

**АННОТАЦИЯ**

В данном отчёте описан процесс реализации курсовой работы на тему «Веб-сервис обработки студенческих работ».

Главной целью данной работы являлась разработка веб-сервиса для обработки студенческих работ на антиплагиат по внутренней БД. В задачи данного проекта входили: выбор необходимых функций, дружелюбный интерфейс.

В 1-ой главе представлено описание предварительной подготовки – определение необходимого функционала.

В 2-ой главе представлено выбор и обоснование технологий.

В 3-ей главе представлено описание архитектуры MVC.

В 4-ой главе представлены подробности разработки серверной части.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc121582719)

[ОСНОВНАЯ ЧАСТЬ 6](#_Toc121582720)

[**1.** **Анализ предметной области** 6](#_Toc121582721)

[**1.1.** **Анализ аналогичных веб-сервисов** 6](#_Toc121582722)

[**1.2.** **Определение необходимого функционала** 6](#_Toc121582723)

[**2.** **Выбор и обоснование технологий** 7](#_Toc121582724)

[3. Разработка архитектуры приложения на основе выбранного паттерна 9](#_Toc121582725)

[4. Разработка серверной части интернет-ресурса 10](#_Toc121582726)

[**4.1.** **Структура MySQL** 10](#_Toc121582727)

[**4.2.** **Структура контроллеров** 10](#_Toc121582728)

[**4.3.** **Структура моделей** 15](#_Toc121582729)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 16](#_Toc121582730)

# **ВВЕДЕНИЕ**

Обучающиеся нередко проверяют свои работы на антиплагиат.

Целью данной работы является создание веб-сервиса проверки работ на антиплагиат, которое предоставило бы пользователям удобный интерфейс и неограниченное количество проверок.

Задачи, которые мне предстоит решить для достижения данной цели:

* определение необходимого функционала;
* создание взаимодействий между клиентской частью, серверной и базой данных;
* реализация возможности добавления файлов и удаления старых;
* тестирование и отладка веб-сервиса.

# **ОСНОВНАЯ ЧАСТЬ**

## **Анализ предметной области**

### **Анализ аналогичных веб-сервисов**

Возьмем популярные веб-сервисы обработки студенческих работ с точки зрения антиплагиата.

Веб-сервис “Antiplagiat.ru” [1] является по моему мнению самым популярным для проверки студенческих работ. В нем есть личный кабинет пользователя, в котором он может выбрать удобный для себя тариф и проверять далее разные работы на антиплагиат.

В правом верхнем углу есть кнопка “Меню”, через нее можно перейти в “Профиль”, “Баланс”, “Кабинет”, “Тарифы” и “Служба поддержки”. Это не все функции, но я возьму только важные.

Если поменять тариф на ”Full”, то открываются все возможности данного веб-сервеса, например, такие как неограниченная проверка файлов по времени и снятие ограничения на размер файлов.

### **Определение необходимого функционала**

Главная страница будет автоматически открываться на описание веб-сервиса, а в правом верхнем углу будут кнопки “Регистрация” и “Вход”.

После входа в личный кабинет пользователь автоматически попадает на страницу “Проверить документ”. В правом верхнем угла будет кнопка “Выйти”. Так же на странице проверки документов будут следующие функции:

* Добавить документ на проверку
* Удалить уже проверенный документ
* Выгрузить загруженный файл

У всех проверенных файлов будет стоять название файла, дата проверки и процент оригинальности.

## **Выбор и обоснование технологий**

Для реализации кода моего проекта были использованы: паттерн MVC, язык разметки HTML5, формальный язык описания внешнего вида CSS3, язык программирования JavaScript, база данный MySQL, программной библиотеке NodeJS [2] и фреймворк ReactJS [3].

У паттерна MVC есть модель, контроллер и представление (Рисунок 3.1).

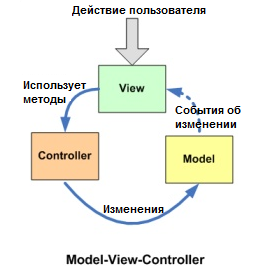


Рисунок 3.1 – Работа паттерна MVC

Основная идея этого паттерна в том, что и контроллер и представление зависят от модели, но модель никак не зависит от этих двух компонент.

Для разрабатываемого приложения важна правильная архитектура серверной части. Клиентская часть не является целью данной работы, кроме того, с учетом подхода к архитектуре, серверная часть не должна зависеть от клиентской. Для клиентской части обычно используется HTML, CSS, JavaScript. Однако при разработке с использованием данного набора сложно производить генерацию большого количества данных. Поэтому было принято решение использовать ReactJS – библиотекy для создания пользовательских интерфейсов на основе UI-компонентов. Использование библиотеки сэкономит время на разработку клиентского представления. Также создание клиентской части с использованием ReactJS позволяет оторвать клиент от сервера – клиент будет создавать представления лишь на основе ответа сервера – преобразуя данные Json-объектов.

Для хранения данных используется MySQL, обеспечивающую масштабируемость, безопасность и повышенную скорость работы.

Для реализации самого проекта потребуется ПК или мобильное устройство, доступ в интернет и прикладное программное обеспечение - браузер.

# **Разработка архитектуры приложения на основе выбранного паттерна**

В моей работе в роли модели выступают пользователи и файлы, которые связаны с БД. В качестве контроллеров у нас методы работы с пользователями и файлами. В качестве представления у нас выступает клиентская часть.

В контроллере пишется весь функционал. Представлением, за что выступает ReactJS отображает пользователям все изменения.

По итогу получилась такая структура (Рисунок 3.1).

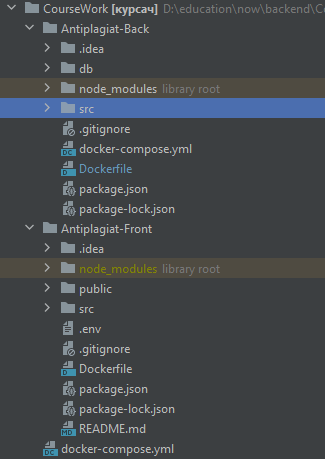


Рисунок 3.1 – Структура архитектуры с паттерном MVC

# **Разработка серверной части интернет-ресурса**

## **Структура MySQL**

В базе данных есть 2 таблицы: пользователи и файлы.

В таблице пользователей есть id, login и password(Листинг 4.1).

Листинг 4.1 – код таблицы пользователей

CREATE TABLE IF NOT EXISTS `mydb`.`users` (

`id` INT NOT NULL AUTO\_INCREMENT,

`login` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

В таблице файлов есть id, user\_id, content, file, filename, overlap, createdAt, login и password(Листинг 4.2).

Листинг 4.2 – код таблицы файлов

CREATE TABLE IF NOT EXISTS `mydb`.`files` (

`id` INT NOT NULL AUTO\_INCREMENT,

`user\_id` INT NOT NULL,

`content` TEXT NOT NULL,

`file` LONGBLOB,

`filename` TEXT,

`overlap` INT,

`createdAt` DATETIME,

PRIMARY KEY (`id`))

## **Структура контроллеров**

Контроллеры содержат в себе весь функционал.

Рассмотрим функционал пользователей:

Листинг 4.2.1 – Метод AuthRoute

export default function authRoute(app, path) {

app.get(`${path}/auth`, async (req, res) => {

const {login, password} = req.query

const user = await User.findOne({

where: {

login, password

}

})

if (!user) {

sendError(res, 404, "Пользователь не найден", "User not found")

return;

}

res.send({token: generateToken(user.get("id"), "user")})

})

}

Данный метод позволяет аутентифицировать пользователя с помощью логина и пароля и создает ему токен, однако если такого пользователя нет, то будет выведена ошибка.

Листинг 4.2.2 – Метод register

export default function authRoute(app, path) {

app.get(`${path}/auth`, async (req, res) => {

const {login, password} = req.query

const user = await User.findOne({

where: {

login, password

}

})

if (!user) {

sendError(res, 404, "Пользователь не найден", "User not found")

return;

}

res.send({token: generateToken(user.get("id"), "user")})

})

}

Данный метод позволяет зарегистрироваться новому пользователю, так же проверяет на наличие пользователя с таким же логином и выдает токен.

Листинг 4.2.3 – Метод getUser

export default function getUser(app, path) {

app.get(`${path}`, async (req, res) => {

const {userId} = app.locals.settings;

const user = await User.findOne({

where: {id: userId}

})

const files = await getUserFiles(app)

res.send({

result: {...JSON.parse(JSON.stringify(user)), password: undefined, files},

})

})

}

Данный метод позволяет получить данные пользователя.

Рассмотрим функционал для файлов:

Листинг 4.2.4 – Метод delete

export function deleteFileRoute(app, path) {

app.delete(path, async (req, res) => {

const {fileId} = req.query

const result = await File.destroy({

where: {

id: fileId

}

})

if (result) {

res.send({result: "ok"})

} else {

res.status(405).send({result: "error"})

}

})

}

Данный метод позволяет удалять файл при помощи id этого самого файла.

Листинг 4.2.5 – Метод download

export function generateFileToken(fileObj) {

const needsToken = crypto.createHash("md5").update(fileObj.get("user\_id") + "qweqwe").digest('hex');

return needsToken;

}

export function downloadFilesRoute(app, path) {

app.get(path + "/download", async (req, res) => {

const {fileId, token} = req.query

const fileObj = await FileModel.findOne({

where: {

id: fileId

}

})

if (!fileObj) {

sendError(res, 404, "Invalid file id", "Неверный id файла")

return

}

const needsToken = generateFileToken(fileObj);

if (token !== needsToken) {

sendError(res, 401, "Invalid file token", "Неверный токен для получения файла")

return

}

const blob = fileObj.get("file")

fs.writeFileSync(fileObj.get("filename"), blob)

res.download(fileObj.get("filename"), () => {

fs.unlink(fileObj.get("filename"), () => {

})

})

})

}

Данный метод позволяет скачать загруженный файл. Идет проверка id файла и токена.

Листинг 4.2.6 – Метод upload

function getRegexp(text) {

return `(${text.replaceAll(/[^а-яА-Яa-zA-Z\s]/gmi, "").split(" ").join("|")})`;

}

export function fileUpload(app, path) {

app.post(path + '/upload', async function (req, res) {

const {file} = req.files

const {userId} = app.locals.settings;

if (file.mimetype !== "application/pdf") {

sendError(res, 403, "File not pdf", "Это не пдф файл")

} else {

const text = await new Promise((resolve) => {

let result = ""

new PdfReader().parseBuffer(file.data, (err, item) => {

if (!item && !err) {

resolve(result)

} else {

result += item?.text

}

});

})

const l = await File.findAll({

where: {

content: {

[Op.regexp]: getRegexp(text),

}

}

})

let similarity = 0

if (l.length > 0) {

const objSimilarity = stringSimilarity.findBestMatch(text, l.map((e) => e.get("content")));

console.log(objSimilarity)

if (objSimilarity?.bestMatch?.rating) {

similarity = Math.ceil((objSimilarity.bestMatch.rating \* 100))

}

}

let fileRow = await File.create({

user\_id: userId,

content: text,

file: file.data,

filename: Buffer.from(file.name, 'ascii').toString(),

overlap: similarity

})

fileRow = JSON.parse(JSON.stringify(fileRow))

res.send({file: {...fileRow, content: undefined, file: undefined}})

}

});

}

Данный метод позволяет загрузить файл.

Листинг 4.2.7. – Метод get

export async function getUserFiles(app) {

const {userId} = app.locals.settings;

const files = await File.findAll({

where: {

user\_id: userId

},

attributes: ["id", "user\_id", "filename", "overlap", "createdAt"]

})

const filesResult = JSON.parse(JSON.stringify(files)).map((e, i) => {

return {...e, content: undefined, data: undefined, file: undefined, token: generateFileToken(files[i])}

})

return filesResult;

}

export function getFiles(app, path) {

app.get(path, async (req, res) => {

const filesResult = await getUserFiles(app);

res.send({files: filesResult})

})

}

Данный метод позволяет получить данные по файлу.

## **Структура моделей**

# **СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ**

* Antiplagiat.ru [Электронный ресурс]. URL: <https://www.antiplagiat.ru/> ;