Spring Cloud로 개발하는 마이크로서비스 애플리케이션

+

목차

Part 1

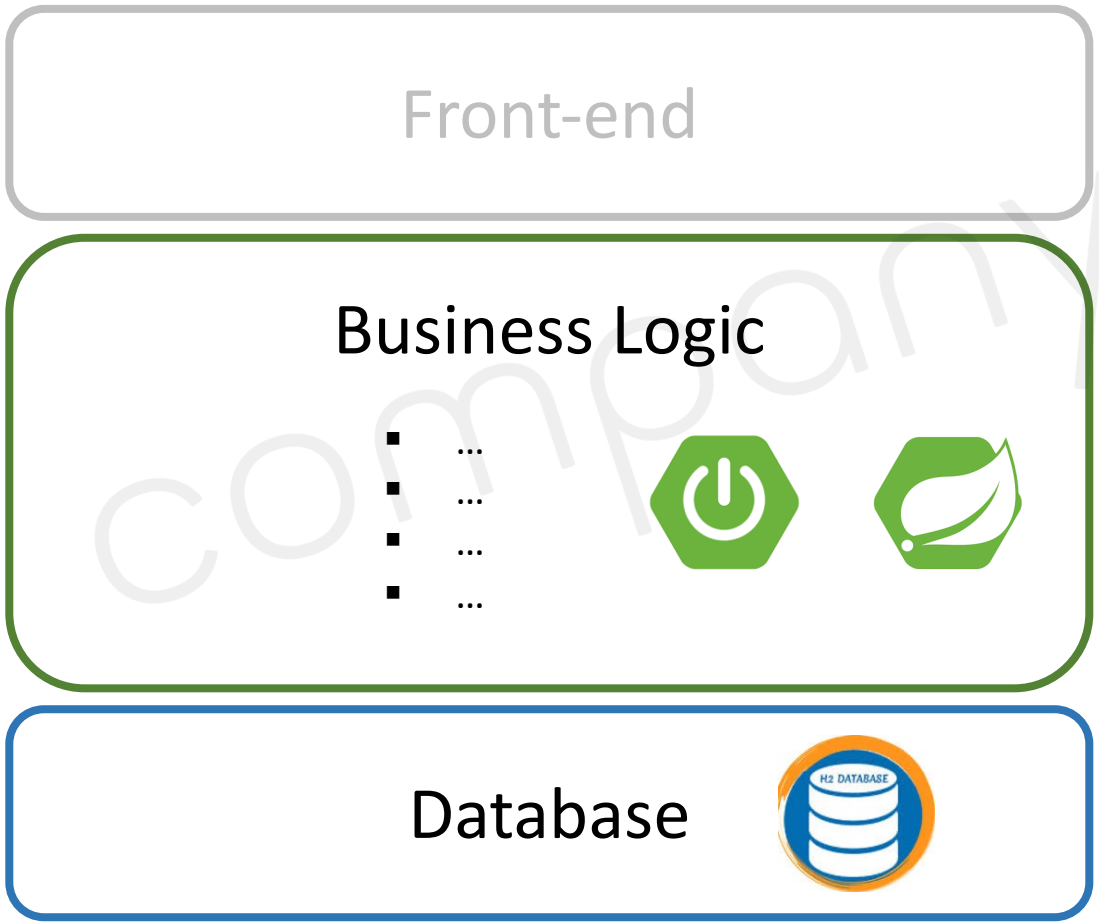
Section 6.

Users Microservice - ②

- Users Microservice – Login
- JWT (Json Web Token)
- API Gateway service – AuthorizationHeaderFilter

Users Microservice 기능 추가 – Login

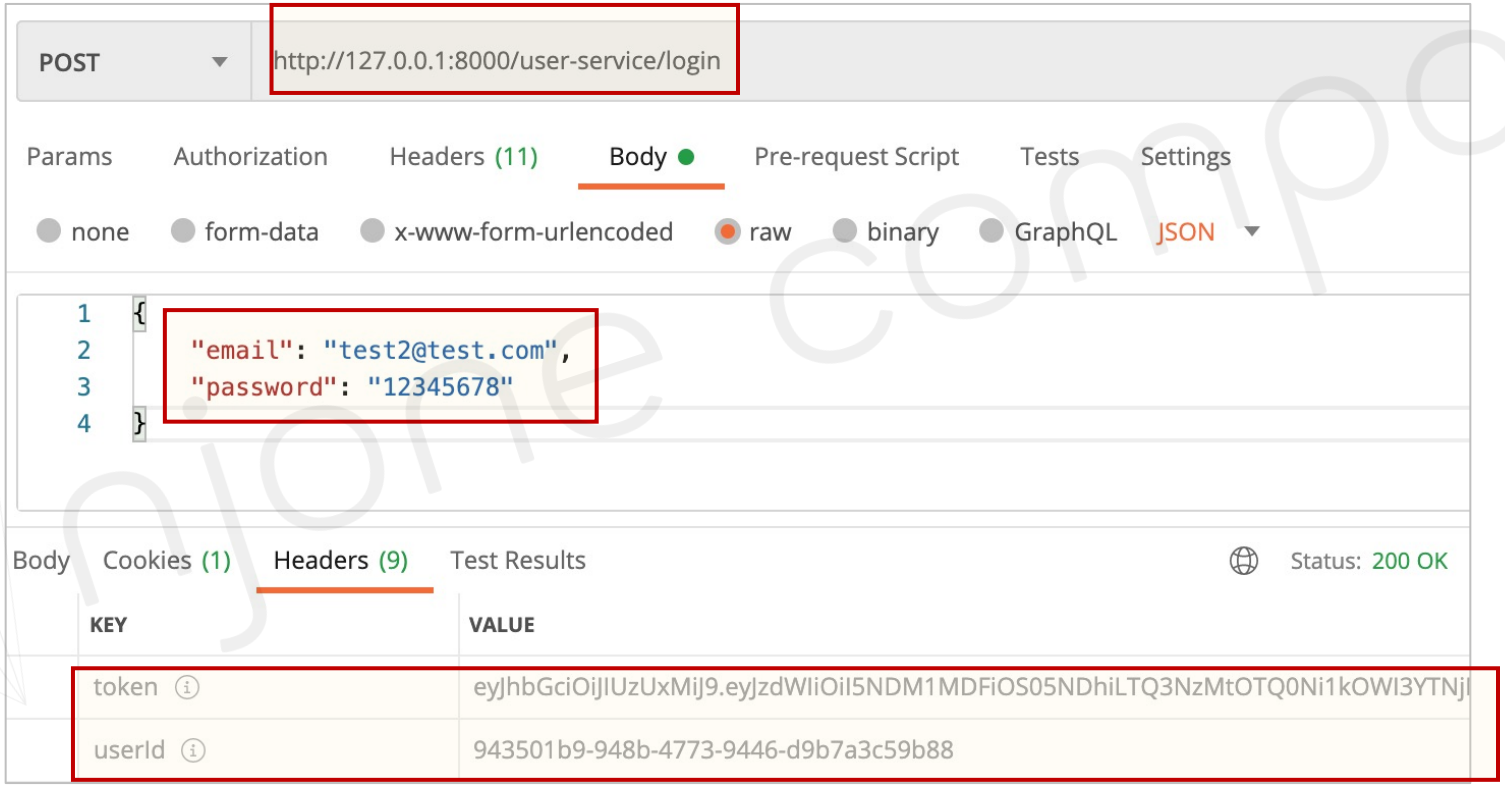
- Features
  - ~~신규 회원 등록~~
  - 회원 로그인
  - ~~상세 정보 확인~~
  - 회원 정보 수정/삭제
  - ~~상품 주문~~
  - ~~주문 내역 확인~~



Users Microservice 기능 추가 – Login

- APIs

기능	URI (API Gateway 사용 시)	URI (API Gateway 미사용 시)	HTTP Method
사용자 로그인	/user-service/login	/login	POST



Users Microservice 기능 추가 – Login

- RequestLogin.java
  - 사용자 로그인 정보를 저장하기 위한 RequestLogin Model 클래스 추가

```java
@Data
public class RequestLogin {
    @NotNull(message = "Email cannot be null")
    @Size(min = 2, message = "Email not be less than two characters")
    @Email
    private String email;

    @NotNull(message = "Password cannot be null")
    @Size(min = 8, message = "Password must be equal or grater than 8 characters and less than 16 characters")
    private String password;
}
```

Users Microservice 기능 추가 – Login

- AuthenticationFilter.java
  - Spring Security를 이용한 로그인 요청 발생 시 작업을 처리해 주는 ***Custom Filter*** 클래스
  - UsernamePasswordAuthenticationFilter 상속
    - https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/web/authentication/UsernamePasswordAuthenticationFilter.html

      org.springframework.security.web.authentication

      Class UsernamePasswordAuthenticationFilter

      java.lang.Object
      org.springframework.web.filter.GenericFilterBean
      org.springframework.security.web.authentication.AbstractAuthenticationProcessingFilter
      org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter

  - attemptAuthentication(), successfulAuthentication() 함수 구현

Users Microservice 기능 추가 – Login

- AuthenticationFilter.java

```java
public class AuthenticationFilter extends UsernamePasswordAuthenticationFilter {
    @Override
    public Authentication attemptAuthentication(HttpServletRequest request,
                                                HttpServletResponse response)
            throws AuthenticationException {
        try {
            RequestLogin creds = new ObjectMapper().readValue(request.getInputStream(), RequestLogin.class);

            return getAuthenticationManager().authenticate(
                    new UsernamePasswordAuthenticationToken(
                            creds.getEmail(),
                            creds.getPassword(),
                            new ArrayList<>()
                    )
            );
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

```java
    @Override
    protected void successfulAuthentication(HttpServletRequest request,
                                            HttpServletResponse response,
                                            FilterChain chain,
                                            Authentication authResult)
            throws IOException, ServletException {

    }
```

Users Microservice 기능 추가 – Login

- WebSecurity.java
  - 사용자 요청에 대해 AuthenticationFilter를 거치도록 수정

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable();
//      http.authorizeRequests().antMatchers("/users/**").permitAll();
    http.authorizeRequests().antMatchers( ...antPatterns: "/**") ExpressionUrlAuthorizati
            .hasIpAddress( ipaddressExpression: "192.168.0.8") ExpressionUrlAuthorizatio
            .and() HttpSecurity
            .addFilter(getAuthenticationFilter());

    http.headers().frameOptions().disable();
}

private AuthenticationFilter getAuthenticationFilter() throws Exception {
    AuthenticationFilter authenticationFilter = new AuthenticationFilter();
    authenticationFilter.setAuthenticationManager(authenticationManager());

    return authenticationFilter;
}
```

Gateway IP

Users Microservice 기능 추가 – Login

- WebSecurity.java
  - 사용자 요청에 대해 AuthenticationFilter를 거치도록 수정

```
@Configuration
@EnableWebSecurity
public class WebSecurity extends WebSecurityConfigurerAdapter {
    private UserService userService;
    private BCryptPasswordEncoder bCryptPasswordEncoder;
    private Environment env;

    @Autowired
    public WebSecurity(UserService userService, Environment env, BCryptPasswordEncoder

    @Override
    protected void configure(HttpSecurity http) throws Exception {...}

    private AuthenticationFilter getAuthenticationFilter() throws Exception {...}

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userService).passwordEncoder(bCryptPasswordEncoder);
    }
}
```

Users Microservice 기능 추가 – Login

- UserDetailsService 등록
  - 기존의 UserService.java 활용

```java
public interface UserService extends UserDetailsService {
    UserDto createUser(UserDto userDto);
    UserDto getUserByUserId(String userId);
    Iterable<UserEntity> getUserByAll();
```

org.springframework.security.core.userdetails

Interface UserDetailsService

All Known Subinterfaces:

UserDetailsManager

```java
public interface UserRepository extends CrudRepository<UserEntity, Long> {
    UserEntity findByUserId(String userId);
```

```java
@Service
public class UserServiceImpl implements UserService {
    @Override
    public UserDetails loadUserByUsername(String username)
            throws UsernameNotFoundException {
        UserEntity userEntity = userRepository.findByEmail(username);

        if (userEntity == null)
            throw new UsernameNotFoundException(username);

        return new User(userEntity.getEmail(), userEntity.getEncryptedPwd(),
                enabled: true, accountNonExpired: true, credentialsNonExpired: true, accountNonLocked: true,
                new ArrayList<>());
    }
```

Users Microservice 기능 추가 – Login

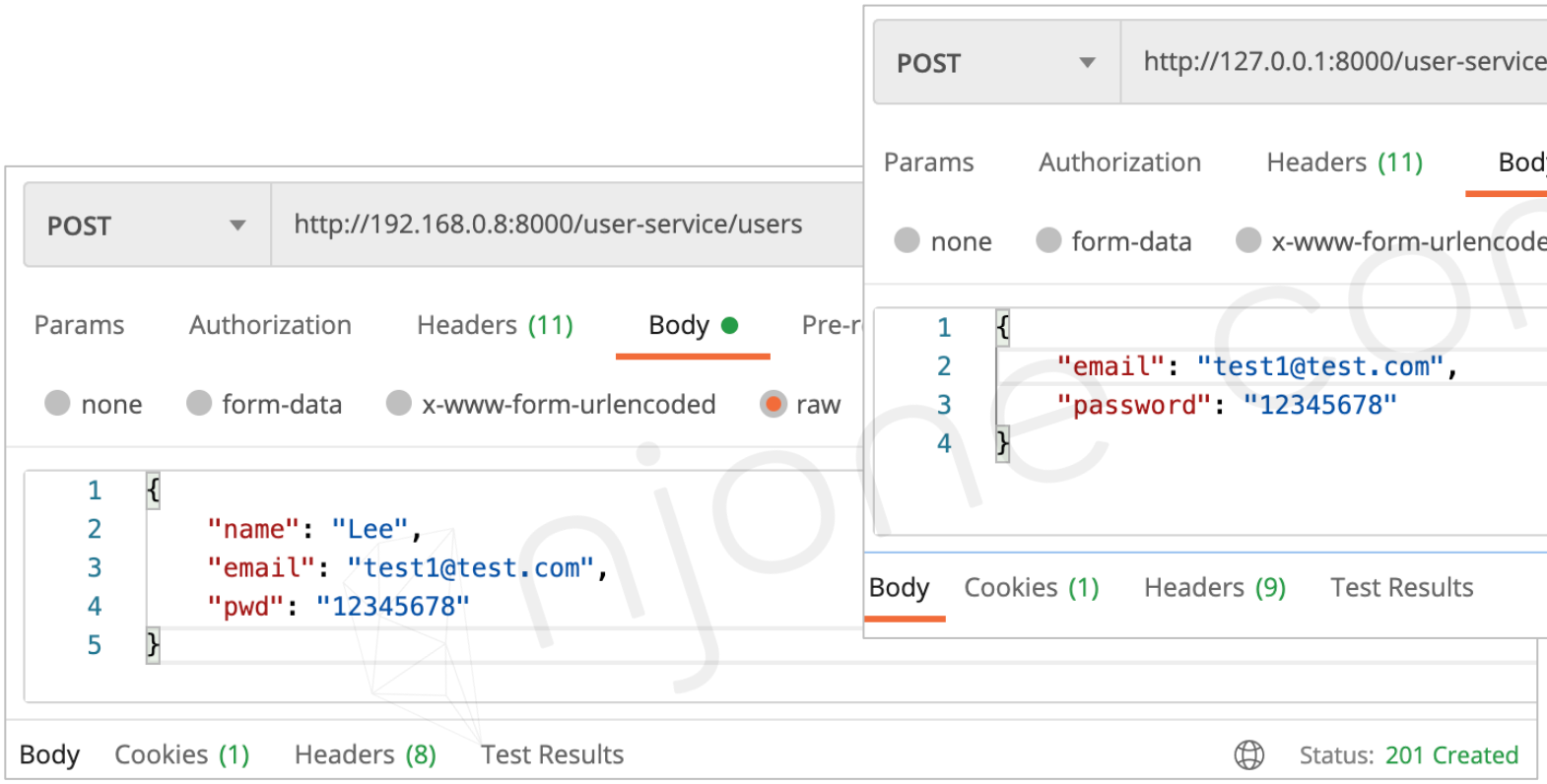
- API Gateway Service 수정
  - User Service에 대한 Routes 정보 수정

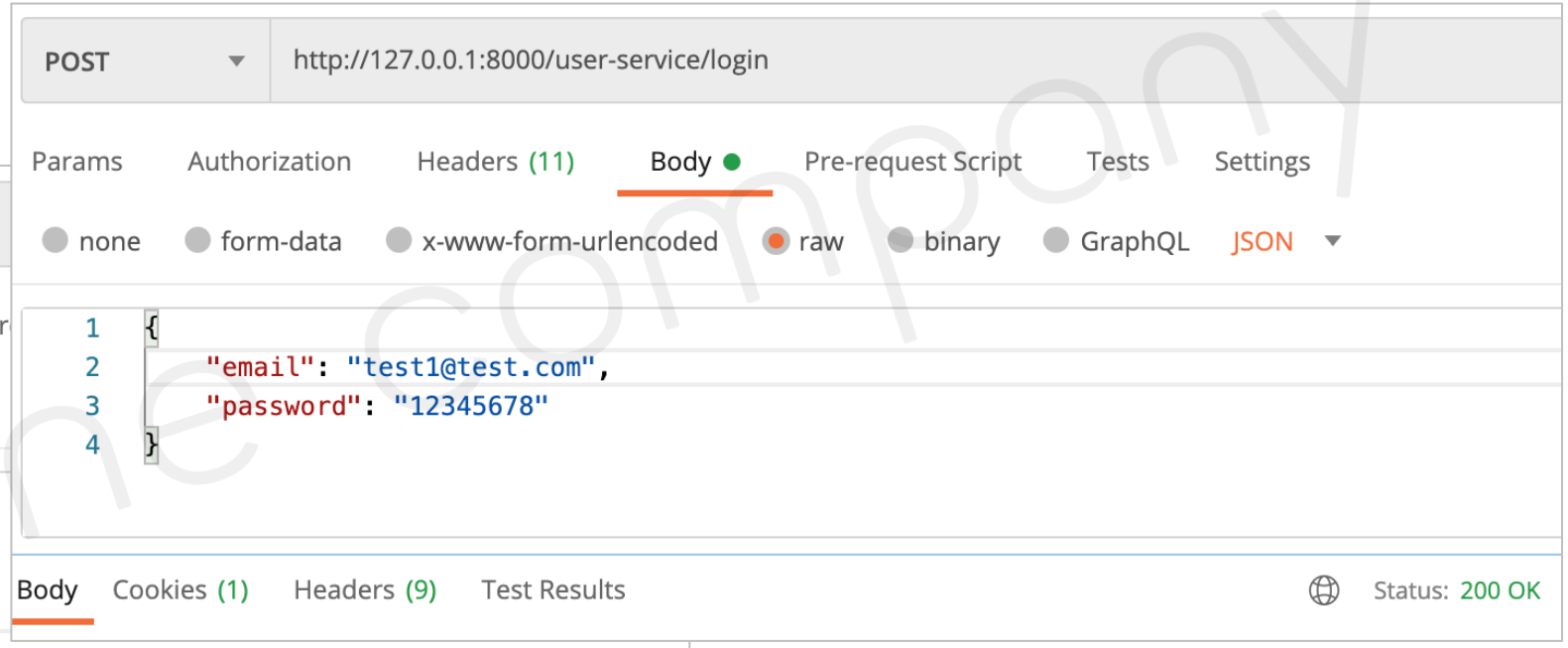
```
routes:
#   - id: user-service
#     uri: lb://USER-SERVICE
#     predicates:
#       - Path=/user-service/**
```

```
- id: user-service
  uri: lb://USER-SERVICE
  predicates:
    - Path=/user-service/login
    - Method=POST
  filters:
    - RemoveRequestHeader=Cookie
    - RewritePath=/user-service/(?<segment>.*), /$\{segment}
- id: user-service
  uri: lb://USER-SERVICE
  predicates:
    - Path=/user-service/users
    - Method=POST
  filters:
    - RemoveRequestHeader=Cookie
    - RewritePath=/user-service/(?<segment>.*), /$\{segment}
- id: user-service
  uri: lb://USER-SERVICE
  predicates:
    - Path=/user-service/**
    - Method=GET
  filters:
    - RemoveRequestHeader=Cookie
    - RewritePath=/user-service/(?<segment>.*), /$\{segment}
```

Users Microservice 기능 추가 – Login

- Test
  - http:///*[Gateway IP]*:8000/user-service/users → 회원가입
  - http:///*[Gateway IP]*:8000/user-service/login → 로그인

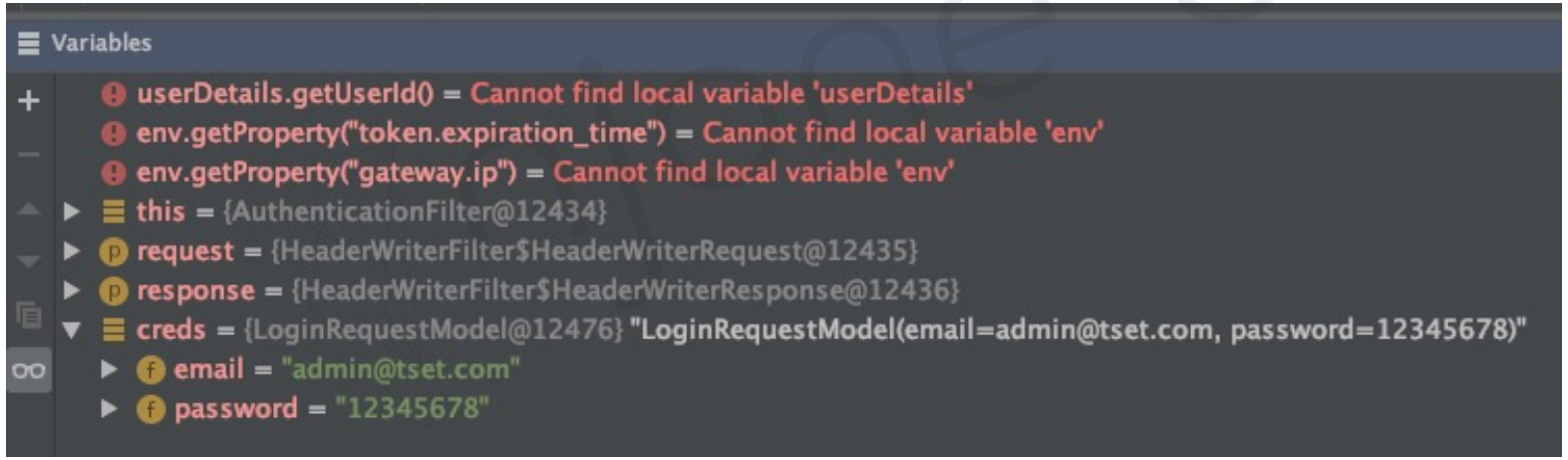




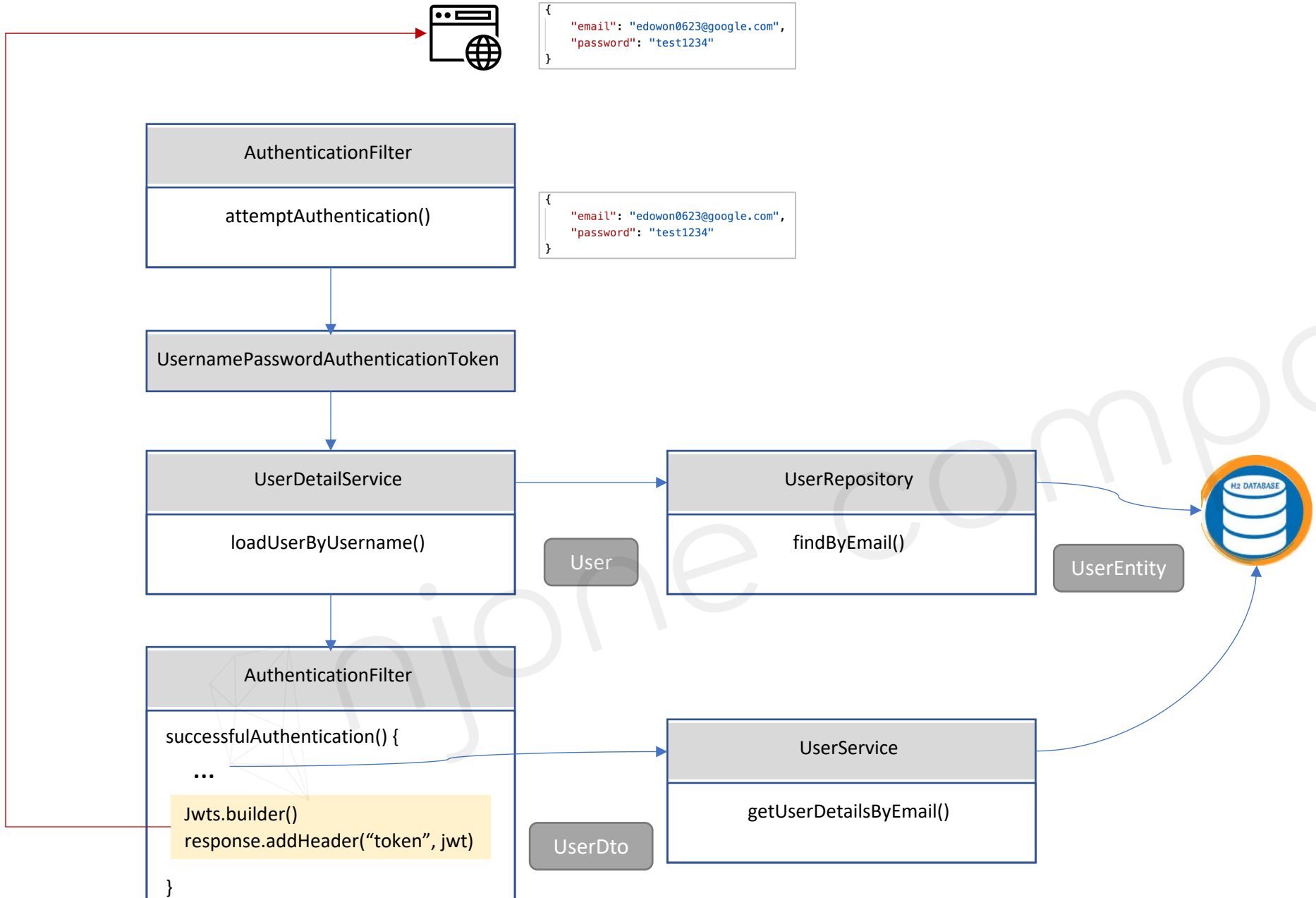
Users Microservice 기능 추가 – Login

- Test
  - Debug

```
42      @Override
43      public Authentication attemptAuthentication(HttpServletRequest request,  reques
44                                                  HttpServletResponse response)  resp
45              throws AuthenticationException {
46          try {
47              LoginRequestModel creds = new ObjectMapper()  creds: "LoginRequestModel
48                      .readValue(request.getInputStream(), LoginRequestModel.class);
49
50              return getAuthenticationManager().authenticate(
51                      new UsernamePasswordAuthenticationToken(
```



Users Microservice 기능 추가 – Login



Users Microservice 기능 추가 – Login

- AuthenticationFilter.java의 successfulAuthentication() 수정
  - 인증 성공 시 사용자에게 Token 발생

```
public class AuthenticationFilter extends UsernamePasswordAuthenticationFilter {
    private UserService userService;
    private Environment env;

    public AuthenticationFilter(UserService usersService,
                                Environment env,
                                AuthenticationManager authenticationManager) {
        this.env = env;
        this.userService = usersService;
        super.setAuthenticationManager(authenticationManager);
    }

    @Override
    protected void successfulAuthentication(HttpServletRequest request,
                                            HttpServletResponse response,
                                            FilterChain chain,
                                            Authentication authResult)
            throws IOException, ServletException {
        String userName = ((User)authResult.getPrincipal()).getUsername();
```

AuthenticationFilter.java

Users Microservice 기능 추가 – Login

- WebSecurity.java의 configure() 수정
  - User Service 사용할 수 있도록 메소드 수정

```
private AuthenticationFilter getAuthenticationFilter() throws Exception {
    AuthenticationFilter authenticationFilter =
            new AuthenticationFilter(userService, env, authenticationManager());

    return authenticationFilter;
}
```

WebSecurity.java

Users Microservice 기능 추가 – Login

- UserService.java, UserServiceImpl.java, UserRepository.java
  - 사용자 인증을 위한 검색 메소드 추가

```java
public interface UserService extends UserDetailsService {
    UserDto createUser(UserDto userDto);
    UserDto getUserByUserId(String userId);
    Iterable<UserEntity> getUserByAll();

    UserDto getUserDetailsByEmail(String email);
}
```

```java
public interface UserRepository extends CrudRepository<UserEntity, Long> {
    UserEntity findByUserId(String userId);
    UserEntity findByEmail(String email);
}
```

UserRepository.java

```java
    @Override
    public UserDto getUserDetailsByEmail(String email) {
        UserEntity userEntity = userRepository.findByEmail(email);

//        if (userEntity == null)
//            throw new UsernameNotFoundException(email);

        return new ModelMapper().map(userEntity, UserDto.class);
    }
}
```

UserServiceImpl.java

Users Microservice 기능 추가 – Login

- AuthenticationFilter.java의 successfulAuthentication() 수정
  - 인증 성공 시 사용자에게 Token 발생

```
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
```

```
token:
  expiration_time: 864000000 #10 days
  secret: user_token
```

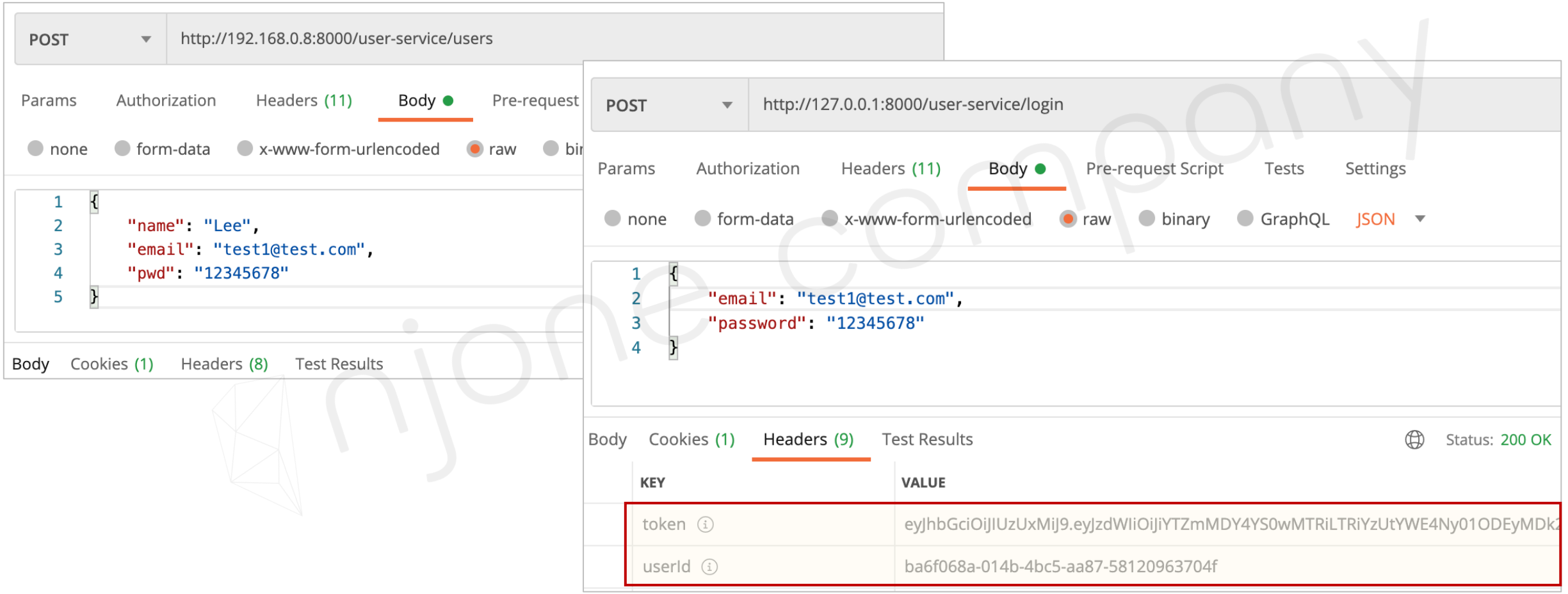
```
@Override
protected void successfulAuthentication(HttpServletRequest request,
                                        HttpServletResponse response,
                                        FilterChain chain,
                                        Authentication authResult)
        throws IOException, ServletException {
    String userName = ((User)authResult.getPrincipal()).getUsername();
    UserDto userDetails = userService.getUserDetailsByEmail(userName);

    String token = Jwts.builder()
            .setSubject(userDetails.getUserId())
            .setExpiration(new Date(System.currentTimeMillis()
                    + Long.parseLong(env.getProperty("token.expiration_time"))))
            .signWith(SignatureAlgorithm.HS512, env.getProperty("token.secret"))
            .compact();

    response.addHeader( s: "token", token);
    response.addHeader( s: "userId", userDetails.getUserId());
}
```

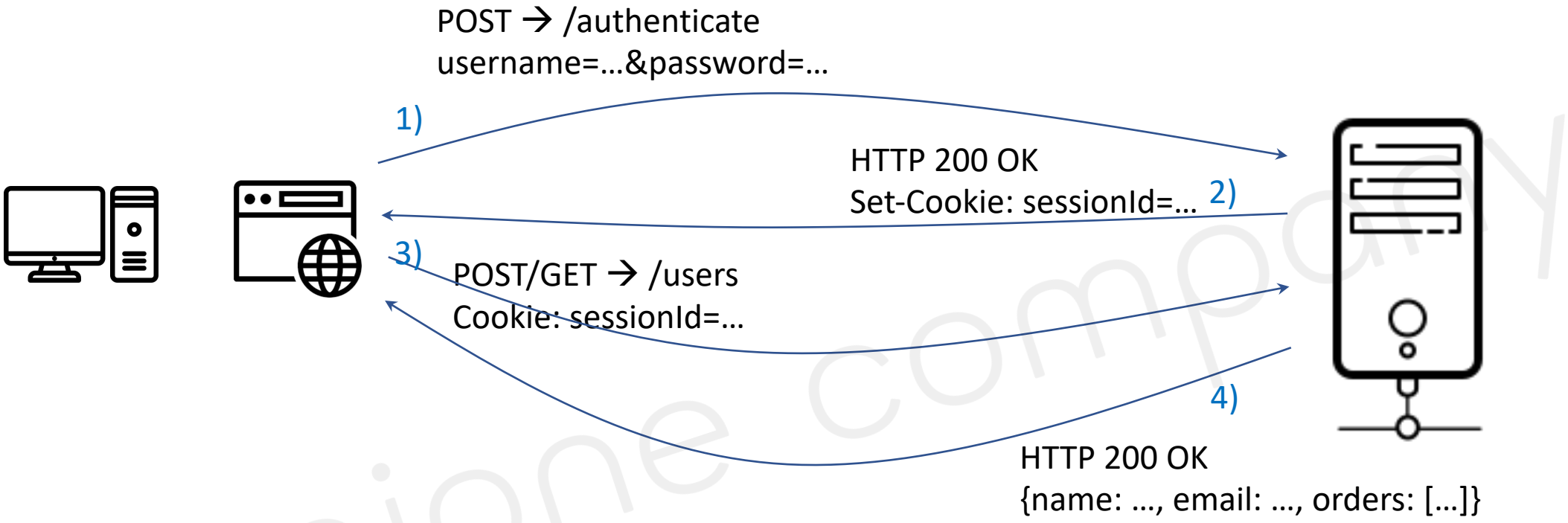
Users Microservice 기능 추가 – Login

- Test
  - http:///*[Gateway IP]*:8000/user-service/users → 회원가입
  - http:///*[Gateway IP]*:8000/user-service/login → 로그인



Users Microservice 기능 추가 – Login

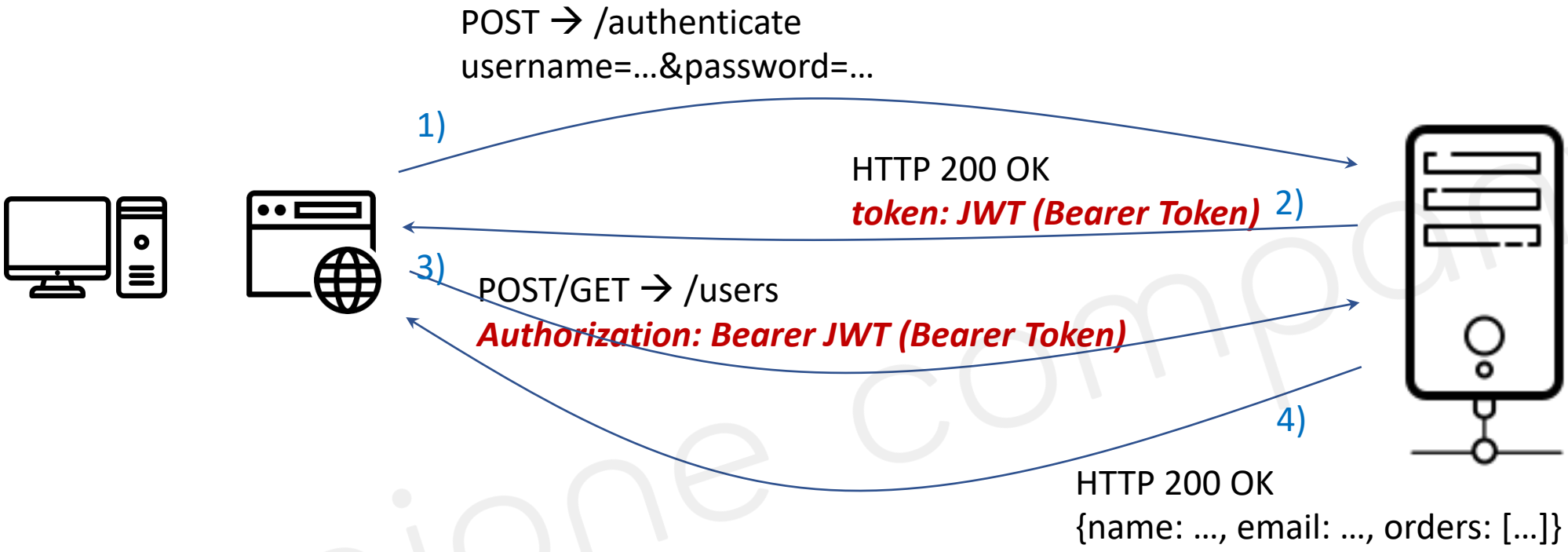
- 전통적인 인증 시스템



- *문제점*
  - *세션과 쿠키는 모바일 애플리케이션에서 유효하게 사용할 수 없음(공유 불가)*
  - *렌더링된 HTMl 페이지가 반환되지만, 모바일 애플리케이션에서는 JSON(or XML)과 같은 포멧 필요*

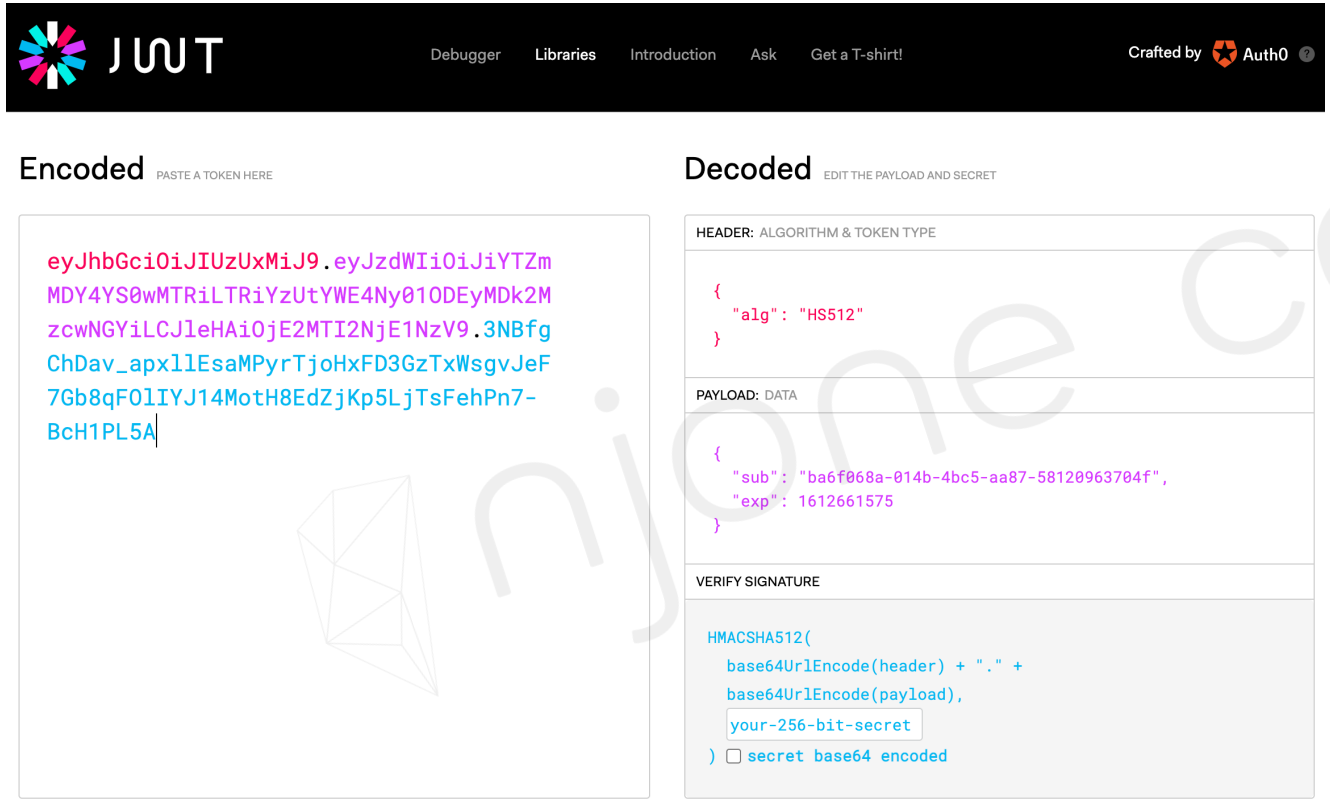
Users Microservice 기능 추가 – Login

- Token 기반 인증 시스템



Users Microservice 기능 추가 – Login

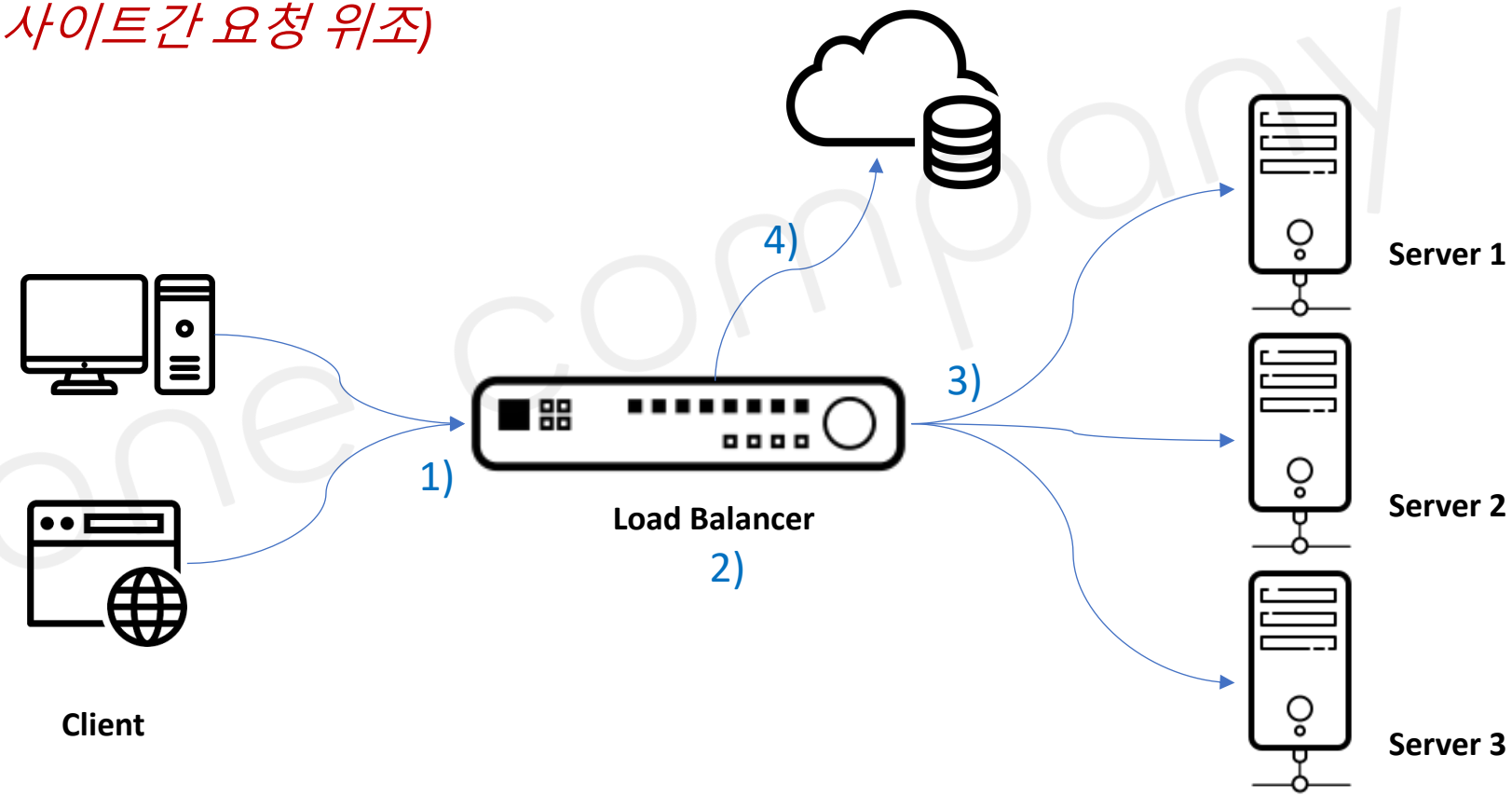
- JWT (JSON Web Token)
  - https://jwt.io/
  - 인증 헤더 내에서 사용되는 토큰 포맷
  - 두 개의 시스템끼리 안전한 방법으로 통신 가능



Language	Libary URL
NodeJS	http://github.com/auth0/node-jsonwebtoken
PHP	http://github.com/firebase/php-jwt
Java	http://github.com/auth0/java-jwt
Ruby	http://github.com/progrium/ruby-jwt
.NET	http://github.com/AzureAD/azure-activedirectory-identitymodel-extensions-for-dotnet
Python	http://github.com/progrium/pyjwt/

Users Microservice 기능 추가 – Login

- JWT (JSON Web Token) 장점
  - *클라이언트 독립적인 서비스 (stateless)*
  - *CDN*
  - *No Cookie-Session (No CSRF, 사이트간 요청 위조)*
  - *지속적인 토큰 저장*



Users Microservice 기능 추가 – Login

- API Gateway service에 Spring Security와 JWT Token 사용 추가
- AuthorizationHeaderFilter.java 추가

```
<!-- jwt token -->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
```

```
@Component
@Slf4j
public class AuthorizationHeaderFilter extends AbstractGatewayFilterFactory<AuthorizationHeaderFilter.Config> {
    Environment env;

    public AuthorizationHeaderFilter(Environment env) {
        this.env = env;
    }

    public static class Config {...}
```

Users Microservice 기능 추가 – Login

- AuthorizationHeaderFilter.java

```
@Override
public GatewayFilter apply(Config config) {
    return (exchange, chain) -> {
        ServerHttpRequest request = exchange.getRequest();

        if (!request.getHeaders().containsKey(HttpHeaders.AUTHORIZATION)) {
            return onError(exchange,  err: "No authorization header", HttpStatus.UNAUTHORIZED);
        }

        String authorizationHeader = request.getHeaders().get(HttpHeaders.AUTHORIZATION).get(0);
        String jwt = authorizationHeader.replace( target: "Bearer",  replacement: "");

        if (!isJwtValid(jwt)) {
            return onError(exchange,  err: "JWT token is not valid", HttpStatus.UNAUTHORIZED);
        }

        return chain.filter(exchange);
    };
}
```

Users Microservice 기능 추가 – Login

- AuthorizationHeaderFilter.java

```
token:
  secret: user_token
```

```
private Mono<Void> onError(ServerWebExchange exchange, String err, HttpStatus httpStatus) {
    ServerHttpResponse response = exchange.getResponse();
    response.setStatusCode(httpStatus);

    log.error(err);
    return response.setComplete();
}
```

```
private boolean isJwtValid(String jwt) {
    boolean returnValue = true;

    String subject = null;

    try {
        subject = Jwts.parser().setSigningKey(env.getProperty("token.secret"))
                              .parseClaimsJws(jwt).getBody()
                              .getSubject();
    } catch (Exception ex) {
        returnValue = false;
    }

    if (subject == null || subject.isEmpty()) {
        returnValue = false;
    }

    return returnValue;
}
```

Users Microservice 기능 추가 – Login

- application.yml

```
- id: user-service
  uri: lb://USER-SERVICE
  predicates:
    - Path=/user-service/**
    - Method=GET
  filters:
    - RemoveRequestHeader=Cookie
    - RewritePath=/user-service/(?<segment>.*), /$\{segment}
    - AuthorizationHeaderFilter
```

Authenticate 필요 X

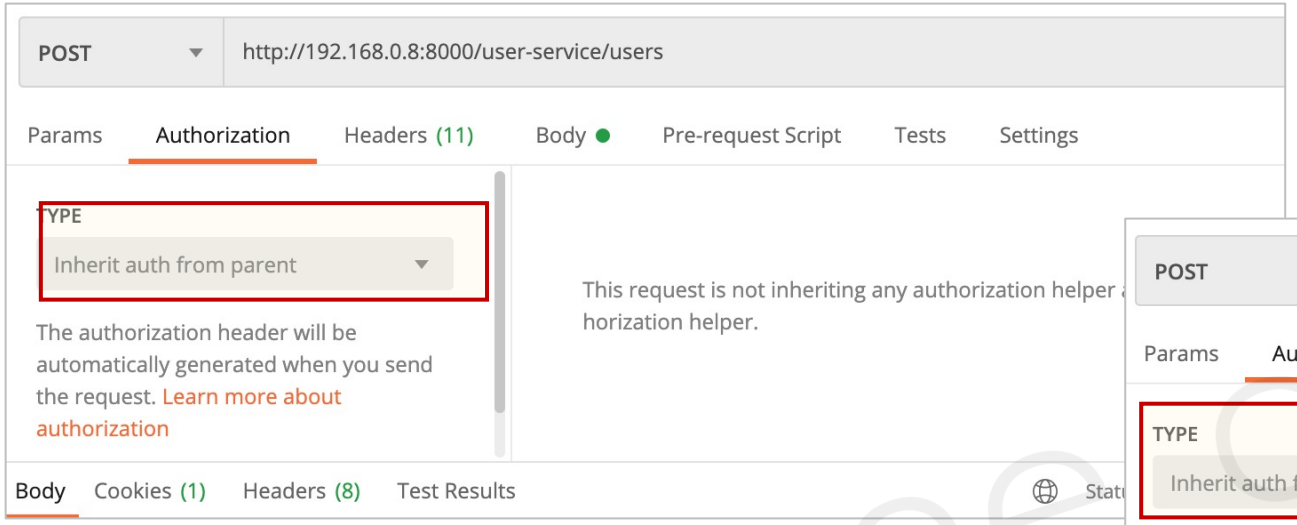
POST → /user-service/users (회원가입)
POST → /user-service/login (로그인)

Authenticate 필요 O

GET → /user-service/users (회원정보 확인)
GET → /user-service/health_check (상태 체크)

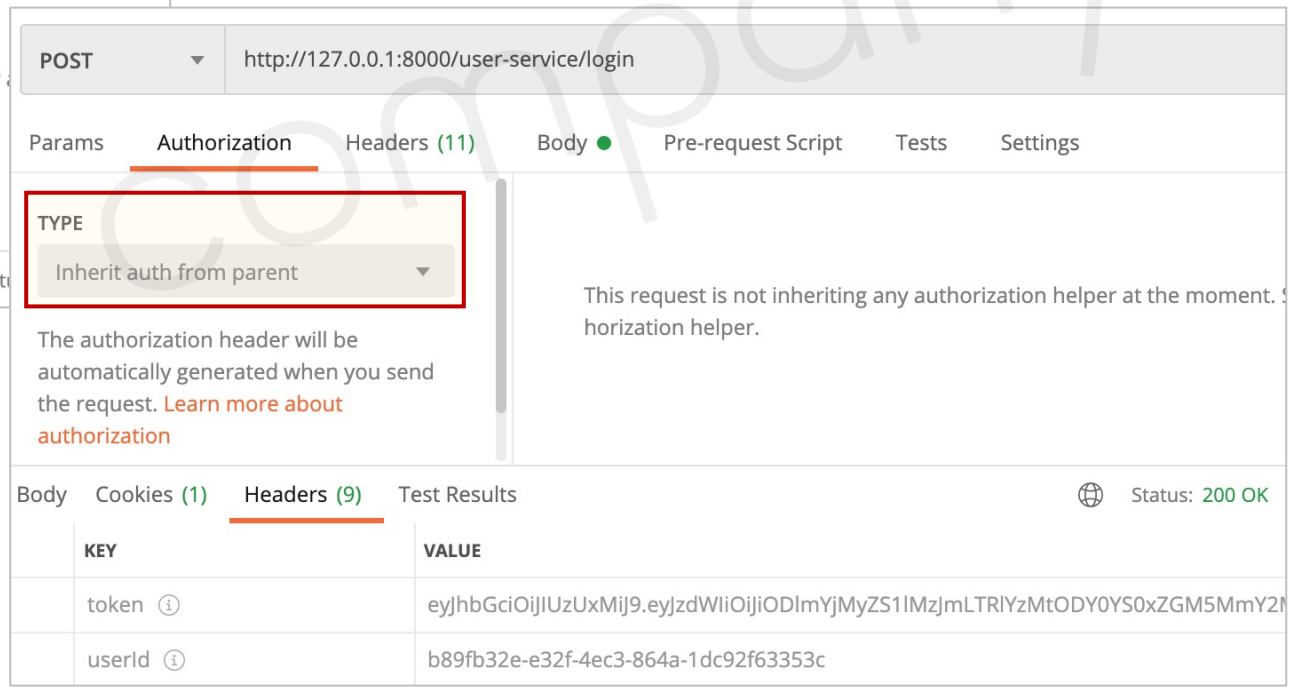
Users Microservice 기능 추가 – Login

- Test
  - Authorization Type → “No Auth” or “Inherit auth from parent”



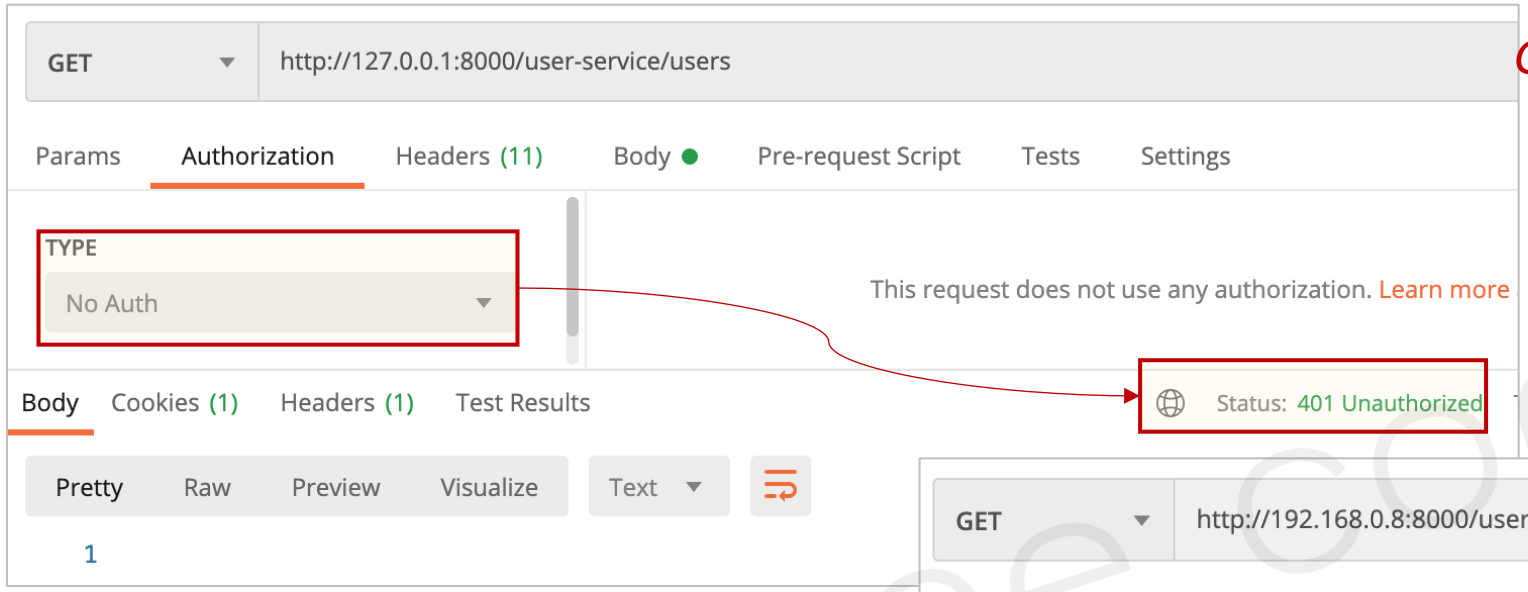
POST → /user-service/users (회원가입)

POST → /user-service/login (로그인)



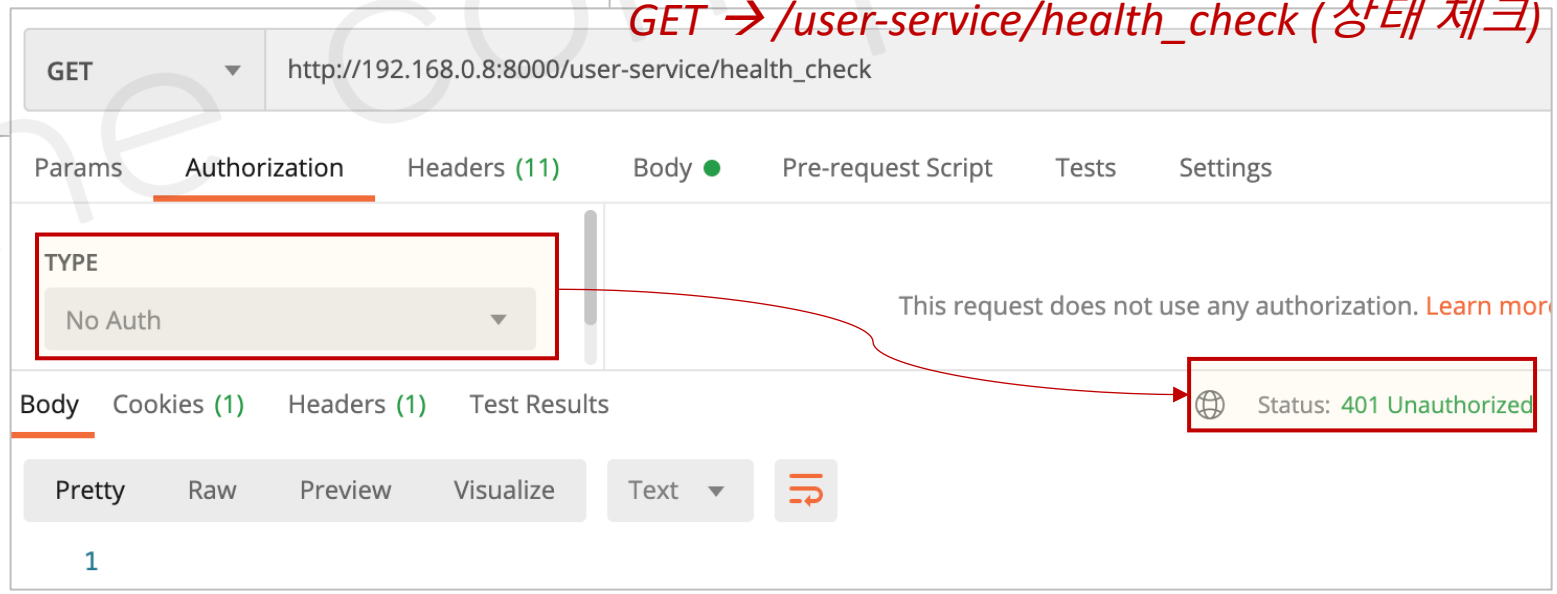
Users Microservice 기능 추가 – Login

- Test: Authorization Type → “No Auth” or “Inherit auth from parent”



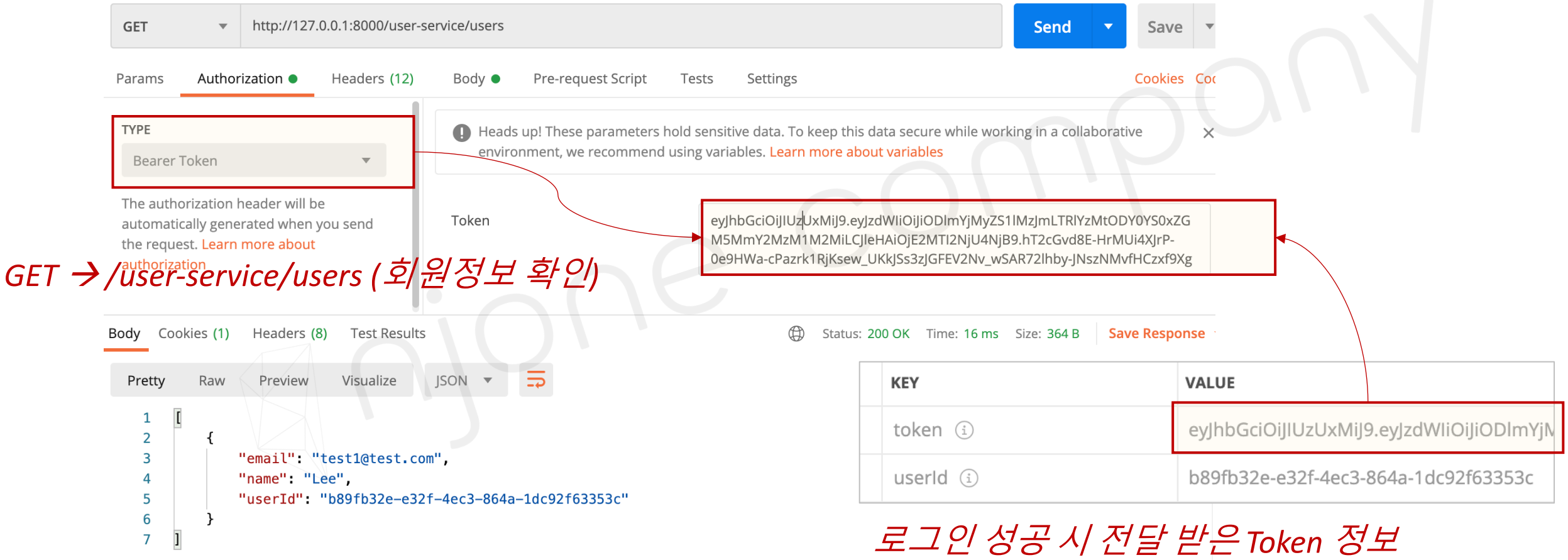
GET → /user-service/users (회원정보 확인)

GET → /user-service/health_check (상태 체크)



Users Microservice 기능 추가 – Login

- Test: Authorization Type → “Bearer Token”
  - Bearer Authentication:
    - API에 접속하기 위해서는 access token을 API 서버에 제출해서 인증 처리.
    - OAuth를 위해서 고안된 방법, RFC 6750

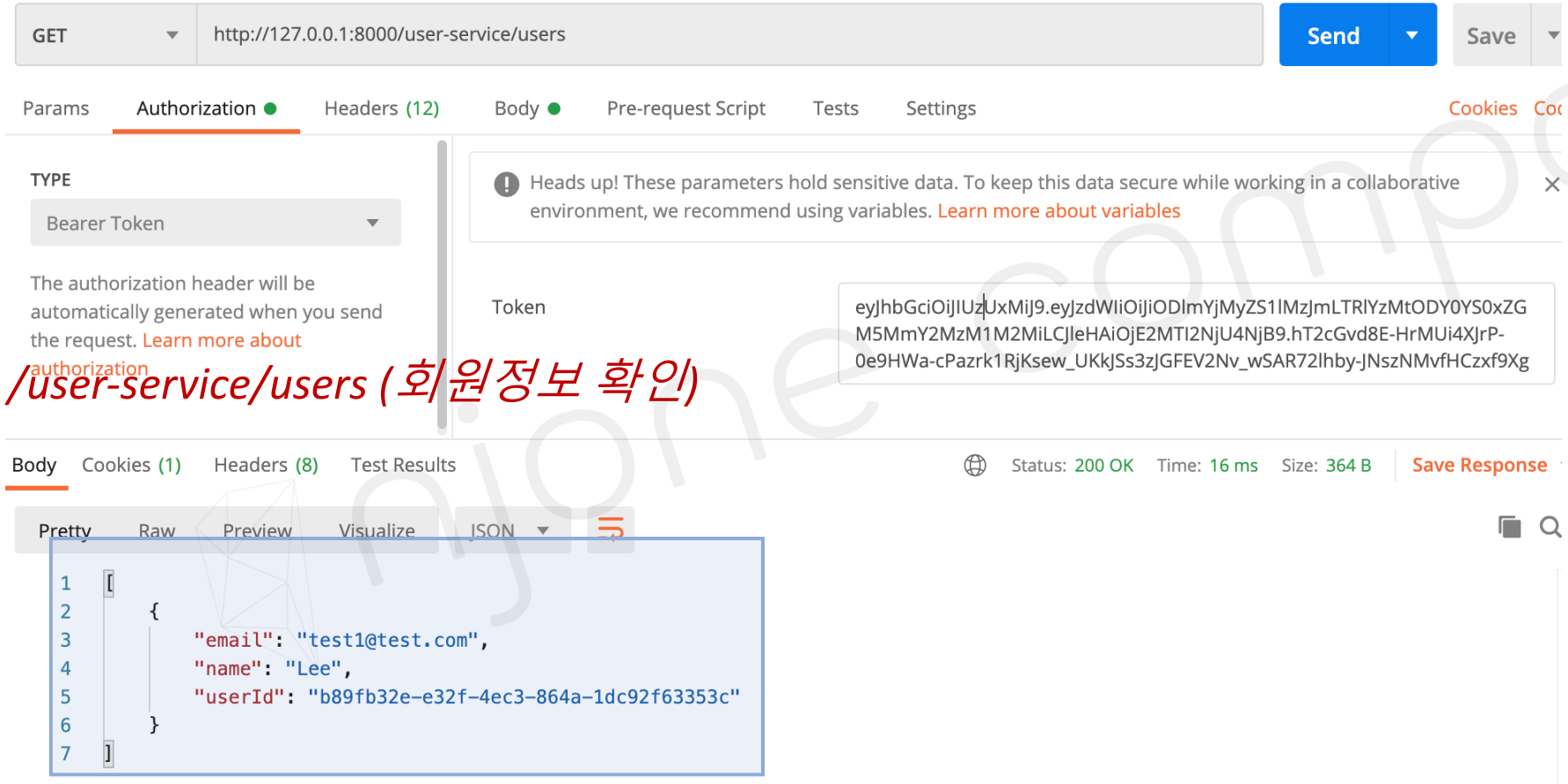


GET → /user-service/users (회원정보 확인)

로그인 성공 시 전달 받은 Token 정보

Users Microservice 기능 추가 – Login

- Test: Authorization Type → “Bearer Token”
  - Bearer Authentication:
    - API에 접속하기 위해서는 access token을 API 서버에 제출해서 인증 처리.
    - OAuth를 위해서 고안된 방법, RFC 6750



GET → /user-service/users (회원정보 확인)