**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue and white logo

Description automatically generated

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

**Đề tài**

**Xây dựng và triển khai Website trò chơi bắn xe tăng nhiều người theo mô hình Client-Server**

**GIẢNG VIÊN HƯỚNG DẪN:**

**ThS. Trần Hồ Thuỷ Tiên.**

**SINH VIÊN THỰC HIỆN:**

|  |  |  |
| --- | --- | --- |
| **1. Nguyễn Thành Trung** | **MSSV: 102220100** | **Nhóm: 22N12A** |
| **2. Văn Thị Kim Dung** | **MSSV: 102220100** | **Nhóm: 22N12A** |

Đà Nẵng dd/mm/yyyyMỤC LỤC

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 8](#_Toc183510291)

[1.1 Mô hình Client-Server. 8](#_Toc183510292)

[1.2 HTTP/ HTTPS là gì? 8](#_Toc183510293)

[1.3 Request và Response. 9](#_Toc183510294)

[1.4 Status Code trong Response. 10](#_Toc183510295)

[1.5 Rest API. 11](#_Toc183510296)

[1.6 Express NodeJS Frameworks. 11](#_Toc183510297)

[1.7 Template Engine Handlebars. 11](#_Toc183510298)

[1.8 Socket.IO 12](#_Toc183510299)

[1.8.1 Khái niệm 12](#_Toc183510300)

[1.8.2 Các tính năng của Socket.IO 13](#_Toc183510301)

[1.9 Hệ quản trị cơ sở dữ liệu MongoDB 14](#_Toc183510302)

[CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG 16](#_Toc183510303)

[2.1. Khái quát đề tài. 16](#_Toc183510304)

[2.2. Lập trình mạng giữa Client và Server. 17](#_Toc183510305)

[2.2.1. Xây dựng hệ thống phía Client và Server. 17](#_Toc183510306)

[2.2.2. Cách thức giao tiếp giữa Client và Server. 17](#_Toc183510307)

[2.3. Biểu đồ Usecase. 19](#_Toc183510308)

[2.3.1. Biểu đồ Usecase tổng quát. 19](#_Toc183510309)

[2.3.2. Biểu đồ usecase phân rã. 20](#_Toc183510310)

[2.3.3. Đặc tả use case. 22](#_Toc183510311)

[2.4 EngineGame. 25](#_Toc183510312)

[2.4.1 EngineGameServer 25](#_Toc183510313)

[2.4.2 EngineGameClient 26](#_Toc183510314)

[2.5 Cơ chế đồng bộ trạng thái trò chơi giữa các người dùng. 27](#_Toc183510315)

[2.6. Sự kiện bàn phím và chuột của người dùng. 28](#_Toc183510316)

[2.6.1. Các sự kiện chính được để người dùng điều khiển xe tăng. 28](#_Toc183510317)

[2.6.2. Các phím được sử dụng để điều khiển xe tăng. 29](#_Toc183510318)

[2.6.3. Các lệnh được gửi lên Server. 29](#_Toc183510319)

[2.7. Thuật toán va chạm. 30](#_Toc183510320)

[**2.7.1. Va chạm của xe tăng và các vật cản.** 30](#_Toc183510321)

[**2.7.2. Va chạm của đạn và các vật cản.** 30](#_Toc183510322)

[2.8. Cơ sở dữ liệu. 32](#_Toc183510323)

[2.8.1. Biểu đồ cơ sở dữ liệu. 32](#_Toc183510324)

[2.8.2. Mô tả chi tiết các collection trong cơ sở dữ liệu. 32](#_Toc183510325)

[2.9. Mã hóa mật khẩu. 34](#_Toc183510326)

[2.9.1. bcrypt: Thuật toán mã hóa mật khẩu 34](#_Toc183510327)

[**2.9.2 Quá trình mã hóa và xác thực mật khẩu** 34](#_Toc183510328)

[**2.9.3 Ưu điểm của việc sử dụng bcrypt** 35](#_Toc183510329)

[**2.10. Cơ chế lưu và phát lại trận đấu đã chơi.** 35](#_Toc183510330)

[2.10.1 Cơ chế lưu trận đấu. 36](#_Toc183510331)

[**2.10.2 Phát lại trận đấu đã chơi.** 36](#_Toc183510332)

[2.11. Cơ chế duy trì đăng nhập bằng Session. 38](#_Toc183510333)

[2.11.1. Cách Session hoạt động. 38](#_Toc183510334)

[2.11.2. Các tính năng của Session. 38](#_Toc183510335)

[CHƯƠNG 3. DEMO ỨNG DỤNG VÀ ĐÁNH GIÁ KẾT QUẢ 40](#_Toc183510336)

[3.1. Demo ứng dụng. 40](#_Toc183510337)

[3.2 Đánh giá kết quả 40](#_Toc183510338)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 41](#_Toc183510339)

[TÀI LIỆU THAM KHẢO 42](#_Toc183510340)

[PHỤ LỤC 43](#_Toc183510341)

DANH SÁCH HÌNH VẼ

[Hình 1. Mô hình Client-Server 8](#_Toc183348934)

[Hình 2. Giao thức HTTP/HTTPS 8](#_Toc183348935)

[Hình 3. HTTP Status Codes 10](#_Toc183348936)

[Hình 4. Giao tiếp bằng SocketIO 12](#_Toc183348937)

[Hình 5. Mô hình truyền tin giữa Client-Server trong SocketIO 13](#_Toc183348938)

[Hình 6. Sender và Receiver trong SocketIO 13](#_Toc183348939)

[Hình 7. Room trong SocketIO 14](#_Toc183348940)

[Hình 8. Cơ sở dữ liệu MongoDB 14](#_Toc183348941)

[Hình 9. Collection và document trong MongoDB 15](#_Toc183348942)

[Hình 10. Kiến trúc dự án tổng quát 16](#_Toc183348943)

[Hình 11. Connection được tạo ra giữa Client và Server 17](#_Toc183348944)

[Hình 12. Giao tiếp hai chiều giữa Client và Server sử dụng Socket.io 18](#_Toc183348945)

[Hình 13. Truyền thông cho tất cả Client 18](#_Toc183348946)

[Hình 14. Các Client tham gia vào room trong SocketIO 19](#_Toc183348947)

[Hình 15. Biểu đồ usecase tổng quát 20](#_Toc183348948)

[Hình 16. Phân rã usecase chơi game 21](#_Toc183348949)

[Hình 17. Phân rã usecase quản lý trang cá nhân 21](#_Toc183348950)

[Hình 18. Phân rã usecase quản lý map 22](#_Toc183348951)

[Hình 19. Đồng bộ trạng thái trò chơi 28](#_Toc183348952)

[Hình 20. Thuật toán va chạm giữa xe tăng và tường 30](#_Toc183348953)

[Hình 21. Thuật toán va chạm giữa đạn và tường 31](#_Toc183348954)

[Hình 22. Biểu đồ cơ sở dữ liệu 32](#_Toc183348955)

[Hình 23. Mã hóa mật khẩu người dùng 35](#_Toc183348956)

[Hình 24. Kiểm tra đăng nhập của người dùng 35](#_Toc183348957)

[Hình 25. Cơ chế lưu trận đấu 36](#_Toc183348958)

[Hình 26. Cơ chế phát lại trận đấu đã lưu 37](#_Toc183348959)

[Hình 27. Duy trì đăng nhập bằng Session 38](#_Toc183348960)

DANH SÁCH BẢNG BIỂU

[Bảng 1. Các status code trong HTTP 11](#_Toc183518350)

[Bảng 2. Các phương thức trong RestAPI 11](#_Toc183518351)

[Bảng 3. Đặc tả usecase trang cá nhân 22](#_Toc183518352)

[Bảng 4. Đặc tả usecase chơi game 23](#_Toc183518353)

[Bảng 5. Đặc tả usecase di chuyển 23](#_Toc183518354)

[Bảng 6. Đặc tả usecase chat thế giới 24](#_Toc183518355)

[Bảng 7. Đặc tả use case xem thông tin xếp hạng 24](#_Toc183518356)

[Bảng 8. Đặc tả usecase xem lịch sử đấu 24](#_Toc183518357)

[Bảng 9. Đặc tả use case quản lý người chơi 25](#_Toc183518358)

[Bảng 10. Đặc tả use case quản lý map 25](#_Toc183518359)

[Bảng 11. Các sự kiện để điều khiển xe tăng 28](#_Toc183518360)

[Bảng 12. Các phím được sử dụng để điều khiển xe tăng 29](#_Toc183518361)

[Bảng 13. Các lệnh được gửi lên Server 29](#_Toc183518362)

[Bảng 14. Danh sách các collection trong mongoDB 32](#_Toc183518363)

[Bảng 15. Collection user 33](#_Toc183518364)

[Bảng 16. Collection map 34](#_Toc183518365)

[Bảng 17. Collection match 34](#_Toc183518366)

[Bảng 18. Collection chat 34](#_Toc183518367)

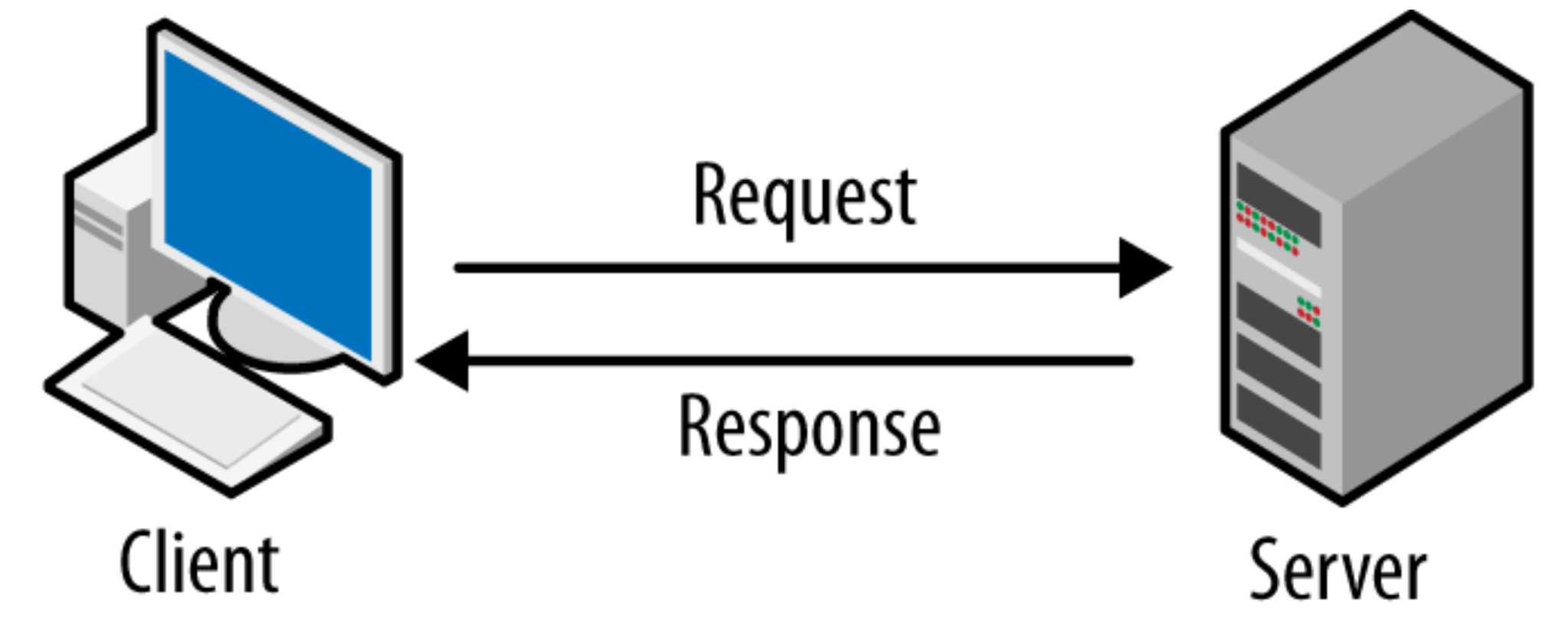
DANH SÁCH CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Ký hiệu** | **Ý nghĩa** |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheet |
| HTTP | Hypertext Transfer Protocol |
| REST | Representational state transfer |
| API | Application programming interface |
| MVC | Model View Controller |
| FTP |  |
| HTTPS |  |
|  |  |
|  |  |
|  |  |
|  |  |

MỞ ĐẦU

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

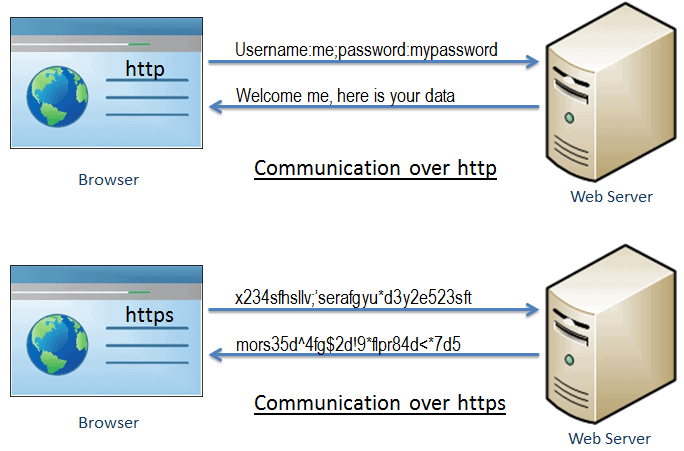
## 1.1 Mô hình Client-Server.



Hình . Mô hình Client-Server

* Mô hình Client-Server là một kiến trúc hệ thống trong đó có hai thành phần chính:
* **Client (Khách hàng):** Là thiết bị hoặc ứng dụng yêu cầu dịch vụ hoặc tài nguyên từ một máy chủ. Client gửi yêu cầu và nhận kết quả từ Server.
* **Server (Máy chủ):** Là thiết bị hoặc hệ thống cung cấp các dịch vụ hoặc tài nguyên cho client. Máy chủ xử lý yêu cầu từ client và trả về dữ liệu hoặc thông tin cần thiết.
* Quá trình hoạt động:
* Client gửi yêu cầu tới Server qua một giao thức (như HTTP).
* Server nhận yêu cầu, xử lý và trả về kết quả (thường là một tài nguyên hoặc dữ liệu).
* Giao tiếp giữa client và server có thể diễn ra qua mạng, và thường sử dụng các giao thức chuẩn như HTTP, HTTPS, FTP, hoặc TCP/IP.

## HTTP/ HTTPS là gì?



Hình . Giao thức HTTP/HTTPS

* **HTTP (HyperText Transfer Protocol):**
* HTTP là một giao thức truyền tải thông tin qua mạng, đặc biệt là cho các trang web. HTTP cho phép các trình duyệt web và máy chủ web giao tiếp với nhau để tải về và hiển thị các trang web. Nó hoạt động trên cơ chế yêu cầu và phản hồi.
* Ví dụ: Khi bạn truy cập một trang web như http://www.trungvadung.com, trình duyệt gửi một yêu cầu HTTP đến máy chủ, và máy chủ sẽ trả về một phản hồi với trang web đó.
* Đặc điểm của HTTP:

Không mã hóa: Dữ liệu truyền đi giữa client và server không được mã hóa, vì vậy có thể bị nghe lén hoặc thay đổi.

Không bảo mật: HTTP không cung cấp bảo mật hoặc tính toàn vẹn cho dữ liệu.

* **HTTPS (HyperText Transfer Protocol Secure):**
* HTTPS là phiên bản bảo mật của HTTP. Nó sử dụng SSL/TLS (Secure Sockets Layer / Transport Layer Security) để mã hóa dữ liệu truyền tải giữa client và server, giúp bảo vệ thông tin khỏi các cuộc tấn công như nghe lén, giả mạo, hoặc thay đổi nội dung dữ liệu.
* Ví dụ: Khi bạn truy cập một trang web có URL https://www.trungvadung.com, trình duyệt và máy chủ sẽ sử dụng HTTPS để bảo mật thông tin trao đổi.
* Đặc điểm của HTTPS:

Mã hóa: Dữ liệu được mã hóa, giúp bảo vệ thông tin khỏi các cuộc tấn công.

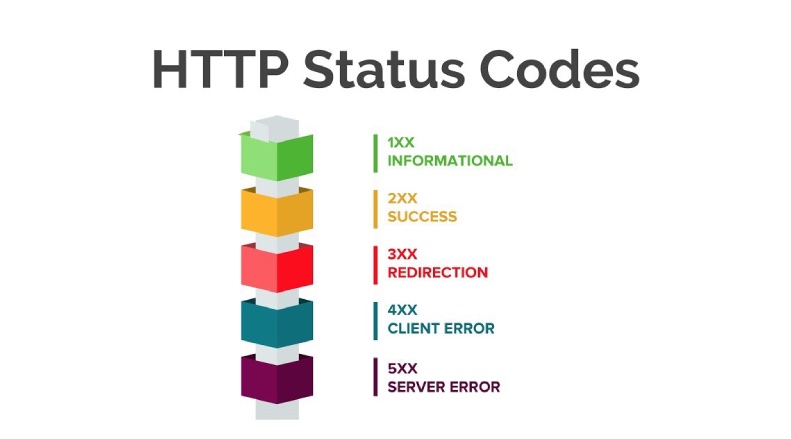
Bảo mật: Dữ liệu được xác thực và bảo vệ, giảm thiểu khả năng bị giả mạo.

Cần chứng chỉ SSL/TLS: Các máy chủ cần có chứng chỉ bảo mật SSL/TLS để sử dụng HTTPS.

## 1.3 Request và Response.

* **Request (Yêu cầu):** Request là thông điệp mà client gửi tới server để yêu cầu một tài nguyên hoặc thực hiện một hành động nào đó. Yêu cầu này thường bao gồm các thông tin như:
* **HTTP Method (Phương thức HTTP)**: Xác định loại yêu cầu (ví dụ: GET, POST, PUT, DELETE).
* **URL (Uniform Resource Locator)**: Địa chỉ của tài nguyên mà client muốn truy cập.
* **Headers (Tiêu đề)**: Các thông tin bổ sung về yêu cầu như kiểu nội dung, thông tin xác thực, v.v.
* **Body (Nội dung)**: Dữ liệu đi kèm trong yêu cầu, thường dùng trong các phương thức như POST hoặc PUT (ví dụ: dữ liệu biểu mẫu, JSON, XML).
* **Response (Phản hồi):** Response là thông điệp mà server gửi lại cho client để trả lời yêu cầu. Phản hồi này thường bao gồm:
* **HTTP Status Code (Mã trạng thái HTTP)**: Chỉ ra kết quả của yêu cầu (ví dụ: 200 OK, 404 Not Found, 500 Internal Server Error).
* **Headers**: Các thông tin bổ sung về phản hồi (ví dụ: kiểu nội dung, thời gian, v.v.).
* **Body (Nội dung)**: Dữ liệu trả về từ server, chẳng hạn như trang web, hình ảnh, thông tin JSON, v.v.

## 1.4 Status Code trong Response.



Hình . HTTP Status Codes

|  |  |
| --- | --- |
| **Status Code** | **Description** |
| 200 OK | Trả về thành công cho các phương thức GET, PUT, PATCH hoặc DELETE. |
| 201 Created | Trả về khi một Resource vừa được tạo thành công. |
| 204 No Content | Trả về khi Resource bị xoá thành công, không có dữ liệu trả về. |
| 304 Not Modified | Client có thể sử dụng dữ liệu cache, resource server không đổi gì. |
| 400 Bad Request | Request không hợp lệ, có thể do thiếu tham số hoặc cấu trúc sai. |
| 401 Unauthorized | Request cần có xác thực, người dùng chưa đăng nhập hoặc không có quyền. |
| 403 Forbidden | Bị từ chối, không có quyền truy cập tài nguyên. |
| 404 Not Found | Không tìm thấy resource từ URI yêu cầu. |
| 405 Method Not Allowed | Phương thức HTTP không cho phép với resource hiện tại. |
| 410 Gone | Resource không còn tồn tại, hoặc phiên bản cũ không còn được hỗ trợ. |
| 415 Unsupported Media Type | Không hỗ trợ kiểu resource này (ví dụ: định dạng tệp không hỗ trợ). |
| 422 Unprocessable Entity | Dữ liệu không được xác thực, thường gặp trong POST hoặc PUT. |
| 429 Too Many Requests | Request bị từ chối do giới hạn số lần gọi API trong một khoảng thời gian. |

Bảng . Các status code trong HTTP

## 1.5 Rest API.

**REST API (Representational State Transfer Application Programming Interface)** là một kiểu kiến trúc cho phép giao tiếp giữa các hệ thống qua HTTP, tuân theo các nguyên lý REST. REST API chủ yếu dùng để tạo, đọc, cập nhật và xóa các tài nguyên từ xa. Các thao tác này được gọi là **CRUD** (Create, Read, Update, Delete) và REST API cung cấp các phương thức HTTP tương ứng với mỗi thao tác.

|  |  |  |
| --- | --- | --- |
| **Chức năng CRUD** | **Phương thức HTTP** | **Mô tả** |
| Create | POST | Tạo mới một tài nguyên. |
| Read | GET | Lấy thông tin tài nguyên. |
| Update | PUT hoặc PATCH | Cập nhật tài nguyên. |
| Delete | DELETE | Xóa tài nguyên. |

Bảng . Các phương thức trong RestAPI

## 1.6 Express NodeJS Frameworks.

* **Express.js** là một framework web dành cho Node.js, được phát triển bởi **TJ Holowaychuk** vào năm 2010. Đây là một trong những framework phổ biến và mạnh mẽ nhất cho Node.js, thường được sử dụng để phát triển các ứng dụng web và API RESTful. Express được thiết kế đơn giản, nhẹ nhàng, nhưng lại cực kỳ linh hoạt và dễ sử dụng, giúp tăng tốc độ phát triển ứng dụng.
* Tính năng chính của Express.js:
* **Routing** trong Express cho phép bạn định nghĩa các URL endpoints mà ứng dụng sẽ phục vụ. Bạn có thể dễ dàng xử lý các yêu cầu HTTP (GET, POST, PUT, DELETE, v.v.) dựa trên đường dẫn URL.
* **Middleware** là một khái niệm quan trọng trong Express. Đây là các hàm được thực thi trong chuỗi xử lý yêu cầu HTTP, có thể được sử dụng để xử lý yêu cầu, thực hiện xác thực, hoặc thêm thông tin vào yêu cầu.
* Express hỗ trợ sử dụng các **template engine** như EJS, Pug, Handlebars, v.v. giúp bạn tạo các ứng dụng web động, nơi nội dung có thể thay đổi tùy thuộc vào dữ liệu.
* Express hỗ trợ việc nhận các tham số trong URL (path parameters) hoặc thông qua query string (dữ liệu gửi qua URL).
* Express hỗ trợ việc phục vụ các tệp tĩnh (như hình ảnh, CSS, JavaScript) thông qua middleware.

## 1.7 Template Engine Handlebars.

* **Mô tả:** Handlebars.js là một template engine (công cụ xử lý mẫu) cho JavaScript, giúp bạn tạo ra các HTML động dựa trên dữ liệu đầu vào. Nó cho phép tách biệt logic khỏi giao diện người dùng, giúp mã nguồn dễ bảo trì và mở rộng.
* **Lợi ích của Handlebars:**
* Tách biệt dữ liệu và giao diện: Giúp mã nguồn sạch sẽ và dễ bảo trì.
* Hỗ trợ mạnh mẽ cho điều kiện và vòng lặp: Handlebars cho phép xử lý logic một cách dễ dàng ngay trong templates.
* **Ứng dụng:**
* Web front-end: Tạo các ứng dụng web động.
* Node.js server-side rendering: Sinh HTML trên server và gửi đến client.

## 1.8 Socket.IO

### 1.8.1 Khái niệm

Socket.io là một module của NodeJS cho phép giao tiếp hai chiều giữa client và server. Giao tiếp hai chiều được kích hoạt khi client có Socket.IO trên trình duyệt và server cũng có tích hợp với Socket.IO. Dữ liệu được gửi có thể ở nhiều định dạng, đơn giản nhất là dùng JSON.

A diagram of a couple of squares

Description automatically generated with medium confidence

Hình . Giao tiếp bằng SocketIO

Socket.io bao gồm hai phần:

* **Thư viện phía máy chủ (Server-side library):** Thư viện này tích hợp với máy chủ web (chủ yếu là Node.js) và xử lý các kết nối đến từ máy khách, quản lý trao đổi dữ liệu và truyền phát tin nhắn đến người dùng được kết nối.
* **Thư viện phía máy khách (Client-side library):** Thư viện này chạy trong trình duyệt web của người dùng và thiết lập kết nối đến máy chủ. Nó cung cấp các chức năng gửi và nhận dữ liệu, cho phép trang web luôn đồng bộ với máy chủ và phản ứng với các cập nhật theo thời gian thực.

A diagram of a server

Description automatically generated

Hình . Mô hình truyền tin giữa Client-Server trong SocketIO

### 1.8.2 Các tính năng của Socket.IO

Socket.IO được xây dựng dựa trên nền tảng của WebSockets, cung cấp các tính năng bổ sung giúp nâng cao khả năng giao tiếp theo thời gian thực cho các ứng dụng web. Sau đây là một số tính năng chính mà Socket.IO cung cấp cho người dùng:

- Kết nối đáng tin cậy: Mặc dù WebSockets là phương thức truyền tải lý tưởng nhưng không phải tất cả các trình duyệt đều hỗ trợ WebSockets. Socket.IO thu hẹp khoảng cách này bằng cách quay trở lại các phương pháp thay thế như Long Polling HTTP nếu cần. Điều này đảm bảo có thể thiết lập kết nối đáng tin cậy bất kể môi trường trình duyệt.

- Tự động kết nối lại: Nếu kết nối giữa máy khách và máy chủ bị gián đoạn, Socket.IO sẽ tự động thử kết nối lại, đảm bảo tính liên tục cho ứng dụng.

- Phát sóng (Broadcasting): Socket.IO cung cấp phương thức io.emit() giúp dễ dàng truyền phát tin nhắn tới nhiều client cùng một lúc. Tình năng này được sử dụng trong các ứng dụng cần được cập nhật theo thời gian thực.

-Tín hiệu xác nhận : Đây là một khía cạnh thiết yếu của giao tiếp thời gian thực. Chúng cho phép server xác nhận rằng tin nhắn đã được client nhận hoặc ngược lại. Socket.IO tạo điều kiện thuận lợi cho việc này bằng cách cho phép các nhà phát triển gắn các hàm gọi lại (callback) vào các sự kiện được phát ra. Điều này trao quyền cho các nhà phát triển xây dựng các hệ thống mạnh mẽ trong đó tính toàn vẹn và độ tin cậy của dữ liệu là quan trọng.

A close-up of a computer code

Description automatically generated

Hình . Sender và Receiver trong SocketIO

- Tạo phòng (Rooms): Socket.IO cũng hỗ trợ việc tạo ra các phòng để nhóm các client thành các nhóm nhằm gửi thông báo hoặc dữ liệu tới toàn bộ nhóm đó. Điều này cho phép gửi thông báo đến một nhóm người dùng cụ thể được kết nối qua các thiết bị hoặc kênh nhất định.

A diagram of a server

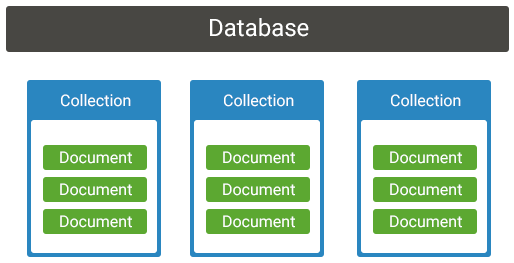
Description automatically generated

Hình . Room trong SocketIO

- Hỗ trợ đa nền tảng: Socket.IO có các triển khai cho nhiều ngôn ngữ và môi trường khác nhau, bao gồm JavaScript (Node.js), Java, Python, Golang, Rust và nhiều ngôn ngữ khác, giúp tích hợp dễ dàng vào các dự án đa dạng.

## 1.9 Hệ quản trị cơ sở dữ liệu MongoDB

MongoDB là một hệ thống cơ sở dữ liệu phi quan hệ (NoSQL), mã nguồn mở.MongoDB lưu trữ dữ liệu dưới dạng tài liệu (document) JSON, cho phép các ứng dụng lưu trữ và truy vấn dữ liệu một cách linh hoạt và hiệu quả.



Hình . Cơ sở dữ liệu MongoDB

* Cơ sở dữ liệu MongoDB lưu trữ tài liệu trong các collections, tương tự như bảng trong cơ sở dữ liệu quan hệ. Mỗi collection có thể chứa nhiều tài liệu (documents) có cấu trúc dữ liệu tùy ý.
* Bên trong collection sẽ có tài liệu (documents). Các tài liệu này sẽ chứa dữ liệu mà bạn muốn lưu trữ trong MongoDB database. Mỗi document có thể chứa nhiều fields dữ liệu, mỗi field được định danh bằng tên và có giá trị tương ứng.
* Các tài liệu (documents) được tạo bằng cách sử dụng các field. Các field là các key-value pair trong tài liệu, nó giống như các cột trong cơ sở dữ liệu quan hệ. Giá trị của fields có thể thuộc bất kỳ loại dữ liệu BSON nào như double, string, boolean,...

A black background with blue and green arrows

Description automatically generated

Hình . Collection và document trong MongoDB

* MongoDB hỗ trợ việc tạo index cho các field dữ liệu trong collection, giúp tăng tốc độ truy vấn. Chúng còn hỗ trợ sao chép dữ liệu giữa các node trong một cluster giúp đảm bảo tính khả dụng và độ tin cậy của hệ thống.
* MongoDB phân tán dữ liệu trên nhiều node, giúp tăng khả năng mở rộng của hệ thống, đồng thời chúng còn hỗ trợ tính toán phân tán bằng cách sử dụng MapReduce giúp xử lý dữ liệu lớn một cách hiệu quả.

# CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

## 2.1. Khái quát đề tài.

Trò chơi bắn xe tăng trực tuyến đa người người chơi thời gian thực được thiết kế và xây dựng theo mô hình Client-Server gồm các ứng dụng chạy trên Client và trên Server. Các công nghệ được sử dụng để xây dựng và phát triển trò chơi:A diagram of a software company

Description automatically generated

Hình 10. Kiến trúc dự án tổng quát

- Node.js: Đóng vai trò là máy chủ (Server), xử lý các yêu cầu từ các máy khách (Client), duy trì và điều phối trạng thái của trò chơi; phân phối dữ liệu qua mạng.

- Express: Framework trong Node.js để phát triển các API và xử lý các yêu cầu từ client.

- Socket.io: Thư viện để tạo ra kết nối thời gian thực giữa Client và Server. Việc sử dụng Socket.io cho phép các sự kiện như di chuyển xe tăng, bắn đạn và cập nhật trạng thái trò chơi được truyền trực tiếp giữa các người chơi qua mạng.

- Handlebars: Được sử dụng để tạo giao diện động trên Client, giúp render HTML dựa trên dữ liệu từ Server.

- Canvas: là một thành phần của HTML5 hỗ trợ vẽ đồ họa dựa trên frame (khung hình), dùng để vẽ các thành phần trong game, tạo animation (hoạt ảnh) liên tục và mượt mà.

- MongoDB: Cơ sở dữ liệu dùng để lưu trữ một lượng dữ liệu lớn liên quan đến người dùng, lịch sử các trận đấu,… của trò chơi

## 2.2. Lập trình mạng giữa Client và Server.

### 2.2.1. Xây dựng hệ thống phía Client và Server.

Trò chơi trực tuyến nhiều người chơi được xây dựng theo kiến trúc Client-Server. Mỗi người chơi chạy một phần mềm gọi là Client, chịu trách nhiệm hiển thị trò chơi và xử lý dữ liệu đầu vào của người chơi. Mỗi Client trao đổi dữ liệu với một Server trung tâm, xác thực hành động của người chơi và phát chúng đến các Client khác.

Client được viết bằng Javascript và chạy trong trình duyệt của người chơi. Server được viết bằng Javascript, sử dụng Node.js và module Express. Client và Server giao tiếp với nhau thông qua Socket.io.

- Server: Tạo một phiên bản Socket.io Server bằng cách sử dụng module HTTP tích hợp của Node.js và Express

- Client: Tạo một phiên bản Socket.io Client bằng cách sử dụng thư viện socket phía Client.

### 2.2.2. Cách thức giao tiếp giữa Client và Server.

Khi người chơi truy cập vào ứng dụng, một kết nối sẽ được thiết lập với Server. Lúc này một socket phía Client sẽ được tạo ra, đây là điểm cuối trong luồng giao tiếp giữa Client và Server. Với Socket.io, chúng ta có thể gửi và nhận tin nhắn thông qua socket, đây là cách cơ bản mà Client và Server sẽ tương tác. Tại đây, socket của máy khách được lưu trữ trong biến hằng socket để sử dụng sau trong tương lai.

A computer and server with text

Description automatically generated with medium confidence

Hình . Connection được tạo ra giữa Client và Server

Server yêu cầu Socket.io lắng nghe sự kiện ‘connection’, được kích hoạt mỗi khi Client kết nối với Server (sử dụng \_io.on(‘connection’) ). Khi có yêu cầu kết nối, nó sẽ gọi hàm callback được đưa ra ở đói số thứ hai ( Socket.connection). Hàm này nhận socket được sử dụng để thiết lập kết nối làm đối số đầu tiên, giống như socket ở Client, nó có thể được sử dụng để truyền và nhận thông điệp.

Lúc này Server và các Client sẽ giao tiếp với nhau thông qua đối tượng socket ở phía Client và Server đã kết nối. Quá trình phát và bắt sự kiện được thực hiện thông qua phương thức socket.emit() và socket.on() trên đối tượng socket ở Server và Client.

A diagram of a computer server

Description automatically generated

Hình . Giao tiếp hai chiều giữa Client và Server sử dụng Socket.io

Các chức năng của trò chơi được thực hiện thông qua các kỹ thuật chính trên socket phía Server là : broadcasting và room.

Kỹ thuật broadcasting sự kiện đến Client:

- Broadcasting sự kiện là phát một sự kiện đến một nhóm Client hoặc tất cả Client. Broadcasting là kỹ thuật được sử dụng hoàn toàn ở phía Server và Client chỉ nhận được sự kiện từ Server như bình thường. Khi người chơi gửi tin nhắn đến phòng chat thì Server sẽ nhận tin nhắn và gửi lại cho tất cả người chơi khác trong phòng chat.

- Việc phát sự kiện tới tất cả người chơi được thực hiện bằng cách sử dụng biến \_io trên Server và nó sẽ phát sự kiện tới tất cả Client đã kết nối.

A diagram of a client

Description automatically generated

Hình . Truyền thông cho tất cả Client

Kỹ thuật tạo room( phòng):

- Room là kỹ thuật được sử dụng hoàn toàn ở Server; Client không liên quan đến nó và Client không biết họ đang ở phòng nào trên Server. Room là một kênh tùy ý mà các socket có thể tham gia và rời khỏi. Với kỹ thuật tạo room, dữ liệu có thể được gửi trực tiếp từ socket này đến một nhóm socket khác bằng cách sử dụng Server làm trung gian. Một socket có thể ở nhiều room cùng một lúc và có thể nhận tất cả sự kiện phát ra từ các socket trong phòng đó từ Server.

-Trò chơi tổ chức hai loại phòng cho người chơi trong hệ thống: phòng dành cho người chơi tại lobby (sảnh) và phòng cho nhóm người chơi trong một trận đấu.

- Khi một người chơi truy cập vào sảnh, socket phía Client phát ra sự kiện kết nối đến sảnh. Socket phía Server nhận sự kiện từ Client và thực hiện phương thức join(“teamMatch”) với đối số là tên phòng.

A computer screen with text and images

Description automatically generated with medium confidence

Hình . Các Client tham gia vào room trong SocketIO

Lúc này, tất cả sự kiện diễn ra tại sảnh sẽ được phát lại đến tất cả socket phía Client đã tham gia vào room “teamMatch” thông qua biến \_io của Server.

Mỗi người chơi khi tham gia vào một trận đấu sẽ được thêm vào một room với tên của room được Server tạo ra. Luồng giao tiếp của các người chơi (di chuyển, trò chuyện,…) được gửi đến tất cả socket đã tham gia vào room.

## 2.3. Biểu đồ Usecase.

### 2.3.1. Biểu đồ Usecase tổng quát.

Hệ thống có 3 tác nhân chính:

* Người chơi.
* Khách vãng lai.
* Quản trị viên.

Các chức năng chính của hệ thống:

* Quản lý trang cá nhân.
* Chơi game.
* Cài đặt di chuyển.
* Quản lý người chơi.
* Quản lý map.
* Chat thế giới.
* Xem lịch sử đấu.
* Xem thông tin xếp hạng.

A diagram of a diagram

Description automatically generated

Hình . Biểu đồ usecase tổng quát

### 2.3.2. Biểu đồ usecase phân rã.

**a.Usecase chơi game.**

A diagram of a diagram

Description automatically generated

Hình . Phân rã usecase chơi game

**b.Usecase quản lý trang cá nhân.**

**A diagram of a diagram

Description automatically generated**

Hình . Phân rã usecase quản lý trang cá nhân

**c. Usecase quản lý map.**

**A diagram of a diagram

Description automatically generated**

Hình . Phân rã usecase quản lý map

### 2.3.3. Đặc tả use case.

**a. Đặc tả use case quản lý trang cá nhân**

|  |  |
| --- | --- |
| **Tên Use Case** | **Quản lý trang cá nhân** |
| Tác nhân chính | Người chơi |
| Mô tả | Người chơi có thể chỉnh sửa và quản lý thông tin cá nhân như tên, ảnh đại diện, và trạng thái. |
| Tiền điều kiện | Người chơi đã đăng nhập vào hệ thống. |
| Luồng chính | |  |  |  | | --- | --- | --- | | **STT** | **Thực hiện bởi** | **Hành động** | | 1 | Người chơi | Truy cập giao diện quản lý cá nhân. | | 2 | Hệ thống | Hiển thị thông tin cá nhân hiện tại. | | 3 | Người chơi | Cập nhật thông tin cần chỉnh sửa. | | 4 | Hệ thống | Lưu lại thay đổi và xác nhận. | |
| Luồng phụ | Nếu lưu thay đổi thất bại, hệ thống thông báo lỗi và yêu cầu người chơi thử lại. |
| Hậu điều kiện | Thông tin cá nhân được cập nhật thành công. |

Bảng . Đặc tả usecase trang cá nhân

**b. Đặc tả use case chơi game**

|  |  |
| --- | --- |
| **Tên Use Case** | **Chơi game** |
| Tác nhân chính | Người chơi |
| Mô tả | Người chơi tham gia vào trận đấu với các người chơi khác trong chế độ nhiều người. |
| Tiền điều kiện | Người chơi đã chọn chế độ chơi game và có kết nối mạng ổn định. |
| Luồng chính | |  |  |  | | --- | --- | --- | | **STT** | **Thực hiện bởi** | **Hành động** | | 1 | Người chơi | Chọn chế độ chơi. | | 2 | Người chơi | Tham gia hoặc tạo phòng. | | 3 | Người chơi | Tham gia trận đấu và chơi game. | | 4a | Người chơi | Bắt đầu game | | 4b | Hệ thống | Cập nhật thông tin người chơi | | 4c | Người dùng | Thao tác trên hệ thống | | 4d | Hệ thống | Xử lí thao tác của người chơi và cập nhật dữ liệu đồng bộ cho tất cả người chơi | | 5 | Hệ thống | Lưu kết quả trận đấu | |
| Luồng phụ | Nếu người chơi mất kết nối trong trận, hệ thống tự động thoát khỏi phòng. |
| Hậu điều kiện | Kết quả trận đấu được lưu lại, và người chơi có thể tiếp tục chơi trận mới. |

Bảng . Đặc tả usecase chơi game

**c. Đặc tả use case cài đặt di chuyển**

|  |  |
| --- | --- |
| **Tên Use Case** | **Cài đặt di chuyển** |
| Tác nhân chính | Người chơi |
| Mô tả | Người chơi tùy chỉnh phím điều khiển để thuận tiện cho thao tác trong game. |
| Tiền điều kiện | Người chơi đã đăng nhập vào hệ thống. |
| Luồng chính | |  |  |  | | --- | --- | --- | | **STT** | **Thực hiện bởi** | **Hành động** | | 1 | Người chơi | Mở giao diện cài đặt điều khiển | | 2 | Hệ thống | Hiển thị danh sách các phím mặc địn | | 3 | Người chơi | Chọn thay đổi phím điều khiển | | 4 | Hệ thống | Lưu cài đặt và áp dụng | |
| Luồng phụ | Nếu lưu cài đặt thất bại, hệ thống yêu cầu người chơi thử lại. |
| Hậu điều kiện | Cài đặt di chuyển được lưu và áp dụng trong game. |

Bảng . Đặc tả usecase di chuyển

**d. Đặc tả use case chat thế giới**

|  |  |
| --- | --- |
| **Tên Use Case** | **Chat thế giới** |
| Tác nhân chính | Người chơi, Khách vãng lai |
| Mô tả | Tất cả người dùng có thể gửi và nhận tin nhắn trong kênh chat công cộng. |
| Tiền điều kiện | Người dùng đang trực tuyến. |
| Luồng chính | |  |  |  | | --- | --- | --- | | **STT** | **Thực hiện bởi** | **Hành động** | | 1 | Người chơi | Mở giao diện chat thế giới | | 2 | Hệ thống | Hiển thị các tin nhắn gần đây. | | 3 | Người chơi | Gửi tin nhắn. | | 4 | Hệ thống | Gửi đến tất cả người dùng trong kênh chat. | |
| Luồng phụ | Nếu gửi tin nhắn thất bại, hệ thống thông báo lỗi. |
| Hậu điều kiện | Tin nhắn được hiển thị trong kênh chat. |

Bảng . Đặc tả usecase chat thế giới

**e. Đặc tả use case xem thông tin xếp hạng**

|  |  |
| --- | --- |
| **Tên Use Case** | **Xem thông tin xếp hạng** |
| Tác nhân chính | Người chơi, Khách vãng lai |
| Mô tả | Người dùng có thể xem bảng xếp hạng các người chơi theo thứ hạng. |
| Tiền điều kiện | Không yêu cầu điều kiện. |
| Luồng chính | |  |  |  | | --- | --- | --- | | **STT** | **Thực hiện bởi** | **Hành động** | | 1 | Người chơi | Mở giao diện xếp hạng. | | 2 | Hệ thống | Hiển thị danh sách xếp hạng theo thứ tự từ cao xuống thấp. | |
| Luồng phụ | Nếu không có dữ liệu, hệ thống hiển thị thông báo. |
| Hậu điều kiện | Người dùng xem được thông tin xếp hạng. |

Bảng . Đặc tả use case xem thông tin xếp hạng

**f. Đặc tả use case xem lịch sử đấu**

|  |  |
| --- | --- |
| **Tên Use Case** | **Xem lịch sử đấu** |
| Tác nhân chính | Người chơi |
| Mô tả | Người chơi xem lại các trận đấu mà họ đã tham gia. |
| Tiền điều kiện | Người chơi đã tham gia ít nhất một trận đấu trước đó. |
| Luồng chính | |  |  |  | | --- | --- | --- | | **STT** | **Thực hiện bởi** | **Hành động** | | 1 | Người chơi | Mở giao diện lịch sử đấu. | | 2 | Hệ thống | Hiển thị danh sách trận đấu đã tham gia. | | 3 | Người chơi | Chọn xem chi tiết trận đấu | | 4 | Hệ thống | Hiển thị thông tin chi tiết về trận đấu. | |
| Luồng phụ | Nếu không có trận đấu nào, hệ thống hiển thị thông báo. |
| Hậu điều kiện | Người chơi xem được thông tin trận đấu đã chọn. |

Bảng . Đặc tả usecase xem lịch sử đấu

**g. Đặc tả use case quản lý người chơi**

|  |  |
| --- | --- |
| **Tên Use Case** | **Quản lý người chơi** |
| Tác nhân chính | Quản trị viên |
| Mô tả | Quản trị viên quản lý tài khoản người chơi (thêm, xóa, sửa thông tin, khóa tài khoản). |
| Tiền điều kiện | Quản trị viên đã đăng nhập. |
| Luồng chính | |  |  |  | | --- | --- | --- | | **STT** | **Thực hiện bởi** | **Hành động** | | 1 | Quản trị viên | Truy cập giao diện quản lý người chơi | | 2 | Hệ thống | Hiển thị danh sách người chơi. | | 3 | Quản trị viên | Thực hiện các thao tác đình chỉ hoặc xóa tài khoản. | | 4 | Hệ thống | Lưu thay đổi. | |
| Luồng phụ | Nếu thực hiện thao tác thất bại, hệ thống thông báo lỗi. |
| Hậu điều kiện | Danh sách người chơi được cập nhật. |

Bảng . Đặc tả use case quản lý người chơi

**h. Đặc tả use case quản lý map**

|  |  |
| --- | --- |
| **Tên Use Case** | **Quản lý map** |
| Tác nhân chính | Quản trị viên |
| Mô tả | Quản trị viên quản lý các bản đồ (thêm, xóa, cập nhật thông tin). |
| Tiền điều kiện | Quản trị viên đã đăng nhập. |
| Luồng chính | |  |  |  | | --- | --- | --- | | **STT** | **Thực hiện bởi** | **Hành động** | | 1 | Quản trị viên | Truy cập giao diện quản lý map. | | 2 | Hệ thống | Hiển thị danh sách bản đồ. | | 3 | Quản trị viên | Thực hiện thao tác thêm, xóa, cập nhật thông tin bản đồ. | | 4 | Hệ thống | Lưu thay đổi. | |
| Luồng phụ | Nếu thực hiện thao tác thất bại, hệ thống thông báo lỗi. |
| Hậu điều kiện | Danh sách bản đồ được cập nhật. |

Bảng . Đặc tả use case quản lý map

## 2.4 EngineGame.

EngineGame là module để xử lý logic và các sự kiện trong game. EngineGame sẽ quyết định trạng thái trò chơi tiếp theo, luồng điều khiển của người và cách trò chơi được hiển thị lên màn hình. EngineGame gồm 2 phần là EngineGameServer và EngineGameClient.

### 2.4.1 EngineGameServer

EngineGameServer được sử dụng để quản lý và xử lý logic các trạng thái trò chơi và đồng bộ nó cho các client thông qua socketIO. EngineGameServer gồm các thành phần sau:

* **Lớp Tank:**
* Mô tả đối tượng xe tăng với các thuộc tính như ID, tọa độ, tốc độ, máu, góc quay, nòng súng, đạn, trạng thái, v.v.
* Các chức năng chính:
* **Di chuyển và xoay:** moveForward, moveBackward, rotateLeft, rotateRight, rotateTurret.
* **Quản lý đạn:** createBullet (tạo đạn), handleBulletCollision (xử lý va chạm của đạn).
* **Kiểm tra biên:** setOutOfCanvas (giữ xe tăng trong vùng chơi).
* **Trạng thái:** getBounds (giới hạn va chạm) và getState (lấy trạng thái hiện tại).
* **Lớp Bullet:**
* Mô tả đạn với các thuộc tính như tọa độ, hướng, số lần va chạm cho phép, sát thương, v.v.
* Các chức năng:
* **Cập nhật vị trí:** updatePosition.
* **Xử lý va chạm:** handleCollision (giảm số lần va chạm hoặc đảo hướng khi đụng).
* **Lớp Utilities:**
* Cung cấp các hàm tiện ích:
* **Tính toán góc quay:** calculusAngle.
* **Va chạm:** isColliding, getCollisionSides.
* **Ngẫu nhiên:** getRandom, getIntRandom.
* **Lớp Game:**
* Quản lý toàn bộ trò chơi:
* **Bản đồ:** setMap, buildMap (thiết lập và xác định tường liền kề để xử lý va chạm).
* **Xe tăng:** addTank (thêm xe tăng), controlTank (điều khiển xe tăng theo hành động), getTank (lấy xe tăng theo ID).
* **Va chạm:** detectAndHandleBulletCollision (xử lý va chạm của đạn với tường hoặc xe tăng).
* **Cập nhật trạng thái:** updateState.
* **Lấy trạng thái:** getState (trả về trạng thái hiện tại của các xe tăng và đạn).
* **Kết thúc và lưu game:** endGame, saveGame.

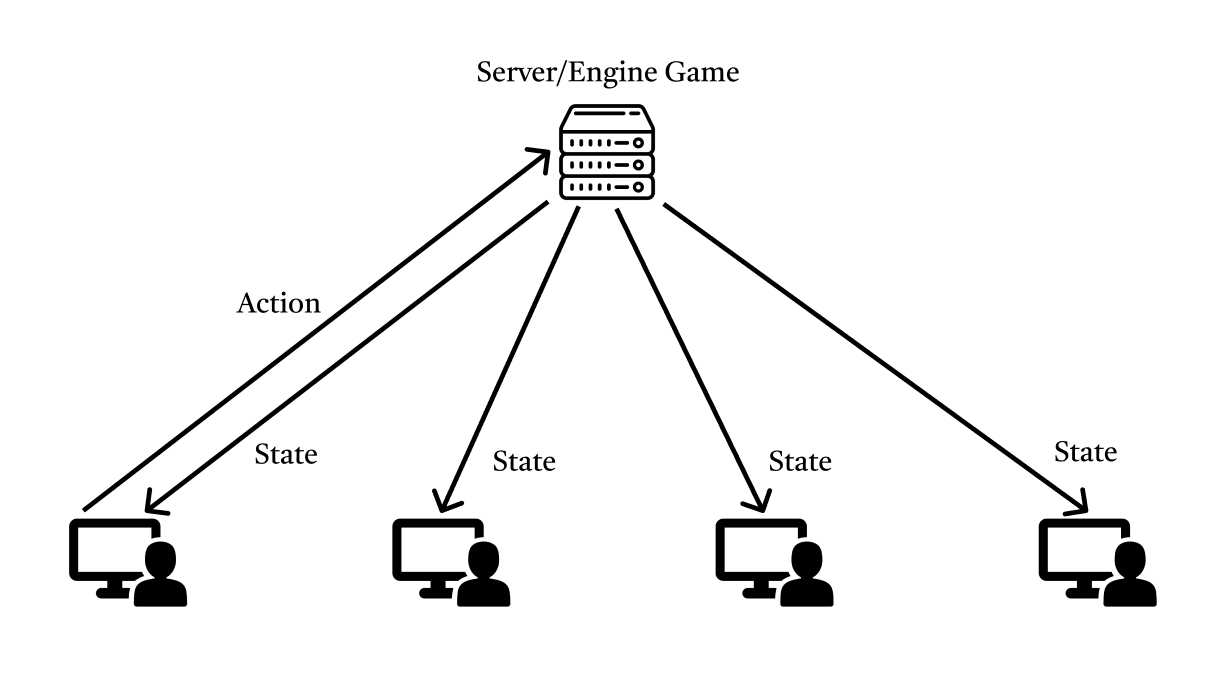
### 2.4.2 EngineGameClient

EngineGameClient nhận các trạng thái từ EngineGameServer và vẽ chúng lên canvas cho người chơi.

* **Lớp Tank:**
* Mô tả đối tượng xe tăng với các thuộc tính tương tự như trong server (ID, vị trí, góc quay, tốc độ, v.v.).
* Thêm chức năng:
* setState(state): Cập nhật trạng thái của xe tăng dựa trên dữ liệu nhận từ server.
* **Lớp Bullet:**
* Mô tả viên đạn (tọa độ, hướng, màu sắc, tốc độ, sát thương).
* **Lớp Utilities:**
* Chứa các hàm tiện ích và vẽ đối tượng trên canvas:
* **Vẽ:** drawTank, drawBullets, drawTrail, drawHealthBar, drawMap, drawTankTurret, drawProtectedTank.
* **Hiệu ứng nổ:** createBlast, explode.
* **Xử lý dữ liệu:** encodeData, decodedData (nén và giải nén dữ liệu game), getSizeData.
* **Lớp Game:**
* Quản lý game trên client:
* **Quản lý bản đồ:** setMap, drawMap (thiết lập và vẽ bản đồ).
* **Quản lý xe tăng:** addTank (thêm xe tăng), setStateGame (cập nhật trạng thái từ server), updateState (vẽ lại trạng thái game).
* **Chế độ phát lại:** replayGame (phát lại trạng thái game lưu trữ, với tốc độ khung hình tùy chỉnh).

## 2.5 Cơ chế đồng bộ trạng thái trò chơi giữa các người dùng.

* **Người chơi gửi hành động lên server**:
* Mỗi người chơi thực hiện một hành động (ví dụ: di chuyển, bắn đạn, hoặc quay xe tăng).
* Hành động này được gửi từ client của người chơi lên server (Engine Game).
* **Server xử lý hành động và cập nhật trạng thái**:
* Server nhận tất cả các hành động từ các người chơi.
* Sau đó, server tính toán trạng thái mới của trò chơi dựa trên logic trò chơi, bao gồm việc di chuyển các đối tượng, xử lý va chạm, hoặc tính toán sát thương.
* **Server gửi trạng thái mới đến tất cả các người chơi**:
* Server gửi trạng thái toàn cục của trò chơi đến từng client.
* Mỗi client cập nhật trạng thái trò chơi trên giao diện của mình để đảm bảo đồng bộ với tất cả người chơi khác.



Hình . Đồng bộ trạng thái trò chơi

## 2.6. Sự kiện bàn phím và chuột của người dùng.

* Người chơi vào trò chơi và hệ thống nhận diện người chơi (có thể qua user\_id).
* Người chơi sử dụng phím để di chuyển và quay xe tăng (phím W, S, A, D).
* Người chơi sử dụng chuột để thay đổi hướng quay của khẩu pháo.
* Khi người chơi nhấn phím Space, lệnh bắn (fire) sẽ được gửi đến máy chủ và xe tăng sẽ bắn.

### 2.6.1. Các sự kiện chính được để người dùng điều khiển xe tăng.

|  |  |  |
| --- | --- | --- |
| **Sự kiện** | **Phím hoặc Chuột** | **Mô tả** |
| keydown | Bất kỳ phím nào | Khi nhấn phím, sự kiện này được kích hoạt và cập nhật trạng thái của phím (lưu vào keysPressed). |
| keyup | Bất kỳ phím nào | Khi thả phím, sự kiện này sẽ kiểm tra nếu phím Space được nhả và thực hiện hành động bắn (gửi dữ liệu "fire"). Đồng thời, trạng thái phím được cập nhật lại thành false. |
| mousemove | Chuột | Khi di chuyển chuột, tọa độ chuột được ghi nhận và gửi đến máy chủ để thay đổi hướng quay của khẩu pháo xe tăng. |

Bảng . Các sự kiện để điều khiển xe tăng

### 2.6.2. Các phím được sử dụng để điều khiển xe tăng.

|  |  |
| --- | --- |
| **Phím** | **Chức năng** |
| W | Di chuyển xe tăng hoặc đầu xe tăng về phía trước (tuỳ chế độ di chuyển: "rotateMode" hoặc "freeMode"). |
| S | Di chuyển xe tăng hoặc đầu xe tăng lùi lại (tuỳ chế độ di chuyển: "rotateMode" hoặc "freeMode"). |
| A | Quay xe tăng sang trái (tuỳ chế độ di chuyển: "rotateMode" hoặc "freeMode"). |
| D | Quay xe tăng sang phải (tuỳ chế độ di chuyển: "rotateMode" hoặc "freeMode"). |
| Space | Thực hiện hành động bắn (khi nhả phím Space, gửi dữ liệu bắn đi). |

Bảng . Các phím được sử dụng để điều khiển xe tăng

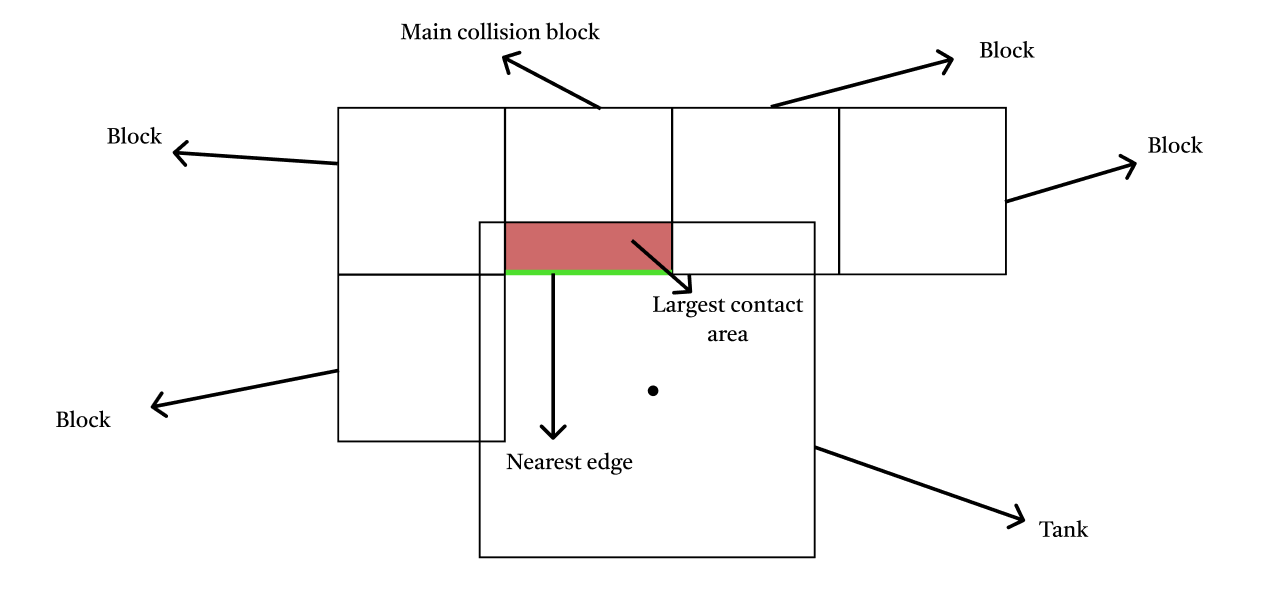
### 2.6.3. Các lệnh được gửi lên Server.

|  |  |
| --- | --- |
| **Lệnh gửi dữ liệu** | **Chức năng** |
| fire | Thực hiện hành động bắn súng khi người chơi nhấn phím Space và thả ra. |
| moveForward | Di chuyển xe tăng hoặc đầu xe tăng về phía trước trong chế độ "rotateMode" hoặc "freeMode". |
| moveBackward | Di chuyển xe tăng hoặc đầu xe tăng lùi lại trong chế độ "rotateMode" hoặc "freeMode". |
| rotateLeft | Quay xe tăng sang trái trong chế độ "rotateMode" hoặc di chuyển đầu xe tăng sang trái trong "freeMode". |
| rotateRight | Quay xe tăng sang phải trong chế độ "rotateMode" hoặc di chuyển đầu xe tăng sang phải trong "freeMode". |
| moveHead | Di chuyển đầu xe tăng trong chế độ "freeMode". |
| moveLeft | Di chuyển xe tăng sang trái trong chế độ "freeMode". |
| moveRight | Di chuyển xe tăng sang phải trong chế độ "freeMode". |
| changeRotationTurret | Thay đổi hướng quay của khẩu pháo xe tăng theo tọa độ chuột (x, y) để điều khiển hướng bắn. |

Bảng . Các lệnh được gửi lên Server

## 2.7. Thuật toán va chạm.

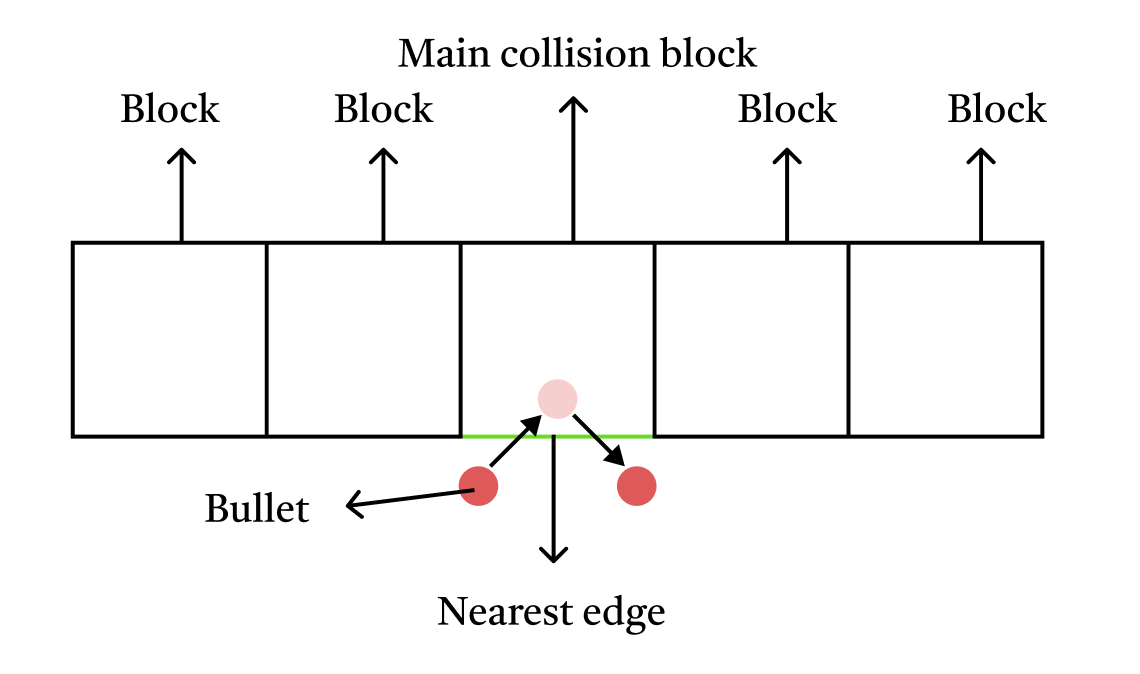
**2.7.1. Va chạm của xe tăng và các vật cản.**



Hình . Thuật toán va chạm giữa xe tăng và tường

* **Xác định vùng tiếp xúc lớn nhất**:
* Đối tượng (xe tăng) kiểm tra va chạm với các khối (block) xung quanh.
* Khối va chạm chính (**main collision block**) được xác định dựa trên diện tích tiếp xúc lớn nhất (vùng đỏ trong hình). Đây là khối mà xe tăng tiếp xúc nhiều nhất.
* **Tính nearest edge (cạnh gần nhất)**:
* Từ tâm của xe tăng, tính khoảng cách đến từng cạnh của **main collision block**.
* Tuy nhiên, không xét những cạnh bị chặn bởi các khối khác (các cạnh này không được tiếp cận trực tiếp).
* **Các bước xét cạnh hợp lệ**:
* Kiểm tra xem cạnh nào của **main collision block** bị "chặn" bởi các khối khác. Một cạnh bị chặn nếu một khối liền kề tiếp xúc với cạnh đó.
* Chỉ các cạnh không bị chặn mới được đưa vào tính toán.
* **Kết quả**:
* Cạnh không bị chặn gần nhất với tâm xe tăng được chọn làm **nearest edge**.
* **Nearest edge** này được sử dụng để xác định cạnh va chạm, từ đó cập nhật trạng thái và không cho phép xe tăng đi xuyên qua.

**2.7.2. Va chạm của đạn và các vật cản.**



Hình . Thuật toán va chạm giữa đạn và tường

* **Xác định khối chứa viên đạn:**
* Dựa trên vị trí của viên đạn, xác định viên đạn đang nằm trong khối nào bằng cách kiểm tra tọa độ viên đạn so với lưới các khối.
* **Xác định các cạnh hợp lệ của khối:**
* Với khối vừa xác định, kiểm tra các cạnh nào không giáp với khối khác (chỉ xét các cạnh "rìa" hoặc "ngoài biên").
* Loại bỏ các cạnh bên trong hoặc cạnh chung với khối khác.
* **Tính khoảng cách từ viên đạn đến các cạnh hợp lệ:**
* Đo khoảng cách từ vị trí viên đạn đến từng cạnh hợp lệ của khối.
* **So sánh để tìm cạnh gần nhất:**
* So sánh các khoảng cách đã tính được, xác định cạnh nào là gần nhất với viên đạn.
* **Trả kết quả:**
* Xác định cạnh gần nhất là cạnh nằm ngang hay cạnh dọc và trả về kết quả.

## 2.8. Cơ sở dữ liệu.

### 2.8.1. Biểu đồ cơ sở dữ liệu.

**A screenshot of a computer

Description automatically generated**

Hình . Biểu đồ cơ sở dữ liệu

-Danh sách các collection trong cơ sở dữ liệu

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên collection** | **Mô tả** |
| 1 | User | Lưu thông tin người dùng trên hệ thống (người chơi, admin) |
| 2 | Map | Lưu thông tin các map trong hệ thống |
| 3 | Match | Lưu thông tin ván đấu của người chơi |
| 4 | Chat | Lưu thông tin trò chuyện của người chơi |

Bảng . Danh sách các collection trong mongoDB

### 2.8.2. Mô tả chi tiết các collection trong cơ sở dữ liệu.

- Collection User

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu** | **Ràng buộc** |
| \_id | objectId | index |
| username | string | unique |
| password | string |  |
| firstname | string |  |
| lastname | string |  |
| email | string |  |
| facebook | string |  |
| avatar | string |  |
| coin | numeric |  |
| created\_at | date |  |
| listFriend | array |  |
| friendRequestSentList | array |  |
| listFriend\_request | array |  |
| level | numeric |  |
| role | string |  |
| rank | string |  |
| history\_match | array |  |
| settings | document |  |
| inGame | boolean |  |
| bio | string |  |
| exp | numeric |  |
| win | numeric |  |
| lose | numeric |  |

Bảng . Collection user

- Collection Map

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu** | **Ràng buộc** |
| \_id | objectId | index |
| dogPosition | document |  |
| catPosition | document |  |
| obstacles | array |  |
| previousMapId | string |  |
| disabled | boolean |  |

Bảng . Collection map

- Collection Match

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu** | **Ràng buộc** |
| \_id | objectId | index |
| dogPlayers | array |  |
| catPlayers | array |  |
| startTime | date |  |
| endTime | date |  |
| winnerTeam | string |  |
| map\_id | objectId |  |
| gameMode | string |  |
| history | string |  |

Bảng . Collection match

- Collection Chat

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu** | **Ràng buộc** |
| \_id | objectId | index |
| userId | objectId |  |
| message | string |  |
| tempstamp | date |  |

Bảng . Collection chat

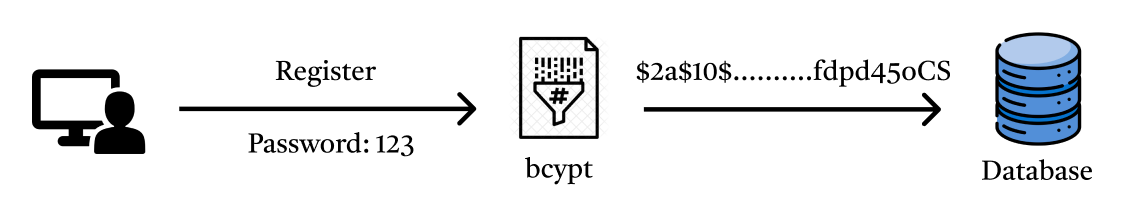
## 2.9. Mã hóa mật khẩu.

### 2.9.1. bcrypt: Thuật toán mã hóa mật khẩu

**bcrypt** là một thuật toán băm mật khẩu dựa trên Blowfish cipher, và được thiết kế để bảo vệ mật khẩu khỏi các cuộc tấn công như brute-force (dùng thử tất cả các khả năng) và dictionary attacks (tấn công từ điển). bcrypt có một số đặc điểm nổi bật:

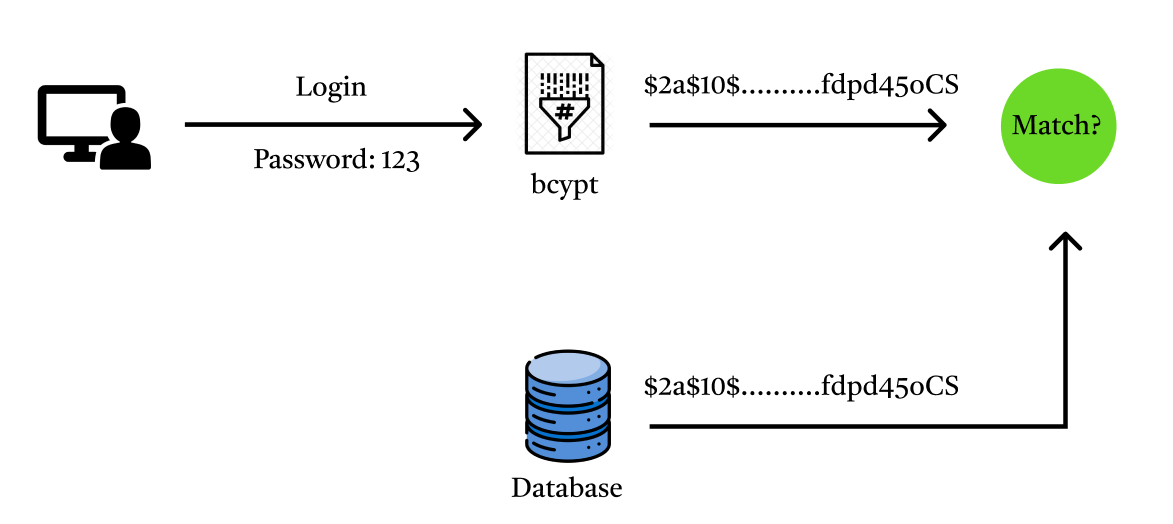
* **Slow hashing:** bcrypt được thiết kế để thực thi chậm, giúp làm giảm tốc độ tấn công brute-force
* **Salting:** bcrypt tự động thêm một giá trị ngẫu nhiên gọi là salt vào mật khẩu trước khi băm, giúp ngăn ngừa các cuộc tấn công với bảng băm (rainbow tables).
* **Adaptive cost factor:** bcrypt cho phép điều chỉnh mức độ khó của việc băm thông qua tham số cost factor.

**2.9.2 Quá trình mã hóa và xác thực mật khẩu**



Hình . Mã hóa mật khẩu người dùng

* Khi người dùng tạo tài khoản hoặc thay đổi mật khẩu, mật khẩu được mã hóa bằng cách sử dụng hàm mã hóa. Kết quả là một giá trị hash được lưu trữ trong cơ sở dữ liệu.



Hình . Kiểm tra đăng nhập của người dùng

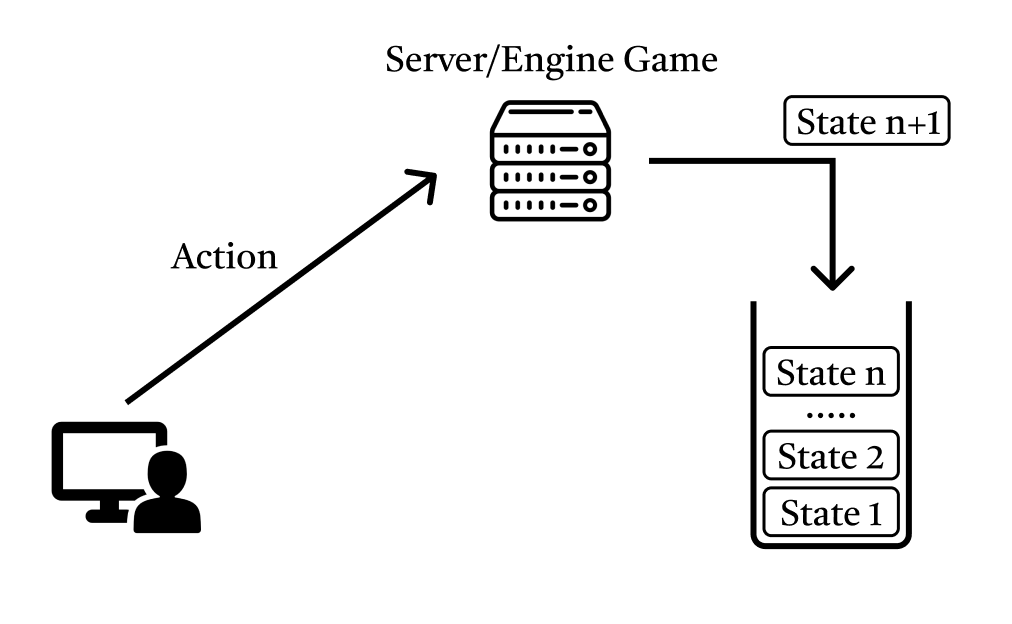
* Khi người dùng đăng nhập, mật khẩu nhập vào sẽ được băm lại và so sánh với giá trị hash đã lưu trữ trong cơ sở dữ liệu. Nếu chúng khớp, người dùng được xác thực thành công.

**2.9.3 Ưu điểm của việc sử dụng bcrypt**

* **Bảo mật cao**: Việc sử dụng salt ngẫu nhiên và cost factor cao giúp bảo vệ mật khẩu khỏi các cuộc tấn công mật khẩu phổ biến.
* **Chống tấn công Brute-force**: bcrypt được thiết kế để chậm lại với các tài nguyên phần cứng mạnh mẽ, làm giảm khả năng thực hiện tấn công brute-force.
* **Bảo vệ mật khẩu ngay cả khi bị rò rỉ**: Ngay cả khi một hacker có được hash của nhiều mật khẩu, họ cũng không thể tính toán lại các mật khẩu gốc từ đó.

**2.10. Cơ chế lưu và phát lại trận đấu đã chơi.**

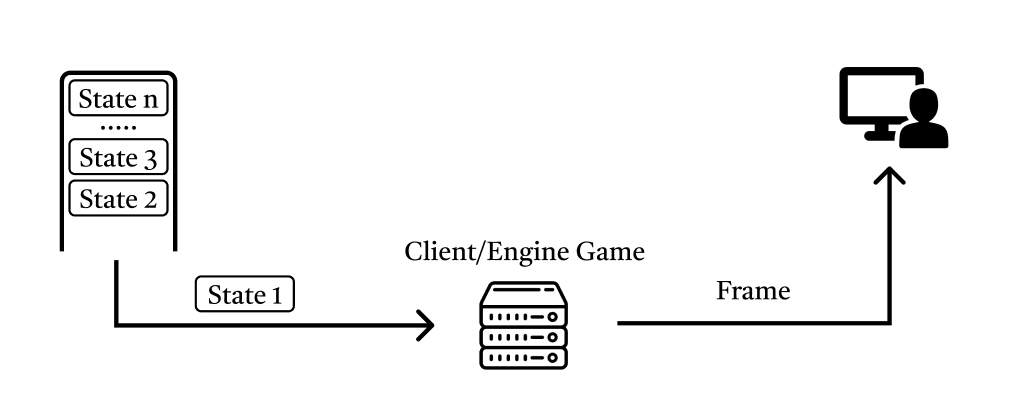
### 2.10.1 Cơ chế lưu trận đấu.



Hình . Cơ chế lưu trận đấu

* **Nhận hành động từ người chơi:**
* Người chơi thực hiện các hành động (di chuyển, tấn công, …).
* Các hành động này được gửi lên **Server/Engine Game** để xử lý.
* **Xử lý và cập nhật trạng thái trò chơi:**
* **Server/Engine Game** nhận hành động, xử lý logic trò chơi (như tính toán va chạm, xử lí các sự kiện, cập nhật trạng thái, …).
* Sau khi xử lý, trò chơi tạo ra trạng thái mới phản ánh kết quả của hành động vừa thực hiện.
* **Lưu trạng thái vào bộ lưu trữ:**
* Trạng thái mới được thêm vào một danh sách chứa tất cả các trạng thái trước đó.
* Bộ lưu trữ này hoạt động như một lịch sử trạng thái trò chơi, bao gồm: Vị trí của các viên đạn và xe tăng, các sự kiện ở trạng thái đó.

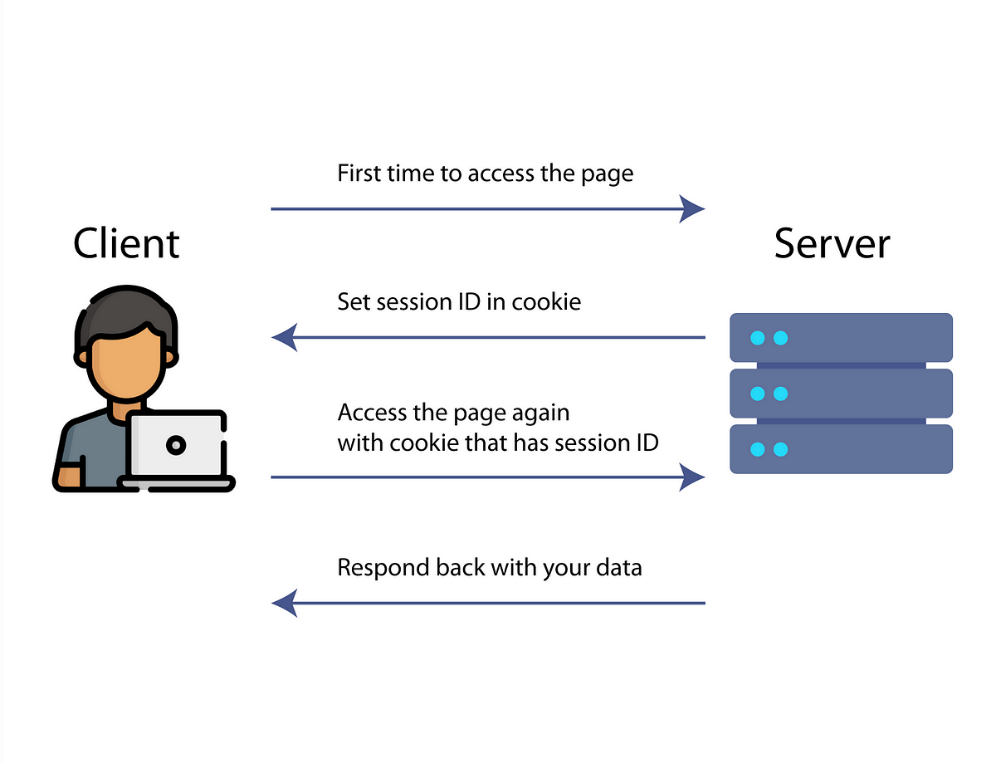
**2.10.2 Phát lại trận đấu đã chơi.**

****

Hình . Cơ chế phát lại trận đấu đã lưu

* **Truy xuất trạng thái từ cơ sở dữ liệu:**
* Các trạng thái trước đó của trò chơi (ví dụ: **State 1, State 2, ..., State n**) đã được lưu trữ trong một cơ sở dữ liệu.
* Khi cần xem lại, người dùng sẽ truy vấn cơ sở dữ liệu để lấy ra trạng thái đầu tiền.
* **Chuyển trạng thái đến Client/Engine Game:**
* Trạng thái được gửi đến **Client/Engine Game**, nơi xử lý và tái tạo lại trạng thái đã lưu.
* Engine Game sẽ giải mã dữ liệu trong **State 1** và tái hiện môi trường, vị trí các xe tăng, viên đạn, sự kiện, ….
* **Kết xuất khung hình (Frame) để hiển thị:**
* Engine Game tạo ra các khung hình (**Frames**) tương ứng với trạng thái đã lấy.
* Các khung hình này sau đó được hiển thị chính xác những gì đã xảy ra trong trạng thái đó.

## 2.11. Cơ chế duy trì đăng nhập bằng Session.



Hình . Duy trì đăng nhập bằng Session

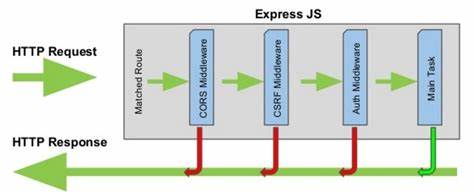
### 2.11.1. Cách Session hoạt động.

* **Client đăng nhập lần đầu tiên:**  
  Khi người dùng lần đầu gửi yêu cầu đến server (ví dụ: truy cập trang web hoặc đăng nhập), server không có bất kỳ thông tin nào để nhận diện họ.
* **Server tạo session ID và lưu trong cookie:**  
  Sau khi xử lý yêu cầu, server sẽ tạo một session (phiên làm việc) và gắn một session ID duy nhất cho phiên này. Session ID được lưu trữ trong một cookie và gửi về trình duyệt của client.
* **Client truy cập lần sau với session ID:**  
  Trong các lần truy cập sau, trình duyệt tự động gửi cookie (bao gồm session ID) cùng với yêu cầu đến server. Server dựa trên session ID để lấy dữ liệu liên quan đến phiên làm việc của người dùng.
* **Server phản hồi với dữ liệu của người dùng:**  
  Sau khi xác thực session ID, server có thể trả về các dữ liệu hoặc tài nguyên cần thiết cho người dùng dựa trên thông tin trong phiên làm việc.

2.11.2. Các tính năng của Session.

* **Lưu trữ trạng thái người dùng**
* Session cho phép ứng dụng web theo dõi trạng thái của người dùng trong suốt phiên làm việc (giữa các yêu cầu HTTP). Vì HTTP là giao thức **stateless** (không duy trì trạng thái), session giúp tạo cảm giác "kết nối liên tục" cho người dùng.
* Ví dụ: Khi người dùng đăng nhập, trạng thái đăng nhập sẽ được lưu trong session để họ không cần đăng nhập lại ở các trang khác.
* **Lưu trữ dữ liệu tạm thời**
* Session có thể được sử dụng để lưu trữ dữ liệu tạm thời mà không cần ghi xuống cơ sở dữ liệu. Dữ liệu này thường được lưu trữ trên bộ nhớ server.
* **Hỗ trợ hết hạn phiên làm việc (Session expiration)**
* Session có thể tự động hết hạn sau một khoảng thời gian nếu người dùng không hoạt động (Idle timeout) hoặc khi phiên đạt giới hạn thời gian (Max-age).
* Ví dụ: Đăng xuất tự động sau 30 phút không sử dụng.
* **Dễ dàng hủy bỏ phiên làm việc (Session destruction)**
* Session có thể được xóa khi người dùng đăng xuất hoặc khi cần bảo vệ thông tin nhạy cảm.
* Ví dụ: Khi người dùng nhấn "Đăng xuất", session sẽ bị hủy bỏ để ngăn truy cập trái phép.

## 2.12. Middleware trong Express.js



Hình . Middleware trong Express.js

### 2.12.1. Middleware là gì?

Middleware trong Express.js là các hàm được thực thi trong quá trình xử lý yêu cầu HTTP. Nó đóng vai trò trung gian giữa yêu cầu của client và phản hồi từ server, giúp xử lý các tác vụ cụ thể như xác thực, quản lý lỗi, hoặc cấu hình bảo mật.

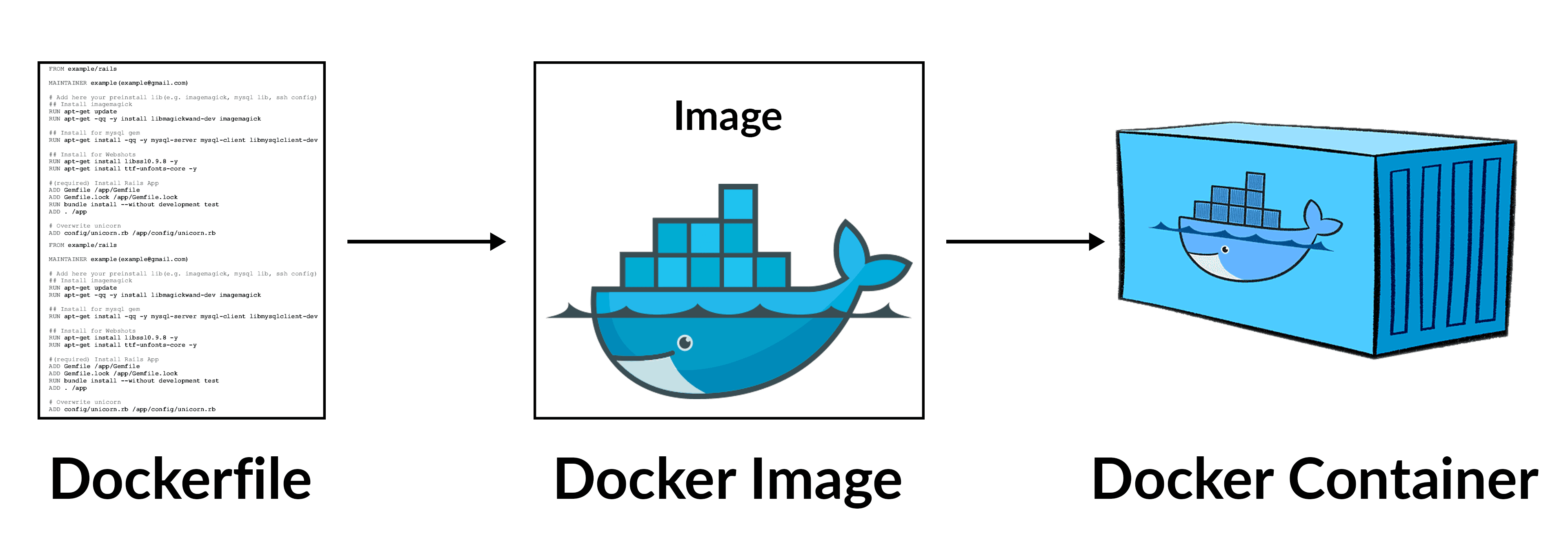
### 2.12.2. Các bước xử lý của middleware

* **Matched Route:** Xác định route tương ứng với yêu cầu HTTP nhận được.
* **CORS Middleware:** Đảm bảo rằng ứng dụng hỗ trợ chia sẻ tài nguyên giữa các nguồn gốc (Cross-Origin Resource Sharing).
* **CSRF Middleware:** Kiểm tra và bảo vệ ứng dụng khỏi các cuộc tấn công giả mạo yêu cầu từ trình duyệt (Cross-Site Request Forgery).
* **Auth Middleware:** Xác thực người dùng, kiểm tra quyền truy cập trước khi tiếp tục xử lý.
* **Main Task:** Thực hiện nhiệm vụ chính như truy vấn cơ sở dữ liệu, xử lý logic ứng dụng và gửi phản hồi về client.

### 2.12.3. Quá trình xử lý yêu cầu và phản hồi

* Khi một yêu cầu HTTP đến (HTTP Request), nó được chuyển qua từng lớp middleware theo thứ tự.
* Mỗi middleware có thể:
  + Chuyển tiếp yêu cầu đến middleware tiếp theo.
  + Kết thúc chuỗi xử lý và gửi phản hồi (HTTP Response).
  + Xử lý lỗi hoặc dừng yêu cầu nếu có vấn đề.

## 2.13. Sử dụng Docker để deploy và triển khai ứng dụng lên Internet



Hình . Deploy bằng docker

**2.13.1. Docker là gì?**

Docker là một nền tảng mã nguồn mở giúp đóng gói, triển khai và chạy các ứng dụng trong các container. Container cung cấp môi trường độc lập, đảm bảo rằng ứng dụng hoạt động nhất quán trên mọi hệ thống, từ máy tính cá nhân đến server triển khai.

**2.13.2. Lợi ích của việc sử dụng Docker**

* **Khả năng tái tạo môi trường:** Docker đảm bảo rằng môi trường phát triển, thử nghiệm và triển khai là giống hệt nhau.
* **Dễ dàng triển khai:** Các container chứa tất cả các thành phần cần thiết (mã nguồn, thư viện, công cụ) giúp triển khai ứng dụng nhanh chóng và dễ dàng.
* **Tính di động:** Các container có thể chạy trên mọi hệ điều hành hỗ trợ Docker, giúp tăng tính linh hoạt khi triển khai.

**2.13.3. Quy trình triển khai ứng dụng bằng Docker**

* **Tạo Dockerfile:**  
  Dockerfile là tệp định nghĩa cách xây dựng image Docker cho ứng dụng.

FROM node:18-alpine

WORKDIR /src

RUN apk add --no-cache make gcc g++ python3

RUN npm config set registry https://registry.npmjs.org

COPY package.json .

RUN npm install --only=production

COPY src/ .

CMD ["npx", "nodemon" , "src/index.js"]

* **Sử dụng docker-compose:**  
  Với các ứng dụng phức tạp bao gồm nhiều dịch vụ (backend, database), Docker Compose giúp quản lý chúng dễ dàng.
* **Build và run docker:**

Truy cập vào VPS bằng ssh và chạy lệnh: docker compose up.

* **Kết quả:**

Website được deploy ở địa chỉ <http://74.208.246.165/>

# CHƯƠNG 3. DEMO ỨNG DỤNG VÀ ĐÁNH GIÁ KẾT QUẢ

## 3.1. Demo ứng dụng.

## 3.2 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

[1] Handlebars, <https://handlebarsjs.com/guide/>, (Truy cập 10/2024)

[2] Socket.io, <https://socket.io/docs/v4/>, (Truy cập 10/2024)

[3] Mongoose, <https://mongoosejs.com/docs/migrating_to_8.html>, (Truy cập 10/2024)

[4] ExpressJS, <https://expressjs.com/en/4x/api.html>, (Truy cập 10/2024)

[5] W3school,HTML Canvas Reference, <https://www.w3schools.com/tags/ref_canvas.asp>, (Truy cập 10/2024)

[6] NodeJS, <https://nodejs.org/api/synopsis.html>, (Truy cập 10/2024)

PHỤ LỤC

|  |
| --- |
| [Đưa code vào đây] |