



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

---

Институт информационных технологий (ИИТ)

Кафедра практической и прикладной информатики(ППИ)

## **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Микросервисная архитектура»

### **Практическое задание № 8**

Студент группы *ИКБО-25-20, Валиков Кирилл*  
*Юрьевич*

---

подпись)

Преподаватель *Лантев И.А.*

---

подпись)

Отчет представлен \_\_\_\_\_ 202\_\_ г.

# 1 Тесты userService

```
1 package com.example.pr6_firstmicroservice.Controllers;
2
3 import com.example.pr6_firstmicroservice.DTO.AppealDTO;
4 import com.example.pr6_firstmicroservice.Models.User;
5 import com.example.pr6_firstmicroservice.Repositories.UserRepository;
6 import com.example.pr6_firstmicroservice.Services.UserServiceClient;
7 import org.junit.jupiter.api.Test;
8 import org.junit.jupiter.api.extension.ExtendWith;
9 import org.mockito.InjectMocks;
10 import org.mockito.Mock;
11 import org.mockito.Mockito;
12 import org.mockito.junit.jupiter.MockitoExtension;
13 import org.springframework.http.HttpStatus;
14 import org.springframework.http.MediaType;
15
16 import java.util.List;
17 import java.util.Optional;
18
19 import static org.junit.jupiter.api.Assertions.*;
20
21 @ExtendWith(MockitoExtension.class)
22 class UserControllerTest {
23
24     6 usages
25     @Mock
26     UserRepository userRepository;
27
28     1 usage
29     @Mock
30     UserServiceClient userServiceClient;
31
32     4 usages
33     @InjectMocks
34     UserController controller;
```

Рисунок 1. UserControllerTest (1)

```
35 @Test
36 void testCreateAppeal_UserServiceClient() {
37     //given
38     AppealDTO appealDTO = AppealDTO.builder().description("Нужна помощь!").user(User.builder().id(1L).build()).build();
39     //when
40     var responseEntity_controller = this.controller.CreateAppeal(appealDTO);
41     Mockito.verify(this.userServiceClient).CreateAppeal(appealDTO);
42     //then
43     assertNotNull(responseEntity_controller);
44     assertEquals(HttpStatus.OK, responseEntity_controller.getStatusCode());
45     assertEquals("expected: \"Всё прошло успешно!\", responseEntity_controller.getBody());
46 }
47
48
49 @Test
50 void testCreateUser() {
51     //given
52     User user = User.builder().id(1L).nickname("Windflaw").build();
53     Mockito.doReturn(user).when(this.userRepository).save(user);
54     //when
55     var responseEntity = this.controller.CreateUser(user);
56     Mockito.verify(this.userRepository).findByNickname(user.getNickname());
57     //then
58     assertNotNull(responseEntity);
59     assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
60     assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
61     assertEquals(user, responseEntity.getBody());
62 }
63
```

Рисунок 2. UserControllerTest (2)

```

65  @Test
66  void testGetUserById() {
67      //given
68      Optional<User> user = Optional.ofNullable(User.builder().id(1L).nickname("Windflaw").build());
69      Mockito.doReturn(user).when(this.userRepository).findById(1L);
70      //when
71      var responseEntity = this.controller.getUserById(1L);
72      Mockito.verify(this.userRepository, Mockito.times( wantedNumberOfInvocations: 2)).findById(1L);
73      //then
74      assertNotNull(responseEntity);
75      assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
76      assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
77      assertEquals(user, responseEntity.getBody());
78  }
79
80  @Test
81  void testGetAllUsers() {
82      //given
83      var users = List.of(User.builder().id(1L).nickname("Windflaw"), User.builder().id(2L).nickname("Valikoy Kirill"));
84      Mockito.doReturn(users).when(this.userRepository).findAll();
85      //when
86      var responseEntity = this.controller.getAllUsers();
87      Mockito.verify(this.userRepository).findAll();
88      //then
89      assertNotNull(responseEntity);
90      assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
91      assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
92      assertEquals(users, responseEntity.getBody());
93  }

```

Рисунок 3. UserControllerTest (3)

```

1  package com.example.pr6_firstmicroservice;
2
3  import org.junit.jupiter.api.Test;
4  import org.springframework.beans.factory.annotation.Autowired;
5  import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
6  import org.springframework.boot.test.context.SpringBootTest;
7  import org.springframework.http.MediaType;
8  import org.springframework.test.web.servlet.MockMvc;
9
10 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
11 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;
12 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
13
14 @SpringBootTest
15 @AutoConfigureMockMvc(printOnlyOnFailure = false)
16 class Pr6FirstMicroserviceApplicationTests {
17
18     2 usages
19     @Autowired
20     MockMvc mockMvc;
21
22

```

Рисунок 4. Pr6FirstMicroserviceApplicationTests (1)

```

23  @Test
24  void testGetAllUsers() throws Exception {
25      mockMvc.perform(get(uriTemplate: "/users/get_all"))
26          .andExpect(status().isOk())
27          .andExpect(content().contentType(MediaType.APPLICATION_JSON))
28          .andExpect(content().json(jsonContent: """
29              [
30                  {
31                      "id":1,
32                      "nickname":"Windflaw"
33                  },
34                  {
35                      "id":2,
36                      "nickname":"Valikov Kirill"
37                  },
38                  {
39                      "id":3,
40                      "nickname":"Denis Pak"
41                  }
42              ]
43          """));
44  }
45
46
47  @Test
48  void testGetUserById() throws Exception {
49      mockMvc.perform(get(uriTemplate: "/users/" + 1))
50          .andExpect(status().isOk())
51          .andExpect(content().contentType(MediaType.APPLICATION_JSON))
52          .andExpect(content().json(jsonContent: """
53              {
54                  "id":1,
55                  "nickname":"Windflaw"
56              }
57          """));
58  }

```

Рисунок 5. Pr6FirstMicroserviceApplicationTests (2)

```

11665 [INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 6.730 s -- in com.example.pr6_firstmicroservice.Pr6FirstMicroserviceApplicationTests
11672 [INFO]
11673 [INFO] Results:
11674 [INFO]
11675 [INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
11676 [INFO]
11677 [INFO] -----
11678 [INFO] BUILD SUCCESS
11679 [INFO] -----
11680 [INFO] Total time: 17.279 s
11681 [INFO] Finished at: 2024-04-27T10:24:44Z
11682 [INFO] -----

```

Рисунок 6. Выполненные тесты userService

## 2 Тест appealService

```
1 package com.example.pr6_secondmicroservice.Controllers;
2
3 import com.example.pr6_secondmicroservice.Models.Appeal;
4 import com.example.pr6_secondmicroservice.Models.Status;
5 import com.example.pr6_secondmicroservice.Models.UserDTO;
6 import com.example.pr6_secondmicroservice.Repositories.AppealRepository;
7 import com.example.pr6_secondmicroservice.Services.AppealServiceClient;
8 import org.junit.jupiter.api.Test;
9 import org.junit.jupiter.api.extension.ExtendWith;
10 import org.mockito.InjectMocks;
11 import org.mockito.Mock;
12 import org.mockito.Mockito;
13 import org.mockito.junit.jupiter.MockitoExtension;
14 import org.springframework.http.HttpStatus;
15 import org.springframework.http.MediaType;
16
17 import java.util.List;
18 import java.util.Optional;
19
20 import static org.junit.jupiter.api.Assertions.*;
21
22 @ExtendWith(MockitoExtension.class)
23 class AppealControllerTest {
24
25     8 usages
26     @Mock
27     AppealRepository appealRepository;
28
29     1 usage
30     @Mock
31     AppealServiceClient appealServiceClient;
32
33     5 usages
34     @InjectMocks
35     AppealController controller;
```

Рисунок 7. AppealControllerTest (1)

```
35 @Test
36 void testUpdateStatus() {
37     //given
38     Optional<Appeal> appeal = Optional.ofNullable(Appeal.builder().id(1L).description("Need help!")
39         .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
40         .status(Status.ACCEPTED_FOR_WORK).build());
41     Appeal appeal_for_db = Appeal.builder().id(1L).description("Need help!")
42         .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
43         .status(Status.DONE).build();
44     Mockito.doReturn(appeal).when(this.appealRepository).findById(1L);
45     //when
46     var responseEntity = this.controller.updateStatus(id: 1L, appeal_for_db);
47     Mockito.verify(this.appealRepository, Mockito.times(wantedNumberOfInvocations: 3)).findById(1L);
48     //then
49     assertNotNull(responseEntity);
50     assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
51     //assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
52     assertEquals(expected: "Статус заявки успешно обновлён", responseEntity.getBody());
53 }
54
55 @Test
56 void testCreateAppeal_AppealServiceClient() {
57     //given
58     Appeal appeal = Appeal.builder().description("Нужна помощь!").user(UserDTO.builder().id(1L).build()).build();
59     UserDTO userindb = UserDTO.builder().id(1L).build();
60     //when
61     var responseEntity = this.controller.createAppeal(appeal);
62     Mockito.verify(this.appealServiceClient).getUserById(userindb.getId());
63     //then
64     assertNotNull(responseEntity);
65     assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
66 }
67
68 }
```

Рисунок 8. AppealControllerTest (2)

```

69      @Test
70      void testGetAppealById() {
71          //given
72          Optional<Appeal> appeal = Optional.ofNullable(Appeal.builder().id(1L).description("Need help!")
73              .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
74              .status(Status.ACCEPTED_FOR_WORK).build());
75          Mockito.doReturn(appeal).when(this.appealRepository).findById(1L);
76          //when
77          var responseEntity = this.controller.GetAppealById(1L);
78          Mockito.verify(this.appealRepository, Mockito.times( wantedNumberOfInvocations: 2)).findById(1L);
79          //then
80          assertNotNull(responseEntity);
81          assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
82          assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
83          assertEquals(appeal, responseEntity.getBody());
84      }
85
86      @Test
87      void testGetAllAppeal() {
88          //given
89          var appeals = List.of(
90              Appeal.builder().id(1L).description("Need help!")
91                  .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
92                  .status(Status.ACCEPTED_FOR_WORK),
93              Appeal.builder().id(2L).description("Помогите, пожалуйста!")
94                  .user(UserDTO.builder().id(2L).nickname("Valikov Kirill").build())
95                  .status(Status.ACCEPTED_FOR_WORK));
96          Mockito.doReturn(appeals).when(this.appealRepository).findAll();
97          //when
98          var responseEntity = this.controller.GetAllAppeal();
99          Mockito.verify(this.appealRepository).findAll();
100          //then
101          assertNotNull(responseEntity);
102          assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
103          assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
104          assertEquals(appeals, responseEntity.getBody());
105      }
106  }
107

```

Рисунок 9. AppealControllerTest (3)

```

108      @Test
109      void testGetAllAppealsByUserId() {
110          //given
111          var all_appeals = List.of(
112              Appeal.builder().id(1L).description("Need help!")
113                  .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
114                  .status(Status.ACCEPTED_FOR_WORK),
115              Appeal.builder().id(2L).description("Помогите, пожалуйста!")
116                  .user(UserDTO.builder().id(2L).nickname("Valikov Kirill").build())
117                  .status(Status.ACCEPTED_FOR_WORK),
118              Appeal.builder().id(3L).description("Нужна срочная помощь!")
119                  .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
120                  .status(Status.ACCEPTED_FOR_WORK));
121          var first_user_appeals = List.of(
122              Appeal.builder().id(1L).description("Need help!")
123                  .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
124                  .status(Status.ACCEPTED_FOR_WORK),
125              Appeal.builder().id(3L).description("Нужна срочная помощь!")
126                  .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
127                  .status(Status.ACCEPTED_FOR_WORK));
128          Mockito.doReturn(first_user_appeals).when(this.appealRepository).findByUserId(1L);
129          //when
130          var responseEntity = this.controller.GetAllAppealsByUserId(1L);
131          Mockito.verify(this.appealRepository).findByUserId(1L);
132          //then
133          assertNotNull(responseEntity);
134          assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
135          assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
136          assertEquals(first_user_appeals, responseEntity.getBody());
137      }
138  }

```

Рисунок 10. AppealControllerTest (4)

```

1 package com.example.pr6_secondmicroservice;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
6 import org.springframework.boot.test.context.SpringBootTest;
7 import org.springframework.http.MediaType;
8 import org.springframework.test.web.servlet.MockMvc;
9
10 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
11 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.put;
12 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;
13 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
14
15 @SpringBootTest
16 @AutoConfigureMockMvc(printOnlyOnFailure = false)
17 class Pr6SecondMicroserviceApplicationTests {
18
19     3 usages
20     @Autowired
21     MockMvc mockMvc;

```

Рисунок 11. Pr6SecondMicroserviceApplicationTests (1)

```

22 @Test
23 void testGetAllAppeal() throws Exception {
24     mockMvc.perform(get("/appeals/get_all"))
25         .andExpect(status().isOk())
26         .andExpect(content().contentType(MediaType.APPLICATION_JSON))
27         .andExpect(content().json(jsonContent: """
28
29             [
30                 {
31                     "id": 1,
32                     "description": "Помогите пожалуйста, уменяю вас",
33                     "user": {
34                         "id": 1,
35                         "nickname": "Windflaw"
36                     },
37                     "status": "ACCEPTED_FOR_WORK"
38                 },
39                 {
40                     "id": 2,
41                     "description": "Need help!",
42                     "user": {
43                         "id": 2,
44                         "nickname": "Valikov Kirill"
45                     },
46                     "status": "ACCEPTED_FOR_WORK"
47                 }
48             ]
49 """));

```

Рисунок 12. Pr6SecondMicroserviceApplicationTests (2)

```

51     @Test
52     void testGetAppealById() throws Exception {
53         mockMvc.perform(get(uriTemplate: "/appeals/" + 1))
54             .andExpect(status().isOk())
55             .andExpect(content().contentType(MediaType.APPLICATION_JSON))
56             .andExpect(content().json(jsonContent: """
57                 {
58                     "id": 1,
59                     "description": "Помогите пазалуйста, умалаяю вас",
60                     "user": {
61                         "id": 1,
62                         "nickname": "Windflaw"
63                     },
64                     "status": "ACCEPTED_FOR_WORK"
65                 }
66             """));
67     }
68
69     @Test
70     void testUpdateStatus() throws Exception {
71         mockMvc.perform(put(uriTemplate: "/appeals/" + 1).contentType(MediaType.APPLICATION_JSON)
72             .content("""
73                 {
74                     "id": 1,
75                     "description": "Помогите пазалуйста, умалаяю вас",
76                     "user": {"id": 1, "nickname": "Windflaw"},
77                     "status": "ACCEPTED_FOR_WORK"
78                 }
79             """))
80             .andExpect(status().isOk())
81             .andExpect(content().string(expectedContent: "Статус заявки успешно обновлён"));
82     }

```

Рисунок 13. Pr6SecondMicroserviceApplicationTests (3)

```

198 [INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 8.692 s -- in com.example.pr6_secondmicroservice.Pr6SecondMicroserviceApplicationTests
206 [INFO]
207 [INFO] Results:
208 [INFO]
209 [INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
210 [INFO]
211 [INFO] -----
212 [INFO] BUILD SUCCESS
213 [INFO] -----
214 [INFO] Total time: 14.669 s
215 [INFO] Finished at: 2024-04-27T10:25:00Z
216 [INFO] -----

```

Рисунок 14. Выполненные тесты appealService