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-21а

направления подготовки

09.03.03 Прикладная информатика

Смыслов Артем Георгиевич

Проверил:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

2022

СОДЕРЖАНИЕ

[1. Постановка задачи на проектирование 3](#_Toc109193178)

[2. Анализ технологий проектирования 5](#_Toc109193179)

[2.1. Обзор реляционных баз данных 5](#_Toc109193180)

[2.2. Обзор языков программирования 7](#_Toc109193181)

[3. Реализация функционала информационной системы 10](#_Toc109193182)

[Заключение 14](#_Toc109193183)

[Список источников 15](#_Toc109193184)

[ПРИЛОЖЕНИЕ А 16](#_Toc109193185)

[ПРИЛОЖЕНИЕ Б 21](#_Toc109193186)

[ПРИЛОЖЕНИЕ В 32](#_Toc109193187)

# 1. Постановка задачи на проектирование

Данная 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, где будут храниться отзывы пользователей, купивших и пользовавшихся продуктом компании. В web-приложении пользователи сами будут вольны оставлять отзыв о продукте или же прочитать их от других пользователей. Благодаря такой системе можно будет быстро и легко анализировать оценки пользователей о продукте, что послужит хорошей связью между компанией и её клиентами.

Интерфейс web-приложения должен быть интуитивно понятен большинству пользователей сети интернет. Стиль web-приложения должен соответствовать компании и особенно продукту, который оно представляет. Также интерфейс должен быть удобен в использовании как для пользователей, так и для работников компании, анализирующих отзывы.

Т.к. это web-приложения, каких-то серьёзных требований к операционной системе, ресурсам памяти или кроссплатформенности нет. В основном всё будет зависеть от того, будет ли работать браузер пользователя интернета. Конечно, желательно, чтобы у ПК пользователя имелось минимальная оперативная память 2 ГБ.

База данных должна храниться в проверенном источнике, чтобы она не была подвержена удалению или изменению. Также должны иметься бэкапы базы данных. Т.к. пользователи вольны оставлять отзывы без регистрации, поэтому особо персональных данных украсть невозможно, потому что их попросту нет.

# 2. Анализ технологий проектирования

2.1. Обзор реляционных баз данных

Реляционная база данных – это набор данных с предопределенными связями между ними. Такие базы данных являются самыми востребованными, ведь они очень удобны для построения связей между данными. Для рассмотрения возьмем такие РСУБД как: SQLite, MySQL и PostegreSQL.

**MySQL -** Одной из самых популярных СУБД на сегодняшний день является MySQL, распространяемая свободно (с некоторыми ограничениями). Эта серверная система способна эффективно функционировать во взаимодействии с интернет-сайтами и веб-приложениями. При этом она проста в освоении, что лишь увеличивает ее популярность, особенно среди «айтишников»-любителей. Примечательно, что результатом огромной популярности MySQL стало появление в интернете множества руководств по освоению системы, а также огромного количества всевозможных плагинов и расширений, упрощающих работу с этой системой. Это в свою очередь лишь придало системе еще большей популярности.

**Преимущества:**

* Простота в использовании. MySQL достаточно легко инсталлируется, а наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных;
* Безопасность. Система изначально создана таким образом, что множество встроенных функций безопасности в ней работают по умолчанию;
* Масштабируемость. Являясь весьма универсальной СУБД, MySQL в равной степени легко может быть использована для работы и с малыми, и с большими объемами данных;
* Скорость. Высокая производительность системы обеспечивается за счет упрощения некоторых используемых в ней стандартов.

**Недостатки:**

* Недостаточная надежность. В вопросах надежности некоторых процессов по работе с данными (например, связь, транзакции, аудит) MySQL уступает некоторым другим СУБД;
* Низкая скорость разработки. Как и многим другим программным продуктам с открытым кодом, MySQL не достает некоторого технического совершенства, что порой сказывается на эффективности процессов разработки.

Сам факт того, что MySQL является самой популярной СУБД на рынке, говорит об ее универсальности и предпочтительности в большинстве ситуаций. В частности, ее используют в следующих ситуациях:

- при распределённых операциях, когда функционала SQLite (другая популярная система) не хватает;

- когда требуется обеспечить высокий уровень безопасности, что MySQL делает с успехом;

- для работы с интернет-страницами и веб-приложениями, поскольку MySQL является наиболее удобной СУБД для этой сферы применения.

**PostgreSQL** – это довольно популярная СУБД. Кроме того, он устойчив по своей природе с высокой производительностью и многозадачностью. PostgreSQL поддерживает много типов данных, таких как строковые, числовые, дата и время, такие как My SQL. Кроме того, он поддерживает типы данных для геометрических фигур, изображений, сетевых адресов, битовых строк, текстового поиска и записей JSON и т. д.

**Преимущества:**

* Легко использовать;
* Имеет пользовательский тип данных;
* Открытый источник;
* Большая поддержка сообщества.

**Недостатки:**

* PostgreSQL не очень хорош в плане производительности;
* Он не так популярен, как другие системы управления базами данных;
* Что касается скорости, PostgreSQL не заслуживает внимания по сравнению с другими инструментами.

Рекомендуется использовать РСУБД PostegreSQL в случаях, где:  
- нужна поддержка внешних ключей, триггеров и представлений, позволяющих скрывать сложности БД от приложения для отказа от сложных команд SQL;

- важна возможность создания селектов (вложенных подзапросов);

- нужна возможность создания сложных команд SQL (за счет соответствия SQL-стандартам ANSI);

- важна целостность данных;

- требуется поддержка MVCC для предоставления одновременного доступа к базе данных большому количеству пользователей на чтение и запись;

**SQLite** — это быстрая и легкая встраиваемая однофайловая СУБД на языке C, которая не имеет сервера и позволяет хранить всю базу локально на одном устройстве. Для работы SQLite не нужны сторонние библиотеки или службы. Это база данных с нулевой конфигурацией, что означает, что, как и другие базы данных, вам не нужно настраивать ее в вашей системе.

**Особенности:**

* SQLite очень маленький и легкий, менее 400 КБ полностью сконфигурирован или менее 250 КБ без дополнительных функций;
* Полная база данных SQLite хранится в одном межплатформенном файле на диске;
* SQLite является автономным, что означает отсутствие внешних зависимостей;
* SQLite не требует отдельного серверного процесса или системы для работы (без сервера);
* SQLite поддерживает большинство функций языка запросов, представленных в стандарте SQL92 (SQL2).

Мой итоговый выбор пал на SQLite, т.к. мне не требовалась большая СУБД для больших баз данных, а также её не нужно было настраивать, только установить.

2.2. Обзор языков программирования

Главной состовляющей любого приложения является язык программирования. При выборе основного языка программирования я опирался на три из существующих: C#, Python, и в дополнение Node.js, что не совсем является языком программирования.

**C#** — объектно-ориентированный, ориентированный на компоненты язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО.

**Преимущества:**

* Большое количество «синтаксического «сахара» — специальных конструкций, разработанных для понимания и написания кода. Они не имеют значения при компиляции;
* Низкий порог вхождения. Синтаксис C# имеет много схожего с другими языками программирования, благодаря чему облегчается переход для программистов. Язык C# часто признают наиболее понятным и подходящим для новичков.

**Недостатки:**

* Приоритетная ориентированность на платформу Windows;
* Язык бесплатен только для небольших фирм, индивидуальных программистов, стартапов и учащихся . Крупной компании покупка лицензионной версии этого языка обойдётся в круглую сумму.

**Python –** это язык программирования, ориентированный на повышение производительности разработчика, читаемости [кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) и его качества, а также на обеспечение переносимости написанных на нём программ.

**Преимущества:**

* Расширяемость и гибкость;
* Стандартизированность. У Python есть единый стандарт для написания кода — Python Enhancement Proposal или PEP, благодаря чему язык остаётся читабельным даже при переходе от одного программиста к другому;
* Широта применения.

**Недостатки:**

* Низкая производительность;
* Динамическая типизация.

**Node** или **Node.js** — программная платформа, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Эта платформа позволяет писать серверный код для динамических веб-страниц и веб-приложений, а также для программ командной строки.

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.

Выбор серверного JavaScript для бэкенда обеспечивает проекту ряд преимуществ:

* рост эффективности разработки благодаря использованию одного языка для фронт- и бэкенда и возможности переиспользования кода;
* возможность использовать npm — самый большой пакетный менеджер;
* более простой по сравнению с другими стеками поиск исполнителей, так как JavaScript входит в число самых популярных языков программирования.

Эта платформа очень пригодится, т.к. мне нужно будет работать не только с самим web-приложением, но и с базой данных, поэтому Node.js и JavaScript будут лучшим выбором.

# 3. Реализация функционала информационной системы

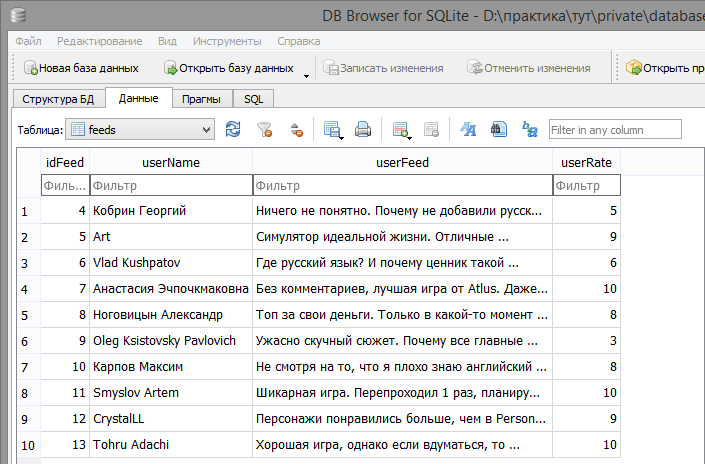
**Логическая модель базы данных**

Рис. 1 – таблица базы данных

**Запросы к базе данных**

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

**FROM** ('таблица; обязательно')

**WHERE** ('условие/фильтрация, например, river = 'Volga'; необязательно')

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

**INSERT INTO** («вставка значения»; необязательно)

**SELECT, FROM** — обязательные элементы запроса, которые определяют выбранные столбцы, их порядок и источник данных. Выбрать все (обозначается как \*), либо же перечислить названия полей, которые ходим выбрать. После «FROM» указать название таблицы.

**WHERE** — необязательный элемент запроса, который используется, когда нужно отфильтровать данные по нужному условию. Очень часто внутри элемента where используются IN / NOT IN для фильтрации столбца по нескольким значениям, AND / OR для фильтрации таблицы по нескольким столбцам.

**ORDER BY** — необязательный элемент запроса, 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** указываются столбцы, которые нужно заполнить информацией. В **VALUES** указана информация, которую нужно сохранить.

**Интерфейс пользователя**

Чтобы открыть web-приложение, необходимо задать команду **node app** в терминале так, чтобы путь был туда, где находятся все файлы web-приложения, а именно «app.js». Далее необходимо вбить в строке браузера localhost:3000 и появится главная страница приложения. Для перехода между страницами нужно использовать верхнее меню.

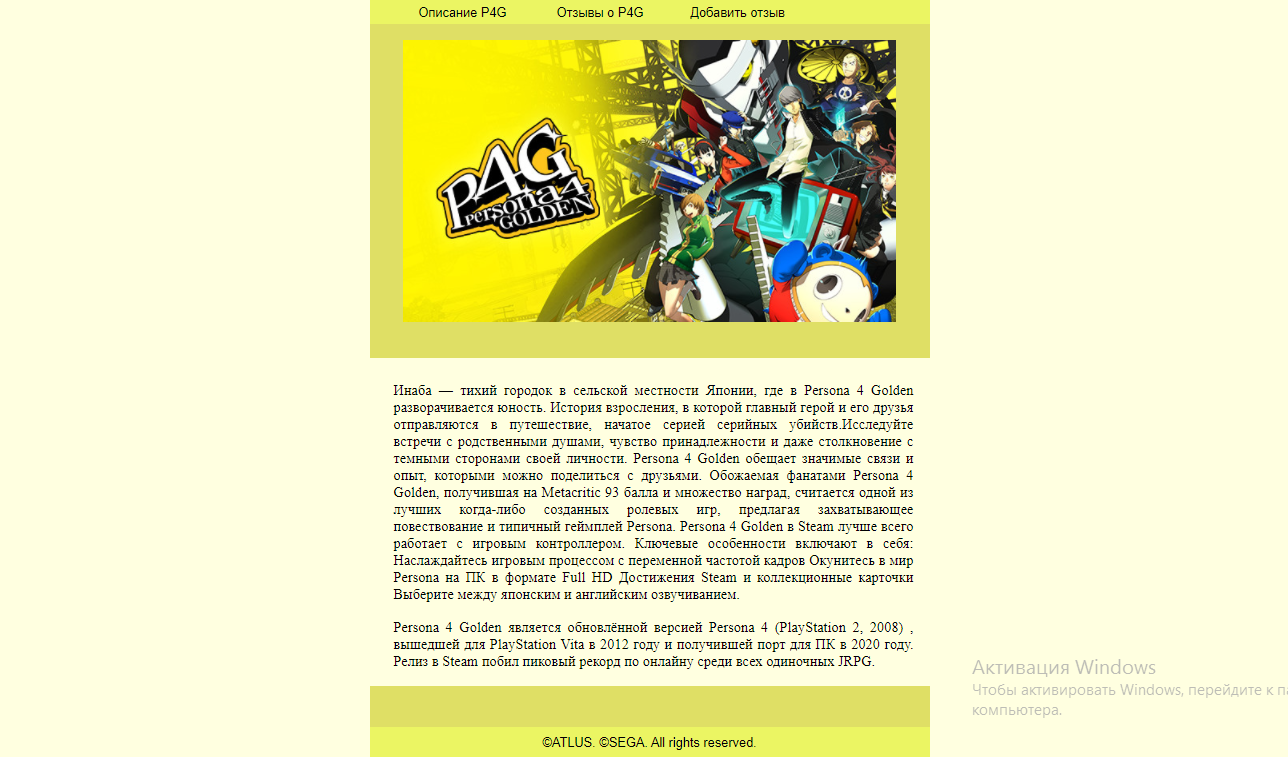
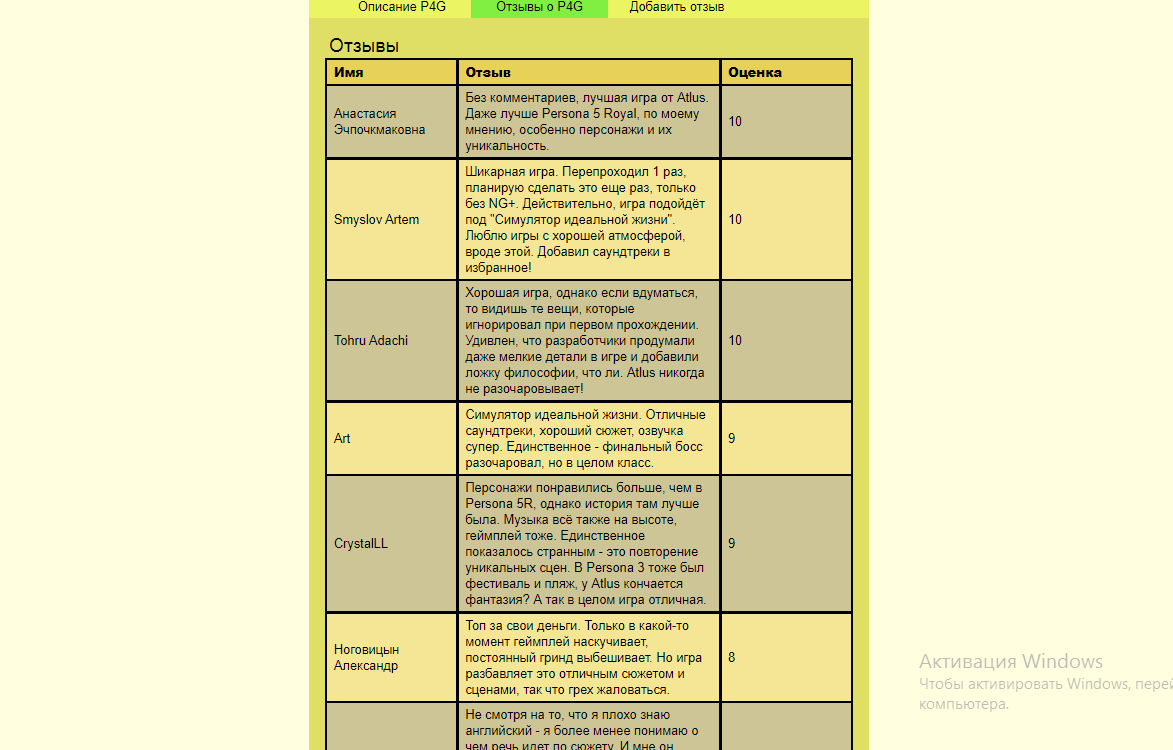
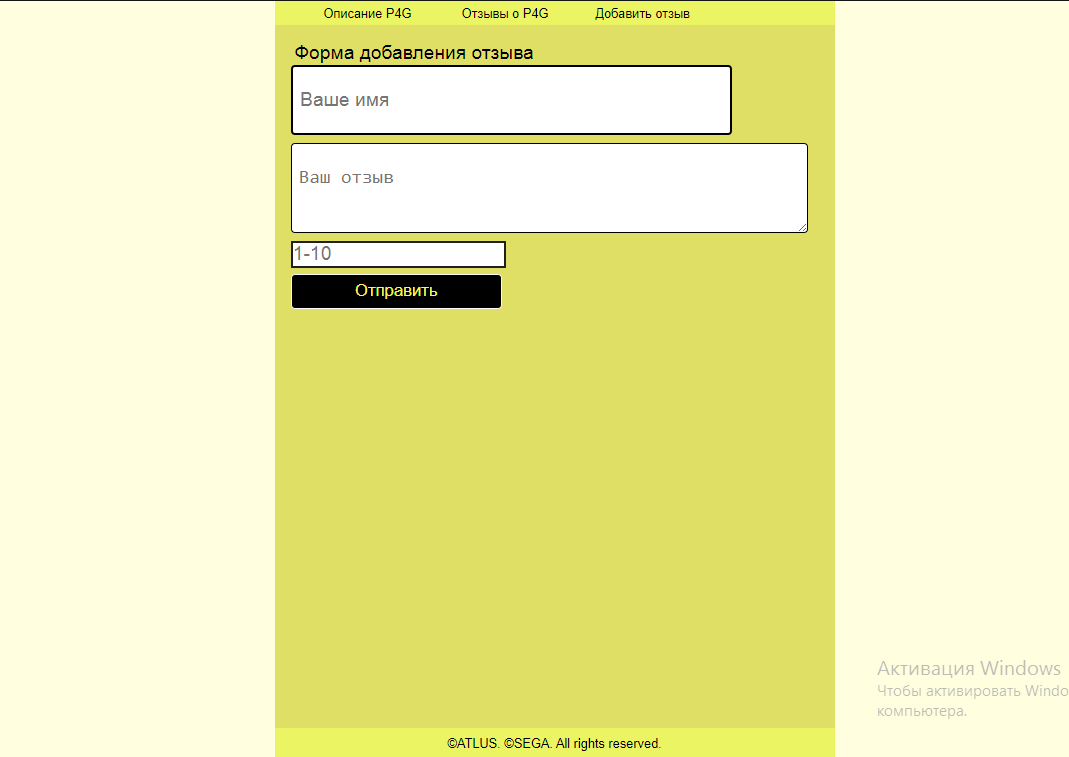
****

Рис. 2 – Начальная страница (описание P4G)

На данной странице пользователя встречает логотип продукции (P4G), а также её описание.

****Рис. 3 – страница отображающая данные отзывов.

Рис. 4 – Страница добавления отзыва.

На данной странице пользователь способен добавить свой отзыв о продукции, достаточно лишь заполнить необходимые поля. Оценка продукта должна быть от 1 до 10, иначе отправить отзыв не получится.

Заключение

Web-приложение были разработано на платформе Node.j, в основном с использованием языка JavaScript. База данных хранится в виде одного файла. Используемая РСУБД может быть любая, но я использовал DB Browser (SQLite). Сделан прототип web-приложения «отзывы о продукции», который в будущем можно улучшать. Web-приложение работает стабильно, отображает все отзывы, присутствует ограничение от 1 до 10 по шкале оценки.

Выполнены задания:

1) добавить поле "Оценка продукта" (добавить отдельный, самый правый столбец таблицы);

- добавить в таблицу БД и в шаблон html-страницы с отзывами и в обработчик события;

- на шаблоне страницы по вводу отзыва добавить поле input типа number для ввода оценки и установить ограничения от 1 до 10 (10 - это высокая оценка).

2) сделать вывод на страницу с отзывами отсортированным с обратном поряде по полю "Оценка продукта"

- вверху таблицы отзывы с высокой оценкой, внизу - с низкой;

3) изменить на свой вкус дизайн меню, фона, кнопок, шрифтов, цветовое оформление и пр.;

4) выбрать для себя тему работы - тот продукт (фирму, фильм, игру и т.д.), о котором будут отзывы

- на стартовой странице поменять логотип и текст описания;

- заменить отзывы, которые были в БД с примерами работы на свои (согласно выбранной теме, ~ штук 10 отзывов).

Данное web-приложение поможет пользователям узнать о качестве продукта просто посмотря на его отзывы, а также поможет компании лучше и легче анализировать мнения своих клиентов о своей продукции.

# 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. Кирилл Мокевкин. Что такое MVC: рассказываем простыми словами. [Электронный ресурс] Режим доступа: <https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami>
10. SkillFactory.Blog – Node.js. [Электронный ресурс] Режим дооступа: <https://blog.skillfactory.ru/glossary/node-js/>

РЕЦЕНЗИЯ

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

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

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

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

2021

1. Общие сведения
2. Наименование системы

Полное наименование разрабатываемой системы – «Официальный сайт о Персона 4 голден».

Краткое наименование – «Персона 4 голден».

1. Наименование заказчика и исполнителя

Организация: ATLUS.

Адрес: ул. Петропавловская, 23.

Телефон: +7 (666) 133-79-69;

Исполнитель: Смыслов Артем Георгиевич.

1. Плановые сроки начала и окончания работ

Дата начала работ: 04.07.2021.

Дата окончания работ: 17.07.2021.

1. Назначение и цели создания системы

К целям создания веб-приложения «Живая лента» можно отнести:

* улучшение оперативного взаимодействия и интеграция модулей;
* автоматизировать опрос сотрудников и студентов университета.

Достижение целей приведёт к следующим положительным результатам:

* руководство компании сможет быстрее и легче получать и анализировать оценочную информации о продукции «Persona 4 Golden».
* пользователи веб-приложения смогут поделиться своим мнением о «Persona 4 Golden» с разработчиками и другими пользователями, а также найдя мнения других пользователей.
* пользователи веб-приложения смогут видеть информационные публикации в отсортированном в порядке убывания оценки продукции.

1. Характеристика объекта автоматизации

Объектом автоматизации является веб-приложение компании ATLUS.

1. Требования к системе

Общие требования к веб-приложению «Персона 4 голден» являются:

* надёжность и работоспособность;
* интуитивно понятный интерфейс;
* лицензионная чистота – применение средств в рамках общего лицензионного соглашения касательно веб-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 веб-приложения «Персона 4 голден» может быть объединение иных подсистем компании ATLUS.

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 веб-приложения «Персона 4 голден» 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 веб-приложения;
* 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 веб-приложения – должен иметь опыт работы с персональным компьютером на уровне опытного пользователя и свободно осуществлять базовые действия с веб-приложением 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 веб-приложение «Персона 4 голден» должно обеспечивать безопасный доступ к данным, предотвращая несанкционированный доступ или модифицирование данных. Модуль аутентификации должен обеспечивать защищённый доступ ко всему программному интерфейсу приложения.

Также при разработке модуля 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

Веб-приложение должно иметь удобный и интуитивно понятный графический пользовательский интерфейс. Диалоговый интерфейс должен соблюдать контекст подсистемы организационной коммуникации компании ATLUS 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.

1. Требования к программному обеспечению

При проектировании веб-приложения «Персона 4 голден» необходимо эффективно использовать используемое на данный момент в проекте веб-приложения применяется паттерн (шаблон построения архитектуры приложения) проектирования MVC(Model-View-Controller), для хранения данных применяется РСУБД DB Browser (SQLite), а в качестве серверного окружения используется программная платформа Visual Studio Code (Node.js).

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 – 512 ГБ;
* 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.

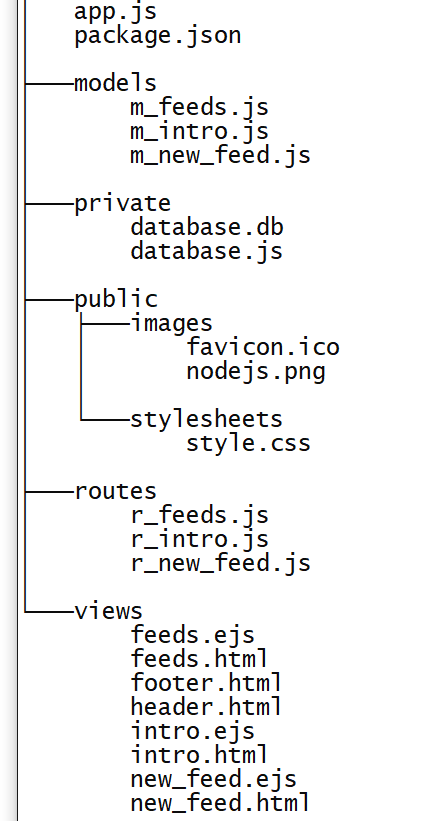
1. Требования к составу и содержанию работ подготовки объекта автоматизации к вводу системы в действие

При подготовке к вводу в эксплуатацию веб-приложения «Персона 4 голден» отдел управления информатизации должен обеспечить выполнение следующих работ:

* определить подразделение и ответственных должностных лиц для внедрения веб-приложения;
* обеспечить пользователей руководством, которое поможет быстрее освоить веб-приложение;
* провести опытную эксплуатацию веб-приложения «Персона 4 голден».

# ПРИЛОЖЕНИЕ Б

**Дерево директорий:**

****

Teddie.png

Nodejs.png

**Листинг кода формы авторизации – 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 формы (библиотеки для построения связи - package.json):**

{

  "dependencies": {

    "better-sqlite3": "^7.5.3",

    "ejs": "^3.1.8",

    "express": "^4.18.1"

  }

}

**Листинг кода формы представления (главной страницы web-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/teddie.png" />

</head>

<body>

    <div class="wrapper">

    <div class="content">

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

        <br>

        <p class="aligncenter">

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

        </p>

        <br><br>

        <div id="intro">

            <p class="intro">

                Инаба — тихий городок в сельской местности Японии, где в Persona 4 Golden разворачивается юность. История взросления, в которой главный герой и его друзья отправляются в путешествие, начатое серией серийных убийств.Исследуйте встречи с родственными душами, чувство принадлежности и даже столкновение с темными сторонами своей личности. Persona 4 Golden обещает значимые связи и опыт, которыми можно поделиться с друзьями.

                Обожаемая фанатами Persona 4 Golden, получившая на Metacritic 93 балла и множество наград, считается одной из лучших когда-либо созданных ролевых игр, предлагая захватывающее повествование и типичный геймплей Persona.

                Persona 4 Golden в Steam лучше всего работает с игровым контроллером.

                Ключевые особенности включают в себя:

                Наслаждайтесь игровым процессом с переменной частотой кадров

                Окунитесь в мир Persona на ПК в формате Full HD

                Достижения Steam и коллекционные карточки

                Выберите между японским и английским озвучиванием.

            </p>

            <p class="intro">

                Persona 4 Golden является обновлённой версией Persona 4 (PlayStation 2, 2008) , вышедшей для PlayStation Vita в 2012 году и получившей порт для ПК в 2020 году. Релиз в Steam побил пиковый рекорд по онлайну среди всех одиночных JRPG.

            </p>

        </div>

    </div>

    <%- include("./footer.html") %>

    </div>

</body>

</html>

**Листинг кода формы представления (страницы web-приложения с отображением 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/teddie.png" />

</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="25%">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.userRate %></td>

                </tr>

                <% } %>

            </table>

        </div>

    </div>

    <%- include("./footer.html") %>

    </div>

</body>

</html>

**Листинг формы представления (страница web-приложения для добавления новых отзывов - 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/teddie.png" />

</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="number" type="number" name="userRate" min="1" max="10" placeholder="1-10">

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

            </form>

        </div>

    </div>

    <%- include("./footer.html") %>

    </div>

</body>

</html>

**Листинг кода формы (работа базы данных – database.js):**  
const pathDB = "./private/database.db";

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

class DataProcessing {

    getFeeds() {

        let query = `

            SELECT userName, userFeed, userRate

            FROM feeds

            ORDER BY userRate DESC`;

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

        return rows;

    }

    insertFeed(name, feed, userRate) {

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

        let query = `

            INSERT INTO feeds (userName,userFeed,userRate)

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

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

    }

}

module.exports = new DataProcessing();

**Листинг кода формы (Описание дизайна web-приложения – style.css):**

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

html, body { height: 100%; }

body {

    background-color: rgb(255, 255, 224);

    font: 16px/20px Timew New Roman, sans-serif;

}

.wrapper {

    display: flex;

    flex-direction: column;

    width: 700px;

    min-height: 100%;

    margin: 0 auto;

    background: #dfdf65;

    font: 16px/20px Timew New Roman, sans-serif;

}

.content {

    flex: 1 0 auto;

}

.desktop {

    margin-left: 20px;

    margin-right: 20px;

}

.aligncenter {

    text-align: center;

}

.intro {

    font: 18px Timew New Roman, serif;

    margin-top: 20px;

    margin-bottom: 10px;

    text-align: justify;

}

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

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

#navbar {

    background-color: rgb(235, 245, 99);

    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: rgb(0, 0, 0);

    text-decoration: none;

    text-align: center;

}

#navbar ul li { float: none; }

#navbar li:hover {

    background-color: #81ee42;

}

.footer {

    flex: 0 0 auto;

    height: 40px;

    background: rgb(235, 245, 99);

    display: flex;

}

.footer-text {

    margin: auto;

    color: #000000;

}

h4 {

    color: #000000;

    font-size: 24px;

    font-weight: lighter;

    margin: 5px;

}

a {

    text-decoration: none;

    font-weight: lighter;

    color: #e9d949;

    font-size: 16px;

}

a:hover {

    color: rgb(87, 72, 32);

}

table {

    width: 100%;

    border-collapse: collapse;

}

td, th {

    border: 3px solid rgb(0, 0, 0);

    padding: 5px 8px 4px 8px;

}

th {

    text-align: justify;

    font-size: 16px;

    font-weight: 800;

    background-color: #e7d159;

    color: rgb(0, 0, 0);

}

tr:nth-child(odd) {

    background-color: #f4e694;

}

tr:nth-child(even) {

    background-color: #cec596;

}

#intro {

    margin:0 auto;

    background: rgb(255, 255, 224);

    padding:10px 20px 10px 30px;

    min-width: 600px;

}

#inline {

    width: 70%;

    margin:0 auto;

    background:rgb(243, 233, 233);

    padding:10px 20px 10px 30px;

    min-width: 600px;

    border-radius: 5px;

}

.txt {

    display:inline-block;

    color:#676767;

    width:80%;

    margin-bottom:10px;

    border:3px solid rgb(0, 0, 0);

    padding:5px 9px;

    font-size:24px;

    line-height:3em;

    border-radius:5px;

}

.txtarea {

    display:inline-block;

    color:#676767;

    resize: vertical;

    width:95%;

    height:100px;

    margin-bottom: 5px;

    border:1px solid rgb(0, 0, 0);

    padding:5px 9px;

    font-size:24px;

    line-height:3em;

    border-radius:5px;

}

.txt:focus, .txtarea:focus {

    border-style:solid;

    border-color:#BABABA;

    color:#444;

}

button.send {

    color:#faf458;

    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:#000000;

}

button.send:hover {

    background:#81ee42;

}

.number{

    border:solid 3px #212020;

    font-size:24px;

    width:40%;

}