

Министерство образования и науки РФ  
Федеральное государственное автономное образовательное учреждение высшего  
образования  
«Омский государственный технический университет»

Факультет (институт) Информационных технологий и компьютерных систем

Кафедра Прикладная математика и фундаментальная информатика

### Лабораторная работа №3

по дисциплине Алгоритмизация и программирование

на тему Разработка WebAPI с использованием фреймворка Oat++.

Справочник музыкальных альбомов.

Студента Эксперт Дианы Дмитриевны

фамилия, имя, отчество полностью

Курс 1

Группа ФИТ-221

Направление (специальность) 02.03.02

Фундаментальная информатика и информационные технологии

код, наименование

Руководитель ст. преподаватель

ученая степень, звание

Федотова И.В.

фамилия, инициалы

Выполнил 09.05.2023

дата, подпись студента

**Работа защищена с количеством баллов**

дата, подпись руководителя

Омск 2023

## Теоретическая часть по фреймворку Oat++

**Фре́ймворк** — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Употребляется также слово «каркас», а некоторые авторы используют его в качестве основного, в том числе не базируясь вообще на англоязычном аналоге. Можно также говорить о каркасном подходе как о подходе к построению программ, где любая конфигурация программы строится из двух частей:

1. Постоянная часть — каркас, не меняющийся от конфигурации к конфигурации и несущий в себе гнёзда, в которых размещается вторая, переменная часть;
2. Сменные модули (или точки расширения).

Фреймворк Oat++ предназначен для написания Backend приложения на языке C++, предоставляет разработчикам набор инструментов реализации различных функций веб-приложений: маршрутизацию запросов, обработку HTTP-запросов и ответов, работу с базами данных и другое. Фреймворк имеет интегрированный набор инструментов для разработки и отладки приложений, что значительно упрощает процесс разработки и сокращает время на развертывание приложения в продакшене. Благодаря модульной архитектуре oat++ разработчики могут создавать масштабируемые и гибкие приложения, которые легко адаптируются к изменяющимся потребностям пользователей

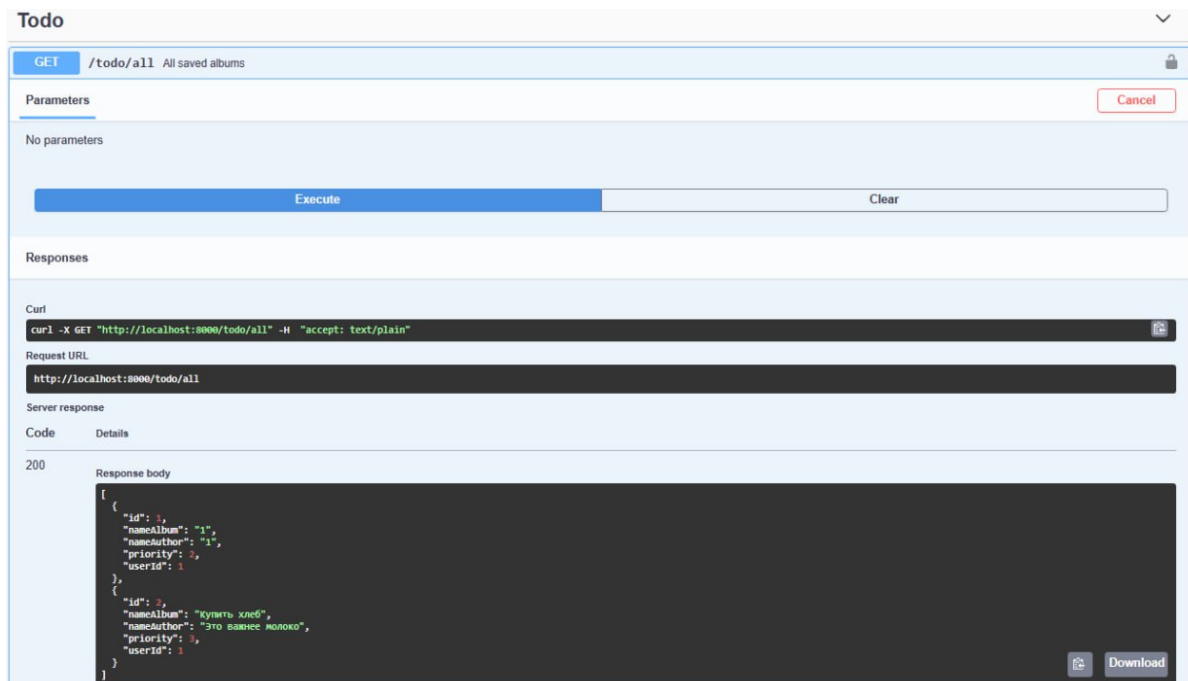
## **Теоретическая часть по базе данных SQLite**

База данных SQLite является самым простым вариантом реализации базы данных. SQLite поддерживает большинство функций и команд, которые доступны в других реляционных базах данных, таких как MySQL и PostgreSQL. Он позволяет хранить данные в таблицах, которые могут быть связаны друг с другом посредством внешних ключей. SQLite также поддерживает индексирование данных, что ускоряет процесс поиска и сортировки.

SQLite может быть использован для хранения данных приложения, кэширования данных, хранение данных сенсоров.

## Скриншоты запросов в Swagger

Вывод альбомов:

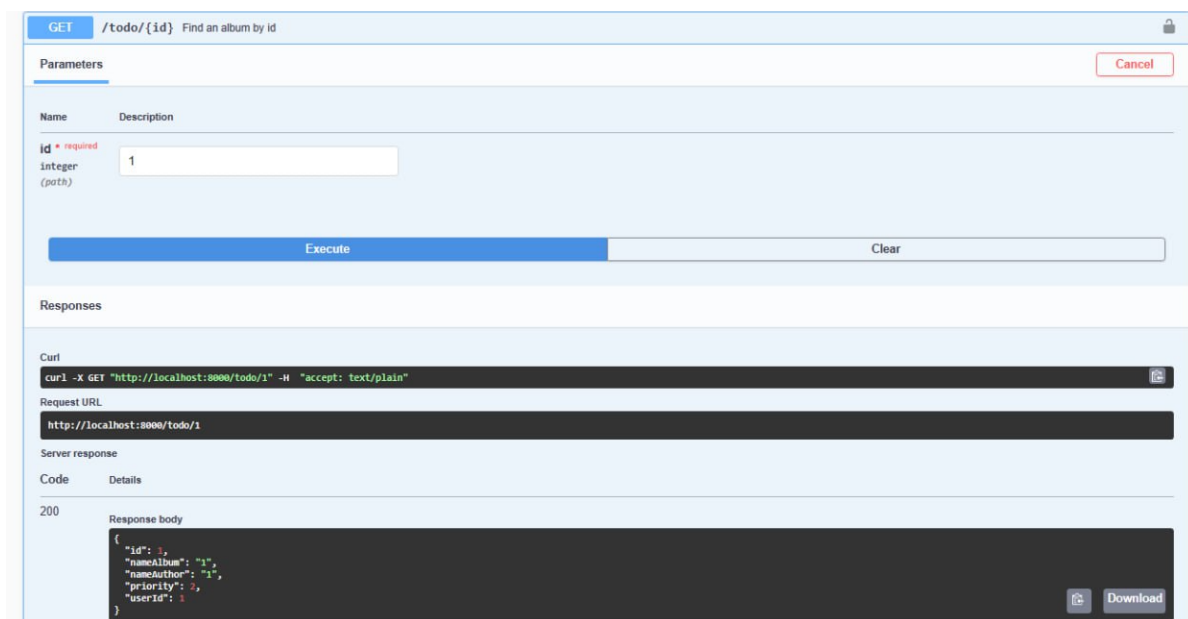


The screenshot shows the Swagger UI for the endpoint `GET /todo/all` with the description "All saved albums". The "Parameters" tab is active, showing "No parameters". Below the "Execute" and "Clear" buttons, the "Responses" section displays a 200 status code. The response body is a JSON array of two album objects. The first object has `id: 1`, `nameAlbum: "1"`, `nameAuthor: "1"`, `priority: 2`, and `userId: 1`. The second object has `id: 2`, `nameAlbum: "Купить хлеб"`, `nameAuthor: "Это самое молоко"`, `priority: 3`, and `userId: 1`. A "Download" button is visible at the bottom right of the response body.

```
curl -X GET "http://localhost:8080/todo/all" -H "accept: text/plain"
```

```
{
  "id": 1,
  "nameAlbum": "1",
  "nameAuthor": "1",
  "priority": 2,
  "userId": 1
},
{
  "id": 2,
  "nameAlbum": "Купить хлеб",
  "nameAuthor": "Это самое молоко",
  "priority": 3,
  "userId": 1
}
```

Вывод альбома по id:



The screenshot shows the Swagger UI for the endpoint `GET /todo/{id}` with the description "Find an album by id". The "Parameters" tab is active, showing a required integer parameter `id` (path) with a value of `1` entered in the input field. Below the "Execute" and "Clear" buttons, the "Responses" section displays a 200 status code. The response body is a JSON object representing a single album: `{ "id": 1, "nameAlbum": "1", "nameAuthor": "1", "priority": 2, "userId": 1 }`. A "Download" button is visible at the bottom right of the response body.

```
curl -X GET "http://localhost:8080/todo/1" -H "accept: text/plain"
```

```
{
  "id": 1,
  "nameAlbum": "1",
  "nameAuthor": "1",
  "priority": 2,
  "userId": 1
}
```

## Изменение альбома:

**PUT** /todo/{id} Editing a Saved Album

Parameters

Name	Description
<b>id</b> * required integer (path)	Description of ID

1

Request body required

application/json

```
{
  "nameAlbum": "One X",
  "nameAuthor": "Three Days Grace",
  "priority": 1,
  "userId": 1
}
```

Execute

Clear

Responses

Curl

```
curl -X PUT "http://localhost:8000/todo/1" -H "accept: text/plain" -H "Content-type: application/json" -d '{"nameAlbum":"One X","nameAuthor":"Three Days Grace","priority":1,"userId":1}'
```

Request URL

```
http://localhost:8000/todo/1
```

Server response

Code	Details
200	<div>Response body</div> <div><pre>{   "nameAlbum": "One X",   "nameAuthor": "Three Days Grace",   "priority": 1,   "userId": 1 }</pre></div> <div>Download</div>

## Удаление альбома:

**DELETE** /todo/{id} Delete album by id

Parameters

Name	Description
<b>id</b> * required integer (path)	

2

Execute

Clear

Responses

Curl

```
curl -X DELETE "http://localhost:8000/todo/2" -H "accept: text/plain"
```

Request URL

```
http://localhost:8000/todo/2
```

Server response

Code	Details
204 <i>Undocumented</i>	<div>Response headers</div> <div><pre>connection: keep-alive content-length: 0 server: oatpp/1.3.0</pre></div>

Responses

Code	Description	Links
200	SUCCESS	No links

## Добавление альбома:

POST

/todo Adding an album

Parameters

Cancel

No parameters

Request body required

application/json

```
{  "nameAlbum": "Metallica",  "nameAuthor": "Master of Puppets",  "priority": 2,  "userId": 1}
```

ExecuteClear

Responses

Curl

```
curl -X POST "http://localhost:8000/todo" -H "accept: text/plain" -H "Content-Type: application/json" -d '{"nameAlbum":"Metallica","nameAuthor":"Master of Puppets","priority":2,"userId":1}'
```

Request URL

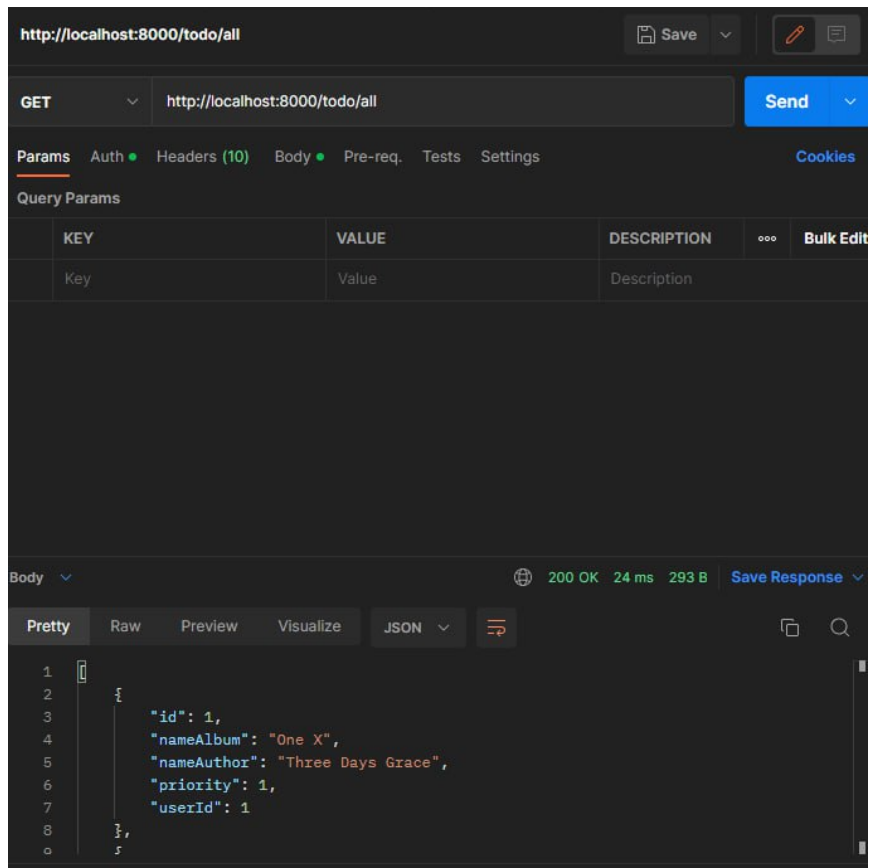
http://localhost:8000/todo

Server response

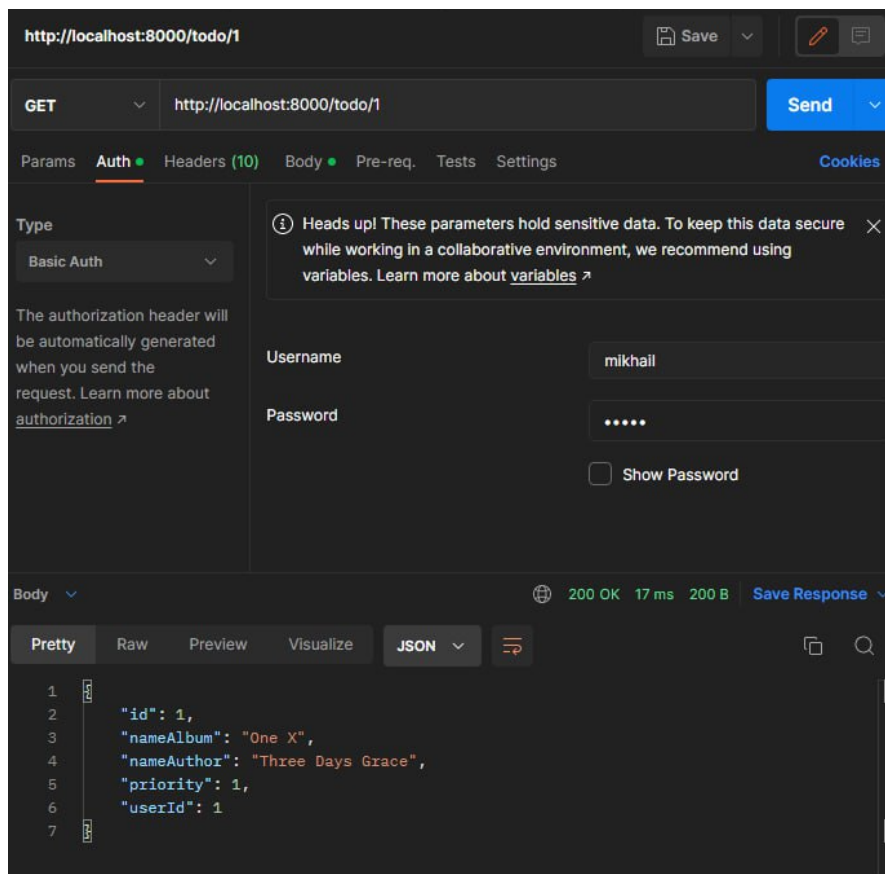
Code	Details
201	Response body <div><pre>{  "nameAlbum": "Metallica",  "nameAuthor": "Master of Puppets",  "priority": 2,  "userId": 1}</pre></div> <div>Download</div>

## Скриншоты запросов в Postman

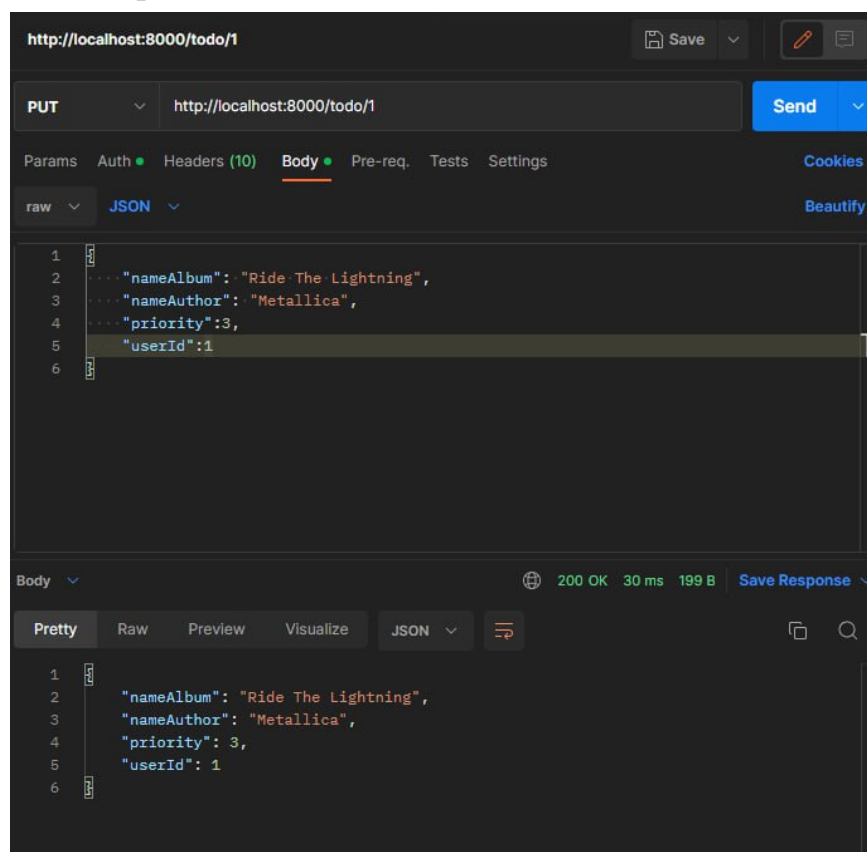
Вывод всех альбомов:



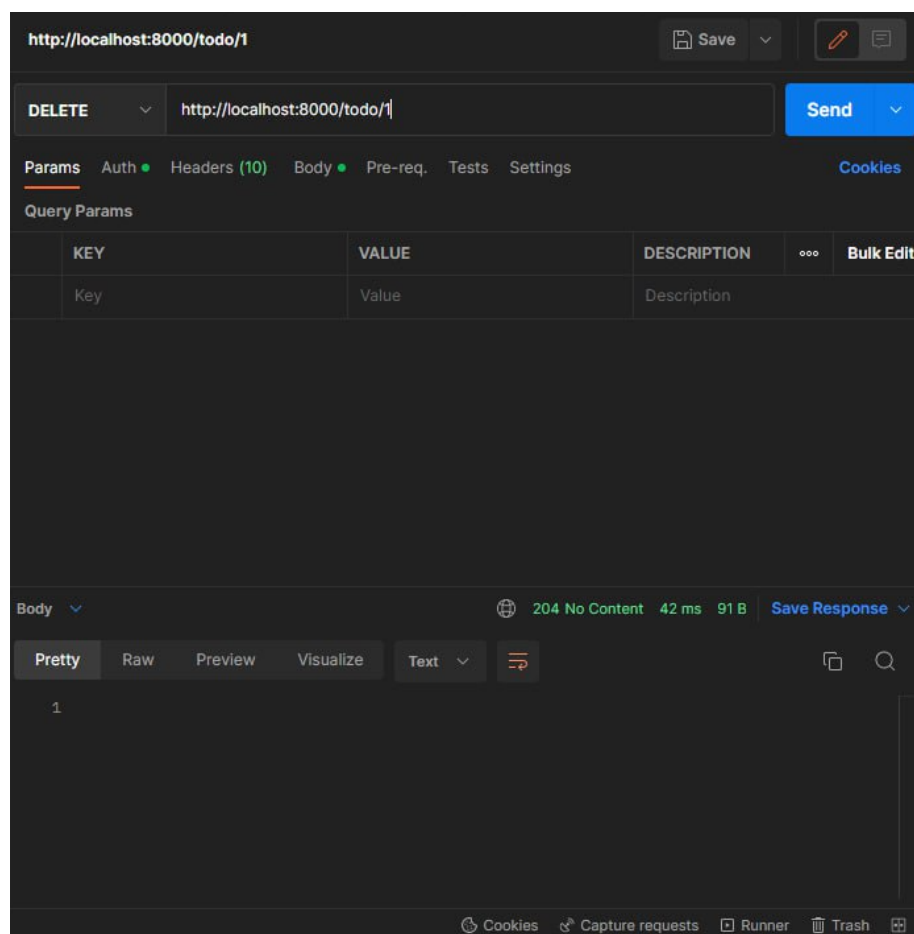
Вывод альбомов по id:



Редактирование альбома по id:



Удаление альбома по id:





### Пример кода с подключением API-контроллера:

```
#include <iostream>

#include <oatpp/network/Server.hpp>

#include "oatpp/web/server/HttpConnectionHandler.hpp"

#include "oatpp/network/tcp/server/ConnectionProvider.hpp"

#include "AppComponent.hpp"

#include "controller/ToDoController.hpp"

#include "oatpp-swagger/Controller.hpp"


void runServer() {
    AppComponent components;

    OATPP_COMPONENT(std::shared_ptr <
oatpp::web::server::HttpRouter>, httpRouter);

    oatpp::web::server::api::Endpoints docEndpoints;

    docEndpoints.append(httpRouter-
>addController(std::make_shared<ToDoController>())->getEndpoints());

    httpRouter-
>addController(oatpp::swagger::Controller::createShared(docEndpoints));

    OATPP_COMPONENT(std::shared_ptr <
oatpp::network::ConnectionHandler>, serverConnectionHandler);

    OATPP_COMPONENT(std::shared_ptr <
oatpp::network::ServerConnectionProvider>, serverConnectionProvider);

    oatpp::network::Server server(serverConnectionProvider,
serverConnectionHandler);

    OATPP_LOGI("App", "Сервер запущен!");

    server.run();
}
```

```
int main() {  
    setlocale(LC_ALL, "Rus");  
    oatpp::base::Environment::init();  
    runServer();  
    return 0;  
}
```