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

**ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ**

(проектно-технологическая практика)

на тему: «Разработка информационной системы

для автоматизации бизнес-процесса»

Выполнил:

студент группы ПИб-21

направления подготовки

09.03.03 Прикладная информатика

Носов Иван Павлович

Проверил:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

Пермь – 2022

РЕЦЕНЗИЯ

на курсовую работу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

Содержание

[Постановка задачи на проектирование 3](#_Toc108724339)

[Анализ технологий проектирования 4](#_Toc108724340)

[**Обзор языков программирования** 4](#_Toc108724341)

[Python 4](#_Toc108724342)

[Java 5](#_Toc108724343)

[JavaScript 5](#_Toc108724344)

[Вывод 6](#_Toc108724345)

[**Анализ технологий хранения данных** 6](#_Toc108724346)

[PostgreSQL 7](#_Toc108724347)

[MySQL 8](#_Toc108724348)

[SQLite 8](#_Toc108724349)

[Вывод 9](#_Toc108724350)

[**Архитектурный паттерн MVC** 9](#_Toc108724351)

[Реализация функционала информационной системы 11](#_Toc108724352)

[Заключение 14](#_Toc108724353)

[Список источников 15](#_Toc108724354)

[ПРИЛОЖЕНИЕ А 16](#_Toc108724355)

[ПРИЛОЖЕНИЕ Б 20](#_Toc108724356)

# Постановка задачи на проектирование

Предприятие, реализующее услуги или производящее товары, нуждается в обратной связи от клиентов. Обратная связь зачастую приходит в виде отзывов о проделанной работе или о качестве товара. В связи с этим ставится проблема – как получать такую обратную связь в виде отзывов. В эпоху всемирного интернета самый простой способ — это получение текстовых отзывов на Интернет-ресурсе.

Такой системой с одной стороны пользуется клиент для получения правдивой информации о предприятии, либо для отправки своего отзыва. С другой стороны, такой системой пользуется предприятие для анализа обратной связи с последующим улучшением качества. Такая система позволяет. Система автоматизирует большую часть процесса сбора и анализа отзывов от потребителей.

Требования к функционалу:

1. Получение отзывов от клиентов
2. Сортировка отзывов по оценке
3. Предоставление отзывов других потребителей
4. Предоставление общей информации о товаре/услуге

Требования к интерфейсу:

1. Удобство
2. Информативность
3. Быстродействие

Требования к защищенности:

1. Доступ к редактированию списка отзывов должен быть только у администратора системы

Требования к кроссплатформенности:

1. Система должна одинаково работать в рамках любой платформы, в которой возможен выход в интернет с помощью веб-браузера.

# Анализ технологий проектирования

## **Обзор языков программирования**

Поскольку система должна собирать отзывы в среде Интернет, то необходимо выбрать язык программирования с заранее существующими фреймворками для работы с веб-страницами.

Фреймворк - программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Популярные языки для работы с веб-страницами посредством фреймворков:

* Python (Django)
* Java (Spring)
* JavaScript (Node.js)

### Python

Python - Высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации.

Однако даже самый популярный фреймворк Django имеет более высокий порог входа для написания веб-сайтов. Даже если сложностей с освоением Python возникнуть не должно, то фреймворк потребует некоторой подготовки.

### Java

Java - строго типизированный объектно-ориентированный язык программирования общего назначения.

Программы на Java транслируются в байт-код Java, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина.

Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание. Часто к недостаткам концепции виртуальной машины относят снижение производительности.

Также как и в языке программирования Python самый популярный фреймворк для Java требует довольно большой подготовки для написания полноценной системы.

**JavaScript**

JavaScript - мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам

Node.js — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера.

Node.js не требует от программиста принципиально новых знаний, в отличие от первых двух фреймворков.

**Вывод**

Опираясь на всё вышеперечисленное, логичнее всего будет сделать выбор в пользу Node.js.

## **Анализ технологий хранения данных**

Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая стока таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей. К этим данным можно получить доступ многими способами, и при этом реорганизовывать таблицы БД не требуется.

В качестве БД для проектируемой ИС было рассмотрено три системы управления базами данных:

* PostgreSQL
* MySQL
* SQLite

### PostgreSQL

PostgreSQL — это самая продвинутая реляционная СУБД, ориентирующаяся в первую очередь на полное соответствие стандартам и расширяемость. PostgreSQL, или Postgres, пытается полностью соответствовать SQL - стандартам ANSI / ISO.

PostgreSQL отличается от других РСУБД тем, что обладает объектно-ориентированным функционалом, в том числе полной поддержкой концепта ACID (Atomicity, Consistency, Isolation, Durability).

Будучи основанным на мощной технологии Postgres отлично справляется с одновременной обработкой нескольких заданий. Поддержка конкурентности реализована с использованием MVCC (Multiversion Concurrency Control), что также обеспечивает совместимость с ACID.

Плюсы:

* Полная SQL-совместимость.
* Сообщество. PostgreSQL поддерживается опытным сообществом.
* Поддержка сторонними организациями. Несмотря на очень продвинутые функции, PostgreSQL используется в многих инструментах связанных с СУБД.
* Расширяемость. PostgreSQL можно программно расширить за счет хранимых процедур.
* Объектно-ориентированность. PostgreSQL — не только реляционная, но и объектно-ориентированная СУБД.

Минусы:

* Производительность. В простых операциях чтения PostgreSQL может уступать своим конкурентам.
* Популярность. Из-за сложности данный инструмент не так популярен, как другие.

### MySQL

MySQL — это самая популярная из всех крупных серверных БД. Разобраться в ней очень просто, да и в сети о ней можно найти большое количество информации. Хотя MySQL и не пытается полностью реализовать SQL- стандарты, она предлагает широкий функционал.

Плюсы:

* Простота. В MySQL легко разобраться, её легко устанавливать, существует много инструментов облегчающих старт работы с СУБД.
* Много функций. MySQL поддерживает большую часть функционала SQL.
* Скорость. Пренебрежение некоторыми стандартами позволяет MySQL работать производительнее.

Минусы:

* Известные ограничения. MySQL не обладает всем функционалом SQL.
* Вопросы к надежности. Некоторые операции проходят в MySQL не безопасно.

### SQLite

SQLite — это библиотека, встраиваемая в приложение, которое её использует. Будучи файловой БД, она предоставляет отличный набор инструментов для более простой (в сравнении с серверными БД) обработки любых видов данных.

Когда приложение использует SQLite, их связь производится с помощью функциональных и прямых вызовов файлов, содержащих данные (например, баз данных SQLite), а не какого-то интерфейса, что повышает скорость и производительность операций.

Плюсы:

* Файловый тип. Вся база хранится в файле, что облегчает её перенос.
* Стандартизированная. SQLite реализует большинство операций SQL.

Минусы:

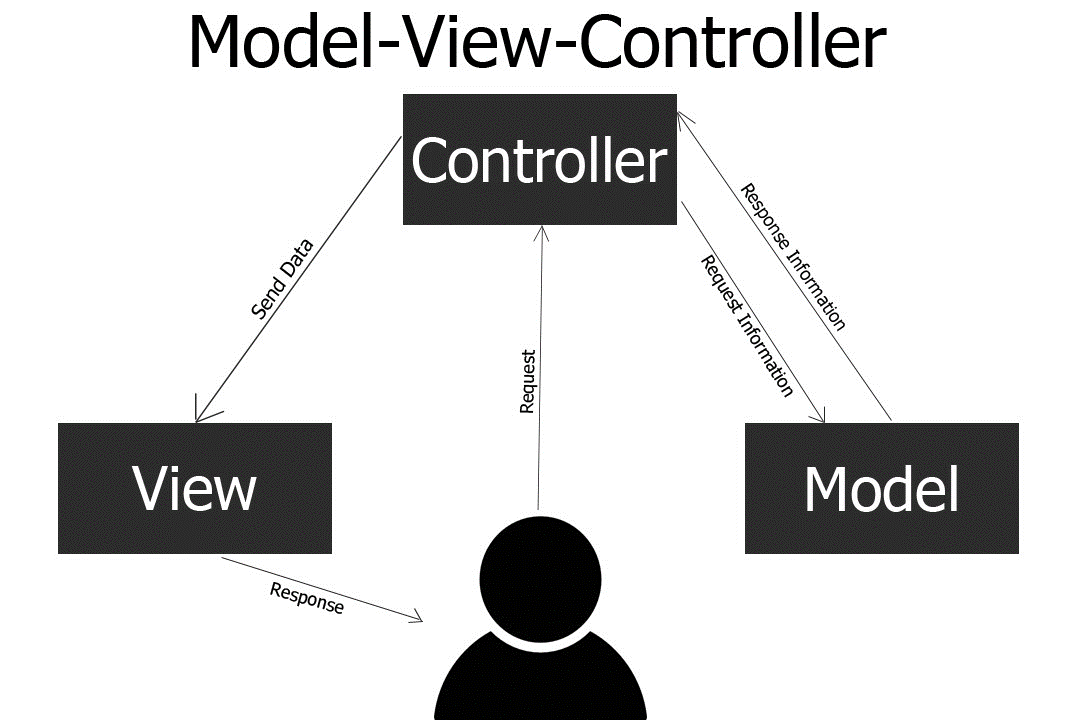
* Отсутствие пользовательского управления.
* Отсутствие возможности настройки.

### Вывод

Опираясь на функционал и требования к производительности проектируемой информационной системе предпочтительным выбором станет SQLite, так как данная СУБД простая и производительная, как и наша система.

## **Архитектурный паттерн MVC**

Шаблон проектирования MVC предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо. Модель (Model) предоставляет данные предметной области представлению и реагирует на команды контроллера, изменяя свое состояние. Представление (View) отвечает за отображение данных предметной области (модели) пользователю, реагируя на изменения модели. Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

  
Рисунок 1 — Схема MVC

Полный цикл работы данной MVC-триады: модели, представления и контроллера переключателя – можно описать следующим образом. При инициализации представления пользователем оно обращается к модели и устанавливает текст метки в соответствии с текущим состоянием переключателя. Пользователь инициирует изменение переключателя, нажимая на определенную кнопку. При этом представление отправляет соответствующую команду контроллеру: включить или выключить. Контроллер интерпретирует команду и изменяет модель. Представление регистрирует изменение модели: по этому событию оно изменяет текст метки для соответствия новому состоянию модели переключателя.

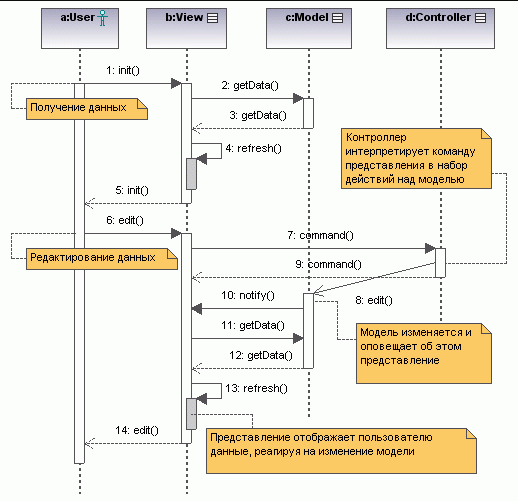


Рисунок 3 — Диаграмма последовательности MVC

# Реализация функционала информационной системы

**Логическая модель базы данных**

В разрабатываемой ИС используется СУБД SQLite с одной таблицей в БД.

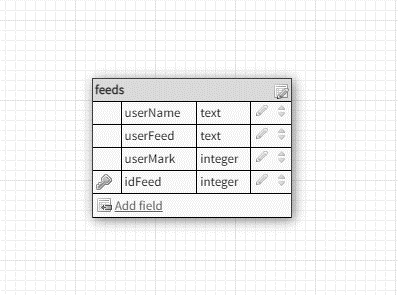


Рисунок 4 — Таблица в БД приложения.

Таблица содержит следующие поля:

* idFeed — уникальный идентификатор отзыва;
* userName — имя пользователя оставившего отзыв;
* userFeed — текст отзыва;
* userMark — условная оценка от 1 до 5.

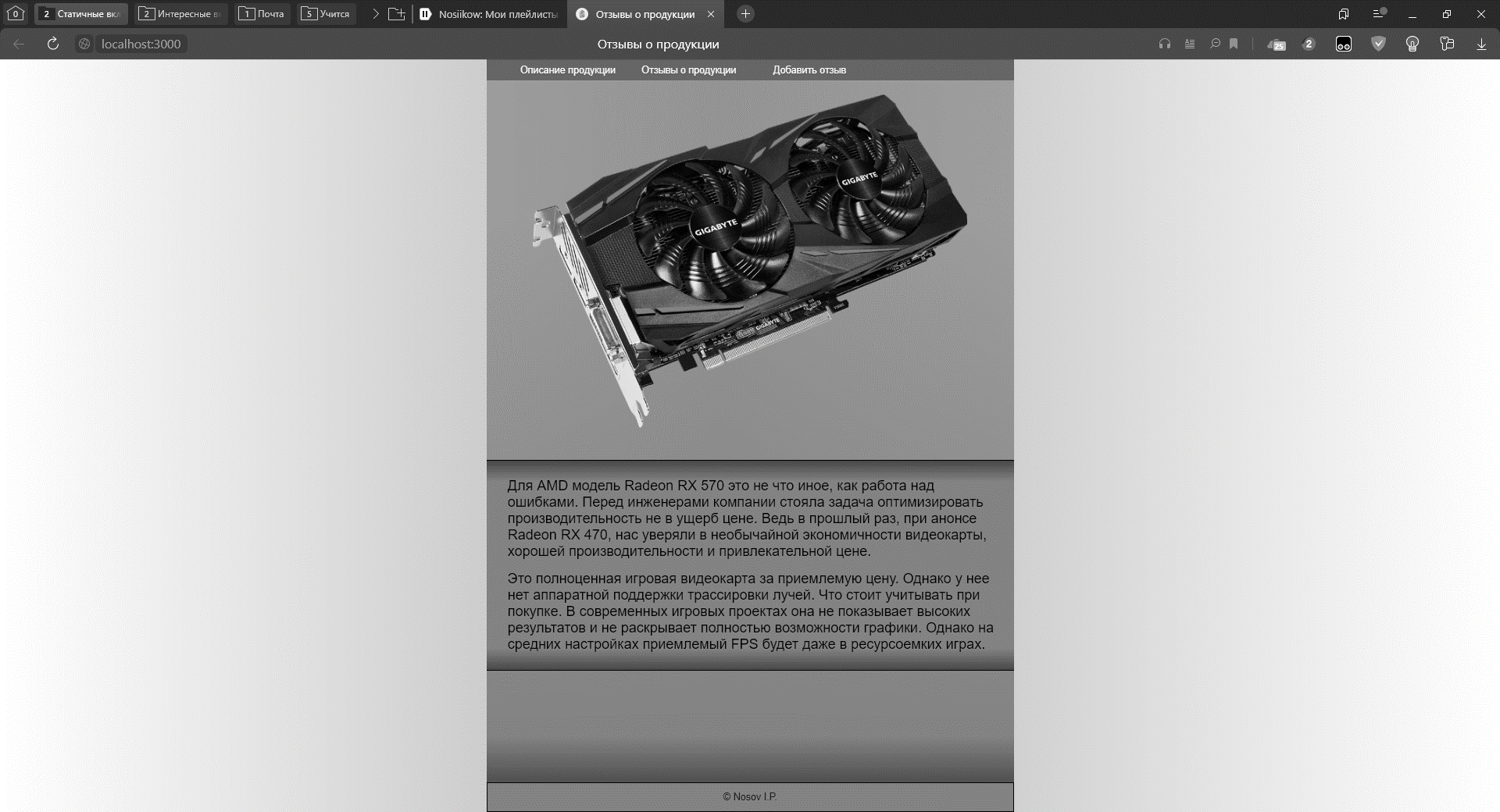
**Запросы к базе данных**

При работе ИС будут использоваться следующие SQL запросы:

* SELECT userName, userFeed, userMark FROM feeds ORDER BY userMark DESC — запрос отвечает за получение всех отзывов и сортировку их по убыванию по полю userMark.
* INSERT INTO feeds ( userName, userFeed, userMark) VALUES (@name, @feed, @mark) — запрос отвечает за добавление нового отзыва в БД.

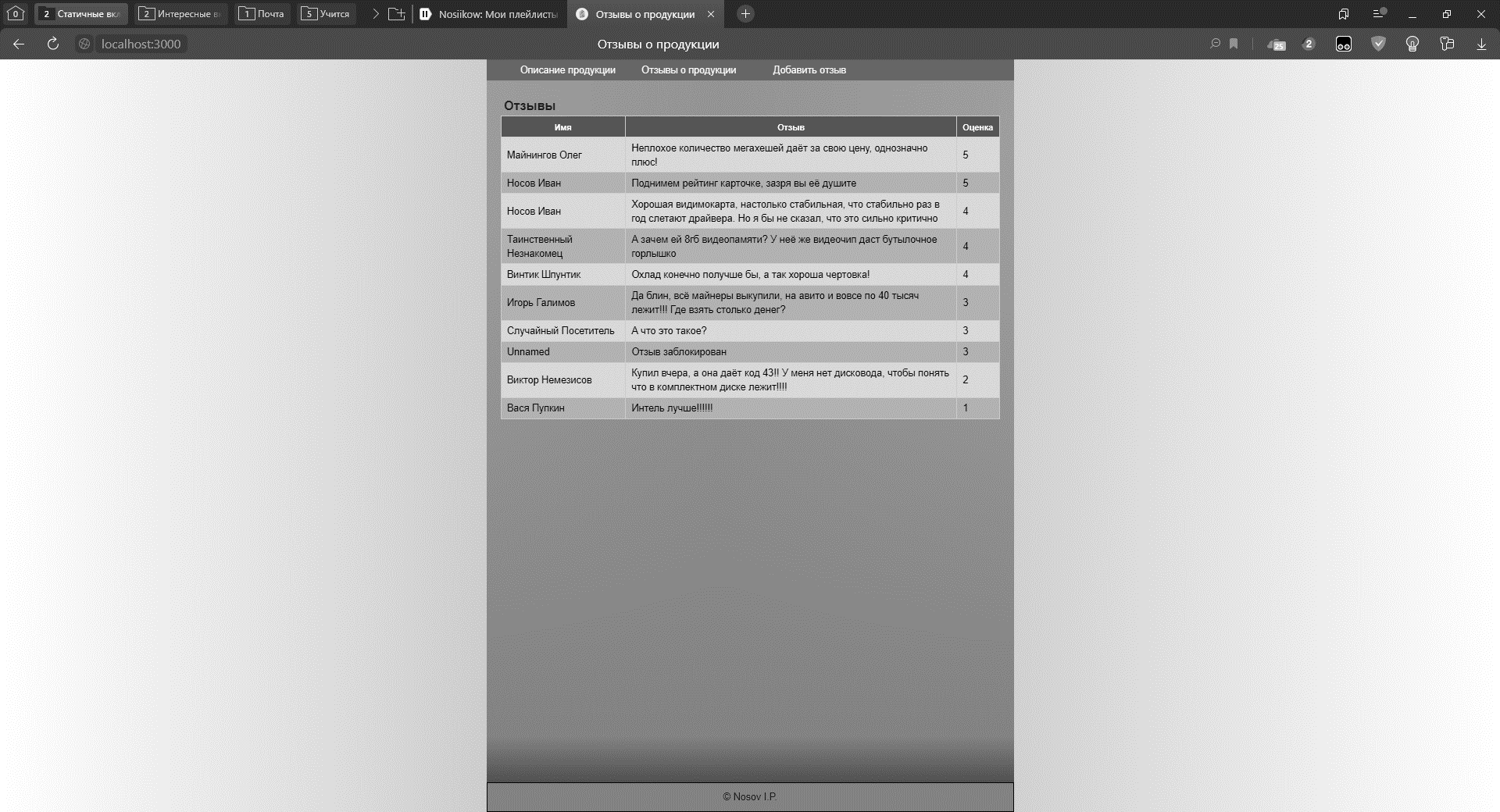
**Интерфейс пользователя**

Приложение имеет три окна (View).

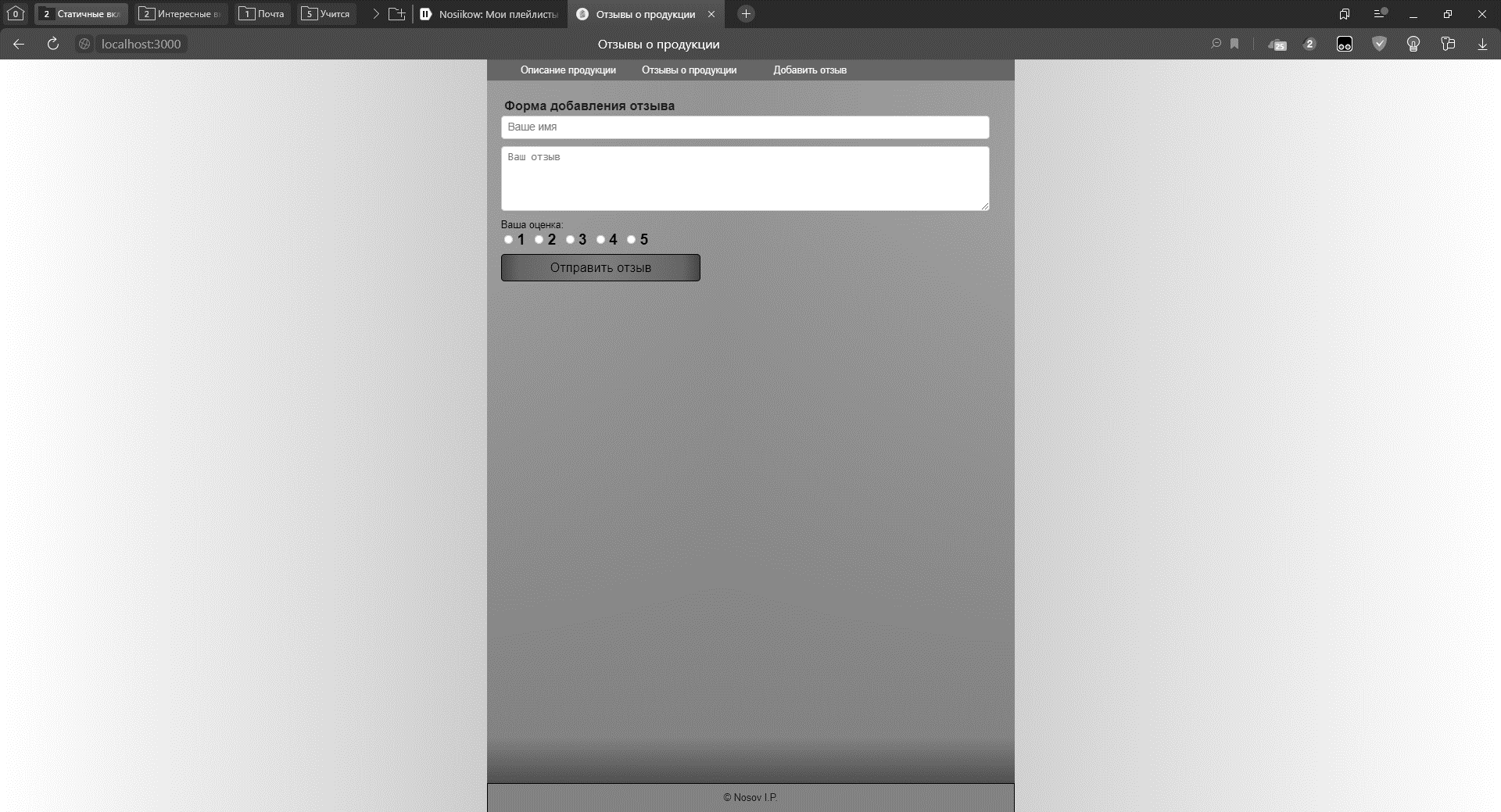
Рисунок 5 — Окно intro.

Первое окно intro, содержит краткую информацию о компании. В шапке сайта есть кнопки переходов к следующим страницам:

* список всех отзывов;
* форма для нового отзыва;
* переход к окну intro;

Рисунок 6 — Окно feeds.

Окно feeds содержит таблицу наполненную всеми отзывами.

Рисунок 7 — Окно new\_feeds.

Окно new\_feeds содержит форму для нового отзыва. После добавления нового отзыва, происходит переадресация на страницу со всеми отзывами, где сразу можно увидеть новый отзыв.

# Заключение

В рамках проектно-технологической работы был проведён анализ популярных языков программирования и фреймворков к ним, а также различные виды СУБД. Был изучен архитектурный паттерн MVC или Model-View-Controller. После анализа была разработана информационная система по сбору и отображению отзывов на продукт.

Система работает стабильно и не требует большого количества ресурсов для продолжительной работы.

В дальнейшем планируется ввести регистрацию и авторизацию пользователя. Данная функция нужна для администратора т.к. есть необходимость в модерации отзывов, т.е. некоторого отбора отзывов (отсеивание нецензурных слов, рекламы и прочего).

Также планируется ввести шифрование данных и генерацию документов по существующим отзывам.

# Список источников

1. ГОСТ 19.401-78 Текст программы. требования к содержанию и оформлению.
2. ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы.
3. ГОСТ 19.701-90 Схемы алгоритмов, программ данных и систем.
4. Алгоритм хеширования MD5. [Электронный ресурс] – Электрон. текстовые дан. Режим доступа: https://ru.wikipedia.org/wiki/MD5
5. МЕТОДИЧЕСКОЕ ПОСОБИЕ по выполнению курсового проекта по дисциплине Объектно-ориентированное программирование. Режим доступа: https://pcoding.ru/pdf/CourseProject.pdf
6. Git Repositories Documentation [Электронный ресурс] – Электрон. текстовые дан. – Microsoft, Режим доступа: https://docs.microsoft.com/ru-ru/azure/devops/repos/git, свободный.
7. MSDN. Техническая документация. MD5 Класс. [Электронный ресурс] Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.security.cryptography.md5?view=netframework-4.8
8. MSDN. Техническая документация. HashAlgorithm.ComputeHash Метод. [Электронный ресурс] Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.security.cryptography.hashalgorithm.computehash?view=netframework-4.8
9. Node.js Техническая документация. [Электронный ресурс] Режим доступа: https://nodejs.org/en/docs/
10. PostgreSQL Техническая документация [Электронный ресурс] Режим доступа: https://nodejs.org/en/docs/
11. SQLite Техническая документация [Электронный ресурс] Режим доступа: https://www.sqlite.org/docs.html

# ПРИЛОЖЕНИЕ А

**Техническое задание**

наименование организации – разработчика ТЗ на АС

УТВЕРЖДАЮ

Руководитель

(должность, наименование предприятия – заказчика АС)

Личная подпись Расшифровка подписи

(печать)

Дата

УТВЕРЖДАЮ

Руководитель

(должность, наименование предприятия – разработчик АС)

Личная подпись Расшифровка подписи

(печать)

Дата

наименование вида АС

наименование объекта автоматизации

сокращённое наименование АС

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

На листах

Действует с

2022

1. Общие сведения

* Наименование системы

Полное наименование разрабатываемой системы – «Автоматизированный модуль сбора отзывов».

Краткое наименование – «Модуль сбора отзывов».

* Наименование заказчика и исполнителя

Организация: ФГБОУ ВО Пермский ГАТУ.

Адрес: ул. Петропавловская, 23.

Телефон: +7 (342) 217-90-66;

Исполнитель: Носов Иван Павлович.

* Плановые сроки начала и окончания работ

Дата начала работ: 04.07.2022.

Дата окончания работ: 17.07.2022.

1. Назначение и цели создания системы

К целям создания модуля сбора отзывов можно отнести:

1. оперативный сбор обратной связи клиентов о продукции;
2. улучшение качества обслуживания клиентов.

Достижение целей приведёт к следующим положительным результатам:

1. выработается доверие у клиентов;
2. обратная связь укажет на слабые места в обслуживании и качестве продукции, что позволит оперативно устранять недостатки;
3. Характеристика объекта автоматизации

Объектом автоматизации является веб-приложение по сбору отзывов о продукции компании.

1. Требования к системе

Общие требования к модулю «Живая лента» являются:

1. надёжность и работоспособность;
2. интуитивно понятный интерфейс;
3. соблюдение информационной безопасности и разграничение прав доступа к данным.
4. Требования к способам и средствам связи для информационного обмена между компонентами

Для обеспечения информационного обмена компоненты подсистемы должны взаимодействовать с базой данных.

1. Перспективы развития, модернизация системы

Дальнейшим развитием модуля сбора отзывов может быть добавление личного кабинета.

1. Требования к квалификации персонала и режиму его работы

Для обеспечения максимальной работоспособности пользователей должны устанавливаться перерывы:

1. через 2 часа после начала смены и через 1,5–2 часа продолжительностью 15 минут;
2. через каждый час работы продолжительностью 10 минут

Для эксплуатации модуля сбора отзывов определены следующие роли:

1. системный администратор – должен быть квалифицированным специалистом умеющим обслуживать БД и модули программы.
2. Требования к надёжности технических средств и программного обеспечения

Надёжность по отношению к техническим средствам должна обеспечиваться использованием в системе средств повышенной отказоустойчивости и их резервированием, а также дублированием носителей информационных банков данных.

Надёжность программного комплекса обеспечивается использованием сертифицированных операционных систем, общесистемных программных средств и инструментальных программных систем, используемых при разработке программного обеспечения. Само программное обеспечение должно обеспечивать защиту от некорректных действий пользователей и ошибочных исходных данных.

1. Требования к безопасности

Разрабатываемый информационный модуль сбора отзывов должен обеспечивать безопасный доступ к данным, предотвращая несанкционированный доступ или модифицирование данных.

1. Требования по эргономике и технической эстетике

Модуль должен иметь удобный и интуитивно понятный графический пользовательский интерфейс. Диалоговый интерфейс должен соблюдать контекст подсистемы.

Пользовательский интерфейс модуля также должен аккомпанировать цветовой гамме и общему стилю компании.

1. Требования к программному обеспечению

При проектировании информационного модуля сбора отзывов необходимо эффективно использовать в качестве серверного окружения использовать платформу Node.js, а для хранения данных применять СУБД SQLite.

1. Требования к техническому обеспечению

Техническое обеспечение системы должно обеспечить бесперебойную работу системы:

1. процессор – AMD Ryzen 5 2600;
2. оперативная память – 16 ГБ;
3. дисковая система – 120 ГБ;
4. сетевой адаптер – 1 Гб/с.
5. Порядок контроля и приёмки системы

По результатам опытной эксплуатации оформляется акт о приёме работ. Акт содержит заключение о соответствии системы техническому заданию.

1. Требования к составу и содержанию работ подготовки объекта автоматизации к вводу системы в действие

При подготовке к вводу в эксплуатацию модуля сбора отзывов необходимо обеспечить выполнение следующих работ:

1. определить подразделение и ответственных должностных лиц для внедрения информационного модуля;
2. провести опытную эксплуатацию модуля сбора отзывов.

# ПРИЛОЖЕНИЕ Б

**Листинг кода приложения – app.js**

const express = require('express');

const app = express();

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

app.set('view engine', 'ejs');

const r\_intro = require('./routes/r\_intro.js');

const r\_feeds = require('./routes/r\_feeds.js');

const r\_new\_feed = require('./routes/r\_new\_feed.js');

app.use(['/intro','/'], r\_intro);

app.use('/feeds', r\_feeds);

app.use('/new\_feed', r\_new\_feed);

app.listen(3000, "localhost", () => console.log(`--> to stop: Ctrl+C`));

**Листинг кода окна intro – r\_intro.js**

const express = require('express');

const router = express.Router();

let { model } = require('../models/m\_intro.js');

router.get('/', (req, res) => {

res.render('intro.ejs', model); // render view

});

module.exports = router;

**Листинг кода view - окна intro – intro.ejs**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title><%= title %></title>

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

<link rel="shortcut icon" href="../public/images/favicon.ico" />

</head>

<body>

<div class="wrapper">

<div class="content">

<%- include("./header.html") %>

<br>

<p class="aligncenter">

<img src="../public/images/rx.png" alt="logo">

</p>

<br><br>

<div id="intro">

<p class="intro">

Для AMD модель Radeon RX 570 это не что иное, как работа над ошибками. Перед инженерами компании стояла задача оптимизировать производительность не в ущерб цене.

Ведь в прошлый раз, при анонсе Radeon RX 470, нас уверяли в необычайной экономичности видеокарты, хорошей производительности и привлекательной цене.

</p>

<p class="intro">

Это полноценная игровая видеокарта за приемлемую цену. Однако у нее нет аппаратной поддержки трассировки лучей.

Что стоит учитывать при покупке. В современных игровых проектах она не показывает высоких результатов и не раскрывает полностью возможности графики.

Однако на средних настройках приемлемый FPS будет даже в ресурсоемких играх.

</p>

</div>

</div>

<%- include("./footer.html") %>

</div>

</body>

</html>

**Листинг кода окна new\_feed – r\_new\_feed.js**

const express = require('express');

const router = express.Router();

let { model } = require("../models/m\_new\_feed.js");

const dp = require("../private/database.js");

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

router.get('/', (req, res) => {

res.render('new\_feed.ejs', model); // render view

});

router.post('/', htmlParser, (req, res) => {

let { userName, userFeed, userMark} = req.body;

dp.insertFeed(userName, userFeed, userMark);

res.redirect("/feeds");

});

module.exports = router;

**Листинг кода view - окна new\_feed – new\_feed.ejs**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title><%= title %></title>

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

<link rel="shortcut icon" href="../public/images/favicon.ico" />

</head>

<body>

<div class="wrapper">

<div class="content">

<%- include("./header.html") %>

<div class="desktop">

<br>

<h4><%= tableName %></h4>

<form action="/new\_feed" method="post"> <!-- вызов события /new\_feed -->

<input class="txt" name="userName" type="name" placeholder="Ваше имя">

<textarea class="txtarea" name="userFeed" placeholder="Ваш отзыв"></textarea>

<p>Ваша оценка:</p>

<h2 class="mark" ><input class="mark1" name="userMark" type="radio" value="1"> 1</h2>

<h2 class="mark" ><input class="mark2" name="userMark" type="radio" value="2"> 2</h2>

<h2 class="mark" ><input class="mark3" name="userMark" type="radio" value="3"> 3</h2>

<h2 class="mark" ><input class="mark4" name="userMark" type="radio" value="4"> 4</h2>

<h2 class="mark" ><input class="mark5" name="userMark" type="radio" value="5"> 5</h2>

<button class="send">Отправить отзыв</button>

</form>

</div>

</div>

<%- include("./footer.html") %>

</div>

</body>

</html>

**Листинг кода окна feeds – r\_feeds.js**

const express = require('express');

const router = express.Router();

const { model } = require("../models/m\_feeds.js");

const dp = require("../private/database.js");

router.get('/', (req, res) => {

model.feeds = dp.getFeeds();

res.render('feeds.ejs', model); // render view

});

module.exports = router;

**Листинг кода view - окна feed – feeds.ejs**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title><%= title %></title>

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

<link rel="shortcut icon" href="../public/images/favicon.ico" />

</head>

<body>

<div class="wrapper">

<div class="content">

<%- include("./header.html") %>

<div class="desktop">

<br>

<h4>Отзывы</h4>

<table>

<tr>

<th width="25%">Имя</th>

<th>Отзыв</th>

<th>Оценка</th>

</tr>

<% for (let feed of feeds) { %>

<tr>

<td class="td"><%= feed.userName %></td>

<td class="td"><%= feed.userFeed %></td>

<td class="ts"><%= feed.userMark %></td>

</tr>

<% } %>

</table>

</div>

</div>

<%- include("./footer.html") %>

</div>

</body>

</html>

**Листинг кода класса для работы с БД – database.js**

const pathDB = "./private/database.db";

const db = require('better-sqlite3')(pathDB);

class DataProcessing {

getFeeds() {

let query = `

SELECT userName, userFeed, userMark

FROM feeds

ORDER BY userMark DESC`;

let rows = db.prepare(query).all();

return rows;

}

insertFeed(name, feed, mark) {

let values = { name: name, feed: feed, mark: mark };

let query = `

INSERT INTO feeds (userName,userFeed,userMark)

VALUES (@name, @feed, @mark)`;

db.prepare(query).run(values);

}

}

module.exports = new DataProcessing();