**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по учебной практике**

Тема: **Тестирование сайта Pinterest**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3384 |  | Конасова Я.О. |
| Студентка гр. 3384 |  | Прокопович Я.Г. |
| Студент гр. 3384 |  | Хальзев А.И. |
| Руководитель |  | Шевелева А.М. |

Санкт-Петербург

2025

**ЗАДАНИЕ**

**на учебную практику**

|  |  |  |
| --- | --- | --- |
| Студентка Конасова Я.О. группы 3384 | | |
| Студентка Прокопович Я.Г. группы 3384 | | |
| Студент Хальзев А.И. группы 3384  Тема практики: Тестирование сайта Pinterest | | |
| Задание на практику: Написать 10 тестов для взаимодействия с пинами в Pinterest. | | |
| Сроки прохождения практики: 25.06.2024 – 08.07.2024 | | |
| Дата сдачи отчета: 05.07.2024 | | |
| Дата защиты отчета: 05.07.2024 | | |
|  | | |
| Студентка |  | Конасова Я.О. |
| Студентка |  | Прокопович Я.Г. |
| Студент |  | Хальзев А.И. |
| Руководитель |  | Шевелева А.М. |

**Аннотация**

Задание направлено на развитие навыков автоматизированного тестирования веб-приложений с использованием Java, библиотеки Selenide, фреймворка JUnit и системы сборки Maven. В качестве объекта тестирования выбрана система Pinterest - сервис для визуального поиска и организации контента.

В рамках работы реализован набор из десяти тестов, охватывающих основные пользовательские сценарии по созданию, управлению и взаимодействию с пинами, включая поиск по ключевым запросам. Каждый тест подробно документирован в виде чеклиста с входными данными, пошаговым описанием действий и критериями ожидаемого результата.

**Summary**

The assignment is aimed at developing the skills of automated testing of web applications using Java, the Selenide library, the JUnit framework and the Maven build system. The Pinterest system, a service for visual search and content organization, was chosen as the testing object.

As part of the work, a set of ten tests has been implemented, covering the main user scenarios for creating, managing, and interacting with pins, including keyword search. Each test is documented in detail in the form of a checklist with input data, a step-by-step description of actions and criteria for the expected result.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Реализуемые тесты | 6 |
| 2. | Описание классов и методов элементов страниц | 0 |
| 2.1. | BaseElement | 0 |
| 2.2. | Button | 0 |
| 2.3. | FileInput | 0 |
| 2.4. | Input | 0 |
| 3. | Описание классов и методов страниц | 0 |
| 3.1.  3.2.  3.3.  3.4.  3.5  4. | BasePage  AccountPage  HomePage  LoginPage  PinCreationPage  Тестирование | 0  0  0  0  0  0 |
|  | Заключение | 0 |
|  | Список использованных источников | 0 |

**введение**

Целью данной работы является создание набора автоматизированных UI‑тестов для веб‑приложения с использованием Java, Selenide, JUnit и Maven, обеспечивающего надёжную проверку пользовательских сценариев и поддерживающего единый стандарт документирования и логирования результатов прогонов.

В рамках задачи предстоит разработать и описать подробные тестовые сценарии для десяти ключевых функций системы: создание пина с загрузкой изображения, сохранение чужого пина в собственную доску, перемещение пина между досками, скрытие пина, редактирование описания пина, добавление комментария к пину, добавление пина в избранное, лайк пина, отправка пина через сообщение и поиск пина по запросу.

**1. реализуемые тестЫ**

В данном разделе представлен подробный чек‑лист функционального тестирования ключевых пользовательских сценариев работы с «пином». Для каждого тестового случая задается последовательность действий и ожидаемый результат на уровне интерфейса и логики. Такой подход обеспечивает всестороннюю проверку критичных функций без избыточных описаний реализации, позволяет оперативно сопоставлять фактическое поведение приложения с требуемыми результатами, упрощает интеграцию тестов и сохраняет прозрачность и воспроизводимость проверок, что в итоге позволяет быстро выявлять отклонения и гарантировать стабильность пользовательского опыта.

Реализуемые тесты представлены в табл. 1.

Таблица 1 – Реализуемые тесты

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название теста | Описание теста | Ожидаемый результат |
| 1 | Создание  пина | 1. Авторизация 2. Нажать **“Создать»**  3. Загрузить изображение, нажав на поле «Выберите файл или перетащите его сюда»  4. Ввести заголовок «Тестовый пин» 5. Нажать **«Опубликовать»**  6. Проверить, опубликовалось ли изображение при помощи отслеживания появления надписи: «Ваше изображение опубликовано!»  7. Перейти в режим просмотра опубликованного изображения. | Появляется новый пин с загруженным изображением и указанным заголовком |
| 2 | Сохранение чужого пина в свою доску | 1. Авторизация  2. Нажать на «Профиль», открывается раздел «Ваши сохраненные идеи»  3. Нажать **кнопку «Создать»**  **4. Нажать на кнопку «Доска»**  **5.** Ввести название «Первая доска»  6. Нажать на кнопку «Создать»  7. Нажать на кнопку «Сохранить» на первом пине на главной странице. 8. Проверить, действительно ли изображение сохранилось, отслеживая появление надписи «Сохранено!» | Появляется уведомление о сохранении пина в нужной доске |
| 3 | Перемещение пина между досками | 1. Авторизация  2. Нажать на «Профиль», открывается раздел «Ваши сохраненные идеи» 3. Нажать **кнопку «Создать»** 4. **Нажать на кнопку «Доска»**  5. Ввести название «Вторая доска»  6. Нажать на кнопку «Создать»  7. Нажать на кнопку «Сохранить» на первом пине в предложенных изображениях для сохранения.  8. Проверить, действительно ли изображение сохранилось, отслеживая появление надписи «Сохранено!»  9. Выбрать поле «Вторая доска»  10. В поле «Поиск» ввести «первая доска»  11. Нажать кнопку «Сохранить»  12. Отследить появление надписи «Сохранено» и «Первая доска»  13. Перейти в доску «Первая доска» | Пин исчезает из исходной доски и появляется в целевой |
| 4 | Скрыть пин | 1. Авторизация  2. Нажать на первый пин в ленте  3. Нажать на кнопку «Другие действия»  4. Нажать на кнопку «Скрыть пин»  5. Нажать на кнопку «Другое»  6. Проверить, действительно ли пин скрыт, отслеживая появление кнопки «Отмена» | Пин заменяется на кнопку для отмены скрытия |
| 5 | Редактирование описания пина | 1. Авторизация  2. Нажать на «Профиль», открывается раздел «Ваши сохраненные идеи» 3. Нажатие на раздел «Пины»  4. Нажатие на раздел «Созданные мной»  5. Выбор первого пина  6. Нажать на кнопку «Другие действия»  7. Нажать на кнопку «Изменить пин»  8. Ввод фразы «Этот пин красивый» в поле «Описание»  9. Нажатие на кнопку «Сохранить»  10. Проверка соответствия поля описания ранее введенной надписи | При просмотре пина отображается новое описание |
| 6 | Добавление комментария к пину | 1. Авторизация  2. Нажать на «Профиль», открывается раздел «Ваши сохраненные идеи»  3. Нажатие на раздел «Пины»  4. Нажатие на раздел «Созданные мной»  5. Нажать на первый пин  6. Ввести комментарий в поле «Добавить комментарий»  7. Нажать на кнопку «Эмодзи»  8. Нажать на эмодзи  9. Нажать на кнопку «Опубликовать»  10. Проверить, действительно ли был добавлен комментарий, путем отслеживания его появления под пином | Комментарий появляется под пином |
| 7 | Занесение пина в избранное | 1. Авторизация  2. Нажать на «Профиль», открывается раздел «Ваши сохраненные идеи»  3. Нажать на раздел «Доски»  4. Нажать на первую доску в разделе  5. Нажать на первый пин в доске  6. Нажать на кнопку «Добавить в избранное» 7. Проверка, добавилось ли изображение в избранное, путем отслеживания появления надписи «Удалить из избранного»  8. Переход обратно в раздел «Ваши сохраненные идеи»  9. Нажать на раздел «Пины»  10. Выбор раздела «Избранное» | Пин отображается в разделе «Избранное» |
| 8 | Лайк пина | 1. Авторизация  2. Нажать на первый пин в ленте 3. Нажать на кнопку «Отреагировать»  4. Проверить, действительно ли был поставлен лайк, путем отслеживания появления надписи «Мне нравится!» | Иконка сердца закрашена, число лайков увеличилось |
| 9 | Поделиться пином через сообщение | 1. Авторизация  2. Нажать на первый пин в ленте 3. Нажать **на кнопку «Общий доступ»**  4. Ввести в поле «Поиск по имени» имя пользователя «axalzev»  5. Нажать на кнопку «Отправить»  6. Проверить, действительно ли был отправлен пин, путем отслеживания появления кнопки «См. чат»  7. Нажатие кнопки «См. чат» | Пин отправлен получателю |
| 10 | Поиск определенного пина по запросу | 1. Авторизация  2. Ввести запрос в поле «Поиск» 3. Нажать Enter  4. Проверить, действительно ли были найдены пины по запросы, путем просматривания полученных пинов | Появляются пины, релевантные запросу |

**2. описание классов и методов Элементов страниц**

**2.1. BaseElement**

Класс BaseElement представляет собой основу для создания объектов, которые инкапсулируют элементы пользовательского интерфейса в автоматизированных UI-тестах, написанных с использованием библиотеки Selenide.

Конструктор класса BaseElement(SelenideElement element) принимает объект типа SelenideElement, который представляет собой конкретный элемент на веб-странице. Этот объект сохраняется в защищённое финальное поле base, чтобы использовать его в дальнейших действиях. Благодаря этому любой наследник класса BaseElementполучает доступ к этому элементу и может выполнять с ним операции.

Метод isDisplayed(String phrase) используется для проверки, отображается ли элемент на странице в течение заданного тайм-аута. Он логирует фразу, переданную в параметре phrase, чтобы зафиксировать, какую проверку производит в данный момент. Затем он вызывает метод shouldBe(Condition.visible, Duration.ofSeconds(8)) на внутреннем элементе base, чтобы убедиться, что элемент стал видимым в течение 8 секунд. Если элемент действительно появляется и становится видимым, метод возвращает true. Если же в течение этого времени элемент не появляется, генерируется исключение, которое перехватывается в блоке catch, и метод возвращает false, тем самым сигнализируя о том, что элемент не был отображён.

Метод getElement() возвращает внутренний элемент base. Он может использоваться в подклассах, когда нужно напрямую взаимодействовать с элементом, например, кликнуть по нему или получить его текст.

Метод byXpath(String xpath) позволяет быстро и удобно получить элемент на странице по его XPath. Он использует встроенную функцию Selenide $x(xpath) и возвращает найденный SelenideElement. Этот метод может быть полезен для создания новых экземпляров BaseElement или других обёрток, предоставляя простой способ поиска элементов без необходимости дублировать логику получения через XPath в каждом классе.

Поле logger позволяет логировать действия, относящиеся к конкретному классу, создающему экземпляр BaseElement. Это делает логирование более информативным, так как в логах можно увидеть, в каком именно классе происходили те или иные проверки и действия.

Таким образом, класс BaseElement создаёт единый подход к работе с веб-элементами в тестах, обеспечивая их удобную и безопасную проверку, доступ к элементу и логирование действий. Он предназначен для наследования другими элемент-классами, представляющими более конкретные части интерфейса.

**2.2. Button**

Класс Button представляет собой обёртку над элементом типа SelenideElement, предназначенную для работы с кнопками на веб-странице. Он наследуется от базового класса BaseElement, который предоставляет базовые механизмы взаимодействия с элементами и логгирования. Основная цель класса - унифицировать доступ к кнопкам по разным селекторам (xpath, id, атрибутам и др.).

Конструктор Button(SelenideElement element) инициализирует объект кнопки, передавая в родительский класс сам элемент SelenideElement, с которым будут проводиться действия.

Метод click(String elementName) выполняет клик по кнопке, при этом логирует информацию о нажатом элементе с помощью логгера. Он обращается напрямую к элементу base, унаследованному из BaseElement.

Метод byXpathCreate(String xpath) создаёт объект Button, используя произвольный xpath, переданный в параметре. Это используется в случае, когда кнопка не может быть найдена по фиксированному атрибуту или тексту.

Метод byPhraseInXpathCreate(String phrase) ищет кнопку по тексту в div, строго равному переданной фразе.

Метод byButtonPhraseInXpathCreate(String phrase) ищет button, содержащую внутри div с заданным текстом.

Метод byButtonTypeCreate(String phrase) находит кнопку по атрибуту type.

Метод byTextContainCreate(String phrase) подходит для поиска любого тега, содержащего указанный текст.

Метод byIdAndHrefCreate(String phrase1, String phrase2) возвращает ссылку (<a>), у которой data-test-id совпадает с phrase1 и href - с phrase2. Это позволяет более точно локализовать элемент.

Метод byIdAndLabelCreate(String phrase1, String phrase2) ищет ссылку <a> внутри контейнера с data-test-id=phrase1 и атрибутом aria-label=phrase2.

Метод byAriaLabelCreate(String phrase) находит кнопку по атрибуту aria-label.

Метод byButtonIdCreate(String phrase) ищет button по значению data-test-id.

Метод byIdCreate(String phrase) находит элемент по id, вне зависимости от тега.

Метод byRoleCreate(String phrase) возвращает ссылку <a>, которая вложена в div с атрибутом role='значение'. XPath выбирает первый найденный элемент.

Метод byGridIdCreate(String phrase) ищет изображение <img> внутри div с data-grid-item-idx, равным переданному значению. Применяется в контексте плиточных сеток.

Метод byHTitleCreate(String phrase) находит заголовок <h2> с заданным значением атрибута title.

Метод byRoleContainsCreate(String phrase1, String phrase2) ищет div, у которого role совпадает с phrase1, и в атрибуте class содержится phrase2. Такой подход используется при динамически формируемых классах.

Метод byDivIdCreate(String phrase) ищет div по атрибуту data-test-id.

Метод bySpanTextCreate(String phrase) ищет элемент span, текст которого строго соответствует переданному значению.

Метод byIdAndContainsCreate(String phrase1, String phrase2) ищет span внутри div с data-test-id=phrase1, текст которого содержит phrase2. Это полезно для частичных совпадений в тексте.

Таким образом, класс Button предоставляет мощный и гибкий способ поиска кнопок и интерактивных элементов на странице, позволяя писать читаемые и поддерживаемые автотесты.

**2.3. FileInput**

Класс FileInput представляет собой специализированный элемент страницы, предназначенный для работы с полями загрузки файлов. Он расширяет базовый класс BaseElement, унаследовав от него базовые возможности взаимодействия с элементом и логгирования.

Конструктор FileInput(SelenideElement element) принимает объект SelenideElement, представляющий собой HTML-элемент поля загрузки, и передаёт его в родительский конструктор, тем самым связывая экземпляр класса с конкретным элементом на веб-странице.

Метод upload(String pathInClasspath) реализует загрузку файла из classpath в указанный элемент. Перед загрузкой он записывает в лог информацию о загружаемом файле. Метод использует встроенный метод Selenide uploadFromClasspath, что позволяет загружать ресурсы, находящиеся в проекте, без необходимости указывать абсолютный путь к ним.

Метод byFileInputTypeCreate(String phrase) создаёт объект FileInput, используя XPath, где значение передаётся через параметр. Такой способ поиска подходит для стандартных полей загрузки файлов, имеющих тип file, и обеспечивает точное определение нужного элемента по атрибуту type.

Метод byXpathCreate(String xpath) предоставляет возможность создать элемент FileInput, используя произвольный XPath, заданный в виде строки. Этот способ даёт гибкость при поиске элементов, структура которых не позволяет использовать стандартные атрибуты.

Таким образом, класс FileInput служит обёрткой для загрузки файлов в веб-интерфейсе и предоставляет унифицированные методы для поиска таких полей и взаимодействия с ними.

**2.4. Input**

Класс Input представляет собой специализированную модель текстового поля на веб-странице. Он наследуется от базового класса BaseElement, что позволяет ему использовать общее логирование и базовые средства взаимодействия с DOM-элементами через библиотеку Selenide.

Конструктор Input(SelenideElement element) принимает в качестве аргумента элемент страницы и передаёт его в родительский класс BaseElement, тем самым связывая конкретный Input с HTML-элементом, с которым он будет взаимодействовать.

Метод setValue(String fieldName, String value) отвечает за ввод текста в поле. Он логирует, какое поле заполняется и каким значением, после чего вызывает метод setValue у базового элемента, тем самым осуществляя симуляцию пользовательского ввода. Это основной способ задания значений в полях ввода.

Метод pressEnter() эмулирует нажатие клавиши Enter в рамках элемента.

Метод byXpathCreate(String xpath) позволяет создать объект Input, используя произвольный XPath, предоставляя гибкость в случае нестандартных или сложно структурированных элементов.

Метод byInputNameCreate(String phrase) создаёт элемент на основе XPath, сформированного с использованием атрибута name.

Метод byIdCreate(String phrase) ищет элемент по уникальному атрибуту id, который крайне удобен и надёжен, когда ID стабилен.

Метод byAriaLabelCreate(String phrase) использует атрибут aria-label для построения XPath.

Метод byInputIdCreate(String phrase) работает схоже с byIdCreate, но с жёсткой привязкой к тегу input. Это позволяет избежать случайного совпадения с другими элементами, не являющимися полями ввода.

Метод byDivAriaCreate(String phrase) позволяет найти поле ввода, если оно реализовано как div с атрибутом aria-label. Этот метод используется, если поле реализовано не через тег input, а через div с редактируемым содержимым.

Метод byIdAndContentCreate(String phrase1, String phrase2) позволяет найти редактируемое текстовое поле внутри контейнера по ID и значению атрибута contenteditable.

Таким образом, класс Input предоставляет удобный и расширяемый интерфейс для работы с различными типами полей ввода на веб-страницах, обеспечивая как простые, так и более гибкие способы их поиска и взаимодействия.

**3. описание классов и методов страниц**

**3.1. BasePage**

Класс BasePage является базовым классом для всех страниц в тестовом фреймворке и служит основой для построения моделей страниц в тестах. Его задача состоит в том, чтобы хранить и предоставлять доступ к базовому элементу страницы, который используется как корень для поиска дочерних элементов.

Конструктор BasePage(SelenideElement baseType) принимает объект SelenideElement, который представляет основной контейнер или корневой элемент страницы, и сохраняет его в защищённом поле baseType. Это позволяет дочерним классам работать с конкретной областью DOM, относящейся к данной странице.

Поле baseType является защищённым и финальным, что гарантирует, что ссылка на корневой элемент страницы не изменится после инициализации и будет доступна в подклассах для поиска и взаимодействия с элементами страницы.

Статический метод byDataTestId(String partialId) служит для поиска элементов страницы по частичному совпадению значения атрибута data-test-id. Он формирует XPath выражение и использует метод $x из Selenide для поиска соответствующего элемента. Этот метод облегчает быстрое и удобное нахождение элементов с заданным тестовым идентификатором, что часто используется для повышения стабильности тестов.

Таким образом, BasePage задаёт каркас для всех страниц, предоставляя общие средства для идентификации и взаимодействия с корневыми элементами, что упрощает создание и поддержку моделей страниц в тестовом проекте.

**3.2. AccountPage**

Класс AccountPage представляет собой модель страницы профиля пользователя. Он расширяет базовый класс BasePage, а значит, наследует общие методы и логику, применимые ко всем страницам.

Метод AccountPage() является конструктором класса и вызывает конструктор родительского класса BasePage, передавая в него селектор, указывающий на элемент с data-test-id="header-profile". Это используется для подтверждения, что страница профиля успешно загружена перед началом любых действий.

Метод chooseBoards() кликает по кнопке, отвечающей за отображение вкладки с досками пользователя. Это позволяет открыть список всех досок на аккаунте.

Метод chooseFirstBoards() выбирает первую доску в списке, кликнув по соответствующему элементу интерфейса. Он используется для перехода на конкретную доску и взаимодействия с её содержимым.

Метод chooseFirstPinBoards() кликает по первому пину, отображаемому на выбранной доске. Он необходим для начала работы с конкретным пином, например, для его сохранения или редактирования.

Метод makePinFavourite() добавляет выбранный пин в список избранного, симулируя клик по кнопке добавления в избранное. Это позволяет протестировать функциональность избранных пинов.

Метод checkFavourites() проверяет, был ли пин действительно добавлен в избранное, и возвращает true, если на странице появилась кнопка "Удалить из избранного", сигнализирующая, что пин находится в избранном.

Метод choosePins() переключает интерфейс на вкладку с пинами пользователя, позволяя просматривать все пины, привязанные к текущему аккаунту.

Метод chooseFavourites() активирует вкладку "Избранное", открывая список пинов, добавленных в избранное пользователем.

Метод chooseMadeByMe() переключает отображение на пины, созданные непосредственно владельцем аккаунта, позволяя работать только с авторским контентом.

Метод choosePin() кликает по первому пину из списка, открывая его для дальнейших действий, таких как лайк, комментирование или сохранение.

Метод chooseLikePin() инициирует отображение меню реакций на пин, позволяя выбрать, какую реакцию пользователь хочет оставить.

Метод checkLikePin() проверяет, доступна ли реакция "Мне нравится!" после открытия меню реакций, возвращая true, если она присутствует.

Метод chooseBoard() выбирает вторую доску по заголовку, используя клик по соответствующему элементу, и используется, например, при переносе или сохранении пина.

Метод chooseFirstPinOnBoard() выбирает первый пин на указанной доске, предоставляя доступ к действиям с этим пином в контексте выбранной доски.

Метод profile() кликает по кнопке, открывающей меню выбора доски для сохранения пина, позволяя задать целевую доску перед сохранением.

Метод chooseFirstBoard() сохраняет пин на первую доску из списка, нажимая на соответствующий элемент. Он используется для подтверждения функциональности сохранения.

Метод chooseEdit() кликает по кнопке, открывающей дополнительные действия с пином, включая редактирование и удаление.

Метод chooseBoardEdit() выбирает действие редактирования пина, запуская интерфейс изменения его содержимого.

Метод chooseDelete() инициирует удаление пина, нажимая на кнопку "Удалить" в интерфейсе редактирования. Это используется для проверки удаления контента.

Метод checkConfirmDeletePin() проверяет, что пин действительно был удалён, и на странице появилось подтверждающее сообщение "Пин удален", возвращая true при успехе.

Метод checkSavedPin() проверяет, был ли пин успешно сохранён, и возвращает true, если отображается индикатор "Сохранено".

Метод chooseOtherActions() кликает по кнопке дополнительных действий, предоставляя доступ к дополнительным функциям редактирования пина.

Метод chooseCorrectPin() выбирает конкретный пин в меню редактирования, позволяя перейти к его изменению.

Метод editDescription(String title) заполняет поле описания пина новым текстом, переданным в параметре title, позволяя протестировать редактирование описания.

Метод saveDescription() сохраняет изменённое описание пина, нажимая на соответствующую кнопку "Сохранить".

Метод checkSavedDescriptionDescription(String title) проверяет, было ли новое описание успешно сохранено, и отображается ли оно на странице, возвращая true при совпадении.

Метод writeComment(String comment) вводит текст комментария к пину, используя переданную строку в качестве содержания комментария.

Метод clickChooseEmoji() открывает меню выбора эмодзи в окне комментирования, позволяя добавить визуальные реакции к сообщению.

Метод chooseEmoji() выбирает конкретный эмодзи — улыбающийся смайлик — для вставки в комментарий.

Метод sendComment() отправляет написанный комментарий, нажимая на кнопку "Опубликовать", что позволяет протестировать процесс добавления комментариев.

Метод checkCommentSend(String title) проверяет, что комментарий с указанным содержанием был успешно добавлен и отображается под пином, возвращая true в случае успеха.

Таким образом, класс AccountPage полностью моделирует поведение страницы профиля, обеспечивая полный набор действий, которые может выполнять пользователь.

**3.3. HomePage**

Класс HomePage представляет главную страницу веб-приложения Pinterest и наследуется от базового класса BasePage. Элементы страницы представлены в виде полей класса, реализованных через обёртки Button и Input, каждая из которых инициализируется через различные способы поиска DOM-элементов. Это позволяет однозначно идентифицировать элементы интерфейса на странице.

Конструктор HomePage вызывает конструктор родительского класса BasePage с локатором byDataTestId("header"). Это означает, что основным элементом, подтверждающим загрузку страницы, является компонент с data-test-id="header". Конструктор устанавливает привязку к конкретной странице.

Метод openPinCreation() открывает интерфейс создания нового пина, нажимая на кнопку, определённую по id и соответствующему значению атрибута href.

Метод openAccountPage() осуществляет переход на страницу профиля пользователя. Кнопка для перехода найдена по id и текстовой метке "Профиль".

Метод openFirstPin() открывает первый пин из пользовательской ленты. Элемент был найден с помощью XPath, указывающего на первый элемент сетки.

Метод sharedAccess() открывает диалог общего доступа, позволяющий отправить пин другому пользователю. Кнопка найдена по aria-label.

Метод writeUserName(String title) вводит имя получателя в поле поиска. Элемент определён по aria-label.

Метод sendPin() отправляет выбранный пин, нажимая на кнопку, определённую по id.

Метод checkMessage() возвращает true, если появилась кнопка перехода к чату, подтверждая успешную отправку пина.

Метод openChat() открывает чат с пользователем, к которому был отправлен пин.

Метод profile() нажимает на кнопку профиля, позволяющую, например, выбрать доску для сохранения или управлять своим контентом.

Метод createBoard() инициирует процесс создания новой доски. Элемент локализован по фразе в XPath "Создать доску".

Метод enterBoardName(String title) вводит название новой доски в соответствующее поле, определённое по id.

Метод addBoard() завершает создание доски, нажимая на кнопку "Создать", найденную по тексту.

Метод checkSavedPin() проверяет, отображается ли надпись "Сохранено", подтверждающая успешное добавление пина в сохранённое.

Метод chooseOtherActions() открывает выпадающее меню дополнительных действий с пином. Кнопка найдена по aria-label.

Метод makePinHidden() скрывает выбранный пин, используя кнопку, определённую по id.

Метод reasonPinHidden() выбирает причину скрытия пина - "Другое", элемент определён по тексту внутри span.

Метод checkHidden() возвращает true, если отображается кнопка "Отмена", что указывает на то, что элемент был скрыт.

Метод writeQuery(String query) вводит поисковый запрос в строку поиска. Поле определено по aria-label.

Метод searchPins() выполняет отправку запроса, имитируя нажатие клавиши Enter в поле поиска.

Таким образом, класс HomePage охватывает основные пользовательские сценарии, выполняемые с главной страницы. Для определения элементов используются различные методы локализации.

**3.4. LoginPage**

Класс LoginPage представляет страницу авторизации пользователя. Он наследуется от BasePage, что позволяет повторно использовать базовую функциональность, общую для всех страниц, включая ожидание загрузки ключевого элемента. Для LoginPage таким элементом выступает компонент с data-test-id="simple-login", который передаётся в конструктор базового класса.

Конструктор LoginPage вызывает конструктор BasePage с локатором byDataTestId("simple-login"), который сигнализирует о полной загрузке страницы входа. Конструктор выполняет только инициализацию страницы.

Метод clickEnterButton() эмулирует нажатие на кнопку входа, которая была найдена по фразе "Войти".

Метод enterEmailField(String email) устанавливает значение в поле ввода email. Поле идентифицируется по атрибуту name.

Метод enterPasswordField(String password) аналогично устанавливает значение в поле пароля, найденное по name.

Метод clickLoginButton() нажимает на кнопку авторизации, определённую по type="submit".

Таким образом, LoginPage предоставляет минимальный, но необходимый набор действий для имитации пользовательского входа: ввод email и пароля, нажатие на кнопку входа, и подтверждение авторизации.

**3.5. PinCreationPage**

Класс PinCreationPage наследуется от BasePage и реализует модель страницы создания пина. Он инкапсулирует логику взаимодействия с ключевыми элементами интерфейса, такими как поле загрузки изображения, поле ввода заголовка и кнопки управления публикацией. Это позволяет тестам работать с этой страницей как с отдельной сущностью и выполнять над ней типовые действия.

Конструктор PinCreationPage() вызывает конструктор родительского класса с локатором по атрибуту data-test-id со значением "pin-creation", что задаёт основную точку привязки для определения страницы в DOM-дереве.

Метод uploadImage(String imagePath) загружает изображение через элемент FileInput, который был найден по типу input "file".

Метод setTitle(String title) устанавливает заголовок пина, заполняя поле ввода, найденное по id со значением "storyboard-selector-title" - уникальному идентификатору текстового поля заголовка.

Метод publish() нажимает на кнопку публикации, которая была найдена по фразе "Опубликовать" внутри XPath.

Метод isPublished() проверяет успешность публикации пина путём проверки отображения кнопки с текстом "Ваш пин опубликован", которая была найдена по частичному совпадению текста.

Метод goToPublishedPin() осуществляет переход к опубликованному пину, кликая по той же кнопке, найденной по частичному тексту "Ваш пин опубликован".

Таким образом, класс PinCreationPage представляет собой обёртку над страницей создания пина, предоставляя удобный интерфейс для взаимодействия с её элементами, что делает тесты более читаемыми, стабильными и повторно используемыми.

**4. тестирование**

Тут может быть ваш текст, если вам, конечно, надо :)

**заключение**

В ходе выполнения работы была успешно создана и документирована система автоматизированного тестирования веб‑приложения Pinterest с использованием Java, библиотеки Selenide, фреймворка JUnit и системы сборки Maven. В качестве основы был разработан набор из десяти UI‑тестов, охватывающих ключевые пользовательские сценарии: от создания и публикации пина с загрузкой изображения до сохранения, перемещения и скрытия пинов, редактирования описаний, добавления комментариев и эмодзи, постановки лайков и избранного, а также отправки пинов через внутренние сообщения и поиска по запросам. Каждый тест подробно описан в виде чек‑листа с чётко определёнными входными данными, последовательностью действий и критериями ожидаемого результата, что обеспечивает воспроизводимость проверок и прозрачность анализа возможных дефектов.

При реализации фреймворка был использован подход Page Object Model: для инкапсуляции элементов пользовательского интерфейса созданы универсальные обёртки BaseElement, Button, Input и FileInput, а для моделирования страниц - классы BasePage, HomePage, LoginPage, AccountPage и PinCreationPage. Такой дизайн позволяет легко поддерживать и расширять набор тестов при изменении интерфейса, обеспечивая повторное использование кода и единый стиль логирования.

Интеграция с Maven и JUnit обеспечила удобство сборки, запуска и формирования отчётов о результатах прогонов в формате XML/HTML, тогда как возможности Selenide по гибким ожиданиям элементов гарантируют устойчивость тестов к сетевым и временным задержкам. Логирование действий каждого теста позволяет быстро локализовать и устранить причины сбоев.

В целом, созданный фреймворк является надёжной и гибкой основой для автоматизации тестирования ключевых функций Pinterest и готов к масштабированию в соответствии с ростом функциональности и требований проекта.

**список использованных источников**

***Ниже представлены примеры библиографического описания, В качестве названия источника в примерах приводится вариант, в котором применяется то или иное библиографическое описание.***

1. Иванов И. И. Книга одного-трех авторов. М.: Издательство, 2010. 000 с.

2. Книга четырех авторов / И. И. Иванов, П. П. Петров, С. С. Сидоров, В. В. Васильев. СПб.: Издательство, 2010. 000 с.

3. Книга пяти и более авторов / И. И. Иванов, П. П. Петров, С. С. Сидоров и др.. СПб.: Издательство, 2010. 000 с.

4. Описание книги под редакцией / под ред. И.И. Иванова СПб., Издательство, 2010. 000 с.

5. Иванов И.И. Описание учебного пособия и текста лекций: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2010. 000 с.

6. Описание методических указаний / сост.: И.И. Иванов, П.П. Петров. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2010. 000 с.

7. Иванов И.И. Описание статьи с одним-тремя авторами из журнала // Название журнала. 2010, вып. (№) 00. С. 000–000.

8. Описание статьи с четырьмя и более авторами из журнала / И. И. Иванов, П. П. Петров, С. С. Сидоров и др. // Название журнала. 2010, вып. (№) 00. С. 000–000.

9. Иванов И.И. Описание тезисов доклада с одним-тремя авторами / Название конференции: тез. докл. III международной науч.-техн. конф., СПб, 00–00 янв. 2000 г. / СПбГЭТУ «ЛЭТИ», СПБ, 2010, С. 000–000.

10. Описание тезисов доклада с четырьмя и более авторами / И. И. Иванов, П. П. Петров, С. С. Сидоров и др. // Название конференции: тез. докл. III международной науч.-техн. конф., СПб, 00–00 янв. 2000 г. / СПбГЭТУ «ЛЭТИ», СПБ, 2010, С. 000–000.

11. Описание электронного ресурса // Наименование сайта. URL: http://east-front.narod.ru/memo/latchford.htm (дата обращения: 00.00.2010).

12. ГОСТ 0.0–00. Описание стандартов. М.: Изд-во стандартов, 2010.

13. Пат. RU 00000000. Описание патентных документов / И. И. Иванов, П. П. Петров, С. С. Сидоров. Опубл. 00.00.2010. Бюл. № 00.

14. Иванов И.И. Описание авторефератов диссертаций: автореф. дисс. канд. техн. наук / СПбГЭТУ «ЛЭТИ», СПБ, 2010.

15. Описание федерального закона: Федер. закон [принят Гос. Думой 00.00.2010] // Собрание законодательств РФ. 2010. № 00. Ст. 00. С. 000–000.

16. Описание федерального постановления: постановление Правительства Рос. Федерации от 00.00.2010 № 00000 // Опубликовавшее издание. 2010. № 0. С. 000–000.

17. Описание указа: указ Президента РФ от 00.00.2010 № 00 // Опубликовавшее издание. 2010. № 0. С. 000–000.