Java Spring RESTful APIs – Xây dựng Backend với Spring Boot

Mục lục

I. Các kiến thức cần nắm

1. JSON

JSON : Javascript Object Notation. Là một định dạng dữ liệu để lữu trữ và trao đổi dữ liệu được sử dụng ở nhiều ngôn ngữ khác nhau: Java, C#,…

Ví dụ về JSON:

{

“id”: 1

“name”: “Trinh Quang Lam”

}

Sử dụng cặp dấu {} để định nghĩa JSON

Các thuộc tính được định nghĩa theo quy luật name:value(ngăn cách nhau bởi dấu :). Thuộc tính name luôn được bọc bởi “ “

Chi tiết: https://www.w3schools.com/js/js_json_intro.asp

2. API
   API (Application Programing Interface) hiểu đơn giản là 1 đường link URL tại phía backend, frontend sẽ gọi tới đường link URL này để lấy/sử dụng dữ liệu

   HTTP (HyperText Transfer Protocol) là giao thức truyền tải siêu văn bản, được dùng để giao tiếp giữa trình duyệt và máy chủ web. Nó xác định cách gửi và nhận dữ liệu qua internet, chẳng hạn như tải trang web, hình ảnh, hoặc gửi biểu mẫu.
   HTTP Method → thao tác CRUD
   POST → Tạo mới 1 thực thể
   GET → lấy thông tin từ Server
   PUT/PATCH → Cập nhật thông tin
   DELETE → Xóa một thực thể

   Cấu trúc HTTP Request:
   - Request Line: method + URL
   - Header Variabled
   - Message body : Json

Chi tiết: https://jsonplaceholder.typicode.com/

3. Response Entity

   Response Entity trong HTTP là phần dữ liệu chính được trả về từ máy chủ trong một **HTTP Response**.

   Để các hệ thống nói chuyện với nhau 1 cách đầy đủ nhất, 1 lời phản hồi response sẽ gồm:

- Thông tin header
- Thông tin status
- Thông tin body

Ví dụ: **ResponseEntity.status(HttpStatus.Ok).headers(Instance_of_HttpHeaders)**

.body(Instance_of_object_send_back_to_client);

Một số HTTP Status hay dùng:

Mã lỗi ám chỉ request thành công:

200 – request succeeded (hay dùng cho method GET/PUT/DELETE)

201 – request created a resource (hay dùng cho method POST)

204 – no content to return (dùng khi muốn có thông báo không có data ở phản hồi)

Mã lỗi ám chỉ request thất bại (lỗi do client)

400 – bad request(lỗi exception, validate)

401 – unauthorized : thường dùng khi client chưa đăng nhập

403 – Forbidden : đã đăng nhập thành công, nhưng không có quyền hạn để thực hiện tác vụ này

404 – Resource not found : không tìm thấy tài nguyên mà client yêu cầu

405 – Method not supported : check cho đúng method khi sử dụng với endpoint

Mã lỗi ám chỉ request thất bại (lỗi do server):

500 – Internal Server error : lỗi xảy ra bên trong Server

503 – Service Unavailable : server không hoạt động nên không có sẵn dịch vụ để sử dụng

4. Xử lý Exception
   @ExceptionHandler: lắng nghe các Exception xảy ra trong controller (phạm vi hẹp)
   @ControllerAdvice: xử lý @ExceptionHandler được chia sẻ tại tất cả controller trong ứng dụng MVC
   @RestControllerAdvice : sử dụng với RESTFul (= @ControllerAdvice + @ResponseBody)

5. JSON Web Token

Trước khi đi vào tìm hiểu JSON Web Token, ta cần hiểu về hai khái niệm Stateful và Stateless

Stateful = state application + full : chứa đầy đủ state của application (thường sử dụng trong mô hình MVC)

Lưu trữ thông tin bên trong ứng dụng, ví dụ như thông tin người dùng đăng nhập

Session: phiên đăng nhập. Được dùng trong mô hình Stateful, cách mà ứng dụng lưu trữ data giữa các lời gọi Request

Stateless: state application + less: không chứa state của application (thường sử dụng trong mô hình REST – dùng để chia tách code Frontend + Backend)

Không lưu trữ thông tin trong ứng dụng

Trong mô hình Stateless, không tồn tại khái niệm "Session", thay vào đấy là : "Token"

- Cơ chế xác thực dựa vào Session(Stateful):

Bước 1: login với username/password

Nếu login thành công, server sẽ tạo và lưu thông tin tại:

  - Client thông qua cookies(lưu SESSION_ID)
  - Server trong memory (RAM) hoặc database

Bước 2: Mỗi lần người dùng F5(refresh) website gửi 1 request từ client lên server, các bước làm tại Server:

  - Client sẽ gửi kèm SESSION_ID (thông qua cookies)
  - Server sẽ kiểm tra SESSION_ID có đang tồn tại không? Nếu có tiếp tục truy cập bình thường, không thì sẽ cho out

Mô hình Stateful với Session chỉ áp dụng hiệu quả khi người lập trình viên muốn kiểm soát cả Frontend và Backend

- Cơ chế xác thực dựa vào Token (Stateless)

Server với Server, mobile app, desktop app,…

Token: là một chuỗi ký tự đã được mã hóa (chỉ Server mới có thể hiểu)

Bước 1: login với username/password

Nếu login thành công, server sẽ tạo token, lưu tại đâu, client sẽ tự quyết định

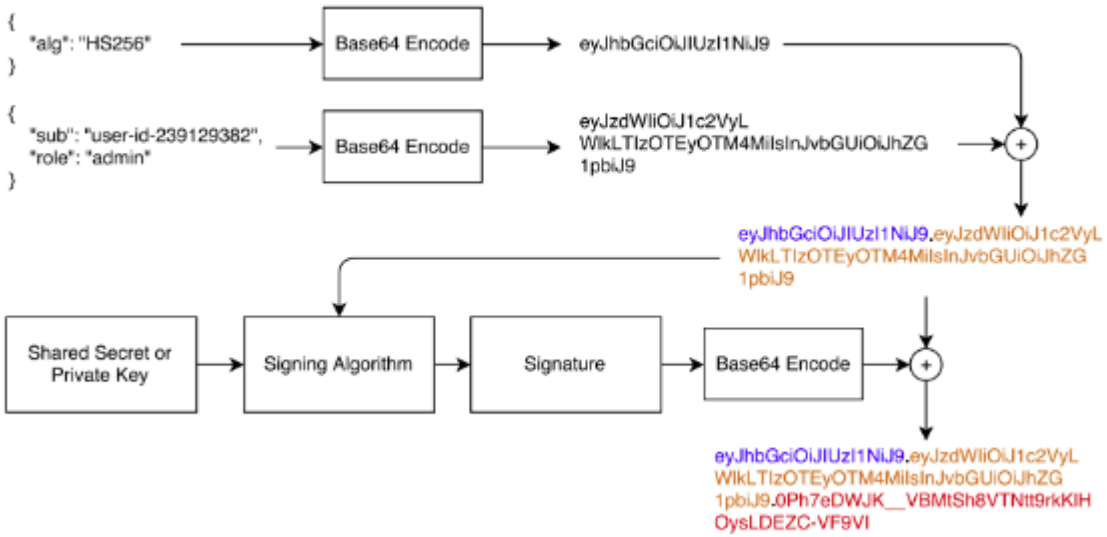
Server không lưu bất cứ thông tin về việc user login

Bước 2: Mỗi lần người dùng F5(refresh) website gửi 1 request từ client lên server, sẽ cần gửi kèm token đã có tại bước 1. Server sẽ giải mã token để biết được user có hợp lệ hay không

Bây giờ ta sẽ đi tìm hiểu về JSON Web Token. JSON Web Token được viết tắt là JWT là một chuỗi ký tự được mã hóa thông qua thuật toán và có tính bảo mật cao. Được sử dụng để trao đổi thông tin giữa các hệ thống với nhau (server-server,client-server)

Cấu trúc của JWT: **header**.**payload**.**signature**

- **Header** (chứa thông tin về thuật toán mã hóa): được encode dưới dạng base64
- **Payload** (chứa data client): encode dưới dạng base64
- **Signature** : được tạo nên từ thuật toán mã hóa +(header+payload)+key dưới dạng base64. Như vậy signature gồm 3 thành phần: thuật toán mã hóa, data của (header+payload) và key(có thể là mật khẩu/ hoặc sử dụng private/public key)



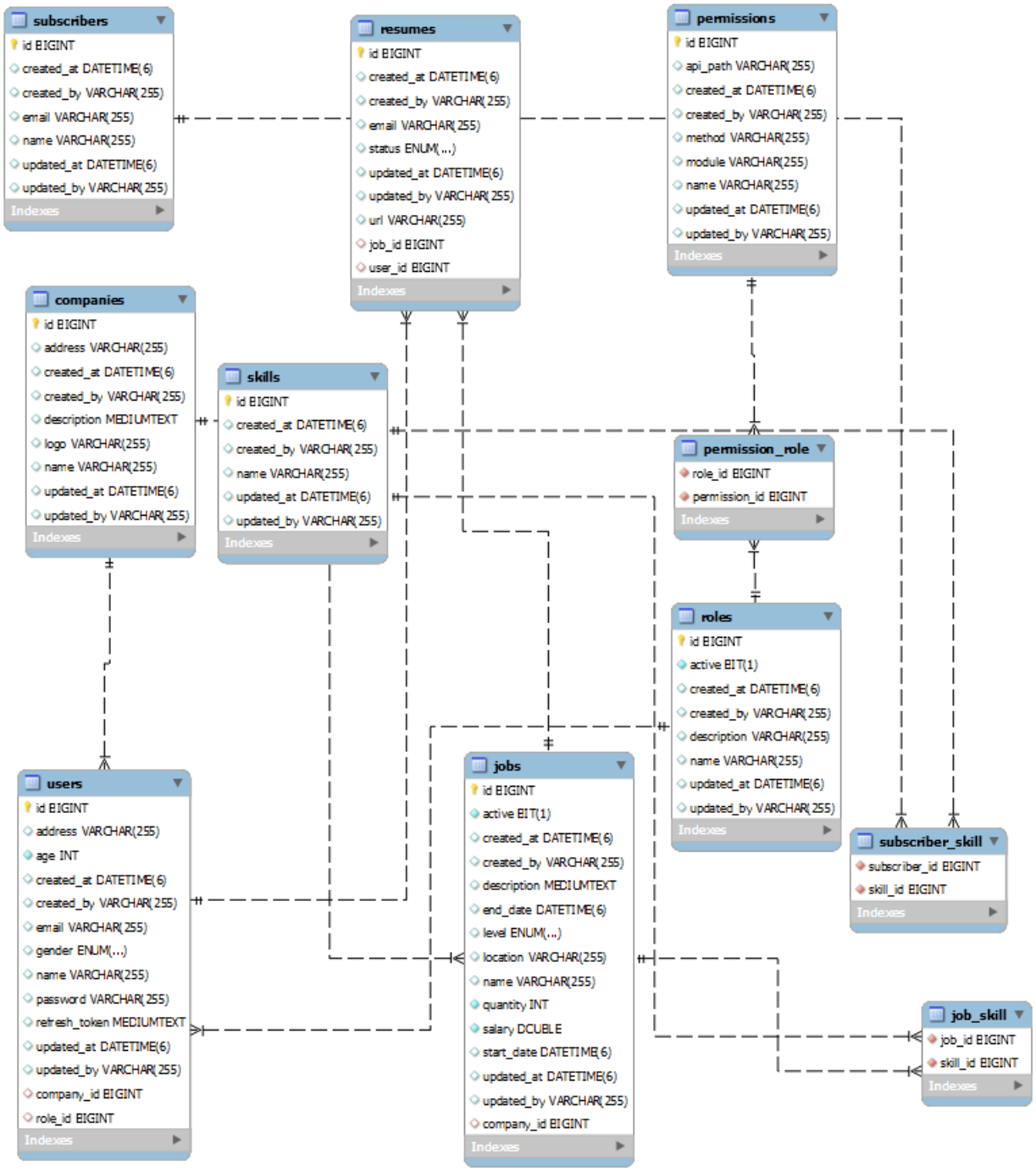
Quá trình tạo ra 1 JWT

Key được tạo ra với một mục đích, cho dù JWT bị lộ ra, nếu không có key → không thể giải mã token được. Có 2 hình thức tạo key phổ biến: một là dùng mật khẩu(hash) và hai là dùng public/private key

OAuth là một chuẩn dùng để xác thực người dùng thông qua token

II. Các Module chính

Phân tích thiết kế:



Các quy tắc viết API:

- Cần thêm tiền tố để biết được version API hiện tại. Ví dụ: /api/v1/...

- Đối với những Entity có quan hệ cha con với nhau, khi viết cần tuân theo thứ tự thực thể cha rồi mới đến thực thể con. Ví dụ: /api/v1/PTIT/B22DCCN482
- Tuân thu RESTful Standards:
    - Sử dụng các phương thức HTTP chuẩn
    - Sử dụng đường dẫn URL rõ ràng
    - Hỗ trợ các trạng thái mã HTTP

1. Modules Company

Model company gồm các trường như hình bên dưới

Id	Long
Name	String
Description	String
Address	String
Logo	String
createdAt	Date
updatedAt	Date
createdBy	String
updatedBy	String

Các API liên quan:

Chức năng	Phương thức	API
Tạo mới 1 công ty	POST	/api/v1/companies
Lấy toàn bộ thông tin công ty	GET	/api/v1/companies
Cập nhật thông tin công ty	PUT	/api/v1/companies
Xóa 1 công ty	DELETE	/api/v1/companies/{id}
Lấy thông tin 1 công ty theo id	GET	/api/v1/companied/{id}

2. Modules User

Model User gồm các trường thông tin như bên dưới:

Id	Long
Name	String
Email	String
Password	String
Age	Int
Gender	String
Address	String
refreshToken	String
createdAt	Date
updatedAt	Date
createdBy	String
updatedBy	String

Các API liên quan:

Chức năng	Phương thức	API
Tạo mới 1 User	POST	/api/v1/users
Xóa 1 User	DELETE	/api/v1/users/{id}
Lấy thông tin User	GET	/api/v1/users/{id}
Lấy thông tin toàn bộ User	GET	/api/v1/users
Cập nhật User	PUT	/api/v1/users
Lấy account user	GET	/api/v1/auth/account
Lấy được access_token	POST	/api/v1/auth/login
Lấy access_token mới	GET	/api/v1/auth/refresh
Logout	POST	/api/v1/auth/logout

Đối với API tạo mới 1 user, cần kiểm tra email đã tồn tại trong hệ thống chưa. Nếu đã tồn tại thông báo lỗi

```
@PostMapping("/users")
@ApiMessage("Create a new user")
public ResponseEntity<ResCreateUserDTO> createNewUser(@Valid @RequestBody User user) throws IdInvalidException {

    boolean isEmailExist = this.userService.isEmailExist(user.getEmail());
    if (isEmailExist) {
        throw new IdInvalidException("Email " + user.getEmail() + " đã tồn tại trong hệ thống.");
    }
    String hashPassword = this.passwordEncoder.encode(user.getPassword());
```

Đối với API login: Nếu người dùng đăng nhập thành công, backend sẽ làm 2 việc:

- Trả về phản hồi cho API, bao gồm access_token và thông tin của user, theo format sau:

{

Access_token: “JSON WEB TOKEN”

User: {

Email:trinhquanglam2k4@gmail.com,

Name:”trinhquanglam”,

Id:”1”}

}

- Ngoài ra, refresh_token sẽ lưu vào cookies

Với browser, để lưu trữ data của user có 3 cách hay dùng nhất:

Local Storage: thông tin được lưu trữ mãi mãi

Session Storage: khi đóng browser là thông tin lưu trữ sẽ clear

Cookies: chỉ mất khi và chỉ khi “bị hết hạn” (không liên quan gì tới việc đóng browser)

Cơ chế thường dùng trong mô hình stateless :

Người dùng sau khi login thành công, sẽ được server trả về:

Access_token: token để định danh người dùng (thời gian sống ngắn)

Refresh_token: nếu access_token bị hết hạn, sử dụng refresh token để renew (thời gian sống lâu hơn)

⇨ Access_token được lưu tại local storage (FE dễ dàng truy cập và sử dụng). Đặt thời gian sống ngắn để giảm thiểu rủi ro

Refresh_token được lưu lại cookies với mục đích là server sử dụng (vì cookies luôn được gửi kèm với mỗi lời gọi request) → lưu ở cookies sẽ an toàn hơn

Giải thích cơ chế JWT:

Do sử dụng mô hình stateless, nên không thể sử dụng session, chúng ta sử dụng token để định danh người dùng

Để truy cập endpoint (APIs) tạo backend, đối với mỗi lời gọi, frontend cần truyền thêm Token (gọi là access_token). Token này sẽ được gán ở header mỗi request (bearer token)

Access_token này có thời gian sống ngắn để đảm bảo an toàn, đồng thời được viết dưới dạng JWT

Ví dụ về access_token:
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJob2lkYW5pdEBnbWFpbC5jb20iLCJleHAiOjE3MjQwNTkzNTEsImlhdCI6MTcxNTQxOTM1MSwidXNlciI6eyJpZCI6MSwiZW1haWwiOiJob2lkYW5pdEBnbWFpbC5jb20iL

CJuYW1lIjoiSOG7j2kgRMOibiBJVCJ9fQ.0V3Ss4AWnHf8f6HtSEWKlchHSJAmHW0ef3WUtDj2fn0vGY
EjbyqIO2nor0lq06DTcAOAthT9BBvQ3gD9M0jThw

3. Modules Job/Resum

4. Modules Permission & Role

5. Modules Subscriber

III. Triển khai

1. Công cụ sử dụng

Ngôn ngữ : Java

Backend: Java Spring

- Spring Boot: cấu hình và chạy dự án một cách nhanh chóng
- Spring Security: xác thực và phân quyền người dùng với JWT sử dụng cơ chế oauth2
- Spring JPA: xử lý và thao tác với cơ sở dữ liệu database với ORM

Frontend: React Vite (Typescript)

Database: MySQL

Build Tool: Gradle - Kotlin

Công cụ khác: Postman, Swagger, Lombok, Git

2. Phương pháp thực hiện

IV. Kết quả

Phụ lục 1: Hướng dẫn cài đặt và chạy ứng dụng

Tài liệu tham khảo