TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM

LẬP TRÌNH WEB (WEBPR330479)

Json Web Token

THS. NGUYỄN HỮU TRUNG

PR33 WEB 0479

Thông tin Giảng viên

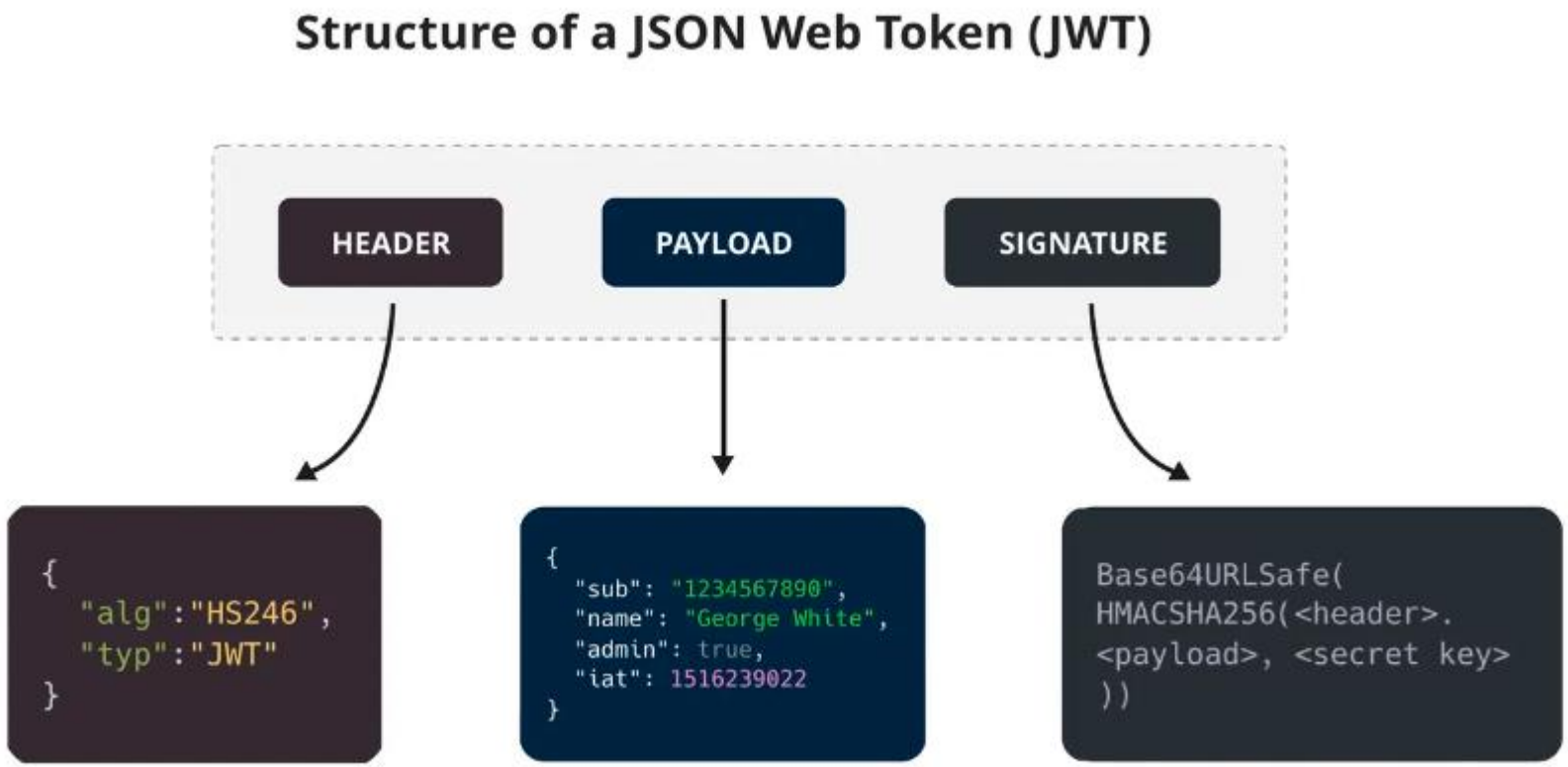
- Ths. Nguyễn Hữu Trung
- Khoa Công Nghệ Thông Tin
- Trường Đại học Sư Phạm Kỹ Thuật TP.HCM
- 090.861.7108
- trungnh@hcmute.edu.vn
- https://www.youtube.com/@baigiai

Nội dung

- Giới thiệu JWT
- Demo JWT trên Spring boot 3 – Security 6

JWT là gì?

- **JWT** là một phương tiện đại diện cho các yêu cầu chuyển giao giữa hai bên Client – Server , các thông tin trong chuỗi **JWT** được định dạng bằng JSON. Trong đó chuỗi Token phải có 3 phần là **header** , phần **payload** và phần **signature** được ngăn bằng dấu “.”
- JWT có thể được ký bằng cách sử dụng một secret (với thuật toán HMAC) hoặc cặp public/private key dùng chuẩn RSA hoặc ECDSA



Cấu trúc của JSON Web Token

- Phần **header** sẽ chứa kiểu dữ liệu và thuật toán sử dụng để mã hóa ra chuỗi JWT

```
{ "typ": "JWT", "alg": "HS256" }
```

- **"typ" (type)** chỉ ra rằng đối tượng là một **JWT**
- **"alg" (algorithm)** xác định thuật toán mã hóa cho chuỗi là **HS256**

- Phần **payload** sẽ chứa các thông tin mình muốn đặt trong chuỗi **Token** như **username** , **userId** , **author** , … ví dụ:

```
{ "user_name": "admin", "user_id": "1", "authorities": "ADMIN", "images": "u1.jpg" }
```

- *Signature* là phần cuối cùng của *JWT*, có chức năng xác thực danh tính người gửi. Phần chữ ký này sẽ được tạo ra bằng cách mã hóa phần **header** , **payload** kèm theo một chuỗi **secret (khóa bí mật)** , ví dụ:

```
data = base64urlEncode( header ) + "." + base64urlEncode( payload ) signature = Hash( data, secret );
```

```
// header.payload.signature
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsidGVzdGp3dHJlc291cmNlaWQiXSwidXNlcl9uYW1lIjoiYWRtaW4iLCJzY29wZSI6WyJyZWFkIiwid3JpdGUiXSwiZXhwIjoxNTEzNzE.
9nRhBWiRoryc8fV5xRpTmw9iyJ6EM7WTGTjvCM1e36Q
```

Cấu trúc của JSON Web Token

- *Payload* trong JWT chứa các *claims*. *Claims* là tập hợp các thông tin đại diện cho một thực thể (*object*) (ví dụ : user_id) và một số thông tin đi kèm. *Claims* sẽ có dạng *Key - Value*. Do đó, chúng ta có thể hiểu rằng, *claims* ám chỉ việc yêu cầu truy xuất tài nguyên cho *object* tương ứng.
- Có 3 kiểu *claims*, bao gồm *registered*, *public*, and *private claims*.
  - *Registered Claims* là các thành phần được xác định trước của *claims*. Thành phần này mặc dù không bắt buộc, nhưng là thành phần nên có để cung cấp một số chức năng và thông tin hữu ích. Một số *registered claims* bao gồm :
    - iss (issuer): Tổ chức, đơn vị cung cấp, phát hành *JWT*.
    - sub (subject): Chủ thể của *JWT*, xác định rằng đây là người sở hữu hoặc có quyền truy cập các *resource* (tài nguyên).
    - aud (audience): Được hiểu là người nhận thông tin, và có thể xác thực tính hợp lệ của *JWT*.
    - exp (expiration time): Thời hạn của *JWT*, vượt quá thời gian này, *JWT* được coi là không hợp lệ.

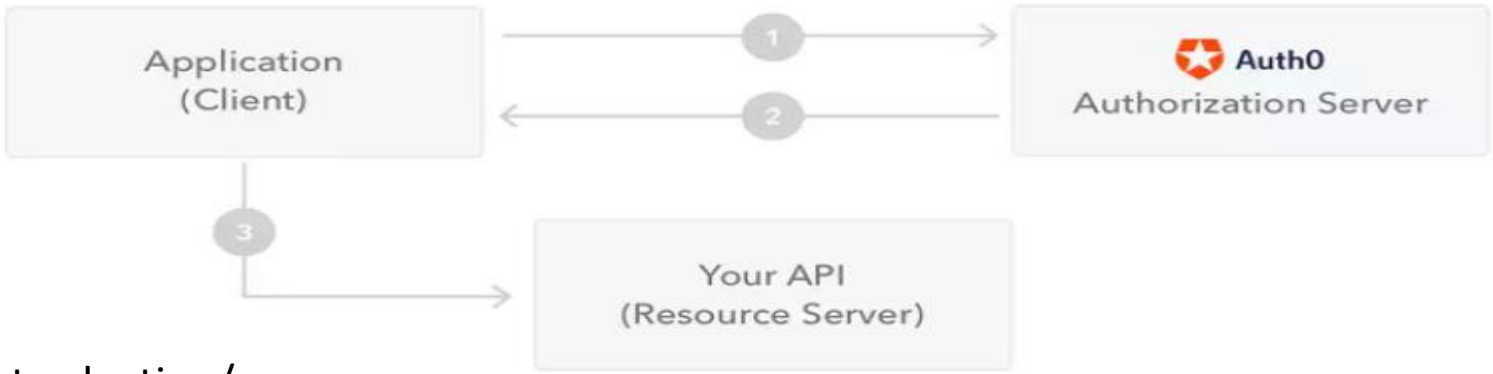
Cấu trúc của JSON Web Token

- Có 3 kiểu *claims*, bao gồm *registered*, *public*, and *private claims*.
  - *Public claims* là các thành phần được xác định bởi người sử dụng *JWT*, được sử dụng rộng rãi trong *JWT*. Mặc dù việc sử dụng *public claims* không phải là bắt buộc, tuy nhiên để tránh xảy ra xung đột. Một số *public claims* điển hình :
    - name, given_name, family_name, middle_name: Thông tin tên nói chung của user
    - email: Thông tin email của user.
    - locale : Địa chỉ của user.
    - profile, picture : Thông tin của trang web gửi đến.
- Các bên sử dụng *JWT* có thể sẽ cần sử dụng đến *claims* không phải là *Registered Claims,* cũng không được định nghĩa trước như *Public Claims.* Đây là phần thông tin mà các bên tự thoả thuận với nhau, không có tài liệu hay tiêu chuẩn nào dành cho *private claims.* **Lưu ý:** Tên *claims* chỉ nên dài ba ký tự vì *JWT* hướng tới sự nhỏ gọn.

JSON Web Tokens hoạt động như thế nào?

- Trong xác thực, khi người dùng đăng nhập thành công bằng thông tin đăng nhập của họ, JSON Web Token sẽ được trả về. Vì token là thông tin xác thực, cần phải hết sức cẩn thận để ngăn chặn các vấn đề bảo mật. Nói chung, bạn không nên giữ token lâu hơn yêu cầu.
- Bạn cũng không nên lưu trữ dữ liệu nhạy cảm trên session vào trong bộ nhớ trình duyệt do thiếu bảo mật.
- Bất cứ khi nào người dùng muốn truy cập URL hoặc resource được bảo vệ, người dùng nên gửi JWT, thêm Authorization trong header với nội dung là Bearer + token. Nội dung của header như sau:

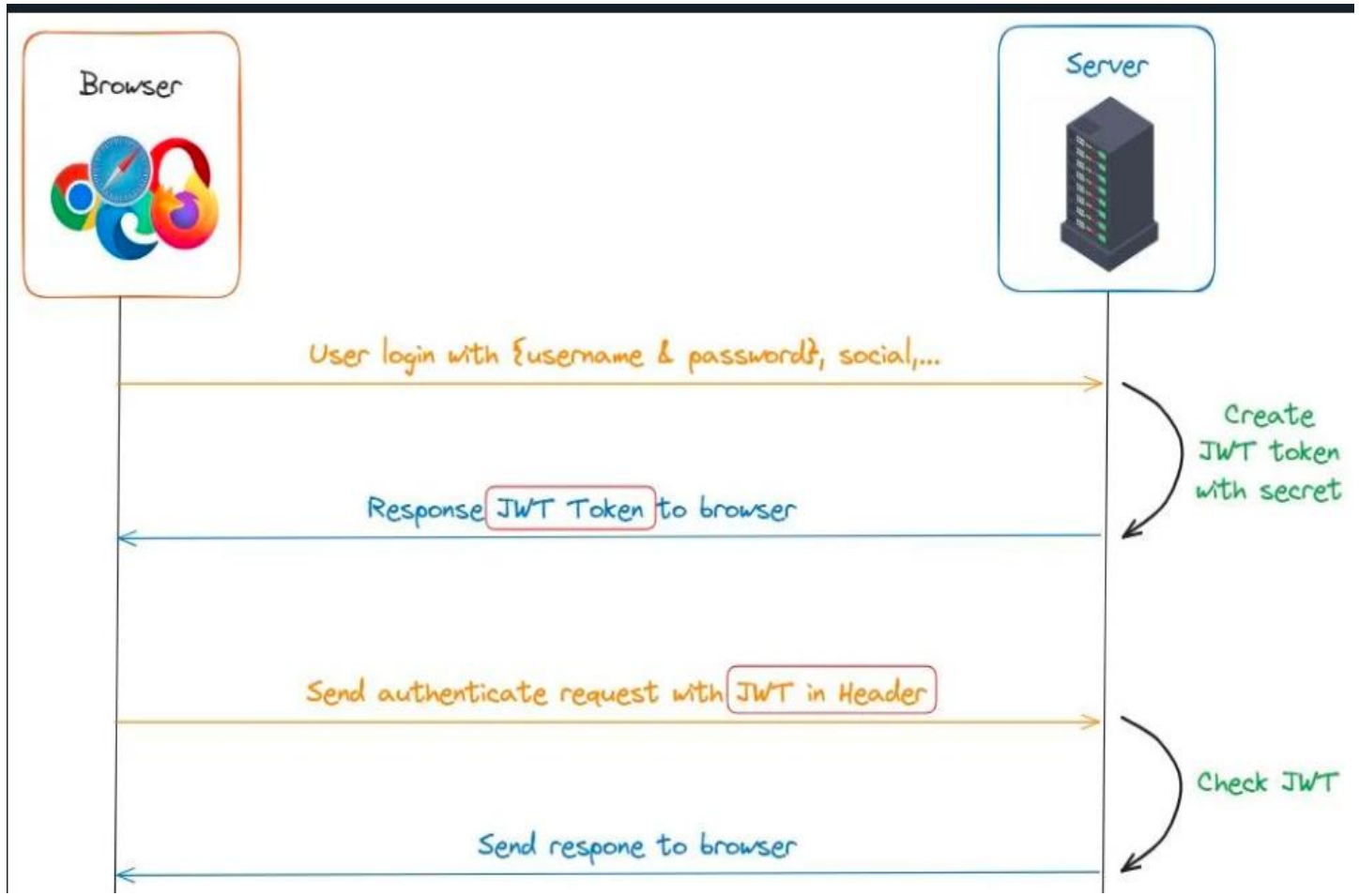
  Authorization: Bearer <token>



https://jwt.io/introduction/

JSON Web Tokens hoạt động như thế nào?

- Client gửi yêu cầu đăng nhập với thông tin xác thực.
- Server xác thực thông tin và tạo JWT token với secret key, sau đó gửi lại cho client.
- Client sẽ lưu trữ JWT token (thường là trong localStorage, sessionStorage, cookies).
- Từ giai đoạn này, tất cả các request client gửi lên server sẽ đính kèm JWT token.
- Server kiểm tra tính hợp lệ của JWT token bằng cách lấy phần Signature (chữ ký) bên trong token, verify xem chữ ký nhận được chính xác là được hash cùng thuật toán và chuỗi secret key hay không.
- Tùy vào từng trường hợp mà server sẽ phản hồi về cho client.



Lý do sử dụng JWT

- **Authentication** (Xác thực): JWT được sử dụng để xác thực người dùng trước khi họ truy cập đến tài nguyên trên server.
- **Authorization** (Uỷ quyền): Khi người dùng đăng nhập thành công, application có thể truy cập vào các tài nguyên thay mặt người dùng đó. Các ứng dụng đăng nhập một lần (*Single Sign-On SSO*) sử dụng *JWT* thường xuyên vì tính nhỏ gọn và dễ dàng triển khai trên nhiều domain.
- **Trao đổi thông tin an toàn**: *JWT* được coi là một cách trao đổi thông tin an toàn vì thông tin đã được *signed* trước khi gửi đi.

Ưu điểm của JWT

- **Gọn nhẹ**: JWT nhỏ gọn, chi phí truyền tải thấp giúp tăng hiệu suốt của các ứng dụng.
- **Bảo mật**: JWT sử dụng các mật mã khoá để tiến hành xác thực người danh tính người dùng. Ngoài ra, cấu trúc của JWT cho phép chống giả mạo nên thông tin được đảm bảo an toan trong quá trình trao đổi.
- **Phổ thông**: JWT được sử dụng dựa trên JSON, là một dạng dữ liệu phổ biến, có thể sử dụng ở hầu hết các ngôn ngữ lập trình. Ngoài ra, triển khai JWT tương đối dễ dàng và tích hợp được với nhiều thiết bị, vì JWT đã tương đối phổ biến.

Khuyết điểm của JWT

- **Kích thước**: Mặc dù trong tài liệu không ghi cụ thể giới hạn, nhưng do được truyền trên *HTTP Header*, vì thế, *JWT* có giới hạn tương đương với *HTTP Header* (khoảng 8KB).
- **Rủi ro bảo mật**: Khi sử dụng *JWT* không đúng cách, ví dụ như không kiểm tra tính hợp lệ của *signature*, không kiểm tra *expire time*, kẻ tấn công có thể lợi dụng sơ hở để truy cập vào các thông tin trái phép. Ngoài ra, việc để thời gian hết hạn của *JWT* quá dài cũng có thể tạo ra kẽ hở tương tự.

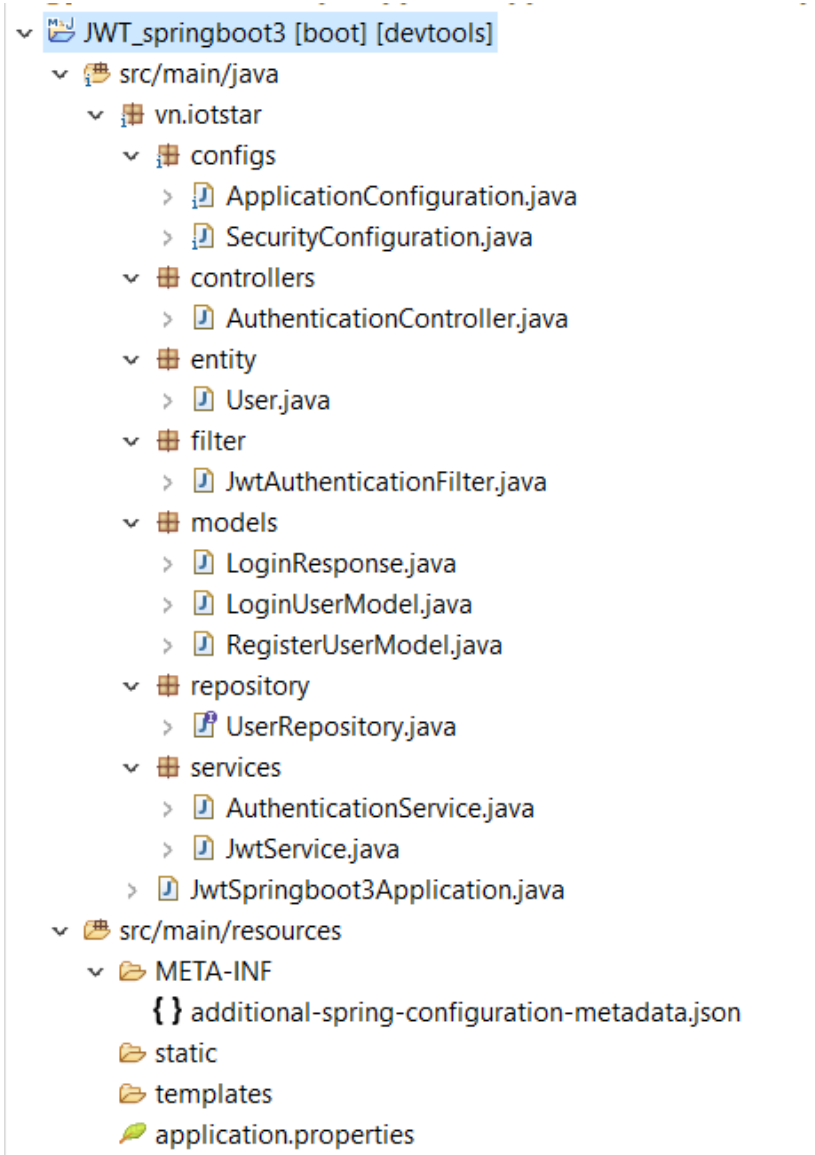
Một số ứng dụng JWT

- **Single Sign-On (SSO)**: *JWT* có thể được sử dụng để cung cấp *single sign-on* cho người dùng. Điều này cho phép họ đăng nhập vào nhiều ứng dụng chỉ với một tài khoản duy nhất.
- **API Authorization**: *JWT* thường được sử dụng để phân quyền cho người dùng đến những tài nguyên cụ thể, từ những *claims* chứa trong JWT đó.
- **User Authentication:** *JWT* cung cấp khả năng xác thực người dùng và cấp quyền cho họ truy cập vào các tài nguyên mong muốn trong hệ thống.
- **Microservices Communication:** *JWT* còn có thể sử dụng cho việc giao tiếp giữa các *service* nhỏ trong hệ thống *microservices*.

- **Sử dụng secret key yếu, dễ dự đoán**: dễ dàng bị tấn công và giải mã dễ dàng.
- **Lưu trữ token không an toàn**: Lưu trữ trong localStorage hay sessionStorage có thể dễ bị tấn công XSS (Cross-Site Scripting).
- **Không thiết lập thời gian hết hạn**: Token sẽ luôn tồn tại vĩnh viễn nếu bạn không thiết lập thời gian hết hạn cho token, nguy cơ về bảo mật nếu như token bị lộ.
- **Truyền qua kết nối không an toàn**: Token dễ bị đánh cắp, luôn sử dụng HTTPs để truyền tải các dữ liệu nhạy cảm.

Demo JWT với Spring Boot 3 – Security 6

- Bước 1: Thêm depandency




```
<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>
</dependency>
<dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-jackson</artifactId>
        <version>0.12.6</version>
</dependency>
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 2: Tạo Entity

```java
@AllArgsConstructor
@NoArgsConstructor
@Data
@Table(name = "users")
@Entity
public class User implements UserDetails {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(nullable = false)
    private Integer id;

    @Column(nullable = false, columnDefinition = "nvarchar(50)")
    private String fullName;

    @Column(unique = true, length = 100, nullable = false)
    private String email;

    @Column(columnDefinition = "nvarchar(500)", nullable = false)
    private String images;

    @Column(nullable = false)
    private String password;

    @CreationTimestamp
    @Column(updatable = false, name = "created_at")
    private Date createdAt;

    @UpdateTimestamp
    @Column(name = "updated_at")
    private Date updatedAt;
```

```java
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return List.of();
    }

    public String getPassword() {
        return password;
    }

    @Override
    public String getUsername() {
        return email;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 3: Tạo Models

```java
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@AllArgsConstructor
@NoArgsConstructor
@Data
public class LoginResponse {
    private String token;

    private long expiresIn;

    public String getToken() {
        return token;
    }
}

```

```java
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@AllArgsConstructor
@NoArgsConstructor
@Data
public class LoginUserModel {
    private String email;
    private String password;
}
```

```java
@AllArgsConstructor
@NoArgsConstructor
@Data
public class RegisterUserModel {
    private String email;

    private String password;

    private String fullName;
}

```

Demo JWT với Spring Boot 3 – Security 6

- Bước 4: Khởi tạo một Interface Repository và services

```java
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import vn.iotstar.entity.User;

import java.util.Optional;

@Repository
public interface UserRepository extends JpaRepository<User, Integer> {
    Optional<User> findByEmail(String email);
}
```

```java
import vn.iotstar.entity.User;
import vn.iotstar.repository.UserRepository;

import java.util.ArrayList;
import java.util.List;

@Service
public class UserService {
    private final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public List<User> allUsers() {
        List<User> users = new ArrayList<>();

        userRepository.findAll().forEach(users::add);

        return users;
    }
}
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 4: Khởi tạo một Interface Repository và services

```java
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import vn.iotstar.entity.User;
import vn.iotstar.models.LoginUserModel;
import vn.iotstar.models.RegisterUserModel;
import vn.iotstar.repository.UserRepository;

@Service
public class AuthenticationService {
    private final UserRepository userRepository;

    private final PasswordEncoder passwordEncoder;

    private final AuthenticationManager authenticationManager;

    public AuthenticationService(
        UserRepository userRepository,
        AuthenticationManager authenticationManager,
        PasswordEncoder passwordEncoder
    ) {
        this.authenticationManager = authenticationManager;
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
    }

    public User signup(RegisterUserModel input) {
            User user = new User();
                user.setFullName(input.getFullName());
                user.setEmail(input.getEmail());
                user.setPassword(passwordEncoder.encode(input.getPassword()));

        return userRepository.save(user);
    }
```

```java
    public User authenticate(LoginUserModel input) {
        authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(
                        input.getEmail(),
                        input.getPassword()
                )
        );

        return userRepository.findByEmail(input.getEmail())
                .orElseThrow();
    }
}
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 4: Khởi tạo một Interface Repository và services

```java
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.io.Decoders;
import io.jsonwebtoken.security.Keys;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

import javax.crypto.SecretKey;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;

@Service
public class JwtService {
    @Value("${security.jwt.secret-key}")
    private String secretKey;

    @Value("${security.jwt.expiration-time}")
    private long jwtExpiration;

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

    public String generateToken(UserDetails userDetails) {
        return generateToken(new HashMap<>(), userDetails);
    }

    public String generateToken(Map<String, Object> extraClaims, UserDetails userDetails) {
        return buildToken(extraClaims, userDetails, jwtExpiration);
    }

    public long getExpirationTime() {
        return jwtExpiration;
    }

    private String buildToken(
            Map<String, Object> extraClaims,
            UserDetails userDetails,
            long expiration
    ) {
        return Jwts.builder()
                .claims(extraClaims)
                .subject(userDetails.getUsername())
                .issuedAt(new Date(System.currentTimeMillis()))
                // the token will be expired in 30 hours
                .expiration(new Date(System.currentTimeMillis() + 1000* 60 * 60 *30))
                .signWith(getSignInKey(), Jwts.SIG.HS256)
                .compact();

    }

    public boolean isTokenValid(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername())) && !isTokenExpired(token);
    }
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 4: Khởi tạo một Interface Repository và services

```
68      private boolean isTokenExpired(String token) {
69          return extractExpiration(token).before(new Date());
70      }
71
72      private Date extractExpiration(String token) {
73          return extractClaim(token, Claims::getExpiration);
74      }
75
76      private Claims extractAllClaims(String token) {
77          return Jwts
78                  .parser()
79                     .verifyWith(getSignInKey())
80                     .build()
81                     .parseSignedClaims(token)
82                     .getPayload();
83
84      }
85
86      private SecretKey getSignInKey() {
87          byte[] keyBytes = Decoders.BASE64.decode(secretKey);
88          return Keys.hmacShaKeyFor(keyBytes);
89      }
90  }
```

File application.properties

```
spring.application.name=JWT_springboot3
server.port=8005
spring.datasource.url=jdbc:mysql://localhost:3306/jwt_springboot3?serverTimezone=UTC&allowPublicKeyRetrieval=true&useSSL=false
spring.datasource.username=root
spring.datasource.password=1234567@a$
## Hibernate properties
spring.jpa.hibernate.ddl-auto=update
spring.jpa.open-in-view=false


security.jwt.secret-key=3cfa76ef14937c1c0ea519f8fc057a80fcd04a7420f8e8bcd0a7567c272e007b
# 1h in millisecond
security.jwt.expiration-time=3600000
```

Tạo mã online: https://www.devglan.com/online-tools/hmac-sha256-online

Demo JWT với Spring Boot 3 – Security 6

- Bước 5: Khởi tạo ApplicationConfiguration

```
13 import vn.iotstar.repository.UserRepository;
14
15 @Configuration
16 public class ApplicationConfiguration {
17     private final UserRepository userRepository;
18
19     public ApplicationConfiguration(UserRepository userRepository) {
20         this.userRepository = userRepository;
21     }
22
23     @Bean
24     UserDetailsService userDetailsService() {
25         return username -> userRepository.findByEmail(username)
26                 .orElseThrow(() -> new UsernameNotFoundException("User not found"));
27     }
28
29     @Bean
30     BCryptPasswordEncoder passwordEncoder() {
31         return new BCryptPasswordEncoder();
32     }
33
34     @Bean
35     public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {
36         return config.getAuthenticationManager();
37     }
38
39     @Bean
40     AuthenticationProvider authenticationProvider() {
41         DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
42
43         authProvider.setUserDetailsService(userDetailsService());
44         authProvider.setPasswordEncoder(passwordEncoder());
45
46         return authProvider;
47     }
48 }
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 6: Khởi tạo một class Filter

```java
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import vn.iotstar.services.JwtService;

import org.springframework.lang.NonNull;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;
import org.springframework.web.servlet.HandlerExceptionResolver;

import java.io.IOException;

@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter {
    private final HandlerExceptionResolver handlerExceptionResolver;

    private final JwtService jwtService;
    private final UserDetailsService userDetailsService;

    public JwtAuthenticationFilter(
        JwtService jwtService,
        UserDetailsService userDetailsService,
        HandlerExceptionResolver handlerExceptionResolver
    ) {
        this.jwtService = jwtService;
        this.userDetailsService = userDetailsService;
        this.handlerExceptionResolver = handlerExceptionResolver;
    }

    @Override
    protected void doFilterInternal(
        @NonNull HttpServletRequest request,
        @NonNull HttpServletResponse response,
        @NonNull FilterChain filterChain
    ) throws ServletException, IOException {
        final String authHeader = request.getHeader("Authorization");

        if (authHeader == null || !authHeader.startsWith("Bearer ")) {
            filterChain.doFilter(request, response);
            return;
        }
        try {
            final String jwt = authHeader.substring(7);
            final String userEmail = jwtService.extractUsername(jwt);

            Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

            if (userEmail != null && authentication == null) {
                UserDetails userDetails = this.userDetailsService.loadUserByUsername(userEmail);

                if (jwtService.isTokenValid(jwt, userDetails)) {
                    UsernamePasswordAuthenticationToken authToken = new UsernamePasswordAuthenticationToken(
                            userDetails,
                            null,
                            userDetails.getAuthorities()
                    );
                    authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
                    SecurityContextHolder.getContext().setAuthentication(authToken);
                }
            }
            filterChain.doFilter(request, response);
        } catch (Exception exception) {
            handlerExceptionResolver.resolveException(request, response, null, exception);
        }
    }
}
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 7: Khởi tạo một class SecurityConfig

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import vn.iotstar.filter.JwtAuthenticationFilter;

import java.util.List;

@Configuration
@EnableWebSecurity
public class SecurityConfiguration {
    private final AuthenticationProvider authenticationProvider;
    private final JwtAuthenticationFilter jwtAuthenticationFilter;

    public SecurityConfiguration(
        JwtAuthenticationFilter jwtAuthenticationFilter,
        AuthenticationProvider authenticationProvider
    ) {
        this.authenticationProvider = authenticationProvider;
        this.jwtAuthenticationFilter = jwtAuthenticationFilter;
    }

    // Configuring HttpSecurity
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity httpSecurity) throws Exception {
        return httpSecurity
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                    //.requestMatchers("/**").permitAll()
                    .requestMatchers("/auth/**").permitAll()
                    .requestMatchers("/login**").permitAll()
                    .requestMatchers("/user/**").permitAll()
                    .requestMatchers(new AntPathRequestMatcher("/images/**")).permitAll()
                    .requestMatchers(new AntPathRequestMatcher("/js/**")).permitAll()
                    .anyRequest()
                    .authenticated()
            )
            .sessionManagement(management -> management
                    .sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .authenticationProvider(authenticationProvider)
            .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class)
            .build();
    }
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 7: Khởi tạo một class SecurityConfig

```
59      @Bean
60      CorsConfigurationSource corsConfigurationSource() {
61          CorsConfiguration configuration = new CorsConfiguration();
62
63          configuration.setAllowedOrigins(List.of("http://localhost:8005"));
64          configuration.setAllowedMethods(List.of("HEAD","GET", "POST", "PUT", "DELETE", "PATCH"));
65          configuration.setAllowCredentials(true);
66          configuration.setAllowedHeaders(List.of("Authorization","Content-Type", "Cache-Control"));
67
68          UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
69
70          source.registerCorsConfiguration("/**",configuration);
71
72          return source;
73      }
74  }
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 8: Tạo class @RestController

```
18 @RequestMapping("/auth")
19 @RestController
20 public class AuthenticationController {
21     private final JwtService jwtService;
22
23     private final AuthenticationService authenticationService;
24
25     public AuthenticationController(JwtService jwtService, AuthenticationService authenticationService) {
26         this.jwtService = jwtService;
27         this.authenticationService = authenticationService;
28     }
29
30     @PostMapping("/signup")
31     @Transactional
32     public ResponseEntity<User> register(@RequestBody RegisterUserModel registerUser) {
33         User registeredUser = authenticationService.signup(registerUser);
34
35         return ResponseEntity.ok(registeredUser);
36     }
37
38     @PostMapping(path="/login")
39     @Transactional
40     public ResponseEntity<LoginResponse> authenticate(@RequestBody LoginUserModel loginUser) {
41         User authenticatedUser = authenticationService.authenticate(loginUser);
42
43         String jwtToken = jwtService.generateToken(authenticatedUser);
44
45         LoginResponse loginResponse = new LoginResponse();
46                 loginResponse.setToken(jwtToken);
47                 loginResponse.setExpiresIn(jwtService.getExpirationTime());
48
49         return ResponseEntity.ok(loginResponse);
50     }
```

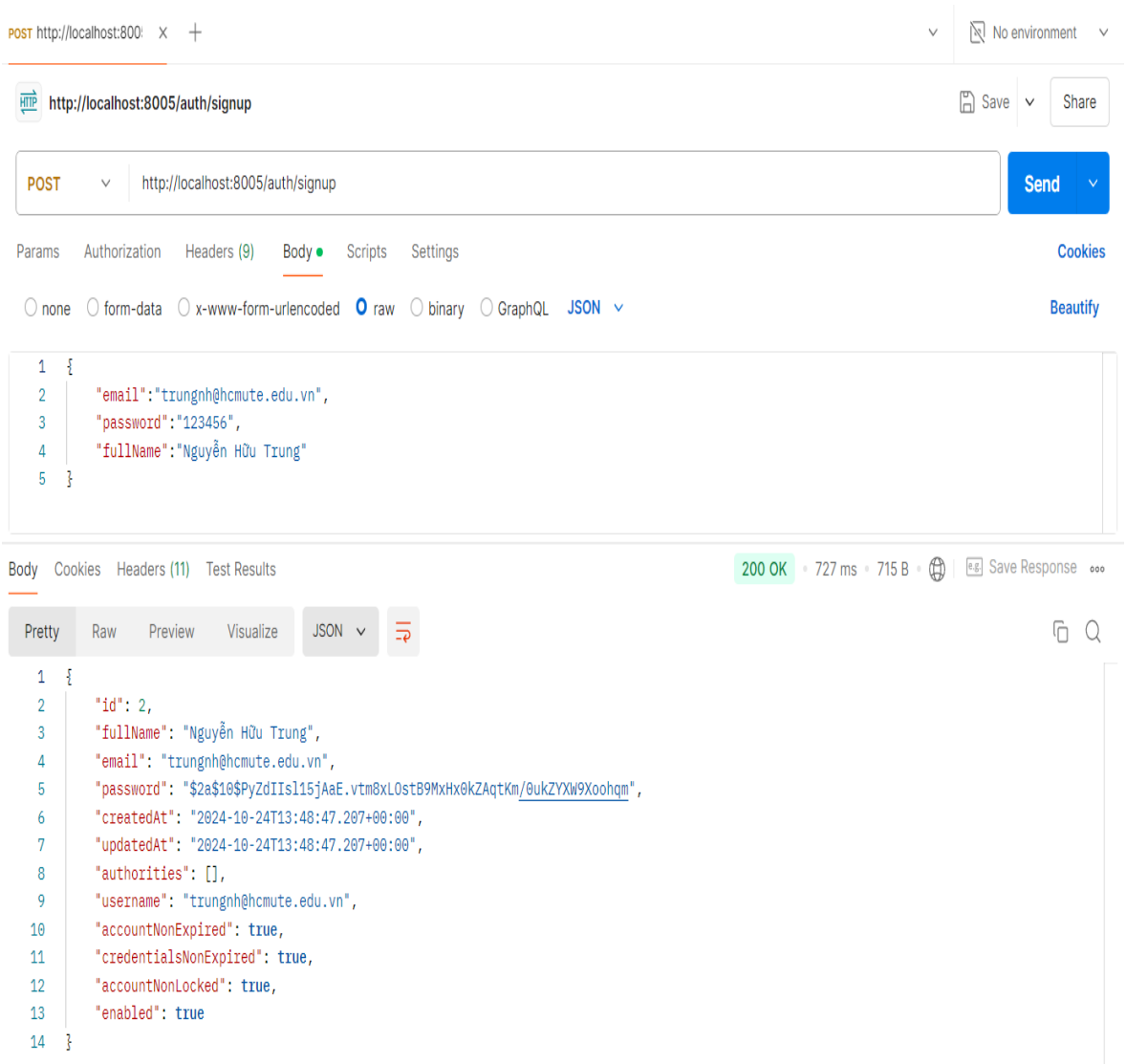
Demo JWT với Spring Boot 3 – Security 6

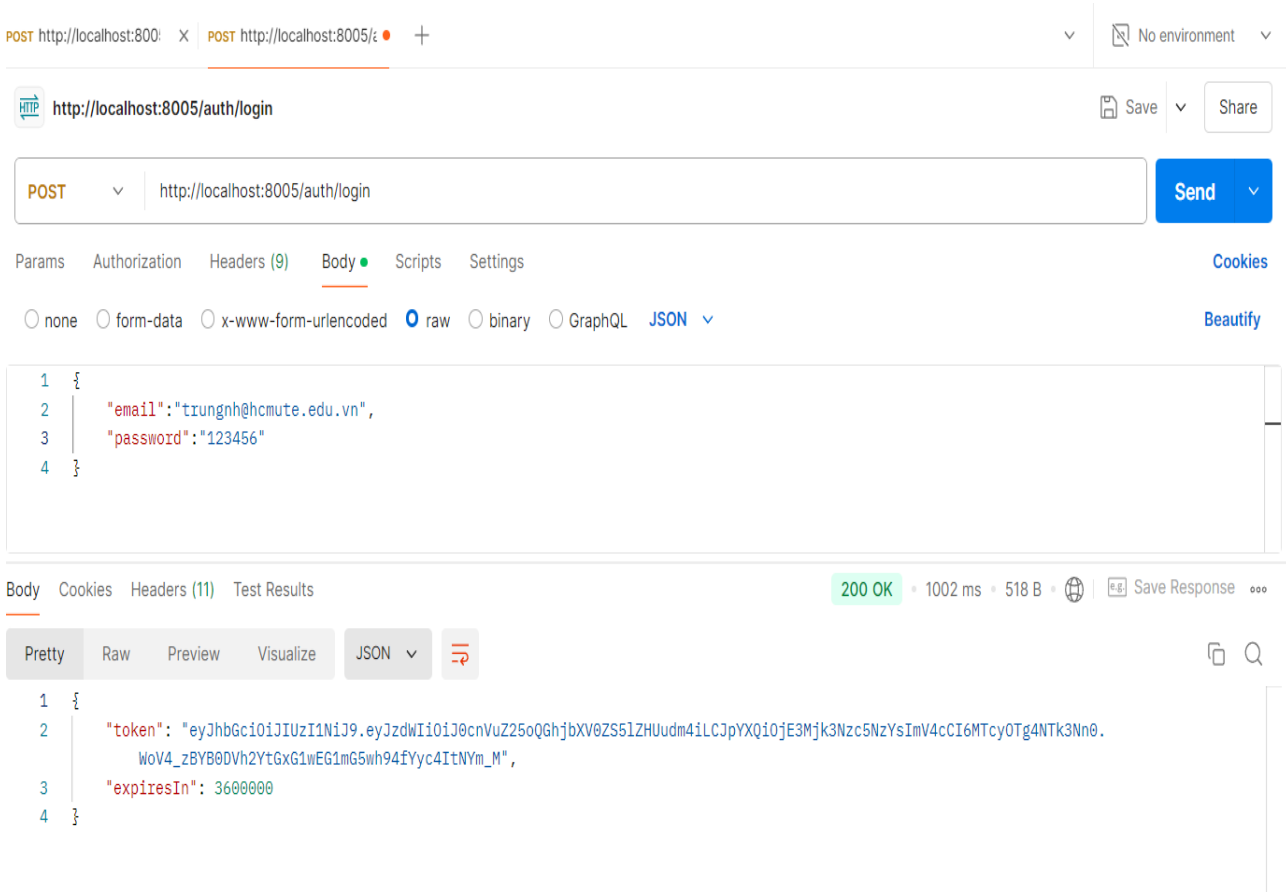
- Bước 8: Tạo class @RestController

```
15 @RequestMapping("/users")
16 @RestController
17 public class UserController {
18     private final UserService userService;
19
20     public UserController(UserService userService) {
21         this.userService = userService;
22     }
23
24     @GetMapping("/me")
25     public ResponseEntity<User> authenticatedUser() {
26         Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
27
28         User currentUser = (User) authentication.getPrincipal();
29
30         return ResponseEntity.ok(currentUser);
31     }
32
33     @GetMapping("/")
34     public ResponseEntity<List<User>> allUsers() {
35         List <User> users = userService.allUsers();
36
37         return ResponseEntity.ok(users);
38     }
39 }
```

Demo JWT với Spring Boot 3 – Security 6

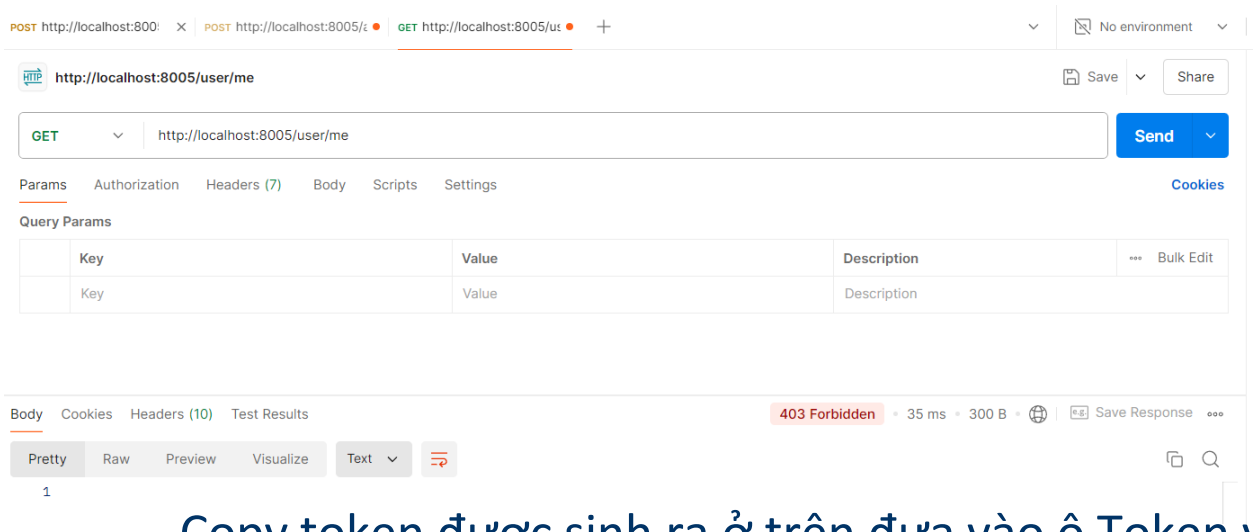
- Bước 9: Test: Tạo tài khoản -> Đăng nhập tài khoản sinh JWT

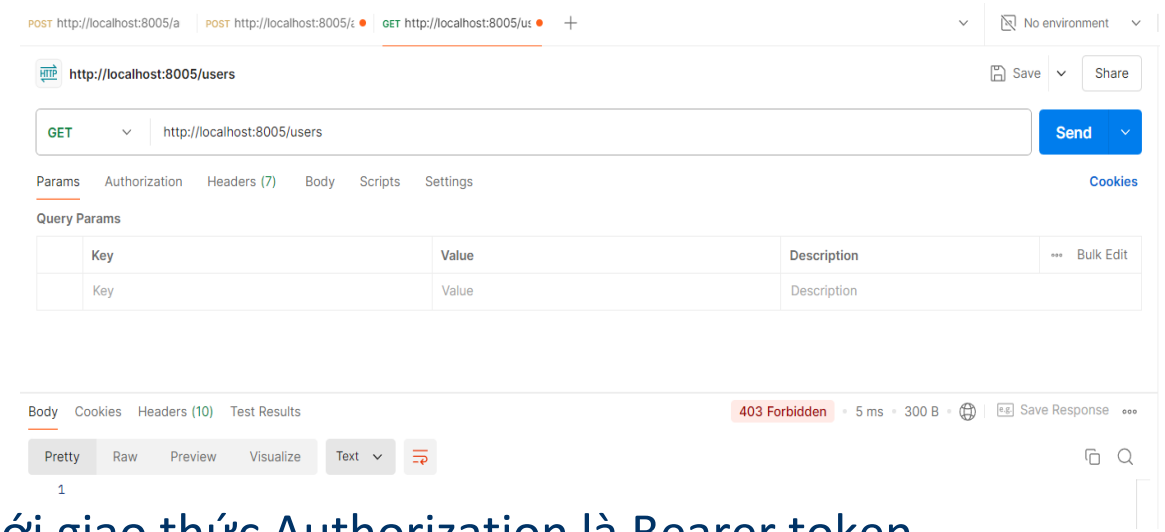




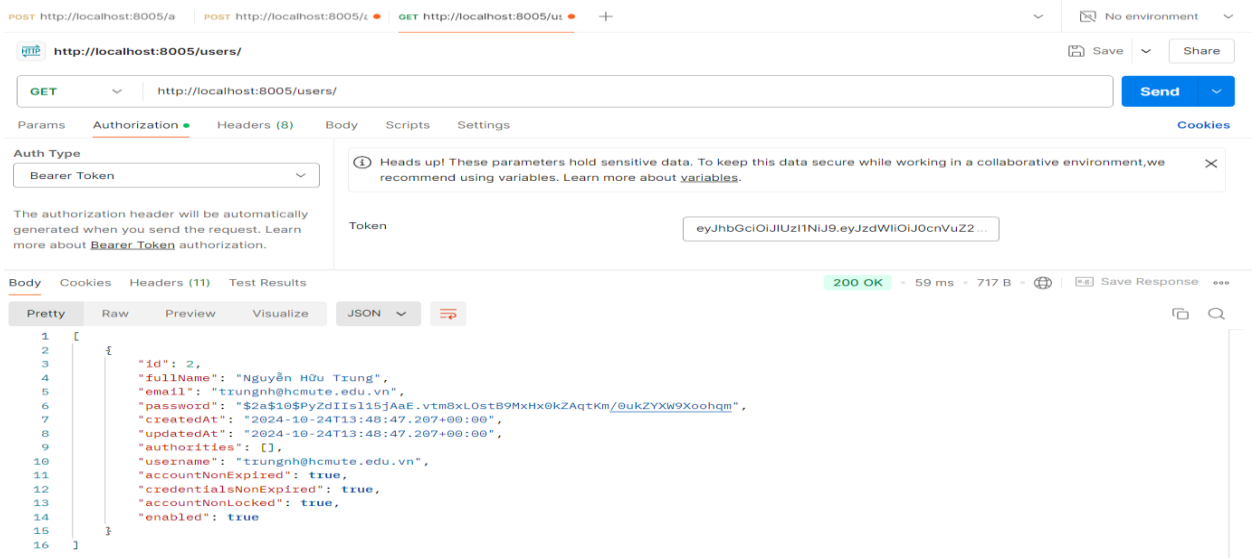
Demo JWT với Spring Boot 3 – Security 6

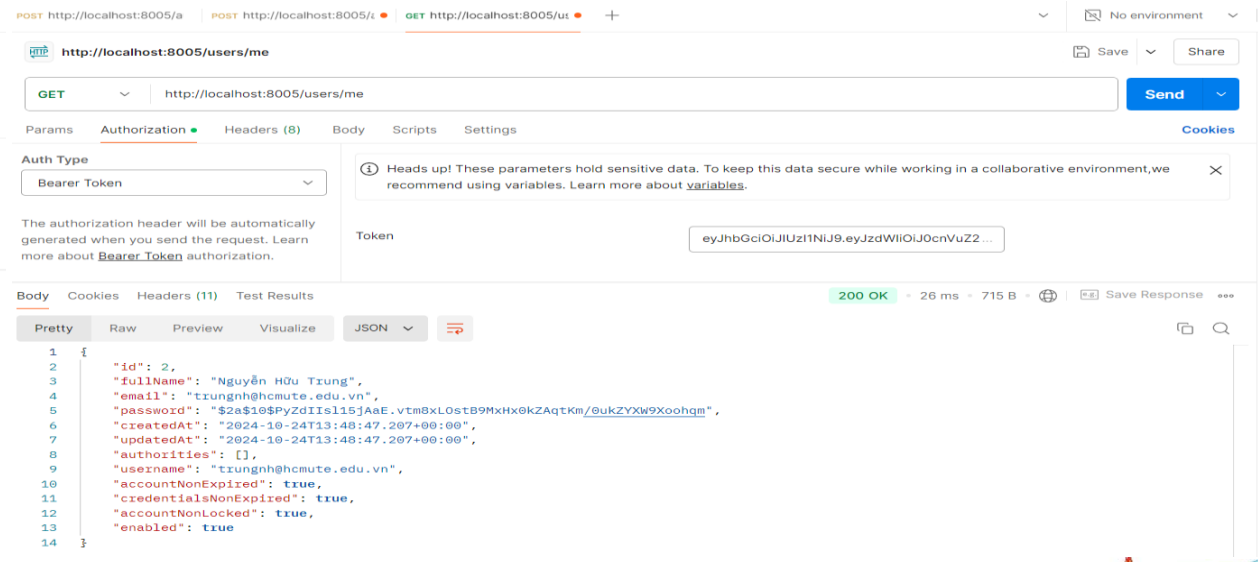
- Bước 9: Test: Các điểm cuối lần lượt /users/me và /users trả về người dùng đã xác thực từ mã thông báo





- Copy token được sinh ra ở trên đưa vào ô Token với giao thức Authorization là Bearer token.





Demo JWT với Spring Boot 3 – Security 6

- Bước 10: Ném Exception

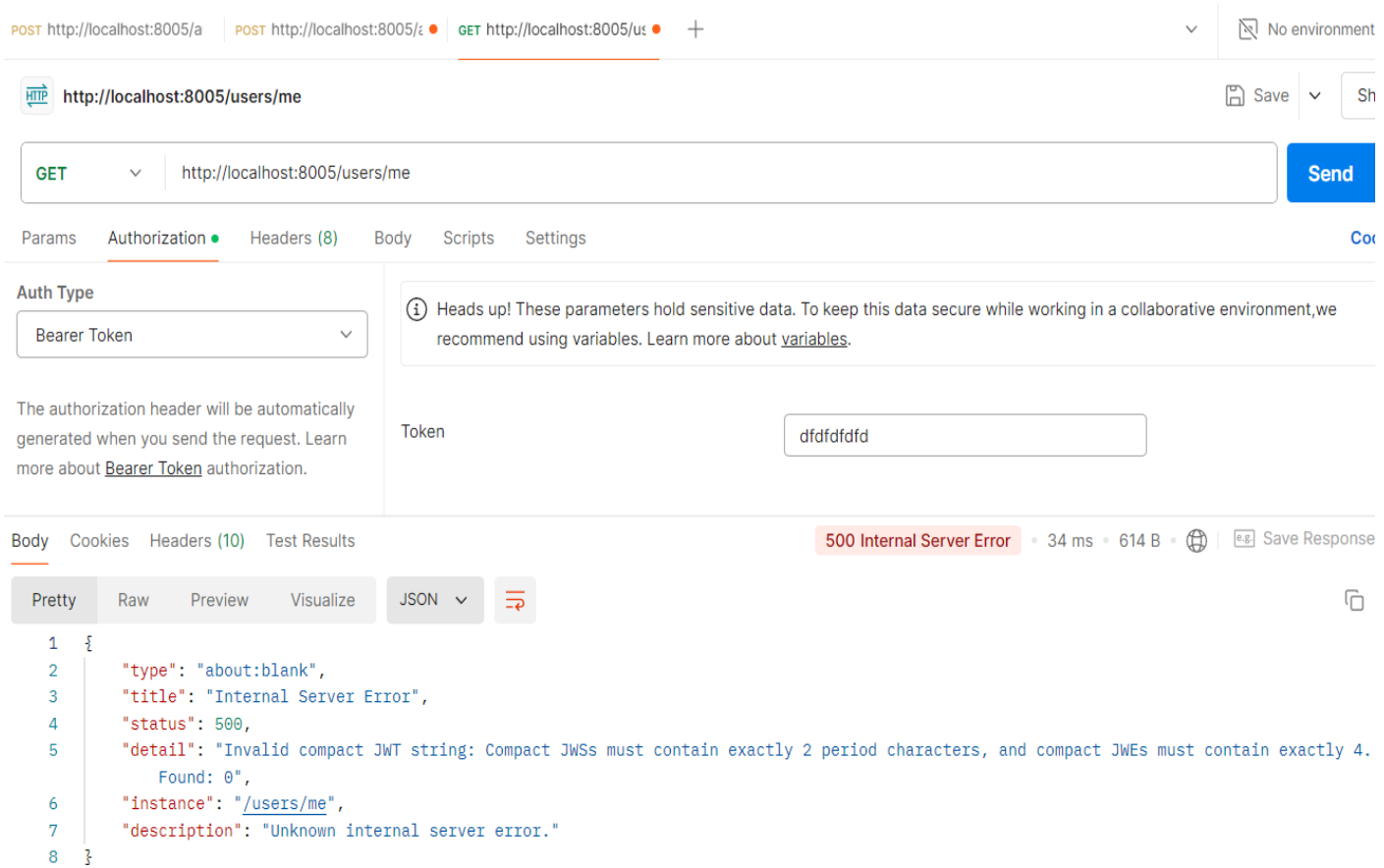
Lỗi xác thực	Ngoại lệ được ném	Mã trạng thái HTTP
Thông tin đăng nhập không hợp lệ	Ngoại lệ BadCredentials	401
Tài khoản bị khóa	Tài khoảnStatusException	403
Không được phép truy cập vào tài nguyên	Ngoại lệ AccessDenied	403
JWT không hợp lệ	Ngoại lệ chữ ký	401
JWT đã hết hạn	Hết hạnJwtException	401

```java
@RestControllerAdvice
public class GlobalExceptionHandler {
    @ExceptionHandler(Exception.class)
    public ProblemDetail handleSecurityException(Exception exception) {
        ProblemDetail errorDetail = null;
        // TODO send this stack trace to an observability tool
        exception.printStackTrace();

        if (exception instanceof BadCredentialsException) {
            errorDetail = ProblemDetail.forStatusAndDetail(HttpStatusCode.valueOf(401), exception.getMessage());
            errorDetail.setProperty("description", "The username or password is incorrect");

            return errorDetail;
        }
        if (exception instanceof AccountStatusException) {
            errorDetail = ProblemDetail.forStatusAndDetail(HttpStatusCode.valueOf(403), exception.getMessage());
            errorDetail.setProperty("description", "The account is locked");
        }
        if (exception instanceof AccessDeniedException) {
            errorDetail = ProblemDetail.forStatusAndDetail(HttpStatusCode.valueOf(403), exception.getMessage());
            errorDetail.setProperty("description", "You are not authorized to access this resource");
        }
        if (exception instanceof SignatureException) {
            errorDetail = ProblemDetail.forStatusAndDetail(HttpStatusCode.valueOf(403), exception.getMessage());
            errorDetail.setProperty("description", "The JWT signature is invalid");
        }
        if (exception instanceof ExpiredJwtException) {
            errorDetail = ProblemDetail.forStatusAndDetail(HttpStatusCode.valueOf(403), exception.getMessage());
            errorDetail.setProperty("description", "The JWT token has expired");
        }
        if (errorDetail == null) {
            errorDetail = ProblemDetail.forStatusAndDetail(HttpStatusCode.valueOf(500), exception.getMessage());
            errorDetail.setProperty("description", "Unknown internal server error.");
        }
        return errorDetail;
    }
}
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 10: Ném Exception

Lỗi xác thực	Ngoại lệ được ném	Mã trạng thái HTTP
Thông tin đăng nhập không hợp lệ	Ngoại lệ BadCredentials	401
Tài khoản bị khóa	Tài khoảnStatusException	403
Không được phép truy cập vào tài nguyên	Ngoại lệ AccessDenied	403
JWT không hợp lệ	Ngoại lệ chữ ký	401
JWT đã hết hạn	Hết hạnJwtException	401



Demo JWT với Spring Boot 3 – Security 6

- Bước 10: Render in Ajax
  - Tạo view login.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="utf-8">
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Login</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">
</head>
<body>
  <div class="container" style="min-height: 500px">
            <form action="" method="post">
                    <label>Email</label>
                    <input type="email" name ="email" id="email" required="required">
                    <label>Password</label>
                    <input type="password" name ="password" id="password"
required="required" autocomplete="on">
                     <button id="Login" type="button">Login</button>
            </form>
  </div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/jquery@3.7.1/dist/jquery.min.js"></script>
<script src="/js/mainjs.js"></script>
</body>
</html>
```

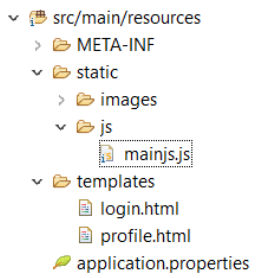
Tạo view profile.html

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4      <title>Spring Boot Rest API with ajax</title>
5      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
6       <!-- Required meta tags -->
7      <meta charset="utf-8">
8      <meta name="viewport" content="width=device-width, initial-scale=1">
9      <!-- Bootstrap CSS -->
10     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
11     integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
12 </head>
13 <body>
14 <div class="container" style="min-height: 500px">
15     <div class="starter-template">
16         <h1>Spring Boot REST API with AJAX Example</h1>
17         <img id="images" src="" alt="" width="100">
18         <div id="profile"></div>
19         <button id="logout">Logout</button>
20     </div>
21 </div>
22   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
23   integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>
24 <script src="https://cdn.jsdelivr.net/npm/jquery@3.7.1/dist/jquery.min.js"></script>
25 <script src="/js/mainjs.js"></script>
26 </body>
27 </html>
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 10: Render in Ajax
  - Tạo file javascript để render API: mainjs.js

```javascript
$(document).ready(function() {
    //Hiển thị thông tin người dùng đăng nhập thành công
     $.ajax({
       type: 'GET',
       url: '/users/me',
       dataType: 'json',
       contentType: "application/json; charset=utf-8",
       beforeSend: function(xhr) {
         if (localStorage.token) {
           xhr.setRequestHeader('Authorization', 'Bearer ' + localStorage.token);
         }
       },
       success: function(data) {
             var json = JSON.stringify(data, null, 4);
            // $('#profile').html(json);
             $('#profile').html( data.fullName);
             $('#images').html(document.getElementById("images").src=data.images);
             //console.log("SUCCESS : ", data);
         //alert('Hello ' + data.email + '! You have successfully accessed to /api/profile.');
       },
       error: function() {
            var json = e.responseText ;
            $('#feedback').html(json);
           // console.log("ERROR : ", e);
               alert("Sorry, you are not logged in.");
       }
     });
```




```javascript
    //Hàm đăng xuất
    $('#logout').click(function() {
        localStorage.clear();
        window.location.href = "/login";
    })

    //hàm Login
    $('#Login').click(function() {
        var email =document.getElementById('email').value;
        var password = document.getElementById('password').value;
        var basicInfo = JSON.stringify({
            email:email,
            password:password
        });
         $.ajax({
           type: "POST",
           url:"/auth/login",
           dataType: 'json',
           contentType: "application/json; charset=utf-8",
           data: basicInfo,
           success: function(data) {
             localStorage.token = data.token;
            // alert('Got a token from the server! Token: ' + data.token);
             window.location.href = "/user/profile";
           },
           error: function() {
             alert("Login Failed");
           }
         });
       });
});
```

Demo JWT với Spring Boot 3 – Security 6

- Bước 10: Render in Ajax
  - Viết Controller: AuthController.java gọi AJAX và cấu hình thêm Security

```java
@Controller
@RequestMapping("/")
public class AuthController {

    @GetMapping("login")
    public String index() {
        return "login";
    }
    @GetMapping("user/profile")
    public String profile() {
        return "profile";
    }
}
```

```
.requestMatchers("/login**").permitAll()
.requestMatchers("/user/**").permitAll()
.requestMatchers(new AntPathRequestMatcher("/images/**")).permitAll()
.requestMatchers(new AntPathRequestMatcher("/js/**")).permitAll()
```