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

ОТЧЕТ

по учебной практике

Тема: UI-тестирование сайта Pinterest

Студенты гр. 3344 Бубякина Ю.В.
Вердин К.К.
Жаворонок Д.Н.

Руководитель Шевелева А.М

Санкт-Петербург
2025

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студенты Бубякина Ю.В., Вердин К.К., Жаворонок Д.Н.

Группа 3344

Тема практики: UI-тестирование сайта Pinterest

Задание на практику:

Создание архитектуры для реализации 10 различных UI-Тестов функционала сайта Pinterest. С использованием таких технологий как Java, Selenide (Selenium), Junit, Maven, логирование. Создание документации к этим тестам

Сроки прохождения практики: 25.06.2025 – 25.06.2025

Дата сдачи отчета: 07.04.2025

Дата защиты отчета: 07.04.2025

Студенты		Бубякина Ю.В. Вердин К.К. Жаворонок Д.Н.
Руководитель		Шевелева А.М

АННОТАЦИЯ

Целью данной практики является освоение автоматизации тестирования веб-приложений с использованием современных технологий: Java, Selenide (на основе Selenium), JUnit для написания тестов и Maven как системы управления проектом. В рамках практики разработано 10 автотестов на определенный функциональный блок выбранной системы (например, работа с постами в социальной сети). Работа ведётся в группах по 3 человека через GitHub, где каждый участник выполняет коммиты и несёт ответственность за определённую часть проекта. Тесты должны быть задокументированы в формате чеклиста с описанием действий, входных данных и ожидаемых результатов. Также предусмотрено логирование выполнения тестов. Итоговый отчет включает описание реализации, UML-диаграммы, демонстрацию работы тестов и заключение по результатам практики

SUMMARY

The goal of this practice is to master the automation of web application testing using modern technologies: Java, Selenide (based on Selenium), JUnit for writing tests, and Maven as a project management system. Within the scope of the practice, 10 automated tests have been developed for a specific functional block of a selected system (for example, working with posts in a social network). The work is carried out in groups of 3 people via GitHub, where each participant makes commits and is responsible for a certain part of the project. The tests must be documented in a checklist format, including descriptions of actions, input data, and expected results. Logging of test execution is also provided. The final report includes an implementation overview, UML diagrams, demonstration of test execution, and a conclusion based on the results of the practice.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель: Разработка автоматизированных UI-тестов для веб-приложения с использованием современных инструментов тестирования.

Задачи:

1.Выбрать систему тестирования

2.Оформить чеклист в виде таблицы с описанием тестов.

3.Организовать работу в GitHub (репозиторий на группу из 3 человек с распределением задач в README.md).

4.Написать 10 автоматизированных тестов и документацию к ним для выбранной системы.

Использовать технологии: Java, Selenide (Selenium), JUnit 5, Maven, логирование.

Для тестирования был выбран сайт Pinterest [1]. Выбор обусловлен тем, что это популярные и функционально насыщенный социальный интернет-сервис, фотохостинг, позволяющий пользователям добавлять в режиме онлайн изображения, помещать их в тематические коллекции и делиться ими с другими пользователями.

1. РЕАЛИЗУЕМЫЕ ТЕСТЫ

1.1. Описание тестов

1. **Проверка авторизации**

Проверяется поведение формы входа при различных вариантах ввода. Тест включает открытие модального окна, попытки входа с неверными данными (неверный пароль, некорректный формат email) и успешную авторизацию. Ожидаемые результаты — отображение соответствующих сообщений об ошибке и переход на главную страницу при корректных данных.

2. **Редактирование профиля**

Проверяется возможность изменения пользовательской информации в настройках профиля. Вводятся новые значения для имени, фамилии и описания, сохраняются, после чего выполняется перезагрузка страницы. Ожидается, что обновлённые данные корректно сохраняются и отображаются.

3. **Открытие и закрытие поста**

Тест проверяет корректность открытия поста из ленты и его закрытия. После клика по посту URL должен соответствовать ссылке элемента. При возврате назад пользователь должен видеть главную страницу без ошибок или перезагрузки.

4. **Поставить и убрать лайк на пост**

Проверяется возможность поставить и убрать лайк у поста. После нажатия на кнопку лайка и перезагрузки страницы меняется SVG-атрибут d элемента path, отражающий состояние кнопки. При повторном нажатии и обновлении страницы атрибут возвращается к исходному значению, что подтверждает отмену лайка.

5. **Сохранение поста в профиль**

Тест проверяет работу кнопки сохранения поста. После нажатия на кнопку её атрибут aria-label должен измениться, отражая новое состояние “Пост сохранен”.

6. **Добавление комментария**

Проверяется возможность оставить комментарий к посту с текстом и прикреплённым изображением. После отправки комментарий должен отображаться под постом с введённым текстом и загруженным изображением.

7. **Скачивание изображения**

Проверяется работа функции скачивания. После нажатия на кнопку «…» и выбора опции «Скачать пост» изображение должно успешно сохраняться на устройство.

8. **Поиск по запросу**

Проверяется корректность работы поиска. При вводе запроса и подтверждении поиска, среди первых 10 изображений хотя бы одно должно содержать в атрибуте aria-label ключевое слово из запроса.

9. **Скрытие поста из ленты**

Тест проверяет возможность скрыть пост по причине «Не интересует». После выбора этой опции появляется уведомление «Пин скрыт», подтверждающее успешное выполнение действия.

10. **Изменение сортировки досок**

Проверяется корректность работы функции сортировки досок в разделе «Сохранённые» профиля пользователя. Тест направлен на подтверждение того, что при выборе в выпадающем меню сортировки критерия «Последний добавленный пин» отображение досок обновляется и соответствует выбранному параметру сортировки.

2. ОПИСАНИЕ КЛАССОВ И МЕТОДОВ, UML ДИАГРАММА

2.1. Описание классов и методов

Для данной работы использовался паттерн Page Object Modal, для реализации которого были созданы классы элементов, страниц и тестов.

Классы элементов

public class BaseElement - Базовый класс для всех элементов пользовательского интерфейса. Инкапсулирует общую функциональность для работы с веб-элементами.

Поля:

protected static final int WAIT_SECONDS - константа, определяющая время ожидания рендера элемента.

protected final SelenideElement baseElement - экземпляр найденного элемента

Методы:

protected BaseElement(String xpath, String attributeValue) - конструктор базового элемента, принимает следующие аргументы: *xPath* - шаблон xPath для поиска элемента *attributeValue* - параметр для подстановки в шаблон Xpath

public boolean isDisplayed() - метод, проверяющий, отображается ли элемент на странице. Ожидает появление элемента в течение стандартного времени ожидания.

public String getAttribute(String attributeName) - метод, возвращающий значение указанного атрибута (*attributeName*) элемента

public class Article extends BaseElement - класс, представляющий элемент статьи на веб-странице, наследуется от базового класса элемента BaseElement. Инкапсулирует функциональность для работы со статьями.

Поля:

private static final String ARIA_LABEL_XPATH - константа. определяющая xPath к элементу по значению атрибута aria-label.

Методы:

public String getHref() - получает значение атрибута href элемента статьи

public String getAriaLabel() - Получает значение атрибута aria-label элемента статьи.

public void click() - выполняет клик по элементу статьи

Статичные фабрики:

public static Article byAriaLabel(String ariaLabel) - Находит элемент статьи по значению атрибута *aria-label*.

public static Article byXpath(String xPath) - Находит элемент статьи по *XPath*.

public class Button extends BaseElement - Класс для работы с элементами типа "кнопка". Предоставляет методы для поиска кнопок по различным атрибутам и выполнения действий над ними. Наследуется от базового класса элемента BaseElement.

Поля:

private static final String ID_XPATH - определяет xPath к элементу кнопки по атрибуту id

private static final String TEXT_XPATH - определяет xPath к элементу кнопки по атрибуту text

private static final String CLASS_XPATH - определяет xPath к элементу кнопки по атрибуту class

private static final String TYPE_XPATH - определяет xPath к элементу кнопки по атрибуту type

private static final String ARIA_LABEL_XPATH - определяет xPath к элементу кнопки по атрибуту aria-label

private static final String DATA_TEST_ID_XPATH - определяет xPath к элементу кнопки по атрибуту data-test-id

Методы:

public void click() - выполняет клик по кнопке.

Фабричные статичные методы:

public static Button byId(String id) - возвращает кнопку по идентификатору (*id*)

public static Button byText(String text) - находит кнопку по тексту внутри неё (*text*)

public static Button byClass(String className) - находит кнопку по названию класса (className)

public static Button byType(String type) - находит кнопку по её типу (type)

public static Button byAriaLabel(String ariaLabel) - Находит кнопку по значению атрибута aria-label.

public static Button byXPath(String xPath) - Находит кнопку по указанному XPath.

public static Button byDataTestId(String dataTestId) - находит кнопку по значению атрибута data-test-id.

public class H2 extends BaseElement - Класс для работы с текстовыми элементами. Предоставляет методы для поиска текстовых элементов и проверки их содержимого.

Поля:

private static final String ID_XPATH - определяет xPath к элементу h2 по значению атрибута id.

private static final String CLASS_XPATH = определяет xPath к элементу h2 по значению атрибута class.

Методы:

public String getText() - возвращает текст элемента.

Фабричные статичные методы:

public static H2 byId(String id) - Находит текстовый элемент по идентификатору.

public static H2 byClass(String className) - Находит текстовый элемент по названию класса.

public static H2 byXPath(String xpath) - Находит H2 по указанному XPath.

public class Image extends BaseElement - Класс для работы с элементами типа "изображение". Предоставляет методы для поиска изображений по различным атрибутам и выполнения действий над ними.

Поля:

private static final String ALT_XPATH

private static final String ELEMENT_TIMING_XPATH

private static final String HREF_ATTRIBUTE

private static final String SRC_ATTRIBUTE

являются константами для шаблонов xPath

Методы:

public void click() - Выполняет клик по изображению.

Фабричные статические методы:

public static Image byAlt(String alt) - Находит изображение по значению атрибута alt.

public class ImagePreview extends BaseElement - класс для работы с превью изображений. Представляет составной элемент, содержащий связанную статью и заголовок.

Поля:

private static final String ARTICLE_XPATH_SUFFIX

private static final String H2_XPATH_SUFFIX

являются константами для xPath суффиксов

private final Article article - элемент статьи;

private final H2 h2 элемент заголовка;

Методы:

public String getPreviewName() - получает заголовок превью изображения

public String getArticleAriaLabel() - Получает значение атрибута aria-label связанной статьи

public String getArticleHref() - Получает ссылку на связанную статью.

public void click() - Выполняет переход по ссылке превью изображения.

Фабричные статические методы:

p*ublic static ImagePreview byXpath(String xPath)* - Находит превью изображения по указанному *XPath*.

public class Input extends BaseElement - Класс для работы с элементами типа "поле ввода". Предоставляет методы для поиска полей ввода по различным атрибутам и выполнения действий над ними.

Поля:

private static final String ID_XPATH

private static final String NAME_XPATH

private static final String CLASS_XPATH

private static final String DATA_TEST_ID_XPATH

являются константами для xPath суффиксов

Методы:

public void clear() - Очищает поле ввода.

public void fill(String value) - Вводит указанное значение в поле ввода.

public void load(File file) - метод, позволяющий загружать файлы

public void click() - Кликает на элемент.

public void pressEnter() - Нажимает Enter для подтверждения.

Фабричные статичные методы:

public static Input byId(String id) - Находит поле ввода по идентификатору id.

public static Input byDataTestId(String id) - Находит поле ввода по dataTextId.

public static Input byName(String name) - Находит поле ввода по имени name.

public class LikeButton extends Button - Класс для работы с кнопкой "лайк". Расширяет функциональность базовой кнопки, добавляя специфичные для кнопки методы.

Поля:

private static final String PATH_XPATH_SUFFIX

private static final String DATA_TEST_ID_XPATH

являются константами для xPath суффиксов

private final Path path - элемент типа path

Методы:

public String getPathD() - Получает значение атрибута 'd' элемента path внутри кнопки. Используется для проверки состояния лайка (заполненный/пустой).

Фабричные статические методы:

public static LikeButton byDataTestId(String dataTestId) - Находит кнопку лайка по значению атрибута data-test-id.

public class MansoryContainer extends BaseElement - Класс для работы с Masonry-контейнером изображений. Представляет контейнер, содержащий превью изображений в виде сетки.

Поля:

private static final String CLASS_XPATH

private static final String IMAGE_PREVIEW_XPATH

являются константами для xPath суффиксов

Методы:

public String getHrefOfFirstImage() - Получает ссылку на статью первого превью изображения в контейнере.

public ImagePreview getNthImagePreview(int n) - Получает превью изображения по порядковому номеру.

public void clickOnImage() - Выполняет клик по первому превью изображения в контейнере.

public String getPreviewName() - Получает название первого превью изображения в контейнере.

public String getNthArticleAriaLabel(int n) - Получает значение атрибута aria-label статьи для n-го превью.

Фабричные статические методы:

public static MansoryContainer byClass(String className) - Находит Masonry-контейнер по названию класса.

public class Path extends BaseElement - Класс для работы с элементами SVG-пути. Предоставляет методы для взаимодействия с тегом <path/> в SVG-графике.

Методы:

public String getD() - Получает значение атрибута 'd' элемента пути. Атрибут 'd' содержит команды для построения SVG-пути.

Фабричные статические методы:

public static Path byXpath(String xPath) - Находит элемент пути по указанному XPath.

public class SearchHeader extends BaseElement - Класс для работы с шапкой поиска. Инкапсулирует функциональность элементов поисковой строки и кнопок в шапке приложения.

Поля:

private static final String CLASS_XPATH - константа, определяющая xPath по названию класса

private final Input searchInput - объект класса Input.

private final Button accountButton - кнопка аккаунта.

Методы:

public void openSearchBar() - Активирует поле поиска, выполняя по нему клик.

public void fillSearchText(String text) - Вводит текст в поле поиска.

public void pressEnter() - Выполняет поиск, нажимая Enter в поле поиска.

public void clickAccountButton() - Нажимает на кнопку аккаунта

Фабричные статические методы:

public static SearchHeader byId(String text) - Находит шапку поиска по идентификатору.

public class Text extends BaseElement - Класс для работы с текстовыми элементами. Предоставляет методы для поиска текстовых элементов и проверки их содержимого.

Поля:

private static final String ID_XPATH

private static final String CLASS_XPATH

private static final String TEXT_XPATH

являются константами для xPath суффиксов

Фабричные методы:

public static Text byId(String id) - Находит текстовый элемент по идентификатору.

public class TextArea extends BaseElement - Класс для работы с текстовыми элементами. Предоставляет методы для поиска текстовых элементов и проверки их содержимого.

Поля:

private static final String ID_XPATH

являются константами для xPath суффиксов

Методы:

public String getText() - Возвращает текст элемента.

public void fill(String value) - вводит текст в элемент

Фабричные методы:

public static TextArea byId(String id) - Находит текстовый элемент по идентификатору.

Классы страниц

public class BasePage - Базовый класс для всех страниц приложения. Инкапсулирует общую функциональность работы со страницами.

Поля:

private static final String BASE_ELEMENT_XPATH - базовый xpath страницы

protected final SelenideElement basePage - элемент страницы

protected final Class<? extends BasePage> pageClass - класс страницы

Методы:

protected BasePage(Class<? extends BasePage> pageClass, String uniqueElementIdentifier) - Конструктор базовой страницы.

pageClass - класс страницы (для рефлексии)

uniqueElementIdentifier - уникальный идентификатор элемента страницы

public <T extends BasePage> T refresh(Class<T> pageClass) - Обновляет текущую страницу. *pageClass* - класс страницы, которую нужно вернуть, *<T>* тип страницы, возвращает новый экземпляр обновлённой страницы

public <T extends BasePage> T page(Class<T> pageClass) - Создаёт экземпляр страницы указанного класса. *pageClass* - класс страницы, *<T>* - тип страницы

public class BoardsPage extends BasePage - Страница досок пользователя. Предоставляет функционал для работы с досками и их сортировкой.

Поля:

private final Button moreOptionsButton - элемент кнопки “показать ещё”.

private final Button alphabeticOrderButton - элемент кнопки сортировки по алфавиту.

private final Button lastAddedOrderButton - элемент кнопки сортировки по дате добавления.

private final MansoryContainer mansoryContainer - элемент контейнера с картинками.

Методы:

public void openSortingOptions() - Открывает меню дополнительных опций сортировки.

public void sortAlphabetically() - Выбирает сортировку досок по алфавиту

public void sortByLastAdded() - Выбирает сортировку досок по дате добавления (последние добавленные).

public String getFirstBoardName() - Получает название первого превью доски в контейнере.

public class LoginPage extends BasePage - Страница аутентификации пользователя. Предоставляет методы для взаимодействия с элементами входа.

Поля:

private final Button loginModalButton - кнопка открытия модал окна логина.

private final Input loginInput - поле ввода логина

private final Input passwordInput - поле ввода пароля

private final Button submitButton - кнопка входа в аккаунт

private final Text incorrectPasswordText - текст, появляющийся при неверном пароле

private final Text incorrectEmailFormatText - текст, появляющийся при неверном формате e-mail.

Методы:

public <T extends BasePage> T openLoginModal(Class<T> nextPageClass) - Открывает модальное окно аутентификации. *nextPageClass* - класс возвращаемой страницы, *<T>* - тип страницы

public <T extends BasePage> T enterLogin(String login, Class<T> nextPageClass) - Вводит логин в поле ввода.

login логин пользователя

nextPageClass класс возвращаемой страницы

<T> тип страницы

public <T extends BasePage> T enterPassword(String password, Class<T> nextPageClass) - Вводит пароль в поле ввода.

password - пароль пользователя

nextPageClass класс возвращаемой страницы

<T> тип страницы

public <T extends BasePage> T clickLoginButton(Class<T> nextPageClass) - Нажимает кнопку входа.

nextPageClass - класс возвращаемой страницы

<T> - тип страницы

public boolean checkIsIncorrectPasswordMessageDisplayed() - Проверяет отображение сообщения о неверном пароле.

public boolean checkIsIncorrectEmailMessageDisplayed() -Проверяет отображение сообщения о неверном формате email.

public class MainPage extends BasePage - главная страница, на которую пользователь переходит после аутентификации

Поля:

private final Button undoActionButton - кнопка отмены действия

private final MansoryContainer mansoryContainer - элемент mansory контейнера

private final SearchHeader searchHeader - элемент поискового хэдэра

private final Button settingsModalButton - элемент кнопки, открывающей настройки

Методы:

public boolean isDisplayed() - Проверяет отображение главной страницы.

public String getHrefOfFirstImage() - Получает ссылку на статью первого изображения.

public <T extends BasePage> T clickOnFirstImage(Class<T> className) - Открывает статью первого изображения. Возвращает экземпляр страницы className

public void openSearchBar() - Активирует поле поиска.

public void fillSearchBar(String searchText) - Вводит текст в поле поиска.

public void submitSearch() - Выполняет поиск.

public void waitForSearchPageLoad(String expectedUrlPart) - Ожидает загрузку страницы по частичному совпадению URL.

public <T extends BasePage> T clickAccountButton(Class<T> className) - Открывает страницу аккаунта.

public String getNthAriaLabel(int n) - Получает значение атрибута aria-label для n-го элемента.

public void clickSettingsModalButton() - Открывает меню настроек.

public <T extends BasePage> T clickSettings(Class<T> className) - Открывает страницу настроек.

public boolean undoDisplayed() - Проверяет, появилась ли возможность отменить действие скрытия

public class PinPage extends BasePage - страница, отображающая пин(пост)

Поля

private final Button backButton - кнопка перехода на главную страницу

private final Button moreOptionsButton - кнопка дополнительных опций

private final Button hidePinButton - кнопка скрытия пина

private final Button notInterestedButton - кнопка “не интересно”

private final LikeButton likeButton - кнопка лайка

private final Button downloadPinButton - кнопка скачивания фото пина

private final Button saveButton - кнопка сохранения пина к себе

private final Button alreadySavedButton - кнопка отмены сохранения пина

private final Button choosePhotoButton - кнопка выбора фото для комментария

private final Input loadPhotoInput - поле выбора фото для комментария

private final SelenideElement commentTextInput - поле для ввода текста комментария

private final Button sendCommentButton - кнопка отправки комментария

private final H2 commentCount - поле, показывающее количество комментариев

public String getCurrentUrl() - Получает ссылку на эту страницу - изображение пина.

public <T extends BasePage> T clickBackButton(Class<T> className) - Возвращается на предыдущую страницу.

public void waitForPinPageLoad(String expectedUrlPart) - Ожидает загрузку страницы пина по частичному совпадению URL

public void clickMoreOptionsButton() - Открывает меню дополнительных опций.

public void clickHidePinButton() - Скрывает текущий пин.

public void clickChoosePhotoButton - Открывает окно загрузки изображения.

public <T extends BasePage> T clickNotInterestedButton(Class<T> className) - Отмечает пин как "Не интересно".

public void clickLikeButton() - Ставит или снимает лайк с поста.

public String getDLikeButton() - Получает данные SVG-пути для кнопки лайка

public void fillText(String text) - Вводит текст комментария.

public void clickSendCommentButton() - Отправляет комментарий.

public void loadPhoto(File file) - Загружает фото.

public void clickDownloadPinButton() - Скачивает текущий пин.

public void clickSaveButton() - Сохраняет текущий пин.

public boolean isPostSaved() - Проверяет, сохранен ли пин

public String getCommentCount() - Возвращает кол-во комментариев.

public class SettingsPage extends BasePage - страница настроек и редактирования информации о пользователе

Поля:

private final Input firstNameInput - поле ввода имени пользователя

private final Input lastNameInput - поле ввода фамилии пользователя

private final TextArea aboutTextArea - поле ввода информации о себе

private final Button saveButton - кнопка сохранения информации

private final Button cancelButton - кнопка сброса введённых данных

Методы:

public void fillFirstName(String firstName) - Вводит имя пользователя.

public void fillLastName(String lastName) - Вводит фамилию пользователя.

public void fillAbout(String about) - Вводит информацию "О себе".

public void clickSaveButton() - Сохраняет изменения профиля.

public String getFirstName() - Получает текущее значение имени пользователя.

public String getLastName() - Получает текущее значение фамилии пользователя.

public String getAbout() - Получает текущую информацию "О себе"

Классы тестов

public class BaseTest - Базовый класс для UI тестов. Содержит общие настройки для всех тестовых классов.

Методы:

@BeforeEach

public void setUp() - Конфигурация тестового окружения перед каждым тестом. Устанавливает параметры браузера, таймауты и базовые настройки.

@AfterEach

public void tearDown() - Закрывает веб-драйвер после каждого теста.

protected MainPage auth(String login, String password) - Выполняет аутентификацию пользователя.

private void configureBrowser() - настраивает параметры браузера.

private void configureLogging() - Настраивает систему логирования.

protected void initTestData() - Инициализирует тестовые данные перед каждым тестом. Загружает учетные данные из .env файла.

public class CommentTest extends BaseTest - Тестовый класс для проверки функциональности комментирования пинов. Проверяет возможность добавления комментария с текстом и изображением.

Поля:

private static final String COMMENT_TEXT - текст комментария

private static final String IMAGE_PATH - путь к изображению

Методы:

@Test

void shouldAddCommentWithTextAndImage() - Проверяет возможность добавления комментария с текстом и изображением.

public class EditProfileTest extends BaseTest - Тестовый класс для проверки функциональности редактирования профиля пользователя. Проверяет возможность изменения имени, фамилии и информации "О себе".

Поля:

private static final String NAME_SUFFIX - суффикс который будет добавлен к имени

private static final String SURNAME_SUFFIX - суффикс который будет добавлен к фамилии

private static final String ABOUT_TEXT - суффикс который будет к полю “О себе”

Методы:

@Test

public void shouldSuccessfullyEditUserProfile() - Проверяет возможность редактирования профиля пользователя.

public class HideTest extends BaseTest - Тестовый класс для проверки функциональности скрытия пинов. Проверяет возможность скрытия пина и отмены этого действия.

Методы:

@Test

public void checkHidePost() - Проверяет функциональность скрытия пина и отмены этого действия.

public class LikeTest extends BaseTest - Тестовый класс для проверки функциональности лайков. Проверяет постановку и снятие лайка, а также сохранение состояния после обновления страницы.

Методы:

public void shouldToggleLikeAndPersistStateAfterRefresh() - Проверяет функциональность лайка

public class LoginTest extends BaseTest - Тестовый класс для проверки функциональности аутентификации. Проверяет сценарии входа с неверными и верными учетными данными.

Поля:

private static final String FAKE_PASSWORD - неверный пароль

private static final String INCORRECT_FORMAT_EMAIL - неверный email

Методы:

public void checkAuth() - проверяет авторизацию

public class OpenClosePostTest extends BaseTest - Тестовый класс для проверки функциональности открытия и закрытия постов. Проверяет корректность перехода на страницу поста и возврата на главную страницу.

Методы:

public void shouldOpenPostAndReturnToMainPage() - проверяет функциональность открытия и закрытия поста

public class SavePostTest extends BaseTest - Тестовый класс для проверки функциональности сохранения пина.

Методы:

public void savePost() - проверяет сохранился ли пост

public class SearchTest extends BaseTest - Тестовый класс для проверки функциональности поиска. Проверяет поиск по запросу.

Поля:

private static final String SEARCH_STR - строка, которая вводится в поле поиска

private static final List<String> SEARCH_KEYWORDS - слова, по которым проверяется, был ли произведён поиск корректно

Методы:

public void checkSearch() - проверяет поиск по запросу

public class SortingTest extends BaseTest - Тестовый класс для проверки функциональности сортировки досок. Проверяет сортировку по нажатию.

Методы:

public void checkSort() - проверяет сортировку досок

public class UploadTest extends BaseTest - Тестовый класс для проверки функциональности загрузки фото.

Методы:

public void checkUpload() - метод, проверяющий скачивается ли фото.

2.2. UML Диаграмма

При помощи плагинов и среды разработки INTELLIJ IDEA была создана UML диаграмма классов (см. рис. 2.2)

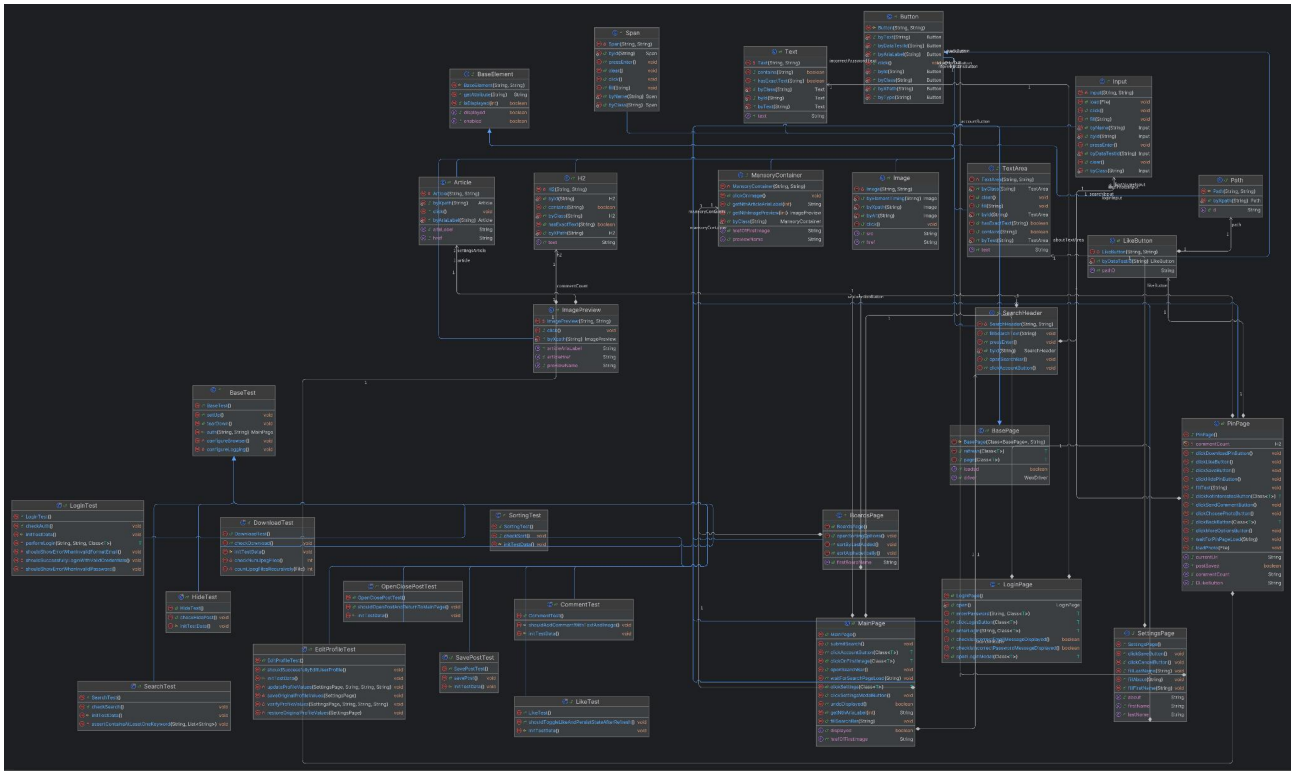

Рис. 2.2 - UML диаграмма

3. ТЕСТИРОВАНИЕ

3.1. Работа тестов

Выполнение теста shouldAddCommentWithTextAndImage

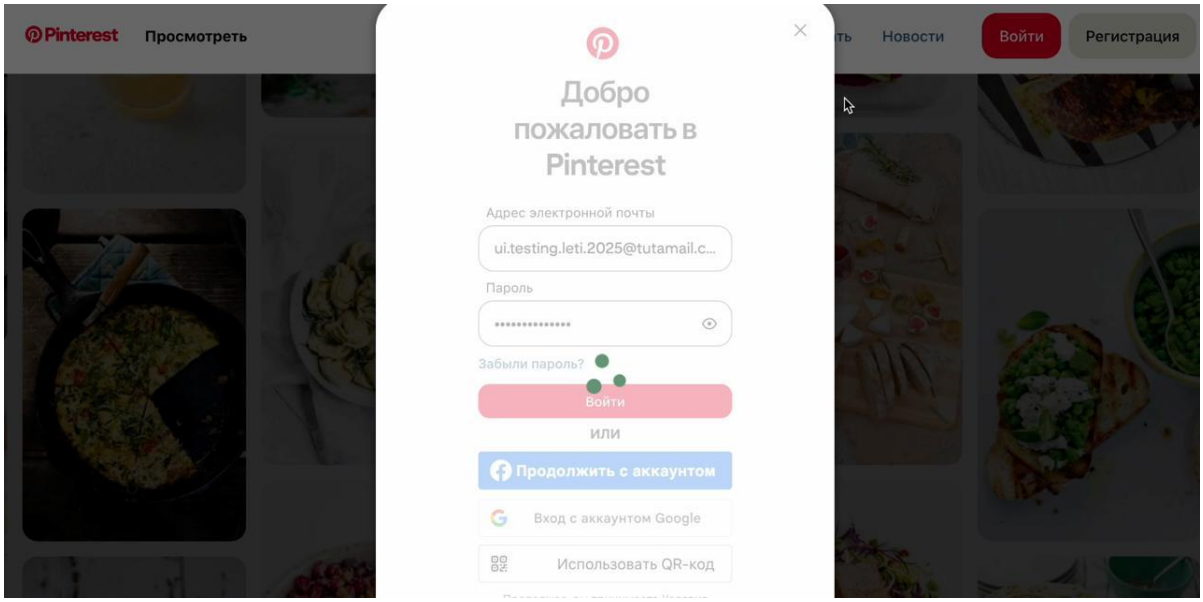

Рис. 3.1.1 - Авторизация на сайте

Наблюдаем, как пользователь вводит данные для входа в систему: логин и пароль. После ввода происходит нажатие на кнопку “Войти”, что инициирует процесс авторизации (см.рис.3.1.1).

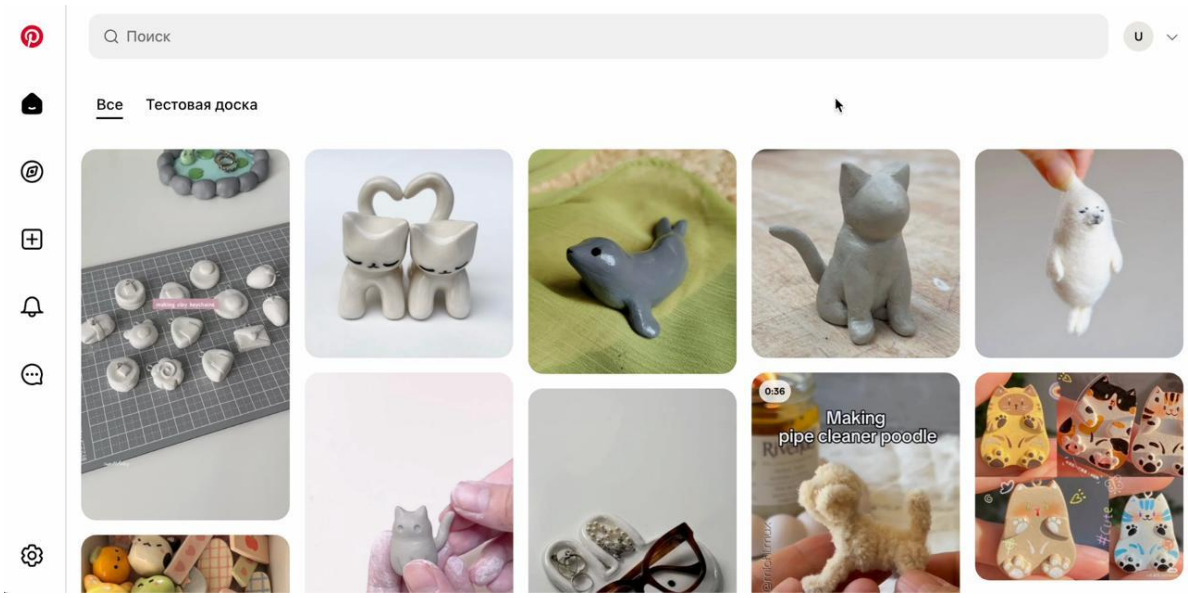

Рис. 3.1.2 - Главная страница сайта

После успешной авторизации пользователь попадает на главную страницу сайта Pinterest (см.рис.3.1.2). Наблюдаем отображение ленты с постами — визуальным контентом, доступным для просмотра, взаимодействия и выбора для дальнейших действий.

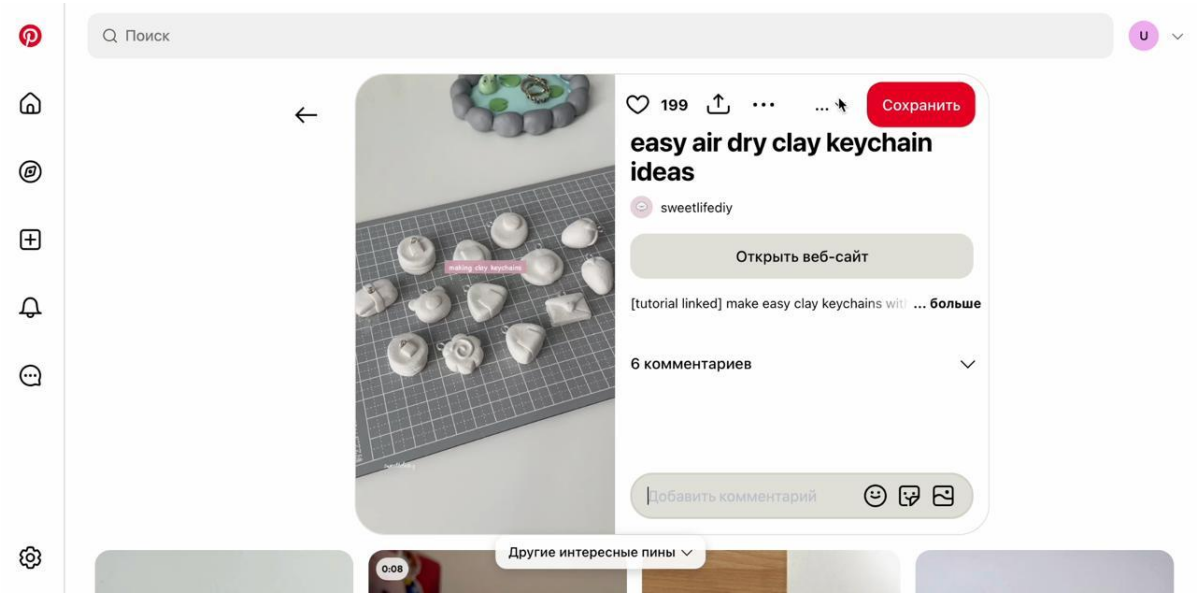

Рис. 3.1.3 - Страница первого поста с главной

Программа тестирования выбирает первый пост из ленты на главной странице (см.рис.3.1.3). Наблюдаем переход на страницу поста, где отображается его подробное содержимое и доступны элементы для взаимодействия, включая поле добавления комментария.

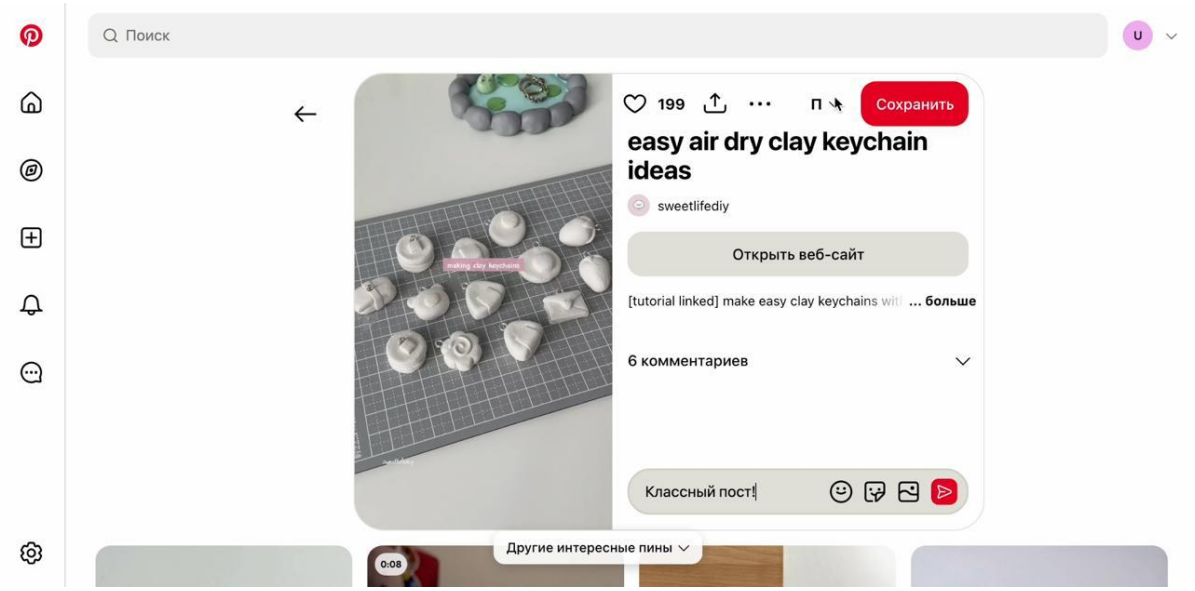

Рис. 3.1.4 - Ввод комментария

Программа тестирования находит поле для ввода комментария и автоматически вписывает текст: “Классный пост!”. Наблюдаем процесс симуляции пользовательского ввода (см.рис.3.1.4)

.

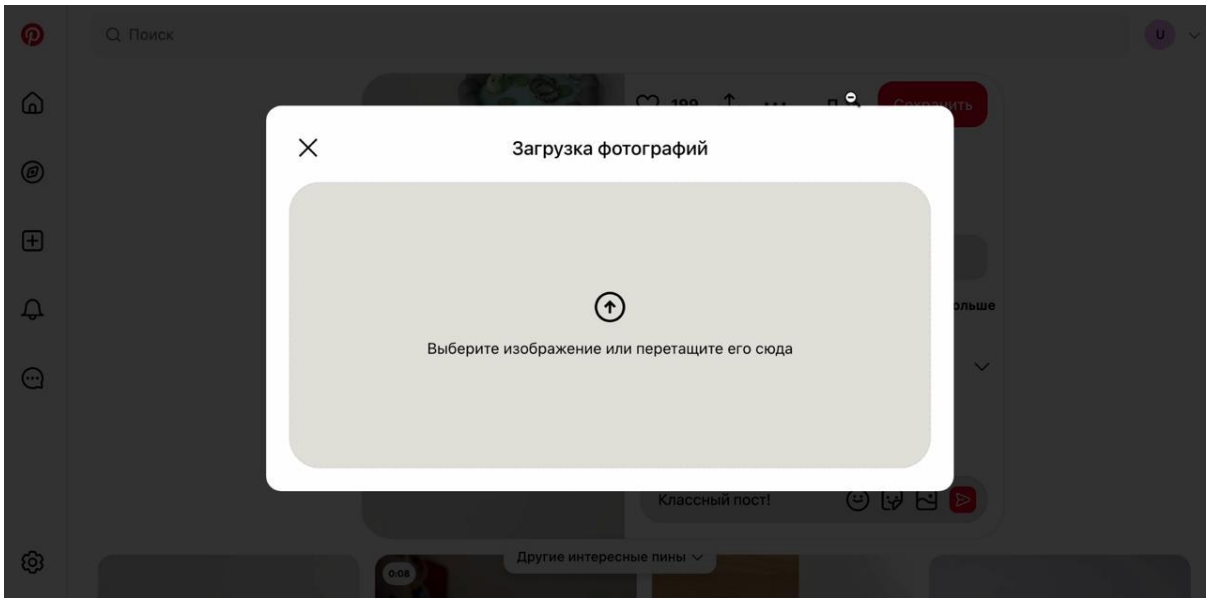

Рис. 3.1.5 - Окошко загрузки изображения

Программа тестирования инициирует действие по открытия окна для прикрепления изображения к комментарию (см.рис.3.1.5).

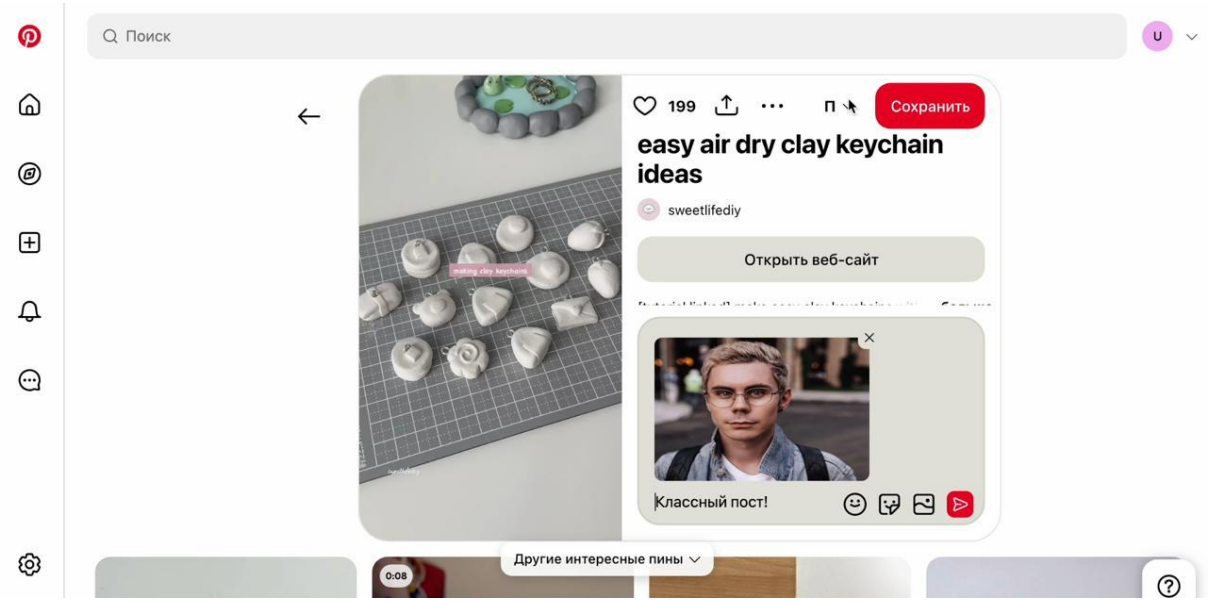

Рис. 3.1.6 - Изображение и текст прикреплены к комментарию

Окно загрузки закрывается, и программа тестирования проверяет, что изображение успешно прикреплено к комментарию вместе с текстом (см.рис.3.1.6). Наблюдаем отображение загруженного изображения и введённого текста в поле комментария.

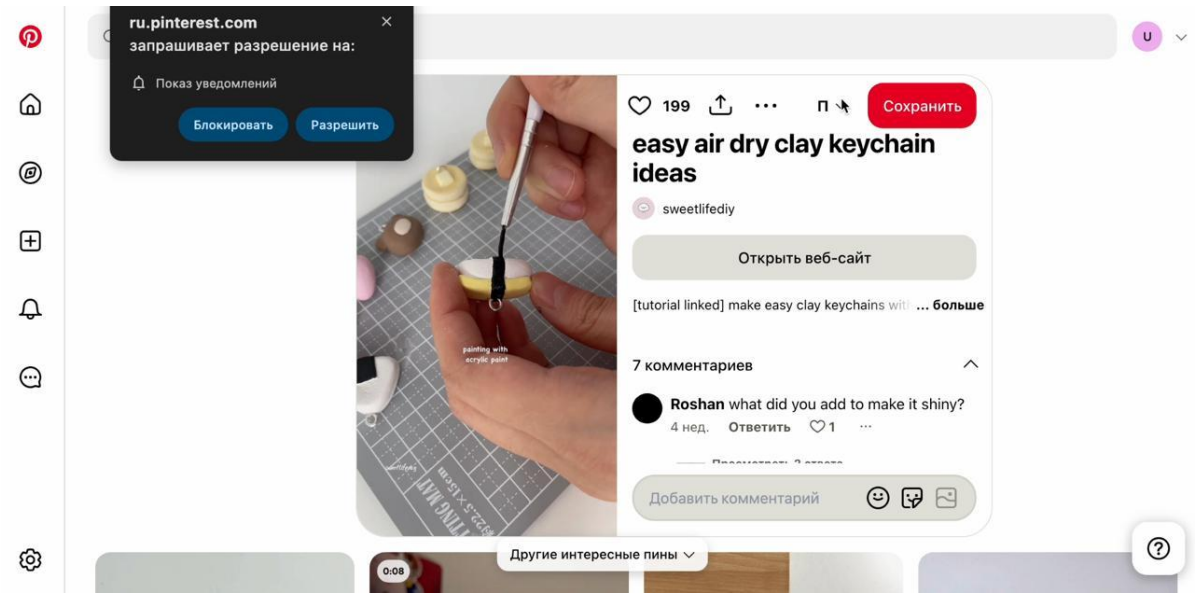

Рис. 3.1.7 - Отправка комментария, изменение количества комментариев у поста

Программа тестирования нажимает кнопку отправки комментария (см.рис.3.1.7). Наблюдаем, как количество комментариев у поста увеличивается на один, подтверждая успешное добавление нового комментария с текстом и изображением.

ЗАКЛЮЧЕНИЕ

В ходе летней практики по автоматизации UI-тестирования веб-приложения Pinterest были достигнуты все поставленные цели и успешно решены задачи:

Разработка и запуск автотестов. Были созданы 8 автоматизированных тестов на основе Java, Selenide и JUnit 5, покрывающих ключевые пользовательские сценарии: авторизация, поиск и фильтрация товаров, работа с корзиной, управление избранным и создание смет. Каждый тест оформлен в виде чеклиста с подробным описанием шагов, входных данных и ожидаемых результатов.

Применение современных инструментов. Для управления проектом использовался Maven. Логирование выполнения тестов обеспечило наглядность отчётов и упростило отладку.

Проектирование структуры. Были реализованы удобные Page Object и элементные абстракции, что повысило читаемость и поддерживаемость кода. UML-диаграмма отразила архитектуру и помогла систематизировать классы и их взаимодействия.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Пинтерест: Pinterest [Электронный ресурс] / Pinterest Inc. - URL: https://ru.pinterest.com/
2. UI-Testing Github Repository [Электронный ресурс]. - URL: https://github.com/VerdinKirill/UI-testing.git