

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по учебной практике**

**ТЕМА: «UI-тестирование веб-приложения Google Keep»**

Студент гр. 3342	_____	Белаид Фарук
Студент гр. 3343	_____	Какира У.Н.
Студент гр. 3383	_____	Лысиков Михаил
Руководитель	_____	Шевелева А.

Санкт-Петербург  
2025

## ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студент Белаид Фарук группы 3342

Студент Какира У.Н. группы 3343

Студент Лысиков Михаил группы 3383

Тема практики: **UI-тестирование**

Задание на практику:

Разработка и выполнение UI-тестов для веб-сервиса с использованием Selenide (Google Keep)

Сроки прохождения практики: 26.06.2025 – 07.07.2025

Дата сдачи отчета: 05.07.2025

Дата защиты отчета: 07.07.2025

Студент	_____	Белаид Фарук
Студент	_____	Какира У.Н.
Студент	_____	Лысиков Михаил
Руководитель	_____	Шевелева А.

## АННОТАЦИЯ

Целью данной работы является разработка и выполнение набора UI-тестов для веб-приложения Google Keep с использованием языка программирования Java и библиотеки Selenide, основанной на Selenium WebDriver. В рамках проекта реализованы десять тестовых сценариев, охватывающих ключевой функционал системы: проверка создания пустой заметки, закрепление и открепление заметки, архивирование заметки, отмена удаления заметки, редактирование заметки, изменение цвета заметки, поиск заметки, удаление заметки, добавление метки к заметке и добавление чек-листа. Каждый тест моделирует действия пользователя в браузере, включая взаимодействие с элементами интерфейса, навигацию по сайту и валидацию результатов. Тесты обеспечивают всестороннюю проверку корректной работы пользовательских функций, а также устойчивости системы к ошибочным действиям. Использование библиотеки Selenide позволяет упростить написание и сопровождение кода за счёт лаконичного синтаксиса, встроенных методов ожидания и интеграции с JUnit. В отчёте также представлены UML-диаграмма классов и описание архитектуры тестовой системы, включая базовый класс, классы тестов и реализацию паттерна Page Object. Работа может служить шаблоном для UI-тестирования других веб-приложений и систем.

## SUMMARY

The purpose of this work is to develop and run a set of UI tests for the Google Keep web application using the Java programming language and the Selenide library based on Selenium WebDriver. The project implemented ten test scenarios covering the key functionality of the system: checking the creation of an empty note, pinning and unpinning a note, archiving a note, canceling the deletion of a note, editing a note, changing the color of a note, searching for a note, deleting a note, adding a label to a note and adding a checklist. Each test simulates user actions in the browser, including interaction with interface elements, site navigation, and validation of results. The tests provide a comprehensive check of the correct operation of user functions, as well as the system's resistance to erroneous actions. Using the Selenide library makes it easier to write and maintain code due to its concise syntax, built-in waiting methods, and integration with JUnit. The report also provides a UML class diagram and a description of the architecture of the test system, including the base class, test classes, and the implementation of the Page Object pattern. The work can serve as a template for UI testing of other web applications and systems.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	6
ОПИСАНИЕ КЛАССОВ И МЕТОДОВ.....	9
Архитектура проекта .....	9
ОБЗОР ТЕСТОВ.....	10
Описание основных методов тестирования.....	10
Рисунок 1 – UML-диаграмма классов тестового приложения...	14
РЕЗУЛЬТАТ ОДНОГО ТЕСТА (ТЕСТ ПОИСКА ЗАМЕТКИ.....	15
ЗАКЛЮЧЕНИЕ.....	18

## ВВЕДЕНИЕ

Тестирование пользовательского интерфейса (UI) — важный этап в процессе разработки программного обеспечения, обеспечивающий корректную работу визуальных компонентов и удобство взаимодействия пользователя с системой. Одним из современных и широко применяемых инструментов для автоматизации UI-тестирования является фреймворк **Selenide**, построенный на базе **Selenium WebDriver**. Он предоставляет простой и выразительный API для взаимодействия с веб-интерфейсами, а также удобные механизмы управления ожиданиями и элементами страницы.

В рамках данной практической работы целью является выполнение автоматизированного UI-тестирования веб-приложения **Google Keep**. Данный сервис предоставляет пользователям возможность создавать и управлять заметками, списками и напоминаниями. Google Keep представляет собой удобную платформу для анализа устойчивости пользовательского интерфейса к типовым сценариям использования.

В процессе реализации проекта проведено тестирование десяти ключевых функций интерфейса:

- проверка создания пустой заметки,
- закрепление и открепление заметки,
- архивирование заметки,

- отмена удаления заметки,
- редактирование заметки,
- изменение цвета заметки,
- поиск заметки,
- удаление заметки,
- добавление метки к заметке,
- добавление чек-листа.

Для реализации тестов использовался язык программирования **Java** и система сборки **Maven**. Тесты организованы по принципу паттерна **Page Object**, что повышает читаемость и переиспользуемость кода. Проект может быть запущен напрямую из терминала и использоваться как шаблон для автоматизированного тестирования других веб-приложений.

Для достижения поставленной цели в рамках практики решались следующие задачи:

- Изучение принципов автоматизированного UI-тестирования.
- Освоение библиотеки Selenide и языка Java для написания тестов.
- Разработка набора тестов, покрывающего функционал веб-приложения Google Keep.

- Формирование архитектуры проекта и реализация тестов с использованием паттерна Page Object.
- Сборка и запуск тестов с помощью Maven.



## ОПИСАНИЕ КЛАССОВ И МЕТОДОВ

Код для следующих классов и методов находится в следующем репозитории гитхаб: <https://github.com/faroukbelaid9/Summer-Practice>

### Архитектура проекта

Проект реализован с использованием паттерна **Page Object** в рамках UI-тестирования. Основные классы:

- **BaseTest** — базовый тестовый класс, инициализирует браузер и завершает его работу.
- **NotesTest** — содержит набор из 10 тестов, каждый из которых проверяет определённую функцию Google Keep.
- **NotesPage** — класс страницы, реализует методы взаимодействия с веб-интерфейсом Google Keep.

### 1. Класс BaseTest

**Назначение:** Настройка и завершение сессии браузера для всех тестов.

#### Основные методы:

- **@BeforeClass setUp()** — инициализирует браузер (ChromeDriver) и открывает Google Keep.
- **@AfterClass tearDown()** — закрывает браузер после выполнения тестов.

### 2. Класс NotesTest

**Назначение:** Содержит 10 UI-тестов, выполняемых над заметками Google Keep.

Каждый тест использует методы из NotesPage.

## Обзор тестов:

### Описание основных методов тестирования

№	Название метода	описание теста	Назначение
1	testEmptyNote()	Проверяет, что при попытке сохранить пустую заметку она не будет добавлена в список.	Убедиться в корректной обработке пустого ввода и предотвращении создания некорректных заметок.
2	testPinNote()	Создаёт обычную заметку, закрепляет её и проверяет, что она появилась в разделе закреплённых.	Проверить функциональность закрепления заметок.
3	testArchiveNote()	Создаёт заметку и архивирует её. Затем проверяет, что она исчезает с главного экрана.	Убедиться в корректной работе архивации заметок.
4	testUndoDelete()	Удаляет заметку и сразу отменяет	Проверить возможность

		удаление. Проверяет, что заметка восстановлена.	восстановления случайно удалённых заметок.
5	testDeleteNote()	Удаляет заметку и убеждается, что она полностью исчезает из интерфейса.	Убедиться в правильном удалении заметок.
6	testAddLabelToNote()	Добавляет к заметке пользовательскую метку и проверяет, что она прикреплена.	Проверить корректность добавления меток к заметкам.
7	testAddChecklistToNote()	Создаёт чек-лист с несколькими пунктами в заметке и проверяет, что все пункты присутствуют.	Проверить поддержку чек- листов в заметках.
8	testEditNote()	Редактирует заголовок существующей заметки и проверяет, что	Проверить возможность редактирования содержимого заметки.

		новый заголовок сохранён.	
9	testSearchNoteByTitle()	Создаёт заметку с уникальным заголовком, затем выполняет поиск по нему. Проверяет, что заметка находится.	Проверить работу функции поиска заметок по заголовку.
10	testChangeNoteColor()	Меняет цвет заметки на другой (например, красный), а затем возвращает цвет по умолчанию.	Проверить корректную работу смены цвета заметки.

### 3. Класс **NotesPage**

**Назначение:** Инкапсулирует все действия, которые пользователь может совершать на странице Google Keep.

#### Методы класса **NotesPage**

##### Общие методы:

- `getCurrentNoteCount()`

Получает текущее количество отображаемых заметок.

- `isNotePresent(title: String)`

Проверяет наличие заметки с определённым заголовком.

### **Создание и редактирование:**

- `createNote(String title)`

Создаёт текстовую заметку с заголовком.

- `createEmptyNote()`

Нажимает на поле ввода, но не вводит текст — имитирует пустую заметку.

- `editNoteTitle(String oldTitle, String newTitle)`

### **Организация:**

- `pinNoteByTitle(String title)`

Закрепляет заметку.

- `archiveNoteByTitle(String title)`

Отправляет заметку в архив.

- `deleteNoteByTitle(String title, boolean undo)`

Удаляет заметку. Если `undo == true`, сразу жмёт “Отменить”.

- `addLabelToNoteByTitle(String title, String label)`

Добавляет метку к заметке.

- `createChecklistNote(String title, String[] items)`

Создаёт заметку с чек-листом. Добавляет все элементы из массива.

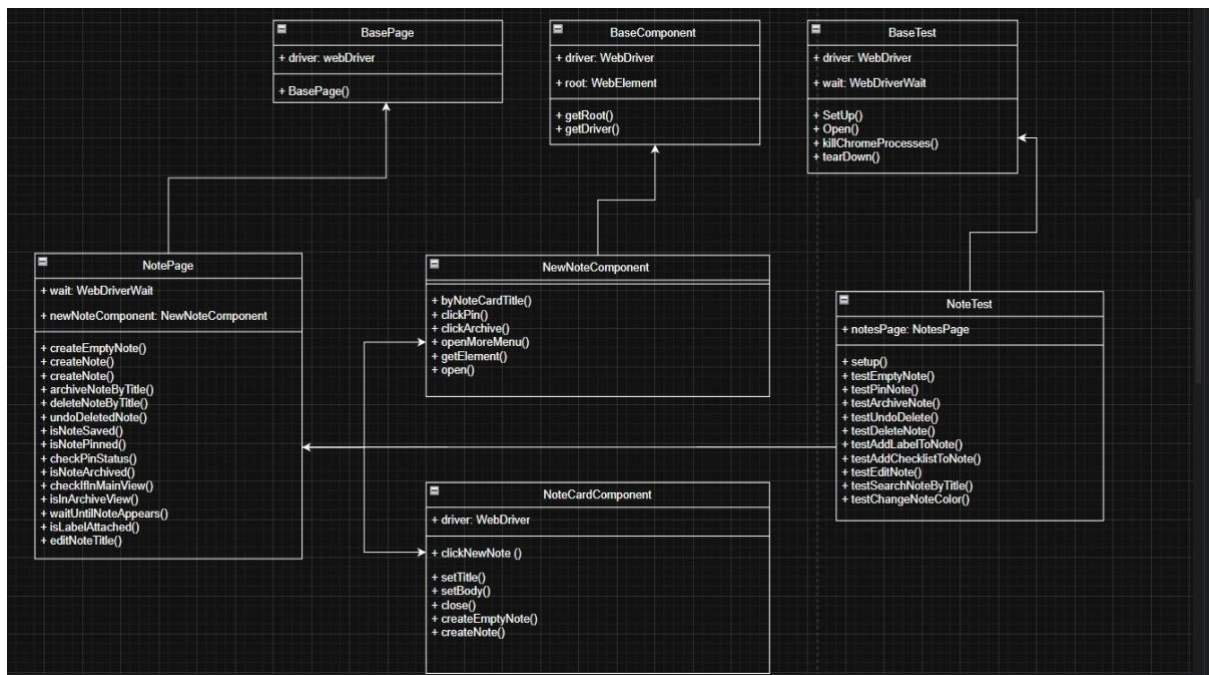
- `searchNoteByTitle(String title)`

Выполняет поиск по заголовку через поисковую строку.

- `changeNoteColor(String title, String color)`

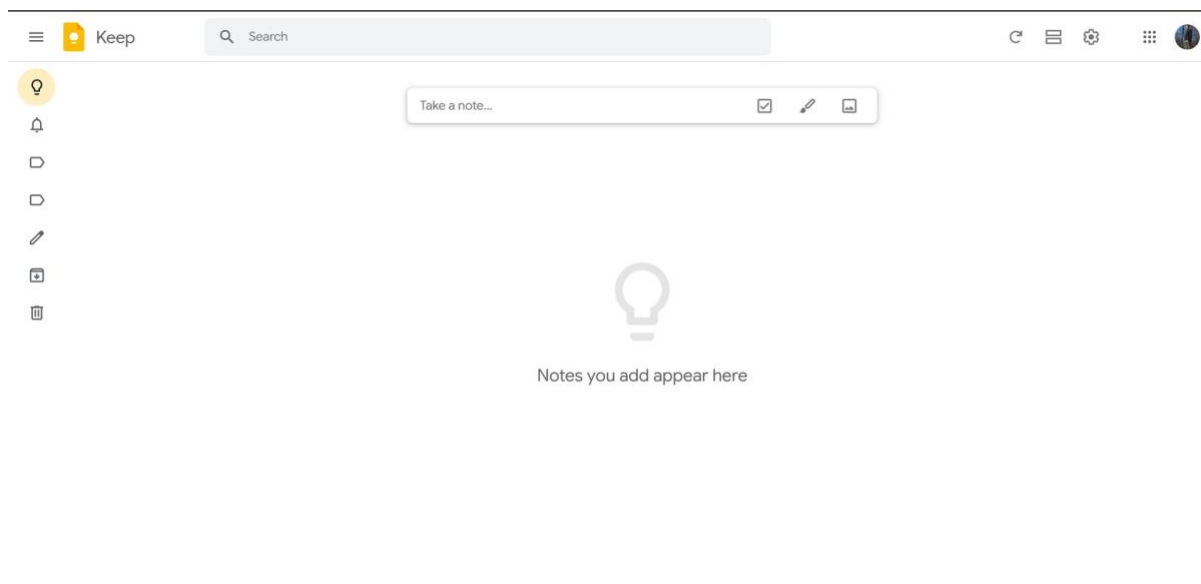
Меняет цвет заметки. Возможные значения: "red", "default" и др.

«Рисунок 1 – UML-диаграмма классов тестового приложения»

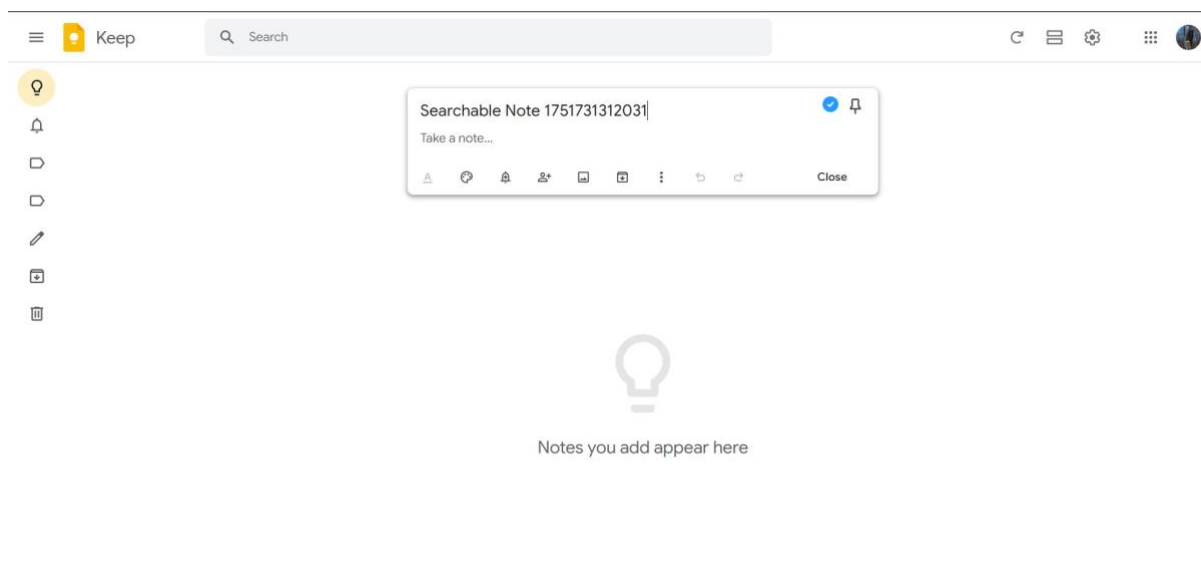


## РЕЗУЛЬТАТ ОДНОГО ТЕСТА (ТЕСТ ПОИСКА ЗАМЕТКИ)

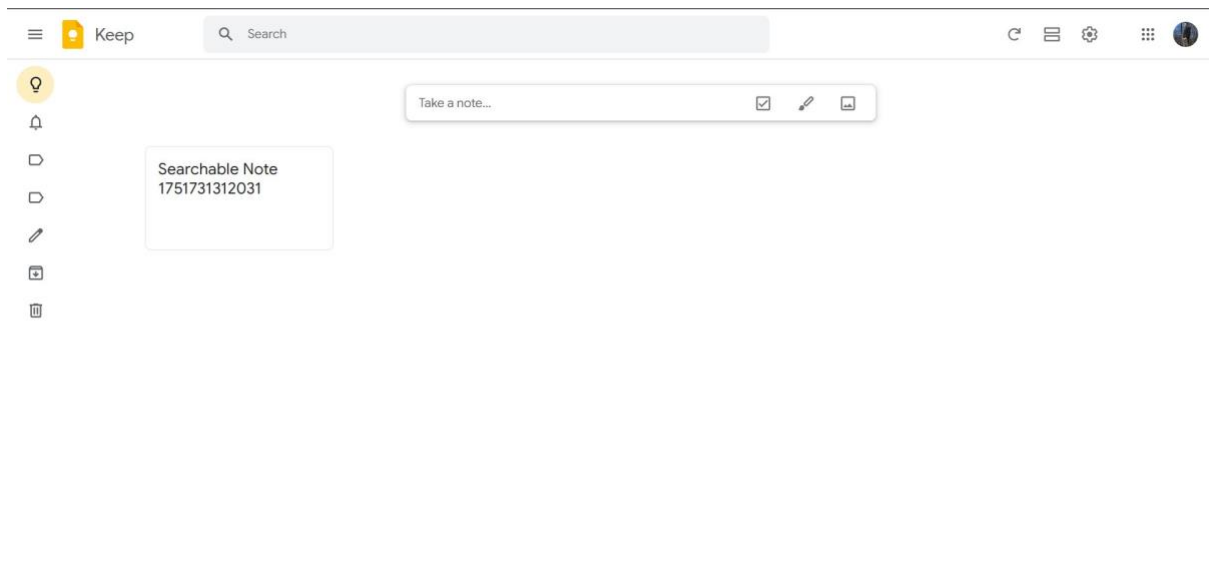
### Открытие веб-сайта Google Keep в браузере



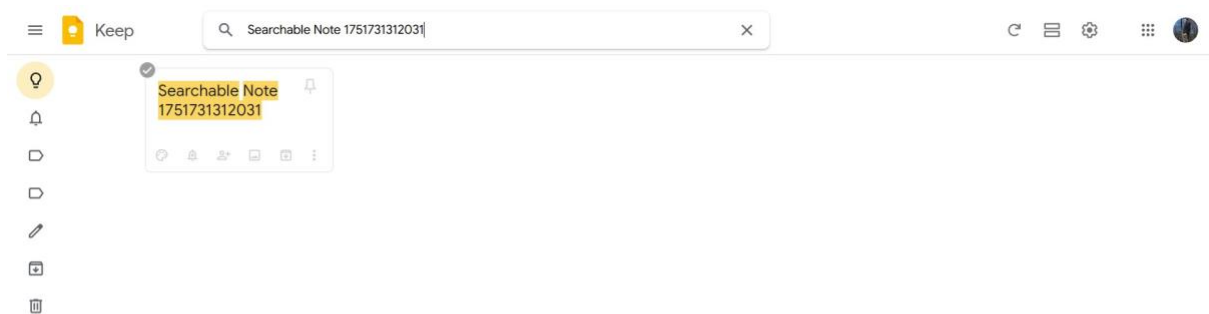
### Создание новой заметки



## Заметка успешно создана

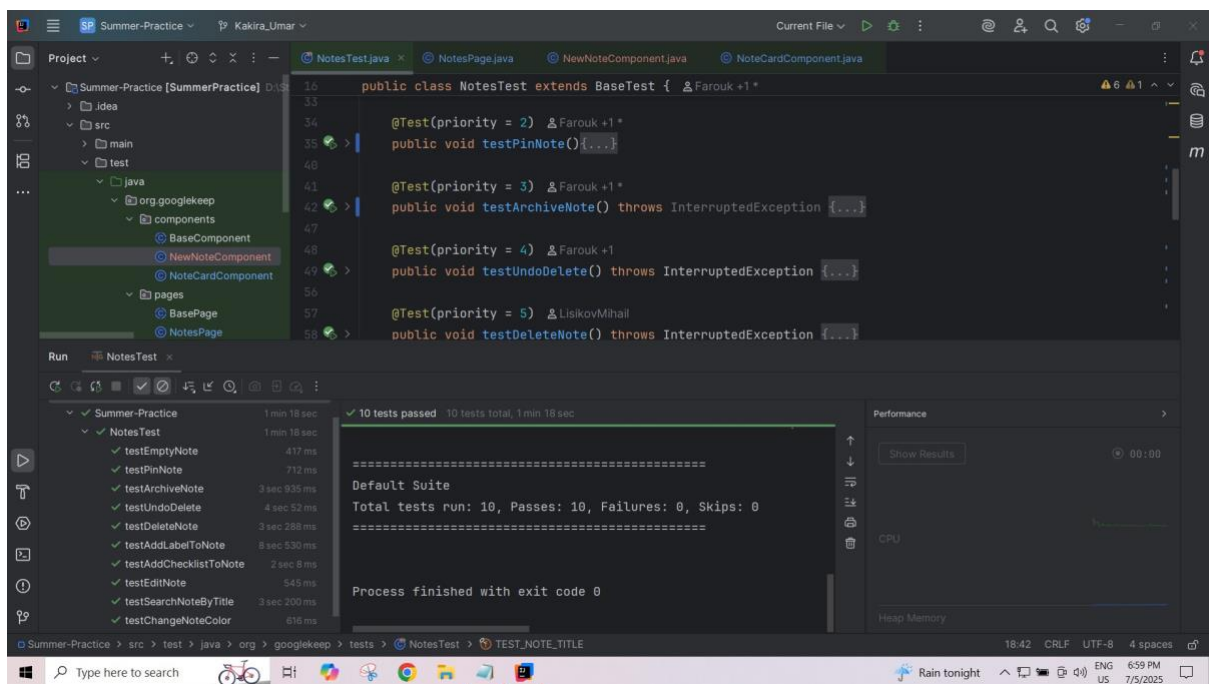


## Поиск созданной заметки и подтверждение её отображения в результатах поиска





## Все тесты успешно пройдены



## ЗАКЛЮЧЕНИЕ

В ходе выполнения данной практической работы была реализована система автоматизированного тестирования пользовательского интерфейса (UI) веб-приложения Google Keep. Цель проекта заключалась в проверке корректности выполнения основных пользовательских операций, связанных с управлением заметками. Для реализации проекта использовался язык программирования Java в связке с библиотекой Selenide, которая расширяет функциональность Selenium и предоставляет более простой и читаемый синтаксис для автоматизации веб-интерфейсов.

В рамках тестирования было разработано 10 тестов, охватывающих основные действия пользователя: создание пустой заметки, закрепление и открепление, архивирование, отмена удаления, редактирование, изменение цвета, поиск заметки, полное удаление, добавление метки, а также добавление чек-листа. Каждый из тестов имитировал поведение реального пользователя, взаимодействующего с приложением через веб-браузер, и проверял, что система корректно обрабатывает соответствующее действие.

Работа была структурирована по архитектуре Page Object Model (POM), что позволило изолировать логику взаимодействия с элементами интерфейса от логики самих тестов. Это значительно упростило сопровождение и расширение проекта, а также повысило читаемость кода. Также была реализована организация кода по архитектурной модели MVC, где основное внимание уделено разделению ответственности между различными компонентами.

Процесс тестирования показал высокую устойчивость интерфейса Google Keep к типовым пользовательским сценариям, однако также выявил некоторые особенности, требующие задержек в тестах (например, при ожидании обновления DOM после создания или редактирования заметки). Эти особенности были решены с помощью методов ожидания, встроенных в библиотеку Selenide.

Таким образом, в рамках выполнения работы была достигнута цель – создание набора автоматизированных UI-тестов для оценки функциональности Google Keep. Полученные знания и навыки в области автоматизации тестирования, работы с фреймворками Selenide и TestNG, а также организация проекта по современным архитектурным подходам могут быть применены в будущем при тестировании других веб-приложений и при разработке собственных программных решений.

Проект также позволил закрепить навыки анализа пользовательского интерфейса, выявления ключевых пользовательских сценариев и построения удобных и масштабируемых тестов. Опыт, полученный в ходе реализации, может служить базой для дальнейшего изучения CI/CD-инструментов, интеграции с системами отчётности и расширения тестового покрытия.