Интеллектуальные информационные системы

Практическое задание 7

Кафедра информатики
Институт кибербезопасности и цифровых технологий
РТУ МИРЭА

Добавление пользовательского интерфейса к API сервису «Библиотека»

В этой практической работе добавим пользовательский интерфейс (Frontend часть) для взаимодействия с API, причем сам Frontend будет находиться в отдельном процессе, фактически это будет отдельный python веб-сервис. Мы не будем использовать классические способы создания Frontend'а, а воспользуемся Python фреймворком для быстрого создания пользовательских интерфейсов - **Streamlit**.

Общая информация о Stremlit

Сайт: https://streamlit.io/

Streamlit — это фреймворк Python с открытым исходным кодом для специалистов по данным и инженеров AI/ML, позволяющий создавать динамические приложения для работы с данными всего за несколько строчек кода.

Особенности Streamlit:

- Основное предназначение — создание прототипов, интерактивных дашбордов и инструментов анализа данных.
- Ориентирован на разработчиков, аналитиков и data scientists.
- Позволяет быстро превратить Python-код в интерактивное приложение без необходимости знаний HTML, CSS или JavaScript.

После установки Streamlit через pip. Можно запускать скрипты Python содержащие код для Frontend'а (например если код содержится в файле myui.py): командой **streamlit run myui.py**, что аналогично запуску команды: **python -m streamlit run myui.py**

Как только вы запустите скрипт, как показано выше, локальный сервер Streamlit запустится, и ваше приложение откроется в новой вкладке в вашем веб-браузере по умолчанию. Приложение — это ваш холст, на котором вы будете рисовать диаграммы, текст, виджеты, таблицы и многое другое.

Добавление пользовательского интерфейса к API сервису «Библиотека»

Процесс разработки интерфейса на Streamlit

Каждый раз, когда вы хотите обновить свое приложение, сохраняйте исходный файл. Когда вы это делаете, Streamlit определяет, есть ли изменения, и спрашивает, хотите ли вы перезапустить свое приложение. Выберите «Всегда перезапускать» в правом верхнем углу экрана, чтобы автоматически обновлять свое приложение каждый раз, когда вы изменяете его исходный код.

Это позволяет вам работать в быстром интерактивном цикле: вы вводите код, сохраняете его, пробуете его вживую, затем пишите еще код, сохраняете его, пробуете его и так далее, пока не будете удовлетворены результатами.

Поток данных в Stremlit

Архитектура Streamlit позволяет вам писать приложения так же, как вы пишете простые скрипты Python. Чтобы так было возможно, приложения Streamlit имеют уникальный поток данных: каждый раз, когда что-то должно быть обновлено на экране, Streamlit перезапускает весь ваш скрипт Python сверху донизу.

Это может произойти в двух ситуациях:

- Всякий раз, когда вы изменяете исходный код своего приложения.
- Всякий раз, когда пользователь взаимодействует с виджетами в приложении. Например, при перетаскивании ползунка, вводе текста в поле ввода или нажатии кнопки.

Более детальную информацию вы найдете в документации к фреймворку:
https://docs.streamlit.io/

Добавление пользовательского интерфейса к API сервису «Библиотека»

Этапы работы:

1. Создайте новую отдельную папку (в не папки прошлого проекта) можете назвать ее front

2. Зайдите в консоль в эту папку и создайте виртуальное окружения python

3. Для простоты этого проекта мы не будет использовать загрузку библиотек из requirements.txt (хотя это надо будет делать если вы будете развивать проект frontend’a и добавлять туда новые библиотеки). Поэтому просто добавим библиотеку Streamlit командой **pip install streamlit**

4. Откройте VSCode в этой папке и создайте файл ui.py

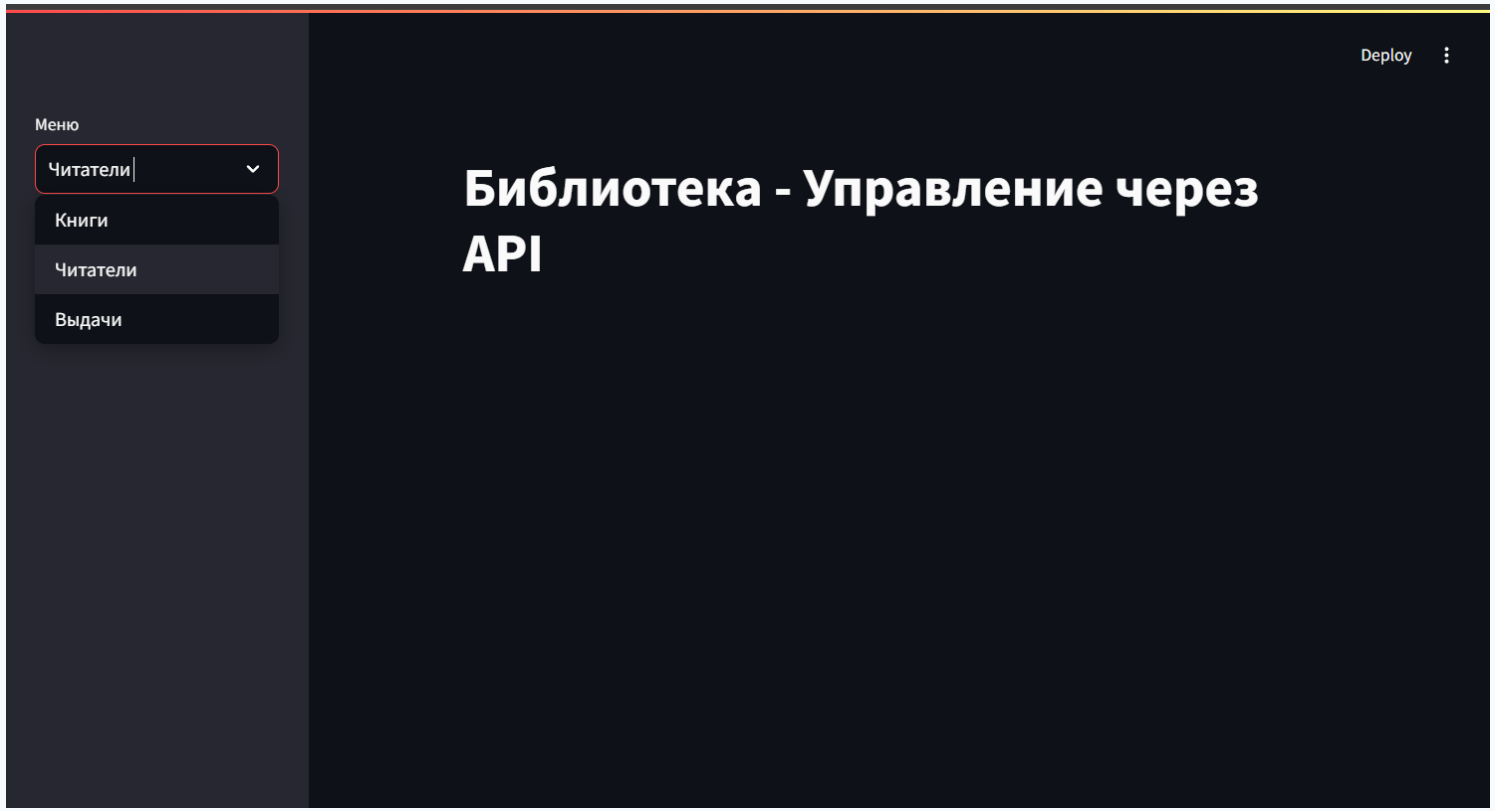
5. Добавьте следующий код:

```python
import streamlit as st
import requests

API_BASE_URL = "http://127.0.0.1:8000"  # Базовый URL вашего API

# Главная функция приложения
def main():
    st.title("Библиотека - Управление через API")

    # Меню
    menu = ["Книги", "Читатели", "Выдачи"]
    choice = st.sidebar.selectbox("Меню", menu)

    if choice == "Книги":
        books_ui()
    elif choice == "Читатели":
        readers_ui()
    elif choice == "Выдачи":
        issues_ui()

def  books_ui():
    pass

def  readers_ui():
    pass

def  issues_ui():
    pass

# Запуск приложения
if __name__ == "__main__":
    main()
```

Добавление пользовательского интерфейса к API сервису «Библиотека»

6. Запустите скрипт из консоли командой **streamlit run ui.py**

7. У вас откроется браузер где будет отображен сгенерированный интерфейс



8. Давайте создадим недостающий код функции для Читателей, Книг и Выдачи.

а. Напишем код функции books_ui:

```
# Интерфейс для работы с книгами
def books_ui():
    st.header("Управление книгами")

    # Добавление книги
    st.subheader("Добавить книгу")
    with st.form("add_book_form"):
        title = st.text_input("Название книги")
        author = st.text_input("Автор книги")
        submitted = st.form_submit_button("Добавить")
        if submitted:
            response = requests.post(f"{API_BASE_URL}/book/",
json={"title": title, "author": author})
            if response.status_code == 200:
                st.success("Книга успешно добавлена!")
            else:
                st.error(f"Ошибка: {response.json().get('detail'
)}")


    # Просмотр списка книг
    st.subheader("Список книг")
    response = requests.get(f"{API_BASE_URL}/book/all")
    if response.status_code == 200:
        books = response.json()
        for book in books:
            st.write(f"ID: {book['id']}, Название: {book['title'
]}, Автор: {book['author']}")
    else:
        st.error("Ошибка при получении списка книг.")
```

b. Напишем код функции readers_ui:

```
# Интерфейс для работы с читателями
def readers_ui():
    st.header("Управление читателями")

    # Добавление читателя
    st.subheader("Добавить читателя")
    with st.form("add_reader_form"):
        name = st.text_input("Имя читателя")
        submitted = st.form_submit_button("Добавить")
        if submitted:
            response = requests.post(f"{API_BASE_URL}/reader/",
json={"name": name})
            if response.status_code == 200:
                st.success("Читатель успешно добавлен!")
            else:
                st.error(f"Ошибка: {response.json().get('detail'
)}")

    # Просмотр списка читателей
    st.subheader("Список читателей")
    response = requests.get(f"{API_BASE_URL}/reader/all")
    if response.status_code == 200:
        readers = response.json()
        for reader in readers:
            st.write(f"ID: {reader['id']}, Имя: {reader['name']}
")
    else:
        st.error("Ошибка при получении списка читателей.")
```

с. Напишем код функции issues_ui:

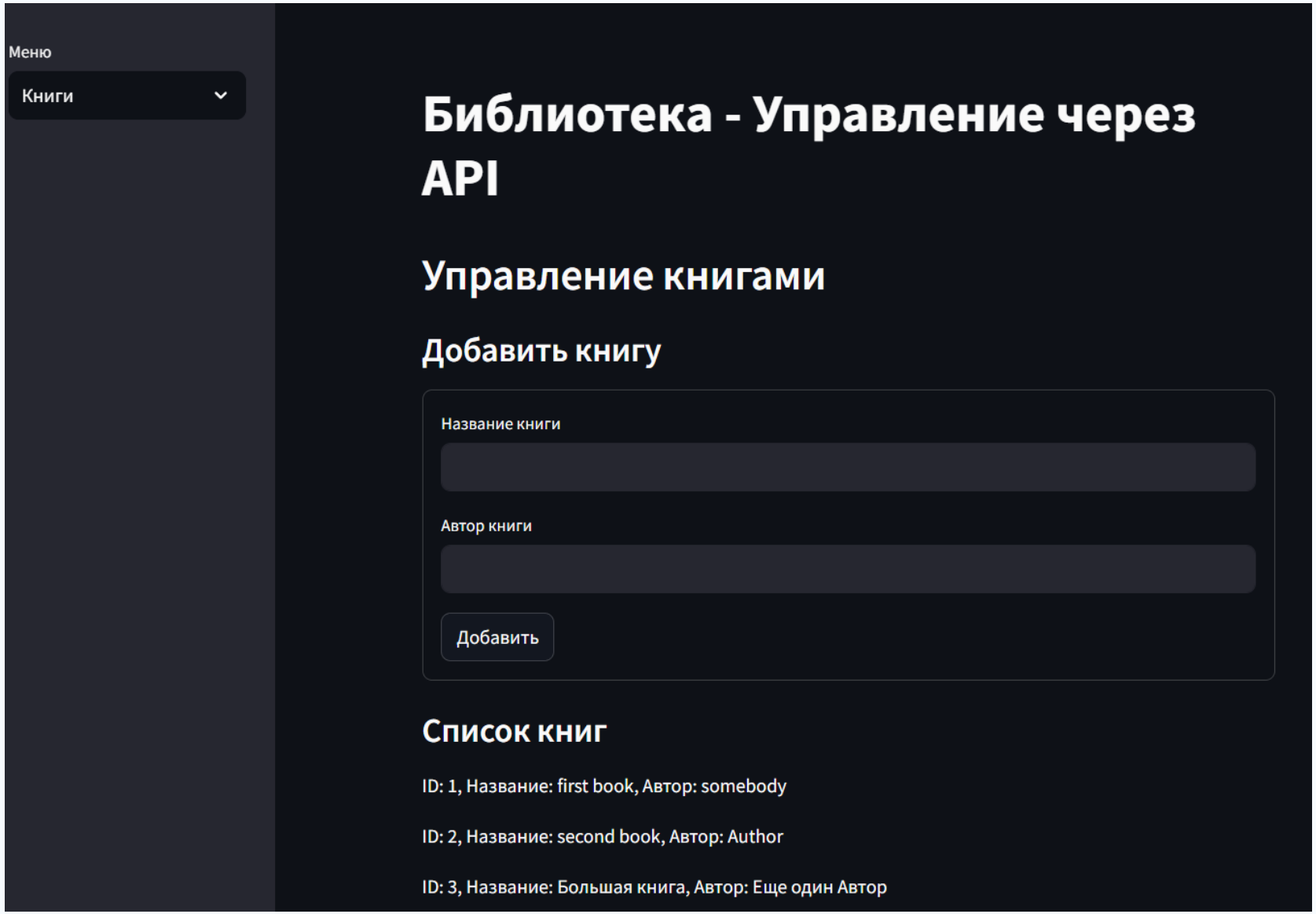
```
# Интерфейс для работы с выдачами
def issues_ui():
    st.header("Управление выдачами")

    # Создание выдачи
    st.subheader("Выдать книгу")
    with st.form("create_issue_form"):
        book_id = st.number_input("ID книги", step=1)
        reader_id = st.number_input("ID читателя", step=1)
        submitted = st.form_submit_button("Выдать книгу")
        if submitted:
            response = requests.post(f"{API_BASE_URL}/issue/",
json={"book_id": book_id, "reader_id": reader_id})
            if response.status_code == 200:
                st.success("Книга успешно выдана!")
            else:
                st.error(f"Ошибка: {response.json().get('detail'
)}")

    # Просмотр всех выдач
    st.subheader("Список выдач")
    response = requests.get(f"{API_BASE_URL}/issue/all")
    if response.status_code == 200:
        issues = response.json()
        for issue in issues:
            st.write(f"ID: {issue['id']}, Книга ID: {issue[
'book_id']}, Читатель ID: {issue['reader_id']}")
    else:
        st.error("Ошибка при получении списка выдач.")
```

Добавление пользовательского интерфейса к API сервису «Библиотека»

9. Сохраните изменения. Зайдите в браузер, выберите любой элемент из выпадающего списка меню и посмотрите как изменился интерфейс. Поскольку Streamlit весь код скрипта каждый раз когда вы взаимодействуете с интерфейсом (нажимайте кнопки или выбираете что-то из выпадающих списков и т.д.), то ваши изменения в коде будут видны сразу на пользовательском интерфейсе.

10. ***Для дальнейшей корректной работы у вас должна быть полностью выполнена практическая работа 5***. Если сейчас вы попробуете понажимать кнопки в созданном вами интерфейсе, например кнопку “Добавить” в пункте меню “Книги”, то получите ошибку. Так как код по кнопке добавить должен вызвать post метод вашего апи по адресу http://127.0.0.1:8000/book. Для этого ваше API из практической работы 5 или 6 должно быть запущено. Поэтому запустите ваш API сервис как это делалось в предыдущих практических и попробуйте еще раз повзаимодействовать с интерфейсом. Если ошибки сохраняются, то проверяйте соответствие ваших методов (маршрутов) выставленных в API, с теми, которые вызываются из вашего кода в ui.py.

Добавление пользовательского интерфейса к API сервису «Библиотека»



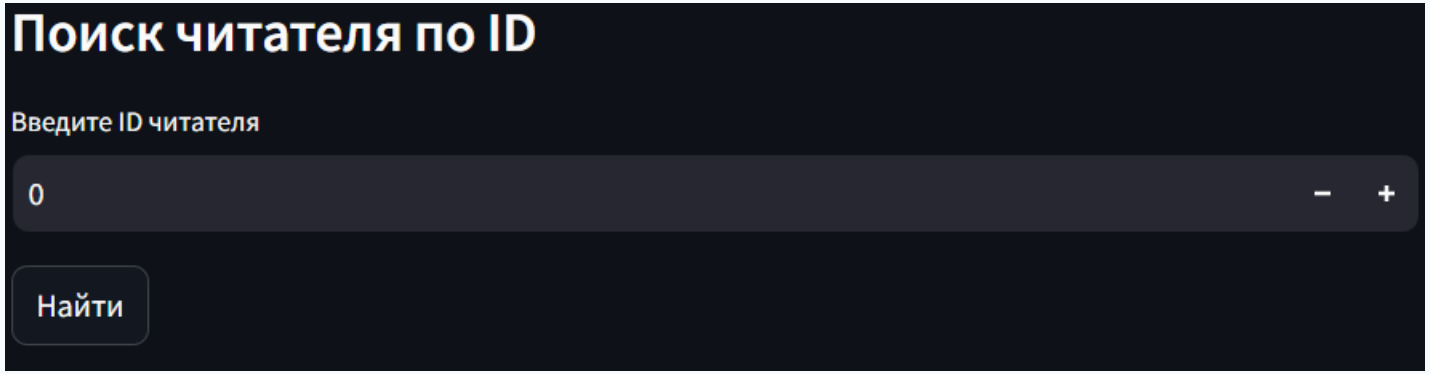
Добавление пользовательского интерфейса к API сервису «Библиотека»

11. После того когда вы убедитесь что все работает, добавим также возможность поиска читателя по ID и удаление конкретной выдачи книги

а. Добавьте код для поиска читателя в метод readers_ui перед кодом “выводом списка читателей”:

```
# Поиск читателя по ID
st.subheader("Поиск читателя по ID")
reader_id = st.number_input("Введите ID читателя", step=1)
if st.button("Найти"):
    response = requests.get(f"{API_BASE_URL}/reader/{int(reader_id)}")
    if response.status_code == 200:
        reader = response.json()
        st.write(f"ID: {reader['id']}, Имя: {reader['name']}")
    else:
        st.error(f"Читатель с ID {reader_id} не найден.")
```

Появится следующие элементы интерфейса:



Добавление пользовательского интерфейса к API сервису «Библиотека»

- b. Добавьте для удаление Выдачи в метод issues_ui перед кодом “просмотр всех выдач”:

```python
# Удаление выдачи
st.subheader("Удалить выдачу")
issue_id = st.number_input("ID выдачи", step=1)
if st.button("Удалить выдачу"):
    response = requests.delete(f"{API_BASE_URL}/issue/{int(issue_id)}")
    if response.status_code == 200:
        st.success(f"Выдача с ID {issue_id} успешно удалена!")
    else:
        st.error(f"Ошибка: {response.json().get('detail')}")
```

Появится следующие элементы интерфейса:

Добавление пользовательского интерфейса к API сервису «Библиотека»

- Еще раз проверьте работоспособность всех элементов интерфейса, а также добавленного поиска и удаления. Понаблюдайте в консоли вашего API какие и в какой момент выполняются запросы при работе с пользовательским интерфейсом:

```
INFO:     127.0.0.1:60316 - "GET /book/all HTTP/1.1" 200 OK
INFO:     127.0.0.1:60320 - "POST /book/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:60321 - "GET /book/all HTTP/1.1" 200 OK
INFO:     127.0.0.1:60322 - "POST /book/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:60323 - "GET /book/all HTTP/1.1" 200 OK
INFO:     127.0.0.1:60324 - "GET /issue/all HTTP/1.1" 200 OK
INFO:     127.0.0.1:60325 - "POST /issue/ HTTP/1.1" 400 Bad Request
INFO:     127.0.0.1:60326 - "GET /issue/all HTTP/1.1" 200 OK
INFO:     127.0.0.1:60327 - "POST /issue/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:60328 - "GET /issue/all HTTP/1.1" 200 OK
INFO:     127.0.0.1:60331 - "GET /issue/all HTTP/1.1" 200 OK
INFO:     127.0.0.1:60332 - "DELETE /issue/2 HTTP/1.1" 200 OK
INFO:     127.0.0.1:60333 - "GET /issue/all HTTP/1.1" 200 OK
INFO:     127.0.0.1:60334 - "POST /issue/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:60335 - "GET /issue/all HTTP/1.1" 200 OK
```