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 JS»

Выполнила работу:

Студент 3 курса очного отделения

Специальности 09.03.04 Программная инженерия

Шифр ПИНб-1258-2018

Ларионов Александр Сергеевич

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

(подпись)

Проверил работу:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

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

(подпись)

Пермь – 2021

Содержание

[ВВЕДЕНИЕ 3](#_Toc76033583)

[Раздел 1. Постановка задачи на проектирование 4](#_Toc76033584)

[Раздел 2. Разработка структуры СУБД. 5](#_Toc76033585)

[Раздел 3. Разработка программного решения 7](#_Toc76033586)

[ЗАКЛЮЧЕНИЕ 15](#_Toc76033587)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 16](#_Toc76033588)

[ПРИЛОЖЕНИЕ 17](#_Toc76033589)

# ВВЕДЕНИЕ

С каждым годом в культурной сфере современного общества возрастает потребность и, соответственно, ценность художественного телевизионного жанра как сериал. Если рассматривать данный вид художественных произведений как своеобразное зеркало, в котором отражается дух эпохи, то можно сделать вывод, что в это зеркало смотрится очень много людей. Сейчас данный вид телепродукции очень востребован аудиторией из самых различных слоев населения - это даёт повод предположить, что сериалы уже стали в какой-то мере предметом для национальной или другой идентификации личности, частью культурного кода. И в качестве реализации данного курсового проекта была выбрана тема «Web-приложение Список сериалов».

Целью разработки является ведение списка сериалов.

Объектом исследования является список сериалов.

Предметом исследования является сайт «Web-приложение Список сериалов».

Основные задачи проекта:

1. Определиться с технологиями.
2. Разобраться с БД и выбрать подходящую для работы СУБД.
3. Осуществить программную реализацию.

# Раздел 1. Постановка задачи на проектирование

Разрабатываемы программный продукт будет решать следующие задачи:

1. Добавление нового сериала.
2. Очистка всего списка сериалов.
3. Редактирование, удаление, сортировка.

Для целей данного курсового проекта было выбрано следующее программное обеспечение: Visual Studio Code, Node.js, СУБД MySQL . Причин для выбора такого программного обеспечения несколько:

Visual Studio Code бесплатный редактор исходного кода, позиционирующий себя как редактор кода для кроссплатформенной разработки web- и облачных приложений с возможностью подключить различные плагины. Node.js фреймворк использующийся для написания серверных JavaScript-приложений. phpMyAdmin – система управления MySQL через web-интерфейс. Выбранная СУБД обеспечивает достаточную скорость обработки запросов, а также удовлетворяет потребностям в индивидуальных подключениях. СУБД MySQL является бесплатной для целей курсового проекта.

# Раздел 2. Разработка структуры СУБД.

Для целей данного курсового проекта была выбрана СУБД MySQL и приложение для работы с СУБД phpMyAdmin . В данной СУБД была разработана БД для хранения и обмена информации.

Для задач, выполняемой программы подойдет простая структура БД.

Таблица показана на рисунке 1, а структура этой таблицы на рисунке 2.

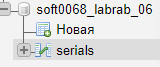


Рисунок 1 – Структура БД

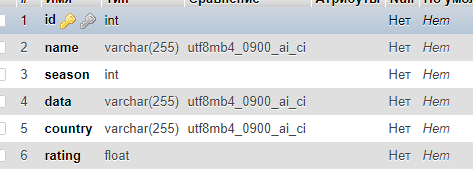


Рисунок 2 – Структура таблицы «Сериалы»

После создания таблицы, создали 6 столбцов «id», «name», «season», «data», «country», «rating». Поле «id» установлено как PRIMARY и имеет AUTO\_INCREMEANT. Столбцы «id», «season», «data» имеют тип данных Int Столбцам «name», «country» был присвоен тип данный Varchar. Столбец «rating» имеет тип данных FLOAT, так как в нем используются десятичные числа.

Далее заполняем таблицу значениями как показано на листинге 1.

INSERT INTO `serials` (name, season, data, country,rating) VALUES('Автатар: Легенда об Аанге',3,2004,'США',8.6);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Гравити фолз',2,2012,'США',8.9);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Друзья',10,1994,'США',9.2);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Рик и Морти',5,2013,'США',8.9);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Клиника',9,2001,'США',8.7);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Теория большого взрыва',12,2007,'США',8.6);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Бесстыжие',11,2011,'США',8.6);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Офис',9,2005,'США',8.3);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Клан Сопрано',6,1999,'США',8.7);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Доктор Хаус',8,2004,'США',8.8);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Во все тяжкие',5,2008,'США',8.9);

Листинг 1 – Заполнение таблицы «Сериалы»



Рисунок 3 – Таблица «Сериалы»

# Раздел 3. Разработка программного решения

На предыдущем этапе была подготовлена БД, заполненная структурами и данными, после начата подготовка к созданию программного решения.

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

Передаваемые в метод настройки конфигурации могут содержать ряд параметров они представлены на листинге 2.

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: "soft0061",

password: "JY9F5Le7",

database: "soft0061\_curs"

};

Листинг 2 –Настройка подключения

Для отображения информации на сайте используют чистый html/сss, но так как работа выполнена Node.js есть возможность использовать web-шаблоны – файлы с расширением hbs под каждую необходимую функциональность был создан собственный web-шаблон, список перечислен на рисунке 4, далее они будет подробно рассмотрены.



Рисунок 4 – Web-шаблоны

Оформление сайта выполнено в простой форме (Рисунок 5). Задний фон был выполнен в светло-зелёном цвете, были добавлены эффекты при наведении на кнопки с жёлтого на темно-жёлтый и изменение цвета текста при наведении на кнопку с чёрного на белый.

На главной странице происходит выполнение следующих функций:

* Добавление сериала;
* Редактирование выбранного сериала;
* Удаление выбранного сериала из списка;
* Сортировка по трём столбцам;
* Сбросить сортировку;
* Очистить таблицу полностью;



Рисунок 5 – Главная страница

Для того чтобы выводить данные из базы данных в программной реализации было сделано следующее (листинг 3). Отправляется запрос в БД при помощи конструкции poll.query где указан запрос SQL. Аналогично это выполнено и для других страниц портала.

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

let query = "SELECT \* FROM serials";

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

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

res.render("index.hbs", {

serials: data

});

});

});

Листинг 3 – Отобразить список сериалов

Для добавления нового сериала в наш список, потребуется нажать кнопку «Добавить сериал», где в последствие происходит заполнение и сохранение данных (рисунок 6).

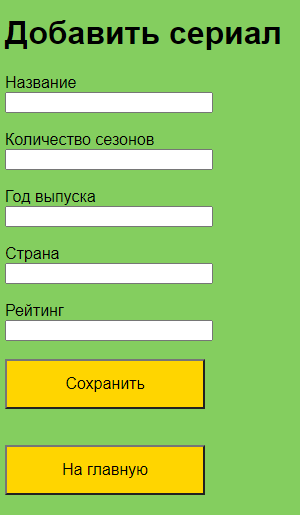


Рисунок 6 – Добавление нового сериала

Страница «Добавить сериал» содержит форму в которой пользователь заносит данные на сервер, метод отправки продемонстрирован на листинге 4, 5.

<!DOCTYPE html>

<html>

<head>

<title>Добавить сериал</title>

<meta charset="utf-8" />

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

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

</head>

<body>

<h1>Добавить сериал</h1>

<form method="POST">

<label>Название</label><br>

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

<label>Количество сезонов</label><br>

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

<label>Год выпуска</label><br>

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

<label>Страна</label><br>

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

<label>Рейтинг</label><br>

<input name="rating" class="field" pattern="\d{1,2}\.\d{0,4}" /><br><br>

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

</form><br><br>

<form action="/" method="GET" style="display: inline;">

<input type="submit" value="На главную" id="sub">

</form>

</body

<html>

Листинг 4 – Web-шаблон «Добавить сериал»

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

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 season = req.body.season;

const data = req.body.data;

const country = req.body.country;

const rating = req.body.rating;

let query = "INSERT INTO serials (name, season, data, country, rating) VALUES (?,?,?,?,?)";

let params = [name, season, data, country, rating];

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

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

res.redirect("/");

});

});

Листинг 5 – Добавить сериал

В данном Web-приложение есть возможность редактировать информацию (рисунок 7).

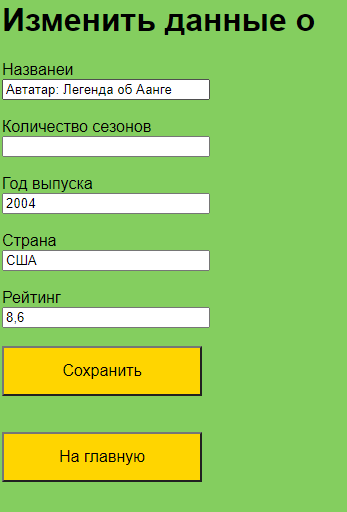


Рисунок 7 – Редактирование информации

В данном случае меняем данные о сериале и сохраняем. При нажатии на кнопку редактировать вызывается метод продемонстрированный на листинге 6.

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

const id = req.params.id;

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

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

res.render("edit.hbs", {

serials: 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.lastname;

const season = req.body.season;

const data = req.body.data;

const country = req.body.country;

const rating = req.body.rating;

let query = "UPDATE serials SET name=?, season=?, data=?, country=?, rating=? WHERE id=?";

let params = [name, season, data, country, rating, id];

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

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

res.redirect("/");

});

});

Листинг 6 – Редактирование данных

Так же существует сортировка по убыванию и возрастанию на трех полях.

<th>Редактор</th>

<th>Название</th>

<th>Количество сезонов&nbsp;<a href="/sort/season.ASC" id="s">▲</a><a href="/sort/season.DESC" id="s">▼</a></th>

<th>Год выпуска&nbsp;<a href="/sort/data.ASC" id="s">▲</a><a href="/sort/data.DESC" id="s">▼</a></th>

<th>Страна</th> <th>Рейтинг&nbsp;<a href="/sort/rating.ASC" id="s">▲</a><a href="/sort/rating.DESC" id="s">▼</a></th>

Листинг 7 – Сортировка полей по убыванию возрастанию

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

const field = req.params.field;

const direct = req.params.direct;

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

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

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

res.render("index.hbs", {

serials: data

});

});

});

Листинг 8 – Метод сортировки

При нажатии на кнопку удалить выбранный элемент вызывается метод продемонстрированный на листинге 9.

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

const id = req.params.id;

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

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

res.redirect("/");

});

});

Листинг 9 – Удаление записи о сериале

На главной странице существует кнопка «очистить список». При нажатии на эту кнопку очищается полностью весь список сериалов, метод приведён на листинге 10.

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

let query\_truncate = "TRUNCATE serials";

const conn = get\_conn();

conn.promise()

.query(query\_truncate)

.then(() => {

conn.promise()

.query('SELECT \* FROM serials')

.then(([data]) => {

res.render('index.hbs', {

serials: data

});

})

.then(conn.end())

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

})

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

});

Листинг 10 – Очистка списка

# ЗАКЛЮЧЕНИЕ

Результатам данного курсового проекта стал полноценное web-приложение, которое решает следующие задачи:

1. Добавление сериалов.
2. Очистка всего списка сериалов.
3. Редактирование, удаление, сортировка.

Были приобретены навыки работы с phpmyadmin. Все поставленные цели достигнуты, задачи выполнены, реализовано веб-приложение. В дальнейшем можно реализовать страницу для каждого сериала с подробной информацией.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 (pdf+epub). – СПб.: Изд-во Питер, 2018. – 816 с.;
2. Кантелон, М. Node.js в действии / М. Кантелон. - М.: Питер, 2018. - 810 c.
3. Дронов, Владимир Александрович Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS / Дронов Владимир Александрович. - М.: БХВ-Петербург, 2018. - 349 c.
4. Роббинс, Дженнифер HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дженнифер Роббинс. - М.: Эксмо, 2018. - 528 c.
5. Гольцман В.И. MySQL 5.0. Библиотека программиста. – СПб.: Изд-во Питер, 2017. – 253 с.;
6. Хэррон, Дэвид Node.js Разработка серверных веб-приложений на JavaScript / Дэвид Хэррон. - М.: ДМК Пресс, 2017. - 144 c.

# ПРИЛОЖЕНИЕ

INSERT INTO `serials` (name, season, data, country,rating) VALUES('Автатар: Легенда об Аанге',3,2004,'США',8.6);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Гравити фолз',2,2012,'США',8.9);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Друзья',10,1994,'США',9.2);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Рик и Морти',5,2013,'США',8.9);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Клиника',9,2001,'США',8.7);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Теория большого взрыва',12,2007,'США',8.6);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Бесстыжие',11,2011,'США',8.6);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Офис',9,2005,'США',8.3);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Клан Сопрано',6,1999,'США',8.7);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Доктор Хаус',8,2004,'США',8.8);

INSERT INTO `serials` (name, season, data, country, rating) VALUES('Во все тяжкие',5,2008,'США',8.9);

**Исходные коды программ**

**Файл index.hbs**

<!DOCTYPE html>

<html>

<head>

<title>Список сериалов</title>

<meta charset="utf-8" />

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

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

</head>

<body>

<center>

<h3>Список сериалов</h1>

<form action="/clear" method="GET" style="display: inline;">

<input type="submit" value="Очистить список" id="sub">

</form>

<form action="/" method="GET" style="display: inline;">

<input type="submit" value="Сбросить сортировку" id="sub">

</form>

<form action="/create" method="GET" style="display: inline;">

<input type="submit" value="Добавить сериал" id="sub">

</form><br><br>

<table>

<tr>

<th>Редактор</th>

<th>Название</th>

<th>Количество сезонов&nbsp;<a href="/sort/season.ASC" id="s">▲</a><a href="/sort/season.DESC" id="s">▼</a></th>

<th>Год выпуска&nbsp;<a href="/sort/data.ASC" id="s">▲</a><a href="/sort/data.DESC" id="s">▼</a></th>

<th>Страна</th>

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

</tr>

{{#each serials}}

<tr>

<td width=180 class="editor">

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

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

</form>

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

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

</form>

</td>

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

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

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

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

<td align = "center">{{this.rating}}</td>

</tr>

{{/each}}

</table>

</center>

</body>

<html>

**Файл edit.hbs**

<!DOCTYPE html>

<html>

<head>

<title>Изменить данные</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 о</h1>

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

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

<label>Названеи</label><br>

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

<label>Количество сезонов</label><br>

<input name="lastName" class="field" value="{{serials.seasons}}" /><br><br>

<label>Год выпуска</label><br>

<input name="city" class="field" value="{{serials.data}}" /><br><br>

<label>Страна</label><br>

<input name="city" class="field" value="{{serials.country}}" /><br><br>

<label>Рейтинг</label><br>

<input name="rating" type="number" class="field" min="0" max="300" value="{{serials.rating}}" /><br><br>

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

</form><br><br>

<form action="/" method="GET" style="display: inline;">

<input type="submit" value="На главную" id="sub">

</form>

</body>

<html>

**Файл create.hbs**

<!DOCTYPE html>

<html>

<head>

<title>Добавить сериал</title>

<meta charset="utf-8" />

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

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

</head>

<body>

<h1>Добавить сериал</h1>

<form method="POST">

<label>Название</label><br>

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

<label>Количество сезонов</label><br>

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

<label>Год выпуска</label><br>

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

<label>Страна</label><br>

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

<label>Рейтинг</label><br>

<input name="rating" class="field" pattern="\d{1,2}\.\d{0,4}" /><br><br>

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

</form><br><br>

<form action="/" method="GET" style="display: inline;">

<input type="submit" value="На главную" id="sub">

</form>

</body>

<html>

**Файл app.js**

const mysql = require("mysql2");

const express = require("express");

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

const pool = mysql.createPool({

host: "pgsha.ru",

port: "35006",

user: "soft0068",

password: "ypHKMRNF",

database: "soft0068\_labrab\_06"

});

const app = express();

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

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

app.set("view engine", "hbs");

/\* отобразить сериалов \*/

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

let query = "SELECT \* FROM serials";

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

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

res.render("index.hbs", {

serials: data

});

});

});

/\* добавить новый сериал \*/

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

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 season = req.body.season;

const data = req.body.data;

const country = req.body.country;

const rating = req.body.rating;

let query = "INSERT INTO serials (name, season, data, country, rating) VALUES (?,?,?,?,?)";

let params = [name, season, data, country, rating];

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

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

res.redirect("/");

});

});

/\* изменить данные сериалу \*/

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

const id = req.params.id;

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

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

res.render("edit.hbs", {

serials: 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.lastname;

const season = req.body.season;

const data = req.body.data;

const country = req.body.country;

const rating = req.body.rating;

let query = "UPDATE serials SET name=?, season=?, data=?, country=?, rating=? WHERE id=?";

let params = [name, season, data, country, rating, id];

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

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

res.redirect("/");

});

});

/\* удалить запись про сериал \*/

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

const id = req.params.id;

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

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

res.redirect("/");

});

});

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

let query\_truncate = "TRUNCATE serials";

const conn = get\_conn();

conn.promise()

.query(query\_truncate)

.then(() => {

conn.promise()

.query('SELECT \* FROM serials')

.then(([data]) => {

res.render('index.hbs', {

serials: data

});

})

.then(conn.end())

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

})

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

});

/\* отсортировать сериалы \*/

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

const field = req.params.field;

const direct = req.params.direct;

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

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

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

res.render("index.hbs", {

serials: 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: "soft0068",

password: "ypHKMRNF",

database: "soft0068\_labrab\_06"

};

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