**TRƯỜNG ĐẠI HỌC LÂM NGHIỆP**

KHOA KINH TẾ VÀ QUẢN TRỊ KINH DOANH

**===o0o===**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN: LẬP TRÌNH MOBILE**

**ĐỀ TÀI:**

**PHÁT TRIỂN PHẦN MỀM ĐỌC TRUYỆN TRANH**



**Giảng viên hướng dẫn: Mai Hà An**

**Sinh viên thực hiện:**

**Họ và Tên: Nguyễn Khả Đăng**

**Mã sinh viên: 2041070501**

**Lớp: K65 - HTTT**

**Hà Nội, 2022**

MỤC LỤC

[CHƯƠNG I. TỔNG QUAN ĐỀ TÀI NGHIÊN CỨU 4](#_Toc153554433)

[1.1 Lý do, mục đích chọn đề tài 4](#_Toc153554434)

[1.2 Mục tiêu và phạm vi nghiên cứu 4](#_Toc153554435)

[1.2.1 Mục tiêu 4](#_Toc153554436)

[1.2.2 Phạm vi nghiên cứu 4](#_Toc153554437)

[CHƯƠNG II. CƠ SỞ LÝ THUYẾT 5](#_Toc153554438)

[2.1. Lập trình mobile 5](#_Toc153554439)

[2.1.1 Định Nghĩa 5](#_Toc153554440)

[2.1.2 Công nghệ lập trình mobile 5](#_Toc153554441)

[2.1.3 Các mô hình lập trình mobile 5](#_Toc153554442)

[2.2 API 6](#_Toc153554443)

[2.2.1 Định Nghĩa 6](#_Toc153554444)

[2.2.2 Một số loại API phổ biến 7](#_Toc153554445)

[CHƯƠNG III. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 8](#_Toc153554446)

[3.1 Thiết kế mô hình UML 8](#_Toc153554447)

[3.1.1 Danh sách các đối tượng sử dụng (Actor) 8](#_Toc153554448)

[3.1.2 Mô tả các use case 8](#_Toc153554449)

[3.1.3 Cơ sở dữ liệu 8](#_Toc153554450)

[3.1.4. Biểu đồ phân rã các UC 9](#_Toc153554451)

[3.1.5. Biểu đồ hoạt động 10](#_Toc153554452)

[CHƯƠNG IV. LẬP TRÌNH VÀ KẾT QUẢ 12](#_Toc153554453)

[4.1. Các hàm chức năng 12](#_Toc153554454)

[4.1.1 Các hàm tương tác với cơ sở dữ liệu 12](#_Toc153554455)

[4.1.2 Hàm tạo request tới server 13](#_Toc153554456)

[4.1.3 Hiển thị danh sách truyện 14](#_Toc153554457)

[4.1.5 Hiển thị chi tiết truyện 15](#_Toc153554458)

[4.1.5 Hàm tạo handle sự kiện lướt ảnh và hiển thị ảnh 18](#_Toc153554459)

[4.2. Kết quả lập trình 19](#_Toc153554460)

[4.2.1 Màn hình chính 19](#_Toc153554461)

[CHƯƠNG V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 22](#_Toc153554462)

[5.1. Kết luận 22](#_Toc153554463)

[5.1.1. Kết quả đạt được 22](#_Toc153554464)

[5.1.2 Hạn Chế 22](#_Toc153554465)

[5.2. Hướng phát triển 22](#_Toc153554466)

# CHƯƠNG I. TỔNG QUAN ĐỀ TÀI NGHIÊN CỨU

## 1.1 Lý do, mục đích chọn đề tài

Ngày nay, cách mạng công nghiệp 4.0 đang diễn ra mạnh mẽ ở trên thế giới, đặc biệt là ở Việt Nam, quốc gia đang chứng kiến sự cải tiến và ứng dụng công nghệ vào nơi làm việc ở tốc độ nhanh hơn bao giờ hết và rất đa dạng trong các ngành nghề khác nhau vì vậy về mặt giải trí cũng được ứng dụng . Khi công nghệ thông tin đã và đang đóng vai trò quan trọng trong xã hội. Nó giúp con người giảm căng thẳng sau mỗi giờ làm việc dài. Do đó việc ứng dụng công nghệ thông tin vào việc giải trí bớt căng thẳng sau những giờ làm việc dài cũng là điều tất yếu.

## 1.2 Mục tiêu và phạm vi nghiên cứu

### 1.2.1 Mục tiêu

Mục tiêu chủ yếu của đề tài là xây dựng một hệ thống đọc truyện nhằm phục vụ việc giải trí giảm căng thẳng và cũng như sở thích. Đảm bảo các tính năng gần gũi với mọi người và dễ dàng sử dụng.

### 1.2.2 Phạm vi nghiên cứu

Mục tiêu chủ yếu của đề tài là xây dựng một website đọc truyện nhằm phục vụ việc giải trí giảm căng thẳng và cũng như sở thích. Đảm bảo các tính năng gần gũi với mọi người và dễ dàng sử dụng.

# CHƯƠNG II. CƠ SỞ LÝ THUYẾT

## Lập trình mobile

### 2.1.1 Định Nghĩa

Lập trình mobile là việc thiết kế và triển khai các ứng dụng phần mềm chạy trên thiết bị di động.

Và để ứng dụng có thể chạy được thì thiết bị di động phải cung cấp một nền tảng hỗ trợ phát triển, được gọi tắt là hệ điều hành. Hệ điều hành đóng vai trò trung gian, tạo điều kiện thuận lợi cho sự tương tác giữa phần cứng máy tính và các chương trình chạy trên nó.

Mặc dù trong lịch sử đã từng tồn tại rất nhiều hệ điều hành khả thi, nhưng ngày nay, phần lớn thị phần được chia cho 2 ứng cử viên hàng đầu: Android (chiếm 80%) và iOS (chiếm 18%). 2% còn lại bao gồm các hệ điều hành Windows Phone và Blackberry, nhưng hầu như tất cả các nhà phát triển mobile đều chọn tập trung nỗ lực của họ vào Android, iOS hoặc kết hợp cả hai.

### 2.1.2 Công nghệ lập trình mobile

Hiện nay, có nhiều công nghệ lập trình khác nhau để xây dựng một ứng dụng mobile. Chúng đều tồn tại các ưu - nhược điểm khác nhau. Nhưng chung quy vẫn là hỗ trợ thư viện và API với nhiều cấp độ và mức phụ thuộc cụ thể.

Một số công nghệ lập trình mạng phổ biến và được sử dụng nhiều nhất hiện nay là Java, Kotlin, React Native, Appcelerator, Xamarin ...

### 2.1.3 Các mô hình lập trình mobile

Lập trình di động là quá trình phát triển ứng dụng cho các thiết bị di động như điện thoại thông minh và máy tính bảng. Có nhiều mô hình lập trình di động khác nhau, tùy thuộc vào nền tảng (iOS, Android, hoặc cả hai) và ngôn ngữ lập trình sử dụng. Dưới đây là một số mô hình lập trình di động phổ biến:

1. **Native App Development:**

* iOS: Sử dụng ngôn ngữ Objective-C hoặc Swift và công cụ phát triển như Xcode.
* Android: Sử dụng Java hoặc Kotlin và công cụ phát triển như Android Studio.

1. **Cross-Platform Development:**

* React Native: Sử dụng JavaScript và React để xây dựng ứng dụng chạy trên cả iOS và Android.
* Flutter: Sử dụng Dart để tạo giao diện người dùng đẹp mắt và ứng dụng chạy trên cả hai nền tảng.
* Xamarin: Sử dụng C# và .NET để phát triển ứng dụng chạy trên cả hai nền tảng.

1. **Web-Based Mobile Apps:**

* Sử dụng HTML, CSS, và JavaScript để phát triển ứng dụng web được tối ưu hóa cho trải nghiệm di động. Các công nghệ như Progressive Web Apps (PWAs) cung cấp trải nghiệm giống như ứng dụng native.

1. **Hybrid App Development:**

* Sử dụng các framework như Apache Cordova hoặc Ionic để đóng gói ứng dụng web trong một đóng gói chương trình di động, có thể triển khai trên nhiều nền tảng.

1. **Augmented Reality (AR) and Virtual Reality (VR):**

* Sử dụng ARKit (cho iOS) hoặc ARCore (cho Android) cho phát triển ứng dụng thực tế ảo và ảo thực tế.

1. **Game Development:**

* Sử dụng các framework như Unity hoặc Unreal Engine để phát triển trò chơi di động, có thể chạy trên cả iOS và Android.

1. **Serverless Mobile Backend:**

* Sử dụng các dịch vụ điện toán đám mây (cloud) như Firebase, AWS Mobile Hub để quản lý backend cho ứng dụng di động mà không cần triển khai và duy trì máy chủ riêng.

Mỗi mô hình có những ưu điểm và nhược điểm riêng, và lựa chọn giữa chúng thường phụ thuộc vào yêu cầu cụ thể của dự án và kiến thức của nhà phát triển.

## 2.2 API

### 2.2.1 Định Nghĩa

API (Application Programming Interface) là một tập hợp các quy tắc, giao thức và công cụ được sử dụng để phát triển các ứng dụng phần mềm và cho phép các ứng dụng khác tương tác với ứng dụng của bạn. API thường được sử dụng để chia sẻ dữ liệu và chức năng giữa các ứng dụng khác nhau và cung cấp tính năng mở rộng cho ứng dụng của bạn.

### 2.2.2 Một số loại API phổ biến

* API Web: API Web là một loại API sử dụng giao thức HTTP để cho phép các ứng dụng tương tác với các dịch vụ web như Google Maps, Facebook, Twitter, và PayPal.
* API RESTful: REST (Representational State Transfer) là một kiến trúc được sử dụng để phát triển các dịch vụ web. API RESTful là một loại API sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thực hiện các yêu cầu và truy xuất dữ liệu.
* API SOAP: SOAP (Simple Object Access Protocol) là một giao thức được sử dụng để truyền thông tin giữa các ứng dụng. API SOAP sử dụng XML để định dạng dữ liệu và cung cấp các tính năng bảo mật và kiểm soát phiên.
* API nhúng: API nhúng được sử dụng để kết nối các thiết bị nhúng như cảm biến, thiết bị điều khiển và robot với các ứng dụng.

# CHƯƠNG III. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 3.1 Thiết kế mô hình UML

### 3.1.1 Danh sách các đối tượng sử dụng (Actor)

Hệ thống sau khi khảo sát xác định được 2 đối tượng tác nhân chính gồm : Quản trị viên (Admin) , người dùng (User). Cụ thể các chức năng của mỗi tác nhân như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tác nhân** | **Mô tả** | **Chức năng** |
| 1 | Người dùng (User) | Là người vào xem truyện , đọc truyện | Xem truyện, xem tập, tìm kiếm, lọc theo thể loại, lưu và lịch sử đọc, zoom ảnh khi đọc truyện |
| 2 | Quản trị viên (admin) | Là người quản lý dữ liệu hệ thống | Thêm, sửa, xóa thông tin tác giả, thể loại, truyện, tập |

Bảng 1: Danh sách các đối tượng sử dụng (Actor)

### 3.1.2 Mô tả các use case

|  |  |  |
| --- | --- | --- |
| **STT** | **Use case** | **Mô tả** |
| 1 | Xem truyện | Xem thông tin truyện |
| 2 | Xem tập | Xem tập truyện |
| 3 | Tìm kiếm | Tìm kiếm truyện theo tên |
| 4 | Lọc | Lọc truyện theo thể loại |
| 5 | Lưu, xóa lịch sử đọc | Lưu lịch sử đọc mỗi khi xem tập và cho phép người dùng xóa lịch sử đọc |
| 6 | Zoom ảnh | Cho phép thu phóng ảnh khi đọc truyện |
| 7 | Cập nhật dữ liệu theo thời gian thực | Ứng dụng mobile sẽ được cập nhật dữ liệu theo web |

Bảng 2: Danh sách các use case chính

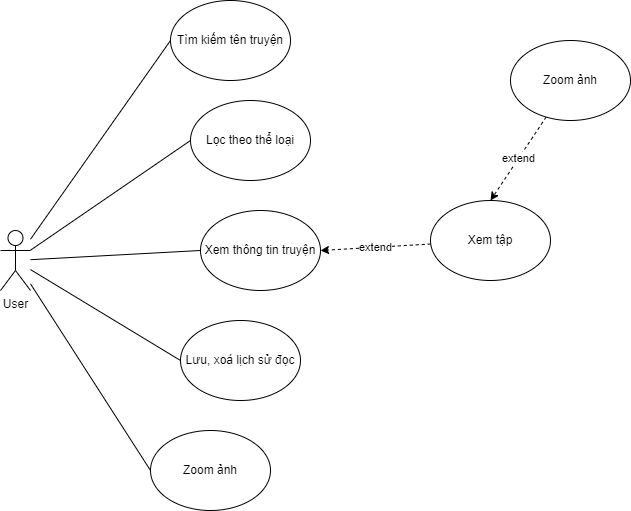
### 3.1.3 Cơ sở dữ liệu

Bảng LichSuDoc

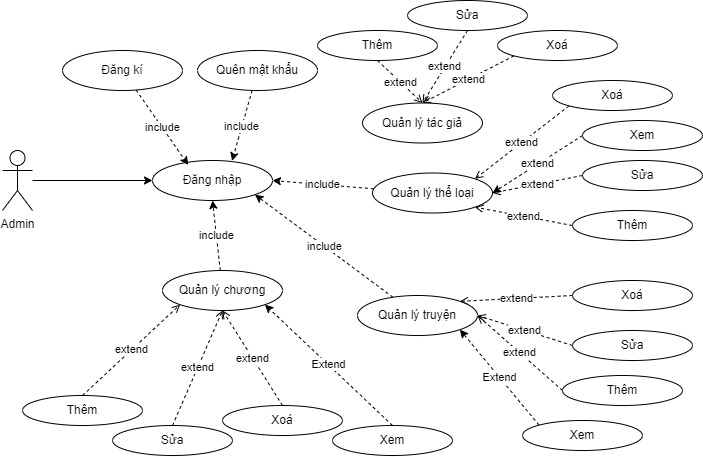
|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Mô tả |
| id | int | PRIMARY KEY AUTOINCREMENT |
| tentruyen | Text | Tên truyện |
| slug | Text | Tên viết tắt của truyện |
| tentap | Text | Tên tập |
| id\_tap | Int | Id tập truyện |
| id\_truyen | Int | Id truyện |
| path | Text | Đường dẫn ảnh đại diện truyện |
| indexPath | Int | Số trang cuối đang đọc của tập |

### 3.1.4. Biểu đồ phân rã các UC

* **Biểu đồ usecase User (lient)**

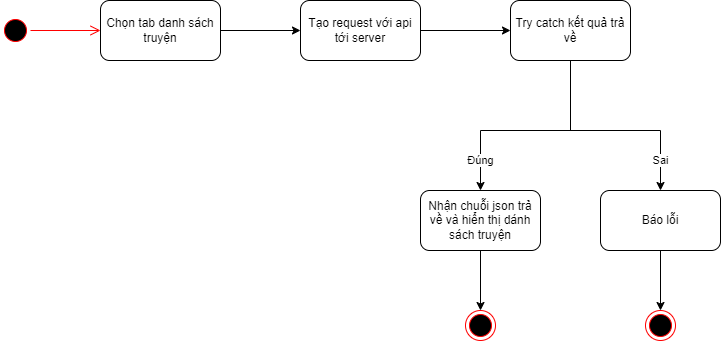
****

* **Biểu đổ usecase Admin (server)**

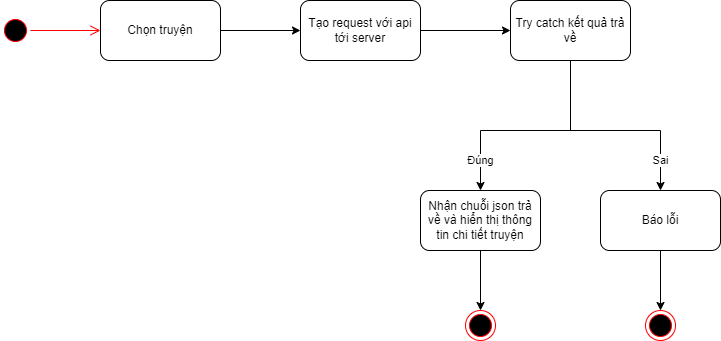
****

### 3.1.5. Biểu đồ hoạt động

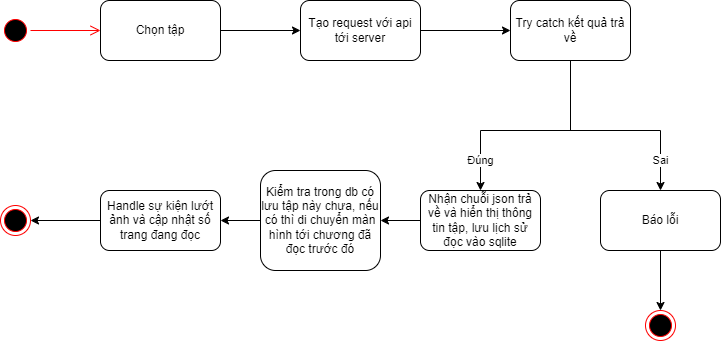
* **Biểu đồ hoạt động xem danh sách truyện**

****

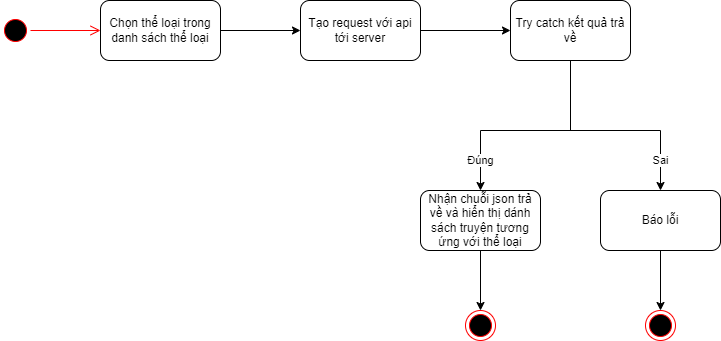
* **Biểu đồ hoạt động xem thông tin chi tiết truyện**

****

* **Biểu đồ hoạt động xem tập (đọc truyện)**



* **Biểu đồ lọc thể loại**

****

# CHƯƠNG IV. LẬP TRÌNH VÀ KẾT QUẢ

## 4.1. Các hàm chức năng

### 4.1.1 Các hàm tương tác với cơ sở dữ liệu

const createTable = () => {

  db.transaction((tx) => {

    tx.executeSql(

      "CREATE TABLE IF NOT EXISTS lichsu (id INTEGER PRIMARY KEY AUTOINCREMENT, tentruyen TEXT, slug TEXT, tentap TEXT, id\_tap INTEGER, id\_truyen INTEGER, path TEXT, indexPath INTEGER);"

    );

  });

  console.log("created table");

};

const insertLichsu = (tentruyen, slug, tentap, id\_tap, id\_truyen, path) => {

  db.transaction((tx) => {

    tx.executeSql(

      "SELECT \* FROM lichsu WHERE id\_truyen = ?",

      [id\_truyen],

      (tx, results) => {

        if (results.rows.length > 0) {

          if (

            results.rows.item(0).id\_truyen == id\_truyen &&

            results.rows.item(0).id\_tap != id\_tap

          ) {

            tx.executeSql(

              "UPDATE lichsu SET tentruyen=?, slug=?, tentap=?, id\_tap=?, path=?, indexPath=? WHERE id\_truyen=?",

              [tentruyen, slug, tentap, id\_tap, path, 0, id\_truyen]

            );

            console.log("update " + id\_truyen);

          }

        } else {

          tx.executeSql(

            "INSERT INTO lichsu (tentruyen, slug, tentap, id\_tap, id\_truyen, path, indexPath) VALUES (?, ?, ?, ?, ?, ?, ?);",

            [tentruyen, slug, tentap, id\_tap, id\_truyen, path, 0]

          );

          console.log("insert new");

        }

      }

    );

  });

};

const showLichsu = () => {

  db.transaction((tx) => {

    tx.executeSql("SELECT \* FROM lichsu", [], (tx, results) => {

      console.log(results.rows.length);

      if (results.rows.length > 0) {

        for (let index = 0; index < results.rows.length; index++) {

          const row = results.rows.item(index);

          console.log("------------" + index + "----------");

          console.log("Tentruyen: " + row.tentruyen);

          console.log("Slug: " + row.slug);

          console.log("Tentap: " + row.tentap);

          console.log("Id\_tap: " + row.id\_tap);

          console.log("Id\_truyen: " + row.id\_truyen);

          console.log("Path: " + row.path);

          console.log("indexPath: " + row.indexPath);

        }

      }

    });

  });

};

db.transaction((tx) => {

  tx.executeSql(

    "SELECT indexPath FROM lichsu where id\_truyen=?",

    [result.truyen.id],

    (tx, dt) => {

      if (dt.rows.length > 0) {

        setIndexPath(dt.rows.item(0).indexPath);

        const wait = new Promise((resolve) => setTimeout(resolve, 700));

        wait

          .then(() => {

            flatListRef.current.scrollToIndex({

              animated: false,

              index: dt.rows.item(0).indexPath,

              viewPosition: 0.5,

            });

          })

          .catch((error) => {

            console.error("Lỗi khi cuộn tới index:", error);

          });

      }

    }

  );

});

### 4.1.2 Hàm tạo request tới server

const fetchData = async url => {

  const response = await fetch('http://127.0.0.1:8000/api/' + url);

  return response.json();

};

export default fetchData;

const fetchDataFromApi = async () => {

  try {

    const result = await fetchData('home');

    setData(result);

  } catch (error) {

    console.error('Lỗi call api, đổi port đê:', error);

  }

};

### 4.1.3 Hiển thị danh sách truyện

<SwiperFlatList

  style={{

    zIndex: 10,

  }}

  autoplay

  autoplayDelay={3}

  autoplayLoop

  index={0}

  data={data.truyen\_view}

  renderItem={({ item }) => (

    <View style={{ margin: 5, flex: 1 }}>

      <TouchableOpacity

        onPress={() => {

          navigation.navigate("Detail", { slug: item.slug });

        }}

      >

        <Image

          style={{ width: (windowWidth - 20) / 2, height: 250 }}

          source={{

            uri: `http://127.0.0.1:8000/${item.path}`,

          }}

        />

        <Text

          numberOfLines={item.tentruyen.length > 20 ? 1 : 0}

          style={{

            color: "#fafafa",

            textAlign: "center",

            maxWidth: (windowWidth - 20) / 2,

            overflow: "hidden",

            fontWeight: "bold",

          }}

        >

          {item.tentruyen}

        </Text>

      </TouchableOpacity>

    </View>

  )}

/>;

### 4.1.5 Hiển thị chi tiết truyện

<ScrollView

  refreshControl={

    <RefreshControl refreshing={refreshing} onRefresh={handleRefresh} />

  }

>

  {data.truyen ? (

    <>

      <View

        style={{

          flexDirection: "row",

          marginBottom: 3,

        }}

      >

        <View style={{ marginTop: 10 }}>

          <Image

            style={{ width: 100, height: 150 }}

            source={{

              uri: `http://127.0.0.1:8000/${data.truyen.path}`,

            }}

          />

          <TouchableOpacity

            onPress={() => {

              navigation.navigate("Reading", {

                slug: data.truyen.slug,

                id: data.max\_tap,

              });

            }}

          >

            <Text

              style={{

                color: "#fafafa",

                marginBottom: 5,

                borderWidth: 2,

                borderColor: "#333",

                marginTop: 5,

                padding: 3,

                borderRadius: 5,

                fontSize: 13,

                textAlign: "center",

                backgroundColor: "green",

              }}

            >

              Đọc mới nhất

            </Text>

          </TouchableOpacity>

        </View>

        <View

          style={{

            marginTop: 10,

            marginLeft: 10,

            flex: 1,

          }}

        >

          <View

            style={{

              backgroundColor: "#333",

              marginBottom: 10,

              padding: 5,

            }}

          >

            <Text

              style={{

                color: "#fafafa",

                marginBottom: 5,

                fontSize: 20,

                fontWeight: "bold",

              }}

            >

              {data.truyen.tentruyen}

            </Text>

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

              <Image

                source={require("../asset/icon/eye.png")}

                resizeMode="contain"

                style={{

                  width: 18,

                  height: 18,

                  tintColor: "#fafafa",

                  marginRight: 5,

                }}

              />

              <Text style={{ color: "#fafafa", marginBottom: 5 }}>

                {data.truyen.view}

              </Text>

            </View>

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

              <Image

                source={require("../asset/icon/waiting.png")}

                resizeMode="contain"

                style={{

                  width: 18,

                  height: 18,

                  tintColor: "#fafafa",

                  marginRight: 5,

                }}

              />

              <Text style={{ color: "#fafafa", marginBottom: 5 }}>

                {" "}

                {data.truyen.status === 0 ? "Đang tiến hành" : "Hoàn thành"}

              </Text>

            </View>

          </View>

          <Text style={{ color: "#fafafa", marginBottom: 5 }}>

            Tác giả: {data.arr\_tacgia.map((item) => item.tentacgia).join(", ")}

          </Text>

          <View style={styles.container}>

            {data.arr\_theloai.map((item, index) => (

              <TouchableOpacity

                key={index}

                onPress={() => {

                  navigation.navigate("ListFilter", { id: item.id });

                }}

              >

                <Text style={styles.text}>{item.tentheloai}</Text>

              </TouchableOpacity>

            ))}

          </View>

        </View>

      </View>

      <Text style={{ color: "#fafafa", marginBottom: 10 }}>

        {data.truyen.mota}

      </Text>

    </>

  ) : (

    <Text

      style={{

        color: "#fafafa",

        textAlign: "center",

        justifyContent: "center",

      }}

    >

      Đang tải...

    </Text>

  )}

</ScrollView>;

### 4.1.5 Hàm tạo handle sự kiện lướt ảnh và hiển thị ảnh

const handleScroll = (event, id\_truyen) => {

  const contentOffsetY = event.nativeEvent.contentOffset.y;

  const currentIndex = Math.floor(contentOffsetY / screenHeight);

  db.transaction(tx => {

    tx.executeSql('UPDATE lichsu SET indexPath=? WHERE id\_truyen=?', [

      currentIndex,

      id\_truyen,

    ]);

  });

  console.log('Index của ảnh:', currentIndex);

};

<FlatList

  ref={flatListRef}

  data={data.arr\_path}

  keyExtractor={index => index.toString()}

  getItemLayout={(data, index) => ({

    length: screenHeight,

    offset: screenHeight \* index,

    index,

  })}

  renderItem={({item}) => (

    <View>

      <ReactNativeZoomableView

        maxZoom={3}

        minZoom={0.5}

        zoomStep={1}

        initialZoom={1}

        bindToBorders={true}

        onZoomAfter={this.logOutZoomState}

        pinchToZoomOutSensitivity={1}

        longPressDuration={1500}

        visualTouchFeedbackEnabled={true}

        >

        <Image

          style={{width: screenWidth, height: screenHeight, flex: 1}}

          resizeMode="stretch"

          source={{uri: `http://127.0.0.1:8000/${item}`}}

        />

      </ReactNativeZoomableView>

    </View>

  )}

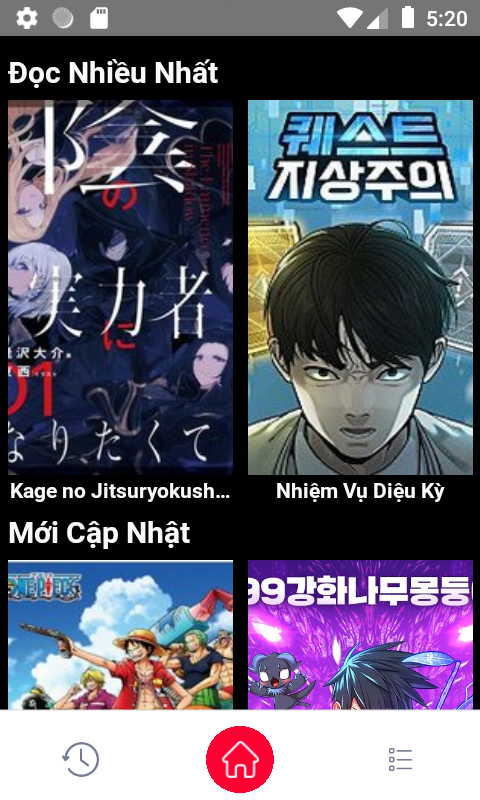
  onScroll={event => handleScroll(event, data.truyen.id)}

/>

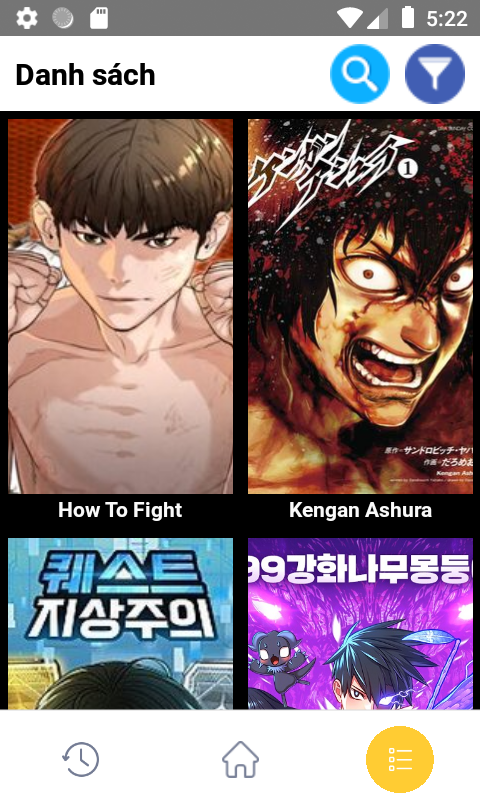
## 4.2. Kết quả lập trình

### 4.2.1 Màn hình chính

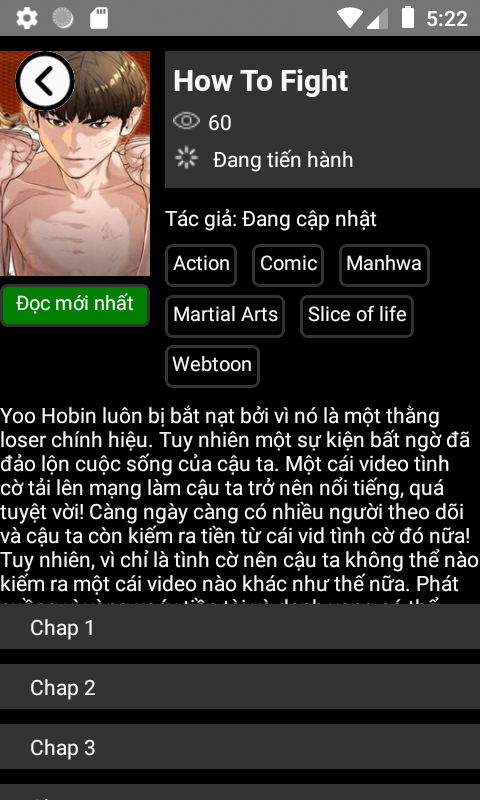
* Màn hình chính



* Màn hình danh sách truyện



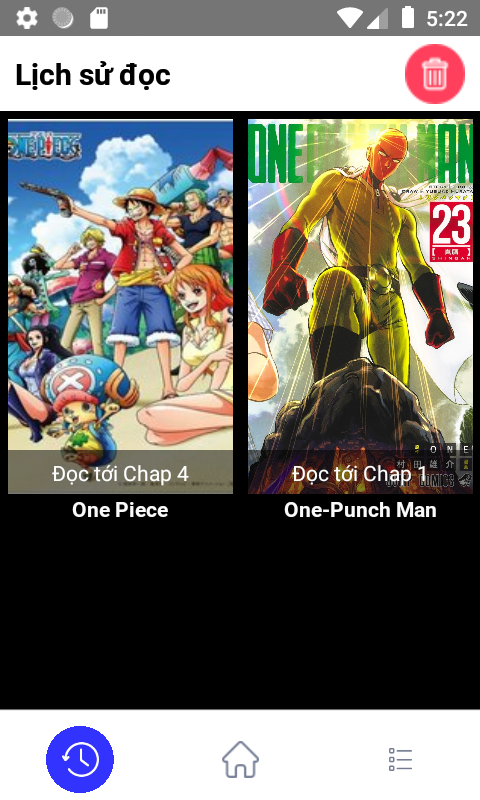
* Màn hình chi tiết truyện



* Màn hình đọc truyện



* Màn hình lịch sử đọc



# CHƯƠNG V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1. Kết luận

### 5.1.1. Kết quả đạt được

Phần mềm đọc truyện có giao diện đơn giản và dễ sử dụng đã hoàn thành các chức năng cơ bản như: Xem thông tin truyện - xem từng tập truyện; Đề xuất truyện vừa được update và có nhiều lượt đọc; Tìm kiếm theo tên truyện; Lọc theo thể loại; Lưu, xoá lịch sử đọc cục bộ; Lưu trang đang đọc cuối cùng của tập theo lịch sử đọc; Zoom ảnh khi đọc truyện; Cập nhật dữ liệu theo thời gian thực.

### 5.1.2 Hạn Chế

Tuy nhiên, phần mềm vẫn còn một số hạn chế như việc xử lý lưu trang đang đọc của người dùng ảnh hưởng tới hiệu năng chung của ứng dụng.

## 5.2. Hướng phát triển

Từ những hạn chế trên, cần có một hướng phát triển mới để đề tài ngày càng hoàn thiện hơn:

* Cải thiện giải thuật lưu trang đọc của người dùng
* Cho phép đăng nhập với tài khoản người dùng trên web và đồng bộ dữ liệu giữa web và mobile (đồng bộ truyện theo dõi, lịch sử…)