

Projektowanie bezpiecznych aplikacji

Konrad Leśkiewicz 46576

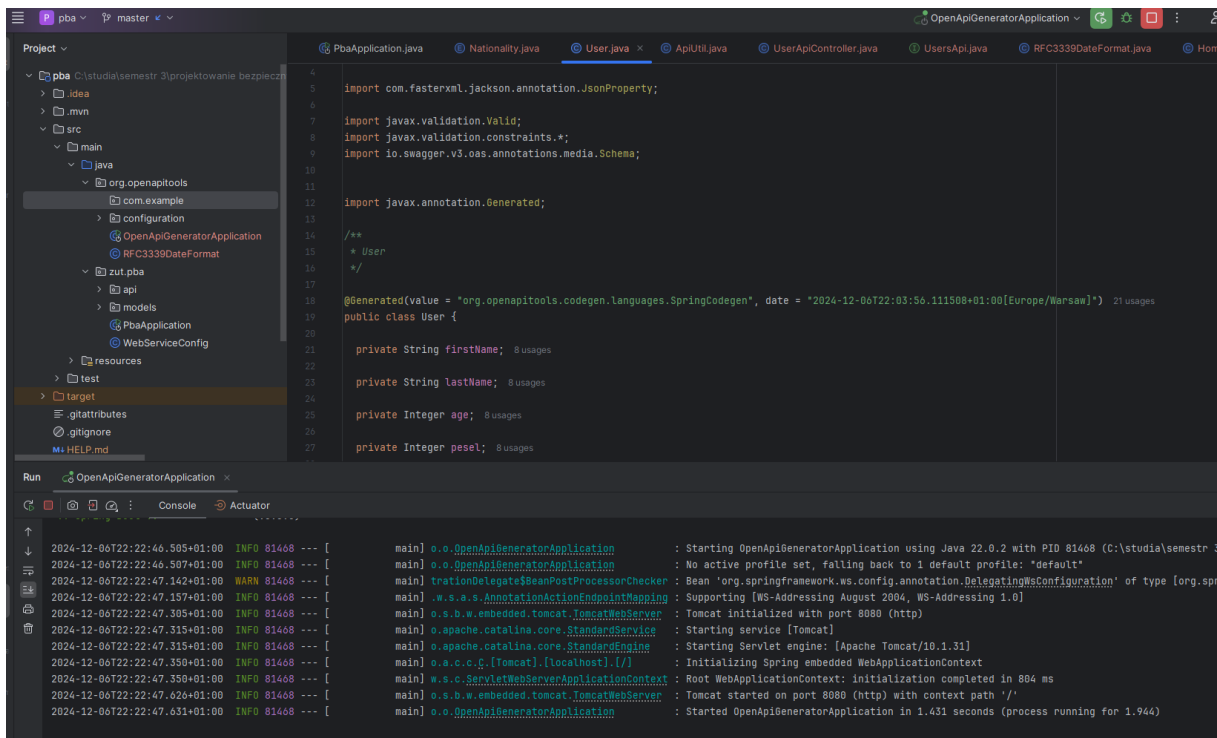
Implementacja – Lab04

Zadanie 1

Wygenerowanie potrzebnych plików na podstawie zawartości pliku api.yaml.

```
[error] Check the path of the OpenAPI spec and try again.
konrad@ASUS-KONRAD:~/pba$ java -jar openapi-generator-cli.jar generate -i api.yaml -g spring -o generated-spring -c config.json
[main] INFO o.o.codegen.DefaultGenerator - Generating with dryRun=false
[main] INFO o.o.c.ignore.CodegenIgnoreProcessor - Output directory (/home/konrad/pba/generated-spring) does not exist, or is inaccessible. No file (.openapi-generator-ignore) will be evaluated.
[main] INFO o.o.codegen.DefaultGenerator - OpenAPI Generator: spring (server)
[main] INFO o.o.codegen.DefaultGenerator - Generator 'spring' is considered stable.
[main] INFO o.o.codegen.languages.SpringCodegen - -----
[main] INFO o.o.c.languages.AbstractJavaCodegen - Environment variable JAVA_POST_PROCESS_FILE not defined so the Java code may not be properly formatted. To define it, try 'export JAVA_POST_PROCESS_FILE="/usr
```

Dodanie plików do projektu i zainstalowanie potrzebnych zależności. Po próbie zbudowania i uruchomienia, operacja zakończyła się sukcesem



Zadanie 2

Prosta implementacja UserApiController na podstawie interfejsu

```

34 @RequestMapping(@PathVariable("${openapi.projektowanieBezpiecznychAplikacjiLab03.base-path:}")
35 public class UserApiController implements UserApi {
36
37     private final NativeWebRequest request; 2 usages
38     private final Map<String, User> userDatabase = new HashMap<>(); // Prosta baza danych w pamięci 6 usages
39
40     @Autowired 1 Konrad Leśkiewicz
41     public UserApiController(NativeWebRequest request) { this.request = request; }
42
43
44
45     @Override 3 usages 1 Konrad Leśkiewicz
46     public Optional<NativeWebRequest> getRequest() { return Optional.ofNullable(request); }
47
48
49
50     @Override no usages new *
51     public ResponseEntity<User> createUser(String xRequestID, OffsetDateTime xDate, User user) {
52         if (user == null || user.getFirstName() == null) {
53             return ResponseEntity.badRequest().build(); // 400 Bad Request
54         }
55         userDatabase.put(user.getFirstName(), user);
56         return ResponseEntity.status(201).body(user); // 201 Created
57     }
58
59     @Override no usages new *
60     public ResponseEntity<Void> deleteUser(String username, String xRequestID, OffsetDateTime xDate) {
61         if (!userDatabase.containsKey(username)) {
62             return ResponseEntity.notFound().build(); // 404 Not Found
63         }
64         userDatabase.remove(username);
65         return ResponseEntity.noContent().build(); // 204 No Content
66     }
67
68     @Override no usages new *

```

Zadanie 3

Wykonanie polecenia POST

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/user
- Body Type:** raw
- Request Body (JSON):**

```

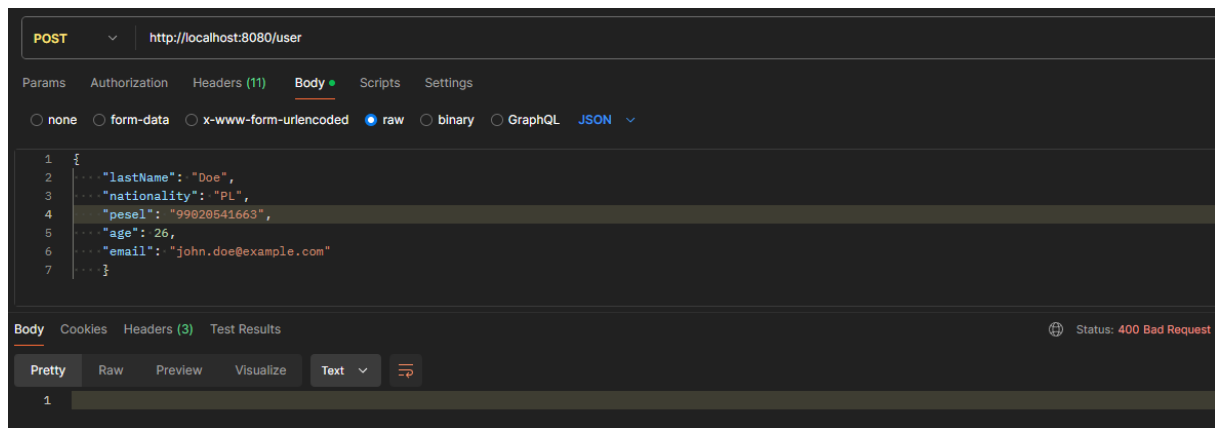
1 {
2   "firstName": "John",
3   "lastName": "Doe",
4   "nationality": "PL",
5   "pesel": "99020541663",
6   "age": 26,
7   "email": "john.doe@example.com"
8 }

```
- Response Status:** 201 Created
- Response Body (JSON):**

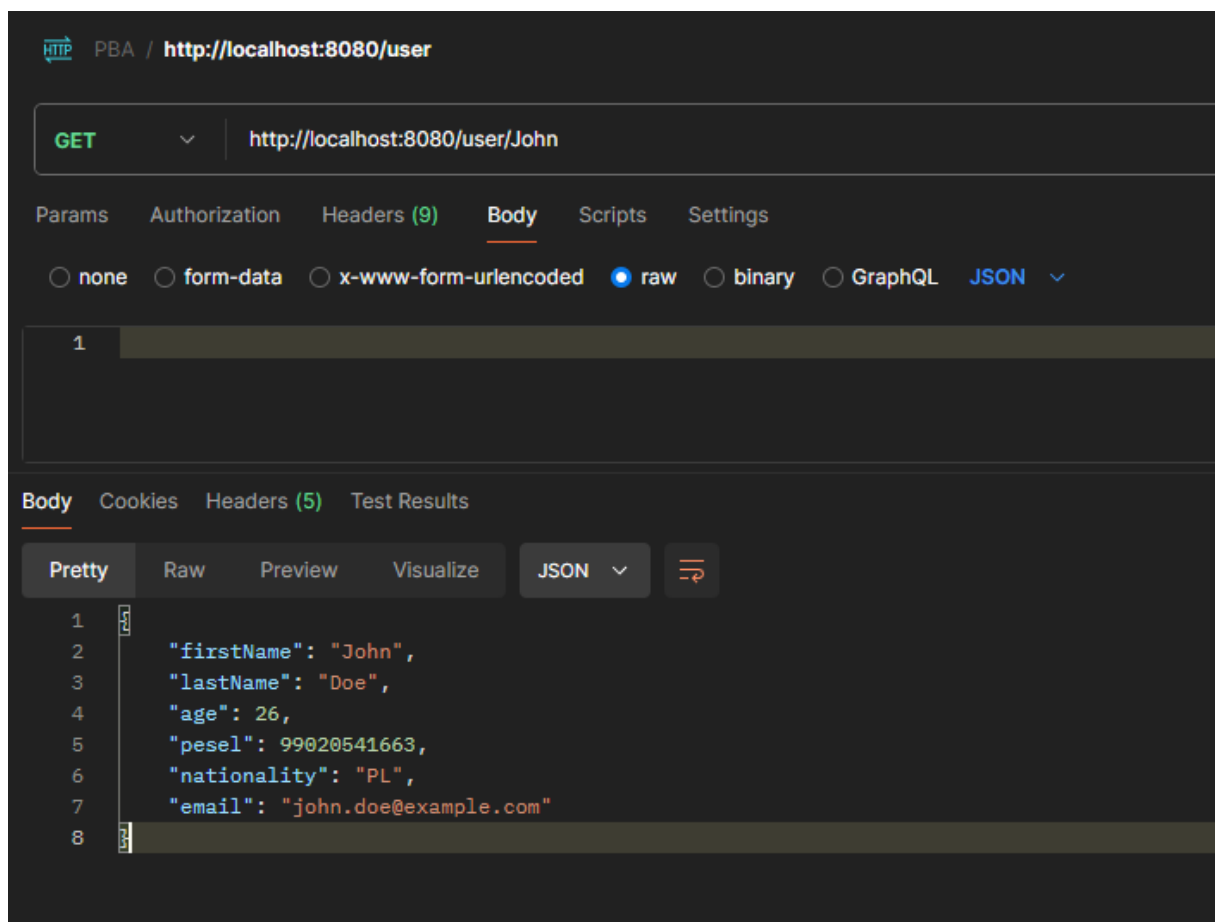
```

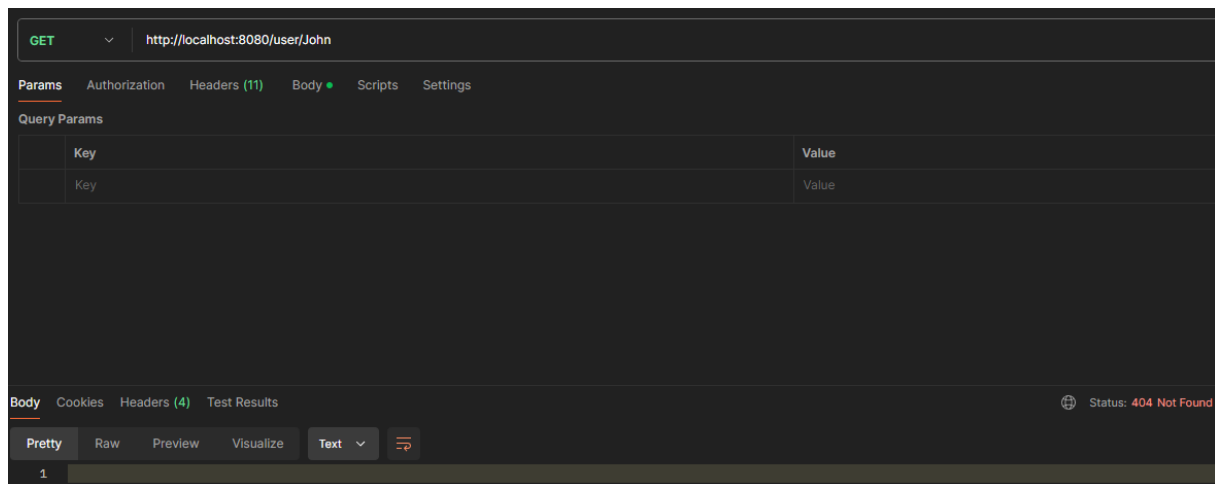
1 {
2   "firstName": "John",
3   "lastName": "Doe",
4   "age": 26,
5   "pesel": "99020541663",
6   "nationality": "PL",
7   "email": "john.doe@example.com"
8 }

```

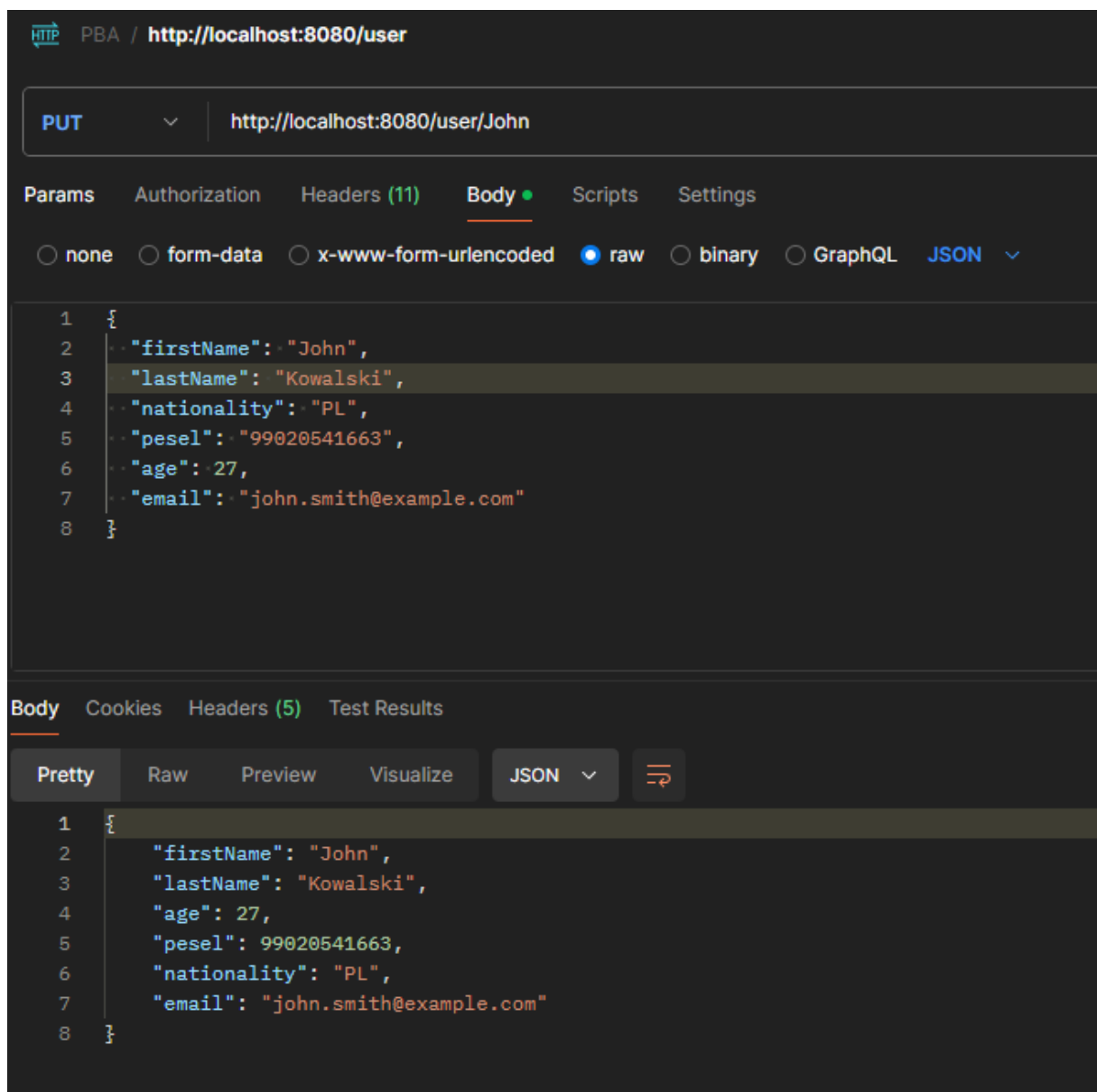


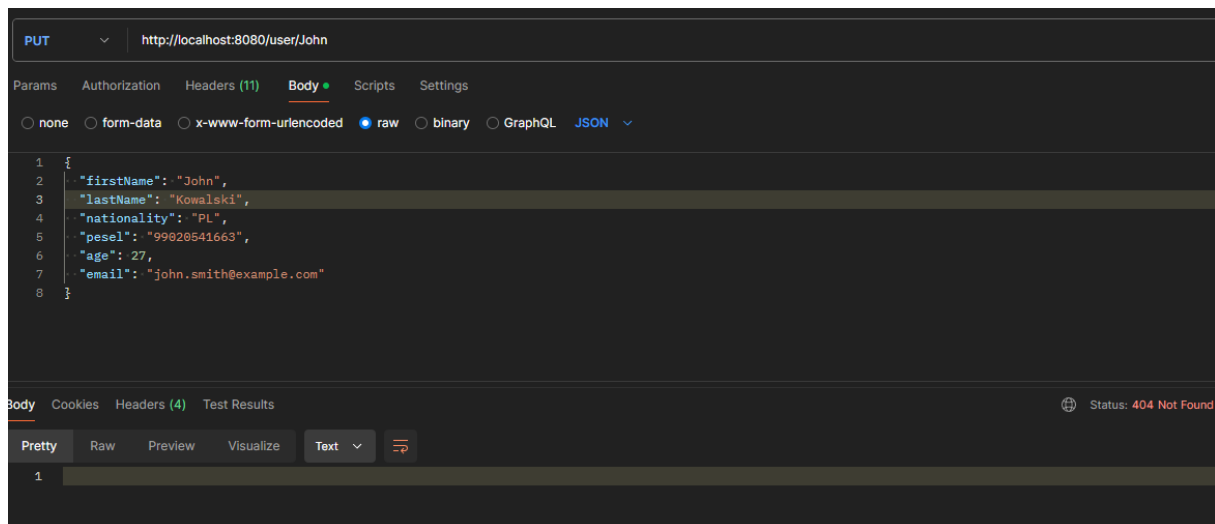
Wykonanie polecenia GET



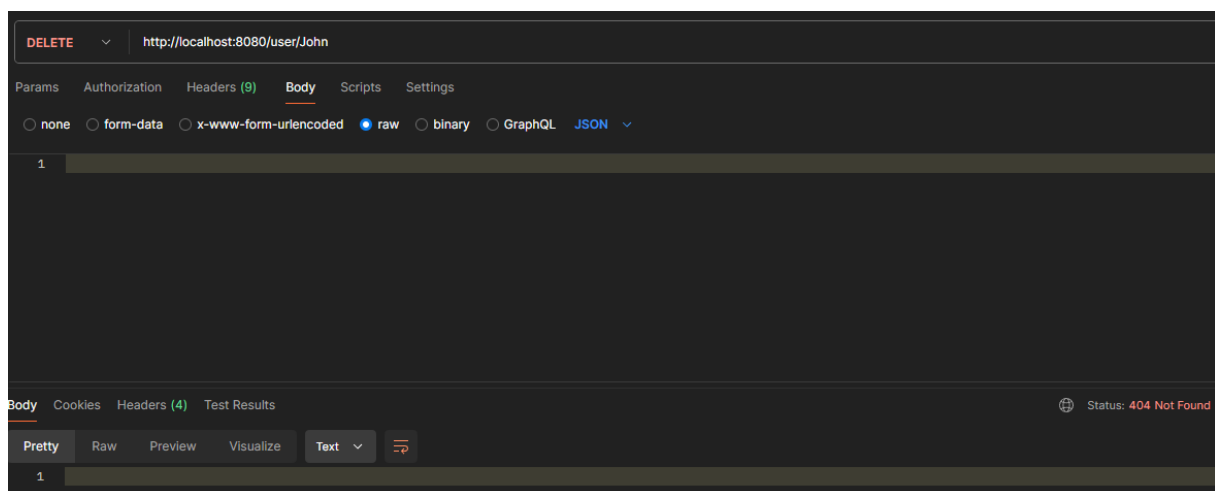
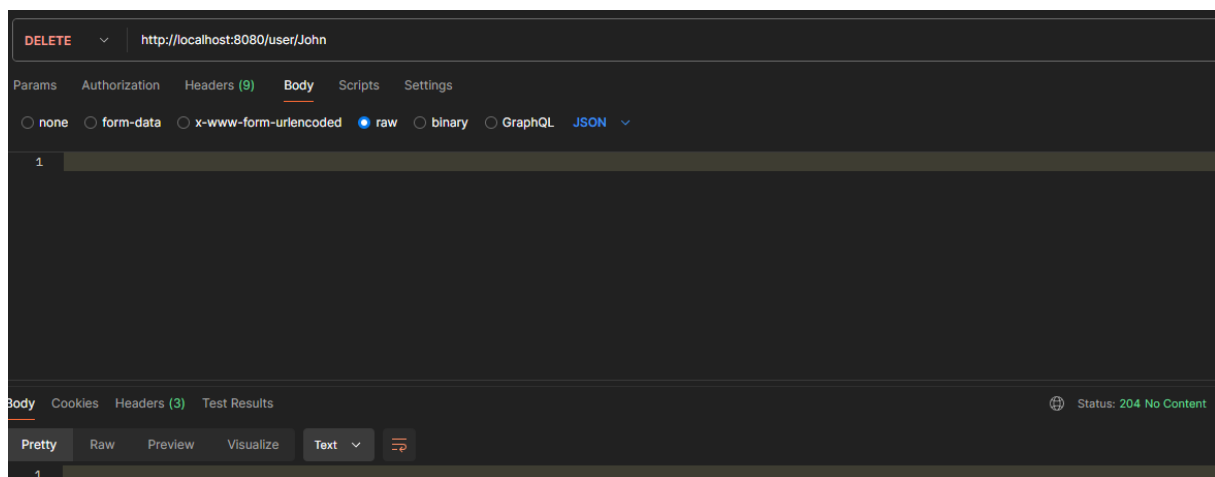


Wykonanie polecenia PUT





Wykonanie polecenia DELETE



Polecenia CURL w osobnym pliku oraz w repozytorium

<https://github.com/leskiewicz/projektowanie-bezpiecznych-aplikacji/tree/master/pba>

<https://github.com/leskiewicz/projektowanie-bezpiecznych-aplikacji/blob/master/pba/lab04-polecenia-curl.txt>