Лабораторная работа 6

Татаринова А.Г.

19 ноября 2025 г.

Цель: получить навыки написания веб-сервиса на Fast-API.

- Лабораторная работа сдается преподавателю лично студентом.
- Сдача лабораторной работы заключается в том, что студент отвечает на вопросы преподавателя по заданиям, а также показывает и комментирует результаты самостоятельно выполненных заданий.
- Всё что написано в заданиях «изучите, узнайте и прочее» нужно изучить и для себя зафиксировать, так как преподаватель может спросить об этом при сдаче лабораторной работы.

(!) В качестве дополнительного руководства можно использовать уроки курса «Python и FastAPI» с ресурса Metanit.

1 Задание

Изучите материалы 1, 2, 3 и 4 глав книги «Создание веб-API Python с помощью FastAPI». Реализуйте пример приложения ToDo по материалам книги только этих глав (Часть 1).

2 Задание

Запустите скрипт app.py в каталоге server проекта **web-lab-6-proj-test**. Через UI Swagger вызовите документацию для api запущенного сервиса через localhost:8085 (надо добавить к адресу «docs»). Попробуйте вызвать каждый из маршрутов сервиса.

Также откройте веб-страницу index.html в каталоге client и проверьте работу соответствующего маршрута.

3 Задание

Напишите веб-сервис, выполняющий поиск научных статей ресурса arxiv.org по их заголовкам.

> Для извлечения данных со страницы используйте библиотеку BeautifulSoup4 (bs4). Ниже показана краткая справка по извлечению данных со страницы.

Для получения списка статей из области компьютерных наук выполняйте запросы, используя библиотеку requests (см. первую лабораторную работу). В своих запросах используйте параметр, хранящий запросные слова. Чтобы понять какой параметр использовать, обратите внимание на параметры адресной строки при поиске на странице https://arxiv.org/search/cs.

Приложение должно выполнять следующие функции:

a) получение списка статей из области компьютерных наук (https://arxiv.org/search/cs), в заголовках (title) которых встречаются заданные пользователем слова (параметры в маршруте),

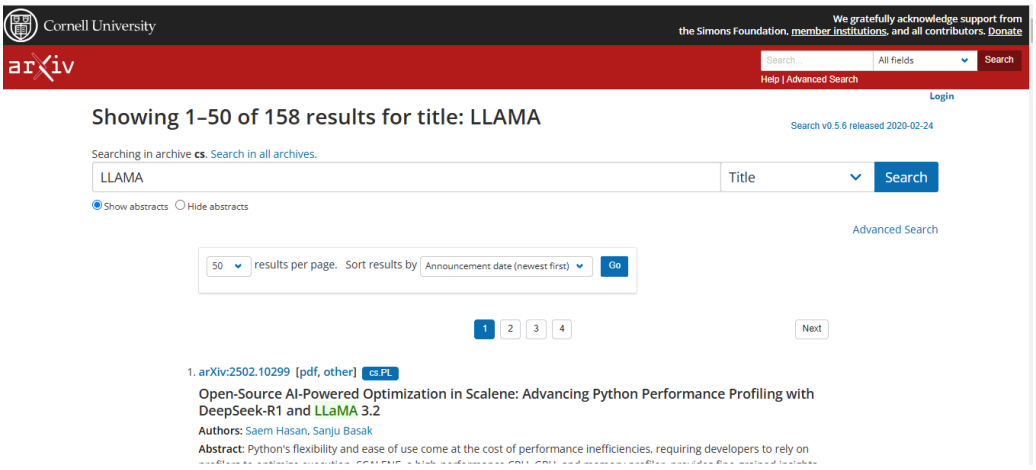
b) получение названия, авторов и аннотации статьи по её ID на сайте arxiv.org (уточнение смотри ниже на поясняющих скриншотах),

c) перевод аннотации выбранной статьи. Пример перевода текста можно посмотреть в коде для второго задания, в котором используется библиотека translate на основе Microsoft Translator (устанавливается как **pip install translate**). Учтите ограничение на длину переводного текста, для этого можно текст аннотации делить на несколько предложений и для них вызывать функцию перевода.

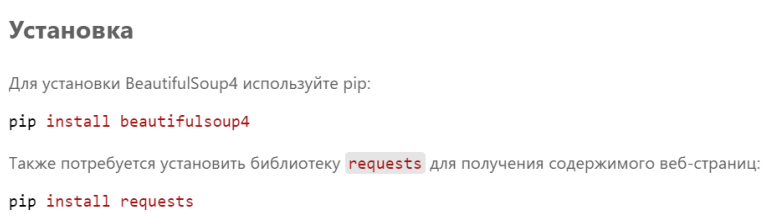
Протестируйте работу веб-сервиса через UI Swagger.

URL страницы с аннотацией статьи составляется по правилу:
arxiv.org/abs/ID статьи.
Например: https://arxiv.org/abs/2502.10299

ID номером статьи является набор цифр, следующий после **arXiv:** в строке перед названием научной статьи:



Краткая справка по использованию библиотеки BeautifulSoup4 (bs4):



Основные шаги работы с BeautifulSoup

1. **Импортирование библиотек:**

```
from bs4 import BeautifulSoup
import requests
```

2. **Получение HTML-кода страницы:** Используйте библиотеку `requests` для загрузки содержимого веб-страницы:

```
url = "https://example.com"
response = requests.get(url)
html_content = response.text  # Получаем HTML-код страницы
```

3. **Создание объекта BeautifulSoup:** Создайте объект BeautifulSoup для анализа HTML:

```
soup = BeautifulSoup(html_content, 'html.parser')  # 'html.parser' — встроенный парсер Python
```

4. **Поиск элементов:** BeautifulSoup предоставляет методы для поиска элементов в HTML-документе.

Основные методы BeautifulSoup

1. Поиск одного элемента

- `.find()`: Находит первый элемент, соответствующий заданному тегу или атрибутам.

```
title = soup.find('title')  # Находит первый тег <title>
print(title.text)  # Выводит текст внутри тега
```

2. Поиск всех элементов

- `.find_all()`: Находит все элементы, соответствующие заданному тегу или атрибутам.

```
links = soup.find_all('a')  # Находит все ссылки (<a>)
for link in links:
    print(link.get('href'))  # Выводит значения атрибута href
```

3. Поиск по CSS-селекторам

- `.select()`: Использует CSS-селекторы для поиска элементов.

```
items = soup.select('.item')  # Находит все элементы с классом 'item'
for item in items:
    print(item.text)
```

4. Извлечение атрибутов

- Используйте `.get()` для доступа к атрибутам элемента.

```
img_tag = soup.find('img')
src = img_tag.get('src')  # Получает значение атрибута src
```

5. Извлечение текста

- `.text`: Возвращает текстовое содержимое элемента.

```
paragraph = soup.find('p').text  # Текст внутри первого <p>
```

6. Работа с вложенными элементами

- Можно обращаться к дочерним элементам через точечную нотацию.

```
header = soup.body.h1  # Находит первый <h1> внутри <body>
print(header.text)
```