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 *Лаптев И.А.*

_____ подпись)

Отчет представлен _____202__г.

1 Тесты userService

```java
package com.example.pró_firstmicroservice.Controllers;

import com.example.pró_firstmicroservice.DTO.AppealDTO;
import com.example.pró_firstmicroservice.Models.User;
import com.example.pró_firstmicroservice.Repositories.UserRepository;
import com.example.pró_firstmicroservice.Services.UserServiceClient;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;

import java.util.List;
import java.util.Optional;

import static org.junit.jupiter.api.Assertions.*;

@ExtendWith(MockitoExtension.class)
class UserControllerTest {


    6 usages
    @Mock
    UserRepository userRepository;

    1 usage
    @Mock
    UserServiceClient userServiceClient;

    4 usages
    @InjectMocks
    UserController controller;


```

Рисунок 1. UserControllerTest (1)

```java
    @Test
    void testCreateAppeal_UserServiceClient() {
        //given
        AppealDTO appealDTO = AppealDTO.builder().description("Нужна помощь!").user(User.builder().id(1L).build()).build();
        //when
        var responseEntity_controller = this.controller.CreateAppeal(appealDTO);
        Mockito.verify(this.userServiceClient).CreateAppeal(appealDTO);
        //then
        assertNotNull(responseEntity_controller);
        assertEquals(HttpStatus.OK, responseEntity_controller.getStatusCode());
        assertEquals( expected: "Всё прошло успешно!", responseEntity_controller.getBody());
    }


    @Test
    void testCreateUser() {
        //given
        User user = User.builder().id(1L).nickname("Windflaw").build();
        Mockito.doReturn(user).when(this.userRepository).save(user);
        //when
        var responseEntity = this.controller.CreateUser(user);
        Mockito.verify(this.userRepository).findByNickname(user.getNickname());
        //then
        assertNotNull(responseEntity);
        assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
        assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
        assertEquals(user, responseEntity.getBody());
    }

```

Рисунок 2. UserControllerTest (2)

```java
    @Test
    void testGetUserById() {
        //given
        Optional<User> user = Optional.ofNullable(User.builder().id(1L).nickname("Windflaw").build());
        Mockito.doReturn(user).when(this.userRepository).findById(1L);
        //when
        var responseEntity = this.controller.GetUserById(1L);
        Mockito.verify(this.userRepository, Mockito.times( wantedNumberOfInvocations: 2)).findById(1L);
        //then
        assertNotNull(responseEntity);
        assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
        assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
        assertEquals(user, responseEntity.getBody());
    }

    @Test
    void testGetAllUsers() {
        //given
        var users = List.of(User.builder().id(1L).nickname("Windflaw"), User.builder().id(2L).nickname("Valikov Kirill"));
        Mockito.doReturn(users).when(this.userRepository).findAll();
        //when
        var responseEntity = this.controller.GetAllUsers();
        Mockito.verify(this.userRepository).findAll();
        //then
        assertNotNull(responseEntity);
        assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
        assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
        assertEquals(users, responseEntity.getBody());
    }
}
```

Рисунок 3. UserControllerTest (3)

```java
package com.example.pr6_firstmicroservice;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.test.web.servlet.MockMvc;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

@SpringBootTest
@AutoConfigureMockMvc(printOnlyOnFailure = false)
class Pr6FirstMicroserviceApplicationTests {

    2 usages
    @Autowired
    MockMvc mockMvc;
```

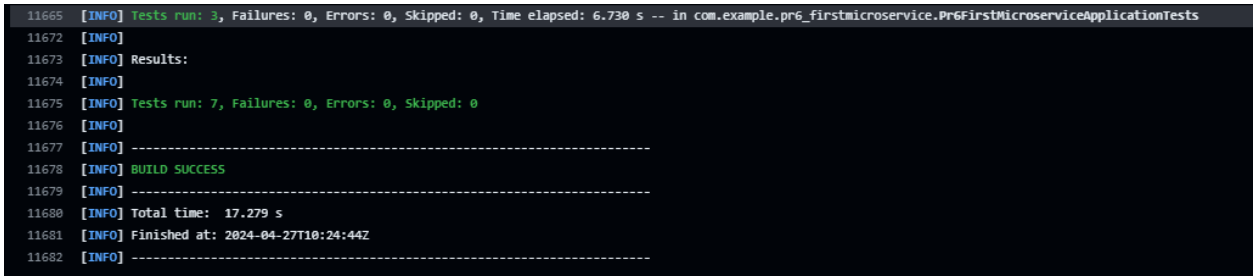
Рисунок 4. Pr6FirstMicroserviceApplicationTests (1)

```
@Test
void testGetAllUsers() throws Exception {
    mockMvc.perform(get( urlTemplate: "/users/get_all"))
            .andExpect(status().isOk())
            .andExpect(content().contentType(MediaType.APPLICATION_JSON))
            .andExpect(content().json( jsonContent: """
                    [
                        {
                            "id":1,
                            "nickname":"Windflaw"
                        },
                        {
                            "id":2,
                            "nickname":"Valikov Kirill"
                        },
                        {
                            "id":3,
                            "nickname":"Denis Pak"
                        }

                    ]
                    """));
}

@Test
void testGetUserById() throws Exception {
    mockMvc.perform(get( urlTemplate: "/users/" + 1))
            .andExpect(status().isOk())
            .andExpect(content().contentType(MediaType.APPLICATION_JSON))
            .andExpect(content().json( jsonContent: """
                    {
                        "id":1,
                        "nickname":"Windflaw"
                    }
                    """));
}
```

Рисунок 5. Pr6FirstMicroserviceApplicationTests (2)



```
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 6.730 s -- in com.example.pr6_firstmicroservice.Pr6FirstMicroserviceApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  17.279 s
[INFO] Finished at: 2024-04-27T10:24:44Z
[INFO] ------------------------------------------------------------------------
```

Рисунок 6. Выполненные тесты userService

2 Тест appealService

```java
package com.example.pr6_secondmicroservice.Controllers;

import com.example.pr6_secondmicroservice.Models.Appeal;
import com.example.pr6_secondmicroservice.Models.Status;
import com.example.pr6_secondmicroservice.Models.UserDTO;
import com.example.pr6_secondmicroservice.Repositories.AppealRepository;
import com.example.pr6_secondmicroservice.Services.AppealServiceClient;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;

import java.util.List;
import java.util.Optional;

import static org.junit.jupiter.api.Assertions.*;

@ExtendWith(MockitoExtension.class)
class AppealControllerTest {

    8 usages
    @Mock
    AppealRepository appealRepository;

    1 usage
    @Mock
    AppealServiceClient appealServiceClient;

    5 usages
    @InjectMocks
    AppealController controller;


```

Рисунок 7. AppealControllerTest (1)

```java
    @Test
    void testUpdateStatus() {
        //given
        Optional<Appeal> appeal = Optional.ofNullable(Appeal.builder().id(1L).description("Need help!")
                .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
                .status(Status.ACCEPTED_FOR_WORK).build());
        Appeal appeal_for_db = Appeal.builder().id(1L).description("Need help!")
                .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
                .status(Status.DONE).build();
        Mockito.doReturn(appeal).when(this.appealRepository).findById(1L);
        //when
        var responseEntity = this.controller.UpdateStatus( id: 1L, appeal_for_db);
        Mockito.verify(this.appealRepository, Mockito.times( wantedNumberOfInvocations: 3)).findById(1L);
        //then
        assertNotNull(responseEntity);
        assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
        //assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
        assertEquals( expected: "Статус заявки успешно обновлён", responseEntity.getBody());
    }

    @Test
    void testCreateAppeal_AppealServiceClient() {
        //given
        Appeal appeal = Appeal.builder().description("Нужна помощь!").user(UserDTO.builder().id(1L).build()).build();
        UserDTO userindb = UserDTO.builder().id(1L).build();
        //when
        var responseEntity = this.controller.CreateAppeal(appeal);
        Mockito.verify(this.appealServiceClient).GetUserById(userindb.getId());
        //then
        assertNotNull(responseEntity);
        assertEquals(HttpStatus.OK, responseEntity.getStatusCode());

    }

```

Рисунок 8. AppealControllerTest (2)

```
    @Test
    void testGetAppealById() {
        //given
        Optional<Appeal> appeal = Optional.ofNullable(Appeal.builder().id(1L).description("Need help!")
                .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
                .status(Status.ACCEPTED_FOR_WORK).build());
        Mockito.doReturn(appeal).when(this.appealRepository).findById(1L);
        //when
        var responseEntity = this.controller.GetAppealById(1L);
        Mockito.verify(this.appealRepository, Mockito.times( wantedNumberOfInvocations: 2)).findById(1L);
        //then
        assertNotNull(responseEntity);
        assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
        assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
        assertEquals(appeal, responseEntity.getBody());
    }

    @Test
    void testGetAllAppeal() {
        //given
        var appeals = List.of(
                Appeal.builder().id(1L).description("Need help!")
                        .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
                        .status(Status.ACCEPTED_FOR_WORK),
                Appeal.builder().id(2L).description("Помогите, пожалуйста!")
                        .user(UserDTO.builder().id(2L).nickname("Valikov Kirill").build())
                        .status(Status.ACCEPTED_FOR_WORK));
        Mockito.doReturn(appeals).when(this.appealRepository).findAll();
        //when
        var responseEntity = this.controller.GetAllAppeal();
        Mockito.verify(this.appealRepository).findAll();
        //then
        assertNotNull(responseEntity);
        assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
        assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
        assertEquals(appeals, responseEntity.getBody());

    }

```

Рисунок 9. AppealControllerTest (3)

```
    @Test
    void testGetAllAppealsByUserId() {
        //given
        var all_appeals = List.of(
                Appeal.builder().id(1L).description("Need help!")
                        .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
                        .status(Status.ACCEPTED_FOR_WORK),
                Appeal.builder().id(2L).description("Помогите, пожалуйста!")
                        .user(UserDTO.builder().id(2L).nickname("Valikov Kirill").build())
                        .status(Status.ACCEPTED_FOR_WORK),
                Appeal.builder().id(3L).description("Нужна срочная помощь!")
                        .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
                        .status(Status.ACCEPTED_FOR_WORK));
        var first_user_appeals = List.of(
                Appeal.builder().id(1L).description("Need help!")
                        .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
                        .status(Status.ACCEPTED_FOR_WORK),
                Appeal.builder().id(3L).description("Нужна срочная помощь!")
                        .user(UserDTO.builder().id(1L).nickname("Windflaw").build())
                        .status(Status.ACCEPTED_FOR_WORK));
        Mockito.doReturn(first_user_appeals).when(this.appealRepository).findByUserId(1L);
        //when
        var responseEntity = this.controller.GetAllAppealsByUserId(1L);
        Mockito.verify(this.appealRepository).findByUserId(1L);
        //then
        assertNotNull(responseEntity);
        assertEquals(HttpStatus.OK, responseEntity.getStatusCode());
        assertEquals(MediaType.APPLICATION_JSON, responseEntity.getHeaders().getContentType());
        assertEquals(first_user_appeals, responseEntity.getBody());
    }
}
```

Рисунок 10. AppealControllerTest (4)

```java
package com.example.pr6_secondmicroservice;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.test.web.servlet.MockMvc;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.put;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

@SpringBootTest
@AutoConfigureMockMvc(printOnlyOnFailure = false)
class Pr6SecondMicroserviceApplicationTests {

    3 usages
    @Autowired
    MockMvc mockMvc;

```

Рисунок 11. Pr6SecondMicroserviceApplicationTests (1)

```java
    @Test
    void testGetAllAppeal() throws Exception {
        mockMvc.perform(get( urlTemplate: "/appeals/get_all"))
                .andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(content().json( jsonContent: """
                        [
                            {
                                "id": 1,
                                "description": "Памагите пазалуйста, умалаяю вас",
                                "user": {
                                    "id": 1,
                                    "nickname": "Windflaw"
                                },
                                "status": "ACCEPTED_FOR_WORK"
                            },
                            {
                                "id": 2,
                                "description": "Need help!",
                                "user": {
                                    "id": 2,
                                    "nickname": "Valikov Kirill"
                                },
                                "status": "ACCEPTED_FOR_WORK"
                            }
                        ]
                        """));
    }
```

Рисунок 12. Pr6SecondMicroserviceApplicationTests (2)

```java
@Test
void testGetAppealById() throws Exception {
    mockMvc.perform(get( urlTemplate: "/appeals/" + 1))
            .andExpect(status().isOk())
            .andExpect(content().contentType(MediaType.APPLICATION_JSON))
            .andExpect(content().json( jsonContent: """
                    {
                        "id": 1,
                        "description": "Памагите пазалуйста, умалаяю вас",
                        "user": {
                            "id": 1,
                            "nickname": "Windflaw"
                        },
                        "status": "ACCEPTED_FOR_WORK"
                    }
                    """));
}

@Test
void testUpdateStatus() throws Exception {
    mockMvc.perform(put( urlTemplate: "/appeals/" + 1).contentType(MediaType.APPLICATION_JSON)
                    .content("""
                    {
                        "id": 1,
                        "description": "Памагите пазалуйста, умалаяю вас",
                        "user": {"id": 1, "nickname": "Windflaw"},
                        "status": "ACCEPTED_FOR_WORK"
                    }
                    """))
            .andExpect(status().isOk())
            .andExpect(content().string( expectedContent: "Статус заявки успешно обновлён"));
}
```

Рисунок 13. Pr6SecondMicroserviceApplicationTests (3)

```
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 8.692 s -- in com.example.pr6_secondmicroservice.Pr6SecondMicroserviceApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  14.669 s
[INFO] Finished at: 2024-04-27T10:25:00Z
[INFO] ------------------------------------------------------------------------
```

Рисунок 14. Выполненные тесты appealService