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

Карпов Максим Валерьевич

Проверил:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

2022

Содержание

[Постановка задачи на проектирование 3](#_Toc108551734)

[Анализ технологий проектирования 4](#_Toc108551735)

[Обзор реляционных баз данных 4](#_Toc108551736)

[MySQL 4](#_Toc108551737)

[PostgreSQL 5](#_Toc108551738)

[SQLite 6](#_Toc108551739)

[Обзор языков программирования 7](#_Toc108551740)

[С# 7](#_Toc108551741)

[Python 7](#_Toc108551742)

[Node.js 8](#_Toc108551743)

[Реализация функционала информационной системы 9](#_Toc108551744)

[Логическая модель базы данных 9](#_Toc108551745)

[Запросы к базе данных 10](#_Toc108551746)

[Интерфейс пользователя 12](#_Toc108551747)

[Заключение 14](#_Toc108551748)

[Список источников 15](#_Toc108551749)

[ПРИЛОЖЕНИЕ А 16](#_Toc108551750)

[ПРИЛОЖЕНИЕ Б 21](#_Toc108551751)

# Постановка задачи на проектирование

Практика посвящена рассмотрению процесса автоматизации бизнес-процессов и выполнению этапов, связанных с разработкой программной реализации.

Этапы автоматизации бизнес-процесса:

* исследуем предприятие;
* выбираем плохо автоматизированный процесс;
* составляем модели процесса «как-есть» и «как-будет»;
* **составляем Техническое Задание на разработку информационной системы;**
* **разрабатываем прототип и проводим апробацию;**
* разрабатываем программную реализацию;
* тестируем, внедряем.

Из приведённой последовательности этапов на данной практике будут задействованы:

* составление Технического задания;
* разработка прототипа;
* апробация информационной системы.

В данную задачу будут интегрированы такие вопросы проектирования и программирования как: архитектура web-приложения, MVC-паттерн проектирования, организация обмена данными между программой и базой данных, передача параметров и данных между формами приложения.

Необходимо создать базу данных с отзывами клиентов. База отзывов может стать частью CRM-системы – системы управления взаимоотношениями с клиентами. Это не только дополнение клиентской базы, но и подход к увеличению продаж услуг и продуктов. С помощью обратной связи можно понять, как работает продукт или услуга, какие есть недостатки, чего действительно хотят пользователи. Тогда можно улучшить продукт или услугу и доработать их до ожиданий клиентов.

# Анализ технологий проектирования

## **Обзор реляционных баз данных**

Реляционные базы данных используются уже очень давно. Они стали популярными благодаря успешным реализациям реляционных моделей в системах управления, оказавшимся весьма удобными для работы с данными.

Наиболее популярные РСУБД:

* SQLite: очень мощная встраиваемая РСУБД.
* MySQL: самая популярная и часто используемая РСУБД.
* PostgreSQL: самая продвинутая и гибкая РСУБД.

### **MySQL**

MySQL — это самая популярная из всех крупных серверных БД. Разобраться в ней очень просто. MySQL и не пытается полностью реализовать SQL-стандарты, она предлагает широкий функционал. Приложения общаются с базой данных через процесс-демон.

**Преимущества**

Простота: MySQL легко устанавливается. Существует много сторонних инструментов, включая визуальные, облегчающих начало работы с БД.

Много функций: MySQL поддерживает большую часть функционала SQL.

Безопасность: в MySQL встроено много функций безопасности.

Мощность и масштабируемость: MySQL может работать с действительно большими объёмами данных, и неплохо походит для масштабируемых приложений.

Скорость: пренебрежение некоторыми стандартами позволяет MySQL работать производительнее.

**Недостатки**

Известные ограничения: по определению, MySQL не может сделать всё, что угодно, и в ней присутствуют определённые ограничения функциональности.

Вопросы надёжности: некоторые операции реализованы менее надёжно, чем в других РСУБД.

Застой в разработке: MySQL является open-source продуктом, но работа над ней сильно заторможена.

**Когда стоит использовать MySQL**

Распределённые операции: когда вам нужен функционал больший, чем может предоставить SQLite, стоит использовать MySQL.

Высокая безопасность: функции безопасности MySQL предоставляют надёжную защиту доступа и использования данных.

Веб-сайты и приложения: большая часть веб-ресурсов вполне может работать с MySQL, несмотря на ограничения. Этот инструмент весьма гибок и прост в обращении, что только на руку в длительной перспективе.

Кастомные решения: если вы работаете над очень специфичным продуктом, MySQL подстроится под ваши потребности благодаря широкому спектру настроек и режимов работы.

**Когда не стоит использовать MySQL**

SQL-совместимость: поскольку MySQL не пытается полностью реализовать стандарты SQL, она не является полностью совместимой с SQL. Из-за этого могут возникнуть проблемы при интеграции с другими РСУБД.

Конкурентность: MySQL одновременные операции чтения-записи могут вызвать проблемы.

Недостаток функций: в зависимости от выбора движка MySQL может недоставать некоторых функций.

### **PostgreSQL**

PostgreSQL — это самая продвинутая РСУБД, ориентирующаяся 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).

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

**Преимущества**

Полная SQL-совместимость.

Сообщество: PostgreSQL поддерживается опытным сообществом 24/7.

Поддержка сторонними организациями: PostgreSQL используется в многих инструментах, связанных с РСУБД.

Расширяемость: PostgreSQL можно программно расширить за счёт хранимых процедур.

Объектно-ориентированность: PostgreSQL — не только реляционная, но и объектно-ориентированная СУБД.

**Недостатки**

Производительность: в простых операциях чтения PostgreSQL может уступать своим соперникам.

Популярность: из-за своей сложности инструмент не очень популярен.

Хостинг: из-за вышеперечисленных факторов проблематично найти подходящего провайдера.

**Когда стоит использовать PostgreSQL**

Целостность данных: если приоритет стоит на надёжность и целостность данных, PostgreSQL — лучший выбор.

Сложные процедуры: если ваша БД должна выполнять сложные процедуры, стоит выбрать PostgreSQL в силу её расширяемости.

Интеграция: если в будущем вам предстоит перемещать всю базу на другое решение, меньше всего проблем возникнет с PostgreSQL.

**Когда не стоит использовать PostgreSQL**

Скорость: если всё, что нужно — это быстрые операции чтения, не стоит использовать PostgreSQL.

Простые ситуации: если вам не требуется повышенная надёжность, поддержка ACID и всё такое, использование PostgreSQL

### **SQLite**

**SQLite** — это изумительная библиотека, встраиваемая в приложение, которое её использует. Будучи файловой БД, она предоставляет отличный набор инструментов для более простой 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.

**Преимущества**

Файловая: вся база данных хранится в одном файле, что облегчает перемещение.

Стандартизированная: SQLite использует SQL; некоторые функции опущены, однако, есть и некоторые новые.

Отлично подходит для разработки и даже тестирования: во время этапа разработки большинству требуется масштабируемое решение. SQLite, со своим богатым набором функций, может предоставить более чем достаточный функционал.

**Недостатки**

Отсутствие пользовательского управления: продвинутые БД предоставляют пользователям возможность управлять связями в таблицах в соответствии с привилегиями, но у SQLite такой функции нет.

Невозможность дополнительной настройки: SQLite нельзя сделать более производительной.

**Когда стоит использовать SQLite**

Встроенные приложения: все портируемые не предназначенные для масштабирования приложения.

Система доступа к дисковой памяти: в большинстве случаев приложения, часто производящие прямые операции чтения/записи на диск, можно перевести на SQLite для повышения производительности.

Тестирование: отлично подойдёт для большинства приложений, частью функционала которых является тестирование бизнес-логики.

**Когда не стоит использовать SQLite**

Многопользовательские приложения: если вы работаете над приложением, доступом к БД в котором будут одновременно пользоваться несколько человек, лучше выбрать полнофункциональную РСУБД — например, MySQL.

Приложения, записывающие большие объёмы данных: одним из ограничений SQLite являются операции записи (единовременное исполнение лишь одной операции записи).

Для нашего приложения подходит база данных SQLite, так как у нас небольшое количества данных.

## **Обзор языков программирования**

В современном мире без языков программирования невозможно обойтись. Языки программирования - сложный, но интересный процесс создания компьютерного мира!

### **С#**

**Сложность:** C# изначально перенял синтаксические конструкции языков С++ и Java, так как его разработали специально, чтобы упростить последние. У него хороший набор фреймворков (.Net, Xamarin) и библиотек.

**Продукты**: Это универсальный язык программирования. На нем можно писать серверную часть сайтов, десктопные приложения, серверные приложения и даже игры. На российском рынке популярен движок Unity 3D, который позволяет на языке C# писать игры для любых платформ: для PC, мобильных устройств, приставок или VR.

На нем удобно писать оконные приложения и веб-приложения или даже корпоративное программное обеспечение (например, банковское обслуживание). Так как язык входит в пятерку самых популярных, он узнаваем, и в целом из-за обширной синтаксической семьи C# его сообщество масштабное и активное.

### **Python**

**Сложность:** подходит для неподготовленного новичка. Обилие готовых библиотек (многие из которых предназначены для работы с big data) и фреймворков, а также простой синтаксис позволяют практически сразу начать программировать.

**Продукты**: Sims 4, YouTube, Instagram, Dropbox, Bittorrent.

**Преимущество** Python — в его многогранности и широком применении. Его можно использовать как в веб-разработке (фреймворк Django, например, достаточно прост для того, чтобы новичок быстро разобрался в основах его использования) и работе с данными, так и в написании скриптов. Можно заниматься анализом данных для банковских систем, автоматизацией простых задач, разрабатывать приложения, совершенствовать голосовые помощники.

### **Node.js**

**Node.js** — среда выполнения кода JavaScript вне браузера. Эта платформа позволяет писать серверный код для динамических веб-страниц и веб-приложений, а также для программ командной строки. С помощью Node.js реализуется парадигма «JavaScript для всего». Она предполагает использование одного языка программирования для разработки веб-приложений вместо применения разных языков для работы над фронтендом и бэкендом.

Node.js — не отдельный язык программирования, а платформа для использования JavaScript на стороне сервера. Если говорить о языке, то как для фронденда, так и для бэкенда используется один и тот же JavaScript. Разница только в наборе API, которые используют фронтендеры и бэкендеры.

Браузерный JavaScript использует Web API, которые обеспечивают доступ к DOM и пользовательскому интерфейсу страниц и веб-приложений. Серверный JavaScript использует API, обеспечивающие доступ к файловой системе приложений, http-запросам, потокам.

То есть Node.js — это технология для использования JS на бэкенде.

Node.js хорошо подходит для разработки RTA — веб-приложений, реагирующих на действия пользователя в режиме реального времени. Например, это может быть онлайн-редактор типа Google Docs, который позволяет работать над одним документом нескольким пользователям одновременно.

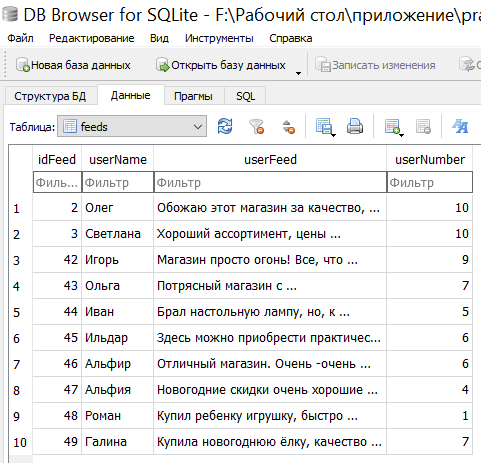
Node.js легко обрабатывает большое количество запросов одновременно и обеспечивает быстродействие приложения. Поэтому серверный JavaScript часто используют для создания SPA — одностраничных веб-приложений, в которых рендеринг выполняется на стороне клиента. Node.js на бэкенде используют Netflix, Uber, eBay, Groupon, Yahoo и другие известные организации и проекты.

Таким образом, Node.js — платформа для запуска кода JavaScript за пределами браузера. Эту технологию используют преимущественно для бэкенд-разработки на JavaScript.

# Реализация функционала информационной системы

## **Логическая модель базы данных**

Используется одна таблица БД с полями idFeed, userName, UserFeed, userNumber:

:

## **Запросы к базе данных**

**Общая структура запроса выглядит следующим образом:**

**SELECT** ('столбцы или \* для выбора всех столбцов; обязательно')

**FROM** ('таблица; обязательно')

**WHERE** ('условие/фильтрация, например, city = 'Moscow'; необязательно')

**GROUP BY** ('столбец, по которому хотим сгруппировать данные; необязательно')

**HAVING** ('условие/фильтрация на уровне сгруппированных данных; необязательно')

**ORDER BY** ('столбец, по которому хотим отсортировать вывод; необязательно')

Предложение **ORDER BY** служит для указания способа сортировки результатов запроса. Оно должно быть последним в запросе. Сортировать можно по одному или нескольким полям. Можно указывать псевдонимы имён полей или номера полей по порядку в списке вывода. Сортировка по возрастанию обычно устанавливается в СУБД по умолчанию. Для явного задания способа сортировки служат два ключевых слова: ASC - по возрастанию и DESC - по убыванию.

В части **INSERT INTO** мы указываем столбцы, которые хотим заполнить информацией. В **VALUES** указана информация, которую нужно сохранить.

При вставке строковых значений их нужно брать в одинарные кавычки.

В БД запрос: вывод на страницу с отзывами, отсортированным в обратном порядке по полю "Оценка продукта" (вверху таблицы отзывы с высокой оценкой, внизу - с низкой).

Таблица: feeds

Поля: idFeed , userName, UserFeed, userNumber

Сортировка по полю userNumber (оценка продукта) в обратном порядке (вверху таблицы отзывы с высокой оценкой, внизу - с низкой).

**Листинг кода формы (**сортировка по полю userNumber )**–** database.js

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

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

class DataProcessing {

    getFeeds() {

        let query = `

            SELECT userName, UserFeed, userNumber

            FROM feeds

            ORDER BY userNumber DESC`;

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

        return rows;

    }

    insertFeed(name, feed, number) {

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

        let query = `

            INSERT INTO feeds (userName,userFeed,userNumber)

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

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

    }

}

module.exports = new DataProcessing();

## **Интерфейс пользователя**

Состоит из трех страниц:

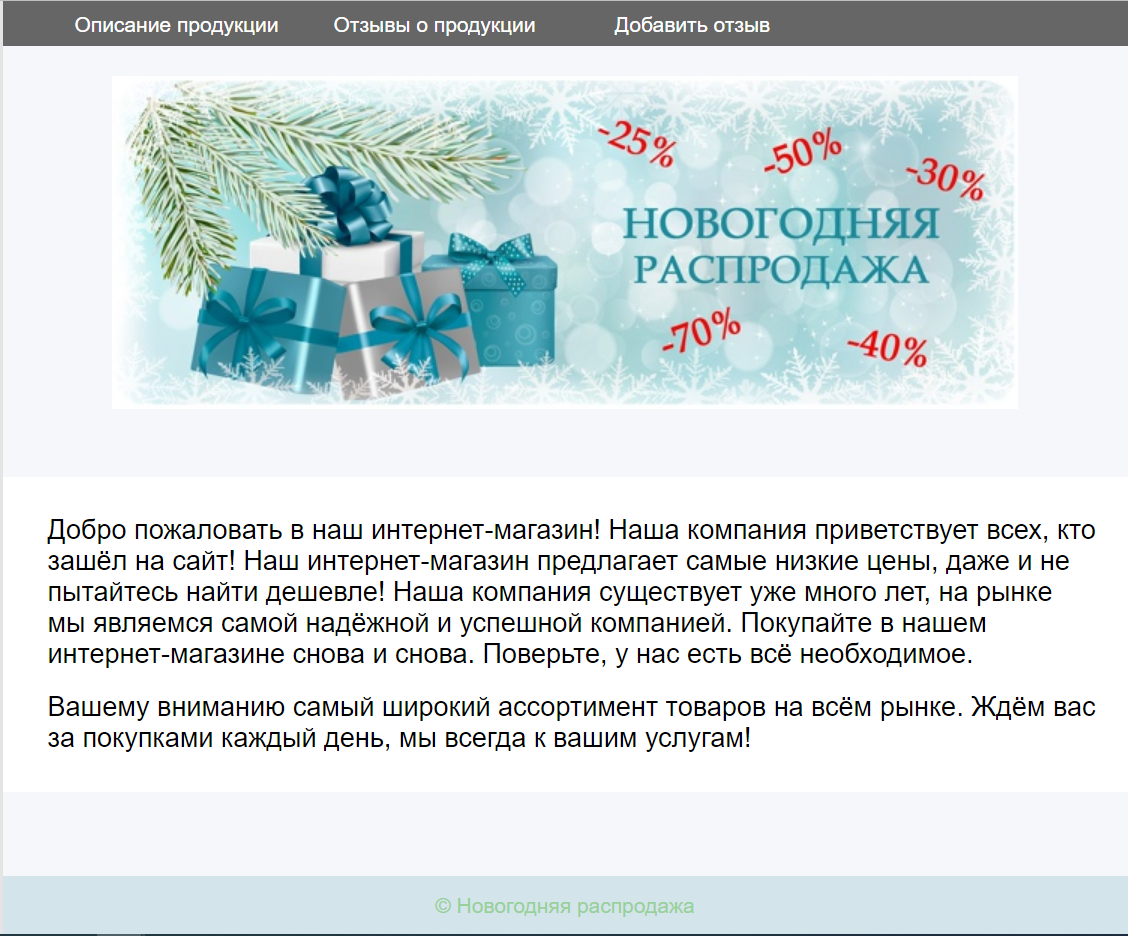
* описание продукции,
* отзывы о продукции,
* добавить отзыв

Запуск приложения: **node app**

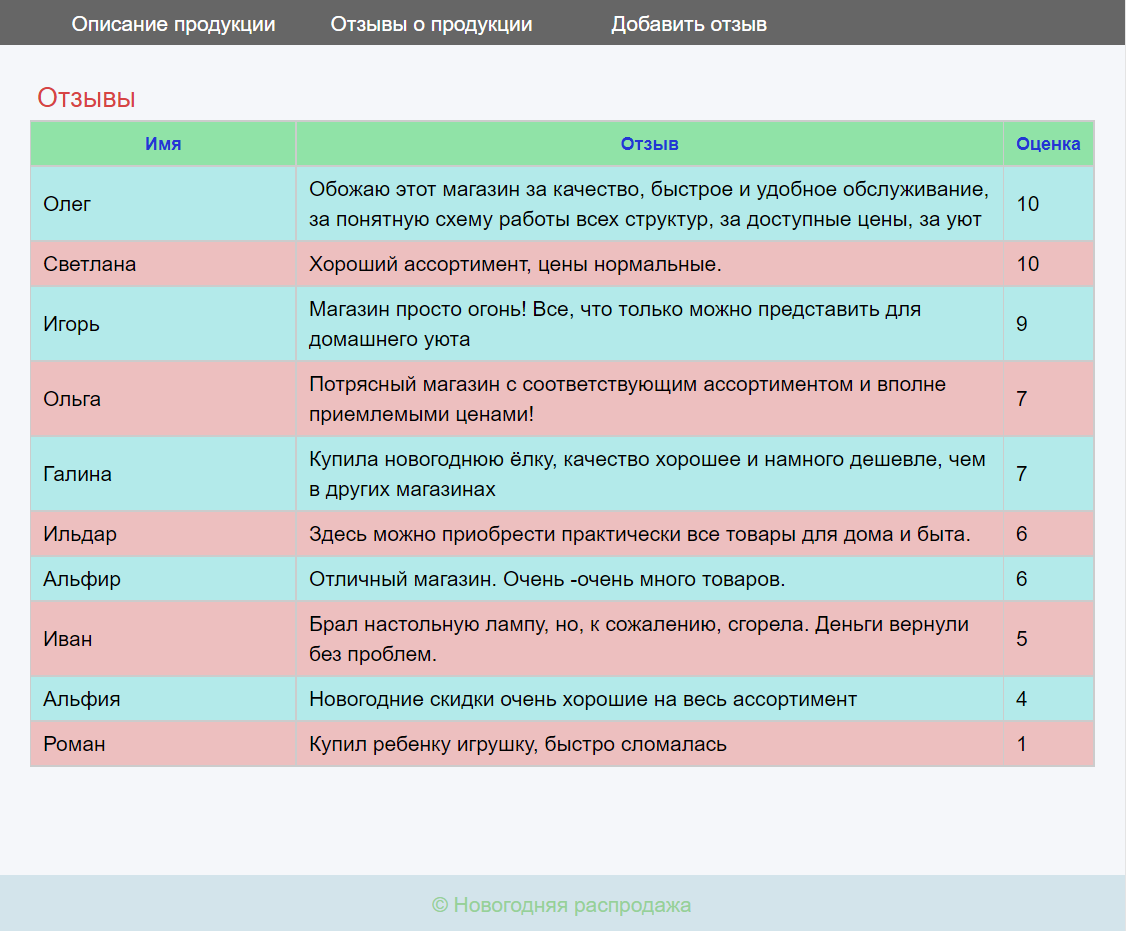
После запуска можно в браузере увидеть главную форму приложения, перейдя по адресу: **localhost:3000/**

**Организация интерфейса пользователя** и функционал визуальных форм приложения.

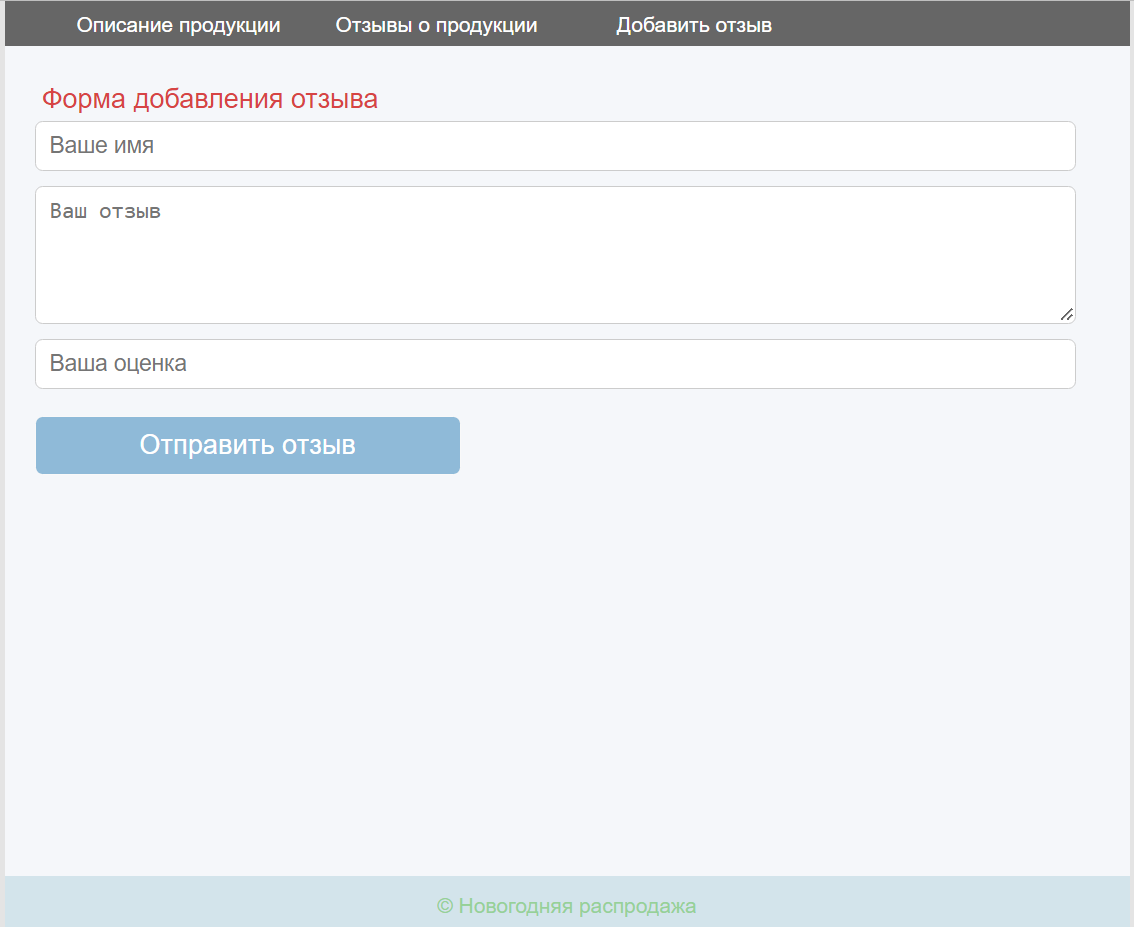
Главная форма приложения:



Дочерняя форма приложения для отображения отзывов:



Форма приложения для набора нового отзыва:



# Заключение

Разработано web-приложение с помощью языка программирования Note.js, который может получить доступ к файловой системе. Данные можно хранить в БД, в текстовом файле и т.д. Если количество данных невелико, то можно использовать базу данных DB Browser (SQLite). Для неё не нужен сервер, данные хранятся в одном файле. Сделан прототип программы (отзывы клиентов о продукте), который можно будет дописать и усовершенствовать.

Выполнены следующие задания:

* добавлено поле "Оценка продукта" (правый столбец таблицы) в таблицу БД и в шаблон html-страницы с отзывами и в обработчик события;
* на шаблоне страницы по вводу отзыва добавлено поле input типа number для ввода оценки и установлено ограничения от 1 до 10 (10 - это высокая оценка);
* сделан вывод на страницу с отзывами, отсортированным в обратном порядке по полю "Оценка продукта" (вверху таблицы отзывы с высокой оценкой, внизу - с низкой);
* разработан дизайн приложения.

Отзывы клиентов — обратная связь поможет довести ваш продукт или услугу до совершенства. Можно узнать, что нравится вашим клиентам, а что их раздражает, что мешает пользоваться вашим продуктом. Спрашивать отзывы и реагировать на них — это способ показать, что вы слышите клиента и готовы ответить на его ожидания.

# 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. Балабанов И.Т. Современные моделирования. / И.Т. Балабанов - СПб: Питер, 2002. - 120 с.
10. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебное пособие. - М.: Финансы и статистика, 2002

# ПРИЛОЖЕНИЕ А

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
2. Наименование системы

Полное наименование разрабатываемой системы – «Интернет-магазин новогодних распродаж».

Краткое наименование – «Интернет-магазин».

1. Наименование заказчика и исполнителя

Организация: Интернет-магазин.

Адрес: ул. Петропавловская, 44.

Телефон: +7 (342) 212-99-66;

Исполнитель: Карпов Максим Валерьевич.

1. 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 web-приложения «Интернет-магазин» можно отнести:

* улучшение оперативного взаимодействия и интеграция web-приложения;
* автоматизировать получение отзывов от клиентов.

Достижение целей приведёт к следующим положительным результатам:

* руководство интернет-магазина сможет быстрее оценивать информацию о предлагаемых товаров и вносить необходимые изменения в предлагаемый ассортимент товаров;
* пользователи web-приложения смогут видеть информационные публикации в отсортированном в порядке убывания оценки о товаре.

1. Характеристика объекта автоматизации

Объектом автоматизации является web-приложение Интернет-магазина.

1. Требования к системе

Общие требования к web-приложению «Интернет-магазин» являются:

* надёжность и работоспособность;
* интуитивно понятный интерфейс;
* лицензионная чистота – применение средств в рамках общего лицензионного соглашения касательно web-приложения;
* соблюдение информационной безопасности и разграничение прав доступа к данным.

1. Требования к способам и средствам связи для информационного обмена между компонентами

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

1. Перспективы развития, модернизация системы

Дальнейшим развитием web-приложения «Интернет-магазин» может быть объединение иных подсистем Интернет-магазина.

1. Требования к квалификации персонала и режиму его работы

Для обеспечения максимальной работоспособности пользователей должны устанавливаться перерывы:

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

Для эксплуатации web-приложения «Интернет-магазин» определены следующие роли:

* системный администратор – должен быть квалифицированным специалистом с практическим опытом выполнения работ по администрированию программных и технических средств. В обязанности входит: установка, модернизация, настройка программного обеспечения, ведение учётных записей web-приложения;
* администратор баз данных – должен быть квалифицированным специалистом с практическим опытом выполнения работ по администрированию РСУБД, проектированию БД, оптимизации производительности, разграничению прав и ролей, а также резервного копирования и обеспечение целостности БД;
* пользователь web-приложения – должен иметь опыт работы с персональным компьютером на уровне опытного пользователя и свободно осуществлять базовые действия с web-приложения посредством браузера с доступом в интернет.

1. 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

Разрабатываемое web-приложение «Интернет-магазин» должно обеспечивать безопасный доступ к данным, предотвращая несанкционированный доступ или модифицирование данных. Web-приложение аутентификации должен обеспечивать защищённый доступ ко всему программному интерфейсу приложения.

Также при разработке web-приложения необходимо соблюдать разграничение прав на публикацию информации отправителю должны быть доступны только те адресаты, которые относятся к его зоне ответственности.

1. Требования по эргономике и технической эстетике

Web-приложение должно иметь удобный и интуитивно понятный графический пользовательский интерфейс. Диалоговый интерфейс должен соблюдать контекст подсистемы организационной коммуникации Интернет-магазина и управления в целом, тем самым действия конечного пользователя должны быть ясны и знакомы.

Пользовательский интерфейс web-приложение также должен аккомпанировать цветовой гамме и общему стилю дизайна Интернет-магазина.

1. Требования к программному обеспечению

При проектировании web-приложения «Интернет-магазин» необходимо эффективно использовать используемое на данный момент в проекте паттерн (шаблон построения архитектуры приложения) проектирования MVC(Model-View-Controller), в качестве серверного окружения используется программная платформа Visual Studio Code (Node.js), а для хранения данных применяется РСУБД DB Browser (SQLite).

1. Требования к техническому обеспечению

Техническое обеспечение системы должно максимально и наиболее эффективно использовать существующее в отделе автоматизации оборудование:

* процессор – 2х Intel Xeon 3.7 ГГц;
* оперативная память – 32 ГБ;
* дисковая система – 1ТБ;
* сетевой адаптер – 1 Гб/с.

1. Порядок контроля и приёмки системы

Приёмо-сдаточные испытания системы проводятся с привлечением сотрудников отдела автоматизации. 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 web-приложение «Интернет-магазин» отдел управления информатизации должен обеспечить выполнение следующих работ:

* определить подразделение и ответственных должностных лиц для внедрения web-приложения;
* обеспечить пользователей руководством, которое поможет быстрее освоить внедрённое web-приложение;
* провести опытную эксплуатацию web-приложения «Интернет-магазин».

# ПРИЛОЖЕНИЕ Б

**Дерево директорий** приложения:

model

m\_feeds.js

m\_intro.js

m\_new\_feed.js

private

│ database.db

│ database.js

│

├───public

│ ├───images

│ │ asd.jpg

│ │ favicon.ico

│ │ nodejs.png

│ │

│ └───stylesheets

│ style.css

│

├───readme

│ 00.js

│ app-tree-.png

│ app\_tree.png

│ feeds.png

│ intro.png

│ new\_feed.png

│

├───routes

│ r\_feeds.js

│ r\_intro.js

│ r\_new\_feed.js

│

└───views

feeds.ejs

feeds.html

footer.html

header.html

intro.ejs

intro.html

new\_feed.ejs

new\_feed.html

**Листинг кода формы авторизации – app.js**

//подключение зависимостей

const express = require('express');

// настройка приложения

const app = express();

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

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

//routing - маршрутизация

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);

// запуск приложения

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

**Листинг кода формы представления** (главная форма приложения) –   
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/asd.jpg" alt="logo">

        </p>

        <br><br>

        <div id="intro">

            <p class="intro">

Добро пожаловать в наш интернет-магазин! Наша компания приветствует всех, кто зашёл на сайт! Наш интернет-магазин предлагает самые низкие цены, даже и не пытайтесь найти дешевле! Наша компания существует уже много лет, на рынке мы являемся самой надёжной и успешной компанией. Покупайте в нашем интернет-магазине снова и снова. Поверьте, у нас есть всё необходимое.

            <p class="intro">

Вашему вниманию самый широкий ассортимент товаров на всём рынке. Ждём вас за покупками каждый день, мы всегда к вашим услугам!

            </p>

        </div>

    </div>

    <%- include("./footer.html") %>

    </div>

</body>

</html>

**Листинг кода формы представления** (дочерняя форма приложения для отображения отзывов) – 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 width="5%">Оценка</th>

                </tr>

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

                <tr>

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

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

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

                </tr>

                <% } %>

            </table>

        </div>

    </div>

    <%- include("./footer.html") %>

    </div>

</body>

</html>

**Листинг кода формы представления** (шаблон для набора нового отзыва) – new\_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><%= 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>

                <input class="txt" name="userNumber" type="number" placeholder="Ваша оценка">

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

            </form>

        </div>

    </div>

    <%- include("./footer.html") %>

    </div>

</body>

</html>

**Листинг кода формы представления** (шаблон для набора нового отзыва) – new\_feeds.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8" />

    <title>Заголовок страницы</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">

        <ul id="navbar">

            <li><a href="./intro.html">Описание продукции</a></li>

            <li><a href="./feeds.html">Отзывы о продукции</a></li>

            <li><a href="./new\_feed.html">Добавить отзыв</a></li>

        </ul>

        <div class="desktop">

            <br>

            <h4>Форма онлайн отзыва</h4>

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

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

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

                <input class="txt" name="nameNumber" type="number" min="1" max="10" placeholder="Ваша оценка">

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

            </form>

        </div>

    </div>

    <div class="footer">

        <p class="footer-text">&copy; </p>

    </div>

    </div>

</body>

</html>

**Листинг кода формы** (библиотеки, которые нужны для работы этого приложения) – package.json

{

  "dependencies": {

    "better-sqlite3": "^7.6.0",

    "ejs": "^3.1.8",

    "express": "^4.18.1",

    "sqlite-sync": "^0.3.7",

    "sqlite3": "^5.0.8"

  }

}

**Листинг кода формы** (дизайн приложения) – style.css

\*{ margin: 0; padding: 0; }

html, body { height: 100%; }

body {

    background-color: rgb(228,228,228);

    font: 14px/20px Arial, sans-serif;

}

.wrapper {

    display: flex;

    flex-direction: column;

    width: 750px;

    min-height: 100%;

    margin: 0 auto;

    background: #F5F7FA;

    font: 14px/20px Arial, sans-serif;

}

.content {

    flex: 1 0 auto;

}

.desktop {

    margin-left: 20px;

    margin-right: 20px;

}

.aligncenter {

    text-align: center;

}

.intro {

    font: 18px Arial, serif;

    margin-top: 15px;

    margin-bottom: 15px;

}

#navbar li:hover ul { display: block; }

#navbar, #navbar ul{ list-style-type: none; }

#navbar {

    background-color: #666;

    height: 30px;

    padding-left: 30px;

    min-width: 600px;

}

#navbar li {

    float: left;

    position: relative;

    height: 100%;

}

#navbar li a {

    display: block;

    padding: 6px;

    width: 160px;

    color: #fff;

    text-decoration: none;

    text-align: center;

}

#navbar ul li { float: none; }

#navbar li:hover {

    background-color: #7cc5e4;

}

.footer {

    flex: 0 0 auto;

    height: 40px;

    background: #d3e4eb;

    display: flex;

}

.footer-text {

    margin: auto;

    color: #96d198;

}

h4 {

    color: #d74444;

    font-size: 18px;

    font-weight: lighter;

    margin: 5px;

}

a {

    text-decoration: none;

    font-weight: lighter;

    color: #0a0909;

    font-size: 14px;

}

a:hover {

    color: rgb(200, 59, 59);

}

table {

    width: 100%;

    border-collapse: collapse;

}

td, th {

    border: 1px solid #CCC;

    padding: 5px 8px 4px 8px;

}

th {

    text-align: center;

    font-size: 12px;

    font-weight: 600;

    background-color: #90e3a7;

    color: rgb(36, 54, 211);

}

tr:nth-child(odd) {

    background-color: #edbfbf;

}

tr:nth-child(even) {

    background-color: #b3eaea;

}

#intro {

    margin:0 auto;

    background:rgb(255, 255, 255);

    padding:10px 20px 10px 30px;

    min-width: 600px;

}

#inline {

    width: 70%;

    margin:0 auto;

    background:rgb(177, 22, 22);

    padding:10px 20px 10px 30px;

    min-width: 600px;

    border-radius: 5px;

}

.txt {

    display:inline-block;

    color:#676767;

    width:95%;

    margin-bottom:10px;

    border:1px solid #CCC;

    padding:5px 9px;

    font-size:15px;

    line-height:1.4em;

    border-radius:5px;

}

.txtarea {

    display:inline-block;

    color:#676767;

    resize: vertical;

    width:95%;

    height:80px;

    margin-bottom: 5px;

    border:1px solid #CCC;

    padding:5px 9px;

    font-size:15px;

    line-height:1.4em;

    border-radius:5px;

}

.txt:focus, .txtarea:focus {

    border-style:solid;

    border-color:#BABABA;

    color:#444;

}

button.send {

    color:#FFFFFF;

    display:block;

    cursor:pointer;

    padding:8px 8px;

    font-size:1.3em;

    border:solid 1px #F9F9F9;

    border-radius:5px;

    width:40%;

    margin: 8px 0px 0px 0px;

    background:#8fbad8;

}

button.send:hover {

    background:#7188c2;

}

**Листинг кода формы** (маршрутизатор) – 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;

РЕЦЕНЗИЯ

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

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

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич