Выпускной проект

**Тестирование веб-приложения Demoblaze**

# Введение

В данной выпускной работе представлено комплексное тестирование веб-приложения Demoblaze, имитирующего работу онлайн магазина цифровой техники. Проект включает в себя разработку тест-плана, создание тестовых случаев, организацию тестовых наборов и чек-листов, а также написание и описание автоматизированных тестов.

# Описание продукта

## Цель и задача продукта

**Цель:** Demoblaze – это демонстрационное веб-приложение, предназначенное для имитации работы онлайн-магазина цифровой техники.

**Задача:** Предоставить платформу для обучения и практики тестирования ПО, а так же для демонстрации возможностей автоматизации тестирования.

## Карта модулей продукта

1. Страница логина — Модуль входа в систему с использованием учетных данных логина и пароля.
2. Страница выбора товаров — Отображение каталога товаров, возможность выбора и добавления товаров корзину.
3. Корзина — Отображение выбранных товаров, возможность удаления товаров, переход к оформлению заказа.
4. Форма подтверждения покупки — Ввод данных для доставки, оплаты, подтверждение заказа.
5. Выпадающий список меню — Доступ к различным разделам сайта, включая разлогин.
6. Страница контактов — Информация о компании.

Тестовые случаи находятся в 5.1

# Цель и задача тестирования продукта

**Цель:** Обеспечить качество и надежность работы веб-приложения  Demoblaze путём проведения комплексного тестирования.

**Задачи:**

1. Создание тест-плана.  
   **1.1.** Определение объема тестирования — Выбор модулей и функций. Определение типов тестирования.  
   **1.2.** Планирование ресурсов —   
   **1.3.** Определение критериев входа и выхода —   
   **1.4.** Планирование отчетности —
2. Разработка тестовых случаев и чек-листов.  
   **2.1.** Создание тестовых случаев —   
   **2.2.** Создание чек листов —   
   **2.3.** Валидация тестовых случаев —
3. Проведение ручного и автоматизированного тестирования.  
   **3.1.** Ручное тестирование —   
   **3.2.** Автоматизация —   
   **3.3.** CI\CD —
4. Анализ результатов тестирования продукта.  
   **4.1.** Анализ дефектов —   
   **4.2.** Оценка качества продукта —   
   **4.3.** Формирование отчетов — Отчетность о проделанных тестах и обнаруженных дефектах. Предоставление отчетов.  
   **4.4.** Рекомендации по улучшению — Предложения по улучшению процесса тестирования и меры по устранению проблем.

# Стратегия тестирования продукта

## Техническое описание проекта

* **Название:** Demoblaze
* **Тех. Документация:** Отсутствует
* **URL:** https://www. Demoblaze.com/
* **Front:** HTML, CSS, JS
* **Back:** Отсутствует
* **Browsers:** Яндекс 24.7.2.1098 x64, Chrome 128.0.6613.85 x64, Firefox 130.0 x64
* **OS:** Win. 10 22H2 19045.5011

## План функционального тестирования

* Проверка всех функциональных возможностей сайта.
* Исследовательское тестирование.
* Тестирование в различных сценариях.

## План тестирования производительности

* Проверка доступности эндпоинтов.
* Проверка времени загрузки страниц.
* Проверка отклика сайта на действия по локаторам.

## План тестирования доступности

* Проверка работы ресурса в различных браузерах и на различных устройствах.
* Проверка отображения всех элементов на всех страницах.
* Проверка удобства пользования ресурсом.

## План тестирования безопасности

* Аудит файловой структуры приложения.
* Работа приложения под нагрузкой.
* Проверка защиты от SQL инъекций.

## Стратегия достижения целей

* Исследовательское тестирование.
* Подготовка тестовой документации.
* Проведение реального тестирования.
* Разработка и написание скриптов авто тестирования.
* Анализ результатов тестирования.

# Список тестовых случаев

## Тестовые случаи

Тестовые случаи составлены в порядке определяемом очередностью доступности функционала в ходе исследовательского тестирования ложась в основу пользовательской истории (5.2)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя | Действия | Примечания |
| 1 | Логин | Страница логина – \*Логин\Пароль\* – “LOGIN” – (Открывается каталог) | Логин: TamaraProcenko Пароль: 1q2w3e4r |
| 2 | Разлогин | Страница логина – \*Логин\Пароль\* – “LOGIN” – (Открывается каталог c категориями цифровой техники) – (Открыта любая страница с меню) – “HOME” – “Log out” |  |
| 3 | Каталог Добавление | Логин - Каталог – 3x “CATEGORIES” | Н. товар: Apple monitor 24 |
| 4 | Корзина Удаление | Логин - Каталог – “CATEGORIES” – “Корзина” – “DELETE” |  |
| 5 | Покупка Успешная | Логин – Каталог – “CATEGORIES ” – “Корзина” – “PLASE ORDER” – \*Name\ COINTRY \City\Credit card\Month\Year\* – “Purchase” – “OK” | Name: ДРУГ ДРУЖИЩЕ  COINTRY:  РОССИЯ City: Новосибирск  Credit card:  123456789876544321 Month: 02  Year:  26 |
| 6 | Покупка Отмена | Логин – Каталог – “CATEGORIES ” – “Корзина” – “PLASE ORDER” – \*Name\ COINTRY \City\Credit card\Month\Year\* – “CLOSE” |  |

## Пользовательская история

Пользовательский путь:

*Логин — Каталог категориями — Добавление товара — Корзина — Подтверждение — Покупка с оплатой — Разлогин*

Оценка удобства:

Логин удобный. Каталог нормальный. Корзина комфортная c описанием преимуществ выбранного продукта. Покупка доступная, кроме того есть возможность выбора товара и оплаты без авторизации.

# Автотестирование

## Описание технологического стека

Стек: Python, Playwright, Selenium, Locust, Pytest, Allure rep.

IDE: PyCharm, Playwright IDE, Playwright WebDriver, Selenium IDE, Selenium WebDriver\*

CI\CD: git, requirements.txt, unix

## Чек лист автоматизации

В рамках проекта была проведена автоматизация тестовых случаев, что позволило значительно повысить эффективность и скорость тестирования. Автоматизации подверглись все тестовые случаи (5.1).

## Реализация

Приложение 1

## Описание работы

В данном проекте тестирование сайта организовано с использованием модульной архитектуры, что обеспечивает удобство поддержки и масштабируемость. Проект состоит из нескольких ключевых файлов, каждый из которых выполняет свою специфическую роль:

Состоит из файлов:

**data\_dbp.py** — содержит все необходимые данные для тестирования, такие как учетные данные пользователей, тестовые данные для заполнения форм и другие константы. Это позволяет централизованно управлять данными и легко их изменять при необходимости.

**locators\_db.py** — хранятся все локаторы (селекторы) для элементов интерфейса и эндпоинты (URL-адреса) для API-запросов. Это обеспечивает единый источник истины для всех локаторов, что упрощает их поддержку и изменение.

**locust\_db.py** — предназначен для нагрузочного тестирования. Он использует библиотеку Locust для имитации большого количества пользователей и проверки производительности сайта под нагрузкой. Это позволяет выявить узкие места и потенциальные проблемы с масштабируемостью.

**test\_dbp\_start.py** и **test\_dbp\_check.py** — представляет собой удобно скомпонованные готовые к запуску тесты, которые проверяют правильность локаторов в файле locators\_dbp.py ,используют шаги из файла test\_steps\_dbp.py. Этот файл объединяет все необходимые шаги в полноценные тестовые сценарии, что упрощает запуск и управление тестаtestми.

**Test\_dbp\_steps.py** и test\_dbp\_checks.ru — находятся пошаговые интерфейс скрипты, которые используют данные из data\_dbp.py и локаторы из locators\_dbp.py. Эти скрипты реализуют основные действия пользователя, такие как логин, добавление товара в корзину, оформление заказа и т.д. Использование шагов позволяет создавать модульные и переиспользуемые тесты.

Отдельно настроен конфигурационный файл **config\_dbp.py**, который предоставляет фикстуры для запуска браузеров и других настроек. Этот файл используется в steps\_db.py для инициализации и управления браузером во время тестирования.

### Преимущества такой архитектуры:

**Модульность:** Каждый файл выполняет свою специфическую роль, что упрощает поддержку и масштабирование.

**Удобство:** Готовые к запуску тесты из .py упрощают управление и запуск тестов.

**Производительность:** Нагрузочное тестирование с использованием Locust позволяет выявить проблемы с производительностью.

## Эксплуатация

В PyCharm в терминале (из корневого каталога) команды:

**Установка зависимостей:**

*pip install -r requirements.txt*

**Запуск всех тестов:**

*pytest -v --headed --slowmo 100 --html=report.html* **test\_saucedemo\_playwright/***run\_sp.py*

-v — это флаг для подробного вывода результатов тестов.

--headed — это флаг для запуска браузера в видимом режиме (без него браузер будет запускаться в фоновом режиме).

--slowmo 100 — это флаг для замедления выполнения тестов на 100 миллисекунд между действиями (полезно для отладки).

**Запуск конкретной группы тестов:**

*pytest -v --headed --slowmo 100 --html=report.html test\_saucedemo\_playwright/run\_sp.py* -m smoke\_market

-m smoke\_market — это флаг для запуска только тех тестов, которые помечены маркером smoke\_market.

**Запуск конкретного теста:**

**pytest -v --headed --slowmo 100 --html=report.html test\_demoblaze\_playwright/run\_sp.py::test\_login\_successful**

**test\_login\_successful – это имя конкретного теста**

## Тест производительности

Сбор страниц и методов: Xenu

Анализ страниц и методов: Приложение 2 (CSV файл списка страниц и методов)

Результаты нагрузки по страницам и методам: Приложение 3 (HTML отчёт работы Locust)

* 1. Карта проекта

Проведем исследовательское тестирование развёрнутого приложения доступного по url https://www.demoblaze.com/

Точкой отчета, главной страницей посчитаем страницу логина.

Выделим функциональные модули проекта:

1. Страница логина
2. Страница выбора товаров
3. Корзина
4. Форма подтверждения покупки
5. Выпадающий список меню
6. Страница контактов
   1. Переходы
7. Страница логина – Страница выбора товаров  
   Ы
8. Страница выбора товаров – Корзина
9. Страница выбора товаров – Выпадающий список меню
10. Страницы выбора товаров – Выпадающий список меню – Страница контактов
11. Корзина – Форма подтверждения заказов
12. Корзина – Выпадающий список меню
13. Корзина – Выпадающий список меню – Страницу выбора товаров
14. Корзина – Выпадающий список меню – Страница контактов
15. Форма подтверждения заказов – Страница выбора товаров
16. Форма подтверждения заказов – Корзина
17. Форма подтверждения заказов – Выпадающий список меню
18. Форма подтверждения заказов – Выпадающий список меню – Страницу выбора товаров
19. Форма подтверждения заказов – Выпадающий список меню – Страница контактов
    1. Эндпоинты
20. Страница логина  
    <https://www.demoblaze.com/index.html>
21. Страница выбора товаров по категориям  
    <https://www.demoblaze.com/index.html>

https://www.demoblaze.com/Samsung1.jpg

https://www.demoblaze.com/nexus1.jpg

https://www.demoblaze.com/iphone1.jpg

1. Корзина

<https://www.demoblaze.com/cart.html>

Подтверждение покупки

<https://www.demoblaze.com/cart.html>

* 1. Описание бизнес процесса

Описание –demoblaze это демонстрационное веб приложение имитирующее магазин по продаже цифровой техники.

Целевая аудитория – Тестировщики ПО, разработчики, студенты и учащиеся, энтузиасты.

Основной бизнес процесс

1. Стратегия тестирования
   1. Техническое описание проекта

Название: Demoblaze

URL: <https://www.demoblaze.com/>

Front: HTML, CSS, JS

Back: (Вероятно отсутствует)

Браузер: Chrome v, Firefox, Safari, Edge, Cпутник, Яндекс.

ОС: Windows 10 22H2

* 1. Функциональное тестирование

Проверка всех функциональных возможностей сайта (см 1.2.)

1. Страница логина
2. Страница выбора товаров
3. Корзина
4. Форма подтверждения покупки
5. Выпадающий список меню
6. Страница контактов

Проведем исследовательское тестирование (см 1.3.)

Тестирование в различных сценариях, составим тестовые случаи и сформируем тестовые наборы и листы

Используем техники тест дизайна

* 1. Тестирование производительности

Проверка доступности эндпоинтов (см 1.4.)

Проверка времени загрузки страниц

Проверка отклика сайта на действия по локаторам (см 3.3)

* 1. Тестирование доступности

Проверка работы ресурса в различных браузерах на различных устройствах, разрешениях и т.д.

Проверка отображения всех элементов на всех страницах

Проверка удобства пользования ресурсом (навигация, доступность, дезигн)

* 1. Тестирование безопасности

Работа приложения под нагрузкой (см 2.3) приложение 3.

Проверка защиты от SQL инъекции

* 1. Стратегия достижения целей и задач (см 1.2)

1. Исследовательское тестирование
2. Подготовка тестовой документации
3. Проведение реального тестирования
4. Разработка и написание скриптов автотестирования
5. Анализ результатов тестрования
6. Тестовые случаи
   1. Список тестовых случаев

Создадим общую базу всех выявленных тестовых случаев  
\*xxxx\* - данные для ввода в соответствующие поля  
“yyyyy” – функциональные элементы страницы

* 1. Чек лист

Чек лист являет собой как чек лист на регресс (включает в себя дымовое) и пригоден к автоматизации

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя | Действия | Примечания |
| Логин | | | |
| 1 | Логин | Страница логина – \*Логин\Пароль\* – “LOGIN” – (Открывается каталог) | Логин: TamaraProcenko Пароль: 1q2w3e4r |
| 2 | Разлогин | Страница логина – \*Логин\Пароль\* – “LOGIN” – (Открывается каталог) – (Открыта любая страница с меню) – “MENU” – “Logout” |  |
| Каталог | | | |
| 3 | Каталог Добавление | Логин - Каталог – 3x “CATEGORIE” | Н-р товар: Apple monitor 24  CATEGORIE-monitors |
| Корзина | | | |
| 4 | Корзина Удаление | Логин - Каталог – “CATEGORIE” – “Корзина” – “DELETE” |  |
| Подтверждение покупки | | | |
| 5 | Покупка Успешная | Логин – Каталог – CATEGORIES ” – “Корзина” – “PLASE ORDER” – \*Name\ COINTRY \City\Credit card\Month\Year\* – “Purchase” – “OK” | Name: ДРУГ ДРУЖИЩЕ  COINTRY:  РОССИЯ City: Новосибирск  Credit card:  123456789876544321 Month: 02  Year:  26 |
| 6 | Покупка Отмена | Логин – Каталог – CATEGORIES ” – “Корзина” – “PLASE ORDER” – \*Name\ COINTRY \City\Credit card\Month\Year\* – “CLOSE” |  |

* 1. Пользовательская история

Пользовательский путь:  
Логин – Каталог – Добавление товара – Корзина – Подтверждение – Покупка – Разлогин

1. Удобно
2. Не удобно

5. Автотестирование

* 1. Описание технологического стека

Стек: Python, Playwright, Locust, Pytest

IDE: PyCharm, Playwright IDE, Playwright driver browser

CI\CD: Отсутствует

Описание: e2e тесты (покрыто регрессионный чеклист) через Playwright и имеется простое нагрузочное с заходами на эндпоинты (1.4) через Locust, запуск осуществляется вручную

* 1. Чек лист автоматизации

Идентичен чек листу на регресс

* 1. Реализация (Скрипты)

Приложение 2

* 1. Описание работы и инструкция по эксплуатации

Как работает, как устроено, как запускать, как анализировать результаты работы

* 1. Тестирование производительности

Сбор эндпоинтов Xenu  
Обработка эндпоинтов (1.2)  
Результаты прогона эндпоинтов (jmetr\locust)

1. ***Итоги***

Проведено ручное тестирование

Проведено автотестирование

Проведено тестирование безопасности

Проведено нагрузочное тестирование

Проведено пользовательское тестирование

***Установлено:***

* Установлено что сайтом удобно пользоваться.
* Установлено при ручном тестировании
* что без авторизации имеется возможность оплаты выбранного продукта
* -не блокируется ввод некорректных данных в ордер для оплаты(возможен неограниченного ввода знаков препинания вместо имени и т.д.), что недопустимо при рабочем сайте
* При незаполнении обязательных строк имени и карты оплаты покупка не подверждается.
* Сайт безопасен, как тестировочный, но актуальные данные в него лучше не вводить, во избежание утечек.
* Выдерживает нагрузку до 1000 пользователей.
* Сайт удобен для проведения учебного тестирования (без реальной покупки и оплаты).

Приложение 1

Config\_dbp.ru

import pytest

from playwright.sync\_api import sync\_playwright

# Список браузеров для тестирования

browsers = ['chromium', 'firefox',  'edge', ]

@pytest.fixture(params=browsers)

def browser(request):

    browser\_name = request.param

    with sync\_playwright() as p:

        if browser\_name == 'chromium':

            browser = p.chromium.launch(headless=False)

        elif browser\_name == 'firefox':

            browser = p.firefox.launch(headless=False)

        elif browser\_name == 'webkit':

            browser = p.webkit.launch(headless=False)

        elif browser\_name == 'edge':

            browser = p.chromium.launch(headless=False, channel="msedge")

        elif browser\_name == 'chrome\_mob':

            browser = p.chromium.launch(headless=False)

            context = browser.new\_context(

                \*\*p.devices['iPhone ']

            )

            page = context.new\_page()

            yield page

            context.close()

            return

        else:

            raise ValueError(f"Unsupported browser: {browser\_name}")

        yield browser

        browser.close()

@pytest.fixture

def page(browser):

    page = browser.new\_page()

    yield page

    page.close()

data\_dbp.ru

# адрес

data\_web\_adress\_demoblaze = "https://www.demoblaze.com/index.html"

# юзеры

data\_username = "TamaraProcenko"

# пароль

data\_password = "1q2w3e4r"

# данные для покупки

data\_order\_modal = "https://www.demoblaze.com/cart.html"

data\_name = "ДРУГ ДРУЖИЩЕ"

data\_Country = "Россия"

data\_City = "новосибирск"

data\_Creditcard = "123456789876544321"

data\_Month = "02"

data\_Year = "26"

# опция выпадающего списка сортировки

data\_cat\_phones = "phones"

data\_cat\_phones\_product\_2= "Nokia lumia 1520"

data\_cat\_laptops = "laptops"

data\_cat\_laptops\_product\_3 = "MacBook air"

data\_cat\_monitors = "monitors"

data\_cat\_monitors\_product\_1 = "Apple monitor 24"

Locators\_dbp.ru

# Локаторы для логина

locator\_button\_login = "a#login2"

locator\_modal\_login = "#logInModal"

locator\_field\_user\_name = "#loginusername"

locator\_field\_user\_pass = "#loginpassword"

locator\_button\_login\_modal = "#logInModal > div > div > div.modal-footer > button.btn.btn-primary"

# Локаторы для выхода

locator\_button\_logout = "#logout2"

# Локаторы для сортировки

locator\_cat = "#cat"

locator\_cat\_phones = "a#itemc[onclick=\"byCat('phone')\"]"

locator\_cat\_laptops = "a#itemc[onclick='byCat(\\'notebook\\')']"

locator\_cat\_monitors = "a#itemc[onclick='byCat(\\'monitor\\')']"

# Локаторы для продуктов

locator\_cat\_phones\_product\_2 = "#tbodyid > div:nth-child(2) > div > div > h4"

locator\_cat\_laptops\_product\_3 = "#tbodyid > div:nth-child(3)"

locator\_cat\_monitors\_product\_1 = "#tbodyid > div:nth-child(1)"

# Локаторы для корзины

locator\_shopping\_cart = "#cartur"

# Локаторы для покупки

locator\_button\_Add\_to\_cart = "a.btn.btn-success.btn-lg"

locator\_button\_plase\_order = "#page-wrapper > div > div.col-lg-1 > button"

locator\_button\_plase\_order\_name = "#name"

locator\_button\_plase\_order\_country = "#country"

locator\_button\_plase\_order\_City = "#city"

locator\_button\_plase\_order\_Creditcard = "#card"

locator\_button\_plase\_order\_Month = "#month"

locator\_button\_plase\_order\_Year = "#year"

locator\_button\_plase\_order\_purchase = "#orderModal > div > div > div.modal-footer > button.btn.btn-primary"

locator\_button\_plase\_order\_ok = "body > div.sweet-alert.showSweetAlert.visible > div.sa-button-container > div > button"

locator\_button\_plase\_order\_close = "#orderModal > div > div > div.modal-footer > button.btn.btn-secondary"

файл test\_dbp\_start.ru

from playwright.sync\_api import sync\_playwright

# Локаторы

locators = {

    "locator\_button\_login": "a#login2",

    "locator\_modal\_login": "#logInModal",

    "locator\_field\_user\_name": "#loginusername",

    "locator\_field\_user\_pass": "#loginpassword",

    "locator\_button\_login\_modal": "#logInModal > div > div > div.modal-footer > button.btn.btn-primary",

    "locator\_button\_logout": "#logout2",

    "locator\_cat": "#cat",

    "locator\_cat\_phones": "a#itemc[onclick=\"byCat('phone')\"]",

    "locator\_cat\_laptops": "a#itemc[onclick='byCat(\\'notebook\\')']",

    "locator\_cat\_monitors": "a#itemc[onclick='byCat(\\'monitor\\')']",

    "locator\_cat\_phones\_product\_2": "#tbodyid > div:nth-child(2) > div > div > h4",

    "locator\_cat\_laptops\_product\_3": "#tbodyid > div:nth-child(3)",

    "locator\_cat\_monitors\_product\_1": "#tbodyid > div:nth-child(1)",

    "locator\_shopping\_cart": "#cartur",

    "locator\_button\_Add\_to\_cart": "a.btn.btn-success.btn-lg",

    "locator\_button\_plase\_order": "#page-wrapper > div > div.col-lg-1 > button",

    "locator\_button\_plase\_order\_name": "#name",

    "locator\_button\_plase\_order\_country": "#country",

    "locator\_button\_plase\_order\_City": "#city",

    "locator\_button\_plase\_order\_Creditcard": "#card",

    "locator\_button\_plase\_order\_Month": "#month",

    "locator\_button\_plase\_order\_Year": "#year",

    "locator\_button\_plase\_order\_purchase": "#orderModal > div > div > div.modal-footer > button.btn.btn-primary",

    "locator\_button\_plase\_order\_ok": "body > div.sweet-alert.showSweetAlert.visible > div.sa-button-container > div > button",

    "locator\_button\_plase\_order\_close": "#orderModal > div > div > div.modal-footer > button.btn.btn-secondary",

    "locator\_modal\_plase\_order": "#orderModal"  # Локатор для модального окна plase\_order

}

def test\_demoblaze\_checking\_locators(): #проверка локаторов на странице входа и выбора товаров

    with sync\_playwright() as p:

        browser = p.chromium.launch(headless=False)

        page = browser.new\_page()

        page.goto("https://www.demoblaze.com/index.html")

        result = page.evaluate("document.querySelector('#login2') !== null")# Проверка локатора для кнопки "Log in"

        print("Кнопка Log in верна")

        assert result, "Кнопка 'Log in' не найдена"

        # Клик по кнопке "Log in" для открытия модального окна

        page.click(locators["locator\_button\_login"])# Клик по кнопке "Log in" для открытия модального окна

        page.wait\_for\_selector(locators["locator\_modal\_login"])# Ожидание открытия модального окна

        modal\_login = page.locator(locators["locator\_modal\_login"])# Проверка, что модальное окно видно

        assert modal\_login.is\_visible(), "Модальное окно логина не видно"

        print("Модальное окно логина видно")

        field\_user\_name = page.locator(locators["locator\_field\_user\_name"])

        assert field\_user\_name.is\_visible(), "Поле ввода имени пользователя не видно"

        print("Поле ввода имени пользователя видно")

        field\_user\_pass = page.locator(locators["locator\_field\_user\_pass"])

        assert field\_user\_pass.is\_visible(), "Поле ввода пароля не видно"

        print("Поле ввода пароля видно")

        button\_login\_modal = page.locator(locators["locator\_button\_login\_modal"])

        assert button\_login\_modal.is\_visible(), "Кнопка 'Log in' в модальном окне не видна"

        print("Кнопка 'Log in' в модальном окне видна")

        result = page.evaluate("document.querySelector('#logout2') !== null")  # Проверка локатора для кнопки "Logout"

        print("Кнопка Logout верна")

        assert result, "Кнопка 'Logout' не найдена"

        # Проверка, что элемент с локатором locator\_cat виден

        cat\_element = page.locator(locators["locator\_cat"])

        assert cat\_element.is\_visible(), "Элемент с локатором 'locator\_cat' не виден"

        print("Элемент с локатором 'locator\_cat' виден")

        cat\_phones\_element = page.locator(locators["locator\_cat\_phones"])

        assert cat\_phones\_element.is\_visible(), "Элемент с локатором 'locator\_cat\_phones' не виден"

        print("Элемент с локатором 'locator\_cat\_phones' виден")

        cat\_monitors\_element = page.locator(locators["locator\_cat\_monitors"])

        assert cat\_monitors\_element.is\_visible(), "Элемент с локатором 'locator\_cat\_monitors' не виден"

        print("Элемент с локатором 'locator\_cat\_monitors' виден")

        cat\_laptops\_element = page.locator(locators["locator\_cat\_laptops"])

        assert cat\_laptops\_element.is\_visible(), "Элемент с локатором 'locator\_cat\_laptops' не виден"

        print("Элемент с локатором 'locator\_cat\_laptops' виден")

        page.wait\_for\_selector(locators["locator\_cat\_phones\_product\_2"])

        # Проверка, что элемент с локатором locator\_cat\_phones\_product\_2 виден

        cat\_phones\_product\_2\_element = page.locator(locators["locator\_cat\_phones\_product\_2"])

        assert cat\_phones\_product\_2\_element.is\_visible(), "Элемент с локатором 'locator\_cat\_phones\_product\_2' не виден"

        print("Элемент с локатором 'locator\_cat\_phones\_product\_2' виден")

        page.wait\_for\_selector(locators["locator\_cat\_laptops\_product\_3"])

        # Проверка, что элемент с локатором locator\_cat\_laptops\_product\_3 виден

        cat\_laptops\_product\_3\_element = page.locator(locators["locator\_cat\_laptops\_product\_3"])

        assert cat\_laptops\_product\_3\_element.is\_visible(), "Элемент с локатором 'locator\_cat\_laptops\_product\_3' не виден"

        print("Элемент с локатором 'locator\_cat\_laptops\_product\_3' виден")

        page.wait\_for\_selector(locators["locator\_cat\_monitors\_product\_1"])

        # Проверка, что элемент с локатором locator\_cat\_monitors\_product\_1 виден

        cat\_monitors\_product\_1\_element = page.locator(locators["locator\_cat\_monitors\_product\_1"])

        assert cat\_monitors\_product\_1\_element.is\_visible(), "Элемент с локатором 'locator\_cat\_monitors\_product\_1' не виден"

        print("Элемент с локатором 'locator\_cat\_monitors\_product\_1' виден")

        # Проверка, что элемент с локатором locator\_shopping\_cart виден

        shopping\_cart\_element = page.locator(locators["locator\_shopping\_cart"])

        assert shopping\_cart\_element.is\_visible(), "Элемент с локатором 'locator\_shopping\_cart' не виден"

        print("Элемент с локатором 'locator\_shopping\_cart' виден")

        browser.close()

def test\_demoblaze\_checkingcart\_locators(): #проверка локаторов на странице корзины

    with sync\_playwright() as p:

        browser = p.chromium.launch(headless=False)

        page = browser.new\_page()

        page.goto("https://www.demoblaze.com/cart.html")

        # Проверка, что элемент с локатором locator\_button\_plase\_order виден

        plase\_order\_button\_element = page.locator(locators["locator\_button\_plase\_order"])

        assert plase\_order\_button\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order' не виден"

        print("Элемент с локатором 'locator\_button\_plase\_order' виден")

        # Клик по кнопке "Place Order" для открытия модального окна оформления заказа

        page.click(locators["locator\_button\_plase\_order"])

        page.wait\_for\_selector(locators["locator\_modal\_plase\_order"])  # Ожидание открытия модального окна

        # Проверка, что модальное окно plase\_order видно

        plase\_order\_modal\_element = page.locator(locators["locator\_modal\_plase\_order"])

        assert plase\_order\_modal\_element.is\_visible(), "Модальное окно 'plase\_order' не видно"

        print("Модальное окно 'plase\_order' видно")

        # Проверка, что элемент с локатором locator\_button\_plase\_order\_name виден

        plase\_order\_name\_element = page.locator(locators["locator\_button\_plase\_order\_name"])

        assert plase\_order\_name\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_name' не виден"

        print("Элемент с локатором 'locator\_button\_plase\_order\_name' виден")

        # Проверка, что элемент с локатором locator\_button\_plase\_order\_country виден

        plase\_order\_country\_element = page.locator(locators["locator\_button\_plase\_order\_country"])

        assert plase\_order\_country\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_country' не виден"

        print("Элемент с локатором 'locator\_button\_plase\_order\_country' виден")

        # Проверка, что элемент с локатором locator\_button\_plase\_order\_City виден

        plase\_order\_City\_element = page.locator(locators["locator\_button\_plase\_order\_City"])

        assert plase\_order\_City\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_City' не виден"

        print("Элемент с локатором 'locator\_button\_plase\_order\_City' виден")

        # Проверка, что элемент с локатором locator\_button\_plase\_order\_Creditcard виден

        plase\_order\_Creditcard\_element = page.locator(locators["locator\_button\_plase\_order\_Creditcard"])

        assert plase\_order\_Creditcard\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_Creditcard' не виден"

        print("Элемент с локатором 'locator\_button\_plase\_order\_Creditcard' виден")

        # Проверка, что элемент с локатором locator\_button\_plase\_order\_Month виден

        plase\_order\_Month\_element = page.locator(locators["locator\_button\_plase\_order\_Month"])

        assert plase\_order\_Month\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_Month' не виден"

        print("Элемент с локатором 'locator\_button\_plase\_order\_Month' виден")

        # Проверка, что элемент с локатором locator\_button\_plase\_order\_Year виден

        plase\_order\_Year\_element = page.locator(locators["locator\_button\_plase\_order\_Year"])

        assert plase\_order\_Year\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_Year' не виден"

        print("Элемент с локатором 'locator\_button\_plase\_order\_Year' виден")

        plase\_order\_purchase\_button\_element = page.locator(locators["locator\_button\_plase\_order\_purchase"])

        assert plase\_order\_purchase\_button\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_purchase' не виден"

        print("Элемент с локатором 'locator\_button\_plase\_order\_purchase' виден")

        # Проверка, что элемент с локатором locator\_button\_plase\_order\_close виден

        plase\_order\_close\_button\_element = page.locator(locators["locator\_button\_plase\_order\_close"])

        assert plase\_order\_close\_button\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_close' не виден"

        print("Элемент с локатором 'locator\_button\_plase\_order\_close' виден")

        browser.close()

def test\_demoblaze\_brausers(page): #запуск теста на различных браузерах с неверными данными для ордера

    page.goto("https://www.demoblaze.com/cart.html")

    # Проверка, что элемент с локатором locator\_button\_plase\_order виден

    plase\_order\_button\_element = page.locator(locators["locator\_button\_plase\_order"])

    assert plase\_order\_button\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order' не виден"

    print("Элемент с локатором 'locator\_button\_plase\_order' виден")

    # Клик по кнопке "Place Order" для открытия модального окна оформления заказа

    page.click(locators["locator\_button\_plase\_order"])

    page.wait\_for\_selector(locators["locator\_modal\_plase\_order"])  # Ожидание открытия модального окна

    # Проверка, что модальное окно plase\_order видно

    plase\_order\_modal\_element = page.locator(locators["locator\_modal\_plase\_order"])

    assert plase\_order\_modal\_element.is\_visible(), "Модальное окно 'plase\_order' не видно"

    print("Модальное окно 'plase\_order' видно")

    # Проверка, что элемент с локатором locator\_button\_plase\_order\_purchase виден

    plase\_order\_purchase\_button\_element = page.locator(locators["locator\_button\_plase\_order\_purchase"])

    assert plase\_order\_purchase\_button\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_purchase' не виден"

    print("Элемент с локатором 'locator\_button\_plase\_order\_purchase' виден")

    # Заполнение формы оформления заказа на неверных данных

    page.fill(locators["locator\_button\_plase\_order\_name"], "3;№")

    page.fill(locators["locator\_button\_plase\_order\_country"], "?Г\*Ш(Щ")

    page.fill(locators["locator\_button\_plase\_order\_City"], "ё4=90")

    page.fill(locators["locator\_button\_plase\_order\_Creditcard"], "Хх,45ву@#@#,,,.")

    page.fill(locators["locator\_button\_plase\_order\_Month"], "111111111111111111111111111111111111111111111")

    page.fill(locators["locator\_button\_plase\_order\_Year"], "класс")

    # Клик по кнопке "Purchase" для подтверждения заказа

    page.click(locators["locator\_button\_plase\_order\_purchase"])

    page.wait\_for\_selector(locators["locator\_button\_plase\_order\_ok"])  # Ожидание открытия модального окна подтверждения

    # Проверка, что элемент с локатором locator\_button\_plase\_order\_ok виден

    plase\_order\_ok\_button\_element = page.locator(locators["locator\_button\_plase\_order\_ok"])

    assert plase\_order\_ok\_button\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_ok' не виден"

    print("Элемент с локатором 'locator\_button\_plase\_order\_ok' виден")

    # Проверка, что элемент с локатором locator\_button\_plase\_order\_close виден

    plase\_order\_close\_button\_element = page.locator(locators["locator\_button\_plase\_order\_close"])

    assert plase\_order\_close\_button\_element.is\_visible(), "Элемент с локатором 'locator\_button\_plase\_order\_close' не виден"

    print("Элемент с локатором 'locator\_button\_plase\_order\_close' виден")

файл test\_dbp\_steps.ru

from playwright.sync\_api import sync\_playwright, Page, BrowserContext

import pytest

from locators\_db import \*

from test\_data\_dbp import \*

@pytest.fixture

def get\_playwright():

    with sync\_playwright() as playwright:

        yield playwright

@pytest.fixture

def browser(get\_playwright):

    # browser = get\_playwright.chromium.launch(headless=False)

    browser = get\_playwright.chromium.launch()

    yield browser

    browser.close()

@pytest.fixture

def browser\_context(browser):

    context = browser.new\_context()

    yield context

    context.close()

@pytest.fixture

def page(browser\_context: BrowserContext):

    page = browser\_context.new\_page()

    yield page

    page.close()

@pytest.fixture

def login(page: Page):

    def login\_function():

        page.goto(data\_web\_adress\_demoblaze)

        page.fill(locator\_field\_user\_name, data\_username)

        page.fill(locator\_field\_user\_pass, data\_password)

        page.click(locator\_button\_login)

        page.screenshot(path=f"screenshot/login.png")

    return login\_function

Файл test\_dbp\_check

from playwright.sync\_api import Page

from test\_dbp\_steps import \*

def test\_demoblaze\_login():

    with sync\_playwright() as p:

        browser = p.chromium.launch()

        page = browser.new\_page()

        page.goto("https://www.demoblaze.com/index.html")

        page.click(locator\_button\_login)# Клик по кнопке "Log in"

        page.wait\_for\_selector(locator\_modal\_login)# Ожидание открытия модального окна

        assert page.query\_selector(locator\_modal\_login) is not None # Проверка, что модальное окно открыто

        page.screenshot(path='screenshots/login.jpeg')# Скриншот модального окна

        page.fill(locator\_field\_user\_name, data\_username)# Ввод логина и пароля

        page.fill(locator\_field\_user\_pass, data\_password)

        page.click(locator\_field\_user\_name)# Клик по кнопке "Log in" внутри модального окна

        page.wait\_for\_timeout(3000)# Ожидание закрытия модального окна

        page.screenshot(path='screenshots/login1.jpeg')# Скриншот после логина

        browser.close()

def test\_demoblaze\_logout():

    with sync\_playwright() as p:

        browser = p.chromium.launch()

        page = browser.new\_page()

        page.goto("https://www.demoblaze.com/index.html")

        page.click(locator\_button\_login)# Клик по кнопке "Log in"

        page.wait\_for\_selector(locator\_modal\_login)# Ожидание открытия модального окна

        assert page.query\_selector(locator\_modal\_login) is not None, "Модальное окно не открыто" # Проверка, что модальное окно открыто

        page.fill(locator\_field\_user\_name, data\_username)# Ввод логина и пароля

        page.fill(locator\_field\_user\_pass, data\_password)

        page.click("button[onclick='logIn()']")# Клик по кнопке "Log in" внутри модального окна

        page.wait\_for\_selector("#logout2")# Ожидание закрытия модального окна и успешного входа

        assert page.query\_selector("#logout2") is not None, "Пользователь не вошел в систему"# Проверка, что пользователь вошел в систему

        page.screenshot(path='screenshots/logged\_in.jpeg')# Скриншот после успешного входа

        page.click("a#logout2")# Клик по кнопке "Log out"

        page.wait\_for\_selector("#login2")# Ожидание закрытия сессии и появления кнопки "Log in"

        assert page.query\_selector("#login2") is not None, "Пользователь не вышел из системы"# Проверка, что пользователь вышел из системы

        page.screenshot(path='screenshots/logout.jpeg')# Скриншот после выхода

        browser.close()

def test\_demoblaze\_cat\_product():

    with sync\_playwright() as p:

        browser = p.chromium.launch()

        page = browser.new\_page()

        page.goto("https://www.demoblaze.com/index.html")

        page.click("a#itemc[onclick='byCat(\\'phone\\')']")# Клик по категории "Phones"

        page.wait\_for\_selector(".card-title")# Ожидание загрузки продуктов в категории "Phones"

        assert page.query\_selector\_all(".card-title") is not None, "Продукты в категории 'Phones' не загружены"# Проверка, что продукты в категории "Phones" загружены

        page.screenshot(path='screenshots/phones\_category.jpeg')# Скриншот страницы с продуктами в категории "Phones"

        page.click("a#itemc[onclick='byCat(\\'notebook\\')']")# Клик по категории "Laptops"

        page.wait\_for\_selector(".card-title")# Ожидание загрузки продуктов в категории "Laptops"

        assert page.query\_selector\_all(".card-title") is not None, "Продукты в категории 'Laptops' не загружены"# Проверка, что продукты в категории "Laptops" загружены

        page.screenshot(path='screenshots/laptops\_category.jpeg')# Скриншот страницы с продуктами в категории "Laptops"

        page.click("a#itemc[onclick='byCat(\\'monitor\\')']")# Клик по категории "Monitors"

        page.wait\_for\_selector(".card-title")# Ожидание загрузки продуктов в категории "Monitors"

        assert page.query\_selector\_all(".card-title") is not None, "Продукты в категории 'Monitors' не загружены"# Проверка, что продукты в категории "Monitors" загружены

        page.screenshot(path='screenshots/monitors\_category.jpeg')# Скриншот страницы с продуктами в категории "Monitors"

        browser.close()

def test\_demoblaze\_get\_product():

    with sync\_playwright() as p:

        browser = p.chromium.launch()

        page = browser.new\_page()

        page.goto("https://www.demoblaze.com/index.html")

        page.click("a#itemc[onclick='byCat(\\'phone\\')']")# Клик по категории "Phones"

        page.wait\_for\_selector(".card-title")# Ожидание загрузки продуктов в категории "Phones"

        assert page.query\_selector\_all(".card-title") is not None, "Продукты в категории 'Phones' не загружены"# Проверка, что продукты в категории "Phones" загружены

        product\_link = page.query\_selector("div.card-block a")# Выбор первого продукта в категории "Phones"

        product\_name = product\_link.text\_content()

        product\_link.click()

        page.wait\_for\_selector("#tbodyid h2")# Ожидание загрузки страницы продукта

        assert page.query\_selector("#tbodyid h2").text\_content() == product\_name, "Неверный продукт открыт"# Проверка, что открыта страница выбранного продукта

        page.screenshot(path='screenshots/product\_page.jpeg')# Скриншот страницы выбранного продукта

        browser.close()

def test\_demoblaze\_purchase():

    with sync\_playwright() as p:

        browser = p.chromium.launch()

        page = browser.new\_page()

        page.goto("https://www.demoblaze.com/index.html")

        page.click("a#itemc[onclick='byCat(\\'phone\\')']")# Клик по категории "Phones"

        page.wait\_for\_selector(".card-title")# Ожидание загрузки продуктов в категории "Phones"

        assert page.query\_selector\_all(".card-title") is not None, "Продукты в категории 'Phones' не загружены"# Проверка, что продукты в категории "Phones" загружены

        product\_link = page.query\_selector("div.card-block a")# Выбор первого продукта в категории "Phones"

        product\_name = product\_link.text\_content()

        product\_link.click()

        page.wait\_for\_selector("#tbodyid h2")# Ожидание загрузки страницы продукта

        assert page.query\_selector("#tbodyid h2").text\_content() == product\_name, "Неверный продукт открыт"# Проверка, что открыта страница выбранного продукта

        page.screenshot(path='screenshots/product\_page.jpeg')# Скриншот страницы выбранного продукта

        page.click("a#cartur")# Добавление продукта в корзину

        page.wait\_for\_selector("#page-wrapper")# Ожидание загрузки страницы корзины

        assert page.query\_selector("#page-wrapper") is not None, "Продукт не добавлен в корзину"# Проверка, что продукт добавлен в корзину

        page.screenshot(path='screenshots/cart\_page.jpeg')# Скриншот страницы корзины

        page.click(locator\_button\_plase\_order)# Клик по кнопке "Place Order"

        page.wait\_for\_selector("#orderModal")# Ожидание открытия модального окна оформления заказа

        assert page.query\_selector("#orderModal") is not None, "Модальное окно оформления заказа не открыто"# Проверка, что модальное окно оформления заказа открыто

        page.fill("#name", "ДРУГ ДРУЖИЩЕ")

        page.fill("#country", "Росси")

        page.fill("#city", "Новосибирск")

        page.fill("#card", "12345678987654432")

        page.fill("#month", "02")

        page.fill("#year", "26")

        page.screenshot(path='screenshots/order\_modal.jpeg')# Скриншот модального окна оформления заказа

        page.click(locator\_button\_plase\_order\_purchase)# Клик по кнопке "Purchase"

        page.wait\_for\_selector("#orderModal")# Ожидание загрузки страницы подтверждения заказа

        assert page.query\_selector("#orderModal") is not None, "Заказ не оформлен"# Проверка, что заказ успешно оформлен

        page.screenshot(path='screenshots/order\_confirmation.jpeg')# Скриншот страницы подтверждения заказа

        browser.close()

def test\_demoblaze\_login\_logout():   # ДЕМОНСТРАЦИОННЫЙ

    with sync\_playwright() as p:

        browser = p.chromium.launch(headless=False, slow\_mo=500)  # Открываем браузер с интерфейсом и замедляем выполнение

        page = browser.new\_page()

        page.goto("https://www.demoblaze.com/index.html")

        # Авторизация

        page.click(locator\_button\_login)  # Клик по кнопке "Log in"

        page.wait\_for\_selector(locator\_button\_login\_modal)  # Ожидание открытия модального окна

        assert page.query\_selector(locator\_button\_login\_modal) is not None  # Проверка, что модальное окно открыто

        page.screenshot(path='screenshots/login.jpeg')  # Скриншот модального окна

        page.fill(locator\_field\_user\_name, data\_username)  # Ввод логина и пароля

        page.fill(locator\_field\_user\_pass, data\_password)

        page.click(locator\_button\_login\_modal)  # Клик по кнопке "Log in" внутри модального окна

        page.wait\_for\_timeout(3000)  # Ожидание закрытия модального окна

        page.screenshot(path='screenshots/login1.jpeg')  # Скриншот после логина

        # Выбор товара

        page.click(locator\_cat\_phones)  # Выбор категории "Phones"

        page.click(locator\_cat\_phones\_product\_2)  # Выбор второго товара в списке

        page.click(locator\_button\_Add\_to\_cart)  # Клик по кнопке "Add to cart"

        page.wait\_for\_timeout(2000)  # Ожидание добавления товара в корзину

        page.screenshot(path='screenshots/product\_page.jpeg')  # Скриншот страницы выбранного продукта

        # Переход в корзину

        page.click(locator\_shopping\_cart)  # Переход в корзину

        page.wait\_for\_timeout(2000)  # Ожидание загрузки корзины

        # Оформление заказа

        page.click(locator\_button\_plase\_order)  # Клик по кнопке "Place Order"

        page.wait\_for\_selector(locator\_button\_plase\_order\_name)  # Ожидание открытия модального окна оформления заказа

        page.fill(locator\_button\_plase\_order\_name, data\_name)  # Заполнение данных для оплаты

        page.fill(locator\_button\_plase\_order\_country, data\_Country)

        page.fill(locator\_button\_plase\_order\_City, data\_City)

        page.fill(locator\_button\_plase\_order\_Creditcard, data\_Creditcard)

        page.fill(locator\_button\_plase\_order\_Month, data\_Month)

        page.fill(locator\_button\_plase\_order\_Year, data\_Year)

        page.click(locator\_button\_plase\_order\_purchase)  # Клик по кнопке "Purchase"

        page.wait\_for\_selector(locator\_button\_plase\_order\_ok)  # Ожидание модального окна подтверждения

        page.click(locator\_button\_plase\_order\_ok)  # Клик по кнопке "OK"

        page.wait\_for\_timeout(2000)  # Ожидание закрытия модального окна

        # Выход из аккаунта

        page.click(locator\_button\_logout)  # Клик по кнопке "Log out"

        page.wait\_for\_timeout(2000)  # Ожидание выхода из аккаунта

        browser.close()

def test\_demoblaze\_login():

    with sync\_playwright() as p:

        browser = p.chromium.launch(headless=False)

        page = browser.new\_page()

        page.goto("https://www.demoblaze.com/index.html")

        # Проверка локатора для кнопки "Log in"

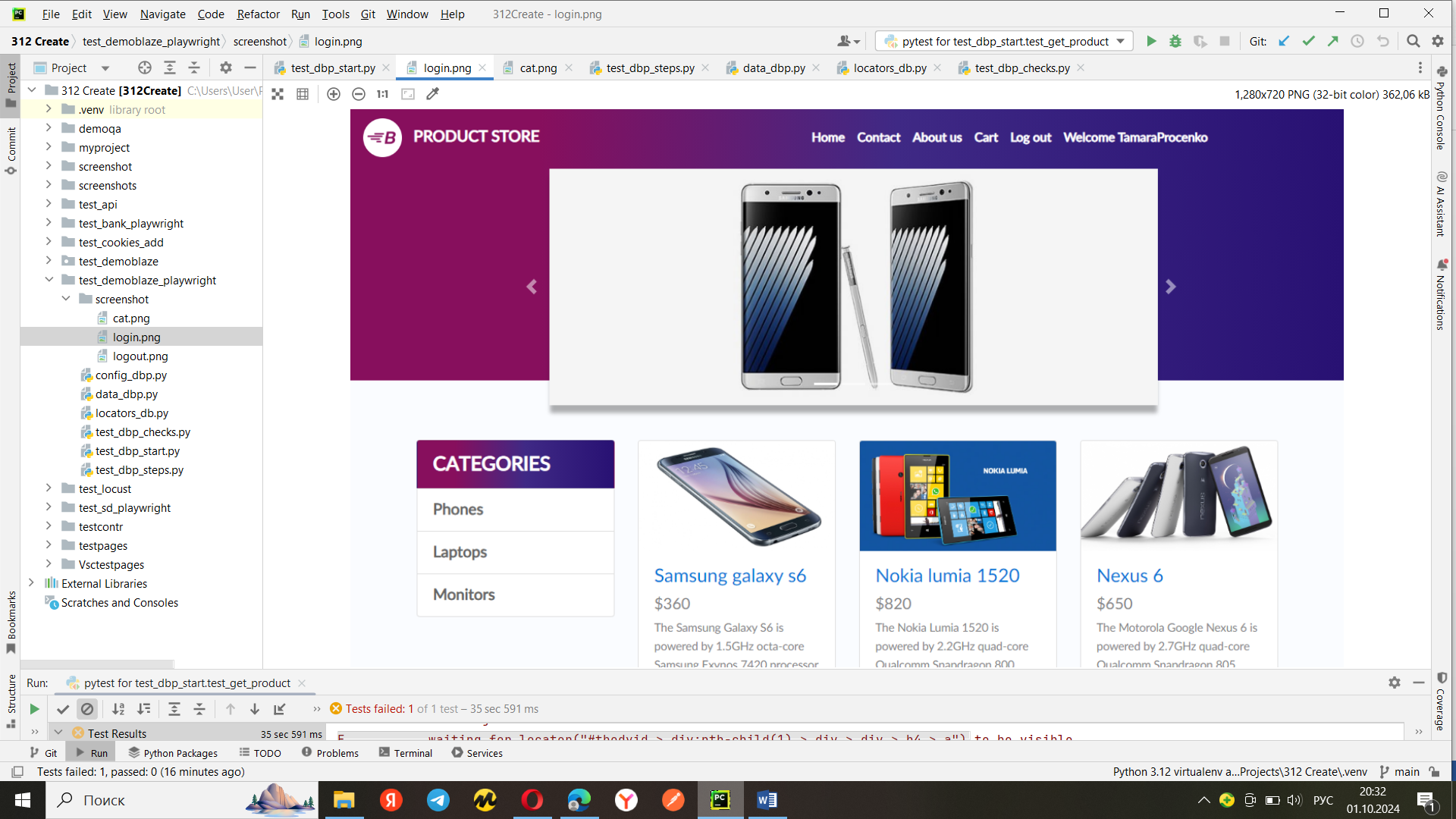
        result = page.evaluate("document.querySelector('#login2') !== null")

        assert result, "Кнопка 'Log in' не найдена"

        browser.close()

Приложение 2

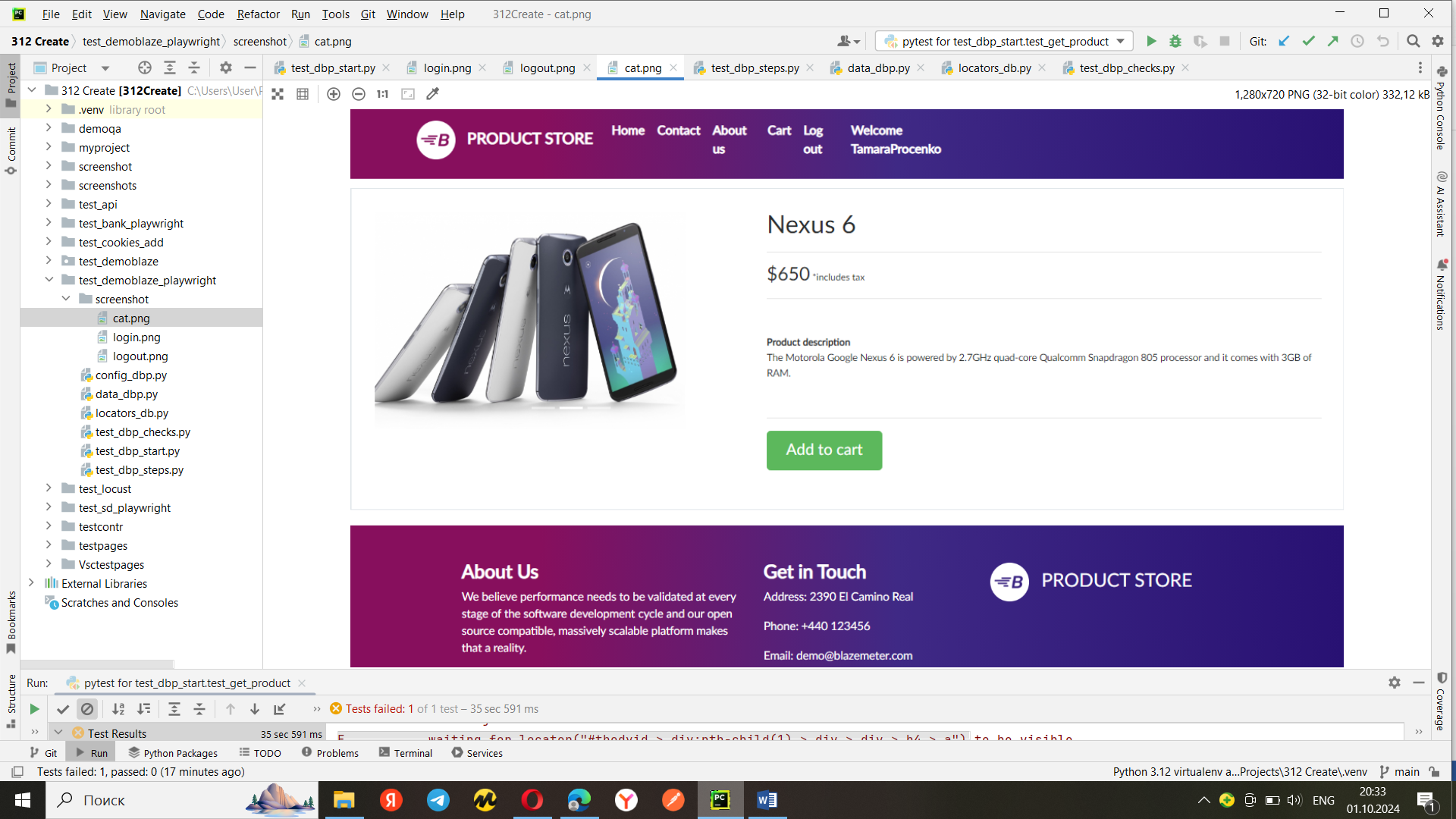
***при запуске тестов выгружаются скриншоты***

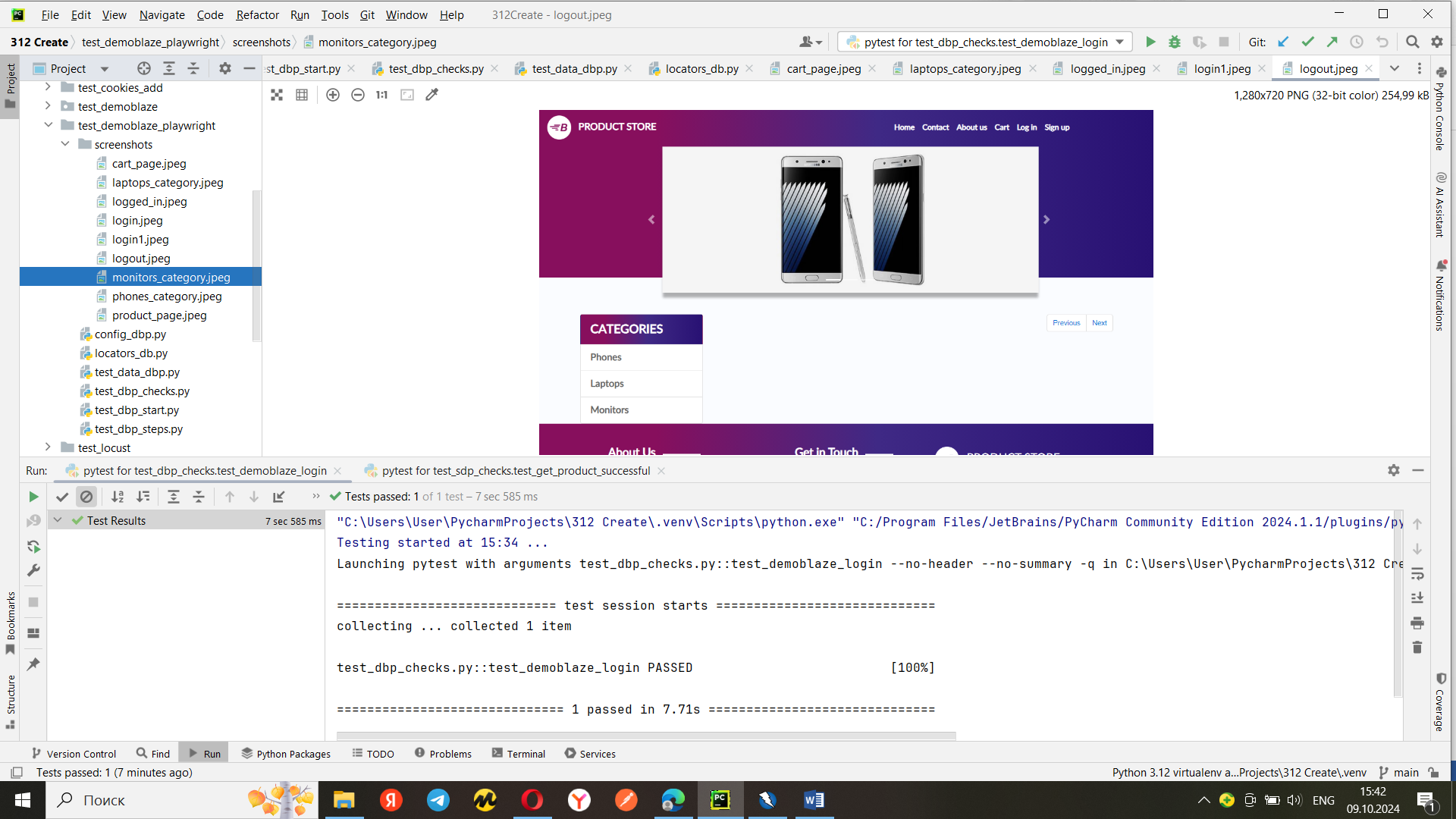
login

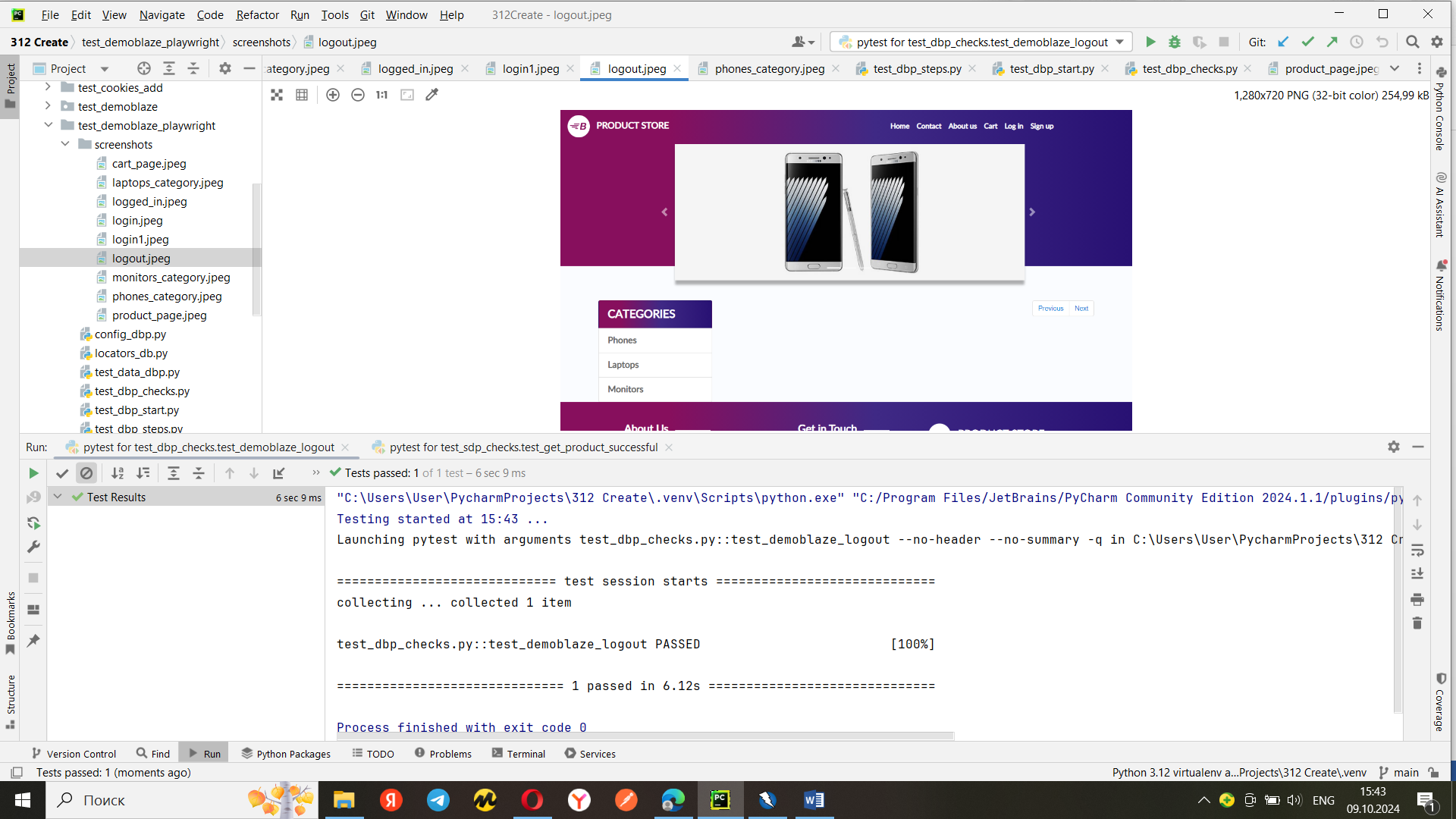
logout

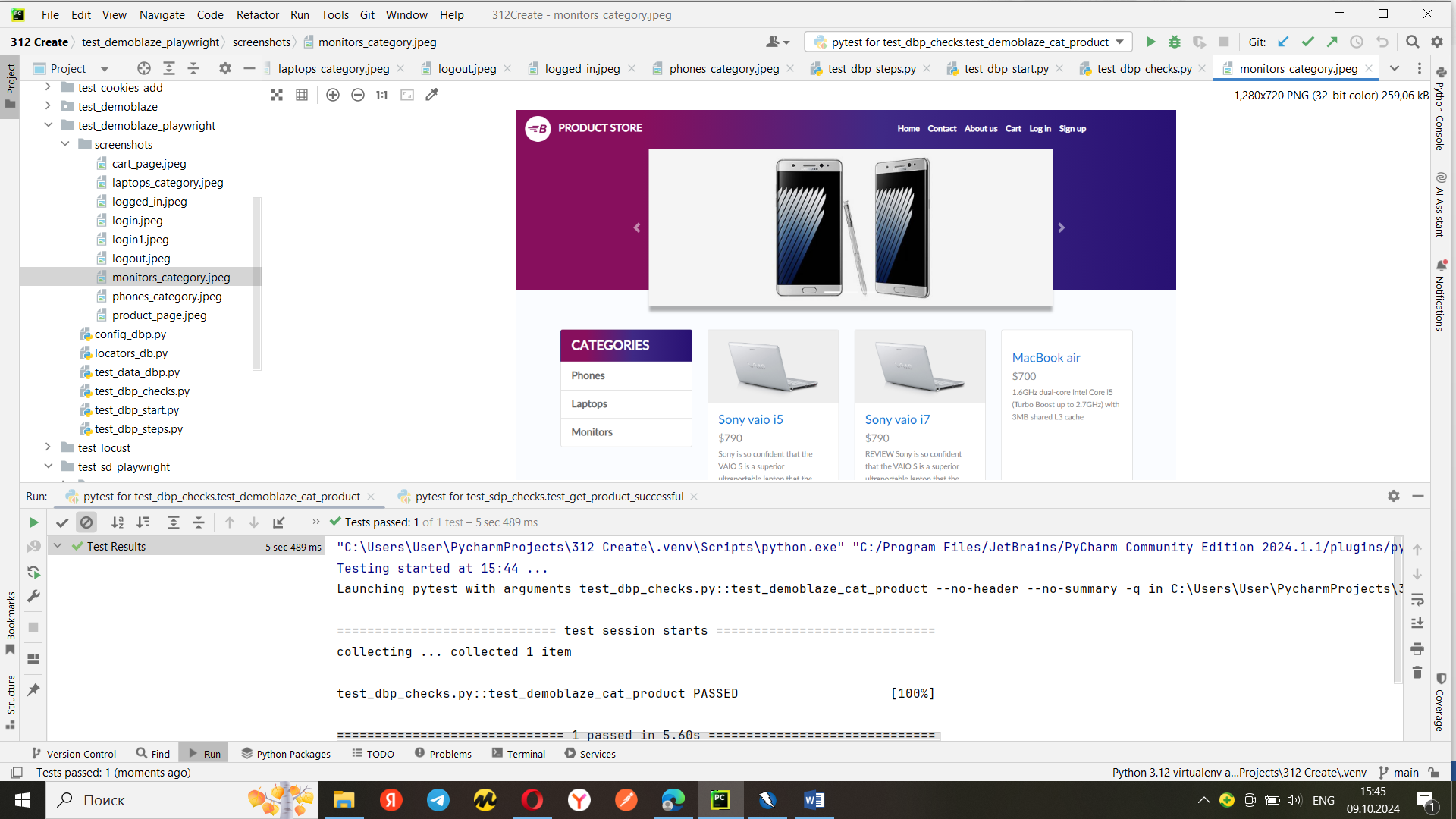


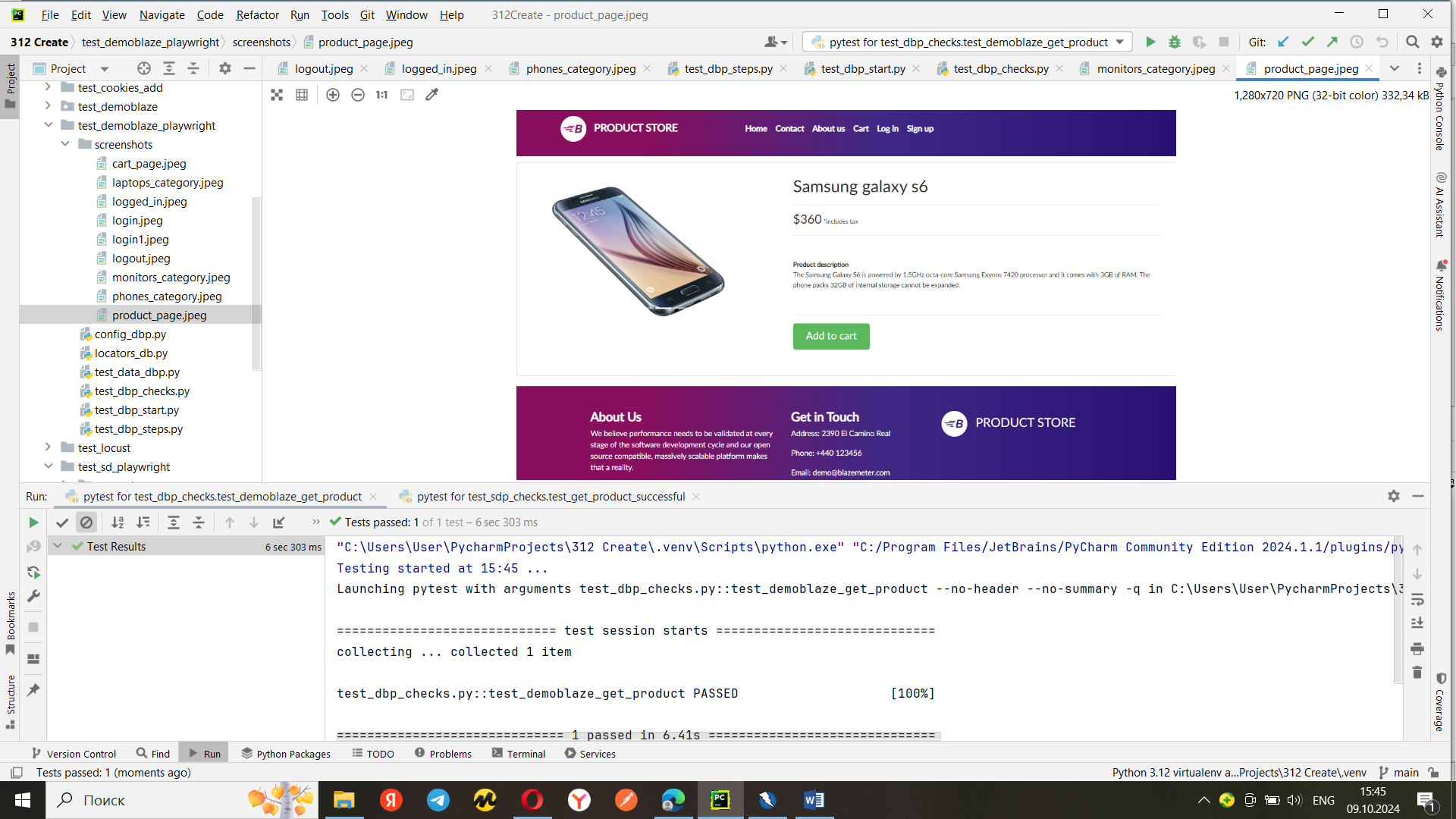
Cat



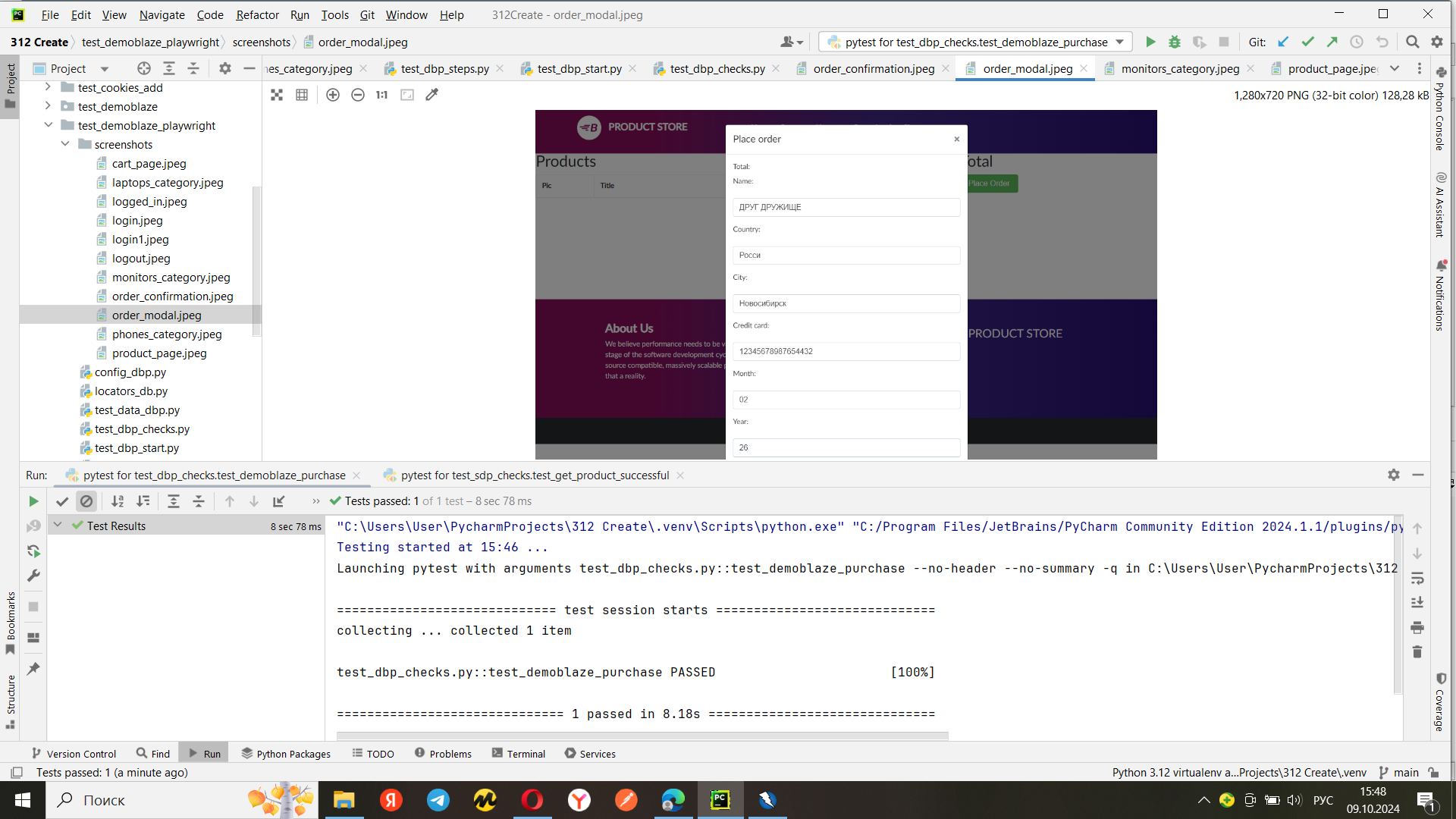




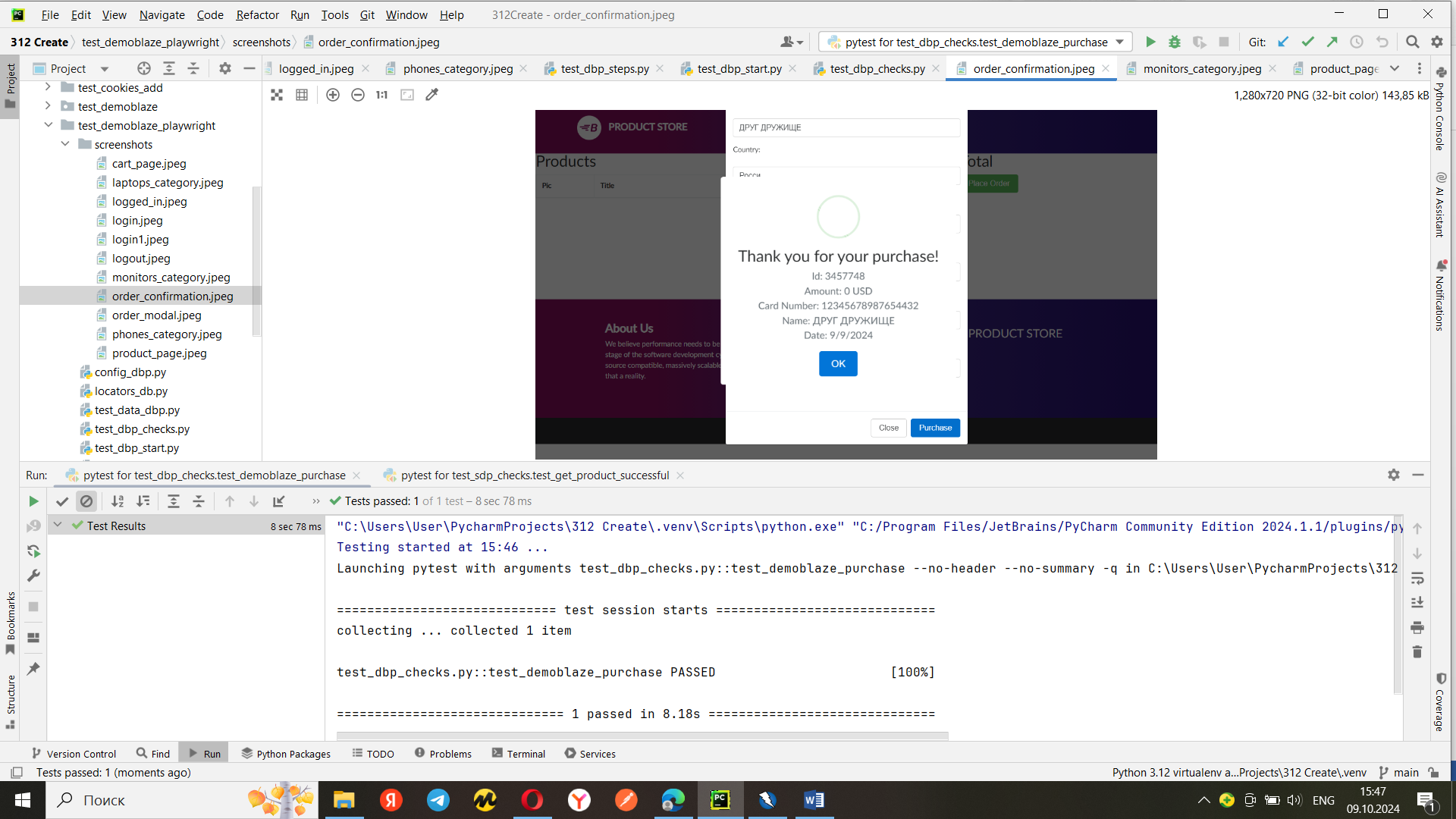




Ввод данных для оплаты

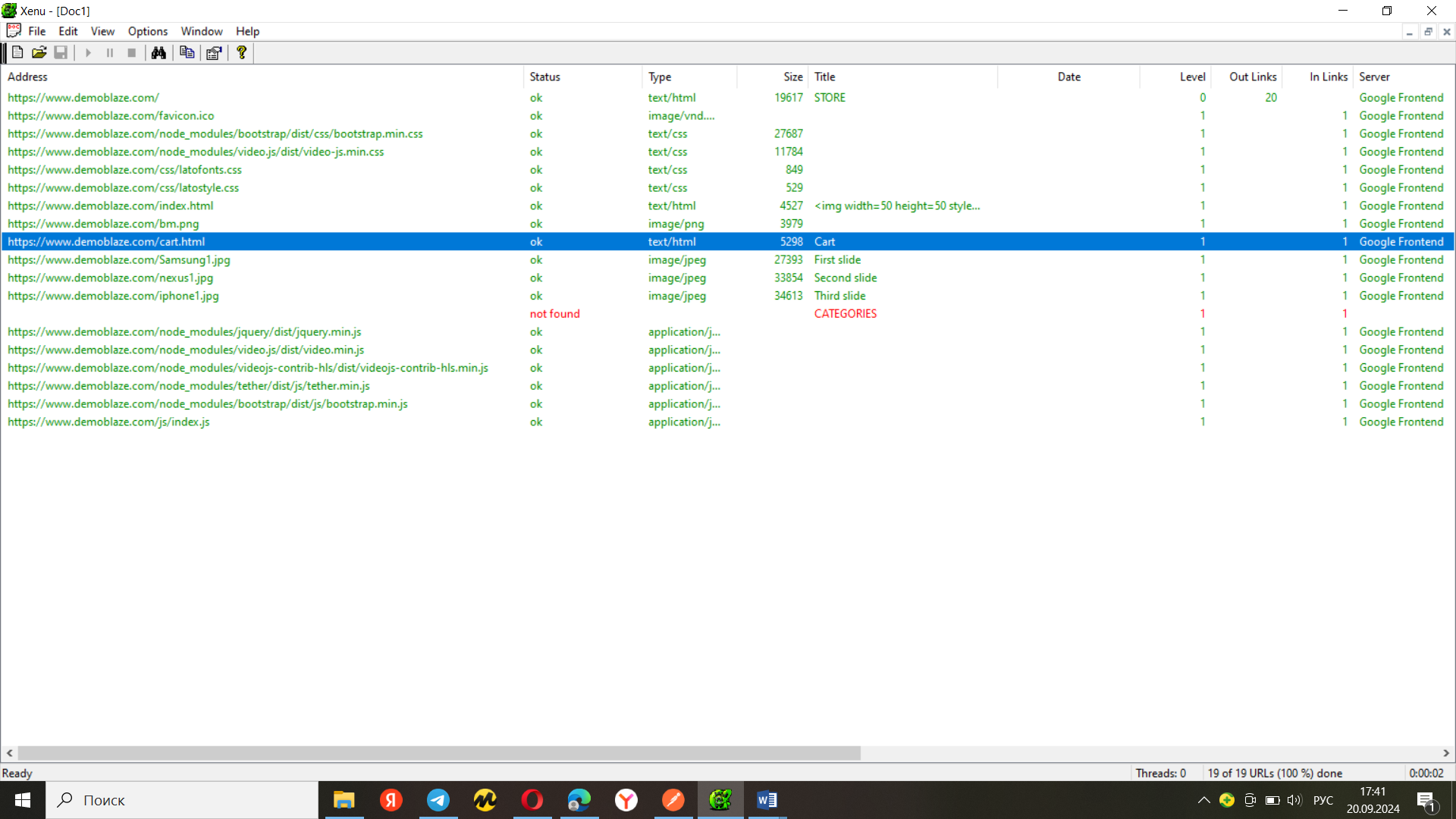


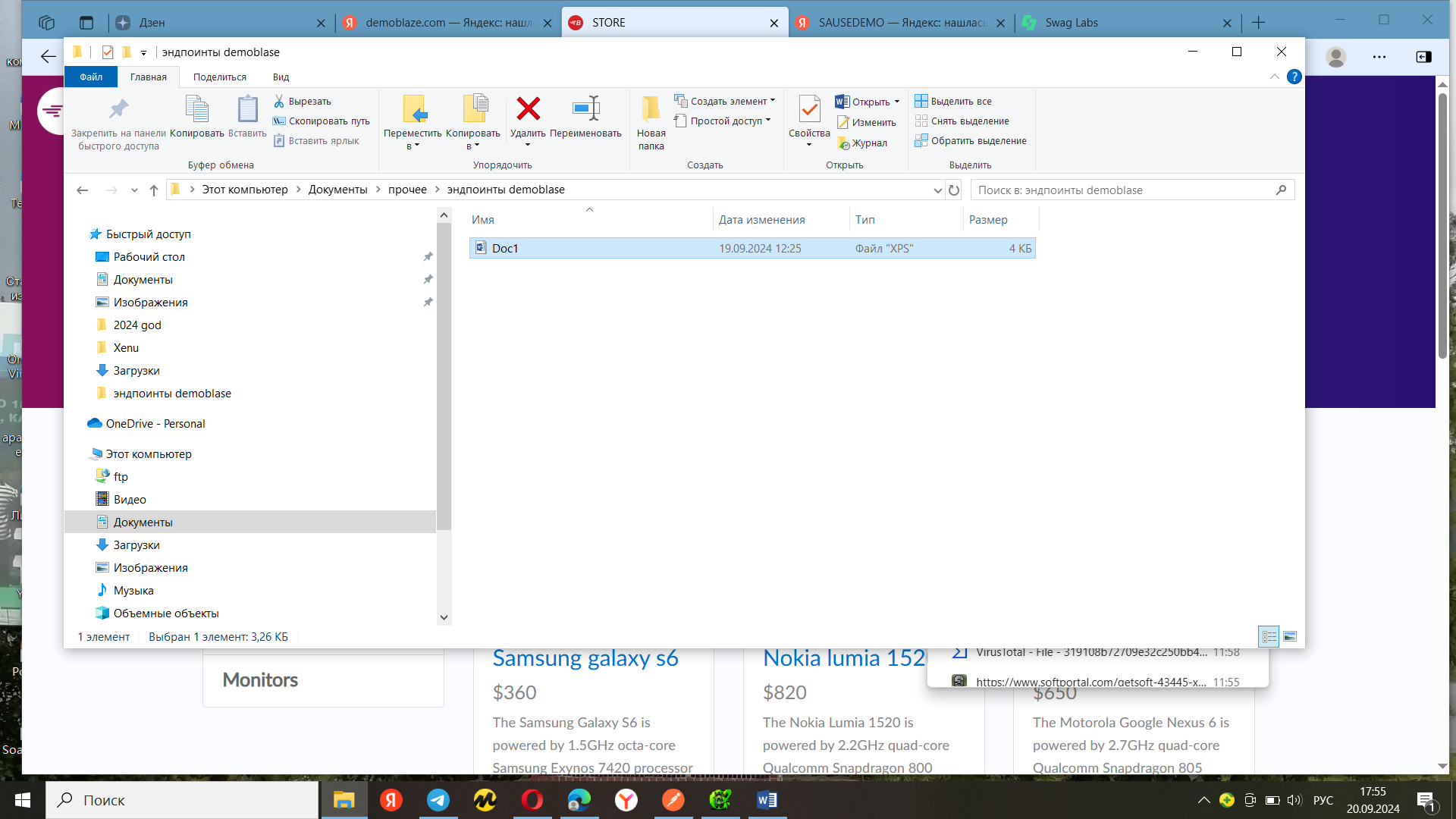
Оплата продукта в корзине

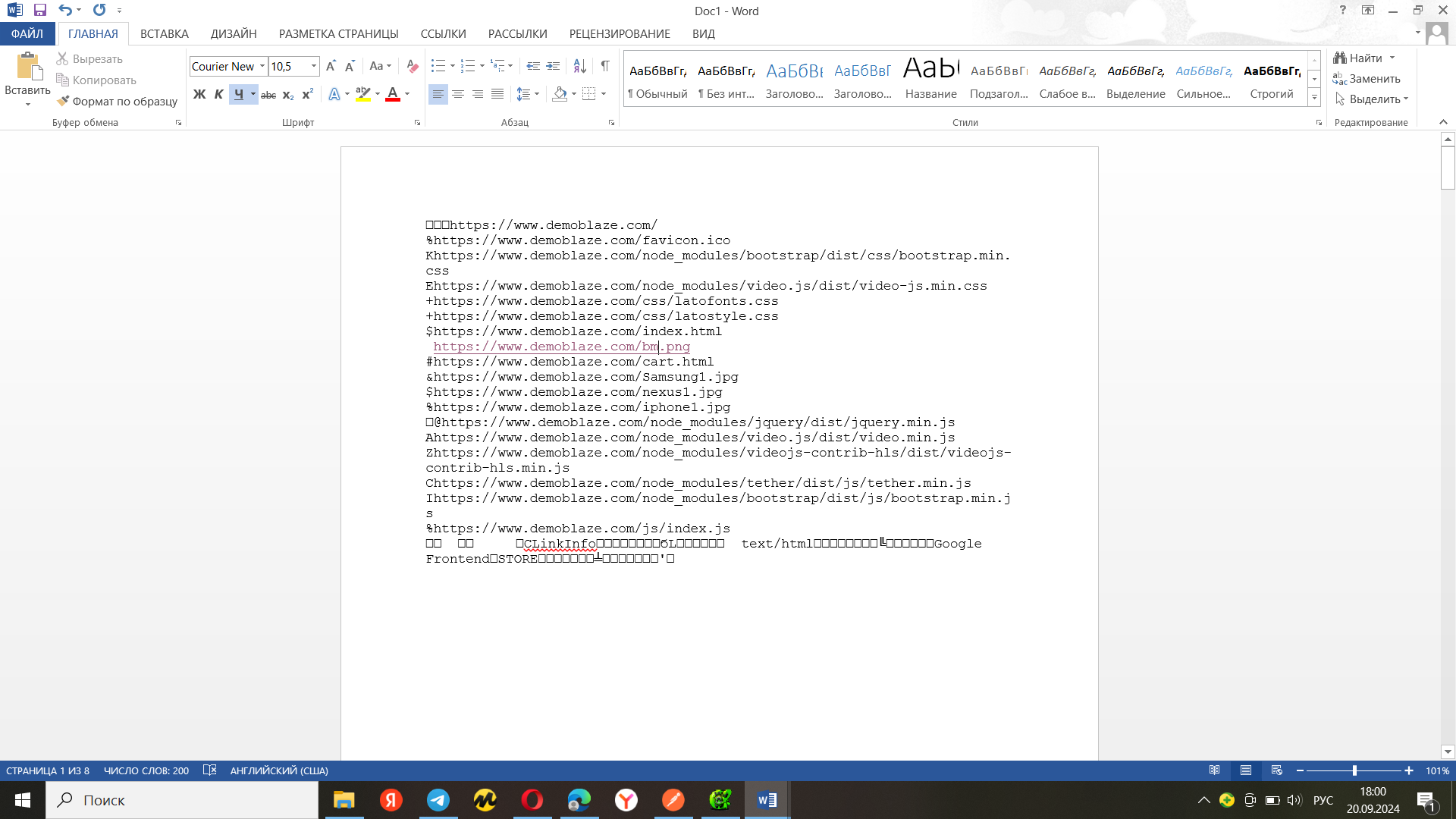


Приложение 3

***ЭНДПОИНТЫ из Xeny***







1. Запускаем Locust:

locust -f locustfile\_db.py --host=http://demoblaze.com

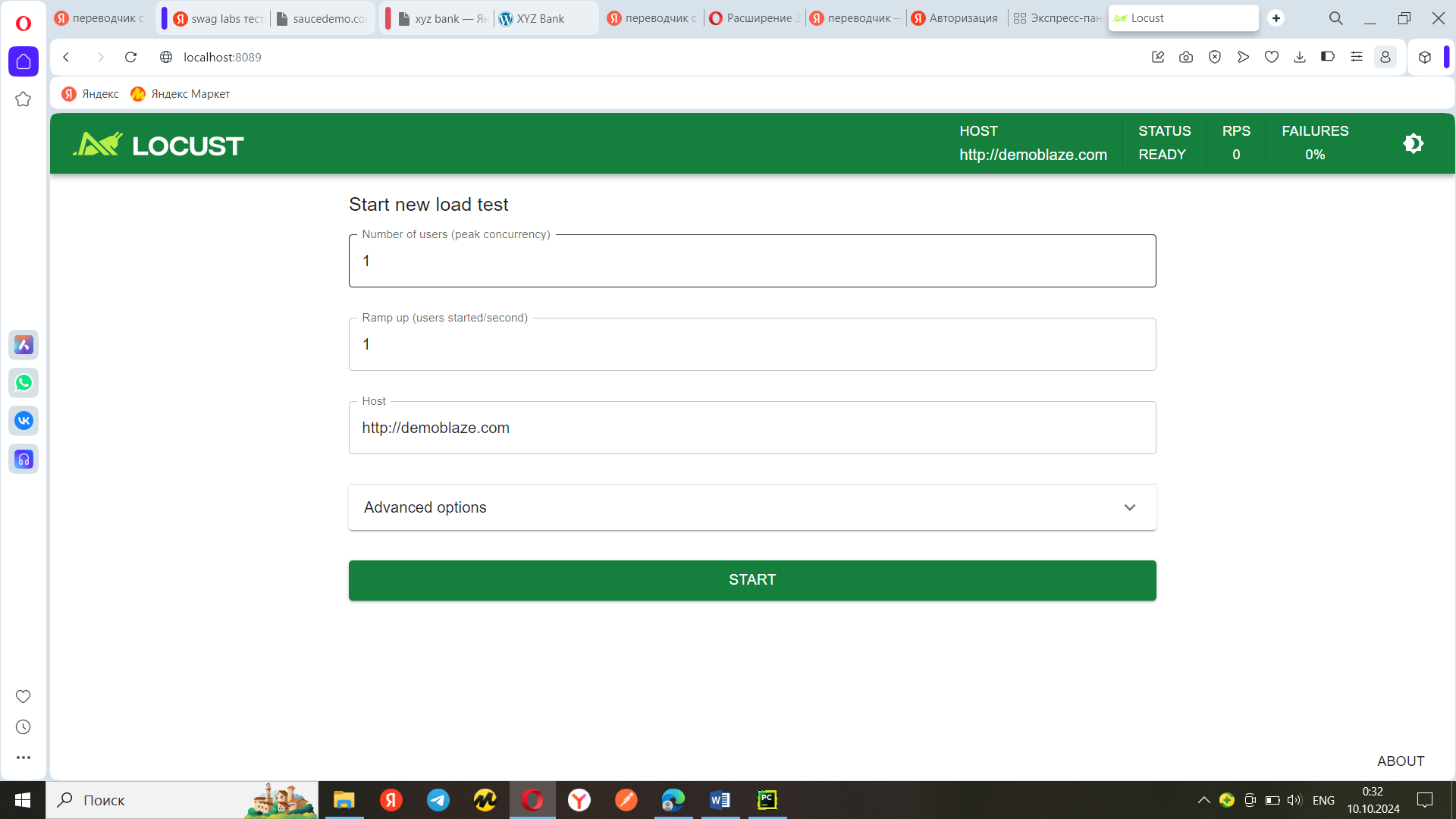
Здесь --host указывает на URL сайта, который вы хотите тестировать.

1. Доступ к веб-интерфейсу Locust

После запуска Locust откройте веб-браузер и перейдите по адресу:

http://localhost:8089

В веб-интерфейсе Locust вы можете указать количество пользователей, которые будут имитироваться, и скорость их набора (ramp-up).



Увеличение до 1000 пользователей со скоростью 1,00 в секунду

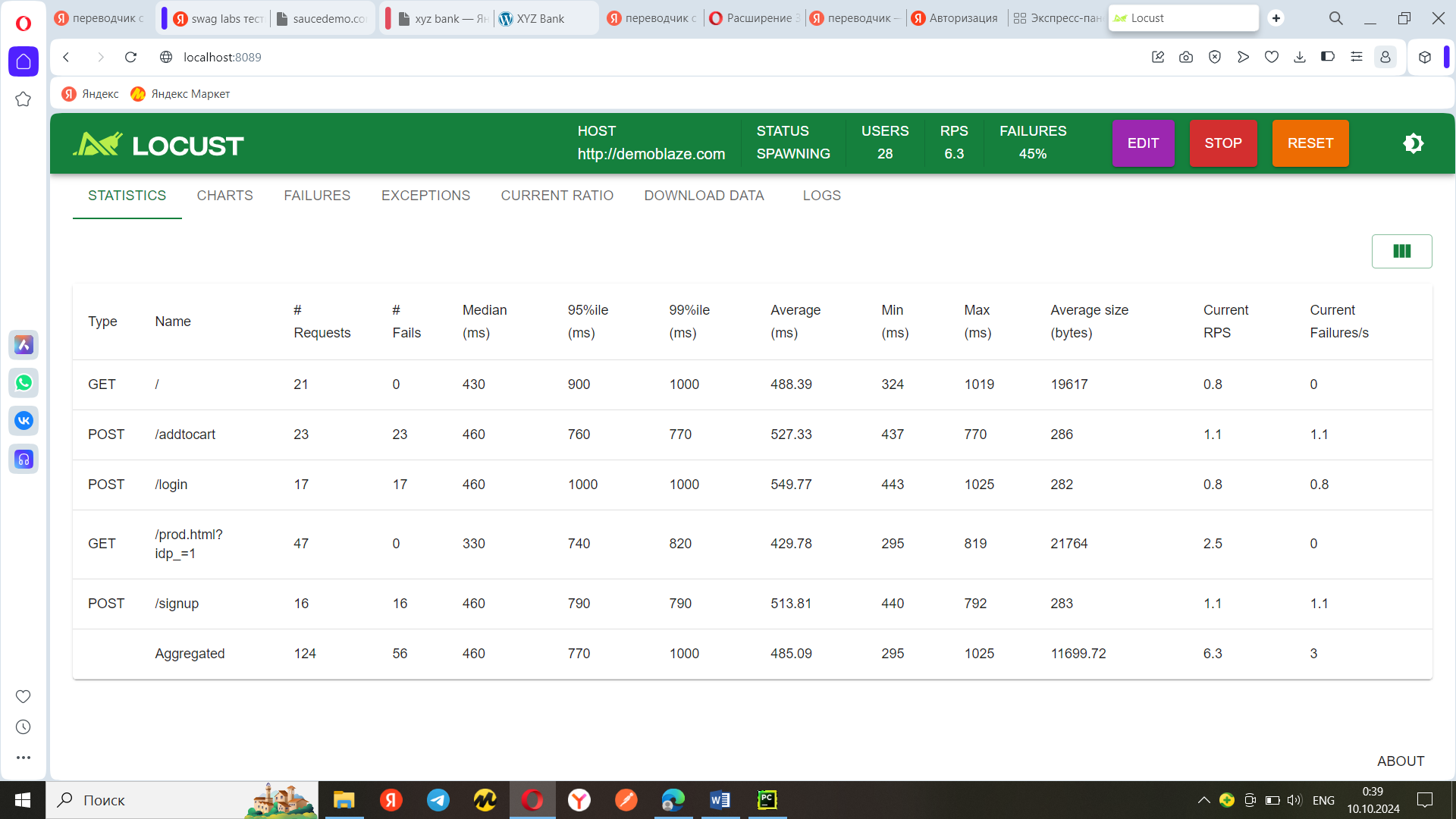
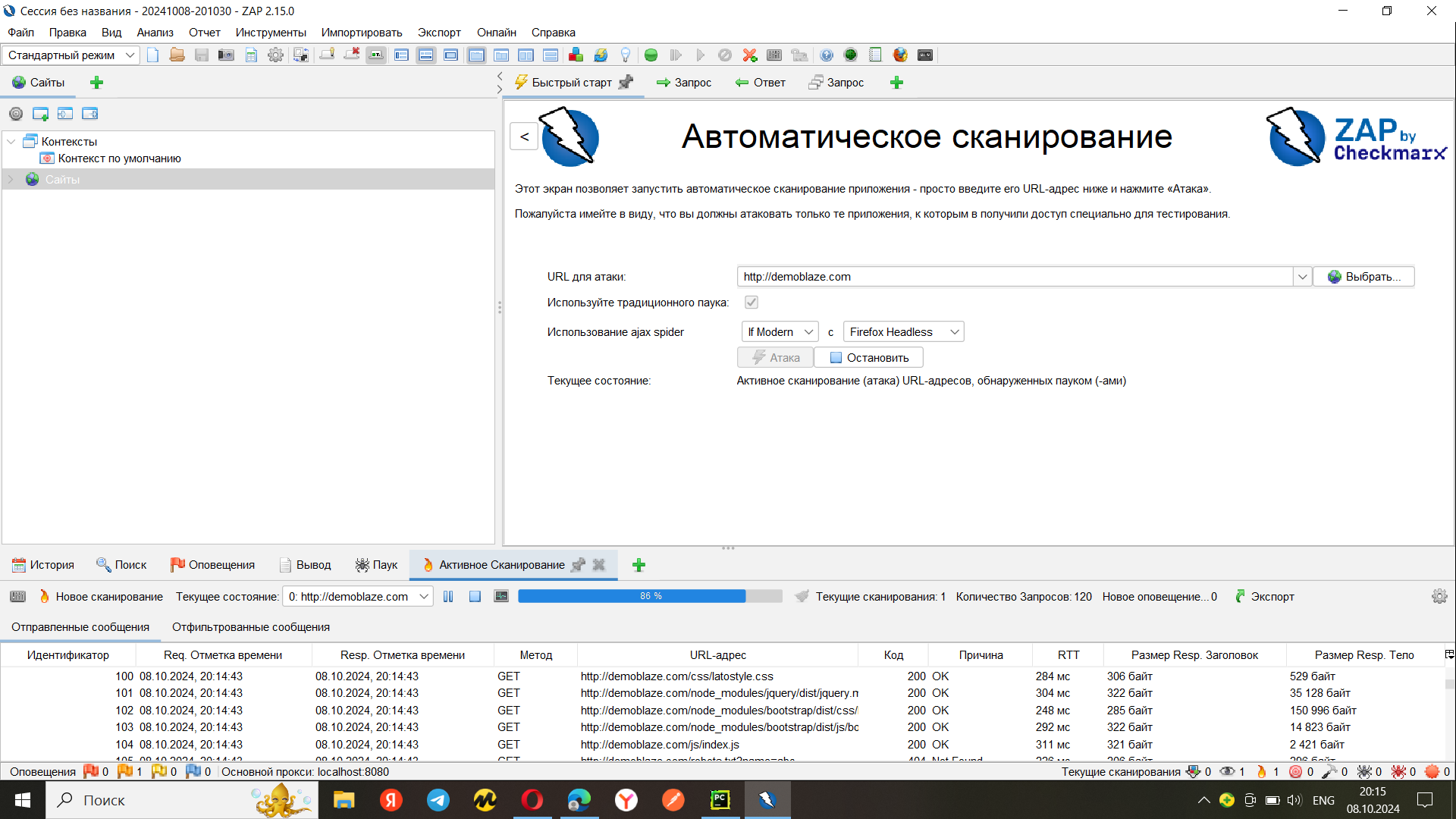


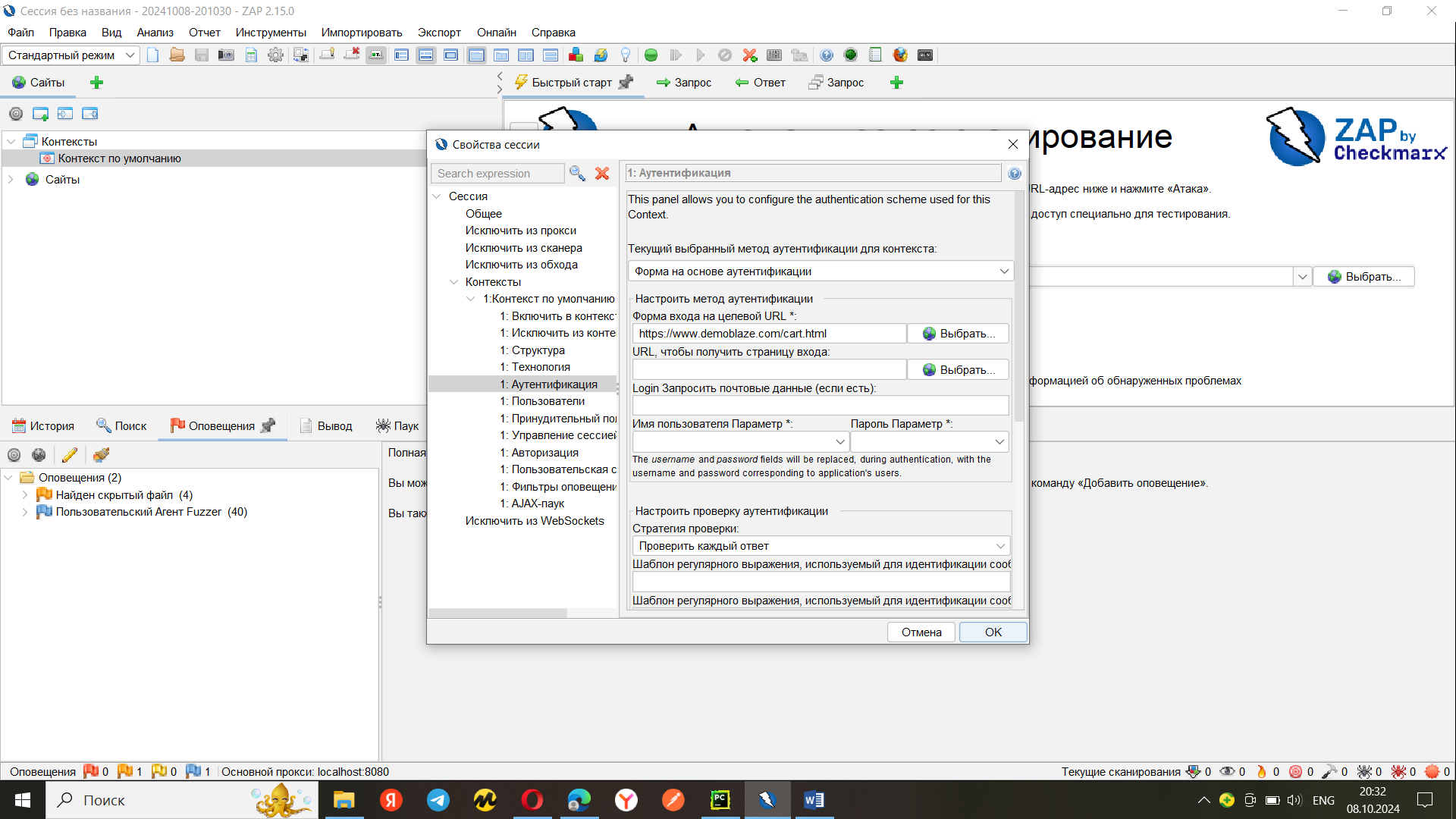
График нагрузки на сайт



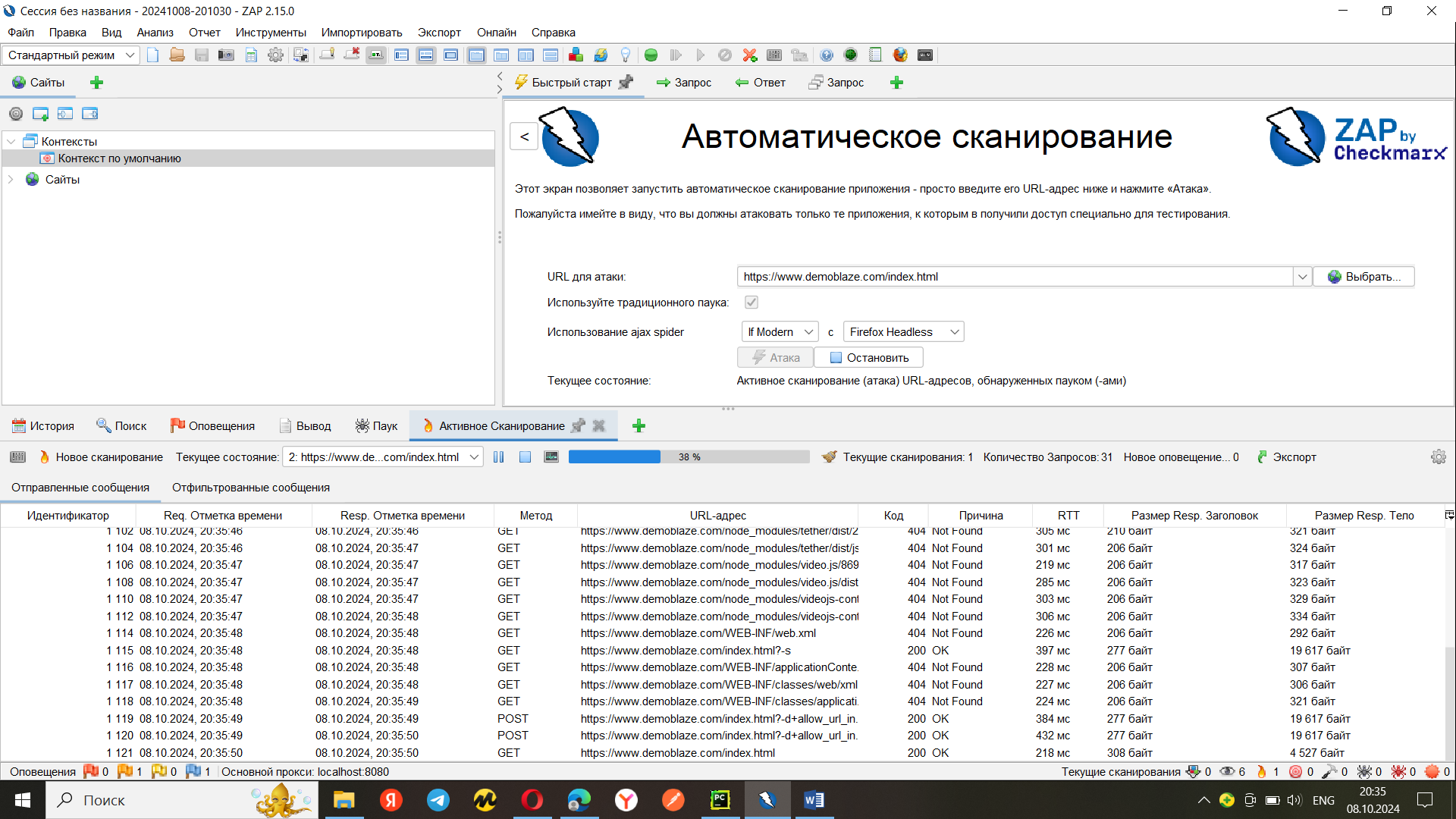
Приложение 4

***Тестирование безопасности***

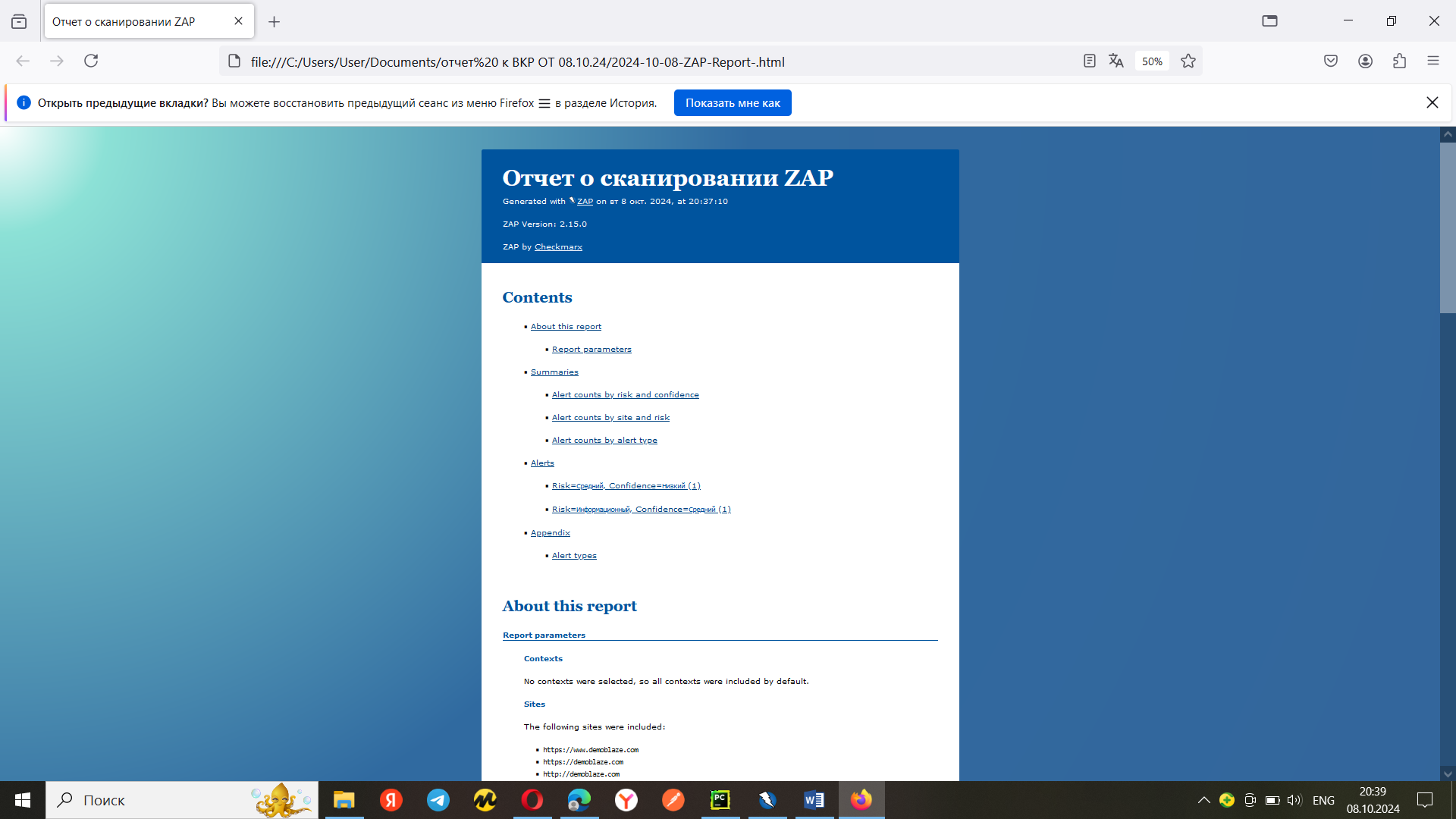


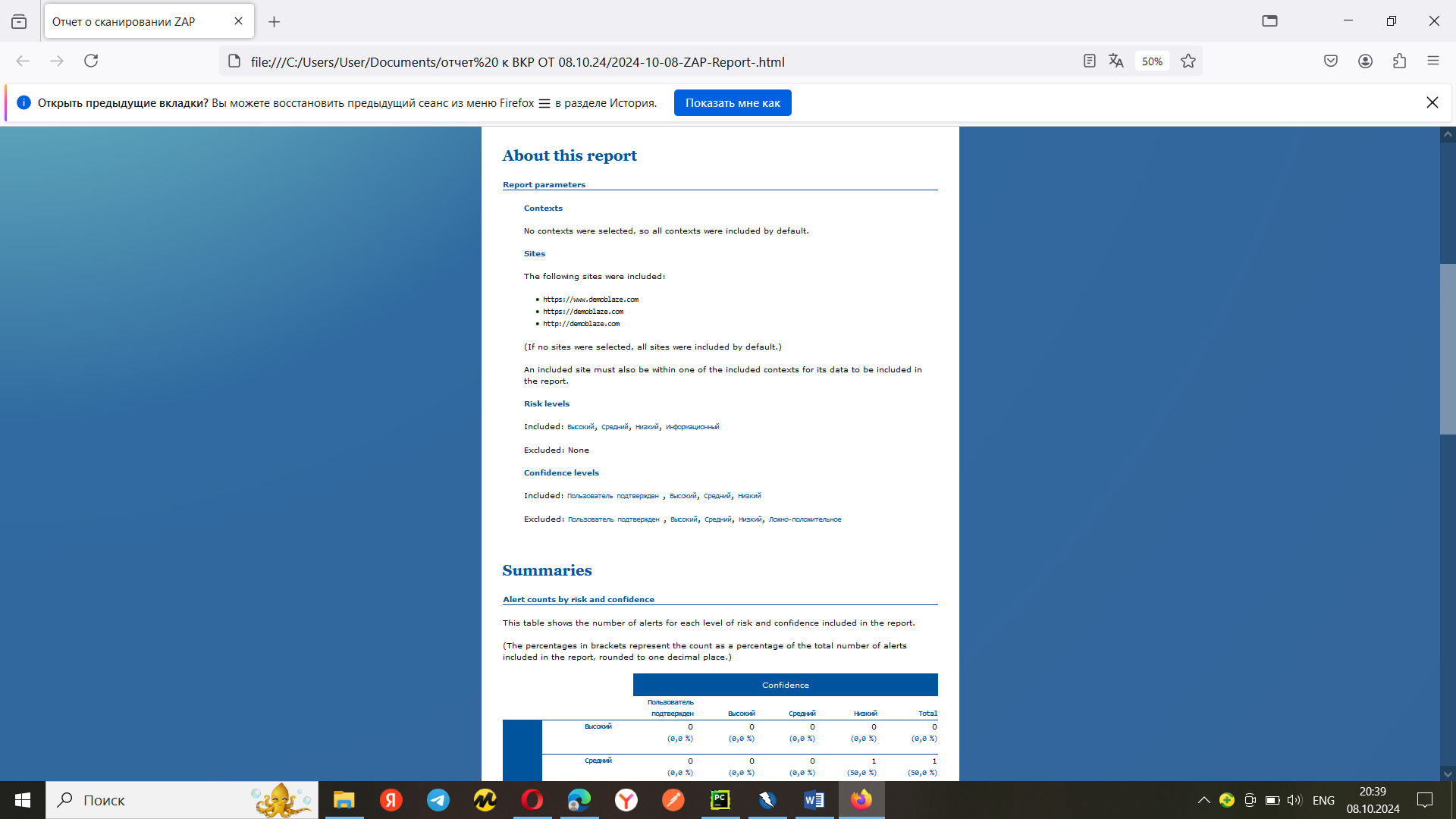


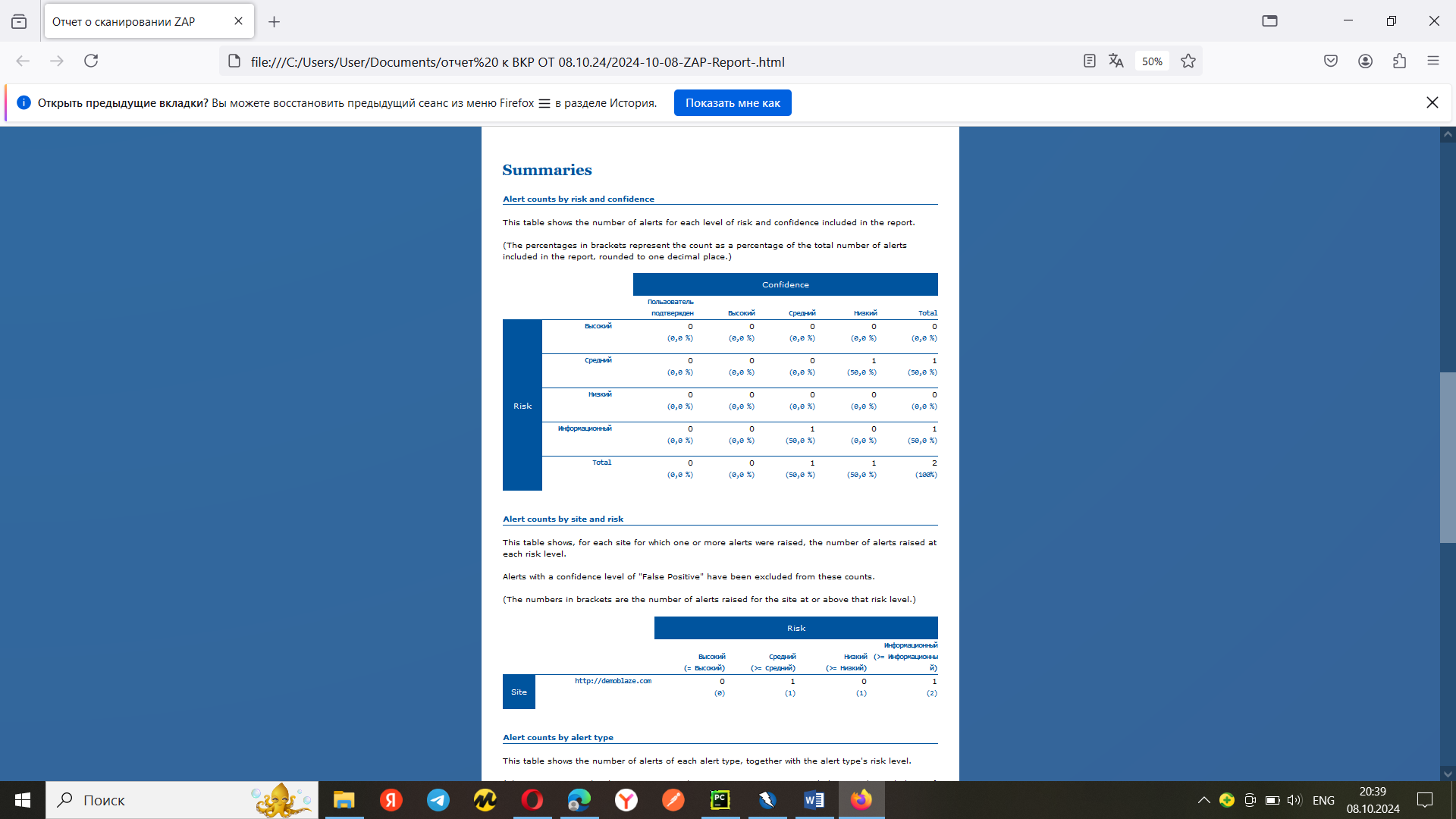
Результат запуска

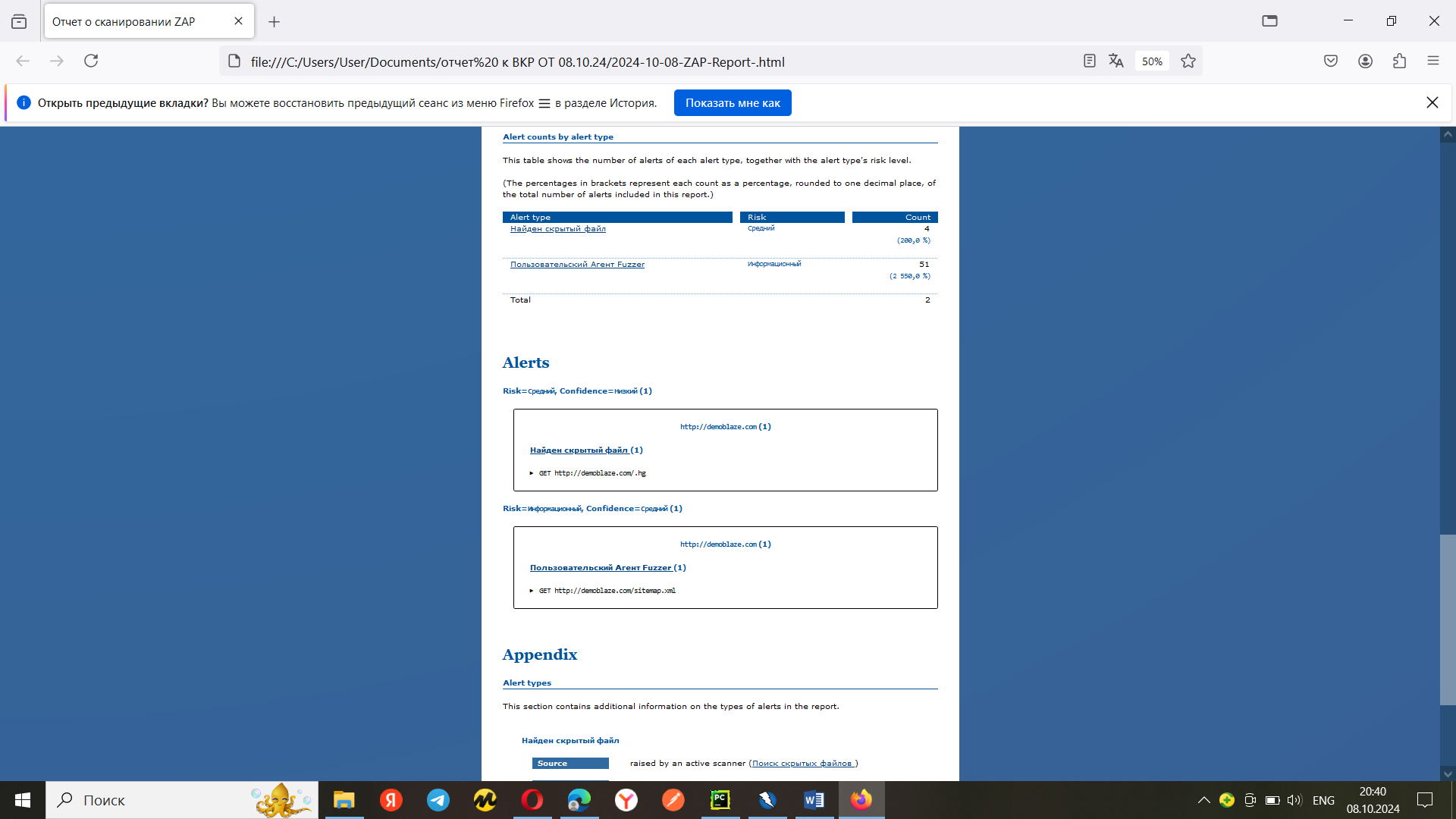


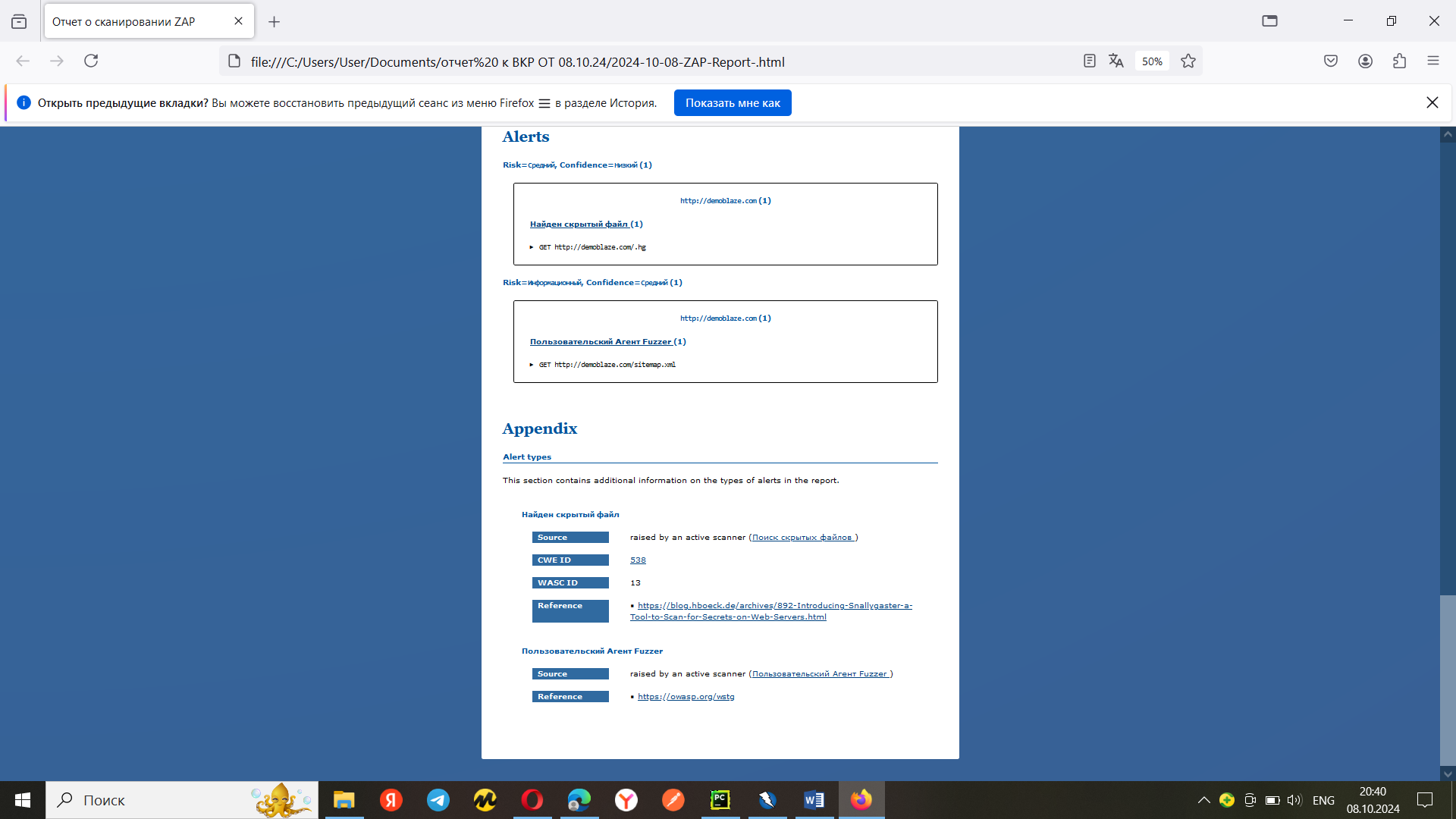
Отчет о сканировании сайта











Отчет показывает, что на сайте demoblaze.com были обнаружены два типа предупреждений:

1. **Найден скрытый файл**: Предупреждение о возможной утечке конфиденциальной информации. Рекомендуется проверить и защитить доступ к подобным файлам.
2. **Пользовательский Агент Fuzzer**: Предупреждение о возможности различий в ответах сайта в зависимости от пользовательского агента. Рекомендуется проверить и стандартизировать ответы сайта для разных типов пользователей.

Эти предупреждения указывают на потенциальные уязвимости, которые могут быть использованы злоумышленниками. Рекомендуется принять меры по устранению этих уязвимостей для повышения безопасности сайта.

***Презентация проекта***