icon\_weather}></Image>

                <Text style={styles.detail\_title}>Gió</Text>

                <Text style={styles.detail\_number}>{weather[i].wind.speed}m/s</Text>

              </View>

            </View>

            </View>

        )

      }

    }

    return null;

  }

}

* WeatherForecast.js (Sơn)

//import thu vien

import React, {Component} from "react";

import{

  View, Text, FlatList, StyleSheet, Image, ActivityIndicator, TextInput, TouchableOpacity

} from "react-native";

//Khai bao class

export default *class* Forecast *extends* Component{

  constructor(props){

    super(props);

    this.state ={

      isLoading: true,

      city: 'Thành phố Hồ Chí Minh',

    }

  }

  getForecast(){

    return fetch('http://api.openweathermap.org/data/2.5/forecast?q='+ this.state.city +'&lang=vi&units=metric&APPID=b5177eb82d0e5d0cbdbbf5a5d2cd19b1')

      .then((response) *=>* response.json())

      .then((responseJson) *=>* {

        this.setState({

          isLoading: false,

          dataSource: responseJson,

        }, *function*(){

        });

      })

      .catch((error) *=>*{

*console*.error(error);

      });

  }

  componentDidMount(){

    this.getForecast();

  }

  render(){

    if(this.state.isLoading){

      return(

        <View style={{flex: 1, padding: 20}}>

          <ActivityIndicator/>

        </View>

      )

    }

*const* {navigate} = this.props.navigation;

    return(

      <View>

        <View style={{flexDirection: "row"}}>

          <TextInput

          placeholder = "Nhập thành phố"

          underlineColorAndroid = "transparent"

          style={{flex: 2}}

          onChangeText = {(TextInputText) *=>* this.setState({ city: TextInputText })} />

          <TouchableOpacity style={{flex: 1/3}} onPress={()*=>*this.getForecast()}>

            <View style={{width:100, height: 50, backgroundColor: "grey", justifyContent: "center"}}>

              <Text>Tìm kiếm</Text>

            </View>

          </TouchableOpacity>

        </View>

        <Text style={styles.thanhpho}>Thành phố(Quận): {this.state.dataSource.city.name}</Text>

        <Text style={styles.thanhpho}>Quốc gia: {this.state.dataSource.city.country}</Text>

        <FlatList data={this.state.dataSource.list}

          renderItem={({item})*=>*{

            return(

              <View>

                <TouchableOpacity onPress={() *=>* navigate('Weather', {key: item.dt\_txt, city: this.state.city})}>

                <View style={styles.flatList}>

                  <View style={{

                    flex: 2,

                    flexDirection: "column",

                  }}>

                    <Text style={styles.flatListItem}>{item.dt\_txt}</Text>

                    <Text style={styles.flatListItem}>{item.weather[0].description}</Text>

                  </View>

                  <View style={{flex: 1/2,

                    flexDirection: "column",}}>

                    <Image

                      style={styles.image}

                      source={{uri: "http://openweathermap.org/img/wn/"+ item.weather[0].icon + "@2x.png"}}

                    />

                  </View>

                  <View style={{

                    flex: 2/3,

                    flexDirection: "column",

                  }}>

                  <Text style={styles.flatListItem}>{`${item.main.temp\_max}°C`}</Text>

                    <Text style={styles.flatListItem}>{`${item.main.temp\_min}°C`}</Text>

                  </View>

                </View>

                </TouchableOpacity>

                <View style={{height: 1, backgroundColor: 'white'}}>

                </View>

              </View>

            )

          }}

          keyExtractor={item *=>* item.dt\_txt}

        >

        </FlatList>

      </View>

    );

  }

}

*var* styles = StyleSheet.create({

  flatListItem: {

    color: 'white',

    padding: 10,

    fontSize: 16,

  },

  flatList:{

    flex: 1,

    flexDirection: "row",

    backgroundColor: 'mediumseagreen'

  },

  image:{

    width: 60,

    height: 60,

    marginTop: 10

  },

  thanhpho:{

    padding: 10,

    fontSize: 16,

    backgroundColor: "tomato",

    color: "white",

  },

});

* index.js

import {createAppContainer} from 'react-navigation';

import {createStackNavigator} from 'react-navigation-stack';

import Forecast from './Forecast';

import Weather from './Weather';

*const* MainNavigator = createStackNavigator({

  Forcast: {

    screen: Forecast,

    navigationOptions: {

      header: null,

    },

  },

  Weather: {screen: Weather},

});

*const* App = createAppContainer(MainNavigator);

export default App;

* 1. Cài đặt và triển khai

Bước 1: đăng ký tài khoản trên trang openweathermap.org để lấy API

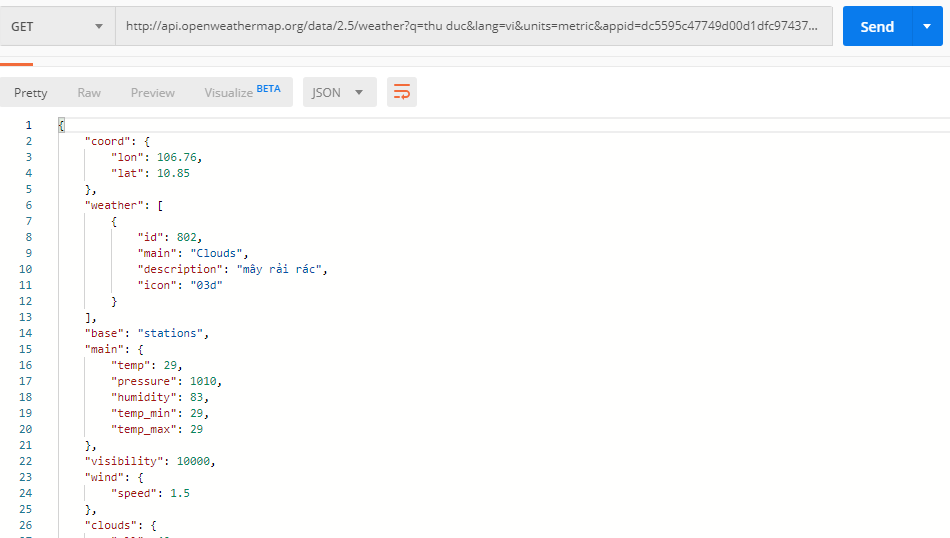
Bước 2: sau khi đăng nhập tài khoản mới đăng ký, vào trang [home.openweathermap.org/api\_keys](http://home.openweathermap.org/api_keys) để lấy API key.



Bước 3: tìm hiểu cách lấy API thời tiết hiện tại ở trang [openweathermap.org/current](https://openweathermap.org/current) và viết code lấy dữ liệu thời tiết hiện tại.

Ví dụ request API thông tin thời tiết hiện tại thông qua tên thành phố:

[http://api.openweathermap.org/data/2.5/weather?q=thu duc&lang=vi&units=metric&APPID=b5177eb82d0e5d0cbdbbf5a5d2cd19b1](http://api.openweathermap.org/data/2.5/weather?q=thu%20duc&lang=vi&units=metric&APPID=b5177eb82d0e5d0cbdbbf5a5d2cd19b1)

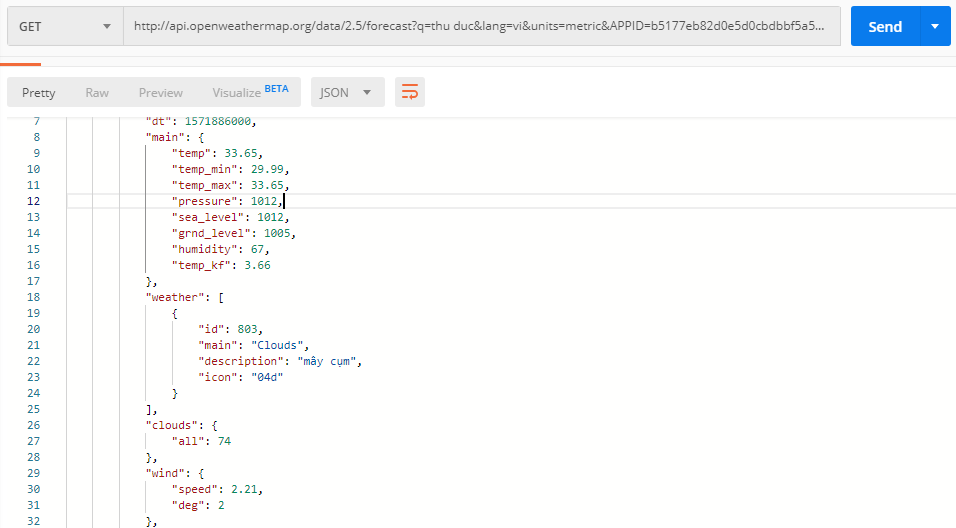


*Hình 8: kết quả trả về dạng json sau khi request*

Bước 4: tìm hiểu cách lấy API thời tiết 5 ngày tiếp theo ở trang [openweathermap.org/forecast5](https://openweathermap.org/forecast5) và viết code lấy dữ liệu thời tiết trong 5 ngày tiếp theo.

Ví dụ request API thông tin thời tiết qua tên thành phố:

[http://api.openweathermap.org/data/2.5/forecast?q=thu duc&lang=vi&units=metric&APPID=b5177eb82d0e5d0cbdbbf5a5d2cd19b1](http://api.openweathermap.org/data/2.5/forecast?q=thu%20duc&lang=vi&units=metric&APPID=b5177eb82d0e5d0cbdbbf5a5d2cd19b1)



*Hình 9: kết quả trả về dạng json sau khi request*

Bước 5: sau khi request API hoạt động được bước tiếp theo là tiến hành code lấy dữ liệu từ API như ở mục 4.3

CHƯƠNG V – KẾT LUẬT VÀ KIẾN NGHỊ

1. Kết luận

Sau khi kết thúc môn học lập trình di động 3 – React Native. Em và bạn cùng nhóm đã học được nội dung cơ bản của React Native. Là 1 ngôn ngữ hỗ trợ đa nền tảng nên khi bắt đầu làm quen chúng em đã có một vài khó khăn trong việc cài đặt và làm quen chẳng hạn như phải tìm hiểu thêm về NPM, node.js, ES6 (JavaScript). Được sự hỗ trợ và giảng dạy của thầy giúp chúng em định hướng được cách học tập và xây dựng ứng dụng cơ bản trong React Native.

Để chứng minh được những kiến thức mình tiếp thu, nhóm 6 quyết định xây dựng ứng dụng hiển thị thông tin thời tiết chi tiết trong ngày hiện tại và 5 ngày tiếp theo, dữ liệu được cập nhật mỗi 3 giờ và dữ liệu được lấy từ API công khai của OpenWeather. Do là API dùng thử và có trả phí nên chúng em chỉ được sử dụng những chức năng cơ bản và hạn chế. Tuy ứng dụng đơn giản và kiến thức còn hạn chế nên chắc sẽ có sai sót đôi chút, mong thầy thông cảm cho nhóm em.

Một lần nữa nhóm em xin cảm ơn thầy Trương Bá Thái, giảng viên giảng dạy bộ môn đã hỗ trợ nhóm em trong suốt quá trình học tập.

1. Kiến nghị

Trong quá trình học và làm đồ án, nhóm em có kiến nghị như sau:

* React Native là ngôn ngữ hỗ trợ đa nền tảng nhưng chủ yếu chúng em chỉ được thực hành trên thiết bị Android là chính và chưa được thử nghiệm trên thiết bị iOS, đó cũng là một hạn chế của nhóm em khi không có thiết bị thật hoặc máy tính đủ mạnh để chạy máy ảo trải nghiệm. Vì thế, em mong nhà trường tạo điều kiện và để tâm hơn phần phát triển ứng dụng phía iOS trong chương trình dạy môn lập trình di động 3.

1. Tài liệu tham khảo

* <https://cameoplus.com/tai-sao-nen-su-dung-let-const-thay-cho-var-trong-javascript/>
* <https://developer.mozilla.org/vi/docs/Web/JavaScript/Reference/Global_Objects/Array>
* <https://www.codehub.vn/ES6-Co-Ban/ES6-Arrow-Function>
* <https://techblog.vn/cach-su-dung-flatlist-component-trong-react-native-phan-1>