

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп'ютерних систем

КУРСОВА РОБОТА
з дисципліни "Компоненти програмної інженерії"

Виконала: Сіренко Вікторія Юріївна
Група: КП-03

Допущено до захисту

1 семестр 2022/2023

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп'ютерних систем

Узгоджено

Керівник роботи

_____/Погорелов В.В./

ЗАХИЩЕНА

"__" _____ 2022р.

з оцінкою _____

_____/Погорелов В.В../

Виконавець роботи

Сіренко Вікторія Юріївна

_____ 2022р.

Аналіз мов програмування та технологій розроблення

1. Мова програмування Python.

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Будучи високо адаптованою мовою програмування, Python дозволяє легко розробляти та підтримувати проекти різного рівня складності. Найбільші переваги Python – це гнучкість, швидкий розвиток, масштабованість і відмінна продуктивність.

2. Pytest.

Pytest - це середовище тестування, засноване на Python. Його використовують для написання та виконання тестового коду. Pytest підходить і для модульного (тестування окремих компонентів), і для функціонального тестування (тестування здатності коду задовольняти бізнес-вимог).

3. Flask.

Flask — мікрофреймворк для вебдодатків, створений з використанням Python. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2.

4. Click.

Click — це пакет Python для створення красивих інтерфейсів командного рядка, які можна компонувати, з мінімальною кількістю коду.

5. Docker.

Docker — це відкрита платформа для розробки, доставки та запуску програм. Docker дозволяє відокремити програми від інфраструктури, для швидкої доставки програмного забезпечення. За допомогою Docker можна керувати інфраструктурою так само, як і програмами. Скориставшись методологіями Docker для швидкої доставки,

тестування та розгортання коду, можна значно зменшити затримку між написанням коду та його запуском у виробництві.

Результати виконання

1. Створення образу контейнера.

Команда *docker build* використовує Dockerfile для створення нового образу контейнера.

```
● kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ docker build -t vika/lab3 .
Sending build context to Docker daemon 29.18kB
Step 1/6 : FROM python:3.8
---> 51a078947558
Step 2/6 : WORKDIR /usr/src/app
---> Using cache
---> 44d103921464
Step 3/6 : COPY . .
---> Using cache
---> f2abc7903bba
Step 4/6 : RUN pip install --no-cache-dir -r requirements.txt
---> Using cache
---> 1c813cd47499
Step 5/6 : EXPOSE 5000
---> Using cache
---> 5e44d4e46c62
Step 6/6 : CMD ["python", "./app.py"]
---> Using cache
---> 83388ddd21d1
Successfully built 83388ddd21d1
Successfully tagged vika/lab3:latest
○ kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$
```

2. Запустимо програму в контейнері, використовуючи команду *docker run* та надішлемо декілька запитів, щоб переконатися, що все працює коректно :

```
○ kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ docker run -p 5000:5000 vika/lab3
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
  WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 192-261-146
172.17.0.1 - - [29/Dec/2022 17:40:40] "POST /createdir HTTP/1.1" 201 -
172.17.0.1 - - [29/Dec/2022 17:44:54] "POST /createdir HTTP/1.1" 400 -
```

3. Спочатку розглянемо всі доступні команди CLI:

```
• kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ python3 cli.py --help
Usage: cli.py [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  appendLog
  consumeBuff
  createBin
  createBuff
  createDir
  createLog
  deleteBin
  deleteBuff
  deleteDir
  deleteLog
  getDirList
  moveBin
  moveBuff
  moveDir
  moveLog
  pushBuff
  readBin
  readLog
```

Будемо надсилати запити, використовуючи CLI:

Створимо першу папку в корені:

```
• kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ python3 cli.py createDir dir1 ""
b'Created directory: dir1/'
```

Створимо другу папку в першій директорії:

```
• kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ python3 cli.py createDir dir2 "dir1/"
b'Created directory: dir1/dir2/'
```

Створимо Log text file в першій директорії:

```
• kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ python3 cli.py createLog log1 "dir1/" "hi, is, it , log1"
b'Created log file: dir1/log1/'
• kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ python3 cli.py getDirList "dir1/"
```

Переглянемо суб директорії та файли, які знаходяться в першій папці:

```
• kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ python3 cli.py getDirList "dir1/"
b'dir2, log1, '
```

Видалимо другу директорію:

```
• kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ python3 cli.py deleteDir "dir1/dir2/"
b'Deleted from: dir1/'
```

Знову переглянемо вміст першої директорії:

```
• kuraga@ubuntu:~/qa_sem1/qa_sem1/sem_1_lab_3$ python3 cli.py getDirList "dir1/"
b'log1, '
```

Створимо третью папку:

```
• kuraga@ubuntu:~/qa_seml/qa_seml/sem_1_lab_3$ python3 cli.py createDir dir3 "dir1/"  
b'Created directory: dir1/dir3/'
```

Перенесемо Log text file з першої директорії в третю:

```
• kuraga@ubuntu:~/qa_seml/qa_seml/sem_1_lab_3$ python3 cli.py moveLog dir1/log1/ dir1/dir3/  
b'Moved: dir1/dir3/log1/'
```

Переглянемо вміст третьої директорії:

```
• kuraga@ubuntu:~/qa_seml/qa_seml/sem_1_lab_3$ python3 cli.py getDirList "dir1/dir3/"  
b'log1, '
```

Спробуємо створити binary file в неіснуючій директорії і отримаємо повідомлення про помилку:

```
• kuraga@ubuntu:~/qa_seml/qa_seml/sem_1_lab_3$ python3 cli.py createBin bin1 dir1/dir2/ hiii  
b'Can not create binary file'  
• kuraga@ubuntu:~/qa_seml/qa_seml/sem_1_lab_3$
```

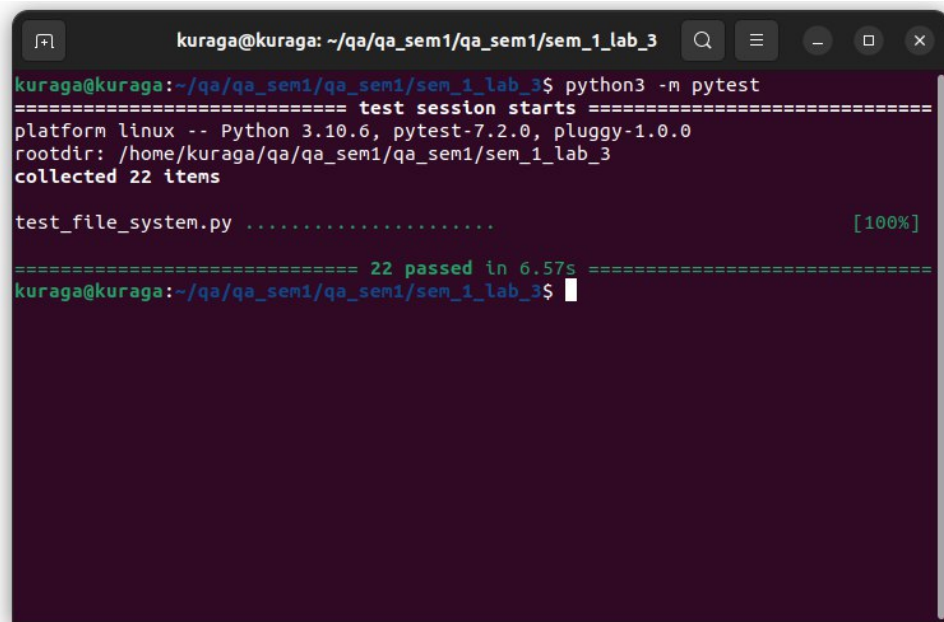
Тепер створимо binary file в першій директорії:

```
• kuraga@ubuntu:~/qa_seml/qa_seml/sem_1_lab_3$ python3 cli.py createBin bin1 dir1/ hiii  
b'Created binary file: dir1/bin1/'
```

Переглянемо всі запити, які прийняв сервер:

```
• kuraga@ubuntu:~/qa_seml/qa_seml/sem_1_lab_3$ /bin/python3 /home/kuraga/qa_seml/qa_seml/sem_1_lab_3/app.py  
* Serving Flask app 'app'  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on all addresses (0.0.0.0)  
* Running on http://127.0.0.1:5000  
* Running on http://10.0.2.15:5000  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 491-460-590  
127.0.0.1 - - [31/Dec/2022 11:51:08] "POST /createdir HTTP/1.1" 201 -  
127.0.0.1 - - [31/Dec/2022 11:51:15] "POST /createdir HTTP/1.1" 201 -  
127.0.0.1 - - [31/Dec/2022 11:52:14] "POST /createlog HTTP/1.1" 201 -  
127.0.0.1 - - [31/Dec/2022 11:52:48] "GET /getlist HTTP/1.1" 200 -  
127.0.0.1 - - [31/Dec/2022 11:53:53] "DELETE /deletedir HTTP/1.1" 200 -  
127.0.0.1 - - [31/Dec/2022 11:53:57] "GET /getlist HTTP/1.1" 200 -  
127.0.0.1 - - [31/Dec/2022 11:54:54] "POST /createdir HTTP/1.1" 201 -  
127.0.0.1 - - [31/Dec/2022 11:56:01] "PUT /moveNode HTTP/1.1" 200 -  
127.0.0.1 - - [31/Dec/2022 11:56:14] "GET /getlist HTTP/1.1" 200 -  
127.0.0.1 - - [31/Dec/2022 12:02:08] "POST /createbinary HTTP/1.1" 400 -  
127.0.0.1 - - [31/Dec/2022 12:02:59] "POST /createbinary HTTP/1.1" 201 -
```

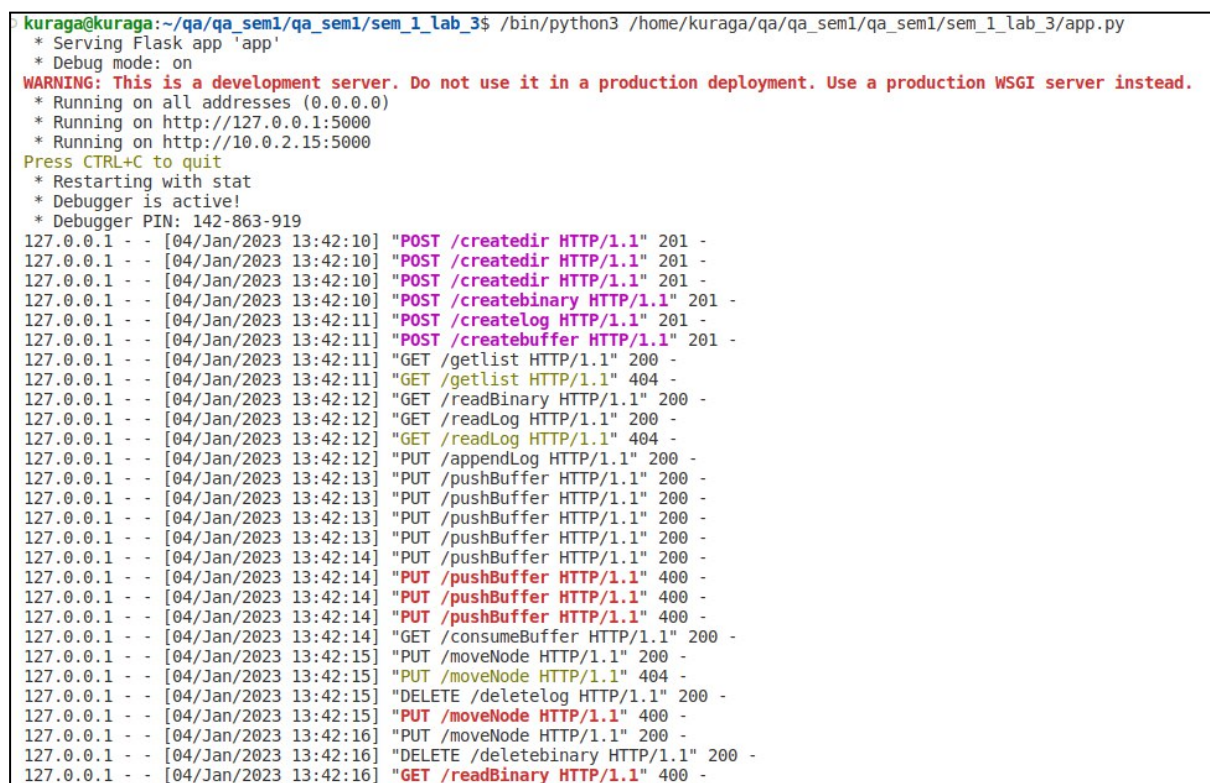

4. Протестуємо створену програму за допомогою pytest:

A terminal window with a dark background and light green text. The title bar shows the user 'kuraga' at host 'kuraga' in the directory '~/qa/qa_sem1/qa_sem1/sem_1_lab_3'. The command 'python3 -m pytest' has been executed. The output shows a test session starting on a Linux platform with Python 3.10.6, pytest 7.2.0, and pluggy 1.0.0. It collected 22 items from 'test_file_system.py' and all 22 tests passed in 6.57 seconds. The terminal window has standard Linux window controls (minimize, maximize, close) and a search icon.

```
kuraga@kuraga: ~/qa/qa_sem1/qa_sem1/sem_1_lab_3
kuraga@kuraga:~/qa/qa_sem1/qa_sem1/sem_1_lab_3$ python3 -m pytest
===== test session starts =====
platform linux -- Python 3.10.6, pytest-7.2.0, pluggy-1.0.0
rootdir: /home/kuraga/qa/qa_sem1/qa_sem1/sem_1_lab_3
collected 22 items

test_file_system.py ..... [100%]

===== 22 passed in 6.57s =====
kuraga@kuraga:~/qa/qa_sem1/qa_sem1/sem_1_lab_3$
```

A terminal window showing the logs of a Flask application. The title bar shows the user 'kuraga' at host 'kuraga' in the directory '~/qa/qa_sem1/qa_sem1/sem_1_lab_3'. The command '/bin/python3 /home/kuraga/qa/qa_sem1/qa_sem1/sem_1_lab_3/app.py' has been executed. The output shows the Flask app starting in debug mode, warning that it is a development server, and running on http://127.0.0.1:5000 and http://10.0.2.15:5000. It then shows a series of HTTP requests and responses, including POST requests for creating directories, binaries, logs, and buffers, and GET requests for reading logs, binaries, and buffers. The terminal window has standard Linux window controls (minimize, maximize, close) and a search icon.

```
kuraga@kuraga:~/qa/qa_sem1/qa_sem1/sem_1_lab_3$ /bin/python3 /home/kuraga/qa/qa_sem1/qa_sem1/sem_1_lab_3/app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.0.2.15:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 142-863-919
127.0.0.1 - - [04/Jan/2023 13:42:10] "POST /createdir HTTP/1.1" 201 -
127.0.0.1 - - [04/Jan/2023 13:42:10] "POST /createdir HTTP/1.1" 201 -
127.0.0.1 - - [04/Jan/2023 13:42:10] "POST /createdir HTTP/1.1" 201 -
127.0.0.1 - - [04/Jan/2023 13:42:10] "POST /createbinary HTTP/1.1" 201 -
127.0.0.1 - - [04/Jan/2023 13:42:11] "POST /createLog HTTP/1.1" 201 -
127.0.0.1 - - [04/Jan/2023 13:42:11] "POST /createbuffer HTTP/1.1" 201 -
127.0.0.1 - - [04/Jan/2023 13:42:11] "GET /getList HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:11] "GET /getList HTTP/1.1" 404 -
127.0.0.1 - - [04/Jan/2023 13:42:12] "GET /readBinary HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:12] "GET /readLog HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:12] "GET /readLog HTTP/1.1" 404 -
127.0.0.1 - - [04/Jan/2023 13:42:12] "PUT /appendLog HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:13] "PUT /pushBuffer HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:13] "PUT /pushBuffer HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:13] "PUT /pushBuffer HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:13] "PUT /pushBuffer HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:14] "PUT /pushBuffer HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:14] "PUT /pushBuffer HTTP/1.1" 400 -
127.0.0.1 - - [04/Jan/2023 13:42:14] "PUT /pushBuffer HTTP/1.1" 400 -
127.0.0.1 - - [04/Jan/2023 13:42:14] "PUT /pushBuffer HTTP/1.1" 400 -
127.0.0.1 - - [04/Jan/2023 13:42:14] "GET /consumeBuffer HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:15] "PUT /moveNode HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:15] "PUT /moveNode HTTP/1.1" 404 -
127.0.0.1 - - [04/Jan/2023 13:42:15] "DELETE /deletelog HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:15] "PUT /moveNode HTTP/1.1" 400 -
127.0.0.1 - - [04/Jan/2023 13:42:16] "PUT /moveNode HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:16] "DELETE /deletebinary HTTP/1.1" 200 -
127.0.0.1 - - [04/Jan/2023 13:42:16] "GET /readBinary HTTP/1.1" 400 -
```

Висновки

Під час виконання курсової роботи, було розроблено HTTP додаток файлової системи та клієнт до програми у вигляді інтерфейсу командного рядка (CLI). Створена програма була запущена в docker контейнері та протестована за допомогою використання pytest.