

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ  
Khoa Công nghệ thông tin

---



**BÁO CÁO BÀI TẬP LỚN**

**MÔN KIỂM THỬ VÀ ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM**

**Mã lớp: INT3117\_1**

**ĐỀ TÀI: CÔNG CỤ KIỂM THỬ  
ỨNG DỤNG DI ĐỘNG APPIUM**

**Giảng viên:** ThS.Nguyễn Thị Thu Trang

**Nhóm thực hiện:** Nhóm 13

**Thành viên:**

Vũ Trọng Đạt (Nhóm trưởng):	18020293
Nguyễn Thanh Huyền:	19020048
Đồng Vũ Hạnh Thảo	19020053

HÀ NỘI – 2021

# MỤC LỤC

<b>Danh mục hình ảnh</b>	<b>3</b>
<b>Danh mục bảng biểu</b>	<b>4</b>
<b>I. Tổng quan công cụ Appium</b>	<b>5</b>
1. Giới thiệu	5
2. Triết lý Appium	6
3. Kiến trúc và cách thức hoạt động	7
3.1 Kiến trúc hoạt động chung	7
3.2 Kiến trúc hoạt động với Android	8
3.3 Kiến trúc hoạt động với iOS	9
<b>II. Hướng dẫn sử dụng</b>	<b>9</b>
1. Cài đặt Appium	9
1.1 Công cụ yêu cầu:	9
1.2 Cài đặt cấu hình	10
1.2.1 Appium Desktop	10
1.2.2 Appium Inspector	11
1.2.3 WebDriverIo	13
2. Các thao tác cơ bản khi sử dụng Appium với WebdriverIO	13
2.1 Khởi tạo project NodeJs	13
2.2 Cấu hình WebdriverIo	14
2.3 Cấu hình kết nối thiết bị và server:	16
2.4 Viết phương thức kiểm thử tự động	17
3. Tổng quát dự án	17
4. Kiểm thử dự án	17
4.1 Danh sách test case	17
4.2 Phân tích AsscessbilityId và giao diện ứng dụng tương ứng.	20
4.2.1 Giao diện trang đăng nhập và đăng ký	20
4.2.2 Các AsscessbilityId sử dụng trong giao diện	20
4.3 Kiểm thử tự động demo	21
<b>III. So sánh với các công cụ kiểm thử khác</b>	<b>23</b>
1. Tổng quan	23
2. Ưu, nhược điểm	25
<b>IV. Kết luận</b>	<b>28</b>
<b>V. Tài liệu tham khảo</b>	<b>29</b>

## Danh mục hình ảnh

Hình 1: Appium	5
Hình 2: Kiến trúc hệ thống sử dụng Appium	7
Hình 3: Kiến trúc hoạt động của Appium trên Android	8
Hình 4: Kiến trúc hoạt động của Appium trên iOS	4
Hình 5: Giao diện chính Appium Desktop	10
Hình 6: Giao diện Appium Desktop khi khởi chạy server	11
Hình 7: Giao diện cài đặt Appium Inspector	12
Hình 8: Giao diện tương tác của Appium Inspector	13
Hình 9: Giao diện tạo mới project Nodejs	14
Hình 10: Giao diện tạo mới project Nodejs	15
Hình 11: Package.json sau khi đã cài đặt đủ các thư viện cần thiết	15
Hình 12: Cấu hình kết nối thiết bị, Appium và WebdriverIO	16
Hình 13: Code mẫu ca kiểm thử	17
Hình 14: Giao diện đăng ký, đăng nhập của ứng dụng	20
Hình 15: Code ca kiểm thử	22

## **Danh mục bảng biểu**

Bảng 1: Một số test case trong chương trình	19
Bảng 2: Đối chiếu AsscessbilityId và thành phần giao diện	21
Bảng 3: So sánh Appium, Selenium và Katalon Studio	25
Bảng 4: So sánh ưu, nhược điểm Appium, Selenium và Katalon Studio	27

## I. Tổng quan công cụ Appium

### 1. Giới thiệu



*Hình 1: Appium*

- + Hiện nay có rất nhiều tool được sử dụng cho testing automation các native app, mobile web app, và hybrid app trên nền tảng iOS và Android. Đặc biệt hơn Appium là 1 cross-platform tool, cho phép người dùng có thể kiểm thử trên nhiều nền tảng, sử dụng chung API, và 1 source code lại được dùng cho nhiều nền tảng.
- + Appium cho phép kiểm thử các ứng dụng di động dạng native app, hybrid app và cả cross-platform app trên các thiết bị vật lý thực (physical device) và cả thiết bị ảo (simulator/emulator):
  - Ứng dụng Native là những ứng dụng được viết riêng cho một loại nền tảng như iOS, Android, Windows Phone bằng các ngôn ngữ tương ứng của mỗi nền tảng đó ví dụ Java trên Android, Object C trên iOS, C# trên winphone. Mỗi Native App chỉ chạy được trên một nền tảng và không thể mang sang các nền tảng khác.
  - Ứng dụng hybrid app là ứng dụng có các thành phần phần cơ bản được viết bằng ngôn ngữ web, nhưng được đặt trong native container, nên vẫn có thể đưa lên các App Store, cho phép người dùng có thể tương tác với nội dung web thông qua app.
  - Ứng dụng cross-platform là ứng dụng chỉ cần một bản code, có thể phiên dịch và sử dụng chúng trên rất nhiều nền tảng khác nhau, giúp tối ưu chi phí phát triển và hiệu năng của hệ thống.
- + Đặc biệt, Appium là "**đa nền tảng**": vì Appium được phát triển từ nền tảng hệ thống Selenium (kế thừa các đối tượng, cấu trúc và cú pháp) nên nó cho phép người dùng viết các test script trên nhiều nền tảng khác nhau (iOS, Android),

sử dụng cùng một API. Điều này cho phép bạn tái sử dụng mã giữa các nền tảng iOS và Android

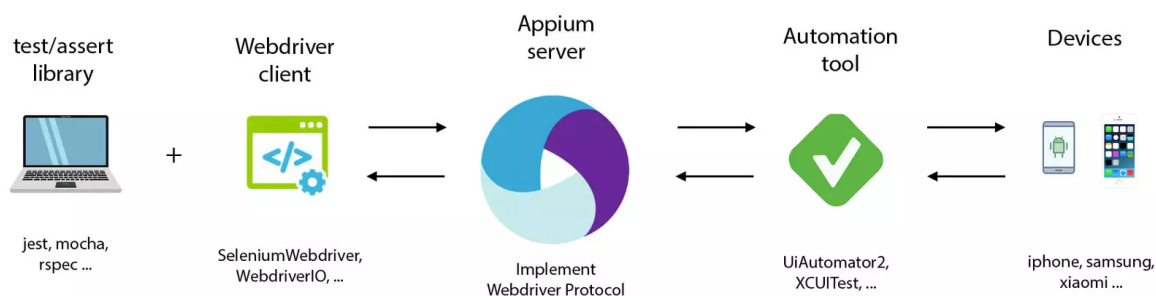
- + Khả năng cross-platforms của Appium đến từ việc sử dụng các thư viện của Appium, thông qua một chương trình server, chuyển các câu lệnh sử dụng trong mã kiểm thử thành các lệnh XCUITest (với iOS) hay UIAutomator (với Android) để tương tác với thiết bị.
- + Appium hỗ trợ kiểm thử tự động phần mềm trên cả emulator, simulator và thiết bị thật.
- + Appium hỗ trợ viết test cho rất nhiều ngôn ngữ như Java, C#, PHP, Python, JavaScript, NodeJS, Perl, Ruby....
- + Trong báo cáo này sẽ tập trung ví dụ Appium với phần mềm Android.

## **2. Triết lý Appium**

- Người dùng không cần phải biên dịch lại ứng dụng của mình hoặc sửa đổi ứng dụng theo bất kỳ cách nào để tiến hành kiểm thử. Appium không yêu cầu bất kỳ tác nhân bổ sung nào trong mã ứng dụng. Do đó, chúng ta không cần phải biên dịch lại hoặc sửa đổi ứng dụng của mình.
- Người dùng không nên bị khóa vào một ngôn ngữ hoặc khuôn khổ cụ thể để viết và chạy chương trình kiểm thử. Appium cho phép viết các bài kiểm tra bằng Ruby, Python, Java, Node.js, Objective-C, PHP và C #,... Người dùng không cần phải học bất kỳ ngôn ngữ / khuôn khổ mới nào để sử dụng nó.
- Một công cụ kiểm thử tự động không nên thực hiện các hành động không cần thiết khi gọi đến các API tự động. Appium sử dụng API WebDriver trên nền tảng NodeJs, vốn đã khá nổi tiếng trong số những người kiểm tra tự động hóa.
- Một công cụ kiểm thử phải là nguồn mở. Appium có sẵn miễn phí và người dùng có thể tự do sửa đổi nó theo yêu cầu của họ. Thực sự là mã nguồn mở trong tinh thần và trong thực tế.

### 3. Kiến trúc và cách thức hoạt động

#### 3.1. Kiến trúc hoạt động chung



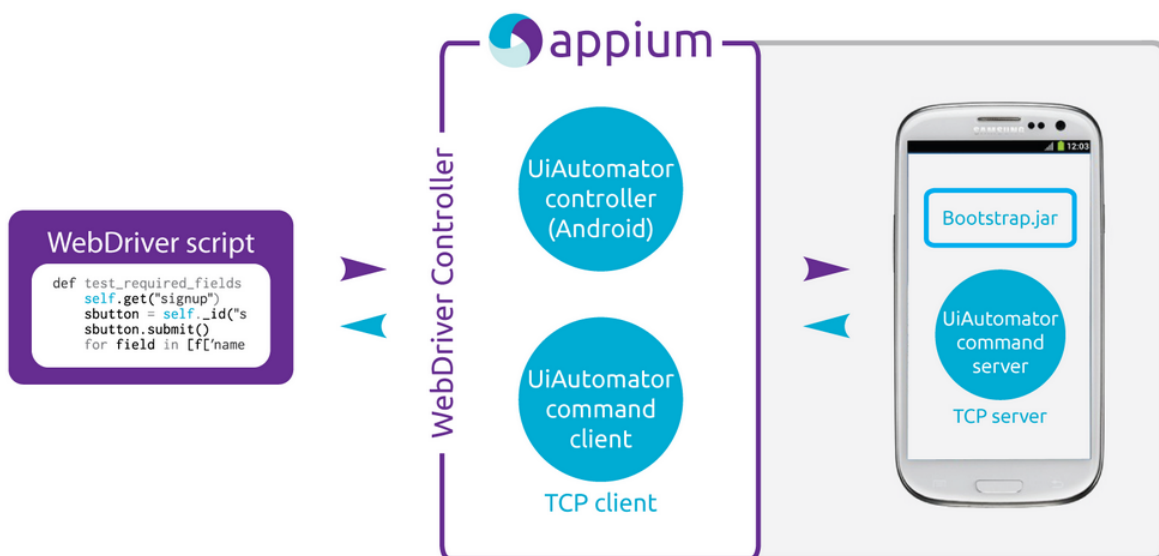
Hình 2: Kiến trúc hệ thống sử dụng Appium

- Appium là một máy chủ HTTP được viết bằng Node.JS tạo ra và xử lý nhiều phiên WebDriver cho các nền tảng khác nhau như iOS và Android. Do đó, trước khi khởi tạo máy chủ Appium, Node.js phải được cài đặt sẵn trên hệ thống.
- Chương trình máy chủ Appium – Appium Server: là một chương trình tạo lập một máy chủ Java, dùng để chuyển các lệnh trong mã kiểm thử thành các lệnh có thể tương tác với thiết bị: XCUITest với iOS hay UIAutomator với Android.
- Hệ thống thư viện Appium: Appium có một hệ thống thư viện dùng để nhận diện và tương tác với các đối tượng UI trên ứng dụng di động. Hệ thống thư viện Appium được cung cấp cho nhiều ngôn ngữ lập trình khác nhau như C#, Java, Python... để kỹ sư kiểm thử tự động có thể chọn ngôn ngữ lập trình quen thuộc cho việc tự động hóa kiểm thử.
- Appium bắt đầu một ca kiểm thử trên thiết bị sinh ra máy chủ và lắng nghe lệnh proxy từ máy chủ Appium chính. Nó gần giống như Selenium server, nhận thức các yêu cầu HTTP từ thư viện selenium client và nó xử lý các yêu cầu theo những cách khác nhau tùy thuộc vào nền tảng.
- Khi máy chủ Appium khởi động, về cơ bản nó sẽ hiển thị API REST cho máy khách. Khách hàng (devices) bắt đầu một phiên với nó bằng cách sử dụng đối tượng JSON 'khả năng mong muốn'. Đến lượt nó, máy chủ bắt đầu một phiên

và trả lời máy khách bằng một ID phiên. Kết quả này là việc tạo ra một phiên giữa máy khách và máy chủ.

- Bây giờ phiên được thiết lập giữa chúng, máy khách sẽ gửi các lệnh tự động hóa đến máy chủ. Máy chủ Appium thực thi các lệnh đó trên thiết bị di động và trả lời ứng dụng khách bằng các kết quả.
- Tại thời điểm này, điều đáng chú ý là Appium không thực hiện các lệnh trên thiết bị di động một cách trực tiếp. Thay vào đó, nó gọi các khuôn khổ tự động hóa (automation tool) do nhà cung cấp cung cấp cụ thể cho nền tảng mà chúng tôi đang thử nghiệm. Các nhà cung cấp như Apple, Google và Microsoft cung cấp các khuôn khổ tự động hóa cho nền tảng của họ. Máy chủ Appium kết nối với các khuôn khổ này và thực hiện tự động hóa trên thiết bị.

### 3.2. Kiến trúc hoạt động với Android



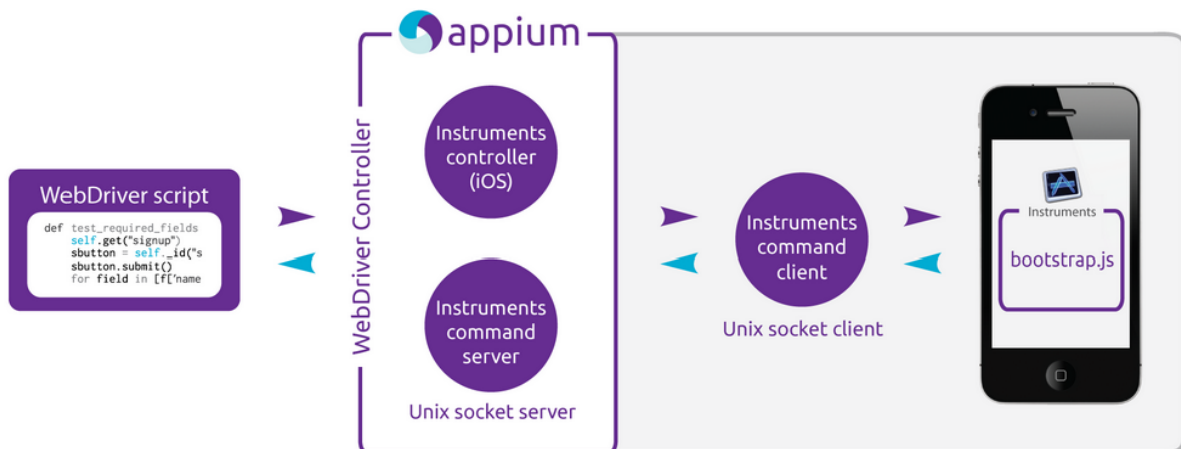
Hình 3: Kiến trúc hoạt động của Appium trên Android

- Trên Android, Appium ủy quyền tập lệnh cho UIAutomator. UIAutomator là framework của Android hỗ trợ chạy các test case trực tiếp bằng Junit trên thiết bị từ dòng lệnh. Nó sử dụng ngôn ngữ lập trình Java, nhưng Appium sẽ làm cho nó chạy từ bất kỳ ngôn ngữ nào được hỗ trợ WebDriver.
- Khi thực thi các tập lệnh, sẽ có một yêu cầu HTTP ở định dạng JSON đến máy chủ Appium. Sau đó, máy chủ Appium gửi lệnh tới UIAutomator cần bootstrap.jar. Nó hoạt động như máy chủ TCP để nghe lệnh của từ người kiểm



thử. Và sau đó nó thực thi lệnh trên các thiết bị Android và trả về kết quả. Chu kỳ này tiếp tục cho đến khi tất cả các lệnh được thực thi.

### 3.3. Kiến trúc hoạt động với iOS



Hình 4: Kiến trúc hoạt động của Appium trên iOS

- Trên iOS, Appium ủy quyền tập lệnh cho XCTest chạy trong môi trường MacOS. Apple cung cấp ứng dụng này có tên là Instruments, được sử dụng để thực hiện nhiều hoạt động như thiết lập hồ sơ, điều khiển và build ứng dụng iOS. Nhưng nó cũng có một thành phần tự động hóa để chúng ta có thể viết một số lệnh trong javascript sử dụng api XCTest để tương tác với giao diện ứng dụng. Appium sử dụng các thư viện tương tự để tự động hóa ứng dụng iOS.
- Khi chúng ta thực thi các tập lệnh, sẽ có một yêu cầu HTTP ở định dạng JSON đến máy chủ Appium. Sau đó, máy chủ Appium gửi lệnh tới Instruments được viết bằng NodeJs, thực thi lệnh trong bootstrap.js trong môi trường instruments iOS. Khi lệnh được thực thi, máy khách lệnh sẽ gửi lại thông điệp đến máy chủ Appium để ghi lại mọi thứ liên quan đến lệnh trong console của nó. Chu kỳ này tiếp tục cho đến khi tất cả các lệnh được thực thi.

## II. Hướng dẫn sử dụng

### 1. Cài đặt Appium

#### 1.1. Công cụ yêu cầu:

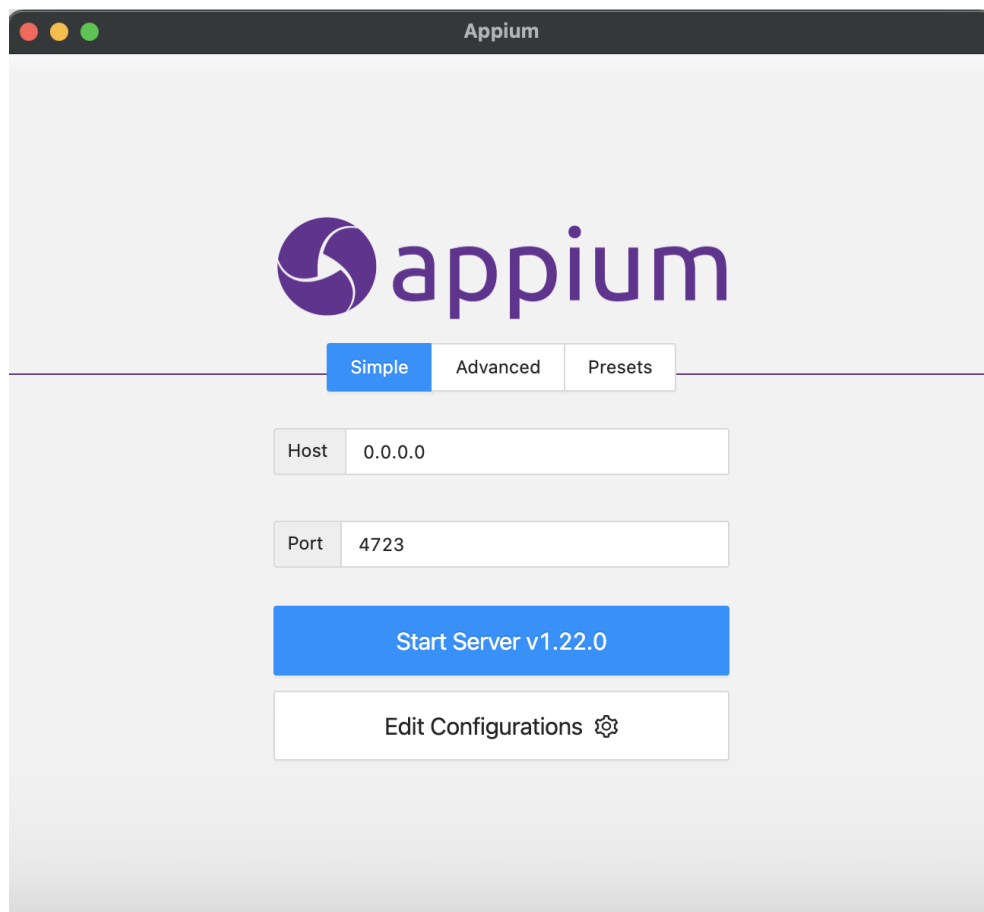
- + NodeJS: <https://nodejs.org/en/>
- + Appium Desktop: <https://appium.io/>
- + Appium Inspector: <https://github.com/appium/appium-inspector/releases>

- + Đối với Android:
  - JavaSDK: <https://www.oracle.com/java/technologies/downloads/>
  - Android Studio: <https://developer.android.com/studio>
- + Đối với IOS:
  - XCode: <https://developer.apple.com/xcode/resources/>
  - WebDriverAgent: <https://github.com/appium/WebDriverAgent>

## 1.2. Cài đặt cấu hình

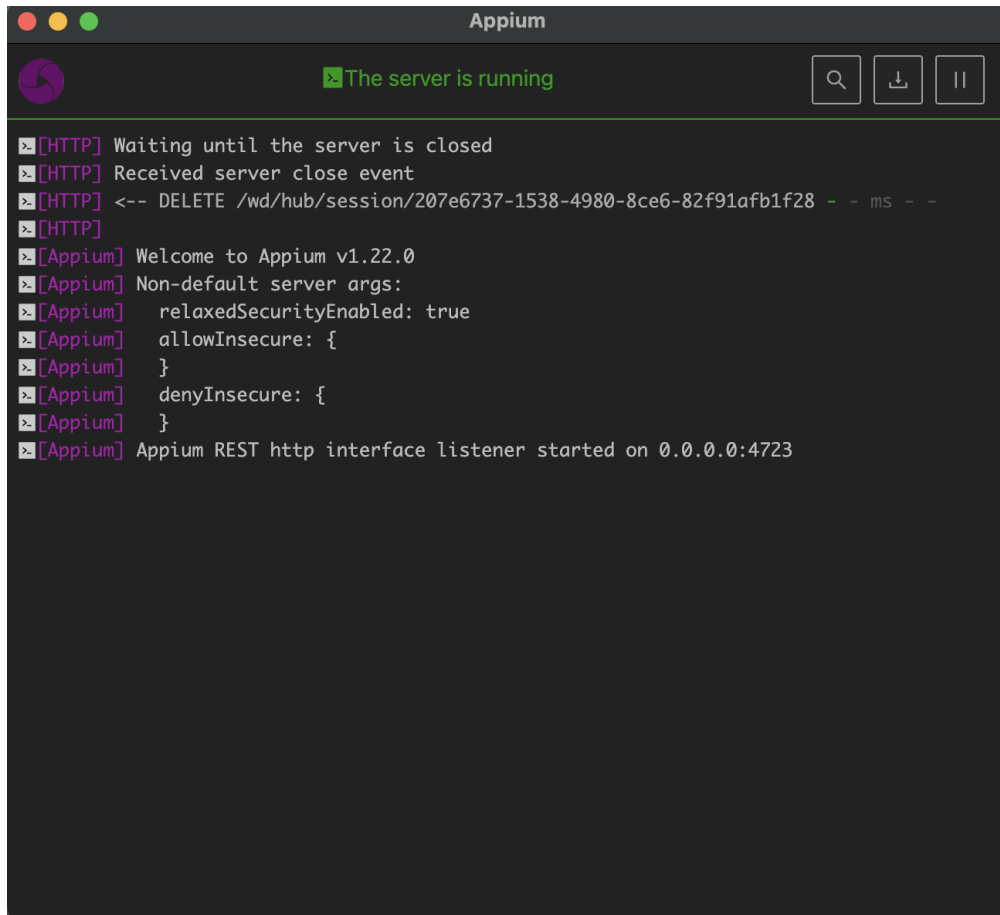
### 1.2.1. Appium Desktop

- + Sau khi cài đặt thành công, click chuột vào phần mềm Appium.
- + Khi Server Appium khởi động, nó sẽ hiển thị host và port mặc định và người dùng có thể tự thay đổi thông số này.



*Hình 5: Giao diện chính Appium desktop*

- + Khi click chuột vào button Start Server, một máy chủ mới sẽ khởi động ở địa chỉ host và port đã cài đặt. Server đầu ra được hiển thị.

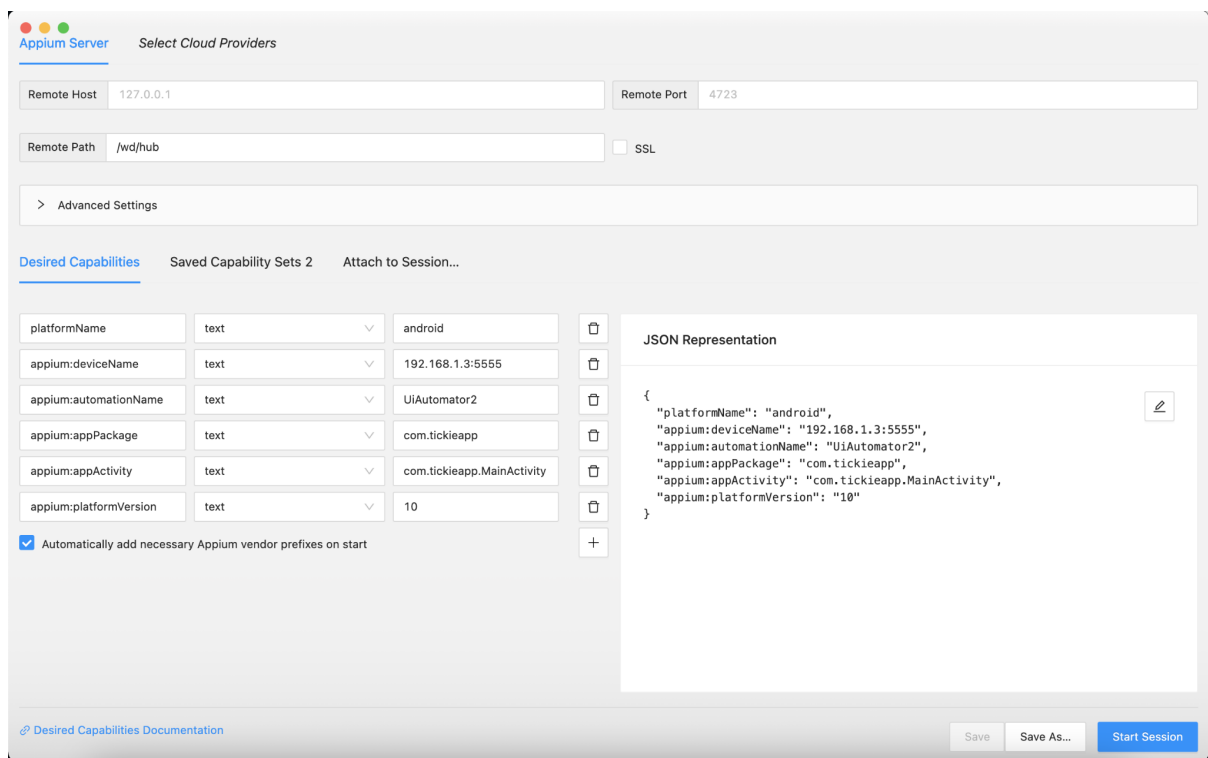


*Hình 6: Giao diện Appium Desktop khi khởi chạy server*

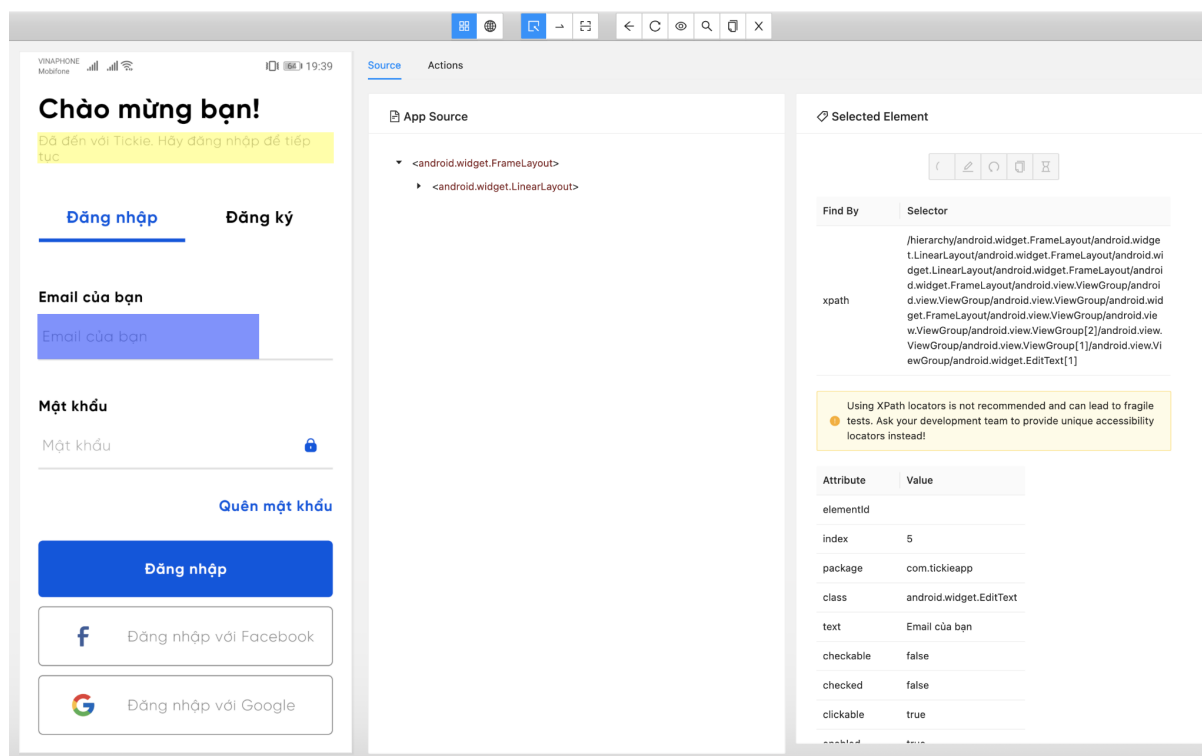
### 1.2.2. Appium Inspector

- + Tổng quan:
  - Appium Inspector là một Appium client với giao diện đồ họa, với chức năng tương tự như WebdriverIO, Appium's Java client, Appium's Python,...
  - Appium Inspector có thể chỉ định máy chủ Appium nào nên sử dụng, các thiết lập để kết nối giữa Appium server và ứng dụng kiểm thử, sau đó tương tác với các phần tử giao diện trong ứng dụng.
  - Appium Inspector hỗ trợ chế độ record, dịch hành động kiểm thử trên giao diện thành tập lệnh để tiến hành kiểm thử tự động
  - Tuy nhiên Appium Inspector không cung cấp phương thức để kiểm thử tự động mà người dùng phải tự thực hiện tất cả bằng tay.

- + Cấu hình: Mục tiêu để khi khởi chạy phiên làm việc, Appium Inspector có thể nhận biết được thiết bị đang dùng để kiểm thử cũng như ứng dụng cần kiểm thử trên thiết bị đó.
  - platformName: android/ios.
  - deviceName: Tên định danh thiết bị đối với máy tính đang kết nối.
  - automationName: Tên framework đối với từng nền tảng. UiAutomator2 dùng cho ứng dụng Android và XCUITest dùng cho ứng dụng iOS.
  - appPackage: Tên định danh ứng dụng.
  - appActivity: Tên định danh module main của ứng dụng.
  - app: Đường dẫn tệp cài đặt .apk hoặc .ipa trên máy tính đang kết nối với thiết bị kiểm thử. Người dùng có thể chỉ định tệp cài đặt cho ứng dụng mà ko cần khai báo appPackage và appActivity.



Hình 7: Giao diện cài đặt Appium Inspector



Hình 8: Giao diện tương tác của Appium Inspector

### 1.2.3. WebDriverIo

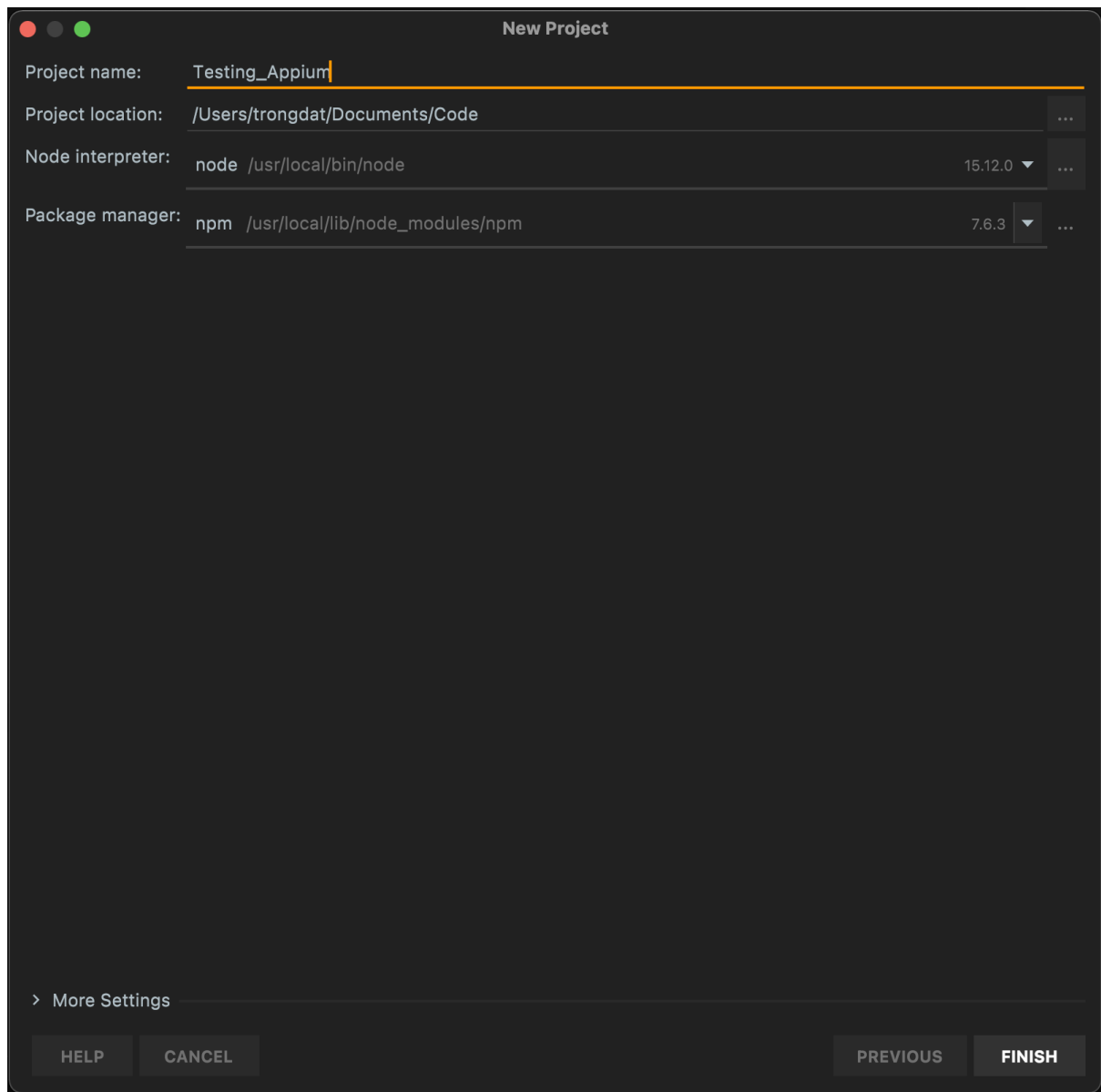
- + WebDriverIO là một tiện ích kiểm thử mã nguồn mở rất mạnh mẽ sử dụng ngôn ngữ NodeJs. Nó giúp người kiểm thử viết các tiến trình kiểm thử bằng Javascript rất nhanh chóng và thuận tiện, giúp có thể thực thi và duy trì các pha kiểm thử này bằng Javascript mà không cần chuyển đổi ngôn ngữ giữa Kotlin và Swift.
- + Về cơ bản, nó sẽ gửi các yêu cầu đến máy chủ Appium thông qua Giao thức WebDriver và xử lý phản hồi của nó.
- + WebdriverIO có thể xây dựng các pha kiểm thử tự động hóa các ứng dụng Android và iOS tương tự như các tương tác của người dùng thông thường mà người dùng không phải tương tác trực tiếp với thiết bị.

## 2. Các thao tác cơ bản khi sử dụng Appium với WebdriverIO

### 2.1. Khởi tạo project NodeJs

- Đối với IntelliJ, chọn File / New / Project.
- Chọn Javascript / Node.js.
- Điền Project name và các tùy chỉnh khác nếu cần và chọn Finish.

- Hoặc có thể sử dụng câu lệnh: `npm init -y` để khởi tạo project qua Terminal.



*Hình 9: Giao diện tạo mới project Nodejs*

## 2.2. Cấu hình WebdriverIo

Sử dụng các câu lệnh sau trong Terminal:

- Cài đặt WebdriverIo CLI: `npm install @wdio/cli`
- Cấu hình Cấu hình WebdriverIo: `npx wdio config`

```
Terminal: Local x + ↵
=====
WDIO Configuration Helper
=====

? Where is your automation backend located? On my local machine
? Which framework do you want to use? mocha
? Do you want to use a compiler? Babel (https://babeljs.io/)
? Where are your test specs located? ./test/specs/**/*.js
? Do you want WebdriverIO to autogenerate some test files? Yes
? Do you want to use page objects (https://martinfowler.com/bliki/PageObject.html)? Yes
? Where are your page objects located? ./test/pageobjects/**/*.js
? Which reporter do you want to use? allure
? Do you want to add a service to your test setup? appium
? What is the base url? http://localhost
```

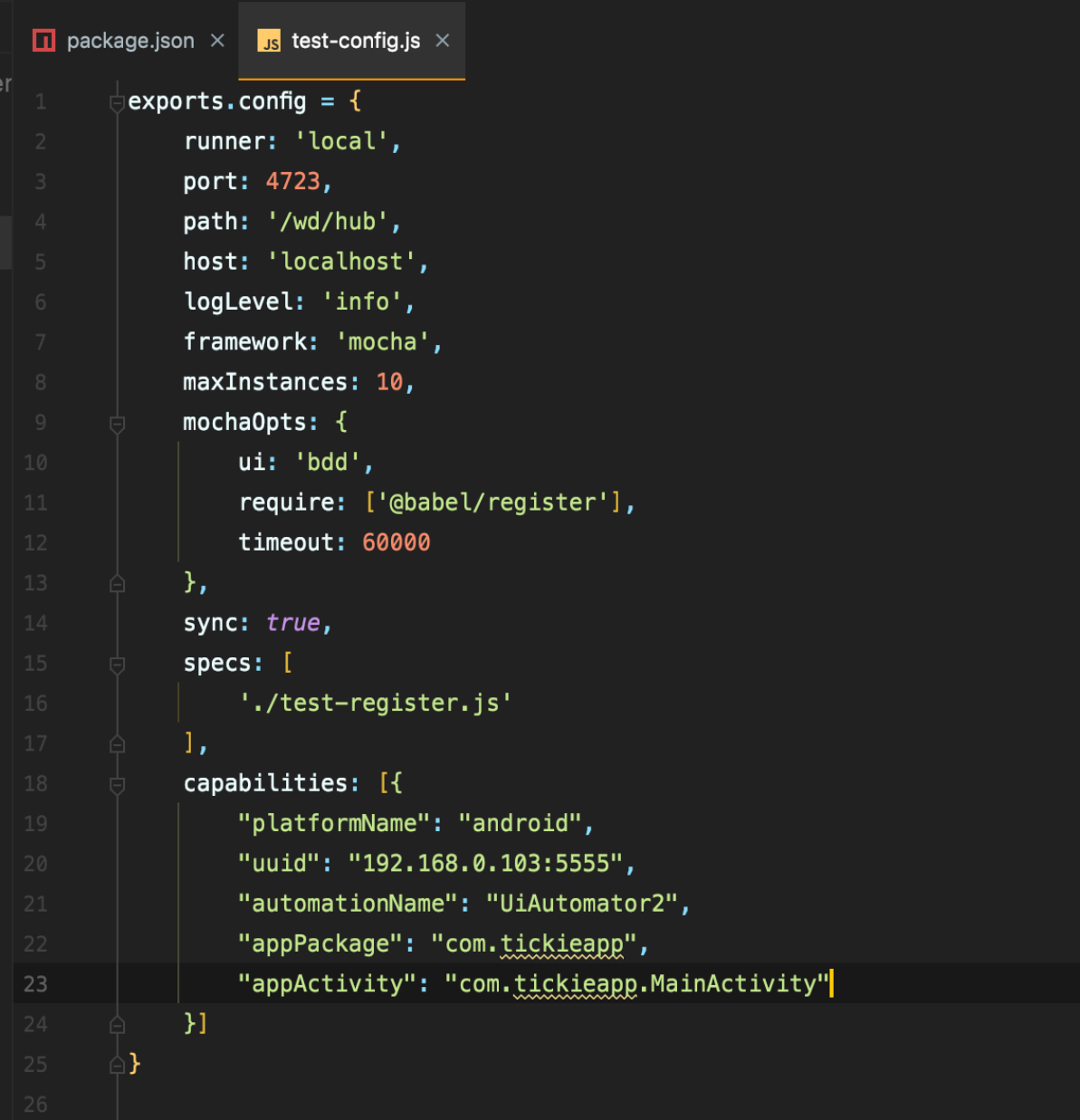
Hình 10: Giao diện cấu hình WebdriverIO

```
Project v [icons] - package.json x
Mobile_Auto_Test_Appium ~/Documents
├── .idea
├── node_modules library root
├── test
│   ├── pageobjects
│   └── specs
│       └── example.e2e.js
├── babel.config.js
├── package.json
├── package-lock.json
├── PlayWithMocha.js
├── test-config.js
├── test-register.js
├── wdio.conf.js
└── External Libraries
> Scratches and Consoles

1 {
2   "name": "Mobile_Auto_Test_Appium",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "wdio"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "@wdio/cli": "^7.16.4",
14    "@wdio/sync": "^7.16.4",
15    "webdriverio": "^7.16.4"
16  },
17  "devDependencies": {
18    "@babel/core": "^7.16.0",
19    "@babel/preset-env": "^7.16.0",
20    "@babel/register": "^7.16.0",
21    "@wdio/allure-reporter": "^7.16.3",
22    "@wdio/appium-service": "^7.16.4",
23    "@wdio/local-runner": "^7.16.4",
24    "@wdio/mocha-framework": "^7.16.4",
25    "@wdio/spec-reporter": "^7.16.4"
26  }
27 }
```

Hình 11: Package.json sau khi đã cài đặt đủ các thư viện cần thiết

### 2.3. Cấu hình kết nối thiết bị và server:

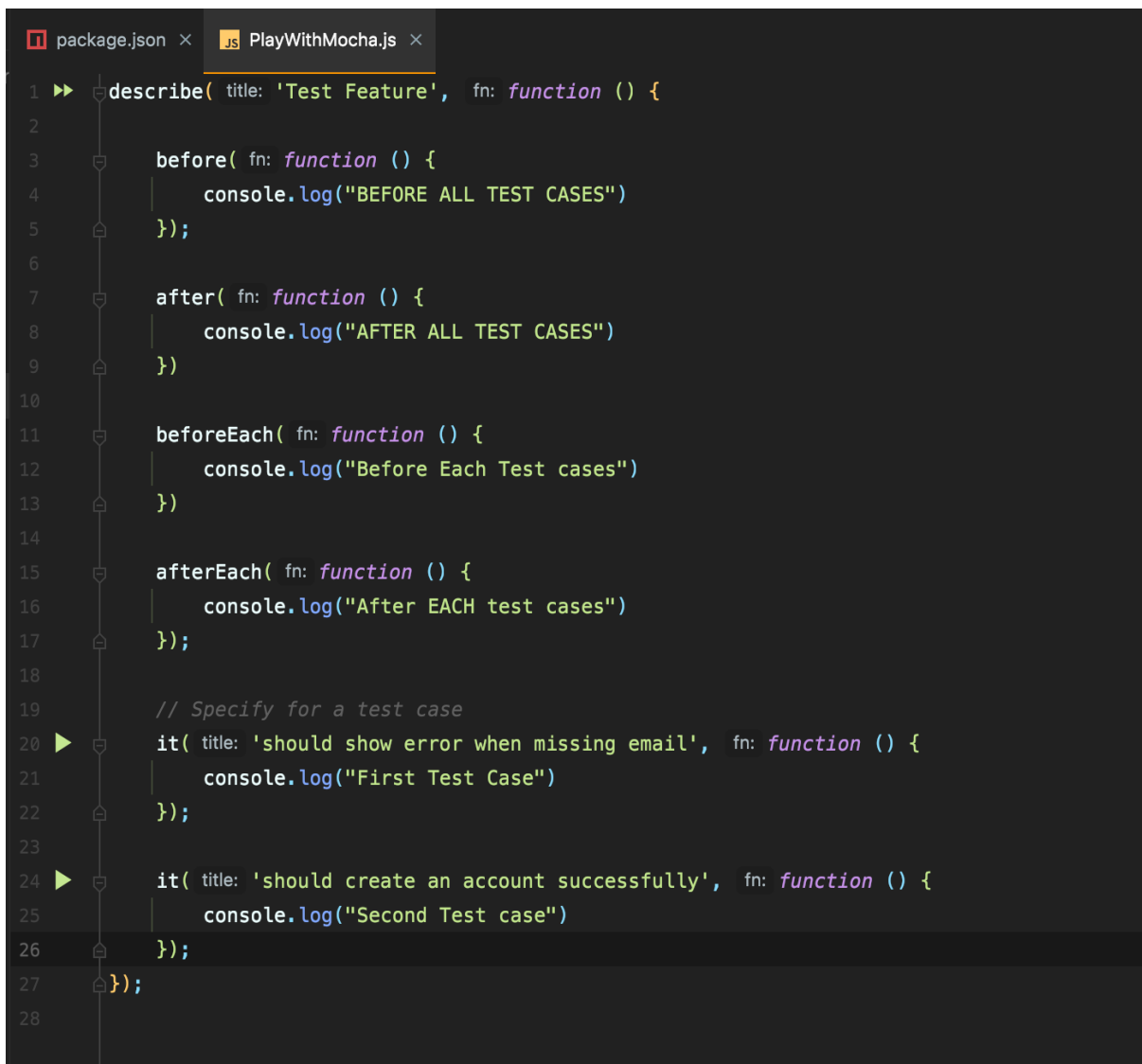


```
1 exports.config = {
2   runner: 'local',
3   port: 4723,
4   path: '/wd/hub',
5   host: 'localhost',
6   logLevel: 'info',
7   framework: 'mocha',
8   maxInstances: 10,
9   mochaOpts: {
10     ui: 'bdd',
11     require: ['@babel/register'],
12     timeout: 60000
13   },
14   sync: true,
15   specs: [
16     './test-register.js'
17   ],
18   capabilities: [{
19     "platformName": "android",
20     "uuid": "192.168.0.103:5555",
21     "automationName": "UiAutomator2",
22     "appPackage": "com.tickieapp",
23     "appActivity": "com.tickieapp.MainActivity"
24   }]
25 }
```

Hình 12: Cấu hình kết nối thiết bị, Appium và WebDriverIO



## 2.4. Viết phương thức kiểm thử tự động



```
1  describe( title: 'Test Feature', fn: function () {
2
3      before( fn: function () {
4          console.log("BEFORE ALL TEST CASES")
5      });
6
7      after( fn: function () {
8          console.log("AFTER ALL TEST CASES")
9      })
10
11     beforeEach( fn: function () {
12         console.log("Before Each Test cases")
13     })
14
15     afterEach( fn: function () {
16         console.log("After EACH test cases")
17     });
18
19     // Specify for a test case
20     it( title: 'should show error when missing email', fn: function () {
21         console.log("First Test Case")
22     });
23
24     it( title: 'should create an account successfully', fn: function () {
25         console.log("Second Test case")
26     });
27 });
28
```

Hình 13: Code mẫu ca kiểm thử

## 3. Tổng quát dự án

- + Mã nguồn chương trình: <https://github.com/dattv155/TickieApp>
- + Nội dung chương trình: Ứng dụng đặt vé xem phim trực tuyến.
- + Các tính năng chính của ứng dụng: Đăng ký / đăng nhập, xem thông tin phim, đặt mua vé xem phim, nhận thông báo, quản lý thông tin người dùng...

## 4. Kiểm thử dự án

### 4.1. Danh sách test case

Test case	Mô tả	Các bước	Expected	Actual
-----------	-------	----------	----------	--------

Id			Output	Output
Register _01	Nhập đủ thông tin yêu cầu, không bị lỗi	<ol style="list-style-type: none"> <li>1. Mở ứng dụng.</li> <li>2. Vào màn hình Đăng nhập.</li> <li>3. Chọn tab Đăng ký</li> <li>4. Điền email chưa đăng ký.</li> <li>5. Điền password.</li> <li>6. Điền confirm password trùng với password đã nhập.</li> <li>7. Chọn Đăng ký</li> </ol>	Hiện thông báo đăng ký thành công, chuyển về màn hình đăng nhập.	
Register _02	Bỏ trống các trường bắt buộc: email, mật khẩu, nhập lại mật khẩu	<ol style="list-style-type: none"> <li>1. Mở ứng dụng.</li> <li>2. Vào màn hình Đăng nhập.</li> <li>3. Chọn tab Đăng ký</li> <li>4. Chỉ điền email.</li> <li>5. Ấn Đăng ký</li> </ol>	Hiện thông báo thiếu trường bắt buộc, không cho chuyển màn hình	
Register _03	Nhập confirm password không giống password	<ol style="list-style-type: none"> <li>1. Mở ứng dụng.</li> <li>2. Vào màn hình Đăng nhập.</li> <li>3. Chọn tab Đăng ký</li> <li>4. Điền email.</li> <li>5. Điền password.</li> <li>6. Điền confirm password khác với password đã nhập.</li> <li>7. Ấn Đăng ký</li> </ol>	Hiện thông báo nhập lại mật khẩu không khớp, không cho chuyển màn hình	

Register_04	Nhập email đã đăng ký	<ol style="list-style-type: none"> <li>1. Mở ứng dụng.</li> <li>2. Vào màn hình Đăng nhập.</li> <li>3. Chọn tab Đăng ký</li> <li>4. Điền email đã đăng ký trước đây.</li> <li>5. Điền password.</li> <li>6. Điền confirm password khác với password đã nhập.</li> <li>7. Ấn Đăng ký</li> </ol>	Hiện thông báo email đã được sử dụng.	
Login_01	Bỏ trống các trường bắt buộc: email, password	<ol style="list-style-type: none"> <li>1. Mở ứng dụng.</li> <li>2. Vào màn hình Đăng nhập.</li> <li>3. Chỉ điền email.</li> <li>4. Ấn Đăng nhập</li> </ol>	Hiện thông báo thiếu trường password, không cho chuyển màn hình	
Login_02	Nhập sai thông tin đăng nhập	<ol style="list-style-type: none"> <li>1. Mở ứng dụng.</li> <li>2. Vào màn hình Đăng nhập.</li> <li>3. Nhập email đã đăng ký.</li> <li>4. Nhập password khác với đã đăng ký</li> </ol>	Hiện thông báo đăng nhập thất bại, sai password, không cho chuyển màn hình	

*Bảng 1: Một số test case trong chương trình*

## 4.2. Phân tích AsscessbilityId và giao diện ứng dụng tương ứng.

### 4.2.1. Giao diện trang đăng nhập và đăng ký

## Chào mừng bạn!

Đã đến với Tickie. Hãy đăng nhập để tiếp tục

[Đăng nhập](#)[Đăng ký](#)

Email của bạn

Email của bạn

Mật khẩu

Mật khẩu

[Quên mật khẩu](#)

Đăng nhập

f Đăng nhập với Facebook

G Đăng nhập với Google

## Chào mừng bạn!

Đã đến với Tickie. Hãy đăng ký để tiếp tục

[Đăng nhập](#)[Đăng ký](#)

Email của bạn

Email của bạn

Mật khẩu

Mật khẩu

Nhập lại mật khẩu

Nhập lại mật khẩu

Đăng ký

f Đăng ký với Facebook

G Đăng ký với Google

Hình 14: Giao diện đăng ký, đăng nhập của ứng dụng

### 4.2.2. Các AsscessbilityId sử dụng trong giao diện

ST T	Mô tả	Thành phần giao diện	AsscessbilityId
1	Nút chọn tab Đăng nhập	<a href="#">Đăng nhập</a>	TabLogin

20

2	Nút chọn tab Đăng ký		TabSignup
3	Trường nhập email đăng nhập		LoginEmailInput
4	Trường nhập password đăng nhập		LoginPasswordInput
4	Trường nhập email đăng ký		SignupEmailInput
5	Trường nhập password đăng ký		SignupPasswordInput
6	Trường nhập confirm password đăng ký		SignupConfirmPasswordInput
7	Nút đăng nhập		LoginButton
8	Nút đăng ký		SignupButton

*Bảng 2: Bảng đối chiếu AccessibilityId và thành phần giao diện*

#### 4.3. Kiểm thử tự động demo

- + Trong demo này sẽ sử dụng Register\_03 để tiến hành kiểm thử tự động với Appium trên thiết bị Android.

```
package.json x JS test-register.js x
1 >> describe( title: 'Register Feature', fn: function () {
2 >> it( title: 'Register_03', fn: function () {
3     const TabSignup = '~TabSignup'
4     const SignupEmailInput = '~SignupEmailInput'
5     const SignupPasswordInput = '~SignupPasswordInput'
6     const SignupConfirmPasswordInput = '~SignupConfirmPasswordInput'
7     const SignupButton = '~SignupButton'
8     // 1. Click on Register tab
9     $(TabSignup).click()
10    // 2. Fill email text field
11    $(SignupEmailInput).setValue('datvt.hrt@gmail.com')
12    // 3. Fill password text field
13    $(SignupPasswordInput).setValue('12345678')
14    // 4. Fill confirm password text field
15    $(SignupConfirmPasswordInput).setValue('11111111')
16    // 5. Click on Register btn
17    $(SignupButton).click()
18    browser.pause( milliseconds: 2000)
19  });
20
21 >> it( title: 'Register_01', fn: function () {
22    const TabSignup = '~TabSignup'
23    const SignupEmailInput = '~SignupEmailInput'
24    const SignupPasswordInput = '~SignupPasswordInput'
25    const SignupConfirmPasswordInput = '~SignupConfirmPasswordInput'
26    const SignupButton = '~SignupButton'
27    // 1. Click on Register tab
28    $(TabSignup).click()
29    // 2. Fill email text field
30    $(SignupEmailInput).setValue('datvt.hrt@gmail.com')
31    // 3. Fill password text field
32    $(SignupPasswordInput).setValue('12345678')
33    // 4. Fill confirm password text field
34    $(SignupConfirmPasswordInput).setValue('12345678')
```

Hình 15: Code ca kiểm thử

+ Giải thích:

- Sử dụng phương thức describe để khai báo pha kiểm thử, có thể gồm một hoặc nhiều ca kiểm thử khác nhau.
- Mỗi phương thức it là một ca kiểm thử. Nó bao gồm title và một callback function để khai báo các hành động mà người dùng muốn thực hiện.

- Trong ca kiểm thử Register\_03, với các AccessibilityId được định nghĩa sẵn có thể tìm ra đúng thành phần giao diện cần trích xuất và gán cho một biến cụ thể.
- Hàm click(): tương ứng với thao tác bấm vào giao diện.
- Hàm setValue(): tương ứng với thao tác nhập dữ liệu vào form.
- Hàm browser.pause(2000): tạm dừng chương trình 2000ms trước khi kết thúc.

### III. So sánh với các công cụ kiểm thử khác

#### 1. Tổng quan

	Appium	Selenium	Katalon Studio
Mục tiêu sử dụng	Được sử dụng để tự động hóa các bài kiểm tra cho các ứng dụng web gốc, kết hợp (.ipa và .apk) và di động.	Được sử dụng để giúp kiểm tra ứng dụng web dễ dàng hơn bằng cách tự động hóa các hành động của trình duyệt bằng cách sử dụng WebDriver.	Được sử dụng để giúp kiểm tra ứng dụng web dễ dàng hơn bằng cách tự động hóa các hành động của trình duyệt bằng cách sử dụng WebDriver
Các dạng hỗ trợ kiểm thử	Mobile app (android, ios) và máy tính để bàn.	Web trên nhiều trình duyệt. Không thể tự động hóa các ứng dụng máy tính để bàn	Web, mobile, Api/ Web services
Nền tảng	Windows  iOS	Windows  MacOs	Windows  OS X

	Android	Linux Solaris	Linux
Ngôn ngữ kịch bản	Java, PHP, Perl, Python	Java,C#,Perl,Python, Javascript, Ruby, PHP	Java / groovy
Kỹ năng ngôn ngữ	Kỹ năng nâng cao cần thiết	Kỹ năng nâng cao cần thiết	Không yêu cầu, đề xuất cho các kịch bản nâng cao
Cài đặt, sử dụng	Dễ dàng cài đặt và sử dụng	Yêu cầu cài đặt và tích hợp các công cụ khác nhau	Dễ dàng cài đặt và sử dụng
Thời gian tạo kịch bản test	Chậm	Chậm	Nhanh do có Katalon recorder
Data-driven testing		Phải hard-code dữ liệu vào trong code	Hỗ trợ nhiều nguồn dữ liệu như CSV, Excel, database
Các keyword dựng sẵn	Phải tự xây dựng	Phải tự xây dựng	Được định nghĩa và tích hợp sẵn



Báo cáo ca kiểm thử	Không tích hợp	Không tích hợp	Cung cấp với giao diện trực quan và thân thiện
Khả năng tích hợp	Không tích hợp	Không tích hợp	Có thể tích hợp với các công cụ kiểm thử, hỗ trợ các hệ thống CI gọi và thực hiện ca kiểm thử.
Kiểm thử hình ảnh	Không tích hợp	Yêu cầu cài đặt thư viện bổ sung	Hỗ trợ tích hợp
Kiểm thử phân tích	Không tích hợp	Không	Katalon analytics
Loại giấy phép	Mã nguồn mở	Mã nguồn mở (apache)	Phần mềm miễn phí
Chi phí	Miễn phí	Miễn phí	Miễn phí

*Bảng 3: Bảng so sánh Appium, Selenium và Katalon Studio*

## 2. Ưu, nhược điểm

	Ưu điểm	Nhược điểm
--	---------	------------

Appium	<ul style="list-style-type: none"> <li>• Nguồn mở và miễn phí</li> <li>• Appium hỗ trợ nhiều platform khác nhau bao gồm: Android, IOS, FirefoxOS</li> <li>• Appium hỗ trợ đa ngôn ngữ: Java, Objective-C, JavaScript với node.js, PHP, Ruby, Python, C #,...</li> <li>• Ngoài ra còn có những ưu điểm khác như không cần mã nguồn để kiểm thử ứng dụng khi bạn có thể kiểm tra trực tiếp, bạn cũng có thể tương tác với các ứng dụng như máy ảnh, lịch vv trong kịch bản kiểm thử nếu cần.</li> </ul>	<ul style="list-style-type: none"> <li>• Không hỗ trợ cho Android API có version &gt; 17, tức là Android &gt; 4.2.</li> <li>• Chạy script rất chậm trên platform iOS.</li> <li>• Hỗ trợ hành động cử chỉ có giới hạn.</li> <li>• Không hỗ trợ cho thông báo toaster</li> </ul>
Selenium	<ul style="list-style-type: none"> <li>• Mã nguồn mở, không mất phí license và bảo trì.</li> <li>• Cộng đồng người dùng và phát triển lớn và tích cực để theo kịp các công nghệ phần mềm.</li> <li>• Tích hợp được các công cụ và framework khác để phát triển, tăng cường khả năng của nó.</li> </ul>	<ul style="list-style-type: none"> <li>• Người kiểm thử cần có kỹ năng và kinh nghiệm lập trình tốt để thiết lập và tích hợp Selenium với các công cụ và framework khác</li> <li>• Mất thời gian thiết lập và tích hợp các framework khác khi build dự án mới</li> <li>• Hỗ trợ chậm từ cộng đồng</li> </ul>

Katalon Studio	<ul style="list-style-type: none"> <li>● Không yêu license và bảo trì (có sẵn các dịch vụ hỗ trợ chuyên dụng có trả tiền nếu cần).</li> <li>● Tích hợp các framework và tính năng cần thiết để tạo và thực hiện các trường hợp thử nghiệm nhanh.</li> <li>● Được xây dựng dựa trên framework Selenium nhưng đã lược bỏ yêu cầu kỹ năng lập trình nâng cao cần thiết cho Selenium.</li> </ul>	<ul style="list-style-type: none"> <li>● Framework mới nổi với một cộng đồng phát triển nhanh chóng.</li> <li>● Các tính năng vẫn đang phát triển</li> <li>● Ngôn ngữ kịch bản hạn chế: chỉ hỗ trợ cho Java/Groovy</li> </ul>
----------------	--	---

*Bảng 4: Bảng so sánh ưu nhược điểm Appium, Selenium và Katalon Studio*

#### IV. Kết luận

Appium là công cụ kiểm thử với nhiều đặc điểm nổi bật như:

- Khả năng kiểm tra hồi quy ứng dụng cực nhanh và đơn giản. Những kỹ sư hoặc lập trình đang nghiên cứu về mobile app nên sử dụng Appium.
- Appium có thể hỗ trợ dưới nhiều nền tảng khác nhau, cụ thể là IOS và Android – hai nền tảng có số người dùng lớn nhất hiện nay
- Appium đa ngôn ngữ. Công cụ này thích hợp với vô số ngôn ngữ lập trình khác nhau như: JavaScript with Node.js, PHPPython,.. Đây là điểm cộng lớn bởi rất nhiều công cụ kiểm thử khác gặp hạn chế về mặt ngôn ngữ lập trình.

Bên cạnh đó, Appium là công cụ tự động hóa có thể khắc phục đa số những nhược điểm của những công cụ khác trên thị trường. Appium có khả năng chạy liên tục trên nhiều trình giả lập, giúp quá trình kiểm thử trở nên vô cùng thuận tiện.

Source code: [https://github.com/dattv155/Mobile\\_Testing\\_Appium](https://github.com/dattv155/Mobile_Testing_Appium)

Slide:

[https://www.canva.com/design/DAEu4RKewSE/IlwV11jc12CjvW531gjtA/view?utm\\_content=DAEu4RKewSE&utm\\_campaign=designshare&utm\\_medium=link&utm\\_source=publishsharelink](https://www.canva.com/design/DAEu4RKewSE/IlwV11jc12CjvW531gjtA/view?utm_content=DAEu4RKewSE&utm_campaign=designshare&utm_medium=link&utm_source=publishsharelink)

#### V. Tài liệu tham khảo

- [1] <https://viblo.asia/p/tim-hieu-khai-quat-ve-test-tool-appium-Ljy5VByz5ra>
- [2] <https://viettuts.vn/appium>
- [3] <https://appium.io/>
- [4] <https://www.katalon.com/resources-center/blog/appium-vs-selenium/>
- [5] <https://www.educba.com/appium-vs-selenium/>