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 на node.js

Выполнил:

Студент 3 курса очного отделения

Специальности 09.03.04 Программная инженерия

Группы ПиНб-31

Константинов Даниил Константинович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Проверил:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Пермь – 2021

Оглавление

[Раздел 1. Постановка задачи на проектирование 4](#_Toc75999665)

[Раздел 2. Разработка структуры БД 5](#_Toc75999666)

[Раздел 3. Разработка программного решения 6](#_Toc75999667)

[ЗАКЛЮЧЕНИЕ 13](#_Toc75999668)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 14](#_Toc75999669)

[ПРИЛОЖЕНИЕ 1 15](#_Toc75999670)

**Введение**

В современном мире трудно представить себе человека, который не пользовался бы информационными технологиями. С развитием этих технологий потребности людей также становятся более разнообразными, никогда не знаешь, когда тебе пригодится список любимых фильмов или комплектующих для будущего или уже собранного компьютера.

Целью разработки является создание клиент-серверного приложения для хранения и изменения списка комплектующих для ПК.

Актуальность разработки заключается в развитии технологий и росте различных потребностей людей, даже простые задачи необходимо решать. К тому же данное приложение способно быть частью какой-нибудь большой системы.

Основные задачи проекта:

1. Определится с технологиями и уточнить задачу.
2. Разработать структуру БД и выбрать подходящую систему управлениями базами данных (СУБД)
3. Осуществить программную реализацию

# Раздел 1. Постановка задачи на проектирование

Разрабатываемый программный продукт будет решать следующие

задачи:

1. Отображение содержимого привязанной таблицы
2. Добавление новых записей.
3. Удаление отдельных записей
4. Редактирование существующих записей.
5. Полная очистка таблицы
6. Возврат таблицы к самому первому состоянию

Для целей данного курсового проекта в качестве технологий разработки были выбраны следующие средства: Visual Studio Code, СУБД MySQL, node.js, HTML, CSS. Причины выбора именно этих средств:

1. Простота проектирования
2. Visual Studio Code распространяется в бесплатном доступе и поддерживается всеми актуальными операционными системами.
3. VS Code представляет собой обычный текстовый редактор с возможностью подключения различных плагинов, что дает возможность работать со всевозможными языками программирования для разработки любого IT-продукта.
4. Программный продукт может быть запущен на слабых устройствах
5. СУБД MySQL является бесплатной для целей курсового проекта.
6. Выбранная СУБД обладает необходимыми характеристиками быстродействия.

# Раздел 2. Разработка структуры БД

Для целей данного курсового проекта выбрана СУБД MySQL и клиентское приложение для работы с этой СУБД phpmyadmin. Университет предоставил студентам место на собственном сервере для работы с БД, поэтому все дальнейшие манипуляции были проведены на нём.

Для задач, выполняемых программой, подойдет однотабличная структура БД, структура таблицы показана на рисунке 1.



Рисунок 1 – Структура таблицы components

После создания таблицы, поместили 5 столбцов «id», «name», «price», «rating», «maker» поставили соответствующие типы данных. Поле «id» следует установить, как Primary и добавить параметр AUTO\_INCREMENT (чтобы индекс сам увеличивался).

Пояснение по использованным типам данных. Поле id является ключевым, и используется для счетчика записей, поэтому используется integer. Поля name – название и maker – производитель, это строки со словами, цифрами и другими символами, поэтому используется varchar. Поле price – цена может оказаться не целым, поэтому целесообразно использовать тип float. Поле rating – Место в рейтинге простое обозначение позиции типа int вполне достаточно.

Порядок создания БД следующий: Сначала была создана таблица components, а затем при помощи запроса insert она была заполнена.

Листинг запросов на наполнение БД находится в приложении 1 к данному курсовому проекту.

Заполненная таблица показана на рисунке 2.

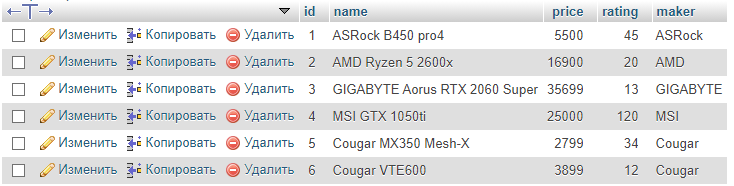


Рисунок 2 – Таблица с данными

# Раздел 3. Разработка программного решения

На предыдущем этапе была подготовлена СУБД, БД была наполнена структурами и данными, далее начата подготовка к созданию программной реализации. Для написания основной части приложения была выбрана программная платформа node.js.

Для работы с MySQL в Node.js можно использовать ряд драйверов. В данном случае было использовано mysql2, он предоставляет большую производительность.

Передаваемые в метод настройки конфигурации могут содержать ряд параметров они представлены на листинге 1.

Наиболее используемые из них:

1. host: хост, на котором запущен сервер mysql.
2. port: номер порта, на котором запущен сервер mysql.
3. user: пользователь MySQL, который используется для подключения.
4. password: пароль для пользователя MySQL
5. database: имя базы данных, к которой идет подключение.

const paramsDB = {

    host: "pgsha.ru",

    port: "35006",

    user: "soft0067",

    password: "J6M6p979",

    database: "soft0067\_labrab06"

};

Листинг 1 – Настройки конфигурации

Для отображения информации на сайте используется html и css, но так как проект выполнена с использованием node.js, имеется возможность использовать шаблоны, hbs файлы. Для каждой страницы был создан свой web-шаблок, список показан на рисунке 3.

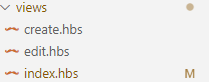


Рисунок 3 – Web-шаблоны

Внешний вид сайта очень простой и показан на рисунке 5. Задний фон – очень светлый оттенок серого, эффект при наведении на ячейки таблицы – выделение их черным цветом. Вся главная страница отрисовывается шаблоном index.hbs – листинг которого в приложении 1.

На главной странице можно выполнить следующие функции:

1. Добавление компонента
2. Очистка таблицы
3. Восстановление первоначальных данных
4. Редактирование записи
5. Удаление записи

Рисунок 4 – Главная форма.

Для получения записей из нашей БД необходимо выполнить код из листинга 2. Отправляется запрос в БД при помощи конструкции pool.query где указываем запрос SQL.

app.get("/", function(req, res) { // получим список абитуриентов

    let query = "SELECT \* FROM components";

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

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

        res.render("index.hbs", {

            components: data

        });

    });

});

Листинг 2 – Отобразить содержимое БД

Для добавления нового компонента необходимо нажать кнопку «Добавить компонент», после чего откроется поле форма добавления компонента – рисунок 5, далее необходимо заполнить все поля и нажать на кнопку «Сохранить».

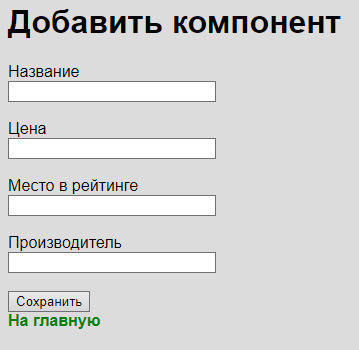


Рисунок 5 – Форма добавления компонента

Страница «Добавить компонент» содержит форму, где пользователь вносит данные, она же в свою очередь отправляет даны на метод /create.

Листинг шаблона create.hbs показан в приложении 1. Код с обращением к БД показан на листинге 3.

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

    res.render("create.hbs");

});

app.post("/create", urlencodedParser, function (req, res) { // сохранить запись в БД

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

    const name = req.body.name;

    const price = req.body.price;

    const rating = req.body.rating;

    const maker = req.body.maker;

    let query = "INSERT INTO components (name, price, rating, maker) VALUES (?,?,?,?)";

    let params = [name, price, rating, maker];

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

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

        res.redirect("/");

    });

});

Листинг 3 – Добавление компонента

Форма редактирования записи показана на рисунке 6. На листинге 4 показан код с обращением к БД, листинг шаблона /edit показан в приложении 1.

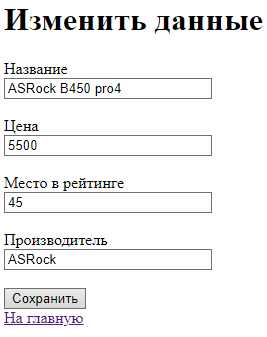


Рисунок 6 – Форма изменения данных

При изменении данных, вызывается метод /edit, в тело которого передается параметр id.

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

    const id = req.params.id;

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

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

        res.render("edit.hbs", {

            component: data[0]

        });

    });

});

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

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

    const id = req.body.id;

    const name = req.body.name;

    const price = req.body.price;

    const rating = req.body.rating;

    const maker = req.body.maker;

    let query = "UPDATE components SET name=?, price=?, rating=?, maker=? WHERE id=?";

    let params = [name, price, rating, maker, id];

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

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

        res.redirect("/");

    });

});

Листинг 4 – Изменение данных

Также имеется возможность сортировки по цене и месту в рейтинге, в порядке возрастания и убывания, что показано на рисунке 7.



Рисунок 7 – Сортировка по цене и рейтингу

Реализация сортировки показана на листингах 5 и 6.

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

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

Листинг 5 – Реализация сортировки в HTML

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

    const field = req.params.field;

    const direct = req.params.direct;

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

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

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

        res.render("index.hbs", {

            abiturs: data

        });

    });

});

Листинг 6 – Реализация сортировки в node.js

При нажатии на кнопку удаление, вызывается серверный метод delete в тело метода передается параметр id.

В первой строчки метода определяется const id, в которую поместили полученную от клиентской части информацию. Далее отправляем запрос в БД при помощи конструкции pool.query где указываем запрос SQL с параметром который передан в квадратных скобках. Сallback функция в случае ошибки выводит в консоль информацию об этой ошибке, либо при помощи res.redirect переправляет на главную страницу, заставляя попутно сервер «отрисовать» данные заново см. листинг 7.

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

    const id = req.params.id;

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

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

        res.redirect("/");

    });

});

Листинг 7 – Реализация удаления записи

Так же есть кнопки «Очистить список» и «Восстановить данные». При очистке списка происходит полное удаление информации с таблицы, вызывается серверный метод /remove, при восстановлении данных, таблица приходит к первоначальному виду, здесь вызывается серверный метод /backup см. рисунок 8, 9, листинг 8, 9.



Рисунок 8 – Очистить таблицу

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

    let query = "TRUNCATE TABLE components";

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

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

        res.redirect("/");

    });

});

Листинг 8 – Очистить таблицу



Рисунок 9 – Вернуть БД в первоначальное состояние

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

    const ctj = require('./utils').csv\_to\_json;

    const get\_conn = require('./utils').get\_conn;

    let array = ctj('./csv/data.csv');

    let inserted\_rows = array.map(item => Object.values(item));

    let query\_truncate = "TRUNCATE components";

    let query\_insert = "INSERT INTO components (name, price, rating, maker) VALUES ? ";

    const conn = get\_conn();

    conn.promise()

        .query(query\_truncate)

        .then(() => {

            conn.promise()

                .query(query\_insert, [inserted\_rows])

                .catch((err) => console.error('ins -> ', err));

        })

        .then(() => {

            conn.promise()

                .query("SELECT \* FROM components")

                .then(([data]) => {

                    res.render("index.hbs", {

                    components: data

                    });

                })

                .then(conn.end())

                .catch((err) => console.error('sel ->', err));

        })

        .catch((err) => console.error('trunc ->', err));

})

Листинг 9 – Вернуть БД к первоначальному состоянию

# ЗАКЛЮЧЕНИЕ

Результатом курсового проекта стал полноценный web-сайт, который решает следующие задачи:

1. Добавление компонента.
2. Восстановление данных БД
3. Очистка все таблицы
4. Взаимодействие со справочниками (отображение, редактирование, удаление, сортировка, поиск)

Были приобретены навыки работы с phpmyadmin. Все поставленные цели достигнуты, задачи выполнены, реализован веб-сайт. Имеется возможность, при желании конечно же, дальнейшего развития.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Бен Хенник, HTML и CSS Путь к совершенству. – СПб.: Изд-во Питер, 2018. – 336 с.;
2. Гольцман В.И. MySQL 5.0. Библиотека программиста. – СПб.: Изд-во Питер, 2017. – 253 с.;
3. Кантелон, М. Node.js в действии / М. Кантелон. - М.: Питер, 2018. - 810 c.
4. Харрис Э. PHP/MySQL для начинающих. – Минск.: Изд-во Кудиц-Образ, 2016. – 384 с.;

Хэррон, Дэвид Node.js Разработка серверных веб-приложений на JavaScript / Дэвид Хэррон. - М.: ДМК Пресс, 2017. - 144 c.

# ПРИЛОЖЕНИЕ 1

INSERT INTO `components`(`name`, `price`, `rating`, `maker`) VALUES

("ASRock B450 pro4", 5500, 45, "ASRock"),

("AMD Ryzen 5 2600x", 16900, 20, "AMD"),

("GIGABYTE Aorus RTX 2060 Super", 35699, 13, "GIGABYTE"),

("MSI GTX 1050ti", 25000, 120, "MSI"),

("Cougar MX350 Mesh-X", 2799, 34, "Cougar"),

("Cougar VTE600", 3899, 12, "Cougar");

**Исходные коды программы**

**Файл create.hbs**

<!DOCTYPE html>

<html>

<head>

    <title>create component</title>

    <meta charset="utf-8" />

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

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

</head>

<body>

    <h1>Добавить компонент</h1>

    <form method="POST">

        <label>Название</label><br>

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

        <label>Цена</label><br>

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

        <label>Место в рейтинге</label><br>

        <input name="rating" type="number" class="field" min="0" max="3000" /><br><br>

        <label>Производитель</label><br>

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

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

    </form>

    <a href="/">На главную</a>

</body>

<html>

**Файл edit.hbs**

<!DOCTYPE html>

<html>

<head>

    <title>edit component</title>

    <meta charset="utf-8" />

    <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="{{component.id}}" />

        <label>Название</label><br>

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

        <label>Цена</label><br>

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

        <label>Место в рейтинге</label><br>

        <input name="rating" type="number" class="field" min="0" max="3000" value="{{component.rating}}" /><br><br>

        <label>Производитель</label><br>

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

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

    </form>

    <a href="/">На главную</a>

</body>

<html>

**Файл index.hbs**

<!DOCTYPE html>

<html>

<head>

    <title>MySQL</title>

    <meta charset="utf-8" />

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

</head>

<body>

    <center>

    <h3>Список компонентов</h1>

    <p><a href="/create">Добавить компонент</a></p>

    <p><a href="/remove">Очистить таблицу</a></p>

    <p><a href="/backup">Вернуть БД к исходному состоянию</a></p>

    <table>

        <tr>

            <th>Редактор</th>

            <th>Название</th>

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

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

            <th>Производитель</th>

        </tr>

        {{#each components}}

            <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.name}}</td>

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

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

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

            </tr>

        {{/each}}

    </table>

    </center>

</body>

<html>

**Файл app.js**

// npm i hbs express mysql2 ИЛИ npm i если есть package.json

const mysql = require("mysql2");

const express = require("express");

const pool = mysql.createPool({

    host: "pgsha.ru",

    port: "35006",

    user: "soft0067",

    password: "J6M6p979",

    database: "soft0067\_labrab06"

});

const app = express();

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

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

app.set("view engine", "hbs");

/\* отобразить абитуриентов \*/

app.get("/", function(req, res) { // получим список абитуриентов

    let query = "SELECT \* FROM components";

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

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

        res.render("index.hbs", {

            components: data

        });

    });

});

/\* добавить нового абитуриента \*/

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

    res.render("create.hbs");

});

app.post("/create", urlencodedParser, function (req, res) { // сохранить запись в БД

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

    const name = req.body.name;

    const price = req.body.price;

    const rating = req.body.rating;

    const maker = req.body.maker;

    let query = "INSERT INTO components (name, price, rating, maker) VALUES (?,?,?,?)";

    let params = [name, price, rating, maker];

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

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

        res.redirect("/");

    });

});

/\* изменить данные абитуриента \*/

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

    const id = req.params.id;

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

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

        res.render("edit.hbs", {

            component: data[0]

        });

    });

});

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

    let query = "TRUNCATE TABLE components";

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

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

        res.redirect("/");

    });

});

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

    const ctj = require('./utils').csv\_to\_json;

    const get\_conn = require('./utils').get\_conn;

    let array = ctj('./csv/data.csv');

    let inserted\_rows = array.map(item => Object.values(item));

    let query\_truncate = "TRUNCATE components";

    let query\_insert = "INSERT INTO components (name, price, rating, maker) VALUES ? ";

    const conn = get\_conn();

    conn.promise()

        .query(query\_truncate)

        .then(() => {

            conn.promise()

                .query(query\_insert, [inserted\_rows])

                .catch((err) => console.error('ins -> ', err));

        })

        .then(() => {

            conn.promise()

                .query("SELECT \* FROM components")

                .then(([data]) => {

                    res.render("index.hbs", {

                    components: data

                    });

                })

                .then(conn.end())

                .catch((err) => console.error('sel ->', err));

        })

        .catch((err) => console.error('trunc ->', err));

})

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

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

    const id = req.body.id;

    const name = req.body.name;

    const price = req.body.price;

    const rating = req.body.rating;

    const maker = req.body.maker;

    let query = "UPDATE components SET name=?, price=?, rating=?, maker=? WHERE id=?";

    let params = [name, price, rating, maker, id];

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

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

        res.redirect("/");

    });

});

/\* удалить запись про абитуриента \*/

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

    const id = req.params.id;

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

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

        res.redirect("/");

    });

});

/\* отсортировать абитуриентов \*/

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

    const field = req.params.field;

    const direct = req.params.direct;

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

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

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

        res.render("index.hbs", {

            abiturs: data

        });

    });

});

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"; // Windows / Linux

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

});

**Файл utils.js**

const fs = require('fs');

const csvjson = require('csvjson');

const mysql = require("mysql2"); // npm i mysql2

const mysqlp = require("mysql2/promise");

const paramsDB = {

    host: "pgsha.ru",

    port: "35006",

    user: "soft0067",

    password: "J6M6p979",

    database: "soft0067\_labrab06"

};

function get\_connection() {

    return mysql.createConnection(paramsDB);

}

function get\_pool() {

    return mysql.createPool(paramsDB);

}

function get\_pool\_p() {

    return mysqlp.createPool(paramsDB);

}

function csv\_to\_json(nameFile) {

    let textCSV = fs.readFileSync(nameFile, 'utf-8');

    return csvjson.toObject(textCSV, { delimiter: ',' });

}

function write\_to\_csv(array, nameFile) {

    let fw = fs.createWriteStream(nameFile);

    fastcsv

        .write(array, { headers: true })

        .pipe(fw);

}

module.exports.get\_conn = get\_connection;

module.exports.get\_pool = get\_pool;

module.exports.get\_pool\_p = get\_pool\_p;

module.exports.csv\_to\_json = csv\_to\_json;

module.exports.write\_to\_csv = write\_to\_csv;

РЕЦЕНЗИЯ

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

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

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич