*Link git security: https://github.com/ngoctan1234/Spring\_security\_Auth\_Bearer.git*

*Link source api:* [*https://github.com/ngoctan1234/Source\_springboot\_api.git*](https://github.com/ngoctan1234/Source_springboot_api.git)

*Link sql: https://drive.google.com/file/d/1zn-2CIa5d6ZWfM8xfDpnpQB94n62Xe5v/view?usp=sharing*

*Luồng xác thực trong Spring Security thường bao gồm một chuỗi các bước từ khi một yêu cầu được gửi đến khi người dùng được xác thực và cho phép truy cập vào tài nguyên. Dưới đây là quy trình cơ bản:*

### ***1. Nhận yêu cầu***

* *Khi một yêu cầu đến ứng dụng, Spring Security sẽ xử lý yêu cầu đó thông qua chuỗi bộ lọc (filter chain).*

### ***2. Áp dụng bộ lọc xác thực***

*Spring Security sẽ kiểm tra các bộ lọc đã được định nghĩa, thường là:*

* ***UsernamePasswordAuthenticationFilter****: Dùng để xác thực người dùng qua username và password.*
* ***JwtAuthenticationFilter****: Nếu sử dụng JWT, bộ lọc này sẽ kiểm tra token trong tiêu đề của yêu cầu.*

### ***3. Kiểm tra thông tin xác thực***

*Trong bộ lọc xác thực, thông tin người dùng (như username và password hoặc token) sẽ được lấy từ yêu cầu và kiểm tra:*

* ***Xác thực bằng username và password****:*
  + *Nếu thông tin hợp lệ, một đối tượng Authentication sẽ được tạo ra và lưu trữ trong SecurityContextHolder.*
* ***Xác thực bằng token****:*
  + *Token sẽ được kiểm tra tính hợp lệ (có thể sử dụng thư viện JWT).*
  + *Nếu token hợp lệ, một đối tượng Authentication tương ứng sẽ được tạo ra.*

### ***4. Cập nhật SecurityContext***

*Khi xác thực thành công, Spring Security sẽ cập nhật SecurityContext với thông tin người dùng đã xác thực. Điều này giúp hệ thống biết rằng người dùng đã được xác thực cho các yêu cầu tiếp theo.*

### ***5. Xử lý quyền truy cập***

*Sau khi xác thực, Spring Security sẽ kiểm tra các quyền truy cập (authorization) dựa trên các cấu hình đã định nghĩa trong ứng dụng. Điều này thường dựa trên vai trò (roles) hoặc quyền (authorities) của người dùng.*

### ***6. Trả về phản hồi***

*Nếu người dùng có quyền truy cập, ứng dụng sẽ xử lý yêu cầu và trả về phản hồi cho người dùng. Nếu không, một phản hồi lỗi sẽ được trả về (ví dụ: 403 Forbidden).*

### ***Sơ đồ luồng xác thực***

*1. Yêu cầu đến ứng dụng*

*↓*

*2. Bộ lọc xác thực*

*↓*

*3. Kiểm tra thông tin xác thực*

*├── Nếu thành công:*

*│ ├── Tạo đối tượng Authentication*

*│ └── Cập nhật SecurityContext*

*↓*

*4. Kiểm tra quyền truy cập*

*↓*

*5. Phản hồi*

*2. OncePerRequestFilter*

* *OncePerRequestFilter trong Spring Security được sử dụng để đảm bảo rằng một bộ lọc chỉ chạy một lần cho mỗi yêu cầu HTTP.*
* *Điều này rất quan trọng trong bối cảnh xác thực, nơi bạn muốn thực hiện một số thao tác như xác thực người dùng hoặc kiểm tra token mà không cần phải lặp lại chúng trong các bộ lọc khác.*
* *doFilterInternal: Phương thức này thực hiện logic chính của bộ lọc. Nó kiểm tra xem yêu cầu có cần xác thực hay không.*

*// Kiểm tra nếu yêu cầu không cần xác thực*

*if (isBypassToken(request)) {*

*filterChain.doFilter(request, response); // Bỏ qua xác thực*

*return;*

*}*

* *Cách kiểm tra xem header Authorization có tồn tại và có định dạng đúng không (bắt đầu bằng "Bearer "). Nếu không, trả về lỗi 401 Unauthorized.*

*final String authHeader = request.getHeader("Authorization");*

*if (authHeader == null || !authHeader.startsWith("Bearer ")) {*

*response.sendError(HttpServletResponse.SC\_UNAUTHORIZED, "authHeader null or not started with Bearer");*

*return;*

*}*

* *Phương thức isByPassToken:Phương thức này kiểm tra xem yêu cầu có phải là một trong những yêu cầu không cần xác thực hay không (ví dụ: đăng nhập hoặc đăng ký).*

*-*

*3. Giải thích lớp JwtTokenUtil,*

* *nơi bạn quản lý các hoạt động liên quan đến JWT (JSON Web Token)*
* *cung cấp các phương thức tiện ích để tạo, xác thực và trích xuất thông tin từ JWT. Điều này giúp quản lý xác thực người dùng một cách an toàn và hiệu quả trong ứng dụng*

1. *generateToken:*

* *Mục đích: Tạo một JWT cho người dùng.*
* *Claims: Thông tin bổ sung (như username) được lưu trữ trong token.*
* *Hết hạn: Token sẽ hết hạn sau một khoảng thời gian nhất định (được xác định bởi biến expiration).*
* *Ký: Token được ký bằng một khóa bí mật để đảm bảo tính toàn vẹn.*

1. *getSignInKey*

* ***Mục đích****: Lấy khóa bí mật để ký và xác thực token.*
* ***Chuyển đổi****: Khóa bí mật được mã hóa dưới dạng Base64 và được giải mã để sử dụng.*

1. *generateSecretKey*

* ***Mục đích****: Tạo ra một khóa bí mật ngẫu nhiên mới.*
* ***Kích thước****: Khóa này có độ dài 256-bit.*
* ***Mã hóa****: Khóa được mã hóa thành dạng Base64 để có thể lưu trữ hoặc truyền tải.*

1. *extractAllClaims*

* ***Mục đích****: Giải mã và lấy tất cả các claims từ token.*
* ***Xử lý lỗi****: Nếu token không hợp lệ, sẽ ném ra một ngoại lệ.*

1. *extractClaim*

* ***Mục đích****: Trích xuất một claim cụ thể từ token bằng cách sử dụng một hàm giải quyết (claimsResolver).*
* ***Tính linh hoạt****: Bạn có thể sử dụng hàm này để lấy bất kỳ claim nào bạn cần.*

1. *extractUsername*

* ***Mục đích****: Lấy username từ token (được lưu trong claim subject).*
* ***Dễ sử dụng****: Dùng phương thức extractClaim để truy cập claim này*

1. *validateToken*

* ***Mục đích****: Xác minh xem token có hợp lệ hay không.*
* ***Điều kiện****: Token phải chứa đúng số điện thoại của người dùng và chưa hết hạn.*

*CREATE TABLE users(*

*id INT PRIMARY KEY AUTO\_INCREMENT,*

*fullname VARCHAR(100) DEFAULT '',*

*phone\_number VARCHAR(10) NOT NULL,*

*address VARCHAR(200) DEFAULT '',*

*password VARCHAR(100) NOT NULL DEFAULT '',*

*created\_at DATETIME,*

*updated\_at DATETIME,*

*is\_active TINYINT(1) DEFAULT 1,*

*date\_of\_birth DATE,*

*facebook\_account\_id INT DEFAULT 0,*

*google\_account\_id INT DEFAULT 0*

*);*

*ALTER TABLE users ADD COLUMN role\_id INT;*

*CREATE TABLE roles(*

*id INT PRIMARY KEY,*

*name VARCHAR(20) NOT NULL*

*);*

*ALTER TABLE users ADD FOREIGN KEY (role\_id) REFERENCES roles (id);*

*CREATE TABLE tokens(*

*id int PRIMARY KEY AUTO\_INCREMENT,*

*token varchar(255) UNIQUE NOT NULL,*

*token\_type varchar(50) NOT NULL,*

*expiration\_date DATETIME,*

*revoked tinyint(1) NOT NULL,*

*expired tinyint(1) NOT NULL,*

*user\_id int,*

*FOREIGN KEY (user\_id) REFERENCES users(id)*

*);*

*--hỗ trợ đăng nhập từ Facebook và Google*

*CREATE TABLE social\_accounts(*

*id INT PRIMARY KEY AUTO\_INCREMENT,*

*provider VARCHAR(20) NOT NULL COMMENT 'Tên nhà social network',*

*provider\_id VARCHAR(50) NOT NULL,*

*email VARCHAR(150) NOT NULL COMMENT 'Email tài khoản',*

*name VARCHAR(100) NOT NULL COMMENT 'Tên người dùng',*

*user\_id int,*

*FOREIGN KEY (user\_id) REFERENCES users(id)*

*);*

*--Bảng danh mục sản phẩm(Category)*

*CREATE TABLE categories(*

*id INT PRIMARY KEY AUTO\_INCREMENT,*

*name varchar(100) NOT NULL DEFAULT '' COMMENT 'Tên danh mục, vd: đồ điện tử'*

*);*

*--Bảng chứa sản phẩm(Product): "laptop macbook air 15 inch 2023", iphone 15 pro,...*

*CREATE TABLE products (*

*id INT PRIMARY KEY AUTO\_INCREMENT,*

*name VARCHAR(350) COMMENT 'Tên sản phẩm',*

*price FLOAT NOT NULL CHECK (price >= 0),*

*thumbnail VARCHAR(300) DEFAULT '',*

*description LONGTEXT DEFAULT '',*

*created\_at DATETIME,*

*updated\_at DATETIME,*

*category\_id INT,*

*FOREIGN KEY (category\_id) REFERENCES categories (id)*

*);*

*CREATE TABLE product\_images(*

*id INT PRIMARY KEY AUTO\_INCREMENT,*

*product\_id INT,*

*FOREIGN KEY (product\_id) REFERENCES products (id),*

*CONSTRAINT fk\_product\_images\_product\_id*

*FOREIGN KEY (product\_id)*

*REFERENCES products (id) ON DELETE CASCADE,*

*image\_url VARCHAR(300)*

*);*

*--Đặt hàng - orders*

*CREATE TABLE orders(*

*id INT PRIMARY KEY AUTO\_INCREMENT,*

*user\_id int,*

*FOREIGN KEY (user\_id) REFERENCES users(id),*

*fullname VARCHAR(100) DEFAULT '',*

*email VARCHAR(100) DEFAULT '',*

*phone\_number VARCHAR(20) NOT NULL,*

*address VARCHAR(200) NOT NULL,*

*note VARCHAR(100) DEFAULT '',*

*order\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,*

*status VARCHAR(20),*

*total\_money FLOAT CHECK(total\_money >= 0)*

*);*

*ALTER TABLE orders ADD COLUMN `shipping\_method` VARCHAR(100);*

*ALTER TABLE orders ADD COLUMN `shipping\_address` VARCHAR(200);*

*ALTER TABLE orders ADD COLUMN `shipping\_date` DATE;*

*ALTER TABLE orders ADD COLUMN `tracking\_number` VARCHAR(100);*

*ALTER TABLE orders ADD COLUMN `payment\_method` VARCHAR(100);*

*--xóa 1 đơn hàng => xóa mềm => thêm trường active*

*ALTER TABLE orders ADD COLUMN active TINYINT(1);*

*--Trạng thái đơn hàng chỉ đc phép nhận "một số giá trị cụ thể"*

*ALTER TABLE orders*

*MODIFY COLUMN status ENUM('pending', 'processing', 'shipped', 'delivered', 'cancelled')*

*COMMENT 'Trạng thái đơn hàng';*

*CREATE TABLE order\_details(*

*id INT PRIMARY KEY AUTO\_INCREMENT,*

*order\_id INT,*

*FOREIGN KEY (order\_id) REFERENCES orders (id),*

*product\_id INT,*

*FOREIGN KEY (product\_id) REFERENCES products (id),*

*price FLOAT CHECK(price >= 0),*

*number\_of\_products INT CHECK(number\_of\_products > 0),*

*total\_money FLOAT CHECK(total\_money >= 0),*

*color VARCHAR(20) DEFAULT ''*

*);*

*JWT*

*JSON Web Token (JWT) là một phương thức phổ biến để xác thực và truyền tải thông tin an toàn giữa client và server. Dưới đây là tổng quan về cách JWT hoạt động trong Spring Boot:*

### ***1. Cấu trúc của JWT***

*JWT bao gồm ba phần:*

* ***Header****: Thông tin về loại token và thuật toán mã hóa (ví dụ: HMAC SHA256).*
* ***Payload****: Chứa thông tin mà bạn muốn truyền tải, như user ID, quyền hạn, và thời gian hết hạn.*
* ***Signature****: Được tạo ra bằng cách mã hóa header và payload với một khóa bí mật.*
* *Ví dụ: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c*

*2. Cài đặt*

1. *Thư viện*

*<dependency>*

*<groupId>io.jsonwebtoken</groupId>*

*<artifactId>jjwt-jackson</artifactId>*

*<version>0.12.6</version>*

*<scope>runtime</scope>*

*</dependency>*

*<dependency>*

*<groupId>io.jsonwebtoken</groupId>*

*<artifactId>jjwt-api</artifactId>*

*<version>0.12.6</version>*

*</dependency>*

*<dependency>*

*<groupId>io.jsonwebtoken</groupId>*

*<artifactId>jjwt-impl</artifactId>*

*<version>0.12.6</version>*

*<scope>runtime</scope>*

*</dependency>*

1. *Cấu hình JWT trong spring boot*

*import io.jsonwebtoken.Claims;*

*import io.jsonwebtoken.JwtBuilder;*

*import io.jsonwebtoken.Jwts;*

*import io.jsonwebtoken.SignatureAlgorithm;*

*import org.springframework.stereotype.Component;*

*import java.util.Date;*

*import java.util.HashMap;*

*import java.util.Map;*

*@Component*

*public class JwtUtil {*

*private final String SECRET\_KEY = "your\_secret\_key";*

*private final long EXPIRATION\_TIME = 1000 \* 60 \* 60; // 1 hour*

*public String generateToken(String username) {*

*Map<String, Object> claims = new HashMap<>();*

*return createToken(claims, username);*

*}*

*private String createToken(Map<String, Object> claims, String subject) {*

*long now = System.currentTimeMillis();*

*Date issuedAt = new Date(now);*

*Date expiration = new Date(now + EXPIRATION\_TIME);*

*JwtBuilder builder = Jwts.builder()*

*.setClaims(claims)*

*.setSubject(subject)*

*.setIssuedAt(issuedAt)*

*.setExpiration(expiration)*

*.signWith(SignatureAlgorithm.HS256, SECRET\_KEY);*

*return builder.compact();*

*}*

*public boolean validateToken(String token, String username) {*

*String extractedUsername = extractUsername(token);*

*return (extractedUsername.equals(username) && !isTokenExpired(token));*

*}*

*private boolean isTokenExpired(String token) {*

*return extractExpiration(token).before(new Date());*

*}*

*private Date extractExpiration(String token) {*

*return Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token).getBody().getExpiration();*

*}*

*public String extractUsername(String token) {*

*return Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token).getBody().getSubject();*

*}*

*}*

1. *secret key*

*Khóa bí mật (secret key) là một thành phần quan trọng trong việc sử dụng JSON Web Tokens (JWT) và các phương thức mã hóa khác. Đây là khóa mà bạn sử dụng để ký và xác thực các token nhằm đảm bảo rằng thông tin trong token là chính xác và chưa bị thay đổi.*

### ***Chức năng của khóa bí mật***

* ***Ký token****: Khi bạn tạo một JWT, khóa bí mật được sử dụng để tạo chữ ký cho token. Điều này đảm bảo rằng token không thể bị giả mạo.*
* ***Xác thực token****: Khi một client gửi token đến server, server sẽ sử dụng khóa bí mật để xác minh rằng token đó được ký bởi chính server. Nếu chữ ký không khớp, token bị coi là không hợp lệ.*

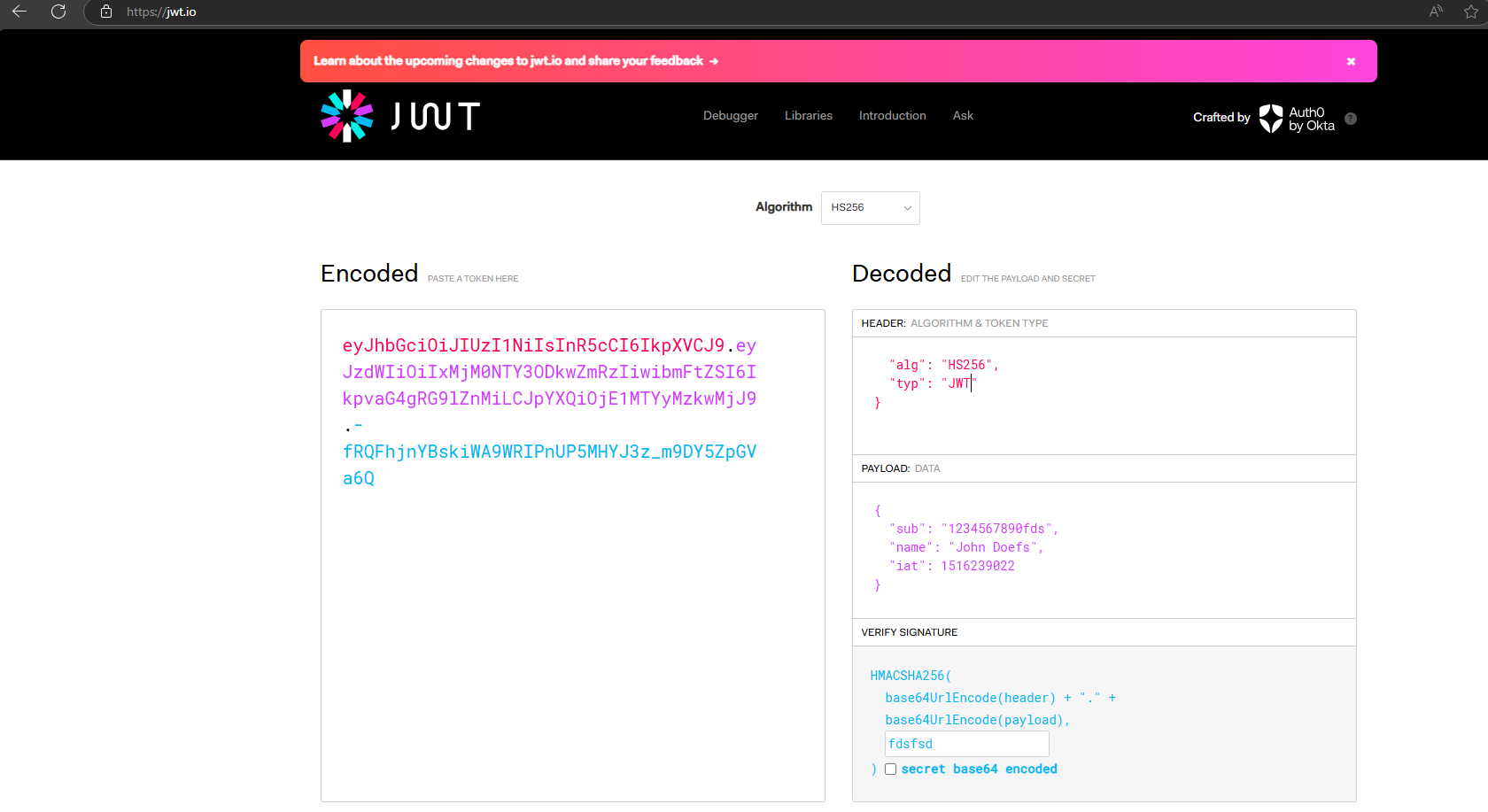
### ***Tính bảo mật***

* ***Bí mật****: Khóa bí mật nên được giữ kín và không được công khai. Nếu kẻ tấn công có được khóa bí mật, họ có thể tạo ra các token giả mạo.*
* ***Độ dài****: Khóa bí mật nên đủ dài và phức tạp để tránh việc bị đoán hoặc bẻ khóa. Thông thường, khóa nên có ít nhất 256 bit.*
* ***Không đưa vào mã nguồn****: Tránh việc để trực tiếp khóa bí mật trong mã nguồn. Hãy sử dụng application.properties hoặc biến môi trường.*
* ***Quản lý bí mật****: Nếu ứng dụng của bạn chạy trên nền tảng đám mây, hãy xem xét sử dụng dịch vụ quản lý bí mật (như AWS Secrets Manager, Azure Key Vault, v.v.) để lưu trữ và truy xuất khóa bí mật một cách an toàn.*

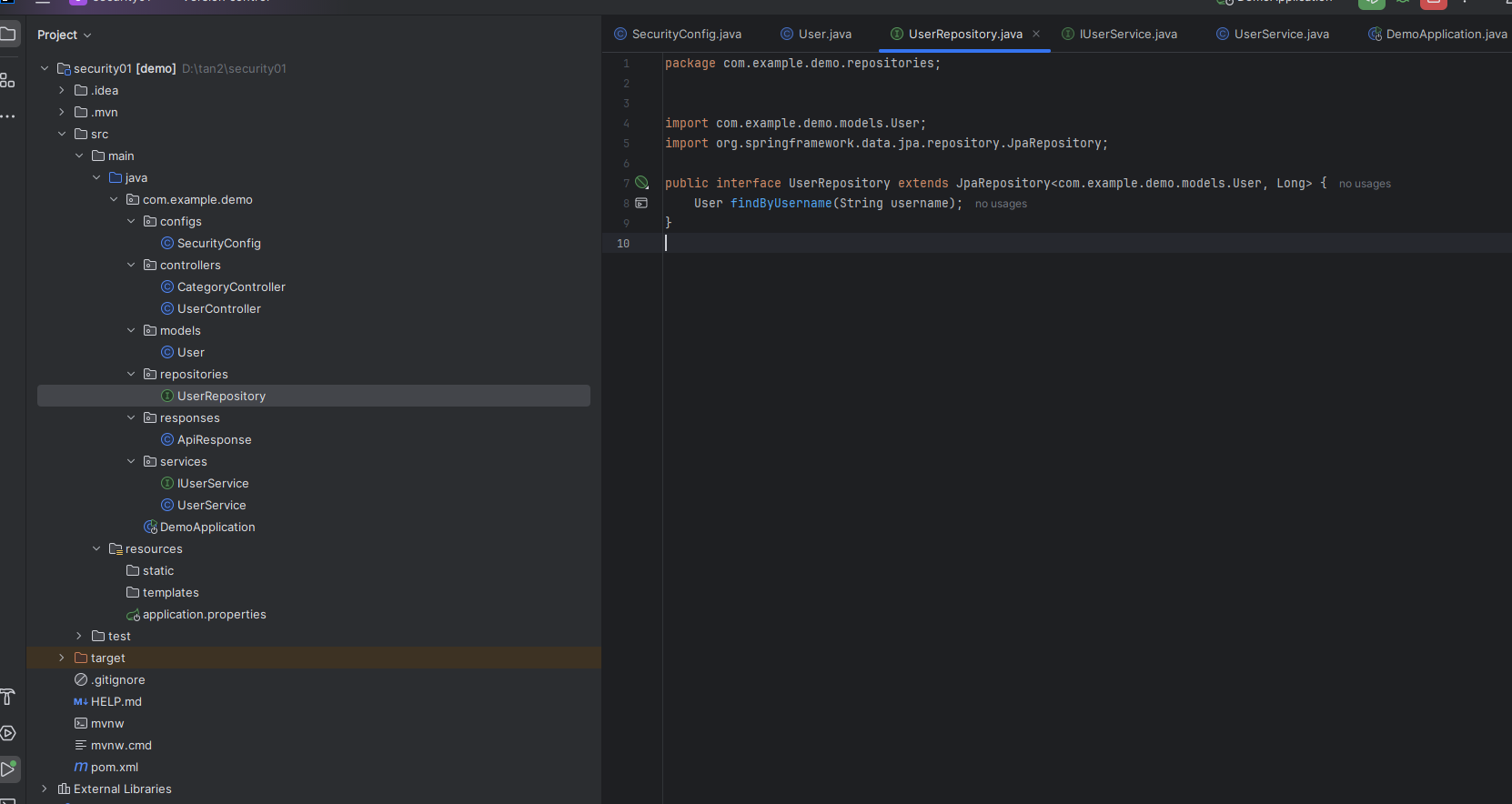
1. *Ví dụ*jwt.secret=my\_super\_secret\_key

jwt.expiration=3600000 # Thời gian hết hạn token (1 giờ)

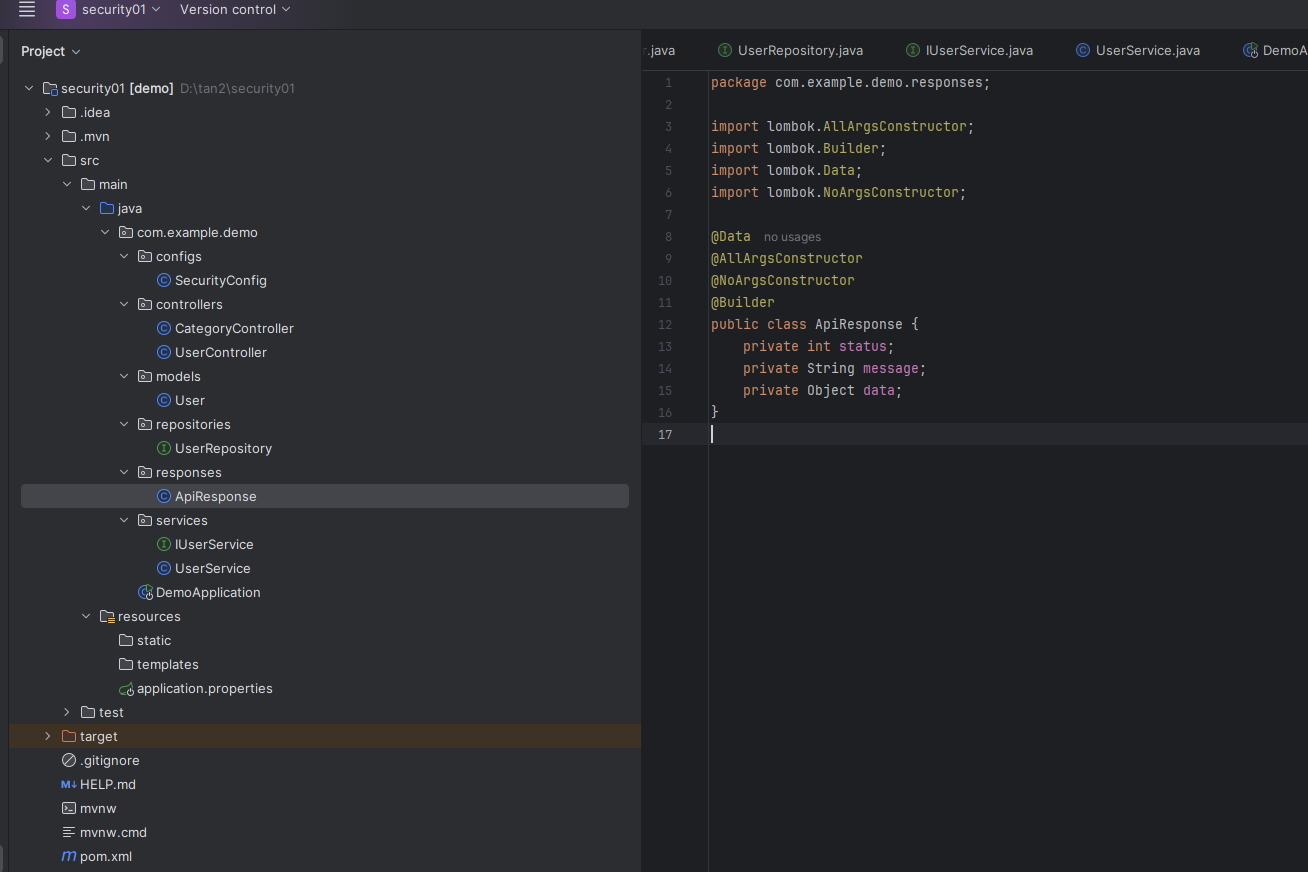
*3. Minh họa JWT*

**

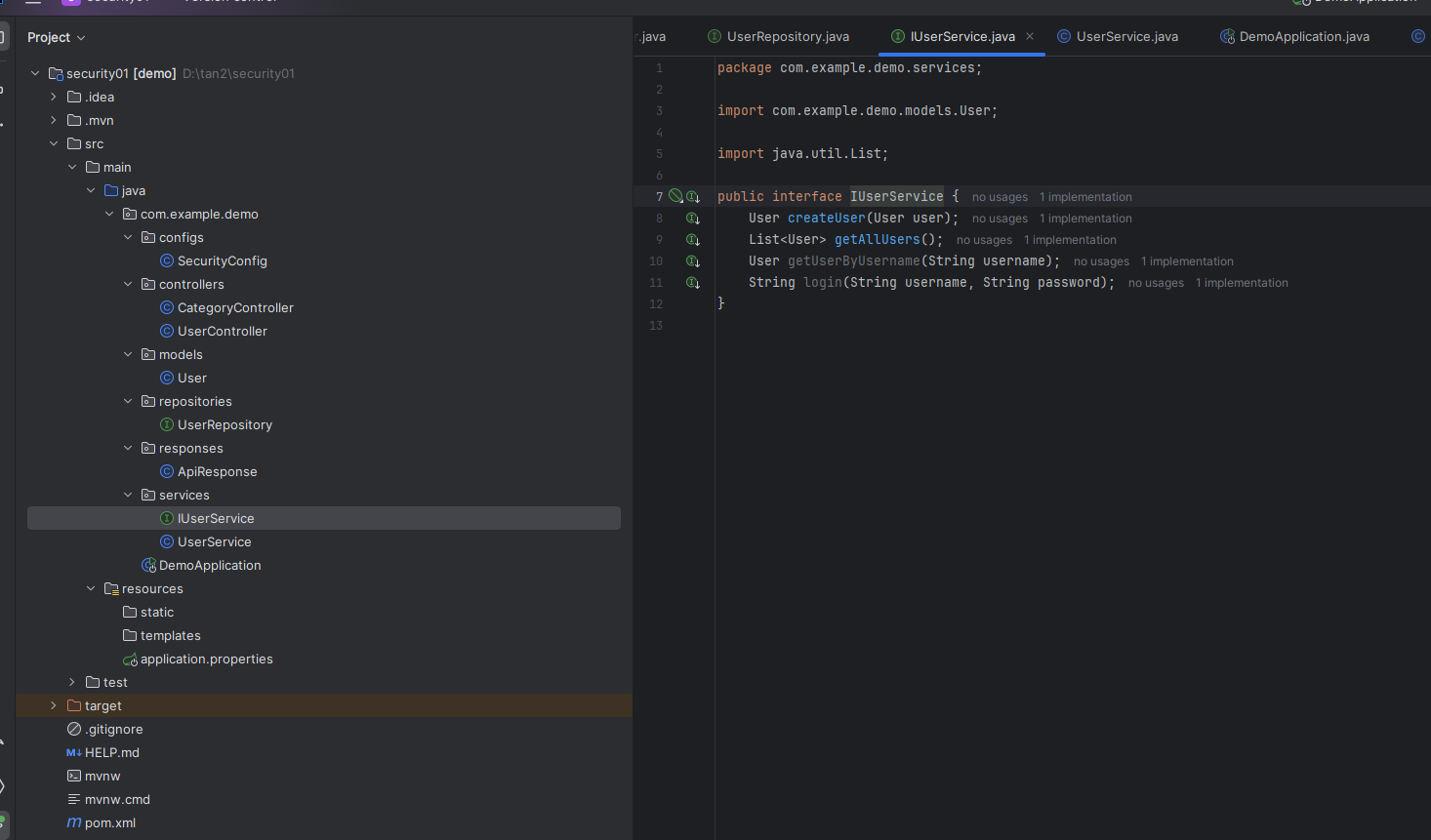
*Code*

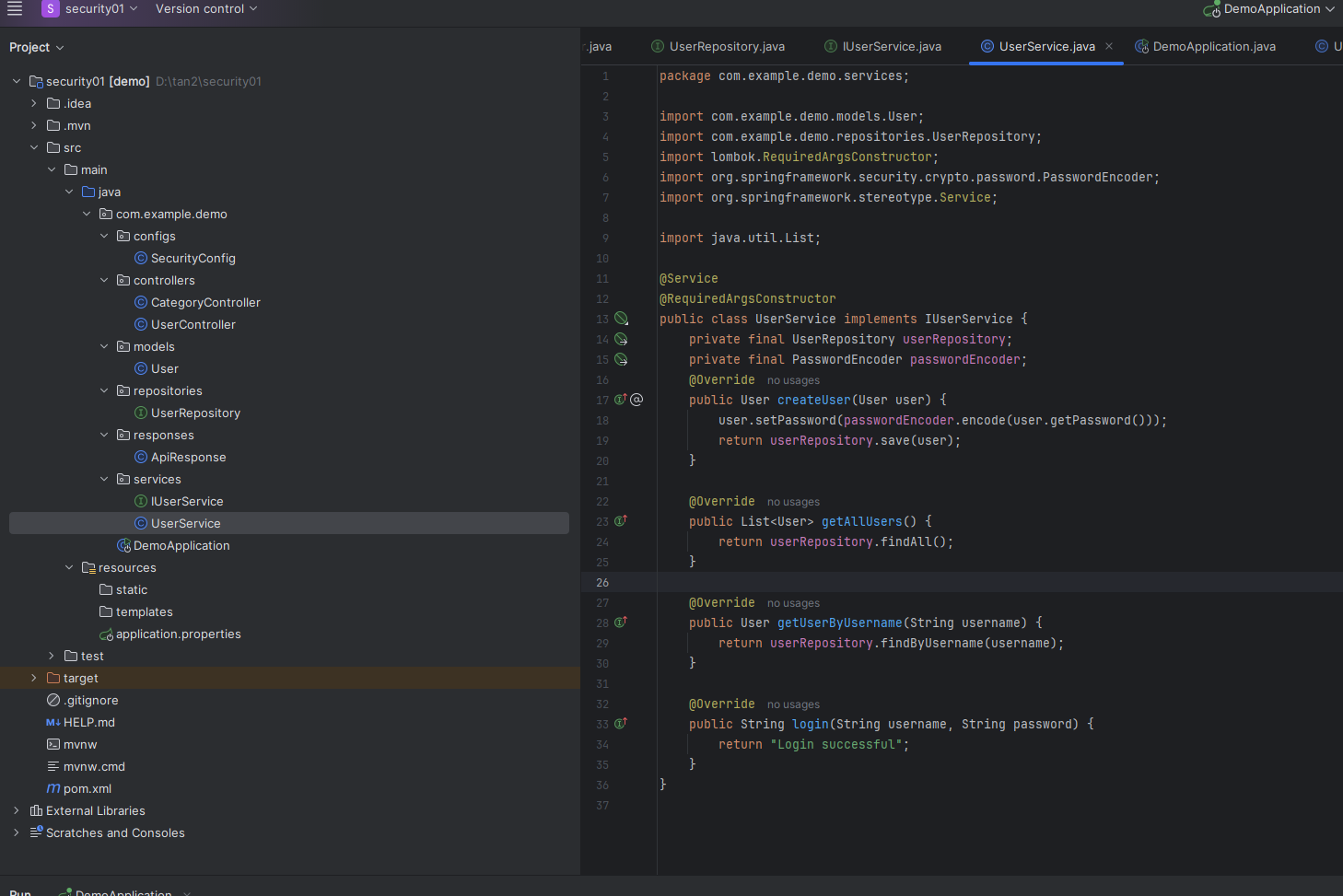
**

*response*

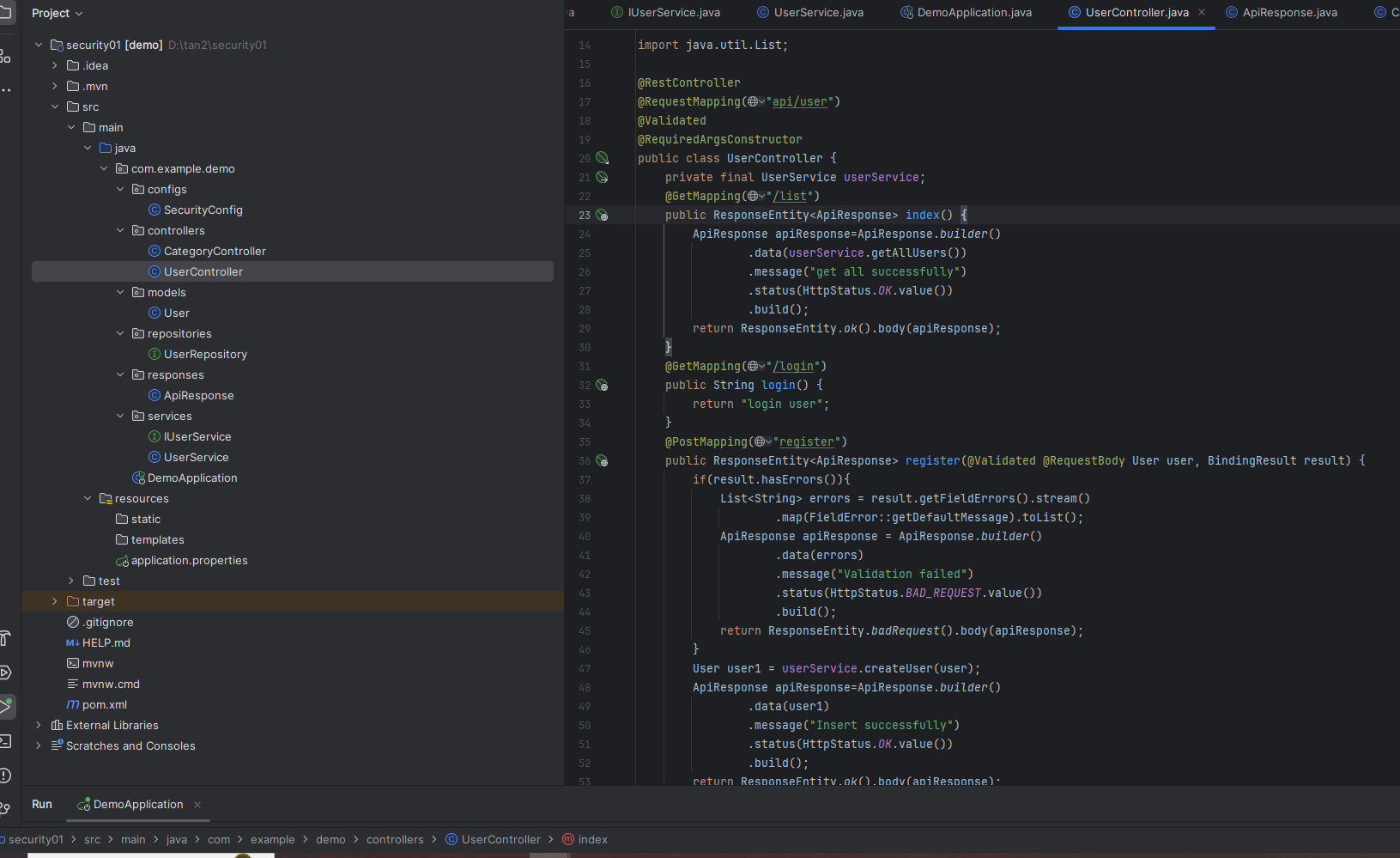
**

*service*

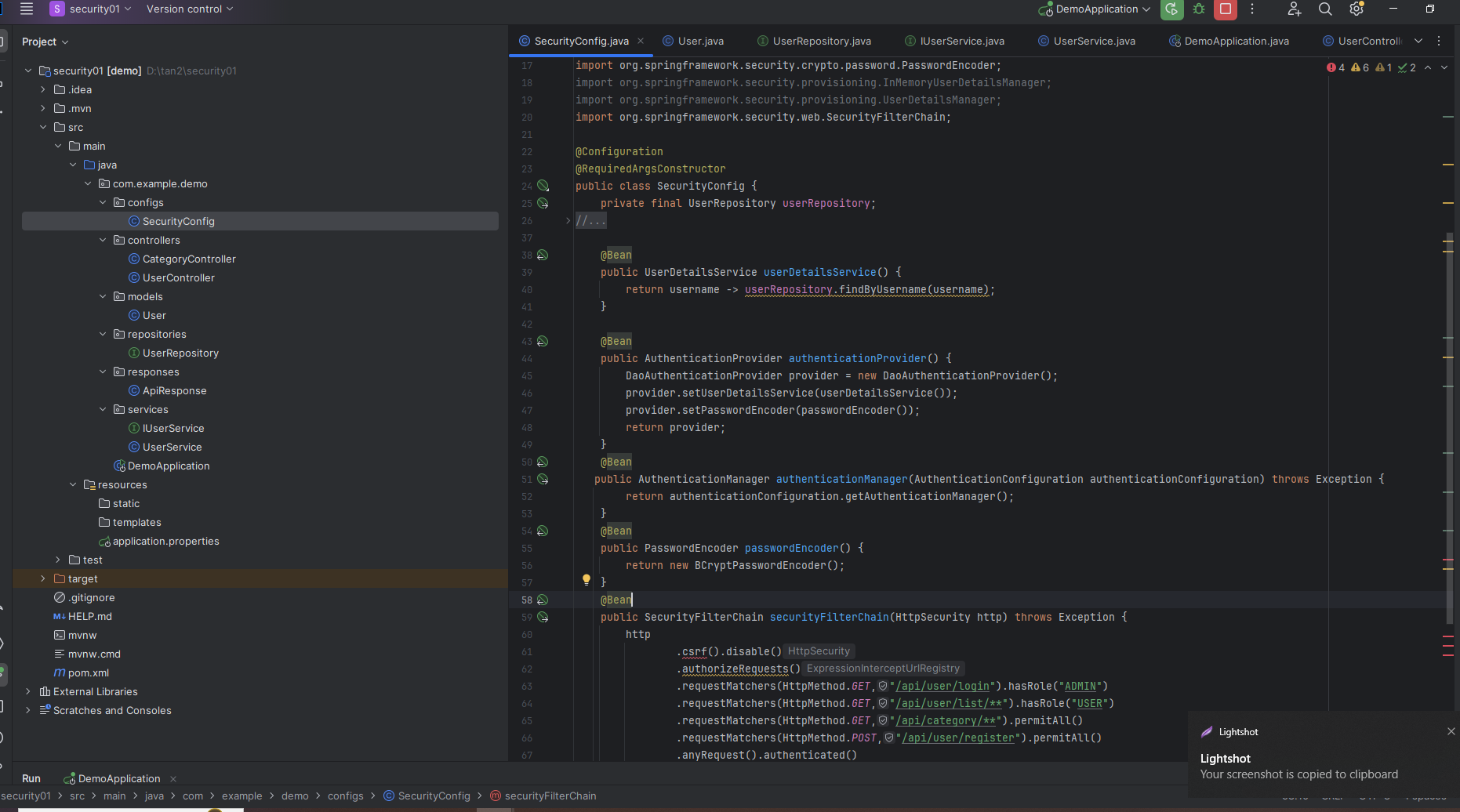
**

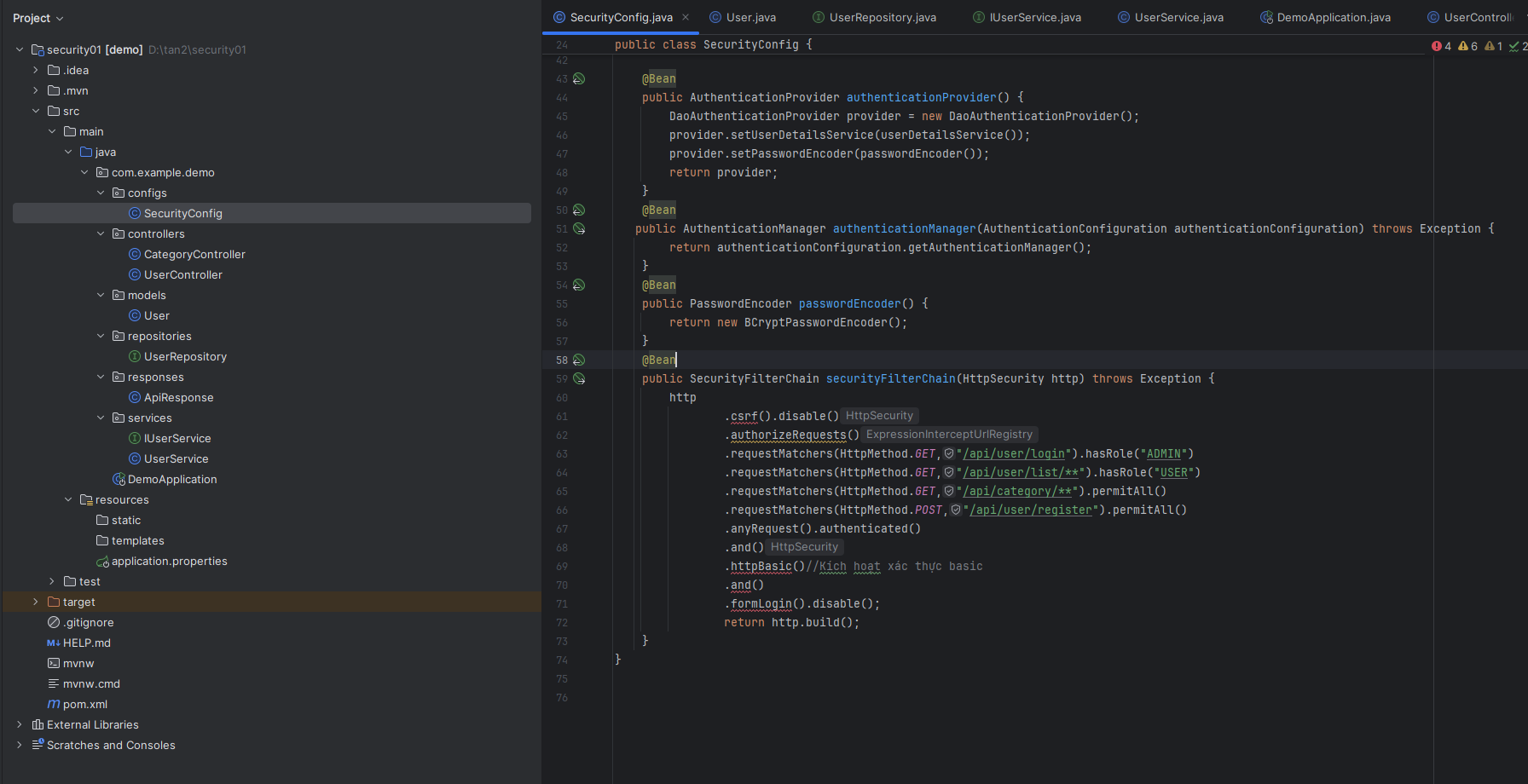
**

*controller*

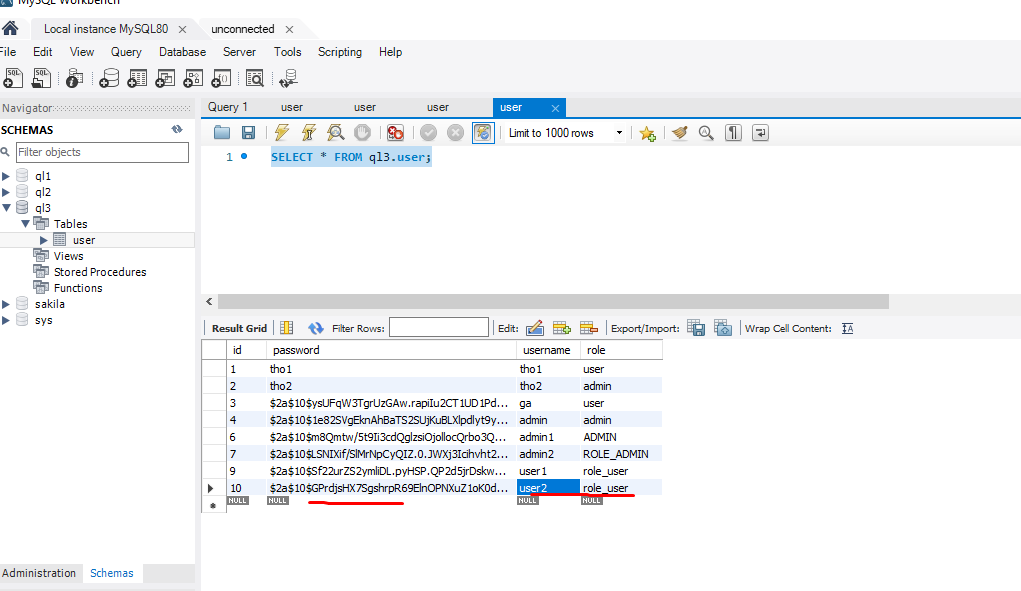
**

*config*

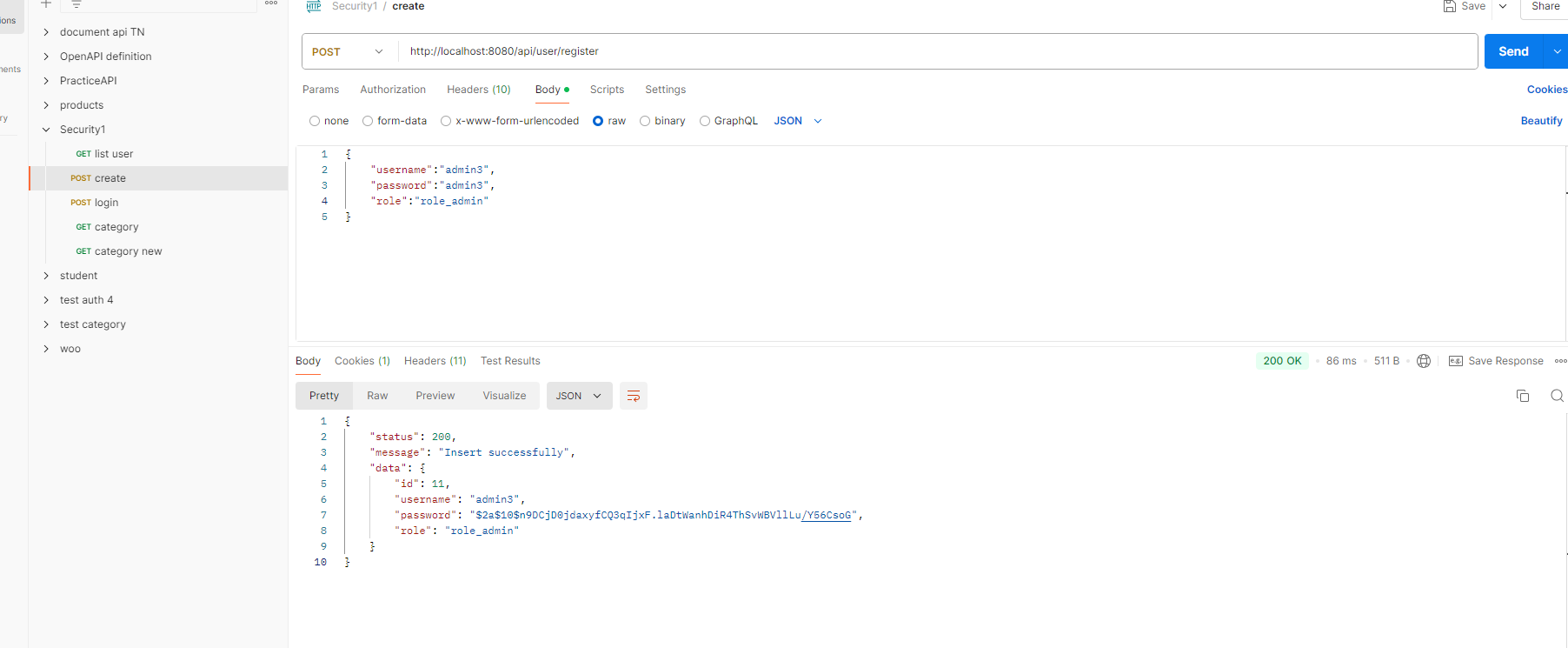
**

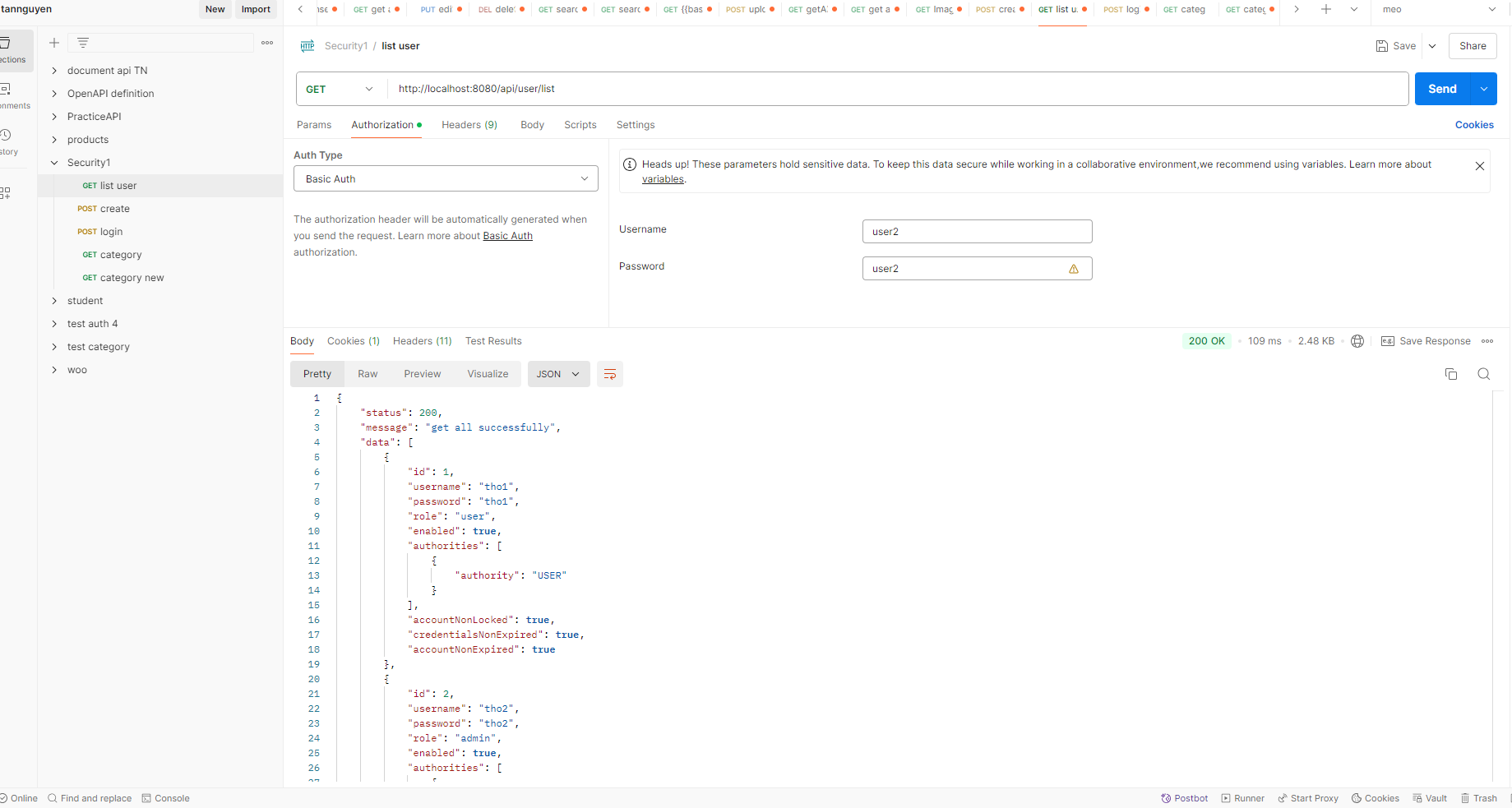
**

*database*

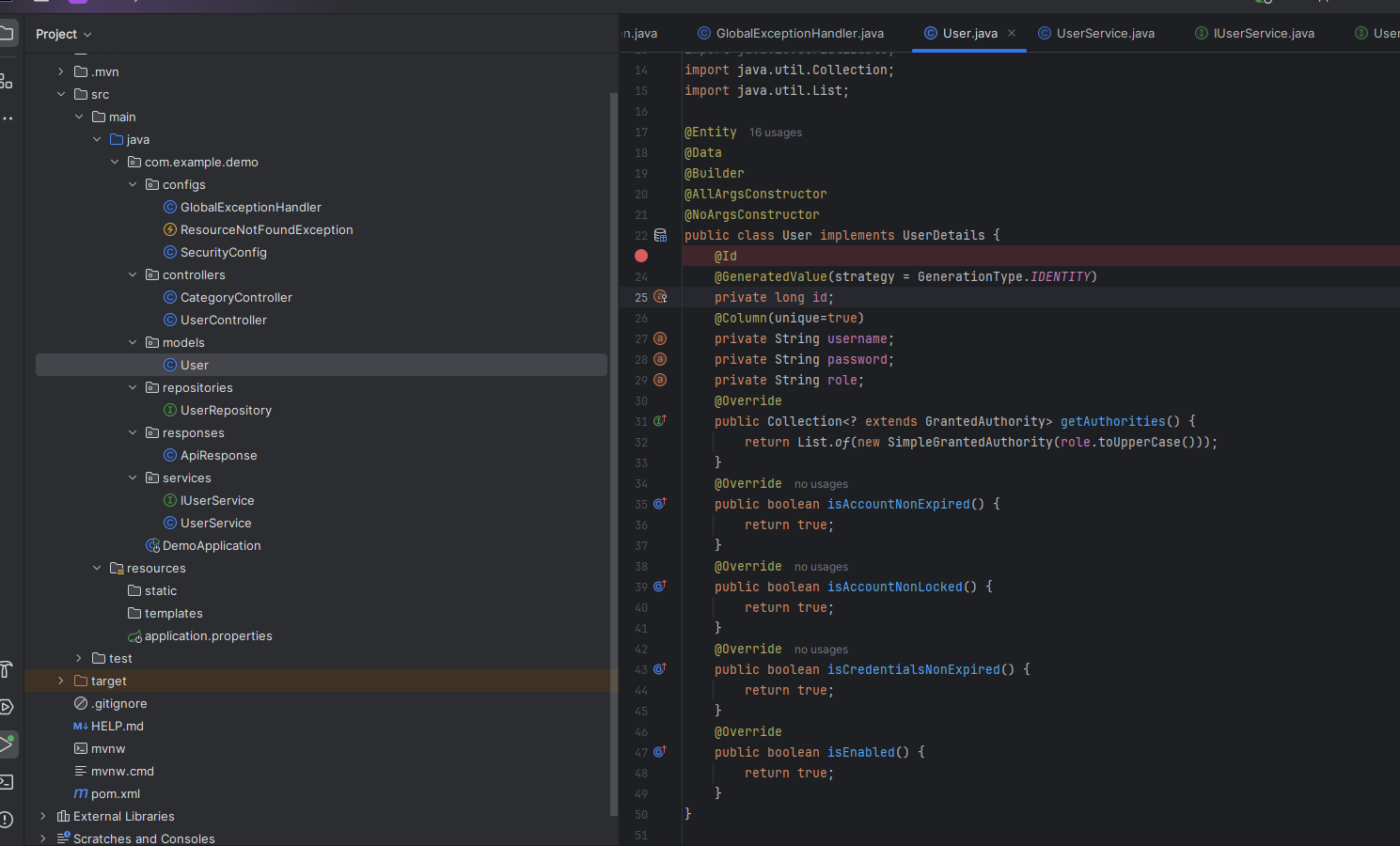
**

*postman*

**

**

*Model*

**