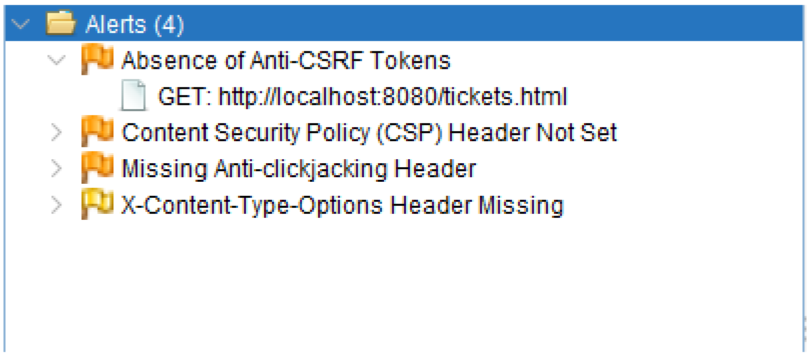
Secure Coding project

Valentin Hervé

Table of contents

Module 1: ZAP - Burp

ZAP alerts before fixing vulnerabilities



```
Alerts (4)
  Absence of Anti-CSRF Tokens
    GET: http://localhost:8080/tickets.html
  Content Security Policy (CSP) Header Not Set
  Missing Anti-clickjacking Header
  X-Content-Type-Options Header Missing
```

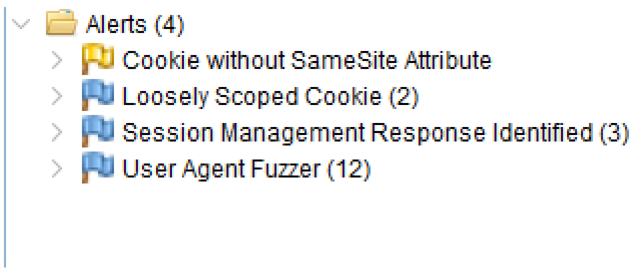
In order to fix all these vulnerabilities, I create a SecurityConfig class. The following method fix all these 3 vulnerabilities:

```
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
    http
            .csrf(csrf -> csrf
                    .ignoringRequestMatchers("/mvc/**", "/rest/**",
"/auth/api/v1/**")
            ) // Protect against anti csrf token
            .cors(AbstractHttpConfigurer::disable)
            .authorizeHttpRequests((authorize) -> authorize
                    .requestMatchers("/auth/api/v1/login").permitAll()

.requestMatchers("/auth/api/v1/refreshToken").permitAll()
                    .requestMatchers("/auth/api/v1/**").authenticated()
                    .requestMatchers("/mvc/**").authenticated()
                    .requestMatchers("/rest/**").authenticated())
            .sessionManagement((session) -> session

.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .authenticationProvider(authenticationProvider())
            .addFilterBefore(jwtAuthFilter,
UsernamePasswordAuthenticationFilter.class)
            .headers(headers ->
                    headers.xssProtection(
                            xss ->
xss.headerValue(XXssProtectionHeaderWriter.HeaderValue.ENABLED_MODE_BLOCK)
                    ).contentSecurityPolicy(
                            cps -> cps.policyDirectives("script-src 'self';
style-src 'self'; img-src 'self'; connect-src 'self'; frame-src 'self';
frame-ancestors 'self'; font-src 'self'; media-src 'self'; object-src
'self'; manifest-src 'self'; form-action 'self';")
                    )); // Set the Content Security Policy header

    return http.build();
}
```

- '.csrf' adds the anti-csrf tokens,

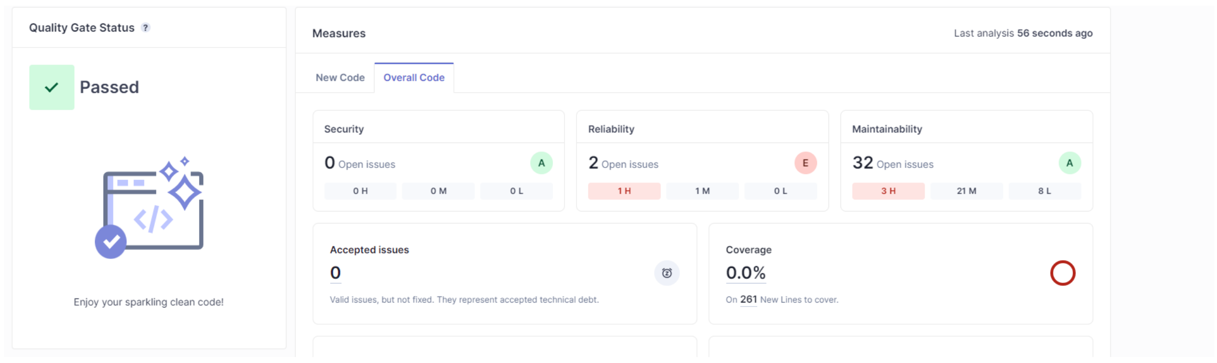
- ‘.headers.contentSecurityPolicy’ sets the Content Security Policy header protecting things like scripts to be loaded from outside of the site,
- ‘frame-ancestors 'self'’ protects against ‘ClickJacking’ attacks.

Here are the ZAP alerts after fixing these 3 vulnerabilities:



```
Alerts (4)
  Cookie without SameSite Attribute
  Loosely Scoped Cookie (2)
  Session Management Response Identified (3)
  User Agent Fuzzer (12)
```

Module 2: SonarQube



Quality Gate Status

Passed

Enjoy your sparkling clean code!

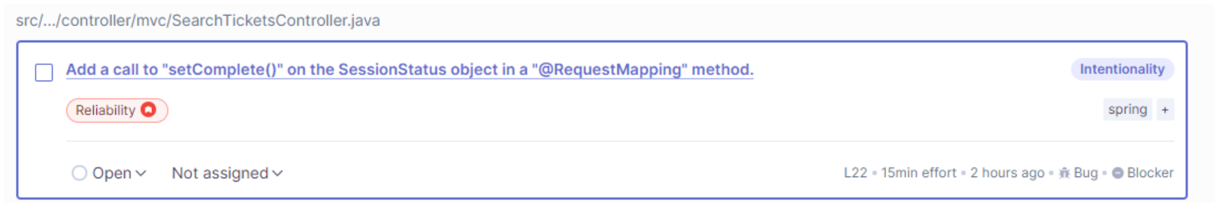
Measures — Last analysis 56 seconds ago

New Code | Overall Code

Security	Reliability	Maintainability
0 Open issues (A)	2 Open issues (E)	32 Open issues (A)
0 H, 0 M, 0 L	1 H, 1 M, 0 L	3 H, 21 M, 8 L

Accepted issues: 0 — Valid issues, but not fixed. They represent accepted technical debt.

Coverage: 0.0% — On 261 New Lines to cover.



src/.../controller/mvc/SearchTicketsController.java

Add a call to "setComplete()" on the SessionStatus object in a "@RequestMapping" method.

Intentionality

Reliability

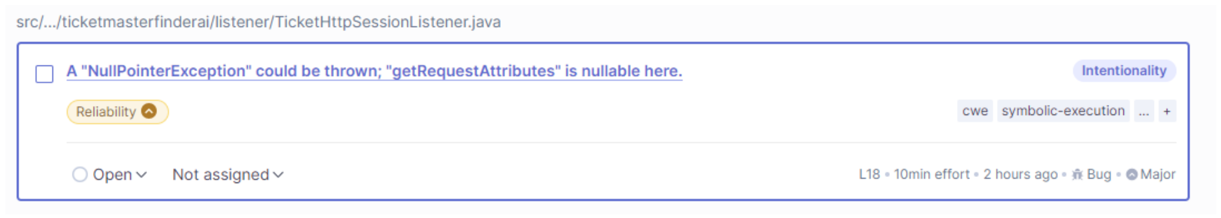
spring +

Open · Not assigned

L22 · 15min effort · 2 hours ago · Bug · Blocker

```
@PostMapping("/ticket")
public String filterTickets(Model model, TicketFormDTO ticketFormDTO,
SessionStatus sessionStatus) {
    List<TicketDTO> ticketDTOList =
ticketService.filterByCriteria(ticketFormDTO);
    model.addAttribute("tickets", ticketDTOList);
    sessionStatus.setComplete() // added
    return "redirect:/mvc/ticket";
}
```

I followed the instructions of SonarQube and added this ‘sessionStatus.setComplete’ line in the SearchTicketsController.java file.


src/.../ticketmasterfinderai/listener/TicketHttpSessionListener.java

A "NullPointerException" could be thrown; "getRequestAttributes" is nullable here.

Intentionality

Reliability

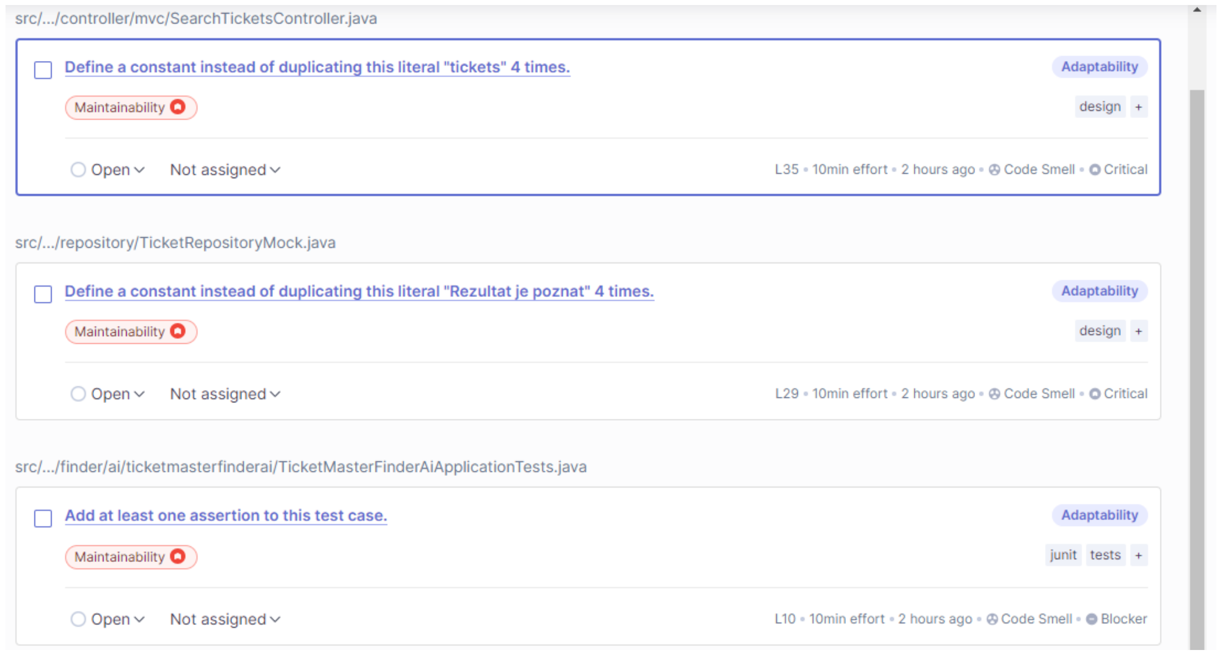
cwe symbolic-execution ... +

Open Not assigned

L18 • 10min effort • 2 hours ago • Bug • Major

```
@Override
public void sessionCreated(final HttpSessionEvent event) {

    try {
        HttpServletRequest request =
                ((ServletRequestAttributes)
Objects.requireNonNull(RequestContextHolder.getRequestAttributes())).getReq
uest();

        String ipAddress =  request.getRemoteAddr();

        System.out.println("Session created! IP address = " + ipAddress);
    }
    catch (NullPointerException e) {
        System.out.println("Session not created! " + e.getMessage());
    }
}
```

I added this ‘Objects.requireNonNull‘ method to protect against a nullable attribute.


src/.../controller/mvc/SearchTicketsController.java

Define a constant instead of duplicating this literal "tickets" 4 times.

Adaptability

Maintainability

design +

Open Not assigned

L35 • 10min effort • 2 hours ago • Code Smell • Critical

src/.../repository/TicketRepositoryMock.java

Define a constant instead of duplicating this literal "Rezultat je poznat" 4 times.

Adaptability

Maintainability

design +

Open Not assigned

L29 • 10min effort • 2 hours ago • Code Smell • Critical

src/.../finder/ai/ticketmasterfinderai/TicketMasterFinderAiApplicationTests.java

Add at least one assertion to this test case.

Adaptability

Maintainability

junit tests +

Open Not assigned

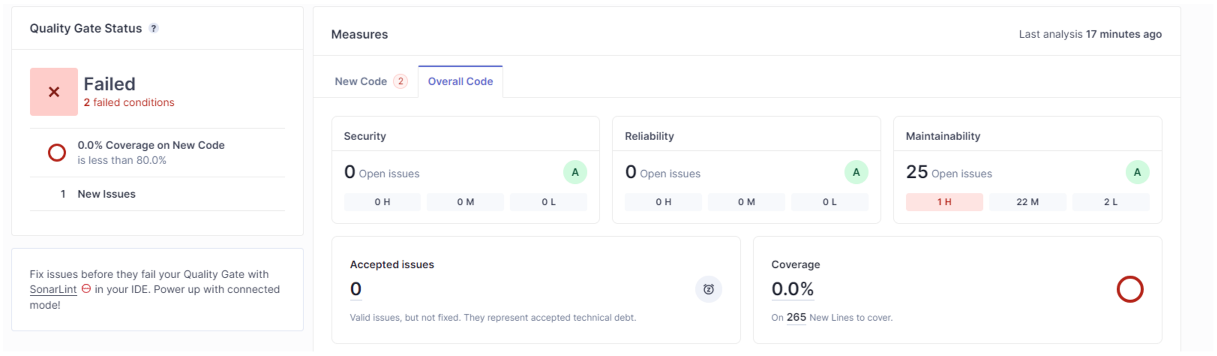
L10 • 10min effort • 2 hours ago • Code Smell • Blocker

```
private static final String TICKETS = "tickets";
```

```
private static final String TICKET_DESCRIPTION = "Rezultat je poznat";
```

I defined some constants where it was required where string were duplicated.

Here is the SonarQube analysis after fixing these problems.



Quality Gate Status

Failed
2 failed conditions

0.0% Coverage on New Code
is less than 80.0%

1 New Issues

Fix issues before they fail your Quality Gate with SonarLint in your IDE. Power up with connected mode!

Measures — Last analysis 17 minutes ago

New Code 2 | Overall Code

Security: 0 Open issues — A — 0 H, 0 M, 0 L

Reliability: 0 Open issues — A — 0 H, 0 M, 0 L

Maintainability: 25 Open issues — A — 1 H, 22 M, 2 L

Accepted issues: 0
Valid issues, but not fixed. They represent accepted technical debt.

Coverage: 0.0%
On 265 New Lines to cover.

MODULE 3: JWT Access and refresh token

For the access token, I added an authentication filter in my security configuration. I added these lines that tells that everybody can access the url '/auth/api/v1/login' and '/auth/api/v1/refreshToken', but you need to be authenticated to access the others.

```
.authorizeHttpRequests((authorize) -> authorize
        .requestMatchers("/auth/api/v1/login").permitAll()
        .requestMatchers("/auth/api/v1/refreshToken").permitAll()
        .requestMatchers("/auth/api/v1/**").authenticated()
        .requestMatchers("/mvc/**").authenticated()
        .requestMatchers("/rest/**").authenticated())
```

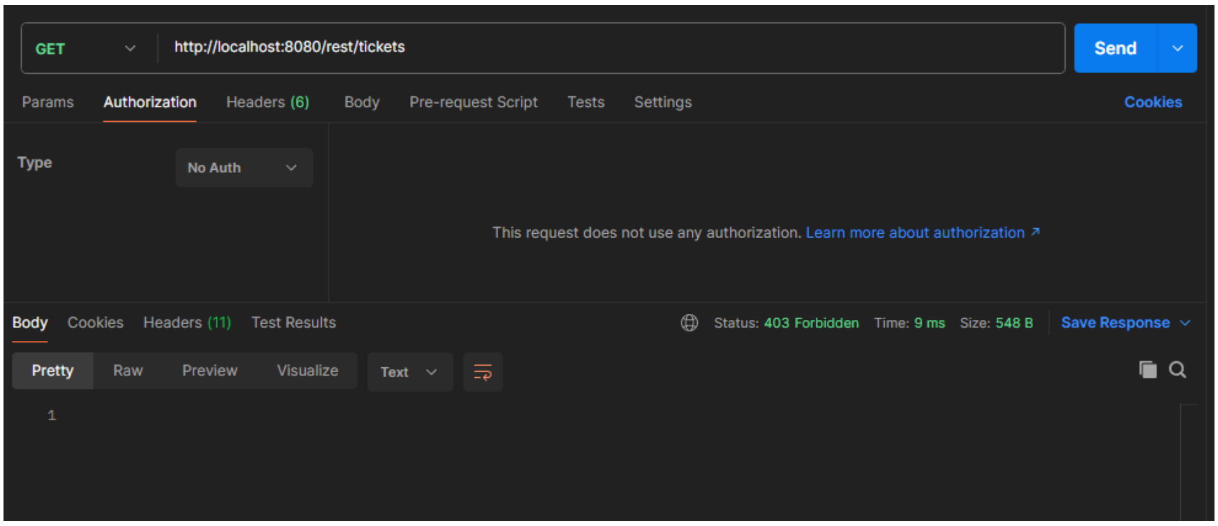
I created a table users, roles and user_roles to create user and assign them a role with a certain access level.

I finally created the AuthController file, which will allow users to login by giving and username and a password. If both match and are in the users database, then the server will send an access token, which will be required to access some data.

For the refresh token, I added a refresh token table in the database, and I added a /refreshToken POST endpoint in my AuthController file so that if we send a valid refresh token in the database it will send back the access token.

Here is an example of use:

First let's try accessing some data without being authenticated:



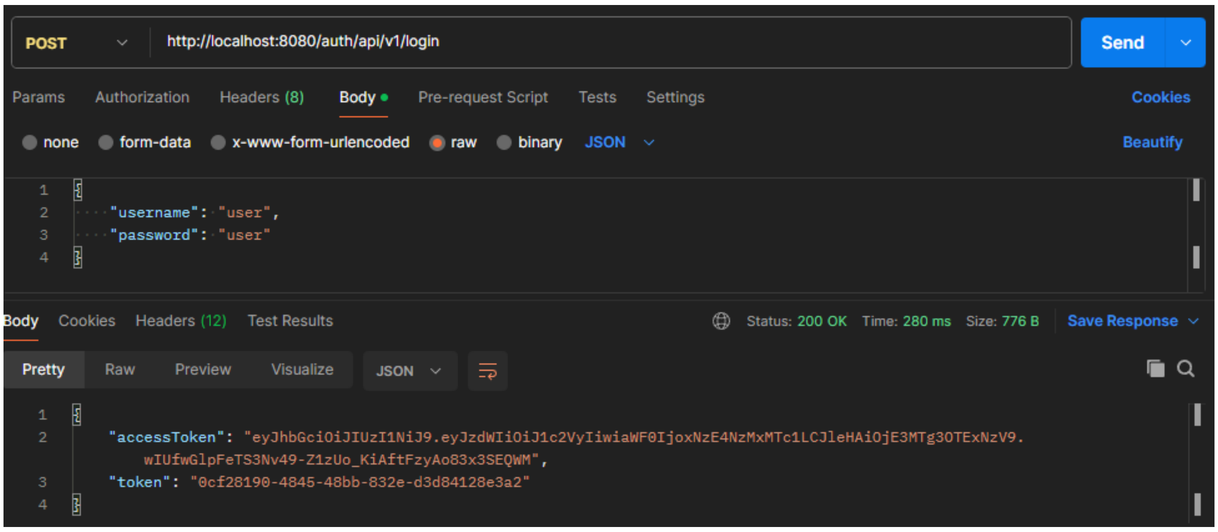
GET http://localhost:8080/rest/tickets

Authorization — Type: No Auth

This request does not use any authorization. Learn more about authorization ↗

Status: 403 Forbidden Time: 9 ms Size: 548 B

As you can see, we have the expected 403 error, forbidden access. Let’s then try to authenticate ourselves as a regular user:



POST http://localhost:8080/auth/api/v1/login

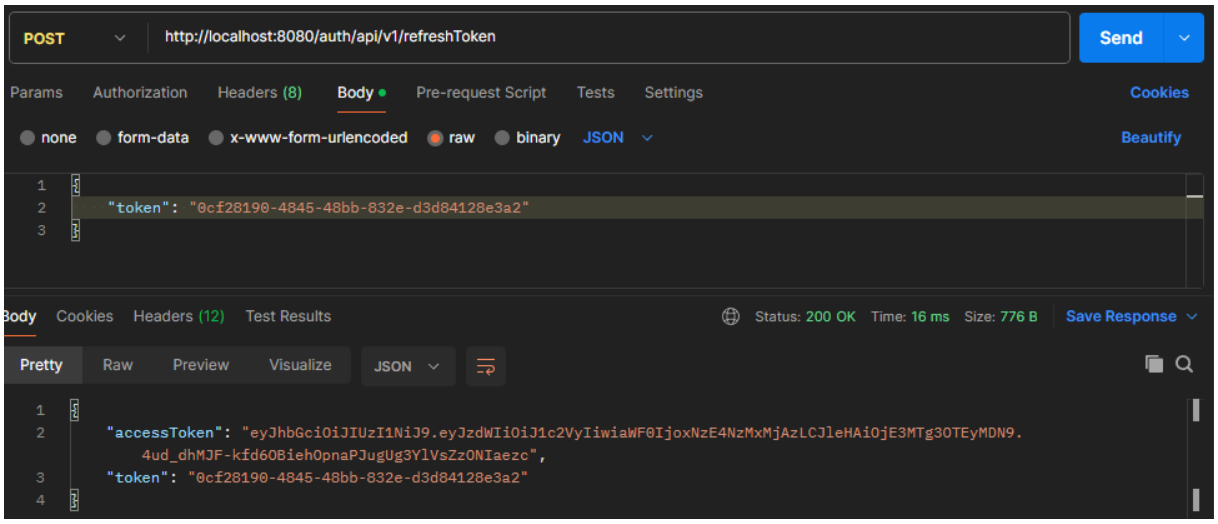
Body — raw — JSON

```
{
    "username": "user",
    "password": "user"
}
```

Status: 200 OK Time: 280 ms Size: 776 B

```
{
    "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzE4NzMxMTc1LCJleHAiOjE3MTg3OTExNzV9.
        wIUfwGlpFeTS3Nv49-Z1zUo_KiAftFzyAo83x3SEQWM",
    "token": "0cf28190-4845-48bb-832e-d3d84128e3a2"
}
```

As you can see, we now have an access token and a token. We can use this token to get our access token back.



POST http://localhost:8080/auth/api/v1/refreshToken

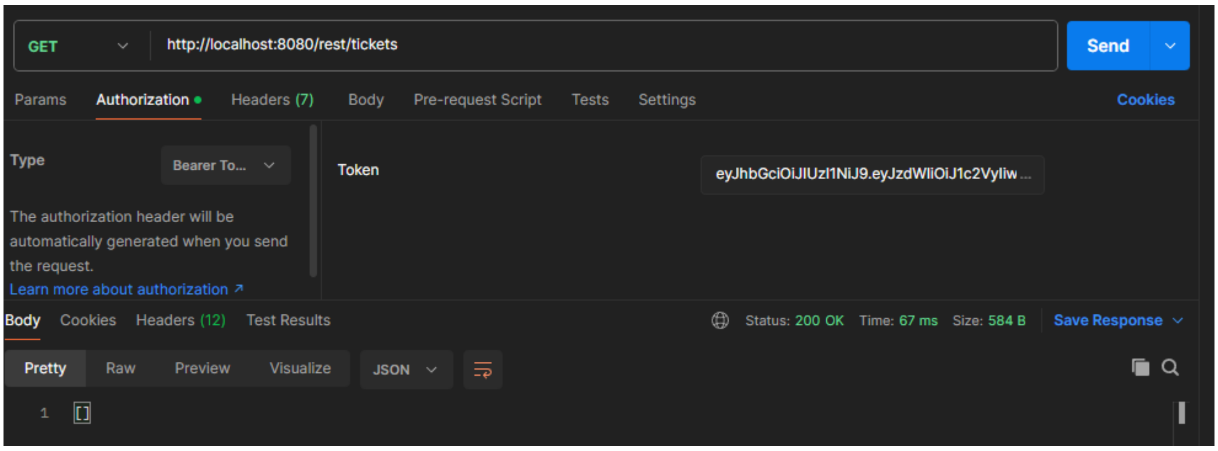
Body — raw — JSON

```
{
    "token": "0cf28190-4845-48bb-832e-d3d84128e3a2"
}
```

Status: 200 OK Time: 16 ms Size: 776 B

```
{
    "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzE4NzMxMjAzLCJleHAiOjE3MTg3OTEyMDN9.
        4ud_dhMJF-kfd6OBiehOpnaPJugUg3YlVsZzONIaezc",
    "token": "0cf28190-4845-48bb-832e-d3d84128e3a2"
}
```

We can use this access token to authenticate ourselves (as a regular user):


```
GET  http://localhost:8080/rest/tickets
```
Authorization — Type: Bearer Token. Token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiw...

The authorization header will be automatically generated when you send the request.

Status: 200 OK Time: 67 ms Size: 584 B

```
1  []
```

We now have access to these data.

MODULE 4: SQL injections

The application uses parameterized queries for database operations, which protects us against SQL injection. Parameterized queries ensure that the parameters are treated as data and not executable SQL code like 'WHERE 1=1'. For example:

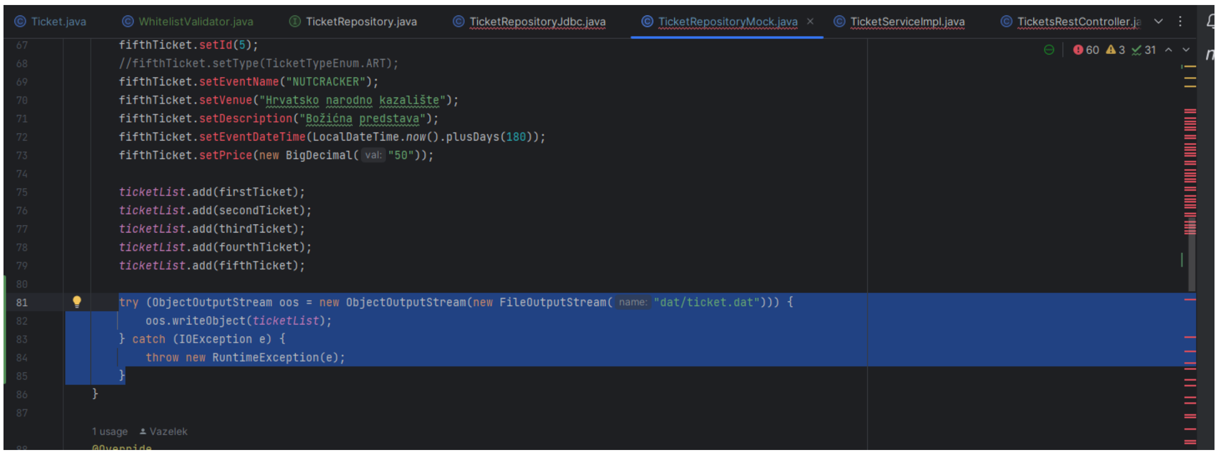
```
String sql = "SELECT * FROM TICKET WHERE ID = ?";
return Optional.ofNullable(
        jdbcTemplate.queryForObject(sql, new TicketRowMapper(), id));
```

Here, id is passed as a parameter to the query and not concatenated into the SQL string, which prevents SQL injection. On the contrary, something like this is very vulnerable, as there is no verifications of what param 'id' is. It could be something like '1 OR 1=1', which will then return all the tickets in the database:

```
String sql = "SELECT * FROM TICKET WHERE ID = " + id;
```

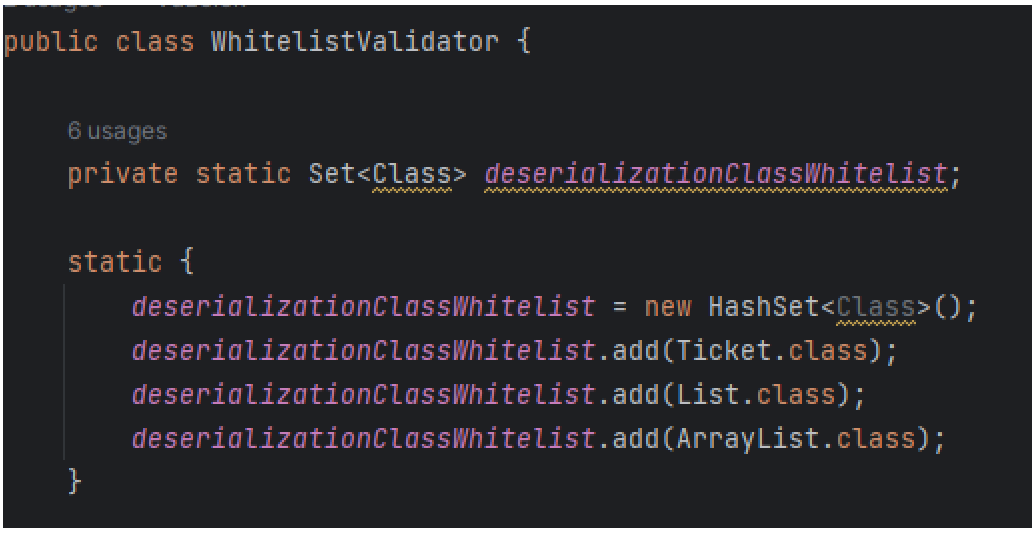
MODULE 5: Serialization

I implemented serialization as follows:


```
        fifthTicket.setId(5);
        //fifthTicket.setType(TicketTypeEnum.ART);
        fifthTicket.setEventName("NUTCRACKER");
        fifthTicket.setVenue("Hrvatsko narodno kazalište");
        fifthTicket.setDescription("Božićna predstava");
        fifthTicket.setEventDateTime(LocalDateTime.now().plusDays(180));
        fifthTicket.setPrice(new BigDecimal( val: "50"));

        ticketList.add(firstTicket);
        ticketList.add(secondTicket);
        ticketList.add(thirdTicket);
        ticketList.add(fourthTicket);
        ticketList.add(fifthTicket);

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream( name: "dat/ticket.dat"))) {
            oos.writeObject(ticketList);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    1 usage  Vazelek
    @Override
```

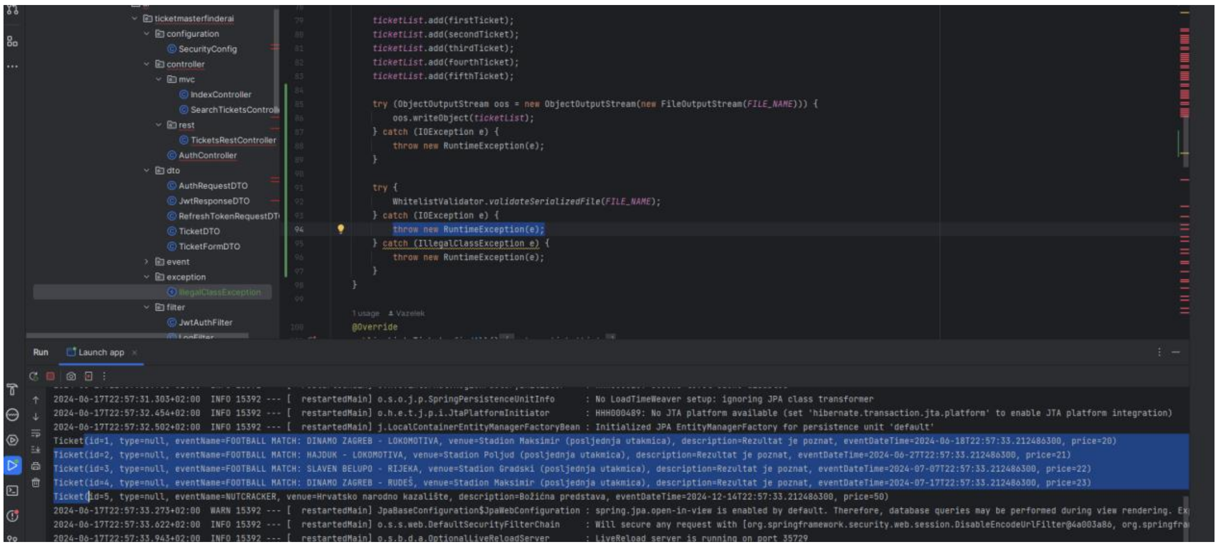
Here, we serialize the ticketList object in the ‘dat/ticket.dat’ file

For the deserialization, I created a WhiteListValidator, which will deserialize objects only if the object is in the whitelist, else it will throw an Illegal Class Exception:



```java
public class WhitelistValidator {

    6 usages
    private static Set<Class> deserializationClassWhitelist;

    static {
        deserializationClassWhitelist = new HashSet<Class>();
        deserializationClassWhitelist.add(Ticket.class);
        deserializationClassWhitelist.add(List.class);
        deserializationClassWhitelist.add(ArrayList.class);
    }
```

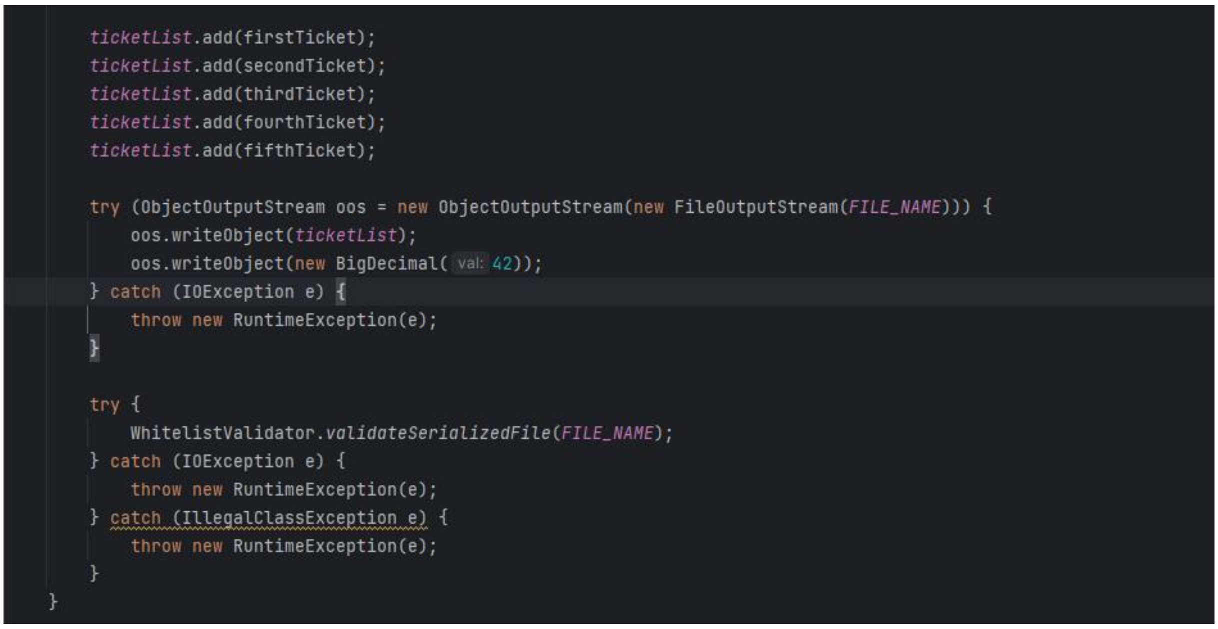
Only these 3 classes can be deserialized. The validateSerializedFile file will takes as input a binary file and then try to deserialize everything accordingly to the whitelist. If everything went well, you might see a result as follows, with the data that we just deserialized:



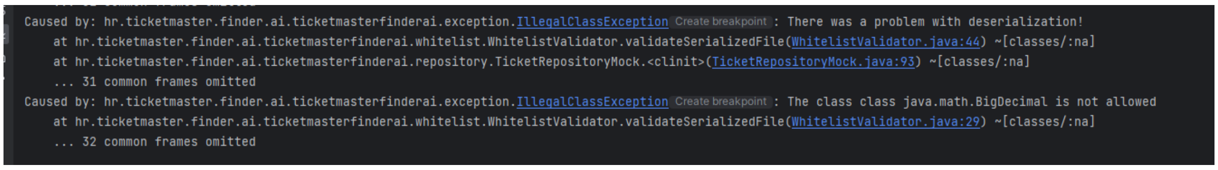
```java
        ticketList.add(firstTicket);
        ticketList.add(secondTicket);
        ticketList.add(thirdTicket);
        ticketList.add(fourthTicket);
        ticketList.add(fifthTicket);

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
            oos.writeObject(ticketList);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        try {
            WhitelistValidator.validateSerializedFile(FILE_NAME);
        } catch (IOException e) {
            throw new RuntimeException(e);
        } catch (IllegalClassException e) {
            throw new RuntimeException(e);
        }
    }

    1 usage   Vazelek
    @Override
```

```
2024-06-17T22:57:31.303+02:00  INFO 15392 --- [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class transformer
2024-06-17T22:57:32.454+02:00  INFO 15392 --- [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
2024-06-17T22:57:32.502+02:00  INFO 15392 --- [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
Ticket(id=1, type=null, eventName=FOOTBALL MATCH: DINAMO ZAGREB - LOKOMOTIVA, venue=Stadion Maksimir (posljednja utakmica), description=Rezultat je poznat, eventDateTime=2024-06-18T22:57:33.212486300, price=20)
Ticket(id=2, type=null, eventName=FOOTBALL MATCH: HAJDUK - LOKOMOTIVA, venue=Stadion Poljud (posljednja utakmica), description=Rezultat je poznat, eventDateTime=2024-06-27T22:57:33.212486300, price=21)
Ticket(id=3, type=null, eventName=FOOTBALL MATCH: SLAVEN BELUPO - RIJEKA, venue=Stadion Gradski (posljednja utakmica), description=Rezultat je poznat, eventDateTime=2024-07-07T22:57:33.212486300, price=22)
Ticket(id=4, type=null, eventName=FOOTBALL MATCH: DINAMO ZAGREB - RUDEŠ, venue=Stadion Maksimir (posljednja utakmica), description=Rezultat je poznat, eventDateTime=2024-07-17T22:57:33.212486300, price=23)
Ticket(id=5, type=null, eventName=NUTCRACKER, venue=Hrvatsko narodno kazalište, description=Božićna predstava, eventDateTime=2024-12-14T22:57:33.212486300, price=50)
2024-06-17T22:57:33.273+02:00  WARN 15392 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Ex
2024-06-17T22:57:33.622+02:00  INFO 15392 --- [  restartedMain] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request with [org.springframework.security.web.session.DisableEncodeUrlFilter@4a003a86, org.springfra
2024-06-17T22:57:33.943+02:00  INFO 15392 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
```

Else, an exception is thrown and the processus exits:



```java
        ticketList.add(firstTicket);
        ticketList.add(secondTicket);
        ticketList.add(thirdTicket);
        ticketList.add(fourthTicket);
        ticketList.add(fifthTicket);

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
            oos.writeObject(ticketList);
            oos.writeObject(new BigDecimal( val: 42));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        try {
            WhitelistValidator.validateSerializedFile(FILE_NAME);
        } catch (IOException e) {
            throw new RuntimeException(e);
        } catch (IllegalClassException e) {
            throw new RuntimeException(e);
        }
    }
```

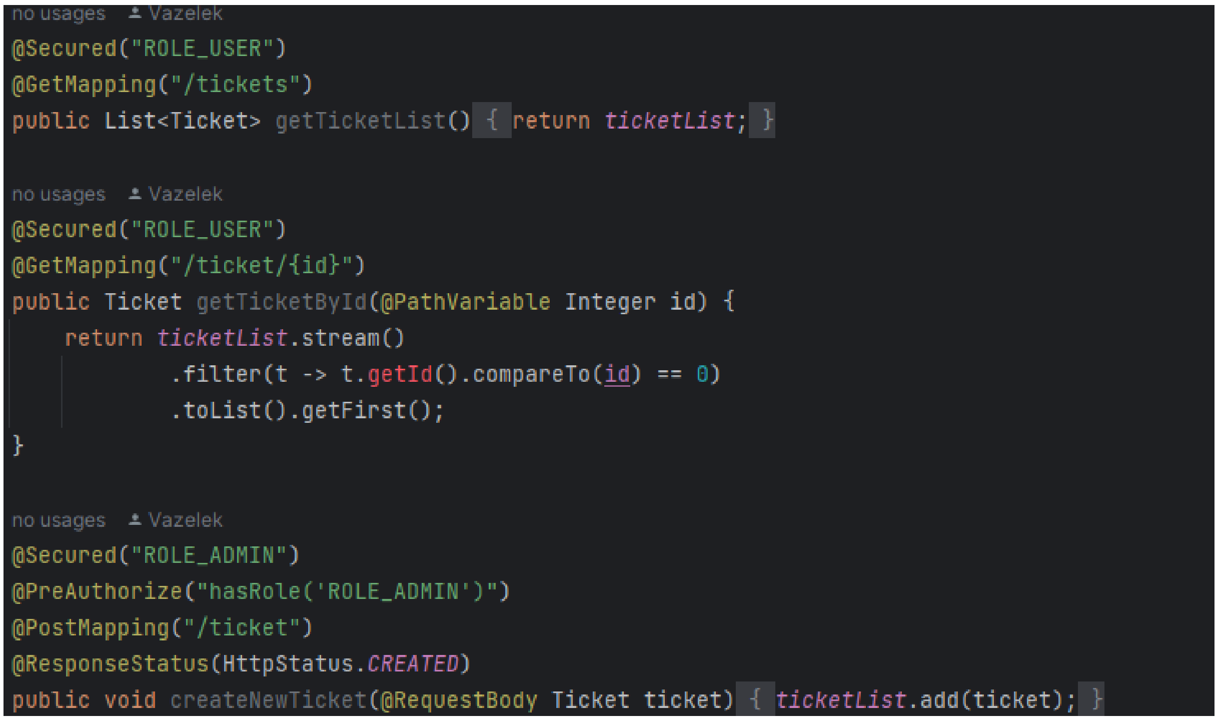


```
Caused by: hr.ticketmaster.finder.ai.ticketmasterfinderai.exception.IllegalClassException Create breakpoint : There was a problem with deserialization!
    at hr.ticketmaster.finder.ai.ticketmasterfinderai.whitelist.WhitelistValidator.validateSerializedFile(WhitelistValidator.java:44) ~[classes/:na]
    at hr.ticketmaster.finder.ai.ticketmasterfinderai.repository.TicketRepositoryMock.<clinit>(TicketRepositoryMock.java:93) ~[classes/:na]
    ... 31 common frames omitted
Caused by: hr.ticketmaster.finder.ai.ticketmasterfinderai.exception.IllegalClassException Create breakpoint : The class class java.math.BigDecimal is not allowed
    at hr.ticketmaster.finder.ai.ticketmasterfinderai.whitelist.WhitelistValidator.validateSerializedFile(WhitelistValidator.java:29) ~[classes/:na]
    ... 32 common frames omitted
```

MODULE 6: Authentication good practices

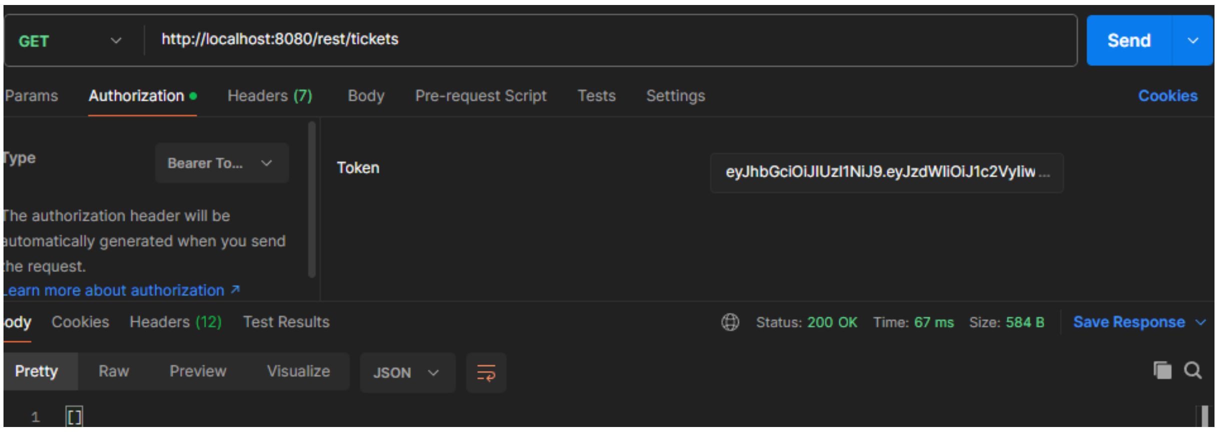
Protecting access to confidential data has already been done to some extent by allowing only certain requests from unauthenticated users in the security configuration.

The second thing that I have done is adding these ‘@Secured("ROLE_ADMIN")’ and ‘@PreAuthorize("hasRole('ROLE_ADMIN')")’ tags in my controller:



```
no usages  Vazelek
@Secured("ROLE_USER")
@GetMapping("/tickets")
public List<Ticket> getTicketList() { return ticketList; }

no usages  Vazelek
@Secured("ROLE_USER")
@GetMapping("/ticket/{id}")
public Ticket getTicketById(@PathVariable Integer id) {
    return ticketList.stream()
            .filter(t -> t.getId().compareTo(id) == 0)
            .toList().getFirst();
}

no usages  Vazelek
@Secured("ROLE_ADMIN")
@PreAuthorize("hasRole('ROLE_ADMIN')")
@PostMapping("/ticket")
@ResponseStatus(HttpStatus.CREATED)
public void createNewTicket(@RequestBody Ticket ticket) { ticketList.add(ticket); }
```

By doing it, I ensure that all users can access the get requests, but only admins can post. Let’s try it using postman. I still have my classic user token:



GET http://localhost:8080/rest/tickets

Type: Bearer To...

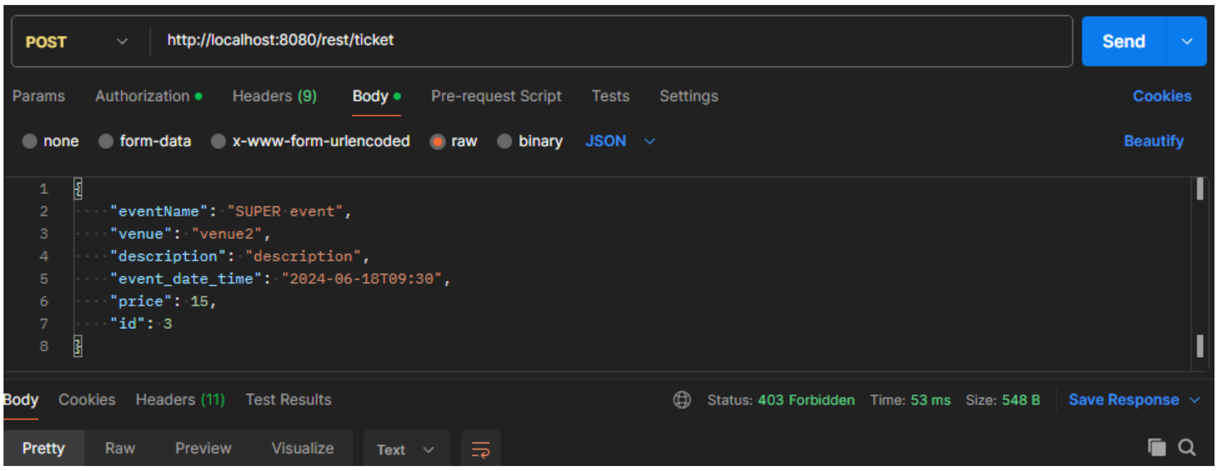
The authorization header will be automatically generated when you send the request.

Token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiw...

Status: 200 OK Time: 67 ms Size: 584 B

```
[]
```

As you can see, I can do the GET request and get a 200 response code, but if I try the POST request:

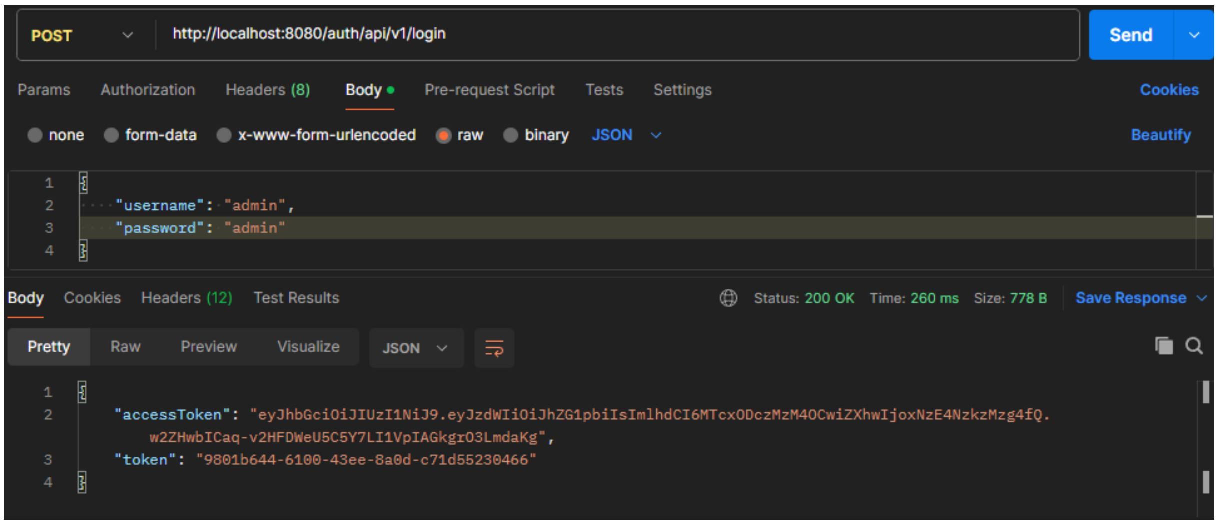


POST http://localhost:8080/rest/ticket

```
{
    "eventName": "SUPER event",
    "venue": "venue2",
    "description": "description",
    "event_date_time": "2024-06-18T09:30",
    "price": 15,
    "id": 3
}
```

Status: 403 Forbidden Time: 53 ms Size: 548 B

As a regular user, it is not working, but as an admin user:

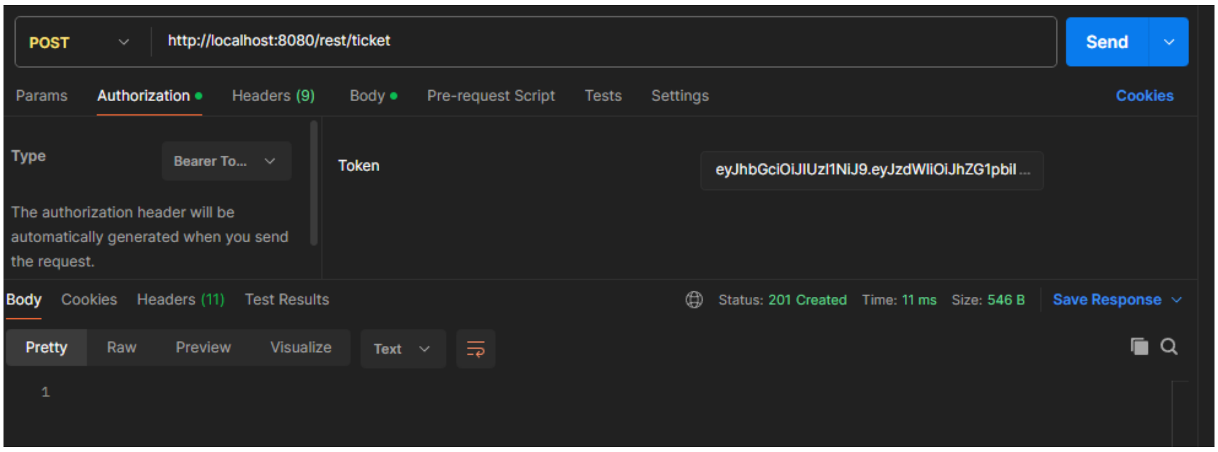


POST http://localhost:8080/auth/api/v1/login

```
{
    "username": "admin",
    "password": "admin"
}
```

Status: 200 OK Time: 260 ms Size: 778 B

```
{
    "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlhdCI6MTcxODczMzM4OCwiZXhwIjoxNzE4NzkzMzg4fQ.
        w2ZHwbICaq-v2HFDWeU5C5Y7LI1VpIAGkgrO3LmdaKg",
    "token": "9801b644-6100-43ee-8a0d-c71d55230466"
}
```



POST http://localhost:8080/rest/ticket

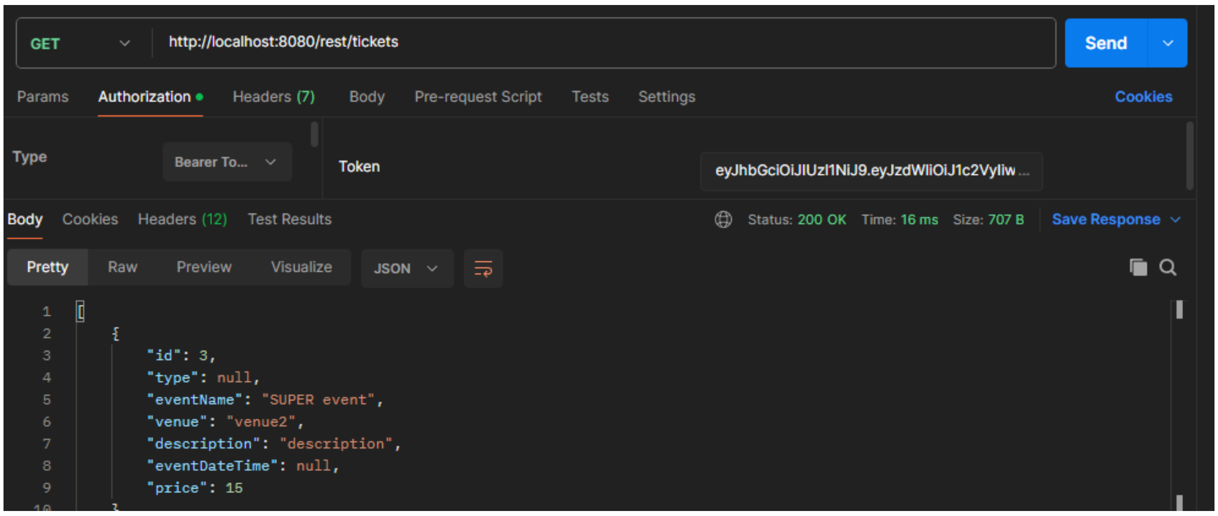
Authorization: Type Bearer To...

The authorization header will be automatically generated when you send the request.

Token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbil ...

Status: 201 Created Time: 11 ms Size: 546 B

I have the status 201 Created which means that the data have been sent, and I can get them (still as an admin):



GET http://localhost:8080/rest/tickets

Authorization: Type Bearer To...

Token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2Vyliw ...

Status: 200 OK Time: 16 ms Size: 707 B

```
[
    {
        "id": 3,
        "type": null,
        "eventName": "SUPER event",
        "venue": "venue2",
        "description": "description",
        "eventDateTime": null,
        "price": 15
    }
```