A white rectangular frame with black lines

Description automatically generated

**Sinh viên thực hiện:**

**1. Hoàng Thị Kim Phụng**

**2.**

**3.**

**4.**

**5.**

Đà Nẵng, tháng 11, năm 2024

**HỌC PHẦN: AN TOÀN THÔNG TIN**

**BÁO CÁO: CUỐI KỲ**

**SỬ DỤNG JWT TRONG VIỆC ĐĂNG KÝ, ĐĂNG NHẬP, XÁC THỰC VÀ PHÂN QUYỀN**

**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM**

**KHOA TIN HỌC**



# **I. Tổng quan**

## **1.1. Đặt vấn đề**

Trong các ứng dụng web hiện đại, bảo mật là yếu tố quan trọng để bảo vệ dữ liệu người dùng, với xác thực (authentication) và phân quyền (authorization) đóng vai trò then chốt. Xác thực giúp xác định danh tính người dùng, trong khi phân quyền đảm bảo người dùng chỉ truy cập được các tài nguyên họ được phép. JSON Web Tokens (JWT) là công cụ quan trọng trong việc bảo mật dữ liệu khi truyền tải qua mạng, nhờ vào tính bảo mật, nhỏ gọn, tự chứa và linh hoạt. JWT giúp mã hóa và ký thông tin, đảm bảo dữ liệu không bị thay đổi trong quá trình truyền tải, góp phần bảo vệ dữ liệu người dùng hiệu quả.

## **1.2. Các nghiên cứu liên quan**

- Để thực hiện được đề tài sử dụng JWT trong việc xây dựng chức năng đăng ký, đăng nhập, xác thực và phân quyền, cần nắm vững và có các kiến thức sau:  
+ Kiến thức về JWT (Json Web Tokens): cấu trúc và cách hoạt động của JWT, phân biệt được access token và refresh token

+ Kiến thức về xác thực (Authentication) và phân quyền (Authorization)

+ Kiến thức về Frontend:

* React JS: để xây dựng giao diện người dùng
* Redux Toolkit: để quản lý trạng thái ứng dụng
* Axios: thư viện này thực hiện các yêu cầu của HTTP
* Redux Persist: giúp duy trì phiên đăng nhập

+ Kiến thức về Backend:

* Node.js: là môi trường chạy JavaScript phía server
* Express: là framework cho Node.js để xây dựng REST API
* MongoDB: là cơ sở dữ liệu NoSQL
* Mongoose: đây là thư viện của MongoDB và Node.js để quản lý dữ liệu
* .env files: dùng để bảo mật thông tin nhạy cảm như JWT secret

+ Kiến thức về bảo mật Web

+ Kiến thức về testing API và kiểm tra bảo mật

## **1.3. Mục tiêu**

- Giúp xây dựng các hệ thống xác thực và phân quyền cho người dùng dựa trên JWT

- Đảm bảo tính bảo mật và hiệu quả khi xử lý các yêu cầu đăng ký, đăng nhập và phân quyền cho người dùng

## **1.4. Phạm vi đề tài**

- Phạm vi nghiên cứu bao gồm việc triển khai hệ thống xác thực và phân quyền trong một ứng dụng web. Các chức năng chính sẽ bao gồm:

+ Sử dụng JWT để xác thực người dùng.

+ Phân quyền người dùng dựa trên các loại token (access token và refresh token).

+ Lưu trữ và kiểm tra token trong header của các yêu cầu API.

## **1.5. Phương pháp nghiên cứu**

### **1.5.1. Phân tích hệ thống**

- Xác định các thực thể:

+ Người dùng (user): các đối tượng đăng ký, đăng nhập và tương tác với hệ thống gồm user và admin

+ Token: để lưu trữ thông tin xác thực của người dùng sau khi đăng nhập sau đó hệ thống sẽ quản lý các access token và refresh token để duy trì phiên làm việc và thực hiện phân quyền

- Xác định các mối quan hệ: người dùng – vai trò, người dùng – token, người dùng – dữ liệu, token – API

- Xác định các chức năng cụ thể mà hệ thống cần cung cấp gồm: đăng ký, đăng nhập, xác thực và phân quyền

### **1.5.2. Công cụ và công nghệ**

- Node.js và Express: Xây dựng backend và REST API cho đăng ký, đăng nhập, xác thực, phân quyền.

- MongoDB và Mongoose: Quản lý và lưu trữ dữ liệu người dùng, token trong cơ sở dữ liệu.

- JWT: Cung cấp mã hóa thông tin xác thực, duy trì phiên làm việc của người dùng.

- Redux Toolkit và Axios: Quản lý trạng thái người dùng trên frontend, gửi yêu cầu API có bảo mật.

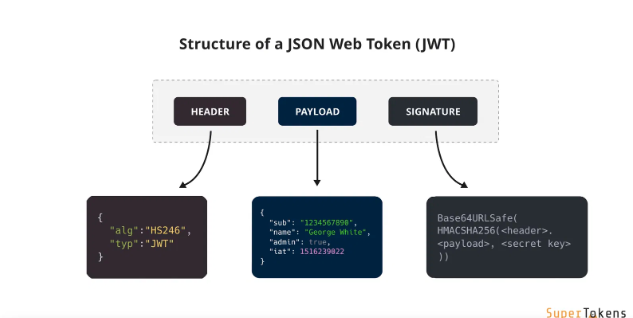
- Thunder Client hoặc Postman: Kiểm thử API, đảm bảo các chức năng hoạt động đúng và an toàn.

# **II. Cơ sở lý thuyết**

## **2.1. JWT (Json Web Tokens)**

- JWT - "JSON Web Token" là một tiêu chuẩn mở cho việc tạo ra các token truy cập an toàn dựa trên JSON. JWT thường được sử dụng để xác thực và ủy quyền người dùng trong ứng dụng web và di động.

- Cấu trúc của một JSON Web Token (JWT) bao gồm ba phần chính và phần tách nhau bằng một dấu chấm (.), mỗi phần phục vụ cho một mục đích cụ thể nhằm đảm bảo tính toàn vẹn và xác thực của token:



+ Header: chứa metadata về loại token và thuật toán ký được sử dụng để mã hóa token và có dạng như trên.

🡪 Trong đó: “alg” là thuật toán mã hóa được sử dụng ở đây là HMAC SHA – 246, “typ” là loại token JWT

+ Payload: sẽ chứa các claim (tuyên bố) về người dùng và các thông tín khác. Co ba loại claim chính:

* Registered claims: là các claims được định nghĩa sẵn bởi chuẩn JWT, gồm "iss" (Issuer), "sub" (Subject), "exp" (Expiration Time), "nbf" (Not Before), "iat" (Issued At) và "jti" (JWT ID).
* Public claims: được sử dụng không xung đột với các standard claim, nhưng cần thống nhất với các entities sử dụng JWT.
* Private claims: được sử dụng để chuyển thông tin giữa các bên, nhưng không được định nghĩa trong chuẩn JWT.

+ Signature: được tạo ra bằng cách mã hóa header và payload (dùng thuật toán chỉ định trong header và một secret key) để xác minh tính hợp lệ của token

🡪 Chuỗi token có dạng:

"{base64url-encoded header}.{base64url-encoded payload}.{signature}"

## **2.2. Authentication và Authorization**

|  | **Authentication (xác thực)** | **Authorization (phân quyền)** |
| --- | --- | --- |
| Khái niệm | Là quá trình xác minh ai đang truy cập hệ thống. | Là quá trình xác định người dùng có quyền gì trong hệ thống sau khi đã được xác thực. |
| Mục tiêu | Xác nhận danh tính của người dùng | Xác định quyền truy cập và hành động mà người dùng có thể thực hiện |
| Quá trình | Kiểm tra thông tin đăng nhập (tên người dùng, mật khẩu, ...) | Kiểm tra quyền của người dùng dựa trên vai trò, quyền hạn |
| Ví dụ | Người dùng nhập tên và mật khẩu, hệ thống xác thực và cho phép đăng nhập | Sau khi đăng nhập, người dùng có quyền truy cập vào các tài nguyên nhất định (admin, xem dữ liệu, ...) |
| Phương pháp | Sử dụng mật khẩu, mã OTP, sinh trắc học, xác thực đa yếu tố (MFA) | Sử dụng vai trò (role), nhóm quyền (permission) hoặc chính sách phân quyền |

## **2.3. Access Token và Refresh Token**

- Access Token và Refresh Token đang dần thay thế session và cookie để xác thực và phân quyền người dùng, nhờ vào khả năng hoạt động hiệu quả hơn trong môi trường phân tán và bảo mật cao hơn.

|  |  |  |
| --- | --- | --- |
|  | Access Token | Refresh Token |
| Khái niệm | Là mã thông báo dùng để xác thực và cấp quyền truy cập vào tài nguyên hệ thống. | Mã thông báo giúp làm mới Access Token khi hết hạn. |
| Mục đích | Cung cấp quyền truy cập tạm thời đến các tài nguyên bảo mật. | Duy trì phiên làm việc mà không cần người dùng đăng nhập lại |
| Thời gian hiệu lực | Ngắn (vài phút đến vài giờ). | Dài hơn Access Token (vài tuần đến vài tháng). |
| Cách dùng | - Access Token được gửi cùng với các yêu cầu đến server, thông qua header HTTP (thường là trong Authorization).  Ví dụ: Authorization: Bearer <access\_token>. | Khi Access Token hết hạn, client gửi Refresh Token đến server để yêu cầu cấp lại Access Token mới. |
| Bảo mật | Thường được mã hóa hoặc sử dụng các thuật toán bảo mật (JWT, OAuth) | Cần được bảo mật khi lưu trữ vì nó có thể được sử dụng để lấy lại Access Token. |
| Lưu trữ | Lưu trữ trên client (localStorage, sessionStorage) hoặc trong bộ nhớ. | Được lưu trữ an toàn trong HttpOnly cookies hoặc client-side storage |
| Quyền hạn | Cấp quyền truy cập hạn chế vào các tài nguyên, thường chứa thông tin về vai trò và quyền người dùng. | Không cấp quyền truy cập trực tiếp, chỉ giúp làm mới Access Token. |

## **2.4. REST API và bảo mật API với JWT**

- REST API là kiểu kiến trúc API sử dụng các phương thức HTTP (GET, POST, PUT, DELETE) để quản lý tài nguyên theo nguyên tắc không trạng thái (stateless) và dễ truy cập.

- JWT là phương thức xác thực người dùng trong API RESTful, bao gồm token mã hóa ba phần (header, payload, signature) giúp xác thực và phân quyền người dùng.

- Cách hoạt động của JWT trong API:

+ Đăng nhập: Server tạo JWT (access token) khi người dùng đăng nhập.

+ Xác thực yêu cầu: JWT được gửi trong header mỗi lần client gửi yêu cầu.

+ Phân quyền: JWT chứa quyền hạn để xác định tài nguyên người dùng có thể truy cập.

+ Refresh token: Khi access token hết hạn, refresh token tạo lại token mới.

- Bảo mật JWT trong API:

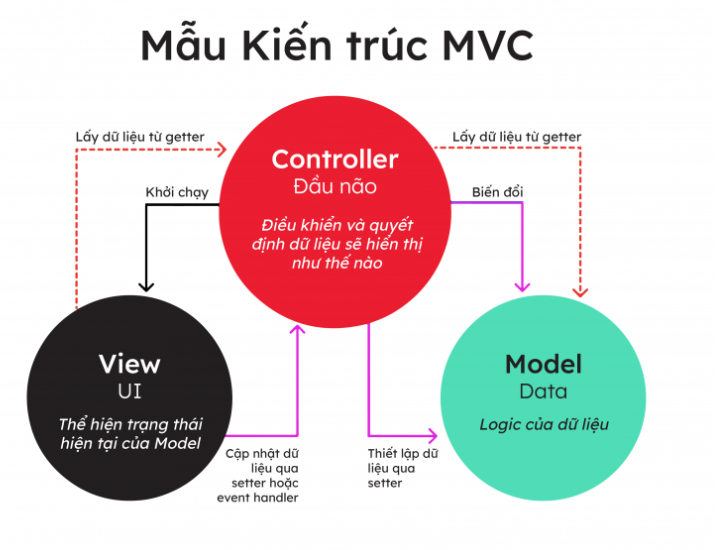
+ Mã hóa token với secret key để bảo vệ tính toàn vẹn.

+ Thời gian hiệu lực của access token và refresh token dài hơn giúp an toàn hơn.

+ Lưu trữ an toàn trong HTTP-only cookies hoặc local storage để chống tấn công.

## **2.5. Mô hình MVC (Model-View-Controller)**

- Mô hình MVC (Model – View – Controller) : là một mẫu kiến trúc/mẫu thiết kế (design pattern) tách ứng dụng thành ba thành phần logic chính: Model, View và Controller. Mỗi thành phần kiến trúc được xây dựng để xử lý các khía cạnh phát triển cụ thể của một ứng dụng.



+ Model ( Mô hình)

* Là phần back-end chứa tất cả logic dữ liệu của ứng dụng. Dữ liệu ở đây có thể là dữ liệu đang được truyền giữa các thành phần View và Controller hoặc bất kỳ dữ liệu nào khác liên quan đến logic của doanh nghiệp

VD:

Giả sử bạn đang phát triển một ứng dụng mua sắm. Ở đây, Model sẽ chỉ định giỏ hàng sẽ bao gồm những dữ liệu nào — như mặt hàng, giá cả, ... — và những dữ liệu nào đã có sẵn trong giỏ hàng.

+ View (Giao diện)

* View là các thành phần hiển thị giao diện người dùng (UI) của ứng dụng. Thông thường, giao diện người dùng này được tạo từ dữ liệu Model.

VD:

Trong ứng dụng mua sắm, View quyết định cách hiển thị giỏ hàng và sử dụng dữ liệu từ Model để hiển thị. Nó bao gồm các thành phần giao diện người dùng như nút bấm, danh sách thả xuống mà người dùng tương tác.

+ Controller (Bộ điều khiển)

* Là các thành phần xử lý tương tác của người dùng để làm việc với Model (cập nhật logic dữ liệu) hoặc/ và với View (cập nhật hiển thị giao diện người dùng).

VD:

Trong ứng dụng mua sắm, người dùng có thể thêm hoặc xóa mặt hàng trong giỏ hàng qua các nút bấm. Những hành động này được gửi đến Controller, Controller sẽ cập nhật Model và gửi dữ liệu mới đến View để hiển thị.

## **2.6. Middleware và Bảo mật Token**

- Middleware trong ứng dụng web là các hàm xử lý được thực thi trong quá trình xử lý request, trước khi đến route handler, có thể thực hiện các nhiệm vụ như: xác thực người dùng, kiểm tra quyền truy cập, ghi log và quản lý lỗi 🡪 giúp đảm bảo logic này được áp dụng cho tất cả các request mà không phải lặp lại trong từng route handler

- Bảo mật Token: Sử dụng httpOnly Cookies để bảo vệ refresh token, tránh lộ token qua các lỗ hổng XSS.

# **III. Phân tích hệ thống**

## **3.1. Một số thực thể quan trọng**

- Trong hệ thống quản lý xác thực và phân quyền, các thực thể chính gồm:

+ User (người dùng):

|  |  |  |
| --- | --- | --- |
| **STT** | **Thuộc tính** | **Mô tả** |
| 1 | userID | ID người dùng là duy nhất |
| 2 | username | Tên người dùng |
| 3 | password | Mật khẩu cần được mã hóa |
| 4 | role | Vị trí vai trò của người dùng (VD: user, admin) |
| 5 | email | Email của người dùng để xác nhận tài khoản |

+ Token (mã thông báo)

|  |  |  |
| --- | --- | --- |
| **STT** | **Thuộc tính** | **Mô tả** |
| 1 | accessToken | Token cấp quyền truy cập ngắn hạn |
| 2 | refreshToken | Token cấp lại quyền truy cập khi accessToken hết hạn |
| 3 | expiryDate | Ngày hết hạn của từng token |

+ Role (vai trò)

|  |  |  |
| --- | --- | --- |
| **STT** | **Thuộc tính** | **Mô tả** |
| 1 | roleId | ID của vai trò |
| 2 | roleName | Tên vai trò, ví dụ: admin, editor, viewer |
| 3 | permissions | Các quyền được gán cho vai trò |

## **3.2. Một số quan hệ quan trọng**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên quan hệ** | **Mối quan hệ** | **Mô tả** |
| 1 | User và Token | 1 – n | Đảm bảo rằng mỗi phiên đăng nhập của người dùng được bảo mật bằng một cặp token riêng, giúp quản lý quyền truy cập hiệu quả. |
| 2 | User và Role | 1 – n | Vai trò của User sẽ quyết định họ có quyền truy cập đến những tài nguyên nào. |
| 3 | Role và Permission | 1 – n | Quan hệ này giúp hệ thống mở rộng khả năng kiểm soát quyền truy cập, cho phép thiết lập các quyền tùy chỉnh cho từng vai trò. |

## **3.3. Thiết kế hệ thống**

- Backend:

+ Server: Dùng Node.js và Express.js để xử lý các yêu cầu từ client, thực hiện xác thực và quản lý phiên làm việc của người dùng.

+ Cơ sở dữ liệu: MongoDB sẽ được dùng để lưu trữ thông tin người dùng, vai trò, token và các quyền hạn liên quan. Mongoose ORM giúp quản lý và truy vấn dữ liệu.

+ Token Management: Sử dụng JWT để cấp Access Token và Refresh Token. Các token này sẽ được lưu trữ ở client và chỉ gửi về server khi cần.

+ Authentication Middleware: Middleware sẽ kiểm tra Access Token trước khi cho phép truy cập vào các tài nguyên, bảo vệ các endpoint yêu cầu xác thực.

+ Refresh Token Endpoint: Khi Access Token hết hạn, client có thể yêu cầu Refresh Token để lấy một Access Token mới mà không cần đăng nhập lại.

- Frontend:

+ Giao diện người dùng: Tạo các giao diện cho đăng ký, đăng nhập, và truy cập vào tài nguyên hệ thống.

+ Quản lý trạng thái: Sử dụng Redux Toolkit để quản lý trạng thái người dùng và token, đảm bảo việc cập nhật trạng thái truy cập diễn ra đồng bộ.

+ Tương tác với API: Sử dụng Axios để gửi yêu cầu HTTP đến backend và truyền token qua header để xác thực.

- Quy trình hoạt động tổng quan:

+ Đăng nhập/Đăng ký: Khi người dùng đăng nhập hoặc đăng ký thành công, server cấp Access Token và Refresh Token và gửi về cho client.

+ Truy cập tài nguyên: Client gửi Access Token kèm các yêu cầu đến API bảo mật. Server kiểm tra tính hợp lệ của token trước khi cho phép truy cập.

+ Làm mới Token: Khi Access Token hết hạn, client sẽ gửi Refresh Token đến endpoint /refresh-token để lấy token mới.

+ Phân quyền: Dựa trên vai trò và quyền hạn của người dùng, API sẽ kiểm tra và cho phép truy cập tương ứng đến từng tài nguyên.