TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &

TRUYỀN THÔNG VIỆT - HÀN

**KHOA KHOA HỌC MÁY TÍNH**



**ĐỒ ÁN MÔN HỌC LẬP TRÌNH MẠNG**

Đề tài: **SỬ DỤNG GIAO THỨC HTTP VÀ ỨNG DỤNG MÔ HÌNH CLIENT-SERVER ĐỂ XÂY DỰNG WEBSERVER**

**Thành viên:                                  Nguyễn Ngọc Sỹ 22IT250**

**Phan Nhật Tân 22IT260**

**Giảng viên hướng dẫn:             THS. NGUYỄN THANH CẨM**

**Đà Nẵng, tháng 10 năm 2024**

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &

TRUYỀN THÔNG VIỆT - HÀN

**KHOA KHOA HỌC MÁY TÍNH**



**ĐỒ ÁN MÔN HỌC LẬP TRÌNH MẠNG**

Đề tài: **SỬ DỤNG GIAO THỨC HTTP VÀ ỨNG DỤNG MÔ HÌNH CLIENT-SERVER ĐỂ XÂY DỰNG WEBSERVER**

**Thành viên:                                  Nguyễn Ngọc Sỹ 22IT250**

**Phan Nhật Tân 22IT260**

**Giảng viên hướng dẫn:             THS. Nguyễn Thanh Cẩm**

**Đà Nẵng, tháng 10 năm 2024**

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

Giảng viên hướng dẫn

**THS.Nguyễn Thanh Cẩm**

**LỜI CẢM ƠN**

Để hoàn thành bài báo cáo môn lập trình mạng này, trước tiên nhóm em xin gửi lời cảm ơn đến các thầy cô giảng viên trong khoa Khoa học máy tính Trường Công nghệ Thông tin và Truyền thông Việt – Hàn.

Đặc biệt nhóm em xin gửi đến thầy Nguyễn Thanh Cẩm đã tận tình hướng dẫn, giúp đỡ trong việc nâng cao kiến thức chuyên môn, phát huy cho chúng em tính tự học, tự tìm hiểu, từ đó hình thành cho chúng em phương pháp nghiên cứu, tìm tòi, tiếp cận với khoa học công nghệ. Đồng thời trau dồi cho chúng em kỹ năng học tập năng động và sáng tạo.

Vì kiến thức còn hạn chế, trong quá trình tìm hiểu về đề tài này chúng em không thể tránh những sai sót. Chúng em rất mong nhận được nhận xét, đánh giá, góp ý từ các thầy, cô để rút ra kinh nghiệm từ bài báo cáo học phần này và chuẩn bị tốt cho những bài báo cáo học phần sau. Chúng em xin chân thành cảm ơn!

**MỤC LỤC**

[MỞ ĐẦU 7](#_Toc181276810)

[1. Giới thiệu đề tài 7](#_Toc181276811)

[**1.1 Lý do chọn đề tài** 7](#_Toc181276812)

[**1.2 Mục tiêu dự kiến** 7](#_Toc181276813)

[**1.3 Kết quả đề tài** 8](#_Toc181276814)

[2. Kế hoạch thực hiện 9](#_Toc181276815)

[CHƯƠNG 1 CƠ SỞ LÝ THUYẾT 10](#_Toc181276816)

[1.1. Khái niệm về lập trình mạng 10](#_Toc181276817)

[1.2. Các mô hình mạng 10](#_Toc181276818)

[**1.2.1. Mô hình OSI (Open Systems Interconnection)** 10](#_Toc181276819)

[**1.2.2. Mô hình TCP/IP** 11](#_Toc181276820)

[**1.2.3 Các mô hình ứng dụng mạng** 12](#_Toc181276821)

[**a) Mô hình Client – Server** 12](#_Toc181276822)

[**b) Mô hình Peer-to-Peer (P2P)** 13](#_Toc181276823)

[**c) Hệ thống Phân tán** 14](#_Toc181276824)

[**1.2.4 Các giao thức mạng** 15](#_Toc181276825)

[**a) TCP/IP** 15](#_Toc181276826)

[**b) UDP** 15](#_Toc181276827)

[**c) HTTP/HTTPS** 15](#_Toc181276828)

[**d) SMTP** 16](#_Toc181276829)

[**e) RMI** 16](#_Toc181276830)

[1.3 Công nghệ sử dụng 17](#_Toc181276831)

[**1.3.1 Java** 17](#_Toc181276832)

[**Môi trường** 17](#_Toc181276833)

[**Thư viện** 18](#_Toc181276834)

[**1.3.2 MySQL** 18](#_Toc181276835)

[Chương 2: Phân tích & Thiết kế Hệ thống 19](#_Toc181276836)

[2.1. Giới thiệu bài toán của đồ án 19](#_Toc181276837)

[2.2. Yêu cầu hệ thống 20](#_Toc181276838)

[**2.2.1. Yêu cầu chức năng** 20](#_Toc181276839)

[**2.2.2. Yêu cầu phi chức năng** 21](#_Toc181276840)

[**2.2.3 Các tác nhân** 21](#_Toc181276841)

[2.3. Sơ đồ Use Case (UC) 22](#_Toc181276842)

[**2.3.1. Xác định các Use Case** 22](#_Toc181276843)

[**2.3.2. Biểu đồ Use Case** 23](#_Toc181276844)

[2.4 Mô hình hóa cấu trúc 23](#_Toc181276845)

[2.5 Mô hình hóa hành vi 24](#_Toc181276846)

[**2.5.1 Biểu đồ hoạt động** 24](#_Toc181276847)

[**2.5.2 Biểu đồ trạng thái** 26](#_Toc181276848)

[**2.5.3 Biểu đồ trình tự** 27](#_Toc181276849)

[**2.5.4. Đặc tả Use Case chính** 27](#_Toc181276850)

[Chương 3: Thực hiện Chương trình (Giao diện, Màn hình chính) 29](#_Toc181276851)

[3.1. Giao diện Server 29](#_Toc181276852)

[3.2. Hướng dẫn sử dụng và giới thiệu kết quả 30](#_Toc181276853)

[**3.2.1. Khởi động Server** 30](#_Toc181276854)

[**3.2.2. Gửi yêu cầu từ Client** 30](#_Toc181276855)

[**3.2.3. Xem kết quả phản hồi** 31](#_Toc181276856)

[3.3. Kết quả thử nghiệm 32](#_Toc181276857)

[Kết luận và Hướng phát triển 33](#_Toc181276858)

[1. Kết luận 33](#_Toc181276859)

[2. Hướng phát triển 34](#_Toc181276860)

[Tài liệu tham khảo 35](#_Toc181276861)

**MỤC LỤC HÌNH ẢNH**

[Hình 1. 1 Mô hình OSI 10](#_Toc181276752)

[Hình 1. 2 Mô hình TCP/IP 11](#_Toc181276753)

[Hình 1. 3 Mô hình Client – Server 12](#_Toc181276754)

[Hình 1. 4 Mô hình Peer-to-Peer 13](#_Toc181276755)

[Hình 2. 1 Usecase diagram 22](#_Toc181276756)

[Hình 2. 2 Class diagram 23](#_Toc181276757)

[Hình 2. 3 Activities diagram 24](#_Toc181276758)

[Hình 2. 4 State diagram 25](#_Toc181276759)

[Hình 2. 5 Sequence diagram 26](#_Toc181276760)

[Hình 3.1 Giao Diện Server 29](#_Toc181276761)

[Hình 3.2 Giao Diện Home 30](#_Toc181276762)

[Hình 3.3 Giao Diện Trang Contact 31](#_Toc181276763)

[Hình 3.4 Phản Hồi Bên Server 32](#_Toc181276764)

# **MỞ ĐẦU**

## 1. Giới thiệu đề tài

### **1.1 Lý do chọn đề tài**

Đề tài "Sử dụng giao thức HTTP và ứng dụng mô hình Client-Server để xây dựng WebServer" được chọn vì tầm quan trọng của giao thức HTTP trong việc truyền tải thông tin trên Internet. HTTP (HyperText Transfer Protocol) là nền tảng cho việc giao tiếp giữa máy chủ và trình duyệt web, cho phép người dùng truy cập và tương tác với các nội dung trực tuyến.

Trong kỷ nguyên số ngày nay, với sự gia tăng mạnh mẽ của dịch vụ trực tuyến, việc hiểu rõ cách thức hoạt động của một WebServer trở nên cực kỳ quan trọng. Đề tài này không chỉ giúp sinh viên nắm bắt được các nguyên lý cơ bản của lập trình mạng mà còn rèn luyện kỹ năng lập trình thực tế thông qua việc xây dựng một WebServer đơn giản.

Mục tiêu của đề tài là phát triển một WebServer có khả năng xử lý các yêu cầu HTTP cơ bản, từ đó giúp hiểu rõ hơn về các thành phần của mô hình Client-Server, cấu trúc HTTP và cách thức mà WebServer phục vụ cho người dùng qua Internet.

### **1.2 Mục tiêu dự kiến**

Mục tiêu của đề tài này bao gồm:

* **Xây dựng WebServer đơn giản**: Phát triển một WebServer có khả năng xử lý các yêu cầu HTTP cơ bản từ client, bao gồm các yêu cầu GET và POST.
* **Xử lý yêu cầu và phản hồi**: Server sẽ nhận các yêu cầu từ client, xử lý chúng và gửi phản hồi tương ứng, bao gồm các tài nguyên như HTML, CSS, và hình ảnh.
* **Hỗ trợ nhiều loại nội dung**: Hệ thống sẽ cho phép phục vụ nhiều loại nội dung khác nhau, từ các tệp HTML tĩnh đến các tài nguyên động như hình ảnh và tệp văn bản.
* **Giao tiếp Client-Server**: Thiết lập giao tiếp giữa client và server qua giao thức HTTP, đảm bảo rằng mọi yêu cầu đều được xử lý một cách hiệu quả và đáng tin cậy.
* **Quản lý phiên làm việc**: Hệ thống sẽ có khả năng quản lý phiên làm việc của người dùng, cho phép theo dõi và duy trì thông tin về các phiên giao dịch.

### **1.3 Kết quả đề tài**

Kết quả của đề tài sẽ bao gồm:

* **Hệ thống WebServer hoạt động hiệu quả**: Hệ thống sẽ cho phép người dùng gửi yêu cầu và nhận phản hồi một cách nhanh chóng, chính xác. Các yêu cầu từ client sẽ được xử lý mà không gặp phải độ trễ lớn.
* **Giao tiếp Client-Server ổn định**: Giao tiếp giữa client và server sẽ được thiết lập thông qua HTTP, với khả năng xử lý các yêu cầu đồng thời, đảm bảo tính ổn định của hệ thống.
* **Hỗ trợ nhiều loại dữ liệu**: WebServer sẽ có khả năng phục vụ nhiều loại dữ liệu khác nhau, cho phép người dùng truy cập các nội dung đa dạng và phong phú.
* **Tài liệu hướng dẫn sử dụng**: Tài liệu sẽ được biên soạn để hướng dẫn người dùng cách sử dụng hệ thống, bao gồm cách gửi yêu cầu và nhận phản hồi từ server.
* **Kết quả thử nghiệm**: Hệ thống sẽ trải qua một loạt các bài kiểm tra để đảm bảo rằng mọi chức năng hoạt động đúng như mong đợi. Thời gian phản hồi sẽ được ghi nhận và đánh giá để đảm bảo tính hiệu quả của WebServer.

## 

## 

## 2. Kế hoạch thực hiện

|  |  |
| --- | --- |
| **Thời gian** | **Nội dung thực hiện** |
| Tuần thứ 3 - Tuần thứ 5. | - Thu thập tài liệu, nghiên cứu về giao thức phân tán  - Phân tích và xác định yêu cầu hệ thống.  - Xây dựng sơ đồ use-case, sơ đồ tuần tự và phác thảo kiến trúc hệ thống. |
| Tuần thứ 6 - Tuần thứ 7 | - Thiết kế giao diện người dùng.  - Phát triển các tính năng server. |
| Tuần thứ 8 - Tuần thứ 9 | - Triển khai mô hình phân tán  - Phát triển chức năng rút tiền ,chuyển tiền ,nạp tiền,… |
| Tuần thứ 10 - Tuần thứ 11 | - Phát triển tính năng quản lý người dùng: đăng ký, đăng nhập.  - Kiểm thử tính năng bảo mật và quyền riêng tư. |
| Tuần thứ 12 - Tuần thứ 13 | - Cải tiến giao diện người dùng và tối ưu hóa trải nghiệm người dùng. |
| Tuần thứ 14 - Tuần thứ 15 | - Đánh giá và tối ưu hóa ứng dụng về hiệu năng của hệ thống  - Hoàn thiện chương trình demo và chuẩn bị tài liệu hướng dẫn sử dụng. |
| Tuần thứ 16 - Tuần thứ 17 | - Viết báo cáo tổng kết và chuẩn bị bài thuyết trình. |

Bảng Kế hoạch thực hiện

# **CHƯƠNG 1 CƠ SỞ LÝ THUYẾT**

## 1.1. Khái niệm về lập trình mạng

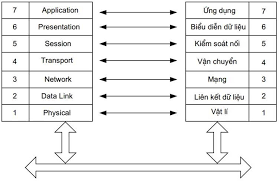
* Lập trình mạng (Network Programming) là hành động sử dụng mã máy tính để viết các chương trình hoặc quy trình có thể giao tiếp với các chương trình hoặc quy trình khác trên mạng. Các lập trình viên sử dụng các ngôn ngữ lập trình, thư viện mã và giao thức khác nhau để thực hiện công việc.
* Chính xác hơn, khả năng lập trình mạng là quá trình sử dụng mã, các khái niệm dựa trên vòng đời phát triển phần mềm và các công cụ khác để làm cho mạng thực hiện các hành động. Hiểu đơn giản, đây là một trong những nhiệm vụ cơ bản được sử dụng để phát triển các ứng dụng, hệ thống doanh nghiệp. Nó có thể bao gồm hệ các chương trình như phần mềm kế toán, nhân sự,… hoặc đến những ứng dụng giải trí như trò chơi, điều khiển…

## 1.2. Các mô hình mạng

### **1.2.1. Mô hình OSI (Open Systems Interconnection)**

Mô hình OSI gồm 7 tầng, được phát triển bởi ISO (International Organization for Standardization), giúp chuẩn hóa các giao thức và công nghệ mạng.

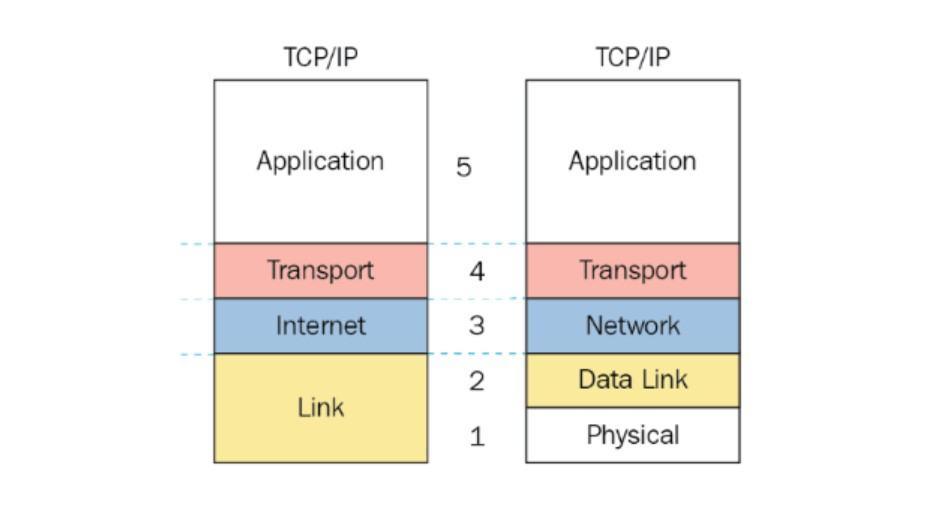
* Tầng 1: Physical (Vật lý): Xử lý các tín hiệu vật lý (bitstream) trên phương tiện truyền dẫn, ví dụ như dây cáp mạng.
* Tầng 2: Data Link (Liên kết dữ liệu): Đảm bảo dữ liệu được truyền đi một cách chính xác giữa hai thiết bị liền kề.
* Tầng 3: Network (Mạng): Xác định đường đi cho dữ liệu từ nguồn đến đích, sử dụng địa chỉ IP.
* Tầng 4: Transport (Vận chuyển): Đảm bảo dữ liệu được truyền tin cậy giữa hai thiết bị, ví dụ qua giao thức TCP hoặc UDP.
* Tầng 5: Session (Phiên): Quản lý phiên giao tiếp giữa hai ứng dụng.
* Tầng 6: Presentation (Trình bày): Chuyển đổi dữ liệu từ định dạng mạng thành định dạng ứng dụng có thể hiểu.
* Tầng 7: Application (Ứng dụng): Cung cấp các dịch vụ cho người dùng cuối (ví dụ HTTP, FTP).



Hình 1. 1 Mô hình OSI

### **1.2.2. Mô hình TCP/IP**

Mô hình TCP/IP có 4 tầng, được xem là hiện thực hóa của mô hình OSI trên Internet.

* Tầng 1: Link (Liên kết): Tương ứng với tầng Physical và Data Link trong OSI.
* Tầng 2: Internet: Tương ứng với tầng Network trong OSI, cung cấp khả năng định tuyến qua giao thức IP.
* Tầng 3: Transport (Vận chuyển): Tương ứng với tầng Transport của OSI, với hai giao thức chính là TCP và UDP.
* Tầng 4: Application (Ứng dụng): Cung cấp các giao thức và dịch vụ ứng dụng, ví dụ HTTP, SMTP, FTP.

Hình 1. 2 Mô hình TCP/IP

### **1.2.3 Các mô hình ứng dụng mạng**

#### **a) Mô hình Client – Server**

Mô tả: Mô hình này có một máy chủ (server) trung tâm cung cấp dịch vụ cho các máy khách (client). Máy chủ chịu trách nhiệm xử lý yêu cầu từ các máy khách và gửi lại kết quả tương ứng. Client chủ yếu là các máy tính người dùng gửi yêu cầu về dữ liệu hoặc dịch vụ đến server.

Cấu trúc:

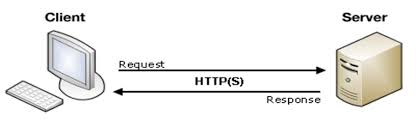
* Server: Lưu trữ dữ liệu và tài nguyên, xử lý yêu cầu từ client.
* Client: Gửi yêu cầu dịch vụ và nhận kết quả từ server.

Ưu điểm:

* Dễ quản lý, triển khai, và bảo mật vì tất cả dữ liệu tập trung ở server.
* Tính bảo mật cao hơn do dữ liệu không phân tán giữa các client.

Nhược điểm:

* Phụ thuộc vào máy chủ: Nếu server bị lỗi hoặc quá tải, toàn bộ hệ thống có thể ngừng hoạt động.
* Tính mở rộng hạn chế do khả năng xử lý của server có giới hạn.



Hình 1. 3 Mô hình Client – Server

#### **b) Mô hình Peer-to-Peer (P2P)**

Mô tả: Trong mô hình này, không có sự phân biệt rõ ràng giữa máy khách và máy chủ. Mọi nút trong mạng đều có vai trò như một "peer" (đồng đẳng), và các peer có thể trực tiếp trao đổi dữ liệu, tài nguyên với nhau mà không cần qua một máy chủ trung tâm.

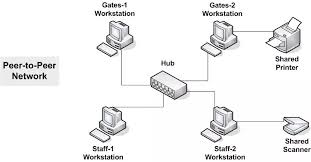
Cấu trúc:Các nút (peer) có quyền ngang nhau và có thể đóng vai trò như máy khách hoặc máy chủ tùy thuộc vào ngữ cảnh.

Ưu điểm:

* Không có sự phụ thuộc vào một server trung tâm, giảm rủi ro "điểm thất bại" (single point of failure).
* Khả năng mở rộng lớn hơn do nhiều nút có thể tham gia vào mạng và cung cấp dịch vụ cho nhau.

Nhược điểm:

* Quản lý, đồng bộ và bảo mật phức tạp hơn vì không có điểm tập trung để quản lý dữ liệu.
* Tính nhất quán dữ liệu có thể khó duy trì giữa các nút.



Hình 1. 4 Mô hình Peer-to-Peer

#### **c) Hệ thống Phân tán**

Mô tả: Hệ thống phân tán là một tập hợp các máy tính (node) liên kết với nhau qua mạng, mỗi node có thể chịu trách nhiệm lưu trữ dữ liệu và xử lý một phần công việc của hệ thống. Các node trong hệ thống phân tán có thể được phân tán tại nhiều địa điểm khác nhau và phối hợp với nhau để cung cấp dịch vụ tổng thể.

Cấu trúc:

* Mỗi node có khả năng hoạt động độc lập, nhưng các node cần phối hợp với nhau để thực hiện các nhiệm vụ phức tạp.
* Dữ liệu và tài nguyên được phân tán giữa các node khác nhau.

Ưu điểm:

* Tính chịu lỗi cao hơn: nếu một hoặc một vài node gặp sự cố, các node khác vẫn có thể tiếp tục hoạt động.
* Khả năng mở rộng mạnh mẽ và hiệu quả cao trong xử lý công việc lớn.

Nhược điểm:

* Quản lý và đồng bộ dữ liệu giữa các node phức tạp, đặc biệt là khi số lượng node tăng lên.
* Các vấn đề về độ trễ mạng và tính nhất quán dữ liệu có thể ảnh hưởng đến hiệu suất.

### **1.2.4 Các giao thức mạng**

#### **a) TCP/IP**

* TCP/IP là một tập hợp các giao thức dùng cho việc truyền thông tin trên mạng Internet. Đây là mô hình cơ bản nhất của mạng Internet, với hai thành phần chính là TCP và IP.
* IP (Internet Protocol): Được sử dụng để định vị và định tuyến các gói tin giữa các thiết bị trên mạng, mỗi thiết bị được xác định bằng một địa chỉ IP.
* TCP (Transmission Control Protocol): Đảm bảo việc truyền tin được thực hiện một cách tin cậy, kiểm soát các gói dữ liệu, đảm bảo chúng đến đúng thứ tự và phát hiện lỗi.

#### **b) UDP**

* UDP là một giao thức không kết nối và không đáng tin cậy như TCP. Dữ liệu được gửi dưới dạng các gói tin (datagram) mà không có cơ chế kiểm tra tính đúng đắn hay mất gói.
* Ưu điểm: Tốc độ nhanh, sử dụng cho các ứng dụng cần thời gian thực như streaming video, audio, hoặc game trực tuyến.
* Nhược điểm: Không đảm bảo dữ liệu được gửi đến đích đúng thứ tự hoặc không bị mất gói.

#### **c) HTTP/HTTPS**

* HTTP: Giao thức sử dụng để trao đổi thông tin giữa các máy khách (client) và máy chủ (server) trên web. HTTP hoạt động dựa trên yêu cầu (request) từ client và phản hồi (response) từ server.
* HTTPS: Là phiên bản bảo mật của HTTP, trong đó dữ liệu được mã hóa bằng SSL/TLS để bảo mật thông tin khi truyền trên Internet.
* Ứng dụng: HTTP/HTTPS là nền tảng cho việc truy cập và truyền thông tin trên web, như truy cập trang web, gửi và nhận dữ liệu từ máy chủ.

#### **d) SMTP**

* SMTP là giao thức chuẩn dùng để gửi email qua mạng Internet. Nó chủ yếu chịu trách nhiệm truyền tải email từ client đến server và từ server này đến server khác.
* Ưu điểm: Đơn giản và hiệu quả cho việc truyền email.
* Nhược điểm: SMTP không mã hóa email, vì vậy thông tin có thể bị chặn trên đường truyền.

#### **e) RMI**

Mô tả: RMI là một kỹ thuật cho phép một đối tượng Java trên một máy tính (máy khách) gọi phương thức của đối tượng từ xa (máy chủ) trên một máy tính khác, giống như đang gọi một phương thức cục bộ.

Đặc điểm:

* RMI sử dụng cơ chế truyền dữ liệu giữa các đối tượng trong hệ thống phân tán, giúp lập trình viên tương tác với các đối tượng từ xa mà không cần lo lắng về chi tiết giao tiếp mạng.
* Dữ liệu được truyền dưới dạng tuần tự hóa (serialization), cho phép truyền các đối tượng Java phức tạp qua mạng.
* Hỗ trợ các tính năng như bảo mật (SSL), khả năng chịu lỗi, và cung cấp nhiều mô hình triển khai trong các môi trường phân tán.

Ưu điểm:

* Đơn giản hóa việc lập trình hệ thống phân tán vì các cuộc gọi từ xa trong RMI có thể được thực hiện như các cuộc gọi cục bộ.
* Java cung cấp các công cụ tích hợp sẵn để dễ dàng triển khai các dịch vụ RMI.

Nhược điểm:

* Phụ thuộc vào Java, cả client và server phải chạy trên môi trường Java.
* Không linh hoạt như một số giải pháp mới như REST hoặc gRPC, đặc biệt khi tích hợp với các hệ thống không sử dụng Java.

Ứng dụng:

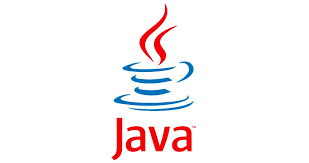
* RMI được sử dụng phổ biến trong các hệ thống Java phân tán để quản lý các đối tượng từ xa, ví dụ như trong các hệ thống ngân hàng, quản lý giao dịch, hay các hệ thống liên quan đến dữ liệu thời gian thực.

Ví dụ: Một hệ thống ATM phân tán sử dụng RMI để cho phép các client trên các máy khác nhau tương tác với máy chủ để thực hiện các giao dịch như rút tiền, kiểm tra số dư.

## 1.3 Công nghệ sử dụng

### **1.3.1 Java**

Java là một ngôn ngữ lập trình hướng đối tượng, mạnh mẽ và đa nền tảng, được phát triển bởi Sun Microsystems (nay thuộc Oracle Corporation) vào năm 1995. Java nổi tiếng với khẩu hiệu "Viết một lần, chạy mọi nơi" (Write Once, Run Anywhere - WORA), nghĩa là mã Java có thể chạy trên nhiều nền tảng khác nhau mà không cần thay đổi, miễn là máy đó có Java Virtual Machine (JVM).



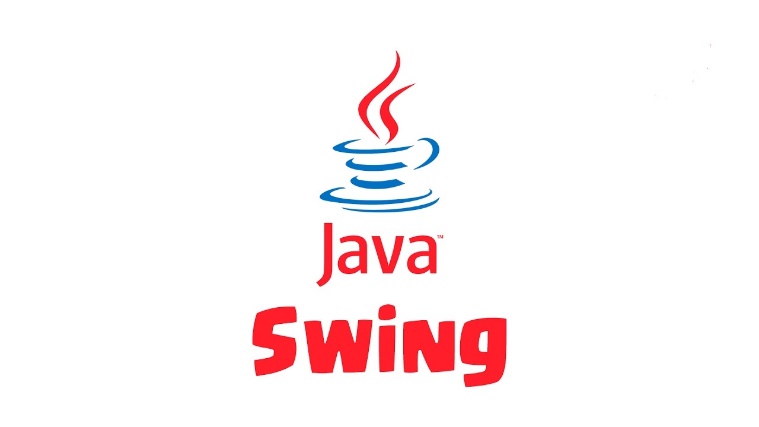
#### **Môi trường**

IntelliJ IDEA là một môi trường phát triển tích hợp (IDE - Integrated Development Environment) mạnh mẽ, mã nguồn mở, được phát triển bởi JetBrains. IntelliJ IDEA hỗ trợ nhiều ngôn ngữ lập trình, nhưng nổi bật nhất là Java. Ngoài Java, nó còn hỗ trợ các ngôn ngữ khác như Kotlin, Groovy, Scala, HTML5, JavaScript, và Python. IntelliJ được biết đến với nhiều tính năng thông minh như tự động hoàn thành mã, phân tích mã, và hỗ trợ quản lý dự án, giúp lập trình viên làm việc hiệu quả hơn.

#### **Thư viện**

MySQL Connector/J là thư viện JDBC (Java Database Connectivity) chính thức của MySQL, cho phép các ứng dụng Java tương tác với cơ sở dữ liệu MySQL. Đây là một công cụ mạnh mẽ, dễ sử dụng và hỗ trợ đầy đủ các tính năng của MySQL, giúp xây dựng và quản lý các ứng dụng cơ sở dữ liệu một cách hiệu quả.

Java Swing là một thư viện (framework) trong Java được sử dụng để phát triển giao diện người dùng (GUI - Graphical User Interface) cho các ứng dụng desktop. Swing là một phần của Java Foundation Classes (JFC) và cung cấp một bộ các thành phần GUI mạnh mẽ để tạo ra các ứng dụng với giao diện hấp dẫn và dễ sử dụng.



### **1.3.2 MySQL**

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở (RDBMS) nổi tiếng, được phát triển bởi Oracle Corporation. Đây là một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất trên thế giới, đặc biệt trong lĩnh vực phát triển ứng dụng web và dịch vụ.

**MySQL Workbench** là một công cụ phát triển và quản lý cơ sở dữ liệu mã nguồn mở dành cho MySQL. Nó cung cấp một giao diện đồ họa thân thiện giúp lập trình viên và quản trị viên cơ sở dữ liệu dễ dàng quản lý và thiết kế cơ sở dữ liệu. MySQL Workbench tích hợp nhiều tính năng mạnh mẽ, bao gồm thiết kế sơ đồ cơ sở dữ liệu, truy vấn SQL, quản lý phiên bản và sao lưu dữ liệu.



**Đặc điểm của MySQL Workbench**:

* **Thiết kế đồ họa**: Cung cấp công cụ thiết kế cơ sở dữ liệu trực quan cho phép người dùng tạo và chỉnh sửa sơ đồ cơ sở dữ liệu mà không cần phải viết mã SQL.
* **Quản lý cơ sở dữ liệu**: Cho phép người dùng dễ dàng quản lý người dùng, quyền truy cập và các phiên bản của cơ sở dữ liệu.
* **Truy vấn và phân tích**: Hỗ trợ viết và chạy các truy vấn SQL, cũng như phân tích kết quả với công cụ trực quan.

**Lợi ích**:

* **Dễ sử dụng**: Giao diện đồ họa giúp người dùng không cần phải có kiến thức sâu về SQL để quản lý cơ sở dữ liệu.
* **Tiết kiệm thời gian**: Các công cụ tự động hóa và tính năng trực quan giúp tăng tốc quá trình thiết kế và quản lý cơ sở dữ liệu.
* **Hỗ trợ đa nền tảng**: MySQL Workbench có thể chạy trên nhiều hệ điều hành khác nhau, bao gồm Windows, macOS và Linux.
* **Tích hợp nhiều tính năng**: Cung cấp nhiều công cụ hỗ trợ như so sánh cơ sở dữ liệu, quản lý dự án và phát hiện vấn đề hiệu suất.

# **Chương 2: Phân tích & Thiết kế Hệ thống**

## 2.1. Giới thiệu bài toán của đồ án

Trong bối cảnh phát triển công nghệ thông tin hiện nay, việc tạo ra các ứng dụng web đang trở thành một xu hướng phổ biến. Đồ án này tập trung vào việc xây dựng một WebServer cơ bản, có khả năng lắng nghe và xử lý các yêu cầu HTTP từ client. Mục tiêu là tạo ra một môi trường nơi client, như trình duyệt web, có thể gửi yêu cầu để truy cập các tài nguyên trên server. Khi người dùng nhập URL vào trình duyệt và nhấn Enter, một yêu cầu HTTP sẽ được gửi đến server, nơi mà yêu cầu này sẽ được phân tích và xử lý, trả về nội dung phù hợp cho client.

WebServer không chỉ đơn thuần là một chương trình máy tính; nó còn là cầu nối giữa người dùng và dữ liệu. Điều này có nghĩa là server cần phải xử lý các yêu cầu một cách hiệu quả và đáng tin cậy để đảm bảo rằng người dùng có được trải nghiệm tốt nhất khi sử dụng ứng dụng web.

## 2.2. Yêu cầu hệ thống

### **2.2.1. Yêu cầu chức năng**

* **Server:**
  + **Lắng nghe các yêu cầu HTTP từ client:** Server sẽ liên tục ở trạng thái chờ và sẵn sàng nhận yêu cầu từ client. Điều này yêu cầu server phải hoạt động liên tục và có khả năng quản lý nhiều kết nối đồng thời.
  + **Phân tích và xử lý các yêu cầu GET:** Server phải có khả năng xử lý các yêu cầu GET từ client, bao gồm việc xác định tài nguyên được yêu cầu và trả về nội dung chính xác.
  + **Gửi phản hồi:** Server cần phải trả về tài nguyên dưới dạng HTML, hình ảnh hoặc các tệp tĩnh khác một cách nhanh chóng và chính xác.
  + **Ghi log yêu cầu:** Server sẽ ghi lại thông tin về từng yêu cầu mà nó nhận được, bao gồm địa chỉ IP của client, phương thức HTTP, thời gian gửi yêu cầu và URL yêu cầu. Điều này giúp theo dõi hoạt động của server và phục vụ cho việc phân tích và bảo trì hệ thống sau này.
* **Client:**
  + **Gửi yêu cầu HTTP đến server:** Client cần có khả năng gửi các yêu cầu đến server thông qua giao thức HTTP. Việc này có thể được thực hiện thông qua các trình duyệt web hoặc các công cụ kiểm tra API.
  + **Nhận và hiển thị phản hồi từ server:** Client sẽ nhận phản hồi từ server và hiển thị nó trên giao diện người dùng, thường là trình duyệt web. Người dùng cần có khả năng tương tác với nội dung một cách mượt mà.

### **2.2.2. Yêu cầu phi chức năng**

* **Thời gian phản hồi:** Hệ thống cần phải đảm bảo rằng thời gian phản hồi cho các yêu cầu đơn giản là dưới 1 giây. Điều này đặc biệt quan trọng trong bối cảnh người dùng ngày càng mong đợi tốc độ cao và trải nghiệm liền mạch.
* **Khả năng mở rộng:** Hệ thống cần có khả năng mở rộng để xử lý nhiều yêu cầu đồng thời từ nhiều client khác nhau mà không làm giảm hiệu suất. Điều này có thể đạt được bằng cách sử dụng các kỹ thuật lập trình bất đồng bộ hoặc đa luồng.
* **Tính bảo mật:** Mặc dù trong giai đoạn phát triển này chưa yêu cầu mã hóa, hệ thống cần được thiết kế sao cho có thể tích hợp các biện pháp bảo mật trong tương lai như mã hóa dữ liệu và xác thực người dùng.
* **Khả năng phục hồi:** Hệ thống phải có khả năng xử lý lỗi và cung cấp phản hồi thích hợp khi có sự cố xảy ra, như tài nguyên không tồn tại hoặc lỗi server.

### **2.2.3 Các tác nhân**

Dựa trên mô tả bài toán, các tác nhân chính của hệ thống có thể được xác định như sau:

* **Client (Người dùng Web)**: Đây là người dùng truy cập vào hệ thống thông qua trình duyệt web. Họ có thể thực hiện các yêu cầu HTTP (GET, POST) để truy cập các tài nguyên hoặc gửi dữ liệu.
* **Admin (Quản trị viên)**: Đây là người quản lý hệ thống WebServer, có khả năng giám sát và quản lý hệ thống, xem nhật ký hoạt động và xử lý các vấn đề phát sinh.

## 2.3. Sơ đồ Use Case (UC)

### **2.3.1. Xác định các Use Case**

Các tác nhân và chức năng của hệ thống được thể hiện qua các Use Case (UC) dưới đây:

**Tác nhân "Client" (Người dùng Web) có các UC sau:**

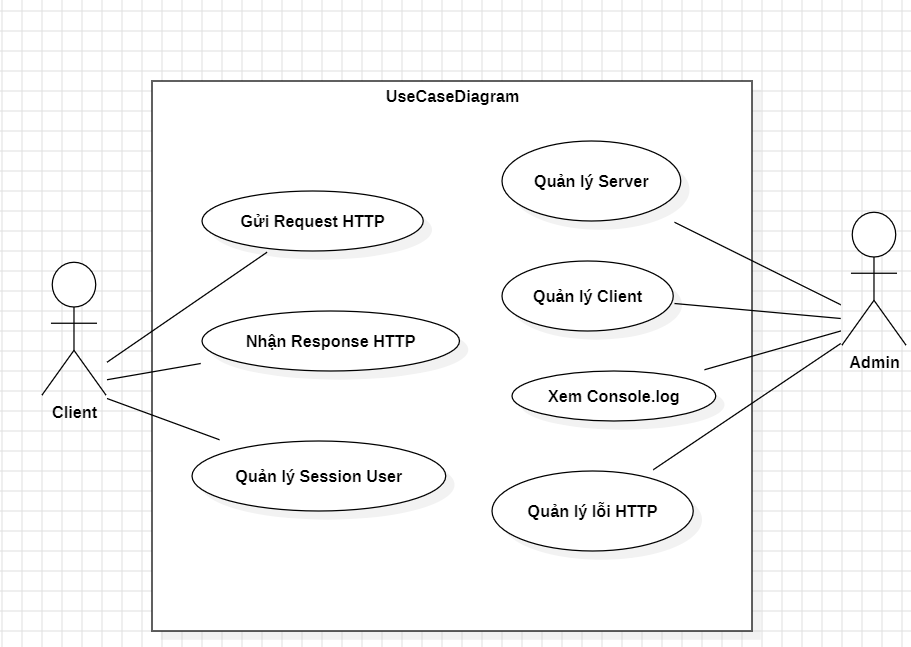
* **UC1: Gửi yêu cầu HTTP (GET)**:
  + Mô tả: Người dùng gửi yêu cầu truy cập trang web (trang HTML, tài nguyên hình ảnh, CSS) bằng cách nhập URL trên trình duyệt.
* **UC2: Nhận phản hồi từ WebServer**:
  + Mô tả: Sau khi gửi yêu cầu, người dùng nhận lại phản hồi từ WebServer dưới dạng trang HTML, tài nguyên hình ảnh, hoặc mã lỗi HTTP.
* **UC3: Gửi yêu cầu POST (Form dữ liệu)**:
  + Mô tả: Người dùng có thể gửi thông tin từ form (ví dụ: đăng ký, đăng nhập) qua yêu cầu POST tới WebServer.
* **UC4: Quản lý session người dùng**:
  + Mô tả: WebServer duy trì phiên làm việc của người dùng để lưu trữ trạng thái đăng nhập hoặc các hoạt động của người dùng trong quá trình truy cập.

**Tác nhân "Admin" (Quản trị viên) có các UC sau:**

* **UC5: Quản lý phiên bản WebServer**:
  + Mô tả: Admin có thể giám sát và quản lý phiên bản của WebServer, bao gồm việc xem nhật ký (log), cập nhật và giám sát hoạt động của server.
* **UC6: Xem nhật ký (Log)**:
  + Mô tả: Admin có thể truy cập và xem nhật ký các yêu cầu và phản hồi của WebServer để giám sát và xử lý các vấn đề về hiệu suất hoặc lỗi hệ thống.
* **UC7: Quản lý lỗi HTTP**:
  + Mô tả: Admin có thể theo dõi và xử lý các lỗi HTTP xuất hiện trong quá trình xử lý yêu cầu từ người dùng (ví dụ: lỗi 404, lỗi 500).

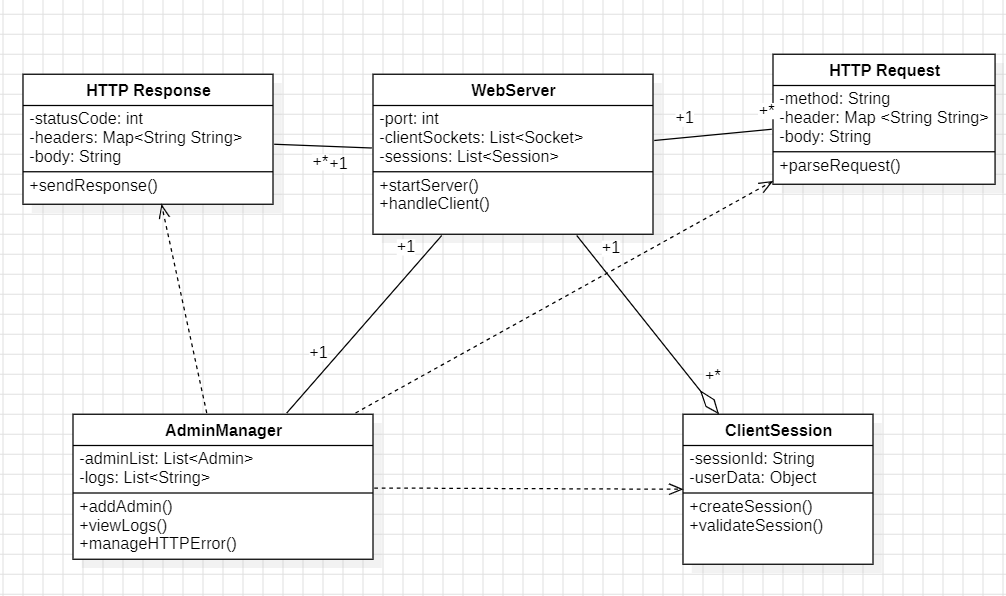
### **2.3.2. Biểu đồ Use Case**

Usecase tổng quát:



Hình 2. 1 Usecase diagram

## 2.4 Mô hình hóa cấu trúc

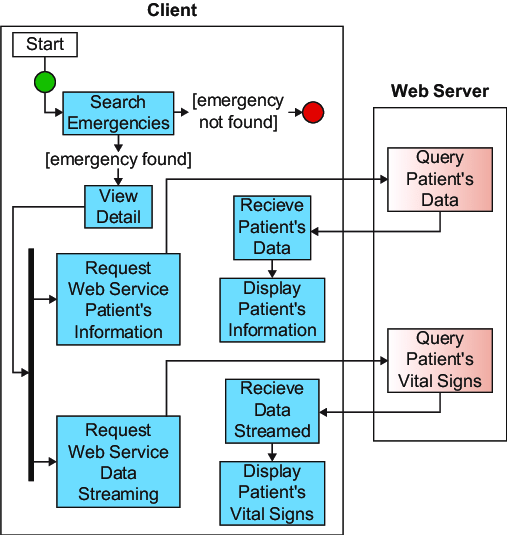


Hình 2. 2 Class diagram

## 2.5 Mô hình hóa hành vi

### **2.5.1 Biểu đồ hoạt động**

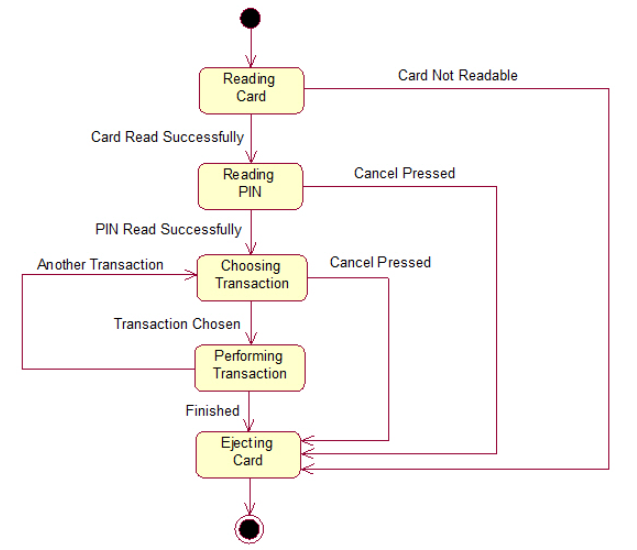
* [Activity diagram for data streaming from the web server to the WebServer:](https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FActivity-diagram-for-data-streaming-from-the-web-server-to-the-client_fig4_315858332&psig=AOvVaw1BDoTUp9xeeD8l90sZ-_zt&ust=1730440386568000&source=images&cd=vfe&opi=89978449&ved=2ahUKEwj__dH19reJAxXjgVYBHbZrAC4Qjhx6BAgAEBo" \t "_blank)



Hình 2. 3 Activities diagram

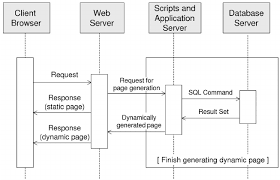
### **2.5.2 Biểu đồ trạng thái**

* Quy Trình Giao Dịch Tại Máy ATM



Hình 2. 4 State diagram

### **2.5.3 Biểu đồ trình tự**



Hình 2. 5 Sequence diagram

### **2.5.4. Đặc tả Use Case chính**

**Use Case 1: Gửi yêu cầu HTTP (GET)**

* **Mô tả:** Đây là yêu cầu cơ bản của hệ thống WebServer. Người dùng nhập URL trên trình duyệt, và yêu cầu này được gửi tới WebServer thông qua giao thức HTTP. WebServer sẽ tìm kiếm và phản hồi lại tài nguyên tương ứng.
* **Dòng sự kiện chính:**
  1. Người dùng nhập URL và nhấn Enter.
  2. Trình duyệt gửi yêu cầu HTTP GET tới WebServer.
  3. WebServer nhận yêu cầu và tìm kiếm tài nguyên.
  4. Nếu tài nguyên tồn tại, WebServer gửi về phản hồi HTTP 200 kèm tài nguyên (HTML, CSS, hình ảnh).
  5. Nếu tài nguyên không tồn tại, WebServer phản hồi mã lỗi 404.

**Use Case 2: Nhận phản hồi HTTP**

* **Mô tả:** Sau khi WebServer nhận yêu cầu từ người dùng, nó sẽ gửi phản hồi lại. Phản hồi này có thể là một trang HTML, tài nguyên hình ảnh, hoặc mã lỗi nếu có sự cố xảy ra.
* **Dòng sự kiện chính:**
  1. WebServer xử lý yêu cầu.
  2. WebServer gửi tài nguyên (HTML, CSS, hình ảnh) hoặc mã lỗi cho người dùng.
  3. Trình duyệt hiển thị nội dung phản hồi cho người dùng**.**

**Use Case 5: Quản lý phiên bản WebServer**

* **Mô tả:** Admin giám sát và quản lý các phiên bản của WebServer, bao gồm việc kiểm tra trạng thái hoạt động và xử lý sự cố nếu có.
* **Dòng sự kiện chính:**
  1. Admin truy cập trang quản lý WebServer.
  2. Admin kiểm tra phiên bản và nhật ký hoạt động.
  3. Admin thực hiện các thao tác quản lý (cập nhật, khởi động lại nếu cần).

**Use Case 6: Xem nhật ký**

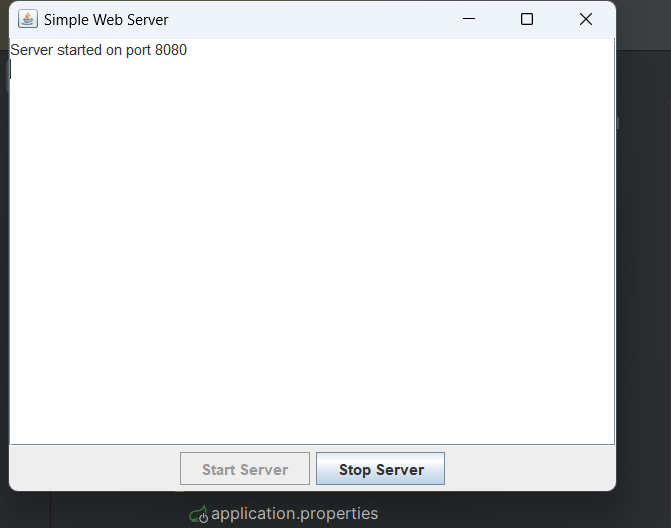
* **Mô tả:** Admin có thể xem các bản ghi hoạt động của WebServer để giám sát hệ thống và xử lý các sự cố nếu có lỗi xảy ra.
* **Dòng sự kiện chính:**
  1. Admin đăng nhập vào trang quản lý.
  2. Admin xem các nhật ký hoạt động và phân tích lỗi**.**

# **Chương 3: Thực hiện Chương trình (Giao diện, Màn hình chính)**

## 3.1. Giao diện Server

Giao diện của server sẽ được triển khai trên dòng lệnh (CLI), cho phép người dùng theo dõi hoạt động của server một cách dễ dàng. Dưới đây là các thành phần của giao diện server:

* **Thông báo trạng thái:** Khi server khởi động thành công, nó sẽ hiển thị một thông báo trên màn hình, chẳng hạn như "Server đang chạy tại địa chỉ http://localhost:8080".
* **Log yêu cầu:** Mỗi khi server nhận được yêu cầu từ client, nó sẽ ghi lại thông tin chi tiết, bao gồm:
  + **IP của client:** Địa chỉ IP từ nơi yêu cầu được gửi đến.
  + **Thời gian gửi yêu cầu:** Thời gian chính xác mà yêu cầu được gửi.
  + **Phương thức HTTP:** Phương thức mà client đã sử dụng (GET hoặc POST).
  + **URL yêu cầu:** Địa chỉ URL mà client đã yêu cầu.
  + **Mã trạng thái phản hồi:** Mã trạng thái mà server đã gửi về cho client (ví dụ: 200 OK, 404 Not Found).



Hình 3.1 Giao Diện Server

## 3.2. Hướng dẫn sử dụng và giới thiệu kết quả

### **3.2.1. Khởi động Server**

Để khởi động server, người dùng cần thực hiện các bước sau:

1. **Mở terminal hoặc command prompt.**
2. **Điều hướng tới thư mục chứa chương trình server:** Sử dụng lệnh cd để chuyển đến thư mục chứa mã nguồn của server.
3. **Chạy lệnh:** Gõ lệnh java WebServer để khởi động server. Khi server hoạt động, nó sẽ hiển thị thông báo xác nhận trên màn hình.
4. **Xác minh trạng thái:** Người dùng có thể kiểm tra trạng thái của server bằng cách mở trình duyệt và truy cập địa chỉ http://localhost:8080.

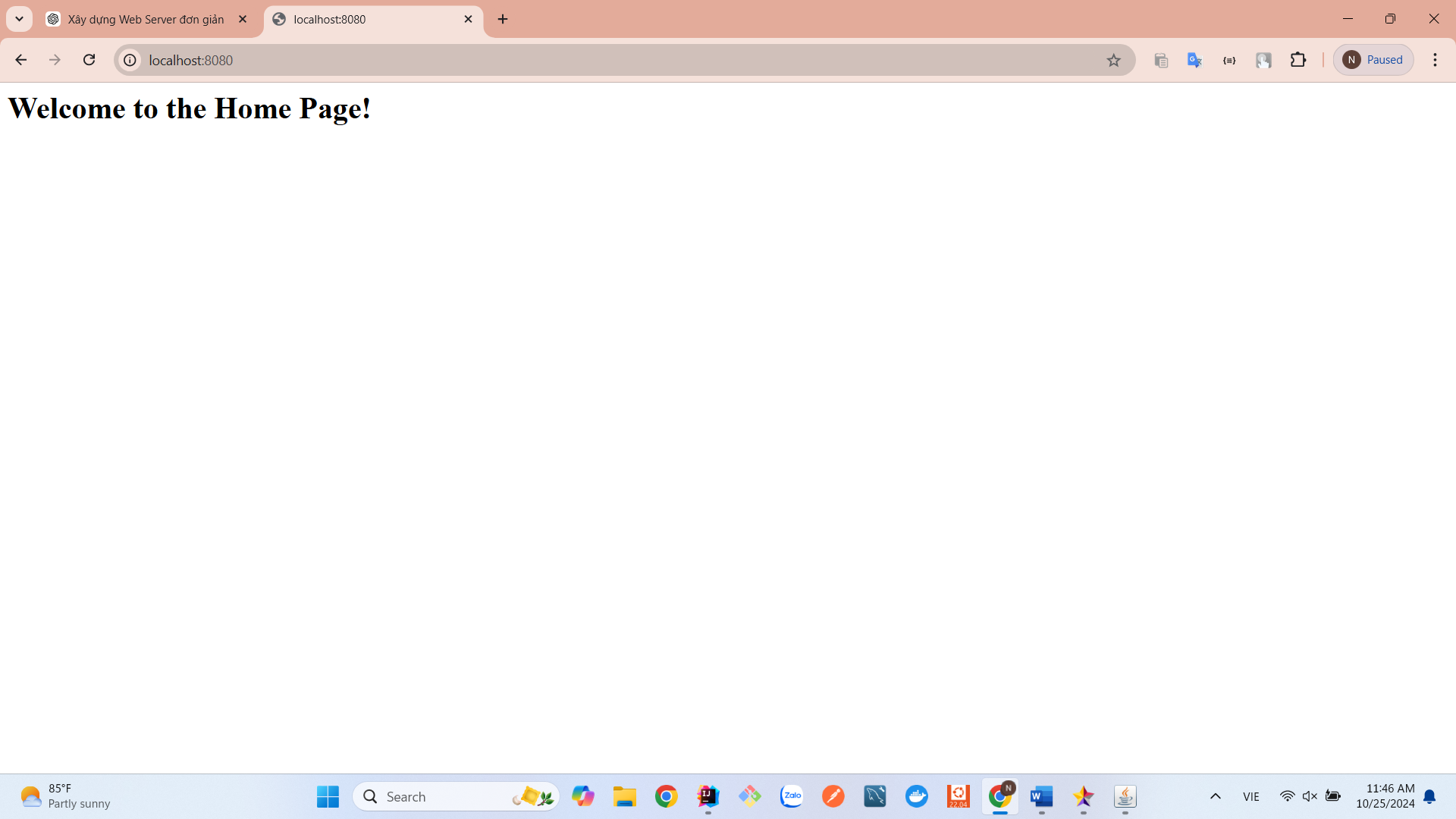
### **3.2.2. Gửi yêu cầu từ Client**

Client có thể gửi yêu cầu tới server bằng cách mở trình duyệt web và nhập URL tương ứng với địa chỉ IP và cổng mà server đang chạy. Ví dụ: http://localhost:8080/index.html. Trình duyệt sẽ gửi yêu cầu HTTP GET tới server, và server sẽ xử lý yêu cầu này.

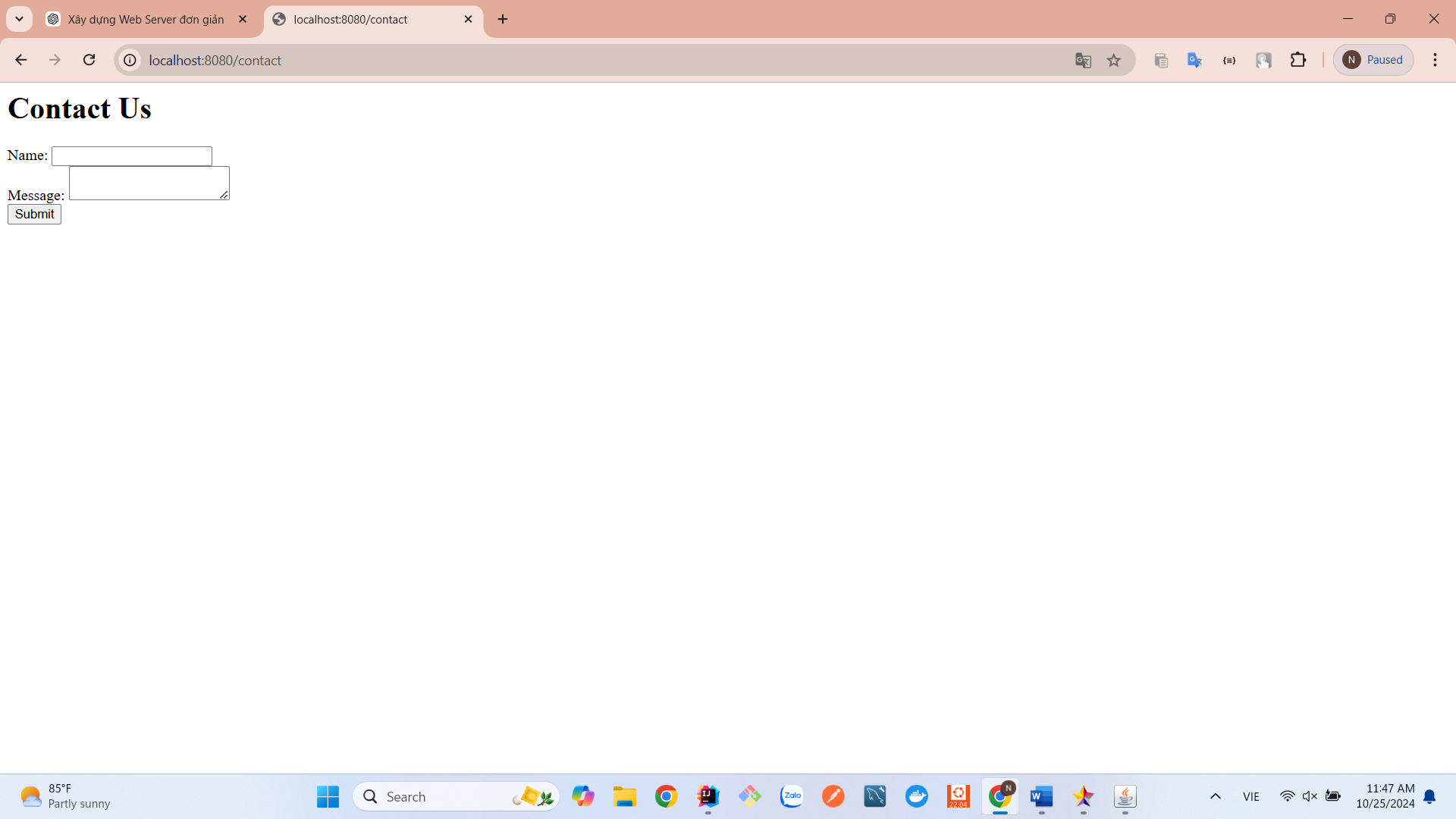
* **Ví dụ yêu cầu:** Khi người dùng nhập http://localhost:8080/home, server sẽ tìm kiếm tệp home.html trong thư mục chứa tài nguyên và trả về nội dung nếu tệp tồn tại.

### **3.2.3. Xem kết quả phản hồi**

* **Nếu tài nguyên tồn tại:** Server sẽ trả về nội dung của tệp HTML tương ứng với yêu cầu. Client sẽ hiển thị nội dung này trên trình duyệt.
* **Nếu tài nguyên không tồn tại:** Server sẽ trả về mã lỗi 404 Not Found cùng với một thông báo cụ thể. Client sẽ hiển thị thông báo lỗi này, giúp người dùng biết rằng tài nguyên không thể tìm thấy.



Hình 3.2 Giao Diện Home

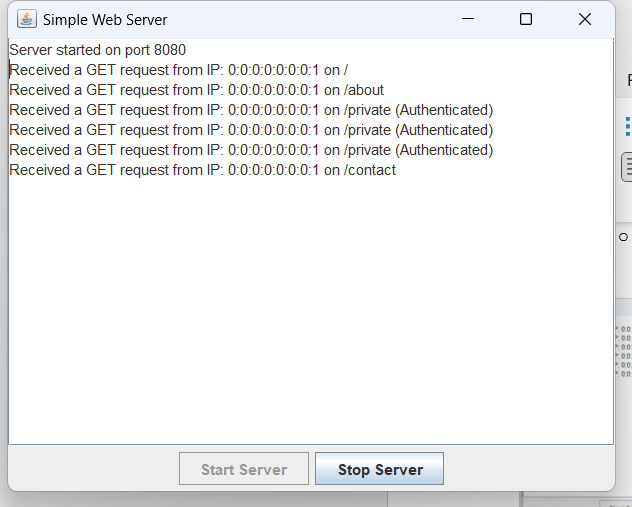


Hình 3.3 Giao Diện Trang Contact

## 3.3. Kết quả thử nghiệm

Trong quá trình phát triển, hệ thống đã được thử nghiệm qua nhiều tình huống khác nhau để đảm bảo tính ổn định và hiệu suất:

* **Thử nghiệm yêu cầu GET:**
  + **Kịch bản:** Người dùng gửi yêu cầu đến một tệp HTML tồn tại.
  + **Kết quả:** Server đã trả về nội dung chính xác trong vòng chưa đầy 1 giây, xác nhận rằng hệ thống hoạt động hiệu quả.
* **Thử nghiệm yêu cầu không tồn tại:**
  + **Kịch bản:** Người dùng nhập URL không hợp lệ.
  + **Kết quả:** Server trả về mã lỗi 404 kèm theo thông báo rõ ràng về sự cố, cho thấy hệ thống có khả năng xử lý lỗi hiệu quả.
* **Thử nghiệm đa người dùng:**
  + **Kịch bản:** Nhiều client gửi yêu cầu đồng thời.
  + **Kết quả:** Server đã xử lý đồng thời các yêu cầu mà không gặp phải độ trễ lớn, chứng minh khả năng mở rộng của hệ thống.



Hình 3.4 Phản Hồi Bên Server

# **Kết luận và Hướng phát triển**

## 1. Kết luận

Thông qua đồ án này, chúng tôi đã xây dựng thành công một WebServer đơn giản có khả năng xử lý các yêu cầu HTTP GET từ client. Hệ thống hoạt động ổn định, cho phép người dùng gửi yêu cầu và nhận phản hồi một cách nhanh chóng và chính xác. Quá trình phát triển không chỉ giúp chúng tôi hiểu rõ hơn về giao thức HTTP mà còn về mô hình Client-Server, cách thức quản lý tài nguyên và giao tiếp qua mạng. Kết quả cho thấy rằng hệ thống hoạt động hiệu quả, có thể xử lý nhiều yêu cầu mà không gặp phải độ trễ lớn.

Hơn nữa, việc ghi log yêu cầu không chỉ giúp theo dõi hoạt động của server mà còn là cơ sở để cải thiện hiệu suất và bảo mật trong tương lai. Các thử nghiệm đã được thực hiện để đảm bảo rằng server có thể xử lý nhiều tình huống khác nhau mà không gặp phải sự cố.

## 2. Hướng phát triển

Trong tương lai, chúng tôi dự kiến mở rộng và cải thiện hệ thống với các tính năng sau:

* **Xử lý yêu cầu POST:** Cho phép client gửi dữ liệu tới server, phục vụ cho các ứng dụng web yêu cầu gửi dữ liệu như biểu mẫu đăng ký, đăng nhập, v.v. Điều này sẽ cần xây dựng một cách xử lý và lưu trữ dữ liệu hiệu quả.
* **Hỗ trợ HTTPS:** Tích hợp chứng chỉ SSL để mã hóa thông tin giữa client và server, tăng cường tính bảo mật cho dữ liệu truyền tải. Điều này rất quan trọng trong bối cảnh hiện nay khi mà bảo mật thông tin người dùng ngày càng được chú trọng.
* **Hỗ trợ nội dung động:** Server có thể xử lý các yêu cầu động từ cơ sở dữ liệu, cho phép tạo ra nội dung tùy chỉnh dựa trên thông tin người dùng. Điều này có thể bao gồm việc tích hợp với các hệ quản trị cơ sở dữ liệu như MySQL hoặc MongoDB.
* **Tích hợp công nghệ WebSocket:** Cho phép giao tiếp hai chiều giữa client và server, nâng cao trải nghiệm người dùng với khả năng cập nhật nội dung theo thời gian thực. Điều này đặc biệt hữu ích cho các ứng dụng yêu cầu tương tác liên tục, chẳng hạn như chat hay thông báo thời gian thực.
* **Cải thiện giao diện người dùng:** Phát triển giao diện người dùng đồ họa cho client, giúp dễ dàng tương tác hơn và cải thiện trải nghiệm người dùng tổng thể. Sử dụng các công nghệ như HTML5, CSS3 và JavaScript sẽ giúp tạo ra một giao diện thân thiện và hiện đại hơn.

# 

# **Tài liệu tham khảo**

1. Slide và giáo trình môn Lập Trình Mạng.
2. Sách "Computer Networking: A Top-Down Approach" của James F. Kurose và Keith W. Ross.
3. Tài liệu chính thức của Java về java.net package.
4. Tham khảo đồ án của sinh viên khóa trước về chủ đề lập trình mạng.
5. Các website:
   * <https://docs.oracle.com/javase/8/docs/api/java/net/>
   * <https://developer.mozilla.org/en-US/docs/Web/HTTP>