**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по «UI-тестирование» практике**

Тема: Тестирование Википедии

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342  Студент гр. 3381  Студент гр. 3342 |  | Русанов А.И.  Спиридонов А.Н.  Смирнова Е.С. |
| Руководитель |  | Шевелева А.М. |

Санкт-Петербург

2025

**ЗАДАНИЕ**

**на «UI-тестирован» практику**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 Русанов А.И.  Студент гр. 3381 Спиридонов А.Н.  Студент гр. 3342 Смирнова Е.С. | | |
|  | | |
| Тема практики: Тестирование Википедии | | |
| Задание на практику:  Нужно написать 10 тестов по одному определенному блоку / функционалу системы. Например, работа с постами в вк – создание поста, поделиться постом на своей странице, добавить комментарий к посту, лайкнуть пост, поделиться постом в сообщении, удалить пост, закрепить пост, добавить в архив, отключить комментарии. | | |
| Сроки прохождения практики: 25.06.2025 – 08.07.2025 | | |
| Дата сдачи отчета: 07.07.2025 | | |
| Дата защиты отчета: 07.07.2025 | | |
|  | | |
| Студенты |  | Русанов А.И.  Спиридонов А.Н.  Смирнова Е.С. |
| Руководитель |  | Шевелева А.М. |

**Аннотация**

Цель практики — освоение автоматизированного тестирования веб-приложений на примере Википедии [[1](#Bookmark_1)] с использованием Java, Selenide [[3](#Bookmark_2)], Junit [[4](#Bookmark_4)] и Maven [[5](#Bookmark_5)]. Разработано 10 автотестов для ключевых сценариев: авторизация, поиск статей, работа с медиаконтентом, генерация PDF и внешними ссылками. Командная реализация ведётся через GitHub [[2](#Bookmark_3)] с распределением задач и документированием тест-кейсов в формате чек-листов. Проект включает сквозное логирование, анализ специфики многоязычного контента и итоговый отчёт с архитектурным описанием и демонстрацией результатов.

**Summary**

The purpose of the practice is to master automated testing of web applications using the example of Wikipedia [[1](#Bookmark_1)] using Java, Selenide [[3](#Bookmark_2)] , Junit [[4](#Bookmark_4)] and Maven [[5](#Bookmark_5)]. 10 autotests have been developed for key scenarios: authorization, article search, working with media content, PDF generation and external links. Team implementation is conducted via GitHub [[2](#Bookmark_3)] with assignment of tasks and documentation of test cases in the format of checklists. The project includes end-to-end logging, an analysis of the specifics of multilingual content, and a final report with an architectural description and demonstration of the results.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Первый Раздел | 6 |
| 1.1. | Описание тестов | 6 |
| 2. | Второй Раздел | 12 |
| 2.1. | Описание классов и методов | 12 |
| 2.1.1 | Классы элементов | 12 |
| 2.1.2 | Классы страниц | 14 |
| 2.1.3 | Классы тестов | 18 |
| 2.2. | UML диаграмма | 26 |
| 3. | Третий Раздел | 22 |
| 3.1. | Тестирование одного теста | 22 |
| 3.2. | Успешная отработка всех тестов | 27 |
|  | Заключение | 28 |
|  | Список использованных источников | 30 |

**введение**

Целью данной практической работы является освоение комплексного подхода к автоматизированному тестированию веб-приложений на примере Википедии с использованием современных инструментов: Java в качестве базового языка программирования, Selenide для эффективного взаимодействия с браузером, JUnit как тестового фреймворка и Maven для управления зависимостями и сборкой проекта. В рамках проекта разработан набор из 10 ключевых автотестов, обеспечивающих комплексную проверку критически важных функциональных блоков системы:

1. Работа со списком наблюдения
2. Авторизация с неверными данными
3. Загрузка медиафайлов
4. Генерация короткого URL
5. Поиск статей
6. Кнопка "Цитировать"
7. Раздел "Ссылки"
8. Сноски и примечания
9. Переключение языка
10. Экспорт в PDF

Командная разработка реализована через GitHub с распределением зон ответственности между участниками: каждый член команды отвечал за реализацию, документирование и валидацию конкретных тест-кейсов в формате структурированных чек-листов с детализацией шагов, входных данных и ожидаемых результатов.

Итоговый отчёт содержит: архитектурное описание решения с UML-диаграммами взаимодействия компонентов и демонстрацию работы всех тестов.

**1. ПЕРВЫЙ РАЗДЕЛ**

## 1.1 Описание тестов

## **Предустановленный тест:** Найти случайные статьи, содержащие все необходимые элементы для последующих тестов, и записать их названия в JSON-файл.

1. **Проверка работы списка наблюдения:** Проверить, что кнопка "Следить" корректно меняет состояние (активное/неактивное) на ранее найденной статье после авторизации.
2. **Авторизация с неверными данными:** Проверить, что система выдает ошибку при попытке входа с неверным логином (test\_invalid) и паролем (wrong123).
3. **Проверка загрузки медиафайлов:** Проверить открытие и закрытие галереи изображений при клике на картинку в ранее найденной статье.
4. **Проверка генерации короткого URL:** Проверить, что функция "Получить короткий URL" для ранее найденной статьи создает валидную короткую ссылку (https://w.wiki/...), ведущую на ту же статью.
5. **Поиск статьи (существующей/несуществующей):** Проверить, что поиск существующей статьи открывает ее, а поиск заведомо несуществующей (dfgdfgdfgdfg) выдает сообщение об отсутствии результатов.
6. **Проверка кнопки "Цитировать страницу":** Проверить, что нажатие кнопки "Цитировать страницу" в боковом меню ранее найденной статьи открывает окно с библиографическими данными в разных форматах.
7. **Проверка раздела "Ссылки":** Проверить наличие, кликабельность и работоспособность (переход на внешний ресурс, HTTP 200) первой ссылки в разделе "Ссылки" ранее найденной статьи.
8. **Проверка отображения сносок и примечаний:** Проверить кликабельность сносок в тексте ранее найденной статьи, переход по якорной ссылке и работу кнопки возврата "Обратно к тексту".
9. **Переключение языка страницы:** Проверить, что нажатие на ссылку "English" в разделе "На других языках" ранее найденной статьи корректно переводит на английскую версию (смена домена, шаблонная строка).
10. **Проверка работы "Скачать как PDF":** Проверить корректность скачивания (название, отсутствие ошибок, возможность открытия) PDF-файла ранее найденной статьи через функцию "Скачать как PDF".

**2. ВТОРОЙ РАЗДЕЛ**

**2.1. Описание классов и методов**

Полный реализованный код можно посмотреть в Github репозитории [[6](#Bookmark_6)].

**2.1.1 Классы элементов**

BaseElement – Базовый абстрактный класс для работы с веб-элементами через Selenide, поддерживающий CSS и XPath-локаторы.

* BaseElement(String selector) – Конструктор, инициализирует элемент по CSS-селектору.
* BaseElement(String locator, boolean isXpath) – Создает элемент через XPath (если isXpath=true) или CSS.
* isAvailable() – Комбинированная проверка существования и видимости (exists() && isDisplayed()).
* isDisplayed() – Проверяет видимость с дефолтным таймаутом (10 сек).
* exists() – Определяет наличие элемента в DOM.
* isDisplayed(Duration timeout) – Проверяет видимость с кастомным таймаутом.

ArticleTitleElement – Класс для работы с заголовками статей Википедии, наследуемый от BaseElement с поддержкой XPath-локаторов.

* ArticleTitleElement(String selector, boolean xpath) – Конструктор с возможностью указания XPath-селектора
* getText() – Возвращает текст заголовка статьи
* byId(String id) – Фабричный метод для создания элемента по ID через XPath

ArticleSection – Класс для работы с разделами статей Википедии, наследуемый от BaseElement с поддержкой XPath-локаторов.

* ArticleSection(String selector, boolean xpath) – Конструктор с возможностью указания XPath-селектора
* byId(String id) – Фабричный метод для создания элемента по ID через XPath

ClickableElement – Абстрактный класс для кликабельных элементов, наследующий функциональность BaseElement.

* ClickableElement(String selector) – Конструктор с CSS-селектором
* ClickableElement(String locator, boolean isXpath) – Конструктор с поддержкой XPath
* click() – Выполняет клик по элементу с дефолтным таймаутом (10 сек)
* click(Duration timeout) – Выполняет клик с указанным временем ожидания

FollowButton – Класс для работы с кнопкой подписки на статьи Википедии, наследующий функциональность ClickableElement.

* FollowButton(String watch\_id, String unwatch\_id, boolean xpath) – Конструктор, принимающий ID кнопок "Следить"/"Отписаться" и флаг использования XPath
* isActive() – Проверяет активное состояние подписки (наличие кнопки "Отписаться")
* unwatchButtonShouldBeVisible() – Проверяет видимость кнопки "Отписаться" с таймаутом 3 секунды
* watchButtonShouldBeVisible() – Проверяет видимость кнопки "Следить" с таймаутом 3 секунды
* byIds(String watch\_id, String unwatch\_id) – Фабричный метод для создания элемента по ID кнопок через XPath

GalleryCloseButton – Класс для работы с кнопкой закрытия галереи изображений, наследующий функциональность ClickableElement.

* GalleryCloseButton(String selector, boolean xpath) – Конструктор с возможностью указания XPath-селектора
* clickAndWaitUntilClosed() – Выполняет клик с ожиданием закрытия (таймаут 10 сек)
* waitUntilOpened() – Ожидает появления кнопки (таймаут 10 сек)
* byTitle(String title) – Фабричный метод для создания элемента по атрибуту title через XPath

InputElement – Класс для работы с элементами ввода (input fields), наследующий функциональность BaseElement.

* InputElement(String selector, boolean xpath) – Конструктор с возможностью указания XPath-селектора
* fill(String text) – Заполняет поле ввода указанным текстом
* clear() – Очищает содержимое поля ввода
* byId(String id) – Фабричный метод для создания элемента по ID через XPath
* byName(String name) – Фабричный метод для создания элемента по атрибуту name через XPath

InteractiveElement – Класс для работы с интерактивными элементами интерфейса, наследующий функциональность ClickableElement.

* InteractiveElement(String selector) – Конструктор с CSS-селектором
* InteractiveElement(String locator, boolean isXpath) – Конструктор с поддержкой XPath
* byId(String id) – Фабричный метод для создания элемента по ID через XPath
* byType(String type) – Фабричный метод для создания элемента по атрибуту type через XPath
* byTitle(String title) – Фабричный метод для создания элемента по атрибуту title через XPath
* byName(String name) – Фабричный метод для создания элемента по атрибуту name через XPath
* bySelector(String css\_selector) – Фабричный метод для создания элемента по CSS-селектору

LanguageSwitchButton – Класс для работы с кнопками переключения языка, наследующий функциональность InteractiveElement.

* LanguageSwitchButton(String selector, boolean xpath) – Конструктор с возможностью указания XPath-селектора
* byLangValue(String lang\_value) – Фабричный метод для создания элемента по атрибуту lang через XPath

SearchElement – Класс для работы с поисковой строкой, наследующий функциональность InputElement.

* SearchElement(String selector, boolean xpath) – Конструктор с возможностью указания XPath-селектора
* search(String query) – Вводит текст запроса и имитирует нажатие Enter
* byClass(String class\_name) – Фабричный метод для создания элемента по классу через XPath

ShortUrlButton – Класс для работы с кнопкой генерации коротких URL, наследующий функциональность InteractiveElement.

* ShortUrlButton(String selector, boolean xpath) – Конструктор с возможностью указания XPath-селектора
* getShortUrlFromDialog() – Получает сгенерированный короткий URL из диалогового окна
* byId(String id) – Фабричный метод для создания элемента по ID через XPath

UserIcon – Класс для работы с иконкой пользователя, наследующий функциональность InteractiveElement.

* UserIcon(String selector, boolean xpath) – Конструктор с возможностью указания XPath-селектора
* getText() – Возвращает текст элемента (например, имя пользователя)
* byId(String id) – Фабричный метод для создания элемента по ID через XPath

WrongLoginMessage – Класс для работы с сообщением о неверном логине, наследующий функциональность BaseElement.

* WrongLoginMessage(String selector, boolean xpath) – Конструктор с возможностью указания XPath-селектора
* getMessageText() – Возвращает текст сообщения об ошибке (с удалением лишних пробелов)
* byClass(String class\_name) – Фабричный метод для создания элемента по классу через XPath

BodyElement – Класс для работы с основным контейнером страницы (тег <body>), наследующий функциональность BaseElement.

* BodyElement() – Конструктор без параметров, использующий CSS-селектор "body" по умолчанию

CitationHeading – Класс для работы с заголовком цитирования, наследующий функциональность BaseElement.

* CitationHeading() – Конструктор без параметров, использующий CSS-селектор "#firstHeading" по умолчанию
* getText() – Возвращает текст заголовка с проверкой видимости элемента

CiteButton – Класс для работы с кнопкой цитирования, наследующий функциональность BaseElement.

* CiteButton(String selector) – Конструктор с указанием CSS-селектора
* byDefault() – Фабричный метод для создания элемента с селектором "#t-cite" по умолчанию
* isAvailable() – Проверяет доступность кнопки (существование и видимость)

ScrollableElement – Абстрактный класс для работы с прокручиваемыми элементами, наследующий функциональность BaseElement.

* ScrollableElement(String selector) – Конструктор с указанием CSS-селектора
* scrollTo() – Прокручивает страницу до данного элемента

ReferencesSection – Класс для работы с разделом ссылок, наследующий функциональность ScrollableElement.

* ReferencesSection(String selector) – Конструктор с указанием CSS-селектора
* byDefault() – Фабричный метод для создания элемента с селектором "#Ссылки" по умолчанию
* clickFirstLink() – Кликает по первой ссылке в разделе
* hasLinks() – Проверяет наличие ссылок в разделе (возвращает true, если есть хотя бы одна ссылка)

BackToTextButton – Класс для работы с кнопкой "Обратно к тексту", наследующий функциональность BaseElement.

* BackToTextButton(String selector, boolean xpath) – Конструктор с указанием селектора и флага использования XPath
* byDefault() – Фабричный метод для создания элемента с XPath-селектором "//a[@title='Обратно к тексту']" по умолчанию

DownloadButton – Класс для работы с кнопкой скачивания, наследующий функциональность BaseElement.

* DownloadButton(String selector, boolean xpath) – Конструктор с указанием селектора и флага использования XPath
* byText(String id) – Фабричный метод для создания элемента по тексту кнопки
* downloadPdf() – Выполняет скачивание PDF-файла и возвращает объект File

FootnoteElement – Класс для работы с элементами сноски, наследующий функциональность BaseElement.

* FootnoteElement(String selector, boolean xpath) – Конструктор с указанием селектора и флага использования XPath
* byFootnoteNumber(String number) – Фабричный метод для создания элемента сноски по её номеру (использует XPath)

**2.1.2 Классы страниц**

BasePage – Абстрактный базовый класс для всех страниц приложения.

* BasePage(String url) – Конструктор для страниц с прямым URL
* BasePage() – Конструктор для страниц, открываемых через взаимодействия
* open() – Открывает страницу по указанному в конструкторе URL
* open(String given\_url) – Открывает страницу по переданному URL
* getTitle() – Возвращает заголовок текущей страницы
* getCurrentUrl() – Возвращает текущий URL страницы

ArticlePage – Класс страницы статьи, наследующий функциональность BasePage.

Элементы страницы:

* titleElement - заголовок статьи
* englishButton - кнопка переключения на английскую версию
* pdfButton - кнопка скачивания PDF
* shortUrlButton - кнопка получения короткой ссылки
* referencesSection - раздел "Ссылки"
* notesSection - раздел "Примечания"
* citeButton - кнопка цитирования
* followButton - кнопка подписки на статью
* galleryCloseButton - кнопка закрытия галереи
* infoboxImage - изображение в инфобоксе
* backToTextButton - кнопка "Назад к тексту"
* footnote - элемент сноски

Основные методы:

* getArticleTitle() – Возвращает заголовок статьи
* openShortUrlDialogWindow() – Открывает диалоговое окно короткой ссылки
* getShortUrlFromDialogWindow() – Получает короткую ссылку из диалога
* hasImages() – Проверяет наличие изображений в статье
* hasEnglishVersion() – Проверяет доступность английской версии (таймаут 3 сек)
* hasFollowButton() – Проверяет наличие кнопки подписки
* scrollToFooter() – Прокручивает страницу к подвалу
* hasPdfDownload() – Проверяет доступность PDF-скачивания
* hasCiteOption() – Проверяет доступность цитирования
* hasShortUrlOption() – Проверяет доступность короткой ссылки
* hasReferencesSection() – Проверяет наличие раздела ссылок
* hasNotesSection() – Проверяет наличие раздела примечаний
* isArticleWatched() – Проверяет активное состояние подписки
* toggleFollowButton() – Переключает состояние подписки
* waitUntilWatchButtonIsVisible() – Ожидает появления кнопки "Следить"
* waitUntilUnwatchButtonIsVisible() – Ожидает появления кнопки "Отслеживается"
* openGallery() – Открывает галерею изображений
* waitUntilGalleryIsOpened() – Ожидает открытия галереи
* closeGallery() – Закрывает галерею (таймаут 3 сек)
* isGalleryVisible() – Проверяет видимость галереи
* switchToEnglish() – Переключает на английскую версию статьи
* hasEnglishTemplate() – Проверяет наличие английского шаблона
* clickFootnote() – Кликает по сноске
* isBackToTextButtonVisible() – Проверяет видимость кнопки "Назад к тексту"
* clickBackToTextButton() – Кликает кнопку "Назад к тексту"
* urlContainsAnchor() – Проверяет наличие якоря в URL
* openPdfDownload() – Открывает страницу скачивания PDF
* clickCiteButton() – Открывает страницу цитирования
* scrollToReferencesSection() – Прокручивает до раздела "Ссылки"
* referencesSectionHasLinks() – Проверяет наличие ссылок в разделе
* clickFirstReferenceLink() – Кликает первую ссылку в разделе ссылок

CitationPage – Класс страницы цитирования, наследующий функциональность BasePage.

Элементы страницы:

* citationHeading - заголовок раздела цитирования

Основные методы:

* CitationPage() – Конструктор по умолчанию (без указания URL)
* getCitationHeadingText() – Возвращает текст заголовка раздела цитирования

LoginPage – Класс страницы авторизации.

Элементы страницы:

* loginField - поле ввода логина
* passwordField - поле ввода пароля
* loginAttemptButton - кнопка входа
* wrongLoginMessage - сообщение об ошибке авторизации

Основные методы:

* enterLogin(String login) – Вводит логин в соответствующее поле
* enterPassword(String password) – Вводит пароль в соответствующее поле
* submit() – Выполняет попытку входа (клик по кнопке)
* getErrorMessage() – Возвращает текст сообщения об ошибке при неверной авторизации

MainPage – Класс главной страницы, наследующий функциональность BasePage.

Элементы страницы:

* randomPageButton - кнопка случайной статьи
* searchElement - поле поиска
* loginButton - кнопка входа
* logoutButton - кнопка выхода
* userIcon - иконка пользователя

Основные методы:

* MainPage() – Конструктор с URL главной страницы
* openRandomArticle() – Открывает случайную статью
* openArticle(String articleName) – Открывает статью по названию
* clickLoginButton() – Переходит на страницу авторизации
* isLoggedIn(String username) – Проверяет вход пользователя
* logout() – Выходит из системы
* byDefault() – Фабричный метод для создания экземпляра

PdfDownloadPage – Класс страницы скачивания PDF, наследующий функциональность BasePage.

* downloadPdf() – Выполняет скачивание PDF-файла

WindowManager – Класс для управления окнами и прокруткой страницы.

* scrollToFooter() – Плавно прокручивает страницу до конца

AuthService – Сервис для авторизации пользователя.

* login() – Выполняет авторизацию на сайте

TestDataLoader – Класс для загрузки тестовых данных из JSON-файла.

* getTestArticle(String key) – Возвращает тестовые данные по ключу

TestResults – Класс для сохранения и загрузки результатов тестирования.

* init() – Инициализирует загрузку результатов из файла
* addResult(String testName, String articleTitle) – Добавляет новый результат теста
* getResults() – Возвращает копию текущих результатов
* saveToFile() – Сохраняет результаты в JSON-файл

**2.1.3 Классы тестов**

BaseTest – Базовый класс для тестов, содержащий общие настройки.

* setup() – Настраивает параметры браузера перед всеми тестами
* init() – Выполняет инициализацию перед каждым тестом
* openMainPage() – Открывает главную страницу и инициализирует mainPage
* auth() – Выполняет авторизацию перед каждым тестом

GalleryTest – Тестовый класс для проверки работы галереи изображений.

* shouldOpenAndCloseGallery() – Проверяет открытие и закрытие галереи

InvalidLoginTest – Тестовый класс для проверки ввода неверных учетных данных.

* auth() – Переопределенный метод подготовки: открывает главную страницу и выполняет выход, если пользователь авторизован
* shouldShowErrorOnInvalidCredentials() – Основной тест: переходит на страницу входа, вводит неверные учетные данные и проверяет сообщение об ошибке

ShortUrlGenerationTest – Тестовый класс для проверки генерации коротких URL.

* shouldGenerateValidShortUrl() – Проверяет работу генератора коротких ссылок: загружает тестовую статью, открывает диалог генерации короткого URL, проверяет формат сгенерированной ссылки (должна начинаться с <https://w.wiki/>), открывает статью по короткой ссылке и сравнивает заголовки оригинальной статьи и статьи по короткой ссылке

WatchButtonTest – Класс тестирования функционала кнопки подписки на статьи.

* shouldToggleWatchStatus() – Проверяет переключение состояния подписки: загружает тестовую статью, запоминает начальное состояние подписки, переключает состояние кнопки и проверяет изменение статуса подписки

FootnoteTest – Класс тестирования работы сносок в статьях.

* footnoteShouldWorkCorrectly() – Проверяет полный цикл работы со сносками: загрузка тестовой статьи, клик по сноске, проверка URL с якорем сноски, проверка видимости кнопки возврата, клик по кнопке возврата и проверка URL после возврата

LanguageSwitchTest – Класс тестирования переключения языковой версии статьи.

* shouldSwitchToEnglishVersion() – Проверяет переключение на английскую версию: загрузка тестовой статьи, проверка доступности английской версии, переключение языка, проверка URL (должен содержать en.wikipedia.org), проверка сохранения заголовка и проверка шаблонной строки на английском

PdfDownloadTest – Класс тестирования скачивания PDF-версии статьи.

* shouldDownloadPdfCorrectly() – Проверяет процесс скачивания PDF: загрузка тестовой статьи, проверка наличия кнопки PDF, открытие страницы экспорта, скачивание файла и проверки файла (существование, размер, валидность)
* isPdfValid(File file) – Вспомогательный метод проверки PDF

CiteButtonTest – Класс тестирования функционала кнопки цитирования.

* citeButtonOpensCitationPage() – Проверяет работу кнопки цитирования: загружает тестовую статью, кликает кнопку цитирования, проверяет заголовок открывшейся страницы цитирования и сравнивает с ожидаемым значением "Библиографические ссылки на статью"

SearchTest – Класс тестирования поиска статей.

* searchExistingArticle() – Проверяет поиск существующей статьи: загружает тестовую статью и сравнивает заголовок открытой статьи с ожидаемым
* searchNonExistingArticle() – Проверяет поиск несуществующей статьи: пытается открыть статью с заведомо неверным названием и проверяет, что заголовок не соответствует тестовому ключу

ReferencesTest – Класс тестирования раздела ссылок в статьях.

* firstReferenceLinkShouldOpenExternalResource() – Проверяет работу ссылок в разделе "Ссылки": загружает тестовую статью, прокручивает до раздела ссылок, проверяет наличие ссылок в разделе, кликает первую ссылку, проверяет переход на внешний ресурс (URL не содержит wikipedia.org) и проверяет корректность загрузки страницы

**2.2. UML диаграмма**

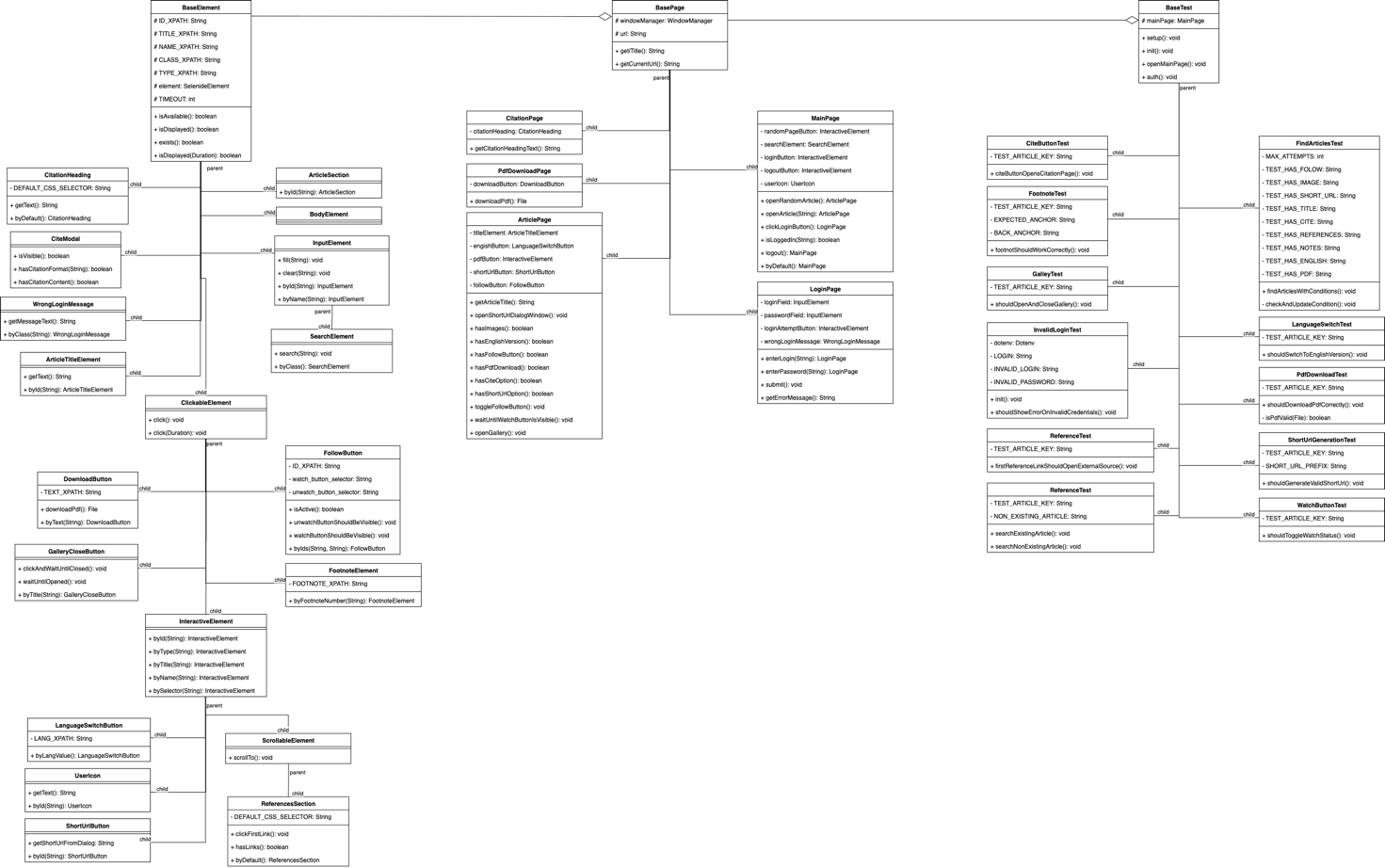
****

Рисунок 1 – UML диаграмма классов

**3. ТРЕТИЙ РАЗДЕЛ**

**3.1.** Тестирование одного теста

Название теста – ShortUrlGenerationTest

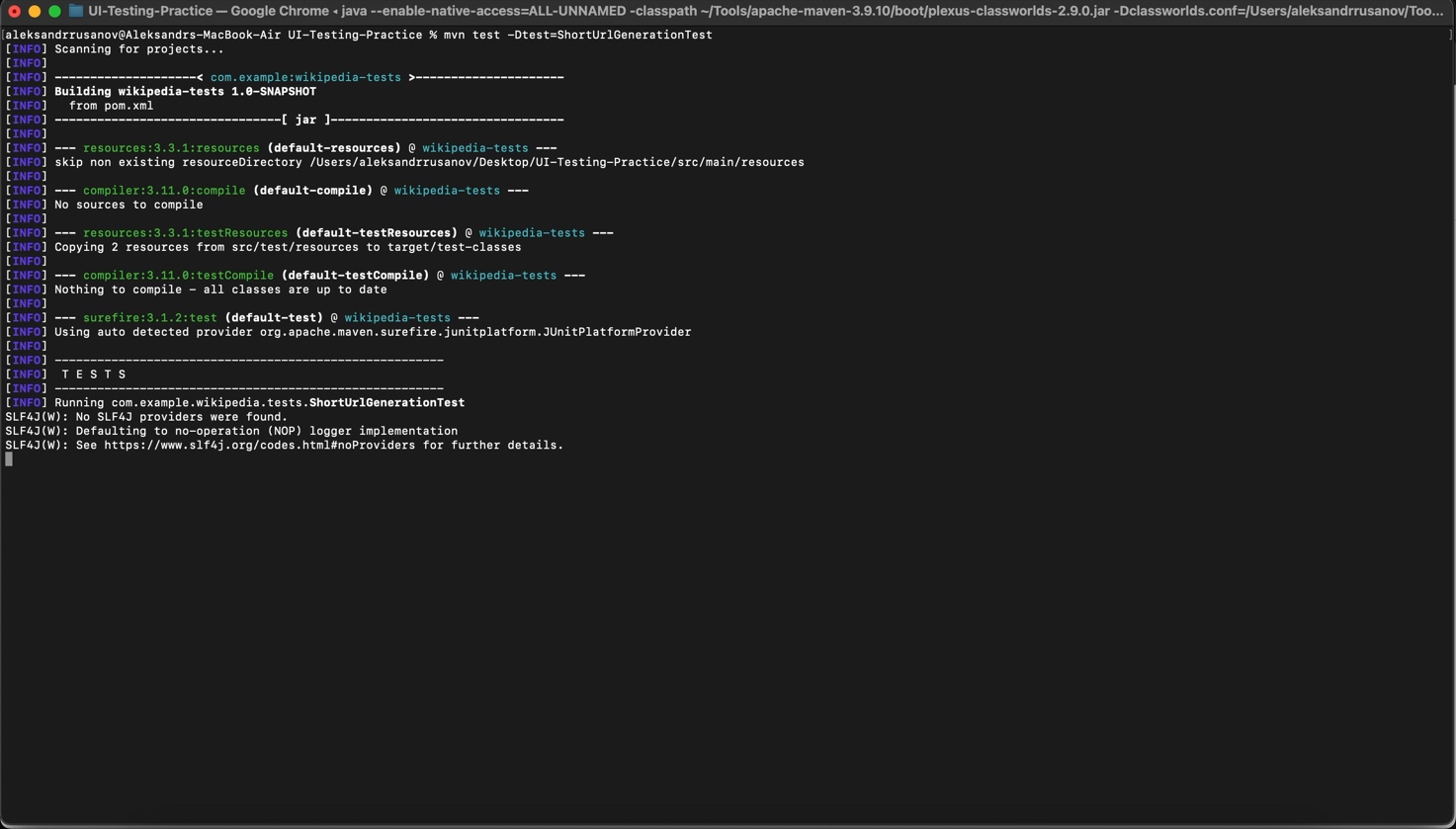
****

Рисунок 2 – Запуск ShortUrlGenerationTest

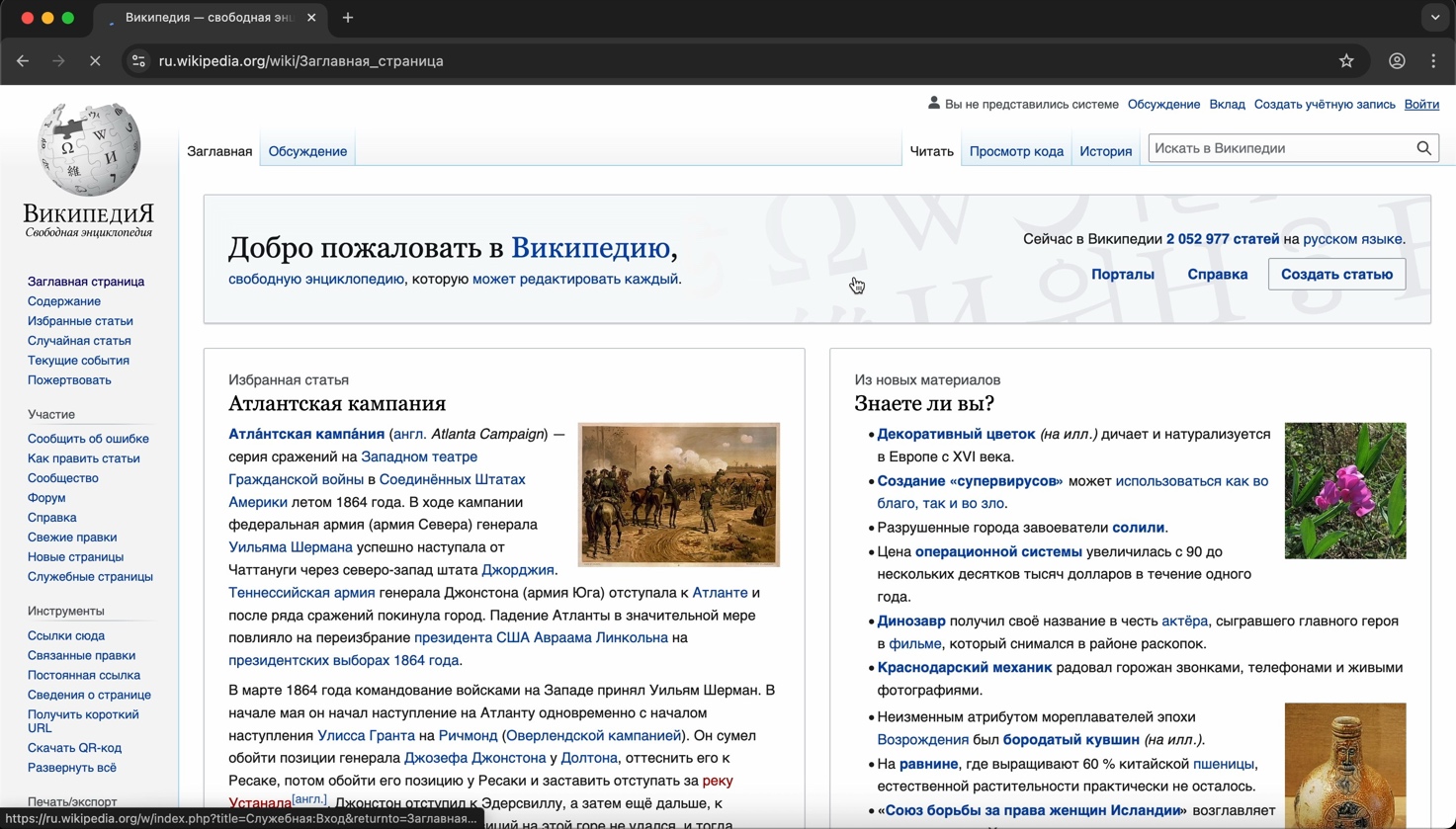


Рисунок 3 – Открытие главной страницы

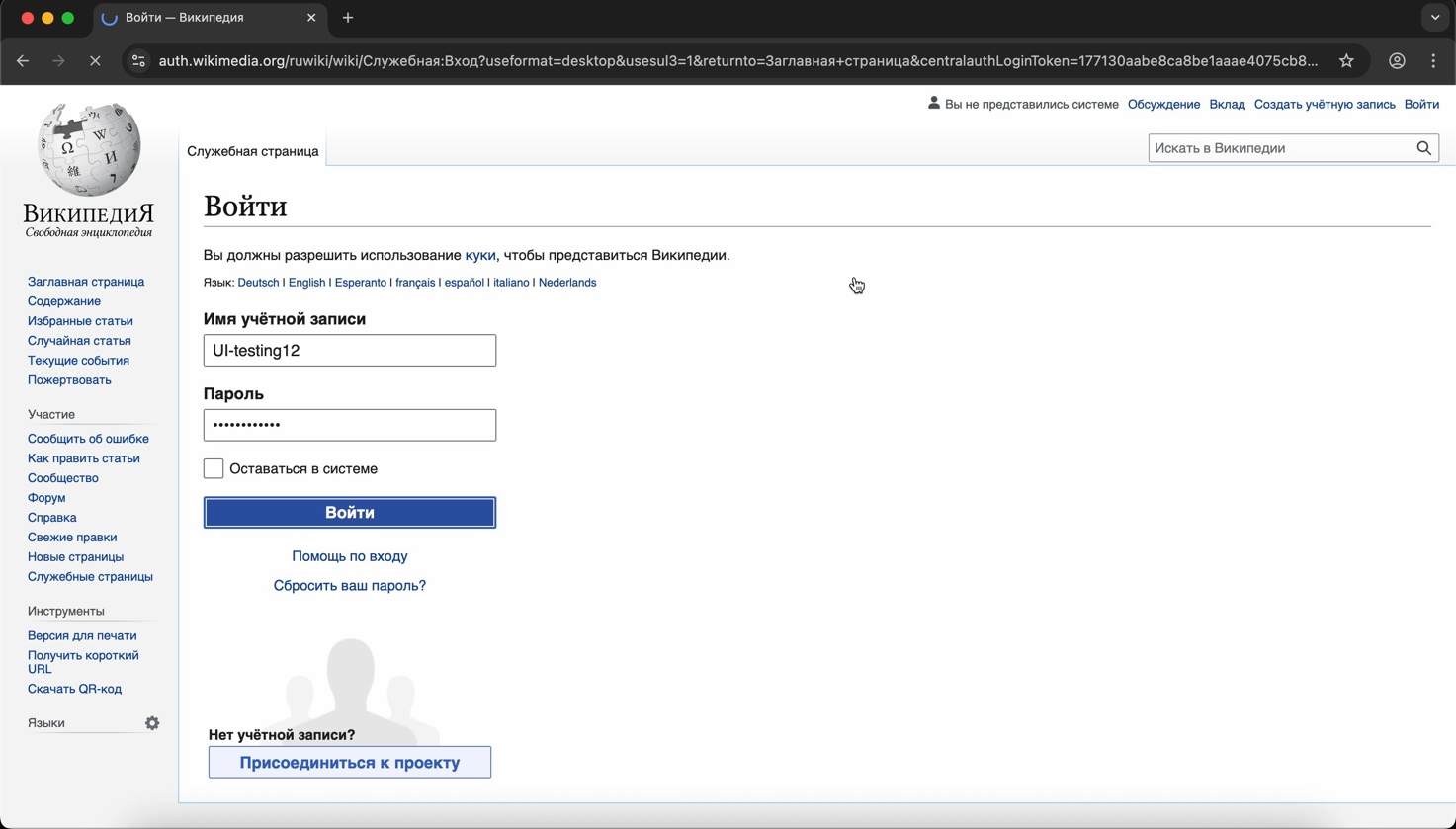


Рисунок 4 – Авторизация пользователя

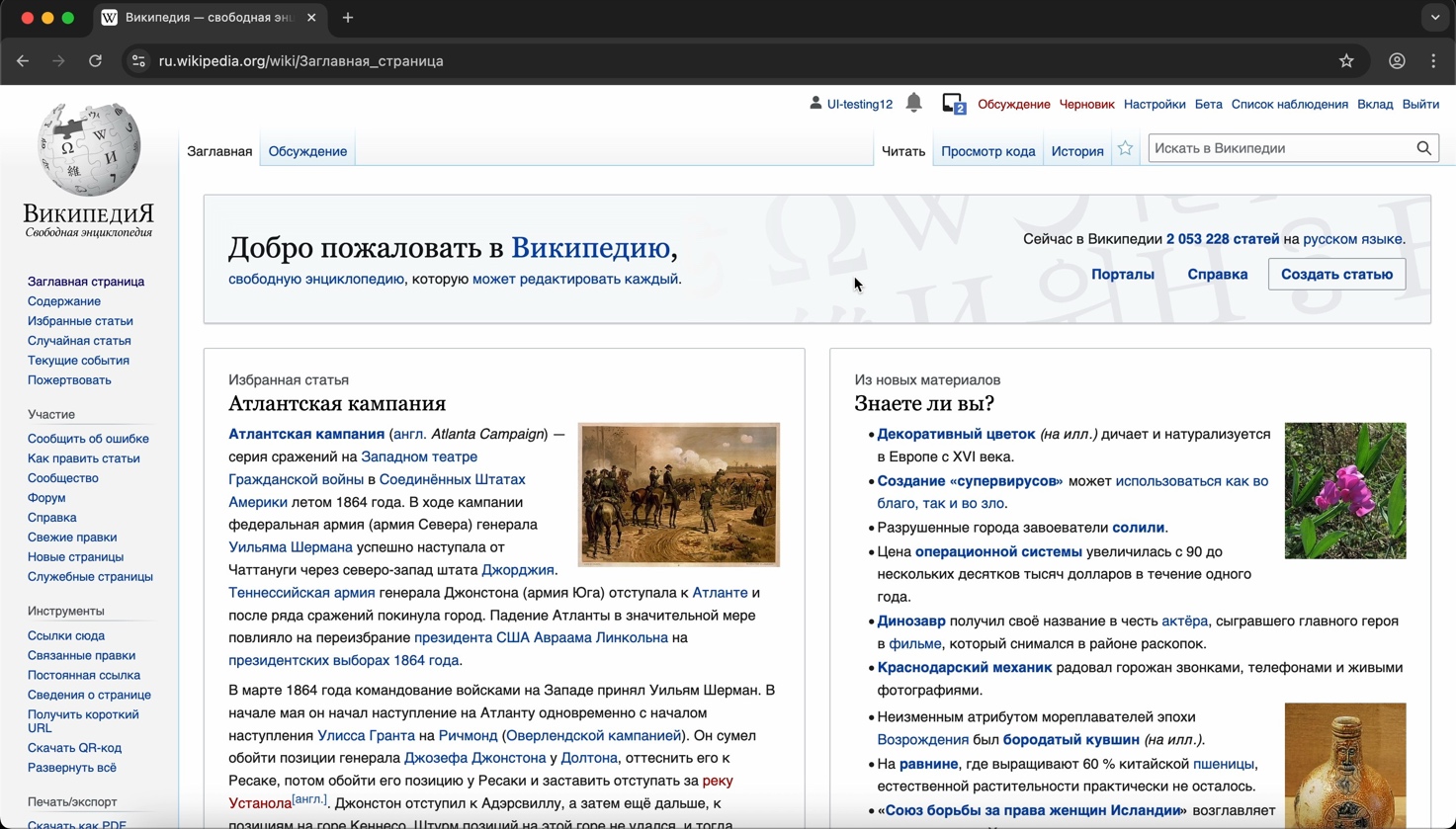


Рисунок 5 – Пользователь авторизован

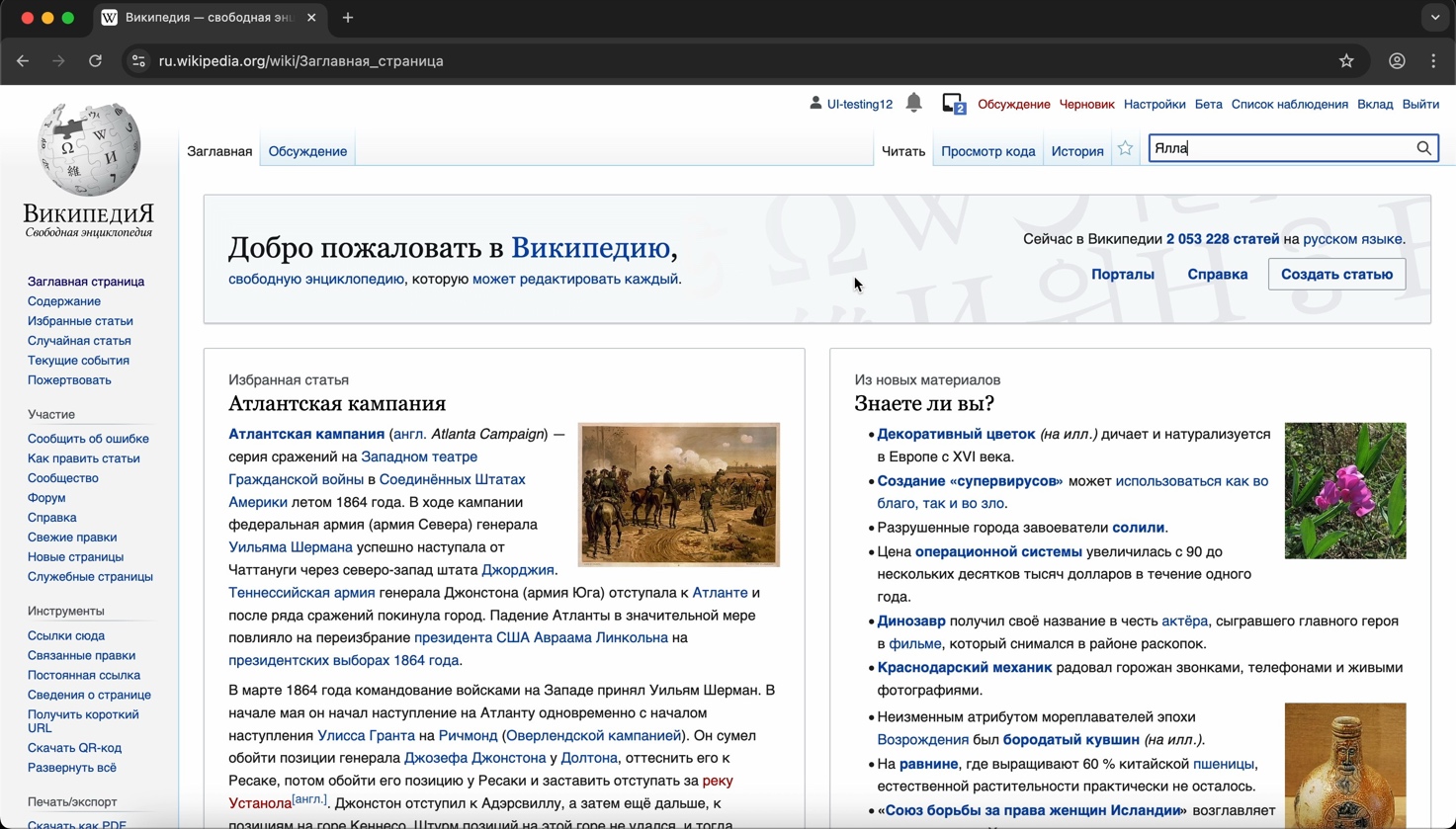


Рисунок 6 – Ввод названия статьи в поле поиска

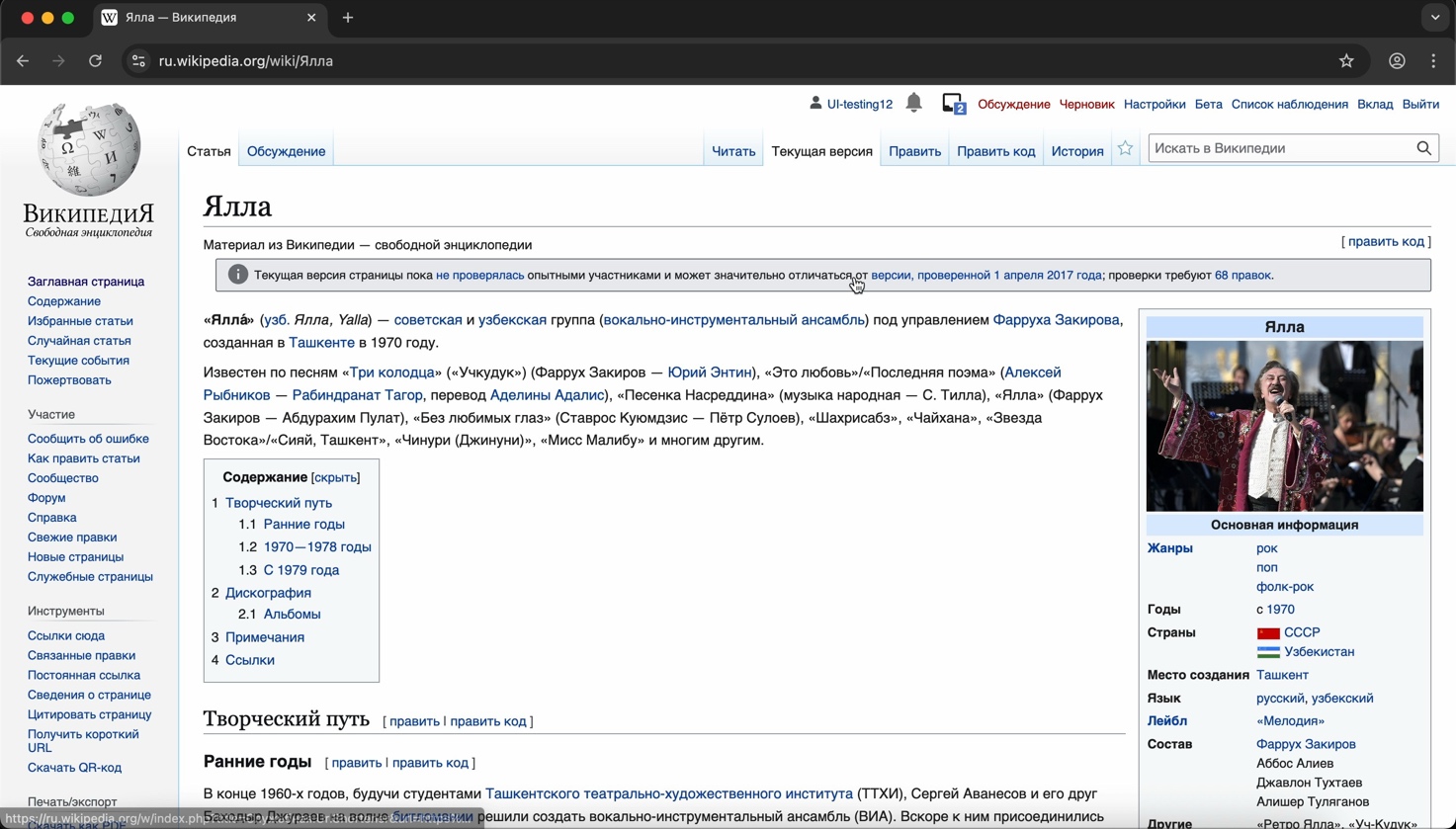


Рисунок 7 – Открытие страницы статьи

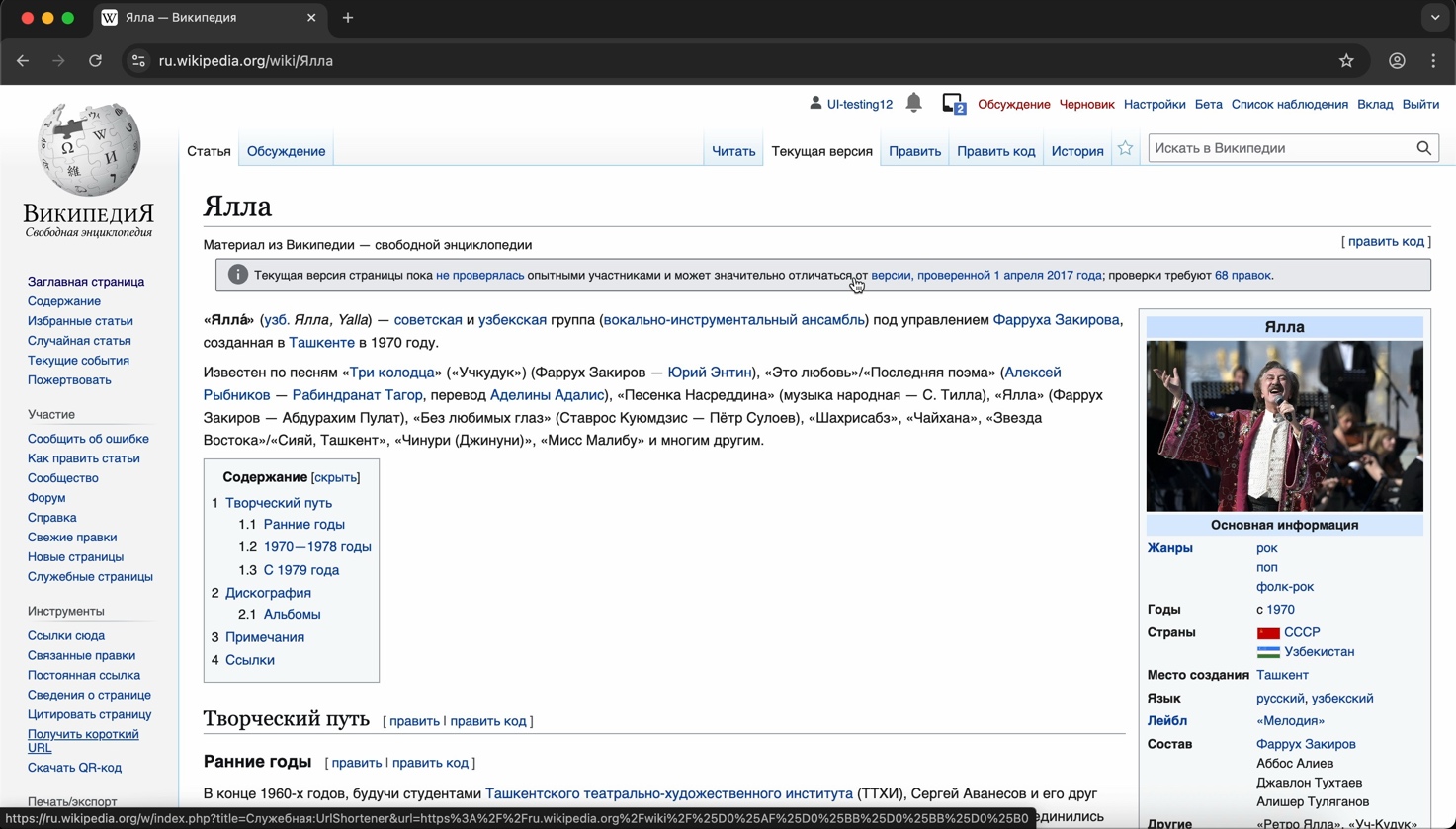
****

Рисунок 8 – Клик по кнопке «Получить короткий URL»

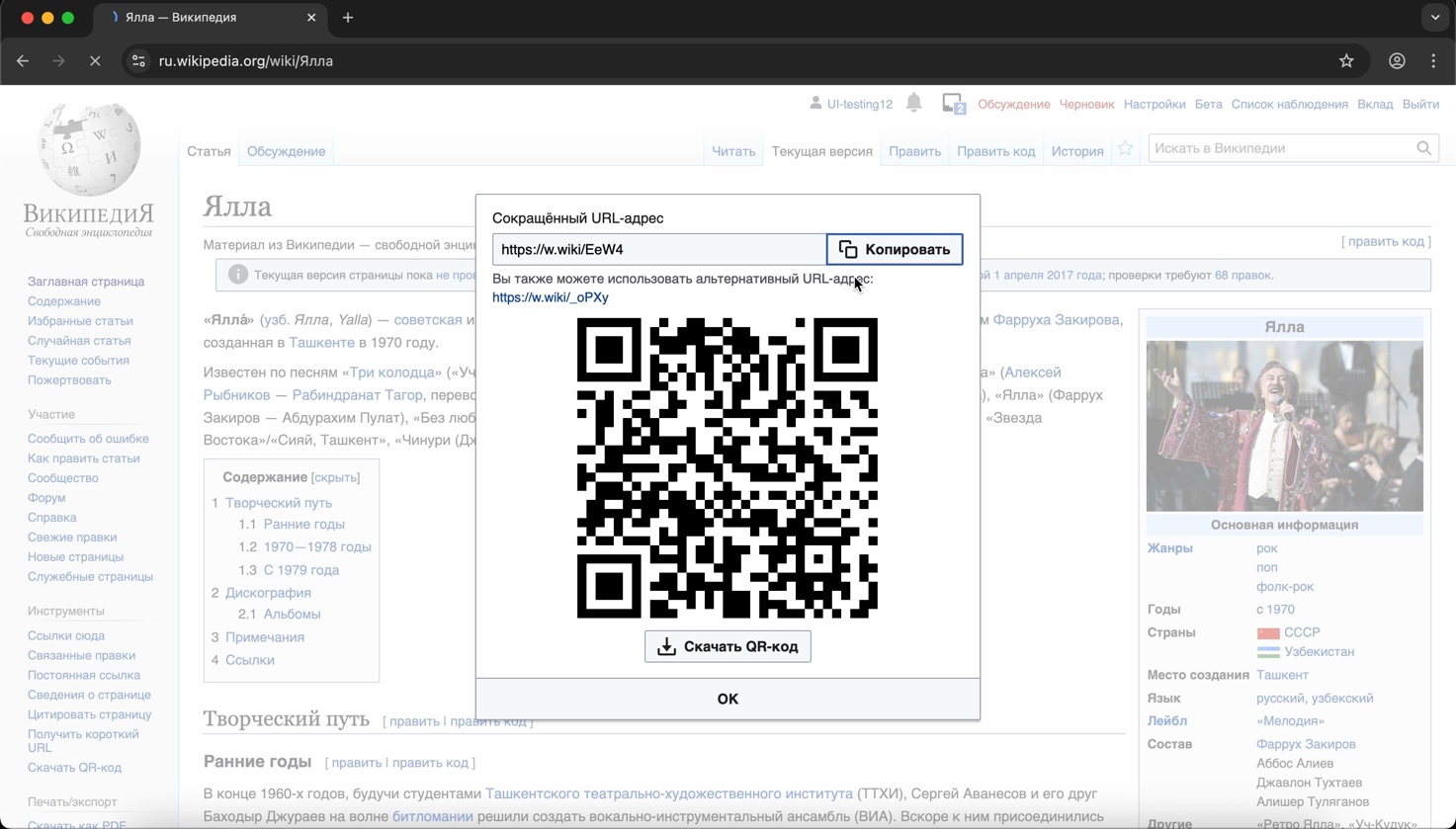
****

Рисунок 9 – Открытие модального окна с сокращенным URL

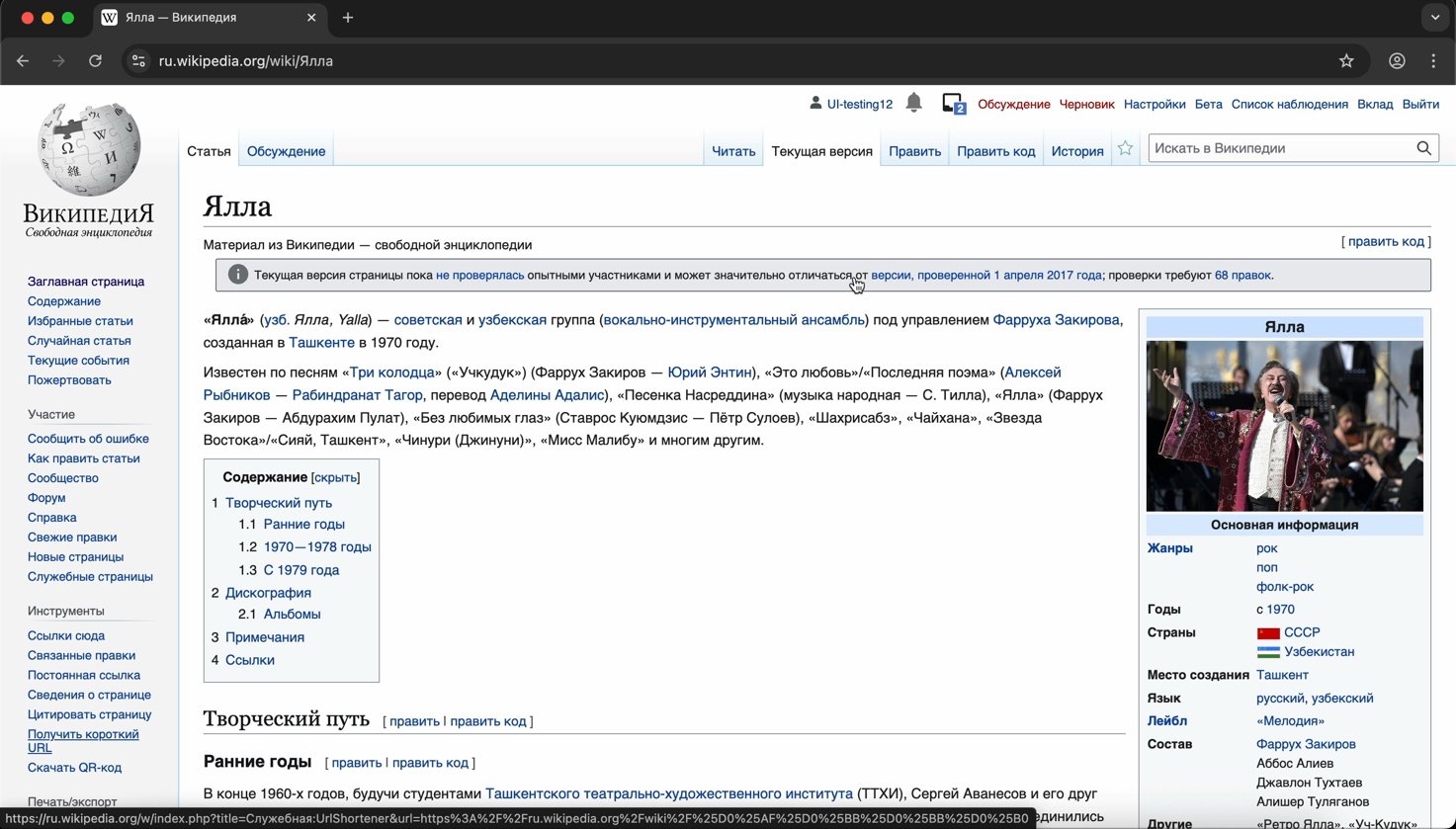
****

Рисунок 10 – Открытие страницы по сокращенному URL

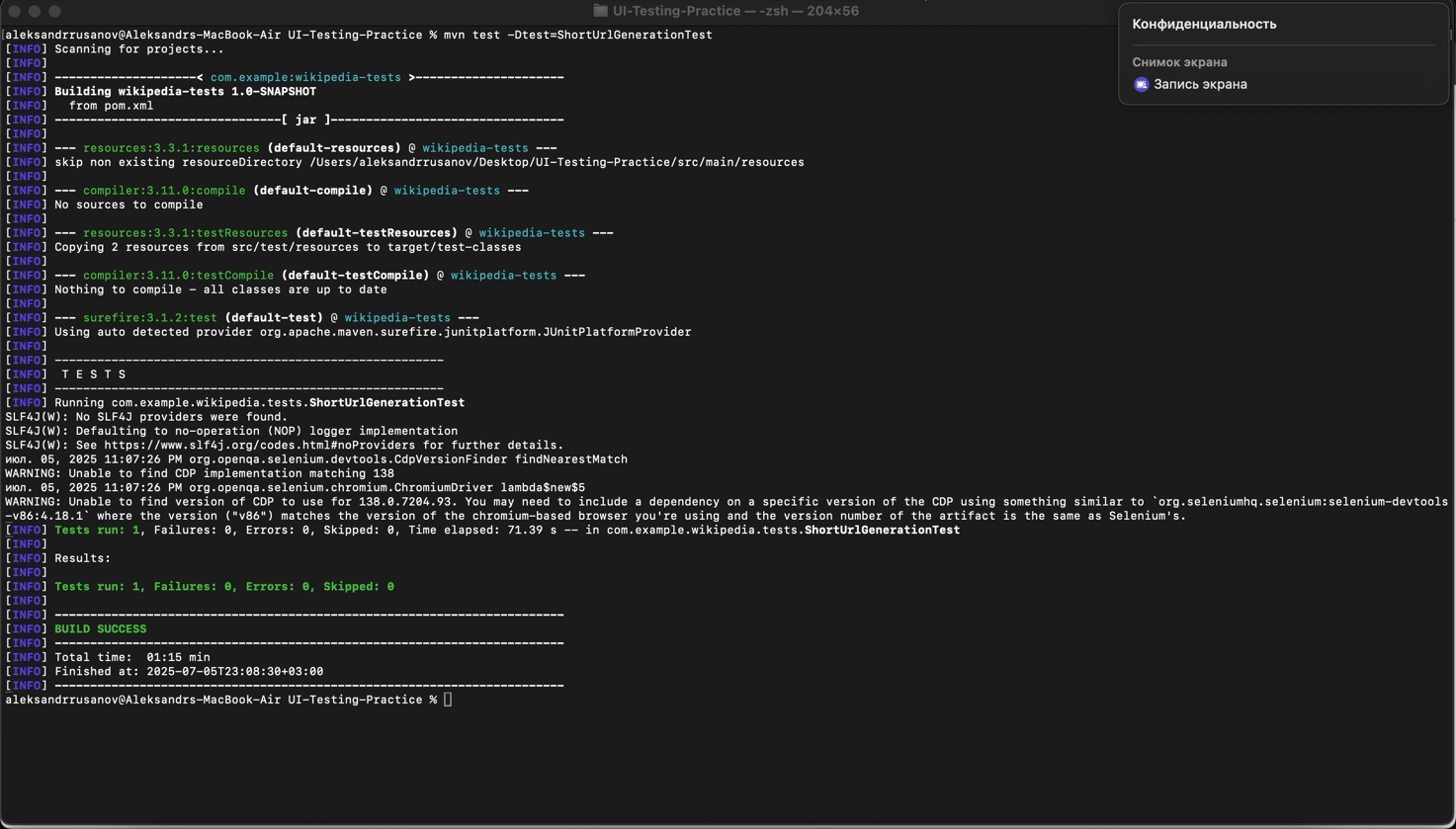
****

Рисунок 11 – Успешное завершение теста

**3.2.** Успешная отработка всех тестов

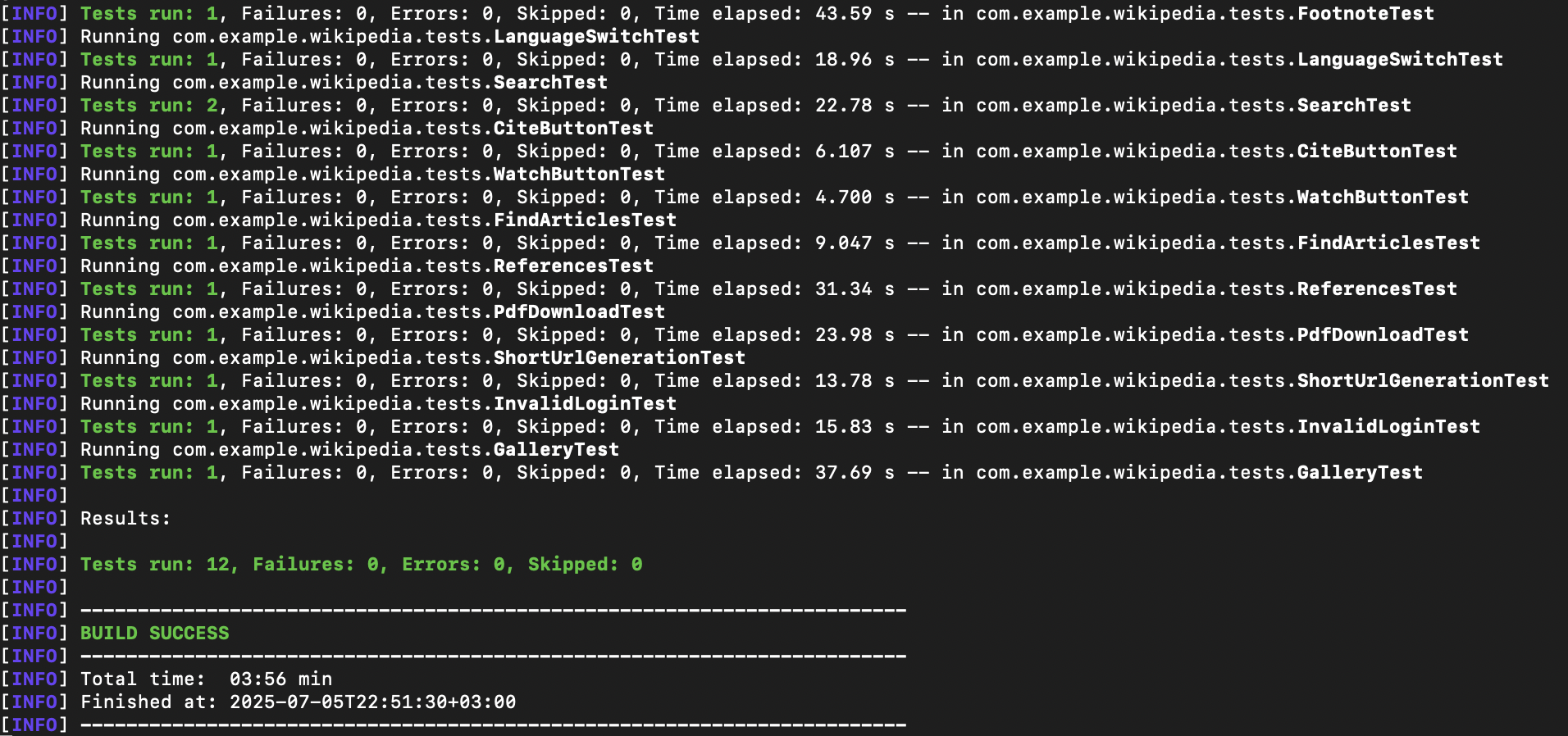
****

Рисунок 12 – Успешная отработка всех тестов

**заключение**

В ходе практической работы была разработана архитектура для реализации десяти UI-тестов сайта «Википедия», основанная на базовых классах элементов и страниц. Базовый класс элемента (BaseElement) определил общий интерфейс для всех компонентов, включая методы проверки видимости, наличия в DOM и другие. От него были унаследованы специализированные классы, такие как ArticleTitleElement, FollowButton, GalleryCloseButton, SearchElement и другие, которые расширили функциональность базового класса для работы с конкретными элементами интерфейса Википедии.

Аналогичным образом был построен базовый класс страницы (BasePage), отвечающий за навигацию, проверку URL и другие общие действия. От него наследовались классы страниц, такие как ArticlePage, LoginPage, MainPage, CitationPage и PdfDownloadPage, которые реализовали специфичные для каждой страницы методы и элементы. Это позволило создать структурированную и легко расширяемую систему тестирования.

В ходе работы были успешно реализованы все 10 запланированных тестов, включая проверку авторизации, работы со списком наблюдения, загрузки медиафайлов, генерации коротких URL, поиска статей и других ключевых функций Википедии. Тесты покрыли основные сценарии взаимодействия пользователя с сайтом, обеспечив высокий уровень проверки функциональности.

Использование таких инструментов, как Java, Selenide, JUnit и Maven, позволило создать надежные и стабильные автотесты. Командная работа через GitHub обеспечила эффективное распределение задач и контроль версий. Реализованная архитектура обладает высокой гибкостью и масштабируемостью: добавление новых тестов или изменение существующих механизмов взаимодействия с элементами требует минимальных правок благодаря наследованию и инкапсуляции.

Таким образом, поставленная цель — освоение автоматизированного тестирования веб-приложений на примере Википедии — была достигнута. Полученные результаты демонстрируют соответствие разработанного решения современным требованиям к UI-тестированию и могут служить основой для дальнейшего расширения тестового покрытия.

**список использованных источников**

1. Википедия: О проекте [Электронный ресурс] / Wikimedia Foundation. – URL: <https://ru.wikipedia.org/wiki/Википедия:О_проекте>
2. GitHub Documentation [Электронный ресурс] / GitHub Inc. – URL: [https://docs.github.com](https://docs.github.com/#_blank)
3. Selenide: лаконичные и стабильные UI тесты на Java [Электронный ресурс]. – URL: [https://selenide.org](https://selenide.org/#_blank)
4. JUnit 5 User Guide [Электронный ресурс] / JUnit Team. – URL: [https://junit.org/junit5/docs/current/user-guide](https://junit.org/junit5/docs/current/user-guide#_blank)
5. Apache Maven Documentation [Электронный ресурс] / The Apache Software Foundation. – URL: <https://maven.apache.org/guides>
6. **UI-Testing-Practice Github repository [Электронный ресурс] – URL:** <https://github.com/rusanovaleksandr/UI-Testing-Practice>