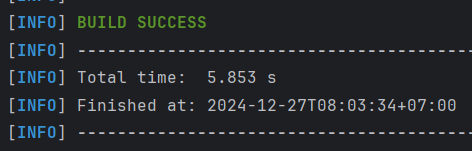
**TEAM CORE**

**Common-core** có các class, chức năng dùng chung cho toàn bộ ứng dụng, bao gồm các chức năng chính sau:

* Tự động điền các trường creadted\_at, created\_by, updated\_at, updated\_by trong các bảng trong db (auditing) thông qua class BaseEntity.
* Có 1 chuẩn trả về BaseResponse (class BaseResponse), có hàm tạo ResponseEntity OK, CREATED với các tham số được truyền vào
* Xử lý tập trung cho exception
* Convert BaseException thành ResponseEntity<BaseResponse> trong GlobalExceptionHandler class

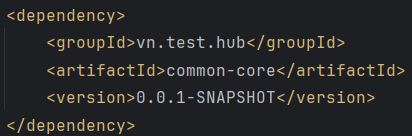
**Version: 0.0.1-SNAPSHOT**

Các bước sử dụng:  
**B1: Build project**

Sau khi tải toàn bộ code về, trên terminal chuyển vào thư mục common-core vừa tải thông qua lệnh cd.  
Sau đó chạy lệnh “mvn clean install”. Sau 1 thời gian trên terminal xuất hiện như sau  


**B2: Bổ xung dependency common-core vào các file cần thiết**

Tại file pom.xml root, pom.xml của từng module, thêm dependency common-core  
Ví dụ, thêm nội dung sau vào file pom.xml root



**B3. Sử dụng các tính năng  
1. Đối với auditing**

Tạo và cấu hình class BaseEntity với các trường thông tin chính (thông tin này không bao gồm Id, creadted\_at, created\_by, updated\_at, updated\_by và deleted; do các thuộc tính đã được định nghĩa trong BaseEntity class). Sau đó extend class vn.test.hub.common.entity.BaseEntity.

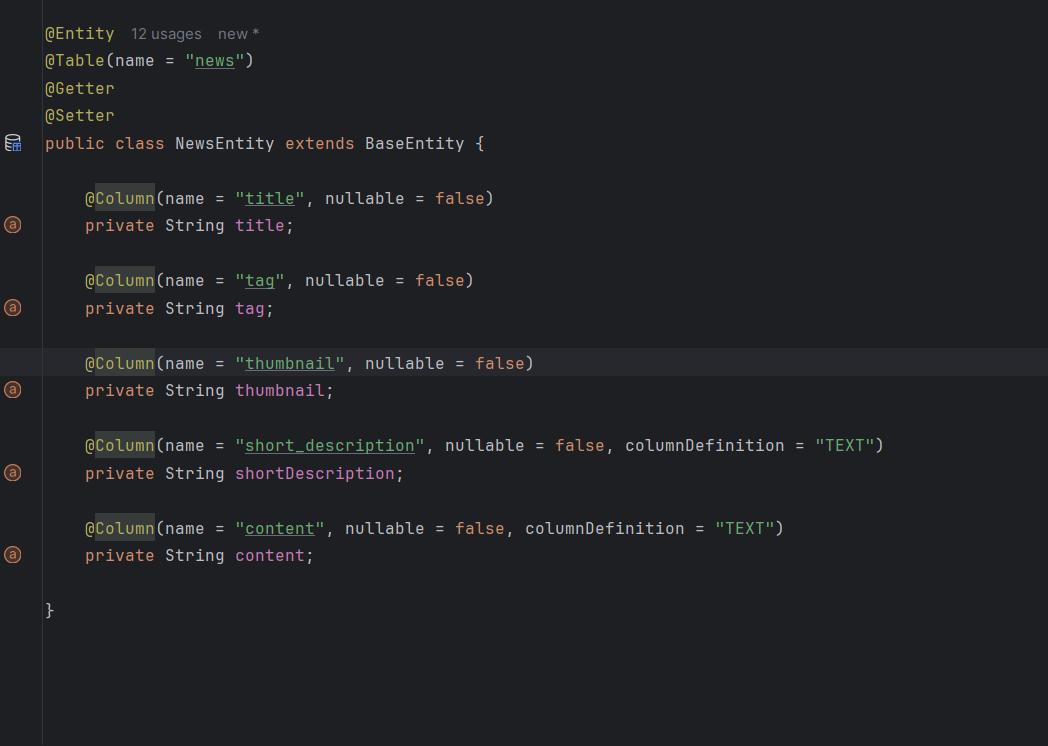
Các trường created\_at, created\_by, updated\_at, updated\_by sẽ được tự đọc tạo ra trường created\_at, updated\_at có giá trị default là “Anonymous” và sẽ được cập nhật sau khi thống nhất với Team Auth



**Class BaseEntity**

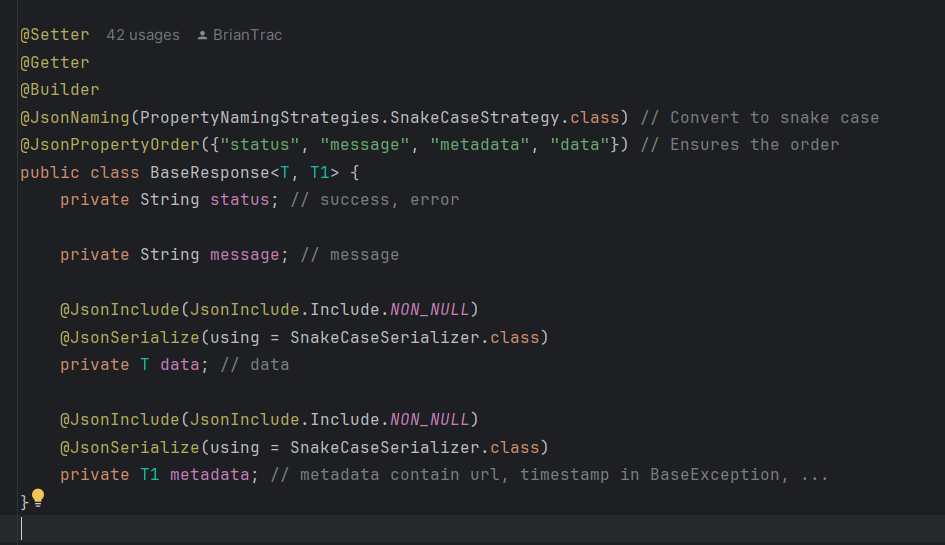


Ví dụ: class NewsEntity



**2. Chuẩn BaseResponse**

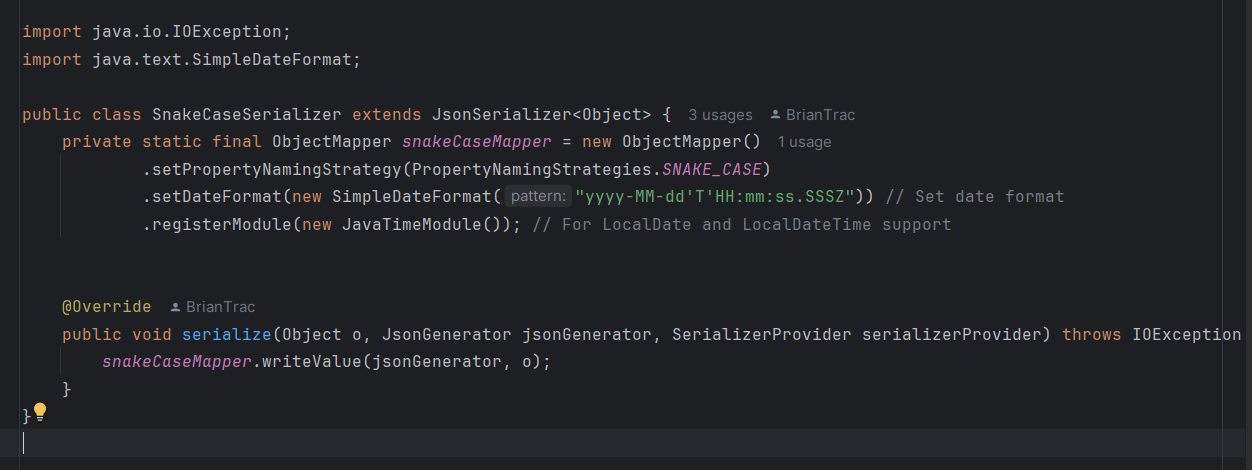
Class BaseResponse chứa các field được trả về dưới dạng Json theo thứ tự như sau



Trong đó:  
 - status: chỉ trạng thái thành công hay thất bại (success | error)

* message: mô tả trạng thái lỗi hoặc message khi thành công
* Data: dữ liệu trả về
* Metadata: dữ liệu bổ sung cho data, nếu là exception thì mặc định sẽ gồm các trường code, url, và timestamp.

- Utils SnakeCaseSerializer để convert nested object thành snake\_case. Bởi vì mặc định, Jackson's @JsonNaming không có propagate đến nested objects, vì vậy cần 1 serializer để làm điều đó



1. **Sử dụng Pagination**

+, Nếu có thông tin phân trang thì có các trường sau trong class PaginationInfo trong đó (

limit: số items trên 1 page;

curentPage: trang hiện tại;

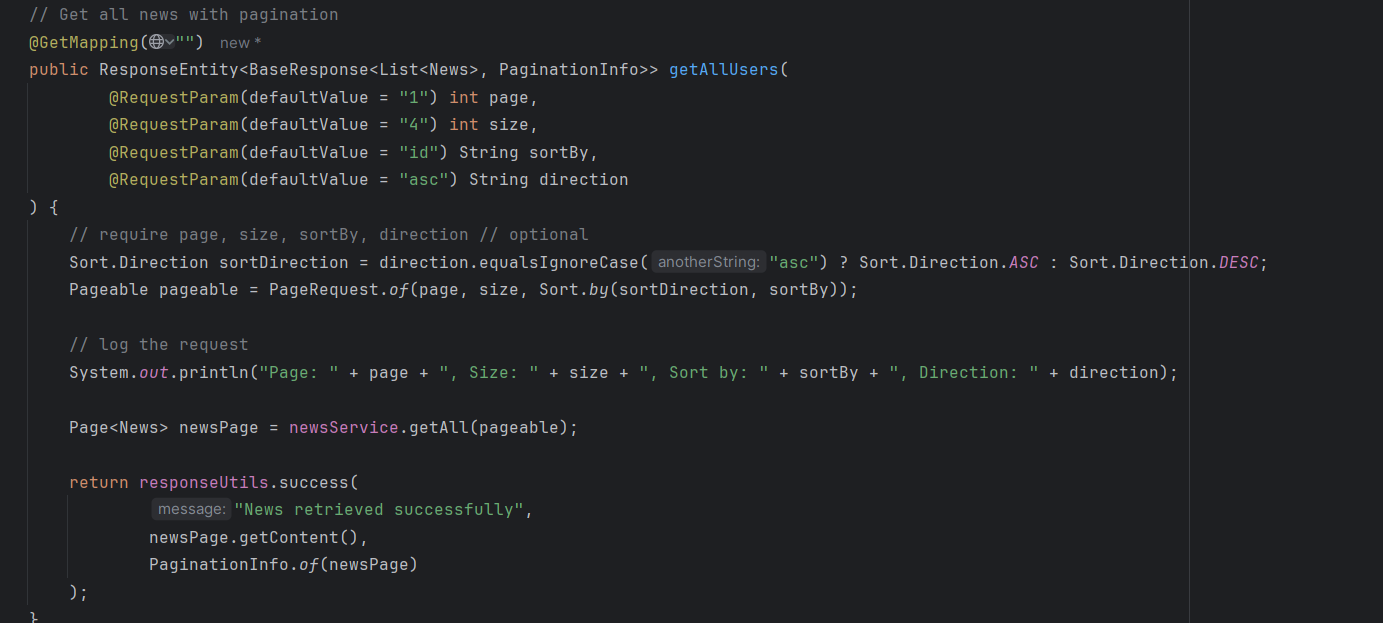
totalPages: tổng số trang;

)

+ Trong PaginationInfo có method **of** trả về PaginationInfo, khi mà user chuyền vào org.springframework.data.domain.Page<?> nó sẽ giúp mình trả về PaginationInfo

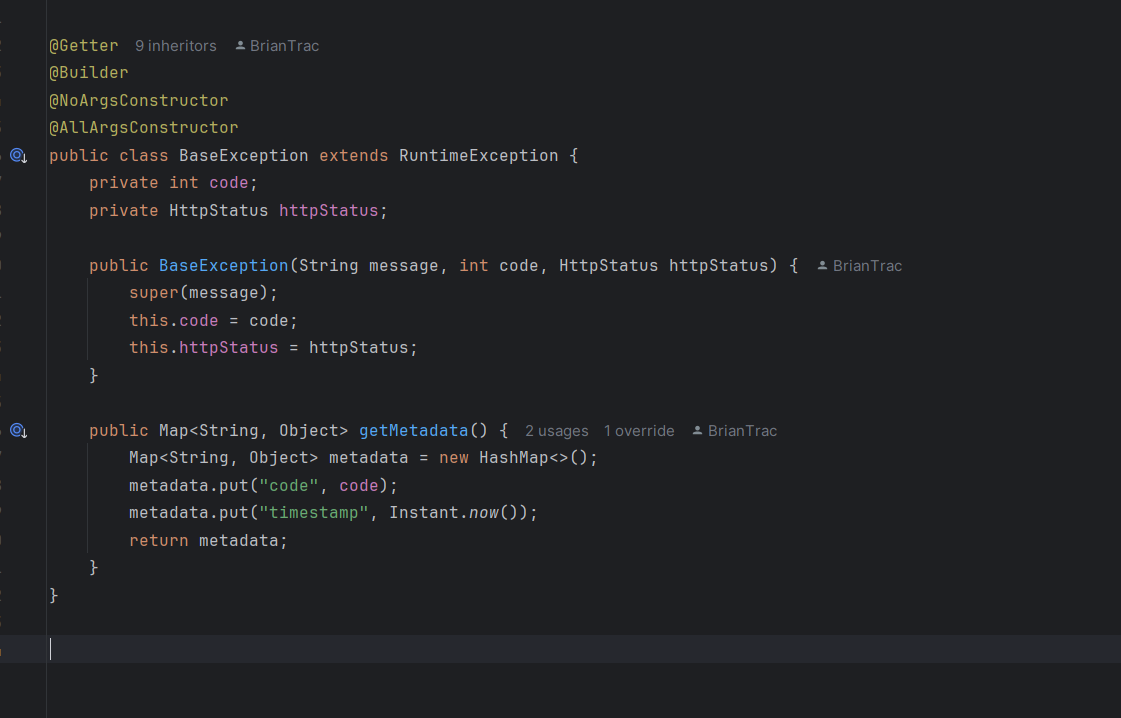


Ví dụ: sử dụng PaginationInfo với News service

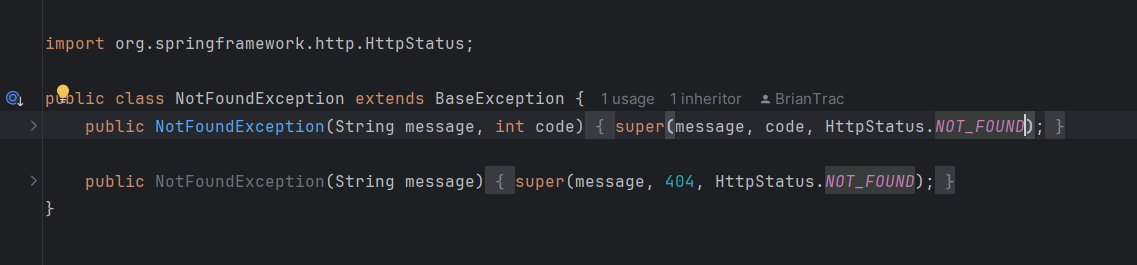


1. **Tự định nghĩa Exception:**

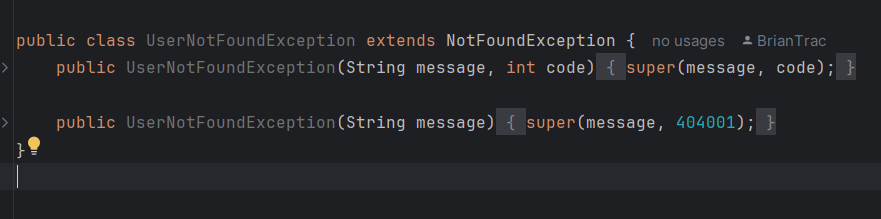
+ User có thể tự định nghĩa custom Exception cho riêng mình dựa vào BaseException

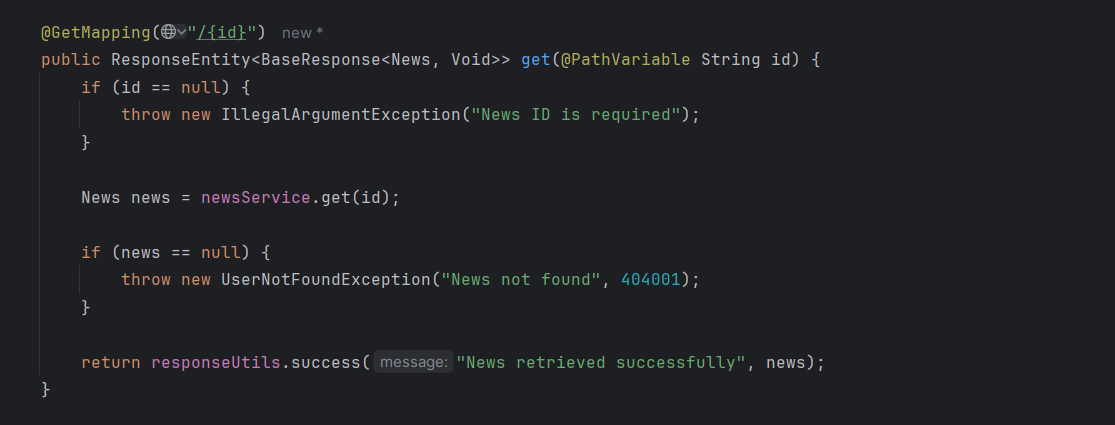


+ Sử dụng một exception đã có sẵn trong common-core có thể chuyền vào message và code để tự định nghĩa lại message và code; hoặc có thể chỉ cần chuyền vào message thì code sẽ lấy giá trị mặc định trong common-core

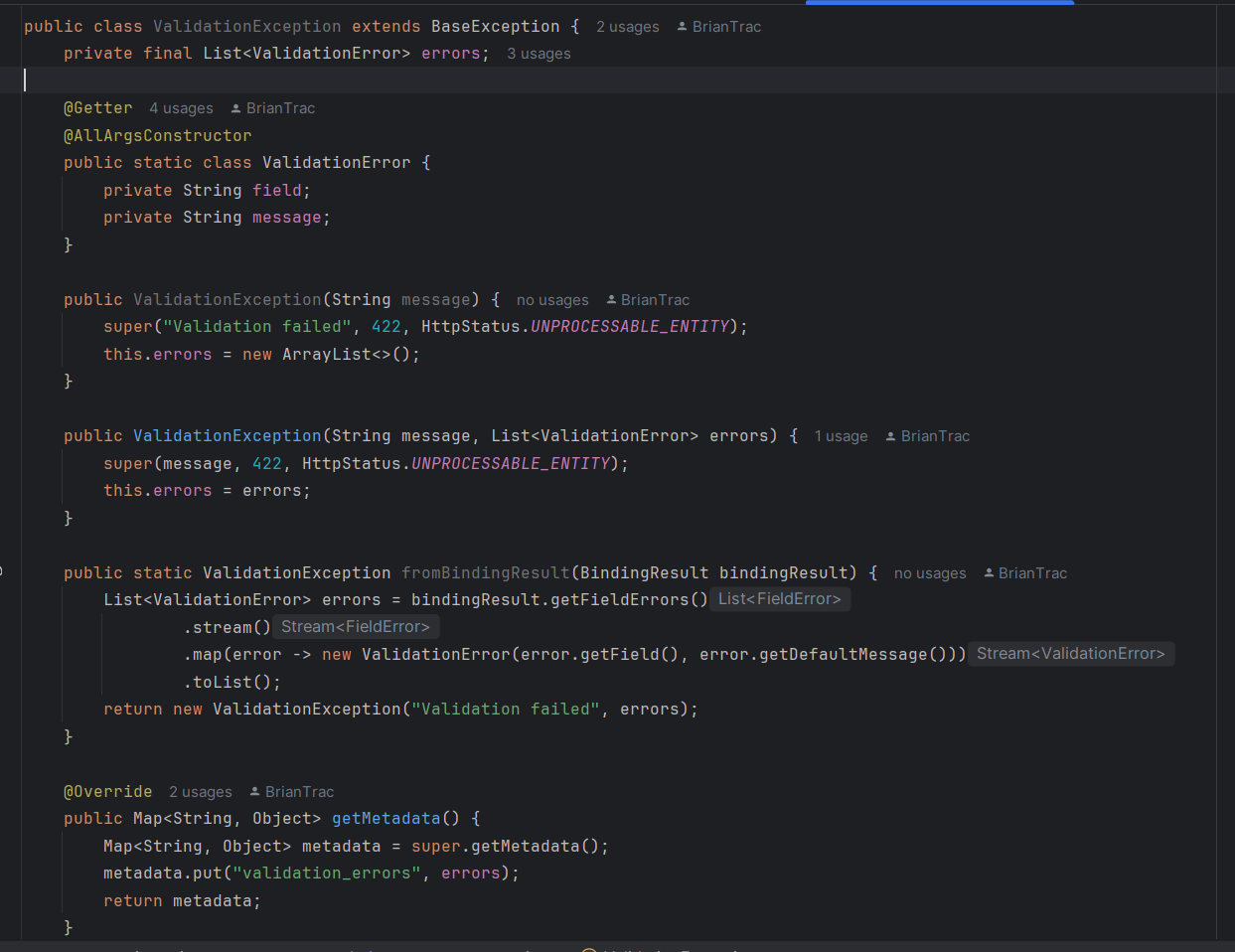


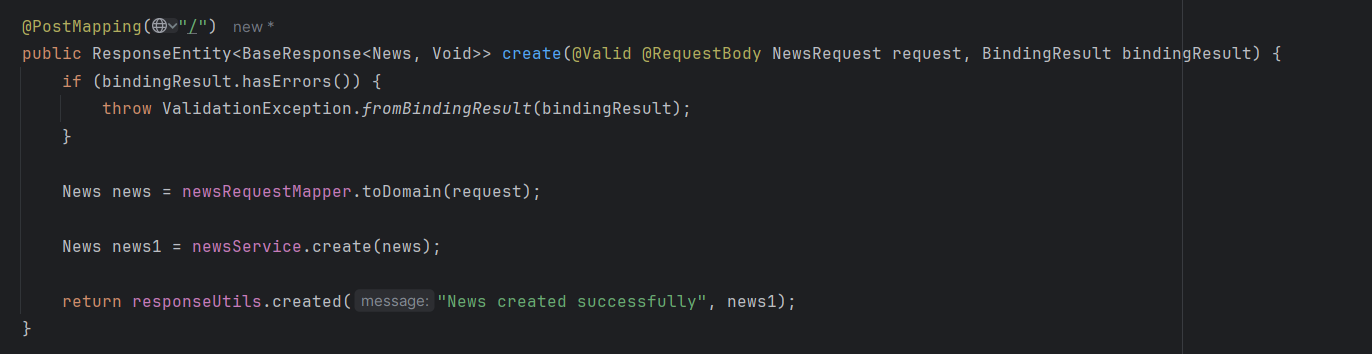
+ Nếu muốn thêm một Exception mới như UserNotFoundException người dùng có thể định nghĩa như sau:





+ Ví dụ khác phức tạp hơn về ValidationException (được ném ra khi quy phạm @NotBlank, @Size trong @Valid của import jakarta.validation.Valid; )





+ User có thể mở rộng **metadata** bằng cách @Override hàm getMetadata() ở class BaseException như ví dụ trên

+ Các Exception đã được định nghĩa sẵn:

**BadRequestException(400)**: Được sử dụng khi yêu cầu từ client không hợp lệ hoặc thiếu thông tin bắt buộc.

**ConflictException(409)**: Dùng khi có xung đột xảy ra, chẳng hạn dữ liệu gửi lên đã tồn tại (ví dụ: tạo một tài khoản với email đã tồn tại).

**ForbiddenException(403)**: Xảy ra khi người dùng không có quyền truy cập vào tài nguyên hoặc hành động yêu cầu.

**IllegalArgumentException(400):**Sử dụng khi một đối số (argument) truyền vào không hợp lệ hoặc không thỏa mãn điều kiện logic

**NotFoundException(404):** Được sử dụng khi tài nguyên yêu cầu không tồn tại hoặc không tìm thấy.

**TooManyRequestsException(429):** Dùng khi người dùng gửi quá nhiều yêu cầu trong một khoảng thời gian giới hạn (thường liên quan đến rate limiting).

**UnauthorizedException(401):** Xảy ra khi người dùng chưa được xác thực hoặc thông tin xác thực không hợp lệ.

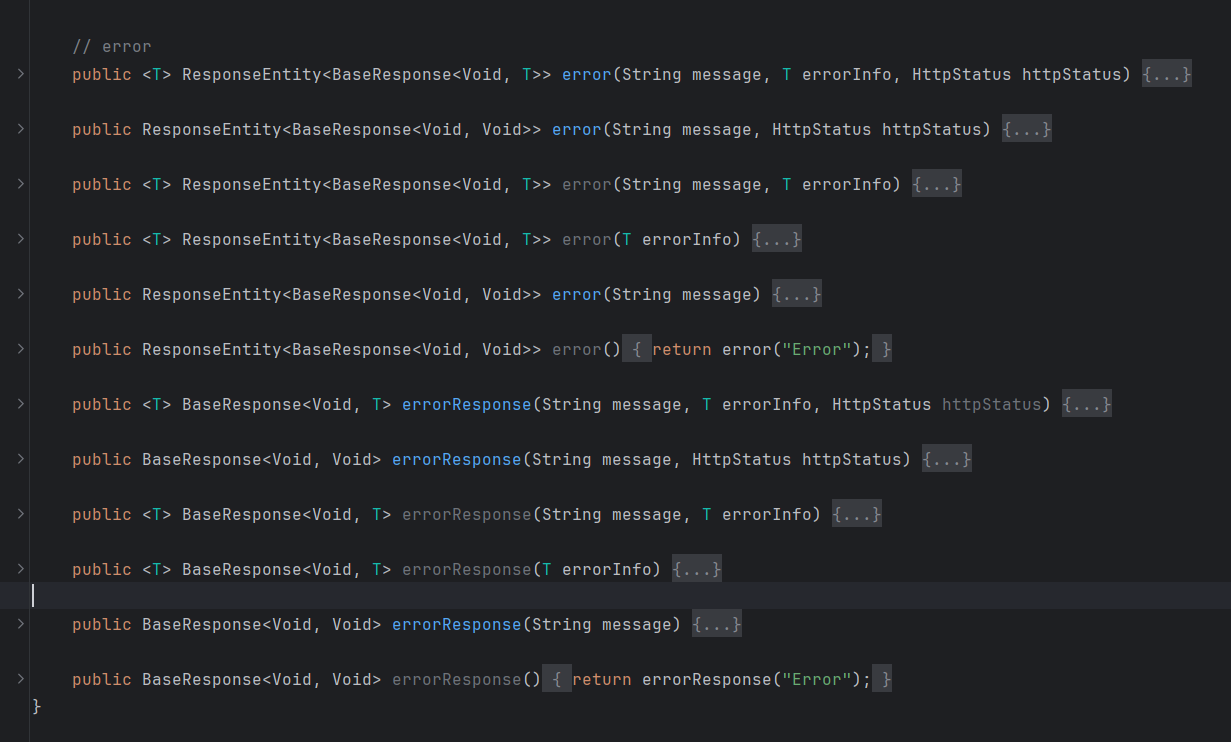
**ValidationException(422):** Xảy ra khi dữ liệu gửi từ client không hợp lệ, chẳng hạn vi phạm quy tắc kiểm tra dữ liệu (validation rules).

**UserNotFoundException(404001):** Một dạng cụ thể của NotFoundException, được sử dụng khi người dùng không tìm thấy do business quy định, thường đi kèm mã lỗi chi tiết (404001)

1. **Hàm ResponseUtils**

Gồm các method: generateSuccessResponse, generateCreatedResponse, generateErrorResponse với các tham số message, data, metadata (đã được overloading)





Sử dụng ResponseUtils cần phải báo private final ResponseUtils responseUtils

