**Технології серверного програмного забезпечення**

**Лабораторна робота 1**

**Налаштування шаблону проекту**

**Мета роботи:** Налаштувати середовище для подальшого виконання лабораторних робіт з дисципліни, розробити базове REST API

**Завдання:**

* Навчитись налаштуванню середовища, типового для розробки бекенду на python
* Розробити healthcheck ендпоінт веб застосунку
* Налаштування деплою застосунку
* Тестування REST API за допомогою insomnia

**Практичне завдання:**

Практичне завдання полягає в налаштуванні шаблону проекту для веб застосунку. Потрібно створити базовий проект з ендпоінтом який буде вітати користувача. Налаштувати для нього докер контейнер, а також конфігурацію для docker-compose. Після цього задеплоїти цей маленький проект за допомогою render.com.

**Вимоги до виконання:**

* Виконане завдання повинно знаходитись у вигляді репозиторію в системі контролю версій(рекомендовано github, проте при великому бажанні можна і інші).
* Повинні бути структуровані коміти - репозиторій, в якому весь код залитий одним комітом з повідоленням *“Initial commit”* не годиться. Коміти повинні бути осмисленні з відповідними повідомленнями.
* В README файлі повинні міститись повні інструкції для запуску проекту локально.
* Проект повинен бути задеплоєний та мати *GET* ендпоінт */healthcheck* який повинен віддавати код відповіді 200 і в тілі відповіді містити json з поточною датою та статусом сервісу. Детальні інструкції містяться нижче в розділі “Методичні рекомендації”
* Для успішної здачі лабораторної потрібно надати в classroom: посилання на репозиторій та посилання на задеплоєний проект.

**Методичні рекомендації:**

1. Створити проект в git та клонувати його
2. Перейти в папку проекту
3. Встановити python в системі(цей крок опціональний, можете одразу розробляти в докері) - для встановлення та управлінням версіями python рекомендую використовувати утиліту pyenv- <https://github.com/pyenv/pyenv>
4. Створити віртуальне середовище за допомогою venv:  
   *> python3 -m venv env*
5. Активувати віртуальне середовище

*> source ./env/bin/activate*

1. Встановити flask допомогою команди

*> pip install flask*

1. Записати всі залежності проекту в файл *requirements.txt* за допомогою команди

*> pip freeze > requirements.txt*

Цю команду потрібно буде також повторно виконати якщо в процесі виконання роботи ви імпортуєте сторонні модулі

1. В репозиторії створити папку в якій буде міститись модуль проекту
2. Створити файли *\_\_init\_\_.py* та *views.py* в яких буде реалізовано основний код застосунку. Проте, бізнес-логіку застосунку та початкові дані рекомендую винести в інші файли по мірі розробки лабораторної.
3. В файлі *\_\_init\_\_.py* потрібно створити змінну app та імпортувати файли які ви будете використовувати:

from flask import Flask

app = Flask(\_\_name\_\_)

import <your app module name>.views

1. У файлі views.py реалізуйте ендпоінт healthcheck. Як реалізувати простий ендпоінт в flask можна почитати тут - <https://flask.palletsprojects.com/en/2.3.x/quickstart/#a-minimal-application>
2. Тепер після реалізації ендпоінту у *views.py* ви зможете запустити застосунок командою:

> flask run –host 0.0.0.0 -p <your port>

1. Після того, як ви впевнились що з базовим ендоінтом застосунок запускається, переходьте до налаштування Docker
2. Встановіть docker з сайту <https://www.docker.com/>
3. Далі потрібно створити *Dockerfile* в папці репозиторію. Він має бути такого вигляду:

*FROM python:3.11.3-slim-bullseye(or other version you use)*

*WORKDIR /app*

*COPY requirements.txt .*

*RUN python -m pip install -r requirements.txt*

*COPY . /app*

*CMD flask --app <your app name> run -h 0.0.0.0 -p $PORT*

1. Далі потрібно збілдити image також командою:

> docker build . -t <image\_name>:latest

1. Якщо image упішно збілдився потрібно його запустити та перевірити чи працює застосунок

> docker run -it --rm --network=host -e PORT=<your\_port> <image\_name>:latest

1. Переходьте до створення *docker-compose.yaml* файлу для зручності запуску контейнерів. Він має бути такого виду:

version: '3'

services:

<app\_name>:

restart: always

build:

context: .

dockerfile: Dockerfile

environment:

PORT: "<your\_port>"

ports:

- "<your\_port>:8080"

1. Після цього встановіть docker-compose та спробуйте збілдити та запустити контейнер за допомогою команд:

> docker-compose build

> docker-compose up

1. Далі потрібно задеплоїти застосунок. В методичних рекомендаціях описано деплой на render.com, він безкоштовний для найпростіших інстансів. Проте, якщо ви маєте доступ до деплою на інших провайдерах - то можете використовувати їх, проте подальші етапи будуть відрізнятись для вас.
2. Зареєструйтесь на render.com
3. Створіть новий сервіс типу Web Service, в меню підключіть свій репозиторій з проектом, назвіть проект(бажано тим же ім’ям що й репозиторій), виберіть бажаний регіон, гілку з якох буде деплоїтись застосунок. Натисніть створити.
4. Після цього, якщо ви все зробили правильно, запуститься процес деплою і через деякий час ваш застосунок буде доступний за адресою вказаною на сторінці дашборду в render.com. Ви можете перейти на сторінку сервісу та вручну перевірити його роботу.

**Критерії оцінювання:**

Всього можна отримати 20 балів за лабораторну

Розподіл балів за лабораторну 1:

5 балів - healthcheck працює коректно

8 балів - налаштований Docker та docker-compose

5 балів - застосунок коректно задеплоєний

2 бали - коректна робота з git: повідомлення комітів, адекватна кількість комітів, наявний README.md з інструкціями по запуску

Також на розсуд перевіряючого може бути доставлено 1-2 бали за якісь цікаві рішення.

**Корисні посилання**

1. <https://docs.docker.com/>
2. <https://docs.docker.com/compose/compose-file/compose-file-v3/#compose-file-structure-and-examples>
3. <https://docs.github.com/en/actions/security-guides/encrypted-secrets>
4. <https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#general-guidelines-and-recommendations>