Client (gửi HTTP) -> SecurityConfig cấu hình các quy tắc bảo mật, bao gồm xác thực, ủy quyền, ..

-> Sử dụng SecurityFilter để xử lý việc xác thực người dùng

-> SecurityFilter lấy chuỗi token JWT từ yêu cầu HTTP, kiểm tra và xác thực token, xử lý yêu cầu HTTP

-> JWTTokenProvider (tạo và giải mã token)

Controller nhận yêu cầu từ client, xác thực người dùng (sử dụng JWT hoặc cơ chế khác), gọi các phương thức từ service để xử lý logic và trả về kết quả cho client.

\*com.project.movie.confifg

- BeanConfiguration: Cấu hình Bean

- JwtTokenProvider: Cấu hình JWT như tạo JWT, validate JWT, lấy dữ liệu sau khi giải mã JWT

- PasswordEncoderConfig: Cấu hình mã hóa password khi đăng ký

- SecurityConfig: cấu hình security

- SecurityFilter: xử lý các yêu cầu đến bằng cách kiểm tra xem người dùng đã đăng nhập chưa.

Nếu người dùng đã đăng nhập, ta cho phép yêu cầu đi tiếp tới các Controller, ngược lại ta quay lại trang đăng nhập.

SecurityFilter sẽ kiểm tra xem một yêu cầu cụ thể có được phép truy cập tài nguyên được yêu cầu hay không bằng cách xác minh thông tin đăng nhập và các quyền của người dùng.

\*com.project.movie.constant

- AppConstant: định nghĩa các biến hằng cho ứng dụng

\*com.project.movie.controller: chứa các file controller cho Auth, Genre, Movie, Review

\*\*\*package domain thường được sử dụng để chứa các đối tượng đại diện cho các thực thể trong hệ thống,

chẳng hạn như các đối tượng được lưu trong cơ sở dữ liệu hoặc các đối tượng được sử dụng trong logic xử lý.

\*com.project.movie.domain.common

- ApiResponse: đối tượng chứa thông tin phản hồi response

- ApiResponseStatus: đối tượng chứa status (mã phản hồi), message (thông điệp phản hồi)

- BaseDao: BaseDao là một interface hoặc abstract class chứa các phương thức cơ bản (CRUD) để truy cập và thao tác dữ liệu với cơ sở dữ liệu,

bao gồm các phương thức như tạo, đọc, cập nhật và xóa dữ liệu

\*com.project.movie.domain.dao: chứa các class định nghĩa các cột thuộc tính của các thực thể như User, Movie, Genre, Review trong database

\*com.project.movie.domain.dto: định nghĩa các đối tượng dùng để truyền dữ liệu giữa các thành phần khác nhau trong ứng dụng.

Các đối tượng DTO này được sử dụng để chuyển dữ liệu giữa lớp Controller và Service

\*com.project.movie.repository: chứa các repository của ứng dụng

\*com.project.movie.service: Chứa các class của từng đối tượng để thao tác dữ liệu với database

\*com.project.movie.util: chứa các class dùng để custom request được gửi từ client, response trả về từ server.

\*DAO: DAO (Data Access Object) là một mẫu thiết kế. mỗi đối tượng dữ liệu sẽ có một DAO tương ứng,

chịu trách nhiệm thực hiện các thao tác CRUD liên quan đến đối tượng đó

\*JPA: JPA (Java Persistence API): Là một chuẩn API trong Java để quản lý đối tượng trong cơ sở dữ liệu quan hệ.

JPA cung cấp một tập hợp các API và annotation để tương tác với cơ sở dữ liệu, thực hiện các thao tác CRUD và quản lý mối quan hệ giữa các đối tượng.

\*Repository: Repository thường được sử dụng để triển khai DAO và JPA. Repository cung cấp các phương thức để thao tác với cơ sở dữ liệu,

bao gồm truy vấn, thêm, sửa, xóa dữ liệu. Repository thường được định nghĩa bằng cách sử dụng JPA annotation và kế thừa các interface hoặc lớp abstract được cung cấp bởi JPA framework.

==> DAO là một mẫu thiết kế để tách biệt logic truy cập dữ liệu, JPA là một chuẩn API trong Java để quản lý đối tượng trong cơ sở dữ liệu quan hệ,

và repository là một thành phần trong kiến trúc của ứng dụng dùng để lưu trữ và truy xuất dữ liệu, thường triển khai DAO và sử dụng JPA.

\*Hibernate: Hibernate là một framework ORM (Object-Relational Mapping). Nó cung cấp một cách tiện lợi và mạnh mẽ

để ánh xạ đối tượng Java vào các bảng trong cơ sở dữ liệu, và thực hiện các thao tác CRUD

\*Lombok: một thư viện Java cung cấp các annotation để tự động tạo các phần mã nguồn như getter, setter, constructor và các phương thức khác cho các lớp Java.

---

\* Mô tả tương tác:

1. Client: Đại diện cho phía người dùng hoặc ứng dụng giao diện người dùng.

Client gửi yêu cầu HTTP (ví dụ: GET, POST, PUT, DELETE) tới server thông qua API (Application Programming Interface).

2. JWT (JSON Web Token): Được sử dụng để xác thực và ủy quyền người dùng. Khi người dùng đăng nhập thành công,

server sẽ tạo một JWT chứa thông tin về người dùng (ví dụ: ID, tên, vai trò) và ký số để đảm bảo tính xác thực.

JWT được gửi từ server tới client và được bao gồm trong các yêu cầu sau này để xác thực người dùng.

3. Security: Đây là một thành phần quan trọng trong hệ thống, được sử dụng để kiểm soát quyền truy cập và bảo vệ các tài nguyên.

Security có thể sử dụng các cơ chế xác thực (ví dụ: JWT, OAuth) để xác định danh tính của người dùng và kiểm tra quyền truy cập của họ.

4. Controller: Là thành phần trong mô hình MVC (Model-View-Controller) và được sử dụng để xử lý yêu cầu từ client.

Controller nhận yêu cầu từ client, xác thực người dùng (sử dụng JWT hoặc cơ chế khác),

gọi các phương thức từ service để xử lý logic và trả về kết quả cho client.

5. Service: Đây là lớp chứa logic kinh doanh của ứng dụng. Service nhận thông tin từ controller,

gọi các phương thức từ repository (hoặc DAO) để truy vấn và cập nhật dữ liệu trong cơ sở dữ liệu, thực hiện các thao tác xử lý logic và trả về kết quả cho controller.

6. Repository (DAO): Được sử dụng để tương tác với cơ sở dữ liệu. Repository cung cấp các phương thức để thực hiện truy vấn (ví dụ: SELECT, INSERT, UPDATE, DELETE) và trả về dữ liệu cho service.

Repository có thể sử dụng các công nghệ ORM (Object-Relational Mapping) như Hibernate để thực hiện truy vấn dễ dàng và giảm sự phụ thuộc vào cơ sở dữ liệu cụ thể.

7. DTO (Data Transfer Object): Là các đối tượng được sử dụng để truyền dữ liệu giữa client và server.

DTO giúp định nghĩa các trường dữ liệu cần truyền đi hoặc nhận về và giúp giảm kích thước dữ liệu truyền qua mạng và tăng tính bảo mật.

8. DAO (Data Access Object): Đây là một mẫu thiết kế (design pattern) được sử dụng để trừu tượng hóa việc truy cập dữ liệu từ lớp service.

DAO cung cấp các phương thức để truy xuất, thêm, sửa, xóa dữ liệu từ cơ sở dữ liệu.

Trong một ứng dụng sử dụng ORM, DAO thường được thay thế bằng Repository.

==> client gửi yêu cầu tới controller,

controller sử dụng service để xử lý logic và gọi repository (hoặc DAO) để truy cập dữ liệu từ cơ sở dữ liệu.

DTO được sử dụng để truyền dữ liệu giữa client và server, và JWT được sử dụng để xác thực và ủy quyền người dùng.

\* Token

1. JWTTokenProvider:

Class này là một thành phần cung cấp các phương thức để xử lý JWT (JSON Web Token).

Các phương thức chính:

setUpSecretKey(): Được gọi trong phương thức @PostConstruct, tạo khóa bí mật từ chuỗi key được cung cấp.

generateToken(): Tạo ra một chuỗi JWT từ thông tin xác thực và vai trò (role) của người dùng.

validateToken(): Xác thực và kiểm tra tính hợp lệ của token trong yêu cầu HTTP.

getUsername(): Trích xuất tên người dùng từ token.

getRole(): Trích xuất vai trò (role) của người dùng từ token.

Class này sử dụng thư viện io.jsonwebtoken để tạo và giải mã JWT.

2. PasswordEncoderConfig:

Class này định nghĩa một bean passwordEncoder() dùng để mã hóa mật khẩu.

Trong ví dụ này, sử dụng BCryptPasswordEncoder để mã hóa mật khẩu.

3. SecurityConfig:

Class này cấu hình bảo mật trong ứng dụng.

Các điểm chính:

Cấu hình CORS để cho phép các yêu cầu từ các nguồn khác nhau.

Cấu hình các quyền truy cập dựa trên URL và phương thức HTTP.

Sử dụng SecurityFilter để xử lý việc xác thực người dùng và gắn kết thông tin xác thực vào SecurityContextHolder.

4. SecurityFilter:

Class này là một OncePerRequestFilter để xử lý việc xác thực người dùng và gắn kết thông tin xác thực vào SecurityContextHolder cho mỗi yêu cầu đến server.

Các phương thức chính:

doFilterInternal(): Xử lý yêu cầu HTTP, kiểm tra và xác thực token, sau đó gắn kết thông tin xác thực vào SecurityContextHolder.

getJwtFromRequest(): Lấy chuỗi token JWT từ yêu cầu HTTP.