|  |  |
| --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет» | |
|  | Факультет информационных технологий  Кафедра технологий программирования |
| Лабораторная работа №5 по курсу «Разработка и анализ требований»  «Реализация Microservices архитектуры»  Вариант 1 | |
| Выполнил | Студент гр. 21-ИТ-1  Чиникайло А.П. |
| Проверил | Преподаватель  Васильева Д.М. |
| Полоцк, 2023г. | |

**Цель работы:** ознакомление с основами Microservices архитектуры, её

реализацией, и инструментами для работы с микросервисами.

**Ход работы**

**Задание:**

1. Установите Docker и Docker Compose на свой компьютер. Затем

создайте Docker Compose файл для запуска нескольких контейнеров, включая

базу данных (например, PostgreSQL) и веб-сервер (например, Nginx).

Предоставьте файлы конфигурации для каждого контейнера.

2. Расширьте структуру файлов вашего микросервиса, чтобы она

включала в себя несколько компонентов, например, модуль аутентификации,

работу с данными и веб-интерфейс. Обеспечьте взаимодействие между

компонентами.

3. Создайте Dockerfile для вашего микросервиса, упакуйте его в Docker-

контейнер и опубликуйте образ в вашем репозитории на Docker Hub.

4. Реализуйте простой RESTful API для ваших микросервисов.

Используйте Flask (Python) или любой другой фреймворк на ваш выбор.

5. Разверните свои микросервисы в кластере Kubernetes. Создайте

манифесты для развертывания, обеспечьте доступность сервисов через службы

Kubernetes (по варианту).

**Вариант 1**

Название сервиса catalog-service

Версия образа v1

Реализация сервиса inventory-service представлена в листинге 1.

Листинг 1 – inventory-service

from flask import Flask, request, jsonify

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

inventory = [

{'element': "el"}

]

@app.route('/inventory')

def get\_elements():

return jsonify(inventory)

@app.route('/inventory/add', methods=['POST'])

def create\_element():

data = request.get\_json()

element = data.get('element')

inventory\_element = {'element': element}

inventory.append(inventory\_element)

return jsonify({'message': 'Element created successfully'}), 201

@app.route('/inventory/<int:element\_id>', methods=['GET'])

def get\_element(element\_id):

if element\_id >= len(inventory):

return jsonify({'error': 'Element not found'}), 404

return jsonify(inventory[element\_id])

@app.route('/inventory/<int:element\_id>', methods=['PUT'])

def update\_element(element\_id):

if element\_id >= len(inventory):

return jsonify({'error': 'Element not found'}), 404

data = request.get\_json()

inventory[element\_id]['element'] = data.get('element')

return jsonify({'message': 'Element updated successfully'})

@app.route('/inventory/<int:element\_id>', methods=['DELETE'])

def delete\_element(element\_id):

if element\_id >= len(inventory):

return jsonify({'error': 'Element not found'}), 404

del inventory[element\_id]

return jsonify({'message': 'Element deleted successfully'})

@app.route('/inventory/len')

def inventory\_len():

return jsonify({'message': f'Inventory countains, {len(inventory)}, elements'})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5001)

Необходимо поместить микросервис в докер. Инструкции к созданию докера указаны в листинге 2.

Листинг 2 – Инструкции к созданию докера

FROM python:3.9

COPY requirements.txt ./

RUN pip install --no-cache-dir --upgrade pip \

&& pip install --no-cache-dir -r requirements.txt

WORKDIR /5

COPY . /5

EXPOSE 5001

CMD ["python", "app.py"]

Также к в инструкциях к докеру присутствует файл requirements.txt с необходимыми для работы библиотеками python. Содержимое requirements.txt представлено в листинге 3.

Листинг 3 – requirements.txt

flask

**Вывод:** В ходе данный лабораторной работе был ознакомлен с основами Microservices архитектуры, её реализацией, и инструментами для работы с микросервисами.