Министерство сельского хозяйства Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Пермский государственный аграрно-технологический университет

имени академика Д.Н. Прянишникова»

Кафедра Информационных технологий

и программной инженерии

КУРСОВОЙ ПРОЕКТ

по дисциплине: Программная инженерия

объектно-ориентированного программирования

на тему: Разработка клиент-серверного приложения.

Выполнил:

студент 3-го курса очного отделения

специальности 09.03.04 Программная инженерия

шифр ПИнб-1711-2018

Лыкасова Елизавета Романовна

Проверил:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

Пермь – 2021

Оглавление

[Введение 3](#_Toc76031668)

[Постановка задачи на проектирование 3](#_Toc76031669)

[Разработка структуры базы данных 5](#_Toc76031670)

[Разработка приложения 7](#_Toc76031671)

[Заключение 19](#_Toc76031672)

[Список используемых источников 20](#_Toc76031673)

[Приложение 21](#_Toc76031674)

# Введение

Неоценима роль книги в жизни человека. Книга учит нас мыслить, выражать свои чувства, делает нас добрее и лучше. Из книг мы узнаем много нового. Чтение книг помогает нам развиваться духовно, повышает уровень интеллекта, учит грамотно писать.

# Постановка задачи на проектирование

У пользователя есть список книг, которые он прочитал и оценил, для того чтобы автоматизировать процесс добавления, редактирования и удаления книг, было принято решение создать web-приложение.

Целью проектирования является разработка простого и удобного web-приложения для работы с базой данных, в которой содержится таблица со списком книг, прочитанных пользователем.

Предметная область: прочитанные книги.

Для реализации поставленной цели необходимо решить следующие задачи:

1. Изучить предметную область.
2. Выбрать технологии для реализации проекта.
3. Разработать структуру базы данных.
4. Разработать приложение.

Для достижения цели и выполнения задач были выбраны следующие технологии:

Visual Studio Code - бесплатный и очень популярный редактор кода от Microsoft. Редактор поможет в работе веб-разработчикам и верстальщикам всех уровней. С одной стороны он подходит новичкам, потому что его интерфейс интуитивно прост и понятен. С другой стороны в VS Code встроены много возможностей, которые интересны опытным разработчикам. Редактор поддерживает множество языков программирования и легко настраивается под пользователя. Можно установить различные сочетания клавиш и цветовые схемы.

Node.js - это серверная платформа для работы с JavaScript через движок V8. JavaScript выполняет действие на стороне клиента, а Node - на сервере. С помощью Node можно писать полноценные приложения. Node умеет работать с внешними библиотеками, вызывать команды из кода на JavaScript и выполнять роль веб-сервера.

Express - это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

MySQL это система управления реляционными базами данных с открытым исходным кодом (СУРБД) с моделью клиент-сервер. СУРБД - это программное обеспечение или служба, используемая для создания и управления базами данных на основе реляционной модели.

phpMyAdmin - это приложение написанное на PHP и обеспечивающее полноценную, в том числе удаленную, работу с базами данных MySQL через браузер. Так как phpMyAdmin позволяет во многих случаях обойтись без непосредственного ввода команд SQL, то работа с базами данных становится вполне посильной задачей даже для человека весьма поверхностно знакомого с MySQL.

# Разработка структуры базы данных

Разработка структуры начинается с создания базы данных. Созданная база данных представлена на рисунке 1.

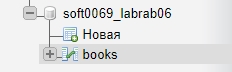


Рисунок 1 - Структура базы данных

Для достижения поставленных целей понадобится лишь одна таблица, создадим ее также в phpmyadmin. Добавим поля id, title, author, genre, publicationDate и rating. Поле id сделаем автоинкрементным и присвоим ему первичный ключ. Структура таблицы books представлена на рисунке 2.



Рисунок 2 - Структура таблицы books

Как видно на рисунке 2, поля id и publicationDate имеют числовой тип данных (int), поля title, author и genre текстовый (varchar), а поле rating тип данных float.

Для того чтобы наполнить таблицу данными, был использован SQL запрос, текст запроса представлен в листинге №1.

Листинг №1 – SQL запрос для добавления строк в таблицу

INSERT INTO `books`(`id`, `title`, `author`, `genre`, `publicationDate`, `rating`) VALUES

(NULL,'1984','Джордж Оруэлл','Фантастика',1949,4.5),

(NULL,'Мастер и Маргарита','Михаил Булгаков','Мистика',1967,4.7),

(NULL,'Три товарища','Эрих Мария Ремарк','Зарубежная классика',1936,4.7),

(NULL,'Портрет Дориана Грея','Оскар Уайльд','Фантастика',1890,4.6),

(NULL,'Маленький принц','Джордж Оруэлл','Зарубежная классика',1943,4.5),

(NULL,'Над пропастью во ржи','Джером Д. Сэлинджер','Зарубежная классика',1951,4.1),

(NULL,'Вино из одуванчиков','Рей Брэдбери','Зарубежная классика',1957,4.2),

(NULL,'Убить пересмешника','Харпер Ли','Зарубежная классика',1960,4.8),

(NULL,'Преступление и наказание','Федор Достоевский','Русская классика',1866,4.8),

(NULL,'Хлеб с ветчиной','Чарльз Буковски','Современная зарубежная литература',1982,4.3),

(NULL,'Женщины','Чарльз Буковски','Современная зарубежная литература',1978,3.9),

(NULL,'Идиот','Фёдор Достоевский','Фантастика',1869,4.7);

Заполненная таблица представлена на рисунке 3.



Рисунок 3 - Заполненная таблица books

# Разработка приложения

Вывод данных и работа с данными из базы реализована через Node.js и фреймворк Express.

Приложение Express для определения визуального интерфейса использует не стандартные файлы html, а представления (views). В этих представлениях можно размещать шаблоны, вместо которых программа на js в процессе рендеринга будет вставлять какое-то динамическое содержимое. Для управления представлениями во время работы приложения следует подключить один из существующих движков (view engine): Handlebars, Pug, Jade, Dust, Nunjucks, EJS. Мы будем использовать движок Handlebars, именно поэтому файлы представлений будут иметь расширение hbs.

Также в работе нам понадобится пакет mysql2 для работы с СУБД MySQL.

Для того чтобы установить Express, mysql2 и Handlebars в консоль была введена следующая команда:

npm i express mysql2 hbs

В папке app создадим таблицу стилей style.css в папке css, файл app.js и несколько представлений: index.hbs, create.hbs и edit.hbs в папке views.

Структура файлов приложения представлена на р исунке 4.

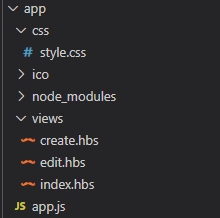


Рисунок 4 - Структура файлов web-приложения

Начнем с файла app.js. Основной функционал приложения состоит в методах фреймворка Express, таких как get и post. Используя метод get мы можем получить данные от пользователя через форму, в нашем случае, мы будем добавлять новые строки в базу данных, выводя на экран форму, описанную в представлении create.hbs, о нем позже. После того как данные введены пользователем, запускается метод post, отправляющий данные в базу.

Оба метода используют пул SQL запросов, выполняя которые, метод post извлекает данные из форм в формате ключ-значение и отправляет их в базу, после чего, метод get выводит данные из базы на экран.

Рассмотрим файл app.js подробнее. В листинге №2 представлена часть кода программы app.js.

Листинг №2 – app.js часть 1.

const mysql = require("mysql2");

const express = require("express");

const paramsDB = {

host: "pgsha.ru",

port: "35006",

user: "soft0069",

password: "y61cf4vJ",

database: "soft0069\_labrab06"

};

const pool = mysql.createPool(paramsDB);

function get\_connection() {

return mysql.createConnection(paramsDB);

}

const app = express();

const urlencodedParser = express.urlencoded({extended: false});

app.use('/css', express.static(\_\_dirname + '/css'));

app.set("view engine", "hbs");

Здесь мы подключаем необходимые библиотеки, задаем параметры подключения к БД, создаем pool подключения, определяем специальный парсер для метода post, а также подключаем таблицу стилей и движок Handlebars.

Дальше в программе app.js идут сами методы get и post.

Начнем с метода, который выводит таблицу на экран, код представлен в листинге №3.

Листинг №3 – Вывод таблицы на экран пользователя.

app.get("/", function(req, res) {

    let query = "SELECT \* FROM books";

    pool.query(query, function(err, data) {

        if (err) return console.log(err);

        res.render("index.hbs", {

            books: data

        });

    });

});

Здесь с помощью запроса SELECT из таблицы books выбираются все записи и после этого методом render выводятся на экран в представление index.hbs

Дальше приведен листинг представления главной страницы:

Листинг №4 – index.hbs

<!DOCTYPE html>

<html>

<head>

  <title>Список книг</title>

  <meta charset="utf-8" />

  <link rel="shortcut icon" href="../ico/favicon.ico" type="image/x-icon">

  <link rel="stylesheet" type="text/css" href="/css/style.css" />

</head>

<body>

  <center>

  <h3>Список прочитанных книг</h3>

  <br>

  <a href="/create" class="btn">Добавить книгу</a>

  <a href="/recover" class="btn">Восстановить список</a>

  <a href="/clear" class="btn">Очистить список</a>

  <table>

    <tr>

      <th>Редактор</th>

      <th>Название</th>

      <th>Автор&nbsp;<a href="/sort/author.ASC">▲</a><a href="/sort/author.DESC">▼</a></th>

      <th>Жанр</th>

      <th>Дата публикации&nbsp;<a href="/sort/publicationDate.ASC">▲</a><a href="/sort/publicationDate.DESC">▼</a></th>

      <th>Оценка&nbsp;<a href="/sort/rating.ASC">▲</a><a href="/sort/rating.DESC">▼</a></th>

    </tr>

    {{#each books}}

    <tr>

      <td width=180 class="editor">

       <form action="/edit/{{this.id}}" method="GET" style="display:inline;">

       <input type="submit" value="Изменить" />

       </form>

       <form action="/delete/{{this.id}}" method="POST" style="display:inline;">

       <input type="submit" value="Удалить" />

       </form>

          </td>

          <td>{{this.title}}</td>

          <td>{{this.author}}</td>

          <td>{{this.genre}}</td>

          <td>{{this.publicationDate}}</td>

          <td>{{this.rating}}</td>

        </tr>

        {{/each}}

    </table>

    </center>

</body>

<html>

Index.hbs используется для отображения главной страницы web-приложения, на нем расположены несколько кнопок для работы с данными и сама таблица.

Кнопки используются для перехода к другим представлениям (create и edit), а также для выполнения функций /delete, /recover и /clean, для восстановления и очистки базы данных.

Внешний вид представления index.hbs представлен на рисунке 5.



Рисунок 5 - Внешний вид index.hbs

Функционал приложения также позволяет добавлять, удалять и изменять данные в базе. Для того чтобы добавить запись нужно нажать на кнопку «Добавить книгу» на главной странице, после чего с помощью метода get мы выводим представление create.hbs. Данный метод представлен в листинге №5.

Листинг №5 – метод get для вывода create.hbs

app.get("/create", function(req, res) {

    res.render("create.hbs");

});

Рассмотрим представление create.hbs, которое описывают форму для добавления данных в таблицу. В листинге №6 представлен лишь основной код, определяющий функционал добавления новой записи.

Листинг №6 – create.hbs

<body>

    <h1>Добавить книгу</h1>

    <form method="POST">

        <label>Название</label><br>

        <input name="title" class="field" /><br><br>

        <label>Автор</label><br>

        <input name="author" class="field" /><br><br>

        <label>Жанр</label><br>

        <input name="genre" class="field" /><br><br>

        <label>Дата публикации</label><br>

        <input name="publicationDate" type="number" class="field" min="100" max="2025" /><br><br>

        <label>Оценка</label><br>

        <input name="rating" type="number" min="1" max="10" step="1" class="field" /><br><br>

        <input type="submit" value="Сохранить" />

    </form>

    <br>

    <a href="/">На главную</a>

</body>

То, как выглядит форма добавления новой записи в базу можно увидеть на рисунке 6.

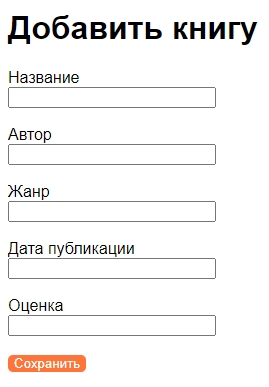


Рисунок 6 - Внешний вид create.hbs

После заполнения формы и нажатия кнопки «Сохранить», метод post отправляет данные в базу, добавляя новую запись, код представлен в листинге №7.

Листинг №7 – метод post для добавления новой записи

app.post("/create", urlencodedParser, function (req, res) {

    if (!req.body) return res.sendpublicationDate(400);

    const title = req.body.title;

    const author = req.body.author;

    const genre = req.body.genre;

    const publicationDate = req.body.publicationDate;

    const rating = req.body.rating;

    let query = "INSERT INTO books (title, author, genre, publicationDate, rating) VALUES (?,?,?,?,?)";

    let params = [title, author, genre, publicationDate, rating];

    pool.query(query, params, function(err, data) {

        if (err) return console.error(err);

        res.redirect("/");

    });

});

Параметры извлекаются из формы и передаются в запрос INSERT INTO, который вставляет новую запись в таблицу.

Для того чтобы изменить запись нужно нажать на кнопку «Изменить» на главной странице, работает это почти также, как и добавление новой записи. Сначала пользователь заполняет форму, потом данные отправляются на сервер.

Код методов get и post для изменения записи представлен в листинге №8.

Листинг №8 – методы get и post для изменения записи

app.get("/edit/:id", function(req, res) {

    const id = req.params.id;

    pool.query("SELECT \* FROM books WHERE id=?", [id], function(err, data) {

        if (err) return console.error(err);

        res.render("edit.hbs", {

            books: data[0]

        });

    });

});

app.post("/edit", urlencodedParser, function (req, res) {

    if (!req.body) return res.sendpublicationDate(400);

    const id = req.body.id;

    const title = req.body.title;

    const author = req.body.author;

    const genre = req.body.genre;

    const publicationDate = req.body.publicationDate;

    const rating = req.body.rating;

    let query = "UPDATE books SET title=?, author=?, genre=?, publicationDate=?, genre=? WHERE id=?";

    let params = [title, author, genre, publicationDate, genre, id];

    pool.query(query, params, function(err, data) {

        if (err) return console.error(err);

        res.redirect("/");

    });

});

Форма edit.hbs представлена на рисунке 7.

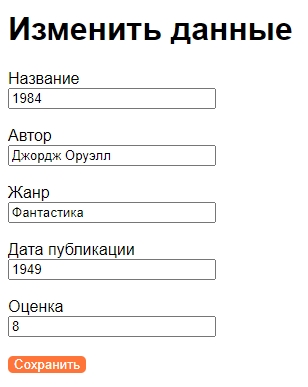


Рисунок 7 - Внешний вид edit.hbs

Далее рассмотрим удаление записи из таблицы. Для этого нужно нажать на кнопку «Удалить». Здесь используется только метод post, так как выводить какое-либо представление пользователю не требуется, нужно лишь посредством SQL запроса удалить нужную запись. Код представлен в листинге №9.

Листинг №9 – метод post для удаления записи

app.post("/delete/:id", function(req, res) {

    const id = req.params.id;

    pool.query("DELETE FROM books WHERE id=?", [id], function(err, data) {

        if (err) return console.log(err);

        res.redirect("/");

    });

});

После удаления записи, загружается главная страница web-приложения.

Сортировка записей в таблице происходит по полям rating и publicationDate для упорядочивания записей по оценке пользователя и дате публикации книги, а также по полю author, для группировки книг, написанных одним автором. Для того чтобы выполнить сортировку, нужно нажать на одну из стрелочек около названий столбцов таблицы, где стрелка вверх будет означать сортировку по возрастанию, стрелка вниз по убыванию.

Сортировка наоборот реализована лишь методом get, ведь отправлять данные в базу не требуется, вместо этого хватает лишь отобразить изменения на главной странице web-приложения. Код представлен в листинге №10.

Листинг №10 – метод get для сортировки записей.

app.get("/sort/:field.:direct", function(req, res) {

    const field = req.params.field;

    const direct = req.params.direct;

    let query = "SELECT \* FROM books ORDER BY " + field + " " + direct;

    pool.query(query, function(err, data) {

        if (err) return console.log(err);

        res.render("index.hbs", {

            books: data

        });

    });

});

В SQL запрос передается поле, по которому происходит сортировка и порядок (по убыванию, по возрастанию). Далее обновляется главная страница index.hbs. Результат сортировки по полю rating представлен на рисунке 8.



Рисунок 8 - Сортировка по полю Оценка, по убыванию

Самыми сложными в реализации являются методы для очистки и восстановления таблицы. Сначала рассмотрим очистку таблицы. Для того чтобы удалить все данные из таблицы, нужно нажать кнопку «Очистить список» на главной странице. После нажатия на кнопку выполняется метод get, код представлен на листинге №11.

Листинг №11 – метод get для очистки таблицы.

app.get("/clear", function(req, res) {

    let query\_truncate = "TRUNCATE books";

    const conn = get\_connection();

    conn.promise()

    .query(query\_truncate)

    .then(() => {

        conn.promise()

            .query('SELECT \* FROM books')

            .then(([data]) => {

                res.render('index.hbs', {

                    books: data

                });

            })

            .then(conn.end())

            .catch((err) => console.error(err));

    })

    .catch((err) => console.error(err));

});

Здесь с использованием промисов выполняется сначала удаление данных из таблицы, а затем вывод данных на главную страницу. Результат выполнения метода представлен на рисунке 9.



Рисунок 9 - Результат очистки таблицы

И, пожалуй, самым сложным методом в данном web-приложении является восстановление таблицы. Для восстановления списка прочитанных книг нужно нажать на кнопку «Восстановить список», после чего запускается метод get. Код представлен в листинге №12.

Листинг №12 – метод get для восстановления таблицы.

app.get("/recover", function(req, res) {

    let query\_truncate = "TRUNCATE `books`";

    let query\_insert = "INSERT INTO `books` \

    (`id`, `title`, `author`, `genre`, `publicationDate`, `rating`) VALUES \

    (NULL,'1984','Джордж Оруэлл','Фантастика',1949,8), \

    (NULL,'Мастер и Маргарита','Михаил Булгаков','Мистика',1967,9), \

    (NULL,'Три товарища','Эрих Мария Ремарк','Зарубежная классика',1936,7), \

    (NULL,'Портрет Дориана Грея','Оскар Уайльд','Фантастика',1890,7), \

    (NULL,'Маленький принц','Джордж Оруэлл','Зарубежная классика',1943,7), \

    (NULL,'Над пропастью во ржи','Джером Д. Сэлинджер','Зарубежная классика',1951,10), \

    (NULL,'Вино из одуванчиков','Рей Брэдбери','Зарубежная классика',1957,6), \

    (NULL,'Убить пересмешника','Харпер Ли','Зарубежная классика',1960,9), \

    (NULL,'Преступление и наказание','Федор Достоевский','Русская классика',1866,8), \

    (NULL,'Хлеб с ветчиной','Чарльз Буковски','Современная зарубежная литература',1982,10), \

    (NULL,'Женщины','Чарльз Буковски','Современная зарубежная литература',1978,8), \

    (NULL,'Идиот','Фёдор Достоевский','Фантастика',1869,7);";

    const conn = get\_connection();

    conn.promise()

        .query(query\_truncate)

        .then(() => {

            conn.promise()

                .query(query\_insert)

                .catch((err) => console.error(err));

        })

        .then(() => {

            conn.promise()

                .query('SELECT \* FROM books')

                .then(([data]) => {

                    res.render('index.hbs', {

                        books: data

                    });

                })

        .then(conn.end())

        .catch((err) => console.error(err));

        })

        .catch((err) => console.error(err));

})

Здесь сначала выполняется SQL запрос на удаление данных из таблицы, а затем в таблицу записываются изначальные значения. После всех преобразований таблица выводится на главный экран.

Все методы были рассмотрены, осталась только таблица стилей, отвечающая за внешний вид web-приложения. Цвета подобраны так, чтобы было интуитивно понятно, что является кликабельным элементом (кнопкой), также для этого добавлен псевдокласс :hover, который преобразует css-код элемента страницы при наведении на него курсором. Например при наведении на кнопку, она меняет свой цвет (рисунок 10), что говорит пользователю о том, что элемент кликабелен.



Рисунок 10 - Пользователь навел курсором на кнопку "Добавить книгу"

# Заключение

В ходе выполнения данной работы было создано web-приложение по работе с базой данных MySQL.

А именно использование данного web-приложения позволяет:

1. Добавлять, изменять и удалять записи в таблице.

2. Полностью очищать и восстанавливать таблицу.

3. Сортировать записи по конкретному полю и направлению сортировки.

На данном этапе развития функционал приложения логически завершён. В случае же продолжения разработки в сторону расширения функционала следует добавить возможность сортировки по всем полям и возможность поиска конкретной строки таблицы по ключевому слову.

Все задачи выполнены, цель достигнута.

# Список используемых источников

1. Справочник по HTML [Электронный ресурс]. – ссылка http://htmlbook.ru/html
2. Справочник по CSS [Электронный ресурс]. – ссылка http://htmlbook.ru/css
3. Express documentation — DevDocs [Электронный ресурс]. – ссылка https://devdocs.io/express/
4. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. 2-е издание [Текст] Браун Итан – 2021.
5. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 [Текст] Робин Никсон – 2018.
6. MySQL, HTML5 и CSS 3. Разработка современных динамических Web-сайтов [Текст] Владимир Дронов – 2016.

# Приложение

**SQL-запрос на создание таблицы:**

CREATE TABLE `soft0069\_labrab06`.`books` ( `id` INT NOT NULL AUTO\_INCREMENT , `title` VARCHAR(80) NOT NULL , `author` VARCHAR(60) NOT NULL , `genre` VARCHAR(30) NOT NULL , `publicationDate` YEAR(4) NOT NULL , `rating` FLOAT NOT NULL , PRIMARY KEY (`id`)) ENGINE = InnoDB;

**SQL-запрос на заполнение таблицы:**

INSERT INTO `books`(`id`, `title`, `author`, `genre`, `publicationDate`, `rating`) VALUES

(NULL,'1984','Джордж Оруэлл','Фантастика',1949,8),

(NULL,'Мастер и Маргарита','Михаил Булгаков','Мистика',1967,9),

(NULL,'Три товарища','Эрих Мария Ремарк','Зарубежная классика',1936,7),

(NULL,'Портрет Дориана Грея','Оскар Уайльд','Фантастика',1890,7),

(NULL,'Маленький принц','Джордж Оруэлл','Зарубежная классика',1943,7),

(NULL,'Над пропастью во ржи','Джером Д. Сэлинджер','Зарубежная классика',1951,10),

(NULL,'Вино из одуванчиков','Рей Брэдбери','Зарубежная классика',1957,6),

(NULL,'Убить пересмешника','Харпер Ли','Зарубежная классика',1960,9),

(NULL,'Преступление и наказание','Федор Достоевский','Русская классика',1866,8),

(NULL,'Хлеб с ветчиной','Чарльз Буковски','Современная зарубежная литература',1982,10),

(NULL,'Женщины','Чарльз Буковски','Современная зарубежная литература',1978,8),

(NULL,'Идиот','Фёдор Достоевский','Фантастика',1869,7);

**app.js:**

const mysql = require("mysql2");

const express = require("express");

const paramsDB = {

    host: "pgsha.ru",

    port: "35006",

    user: "soft0069",

    password: "y61cf4vJ",

    database: "soft0069\_labrab06"

};

const pool = mysql.createPool(paramsDB);

function get\_connection() {

    return mysql.createConnection(paramsDB);

}

const app = express();

const urlencodedParser = express.urlencoded({extended: false});

app.use('/css', express.static(\_\_dirname + '/css'));

app.set("view engine", "hbs");

app.get("/", function(req, res) {

    let query = "SELECT \* FROM books";

    pool.query(query, function(err, data) {

        if (err) return console.log(err);

        res.render("index.hbs", {

            books: data

        });

    });

});

app.get("/create", function(req, res) {

    res.render("create.hbs");

});

app.post("/create", urlencodedParser, function (req, res) {

    if (!req.body) return res.sendpublicationDate(400);

    const title = req.body.title;

    const author = req.body.author;

    const genre = req.body.genre;

    const publicationDate = req.body.publicationDate;

    const rating = req.body.rating;

    let query = "INSERT INTO books (title, author, genre, publicationDate, rating) VALUES (?,?,?,?,?)";

    let params = [title, author, genre, publicationDate, rating];

    pool.query(query, params, function(err, data) {

        if (err) return console.error(err);

        res.redirect("/");

    });

});

app.get("/edit/:id", function(req, res) {

    const id = req.params.id;

    pool.query("SELECT \* FROM books WHERE id=?", [id], function(err, data) {

        if (err) return console.error(err);

        res.render("edit.hbs", {

            books: data[0]

        });

    });

});

app.post("/edit", urlencodedParser, function (req, res) {

    if (!req.body) return res.sendpublicationDate(400);

    const id = req.body.id;

    const title = req.body.title;

    const author = req.body.author;

    const genre = req.body.genre;

    const publicationDate = req.body.publicationDate;

    const rating = req.body.rating;

    let query = "UPDATE books SET title=?, author=?, genre=?, publicationDate=?, genre=? WHERE id=?";

    let params = [title, author, genre, publicationDate, genre, id];

    pool.query(query, params, function(err, data) {

        if (err) return console.error(err);

        res.redirect("/");

    });

});

app.post("/delete/:id", function(req, res) {

    const id = req.params.id;

    pool.query("DELETE FROM books WHERE id=?", [id], function(err, data) {

        if (err) return console.log(err);

        res.redirect("/");

    });

});

app.get("/sort/:field.:direct", function(req, res) {

    const field = req.params.field;

    const direct = req.params.direct;

    let query = "SELECT \* FROM books ORDER BY " + field + " " + direct;

    pool.query(query, function(err, data) {

        if (err) return console.log(err);

        res.render("index.hbs", {

            books: data

        });

    });

});

app.get("/recover", function(req, res) {

    let query\_truncate = "TRUNCATE `books`";

    let query\_insert = "INSERT INTO `books` \

    (`id`, `title`, `author`, `genre`, `publicationDate`, `rating`) VALUES \

    (NULL,'1984','Джордж Оруэлл','Фантастика',1949,8), \

    (NULL,'Мастер и Маргарита','Михаил Булгаков','Мистика',1967,9), \

    (NULL,'Три товарища','Эрих Мария Ремарк','Зарубежная классика',1936,7), \

    (NULL,'Портрет Дориана Грея','Оскар Уайльд','Фантастика',1890,7), \

    (NULL,'Маленький принц','Джордж Оруэлл','Зарубежная классика',1943,7), \

    (NULL,'Над пропастью во ржи','Джером Д. Сэлинджер','Зарубежная классика',1951,10), \

    (NULL,'Вино из одуванчиков','Рей Брэдбери','Зарубежная классика',1957,6), \

    (NULL,'Убить пересмешника','Харпер Ли','Зарубежная классика',1960,9), \

    (NULL,'Преступление и наказание','Федор Достоевский','Русская классика',1866,8), \

    (NULL,'Хлеб с ветчиной','Чарльз Буковски','Современная зарубежная литература',1982,10), \

    (NULL,'Женщины','Чарльз Буковски','Современная зарубежная литература',1978,8), \

    (NULL,'Идиот','Фёдор Достоевский','Фантастика',1869,7);";

    const conn = get\_connection();

    conn.promise()

        .query(query\_truncate)

        .then(() => {

            conn.promise()

                .query(query\_insert)

                .catch((err) => console.error(err));

        })

        .then(() => {

            conn.promise()

                .query('SELECT \* FROM books')

                .then(([data]) => {

                    res.render('index.hbs', {

                        books: data

                    });

                })

        .then(conn.end())

        .catch((err) => console.error(err));

        })

        .catch((err) => console.error(err));

})

app.get("/clear", function(req, res) {

    let query\_truncate = "TRUNCATE books";

    const conn = get\_connection();

    conn.promise()

    .query(query\_truncate)

    .then(() => {

        conn.promise()

            .query('SELECT \* FROM books')

            .then(([data]) => {

                res.render('index.hbs', {

                    books: data

                });

            })

            .then(conn.end())

            .catch((err) => console.error(err));

    })

    .catch((err) => console.error(err));

});

app.listen(3000, function() {

    console.log("смотрим работу через браузер - http://localhost:3000/");

    let isWin = process.platform === "win32";

    let hotKeys = isWin? "Ctrl+C": "Ctrl+D";

    console.log(`остановить сервер - ${hotKeys}`);

});

//Выполнила: Лыкасова Елизавета ПИНб-31

**style.css:**

body {

    background-color: rgb(255, 255, 255);

    font-family: Arial, Helvetica, Verdana, sans-serif;

    color:rgb(0, 0, 0);

}

table {

    margin-top: 20px;

    border-collapse: collapse;

    width: 60%;

}

th, td {

    text-align: center;

    padding: 4px;

}

th {

    background: rgb(255, 193, 142);

    height: 32px;

}

td {

    border-top: 1px solid rgb(255, 192, 141);

    background: #ffdfd2;

}

table tr th a{

    font-size: 8pt;

    color:rgb(0, 0, 0);

}

table tr th a:hover {

    font-size: 9pt;

}

td.editor {

    border-right: 1px solid rgb(255, 193, 142);

}

input[type=submit] {

    color: rgb(255, 255, 255);

    background: #ff7439;

    border: 0;

    border-radius: 5px;

}

input[type=submit]:hover {

    background: #e0571c;

}

a {

    text-decoration: none;

    color: rgb(255, 255, 255);

    font-weight: 400;

}

.upper {

    background-color: #fff;

    border-bottom: 1px solid rgb(150, 150, 150);

}

.btn {

    font-style: 20px;

    font-family: Arial, Helvetica, Verdana, sans-serif;

    color: rgb(255, 255, 255);

    background: #FF4C00;

    text-decoration: none;

    padding: 10px 15px;

    border-radius: 5px;

    background-size: 100%;

}

.btn:hover {

    background: #e24400;

}

**index.hbs:**

<!DOCTYPE html>

<html>

<head>

    <title>Список книг</title>

    <meta charset="utf-8" />

    <link rel="shortcut icon" href="../ico/favicon.ico" type="image/x-icon">

    <link rel="stylesheet" type="text/css" href="/css/style.css" />

</head>

<body>

    <center>

    <h3>Список прочитанных книг</h3>

    <br>

    <a href="/create" class="btn">Добавить книгу</a>

    <a href="/recover" class="btn">Восстановить список</a>

    <a href="/clear" class="btn">Очистить список</a>

    <table>

        <tr>

            <th>Редактор</th>

            <th>Название</th>

            <th>Автор&nbsp;<a href="/sort/author.ASC">▲</a><a href="/sort/author.DESC">▼</a></th>

            <th>Жанр</th>

            <th>Дата публикации&nbsp;<a href="/sort/publicationDate.ASC">▲</a><a href="/sort/publicationDate.DESC">▼</a></th>

            <th>Оценка&nbsp;<a href="/sort/rating.ASC">▲</a><a href="/sort/rating.DESC">▼</a></th>

        </tr>

        {{#each books}}

            <tr>

                <td width=180 class="editor">

                    <form action="/edit/{{this.id}}" method="GET" style="display:inline;">

                        <input type="submit" value="Изменить" />

                    </form>

                    <form action="/delete/{{this.id}}" method="POST" style="display:inline;">

                        <input type="submit" value="Удалить" />

                    </form>

                </td>

                <td>{{this.title}}</td>

                <td>{{this.author}}</td>

                <td>{{this.genre}}</td>

                <td>{{this.publicationDate}}</td>

                <td>{{this.rating}}</td>

            </tr>

        {{/each}}

    </table>

    </center>

</body>

<html>

**create.hbs:**

<!DOCTYPE html>

<html>

<head>

    <title>Добавить книгу</title>

    <meta charset="utf-8" />

    <link rel="shortcut icon" href="../ico/favicon.ico" type="image/x-icon">

    <link rel="stylesheet" type="text/css" href="/css/style.css" />

    <style>.field{width:200px;}</style>

</head>

<body>

    <h1>Добавить книгу</h1>

    <form method="POST">

        <label>Название</label><br>

        <input name="title" class="field" /><br><br>

        <label>Автор</label><br>

        <input name="author" class="field" /><br><br>

        <label>Жанр</label><br>

        <input name="genre" class="field" /><br><br>

        <label>Дата публикации</label><br>

        <input name="publicationDate" type="number" class="field" min="100" max="2025" /><br><br>

        <label>Оценка</label><br>

        <input name="rating" type="number" min="1" max="10" step="1" class="field" /><br><br>

        <input type="submit" value="Сохранить" />

    </form>

    <br>

    <a href="/">На главную</a>

</body>

<html>

**edit.hbs:**

<!DOCTYPE html>

<html>

<head>

    <title>Изменить данные</title>

    <meta charset="utf-8" />

    <link rel="shortcut icon" href="../ico/favicon.ico" type="image/x-icon">

    <link rel="stylesheet" type="text/css" href="/css/style.css" />

    <style>.field{width:200px;}</style>

</head>

<body>

    <h1>Изменить данные</h1>

    <form action="/edit" method="POST">

        <input type="hidden" name="id" value="{{books.id}}" />

        <label>Название</label><br>

        <input name="title" class="field" value="{{books.title}}" /><br><br>

        <label>Автор</label><br>

        <input name="author" class="field" value="{{books.author}}" /><br><br>

        <label>Жанр</label><br>

        <input name="genre" class="field" value="{{books.genre}}" /><br><br>

        <label>Дата публикации</label><br>

        <input name="publicationDate" type="number" class="field" min="100" max="2025" value="{{books.publicationDate}}" /><br><br>

        <label>Оценка</label><br>

        <input name="rating" type="number" class="field" min="1" max="10" step="1" value="{{books.rating}}" /><br><br>

        <input type="submit" value="Сохранить" />

    </form>

    <br>

    <a href="/">На главную</a>

</body>

<html>

РЕЦЕНЗИЯ

на курсовой проект \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич