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-31

направления подготовки

09.03.03 Прикладная информатика

Полежаев Егор Вадимович

Проверил:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

2023

Содержание

[Постановка задачи на проектирование 3](#_Toc125566314)

[Анализ технологий проектирования 4](#_Toc125566315)

[Реализация функционала информационной системы 9](#_Toc125566316)

[Заключение 13](#_Toc125566317)

[Список источников 14](#_Toc125566318)

[ПРИЛОЖЕНИЕ А 15](#_Toc125566319)

[ПРИЛОЖЕНИЕ Б 21](#_Toc125566320)

# Постановка задачи на проектирование

Данная программа способствует сбору, хранению и обработке отзывов о товаре. Пользователю программы она дает возможность рассказать о своем опыте и дать оценку, а бизнесмену или обладателю программы понять как его товар видят покупатели и что следует доработать, изменить или наоборот оставить как есть.

Общими требованиями являются:

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

Функциональными требованиями являются:

Для обеспечения информационного обмена компоненты подсистемы должны взаимодействовать с объединённой информационной базой данных. Благодаря хранению данных в различных схемах, таких как мероприятия или поручения программа может объединить эти данные представив их как единый информационный поток.

Требования к интерфейсу:

Программа должна иметь удобный и интуитивно понятный графический пользовательский интерфейс. Диалоговый интерфейс должен соблюдать контекст подсистемы организационной коммуникации университета и управления в целом, тем самым действия конечного пользователя должны быть ясны и знакомы.

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

Требования к защищённости:

Разрабатываемая информационная система должна обеспечивать безопасный доступ к данным, предотвращая несанкционированный доступ или модифицирование данных. Модуль аутентификации должен обеспечивать защищённый доступ ко всему программному интерфейсу приложения, за исключением статичной формы авторизации.

Требования к кроссплатформенности, к операционной системе, к ресурсам памяти:

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.

# Анализ технологий проектирования

SQLite

SQLite – это внутрипроцессная библиотека, которая реализует автономный, безсерверный, транзакционный механизм базы данных SQL с нулевой конфигурацией. Это база данных с нулевой конфигурацией, что означает, что, как и другие базы данных, вам не нужно настраивать ее в вашей системе.

Движок SQLite не является автономным процессом, как другие базы данных, вы можете связать его статически или динамически в соответствии с вашими требованиями к вашему приложению. SQLite получает доступ к своим файлам хранения напрямую.

SQLite Краткая история

2000 – Д. Ричард Хипп разработал SQLite для администрирования программы, не требующей администрирования.

2000 – В августе выпущен SQLite 1.0 с GNU Database Manager.

2011 – Hipp объявил о добавлении интерфейса UNQl к базе данных SQLite и разработке UNQLite (базы данных, ориентированной на документы).

2000 – Д. Ричард Хипп разработал SQLite для администрирования программы, не требующей администрирования.

2000 – В августе выпущен SQLite 1.0 с GNU Database Manager.

2011 – Hipp объявил о добавлении интерфейса UNQl к базе данных SQLite и разработке UNQLite (базы данных, ориентированной на документы).

Почему SQLite?

SQLite не требует отдельного серверного процесса или системы для работы (без сервера).

SQLite поставляется с нулевой конфигурацией, что означает отсутствие необходимости в настройке или администрировании.

Полная база данных SQLite хранится в одном межплатформенном файле на диске.

SQLite очень маленький и легкий, менее 400 КБ полностью сконфигурирован или менее 250 КБ без дополнительных функций.

SQLite является автономным, что означает отсутствие внешних зависимостей.

Транзакции SQLite полностью совместимы с ACID, что обеспечивает безопасный доступ из нескольких процессов или потоков.

SQLite поддерживает большинство функций языка запросов, представленных в стандарте SQL92 (SQL2).

SQLite написан на ANSI-C и предоставляет простой и удобный API.

SQLite доступен в UNIX (Linux, Mac OS-X, Android, iOS) и Windows (Win32, WinCE, WinRT).

MySQL

MySQL — это реляционная система управления базами данных с открытым исходным кодом. В настоящее время эта СУБД одна из наиболее популярных в веб-приложениях — подавляющее большинство CMS использует именно MySQL (часто только её, без альтернатив), а почти все веб-фреймворки поддерживают MySQL уже на уровне базовой конфигурации (без дополнительных модулей).

Из преимуществ СУБД MySQL стоит отметить простоту использования, гибкость, низкую стоимость владения (относительно платных СУБД), а также масштабируемость и производительность.

MySQL позволяет хранить целочисленные значения со знаком и беззнаковые, длиной в 1, 2, 3, 4 и 8 байтов, работает со строковыми и текстовыми данными фиксированной и переменной длины, позволяет осуществлять SQL-команды SELECT, DELETE, INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов и функций в SELECT- и WHERE- частях запросов, работает с GROUP BY и ORDER BY, поддерживает групповые функции COUNT(), AVG(), STD(), SUM(), MAX() и MIN(), позволяет использовать JOIN в запросах, в т.ч. LEFT OUTER JOIN и RIGHT OUTER JOIN, поддерживает репликацию, транзакции, работу с внешними ключами и каскадные изменения на их основе, а также обеспечивает многие другие функциональные возможности.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Есть и другие типы таблиц, разработанные сообществом.

СУБД MySQL появилась в 1995. Написана на C и C++, протестирована на множестве различных компиляторов и работает на различных платформах. C 2010 года разработку и поддержку MySQL осуществляет корпорация Oracle. Продукт распространяется как под GNU GPL, так и под собственной коммерческой лицензией. Однако по условиям GPL, если какая-либо программа включает исходные коды MySQL, то и эта программа тоже должна распространяться по лицензии GPL. Для нежелающих открывать исходные тексты своих программ как раз предусмотрена коммерческая лицензия, которая, в дополнение к возможности разработки под «закрытой» лицензией, обеспечивает качественную сервисную поддержку. Сообществом разработчиков MySQL созданы различные ответвления — Drizzle, OurDelta, Percona Server и MariaDB, все эти ответвления уже существовали на момент получения прав на MySQL корпорацией Oracle.

Python

Python — популярнейший язык, используемый в разных ИТ-сферах: и в машинном обучении, и при разработке программных приложений, и для написания скриптов автоматизации, и во многих других случаях. В этой статье мы расскажем историю появления Python и поговорим, какими особенностями он обладает.

Python является высокоуровневым языком, который доминирует во многих областях. Его можно смело назвать универсальным языком общего назначения.

Судите сами — «Питону» удалось:

— захватить часть рынка web-разработки;

— обосноваться на рынке десктопных приложений;

— стать лидером в сфере машинного обучения.

Благодаря своей универсальности, «Пайтон» выделяется среди других языков (languages), вызывая большой интерес. Поклонники Python нередко называют его языком номер 1 (Python — is a language № One). А современные рейтинги и сервисы статистики PYPL, TIOBE и statista.com отчасти это подтверждают, т. к. там «Пито» стабильно находится в топ-3, причем уже не первый год подряд.

NODE JS

Node (или более формально Node.js) - кросплатформенная среда исполнения с открытым исходным кодом, которая позволяет разработчикам создавать всевозможные серверные инструменты и приложения используя язык JavaScript. Среда исполнения предназначена для использования вне контекста браузера (т.е. выполняется непосредственно на компьютере или на серверной ОС). Таким образом, среда исключает API-интерфейсы JavaScript для браузера и добавляет поддержку более традиционных OS API-интерфейсов, включая библиотеки HTTP и файловых систем.

С точки зрения веб-серверной разработки Node имеет ряд преимуществ:

Отличная производительность! Node был разработан для оптимизации пропускной способности и масштабируемости в веб-приложениях и очень хорошо справляется со многими распространёнными проблемами веб-разработки (например, веб-приложения реального времени).

Код написан на «обычном старом JavaScript», а это означает, что затрачивается меньше времени при написании кода для браузера и веб-сервера связанное с «переключением технологий» между языками.

JavaScript является относительно новым языком программирования и имеет преимущества от улучшения дизайна языка по сравнению с другими традиционными языками для веб-серверов (например, Python, PHP, и т.д.). Многие другие новые и популярные языки компилируются/конвертируются в JavaScript, поэтому вы можете также использовать CoffeeScript, ClosureScript, Scala, LiveScript, etc.

Менеджер пакетов Node (NPM) обеспечивает доступ к сотням тысяч многоразовых пакетов. Он также имеет лучшее в своём классе разрешение зависимостей и может также использоваться для автоматизации большинства инструментов построения.

Он портативен, имеет версии для Microsoft Windows, OS X, Linux, Solaris, FreeBSD, OpenBSD, WebOS, и NonStop OS. Кроме того, он имеет хорошую поддержку среди многих хостинг-провайдеров, которые часто предоставляют конкретную инфраструктуру и документацию для размещения сайтов, работающих на Node.

Он имеет очень активную стороннюю экосистему и сообщество разработчиков, которые всегда готовы помочь.

Вы можете использовать Node.js для создания простого веб сервера используя пакет Node HTTP.

Почему Node.js?

Язык отвечает всем минимальным требованиям для создания программы, легко читаем, исправляем и быстр в написании. Паралельно с этой на превый вид простотой обладает широкими возможностями и большими мощностями.

MVC

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Концепция MVC была описана Трюгве Реенскаугом в 1978 году, работавшим в научно-исследовательском центре «Xerox PARC» над языком программирования «Smalltalk». Позже, Стив Бурбек реализовал шаблон в Smalltalk-80.  
 Окончательная версия концепции MVC была опубликована лишь в 1988 году в журнале Technology Object.  
 Впоследствии шаблон проектирования стал эволюционировать. Например, была представлена иерархическая версия HMVC; MVA, MVVM.  
 Дальнейший виток популярности привнесло развитие фреймворков, ориентированных на быструю развёртку, на языках Python, PHP и Ruby: Django, Laravel и Ruby on Rails соответственно. На момент 2017 года, фреймворки с MVC заняли заметные позиции по отношению к остальным фреймворкам без этого шаблона.

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

# 

# Реализация функционала информационной системы

**Логическая модель базы данных**

Сведения о формате CSV

CSV-файлы (значения, разделенные запятыми) обычно используются для обмена табличными данными между системами в обычном тексте. В основном они содержат строку заголовка, которая предоставляет имена столбцов для данных, но в остальном они считаются частично структурированными. Это связано с тем, что CSV-файлы изначально не могут представлять иерархические или реляционные данные. Связи между данными обычно обрабатываются с использованием нескольких CSV-файлов. Внешние ключи хранятся в столбцах одного или нескольких файлов, однако связи между этими файлами не выражаются самим форматом. Файлы в формате CSV могут использовать другие разделители, помимо запятых, например символы табуляции или пробелы.

Несмотря на ограничения, CSV-файлы являются популярным выбором для обмена данными, так как они поддерживаются широким рядом бизнес-приложений, потребительских и научных приложений. Например, программы баз данных и электронных таблиц могут импортировать и экспортировать CSV-файлы. Аналогичным образом большинство модулей обработки пакетных и потоковых данных (например, Spark и Hadoop) изначально поддерживают сериализацию и десериализацию CSV-файлов и предлагают способы применения схемы при чтении. Они упрощают работу с данными, предоставляя варианты выполнения запроса и сохраняя сведения в более эффективном формате для быстрой обработки.

CSV-файлы чаще используются для экспорта и импорта данных или их обработки для задач аналитики и машинного обучения. Файлы в формате JSON имеют те же преимущества, но чаще используются в решениях обмена горячими данными. Документы JSON часто отправляются с веб- и мобильных устройств, выполняющих онлайн-транзакции, устройств Интернета вещей для односторонней или двунаправленной связи или клиентских приложений, обменивающихся данными со службами SaaS и PaaS или бессерверными архитектурами.

CSV-файлы облегчают обмен данными между разными системами или устройствами. Частично структурированные форматы позволяют переносить практически любые типы данных, а универсальная поддержка упрощает работу. Оба формата могут использоваться как источник истины в случаях, когда обработанные данные хранятся в двоичных форматах для более эффективного выполнения запроса.

**Запросы к базе данных**

Пример №1

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

md\_index.feeds = get\_records('./private/feeds.csv');

res.render('index', md\_index);

});

При переходе на главную страницу информационной системы объявляется функция с участием csv-файлу в последствии на страницу подаются данные

Пример №2

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

let record = get\_record(req.body.name, req.body.feed, req.body.estimation); // находим по имени в шаблоне

fs.appendFileSync('./private/feeds.csv', record);

res.redirect("/"); // возвращаемся на главную

});

С формы обратной связи происходит запись данных которые ввел пользователь. Все они передаются в csv-файл который будет дальше использоваться для вывода отзывов на главную страницу сайта

**Интерфейс пользователя**

При переходе на главную страницу сайта пользователю выводится актуальная таблица с отзывами в которой содержатся столбцы «Дата», «Время», «Имя», «Отзыв», «Оценка»(Рис.1.1). Она несет лишь информационную функцию и на прямую не взаимодействует с пользователем, однако есть 3 кнопки «Оставить новый отзыв», «Наивысшая оценка» и «Наименьшая оценка», они как раз и способствуют интерактивности с ИС.  
 При нажатии на «Наивысшая оценка» или «Наименьшая оценка» данные в таблице сортируются в соответствии с нажатой кнопкой(Рис.1.2). При нажатии на «Оставить новый отзыв» пользователь переходит на новую страницу в которой может заполнить свободные поля и ввести новые данные которые отобразятся в таблице на главной странице.

Последовательность работы:

1) Переход на главную страницу. Вывод таблицы и кнопок.  
 1.1) Нажатие «Наивысшая оценка» - сортировка по убыванию

1.2) Нажатие «Наименьшая оценка» - сортировка по возрастанию

2) Переход на страницу «Форма онлайн отзыва». Вывод поля и кнопок.

2.1) Ввод в поля данные и нажатие кнопки «Отправить отзыв»

\*Возвращает пользователя на главную страницу, данные обно вятся и в ней появится новая запись.

2.2.) Нажатие кнопки «Вернуться»

\* Возвращает пользователя на главную страницу, данные НЕ об новятся и в ней НЕ появится новая запись.

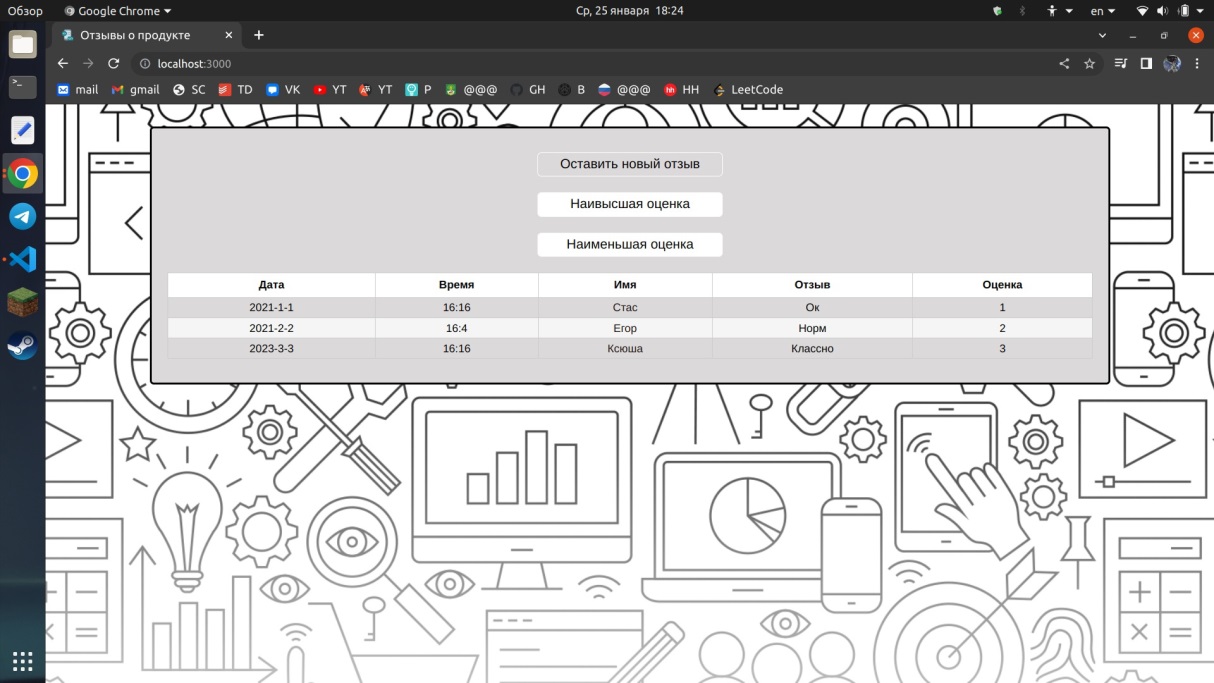


Рис.1.1 Главная страница

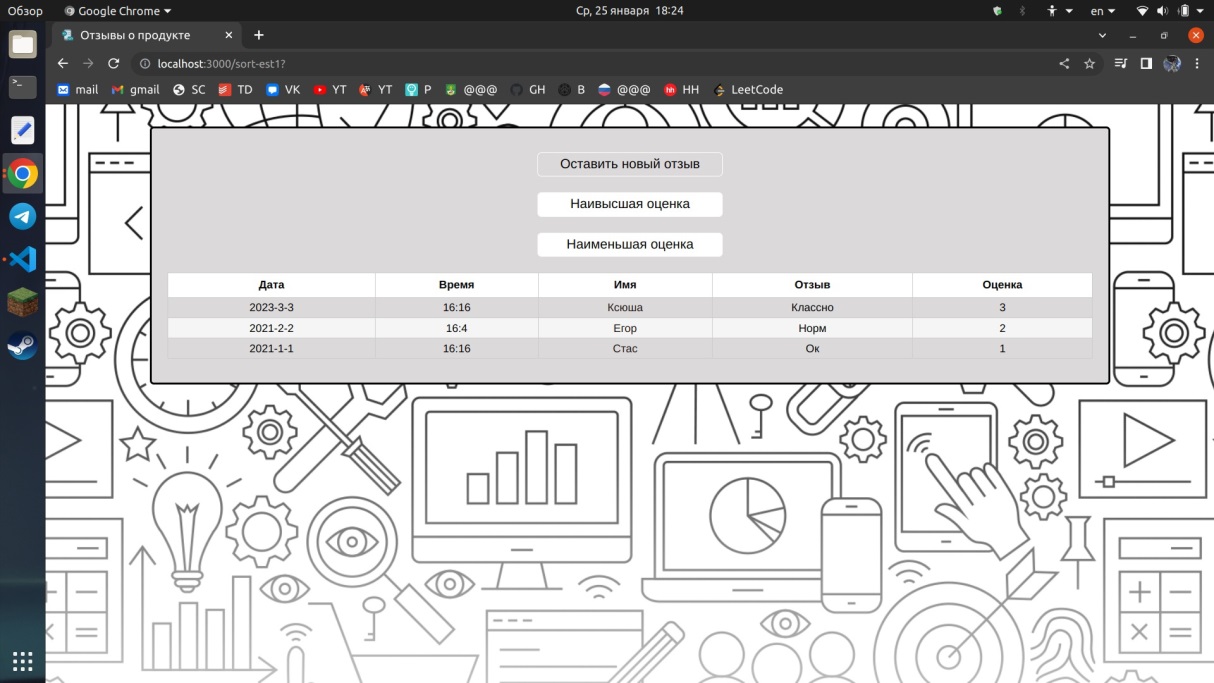


Рис.1.2 Главная страница(отсортированная)

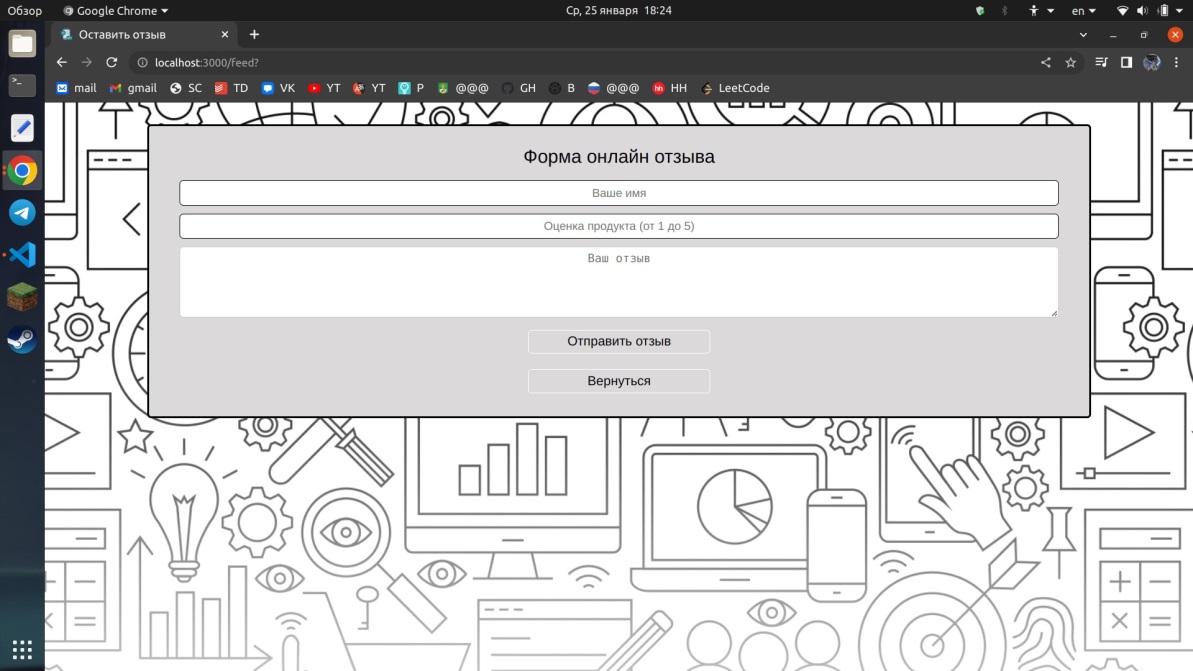


Рис.2 Форма онлайн отзыва

# Заключение

В процессе работы производился анализ бизнес-процесса который помог грамотно понять необходимый функционал информационной системы, определение модулей который необходимы к добавлению, а так же тех которые наоборот не нужны и будут лишь замедлять и усложнять всю информационную систему.

Была спроектирована ,и в последствии реализована, архитектура приложения по схеме MVC с соблюдением всех особенностей используемой схемы и основных положений логики.

Данная автоматизация бизнес-процесса крайне простая и быстрая в реализации, однако её эффективность всё ещё остается большой, т.к. без обладания отзывами продавец/бизнесмен или кто-либо ещё рискует в долгосрочной перспективе уменьшить свой доход вследствие незнания о своих недочетах или минусах своего товара.

В перспективе эта информационная система может получить дополнительные модули, например, модуль позволяющий регистрироваться пользователю, вводить свои личные данные для последующего использования и анализа, модуль позволяющий редактировать введенный комментарий, модуль позволяющий переводить сайт на иностранные языки.

# 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. СЕРИАЛИЗАЦИЯ СТРУКТУР ДАННЫХ ДЛЯ ХРАНЕНИЯ И ПЕРЕДАЧИ В ИНФОРМАЦИОННЫХ СИСТЕМАХ. МЕТОДЫ И СРЕДСТВА Режим доступа: [http://book.sarov.ru/wp-content/uploads/2018/12/16-molodej-v2-2018-45.pdf](http://book.sarov.ru/wp-content/uploads/2018/12/16-molodej-v2-2018-45.pdf%20%2010) .
10. Работа с CSV- и JSON-файлами в контексте решений для работы с данными. Режим доступа: <https://learn.microsoft.com/ru-ru/azure/architecture/data-guide/scenarios/csv-and-json>

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

2021

1. Общие сведения
2. Наименование системы

Полное наименование разрабатываемой системы – «Автоматизированный модуль «Отзывы о продукте».

Краткое наименование – «Отзывы о продукте».

1. Наименование заказчика и исполнителя

Организация: ФГБОУ ВО Пермский ГАТУ.

Адрес: ул. Петропавловская, 23.

Телефон: +7 (342) 217-90-66;

Исполнитель: Полежаев Егор Вадимович.

1. Плановые сроки начала и окончания работ

Дата начала работ: 01.09.2022.

Дата окончания работ: 31.12.2022.

1. Назначение и цели создания системы

К целям создания модуля «Отзывы о продукте» можно отнести:

* улучшение оперативного взаимодействия и интеграция модулей;
* автоматизировать опрос покупателей продуктов.

Достижение целей приведёт к следующим положительным результатам:

* владелец сможет получить актуальную картину качества своего продукта и своих услуг;
* пользователи смогут видеть отзывы в отсортированном в порядке убывания/возрастания оценки;

1. Характеристика объекта автоматизации

Объектом автоматизации является бизнес-процесс.

1. Требования к системе

Общие требования к модулю «Отзывы о продукте» являются:

* надёжность и работоспособность;
* интуитивно понятный интерфейс;
* лицензионная чистота – применение средств в рамках общего лицензионного соглашения касательно корпоративного портала;
* 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. Благодаря хранению данных в различных схемах, таких как мероприятия или поручения модуль «Отзывы о продукте» может объединить эти данные представив их как единый информационный поток.

1. Перспективы развития, модернизация системы

Дальнейшим развитием модуля «Отзывы о продукте» может быть объединение иных подсистем корпоративного портала.

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 модуля «Отзывы о продукте» 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 портала;
* администратор баз данных – должен быть квалифицированным специалистом с практическим опытом выполнения работ по администрированию СУБД, проектированию БД, оптимизации производительности, разграничению прав и ролей, а также резервного копирования и обеспечение целостности БД;
* пользователь портала – должен иметь опыт работы с персональным компьютером на уровне опытного пользователя и свободно осуществлять базовые действия с корпоративным порталом 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

Разрабатываемый информационный модуль «Отзывы о продукте» 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 организационной коммуникации университета и управления в целом, тем самым действия конечного пользователя должны быть ясны и знакомы.

1. Требования к программному обеспечению

При проектировании информационного модуля «Живая лента» необходимо эффективно использовать используемое на данный момент в проекте корпоративного портала применяется веб-фреймворк Node.js, в качестве серверного окружения используется программная платформа Node.js, а для хранения данных применяется CSV-файлы.

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 – 2 х 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 модуля «Отзывы о продукте» отдел управления информатизации должен обеспечить выполнение следующих работ:

* определить подразделение и ответственных должностных лиц для внедрения информационного модуля;
* обеспечить пользователей руководством, которое поможет быстрее освоить внедрённый модуль;
* провести опытную эксплуатацию модуля «Отзывы о продукте».

# ПРИЛОЖЕНИЕ Б

**Листинг кода формы обратной связи**

**1. model-feed.js**

const md\_feed = {

title: "Оставить отзыв",

heading: "Форма онлайн отзыва",

button: "Отправить отзыв"

};

module.exports = { md\_feed }

**2. feed.js**

const express = require('express');

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

const fs = require('fs');

const { md\_feed } = require('../models/model-feed');

const router = express.Router();

// добавим данные в файл

const get\_record = (nameUser, feed, estimation) => {

let date = new Date();

let year = date.getFullYear();

let month = date.getMonth()+1;

let day = date.getDate();

let hour = date.getHours();

let min = date.getMinutes();

let date\_time = `${year}-${month}-${day}`;

let time\_date = `${hour}:${min}`;

feed = feed.replace(/(\r\n|\n|\r)/gm, ' ');

return `${date\_time}|${time\_date}|${nameUser}|${feed}|${estimation}\n`;

};

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

res.render('feed', md\_feed); // render view

});

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

let record = get\_record(req.body.name, req.body.feed, req.body.estimation); // находим по имени в шаблоне

fs.appendFileSync('./private/feeds.csv', record);

res.redirect("/"); // возвращаемся на главную

});

module.exports = router;

**3. feed.еjs**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

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

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

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

</head>

<body>

<br>

<div id="inline">

<h2><%=heading%></h2>

<form action="/feed" method="post"> <!-- вызов события /feed -->

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

<input id="estimation" class="txt" name="estimation" type="estimation" placeholder="Оценка продукта (от 1 до 5)">

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

<button id="send"><%=button%></button>

</form>

<form> <!-- вызов главной страницы - событие / -->

<button id="back" type="submit" formaction="/">Вернуться</button>

</form>

</div>

</body>

</html>

**Листинг кода главной страницы**

**1. model-index.js**

const md\_index = {

title: "Отзывы о продукте",

heading: "Отзывы",

button: "Оставить новый отзыв",

feeds: []

}

module.exports = { md\_index }

**2. index.js**

const express = require('express');

const fs = require('fs');

const csvjson = require('csvjson'); // npm i csvjson

let { md\_index } = require('../models/model-index');

const router = express.Router();

const get\_records = (nameFile, del='|') => {

let textCSV = fs.readFileSync(nameFile, { encoding: 'utf8' }); // из csv файла

return csvjson.toObject(textCSV, { delimiter: del }); // в массив json

}

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

md\_index.feeds = get\_records('./private/feeds.csv');

res.render('index', md\_index);

});

router.get('/sort-est1', (req, res, next) => {

md\_index.feeds = get\_records('./private/feeds.csv');

md\_index.feeds = md\_index.feeds

.sort((a,b) => a.estimation>b.estimation? -1: +1);

res.render('index', md\_index);

});

router.get('/sort-est2', (req, res, next) => {

md\_index.feeds = get\_records('./private/feeds.csv');

md\_index.feeds = md\_index.feeds

.sort((a,b) => a.estimation<b.estimation? -1: +1);

res.render('index', md\_index);

});

module.exports = router;

**3. index.еjs**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

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

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

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

</head>

<body>

<br>

<div id="inline">

<br>

<form action="/feed" method="get"> <!-- вызов события /feed -->

<button id="send"><%=button%></button>

</form>

<!-- <h4><%=heading%></h4> -->

<form action="/sort-est1" method="get"> <!-- Сортировка 1 -->

<button id="sort-est1">Наивысшая оценка</button>

</form>

<form action="/sort-est2" method="get"> <!-- Сортировка 2 -->

<button id="sort-est2">Наименьшая оценка</button>

</form>

<table>

<tr>

<th>Дата</th>

<th>Время</th>

<th>Имя</th>

<th>Отзыв</th>

<th>Оценка</th>

</tr>

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

<tr>

<td class="name"><%=feed.date%></td>

<td class="name"><%=feed.time%></td>

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

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

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

</tr>

<%} %>

</table>

<br>

<!-- <button id="sort-time"><%=button%></button> -->

<!-- <button id="sort-date/time"><%=button%></button> -->

</div>

</body>

</html>

**Листинг кода приложения**

// подключение зависимостей

const express = require('express');

const params = require('./private/config.json').debug;

// настройка приложения

const app = express();

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

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

// routing - маршрутизация

const router\_index = require('./routes/index');

const router\_feed = require('./routes/feed');

app.use('/', router\_index);

app.use('/feed', router\_feed);

// запуск приложения

app.listen(params.port, params.hostname, () => {

console.log(`> ${params.hostname}:${params.port}/\n> to stop: Ctrl+C`);

});