```
mysql> desc users;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| email    | varchar(255) | NO   | PRI | NULL    |       |
| fname    | varchar(255) | YES  |     | NULL    |       |
| password | varchar(255) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

```
mysql> desc roles;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| name        | varchar(255) | NO   | PRI | NULL    |       |
| description | varchar(255) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
mysql> desc users_roles;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| user_fk  | varchar(255) | NO   | PRI | NULL    |       |
| role_fk  | varchar(255) | NO   | PRI | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

```
### @name="LOGIN"
POST http://localhost:8080/auth/signin
Content-Type: application/json
Accept: application/json

{
  "email": "sam@adobe.com",
  "password": "secret"
}
```

```
// login
@PostMapping("/signin")
public ResponseEntity<JwtAuthenticationResponse> signin(@RequestBody SigninRequest request) {
    return ResponseEntity.ok(authenticationService.signin(request));
}
```

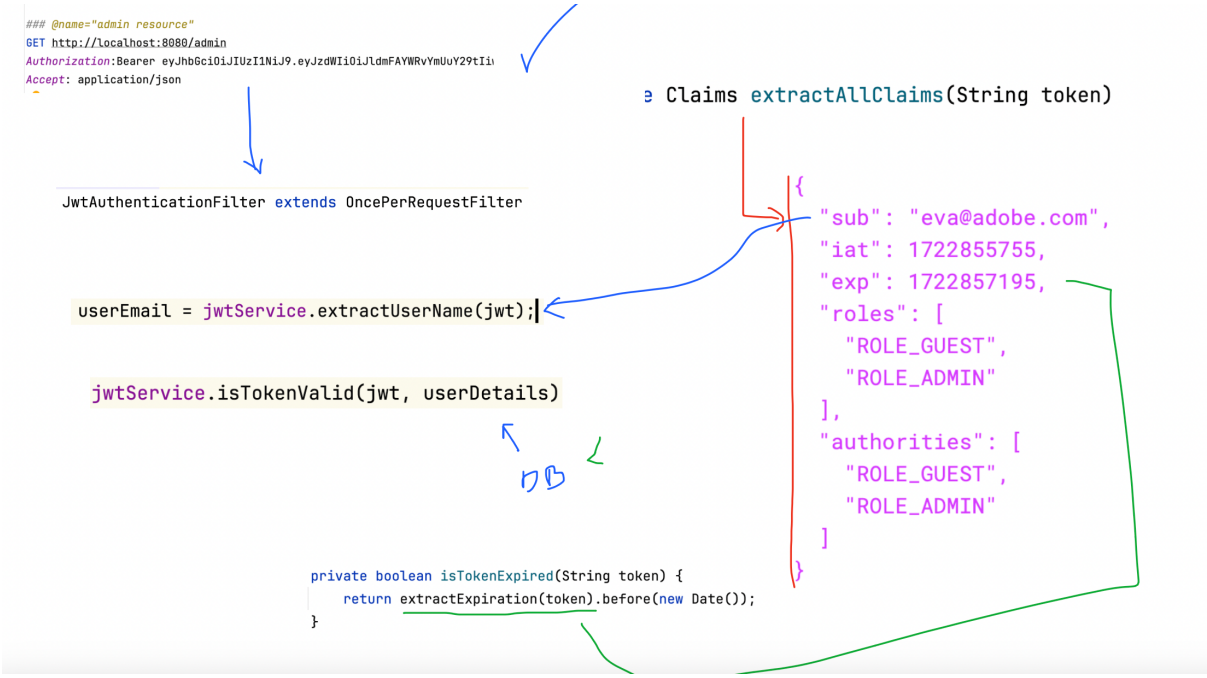
AuthenticationService

```
@Bean
AuthenticationProvider authenticationProvider() {
    DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
    authProvider.setUserDetailsService(userService.userDetailsService());
    authProvider.setPasswordEncoder(passwordEncoder());
    return authProvider;
}
```

```
// login
public JwtAuthenticationResponse signin(SigninRequest request) {
    authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(request.getEmail(), request.getPassword()));
    var user = userRepository.findByEmail(request.getEmail())
            .orElseThrow(() -> new IllegalArgumentException("Invalid email or password."));
    System.out.println(user);
    var jwt = jwtService.generateToken(user);
    return JwtAuthenticationResponse.builder().token(jwt).build();
}
```

```
public interface UserDao extends JpaRepository<User, Integer> {
    // Since email is unique, we'll find users by email
    Optional<User> findByEmail(String email);
}
```

```
private String generateToken(Map<String, Object> extraClaims, UserDetails userDetails) {
    Collection<String> authorities = userDetails.getAuthorities() Collection<capture of extends GrantedAuthority>
            .stream() Stream<capture of extends GrantedAuthority>
            .map(GrantedAuthority::getAuthority) Stream<String>
            .collect(Collectors.toList());
    User user = (User) userDetails;
    return Jwts.builder()
            .setClaims(extraClaims)
            .setSubject(user.getEmail())
            .setIssuedAt(new Date(System.currentTimeMillis()))
```


@name="admin resource"
GET http://localhost:8080/admin
Authorization:Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJldmFAYWRvYmUuY29tIi
Accept: application/json
JwtAuthenticationFilter extends OncePerRequestFilter
userEmail = jwtService.extractUserName(jwt);
jwtService.isTokenValid(jwt, userDetails)
e Claims extractAllClaims(String token)
{
"sub": "eva@adobe.com",
"iat": 1722855755,
"exp": 1722857195,
"roles": [
"ROLE_GUEST",
"ROLE_ADMIN"
],
"authorities": [
"ROLE_GUEST",
"ROLE_ADMIN"
]
}
private boolean isTokenExpired(String token) {
return extractExpiration(token).before(new Date());
}