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-1748-2018

Орлов Максим

Проверил:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

Пермь – 2021

Оглавление

[Введение 3](#_Toc88506531)

[Постановка задачи на проектирование 3](#_Toc88506532)

[Разработка структуры базы данных 5](#_Toc88506533)

[Разработка приложения 7](#_Toc88506534)

[Заключение 19](#_Toc88506535)

[Список используемых источников 20](#_Toc88506536)

[Приложение 21](#_Toc88506537)

# Введение

Мы даже не задумываемся, насколько важна и значительна роль музыки для каждого из нас. Доказано, что они оказывают благотворное влияние на наше развитие. В различных случаях музыка влияет на эмоции, чувства, психику человека, улучшает производительность работы, успокаивает, снимает усталость при переутомлении, расслабляет, стрессом, депрессией, способна даже поднимать настроение, когда становится тоскливо.

# Постановка задачи на проектирование

У пользователя есть список прослушанных музыкальных альбомов и он хочет вести список, для возможности делиться своим списком с другими людьми или для быстрого поиска своих любимых альбомов. Для того чтобы автоматизировать процесс добавления, редактирования и удаления альбомов, 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 музыкальных альбомов, прослушанных пользователем.

Предметная область: прослушанные музыкальные альбомы.

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 web-приложение.

Для достижения цели и выполнения задач были выбраны следующие технологии:

**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.

**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.

# Разработка структуры базы данных

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



Рисунок - Структура базы данных

Для достижения поставленных целей понадобится лишь одна таблица, создадим ее также в phpmyadmin, легче всего это сделать с помощью удобного пользовательского интерфейса. Добавим поля id, author, name, releaseYear, songsCount и rating. Поле id сделаем автоинкрементным для автоматической нумерации записей и присвоим ему первичный ключ. Структура таблицы albums представлена на рисунке 2.

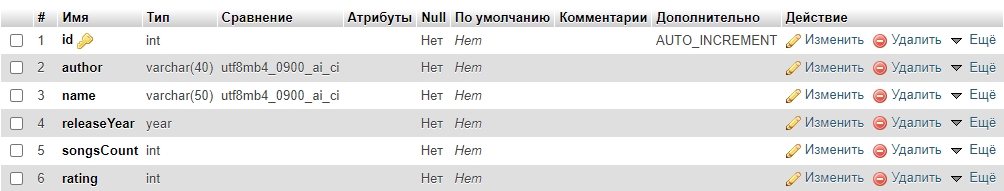


Рисунок - Структура таблицы albums

Как видно на рисунке 2, поля id, songsCount и rating имеют числовой тип данных (int), поля author и name текстовый (varchar), а поле releaseYear тип данных year.

Заполним таблицу данными, используя SQL запрос, текст запроса представлен в листинге №1.

Листинг №1 – SQL запрос для добавления строк в таблицу

INSERT INTO `albums`(`id`, `author`, `name`, `releaseYear`, `songsCount`, `rating`) VALUES

(NULL,'pyrokinesis','Моя милая пустота',2019,17,9),

(NULL,'pyrokinesis','Питер, чай, не Франция',2020,12,8),

(NULL,'pyrokinesis','Корми демонов по расписанию',2018,8,8),

(NULL,'pyrokinesis','Терновый венец эволюции',2017,10,7),

(NULL,'pyrokinesis','Eclipse',2016,9,6),

(NULL,'Слава КПСС','Русское поле',2016,7,7),

(NULL,'Radiohead','OK Computer',1997,12,7),

(NULL,'System Of A Down','Toxicity',2001,15,8),

(NULL,'System Of A Down','Mezmerize',2005,11,7),

(NULL,'System Of A Down','Hypnotize',2005,13,7),

(NULL,'Hawaiian Sadness','Дни дождей',2019,8,7)

Заполненная таблица albums представлена на рисунке 3.



Рисунок 3 - Заполненная таблица albums

# Разработка приложения

Реализация вывода данных на экран будет вестись с помощью веб-фреймворка Express и движка Handlebars, работа с базой данных с помощью Node.js.

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. Для работы был выбран 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 нужные пакеты в консоль была введена следующая команда:

npm install express mysql2 hbs

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

Структура файлов приложения представлена на рисунке 4.

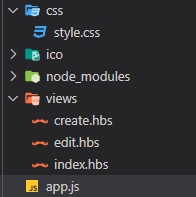


Рисунок 4 - Структура файлов web-приложения

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. После того как данные введены пользователем, запускается метод post, отправляющий данные в базе.

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: "soft0055",

    password: "TdAfEaYR",

    database: "soft0055\_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 albums";

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

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

        res.render("index.hbs", {

            albums: data

        });

    });

});

Здесь посредством запроса SELECT из таблицы albums 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 альбомов</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 любимых альбомов</h3>

    <br>

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

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

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

    <table>

        <tr>

            <th>Редактор</th>

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

            <th>Название</th>

            <th>Год релиза&nbsp;<a href="/sort/releaseYear.ASC">▲</a><a href="/sort/releaseYear.DESC">▼</a></th>

            <th>Количество песен</th>

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

        </tr>

        {{#each albums}}

            <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.author}}</td>

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

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

                <td>{{this.songsCount}}</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 запросов по эндпоинтам - /delete, /recover и /clean, для удаления записи, восстановления таблицы и очистки базы данных.

Запустим сервер, чтобы посмотреть что из себя представляет приложение.

Внешний вид представления index.hbs представлен на рисунке 5.



Рисунок 5 - Внешний вид index.hbs

Для того чтобы добавить запись нужно нажать на кнопку «Добавить альбом» 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>

    <div class="wrapper">

        <div class="content">

            <h1>Добавить альбом</h1>

            <form method="POST">

                <label>Исполнитель</label><br>

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

                <label>Название</label><br>

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

                <label>Год релиза</label><br>

                <input name="releaseYear" min="1800" max="2050" step="1" type="number" class="field" /><br><br>

                <label>Количество песен</label><br>

                <input name="songsCount" type="number" class="field" min="1" max="101" step="1" /><br><br>

                <label>Оценка</label><br>

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

                <input class="btn\_main btn\_save" type="submit" value="Сохранить" />

            </form>

            <br>

            <a class="btn\_main" href="/">На главную</a>

        </div>

    </div>

</body>

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

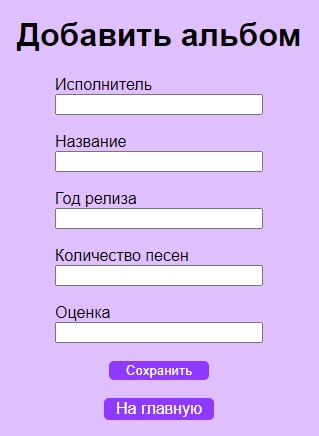


Рисунок 6 - Внешний вид create.hbs

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.sendStatus(400);

    const author = req.body.author;

    const name = req.body.name;

    const releaseYear = req.body.releaseYear;

    const songsCount = req.body.songsCount;

    const rating = req.body.rating;

    let query = "INSERT INTO albums (author, name, releaseYear, songsCount, rating) VALUES (?,?,?,?,?)";

    let params = [author, name, releaseYear, songsCount, rating];

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

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

        res.redirect("/");

    });

});

Здесь запрос имеет параметры, которые сначала извлекаются из формы create, после чего интегрируются в запрос.

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 albums WHERE id=?", [id], function(err, data) {

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

        res.render("edit.hbs", {

            albums: data[0]

        });

    });

});

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

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

    const id = req.body.id;

    const author = req.body.author;

    const name = req.body.name;

    const releaseYear = req.body.releaseYear;

    const songsCount = req.body.songsCount;

    const rating = req.body.rating;

    let query = "UPDATE albums SET author=?, name=?, releaseYear=?, songsCount=?, rating=? WHERE id=?";

    let params = [author, name, releaseYear, songsCount, rating, id];

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

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

        res.redirect("/");

    });

});

Форма edit.hbs представлена на рисунке 7.

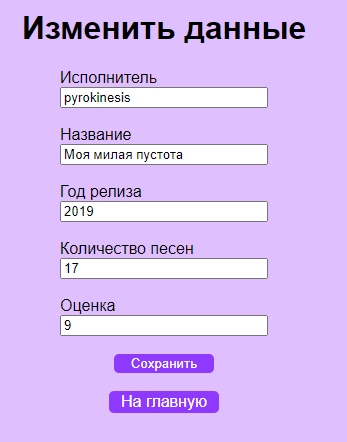


Рисунок 7 - Внешний вид edit.hbs

Для того чтобы удалить запись из таблицы нужно нажать на кнопку «Удалить». Здесь используется только метод post, так как выводить какое-либо представление пользователю не нужно, требуется 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 albums WHERE id=?", [id], function(err, data) {

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

        res.redirect("/");

    });

});

После удаления записи, нас перенаправляет на главную страницу web-приложения.

Сортировка наоборот реализована лишь методом get, ведь отправлять данные в базу не требуется, вместо этого хватает лишь отобразить изменения на главной странице web-приложения.

Сортировка записей в таблице происходит по полям releaseYear и rating для упорядочивания записей по году релиза и оценке, а также по полю author, для группировки альбомов одного исполнителя. Для того чтобы выполнить сортировку, нужно нажать на одну из стрелочек около названий столбцов таблицы.

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 albums ORDER BY " + field + " " + direct;

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

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

        res.render("index.hbs", {

            albums: data

        });

    });

});

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



Рисунок 8 - Сортировка по полю Год релиза, по убыванию

Далее рассмотрим методы для очистки и восстановления таблицы. Начнем с отчистки. 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 albums";

    const conn = get\_connection();

    conn.promise()

    .query(query\_truncate)

    .then(() => {

        conn.promise()

            .query('SELECT \* FROM albums')

            .then(([data]) => {

                res.render('index.hbs', {

                    albums: data

                });

            })

            .then(conn.end())

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

    })

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

});

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

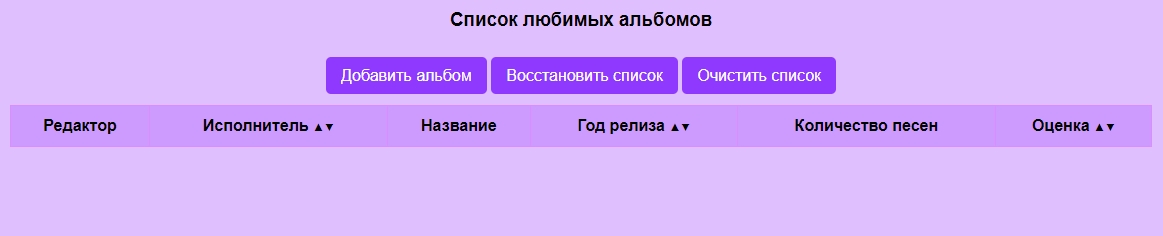


Рисунок 9 - Результат очистки таблицы

И, пожалуй, самым сложным методом в данном web-приложении является восстановление таблицы. Для восстановления списка альбомов 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("/restore", function(req, res) {

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

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

    (`id`, `author`, `name`, `releaseYear`, `songsCount`, `rating`) VALUES \

    (NULL,'pyrokinesis','Моя милая пустота',2019,17,9), \

    (NULL,'pyrokinesis','Питер, чай, не Франция',2020,12,8), \

    (NULL,'pyrokinesis','Корми демонов по расписанию',2018,8,8), \

    (NULL,'pyrokinesis','Терновый венец эволюции',2017,10,7), \

    (NULL,'pyrokinesis','Eclipse',2016,9,6), \

    (NULL,'Слава КПСС','Русское поле',2016,7,7), \

    (NULL,'Radiohead','OK Computer',1997,12,7), \

    (NULL,'System Of A Down','Toxicity',2001,15,8), \

    (NULL,'System Of A Down','Mezmerize',2005,11,7), \

    (NULL,'System Of A Down','Hypnotize',2005,13,7), \

    (NULL,'Hawaiian Sadness','Дни дождей',2019,8,7);";

    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 albums')

                .then(([data]) => {

                    res.render('index.hbs', {

                        albums: 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 и представления 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 "Очистить список"

# 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, поле для обложки альбома 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 `soft0055\_labrab06`.`albums` ( `id` INT NOT NULL AUTO\_INCREMENT , `author` VARCHAR(40) NOT NULL , `name` VARCHAR(50) NOT NULL , `releaseYear` YEAR NOT NULL , `songsCount` INT NOT NULL , `rating` INT NOT NULL , PRIMARY KEY (`id`)) ENGINE = InnoDB;

**SQL-запрос на заполнение таблицы:**

INSERT INTO `albums`(`id`, `author`, `name`, `releaseYear`, `songsCount`, `rating`) VALUES

(NULL,'pyrokinesis','Моя милая пустота',2019,17,9),

(NULL,'pyrokinesis','Питер, чай, не Франция',2020,12,8),

(NULL,'pyrokinesis','Корми демонов по расписанию',2018,8,8),

(NULL,'pyrokinesis','Терновый венец эволюции',2017,10,7),

(NULL,'pyrokinesis','Eclipse',2016,9,6),

(NULL,'Слава КПСС','Русское поле',2016,7,7),

(NULL,'Radiohead','OK Computer',1997,12,7),

(NULL,'System Of A Down','Toxicity',2001,15,8),

(NULL,'System Of A Down','Mezmerize',2005,11,7),

(NULL,'System Of A Down','Hypnotize',2005,13,7),

(NULL,'Hawaiian Sadness','Дни дождей',2019,8,7)

**app.js:**

const mysql = require("mysql2");

const express = require("express");

const paramsDB = {

    host: "pgsha.ru",

    port: "35006",

    user: "soft0055",

    password: "TdAfEaYR",

    database: "soft0055\_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 albums";

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

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

        res.render("index.hbs", {

            albums: data

        });

    });

});

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

    res.render("create.hbs");

});

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

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

    const author = req.body.author;

    const name = req.body.name;

    const releaseYear = req.body.releaseYear;

    const songsCount = req.body.songsCount;

    const rating = req.body.rating;

    let query = "INSERT INTO albums (author, name, releaseYear, songsCount, rating) VALUES (?,?,?,?,?)";

    let params = [author, name, releaseYear, songsCount, 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 albums WHERE id=?", [id], function(err, data) {

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

        res.render("edit.hbs", {

            albums: data[0]

        });

    });

});

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

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

    const id = req.body.id;

    const author = req.body.author;

    const name = req.body.name;

    const releaseYear = req.body.releaseYear;

    const songsCount = req.body.songsCount;

    const rating = req.body.rating;

    let query = "UPDATE albums SET author=?, name=?, releaseYear=?, songsCount=?, rating=? WHERE id=?";

    let params = [author, name, releaseYear, songsCount, rating, 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 albums 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 albums ORDER BY " + field + " " + direct;

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

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

        res.render("index.hbs", {

            albums: data

        });

    });

});

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

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

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

    (`id`, `author`, `name`, `releaseYear`, `songsCount`, `rating`) VALUES \

    (NULL,'pyrokinesis','Моя милая пустота',2019,17,9), \

    (NULL,'pyrokinesis','Питер, чай, не Франция',2020,12,8), \

    (NULL,'pyrokinesis','Корми демонов по расписанию',2018,8,8), \

    (NULL,'pyrokinesis','Терновый венец эволюции',2017,10,7), \

    (NULL,'pyrokinesis','Eclipse',2016,9,6), \

    (NULL,'Слава КПСС','Русское поле',2016,7,7), \

    (NULL,'Radiohead','OK Computer',1997,12,7), \

    (NULL,'System Of A Down','Toxicity',2001,15,8), \

    (NULL,'System Of A Down','Mezmerize',2005,11,7), \

    (NULL,'System Of A Down','Hypnotize',2005,13,7), \

    (NULL,'Hawaiian Sadness','Дни дождей',2019,8,7);";

    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 albums')

                .then(([data]) => {

                    res.render('index.hbs', {

                        albums: data

                    });

                })

        .then(conn.end())

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

        })

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

})

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

    let query\_truncate = "TRUNCATE albums";

    const conn = get\_connection();

    conn.promise()

    .query(query\_truncate)

    .then(() => {

        conn.promise()

            .query('SELECT \* FROM albums')

            .then(([data]) => {

                res.render('index.hbs', {

                    albums: 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}`);

});

//Выполнил: Орлов Максим, ПИНб-31

**style.css:**

body {

    background-color: rgb(224, 191, 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(205, 154, 253);

    border: 1px solid rgb(219, 141, 255);

    height: 32px;

}

td {

    border-top: 1px solid rgb(219, 141, 255);

    background: #ddadfd;

}

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(219, 141, 255);

}

.field {

    width:200px;

}

input[type=submit] {

    width: 100px;

    color: rgb(255, 255, 255);

    background: #8f39ff;

    border: 0;

    border-radius: 5px;

    cursor: pointer;

}

input[type=submit]:hover {

    background: #a21ce0;

    cursor: pointer;

}

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: #8f39ff;

    text-decoration: none;

    padding: 10px 15px;

    border-radius: 5px;

    background-size: 100%;

}

.btn:hover {

    background: #a21ce0;

}

.btn\_main {

    width: 100px;

    text-decoration: none;

    color: rgb(255, 255, 255);

    background: #8f39ff;

    padding: 2px 5px;

    border-radius: 5px;

    text-align: center;

}

.btn\_main:hover {

    background: #a21ce0;

}

.btn\_save {

    margin-left: 54px;

}

.wrapper {

    display: flex;

    justify-content: center;

}

.content {

    display: flex;

    flex-direction: column;

    align-items: center;

}

**index.hbs:**

<!DOCTYPE html>

<html>

<head>

    <title>Список альбомов</title>

    <meta charset="utf-8" />

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

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

</head>

<body>

    <center>

    <h3>Список любимых альбомов</h3>

    <br>

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

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

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

    <table>

        <tr>

            <th>Редактор</th>

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

            <th>Название</th>

            <th>Год релиза&nbsp;<a href="/sort/releaseYear.ASC">▲</a><a href="/sort/releaseYear.DESC">▼</a></th>

            <th>Количество песен</th>

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

        </tr>

        {{#each albums}}

            <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.author}}</td>

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

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

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

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

            </tr>

        {{/each}}

    </table>

    </center>

</body>

<html>

**create.hbs:**

<!DOCTYPE html>

<html>

<head>

    <title>Добавить альбом</title>

    <meta charset="utf-8" />

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

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

</head>

<body>

    <div class="wrapper">

        <div class="content">

            <h1>Добавить альбом</h1>

            <form method="POST">

                <label>Исполнитель</label><br>

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

                <label>Название</label><br>

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

                <label>Год релиза</label><br>

                <input name="releaseYear" min="1800" max="2050" step="1" type="number" class="field" /><br><br>

                <label>Количество песен</label><br>

                <input name="songsCount" type="number" class="field" min="1" max="101" step="1" /><br><br>

                <label>Оценка</label><br>

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

                <input class="btn\_main btn\_save" type="submit" value="Сохранить" />

            </form>

            <br>

            <a class="btn\_main" href="/">На главную</a>

        </div>

    </div>

</body>

<html>

**edit.hbs:**

<!DOCTYPE html>

<html>

<head>

    <title>Изменить данные</title>

    <meta charset="utf-8" />

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

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

</head>

<body>

    <div class="wrapper">

        <div class="content">

            <h1>Изменить данные</h1>

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

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

                <label>Исполнитель</label><br>

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

                <label>Название</label><br>

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

                <label>Год релиза</label><br>

                <input name="releaseYear" type="number" class="field" value="{{albums.releaseYear}}" /><br><br>

                <label>Количество песен</label><br>

                <input name="songsCount" type="number" class="field" min="1" max="101" step="1" value="{{albums.songsCount}}" /><br><br>

                <label>Оценка</label><br>

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

                <input class="btn\_main btn\_save" type="submit" value="Сохранить" />

            </form>

            <br>

            <a class="btn\_main" href="/">На главную</a>

        </div>

    </div>

</body>

<html>

РЕЦЕНЗИЯ

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

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

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич