|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №2**

по дисциплине «Тестирование и верификация программного обеспечения»

|  |  |
| --- | --- |
| **Выполнили:**  Студенты группыИКБО-01-19 | Субботин Е.В.  Кузин Д.О.  Залетин Н.А.  Леутина Т.А. |
| **Проверил:** | ассистент Трушин С. М. |

2021 г.

**Содержание**

[Состав команды 3](#_Toc84324093)

[1 Описание ПО 4](#_Toc84324094)

[1.1 Документация своего проекта 4](#_Toc84324095)

[1.2 Документация проекта партнёра (Леутина Таисия) 4](#_Toc84324096)

[1.3 Выбор ПО для тестирования 4](#_Toc84324097)

[2 Модульное тестирование 5](#_Toc84324098)

[2.1 Выбор классов для тестирования 5](#_Toc84324100)

[2.2 Тестирование класса CommentService 5](#_Toc84324101)

[2.3 Тестирование класса PostServiceImpl 7](#_Toc84324102)

[2.4 Тестирование класса UserService 11](#_Toc84324103)

[3 Оценка качества чужого кода, соответствия документации и его надежности 14](#_Toc84324104)

Состав команды

* Субботин Евгений
* Кузин Данил
* Залетин Никита
* Леутина Таисия

# Описание ПО

## Документация своего проекта

Ссылка на ЯндексДиск с документацией:

https://disk.yandex.ru/d/g73OkwhQzzEuVA

Ссылка на гит-репозиторий с проектом: https://github.com/Shambonik/java\_kr

## Документация проекта партнёра (Леутина Таисия)

Ссылка на гит-репозиторий с проектом:

https://github.com/Evgeniy638/catlog

Ссылка на облако с документацией:

https://drive.google.com/drive/folders/1cSNr86fbitNDk2k0\_Slt9Ro3RHEeQq6c?usp=sharing

## Выбор ПО для тестирования

В качестве ПО для тестирования был выбран JUNIT 5, для создания «объектов-заглушек» использовался Mockito, также применялись библиотеки проверки утверждений, такие как AssertJ. Для генерации документации использовался Javadoc.

# Модульное тестирование



## Выбор классов для тестирования

Для тестирования были выбраны классы CommentService, PostServiceImpl и UserService, представленные на Рисунке 2.1.

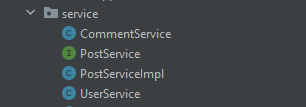


Рисунок 2.1 – Классы для тестирования

## Тестирование класса CommentService

Исходный код класса CommentServiceTest.java (Листинг 2.1)

Листинг 2.1 - Файл CommentServiceTest.java

@SpringBootTest  
public class CommentServiceTest {  
 @InjectMocks  
 CommentService commentService;  
  
 @Mock  
 PostDAO postDAO;  
  
 @Mock  
 UserDAO userDAO;  
  
 @Mock  
 CommentDAO commentDAO;  
  
 @Test  
 public void getCommentsByPostId() {  
 int postId = 1;  
  
 Comment comment1 = new Comment();  
 comment1.setId(1);  
 comment1.setHasAnswers(true);  
 comment1.setPostId(postId);  
  
 Comment comment2 = new Comment();  
 comment2.setId(2);  
 comment2.setPostId(postId);  
 comment2.setHeadCommentId(1);  
  
 Comment comment3 = new Comment();  
 comment3.setId(3);  
 comment3.setPostId(postId);  
 comment3.setHeadCommentId(1);  
  
 Mockito  
 .*when*(commentDAO.getCommentsByPostId(comment1.getPostId()))  
 .thenReturn(List.*of*(comment1));  
  
 Mockito  
 .*when*(commentDAO.getAnswersByCommentId(comment1.getPostId()))  
 .thenReturn(List.*of*(comment2, comment3));  
  
 List<HeadCommentDTO> result = commentService.getCommentsByPostId(comment1.getPostId());  
  
 Assertions.*assertEquals*(1, result.size());  
 Assertions.*assertEquals*(comment1.getId(), result.get(0).getId());  
   
 List<RepliesDTO> replies = result.get(0).getReplies();  
   
 Assertions.*assertEquals*(2, replies.size());  
 Assertions.*assertEquals*(comment2.getId(), replies.get(0).getId());  
 Assertions.*assertEquals*(comment3.getId(), replies.get(1).getId());  
 }  
  
 @Test  
 public void getAnswersByCommentId() {  
 int postId = 1;  
 int headId = 0;  
  
 Comment comment1 = new Comment();  
 comment1.setId(1);  
 comment1.setPostId(postId);  
 comment1.setHeadCommentId(headId);  
  
 Comment comment2 = new Comment();  
 comment2.setId(2);  
 comment2.setPostId(postId);  
 comment2.setHeadCommentId(headId);  
  
 Mockito  
 .*when*(commentDAO.getAnswersByCommentId(headId))  
 .thenReturn(List.*of*(comment1, comment2));  
  
 List<Comment> result = commentService.getAnswersByCommentId(headId);  
  
 Assertions.*assertEquals*(2, result.size());  
 Assertions.*assertEquals*(comment1.getId(), result.get(0).getId());  
 Assertions.*assertEquals*(comment2.getId(), result.get(1).getId());  
 }  
  
 @Test  
 public void createComment() {  
 Comment comment1 = new Comment();  
 comment1.setId(1);  
 comment1.setPostId(1);  
  
 Mockito  
 .*when*(postDAO.getCountComments(1))  
 .thenReturn(1);  
  
 Assertions.*assertEquals*(1, commentService.createComment(comment1));  
 }  
}

Было проведено три теста для методов getCommentByPostId, getAnswerByCommentId, createComment. Тесты пройдены успешно и их результаты приведены на Рисунке 2.2.

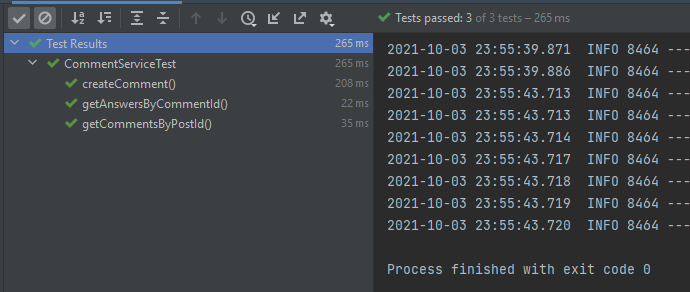


Рисунок 2.2 – Тестирование класса CommentService

## Тестирование класса PostServiceImpl

Исходный код класса PostServiceTest.java (Листинг 2.2)

Листинг 2.2 - Файл PostServiceTest.java

@SpringBootTest  
public class PostServiceTest {  
 @InjectMocks  
 PostServiceImpl postService;  
  
 @Mock  
 PostDAO postDAO;  
  
 @Test  
 public void create() {  
 Post newPost = new Post();  
 newPost.setText("новый пост");  
  
 Post resultPost = new Post();  
 resultPost.setText(newPost.getText());  
 resultPost.setId(1);  
  
 Mockito.*when*(postDAO.save(newPost)).thenReturn(resultPost);  
  
  
  
 Assertions.*assertEquals*(resultPost, postService.create(newPost, ""));  
 }  
  
 @Test  
 public void findByNickname() {  
 String nick = "nick";  
  
 Post post1 = new Post();  
 post1.setId(1);  
 post1.setAuthorNickname(nick);  
  
 Post post2 = new Post();  
 post2.setId(2);  
 post2.setAuthorNickname(nick);  
  
 Mockito  
 .*when*(postDAO.findByNickname(nick, 1, Integer.*MAX\_VALUE*))  
 .thenReturn(List.*of*(post2, post1));  
  
 List<Post> posts = postService.findByNickname(nick, 1, Integer.*MAX\_VALUE*);  
  
 Assertions.*assertEquals*(2, posts.size());  
 Assertions.*assertEquals*(post2, posts.get(0));  
 Assertions.*assertEquals*(post1, posts.get(1));  
 }  
  
 @Test  
 public void findByTags() {  
 Tag tag = new Tag();  
 tag.setName("test");  
  
 List<Tag> tagList = new ArrayList<>();  
 tagList.add(tag);  
  
 Post post1 = new Post();  
 post1.setId(1);  
 post1.setTagList(tagList);  
  
 Post post2 = new Post();  
 post2.setId(2);  
 post1.setTagList(tagList);  
  
 List<String> tagNames = List.*of*(tagList.get(0).getName());  
  
 Mockito  
 .*when*(postDAO.findByTag(tagNames, 1, Integer.*MAX\_VALUE*))  
 .thenReturn(List.*of*(post2, post1));  
  
 List<Post> posts = postService.findByTags(tagNames, 1, Integer.MAX\_VALUE);  
  
 Assertions.assertEquals(2, posts.size());  
 Assertions.assertEquals(post2, posts.get(0));  
 Assertions.assertEquals(post1, posts.get(1));  
 }  
  
 @Test  
 public void findAll() {  
 Post post1 = new Post();  
 post1.setId(1);  
  
 Post post2 = new Post();  
 post2.setId(2);  
  
 Mockito  
 .when(postDAO.findAll(1, Integer.MAX\_VALUE))  
 .thenReturn(List.of(post2, post1));  
  
 List<Post> posts = postService.findAll(1, Integer.MAX\_VALUE);  
  
 Assertions.assertEquals(2, posts.size());  
 Assertions.assertEquals(post2, posts.get(0));  
 Assertions.assertEquals(post1, posts.get(1));  
 }  
  
 @Test  
 public void createLike() {  
 int postId = 1;  
 int userId = 1;  
  
 PostDAO.AnswerLikes answerLikes = new PostDAO.AnswerLikes();  
 answerLikes.countLikes = 1;  
  
 Mockito  
 .when(postDAO.getLikes(postId, userId))  
 .thenReturn(answerLikes);  
  
 Assertions.assertEquals(answerLikes.countLikes, postService.createLike(userId, postId));  
 }  
  
 @Test  
 public void deleteLike() {  
 int postId = 1;  
 int userId = 1;  
  
 PostDAO.AnswerLikes answerLikes = new PostDAO.AnswerLikes();  
 answerLikes.countLikes = 0;  
  
 Mockito  
 .when(postDAO.getLikes(postId, userId))  
 .thenReturn(answerLikes);  
  
 Assertions.assertEquals(answerLikes.countLikes, postService.deleteLike(userId, postId));  
 }  
  
 @Test  
 public void getLikesInfo() {  
 List<Integer> postsIds = List.of(1, 2);  
 int userId = 1;  
  
 PostDAO.AnswerLikes answerLikes1 = new PostDAO.AnswerLikes();  
 answerLikes1.hasLike = false;  
  
 PostDAO.AnswerLikes answerLikes2 = new PostDAO.AnswerLikes();  
 answerLikes2.hasLike = true;  
  
 Mockito.when(postDAO.getLikes(postsIds.get(0), userId)).thenReturn(answerLikes1);  
 Mockito.when(postDAO.getLikes(postsIds.get(1), userId)).thenReturn(answerLikes2);  
  
 List<LikeByIdDTO> likesInfo = postService.getLikesInfo(postsIds, userId);  
  
 Assertions.assertEquals(postsIds.size(), likesInfo.size());  
  
 Assertions.assertEquals(postsIds.get(0), likesInfo.get(0).getPostId());  
 Assertions.assertEquals(answerLikes1.hasLike, likesInfo.get(0).isHasLike());  
  
  
 Assertions.assertEquals(postsIds.get(1), likesInfo.get(1).getPostId());  
 Assertions.assertEquals(answerLikes2.hasLike, likesInfo.get(1).isHasLike());  
 }  
  
 @Test  
 public void findMatchesByTags() {  
 List<String> tagsNames = List.of("ко");  
  
 List<String> expected = List.of("кот", "котики");  
  
 Mockito.when(postDAO.findMatchesByTags(tagsNames)).thenReturn(expected);  
  
 Assertions.assertEquals(expected, postService.findMatchesByTags(tagsNames));  
 }  
  
 @Test  
 public void getImage() {  
 int postId = 1;  
  
 Image image1 = new Image();  
 image1.setId(1);  
  
 Image image2 = new Image();  
 image2.setId(1);  
  
 Mockito.when(postDAO.getImages(postId)).thenReturn(List.of(image1, image2));  
  
 List<Image> res = postService.getImage(postId);  
  
 Assertions.assertEquals(2, res.size());  
 Assertions.assertEquals(image1.getId(), res.get(0).getId());  
 Assertions.assertEquals(image2.getId(), res.get(1).getId());  
 }  
  
 @Test  
 public void getInfoAboutCommentsAndLikes() {  
 Integer postId = 1;  
 Integer userId = 1;  
  
 PostDAO.AnswerLikes answerLikes = new PostDAO.AnswerLikes();  
 answerLikes.countLikes = 1;  
 answerLikes.hasLike = true;  
  
 int countComments = 3;  
  
 Mockito  
 .when(postDAO.getLikes(postId, userId))  
 .thenReturn(answerLikes);  
  
 Mockito  
 .when(postDAO.getCountComments(postId))  
 .thenReturn(countComments);  
  
 InfoAboutCommentsAndLikesDTO result = postService.getInfoAboutCommentsAndLikes(postId, userId);  
  
 Assertions.assertEquals(countComments, result.getCountComments());  
 Assertions.assertEquals(postId, result.getPostId());  
 Assertions.assertEquals(answerLikes.hasLike, result.isHasLike());  
 Assertions.assertEquals(answerLikes.countLikes, result.getCountLikes());  
 }  
}

Было проведено 10 тестов для методов findMatchesByTags, create, findAll, findByTags, getLikesInfo, createLike, findByNickname, deleteLike, getImage, getInfoAboutCommentsAndLikes. Тесты пройдены успешно. Их результаты приведены на Рисунке 2.3.

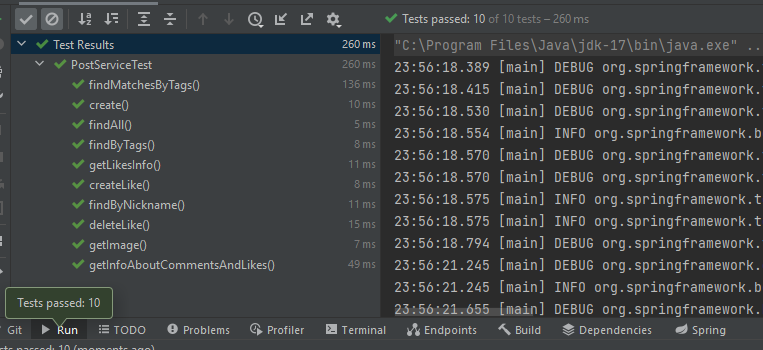


Рисунок 2.3 – Тестирование класса PostServiceImpl

## Тестирование класса UserService

Исходный код класса UserServiceTest.java (Листинг 2.3)

Листинг 2.3 - Файл UserServiceTest.java

@SpringBootTest  
public class UserServiceTest {  
 @InjectMocks  
 UserService userService;  
  
 @Mock  
 UserDAO userDAO;  
  
 @Test  
 public void getUserByAuthorization() {  
 String nickname = "nickname";  
 String password = "passssss";  
 String auth = userService.codeBase64(nickname, password);  
  
 User user = new User();  
 user.setNickname(nickname);  
  
 Mockito.*when*(userDAO.findByUsername(user.getNickname())).thenReturn(user);  
  
 try {  
 Assertions.*assertEquals*(user, userService.getUserByAuthorization(auth));  
 } catch (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Test  
 public void getAvatarImg() {  
 String nickname = "nickname";  
 String img = "picture";  
  
 Mockito.*when*(userDAO.getAvatarImg(nickname)).thenReturn(img);  
  
 Assertions.*assertEquals*(img, userService.getAvatarImg(nickname));  
 }  
  
 @Test  
 public void getCountPosts() {  
 String nickname = "nickname";  
 int countPosts = 5;  
  
 Mockito.*when*(userDAO.getCountPosts(nickname)).thenReturn(countPosts);  
  
 Assertions.*assertEquals*(countPosts, userService.getCountPosts(nickname));  
 }  
  
 @Test  
 public void getCountLikes() {  
 String nickname = "nickname";  
 int countLikes = 5;  
  
 Mockito.*when*(userDAO.getCountLikes(nickname)).thenReturn(countLikes);  
  
 Assertions.*assertEquals*(countLikes, userService.getCountLikes(nickname));  
 }  
}

Было проведено 4 теста для методов getCountLikes, getCountPosts, getUserByAuthorization, getAvatarImg. Тесты пройдены успешно. Их результаты приведены на Рисунке 2.4

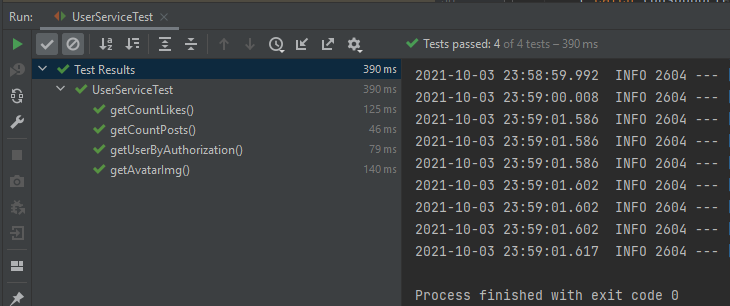


Рисунок 2.4 – Тестирование класса UserService

# Оценка качества чужого кода, соответствия документации и его надежности

Код хорошо читается, в ходе тестирования не были выявлены несоответствия кода документации, а также код прошёл все необходимые тесты. Оценка: 5 баллов.