МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

Факультет информационных технологий и программной инженерии (ИТПИ)

Кафедра программной инженерии и вычислительной техники

Лабораторная работа №6

по дисциплине «Процессы жизненного цикла программного обеспечения»

Выполнил:

студент 4-го курса

группы ИКПИ-14

Хмиль Даниил Игоревич

Подпись_____

Принял:

Пачин Андрей Владимирович

Подпись_______

Санкт-Петербург

2025

1. Постановка задачи

Цель: автоматизация тестирования клиент-серверного приложения для управления недвижимостью с использованием инструментов автоматизированного тестирования.

Задачи:

1. Реализовать автотесты для проверки функциональности приложения.
2. Обеспечить контроль результатов выполнения сценариев.
3. Выявить и зафиксировать дефекты.

2. Описание реализованных автотестов

Инструменты и подходы:

Язык программирования: Java, среда разработки IntelliJ Community Edition.

Фреймворки для тестирования: JUnit 5 совместно с TestFX для автоматизации действий пользователя.

Подход: Автоматизированное GUI-интеграционное тестирование с элементами E2E (проверяет полный сценарий работы пользователя от входа в систему до выполнения целевого действия).

Описание автотестов:

- void testCreateNewAd() – проверка создания нового объявления, здесь же проверяется отображение новых данных в таблице и авторизация.
- void testBookAd() - проверка бронирование объявления, здесь же проверяется отображение новых данных в таблице, и переход между окнами приложения.
- void testClientInterface() – проверка соответствия интерфейса роли клиента, здесь же проверяется авторизация.
- void testInvalidPriceInput() – проверка появления ошибки при попытке размещения объявления с ценой неверного формата.
- void testInvalidAreaInput() - проверка появления ошибки при попытке размещения объявления с площадью неверного формата

- void testMissingData() - проверка появления ошибки при создании пустого объявления, здесь же проверяется авторизация.

Используемые тест-кейсы представлены в таблице 1.

Таблица 1. Тест-кейсы

Тест-сьют	**Тест-кейс(тип)**	**Дата**	**Предусловие**	**Шаги**	**Ожидаемый результат**	**Результат**
TS1	1.1 Создание нового объявления (позитивный)	20.03.2025	Пользователь авторизован как «Агент»	1. Нажать на кнопку «Создать объявление» 2. Заполнить все поля (тип, вид сделки, агентство, площадь, цена, количество комнат) 3. Нажать кнопку «Разместить»	Появление новой позиции в таблицах «Рынок недвижимости» и «Ваши объявления»	Успешно
	1.4 Бронирование объявления клиентом (позитивный)	20.03.2025	Пользователь авторизован как «Клиент», объявление создано	1. Выбрать двойным нажатием левой кнопкой мыши любое объявление в таблице «Рынок недвижимости» 2. Нажать кнопку «Откликнуться»	Изменение статуса объявления в таблице на «Забронировано» и сокрытие позиции для других пользователей	Успешно
TS2	2.1 Переход между окнами (позитивный)	21.03.2025	Пользователь авторизован	1. Перейти из главного меню в окно «Ваши объявления. 2. Выбрать двойным нажатием левой	Корректный переход между окнами без возникновения ошибок	Успешно

				кнопкой мыши любое объявление 3. Нажать кнопку «Отмена» 4. Нажать кнопку «назад»		
	2.2 Разный интерфейс для агента и клиента (позитивный)	21.03.2025	Пользователь авторизован, объявления созданы	1. Для агента: убедиться, что имеются кнопки «Создать объявление» и «Ваши объявления» на главном окне. Для клиента: убедиться, что кнопки описанные выше отсутствуют, присутствует кнопка «Ваши сделки» 2. Для агента: в окне ваши объявления убедиться, что есть возможность удалить и редактировать объявление. Для клиента: убедиться в отсутствии кнопок удаления и редактирования объявления.	Корректное отображение данных для каждой из ролей пользователей.	Успешно

				Должна присутствовать кнопка «Откликнуться»		
	2.3 Отображение данных в таблице объявлений (позитивный)	21.03.2025	Пользователь авторизован, объявления созданы	1. Проверить отображение данных в таблице «Рынок недвижимости» 2.1. Для агента перейти в окно «Ваши объявления» и проверить объявления 2.2 Для клиента перейти в окно «Ваши сделки» и проверить объявления	Корректное отображение всех данных в таблице. Ничего не должно перекрываться или не вмещаться.	Успешно
TS3	3.1 Ввод неверного типа данных в поля «Цена» и «Площадь» (негативный)	21.03.2025	Пользователь авторизован как «Агент»	1. Нажать кнопку «Создать объявление» 2. Написать в поле «Цена» и/или «Площадь» отрицательное, нулевое или буквенное значение. 3. Нажать на кнопку разместить	Появление окна с ошибкой и невозможность создания объявления	Успешно
	3.2 Ввод не всех данных для объявления	21.03.2025	Пользователь авторизован как «Агент»	1. Нажать кнопку	Появление окна с ошибкой и невозможность	Успешно

	(негативный)			«Создать объявление» 2. Нажать кнопку «Разместить»	создания объявления	

3. Код автотестов

Класс TestFXBase.java – содержит функции для установления соединения с сервером и запуска приложения.

```
package com.example.coursework;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.testfx.api.FxToolkit;
import org.testfx.framework.junit5.ApplicationTest;

import javafx.application.Platform;
import javafx.stage.Stage;

public abstract class TestFXBase extends ApplicationTest {

    @BeforeEach
    public void setUpClass() throws Exception {
        try {
            new ClientSomthing("localhost", 4004);
        } catch (Exception e) {
            System.out.println("Сервер не обнаружен");
            System.exit(0);

        }
        ApplicationTest.launch(Client.class);
    }

    @Override
    public void start(Stage stage) throws Exception {
        stage.show();
    }

    @AfterEach
    public void tearDown() throws Exception {
        FxToolkit.hideStage();
```

```
            Platform.runLater(() -> {
                // Очистка ресурсов
            });
        }
    }
```

Класс ClientTests.java - содержит тесты от лица Клиента

```
package com.example.coursework;
import org.junit.jupiter.api.Test;
import org.testfx.matcher.base.NodeMatchers;
import org.testfx.matcher.control.TableViewMatchers;

import static org.testfx.api.FxAssert.verifyThat;
import static org.testfx.matcher.control.LabeledMatchers.hasText;

public class ClientTests extends TestFXBase {

    @Test
    void testBookAd() {
        // Авторизация клиента
        clickOn("#loginField").write("Client");
        clickOn("#passwordField").write("Clientpass");
        clickOn("#authorizationBtn");

        // Бронирование
        doubleClickOn("#realPropertyTable .table-row-cell");
        clickOn("#respondBtn");
        clickOn("#yourDealBtn");

        verifyThat("#realPropertyTable", NodeMatchers.isNotNull());
    }

    @Test
    void testClientInterface() {
        clickOn("#loginField").write("Client");
        clickOn("#passwordField").write("Clientpass");
        clickOn("#authorizationBtn");
        verifyThat("#yourDealBtn", NodeMatchers.isVisible());
    }

}
```

Класс AgentTests.java – содержит тесты от лица Агента

```
package com.example.coursework;

import org.junit.jupiter.api.Test;
import org.testfx.matcher.base.WindowMatchers;
import org.testfx.matcher.control.TableViewMatchers;

import static org.testfx.api.FxAssert.verifyThat;
import static org.testfx.matcher.control.LabeledMatchers.hasText;

public class AgentTests extends TestFXBase {

    @Test
    void testCreateNewAd() {
        // Авторизация агента
        clickOn("#loginField").write("Daniil");
        clickOn("#passwordField").write("Khmil");
        clickOn("#authorizationBtn");

        // Создание объявления
        clickOn("#createAdBtn");
        clickOn("#typeField").clickOn("Квартира");
        clickOn("#dealTypeField").clickOn("Покупка");
        clickOn("#agencyField").write("Агентство");
        clickOn("#areaField").write("50");
        clickOn("#priceField").write("1000000");
        clickOn("#roomsField").clickOn("2");
        clickOn("#applyBtn");

        // Проверка результата
        verifyThat("#realPropertyTable", TableViewMatchers.hasTableCell("Агентство"));
    }

}
```

Класс NegativeTests.java – содержит негативные тесты

```
package com.example.coursework;
import org.junit.jupiter.api.Test;
import org.testfx.matcher.base.NodeMatchers;
```

```
import org.testfx.matcher.base.WindowMatchers;
import org.testfx.matcher.control.TableViewMatchers;

import static org.testfx.api.FxAssert.verifyThat;
import static org.testfx.matcher.control.LabeledMatchers.hasText;
public class NegativeTests extends TestFXBase {

    @Test
    void testInvalidPriceInput() {
        loginAsAgent();
        clickOn("#createAdBtn");
        clickOn("#createAdBtn");
        clickOn("#typeField").clickOn("Квартира");
        clickOn("#dealTypeField").clickOn("Покупка");
        clickOn("#agencyField").write("Агентство");
        clickOn("#areaField").write("50");
        clickOn("#priceField").write("abc");
        clickOn("#roomsField").clickOn("2");
        clickOn("#applyBtn");

        verifyThat(window("Ошибка"), WindowMatchers.isShowing());
    }

    @Test
    void testInvalidAreaInput() {
        loginAsAgent();
        clickOn("#createAdBtn");
        clickOn("#createAdBtn");
        clickOn("#createAdBtn");
        clickOn("#typeField").clickOn("Квартира");
        clickOn("#dealTypeField").clickOn("Покупка");
        clickOn("#agencyField").write("Агентство");
        clickOn("#areaField").write("abc");
        clickOn("#priceField").write("10000");
        clickOn("#roomsField").clickOn("2");
        clickOn("#applyBtn");

        verifyThat(window("Ошибка"), WindowMatchers.isShowing());
    }
```

```
    @Test
    void testMissingData() {
        loginAsAgent();
        clickOn("#createAdBtn");
        clickOn("#applyBtn");

        verifyThat(window("Ошибка"), WindowMatchers.isShowing());
    }

    private void loginAsAgent() {
        clickOn("#loginField").write("Daniil");
        clickOn("#passwordField").write("Khmil");
        clickOn("#authorizationBtn");
    }
}
```

4. Отчёт по тестированию

В результате автоматизированного тестирования было разработано и проведено 6 тестов (функций), которые полностью покрывают указанные выше тест-кейсы, дефектов не обнаружено. Результаты описаны в таблице 2.

Таблица 2. Результаты тестирования

Тест-кейс	**Результат**
1.1 Создание нового объявления	Успешно
1.4 Бронирование объявления клиентом	Успешно
2.1 Переход между окнами	Успешно
2.2 Разный интерфейс для агента и клиента	Успешно
2.3 Отображение данных в таблице объявлений	Успешно
3.1 Ввод неверного типа данных в поля «Цена» и «Площадь»	Успешно

3.2 Ввод не всех данных для объявления	Успешно

Итого: 100% процентов успешных результатов автоматизированного тестирования.

5. Вывод

В ходе автоматизированного тестирования было разработано 6 автотестов (функций), которые покрывают все тест-кейсы. У всех тестов результат положительный, дефектов не обнаружено, приложение готово к эксплуатации.

6. Список использованных источников

1. Хмиль Д.И. Документ Курсовая работа Java [docx] – 2023. – Локальный файл на компьютере автора.
2. Пачин А.В. Тестирование программного обеспечения (или процессы ЖЦ ПО) – лабораторный практикум URL: https://docs.google.com/document/d/1Po2DY8b9JfhQ6BAhzQM-3SgS4ifLlozO/edit?pli=1&tab=t.0 (дата обращения 04.04.25)