**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PBL4: DỰ ÁN HỆ ĐIỀU HÀNH & MẠNG MÁY TÍNH**

**Đề tài 512. Xây dựng ứng dụng trên Raspberry giám sát sức khỏe hệ thống và điều khiển Raspberry từ xa.**

**SINH VIÊN THỰC HIỆN:**

**Trần Nữ Hoài Anh LỚP: 20TCLC\_DT1 NHÓM: 20.13A**

**Nguyễn Vũ Ngọc Linh LỚP: 20TCLC\_DT1 NHÓM: 20.13A**

**Trần Thị Ngọc Trâm LỚP: 20TCLC\_DT2 NHÓM: 20.13A**

**GIẢNG VIÊN HƯỚNG DẪN: ThS. Nguyễn Thế Xuân Ly**

**Đà Nẵng, 12/2022**

**LỜI CẢM ƠN**

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Trường Đại học Bách Khoa - Đại học Đà Nẵng đã cho chúng em một môi trường học tập thật tốt và bổ ích. Chúng em cũng xin gửi lời cảm ơn các thầy cô trong khoa Công Nghệ Thông Tin đã đem môn học PBL này vào chương trình dạy học để giúp chúng em có thể tiếp xúc với các đồ án nhằm học hỏi được các kinh nghiệm cần thiết và quý báu cho sau này. Đặc biệt, chúng em cũng xin chân thành cảm ơn thầy Nguyễn Thế Xuân Ly đã tận tình hướng dẫn chỉ bảo chúng em cách làm trong suốt quá trình thực hiện đồ án để chúng em có thể hoàn thành nó một cách tốt nhất.

Trong quá trình thực hiện đề tài này, chúng em sẽ không tránh khỏi những sai sót không đáng có do chưa có nhiều kinh nghiệm cũng như vốn kiến thức còn chưa đủ nên rất mong thầy (cô) có thể góp ý giúp cho chúng em để chúng em có thể sửa đổi và thực hiện tốt hơn cho các đồ án sắp tới.

Chúng em xin chân thành cảm ơn ạ!

MỤC LỤC

[MỤC LỤC 2](#_Toc90369523)

[DANH SÁCH HÌNH VẼ 3](#_Toc90369524)

[DANH SÁCH BẢNG BIỂU 4](#_Toc90369525)

[DANH SÁCH CÁC TỪ VIẾT TẮT 5](#_Toc90369526)

[MỞ ĐẦU (GIỚI THIỆU ĐỀ TÀI) 6](#_Toc90369527)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 7](#_Toc90369528)

[1.1. …. 7](#_Toc90369529)

[1.2. …. 7](#_Toc90369530)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 7](#_Toc90369531)

[2.1. …. 7](#_Toc90369532)

[2.2. …. 7](#_Toc90369533)

[CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 7](#_Toc90369534)

[3.1. …. 7](#_Toc90369535)

[3.2. …. 7](#_Toc90369536)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 7](#_Toc90369537)

[TÀI LIỆU THAM KHẢO 8](#_Toc90369538)

[[1] Tên tác giả, Tên tài liệu, Tên nhà xuất bản, năm xuất bản 8](#_Toc90369539)

[[2] Tên chủ sở hữu, Tên bài viết, url, ngày truy cập 8](#_Toc90369540)

[PHỤ LỤC 9](#_Toc90369541)

DANH SÁCH HÌNH VẼ

No table of figures entries found.

DANH SÁCH BẢNG BIỂU

No table of table entries found.

DANH SÁCH CÁC TỪ VIẾT TẮT

No table of abbreviation entries found.

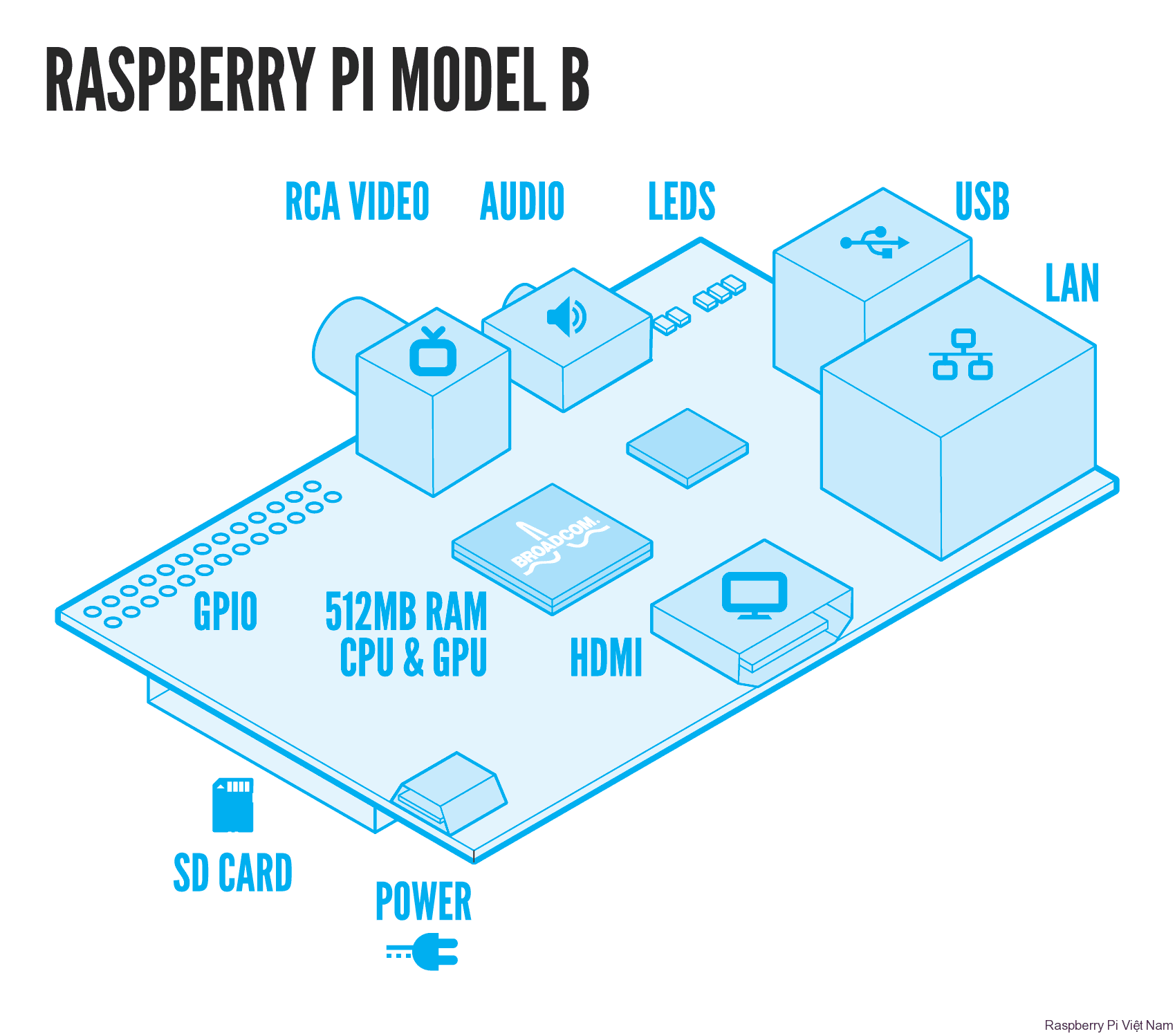
MỞ ĐẦU (GIỚI THIỆU ĐỀ TÀI)

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

## Giới thiệu về Raspberry Pi

Raspberry Pi là cái máy tính kích cỡ như iPhone và chạy HĐH Linux. Đặc tính của Raspberry Pi xây dựng xoay quanh bộ xử lí SoC Broadcom BCM2835 (là chip xử lí mobile mạnh mẽ có kích thước nhỏ hay được dùng trong điện thoại di động) bao gồm CPU, GPU, bộ xử lí âm thanh /video, và các tính năng khác,… tất cả được tích hợp bên trong chip có điện năng thấp này .

Về cơ bản Raspberry Pi có khá nhiều OS linux chạy được nhưng vẫn có sự thiếu vắng của Ubuntu (do CPU ARMv6). Một số Distributions Linux có thể chạy được trên Raspberry Pi như là Raspbian, Pidora, openSUSE, OpenWRT, OpenELEC,….



*Ảnh 1.1. Sơ đồ cấu tạo của Raspberry Pi*

## Giới thiệu về Zabbix

Zabbix là một phần mềm mã nguồn mở được công bố lần đầu 2001, có chức năng giám sát các dịch vụ mạng và  tài nguyên công nghệ khác như máy chủ, phần cứng,.. một cách nhanh chóng, hiệu quả. Kết quả phân tích, thống kê số liệu được thể hiện một cách trực quan; các thông báo về vấn đề tiềm ẩn được cập nhật chính xác và kịp thời.

Tính đến nay, Zabbix là nền tảng được nhiều công ty có quy mô lớn sử dụng khi nó có thể theo dõi và thu thập chỉ số thực từ hàng triệu server trên toàn thế giới.  Zabbix cung cấp cho người dùng cả phiên bản tính phí và trả phí, đây là một chiến lược giúp hệ thống trở lên thân thiện với mọi phân khúc khách hàng.

### Ưu điểm:

- Đây là một công cụ mã nguồn mở, dễ phát triển và mở rộng theo ý người dùng. Chi phí đầu tư ban đầu thấp

- Thực hiện chức năng giám sát toàn diện trên các thiết bị phần cứng và dịch vụ mạng

- Hỗ trợ tốt các máy chủ đặt trên hệ điều hành Linux

- Giao diện thân thiện và đẹp mắt

- Phân quyền user linh động và dễ thực hiện

- Thông báo các sự cố nhanh chóng qua email hoặc app

- Các chức năng theo dõi thống kê được thực hiện chủ động, dễ thiết lập và sửa đổi

- Sở hữu tài nguyên công cụ lớn với nhiều plugin hỗ trợ cho các dịch vụ hệ thống khác nhau

- Có tính năng chứng thực người dùng

- Kết quả được trả về dưới dạng biểu đồ trực quan, dễ phân tích và đánh giá.

### Nhược điểm

- Zabbix không hỗ trợ giao diện web mobile.

- Thiết kế template/alerting rule của hệ thống được người dùng đánh giá là không mấy thân thiện vì đôi khi nó yêu cầu cao về kiến thức code của người dùng.

- Sẽ gặp tình trạng mất ổn định và các vấn đề hiệu suất về PHP và Database khi sử dụng hệ thống mạng lớn hơn 1000+ node.

## JSP, Servlet

### JSP

[JSP](https://vi.wikipedia.org/wiki/JSP) là viết tăt của **Java server page** hay **Java Scripting Preprocessor** – “Bộ tiền xử lý văn lệnh Java”. Đây là một công nghệ Java cho phép các nhà phát triển tạo nội dung HTML, XML hay một số định dạng khác giúp cho trang web sinh động hớn. Các JSP tag đặc biệt, hầu hết**bắt đầu** với**<%**và **kết thúc**với **%>.**

JSP thường được**làm view trong ứng dụng mvc pattern**. Thực ra, JSP vẫn có thể đáp ứng những yêu cầu khác nhưng để thuận tiện cho việc debug hay tái sử dụng các đoạn mã thì người ta thường dùng làm view còn servlet sẽ làm controller.)

1. Những thành phần của một trang JSP

* Thẻ Root: Thẻ này sẽ chứa các thuộc tính, thông tin của trang JSP.
* Comment:  Cũng như trang HTML , trong JSP, bạn cũng có thể comment với kí hiệu này: *<! your comment –>*
* Declaration: khai báo biến hoặc phương thức của java ngay trong trang JSP. Nhưng nếu như khai báo quá nhiều trong trang thì sẽ bị nhầm lẫn giữa code JSP và code java. Cú pháp là *<%! your code %>*
* Expression: được sử dụng để chèn một giá trị vào trong trang một cách trực tiếp.
* Thẻ biểu thức JSP**:**được sử dụng để đánh giá một biểu thức và định hướng các output đến một trình duyệt web phù hợp. Cú pháp khai báo là: *<%= your code %>*
* Scriptlet Tag: cho phép bạn viết mã java trong trang JSP. Cú pháp như sau: *<% <i> mã java </ i>%>*
* Directive Elements

1. “Chu kỳ sống” của JSP



*Ảnh 1.3.1. Chu kỳ sống của JSP*

1. Ưu điểm của JSP

* Hỗ trợ cho việc thiết kế giao diện web dễ dàng hơn.
* Có vai trò lớn trong việc cho phép thiết kế web tạo nên những trang web động.
* Có thể viết một nơi và chạy bất cứ nơi nào.

1. Nhược điểm của JSP

* Tiêu tốn **dung lượng** lưu trữ phía server**gấp đôi.**
* Lần đầu tiên truy cập vào trang JSP sẽ mất nhiều thời gian chờ.

### Servlet

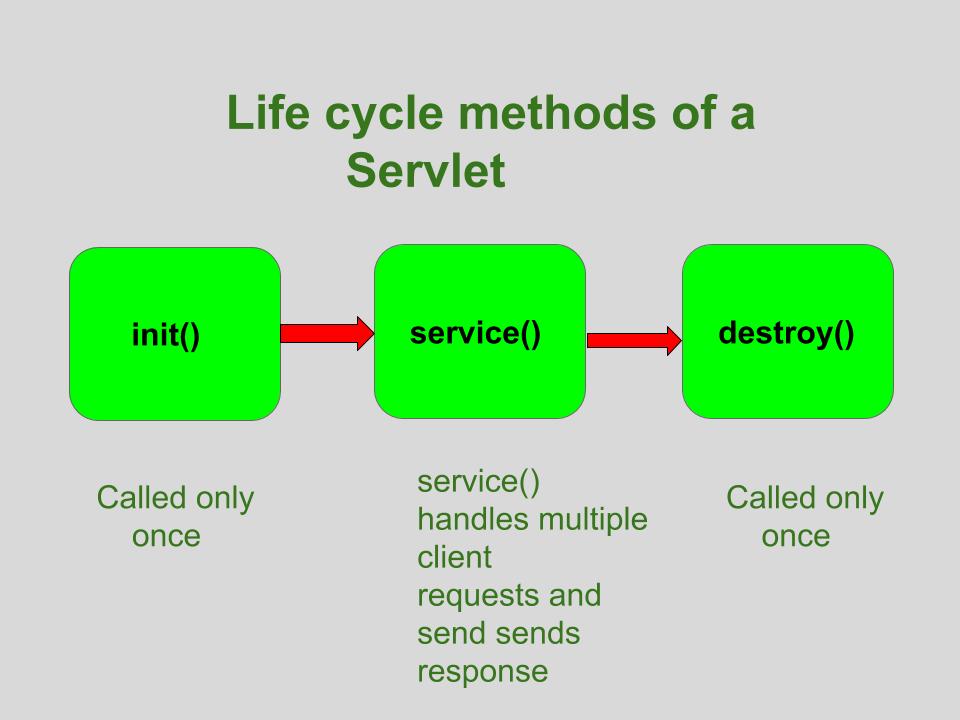
Java Servlet là chương trình chạy trên một Web hoặc ứng dụng máy chủ (Application Server). Nó hoạt động như một lớp trung gian giữa một yêu cầu đến từ một trình duyệt Web hoặc HTTP khách (Client) và cơ sở dữ liệu hoặc các ứng dụng trên máy chủ HTTP (HTTP Server). Hiểu đơn giản, Servlet là một chương trình chạy trên môi trường Web Server hoặc môi trường Application có thực thi mã java với nhiệm vụ chính là giúp thực thi câu lệnh một cách độc lập giúp kết nối các lớp với nhau.

Tóm lại, JSP là mở rộng của Servlet. Trong thực tế, JSP và Servlet được sử dụng đồng thời để phát triển ứng dụng. Nếu JSP đại diện cho trang web thì Servlet đại diện cho các thành phần Java. Trong mô hình [MVC](https://wiki.tino.org/mvc-la-gi/), Servlet xử lý phần controller còn JSP xử lý phần view.

1. Công dụng của Servlet

* Nhận client request và lấy thông tin từ request: Đọc dữ liệu rõ ràng do client (trình duyệt) gửi.
* Xử lý nghiệp vụ và phát sinh chuyên môn (bằng cách truy cập database): Quá trình xử lý dữ liệu và tạo ra các kết quả này có thể yêu cầu nói chuyện với một cơ sở dữ liệu, thực hiện một cuộc gọi RMI hoặc CORBA, gọi một dịch vụ Web, hoặc tính trực tiếp phản hồi.
* Tạo và gửi request đến client hoặc tại request mới đến Servlet mới hoặc JSP mới: Không chỉ gửi dữ liệu rõ ràng tới client (trình duyệt) dưới nhiều định dạng như văn bản (HTML hoặc XML), nhị phân (hình ảnh GIF), Excel, … mà còn gửi phản hồi HTTP ẩn cho client (trình duyệt). Điều này bao gồm nói với trình duyệt hoặc các trình khách khác loại tài liệu đang được trả về (ví dụ, HTML), thiết lập cookie và các tham số bộ nhớ đệm, và các tác vụ khác.

1. “Chu kỳ sống” của Servlet



*Ảnh 1.3.2. Chu kỳ sống của Servlet*

## API

API là viết tắt của Giao diện lập trình ứng dụng (Application Programming Interface). API là một trung gian phần mềm cho phép hai ứng dụng giao tiếp với nhau, có thể sử dụng cho web-based system, operating system, database system, computer hardware, or software library. API cho phép một ứng dụng giao tiếp với ứng dụng khác thông qua các lệnh đơn giản và cách các lệnh này được gửi và định dạng mà dữ liệu được truy xuất.

API được xây dựng trên chính 2 thành phần: Request và Response

Diagram

Description automatically generated

*Ảnh 1.4. Cách thức API hoạt động*

### Request

#### URL

Là địa chỉ duy nhất cho 1 request, thường là đường dẫn tới một hàm xử lí logic.

#### Method

HTTP request có tất cả 9 loại method, 2 loại được sử dụng phổ biến nhất là GET và POST

* GET: Sử dụng để lấy thông tin từ server theo URI đã cung cấp.
* HEAD: Giống với GET nhưng response trả về không có body, chỉ có header.
* POST: Gửi thông tin tới sever thông qua các parameters HTTP.
* PUT: Ghi đè tất cả thông tin của đối tượng với những gì được gửi lên.
* PATCH: Ghi đè các thông tin được thay đổi của đối tượng.
* DELETE: Xóa resource trên server.
* CONNECT: Thiết lập một kết nối tới server theo URI.
* OPTIONS: Mô tả các tùy chọn giao tiếp cho resource.
* TRACE: Thực hiện một bài test loop-back theo đường dẫn đến resource.

#### Headers

Là nơi chứa các thông tin cần thiết của 1 request nhưng end-users không biết có sự tồn tại của nó. Ví dụ: độ dài của request body, thời gian gửi request, loại thiết bị đang sử dụng, loại định dạng cái response mà client có đọc được…

#### Body

Là nơi chứa thông tin mà client sẽ điền.

### Response

Sau khi nhận được request từ phía client, server sẽ xử lý cái request đó và gửi ngược lại cho client 1 cái response. Cấu trúc của 1 response tương đối giống phần request nhưng Status code sẽ thay thế cho URL và Method. Phần Header và body tương đối giống với request. Tóm lại, nó có cầu trúc 3 phần:

* Status code
* Headers
* Body

Status code (Mã hóa trạng thái thường được gọi là mã trạng thái) là một số nguyên 3 ký tự, trong đó ký tự đầu tiên của Status-Code định nghĩa loại Response và hai ký tự cuối không có bất cứ vai trò phân loại nào. Có 5 giá trị của ký tự đầu tiên:

* 1xx: Information (Thông tin): Khi nhận được những mã như vậy tức là request đã được server tiếp nhận và quá trình xử lý request đang được tiếp tục.
* 2xx: Success (Thành công): Khi nhận được những mã như vậy tức là request đã được server tiếp nhận, hiểu và xử lý thành công
* 3xx: Redirection (Chuyển hướng): Mã trạng thái này cho biết client cần có thêm action để hoàn thành request
* 4xx: Client Error (Lỗi Client): Nó nghĩa là request chứa cú pháp không chính xác hoặc không được thực hiện.
* 5xx: Server Error (Lỗi Server): Nó nghĩa là Server thất bại với việc thực hiện một request nhìn như có vẻ khả thi.

## SSH

SSH là giao thức đăng nhập vào server từ xa, cho phép người dùng kiểm soát, chỉnh sửa và quản trị dữ liệu của server thông qua nền tảng Internet. SSH là viết tắt của Secure Socket Shell. SSH cũng giúp việc kết nối của mạng lưới máy chủ và máy khách an toàn, hiệu quả và bảo mật thông tin tốt hơn. Tương tự SSL, SSH cũng có chức năng mã hóa dữ liệu đường truyền. Điều này phục vụ cho quá trình truyền tập tin, chạy chương trình.

SSH hoạt động ở tầng thứ 4 trong mô hình TCP/IP. Nó cho phép tương tác giữa máy chủ và máy khách, sử dụng cơ chế mã hoá nhằm ngăn chặn các hiện tượng nghe trộm, đánh cắp thông tin trên đường truyền. Đây là điều mà các giao thức trước đây như telnet, rlogin không đáp ứng được.

### SSH có bảo mật không?

Giao thức SSH cung cấp các dịch vụ của máy chủ như xác thực, mã hóa, xác minh tính toàn vẹn dữ liệu và xác thực khách hàng.

Để xác minh mã hóa và toàn vẹn dữ liệu, một số thuật toán được cung cấp mà mỗi sản phẩm SSH có thể thực hiện trong một khoảng thời gian nhất định. Trong khi đó, xác thực khách hàng lại được SSH thực hiện bằng cách sử dụng mật khẩu, khóa công khai, chỉ 1 tài khoản login trên 1 thời điểm (single sign-on), và các phương pháp khác.

Vì vậy, khi thực hiện đúng và sử dụng chúng, giao thức này hoàn toàn an toàn để bảo mật thông tin và chống lại hầu hết tất cả các cuộc tấn công mật mã.

### Quy trình hoạt động của SSH

**Bước 1**: Định danh host – xác định định danh của hệ thống tham gia phiên làm việc SSH. Đơn giản hơn, đây là bước khởi tạo kết nối SSH. Kết nối này là một kênh giao tiếp bảo mật giữa server và client.  
**Bước 2**: Mã hoá dữ liệu – thiết lập kênh làm việc mã hoá. Sau khi Client xác định được định danh của Server 1 kết nối bảo mật đối xứng được hình thành giữa 2 bên.  
**Bước 3**: Chứng thực và giải mã – xác thực người sử dụng có quyền đăng nhập hệ thống. Kết nối này sẽ được sử dụng để Server xác thực Client.

### Ưu điểm của SSH

Lý do khiến nhiều người lựa chọn sử dụng giao thức này xuất phát từ những ưu điểm vượt trội của SSH so với các giao thức khác. SSH có khả năng mã hóa và truyền tải dữ liệu an toàn giữa mạng lưới host và client. Host đại diện cho máy chủ từ xa bạn muốn kết nối tới và client là máy tính của bạn dùng để truy cập tới host.

Lợi thế thứ hai là có nhiều cách khác nhau giúp việc mã hóa qua SSH được thực hiện hiệu quả và toàn vẹn như:

* Symmetrical encryption
* Asymmetrical encryption
* Hashing

# CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## Tác nhân

Hệ thống chỉ có 1 tác nhân: User

Để có thể hình dung rõ hơn về tác nhân cũng như các chức năng của hệ thống, hãy xem các sơ đồ use cases được trình bày sau đây.

## Biểu đồ ca sử dụng

### Use case tổng quát

Diagram

Description automatically generated

### Use case quản lý người dùng

Diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã Use case** | UC001 | | |
| **Tên Use case** | Quản lý người dùng | | |
| **Tác nhân** | Người dùng | | |
| *Đăng nhập:* | | | |
| **Mô tả** | Tác nhân đăng nhập vào hệ thống để sử dụng các chức năng hệ thống | | |
| **Sự kiện kích hoạt** | Khởi động ứng dụng | | |
| **Tiền điều kiện** | Tác nhân đã có tài khoản trên hệ thống | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1 | Người dùng | Khởi động ứng dụng |
| 2 | Hệ thống | Yêu cầu đăng nhập |
| 3 | Người dùng | Nhập username và mật khẩu |
| 4 | Người dùng | Click chuột vào Log in để đăng nhập |
| 5 | Hệ thống | Kiểm tra tên tài khoản và mật khẩu có hợp lệ trong hệ thống hay không |
| 6 | Hệ thống | Hiển thị cửa sổ với người dùng |
| **Luồng sự kiện thay thế** | **STT** | **Thực hiện bởi** | **Hành động** |
| 7 | Hệ thống | Thông báo trang lỗi: Cần kiểm tra lại username và password. |
| **Hậu điều kiện** | Người dùng đăng nhập được vào hệ thống | | |
| *Thay đổi mật khẩu:* | | | |
| **Mô tả** | Tác nhân sửa mật khẩu đăng nhập | | |
| **Sự kiện kích hoạt** | Chọn button “Change password” trên thanh sidebar | | |
| **Tiền điều kiện** | Tác nhân đã đăng nhập trên hệ thống | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Chọn chức năng đổi mật khẩu |
| 2. | Hệ thống | Hiển thị chức năng đổi mật khẩu |
| 3. | Người dùng | Nhập mật khẩu mới và xác nhận lại mật khẩu mới |
| 4. | Người dùng | Yêu cầu đổi mật khẩu |
| 5. | Hệ thống | Kiểm tra lại mật khẩu có đạt được những điều kiện chưa (dài trên 8 ký tự, không được chứa username, … ) |
| 6. | Hệ thống | Đổi lại mật khẩu cho user |
| **Luồng sự kiện thay thế** | **STT** | **Thực hiện bởi** | **Hành động** |
| 7. | Hệ thống | Hiển thị cửa sổ thông báo đổi mật khẩu không thành công kèm theo lỗi mà người dùng mắc phải |
| **Hậu điều kiện** | Người dùng thay đổi được mật khẩu | | |

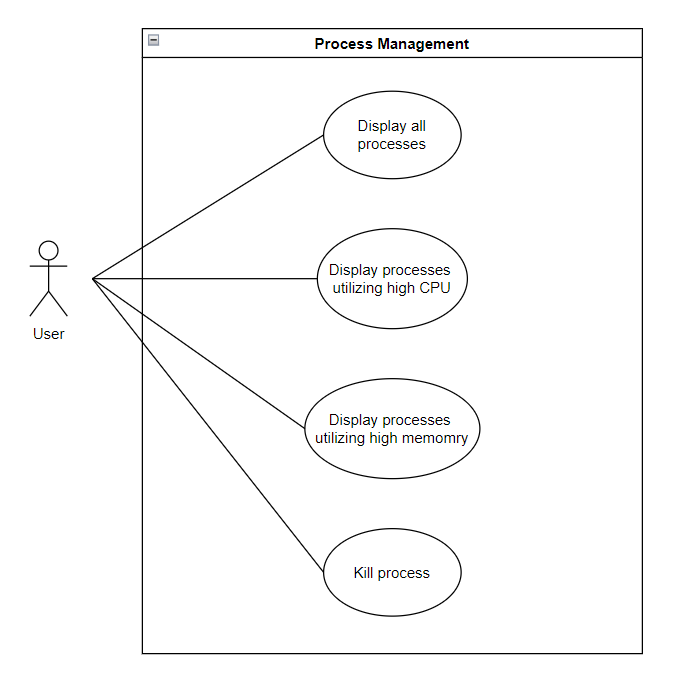
### Use case quản lý thiết bị Raspberry

Diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã Use case** | UC002 | | |
| **Tên Use case** | Quản lý thiết bị | | |
| **Tác nhân** | Người dùng | | |
| **Mô tả** | Tác nhân muốn thực hiện các chức năng “Thêm”, “Xem”, “Sửa”, “Xóa” thông tin thiết bị Raspberry có trong hệ thống. | | |
| **Sự kiện kích hoạt** | Click Overview hoặc Hosts trên thanh sidebar bên trái màn hình | | |
| **Tiền điều kiện** | Tác nhân đăng nhập thành công | | |
| *Xem tổng quan về các thiết bị có trong hệ thống:* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Chọn Overview trên thanh sidebar |
| 2. | Hệ thống | Hiển thị thông tin tổng quát về các thiết bị có trong hệ thống bằng các biểu đồ Availability, CPU Load Average, CPU Utilization, Memory Utilization. |
| *Hiển thị danh sách các thiết bị trong hệ thống:* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Chọn Hosts trên thanh sidebar |
| 2. | Hệ thống | Hiển thị danh sách các thiết bị Raspberry trong hệ thống. |
| *Hiển thị thông tin chi tiết về 1 thiết bị* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Chọn Hosts trên thanh sidebar. |
| 2. | Hệ thống | Hiển thị danh sách các thiết bị Raspberry trong hệ thống. |
| 3. | Người dùng | Chọn vào biểu tượng con mắt  tương ứng với thiết bị ta muốn xem. |
| 4. | Hệ thống | Hiển thị toàn bộ thông tin về thiết bị mà ta đã chọn. |
| *Thêm mới 1 thiết bị vào hệ thống:* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Chọn Hosts trên thanh sidebar. |
| 2. | Hệ thống | Hiển thị danh sách các thiết bị Raspberry trong hệ thống. |
| 3. | Người dùng | Click vào button “Create new host” |
| 4. | Hệ thống | Hiển thị màn hình tạo mới thiết bị. |
| 5. | Người dùng | Nhập vào các trường. |
| 6. | Hệ thống | Thêm 1 thiết bị vào hệ thống |
| **Luồng sự kiện thay thế** | 7. | Hệ thống | Thông báo lỗi nếu không nhập đủ trường. |
| 8. | Hệ thống | Thông báo lỗi nếu điền thông tin lỗi. |
| *Sửa thông tin thiết bị:* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Chọn Hosts trên thanh sidebar. |
| 2. | Hệ thống | Hiển thị danh sách các thiết bị Raspberry trong hệ thống. |
| 3. | Người dùng | Click vào biểu tượng cây bút  tương ứng với thiết bị muốn sửa thông tin |
| 4. | Hệ thống | Hiển thị màn hình sửa thông tin thiết bị. |
| 5. | Người dùng | Nhập vào các trường. |
| 6. | Hệ thống | Thêm 1 thiết bị vào hệ thống |
| **Luồng sự kiện thay thế** | 7. | Hệ thống | Thông báo lỗi nếu điền thông tin lỗi. |
| *Xóa thiết bị:* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Chọn Hosts trên thanh sidebar. |
| 2. | Hệ thống | Hiển thị danh sách các thiết bị Raspberry trong hệ thống. |
| 3. | Người dùng | Click vào biểu tượng thùng rác  tương ứng với thiết bị muốn sửa thông tin |
| 4. | Hệ thống | Hiển thị màn hình xác nhận xóa. |
| 5. | Người dùng | Chọn button “Confirm” |
| 6. | Hệ thống | Xóa thiết bị ra khỏi hệ thống |
| **Luồng sự kiện thay thế** | 7. | Hệ thống | Hệ thống không xóa nếu người dùng chọn button “Cancel” khi hệ thống hiển thị màn hình xác nhận xóa. |

### Use case quản lý tiến trình của thiết bị



|  |  |  |  |
| --- | --- | --- | --- |
| **Mã Use case** | UC003 | | |
| **Tên Use case** | Quản lý tiến trình | | |
| **Tác nhân** | Người dùng | | |
| **Mô tả** | Tác nhân muốn thực hiện các chức năng “Xem”, “Xóa” các tiến trình của 1 thiết bị Raspberry có trong hệ thống. | | |
| **Sự kiện kích hoạt** | Click Hosts trên thanh sidebar bên trái màn hình, chọn vào biểu tượng con mắt  tương ứng với thiết bị Raspberry, chọn button “Show Thread”, đăng nhập SSH. | | |
| **Tiền điều kiện** | Tác nhân đăng nhập hệ thống và đăng nhập SSH thành công thành công | | |
| *Xem toàn bộ tiến trình của thiết bị:* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Đăng nhập SSH thành công. |
| 2. | Hệ thống | Hiển thị toàn bộ tiến trình đang chạy của thiết bị đó. |
| *Xem các tiến trình tiêu tốn CPU nhiều nhất của thiết bị:* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Đăng nhập SSH thành công. |
| 2. | Hệ thống | Hiển thị toàn bộ tiến trình đang chạy của thiết bị đó. |
| 3. | Người dùng | Chọn “Display processes utilizing high CPU” ở đầu trang |
| 4. | Hệ thống | Hiển thị các tiến trình đang chạy mà tiêu tốn nhiều CPU của thiết bị đó. |
| *Xem các tiến trình tiêu tốn nhiều bộ nhớ của thiết bị:* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Đăng nhập SSH thành công. |
| 2. | Hệ thống | Hiển thị toàn bộ tiến trình đang chạy của thiết bị đó. |
| 3. | Người dùng | Chọn “Display processes utilizing high memory” trên thanh sidebar |
| 2. | Hệ thống | Hiển thị các tiến trình đang chạy mà tiêu tốn nhiều bộ nhớ của thiết bị. |
| *Giết tiến trình:* | | | |
| **Luồng sự kiện chính**  **(Thành công)** | **STT** | **Thực hiện bởi** | **Hành động** |
| 1. | Người dùng | Mở 1 danh sách tiến trình bất kỳ (toàn bộ tiến trình, danh sách tiến trình dùng nhiều CPU hoặc danh sách tiến trình dùng nhiều bộ nhớ) |
| 2. | Hệ thống | Hiển thị danh sách các tiến trình. |
| 3. | Người dùng | Click vào biểu tượng  tương ứng với process ta muốn giết. |
| 4. | Hệ thống | Xóa tiến trình đó và quay lại màn hình danh sách các tiến trình. |

## Biểu đồ hoạt động

### Biểu đồ hoạt động cho use case quản lý người dùng

### Biểu đồ hoạt động cho use case quản lý thiết bị Raspberry

### Biểu đồ hoạt động cho use case quản lý luồng của thiết bị

Diagram

Description automatically generated

*Ảnh 2.3.3.1. Biểu đồ hoạt động cho use case xem toàn bộ tiến trình của thiết bị*

*Diagram

Description automatically generated*

*Ảnh 2.3.3.2. Biểu đồ hoạt động cho use case xem các tiến trình tiêu tốn nhiều CPU*

*Diagram

Description automatically generated*

*Ảnh 2.3.3.3. Biểu đồ hoạt động cho use case xem các tiến trình tiêu tốn nhiều bộ nhớ*

## Biểu đồ tuần tự

## Các đối tượng quản lý

### User (Người dùng)

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | userid | int | ID của user |
| 2 | username | String | Tên user |
| 3 | password | String | Mật khẩu user |

### Host (máy tính Raspberry)

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | hostId | String | ID của thiết bị |
| 2 | hostname | String | Tên thiết bị |
| 3 | availability | String | Tính khả dụng |
| 4 | ipAddress | String | Địa chỉ IP |
| 5 | port | String | Cổng |
| 6 | NumberOfProcesses | long | Số tiến trình |
| 7 | NumberOfCPUs | long | Số CPU |
| 8 | SystemUptime | long | Thời gian thiết bị hoạt động |
| 9 | TotalSpace | long | Tổng dung lượng |
| 10 | UsedSpace | long | Dung lượng đã sử dụng |
| 11 | AvailableMemory | long | Bộ nhớ khả dụng |
| 12 | LoadAverage1m | double | Chỉ số tải trung bình trong 1p |
| 13 | LoadAverage5m | double | Chỉ số tải trung bình trong 5p |
| 14 | LoadAverage15m | double | Chỉ số tải trung bình trong 15p |
| 15 | CPUutilization | double | Mức độ sử dụng CPU |
| 16 | DiskUtilization | double | Mức độ sử dụng ổ đĩa |
| 17 | SpaceUtilization | double | Mức độ sử dụng dung lượng |
| 18 | AvailableMemoryInPS | double | Bộ nhớ còn lại |
| 19 | MemoryUtilization | double | Mức độ sử dụng bộ nhớ |

### Process (Tiến trình)

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | ProcessId | String | ID của tiến trình |
| 2 | ProcessName | String | Tên tiến trình |
| 3 | ProcessCpu | Double | Mức độ CPU sử dụng |
| 4 | ProcessMemory | Float | Mức độ bộ nhớ sử dụng |

# 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

## Triển khai

## Đánh giá kết quả

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

TÀI LIỆU THAM KHẢO

[1] Tên tác giả, Tên tài liệu, Tên nhà xuất bản, năm xuất bản

[2] Tên chủ sở hữu, Tên bài viết, url, ngày truy cập

PHỤ LỤC