**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Технологии автоматизации процесса разработки программного обеспечения»**

Тема: Разработка системы автоматизированного тестирования веб-приложений

Вариант 2

|  |  |  |
| --- | --- | --- |
| Студент гр. 9303 |  | Игнашов В.М. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Игнашов В.М. | | |
| Группа 9303 | | |
| Тема работы: Разработка системы автоматизированного тестирования веб-приложений | | |
| Исходные данные:  Необходимо реализовать docker-compose конфигурацию из двух узлов:   * app – контейнер с существующим демонстрационным веб-приложением * tester – контейнер для запуска всех тестов | | |
| Содержание пояснительной записки:  Содержание; Введение; Постановка задачи; Описание Dockerfile; Описание скриптов запуска тестов; Описание docker-compose конфигурации; Заключение; Список использованных источников. | | |
| Предполагаемый объем пояснительной записки:  Не менее 16 страниц. | | |
| Дата выдачи задания: | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Игнашов В.М. |
| Преподаватель |  | Заславский М.М. |

**Аннотация**

В данной курсовой работе описана конфигурация системы для автоматизированного тестирования веб-приложений – демонстрационного и тестового экземпляра ИС ИОТ. Система состоит из двух контейнеров: в одном запускается демонстрационное веб-приложение, второй используется для запуска нескольких этапов тестов, включая форматирование, статический анализ, интеграционные тесты, а также тесты с использованием веб-драйвера Selenium.

**Summary**

This course work describes the configuration of a system for automated testing of web applications - a demo and test instance of the IOT IS. The system consists of two containers: one container runs the demo web application, the second one is used to run several stages of tests, including formatting, static analysis, integration tests, as well as tests using Selenium web driver.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
|  | Постановка задачи | 6 |
| 1. | Описание Dockerfile | 8 |
| 1.1. | Dockerfile для app-контейнера | 8 |
| 1.2. | Dockerfile для tester-контейнера | 9 |
| 2. | Описание скриптов запуска тестов | 11 |
| 2.1. | Скрипт run\_tests.sh для запуска этапов тестирования | 11 |
| 2.2. | Этап форматирования | 11 |
| 2.3. | Этап статического анализа | 11 |
| 2.4. | Этап интеграционного тестирования | 12 |
| 2.5. | Этап selenium-тестирования | 12 |
| 2.6. | Описание пакета с реализованными Selenium-тестами | 13 |
| 3. | Описание docker-compose конфигурации | 14 |
|  | Заключение | 15 |
|  | Список использованных источников | 16 |
|  | Приложение А. Исходный код проекта | 17 |

**введение**

Целью данной работы является реализация docker-compose конфигурации, предназначенной для сборки и запуска контейнеров app и tester. Контейнеры по отдельности выполняют задачи, включающие в себя запуск демонстрационного веб-приложения, а также тестирование данного веб-приложения и ИС ИОТ на нескольких этапах (форматирование, статический анализ, интеграционные тесты, а также тесты с использованием веб-драйвера Selenium).

**Постановка задачи**

Необходимо реализовать docker-compose конфигурацию из двух узлов:

* app - контейнер с существующим демонстрационным веб-приложением.
  + Устанавливать приложение необходимо скачивая репозиторий и копируя файлы из него при сборке вашего контейнера:)
  + Чтобы все заработало, вам придется потратить время и поразбираться - из коробки может не работать.
  + Возможно, вам для выполнения заданий потребуются фиксы в исходник - делайте для них патчи
  + Корнем дерева процессов выступает запущенное веб-приложение
* tester - контейнер для запуска **всех** тестов (состав и особенности тестов задаются в таблице вариантов)
  + Корнем дерева процессов выступает стандартный python http сервер (python -m http.server 3000)
  + Этот сервер должен быть запущен в каталоге контейнера, где будет происходить работа тестовых скриптов
  + Тестовые скрипты запускаются через docker exec

При этом при разработке необходимо учесть следующие требования:

* Dockerfile:
  + Минимальная версия докера Docker version 19.03.13, build 4484c46d9d
  + Базовый образ ubuntu:22.04
  + Не использовать Expose
  + При установке любых пакетов и программ (в том числе в requirements) ВСЕГДА указывать версии
  + Ограничить установку зависимостей apt одной строкой (один RUN)
  + Если настройка одной части приложения состоит из нескольких команд → необходимо разместить их в одном слое (в одном RUN)
* Docker-compose:
  + Минимальная версия docker compose version 1.27.4, build 40524192
  + Все должно собиратся по команде docker-compose build без sudo
  + Не использовать тип сети HOST
  + Не отрывать лишних (непредусмотренных заданием) портов
  + Не использовать порты хост-машины ⇐1024

Параметры конфигурации, заданные для 2 варианта:

|  |  |
| --- | --- |
| Параметр | Требование |
| Проверка на соответствие стилю кодирования / бьютификация | Форматирование Python (yapf) |
| Статический анализ | Анализ по 10 существующим критериям |
| Интеграционные тесты | Проверка на заголовки |
| Selenium-тесты | Создание ОПОП, заполнение вкладок 4, 5, 6. |
| Внешний SSH доступ в контейнеры | В app – по публичному ключу (существующему) |
| Вывод логов работы tester | Каждый этап тестирования - в docker log (stdout + stderr) и в общие файлы (отдельно - для stdout, отдельно - для stderr) |
| Передача параметров в конфигурацию через .env | Порт для веб-сервера |
| Органичения ресурсов | настройки | Ядра процессора |

**1. Описание Dockerfile**

**1.1. Dockerfile для app-контейнера**

Последовательность инструкций создания образа app-контейнера:

1. Базовый образ - ubuntu:22.04.
2. Обновляются пакетные списки и устанавливаются необходимые apt-зависимости:
   1. openssh-server – предоставляет SSH сервер для дальнейшего доступа по SSH
   2. git – система управления версиями для дальнейшего клонирования репозитория с демонстрационным приложением
   3. python3 – интерпретатор Python, необходимый для запуска веб-приложения
   4. python3-pip – пакетный менеджер Python, используемый для установки зависимостей
3. Производится настройка конфигурации SSH сервера для разрешения входа под пользователем root, копируется публичный SSH ключ
4. Клонируется репозиторий с демонстрационным веб-приложением и устанавливается рабочая директория внутри этого репозитория
5. Копируется реализованный patch-файл, изменяющий main.py для корректной работы веб-приложения.
6. Устанавливаются необходимые зависимости Python, используемые в веб-приложении:
   1. Flask – фреймворк для создания веб-приложений
   2. lti – библиотека для реализации LTI веб-приложений
   3. flask-login – расширение Flask для аутентификации
   4. celery – асинхронная очередь задач
7. Задается точка входа для контейнера, запускающая SSH сервер и веб-приложение.

**1.2. Dockerfile для tester-контейнера**

Последовательность инструкций создания образа tester-контейнера:

1. Базовый образ - ubuntu:22.04.
2. Обновляются пакетные списки и устанавливаются необходимые apt-зависимости:
   1. git – система управления версиями для дальнейшего клонирования репозитория с демонстрационным приложением
   2. python3 – интерпретатор Python, необходимый для запуска веб-приложения
   3. python3-pip – пакетный менеджер Python, используемый для установки зависимостей
   4. wget – утилита для загрузки файлов, необходимая для загрузки Google Chrome для использования в Selenium-тестах
   5. xvfb – виртуальный фреймбуфер X, используемый для запуска Google Chrome виртуально в рамках Selenium-тестирования
3. Скачивается и устанавливается браузер Google Chrome для дальнейшего выполнения Selenium-тестов с его использованием.
4. Клонируется демонстрационный проект из репозитория
5. Устанавливаются необходимые зависимости Python, используемые в тестах:
   1. yapf – инструмент для автоматического форматирования Python кода.
   2. pylint – инструмент статического анализа Python кода, проверяющий соответствие стандартам
   3. requests – библиотека для отправки HTTP-запросов, используется в интеграционных тестах
   4. pytest – фреймворк написания и выполнения Python-тестов
   5. selenium – библиотека для автоматизации веб-браузера
   6. webdriver-manager – инструмент для установки веб-драйверов (в частности для браузера Google Chrome)
6. Устанавливается пакет python3-tk, используемый на последнем этапе Selenium-тестов. Данный пакет устанавливается отдельно от других apt-зависимостей для предотвращения интерактивного режима (необходимо установление временной зоны)
7. Копируются файлы из директории /tests внутрь контейнера
8. Задается точка входа для контейнера, запускающая веб-сервер http на порту 3000.

**2. Описание скриптов запуска тестов**

**2.1. Скрипт run\_tests.sh для запуска этапов тестирования**

Запуск различных этапов тестирования возможен с использованием реализованного скрипта run\_tests.sh, запускающего каждый из этапов по отдельности или совместно. В процессе выполнения тестов результаты записываются в монтированную в контейнер папку, создаваемую внутри скрипта.

При выполнении данного скрипта есть возможность передачи дополнительного аргумента, определяющего конкретный этап тестирования: code\_style\_tests, pylint\_tests, integration\_tests, selenium\_tests для форматирования, статического анализа, интеграционных тестов, а также тестов с использованием веб-драйвера Selenium соответственно. Для каждого из этапов тестирования реализована отдельная функция, выполняющая их запуск.

В каждой из функций в первую очередь выводится информация об этапе, после чего выполняются необходимые для запуска команды. Результаты выполнения с помощью команды tee перенаправляются в файлы stderr.log и stdout.log, а также в stdout контейнера.

**2.2. Этап форматирования**

Для выполнения форматирования используется утилита yapf. При выполнении используются параметры `-r` для рекурсивного форматирования всех файлов и каталого в текущей директории, `-vv`, устанавливающий максимальный уровень подробности вывода, а также `-i` для применения изменений к файлам. Форматирование производится в соответствии с PEP 8.

**2.3. Этап статического анализа**

При запуске данного этапа скрипт в случае отсутствия файла \_\_init\_\_.py, создает его для инициализации пакета. Это обеспечивает корректную работу линтера.

Далее последовательно запускаются два вида статического анализа для разных требований.

В первом случае используется файл конфигурации pylintrc, в котором от линтера скрываются файлы тестов и выполняется проверка для 10 чекеров: basic, classes, design, format, imports, nonascii-checker, spelling, string, variables, typecheck.

Во втором случае используется файл конфигурации pylintrc\_custom, где единственным критерием проверки является наличие названий переменных Vadim в любом регистре с помощью параметра bad-names-rgxs с установленным регулярным выражением (?i)vadim.

Созданный \_\_init\_\_.py удаляется.

**2.4. Этап интеграционного тестирования**

Для запуска интеграционных тестов используется фреймворк pytest. Запускается скрипт integration\_tests.py с маркером integration\_tests (установленном внутри pytest.ini).

В ходе выполнения тестов с помощью библиотеки requests скрипт выполняет GET запрос по разным маршрутам в веб-приложении, расположенном на 5000 порту внутри контейнера ci-cd-app. В ответе приложения проверяются заголовки Server, Content-Type, Connection, Content-Length, Allow и их значения.

Также для каждого запроса проверяется значение и его формат внутри заголовка Date с использованием библиотеки datetime. Допустимая погрешность – 10 секунд.

**2.5. Этап selenium тестирования**

Перед запуском selenium-тестов происходит создание и настройка виртуального экрана с помощью виртуального фреймбуфера X xvfb. Его использование необходимо для запуска Google Chrome без создания графического окружения, а также для дальнейшего использования пакета python3-tk внутри тестов без возникновения ошибок. Запуск самих тестов происходит с использованием фреймворка pytest. В качестве начальной точки команде передается реализованный пакет selenium\_tests, в результате которого выполняется следующий сценарий использования: «Создание ОПОП, заполнение вкладок 4, 5, 6.»

В процессе выполнения selenium-тестов собираются скриншоты выполнения на разных этапах, благодаря возможности веб-драйвера делать снимки экрана.

После выполнения функции уничтожается процесс виртуального экрана.

**2.6. Описание пакета с реализованными Selenium-тестами**

Для selenium-тестов был реализован пакет selenium\_tests.

В conftest.py скрипте описана фикстура-генератор, выполняющая инициализацию браузера с параметрами и закрывающая браузер по завершению сессии тестирования.

Тест, реализованный внутри selenium\_test.py выполняет действия описанные внутри actions.py в ходе выполнения которых автоматизированный тест выполняет ожидаемый сценарий:

1. Авторизация в личном кабинете
2. Авторизация за другого пользователя с id (1305)
3. Создание ОПОП документа
4. Редактирование и сохранение ОПОП документа
5. Определение состояния в формате JSON
6. Проверка корректности заполнения данных
7. Удаление документа из системы

Для удобства выполнения действий внутри пакета были определены используемые локаторы, ссылки и JS-скрипты. Взаимодействие с веб-драйвером определено внутри страниц: базовая страница, страница выбора пользователя, страница документа, страница авторизации внутри личного кабинета, страница со списком документов и базовая страница системы.

**3. Описание docker-compose конфигурации**

Конфигурация docker-compose описывается в файле docker-compose.yml и включает в себя описание запуска двух контейнеров (сервисов) – app (ci-cd-app) с помощью образа из Dockerfile\_app и tester (ci-cd-tester) с помощью образа из Dockerfile\_tester.

В процессе запуска контейнера с веб-приложением устанавливается ограничение на максимальное число ядер процессора, а также пробрасываются порты из контейнера на хост-машину:

* "127.0.0.1:${APP\_PORT}:5000" – порт веб-приложения внутри контейнера (5000) становится доступен на хост-машине по порту, указанному в переменных окружения (APP\_PORT).
* "127.0.0.1:2222:22" – порт SSH-сервера внутри контейнера (22) преобразуется в 2222 порт на хост-машине для возможности дальнейшего получения доступа по SSH, используя приватный ключ.

В процессе запуска контейнера с тестирующими скриптами внутрь контейнера передается .env файл с переменными окружения, а также монтируется директория на хостовой системе внутрь контейнера для сохранения результатов тестирования.

Для выполнения тестирования необходимо наличие запущенного контейнера app, соответствующая инструкция указана для контейнера tester.

**заключение**

По итогу выполнения курсовой работы были изучены технологии docker и docker-compose, применены на практике при реализации конфигурации из двух контейнеров – app (для запуска приложения и SSH-сервера) и tester (для запуска процесса тестирования). Была изучена технология Selenium WebDriver, знания применены на практике при реализации автоматизированных тестов.

Процесс тестирования состоит из нескольких этапов, включая форматирование с использованием yapf, статический анализ кода с помощью pylint, а также интеграционное тестирование демонстрационного веб-приложения на корректность заголовков и тестирование ИС ИОТ с помощью веб-драйвера Selenium на сценарии «Создание ОПОП, заполнение вкладок 4, 5, 6». Запуск интеграционных и selenium-тестов выполнялся при помощи фреймворка pytest.

**список использованных источников**

1. Docker Docs [Электронный ресурс]. URL: <https://docs.docker.com/> (дата обращения: 11.04.2024)
2. Pylint 3.1.0 documentation [Электронный ресурс]. URL: <https://pylint.readthedocs.io/en/stable/> (дата обращения: 11.04.2024)
3. pytest: helps you write better programs [Электронный ресурс]. URL: https://docs.pytest.org/en/8.0.x/ (дата обращения: 11.04.2024)
4. The Selenium Browser Automation Project | Selenium [Электронный ресурс]. URL: <https://www.selenium.dev/documentation/> (дата обращения: 11.04.2024)
5. Google/yapf: A formatter for Python files [Электронный ресурс]. URL: <https://github.com/google/yapf> (дата обращения: 11.04.2024)
6. ИС «ИОТ» [Электронный ресурс]. URL: <https://digital.etu.ru/trajectories> (дата обращения 11.04.2024)
7. Linux man pages [Электронный ресурс] URL: <https://linux.die.net/man/> (дата обращения 11.04.2024)

**приложение А**

**Исходный код Проекта**

**Dockerfile\_app:**

FROM ubuntu:22.04  
  
# Installing packages  
RUN apt-get update && apt-get install -y \  
 openssh-server=1:8.9p1-3ubuntu0.6 \  
 git=1:2.34.1-1ubuntu1.10 \  
 python3=3.10.6-1~22.04 \  
 python3-pip=22.0.2+dfsg-1ubuntu0.4  
  
# Allowing to ssh as root and setting public key  
RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd\_config  
COPY ssh-keys/id\_rsa.pub /root/.ssh/authorized\_keys  
  
# Cloning project with web application  
RUN git clone https://github.com/moevm/devops-examples.git  
WORKDIR devops-examples/EXAMPLE\_APP  
  
# Fixing host parameter in main.py file  
COPY app\_run\_host.patch ./  
RUN patch main.py app\_run\_host.patch  
  
# Installing dependencies  
RUN pip3 install \  
 flask==3.0.3 \  
 lti==0.9.5 \  
 flask\_login==0.6.3 \  
 celery==5.3.6  
  
# Starting ssh and web application  
ENTRYPOINT ["bash", "-c", "service ssh start && python3 main.py"]

**Dockerfile\_tester:**

FROM ubuntu:22.04  
  
# Installing packages  
RUN apt-get update && apt-get install -y \  
 git=1:2.34.1-1ubuntu1.10 \  
 python3=3.10.6-1~22.04 \  
 python3-pip=22.0.2+dfsg-1ubuntu0.4 \  
 wget=1.21.2-2ubuntu1 \  
 xvfb=2:21.1.4-2ubuntu1.7~22.04.10  
  
# Installing google chrome for selenium tests  
RUN wget https://dl.google.com/linux/direct/google-chrome-stable\_current\_amd64.deb \  
 && dpkg -i google-chrome-stable\_current\_amd64.deb; apt-get -fy install  
  
# Cloning project with web application  
RUN git clone https://github.com/moevm/devops-examples.git  
WORKDIR devops-examples/EXAMPLE\_APP  
  
# Installing dependencies  
RUN pip install \  
 yapf==0.40.2 \  
 pylint==3.1.0 \  
 requests==2.31.0 \  
 pytest==8.1.1 \  
 selenium==4.19.0 \  
 webdriver-manager==4.0.1  
  
# Installing python3-tk for copying json on the last stage of selenium\_tests  
RUN DEBIAN\_FRONTEND=noninteractive TZ=Europe/Moscow apt-get install -y python3-tk=3.10.8-1~22.04  
  
# Copying tests files  
COPY tests ./tests  
  
# Running the http.server  
ENTRYPOINT ["python3", "-m", "http.server", "3000"]

**docker-compose.yml:**

version: "3"  
services:  
 app:  
 container\_name: ci-cd-app  
 build:  
 dockerfile: ./Dockerfile\_app  
 ports:  
 - "127.0.0.1:${APP\_PORT}:5000"  
 - "127.0.0.1:2222:22"  
 deploy:  
 resources:  
 limits:  
 cpus: "1"  
 tester:  
 container\_name: ci-cd-tester  
 env\_file:  
 - .env  
 build:  
 dockerfile: Dockerfile\_tester  
 volumes:  
 - ./tests/test\_results:/devops-examples/EXAMPLE\_APP/tests/test\_results  
 depends\_on:  
 - app

**app\_run\_host.patch:**

78c78  
< app.run(debug = True)  
---  
> app.run(debug = True, host='0.0.0.0')

**.env.example:**

APP\_PORT=5000  
LOGIN=login  
PASSWORD=password

**/tests/run\_tests.sh:**

#!/bin/bash  
  
CYAN='\033[0;36m'  
NC='\033[0m' # No Color  
  
TESTS\_FOLDER='/devops-examples/EXAMPLE\_APP/tests'  
  
# Create folder if not exist  
mkdir -p ${TESTS\_FOLDER}/test\_results;  
  
  
function code\_style\_tests {  
 echo -e "${CYAN}CODE STYLE TESTS${NC}";  
 # run yapf style tests  
 yapf -ir -vv --style=pep8 . \  
 > >(tee -a ${TESTS\_FOLDER}/test\_results/stdout.log) 2> >(tee -a ${TESTS\_FOLDER}/test\_results/stderr.log >&2) &> >(tee -a /proc/1/fd/1);  
 # output to stderr, stdout and 1 process  
}  
  
function pylint\_tests {  
  
 touch \_\_init\_\_.py; # pylint requires the \_\_init\_\_.py file to exist in target directory  
  
 echo -e "${CYAN}PYLINT 10 TESTS${NC}";  
 # run pylint with 10 enabled  
 pylint $(pwd) -v --rcfile=${TESTS\_FOLDER}/pylintrc \  
 > >(tee -a ${TESTS\_FOLDER}/test\_results/stdout.log) 2> >(tee -a ${TESTS\_FOLDER}/test\_results/stderr.log >&2) &> >(tee -a /proc/1/fd/1);  
  
 echo -e "${CYAN}PYLINT CUSTOM CRITERIA TEST${NC}";  
 # run pylint with custom name criteria  
 pylint $(pwd) -v --rcfile=${TESTS\_FOLDER}/pylintrc\_custom \  
 > >(tee -a ${TESTS\_FOLDER}/test\_results/stdout.log) 2> >(tee -a ${TESTS\_FOLDER}/test\_results/stderr.log >&2) &> >(tee -a /proc/1/fd/1);  
  
 rm \_\_init\_\_.py; # remove added \_\_init\_\_.py  
}  
  
function integration\_tests {  
 echo -e "${CYAN}INTEGRATION TESTS${NC}";  
 # run tests with integration\_tests marker from integration\_tests.py script  
 pytest -s -v -m integration\_tests ${TESTS\_FOLDER}/integration\_tests.py \  
 > >(tee -a ${TESTS\_FOLDER}/test\_results/stdout.log) 2> >(tee -a ${TESTS\_FOLDER}/test\_results/stderr.log >&2) &> >(tee -a /proc/1/fd/1);  
}  
  
function selenium\_tests {  
 echo -e "${CYAN}SELENIUM TESTS${NC}";  
  
 # Attach display  
 exec -a xvfb-run Xvfb :1 -screen 0 1920x1080x16 &> xvfb.log &  
 DISPLAY=:1.0  
 export DISPLAY  
  
 # run tests with selenium\_tests marker from selenium\_tests.py script  
 pytest -s -v -m selenium\_tests -c ${TESTS\_FOLDER}/pytest.ini ${TESTS\_FOLDER}/selenium\_tests \  
 > >(tee -a ${TESTS\_FOLDER}/test\_results/stdout.log) 2> >(tee -a ${TESTS\_FOLDER}/test\_results/stderr.log >&2) &> >(tee -a /proc/1/fd/1);  
 # screenshots from selenium are placed under <project\_directory>/tests/test\_results  
  
 kill $(pgrep -f xvfb-run)  
}  
  
if [ $# -eq 0 ]  
 then  
 code\_style\_tests  
 pylint\_tests  
 integration\_tests  
 selenium\_tests  
 else  
 $1  
fi

**/tests/pylintrc:**

[MAIN]  
ignore=selenium\_tests, integration\_tests.py  
  
[MESSAGES CONTROL]  
disable = all  
enable = basic, classes, design, format, imports, nonascii-checker, spelling, string, variables, typecheck

**/tests/pylintrc\_custom:**

[MESSAGES CONTROL]  
disable = all  
enable=disallowed-name  
  
[BASIC]  
bad-names-rgxs=(?i)vadim

**/tests/pytest.ini:**

[pytest]  
markers =  
 integration\_tests: mark a test as an integration one  
 selenium\_tests: mark a test as a selenium one

**/tests/integration\_tests.py:**

import sys  
  
import pytest  
import requests  
from datetime import datetime  
  
APP\_NAME = 'ci-cd-app'  
APP\_PORT = '5000'  
  
  
@pytest.mark.integration\_tests  
class TestApplication:  
 @pytest.mark.parametrize("suffix,additional\_headers", [  
 ('', {'Content-Length': '589'}),  
 ('files', {'Content-Length': '69'}),  
 ('login', {'Content-Length': '153', 'Allow': ['POST', 'OPTIONS']})  
 ])  
 def test\_headers(self, suffix, additional\_headers):  
 headers\_for\_all = {  
 "Server": 'Werkzeug/3.0.2 Python/3.10.12',  
 "Content-Type": 'text/html; charset=utf-8',  
 "Connection": 'close'  
 }  
  
 url = f'http://{APP\_NAME}:{APP\_PORT}/{suffix}'  
  
 try:  
 actual = requests.get(url).headers  
 except requests.exceptions.RequestException as e:  
 assert False, f"Can't GET: {url=}. {e}"  
 for header, expected in (headers\_for\_all | additional\_headers).items():  
 assert header in actual, f'Missing {header=}'  
 if isinstance(expected, list):  
 assert all((p in actual[header]) for p in expected), f'{expected=}, {actual[header]=}'  
 else:  
 assert expected == actual[header], f'{expected=}, {actual[header]=}'  
  
 # Check Date header:  
 assert 'Date' in actual, f'Missing "Date" header'  
 try:  
 actual\_date = datetime.strptime(actual['Date'], '%a, %d %b %Y %H:%M:%S %Z')  
 except ValueError as e:  
 assert False, f"Can't parse 'Date' header: {actual['Date']}. {e}"  
 assert abs((datetime.utcnow() - actual\_date).seconds) < 10, \  
 f'Incorrect "Date" header: {datetime.utcnow()}, {actual\_date}'

**/tests/selenium\_tests/\_\_init\_\_.py:**

**/tests/selenium\_tests/conftest.py:**

import pytest  
from selenium.webdriver import Chrome  
from selenium.webdriver.chrome.options import Options  
from selenium.webdriver.chrome.service import Service  
from webdriver\_manager.chrome import ChromeDriverManager  
  
  
@pytest.fixture(scope='session')  
def browser():  
 options = Options()  
 options.add\_argument('--no-sandbox')  
 options.add\_argument('--window-size=1920,1080')  
 options.add\_argument('--disable-dev-shm-usage')  
  
 browser = Chrome(service=Service(ChromeDriverManager().install()), options=options)  
 browser.maximize\_window()  
 browser.implicitly\_wait(10)  
 yield browser  
 browser.quit()

**/tests/selenium\_tests/selenium\_test.py:**

import os  
  
import pytest  
from selenium.webdriver.chrome.webdriver import WebDriver  
  
from .actions import TrajectoriesActions  
from .data\_test import DataTest  
  
LOGIN, PASSWORD = os.getenv('LOGIN'), os.getenv('PASSWORD')  
  
  
@pytest.mark.selenium\_tests  
class TestSelenium:  
  
 def test\_fill\_456(self, browser: WebDriver):  
 actions = TrajectoriesActions()  
 actions.authorize\_lk\_etu(browser, LOGIN, PASSWORD)  
 actions.auth\_as\_person\_with\_id(browser, DataTest.PERSON\_ID)  
 actions.create\_opop(browser)  
 actions.edit\_opop\_document(browser)  
 actions.get\_document\_json(browser)  
 actions.check\_if\_data\_is\_saved()  
 actions.remove\_document\_with\_code(browser, DataTest.STUDY\_PLAN.split()[0])

**/tests/selenium\_tests/actions.py:**

import re  
from selenium.webdriver.chrome.webdriver import WebDriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support import expected\_conditions  
  
from .pages.trajectories\_page import TrajectoriesPage  
from .pages.etu\_auth\_page import EtuAuthPage  
from .pages.admin\_fake\_page import AdminFakePage  
from .pages.opop\_list\_page import OpopListPage  
from .pages.document\_page import DocumentPage  
  
from .pages.control.urls import Urls  
from .data\_test import DataTest  
  
  
class TrajectoriesActions:  
 \_content = None  
 \_json\_data = None  
  
 def authorize\_lk\_etu(self, browser: WebDriver, login, password):  
 traj\_page = TrajectoriesPage(browser, Urls.TRAJECTORIES)  
 traj\_page.take\_screenshot()  
 traj\_page.remove\_modal\_if\_exists()  
 traj\_page.accept\_cookies()  
 traj\_page.enter\_by\_etu()  
  
 etu\_lk\_page = EtuAuthPage(traj\_page.get\_browser())  
 etu\_lk\_page.take\_screenshot()  
  
 etu\_lk\_page.authorize\_by\_form(login, password)  
  
 # Auth trajectories with ETU ID:  
 # the first for linking etu id with etu lk  
 # the second for linking trajectories and etu lk)  
 traj\_page = TrajectoriesPage(browser, Urls.TRAJECTORIES)  
 traj\_page.remove\_modal\_if\_exists()  
 traj\_page.enter\_by\_etu()  
 etu\_lk\_page = EtuAuthPage(traj\_page.get\_browser())  
 etu\_lk\_page.wait\_lk\_loaded()  
 etu\_lk\_page.take\_screenshot()  
 traj\_page = TrajectoriesPage(browser, Urls.TRAJECTORIES)  
 traj\_page.remove\_modal\_if\_exists()  
 traj\_page.enter\_by\_etu()  
 # etu\_lk\_page = EtuAuthPage(traj\_page.get\_browser(), transition\_needed=False)  
 # etu\_lk\_page.authorize\_by\_form(LOGIN, PASSWORD)  
  
 def auth\_as\_person\_with\_id(self, browser: WebDriver, person\_id: int):  
 adm\_page = AdminFakePage(browser, Urls.ADMIN\_FAKE)  
 adm\_page.remove\_modal\_if\_exists()  
 adm\_page.take\_screenshot()  
 adm\_page.auth\_as\_person(person\_id)  
  
 def create\_opop(self, browser: WebDriver):  
 opop\_list\_page = OpopListPage(browser, Urls.OPOP\_LIST)  
 opop\_list\_page.remove\_modal\_if\_exists()  
 opop\_list\_page.take\_screenshot()  
 opop\_list\_page.create\_new\_document(DataTest.STUDY\_FIELD, DataTest.STUDY\_PLAN)  
 opop\_list\_page.take\_screenshot()  
 Urls.DOCUMENT\_PAGE = opop\_list\_page.save\_document\_url()  
 print(Urls.DOCUMENT\_PAGE)  
  
 def edit\_opop\_document(self, browser: WebDriver):  
 document\_page = DocumentPage(browser, Urls.DOCUMENT\_PAGE)  
 document\_page.remove\_modal\_if\_exists()  
 document\_page.wait\_loading()  
 document\_page.take\_screenshot()  
 document\_page.set\_implicit\_wait(1)  
 self.\_content = document\_page.fill\_sections()  
 document\_page.take\_screenshot()  
 document\_page.set\_implicit\_wait(10)  
 document\_page.save\_document()  
 document\_page.take\_screenshot()  
  
 def get\_document\_json(self, browser: WebDriver):  
 document\_page = DocumentPage(browser, Urls.DOCUMENT\_PAGE)  
 document\_page.remove\_modal\_if\_exists()  
 document\_page.wait\_loading()  
 document\_page.take\_screenshot()  
 self.\_json\_data = document\_page.get\_document\_in\_json()  
 # print(self.\_json\_data)  
 document\_page.take\_screenshot()  
  
 def check\_if\_data\_is\_saved(self):  
 def get\_value\_by\_json\_pointer(pointer, data\_json):  
 current = data\_json  
 for field in pointer.split('/'):  
 current = current[int(field) if field.isnumeric() else field]  
 return current  
  
 for value, mapping in DataTest.MAPPING\_DATA\_TO\_JSON.items():  
 value = self.\_content[value]  
 if isinstance(mapping, str):  
 expected = get\_value\_by\_json\_pointer(mapping, self.\_json\_data)  
 assert value.strip() in expected, f'Incorrect value for {mapping}. Expected: {expected}. Got: {value}'  
 else:  
 for additional, val in zip(['code', 'value'], re.split(r'\.? ', value, 1)):  
 expected = get\_value\_by\_json\_pointer(mapping[additional], self.\_json\_data)  
 assert val.strip() in expected, f'Incorrect value for {mapping}. Expected: {expected}. Got: {val}'  
  
 def remove\_document\_with\_code(self, browser: WebDriver, code: str):  
 opop\_list\_page = OpopListPage(browser, Urls.OPOP\_LIST)  
 opop\_list\_page.remove\_modal\_if\_exists()  
 opop\_list\_page.take\_screenshot()  
 opop\_list\_page.remove\_document\_with\_code(code)  
 opop\_list\_page.take\_screenshot()

**/tests/selenium\_tests/pages/\_\_init\_\_.py:**

**/tests/selenium\_tests/pages/base\_page.py:**

import os  
from datetime import datetime  
  
from selenium.common import NoSuchElementException  
from selenium.webdriver.chrome.webdriver import WebDriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support import expected\_conditions  
from selenium.webdriver.support.wait import WebDriverWait  
  
from .control.js\_scripts import JSScript  
  
TEST\_RESULTS = os.path.join(os.path.dirname(\_\_file\_\_), '../../test\_results/')  
SCREENSHOT\_FORMAT = "%Y-%m-%d-%H-%M-%S-%f"  
  
class BasePage:  
 \_browser = None  
  
 def \_\_init\_\_(self, browser: WebDriver, transition\_needed=None):  
 self.\_browser = browser  
 if transition\_needed:  
 self.\_browser.get(transition\_needed)  
  
 def take\_screenshot(self):  
 try:  
 os.makedirs(f'{TEST\_RESULTS}/selenium\_screenshots')  
 except FileExistsError:  
 pass  
 filename = f'{TEST\_RESULTS}/selenium\_screenshots/{datetime.utcnow().strftime(SCREENSHOT\_FORMAT)}.png'  
 print('Taken screenshot:', filename)  
 self.\_browser.save\_screenshot(filename)  
  
 def element\_exists(self, locator):  
 try:  
 self.\_browser.find\_element(locator[0], locator[1])  
 except NoSuchElementException:  
 return False  
 return True  
  
 def find\_by\_locator(self, locator):  
 try:  
 elem = self.\_browser.find\_element(locator[0], locator[1])  
 self.\_browser.execute\_script(JSScript.SCROLL\_INTO\_VIEW, elem)  
 except NoSuchElementException:  
 assert False, f'Element not found, {self.\_browser.current\_url=}, {locator=}'  
 return elem  
  
 def find\_multiple\_by\_locator(self, locator):  
 return self.\_browser.find\_elements(locator[0], locator[1])  
  
 def remove\_elements(self, locator):  
 elements = self.find\_multiple\_by\_locator(locator)  
 for el in elements:  
 self.\_browser.execute\_script(JSScript.REMOVE\_ELEMENT, el)  
  
 def wait\_until(self, until\_action):  
 WebDriverWait(self.\_browser, 10).until(  
 until\_action  
 )  
  
 def wait\_url\_change(self, action=None):  
 prev\_url = self.\_browser.current\_url  
 if action:  
 action()  
 self.wait\_until(expected\_conditions.url\_changes(prev\_url))  
  
 def wait\_element\_loaded(self, locator):  
 self.wait\_until(expected\_conditions.element\_to\_be\_clickable(locator))  
  
 def set\_implicit\_wait(self, seconds):  
 self.\_browser.implicitly\_wait(seconds)  
  
 def get\_xpath(self, elm):  
 e = elm  
 xpath = elm.tag\_name  
 while e.tag\_name != "html":  
 e = e.find\_element(By.XPATH, "..")  
 neighbours = e.find\_elements(By.XPATH, "../" + e.tag\_name)  
 level = e.tag\_name  
 if len(neighbours) > 1:  
 level += "[" + str(neighbours.index(e) + 1) + "]"  
 xpath = level + "/" + xpath  
 return "/" + xpath  
  
 def get\_browser(self):  
 return self.\_browser

**/tests/selenium\_tests/pages/trajectories\_page.py:**

from .base\_page import BasePage  
from .control.locators import TrajectoriesLocators, AuthEtuLocators  
  
  
class TrajectoriesPage(BasePage):  
 def remove\_modal\_if\_exists(self):  
 if self.element\_exists(TrajectoriesLocators.NAV\_BAR):  
 self.remove\_elements(TrajectoriesLocators.NAV\_BAR)  
 if self.element\_exists(TrajectoriesLocators.DEV\_SERVER\_MODAL):  
 self.remove\_elements(TrajectoriesLocators.DEV\_SERVER\_MODAL)  
  
 def enter\_by\_etu(self):  
 self.wait\_url\_change(self.find\_by\_locator(TrajectoriesLocators.ENTER\_VIA\_ETU\_ID).click)  
 # Go to ETU ID  
 self.wait\_url\_change(self.find\_by\_locator(AuthEtuLocators.SUBMIT\_BUTTON).click)  
  
 def accept\_cookies(self):  
 if self.element\_exists(TrajectoriesLocators.ACCEPT\_COOKIES):  
 self.find\_by\_locator(TrajectoriesLocators.ACCEPT\_COOKIES).click()

**/tests/selenium\_tests/pages/etu\_auth\_page.py:**

import time  
  
from .base\_page import BasePage  
from .control.locators import AuthEtuLocators  
  
  
class EtuAuthPage(BasePage):  
  
 def authorize\_by\_form(self, login, password):  
  
 if self.element\_exists(AuthEtuLocators.PASSWORD\_FIELD):  
 self.find\_by\_locator(AuthEtuLocators.EMAIL\_FIELD).send\_keys(login)  
 self.find\_by\_locator(AuthEtuLocators.PASSWORD\_FIELD).send\_keys(password)  
 if self.element\_exists(AuthEtuLocators.REMEMBER\_CHECKBOX):  
 self.find\_by\_locator(AuthEtuLocators.REMEMBER\_CHECKBOX).click()  
  
 self.wait\_url\_change(self.find\_by\_locator(AuthEtuLocators.SUBMIT\_BUTTON).click)  
  
 def wait\_lk\_loaded(self):  
 self.wait\_element\_loaded(AuthEtuLocators.LK\_STUDENT\_BODY\_ID)

**/tests/selenium\_tests/pages/admin\_fake\_page.py:**

from selenium.webdriver import ActionChains  
  
from .control.locators import AdminFakeLocators  
from .trajectories\_page import TrajectoriesPage  
  
  
class AdminFakePage(TrajectoriesPage):  
  
 def find\_person\_on\_the\_page(self, person\_id: int):  
 return self.find\_by\_locator(AdminFakeLocators.PERSON\_ID(person\_id))  
  
 def go\_to\_next\_page(self):  
 self.find\_by\_locator(AdminFakeLocators.NEXT\_PAGE).click()  
  
 def auth\_as\_person(self, person\_id):  
 self.take\_screenshot()  
 fr\_id, to\_id = (self.find\_by\_locator(AdminFakeLocators.FROM\_ID),  
 self.find\_by\_locator(AdminFakeLocators.TO\_ID))  
 while not (int(fr\_id.text.replace(',', '')) <= person\_id <= int(to\_id.text.replace(',', ''))):  
 self.go\_to\_next\_page()  
 fr\_id, to\_id = (self.find\_by\_locator(AdminFakeLocators.FROM\_ID),  
 self.find\_by\_locator(AdminFakeLocators.TO\_ID))  
 element = self.find\_person\_on\_the\_page(person\_id)  
 self.take\_screenshot()  
 ActionChains(self.\_browser).double\_click(element).perform()

**/tests/selenium\_tests/pages/opop\_list\_page.py:**

from selenium.webdriver import ActionChains  
  
from .control.locators import OPOPLocators  
from .trajectories\_page import TrajectoriesPage  
  
  
class OpopListPage(TrajectoriesPage):  
  
 def create\_new\_document(self, study\_field, study\_plan):  
 self.find\_by\_locator(OPOPLocators.CREATE\_NEW\_BUTTON).click()  
 self.wait\_element\_loaded(OPOPLocators.CREATE\_NEW\_FINISH\_BUTTON)  
 for inp, value in zip(self.find\_multiple\_by\_locator(OPOPLocators.SELECTS\_TO\_CREATE), [study\_field, study\_plan]):  
 inp.click()  
 locator = OPOPLocators.SELECTS\_TO\_CREATE\_OPTION(value)  
 option = inp.find\_element(locator[0], locator[1])  
 option.click()  
  
 self.take\_screenshot()  
 self.find\_by\_locator(OPOPLocators.CREATE\_NEW\_FINISH\_BUTTON).click()  
  
 def save\_document\_url(self):  
 self.wait\_url\_change()  
 return self.\_browser.current\_url.split('?')[0]  
  
 def remove\_document\_with\_code(self, code):  
 # define document row  
 row = self.find\_by\_locator(OPOPLocators.ROW\_WITH\_CODE(code)).get\_attribute('row-index')  
 self.find\_by\_locator(OPOPLocators.REMOVE\_BUTTON\_FOR\_ROW(row)).click()  
 self.wait\_element\_loaded(OPOPLocators.CONFIRM\_DELETE\_BUTTON)  
 self.find\_by\_locator(OPOPLocators.CONFIRM\_DELETE\_BUTTON).click()

**/tests/selenium\_tests/pages/document\_page.py:**

import json  
import os  
  
import time  
import random  
from tkinter import Tk  
  
from selenium.common import NoSuchElementException  
  
from .trajectories\_page import TrajectoriesPage  
from .control.locators import DocumentPageLocators  
from .control.js\_scripts import JSScript  
  
  
def generate\_random\_numeric(n=2):  
 return ''.join(str(random.randint(1, 9)) for \_ in range(n))  
  
  
class DocumentPage(TrajectoriesPage):  
 result = dict()  
  
 def save\_to\_result(self, upper\_tab\_num, left\_tab\_num, elem):  
 self.result[(upper\_tab\_num, left\_tab\_num, self.get\_xpath(elem))] = elem.text  
  
 def fill\_sections(self):  
 upper\_tabs = self.find\_multiple\_by\_locator(DocumentPageLocators.UPPER\_TABS)  
 for upper\_tab\_num in [2, 4, 5]:  
 # including 2nd because it is necessary for the 4th  
 # excluding 6th because there are no fields to fill  
 self.\_browser.execute\_script(JSScript.SCROLL\_TO\_TOP)  
 upper\_tabs[upper\_tab\_num].click()  
 for left\_tab\_num, left\_tab in enumerate(self.find\_multiple\_by\_locator(DocumentPageLocators.LEFT\_TABS)):  
 left\_tab.click()  
 self.fill\_section(upper\_tab\_num, left\_tab\_num)  
 time.sleep(0.5)  
 return self.result  
  
 def wait\_loading(self):  
 self.wait\_element\_loaded(DocumentPageLocators.TABS\_INCLUDED)  
  
 def fill\_section(self, upper\_tab\_num, left\_tab\_num):  
 for i, ms\_field in enumerate(self.find\_multiple\_by\_locator(DocumentPageLocators.MULTISELECT\_FIELDS)):  
 ms\_field.click()  
 options = self.find\_multiple\_by\_locator(DocumentPageLocators.MULTISELECT\_FIELDS\_OPTION)  
 if len(options) > 0:  
 self.result[f'{upper\_tab\_num},{left\_tab\_num},multiselect,{i}'] = options[0].text  
  
 options[0].click()  
 for i, textarea\_field in enumerate(self.find\_multiple\_by\_locator(DocumentPageLocators.TEXTAREA\_FIELDS)):  
 value = generate\_random\_numeric()  
 textarea\_field.send\_keys(value)  
 self.result[f'{upper\_tab\_num},{left\_tab\_num},textarea\_field,{i}'] = value  
 for i, input\_field in enumerate(self.find\_multiple\_by\_locator(DocumentPageLocators.INPUT\_FIELDS)):  
 value = generate\_random\_numeric()  
 input\_field.send\_keys(value)  
 self.result[f'{upper\_tab\_num},{left\_tab\_num},input\_field,{i}'] = value  
 for i, card\_field in enumerate(self.find\_multiple\_by\_locator(DocumentPageLocators.CARDS\_FIELDS)):  
 self.fill\_card(upper\_tab\_num, left\_tab\_num, card\_field, i)  
  
 def fill\_card(self, upper\_tab\_num, left\_tab\_num, card\_field, card\_id):  
 try:  
 locator = DocumentPageLocators.CARDS\_ADD\_BUTTON  
 card\_field.find\_element(locator[0], locator[1]).click()  
 except NoSuchElementException:  
 pass  
  
 k = 0  
 while True:  
 locator = DocumentPageLocators.UNKNOWN\_INPUT\_VXE  
 for elem in filter(lambda el: len(el.text) == 0 or el.text.isspace(),  
 card\_field.find\_elements(locator[0], locator[1])):  
  
 elem.click()  
 options = self.find\_multiple\_by\_locator(DocumentPageLocators.MULTISELECT\_FIELDS\_OPTION)  
 if len(options) > 0:  
 self.result[f'{upper\_tab\_num},{left\_tab\_num},card,{card\_id},{k}'] = options[0].text  
 options[0].click()  
 else:  
 value = generate\_random\_numeric()  
 try:  
 locator = DocumentPageLocators.INPUT\_FIELDS  
 elem.find\_element(locator[0], locator[1]).send\_keys(value)  
 except NoSuchElementException:  
 locator = DocumentPageLocators.TEXTAREA\_FIELDS  
 elem.find\_element(locator[0], locator[1]).send\_keys(value)  
 self.result[f'{upper\_tab\_num},{left\_tab\_num},card,{card\_id},{k}'] = value  
  
 # search for expanding button  
 locator = DocumentPageLocators.LAST\_EXPANDING\_BUTTON  
 exp\_buttons = card\_field.find\_elements(locator[0], locator[1])  
 locator = DocumentPageLocators.TABLE\_ROWS\_INSIDE\_CARD  
 total\_rows = card\_field.find\_elements(locator[0], locator[1])  
  
 if len(exp\_buttons) == 0 or len(total\_rows) > len(exp\_buttons) + 1:  
 return  
 self.\_browser.execute\_script(JSScript.SCROLL\_INTO\_VIEW, exp\_buttons[-1])  
 exp\_buttons[-1].click()  
  
 k += 1  
  
 def save\_document(self):  
 self.find\_by\_locator(DocumentPageLocators.SAVE\_DOCUMENT\_BUTTON).click()  
 self.wait\_element\_loaded(DocumentPageLocators.SAVED\_INFO)  
  
 def get\_document\_in\_json(self):  
 self.find\_by\_locator(DocumentPageLocators.JSON\_LINK).click()  
 self.\_browser.execute\_script(JSScript.SCROLL\_TO\_TOP)  
 self.wait\_element\_loaded(DocumentPageLocators.JSON\_COPY\_BUTTON)  
 self.find\_by\_locator(DocumentPageLocators.JSON\_COPY\_BUTTON).click()  
 self.wait\_element\_loaded(DocumentPageLocators.JSON\_COPIED\_RESPONSE)  
  
 if os.environ.get('DISPLAY', '') == '':  
 print('no display found. Using :0.0')  
 os.environ.\_\_setitem\_\_('DISPLAY', ':0.0')  
  
 return json.loads(Tk().clipboard\_get().replace('\n', ''))

**/tests/selenium\_tests/pages/control/\_\_init\_\_.py:**

**/tests/selenium\_tests/pages/control/locators.py:**

from selenium.webdriver.common.by import By  
  
PARENT\_NODE = (By.XPATH, "./..")  
PARENT\_NODE\_WITH\_TAG = lambda tag: (By.XPATH, f"/ancestor::{tag}")  
  
  
class AuthEtuLocators:  
 EMAIL\_FIELD = (By.NAME, "email")  
 PASSWORD\_FIELD = (By.NAME, "password")  
 REMEMBER\_CHECKBOX = (By.ID, "remember")  
 SUBMIT\_BUTTON = (By.XPATH, '//button[@type="submit"]')  
 LOGOUT\_LINK = (By.XPATH, '//a[contains(@href="logout")]')  
 LK\_STUDENT\_BODY\_ID = (By.ID, 'body-clone')  
  
  
class TrajectoriesLocators:  
 NAV\_BAR = (By.TAG\_NAME, 'nav')  
 DEV\_SERVER\_MODAL = (By.ID, "devServerModalId\_\_\_BV\_modal\_outer\_")  
 ENTER\_VIA\_ETU\_ID = (By.XPATH, '//button[contains(text(), "ETU ID")]')  
 ACCEPT\_COOKIES = (By.XPATH, '//\*[@data-cy = "cookies-ok-button"]')  
  
  
class AdminFakeLocators:  
 PERSON\_ID = lambda x: (  
 By.XPATH, f'//div[@ref="eCenterContainer"]//div[@col-id="id"]//span[contains(text(),\"{x}\")]')  
 NEXT\_PAGE = (By.XPATH, '//button[contains(text(),"Next")]')  
 FROM\_ID, TO\_ID = (By.XPATH, '//\*[@ref="lbFirstRowOnPage"]'), (By.XPATH, '//\*[@ref="lbLastRowOnPage"]')  
  
  
class OPOPLocators:  
 CREATE\_NEW\_BUTTON = (By.CSS\_SELECTOR, '.row button:nth-of-type(1)')  
 SELECTS\_TO\_CREATE = (By.CSS\_SELECTOR, '.multiselect')  
 SELECTS\_TO\_CREATE\_OPTION = lambda x: (By.XPATH, f'//\*[contains(text(),"{x}")]')  
 CREATE\_NEW\_FINISH\_BUTTON = (  
 By.XPATH, '//div[@id="creationModalId\_\_\_BV\_modal\_content\_"]//button[contains(text(),"Добавить")]'  
 )  
 DISABLED\_OPTIONS = (By.CSS\_SELECTOR, '.multiselect\_\_option--disabled')  
  
 ROW\_WITH\_CODE = lambda code: (By.XPATH, f'//span[text()="{code}"]/ancestor::div[@role="row"]')  
 REMOVE\_BUTTON\_FOR\_ROW = lambda row\_id: (  
 By.XPATH, f'//div[@class="ag-pinned-right-cols-container"]//div[@role="row" and @row-index={row\_id}]//button[2]'  
 )  
 CONFIRM\_DELETE\_BUTTON = (By.XPATH, '//div[@id="deleteModal"]//button[contains(text(), "Удалить")]')  
 DELETED\_INFO = (By.XPATH, '//\*[text()="Удаление выбранного ОПОП выполнено"]')  
  
  
class DocumentPageLocators:  
 TABS\_INCLUDED = (By.XPATH, '//ul[@role="tablist" and contains(@class,"nav-tabs")]')  
 UPPER\_TABS = (By.XPATH, '//ul[@role="tablist" and contains(@class,"nav-tabs")]//li[@role="presentation"]')  
 LEFT\_TABS = (By.XPATH, '//div[@class="tab-content"]//li[@role="presentation"]')  
  
 MULTISELECT\_FIELDS = (By.CSS\_SELECTOR, '.field-multiselect .multiselect')  
 MULTISELECT\_TAG = (By.CSS\_SELECTOR, '.multiselect\_\_tag')  
 MULTISELECT\_FIELDS\_OPTION = (  
 By.XPATH,  
 './/div[contains(@class,"multiselect--active")]//span[contains(@class,"multiselect\_\_option--highlight")]'  
 )  
  
 TEXTAREA\_FIELDS = (By.XPATH, './/textarea[not(@readonly)]')  
 INPUT\_FIELDS = (By.XPATH, './/input[@type and not(@class="multiselect\_\_input")]')  
  
 CARDS\_FIELDS = (By.XPATH, './/\*[@class="card-body" and not(@role)]')  
 CARDS\_ADD\_BUTTON = (By.XPATH, './/button/span[contains(text(),"Добавить")]')  
 UNKNOWN\_INPUT\_VXE = (By.XPATH, './/\*[@class="vxe-tree-cell" or @class="vxe-cell--label"]/ancestor::td')  
 LAST\_EXPANDING\_BUTTON = (By.XPATH, './/button[@title="Добавить на уровень ниже"]')  
 TABLE\_ROWS\_INSIDE\_CARD = (By.XPATH, './/tbody//tr')  
  
 SAVE\_DOCUMENT\_BUTTON = (By.XPATH, '//span[@title="Сохранить документ"]')  
 SAVED\_INFO = (By.XPATH, '//\*[text()="Выполнено сохранение текущего документа ОПОП"]')  
  
 JSON\_LINK = (By.XPATH, '//a[contains(text(), "JSON (dev)")]')  
 JSON\_COPY\_BUTTON = (By.XPATH, '//span[contains(text(),"copy")]')  
 JSON\_COPIED\_RESPONSE = (By.XPATH, '//span[contains(text(),"copied")]')  
 JSON\_CODE\_CONTENT = (By.CSS\_SELECTOR, '.jv-code.open.boxed')

**/tests/selenium\_tests/pages/control/js\_scripts.py:**

class JSScript:  
 SCROLL\_TO\_TOP = "window.scrollTo(0, 0)"  
 SCROLL\_INTO\_VIEW = "arguments[0].scrollIntoView(true);"  
 REMOVE\_ELEMENT = '''  
 var element = arguments[0];  
 element.parentNode.removeChild(element);  
 '''

**/tests/selenium\_tests/pages/control/urls.py:**

class Urls:  
 LK\_ETU = 'https://lk.etu.ru'  
 ID\_ETU = 'https://id.etu.ru'  
 TRAJECTORIES = 'https://dev.digital.etu.ru/trajectories-test'  
 ADMIN\_FAKE = 'https://dev.digital.etu.ru/trajectories-test/admin/fake'  
 OPOP\_LIST = 'https://dev.digital.etu.ru/trajectories-test/documents/opop-list'  
 DOCUMENT\_PAGE = None

**/tests/selenium\_tests/data\_test.py:**

class DataTest:  
 PERSON\_ID = 1305  
  
 STUDY\_FIELD = '11.03.04 Электроника и наноэлектроника - ФЭЛ'  
 STUDY\_PLAN = '11.03.04 Электроника и наноэлектроника (11.03.04\_20\_322.plx, 2023-2024) - каф.ЭПУ'  
  
 MAPPING\_DATA\_TO\_JSON = {  
 "2,0,multiselect,0": {  
 "code": "professional/areas/0/code",  
 "value": "professional/areas/0/title"  
 },  
 "2,0,multiselect,2": "professional/tasks/0/title",  
 "2,0,textarea\_field,0": "professional/objects",  
 "2,1,card,0,0": {  
 "code": "professional/taskRows/0/value/0/code",  
 "value": "professional/taskRows/0/value/0/title"  
 },  
 "2,1,card,0,1": "professional/taskRows/0/\_children/0/value/title",  
 "2,1,card,0,2": "professional/taskRows/0/\_children/0/\_children/0/value",  
 "2,1,card,0,3": "professional/taskRows/0/\_children/0/\_children/1/value",  
 "4,1,card,0,0": {  
 "code": "programResults/generalProfessionalData/0/competence/cipher",  
 "value": "programResults/generalProfessionalData/0/competence/title"  
 },  
 "4,1,card,0,1": {  
 "code": "programResults/generalProfessionalData/0/indicators/0/cipher",  
 "value": "programResults/generalProfessionalData/0/indicators/0/title"  
 },  
 "4,1,card,0,2": {  
 "code": "programResults/generalProfessionalData/0/indicators/1/cipher",  
 "value": "programResults/generalProfessionalData/0/indicators/1/title"  
 },  
 "4,2,card,0,1": {  
 "code": "programResults/professionalData/competences/0/\_children/0/competence/cipher",  
 "value": "programResults/professionalData/competences/0/\_children/0/competence/title"  
 },  
 "4,2,card,0,2": {  
 "code": "programResults/professionalData/competences/0/\_children/0/\_children/0/competenceIndex/cipher",  
 "value": "programResults/professionalData/competences/0/\_children/0/\_children/0/competenceIndex/title"  
 },  
 "4,2,card,1,1": "programResults/professionalData/objects/0/\_children/0/object",  
 "4,2,card,1,2": "programResults/professionalData/objects/0/\_children/1/object",  
 "4,2,card,2,1": "programResults/professionalData/bases/0/\_children/0/base/title",  
 "4,2,card,2,2": "programResults/professionalData/bases/0/\_children/1/base/title",  
 "5,0,input\_field,0": "structure/structure/mainProcent",  
 "5,0,input\_field,1": "structure/structure/block1",  
 "5,0,input\_field,2": "structure/structure/block2",  
 "5,0,input\_field,3": "structure/structure/block3",  
 "5,0,input\_field,4": "structure/structure/all"  
 }