**5.** ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Одним из основных этапов разработки программного средства является его тестирование. Тестирование ПО – это процесс выявление ошибок при разностороннем исследовании приложения. Выявление ошибок осуществляется путём сопоставления, реального и ожидаемого результатов тестов. В более широком смысле тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий две различные цели: продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям и выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Каждый этап разработки программного обеспечения сопровождается написанием большого количества разнообразных тестов. Ниже приведены основные виды тестирования:

— модульное тестирование;

— функциональное тестирование;

— тестирование производительности;

— тестирование совместимости;

— тестирование интерфейса пользователя.

Написание тестов необходимо для снижения рисков нарушения работоспособности приложения при внесении дополнительных изменений. Сам процесс написания тестов является довольно трудозатратным и требующим большого количества времени, но эти затраты окупаются надежностью и стабильности программного средства. По мере усложнения кода проекта стоимость устранения дефектов ПО может экспоненциально возрастать. Инструменты статического и динамического анализа помогают сократить эти затраты благодаря обнаружению программных ошибок на ранних этапах жизненного цикла ПО.

В ходе разработки настоящего программного средства были применены следующие виды тестирования: модульное и функциональное. Модульное тестирование производилось с использованием библиотеки Junit версии 4.12, функциональное проектирование производилось на персональном компьютере Lenovo G700 с установленной ОС Manjaro 17.0.1 Gellivara, процессором Intel Core i7 и оперативной памятью объемом 8 гигабайт.

**5.1** Функциональное тестирование

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях. При тестировании предполагается обработка данных и предсказуемая реакция приложения, когда данные, поданные на вход не являются корректными.

Результаты функционального тестирования приложения приведены в таблице 5.1.

Таблица 5.1 – Тест-кейсы и результаты их выполнения

|  |  |  |  |
| --- | --- | --- | --- |
| Компонент | Название | Шаги и ожидаемый результат | Результат выполнения |
| 1 | 2 | 3 | 4 |
| Окно входа в систему | Аутентификация с корректным логином и паролем | 1 Ввести логин и пароль  2 Нажать кнопку «Login»  Ожидаемый результат: переход на главную страницу приложения | Тест пройден успешно |
| Окно регистрации пользователя | Регистрация пользователя | 1Ввести логин и пароль  2 Ввести адрес электронной почты  3 Ввести имя и фамилию  4 Нажать кнопку «Register»  Ожидаемый результат: получение уведомления об успешной регистрации | Тест пройден успешно |
| Окно доски проекта | Смена статуса задачи | 1 Перетащить задачу из статуса «New» в статус «In progress»  Ожидаемый результат: Отображение задачи в колонке «In progress» | Тест пройден успешно |
| Добавления комментария к задаче | 1 Нажать кнопку «Comment» на задаче  2 Ввести текст комментария  3 Нажать кнопку «Send comment»  Ожидаемы результат: Комментарий добавился к задаче | Тест пройден успешно |

Таблица 5.1 – Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Окно доски проекта | Добавление приложения к задаче | 1 Нажать кнопку «Attach»  2 Выбрать файл  3 Нажать кнопку «Add»  Ожидаемый результат: добавление файла к приложениям задачи | Тест пройден успешно |
| Удаление задачи | 1 Начать кнопку «Х» на задаче  2 Задача удалена  Ожидаемый результат:  задача удалена из списка | Тест пройден успешно |
| Создание новой задачи при наличии необходимых прав | 1 Нажать кнопку «New task»  2 Заполнит поля с названием задачи и описанием  3 Нажать кнопку «Create»  4 Получить уведомление об успешном создании задачи  Ожидаемы результат:  создана новая задача | Тест пройден успешно |
| Окно построения отчетов | Получение отчета по количеству задач в каждом из статусов | 1 Нажать кнопку «Task progress»  2 Получить информацию о задачах на проекте  Ожидаемы результат:  отображен отчет по количеству задач в каждом из статусов | Тест пройден успешно |

Как видно из таблицы 5.1 все функциональные тесты были пройдены успешно. Ошибки, повлекшие за собой некорректное выполнение тестов, были оперативно устранены. В следующем пункте более детально будет рассмотрено модульное тестирование разработанного продукта.

**5.1** Модульное тестирование

Модульное тестирование – процесс в [программировании](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), позволяющий проверить на корректность отдельные модули [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программы. Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к [регрессии](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Основной целью модульного тестирования является разбиение разрабатываемой системы на модули и проверка работоспособности каждого модуля по отдельности. Если при вызове какого-либо метода определенного модуля есть необходимость обратиться к другому модулю, то вместо него подставляется так называемы mock-объект, который по своей сути является своего рода заглушкой.

При разработке настоящего программного средства была использована методология TDD (Test Driven Development). Суть этой методологии в том, что для разрабатываемого модуля изначально пишутся тесты, а позже пишется реализация конкретных методов модуля таким образом, чтобы тесты были пройдены. Разработка через тестирование требует от разработчика создания автоматизированных модульных тестов, определяющих требования к коду непосредственно перед написанием самого кода. Тест содержит проверки условий, которые могут либо выполняться, либо нет. Когда они выполняются, говорят, что тест пройден.

При разработке тестов была использована библиотека Junit версии 4.12. Для подключения библиотеки был использован фреймворк для автоматизации сборки проекта Apache Maven. Ниже приведен пример подключения данной библиотеки.

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

Полный перечень разработанных тестов приведен в приложении Д. Ниже рассматриваются тесты, покрывающие основной функционал данного программного продукта.

Для класса ProjectController ключевым методом является createProject. Данный метод принимает экземпляр класса Project и создает новый проект в системе. В первом случае рассмотрим поведение метода при передаче в него null. При передаче null метод должен выбросить исключение IllegalArgumentException. Ниже приведен листинг для данного теста.

@Test(expected = IllegalArgumentException.class)

public void createProjectTestWithEmptyProject(){

Project project = null;

ProjectController projectController = new ProjectController();

projectController.createProject(null);

}

Ниже приведен тест для случая, где не указан руководитель проекта. Ожидается, что при таком вызове метод выбросит исключение InvalidUserException.

@Test(expected = IllegalArgumentException.class)

public void createProjectTestWithoutLead(){

Project project = new Project();

project.setName(“New Project”);

ProjectController projectController = new ProjectController();

project.setLead(null);

projectController.createProject(project);

}

Для класса TaskController ключевыми методами являются методы createTask и changeTaskStatus.

В следующем тесте производится попытка создания новой задачи без указания проекта, которому она принадлежит. В этом случае ожидается, что метод выбросит исключение InvalidProjectException. Ниже приведен листинг для данного теста.

@Test(expected = InvalidProjectException.class)

public void createTaskTestWithoutProject(){

Task task = new Task();

task.setName(“Create new button”);

task.setDescription(“Description”);

task.setCreator(new User(“vladislav.zavadski”));

task.setProject(null);

TaskController taskController = new TaskController();

taskController.createTask(task);

}

В следующем тесте производится попытка создания задачи для проекта, которого не существует. В таком случае ожидается исключение ProjectNotFoundException. Ниже приведен листинг для данного теста.

@Test(expected = InvalidProjectException.class)

public void createTaskTestWithInvalidProject(){

Task task = new Task();

task.setName(“Create new button”);

task.setDescription(“Description”);

task.setCreator(new User(“vladislav.zavadski”));

Project project = new Project();

project.setId(666);

TaskController taskController = new TaskController();

taskController.createTask(task);

}

Далее будет рассмотрен случай изменения статуса задачи, на статус которого не существует. В данном случае следует ожидать, что метод changeTaskStatus выбросит исключение InvalidTaskStatusException. Листинг данного теста приведен ниже.

@Test(expected = InvalidTaskStatusException.class)

public void changeTaskStatusTestToInvalidStatus(){

Task task = new Task();

task.setId(10);

task.setTaskStatus(new TaskStatus(2));

TaskController taskController = new TaskController();

taskController.changeTaskStatus(task, new TaskStatus(3));

}

В следующем тесте производится попытка изменить статус задачи, которой не существует. В таком случае ожидается, что методом будет выброшено исключение TaskNotFoundException. Листинг для данного теста приведен ниже.

@Test(expected = InvalidTaskStatusException.class)

public void changeTaskStatusTestToInvalidStatus(){

Task task = new Task();

task.setId(666);

task.setTaskStatus(new TaskStatus(2));

TaskController taskController = new TaskController();

taskController.changeTaskStatus(task, new TaskStatus(3));

}

Далее смоделируем ситуацию поиска задач на проекте. В таком случае ожидается, что метод getProjectTasks вернет список всех задач на проекте. В конкретном случает задач всего 4. Листинг кода для данного теста приведен ниже.

@Test

public void getProjectTasksTest(){

TaskController taskController = new TaskController();

List<Task> tasks = taskController.getProjectTasks(2);

assertEquals(2, tasks.size());

}

В таблице 5.2 представлено покрытие тестами функционала наиболее важных модулей программного средства.

Таблица 5.2 – Покрытие модулей тестами

|  |  |
| --- | --- |
| Название модуля | Покрытие тестами |
| Модуль управления задачами | 6 случаев, ошибок не выявлено |
| Модуль отправки уведомлений | 3 случая, 1 ошибка исправлена |
| Модуль доступа к данным | 32 случая, 6 ошибок исправлено |
| Модуль аутентификации | 5 случаев, ошибок не выявлено |

Покрытие unit-тестами вышеперечисленных модулей позволило выявить и исправить 7 ошибок.

**5.3** Вывод

В данной главе были рассмотрены виды тестирования, которые использовались во время разработки программного средства. Использование функционального и модульного тестирования позволило выявить и устранить ошибки. Разработанное программное средство проходит тестовые испытания, что свидетельствует о его работоспособности.