Министерство науки и высшего образования РФ
ФГАОУ ВПО
Национальный исследовательский технологический университет «МИСИС»

Институт компьютерных наук (ИКН)

Кафедра Инфокоммуникационных технологий (ИКТ)

Отчет по лабораторной работе №7
по дисциплине «Разработка сетевых приложений
на языке программирования Python»
на тему «Создание простого REST API на Flask»

Выполнил:
студент группы БИСТ-22-1

Медведев Д. Р.

Проверил:
доц. каф. ИКТ

Стучилин В.В.

Москва, 2025

Цель работы: приобрести навыки создания веб-приложений на Python с использованием микрофреймворка Flask.

Научиться:

обрабатывать HTTP-запросы;

формировать ответы в формате JSON;

организовывать структуру проекта;

понимать разницу между хранением данных в памяти и в базах данных

Задание: Разработать простое REST API для управления списком задач. Задачи должны храниться в

оперативной памяти и иметь следующую структуру:

"id": 1,

"title": "Сделать лабораторную работу"

Требования к API:

1. Реализовать маршруты:

GET /tasks — вернуть список всех задач

POST /tasks — добавить новую задачу (тело запроса:

GET /tasks/<id> — получить задачу по ID

DELETE /tasks/<id> — удалить задачу

2. Реализовать автоматическую генерацию

3. Обрабатывать ошибки:

404 Not Found — если задача не найдена по ID

400 Bad Request — если

title не передан или пустой

{ "title": "..." })

id (инкрементный счётчик).

4. Организовать структуру проекта по папкам и модулям (

app.py ,

views.py ,

data.py).

5. Протестировать работу API с помощью Postman или curl:

curl -X POST -H "Content-Type: application/json" -d '{"title": "Buy milk"}'

http://localhost:5000/tasks

Решение

Структура проекта:

App.py (Главный файл приложения)

Data.py (Работа с данными)

Views.py (HTTP-обработчики)

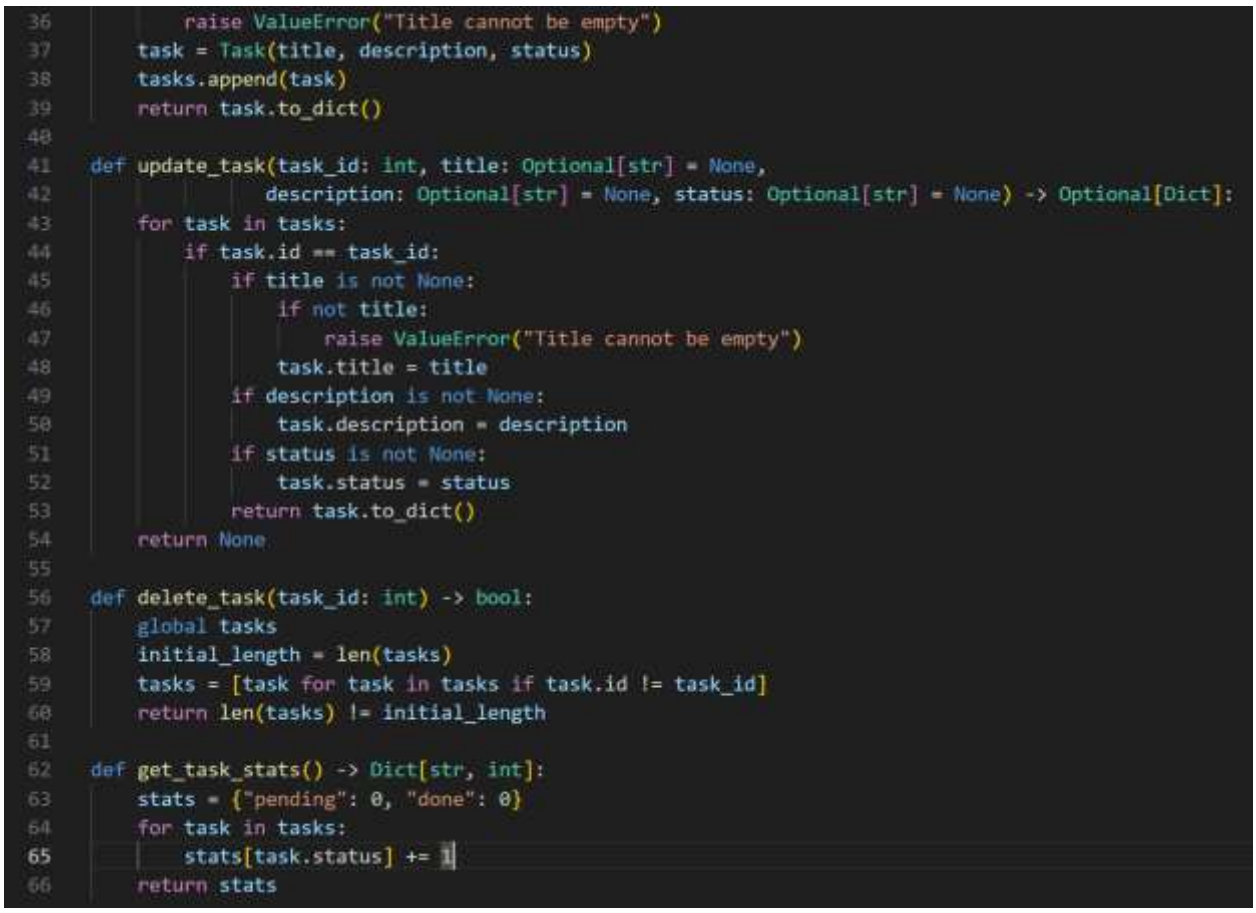
Test_api.py (Тесты)

App.py

```python
from flask import Flask, jsonify  # Добавляем импорт jsonify
from views import tasks_blueprint

app = Flask(__name__)

# Регистрируем blueprint
app.register_blueprint(tasks_blueprint)

@app.errorhandler(404)
def not_found(error):
    return jsonify({"error": "Not found"}), 404

if __name__ == '__main__':
    app.run(debug=True)
```

Data.py

```python
from typing import Dict, List, Optional

tasks = []
current_id = 1

class Task:
    def __init__(self, title: str, description: Optional[str] = None, status: str = "pending"):
        global current_id
        self.id = current_id
        self.title = title
        self.description = description
        self.status = status
        current_id += 1

    def to_dict(self) -> Dict:
        return {
            "id": self.id,
            "title": self.title,
            "description": self.description,
            "status": self.status
        }

def get_all_tasks(status_filter: Optional[str] = None) -> List[Dict]:
    if status_filter:
        return [task.to_dict() for task in tasks if task.status == status_filter]
    return [task.to_dict() for task in tasks]

def get_task_by_id(task_id: int) -> Optional[Dict]:
    for task in tasks:
        if task.id == task_id:
            return task.to_dict()
    return None

def add_task(title: str, description: Optional[str] = None, status: str = "pending") -> Dict:
    if not title:
```



```
        raise ValueError("Title cannot be empty")
    task = Task(title, description, status)
    tasks.append(task)
    return task.to_dict()

def update_task(task_id: int, title: Optional[str] = None,
                description: Optional[str] = None, status: Optional[str] = None) -> Optional[Dict]:
    for task in tasks:
        if task.id == task_id:
            if title is not None:
                if not title:
                    raise ValueError("Title cannot be empty")
                task.title = title
            if description is not None:
                task.description = description
            if status is not None:
                task.status = status
            return task.to_dict()
    return None

def delete_task(task_id: int) -> bool:
    global tasks
    initial_length = len(tasks)
    tasks = [task for task in tasks if task.id != task_id]
    return len(tasks) != initial_length

def get_task_stats() -> Dict[str, int]:
    stats = {"pending": 0, "done": 0}
    for task in tasks:
        stats[task.status] += 1
    return stats
```

Views.py

```python
from flask import Blueprint, request, jsonify
from data import (
    get_all_tasks, get_task_by_id, add_task,
    update_task, delete_task, get_task_stats
)

tasks_blueprint = Blueprint('tasks', __name__)

@tasks_blueprint.route('/tasks', methods=['GET'])
def list_tasks():
    status_filter = request.args.get('status')
    tasks = get_all_tasks(status_filter)
    return jsonify(tasks)

@tasks_blueprint.route('/tasks', methods=['POST'])
def create_task():
    data = request.get_json()
    if not data or 'title' not in data:
        return jsonify({"error": "Title is required"}), 400

    try:
        task = add_task(
            title=data['title'],
            description=data.get('description'),
            status=data.get('status', 'pending')
        )
        return jsonify(task), 201
    except ValueError as e:
        return jsonify({"error": str(e)}), 400

@tasks_blueprint.route('/tasks/<int:task_id>', methods=['GET'])
def get_task(task_id):
    task = get_task_by_id(task_id)
    if not task:
        return jsonify({"error": "Task not found"}), 404
```

```python
    return jsonify(task)

@tasks_blueprint.route('/tasks/<int:task_id>', methods=['PUT'])
def modify_task(task_id):
    data = request.get_json()
    if not data:
        return jsonify({"error": "No data provided"}), 400

    try:
        updated_task = update_task(
            task_id=task_id,
            title=data.get('title'),
            description=data.get('description'),
            status=data.get('status')
        )
        if not updated_task:
            return jsonify({"error": "Task not found"}), 404
        return jsonify(updated_task)
    except ValueError as e:
        return jsonify({"error": str(e)}), 400

@tasks_blueprint.route('/tasks/<int:task_id>', methods=['DELETE'])
def remove_task(task_id):
    if delete_task(task_id):
        return jsonify({"message": "Task deleted"}), 200
    return jsonify({"error": "Task not found"}), 404

@tasks_blueprint.route('/tasks/stats', methods=['GET'])
def task_statistics():
    stats = get_task_stats()
    return jsonify(stats)
```

Test_api.py

```python
import requests

BASE_URL = "http://localhost:5000/tasks"

def print_response(response):
    """Печатает информацию о запросе и ответе"""
    print(f"URL: {response.url}")
    print(f"Status: {response.status_code}")
    print("Response:", response.json())
    print("-" * 50)

def test_api():
    # 1. Создаем несколько задач
    print("1. Создаем задачи:")
    tasks = [
        {"title": "Купить молоко", "status": "pending"},
        {"title": "Сделать ДЗ", "description": "Математика", "status": "pending"},
        {"title": "Позвонить маме", "status": "done"}
    ]

    created_ids = []
    for task in tasks:
        resp = requests.post(BASE_URL, json=task)
        created_ids.append(resp.json()["id"])
        print_response(resp)

    # 2. Получаем все задачи
    print("\n2. Получаем все задачи:")
    resp = requests.get(BASE_URL)
    print_response(resp)

    # 3. Тестируем фильтрацию по статусу
    print("\n3. Фильтруем задачи по статусу 'done':")
    resp = requests.get(BASE_URL, params={"status": "done"})
    print_response(resp)
```

```python

    # 4. Получаем статистику
    print("\n4. Получаем статистику по задачам:")
    resp = requests.get(f"{BASE_URL}/stats")
    print_response(resp)

    # 5. Обновляем задачу
    print("\n5. Обновляем первую задачу:")
    update_data = {
        "title": "Купить органическое молоко",
        "description": "В магазине на углу",
        "status": "done"
    }
    resp = requests.put(f"{BASE_URL}/{created_ids[0]}", json=update_data)
    print_response(resp)

    # 6. Проверяем обновление
    print("\n6. Проверяем обновленную задачу:")
    resp = requests.get(f"{BASE_URL}/{created_ids[0]}")
    print_response(resp)

    # 7. Удаляем задачу
    print("\n7. Удаляем вторую задачу:")
    resp = requests.delete(f"{BASE_URL}/{created_ids[1]}")
    print_response(resp)

    # 8. Проверяем список после удаления
    print("\n8. Проверяем список задач после удаления:")
    resp = requests.get(BASE_URL)
    print_response(resp)

    # 9. Проверяем обработку ошибок
    print("\n9. Тестируем обработку ошибок:")

    # Несуществующий ID
    print("a) Запрос несуществующей задачи:")
    resp = requests.get(f"{BASE_URL}/9999")
    print(f"Status: {resp.status_code}")
    print("Response:", resp.json())

    # Некорректный статус
    print("\nb) Некорректный статус:")
    resp = requests.put(f"{BASE_URL}/{created_ids[0]}", json={"status": "invalid"})
    print(f"Status: {resp.status_code}")
    print("Response:", resp.json())

    # Пустой title
    print("\nc) Пустой title:")
    resp = requests.post(BASE_URL, json={"title": ""})
    print(f"Status: {resp.status_code}")
    print("Response:", resp.json())

if __name__ == "__main__":
    test_api()
```

Результат

Добавление задач

```
C:\Users\dmrme>curl -X POST -H "Content-Type: application/json" -d "{\"title\":\"New task 1
\",\"description\":\"Description\",\"status\":\"pending\"}" http://localhost:5000/tasks
{
  "description": "Description",
  "id": 1,
  "status": "pending",
  "title": "New task 1"
}

C:\Users\dmrme>curl -X POST -H "Content-Type: application/json" -d "{\"title\":\"New task 2
\",\"description\":\"Description 2\",\"status\":\"pending\"}" http://localhost:5000/tasks
{
  "description": "Description 2",
  "id": 2,
  "status": "pending",
  "title": "New task 2"
}
```

Получение всех задач

```
C:\Users\dmrme>curl http://localhost:5000/tasks
[
  {
    "description": "Description",
    "id": 1,
    "status": "pending",
    "title": "New task 1"
  },
  {
    "description": "Description 2",
    "id": 2,
    "status": "pending",
    "title": "New task 2"
  }
]
```

Получение по id

```
C:\Users\dmrme>curl http://localhost:5000/tasks/1
{
  "description": "Description",
  "id": 1,
  "status": "pending",
  "title": "New task 1"
}
```

Удаление

```
C:\Users\dmrme>curl -X DELETE http://localhost:5000/tasks/1
{
  "message": "Task deleted"
}

C:\Users\dmrme>curl http://localhost:5000/tasks
[
  {
    "description": "Description 2",
    "id": 2,
    "status": "pending",
    "title": "New task 2"
  }
]
```

Дополнительные задания:

Изменение задачи по id

```
C:\Users\dmrme>curl -X PUT -H "Content-Type: application/json" -d "{\"title\":\"Changed titl
e\",\"description\":\"Changed desciption\",\"status\":\"done\"}" http://localhost:5000/tasks
/2
{
  "description": "Changed desciption",
  "id": 2,
  "status": "done",
  "title": "Changed title"
}

C:\Users\dmrme>curl http://localhost:5000/tasks
[
  {
    "description": "Changed desciption",
    "id": 2,
    "status": "done",
    "title": "Changed title"
  }
]

C:\Users\dmrme>
```

Фильтрация по статусу

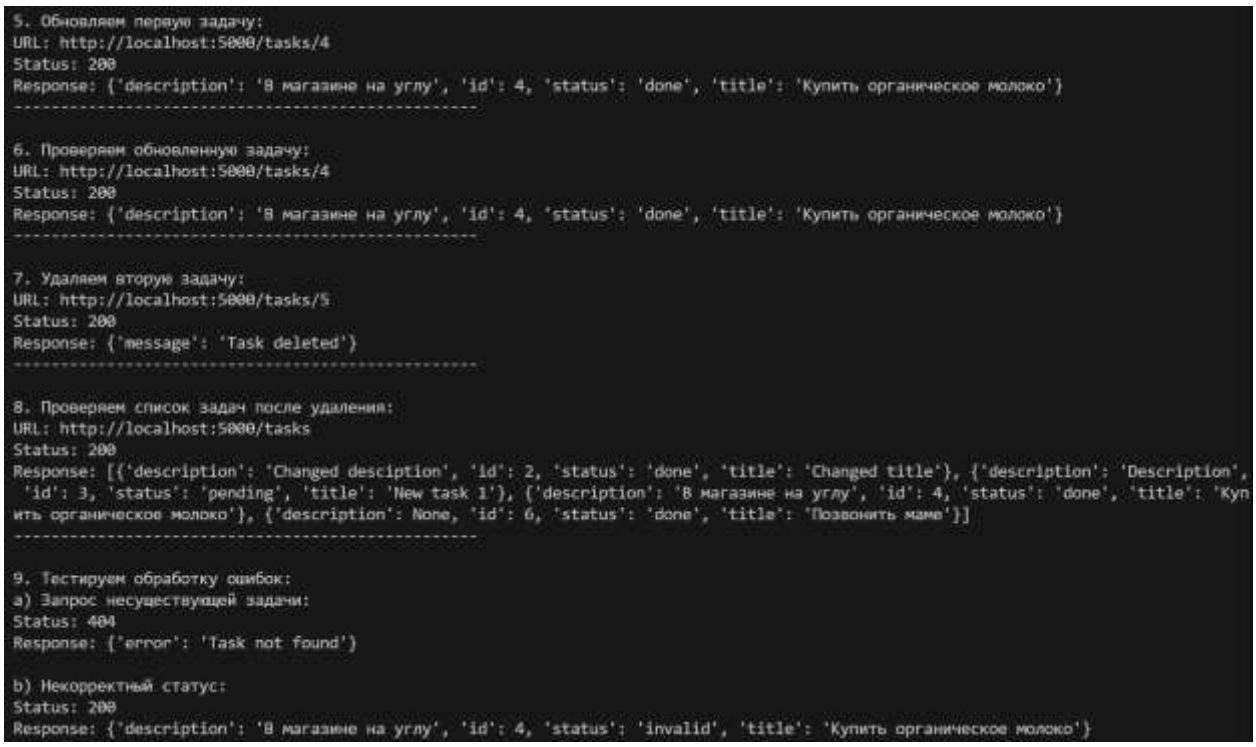
```
C:\Users\dmrme>curl http://localhost:5000/tasks?status=done
[
  {
    "description": "Changed desciption",
    "id": 2,
    "status": "done",
    "title": "Changed title"
  }
]
```

Статистика по статусу

```
C:\Users\dmrme>curl http://localhost:5000/tasks/stats
{
  "done": 1,
  "pending": 1
}
```

Python скрипт тестирования:

```
PS C:\Users\dmrme\Desktop\MISIS\Git\misis\third_course\sixth_semestr\network_applications_python\lab7> python test_api.py
1. Создаем задачи:
URL: http://localhost:5000/tasks
Status: 201
Response: {'description': None, 'id': 4, 'status': 'pending', 'title': 'Купить молоко'}
--------------------------------------------------
URL: http://localhost:5000/tasks
Status: 201
Response: {'description': 'Математика', 'id': 5, 'status': 'pending', 'title': 'Сделать ДЗ'}
--------------------------------------------------
URL: http://localhost:5000/tasks
Status: 201
Response: {'description': None, 'id': 6, 'status': 'done', 'title': 'Позвонить маме'}
--------------------------------------------------

2. Получаем все задачи:
URL: http://localhost:5000/tasks
Status: 200
Response: [{'description': 'Changed desciption', 'id': 2, 'status': 'done', 'title': 'Changed title'}, {'description': 'Description',
 'id': 3, 'status': 'pending', 'title': 'New task 1'}, {'description': None, 'id': 4, 'status': 'pending', 'title': 'Купить молоко'},
 {'description': 'Математика', 'id': 5, 'status': 'pending', 'title': 'Сделать ДЗ'}, {'description': None, 'id': 6, 'status': 'done',
 'title': 'Позвонить маме'}]
--------------------------------------------------

3. Фильтруем задачи по статусу 'done':
URL: http://localhost:5000/tasks?status=done
Status: 200
Response: [{'description': 'Changed desciption', 'id': 2, 'status': 'done', 'title': 'Changed title'}, {'description': None, 'id': 6,
 'status': 'done', 'title': 'Позвонить маме'}]
--------------------------------------------------

4. Получаем статистику по задачам:
URL: http://localhost:5000/tasks/stats
Status: 200
Response: {'done': 2, 'pending': 3}
--------------------------------------------------
```



Вывод: я приобрел навыки создания веб-приложений на Python с использованием микрофреймворка Flask.