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

ОТЧЕТ

По «UI-тестирование» практике

Тема: Тестирование Яндекс Маркета

Студенты гр. 3388 _____ Глебова В.С
Раутио И.А
Басик В.В

Руководитель _____ Шевелева А.М

Санкт-Петербург
2025

ЗАДАНИЕ

НА «UI-ТЕСТИРОВАНИЕ» ПРАКТИКУ

Студенты: Глебова В.С, Раутио И.А, Басик В.В

Группа: 3388

Тема практики: Тестирование Яндекс Маркета

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 _____ Глебова В.С, Раутио И.А, Басик В.В

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 три 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 three 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. Написать 10 автоматизированных тестов для выбранной системы.
2. Использовать технологии: Java, Selenide (Selenium), JUnit 5, Maven, логирование.
3. Организовать работу в GitHub (репозиторий на группу из 3 человек с распределением задач в README.md).
4. Оформить чеклист в виде таблицы с описанием тестов.

Для тестирования был выбран сайт Яндекс Маркет (https://market.yandex.ru). Выбор обусловлен тем, что это популярная и функционально насыщенная торговая площадка с широким ассортиментом товаров и развитой системой поиска, фильтрации и навигации. Такие особенности позволяют протестировать разнообразные сценарии взаимодействия пользователя с веб-интерфейсом, включая авторизацию, поиск, фильтрацию, добавление товаров в корзину и сравнение.

1. ПЕРВЫЙ РАЗДЕЛ

1.1. Чеклист

Тест 1 «Поиск товара»

Шаги теста:

1. Авторизация на сайте
2. Ввести запрос в поисковую строку «iphone 15»
3. Нажать кнопку «Найти»

Ожидаемый результат:

1. Отображается заголовок с текстом запроса
2. Карточки товара содержат ключевые слова

Тест 2 «Фильтр по цене»

Шаги теста:

1. Авторизация
2. Ввести запрос в поисковую строку «ноутбук»
3. Установить минимальную цену 10 тыс.
4. Установить максимальную цену 50 тыс.
5. Применить

Ожидаемый результат:

1. Все товары в выдаче имеют цену в заданном диапазоне

Тест 3 «Релевантность поиска»

Шаги теста:

1. Авторизация
2. Ввести запрос в поисковую строку «iphone 15»
3. Проверка, что у всех введённых товаров в заголовке есть «iphone 15»

Ожидаемый результат:

1. После ввода запроса «iphone 15» и выполнения поиска, на странице отображаются товары, у которых в названии присутствует ключевое слово «iphone 15».
2. Все найденные товары соответствуют поисковому запросу (например: «iPhone 15 128 ГБ», «Чехол для iPhone 15», и т. д.).
3. Отсутствуют товары, не связанные с данным запросом (например, «Samsung Galaxy S24» или «Наушники Bluetooth» без указания iPhone 15 в названии).

Тест 4 «Добавление товара в корзину»

Шаги теста:

1. Авторизация
2. Программа ищет по запросу «ноутбук»

Ожидаемый результат:

1. После выполнения поискового запроса «ноутбук» отображается список товаров, соответствующих данному запросу.
2. Первый товар который выведется, добавляется в корзину
3. При переходе в корзину добавленный товар отображается в списке с указанием наименования, цены и количества

Тест 5 «Удаление товара из корзины»

Шаги теста:

1. Авторизация
2. Заходим в корзину и проверяем есть ли товар
3. Удаляем товар

Ожидаемый результат:

1. Происходит проверка, есть ли товар в корзине, если нету, то выводится «Корзина должна содержать товар»
2. Если корзина прошла проверку и товар имеется в корзине, удаляем его

Тест 6 «Поиск по каталогу»

Шаги теста:

1. Авторизация
2. Открыть категорию «Электроника»
3. Перейти в «Смартфоны»

Ожидаемый результат:

1. После открытия категории «Электроника» отображается список подкатегорий, включая «Смартфоны».
2. При переходе в раздел «Смартфоны» открывается страница с каталогом смартфонов.
3. Отображаются карточки товаров, соответствующие выбранной категории.

Тест 7 «Проверка заголовка главной страницы»

Шаги теста:

1. Авторизация на сайте (если требуется для теста).
2. Открытие главной страницы Яндекс.Маркета.
3. Получение заголовка страницы

Ожидаемый результат:

1. Заголовок страницы содержит текст "Яндекс.Маркет" (например, "Яндекс.Маркет — купить товары в интернет-магазине").
2. Формат заголовка соответствует ожиданиям (отсутствие ошибок в написании, дополнительных символов).

Тест 8 «Проверка пустой корзины»

Шаги теста:

1. Авторизация на сайте (если требуется для теста).
2. Переход в корзину через кнопку в хедере.

3. Проверка, что корзина пустая (например, отображается сообщение "Корзина пуста").

Ожидаемый результат:

1. Сообщение "Корзина пуста" отображается (или аналогичный индикатор).
2. Отсутствуют карточки товаров в корзине.

Тест 9 «Проверка кнопки каталога»

Шаги теста:

1. Авторизация на сайте (если требуется для теста).
2. Клик по кнопке каталога в хедере сайта.
3. Переход в раздел "Электроника" (например, через меню).
4. Проверка отображения меню каталога (наличие заголовка, списка

Ожидаемый результат:

1. Меню каталога отображается (например, заголовок "Электроника" или список товаров).
2. Карточки товаров содержат ключевые слова (например, "смартфоны", "планшеты").
3. подкатегорий).

Тест 10 «Проверка элементов хедера»

Шаги теста:

1. Авторизация на сайте (если требуется для теста).
2. Переход на главную страницу Яндекс.Маркета.
3. Проверка отображения элементов хедера:
    - Кнопка входа/авторизации.

- Кнопка корзины.
- Поисковое поле.

Ожидаемый результат:

1. Кнопка входа отображается (например, "Войти").
2. Кнопка корзины отображается (например, иконка корзины).
3. Поисковое поле доступно для ввода запроса.

2. ВТОРОЙ РАЗДЕЛ

2.1 Описание классов и методов

1. Класс:

YandexMarketTests

Главный тестовый класс, содержащий набор тестов для проверки функциональности Яндекс.Маркет

Методы класса:

- testHomePageTitle()

1) Проверяет заголовок главной страницы
2) Использует authService.authenticate() для аутентификации
3) Проверяет, что заголовок содержит "Яндекс Маркет"

- testSearchFunctionality()

1) Тестирует функциональность поиска
2) Вводит текст поиска (Constants.SEARCH_PHONE)
3) Нажимает кнопку поиска
4) Проверяет отображение заголовка результатов

- testPriceFilter()

1) Тестирует фильтр по цене
2) Ищет ноутбуки (Constants.SEARCH_LAPTOP)
3) Применяет фильтр по цене (мин/макс из Constants)
4) Проверяет, что цены находятся в заданном диапазоне

- testSearchRelevance()

1) Проверяет релевантность поиска
2) Ищет телефон (Constants.SEARCH_PHONE)
3) Проверяет, что заголовки товаров содержат искомый текст

- testAddToCart()

1) Тестирует добавление товара в корзину (с обработкой нескольких вкладок)
2) Использует логирование через LoggerUtil

3) Ищет ноутбук, выбирает первый результат
4) Переключается на новую вкладку с товаром
5) Добавляет товар в корзину и проверяет его наличие

- testRemoveFromCart()

1) Тестирует удаление товара из корзины
2) Проверяет, что товар есть в корзине перед удалением
3) Удаляет товар и проверяет, что корзина пуста

- testCategoryNavigation()

1) Проверяет навигацию по категориям
2) Открывает категорию "Электроника"
3) Проверяет URL на соответствие категории

- testEmptyCart()

1) Проверяет состояние пустой корзины
2) Убеждается, что корзина пуста

- testCatalogButton()

1) Проверяет работу кнопки каталога
2) Открывает категорию "Электроника"
3) Проверяет отображение меню каталога

- testHeaderElements()

1) Проверяет элементы хедера
2) Проверяет отображение кнопки входа, корзины и поля поиска

2. Класс:

AuthService (Сервис аутентификации)

Инкапсулирует логику входа пользователя в систему Яндекс.Маркета. Это вспомогательный сервис, который упрощает авторизацию в тестах, скрывая детали процесса (открытие страницы, ввод логина/пароля и т.д.).

Методы класса:

- authenticate()

1) Логирует начало процесса (LoggerUtil.logInfo).

2) Открывает главную страницу Яндекс.Маркета (driver.get("https://market.yandex.ru")).

3) Переходит на страницу входа через кнопку в хедере (homePage.header().clickLoginButton()).

4) Вводит логин/пароль из Constants и выполняет вход (loginPage.login(Constants.USERNAME, Constants.PASSWORD)).

5) Возвращает HomePage после успешной авторизации.

3. Класс:

Constants (Утилита для хранения констант)

Класс содержит статические константы, используемые в тестах для Яндекс.Маркета. Упрощает поддержку кода, позволяя изменять значения в одном месте.

Основные константы:

- Учетные данные

USERNAME, PASSWORD - логин и пароль для авторизации

- Тестовые данные

SEARCH_PHONE, SEARCH_LAPTOP - поисковые запросы

MIN_PRICE, MAX_PRICE - диапазон цен для фильтрации

- Проверочные значения

EXPECTED_CART_ITEMS_COUNT - ожидаемое количество товаров

EXPECTED_EMPTY_CART_TEXT - текст пустой корзины

- Настройки

DEFAULT_TIMEOUT - стандартное время ожидания (5 сек)

4. Класс:

LoggerUtil

LoggerUtil представляет собой утилитарный класс для централизованного логирования в автоматизированных тестах Яндекс.Маркета. Класс реализован как статический фасад (static facade) над библиотекой Log4j2, что обеспечивает:

1) Единую точку управления логированием во всем проекте
2) Простоту использования без необходимости создания экземпляров
3) Консистентность формата логов
4) Легкую расширяемость функционала

Методы класса:

- logInfo(String message)

Логирует информационное сообщение.

- logError(String message)

Логирует сообщение об ошибке.

- logWarning(String message)

Логирует предупреждение.

5. Класс:

BasePage

Класс BasePage является абстрактным базовым классом для всех страниц в проекте Яндекс.Маркета. Он предоставляет фундаментальную функциональность, которую наследуют все конкретные страницы.

6. Класс:

BaseComponent

Класс BaseComponent является абстрактным базовым классом для всех компонентов страницы в проекте Яндекс.Маркета. Он наследуется от BasePage и предоставляет общую функциональность для работы с веб-элементами компонентов.

7. Класс:

BaseTest

Класс BaseTest является базовым классом для всех тестов, предоставляя общую конфигурацию WebDriver и сервисов.

Текущая реализация:

- Настройка WebDriver

Использует WebDriverManager для автоматического управления драйверами

Конфигурирует ChromeOptions с базовыми параметрами

Устанавливает неявные ожидания (implicit wait)

- Сервисы

Инициализирует AuthService для работы с аутентификацией

- Жизненный цикл теста

@BeforeMethod для настройки перед каждым тестом

@AfterMethod для очистки после каждого теста

8. Класс:

SearchResultsPage

Управление страницей результатов поиска в Яндекс.Маркете. Наследуется от BasePage, содержит методы для работы с фильтрами, товарами и проверки результатов.

Элементы страницы (локаторы):

1) Заголовок результатов

Отображается после выполнения поиска

Используется для проверки успешности поиска

2) Цены товаров

Список элементов с ценами

Применяется для проверки фильтра по цене

3) Первый товар в списке

Кнопка/блок для выбора товара

Используется для перехода на страницу товара

4) Названия товаров

Список элементов с наименованиями

Проверяет релевантность поиска

5) Поля ввода цены

Минимальная и максимальная цена

Для задания диапазона фильтрации

Методы класса:

- isResultsHeaderDisplayed()

1) Ожидает появления заголовка результатов
2) Возвращает true, если заголовок отображается

- applyPriceFilter(int min, int max)

1) Вводит минимальную и максимальную цену из Constants
2) Ожидает обновления результатов после фильтрации

- arePricesInRange(int minPrice, int maxPrice)

1) Проверяет, что все цены товаров попадают в заданный диапазон
2) Преобразует текст цен в числа (удаляет нецифровые символы)

- selectFirstProduct()

1) Кликает на первый товар в списке
2) Ожидает кликабельности элемента

- doProductTitlesContain(String keyword)

1) Проверяет, что хотя бы один товар содержит искомое слово
2) Сравнивает без учета регистра

9. Класс:

ProductPage

ProductPage наследуется от BasePage и реализует функционал для взаимодействия со страницей товара в Яндекс.Маркете. Отвечает за добавление товаров в корзину и переход в неё.

Элементы страницы (локаторы):

1) Кнопка "Добавить в корзину"

Основной элемент для добавления текущего товара в корзину

Расположен в карточке товара

2) Ссылка на корзину

Кликабельный элемент в шапке сайта

Используется для перехода на страницу корзины

3) Иконка корзины

Визуальный индикатор состояния корзины

Отображает количество добавленных товаров

Методы класса:

- addToCart()

1) Нажимает кнопку добавления товара
2) Не возвращает значений (void)

- goToCart()

1) Ожидает кликабельности ссылки
2) Выполняет переход на страницу корзины
3) Возвращает экземпляр CartPage

- waitForCartIconUpdate(String expectedText)

1) Ожидает появления указанного текста в иконке
2) Используется для проверки успешного добавления товара

10. Класс:

LoginPage

Класс LoginPage наследуется от BasePage и представляет собой страницу входа (логина) в системе. Он содержит методы для взаимодействия с элементами страницы и выполнения процесса аутентификации.

Поля класса (WebElement-ы):

- moreOptionsButton – кнопка "Больше вариантов" для отображения дополнительных способов входа.
- usernameInput – поле для ввода имени пользователя.
- submitLoginButton – кнопка подтверждения ввода логина.
- passwordInput – поле для ввода пароля.

- submitPasswordButton – кнопка подтверждения ввода пароля.

Конструктор:

LoginPage(WebDriver driver) – инициализирует страницу, передавая драйвер в родительский класс BasePage.

Методы класса:

login(String username, String password)

Основной метод для выполнения полного процесса входа.

Последовательно вызывает:

- clickMoreOptionsButton()

1) Кликает по кнопке "Больше вариантов" после ожидания ее доступности.
2) Логирует действие через LoggerUtil.logInfo.

- selectLoginByUsername()

1) Выбирает опцию входа по имени пользователя.
2) Пытается найти элемент сначала по одному XPath, затем по альтернативному (если первый не сработал).
3) В случае ошибки логирует проблему через LoggerUtil.logError и выбрасывает исключение.

- enterUsername(String username)

1) Вводит переданное имя пользователя в соответствующее поле после ожидания его видимости.
2) Логирует действие.

- clickSubmitLoginButton()

1) Кликает по кнопке подтверждения логина после ожидания ее доступности.
2) Логирует действие.

- enterPassword(String password)

1) Вводит переданный пароль в соответствующее поле после ожидания его видимости.
2) Логирует действие.

- clickSubmitPasswordButton()

1) Кликает по кнопке подтверждения пароля после ожидания ее доступности.
2) Логирует действие и возвращает новый экземпляр HomePage.

11. Класс:

HomePage

Класс HomePage наследуется от BasePage и представляет главную страницу Яндекс.Маркета. Он содержит методы для взаимодействия с основными элементами страницы, включая поиск, каталог и хедер.

Поля класса:

- header – экземпляр HeaderComponent для работы с элементами хедера.
- searchInput – поле ввода поискового запроса.
- searchButton – кнопка запуска поиска.
- catalogButton – кнопка открытия каталога товаров.
- catalogMenu – меню каталога (после его открытия).

Конструктор:

HomePage(WebDriver driver)

Инициализирует страницу, передавая драйвер в родительский класс BasePage, а также создает экземпляр HeaderComponent для работы с хедером.

Методы класса:

- header()

Возвращает компонент хедера (HeaderComponent), чтобы взаимодействовать с его элементами (например, корзиной, профилем и т. д.).

- isTitleCorrect()

Проверяет, содержит ли заголовок страницы текст "Яндекс Маркет".

Возвращает true, если заголовок корректен, иначе false.

- enterSearchText(String text)

Очищает поле поиска и вводит переданный текст.

clickSearchButton()

Нажимает кнопку поиска (после ввода запроса).

- clickCatalogButton()

Нажимает кнопку каталога (в текущей реализации кликает на catalogMenu, что может быть ошибкой).

- isCatalogMenuDisplayed()

Проверяет, отображается ли меню каталога.

Возвращает true, если меню видно, иначе false.

- openElectronicsCategory()

Открывает каталог (кликает catalogButton), но в текущей реализации не выбирает конкретную категорию (закомментировано).

12. Класс:

CartPage

Класс CartPage наследуется от BasePage и представляет страницу корзины Яндекс.Маркета. Он содержит методы для проверки состояния корзины, удаления товаров и получения информации о количестве товаров.

Поля класса (WebElement-ы):

- cartItemCount – элемент, отображающий количество товаров в корзине.
- removeButton – кнопка удаления товара из корзины.
- submitRemove – кнопка подтверждения удаления товара.
- emptyCartMessage – сообщение о пустой корзине.

- cartItem – контейнер товара в корзине (в текущей реализации не используется).

Конструктор:

CartPage(WebDriver driver)

Инициализирует страницу, передавая драйвер в родительский класс BasePage.

Методы класса:

- isItemInCart()

1) Проверяет, есть ли товары в корзине.
2) Получает текст из cartItemCount и преобразует его в число.
3) Возвращает true, если количество товаров > 0, иначе false.
4) Логирует текущее количество товаров через LoggerUtil.logInfo.
5) В случае ошибки логирует проблему через LoggerUtil.logError и возвращает false.

- removeItem()

1) Удаляет товар из корзины:
2) Кликает кнопку удаления (removeButton).
3) Подтверждает удаление (submitRemove).
4) Ожидает появления сообщения о пустой корзине (emptyCartMessage).
5) В случае ошибки выбрасывает исключение RuntimeException.

- isEmpty()

1) Проверяет, пуста ли корзина:
2) Если отображается emptyCartMessage, возвращает true.
3) Если сообщение не найдено, проверяет текст cartItemCount.
4) Возвращает true, если количество товаров = 0 или текст пуст, иначе false.

- getCartItemsCount()

1) Возвращает количество товаров в корзине:
2) Если товары есть (isItemInCart()), парсит число из cartItemCount.

3) Если корзина пуста, возвращает 0.

13. Класс:

HeaderComponent

Класс HeaderComponent наследуется от BaseComponent и представляет хедер (верхнюю панель) страницы Яндекс.Маркета. Он содержит методы для взаимодействия с основными элементами хедера: кнопками входа, корзины и поиска.

Поля класса (WebElement-ы):

- loginButton – кнопка входа в аккаунт.
- AccButton – кнопка/иконка профиля (используется для проверки отображения).
- cartButton – кнопка корзины.
- cartFromLobbyButton – альтернативная кнопка корзины (например, на главной странице).
- searchInput – поле поиска.

Конструктор:

HeaderComponent(WebDriver driver)

Инициализирует компонент, передавая драйвер в родительский класс BaseComponent.

Методы класса:

- clickLoginButton()

Кликает по кнопке входа (loginButton) и возвращает экземпляр LoginPage.

- clickCartButton()

Кликает по основной кнопке корзины (cartButton) и возвращает экземпляр CartPage.

- clickCartFromLobbyButton()

Кликает по альтернативной кнопке корзины (cartFromLobbyButton) и возвращает экземпляр CartPage.

- isLoginButtonDisplayed()

Проверяет, отображается ли кнопка/иконка профиля (AccButton).

Возвращает true, если элемент виден.

- isCartButtonDisplayed()

Проверяет, отображается ли альтернативная кнопка корзины (cartFromLobbyButton).

Возвращает true, если элемент виден.

- isSearchInputDisplayed()

Проверяет, отображается ли поле поиска (searchInput).

Возвращает true, если элемент виден.

14. Класс:

SearchFilterComponent

Класс SearchFilterComponent наследуется от BaseComponent и представляет компонент фильтров поиска на странице результатов Яндекс.Маркета. Он содержит методы для установки ценового диапазона и применения фильтров.

Поля класса (WebElement-ы):

- minPriceInput – поле для ввода минимальной цены (XPath локатор).
- maxPriceInput – поле для ввода максимальной цены (XPath локатор).
- applyButton – кнопка применения фильтров (CSS локатор).

Конструктор:

SearchFilterComponent(WebDriver driver)

Инициализирует компонент, передавая драйвер в родительский класс BaseComponent.

Методы класса:

- setMinPrice(int price)

Устанавливает минимальную цену:

Очищает поле minPriceInput.

Вводит переданное значение цены.

- setMaxPrice(int price)

Устанавливает максимальную цену:

Очищает поле maxPriceInput.

Вводит переданное значение цены.

- applyFilters()

Кликает по кнопке применения фильтров (applyButton).

15. Класс:

CatalogComponent

Класс CatalogComponent наследуется от BaseComponent и представляет собой компонент каталога в интерфейсе Яндекс.Маркета. Этот класс содержит методы для взаимодействия с элементами каталога на веб-странице.

Поля класса:

- catalogButton - веб-элемент кнопки открытия каталога
- catalogMenu - веб-элемент меню каталога, конкретно пункт электроники

Конструктор:

CatalogComponent(WebDriver driver) - создает новый экземпляр компонента каталога, принимая драйвер веб-браузера в качестве параметра

Методы класса:

- openCatalog() - кликает по кнопке каталога, открывая его

- selectElectronicsCategory() - выбирает категорию "Электроника" в меню каталога
- isCatalogMenuDisplayed() - проверяет, отображается ли меню каталога на странице (возвращает boolean)

16. Класс:

NotificationComponent

Класс NotificationComponent наследуется от BaseComponent и представляет компонент уведомлений в интерфейсе Яндекс.Маркета. Этот класс содержит методы для работы с всплывающими уведомлениями на веб-странице.

Поля класса:

- notificationPopup - веб-элемент всплывающего уведомления
- closeNotificationButton - веб-элемент кнопки закрытия уведомления

Конструктор:

NotificationComponent(WebDriver driver) - создает новый экземпляр компонента уведомлений, принимая драйвер веб-браузера в качестве параметра

Методы класса:

- closeNotificationIfPresent()

1) Ожидает появления всплывающего уведомления
2) Если уведомление появляется, нажимает кнопку "Закрыть"
3) Если уведомление не появляется (выбрасывается исключение), метод завершается без действий

- isNotificationDisplayed()

1) Проверяет, отображается ли уведомление на странице
2) Возвращает true, если уведомление видимо

3) Возвращает false, если уведомление не найдено или не видимо (перехватывает исключение)

17. Класс:

SearchComponent

Класс SearchComponent наследуется от BaseComponent и представляет компонент поиска в интерфейсе Яндекс.Маркета. Этот класс содержит методы для взаимодействия с поисковой строкой и кнопкой поиска.

Поля класса:

- searchInput – поле ввода поискового запроса
- searchButton – кнопка запуска поиска

Конструктор:

SearchComponent(WebDriver driver) – инициализирует компонент, принимая экземпляр WebDriver

Методы класса:

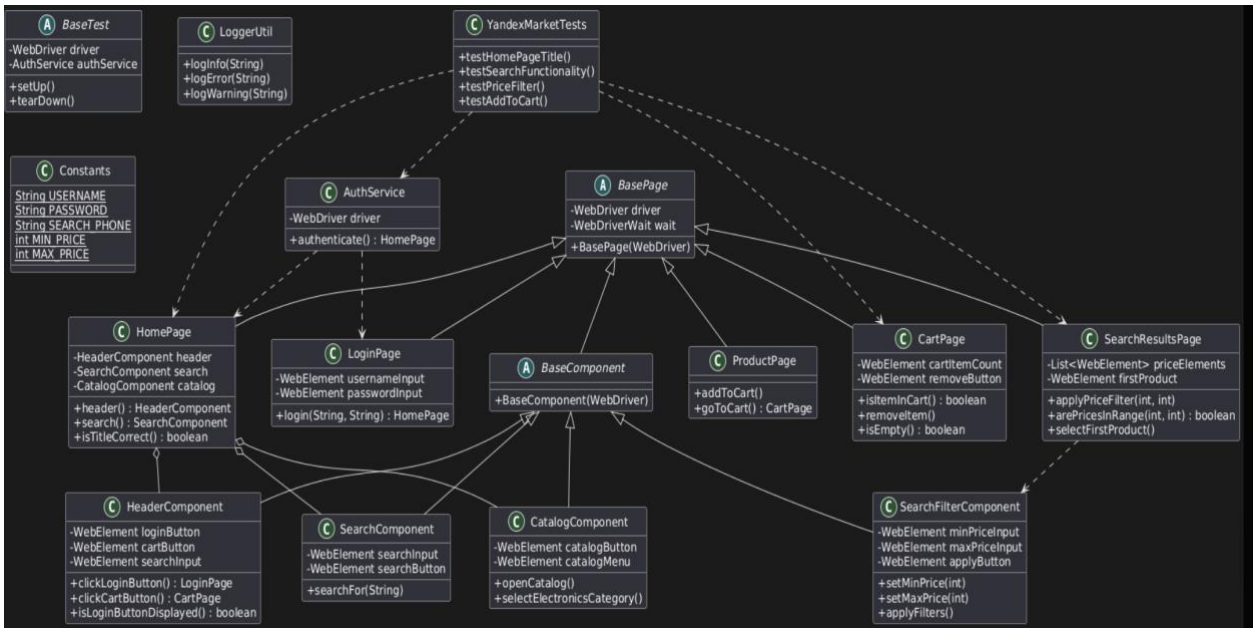
- searchFor(String text)

1) Очищает поле поиска (searchInput.clear()).
2) Вводит переданный текст (searchInput.sendKeys(text)).
3) Нажимает кнопку поиска (searchButton.click()).
4) Назначение:
5) Выполняет поиск товаров по заданному тексту.

- isSearchInputDisplayed()

1) Проверяет, отображается ли поле поиска на странице (searchInput.isDisplayed()).
2) Возвращает true, если поле видимо, и false в противном случае.

2.2. UML-диаграмма



3. ТРЕТИЙ РАЗДЕЛ

1.1. Тестирование одного теста

Для описания выбран тест «Добавление товара в корзину»

1. Программа заходит на сайт Яндекс маркет и проходит регистрацию на сайте. Выбирает войти по логину , вводит почту и пароль(рис.1-рис.3).

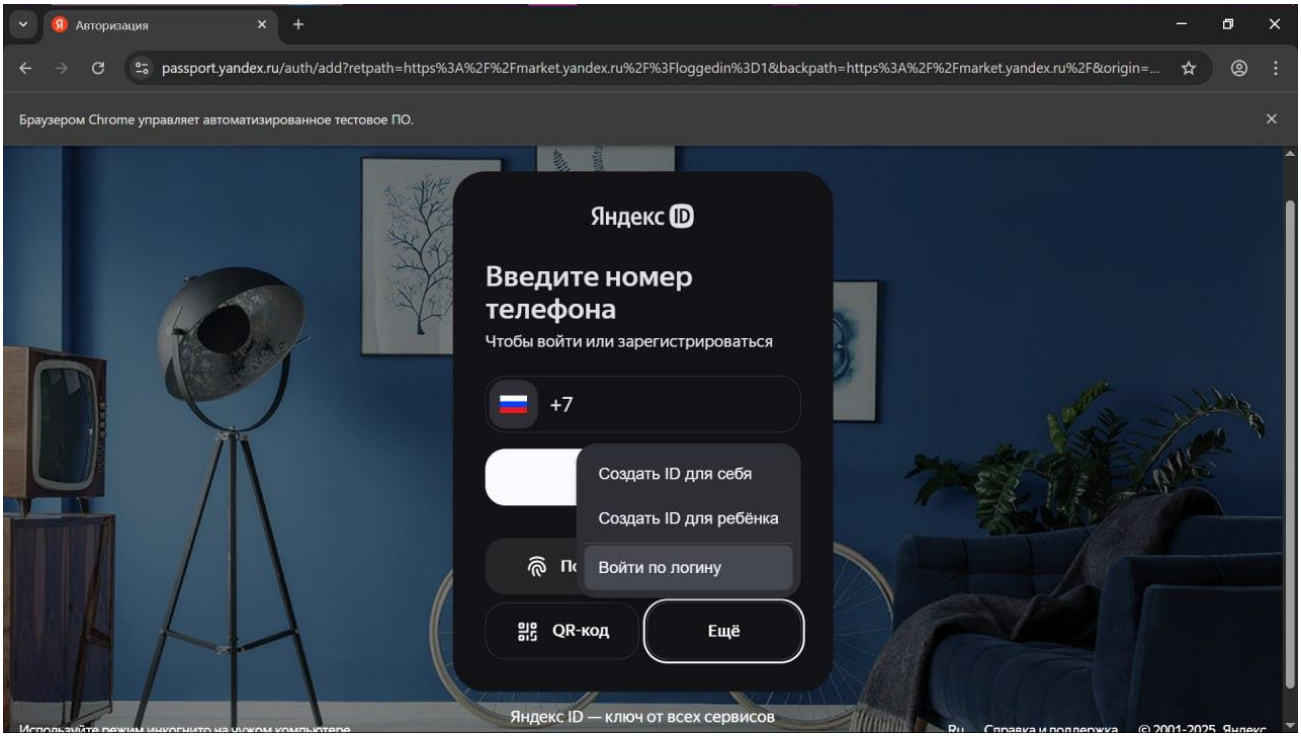


Рисунок 1 - Авторизация

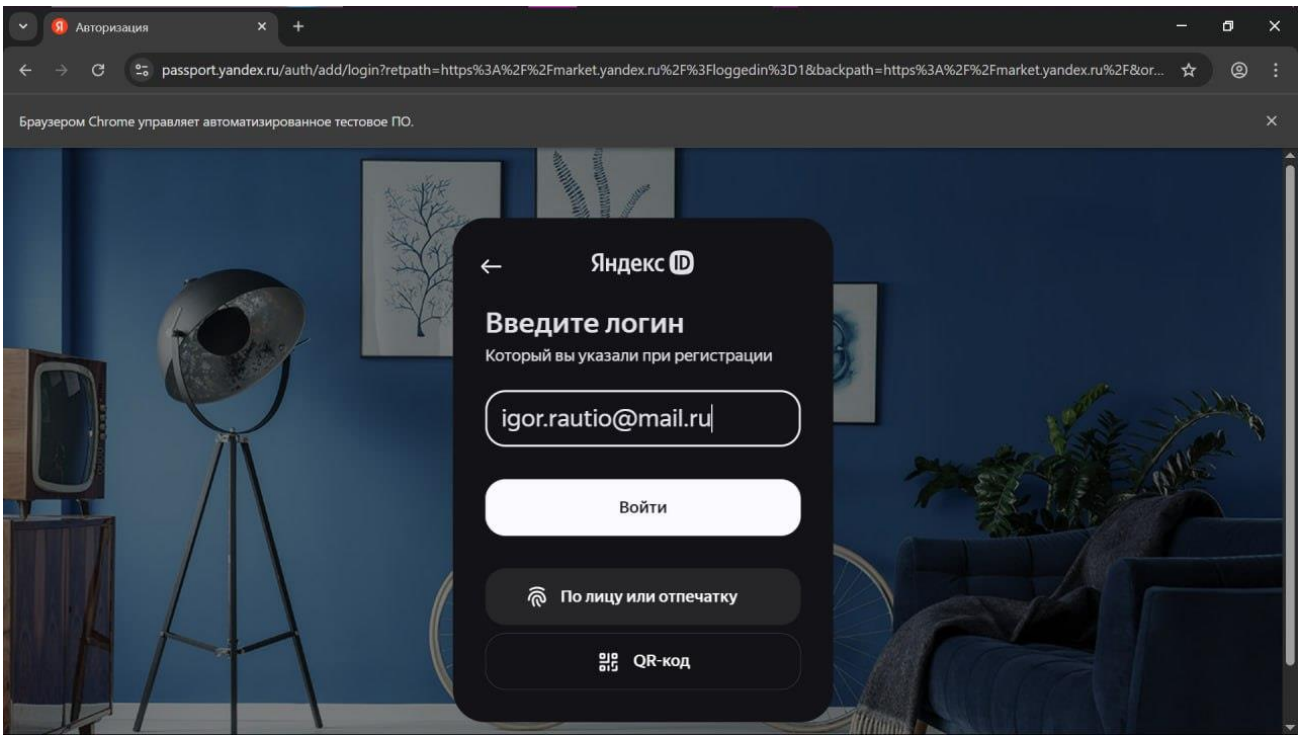


Рисунок 2 – Ввод логина

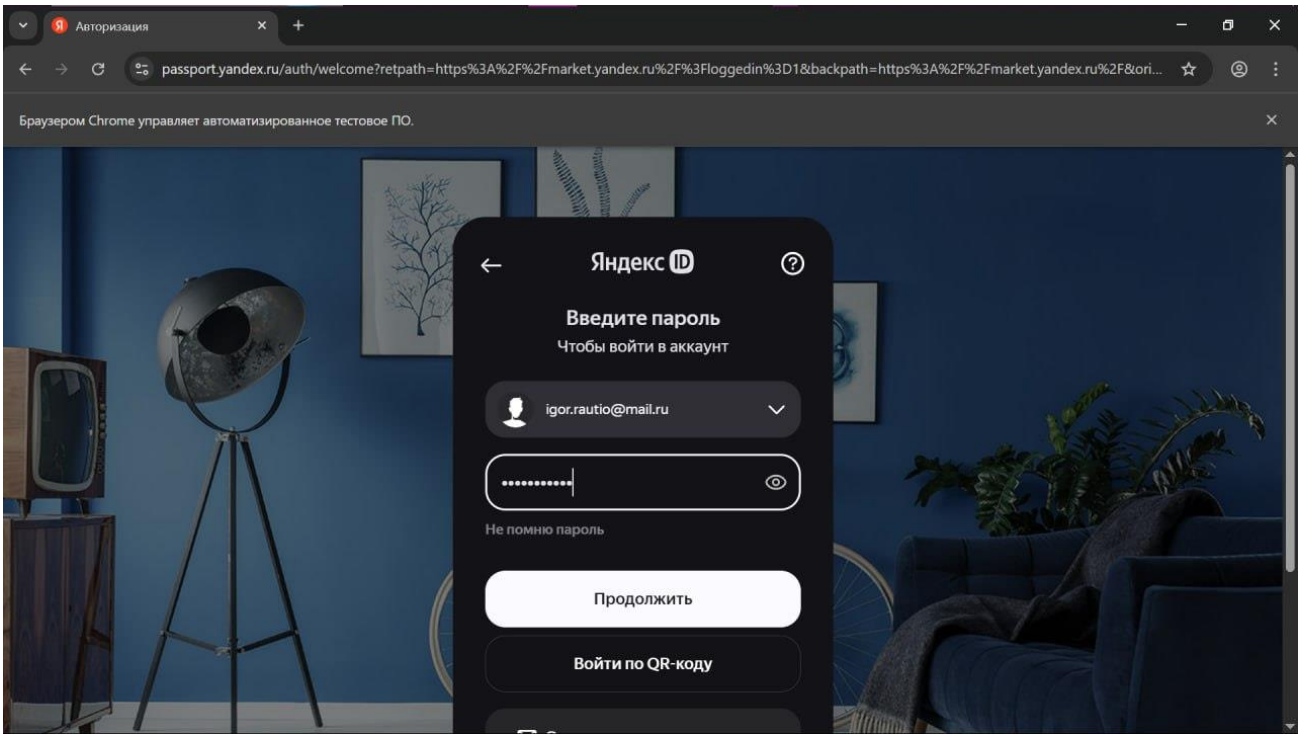


Рисунок 3 – Ввод пароля

2. Далее в поисковой строке вводится запрос «ноутбук», клик по кнопке «Найти», выбор товара по запросу(рис.4)

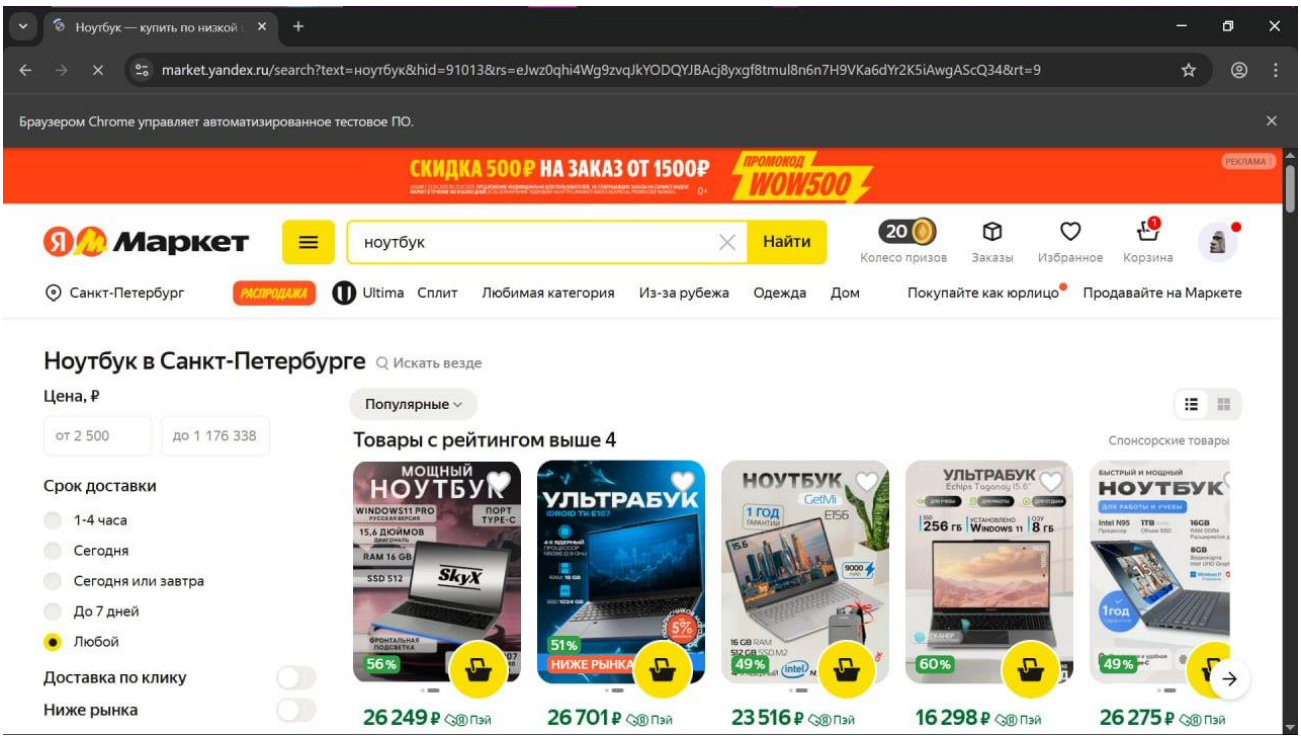


Рисунок 4 – Ввод товара в строку поиска

3. Программа переходит к карточке выбранного товара и добавляет его в корзину(рис.5)

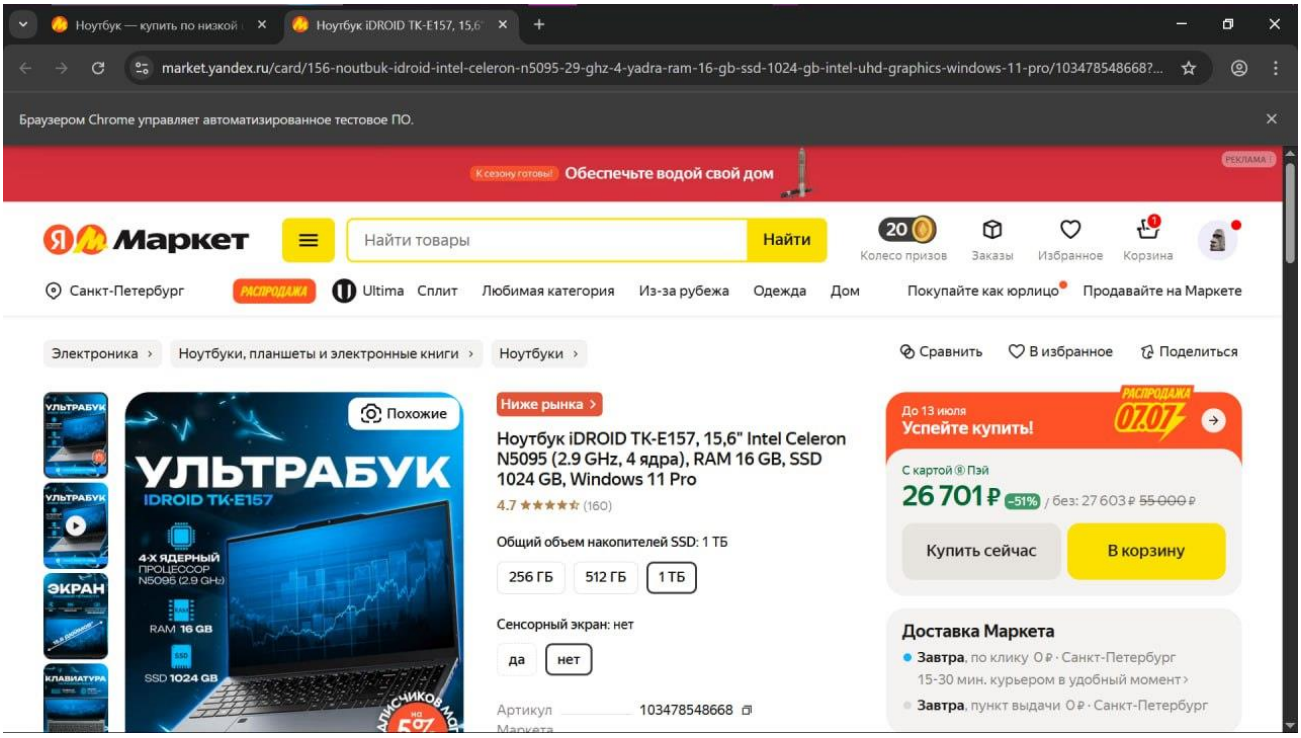


Рисунок 5 – Добавление товара в корзину

4. Происходит проверка, что товар добавлен в корзину(рис.6)

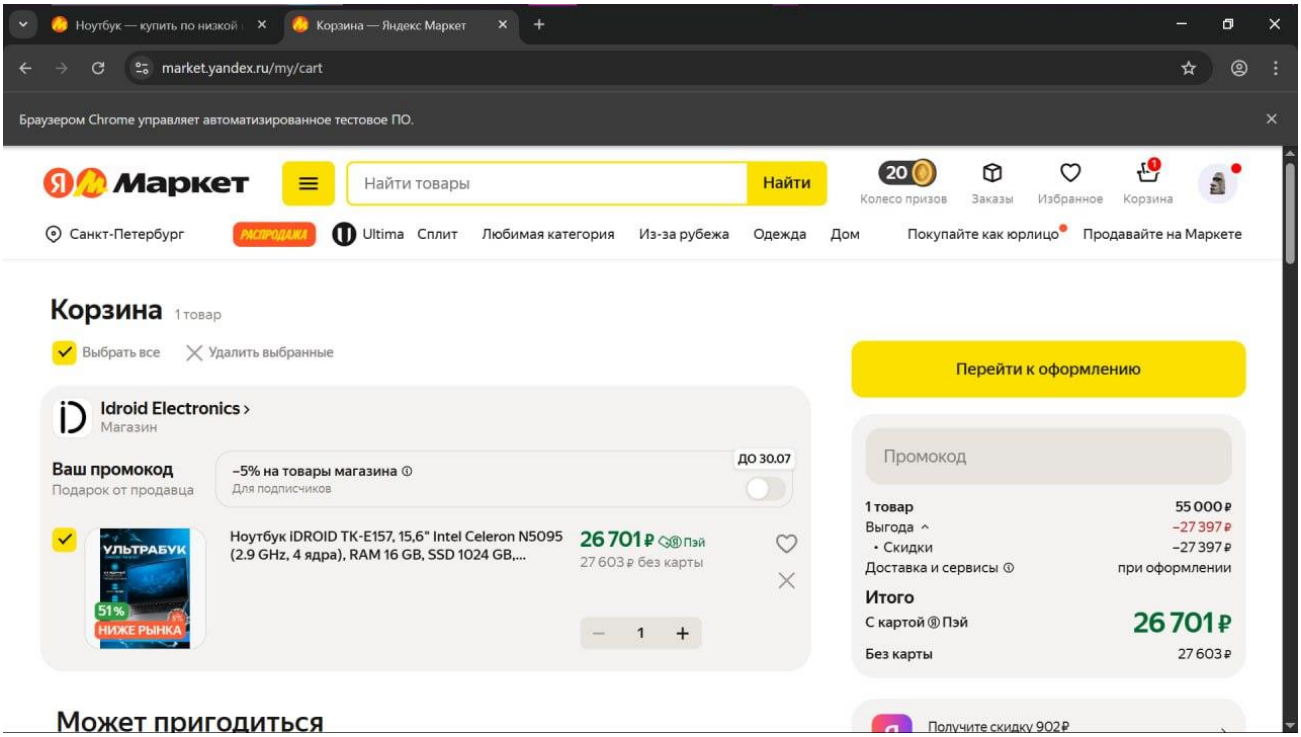


Рисунок 6 – Переход в корзину

5. На рисунке 7 представлены поэтапные шаги.

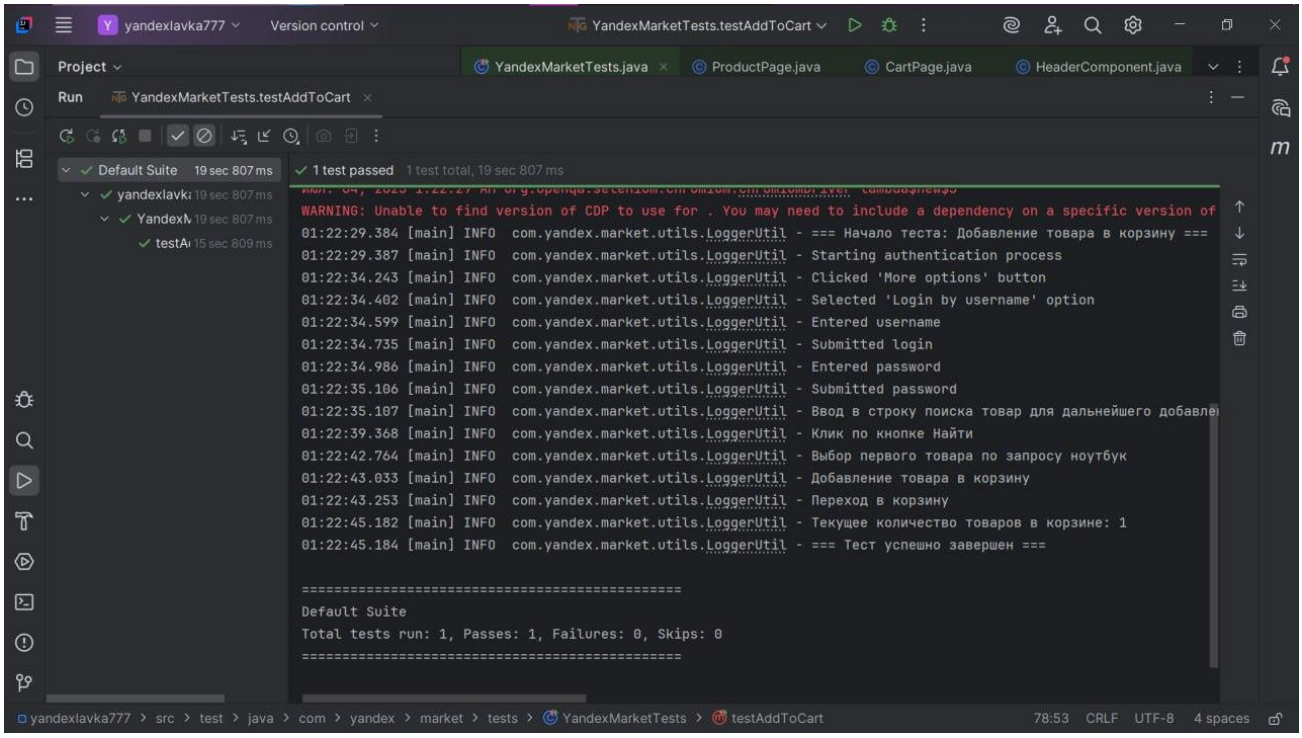
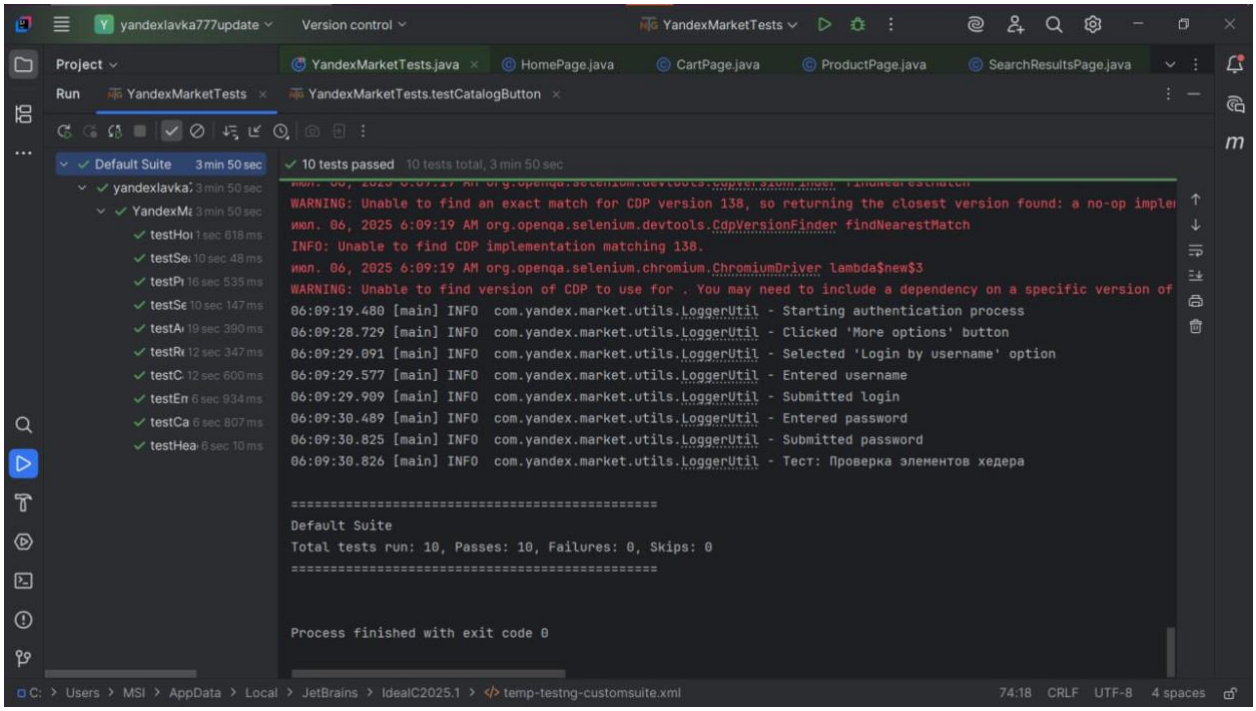


Рисунок 7 – Логгер для тестов

3.2. Успешная обработка всех тестов



ЗАКЛЮЧЕНИЕ

В ходе выполнения практики по UI-тестированию была проведена работа по разработке и реализации автоматизированных тестов для веб-приложения Яндекс.Маркет с использованием современных технологий, таких как Java, Selenide (на основе Selenium), JUnit 5, Maven и инструменты логирования. Было создано 10 автотестов, охватывающих ключевые функциональные блоки системы, включая авторизацию, поиск товаров, фильтрацию по цене, добавление и удаление товаров из корзины, проверку элементов интерфейса и навигацию по каталогу.

Цель практики — освоить методики автоматизации тестирования веб-приложений и применить их на реальном проекте — была успешно достигнута. В процессе работы были закреплены навыки написания чистого, поддерживаемого кода, структурирования проекта, организации совместной работы через GitHub, а также документирование тестовых сценариев в формате чеклиста. Все этапы практики были выполнены в соответствии с заданными требованиями и сроками.

Одним из ключевых результатов стало создание модульной и гибкой архитектуры тестовой системы. Проект был организован с использованием принципов объектно-ориентированного программирования: были выделены абстрактные базовые классы `*BasePage*` и `*BaseComponent*`, от которых наследуются конкретные страницы и компоненты. Это позволило минимизировать дублирование кода, повысить его читаемость и упростить сопровождение. Также были реализованы специализированные сервисы, такие как `*AuthService*` для управления аутентификацией, `*LoggerUtil*` для централизованного логирования, `Constants` для хранения глобальных переменных и параметров. Такой подход способствовал созданию устойчивой и легко расширяемой тестовой среды.

Большое внимание было уделено качеству документирования. Чеклист с описанием всех тестов, шагов, входных данных и ожидаемых результатов обеспечил прозрачность и воспроизводимость тестирования. Каждый тестовый сценарий был тщательно продуман и протестирован на соответствие бизнес-логике приложения. Например, тесты «Поиск товара» и «Релевантность поиска» позволяют убедиться в том, что система правильно обрабатывает пользовательские запросы и возвращает релевантные результаты. Тесты «Добавление товара в корзину» и «Удаление товара из корзины» проверяют корректную работу одного из самых важных пользовательских сценариев — покупки товара.

Также была реализована система логирования, которая позволяет фиксировать все действия во время выполнения тестов. Это помогает быстро находить и анализировать ошибки, а также отслеживать прогресс выполнения тестового набора. Логирование выполнялось с помощью утилитарного класса `*LoggerUtil*`, который предоставляет удобный интерфейс для вывода информации, предупреждений и сообщений об ошибках.

Помимо технической реализации, значительное внимание было уделено организации командной работы. Работа велась в группе из трех человек через систему контроля версий GitHub, где каждый участник нес ответственность за определенную часть проекта. Это способствовало развитию навыков совместной разработки, эффективного взаимодействия и использования Git-стратегий.

Все тесты были успешно запущены и показали высокую степень покрытия проверяемой функциональности. На этапе демонстрации работы тестов были представлены скриншоты, подтверждающие успешное выполнение каждого сценария. Было проверено состояние корзины,

работоспособность поиска, корректность фильтрации и отображение основных элементов интерфейса.

Таким образом, результаты практики показали, что разработанные автотесты полностью соответствуют поставленным задачам. Они обеспечивают надежное покрытие ключевых функций Яндекс.Маркета, могут быть легко адаптированы к изменениям в интерфейсе и расширены для проверки новых возможностей системы. Полученный опыт и навыки позволили не только углубить знания в области автоматизации тестирования, но и получить практические навыки работы с современными инструментами разработки, которые будут полезны в дальнейшей профессиональной деятельности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Васильев А.Н. «Программирование на Java»
2. Яндекс Маркет https://market.yandex.ru
3. https://www.selenium.dev/documentation/webdriver/ui_testing/ *(Официальная документация Selenium по тестированию пользовательского интерфейса)*
4. Мэйвороно, Энтони Test-Driven Development with Selenium — Packt Publishing, 2021. (Подробное описание работы с UI-тестами через Selenium и Selenide
5. https://yandex.ru/support/partners/market/index.html (Официальные материалы по функционалу Яндекс Маркета, полезны для понимания логики интерфейса)

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл BaseComponent.java:

```
package com.yandex.market.base;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

/
 * Базовый класс для компонентов страницы.
 * Обеспечивает переиспользование общих элементов.
 */
public abstract class BaseComponent extends BasePage {

    /
     * Конструктор базового компонента.
     *
     * @param driver WebDriver экземпляр
     */
    protected BaseComponent(WebDriver driver) {
        super(driver);
        PageFactory.initElements(driver, this);
    }
}
```

Файл BasePage.java:

```
package com.yandex.market.base;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;

public abstract class BasePage {
    protected final WebDriver driver;
    protected final WebDriverWait wait;

    protected BasePage(WebDriver driver) {
        this.driver = driver;
        this.wait = new WebDriverWait(driver, Duration.ofSeconds(3));
        PageFactory.initElements(driver, this);
    }
}
```

Файл BaseTest.java:

```
package com.yandex.market.base;

import com.yandex.market.pages.HomePage;
```

```
import com.yandex.market.services.AuthService;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.PageLoadStrategy;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;

import java.time.Duration;

public abstract class BaseTest {
    protected WebDriver driver;
    protected AuthService authService;
    protected HomePage homePage;

    @BeforeMethod
    public void setUp() {
        WebDriverManager.chromedriver().setup();

        ChromeOptions options = new ChromeOptions();
        options.addArguments(
                "--start-maximized",
                "--remote-allow-origins=*",
                "--disable-notifications",
                "--disable-popup-blocking"
        );
        options.setPageLoadStrategy(PageLoadStrategy.NORMAL);

        driver = new ChromeDriver(options);
         driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(1
0));
        authService = new AuthService(driver);
        homePage = authService.authenticate();
    }

    @AfterMethod
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

Файл CatalogComponent.java:

```
package com.yandex.market.components;

import com.yandex.market.base.BaseComponent;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class CatalogComponent extends BaseComponent {
```

```
                                        @FindBy(xpath                    =
"//*[@id=\"/content/header/header/catalogEntrypoint\"]/div/noindex[1]/
button")
    private WebElement catalogButton;

                                        @FindBy(xpath                    =
"//*[@id=\"/MarketNodeHeaderCatalog42\"]/div/div[1]/div/ul/li[5]/a")
    private WebElement catalogMenu;

    public CatalogComponent(WebDriver driver) {
        super(driver);
    }

    public void openCatalog() {
        catalogButton.click();
    }

    public void selectElectronicsCategory() {
        catalogMenu.click();
    }

    public boolean isCatalogMenuDisplayed() {
        return catalogMenu.isDisplayed();
    }
}
```

Файл HeaderComponent.java:

```
package com.yandex.market.components;

import com.yandex.market.base.BaseComponent;
import com.yandex.market.pages.CartPage;
import com.yandex.market.pages.LoginPage;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

/**
 * Компонент хедера страницы.
 */
public class HeaderComponent extends BaseComponent {
                                        @FindBy(xpath                    =
"//*[@id=\"/content/header/header/personalization/profile\"]/div/a")
    private WebElement loginButton;

                                        @FindBy(xpath                    =
"//*[@id=\"/content/header/header/personalization/profile\"]/div/butto
n/img")
    private WebElement AccButton;


                                        @FindBy(xpath                    =
"//*[@id=\"CART_ENTRY_POINT_ANCHOR\"]/a/div/div[1]/span")
    private WebElement cartButton;

    @FindBy(xpath = "//*[@id=\"basketOutline\"]")
    private WebElement cartFromLobbyButton;
```

```
    @FindBy(xpath = "//*[@id=\"header-search\"]")
    private WebElement searchInput;

    public HeaderComponent(WebDriver driver) {
        super(driver);
    }

    public LoginPage clickLoginButton() {
        loginButton.click();
        return new LoginPage(driver);
    }

    public CartPage clickCartButton() {
        cartButton.click();
        return new CartPage(driver);
    }
    public CartPage clickCartFromLobbyButton() {
        cartFromLobbyButton.click();
        return new CartPage(driver);
    }

    public boolean isLoginButtonDisplayed() {
        return AccButton.isDisplayed();
    }

    public boolean isCartButtonDisplayed() {
        return cartFromLobbyButton.isDisplayed();
    }

    public boolean isSearchInputDisplayed() {
        return searchInput.isDisplayed();
    }
}
```

Файл NotificationComponent.java:

```
package com.yandex.market.components;

import com.yandex.market.base.BaseComponent;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.ui.ExpectedConditions;

public class NotificationComponent extends BaseComponent {
    @FindBy(xpath = "//div[contains(@class, 'notification')]")
    private WebElement notificationPopup;

    @FindBy(xpath = "//button[contains(text(), 'Закрыть')]")
    private WebElement closeNotificationButton;

    public NotificationComponent(WebDriver driver) {
        super(driver);
    }

    public void closeNotificationIfPresent() {
```

```
        try {
             wait.until(ExpectedConditions.visibilityOf(notificationPop
up));
            closeNotificationButton.click();
        } catch (Exception e) {
            // Уведомление не появилось, ничего не делаем
        }
    }

    public boolean isNotificationDisplayed() {
        try {
            return notificationPopup.isDisplayed();
        } catch (Exception e) {
            return false;
        }
    }
}
```

Файл SearchComponent.java:

```
package com.yandex.market.components;

import com.yandex.market.base.BaseComponent;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class SearchComponent extends BaseComponent {
    @FindBy(xpath = "//*[@id=\"header-search\"]")
    private WebElement searchInput;

                                        @FindBy(xpath                    =
"//*[@id=\"HeaderSearchLightId\"]/div[1]/button/span")
    private WebElement searchButton;

    public SearchComponent(WebDriver driver) {
        super(driver);
    }

    public void searchFor(String text) {
        searchInput.clear();
        searchInput.sendKeys(text);
        searchButton.click();
    }

    public boolean isSearchInputDisplayed() {
        return searchInput.isDisplayed();
    }
}
```

Файл SearchFilterComponent.java:

```
package com.yandex.market.components;

import com.yandex.market.base.BaseComponent;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
```

```
import org.openqa.selenium.support.FindBy;

/
 * Компонент фильтра поиска.
 * Содержит элементы и методы для работы с фильтрами на странице
результатов.
 */
public class SearchFilterComponent extends BaseComponent {

               @FindBy(xpath      =      "//*[@id=\"range-filter-field-
glprice_25563_min\"]")
    private WebElement minPriceInput;

               @FindBy(xpath      =      "//*[@id=\"range-filter-field-
glprice_25563_max\"]")
    private WebElement maxPriceInput;

    @FindBy(css = "[data-auto='filter-apply']")
    private WebElement applyButton;

    public SearchFilterComponent(WebDriver driver) {
        super(driver);
    }

    /
     * Устанавливает минимальную цену.
     *
     * @param price минимальная цена
     */
    public void setMinPrice(int price) {
        minPriceInput.clear();
        minPriceInput.sendKeys(String.valueOf(price));
    }

    /
     * Устанавливает максимальную цену.
     *
     * @param price максимальная цена
     */
    public void setMaxPrice(int price) {
        maxPriceInput.clear();
        maxPriceInput.sendKeys(String.valueOf(price));
    }

    /
     * Применяет установленные фильтры.
     */
    public void applyFilters() {
        applyButton.click();
    }

}
```

Файл CartPage.java:

```
package com.yandex.market.pages;
import com.yandex.market.base.BasePage;
```

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.ui.ExpectedConditions;
import com.yandex.market.utils.LoggerUtil;
import java.util.concurrent.TimeUnit;

public class CartPage extends BasePage {

    // Основной XPath для количества товаров
                                        @FindBy(xpath                    =
"//*[@id=\"/content/page/fancyPage/@chef\\/cart\\/CartLayout/@chef\\/c
art\\/ChefCartList/@light\\/SlotsTheCreator/MARKET/@chef\\/cart\\/Chef
CartHeader/@chef\\/Cart\\/CartHeader\\/Title\"]/div/div[1]/div")
    private WebElement cartItemCount;
    // Локатор кнопки удаления
                                        @FindBy(xpath                    =
"//*[@id=\"/content/page/fancyPage/@chef\\/cart\\/CartLayout/@chef\\/c
art\\/ChefCartList/@light\\/SlotsTheCreator/MARKET/@chef\\/Cart\\/Cart
Header\\/RemoveButton\"]/div/button/span/span")
    private WebElement removeButton;

                                        @FindBy(xpath                    =
"//*[@id=\"/content/page/fancyPage/@chef\\/cart\\/CartLayout/@chef\\/c
art\\/ChefCartList/@light\\/SlotsTheCreator/MARKET/@chef\\/Cart\\/Cart
Header\\/RemoveButton\"]/div[2]/div[2]/div/div[2]/button[2]")
    private WebElement submitRemove;
    // Локатор сообщения о пустой корзине
                                        @FindBy(xpath                    =
"//*[@id=\"/content/page/fancyPage/@chef\\/cart\\/CartLayout/@chef\\/c
art\\/ChefCartList/@light\\/SlotsTheCreator/@chef\\/cart\\/ChefCartEmp
tyLayout/empty_cart_agitation/empty_cart_agitation\"]/div/div[1]/span[
1]")
    private WebElement emptyCartMessage;

    // Локатор контейнеров товаров
                                        @FindBy(xpath                    =
"//*[@id=\"/content/page/fancyPage/searchContent/searchSerpStatic/cont
ent/content/lazyGenerator/initialContent/serpLayoutItem_666635/serpEnt
ity17514990531420802345427001/incutConstructor/content/content/slot-
f127gfkjsru/addToCartButton\"]/div/div/button/span")
    private WebElement cartItem;

    public CartPage(WebDriver driver) {
        super(driver);
    }

    public boolean isItemInCart() {
        try {
                                            String      countText      =
wait.until(ExpectedConditions.visibilityOf(cartItemCount)).getText();
            int count = Integer.parseInt(countText.replaceAll("[^0-9]",
""));
            LoggerUtil.logInfo("Текущее количество товаров в корзине: "
+ count);
            return count > 0;
        } catch (Exception e) {
```

```
            LoggerUtil.logError("Не удалось проверить количество товаров:
" + e.getMessage());
            return false;
        }
    }


    public void removeItem() {
        try {
             wait.until(ExpectedConditions.elementToBeClickable(removeB
utton)).click();
             wait.until(ExpectedConditions.elementToBeClickable(submitR
emove)).click();
            // Ожидаем обновления корзины
            TimeUnit.SECONDS.sleep(1);
             wait.until(ExpectedConditions.visibilityOf(emptyCartMessag
e));
        } catch (Exception e) {
           throw new RuntimeException("Failed to remove item from cart",
e);
        }
    }

    public boolean isEmpty() {
        try {
                                                                    return
wait.until(ExpectedConditions.visibilityOf(emptyCartMessage)).isDispla
yed();
        } catch (Exception e) {
            try {
                                                String    countText    =
cartItemCount.getText().replaceAll("\\D", "");
              return countText.isEmpty() || Integer.parseInt(countText)
== 0;
            } catch (Exception ex) {
                return true;
            }
        }
    }

    public int getCartItemsCount() {

            if (isItemInCart()) {
                                            WebElement    countElement    =
wait.until(ExpectedConditions.visibilityOf(cartItemCount));
                                                String    countText    =
countElement.getText().replaceAll("\\D", "");
                return Integer.parseInt(countText);
            }
            return 0;
        }
    }
```

Файл HomePage.java:

```
package com.yandex.market.pages;
```

```
import com.yandex.market.base.BasePage;
import com.yandex.market.components.*;
import org.openqa.selenium.WebDriver;

public class HomePage extends BasePage {
    private final HeaderComponent header;
    private final SearchComponent search;
    private final CatalogComponent catalog;
    private final NotificationComponent notification;

    public HomePage(WebDriver driver) {
        super(driver);
        this.header = new HeaderComponent(driver);
        this.search = new SearchComponent(driver);
        this.catalog = new CatalogComponent(driver);
        this.notification = new NotificationComponent(driver);
    }

    public HeaderComponent header() {
        return header;
    }

    public SearchComponent search() {
        return search;
    }

    public CatalogComponent catalog() {
        return catalog;
    }

    public NotificationComponent notification() {
        return notification;
    }

    public boolean isTitleCorrect() {
        return driver.getTitle().contains("Яндекс Маркет");
    }
}
```

Файл LoginPage.java:

```
package com.yandex.market.pages;

import com.yandex.market.base.BasePage;
import com.yandex.market.utils.LoggerUtil;
import org.openqa.selenium.By;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.ui.ExpectedConditions;

import java.util.List;

public class LoginPage extends BasePage {

    @FindBy(xpath = "//*[@id=\"passp:exp-register\"]")
```

```
    private WebElement moreOptionsButton;

    @FindBy(xpath = "//*[@id=\"passp-field-login\"]")
    private WebElement usernameInput;

    @FindBy(xpath = "//*[@id=\"passp:sign-in\"]")
    private WebElement submitLoginButton;

    @FindBy(xpath = "//*[@id=\"passp-field-passwd\"]")
    private WebElement passwordInput;

    @FindBy(xpath = "//*[@id=\"passp:sign-in\"]")
    private WebElement submitPasswordButton;

    public LoginPage(WebDriver driver) {
        super(driver);
    }

    public HomePage login(String username, String password) {
        clickMoreOptionsButton();
        selectLoginByUsername();
        enterUsername(username);
        clickSubmitLoginButton();
        enterPassword(password);
        return clickSubmitPasswordButton();
    }

    private void clickMoreOptionsButton() {
         wait.until(ExpectedConditions.elementToBeClickable(moreOptions
Button)).click();
        LoggerUtil.logInfo("Clicked 'More options' button");
    }

    private void selectLoginByUsername() {
        try {
            // Первый вариант XPath (новая структура)
                                                String      xpath1      =
"//*[@id=\"root\"]/div/div[2]/div[2]/div/div/div[2]/div[3]/div/div/div
/div/form/div/div[3]/div[2]/div/div[2]/div/div/ul/li[3]/button";

            // Второй вариант XPath (старая структура)
                                                String      xpath2      =
"//*[@id=\"root\"]/div/div[2]/div[2]/div/div[2]/div[2]/div[2]/div/div/
div/div/form/div/div[3]/div[2]/div/div[2]/div/div/ul/li[3]/button";

            // Пытаемся найти по первому XPath
                                        List<WebElement>     options     =
driver.findElements(By.xpath(xpath1));

            // Если первый не найден, пробуем второй
            if (options.isEmpty()) {
                options = driver.findElements(By.xpath(xpath2));
            }

            // Если хотя бы один вариант найден
            if (!options.isEmpty()) {
                WebElement element = options.get(0);
```

```
                wait.until(ExpectedConditions.elementToBeClickable(ele
ment)).click();
                    LoggerUtil.logInfo("Selected 'Login by username'
option");
            } else {
                throw new NoSuchElementException("Login option not found
with either XPath");
            }
        } catch (Exception e) {
            // Используем новый метод logError
              LoggerUtil.logError("Error selecting login option: " +
e.getMessage());
            throw e;
        }
    }

    private void enterUsername(String username) {
         wait.until(ExpectedConditions.visibilityOf(usernameInput)).sen
dKeys(username);
        LoggerUtil.logInfo("Entered username");
    }

    private void clickSubmitLoginButton() {
         wait.until(ExpectedConditions.elementToBeClickable(submitLogin
Button)).click();
        LoggerUtil.logInfo("Submitted login");
    }

    private void enterPassword(String password) {
         wait.until(ExpectedConditions.visibilityOf(passwordInput)).sen
dKeys(password);
        LoggerUtil.logInfo("Entered password");
    }

    private HomePage clickSubmitPasswordButton() {
         wait.until(ExpectedConditions.elementToBeClickable(submitPassw
ordButton)).click();
        LoggerUtil.logInfo("Submitted password");
        return new HomePage(driver);
    }
}
```

Файл ProductPage.java:

```
package com.yandex.market.pages;

import com.yandex.market.base.BasePage;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.ui.ExpectedConditions;

public class ProductPage extends BasePage {
                @FindBy(xpath       =       "//*[@data-apiary-widget-
name='@light/AddToCartButton']")
    private WebElement addToCartButton;
```

```
    @FindBy(xpath = "//*[@id=\"basketOutline\"]")
    private WebElement cartLink;

    @FindBy(xpath = "//*[@id=\"basketOutline\"]")
    private WebElement cartIcon;
    public ProductPage(WebDriver driver) {
        super(driver);
    }

    public void clickAddToCartButton() {
        addToCartButton.click();
    }

    public CartPage goToCart() {
         wait.until(ExpectedConditions.elementToBeClickable(cartLink)).
click();
        return new CartPage(driver);
    }

    // Добавляем публичный метод для ожидания обновления иконки
    public void waitForCartIconUpdate(String expectedText) {
         wait.until(ExpectedConditions.textToBePresentInElement(cartIco
n, expectedText));
    }
}
```

Файл SearchResultsPage.java:

```
package com.yandex.market.pages;

import com.yandex.market.base.BasePage;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.ui.ExpectedConditions;

import java.util.List;

import static com.yandex.market.utils.Constants.MAX_PRICE;
import static com.yandex.market.utils.Constants.MIN_PRICE;

/**
 * Класс для работы со страницей результатов поиска.
 */
public class SearchResultsPage extends BasePage {
                                        @FindBy(xpath                    =
"//*[@id=\"/content/page/fancyPage/searchContent/searchTitleWithBreadc
rumbs\"]/div/div/div/h1")
    private WebElement resultsHeader;

                                        @FindBy(xpath                    =
"//*[@id=\"/content/page/fancyPage/searchContent/searchSerpStatic/cont
ent/content/lazyGenerator/initialContent\"]/div")
    private List<WebElement> priceElements;

    @FindBy(xpath = "(//div[@data-idx][contains(@class, 'iqmYz')])[3]")
```

```
    private WebElement firstProduct;

                                        @FindBy(xpath                    =
"//*[@id=\"5md6ki2ysu4\"]/div[1]/div/div[3]/div/div[1]/div/a/div/div/s
pan/span/span[1]")
    private List<WebElement> productTitles;

              @FindBy(xpath      =      "//*[@id=\"range-filter-field-
glprice_25563_min\"]")
    private WebElement minPriceInput;

              @FindBy(xpath      =      "//*[@id=\"range-filter-field-
glprice_25563_max\"]")
    private WebElement maxPriceInput;

      @FindBy(xpath  =  "(//span[contains(@class,  'ds-text_color_price-
term')])[1]")
    private WebElement firstItemPriceElement;

    // Локатор для названия первого товара
    @FindBy(xpath = "(//span[@itemprop='name' and contains(@class, 'ds-
text_color_text-primary')])[1]")
    private WebElement firstItemTitleElement;


    public SearchResultsPage(WebDriver driver) {
        super(driver);
    }

    public boolean isResultsHeaderDisplayed() {
                                                                    return
wait.until(ExpectedConditions.visibilityOf(resultsHeader)).isDisplayed
();
    }

    public void applyPriceFilter(int min, int max) {
        minPriceInput.sendKeys(String.valueOf(MIN_PRICE));
        maxPriceInput.sendKeys(String.valueOf(MAX_PRICE));
         wait.until(ExpectedConditions.stalenessOf(priceElements.get(0)
));
    }

    public boolean arePricesInRange(int minPrice, int maxPrice) {
         wait.until(ExpectedConditions.visibilityOfAllElements(priceEle
ments));
        return priceElements.stream()
                                                .map(element      ->
Integer.parseInt(element.getText().replaceAll("[^0-9]", "")))
                  .allMatch(price -> price >= MIN_PRICE && price <=
MAX_PRICE);
    }

    public void selectFirstProduct() {
         wait.until(ExpectedConditions.elementToBeClickable(firstProduc
t)).click();
    }
```

```
    public boolean doProductTitlesContain(String keyword) {
         wait.until(ExpectedConditions.visibilityOfAllElements(productT
itles));
        return productTitles.stream()
                                                .anyMatch(element     ->
element.getText().toLowerCase().contains(keyword.toLowerCase()));
    }


        // Метод для получения цены первого товара
        public int getFirstItemPrice() {
             wait.until(ExpectedConditions.visibilityOf(firstItemPriceE
lement));
            String priceText = firstItemPriceElement.getText()
                    .replaceAll("[^0-9]", "");
            return Integer.parseInt(priceText);
        }

        // Метод для получения названия первого товара
        public String getFirstItemTitle() {
             wait.until(ExpectedConditions.visibilityOf(firstItemTitleE
lement));
            return firstItemTitleElement.getText();
        }


}
```

Файл AuthService.java:

```
package com.yandex.market.services;

import com.yandex.market.pages.HomePage;
import com.yandex.market.pages.LoginPage;
import com.yandex.market.utils.Constants;
import com.yandex.market.utils.LoggerUtil;
import org.openqa.selenium.WebDriver;

/
 * Сервис для выполнения операций аутентификации.
 * Инкапсулирует логику входа в систему.
 */
public class AuthService {

    private final WebDriver driver;

    public AuthService(WebDriver driver) {
        this.driver = driver;
    }

    /
     * Выполняет аутентификацию пользователя.
     *
     * @return HomePage главная страница после успешной аутентификации
     */
    public HomePage authenticate() {
```

```
        LoggerUtil.logInfo("Starting authentication process");

        HomePage homePage = new HomePage(driver);
        driver.get("https://market.yandex.ru");

        LoginPage loginPage = homePage.header().clickLoginButton();
        return loginPage.login(Constants.USERNAME, Constants.PASSWORD);
    }
}
```

Файл YandexMarketTests.java:

```
package com.yandex.market.tests;
import com.yandex.market.base.BaseTest;
import com.yandex.market.pages.CartPage;
import com.yandex.market.pages.ProductPage;
import com.yandex.market.pages.SearchResultsPage;
import com.yandex.market.utils.Constants;
import com.yandex.market.utils.LoggerUtil;
import org.testng.Assert;
import org.testng.annotations.Test;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;
import java.util.Set;

public class YandexMarketTests extends BaseTest {

    @Test(priority = 1)
    public void testHomePageTitle() {
        LoggerUtil.logInfo("Проверка заголовка главной страницы");
        driver.get("https://market.yandex.ru");
        String actualTitle = driver.getTitle();
        Assert.assertTrue(actualTitle.contains("Яндекс Маркет"),
                    "Заголовок страницы не содержит 'Яндекс Маркет'.
Фактический заголовок: " + actualTitle);
    }

    @Test(priority = 2)
    public void testSearchFunctionality() {
        LoggerUtil.logInfo("Проверка функциональности поиска");
        homePage.search().searchFor(Constants.SEARCH_PHONE);
        SearchResultsPage resultsPage = new SearchResultsPage(driver);
        Assert.assertTrue(resultsPage.isResultsHeaderDisplayed());
    }

    @Test(priority = 3)
    public void testPriceFilter() {
        LoggerUtil.logInfo("Проверка фильтра по цене");
        homePage.search().searchFor(Constants.SEARCH_LAPTOP);
        SearchResultsPage resultsPage = new SearchResultsPage(driver);
                    resultsPage.applyPriceFilter(Constants.MIN_PRICE,
Constants.MAX_PRICE);
        int firstItemPrice = resultsPage.getFirstItemPrice();
           Assert.assertTrue(firstItemPrice >= Constants.MIN_PRICE &&
firstItemPrice <= Constants.MAX_PRICE,
```

```
                String.format("Цена первого товара %d не попадает в
диапазон %d-%d",
                                    firstItemPrice,  Constants.MIN_PRICE,
Constants.MAX_PRICE));
    }

    @Test(priority = 4)
    public void testSearchRelevance() {
        LoggerUtil.logInfo("Проверка релевантности поиска");
        homePage.search().searchFor(Constants.SEARCH_PHONE);
        SearchResultsPage resultsPage = new SearchResultsPage(driver);
        String firstItemTitle = resultsPage.getFirstItemTitle();
         Assert.assertTrue(firstItemTitle.toLowerCase().contains(Consta
nts.SEARCH_PHONE.toLowerCase()),
                "Первый товар '" + firstItemTitle + "' не содержит '" +
Constants.SEARCH_PHONE + "'");
    }

    @Test(priority = 5)
    public void testAddToCart() {
        LoggerUtil.logInfo("=== Тест: Добавление товара в корзину ===");
        String originalWindow = driver.getWindowHandle();

        try {
            homePage.search().searchFor(Constants.SEARCH_LAPTOP);
                        SearchResultsPage   resultsPage   =   new
SearchResultsPage(driver);
            resultsPage.selectFirstProduct();

            new WebDriverWait(driver, Duration.ofSeconds(10))
                    .until(ExpectedConditions.numberOfWindowsToBe(2));

            Set<String> windows = driver.getWindowHandles();
            for (String window : windows) {
                if (!window.equals(originalWindow)) {
                    driver.switchTo().window(window);
                    break;
                }
            }
            // Добавление в корзину (новый код)
            ProductPage productPage = new ProductPage(driver);
               productPage.clickAddToCartButton(); // Клик по кнопке
добавления

            productPage.clickAddToCartButton();

            // Проверка корзины
            CartPage cartPage = productPage.goToCart();
            driver.navigate().refresh();
          Assert.assertTrue(cartPage.isItemInCart(), "Товар не добавлен
в корзину");
            LoggerUtil.logInfo("=== Тест успешно завершен ===");

        } finally {
            if (driver.getWindowHandles().size() > 1) {
                driver.close();
                driver.switchTo().window(originalWindow);
```

```
            }
        }
    }
@Test(priority = 6)
    public void testRemoveFromCart() {
        LoggerUtil.logInfo("Тест: Удаление товара из корзины");
        CartPage cartPage = homePage.header().clickCartButton();
           Assert.assertTrue(cartPage.isItemInCart(), "Корзина должна
содержать товар перед удалением");
        cartPage.removeItem();
         Assert.assertTrue(cartPage.isEmpty(), "Товар не был удален из
корзины");
    }

    @Test(priority = 7)
    public void testCategoryNavigation() {
        LoggerUtil.logInfo("Тест: Навигация по категориям");
        homePage.catalog().openCatalog();
        homePage.catalog().selectElectronicsCategory();
         Assert.assertTrue(driver.getCurrentUrl().contains("elektronika
"),
                "Не удалось перейти в категорию 'Электроника'");
    }

    @Test(priority = 8)
    public void testEmptyCart() {
        LoggerUtil.logInfo("Тест: Проверка пустой корзины");
                                    CartPage        cartPage        =
homePage.header().clickCartFromLobbyButton();
          Assert.assertTrue(cartPage.isEmpty(), "Корзина должна быть
пустой");
    }

    @Test(priority = 9)
    public void testCatalogButton() {
        LoggerUtil.logInfo("Тест: Проверка кнопки каталога");
        homePage.catalog().openCatalog();
        Assert.assertTrue(homePage.catalog().isCatalogMenuDisplayed(),
                "Меню каталога не отображается");
    }

    @Test(priority = 10)
    public void testHeaderElements() {
        LoggerUtil.logInfo("Тест: Проверка элементов хедера");
        Assert.assertTrue(homePage.header().isLoginButtonDisplayed(),
                "Кнопка входа не отображается");
        Assert.assertTrue(homePage.header().isCartButtonDisplayed(),
                "Кнопка корзины не отображается");
        Assert.assertTrue(homePage.search().isSearchInputDisplayed(),
                "Поле поиска не отображается");
    }
}
```

Файл Constants.java:

```
package com.yandex.market.utils;
```

```
/**
 * Класс для хранения констант проекта.
 * Содержит текстовые значения и настройки.
 */
public class Constants {

    public static final String USERNAME = "igor.rautio@mail.ru";
    public static final String PASSWORD = "1Bhcicb1!=1";
    public static final String SEARCH_PHONE = "iPhone 15";
    public static final String SEARCH_LAPTOP = "ноутбук";
    public static final int MIN_PRICE = 10000;
    public static final int MAX_PRICE = 50000;
    public static final String EXPECTED_CART_ITEMS_COUNT = "1";
    public static final int Items_TO_Check = 3;

     public static final String EXPECTED_EMPTY_CART_TEXT = "Корзина
пуста";
    public static final int DEFAULT_TIMEOUT = 5;
}
```

Файл LoggerUtil.java:

```
package com.yandex.market.utils;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class LoggerUtil {
            private static final Logger logger =
LogManager.getLogger(LoggerUtil.class);

    public static void logInfo(String message) {
        logger.info(message);
    }

    // Добавляем метод для логирования ошибок
    public static void logError(String message) {
        logger.error(message);
    }

    // Можно также добавить метод для предупреждений
    public static void logWarning(String message) {
        logger.warn(message);
    }

}
```