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

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 собирается 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:
    a. openssh-server – предоставляет SSH сервер для дальнейшего доступа по SSH
    b. git – система управления версиями для дальнейшего клонирования репозитория с демонстрационным приложением
    c. python3 – интерпретатор Python, необходимый для запуска веб-приложения
    d. 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:
    a. Flask – фреймворк для создания веб-приложений
    b. lti – библиотека для реализации LTI веб-приложений
    c. flask-login – расширение Flask для аутентификации
    d. 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:
    a. git – система управления версиями для дальнейшего клонирования репозитория с демонстрационным приложением
    b. python3 – интерпретатор Python, необходимый для запуска веб-приложения
    c. python3-pip – пакетный менеджер Python, используемый для установки зависимостей
    d. wget – утилита для загрузки файлов, необходимая для загрузки Google Chrome для использования в Selenium-тестах
    e. 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:
    a. yapf – инструмент для автоматического форматирования Python кода.
    b. pylint – инструмент статического анализа Python кода, проверяющий соответствие стандартам
    c. requests – библиотека для отправки HTTP-запросов, используется в интеграционных тестах
    d. pytest – фреймворк написания и выполнения Python-тестов
    e. selenium – библиотека для автоматизации веб-браузера

    f. 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(SCREENS
HOT_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_OPT
ION)
            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_FIE
LDS)):
            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_OPT
ION)
                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/cip
her",
            "value":
"programResults/professionalData/competences/0/_children/0/competence/tit
le"
        },
        "4,2,card,0,2": {
            "code":
"programResults/professionalData/competences/0/_children/0/_children/0/co
mpetenceIndex/cipher",
            "value":
"programResults/professionalData/competences/0/_children/0/_children/0/co
mpetenceIndex/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"
    }
```