ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

CHƯƠNG 8

ĐÁNH GIÁ HIỆU NĂNG ỨNG DỤNG ĐA NỀN TẢNG

Mục tiêu

Sau bài học này sinh viên sẽ:

+ Có cái nhìn tổng quan và giải thích được các vấn đề hiệu năng của React Native cùng Flutter trên cả hai nền tảng
+ Nắm được nguyên lý chung để thực hiện việc so sánh hiệu năng giữa hai nền tảng bất kỳ
+ Biết được các ứng viên xuất sắc nhất để đo đạc hiệu năng ứng dụng

Mục lục

Mục lục

1. Giới thiệu

- Các thiết bị di động sử dụng các nền tảng khác nhau. Các nền tảng phổ biến nhất hiện nay là Android, Apple iOS.
- Sự đa dạng này trong các nền tảng là một thách thức đối với các doanh nghiệp trong việc phát triển các ứng dụng nhắm mục tiêu tất cả hoặc hầu hết các nền tảng từ một cơ sở mã duy nhất.
- Do đó, phát triển ứng dụng đa nền tảng là một giải pháp thay thế để nhắm mục tiêu vào hầu hết các nền tảng hơn là chạy theo phát triển ứng dụng native.

1.1. Background

Hiện có nhiều framework phát triển ứng dụng đa nền tảng trên thị trường.

Framework/Tool	**Development Uses**
Xamarin	C#
React Native	JavaScript
Flutter	Dart
PhoneGap	HTML5, JavaScript
Ionic	AngularJS, JavaScript
Native Script	AngularJS, Vue.js, TypeScript, JavaScript

1.2. Mục tiêu

Ta sẽ so sánh hai bộ phát triển ứng dụng đa nền tảng được sử dụng rộng rãi nhất là React Native và Flutter. Cụ thể là mức sử dụng CPU, GPU và bộ nhớ của một ứng dụng được phát triển bởi React Native và Flutter chạy trên nền tảng iOS và Android.

Hầu hết, các bộ công cụ phát triển ứng dụng đa nền tảng cung cấp bộ core của việc biên dịch mã ứng dụng sang mã native, nhưng vẫn có sự khác biệt giữa các bộ công cụ này.

Mục lục

2. React Native vs Flutter

Hầu hết các framework phát triển ứng dụng di động đa nền tảng không thể cung cấp trải nghiệm người dùng liền mạch ở mọi khía cạnh.

Nhiều app nổi tiếng đã được xây dựng bằng Flutter và RN.

Do đó, React Native và Flutter đều có những ưu và nhược điểm riêng. Ta đi so sánh vài trường hợp cụ thể:

2. React Native vs Flutter

Audio và Video:

+ React Native không cung cấp các core component hoạt động với các API Audio và Video của nền tảng. Tuy nhiên cộng đồng cung cấp các thư viện hỗ trợ, phổ biến nhất là reactnative-sound và react-native-video.
+ Flutter: Flutter core team không cung cấp plugin cho âm thanh, thư viện ngoài phổ biến nhất là flutter_sound, Flutter core team đang phát triển video_player plugin để hiển thị video.

2. React Native vs Flutter

Đồ họa 3D:

+ React Native không cung cấp các core component để hỗ trợ kết xuất 3D. Tuy nhiên cộng đồng cung cấp các thư viện hỗ trợ, phổ biến nhất là react-native-glmodel-view, expo-three.
+ Flutter: Theo tài liệu chính thức, Flutter không hỗ trợ 3D thông qua OpenGL (Open Graphics Library) hoặc các thư viện tương tự và Flutter tập trung vào kết xuất 2D. Tuy nhiên, tài liệu đã đề cập có một kế hoạch dài hạn để đưa ra một API 3D được tối ưu hóa.

2. React Native vs Flutter

Truy cập tệp:

+ React Native không cung cấp các core component để truy cập tệp. Tuy nhiên cộng đồng cung cấp các thư viện hỗ trợ, phổ biến nhất là react-native-fs. Cần có cấu hình nền tảng cụ thể để có quyền thiết lập truy cập hệ thống tệp.
+ Flutter: Flutter core team cung cấp plugin path_provider để tương tác với các vị trí thường được sử dụng trên hệ thống tệp Android và iOS.

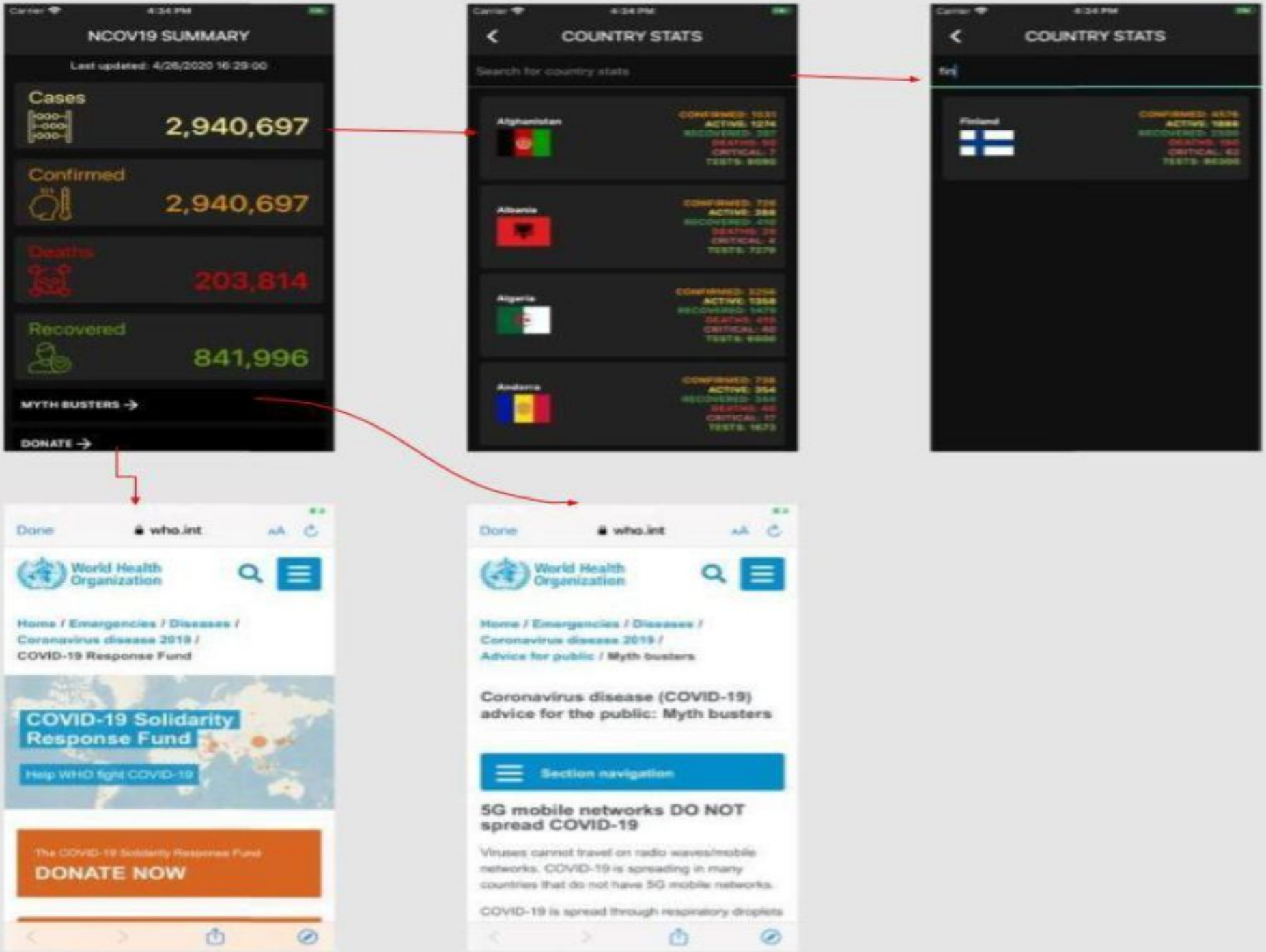
2. React Native vs Flutter

	React Native	**Flutter**
Programming Language	JavaScript (most popular among web developers and easy to adopt to React Native)	Dart (rarely used and less developer community)
Architecture	Flux (uses JavaScript bridge to communicate with native APIs)	SKia (often does not require bridge to communicate with native APIs but bigger in size)
Installation	Via package manager (NPM, HomeBrew)	Binary Download from source
API (UI and beyond)	Less core components	Rich in Widgets
Developer productivity	Depends on JavaScript skill	Depends on Dart skill
Community support	Very high	Rapidly growing
Testing support	Relies on third party for integration or UI testing	Rich set of packages for integration or UI testing
Build & release automation support	iOS deployment manual from Xcode	Command line interface for deployment (Android, iOS)
DevOps and CI/CD support	Not so easy CI/CD setup and relies on third party	Rich command line interface for easy CI/CD setup

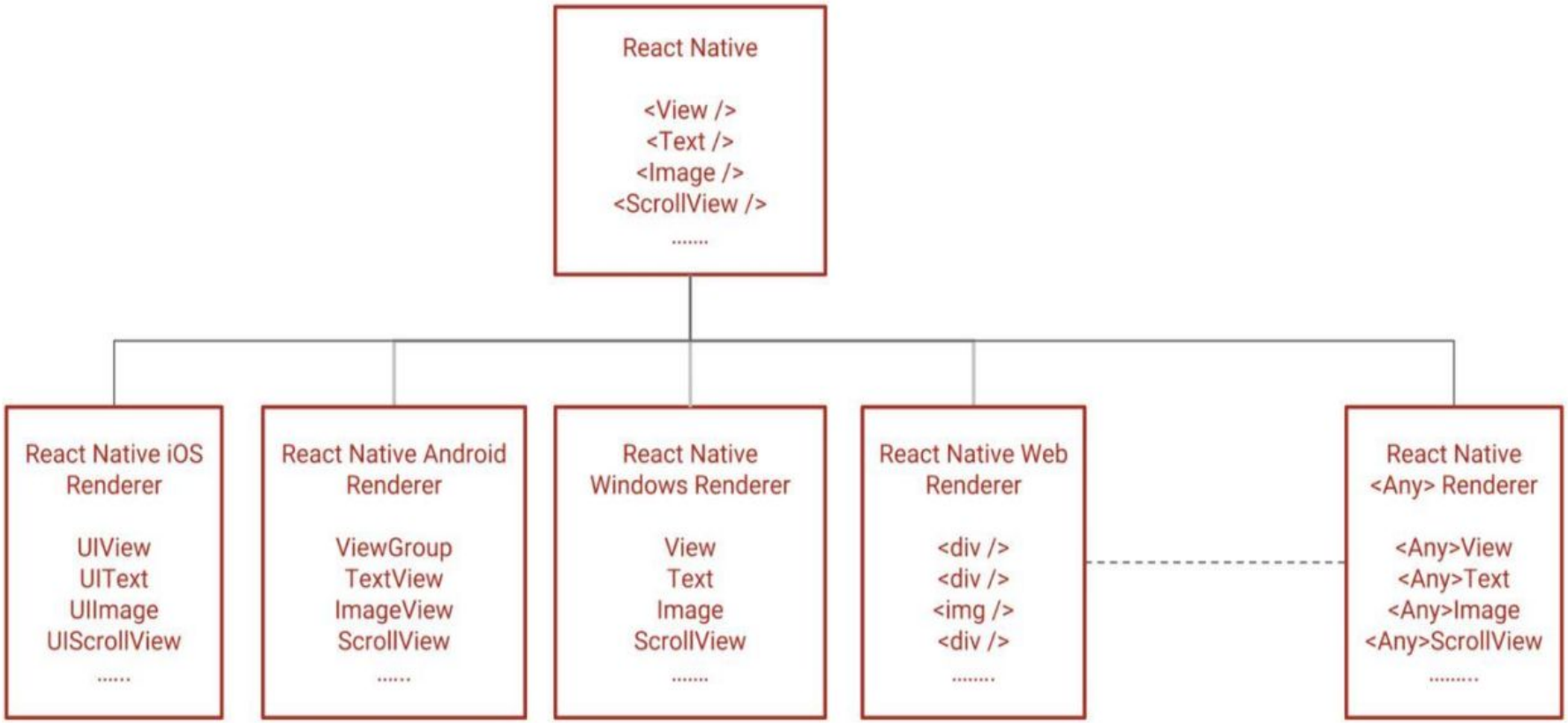
Mục lục

3. Implementations

- Ta sẽ triển khai ứng dụng di động theo dõi đại dịch COVID-19.
- Ứng dụng được khuyên nên dùng để khảo sát hiệu năng chủ yếu vì có các nhà cung cấp REST API miễn phí về chủ đề này.
- Ứng dụng có hai màn hình điều hướng, cụ thể là “NCOV19 SUMMARY” làm landing page và “COUNTRY STATS”.
-


NCOV19 SUMMARY
Last updated: 4/26/2020 16:29:00
Cases
2,940,697
Confirmed
2,940,697
Deaths
203,814
Recovered
841,996
MYTH BUSTERS →
DONATE →
COUNTRY STATS
Search for country stats
Afghanistan
Albania
Algeria
Andorra
COUNTRY STATS
fin
Finland
who.int
World Health Organization
Home / Emergencies / Diseases /
Coronavirus disease 2019 /
COVID-19 Response Fund
COVID-19 Solidarity Response Fund
Help WHO fight COVID-19
The COVID-19 Solidarity Response Fund
DONATE NOW
who.int
World Health Organization
Home / Emergencies / Diseases /
Coronavirus disease 2019 /
Advice for public / Myth busters
Coronavirus disease (COVID-19) advice for the public: Myth busters
Section navigation
5G mobile networks DO NOT spread COVID-19
Viruses cannot travel on radio waves/mobile networks. COVID-19 is spreading in many countries that do not have 5G mobile networks.
COVID-19 is spread through respiratory droplets

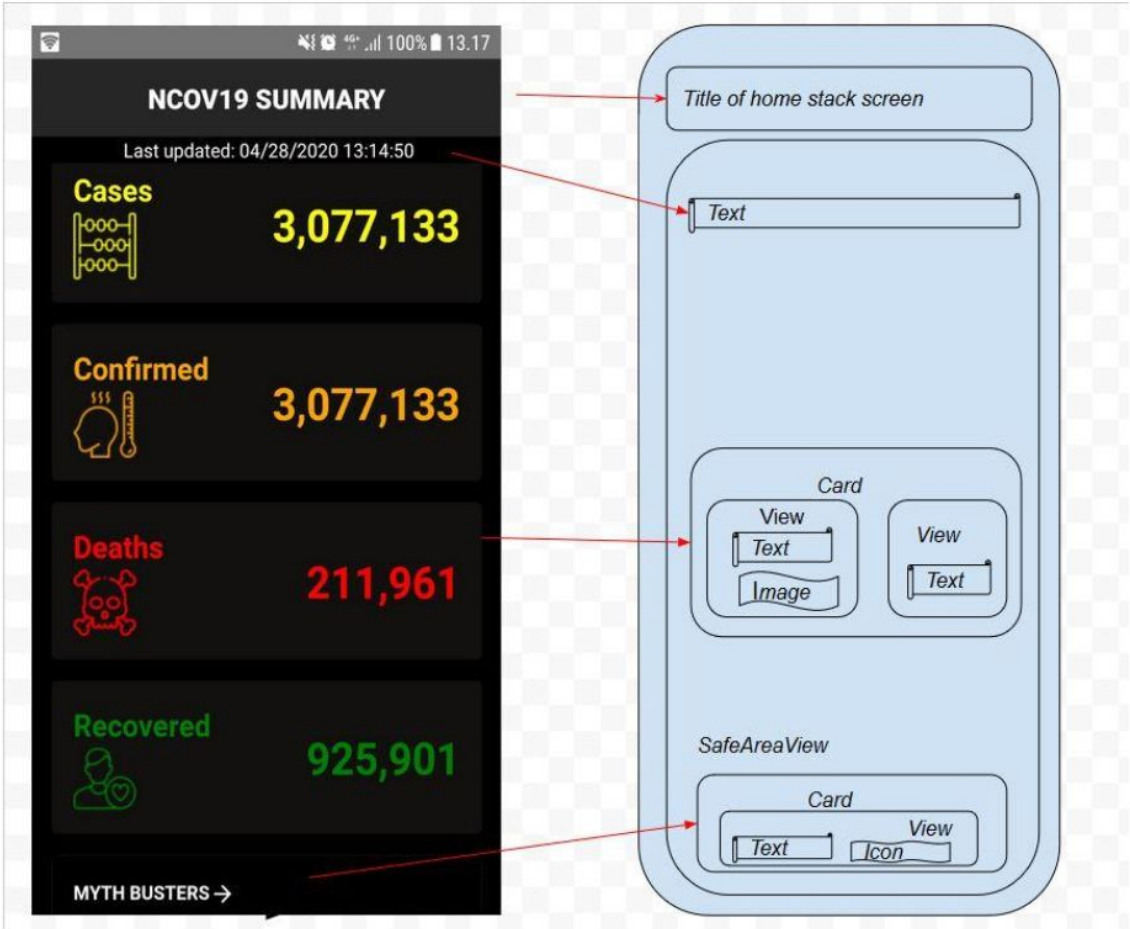
3.1. Ứng dụng di động React Native



Một số phần tử giao diện tương đương của React Native trong các nền tảng khác nhau

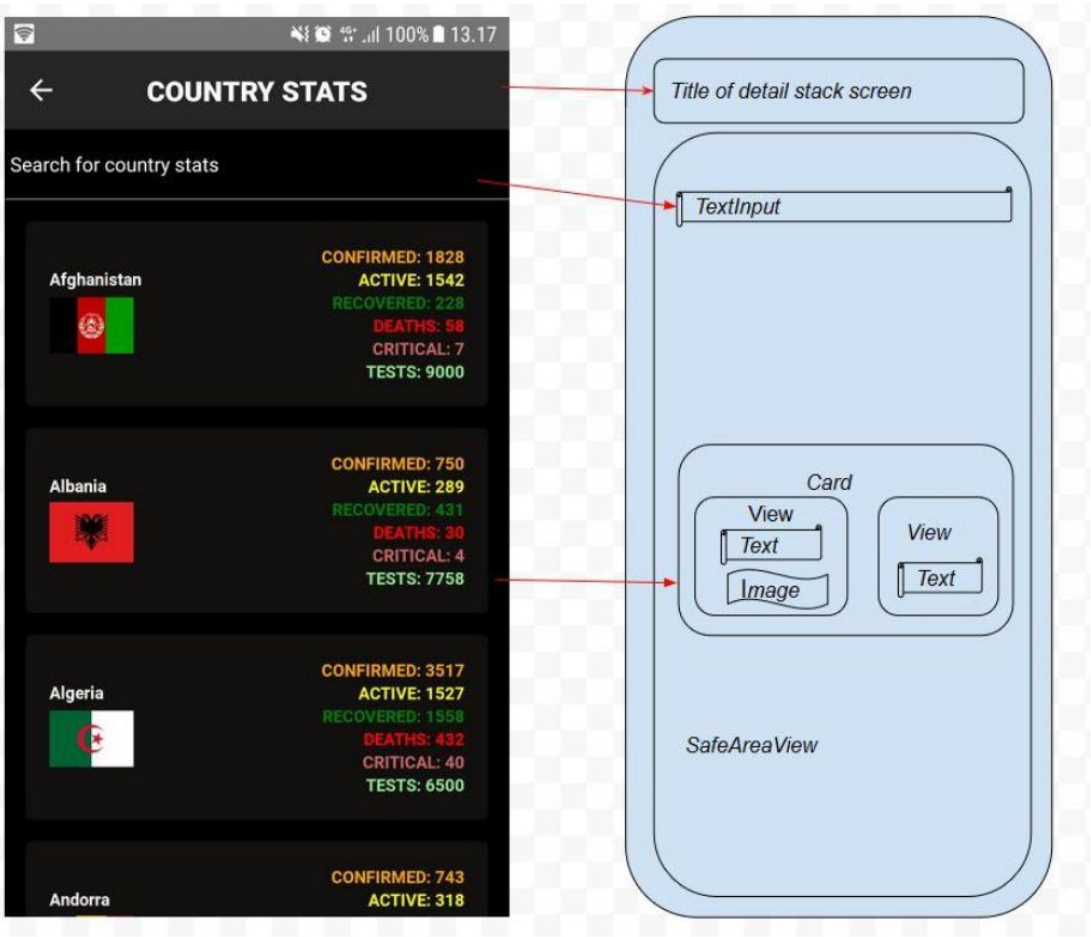
3.1. Ứng dụng di động React Native

Màn hình "NCOV19 SUMMARY" được xây dựng bằng một số core component do React Native cung cấp



3.1. Ứng dụng di động React Native

Màn hình "COUNTRY STATS" được xây dựng bằng một số core component do React Native cung cấp



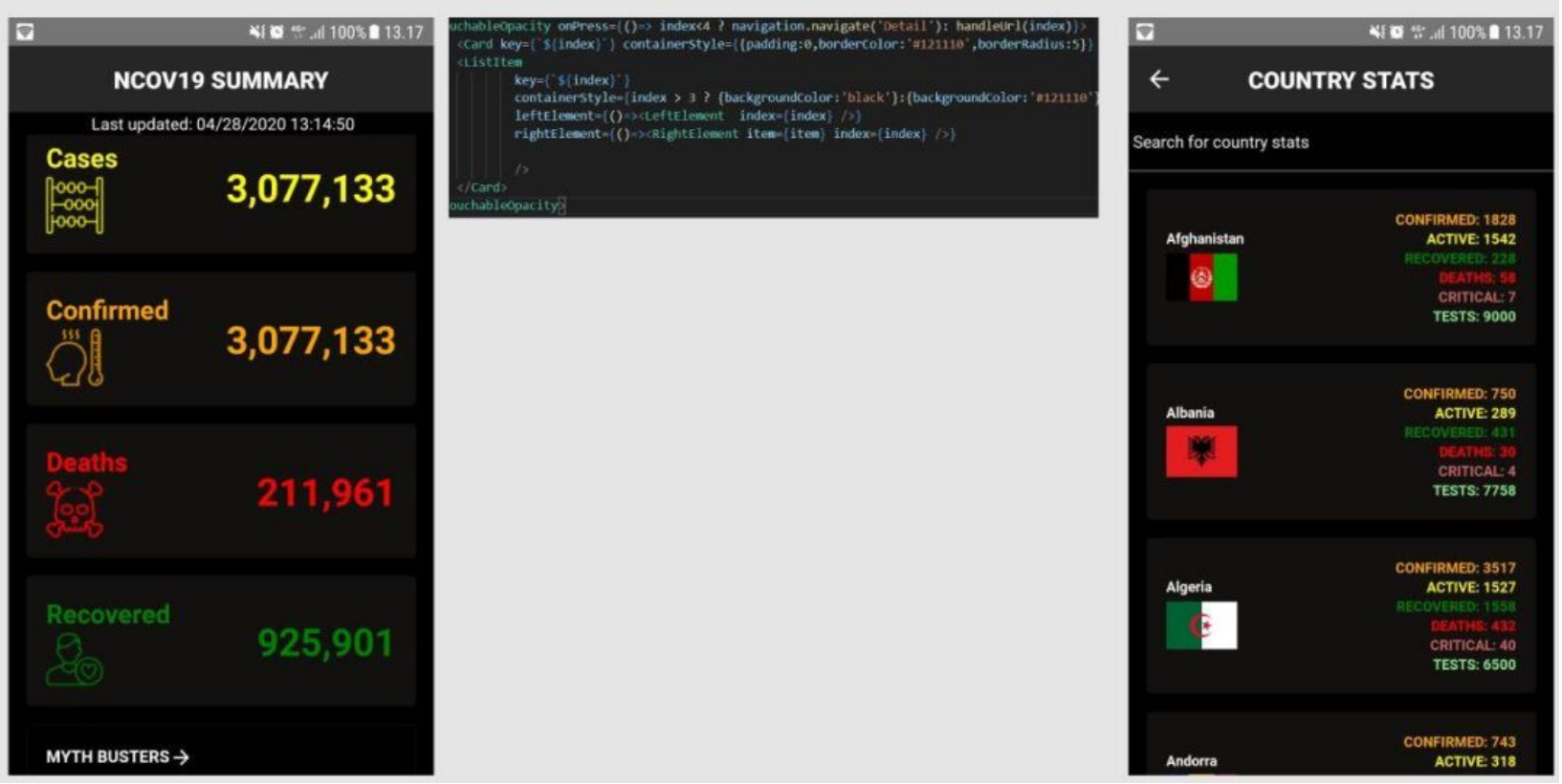
3.1. Ứng dụng di động React Native

Cách cấu hình trình điều hướng (navigator)

Provider là một Redux component, nó cung cấp thuộc tính store để lưu trạng thái ứng dụng

```
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';

import Home from './src/components/Home';
import Detail from './src/components/Details';
import { Provider } from 'react-redux';
import store from './store';


const Stack = createStackNavigator();


const App=() => {
  return (
    <Provider store={store}>
    <NavigationContainer>
        <Stack.Navigator initialRouteName="Home"
                screenOptions={{
                  headerStyle: {
                    backgroundColor: '#262626',

                  },
                  headerTintColor: '#fff',
                  headerTitleAlign:'center',
                  headerTitleStyle: {
                    fontWeight: 'bold',
                    color: 'white',
                  },
                }}>
        <Stack.Screen name="Home" component={Home} options={{ title: 'NCOV19 SUMMARY' }} />
        <Stack.Screen name="Detail" component={Detail} options={{ title: 'COUNTRY STATS' }}/>
      </Stack.Navigator>
    </NavigationContainer>
    </Provider>
  );
};

export default App;
```

3.1. Ứng dụng di động React Native



Cách sử dụng navigation để điều hướng giữa hai màn hình

3.1. Ứng dụng di động React Native

Redux dùng useSelector để lấy một trạng thái cụ thể từ Redux store

Redux dùng useDispatch để gửi action đến reducer

React dùng useEffect để thực hiện các side effect, trong trường hợp này muốn fetch dữ liệu từ API khi rendering 1 component

```
import React ,{useEffect ,useState} from 'react';
import { useDispatch,useSelector} from 'react-redux';
import {countriesAction} from '../../reducers/countriesReducer';
import { Card,ListItem } from 'react-native-elements';
import {
  StyleSheet,
   FlatList, View,ActivityIndicator,SafeAreaView,Text,Image,TextInput
} from 'react-native';



const Detail=()=>{
    const dispatch=useDispatch();
    const countries=useSelector(State=>State.countries);
    const [filteredCountry,setCountry]=useState([])
    const [searchText,setSearch]=useState('');

    //get countries
   useEffect(() => {
     dispatch(countriesAction());

    },[dispatch]);

    useEffect(() => {
      setCountry(
        countries.filter((country)=>{
          return country['country'].toLowerCase().startsWith(searchText);
       })
      )

    },[searchText,countries]);
```

3.2. Ứng dụng di động Flutter

Ta sẽ tạo ứng dụng flutter với các phần tử giao diện giống React Native nhất có thể để so sánh được khách quan.

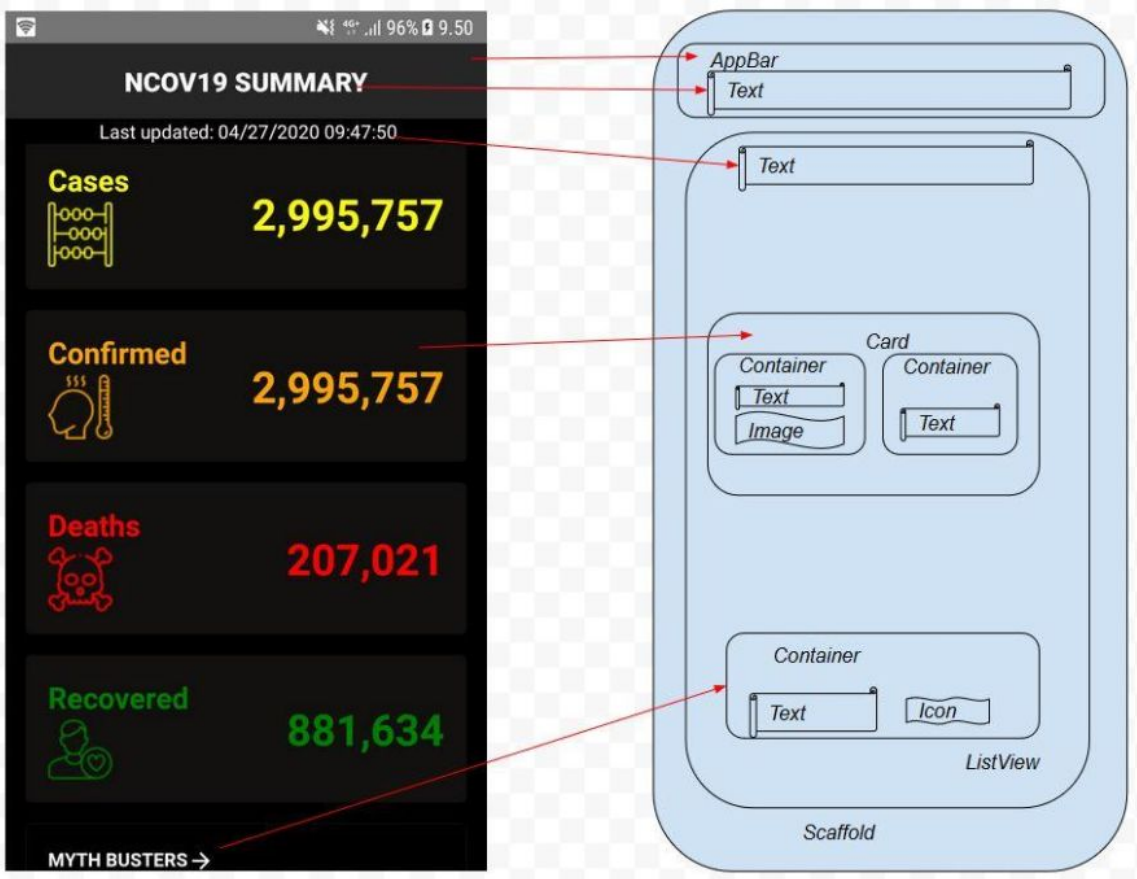
Khái niệm widget tương tự như component.

Flutter cung cấp một bộ widget phong phú dưới dạng core package, trong khi React Native cung cấp một số core UI component

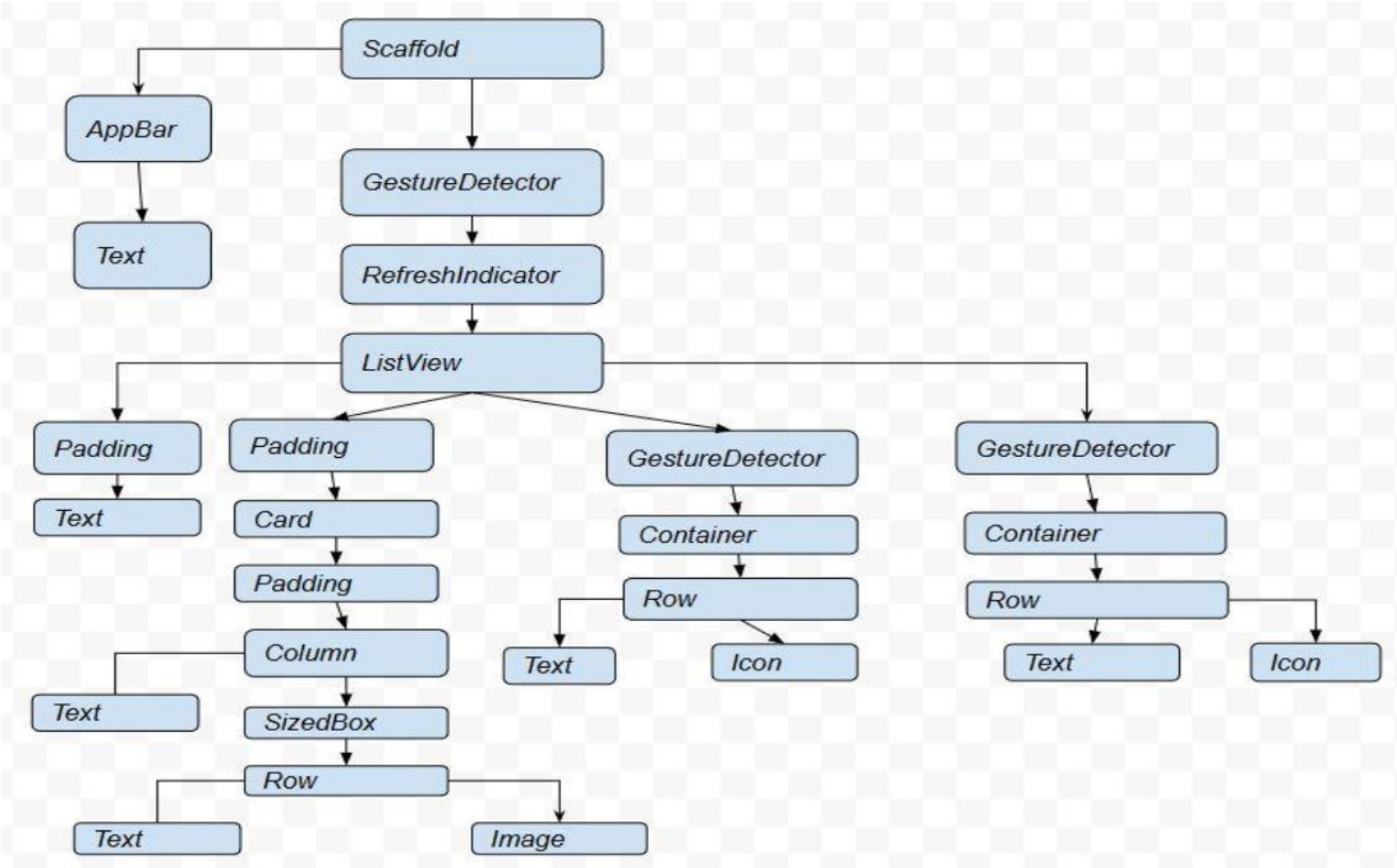
Cộng đồng React Native cung cấp các thư viện thay thế dưới dạng các gói bên ngoài có thể tải xuống để sử dụng.

3.2. Ứng dụng di động Flutter

Màn hình "NCOV19 SUMMARY" được xây dựng bằng một số core widget do Flutter cung cấp



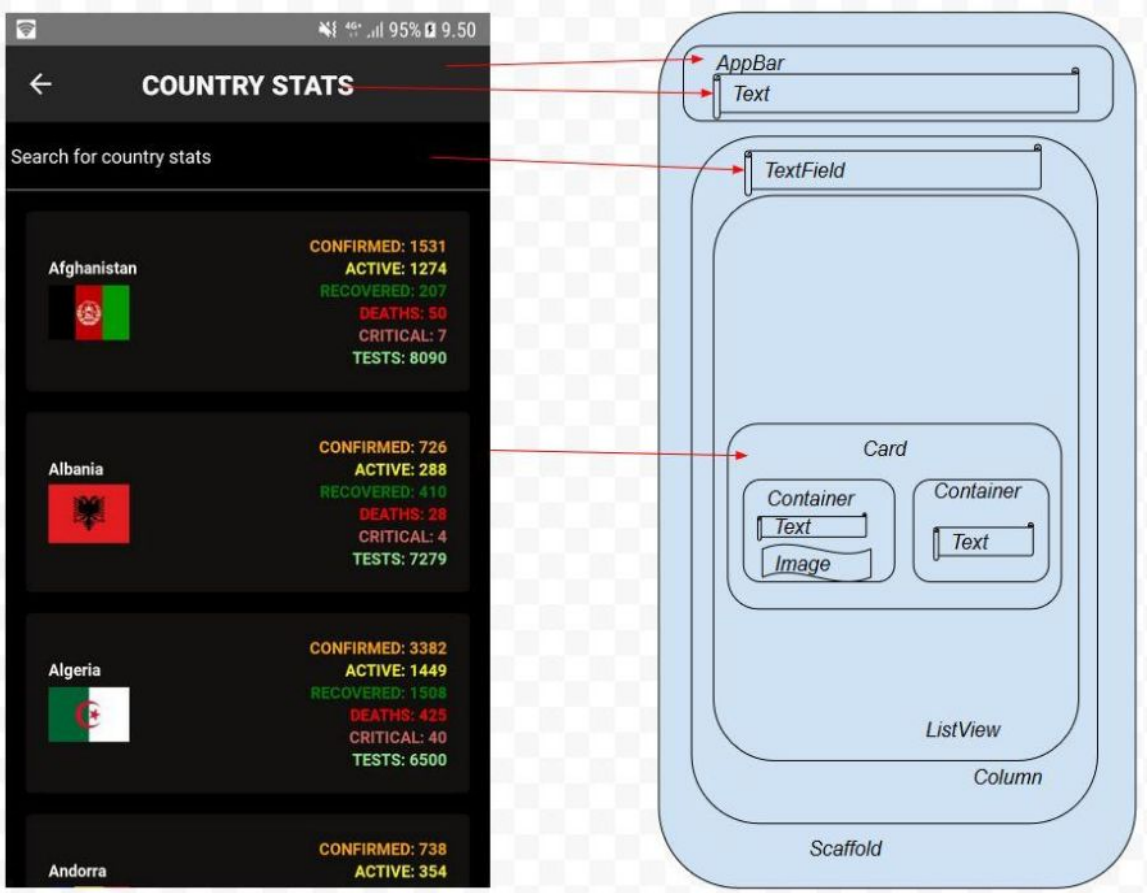
3.2. Ứng dụng di động Flutter



Widget tree cho màn hình "NCOV19 SUMMARY"

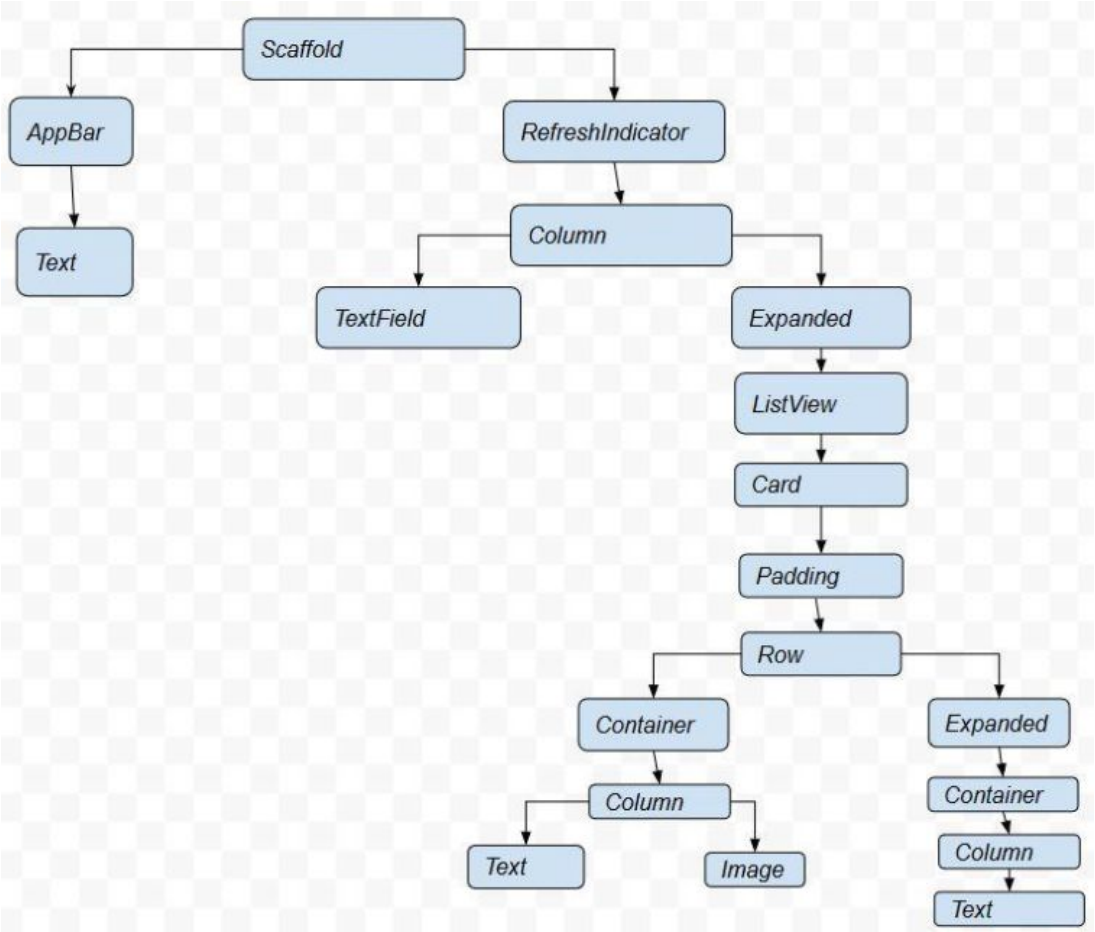
3.2. Ứng dụng di động Flutter

Màn hình "COUNTRY STATS" được xây dựng bằng một số core widget do Flutter cung cấp



3.2. Ứng dụng di động Flutter

Widget tree cho màn hình "COUNTRY STATS"



3.2. Ứng dụng di động Flutter

Cấu hình routes trong Flutter

```dart
import './api/services/api_service.dart';
import './api/services/api.dart';
import './api/repositories/data_repository.dart';
import './ui/dashboard.dart';


void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return Provider<DataRepository> (
      create: (_)=>DataRepository(
        apiService: APIService(API.sandbox()),
      ), // DataRepository
      child:MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'NCOV19 Info',
      theme: ThemeData.dark().copyWith(
        scaffoldBackgroundColor: Color(0xFF101010),
        cardColor: Color(0xFF222222),
      ),
      initialRoute: '/',
      routes: {
        '/':(context)=>Dashboard(),
        '/detail':(context)=>Detail()
      },
    ), // MaterialApp
    ); // Provider
  }
}
```

3.2. Ứng dụng di động Flutter

Cách sử dụng Navigator để điều hướng

```
Future<void> _updateData() async{
  final dataRepository = Provider.of<DataRepository>(context,listen:false);
  final endpointsData = await dataRepository.getAllEndpointData();
  setState(() {
    _endpointsData=endpointsData;
  });
}

@override
Widget build(BuildContext context) {
  final formatter = LastUpdatedDateFormatter(
            lastUpdated: _endpointsData !=null
                ? _endpointsData.values[EndPoint.cases].date
                :null,
            );
  return Scaffold(
    appBar: AppBar(
      centerTitle: true,
      title: Text('NCOV19 SUMMARY'),
    ), // AppBar
    body: _endpointsData ==null ? Center(child: CircularProgressIndicator(),):GestureDetector(

      onTap: () {
          Navigator.pushNamed(context,'/detail');
        },

      child: RefreshIndicator(
     onRefresh: _updateData,
     child:  ListView(
      children:<Widget>[
        LastUpdatedStatusText(
          text:formatter.lastUpdatedStatusText(),
        ), // LastUpdatedStatusText
```

3.2. Ứng dụng di động Flutter

```dart
import '../api/repositories/endpoints_data.dart';
import '../api/services/api.dart';
import './endpoint_card.dart';
import './last_updated_status_text.dart';
import '../api/repositories/data_repository.dart';

class Dashboard extends StatefulWidget{
  @override
  _DashboardState createState()=>_DashboardState();
}

class _DashboardState extends State<Dashboard>{
  EndpointsData _endpointsData;

  @override
  void initState(){
    super.initState();
    _updateData();
  }
  Future<void> _updateData() async{
    final dataRepository = Provider.of<DataRepository>(context,listen:false);
    final endpointsData = await dataRepository.getAllEndpointData();
    setState(() {
      _endpointsData=endpointsData;
    });
  }
```

Cách sử dụng provider

Mục lục

4. So sánh hiệu năng

- Để mô phỏng việc sử dụng thực tế của các ứng dụng, cả hai ứng dụng đều được chạy trên một thiết bị thực có kết nối dữ liệu 4G.
- Thiết bị Android được sử dụng để chạy các ứng dụng là Samsung SM-A510F, trong khi thiết bị iOS được sử dụng để chạy các ứng dụng là iPhone 7.
- Các độ đo là memory, CPU, GPU.

4. So sánh hiệu năng

- Các thao tác chính khi thực hiện phép đo:
  - Scrolling cards: Cuộn lên xuống để xem các phần tử của màn hình "NCOV19 SUMMARY"
  - Opening webview: Mở liên kết "MYTH BUSTERS" và truy cập vào website WHO
  - Rendering listview: Hiển thị danh sách các quốc gia trên màn hình "COUNTRY STATS", cũng bao gồm các network request tới RESTFUL API và tải hình ảnh từ URL
  - Filtering a list: Lọc các thống kê quốc gia cụ thể từ danh sách các quốc gia

4. So sánh hiệu năng

4.1. Ứng dụng iOS

4.2. Ứng dụng Android

4. So sánh hiệu năng

4.1. Ứng dụng iOS

4.2. Ứng dụng Android

4.1. Ứng dụng iOS (1)

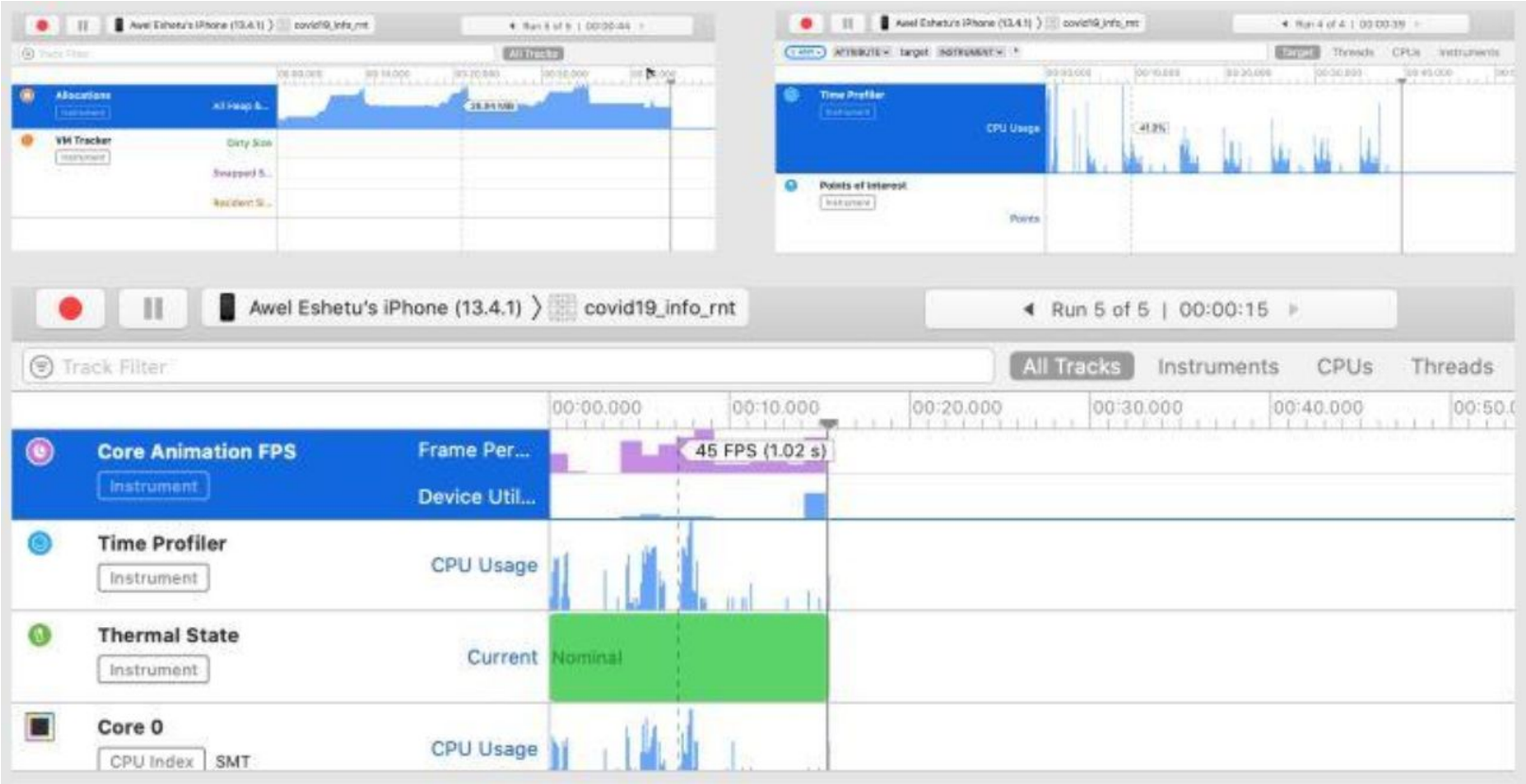
Phân tích hiệu năng được thực hiện nhờ công cụ “Instruments” của Apple, là một trình hiển thị và phân tích hiệu năng ứng dụng, được tích hợp trong Xcode IDE.

Instruments chứa một bộ công cụ có thể được sử dụng để kiểm tra hiệu năng của các ứng dụng iOS.

Trong số các công cụ đa dạng được cung cấp, "Time profiler tool" được sử dụng để phân tích mức sử dụng CPU.

“Core Animation Tool” để phân tích GPU, số khung hình mỗi giây.
“Allocations Tool” để phân tích việc sử dụng bộ nhớ (MiB - mebibyte).

4.1. Ứng dụng iOS



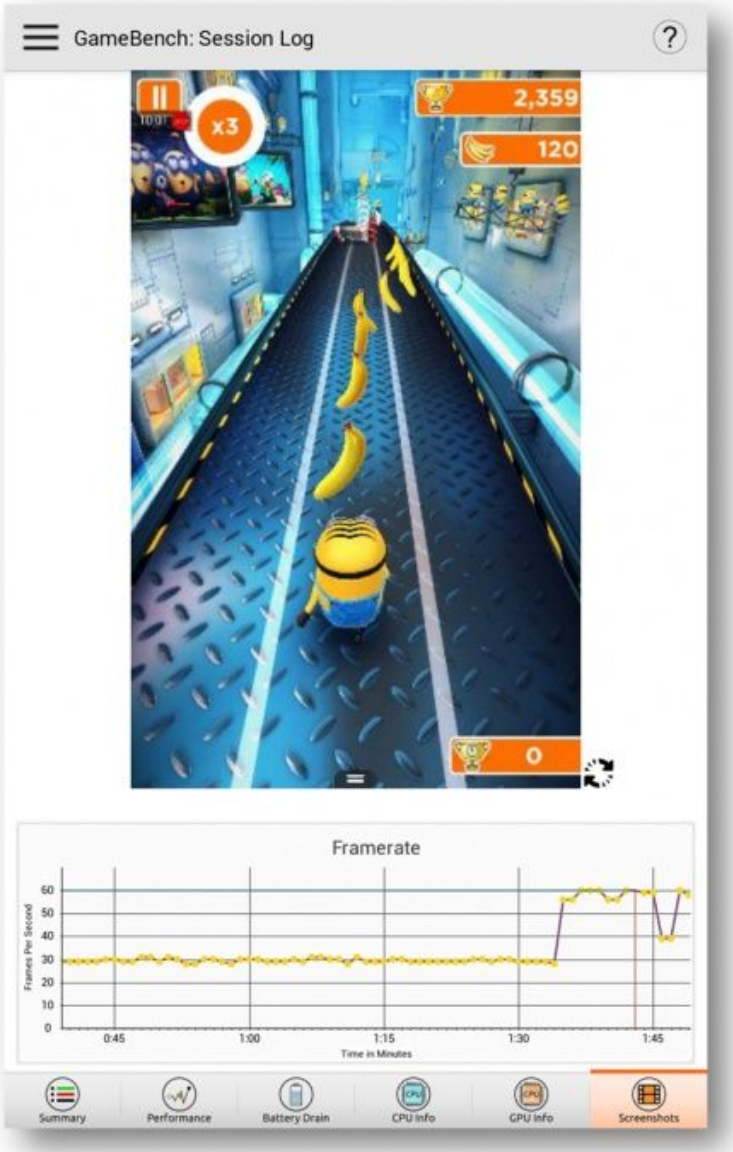
Minh họa công cụ Instruments

4.1. Ứng dụng iOS

4.1.1. CPU

4.1.2. Sử dụng bộ nhớ

4.1.3. GPU

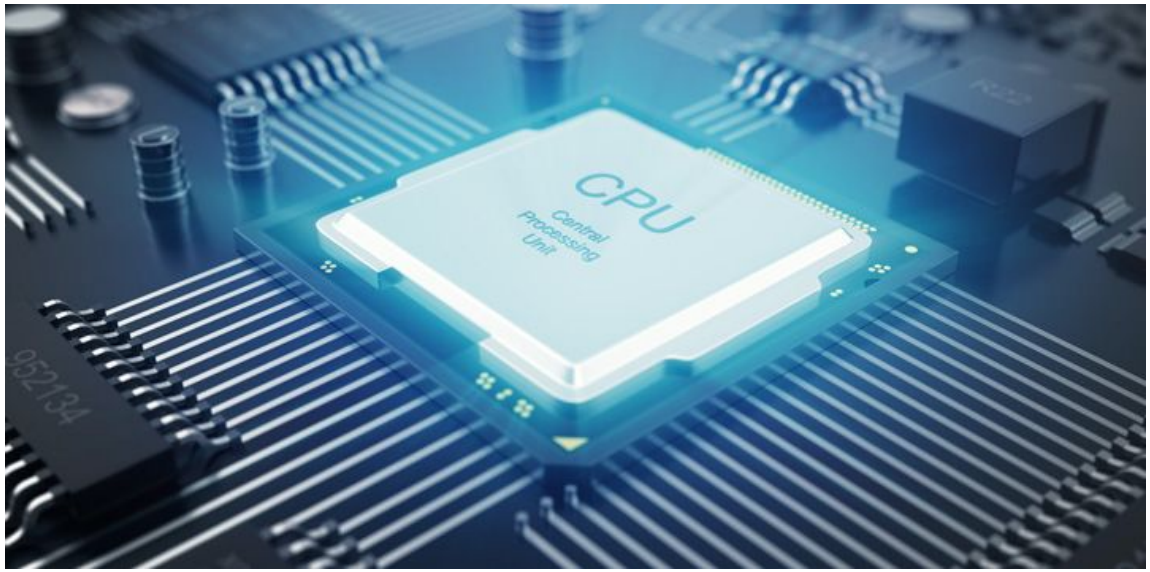


4.1. Ứng dụng iOS

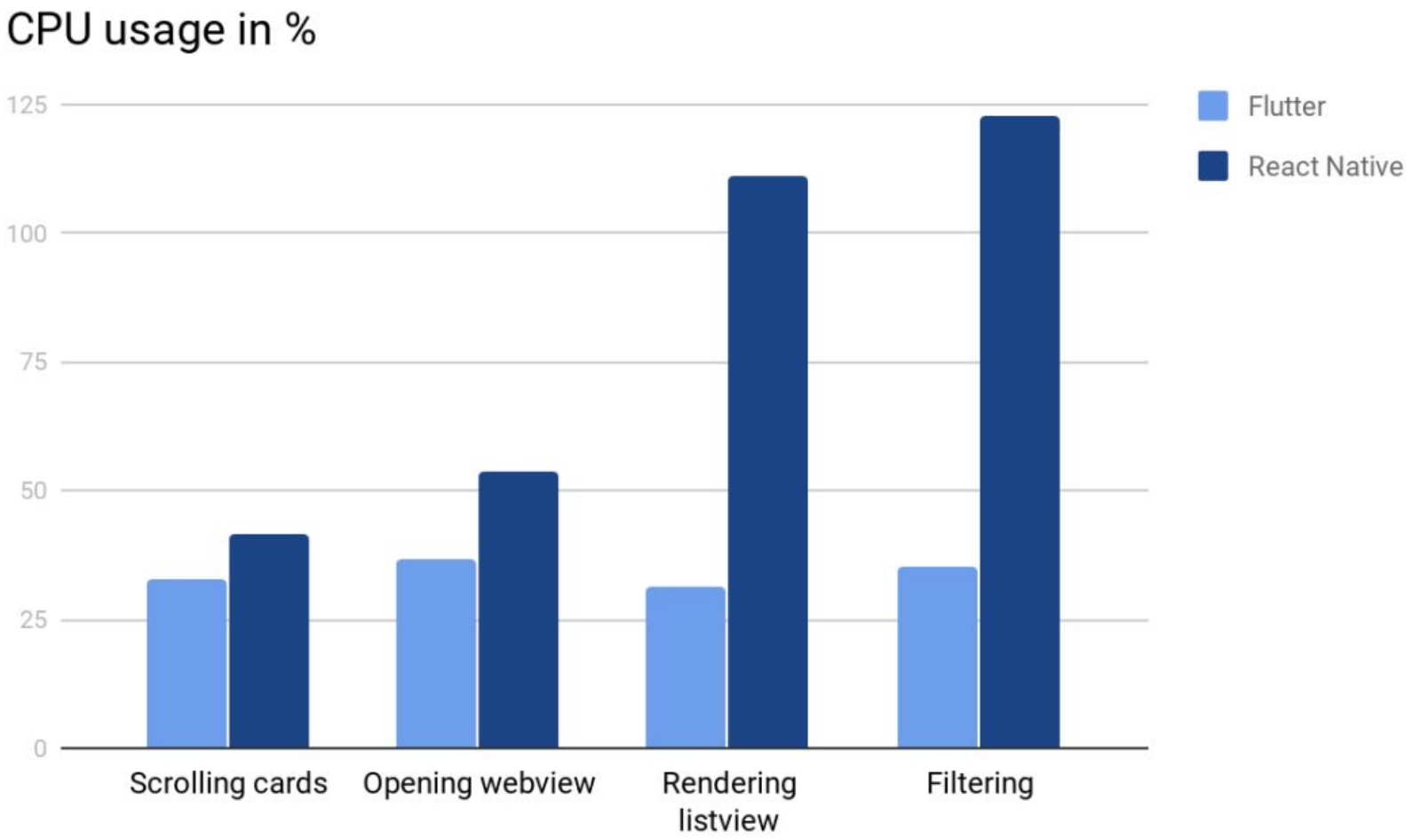
4.1.1. CPU

4.1.2. Sử dụng bộ nhớ

4.1.3. GPU



4.1.1. CPU


CPU usage in %
125
100
75
50
25
0
Flutter
React Native
Scrolling cards
Opening webview
Rendering listview
Filtering

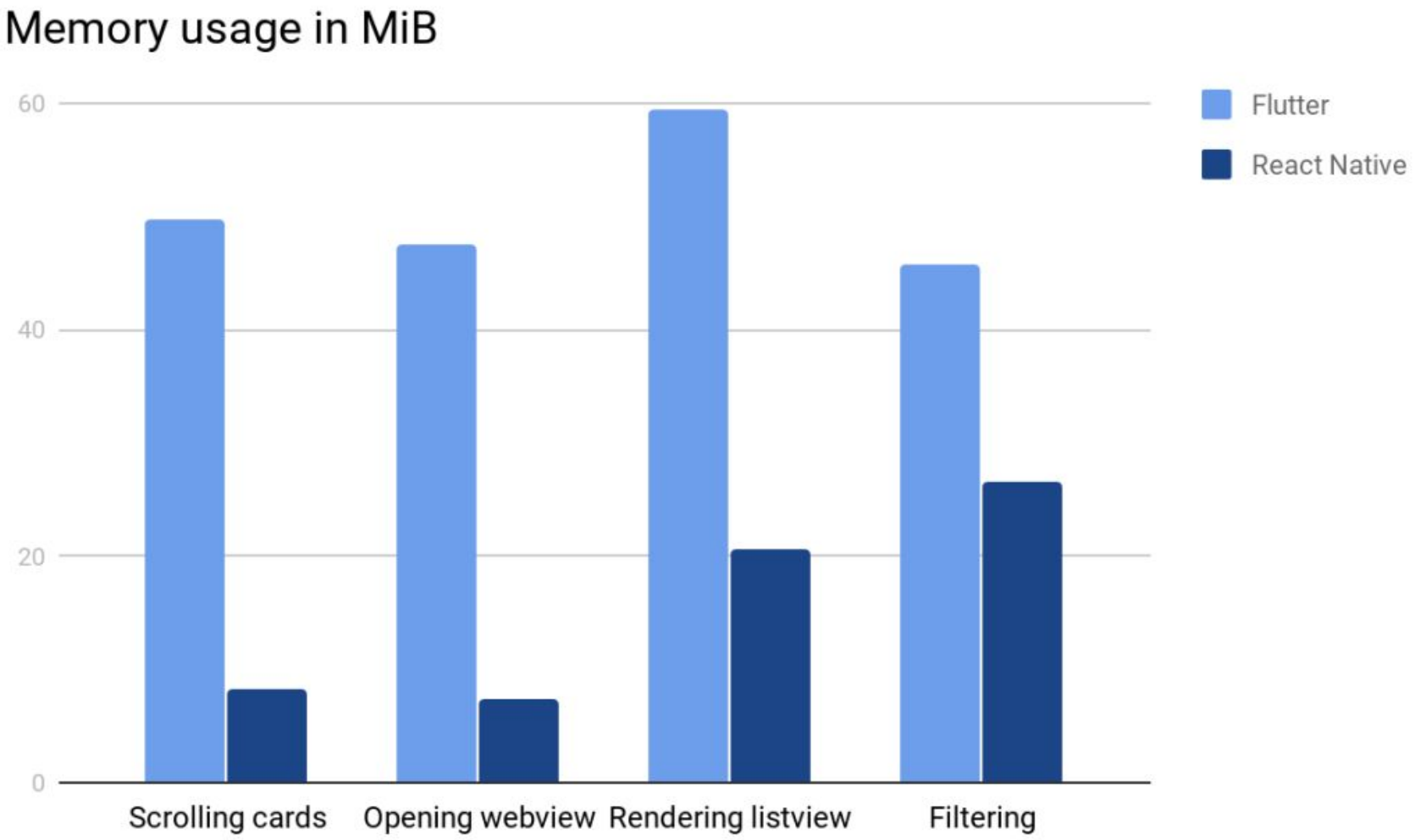
4.1. Ứng dụng iOS

4.1.1. CPU

4.1.2. Sử dụng bộ nhớ

4.1.3. GPU

4.1.2. Sử dụng bộ nhớ

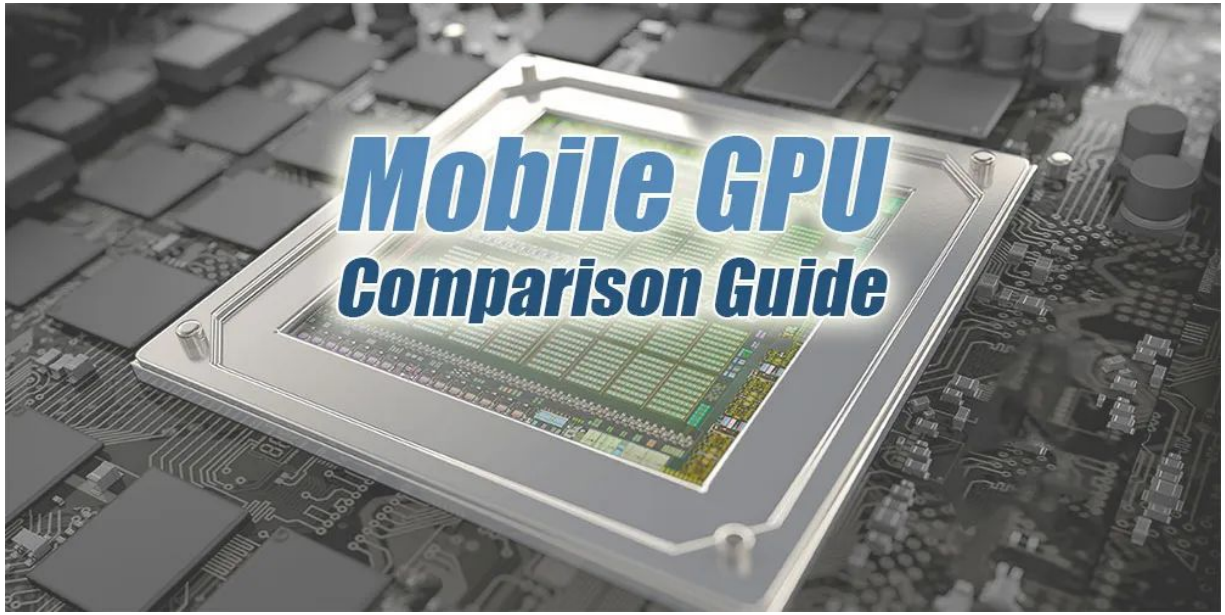


4.1. Ứng dụng iOS

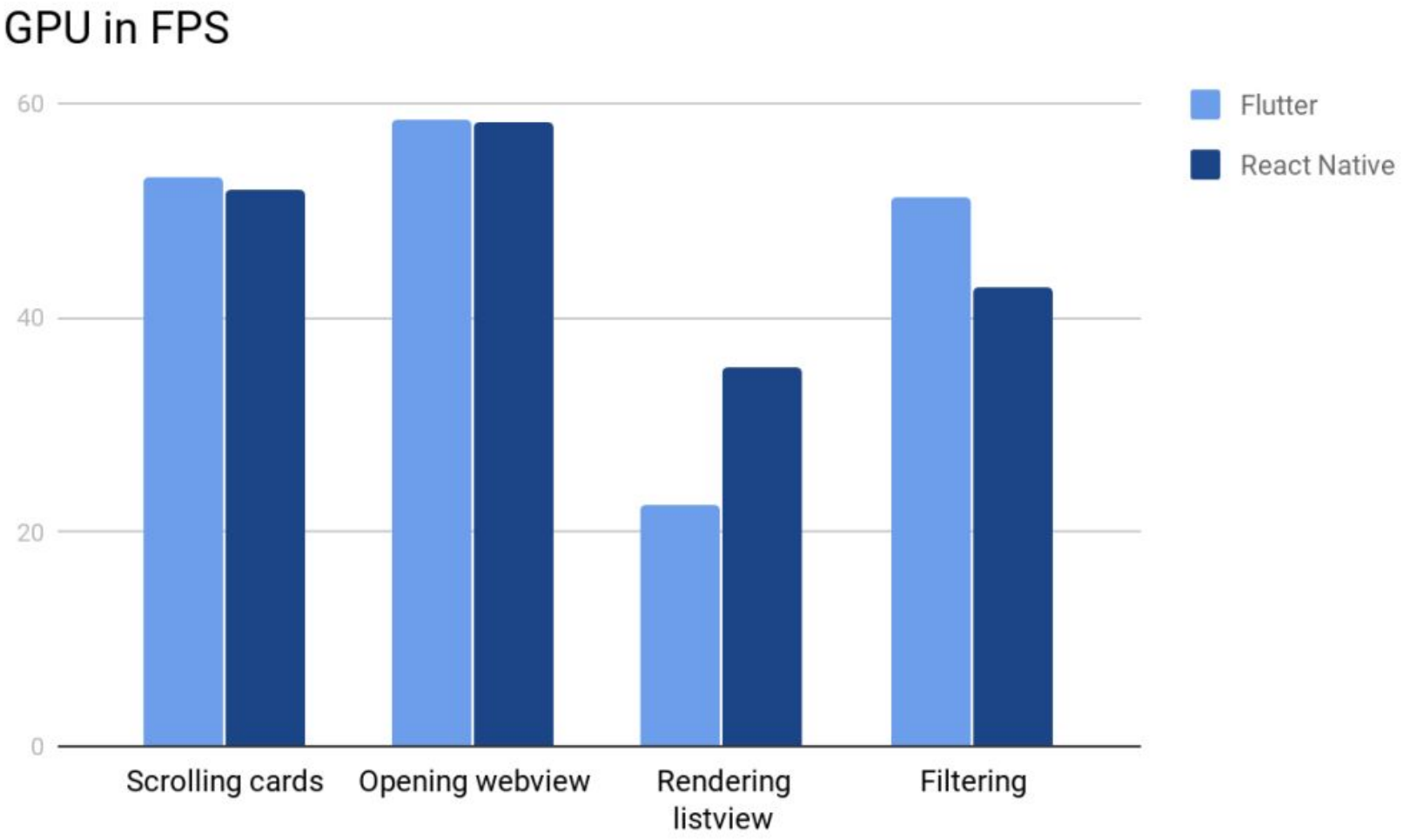
4.1.1. CPU

4.1.2. Sử dụng bộ nhớ

4.1.3. GPU



4.1.3. GPU


GPU in FPS
60
40
20
0
Flutter
React Native
Scrolling cards
Opening webview
Rendering listview
Filtering

4. So sánh hiệu năng

4.1. Ứng dụng iOS

4.2. Ứng dụng Android

4.2. Ứng dụng Android

Phân tích hiệu năng được thực hiện với các công cụ có sẵn trên Android Studio. Cụ thể ta dùng "Profiler" để phân tích CPU, Memory, Network, Energy.

Tuy nhiên Android Studio không cung cấp công cụ theo dõi GPU. Ta sử dụng "GPU rendering tool", một công cụ tích hợp sẵn trên điện thoại Android.

Công cụ này hiển thị trực quan lượng thời gian cần thiết để kết xuất khung hình so với mức chuẩn 16ms (tương đương 62.5 FPS) dưới dạng histogram.

4.2. Ứng dụng Android

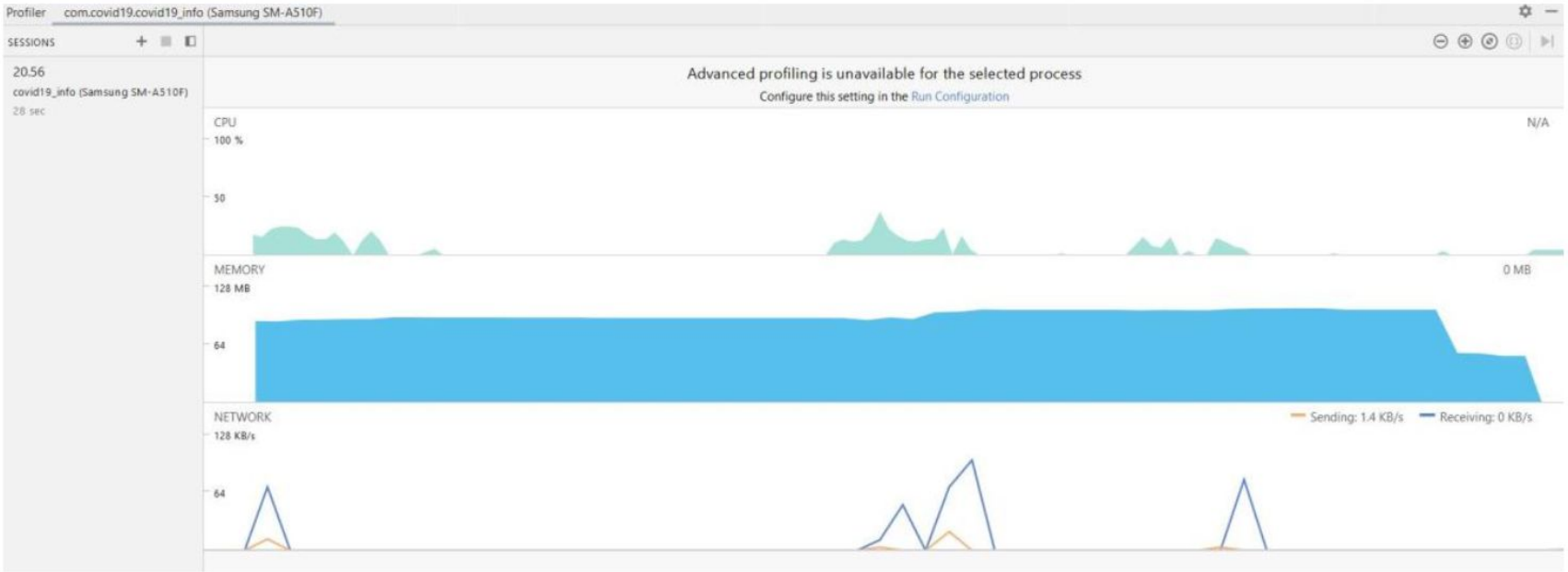
Component of Bar	Rendering Stage	Description
	Swap Buffers	Represents the time the CPU is waiting for the GPU to finish its work. If this bar gets tall, it means the app is doing too much work on the GPU.
	Command Issue	Represents the time spent by Android's 2D renderer issuing commands to OpenGL to draw and redraw display lists. The height of this bar is directly proportional to the sum of the time it takes each display list to execute—more display lists equals a taller red bar.
	Sync & Upload	Represents the time it takes to upload bitmap information to the GPU. A large segment indicates that the app is taking considerable time loading large amounts of graphics.
	Draw	Represents the time used to create and update the view's display lists. If this part of the bar is tall, there may be a lot of custom view drawing, or a lot of work in onDraw methods.
	Measure / Layout	Represents the amount of time spent on onLayout and onMeasure callbacks in the view hierarchy. A large segment indicates that the view hierarchy is taking a long time to process.
	Animation	Represents the time it took to evaluate all the animators that were running that frame. If this segment is large, your app could be using a custom animator that is not performing well or some unintended work is happening as a result of properties being updated.
	Input Handling	Represents the time that the app spent executing code inside of an input event callback. If this segment is large it indicates that the app is spending too much time processing the user input. Consider offloading such processing to a different thread.
	Misc Time / VSync Delay	Represents the time that the app spends executing operations in between two consecutive frames. It might be an indicator of too much processing happening in the UI thread that could be offloaded to a different thread.

GPU profiler component bars

4.2. Ứng dụng Android

Minh họa GPU rendering tool

4.2. Ứng dụng Android



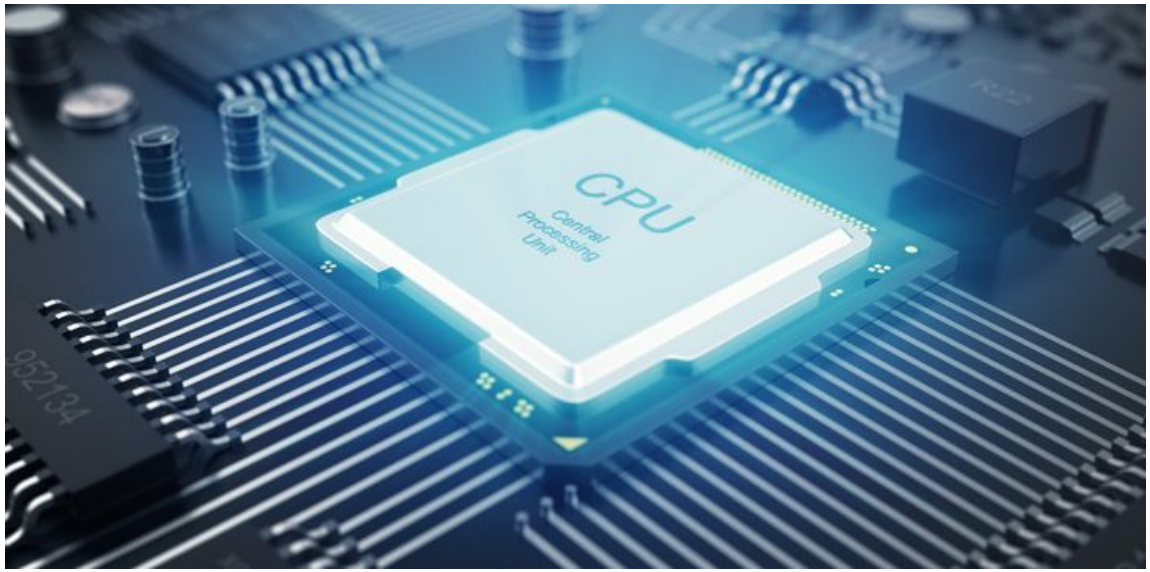
Minh họa Android Studio Profiler

4.2. Ứng dụng Android

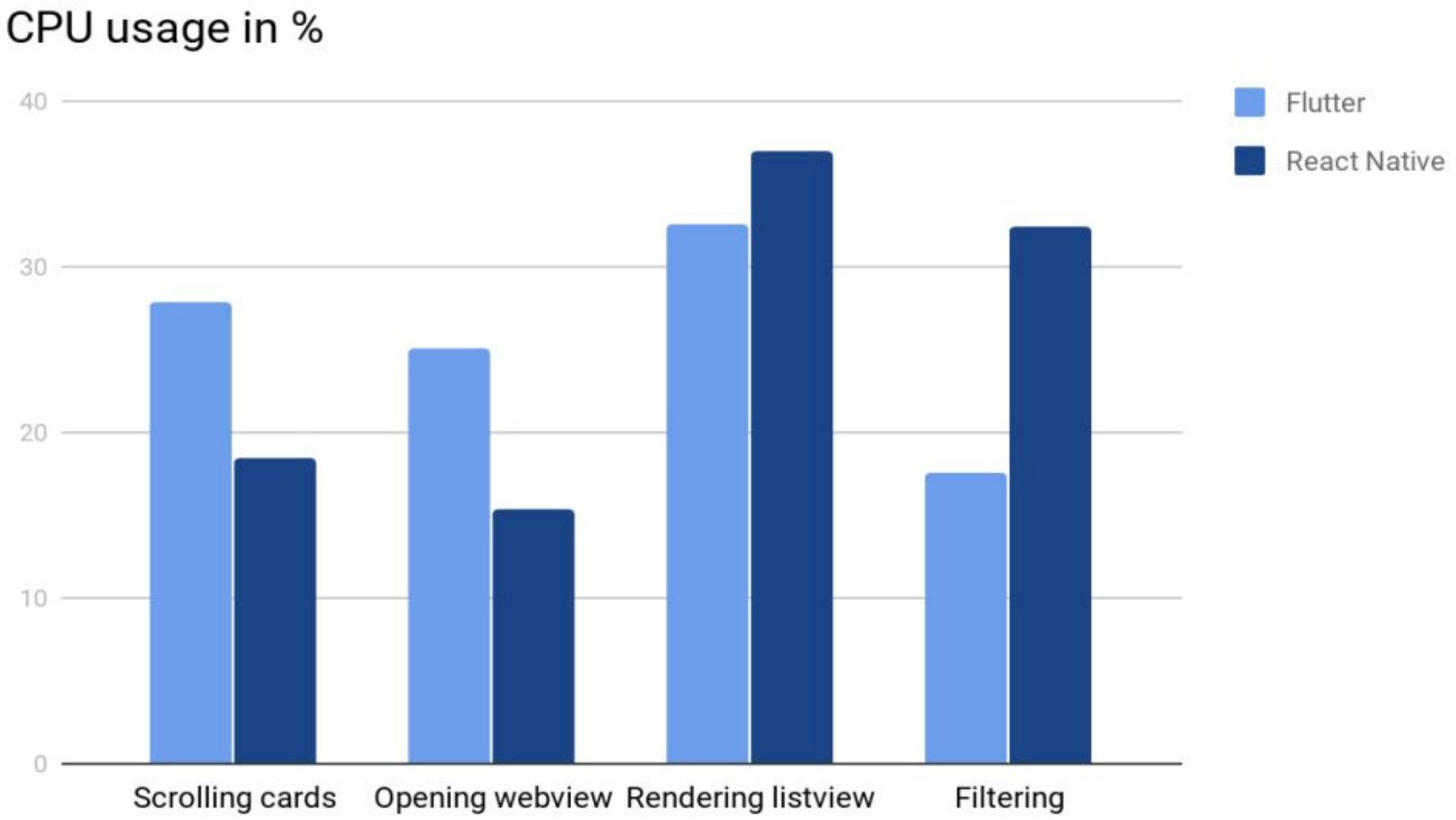
4.2.1. CPU

4.2.2. Sử dụng bộ nhớ

4.2.3. GPU



4.2.1. CPU


CPU usage in %
40
30
20
10
0
Scrolling cards
Opening webview
Rendering listview
Filtering
Flutter
React Native

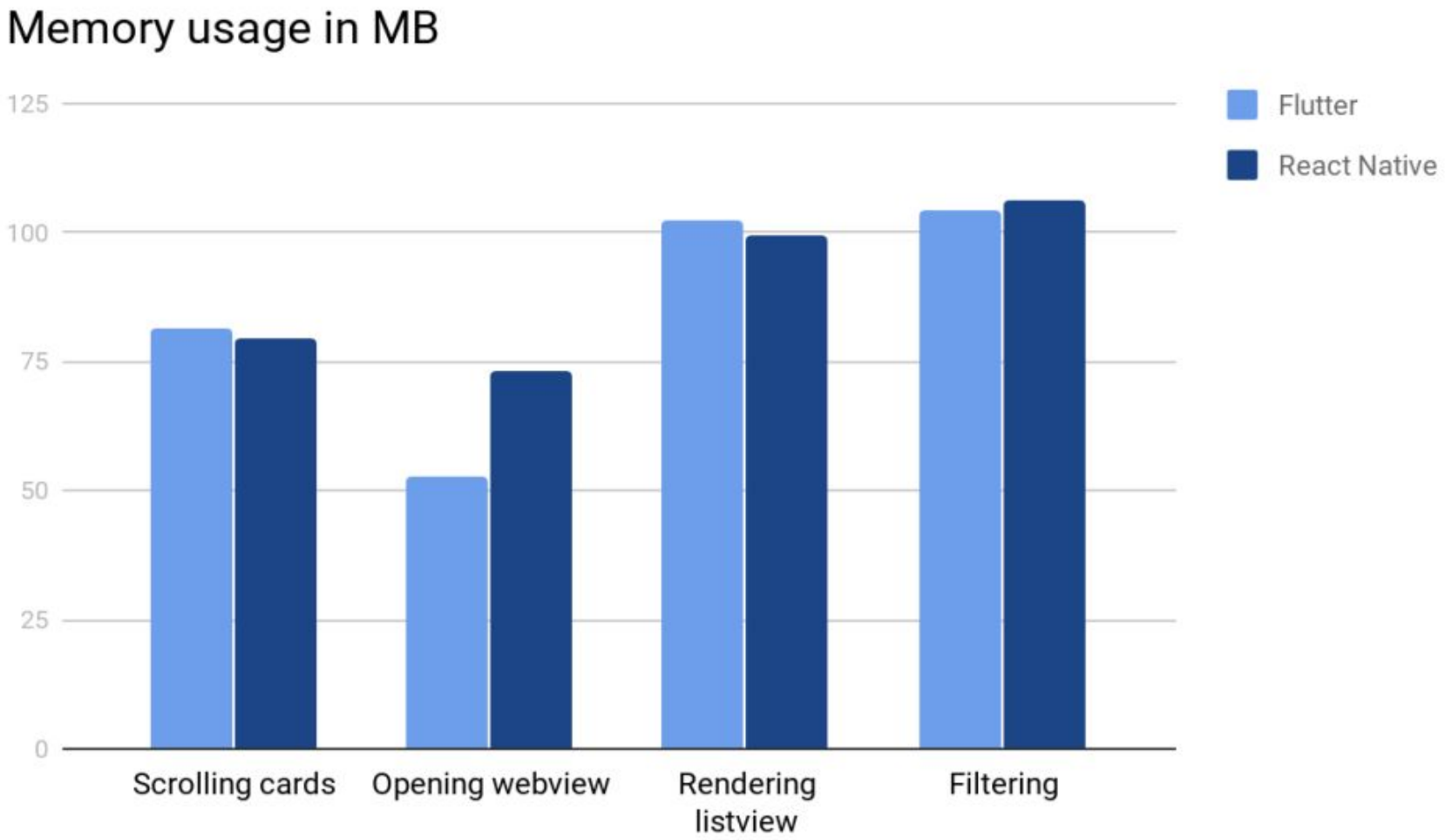
4.2. Ứng dụng Android

4.2.1. CPU

4.2.2. Sử dụng bộ nhớ

4.2.3. GPU

4.2.2. Sử dụng bộ nhớ

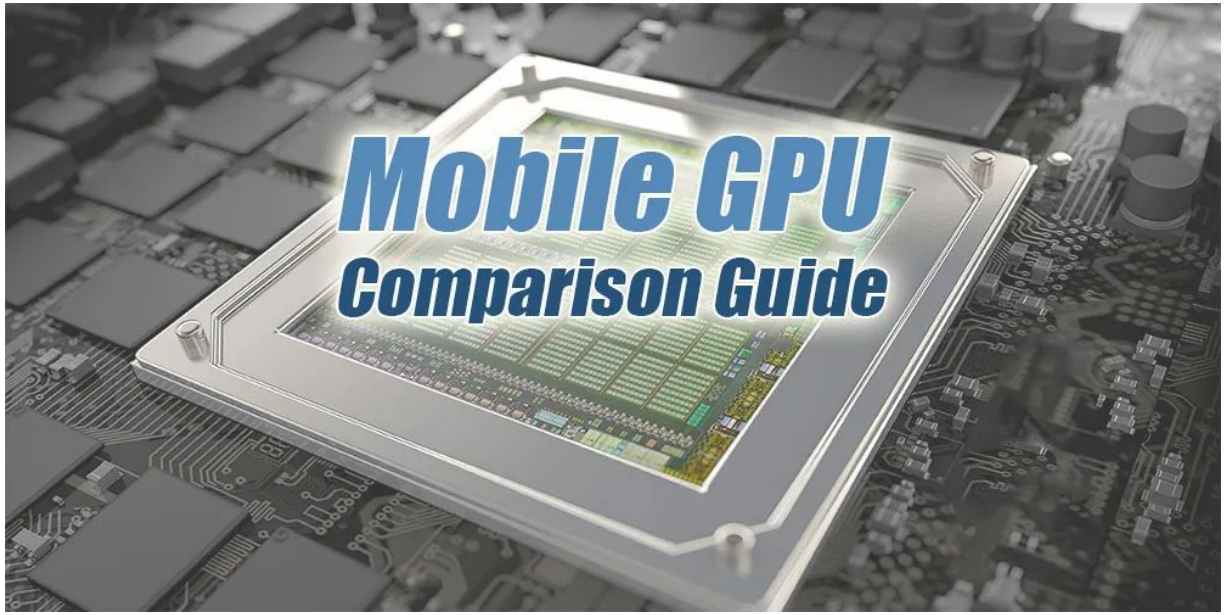


4.2. Ứng dụng Android

4.2.1. CPU

4.2.2. Sử dụng bộ nhớ

4.2.3. GPU

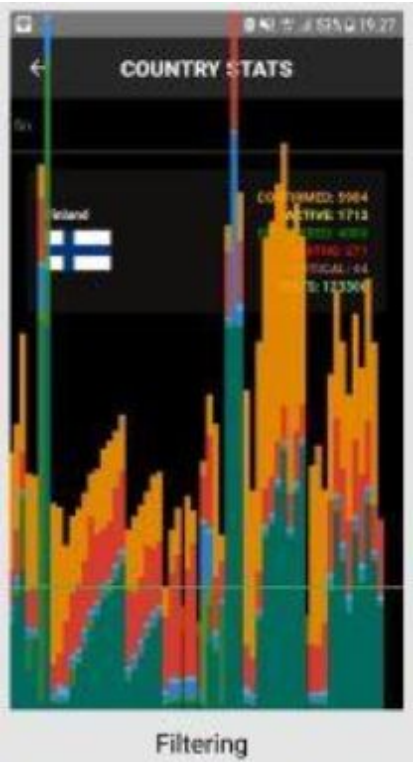


4.2.3. GPU

Scrolling Cards


Opening webview

Rendering listview



Filtering

Câu hỏi: Tại sao cần khảo sát 4 chức năng kia?

Bốn chức năng Scrolling, Openning Webview, Rendering ListView, Filtering khá thường gặp trong các app

Cluster	Included features
c1	chat, share content, update status, call, group chat, invite friend, video call
c2	search flight, load estimation, weather forecast,filter templates, editors, frames and shapes, emoji picker, color picker, picture templates, story templates
c3	create ads, edite ads, insight ads, campaign ads, campaign stats, ad performance, update bids, notifications, alerts
c4	e-commerce, search product, checkout product, buy product, sell product
c5	send message, read message, label message, delete message, notification, calendar integration, instant chat, video call,voice call
c6	record audio, play audio, share, upload, follow, react to uploads, rate, recommend,discover, guide
c7	connect people, meet-up, share photo/video/memory, tag person/location, react to content, watch, buy, sell, follow, notification, create events, invite guests, update status, chat, emoji picker, decorate photo/video
c8	artificial intelligence, self structuring, self reflection, mental health, reports and charts, exercises,tracker

Scrolling, Opening Webview, Rendering ListView, Filtering khá thường gặp

	Scrolling	Opening WebView	Rendering ListView	Filtering
c1	0.25	0.25	0.25	0.25
c2	0.25	0	0.25	0.25
c3	0.25	0	0.25	0.25
c4	0.25	0	0.25	0.25
c5	0.25	0	0.25	0.25
c6	0.25	0.25	0.25	0.25
c7	0.25	0.25	0.25	0.25
c8	0.25	0.25	0	0

	Micro		Small		Medium		Large	
	Target Android	Target iOS	Target Android	Target iOS	Target Android	Target iOS	Target Android	Target iOS
c1	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (83.3%) / F(16.7%)	RN (62.5%) / F(37.5%)
c2	RN(64.3%) / F(35.7%)	RN(60%) / F(40%)	RN(75%) / F(25%)	RN(75%) / F(25%)	RN (66.7%) / F(33.3%)	RN (66.7%) / F(33.3%)	RN (16.7%) / F(83.3%)	RN (14.3%) / F(85.7%)
c3	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (16.7%) / F(83.3%)	RN (14.3%) / F(85.7%)
c4	RN(64.3%) / F(35.7%)	RN(60%) / F(40%)	RN(75%) / F(25%)	RN(75%) / F(25%)	RN (66.7%) / F(33.3%)	RN (66.7%) / F(33.3%)	RN (16.7%) / F(83.3%)	RN (14.3%) / F(85.7%)
c5	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (16.7%) / F(83.3%)	RN (14.3%) / F(85.7%)
c6	RN(75%) / F(25%)	RN(90%) / F(10%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (50%) / F(50%)	RN (25%) / F(75%)
c7	RN (100%)	RN(100%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (83.3%) / F(16.7%)	RN (62.5%) / F(37.5%)
c8	RN(80%) / F(20%)	RN(88.9%) / F(11.1%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	RN (100%)	F(100%)

Mục lục

5. Tóm lược

- So sánh hiệu năng trên nền tảng iOS:
  - CPU: Ứng dụng React Native luôn sử dụng nhiều CPU hơn ứng dụng Flutter. Lý do là React Native sử dụng JavaScript bridge để giao tiếp với các native module và giao tiếp này sử dụng nhiều CPU. Trong khi Flutter không yêu cầu bridge để giao tiếp.
  - Sử dụng bộ nhớ: Ứng dụng React Native quản lý bộ nhớ tốt hơn ứng dụng Flutter. Một nguyên nhân tạo ra kết quả này có thể là React Native được triển khai với Redux trong khi Flutter sử dụng một gói provider đơn giản.
  - GPU: Nhìn chung, Flutter hoạt động tốt hơn một chút so với React Native.

5. Tóm lược

- So sánh hiệu năng trên nền tảng Android:
  - CPU: Cả Flutter và React Native đều quản lý để sử dụng CPU hiệu quả như nhau, kết quả cụ thể tùy thuộc các bài test.
  - Sử dụng bộ nhớ: Nhìn chung có thể nói cả hai framework quản lý bộ nhớ hiệu quả như nhau.
  - GPU: Công cụ "GPU rendering tool" trên Android chưa tương thích với ứng dụng Flutter nên so sánh này chưa được thực hiện.

5. Tóm lược

iOS Platform

	CPU	Memory	GPU
Scrolling cards	Flutter	React Native	Flutter
Opening webview	Flutter	React Native	Flutter
Rendering listview	Flutter	React Native	React Native
Filtering	Flutter	React Native	Flutter

Android Platform

	CPU	Memory	GPU
Scrolling cards	React Native	React Native	NA
Opening webview	React Native	Flutter	NA
Rendering listview	Flutter	React Native	NA
Filtering	Flutter	Flutter	NA

Kết quả so sánh hiệu năng trên từng nền tảng

Mục lục

6. Kết luận

- Một trong những lợi thế chính của React Native là cộng đồng nhà phát triển mạnh mẽ, cung cấp nhiều giải pháp thay thế cho các trường hợp sử dụng khác nhau.
- Vì React Native sử dụng JavaScript nên nó dễ tiếp cận hơn với những nhà phát triển web.
- Mặt trái là việc React Native sử dụng các cầu nối JavaScript để giao tiếp với các native module có ảnh hưởng đến hiệu suất.
- Hơn nữa, React Native yêu cầu styling cho nền tảng cụ thể để làm cho giao diện người dùng hoạt động liền mạch trên một số nền tảng nhất định.

6. Kết luận

- Với các công ty có đủ ngân sách và nguồn lực, Flutter có thể là một lựa chọn tuyệt vời.
- Những người tạo Flutter nói rằng họ lấy cảm hứng từ React Native, việc học Flutter có thể mất nhiều thời gian hơn đối với những nhà phát triển không quen với ngôn ngữ lập trình Dart.
- Mặc dù việc học khó khăn, Flutter có nhiều UI widget hoạt động liền mạch trên các nền tảng khác nhau.

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!

soict.hust.edu.vn/ fb.com/groups/soict