**Отчёт по лабораторной работе №1**

**Тема: Создание простейшего API для работы с данными**

**Выполнил: Гебель Игорь, группа ИСП.23.1А**

Цель работы

Познакомиться с принципами архитектуры REST и реализовать базовое приложение Web API на языке Python с использованием фреймворка FastAPI.

Постановка задачи

Необходимо создать серверное приложение, позволяющее работать с данными предметной области «Библиотека». API должно поддерживать выполнение CRUD-операций:  
- GET — получение информации;  
- POST — добавление новых записей;  
- PUT — редактирование данных;  
- DELETE — удаление записей.

Предметная область: «Библиотека»

Минимальный набор характеристик для хранения данных:  
- номер читательского билета,  
- ФИО читателя,  
- адрес,  
- телефон,  
- название книги,  
- дата выдачи книги,  
- дата возврата книги.

Ход выполнения работы

1. Создание модели данных.  
Реализован класс Book, описывающий сущность «Книга в библиотеке». Каждому объекту автоматически присваивается уникальный id с помощью модуля uuid.  
  
2. Организация условной базы.  
В качестве базы данных используется список books, содержащий несколько тестовых записей.  
  
3. Поиск записи.  
Добавлена вспомогательная функция find\_book(id) для поиска объекта по идентификатору.  
  
4. Реализация API.  
С помощью FastAPI реализованы маршруты:  
- /api/books — получить список всех книг;  
- /api/books/{id} — получить данные по конкретной книге;  
- /api/books (POST) — добавить книгу;  
- /api/users (PUT) — изменить данные;  
- /api/users/{id} (DELETE) — удалить книгу.  
  
5. Клиентская часть.  
В файле index.html реализован интерфейс для работы с API:  
- форма для ввода данных о книге;  
- кнопки «Сохранить» и «Сбросить»;  
- таблица с выводом списка книг и действиями «Изменить»/«Удалить».  
  
6. Работа с запросами.  
На стороне клиента используется JavaScript с методами fetch() для взаимодействия с сервером. Реализованы функции:  
- getBooks() — загрузка всех книг;  
- getBook(id) — загрузка одной книги в форму;  
- createBook(...) — добавление книги;  
- editBook(...) — изменение книги;  
- deleteBook(id) — удаление записи.

Результаты работы

- Разработано Web API, поддерживающее основные операции работы с данными.  
- Реализован простой веб-интерфейс для взаимодействия с сервером.  
- Проверена корректность работы всех CRUD-функций.

Вывод

В ходе лабораторной работы я освоил основы разработки REST API на Python с использованием FastAPI, реализовал базовые операции для предметной области «Библиотека» и подключил интерфейс для работы с данными через браузер.

Main.py:

import uuid  
from fastapi import FastAPI, Body, status  
from fastapi.responses import JSONResponse, FileResponse  
  
  
class Book:  
 def \_\_init\_\_(self, nom\_chit\_b, FIO\_chit, adres, telefon, book\_name, data\_vidachi, data\_sdachi):  
 self.nomer\_bileta = nom\_chit\_b  
 self.FIO = FIO\_chit  
 self.adres = adres  
 self.telefon = telefon  
 self.nazvanie = book\_name  
 self.vidacha = data\_vidachi  
 self.sdacha = data\_sdachi  
 self.id = str(uuid.uuid4())  
  
  
# условная база данных - набор объектов Person  
books = [Book(12, "Хапов Д.С.", "Пушкина 12", 89857632214, "Война и мир", "12.02.2024", "12.04.2024" ), Book(15, "Дрожжина С.Ф.", "Коммунистическая 9", 89168430193, "Капитанская дочка", "03.10.2025", "10.02.2026"), Book(17, "Панчиков И.Д.", "Ленина 2", 89260997468, "Мёртвые души", "09.09.2023", "09.12.2023")]  
  
  
# для поиска пользователя в списке people  
def find\_book(id):  
 for book in books:  
 if book.id == id:  
 return book  
 return None  
  
  
app = FastAPI()  
  
  
@app.get("/")  
async def main():  
 return FileResponse("public/index.html")  
  
  
@app.get("/api/books")  
def get\_books():  
 return books  
  
  
@app.get("/api/books/{id}")  
def get\_book(id):  
 # получаем пользователя по id  
 book = find\_book(id)  
 print(book)  
 # если не найден, отправляем статусный код и сообщение об ошибке  
 if book == None:  
 return JSONResponse(  
 status\_code=status.HTTP\_404\_NOT\_FOUND,  
 content={"message": "Пользователь не найден"}  
 )  
 # если пользователь найден, отправляем его  
 return book  
  
  
@app.post("/api/books")  
def create\_book(data=Body()):  
 book = Book(data["nom\_chit\_b"], data["FIO\_chit"], data["adres"], data["telefon"], data["book\_name"], data["data\_vidachi"], data["data\_sdachi"])  
 # добавляем объект в список people  
 books.append(book)  
 return book  
  
  
@app.put("/api/users")  
def edit\_book(data=Body()):  
 # получаем пользователя по id  
 book = find\_book(data["id"])  
 # если не найден, отправляем статусный код и сообщение об ошибке  
 if book == None:  
 return JSONResponse(  
 status\_code=status.HTTP\_404\_NOT\_FOUND,  
 content={"message": "Пользователь не найден"}  
 )  
 # если пользователь найден, изменяем его данные и отправляем обратно клиенту=  
 book.nomer\_bileta = data["nom\_chit\_b"]  
 book.FIO = data["FIO\_chit"]  
 book.adres = data["adres"]  
 book.telefon = data["telefon"]  
 book.nazvanie = data["book\_name"]  
 book.vidacha = data["data\_vidachi"]  
 book.sdacha = data["data\_sdachi"]  
 return book  
  
  
@app.delete("/api/users/{id}")  
def delete\_book(id):  
 # получаем пользователя по id  
 book = find\_book(id)  
  
 # если не найден, отправляем статусный код и сообщение об ошибке  
 if book == None:  
 return JSONResponse(  
 status\_code=status.HTTP\_404\_NOT\_FOUND,  
 content={"message": "Пользователь не найден"}  
 )  
  
 # если пользователь найден, удаляем его  
 books.remove(book)  
 return book

index.html:

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8" />  
 <title>Библиотека</title>  
 <style>  
 td {padding:5px;}  
 button {margin: 5px;}  
 </style>  
</head>  
<body>  
 <h2>Список книг</h2>  
 <div>  
 <input type="hidden" id="bookId" />  
 <p>  
 Номер читательского билета:<br/>  
 <input id="nom\_chit\_b" type="number" />  
 </p>  
 <p>  
 ФИО читателя:<br />  
 <input id="FIO\_chit" />  
 </p>  
 <p>  
 Адрес:<br />  
 <input id="adres" />  
 </p>  
 <p>  
 Телефон:<br />  
 <input id="telefon" type="text" />  
 </p>  
 <p>  
 Название книги:<br />  
 <input id="book\_name" />  
 </p>  
 <p>  
 Дата выдачи книги читателю:<br />  
 <input id="data\_vidachi" />  
 </p>  
 <p>  
 Дата сдачи книги читателем:<br />  
 <input id="data\_sdachi" />  
 </p>  
  
 <p>  
 <button id="saveBtn">Сохранить</button>  
 <button id="resetBtn">Сбросить</button>  
 </p>  
 </div>  
 <table border="1" cellpadding="5">  
 <thead>  
 <tr>  
 <th>Номер чит. билета</th>  
 <th>ФИО читателя</th>  
 <th>Адрес</th>  
 <th>Телефон</th>  
 <th>Название книги</th>  
 <th>Дата выдачи</th>  
 <th>Дата сдачи</th>  
 <th>Действия</th>  
 </tr>  
 </thead>  
 <tbody></tbody>  
 </table>  
  
 <script>  
 // Получение всех книг  
 async function getBooks() {  
 const response = await fetch("/api/books", {  
 method: "GET",  
 headers: { "Accept": "application/json" }  
 });  
 if (response.ok) {  
 const books = await response.json();  
 const rows = document.querySelector("tbody");  
 rows.innerHTML = ""; // Очистить таблицу перед добавлением  
 books.forEach(book => rows.append(row(book)));  
 }  
 }  
  
 // Получение одной книги и заполнение формы  
 async function getBook(id) {  
 const response = await fetch(`/api/books/${id}`, {  
 method: "GET",  
 headers: { "Accept": "application/json" }  
 });  
 if (response.ok) {  
 const book = await response.json();  
 document.getElementById("bookId").value = book.id;  
 document.getElementById("nom\_chit\_b").value = book.nomer\_bileta;  
 document.getElementById("FIO\_chit").value = book.FIO;  
 document.getElementById("adres").value = book.adres;  
 document.getElementById("telefon").value = book.telefon;  
 document.getElementById("book\_name").value = book.nazvanie;  
 document.getElementById("data\_vidachi").value = book.vidacha;  
 document.getElementById("data\_sdachi").value = book.sdacha;  
 } else {  
 const error = await response.json();  
 alert(error.message);  
 }  
 }  
  
 // Добавление книги  
 async function createBook(nom\_chit\_b, FIO\_chit, adres, telefon, book\_name, data\_vidachi, data\_sdachi) {  
 const response = await fetch("/api/books", {  
 method: "POST",  
 headers: { "Accept": "application/json", "Content-Type": "application/json" },  
 body: JSON.stringify({  
 nom\_chit\_b: parseInt(nom\_chit\_b, 10),  
 FIO\_chit: FIO\_chit,  
 adres: adres,  
 telefon: telefon,  
 book\_name: book\_name,  
 data\_vidachi: data\_vidachi,  
 data\_sdachi: data\_sdachi  
 })  
 });  
 if (response.ok) {  
 const book = await response.json();  
 document.querySelector("tbody").append(row(book));  
 } else {  
 const error = await response.json();  
 alert(error.message);  
 }  
 }  
  
 // Изменение книги  
 async function editBook(id, nom\_chit\_b, FIO\_chit, adres, telefon, book\_name, data\_vidachi, data\_sdachi) {  
 const response = await fetch("/api/users", { // В main.py PUT стоит на /api/users, можно поменять в main.py на /api/books для единообразия  
 method: "PUT",  
 headers: { "Accept": "application/json", "Content-Type": "application/json" },  
 body: JSON.stringify({  
 id: id,  
 nom\_chit\_b: parseInt(nom\_chit\_b, 10),  
 FIO\_chit: FIO\_chit,  
 adres: adres,  
 telefon: telefon,  
 book\_name: book\_name,  
 data\_vidachi: data\_vidachi,  
 data\_sdachi: data\_sdachi  
 })  
 });  
 if (response.ok) {  
 const book = await response.json();  
 const existingRow = document.querySelector(`tr[data-rowid='${book.id}']`);  
 if (existingRow) {  
 existingRow.replaceWith(row(book));  
 }  
 } else {  
 const error = await response.json();  
 alert(error.message);  
 }  
 }  
  
 // Удаление книги  
 async function deleteBook(id) {  
 const response = await fetch(`/api/users/${id}`, { // В main.py DELETE стоит на /api/users/{id}, лучше изменить на /api/books/{id} для единообразия  
 method: "DELETE",  
 headers: { "Accept": "application/json" }  
 });  
 if (response.ok) {  
 const book = await response.json();  
 const tr = document.querySelector(`tr[data-rowid='${book.id}']`);  
 if (tr) tr.remove();  
 } else {  
 const error = await response.json();  
 alert(error.message);  
 }  
 }  
  
 // Очистка формы  
 function reset() {  
 document.getElementById("bookId").value = "";  
 document.getElementById("nom\_chit\_b").value = "";  
 document.getElementById("FIO\_chit").value = "";  
 document.getElementById("adres").value = "";  
 document.getElementById("telefon").value = "";  
 document.getElementById("book\_name").value = "";  
 document.getElementById("data\_vidachi").value = "";  
 document.getElementById("data\_sdachi").value = "";  
 }  
  
 // Создание строки таблицы для книги  
 function row(book) {  
 const tr = document.createElement("tr");  
 tr.setAttribute("data-rowid", book.id);  
  
 tr.innerHTML = `  
 <td>${book.nomer\_bileta}</td>  
 <td>${book.FIO}</td>  
 <td>${book.adres}</td>  
 <td>${book.telefon}</td>  
 <td>${book.nazvanie}</td>  
 <td>${book.vidacha}</td>  
 <td>${book.sdacha}</td>  
 <td>  
 <button onclick="getBook('${book.id}')">Изменить</button>  
 <button onclick="deleteBook('${book.id}')">Удалить</button>  
 </td>  
 `;  
 return tr;  
 }  
  
 // События кнопок  
 document.getElementById("resetBtn").addEventListener("click", () => reset());  
  
 document.getElementById("saveBtn").addEventListener("click", async () => {  
 const id = document.getElementById("bookId").value;  
 const nom\_chit\_b = document.getElementById("nom\_chit\_b").value;  
 const FIO\_chit = document.getElementById("FIO\_chit").value;  
 const adres = document.getElementById("adres").value;  
 const telefon = document.getElementById("telefon").value;  
 const book\_name = document.getElementById("book\_name").value;  
 const data\_vidachi = document.getElementById("data\_vidachi").value;  
 const data\_sdachi = document.getElementById("data\_sdachi").value;  
  
 if (!nom\_chit\_b || !FIO\_chit || !adres || !telefon || !book\_name || !data\_vidachi || !data\_sdachi) {  
 alert("Пожалуйста, заполните все поля!");  
 return;  
 }  
  
 if (id === "") {  
 await createBook(nom\_chit\_b, FIO\_chit, adres, telefon, book\_name, data\_vidachi, data\_sdachi);  
 } else {  
 await editBook(id, nom\_chit\_b, FIO\_chit, adres, telefon, book\_name, data\_vidachi, data\_sdachi);  
 }  
 reset();  
 await getBooks();  
 });  
  
 // Инициализация - загрузить книги при старте  
 getBooks();  
 </script>  
</body>  
</html>