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-2019-3

направления подготовки

09.03.03 Прикладная информатика

Торсунов Илья Павлович

Проверил:

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич

2022

Содержание

[Постановка задачи на проектирование 3](#_Toc105513195)

[Анализ технологий проектирования 5](#_Toc105513196)

[Реализация функционала информационной системы 12](#_Toc105513197)

[Заключение 14](#_Toc105513198)

[Список источников 15](#_Toc105513199)

[ПРИЛОЖЕНИЕ А 16](#_Toc105513200)

[ПРИЛОЖЕНИЕ Б 19](#_Toc105513201)

[ПРИЛОЖЕНИЕ B 25](#_Toc105513202)

# Постановка задачи на проектирование

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

В рамках практики будет разработано web-приложение "**Отзывы о продукте**".

**Используемые технологии:**

|  |  |
| --- | --- |
| * Nodejs * Express * ejs * MVC * csv * json | * re * SQLite * html * css * lodash |

**Архитектура приложения** основана на паттерне проектирования **MVC** (Model-View-Controller).

Составные части паттерна:

**model** - модель - это часть архитектуры приложения:

- включает в себя бизнес-логику приложения (классы, методы, функции обработки данных);

- модель "не знает" о контроллерах и представлениях, не ориентируется на них;

- данные модели могут быть представлены таблицами базы данных или файлами XML, JSON, csv;

- это может быть просто набор объектов или функций для реализации логика приложения.

**view** - представление - это часть архитектуры приложения (примеры представления: HTML-страница, WPF форма, Windows Form):

- отвечает за визуальное отображение данных, полученных от модели

- представление может читать данные, но не может записывать или изменять их

- представление может иметь программный код, который реализует логику отображения данных

**controller** - контроллер - это часть архитектуры приложения:

- совокупность обработчиков событий представления

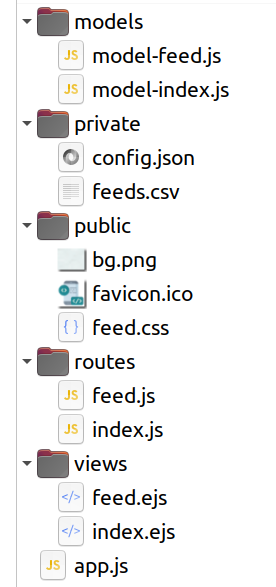
- принимает события от представления

- отдаёт событие на обработку в модель

- возвращает пользователю обновлённое представление

- или выбирает какое именно представление должно быть отображено в ответ.

**Дерево директорий** приложения:



# Анализ технологий проектирования

Обзор технологий для хранения данных

CSV-файлы

Формат CSV используют, чтобы хранить таблицы в текстовых файлах. Данные очень часто упаковывают именно в таблицы, поэтому CSV-файлы очень популярны.

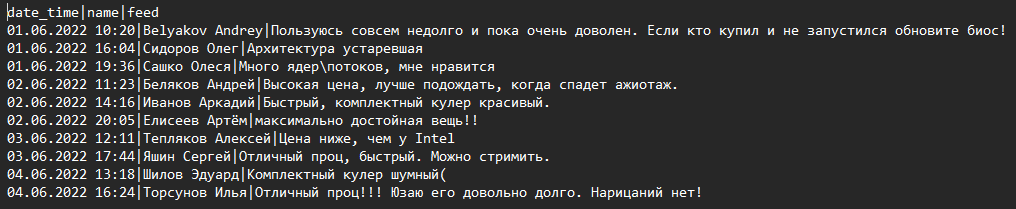


Рисунок 1 – пример csv-файла с отзывами

CSV расшифровывается как comma-separated values — «значения, разделенные запятыми». Но пусть название вас не обманет: разделителями столбцов в CSV-файле могут служить и точки с запятой, и знаки табуляции. Это все равно будет CSV-файл. CSV-файл состоит из строк с данными и разделителей, которые обозначают границы столбцов

У CSV куча плюсов перед тем же форматом Excel: текстовые файлы просты как пуговица, открываются быстро, читаются на любом устройстве и в любой среде без дополнительных инструментов.

Из-за своих преимуществ CSV — сверхпопулярный формат обмена данными, хотя ему уже лет 40. CSV используют прикладные промышленные программы, в него выгружают данные из баз.

Одна беда — текстового редактора для работы с CSV мало. Еще ничего, если таблица простая: в первом поле ID одной длины, во втором дата одного формата, а в третьем какой-нибудь адрес. Но когда поля разной длины и их больше трех, начинаются мучения.



Рисунок 2 – пример csv-файла

Еще хуже с анализом данных — попробуй «Блокнотом» хотя бы сложить все числа в столбце. Я уж не говорю о красивых графиках.

Поэтому CSV-файлы анализируют и редактируют в Excel и аналогах: Open Office, LibreOffice и прочих.

JSON-файлы

JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. Но при этом формат независим от JS и может использоваться в любом языке программирования. Формат JSON был разработан Дугласом Крокфордом.

За счёт своей лаконичности по сравнению с XML формат JSON может быть более подходящим для сериализации сложных структур. Применяется в веб-приложениях как для обмена данными между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения).

Поскольку формат JSON является подмножеством синтаксиса языка JavaScript, то он может быть быстро десериализован встроенной функцией eval().

Структуры данных, используемые JSON, поддерживаются любым современным языком программирования, что и позволяет применять JSON для обмена данными между различными языками программирования и программными системами.

В качестве значений в JSON могут быть использованы:

* запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.
* массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, то есть не содержать ни одного значения. Значения в пределах одного массива могут иметь разный тип.
* число (целое или вещественное).
* литералы true (логическое значение «истина»), false (логическое значение «ложь») и null.
* строка — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты \", \\, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.

Строка очень похожа на литерал одноимённого типа данных в языке Javascript. Число тоже очень похоже на Javascript-число, за исключением того, что используется только десятичный формат (с точкой в качестве разделителя). Пробелы могут быть вставлены между любыми двумя синтаксическими элементами.

Обзор языков программирования

NodeJS

Node или Node.js — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Node.js разработал Райан Даль в 2009 году после двух лет экспериментирования над созданием серверных веб-компонентов. В ходе своих исследований он пришёл к выводу, что вместо традиционной модели параллелизма на основе потоков следует обратиться к событийно-ориентированным системам. Эта модель была выбрана из-за простоты, низких накладных расходов (по сравнению с идеологией «один поток на каждое соединение») и быстродействия. Целью Node является предложить «простой способ построения масштабируемых сетевых серверов».

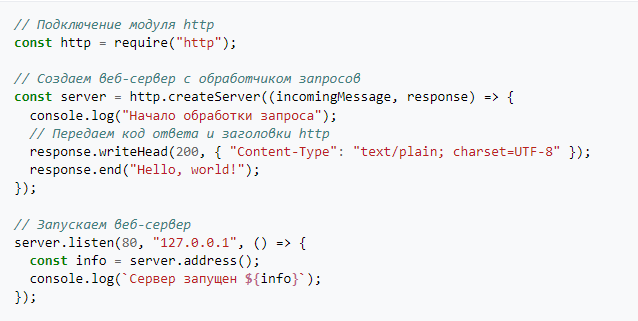


Рисунок 3 - Создание и запуск HTTP-сервера на Node.js

Python

Python — это высокоуровневый интерпретируемый язык программирования общего назначения. Его философия дизайна подчеркивает читабельность кода с использованием значительных отступов.

Python динамически типизируется и собирает мусор. Он поддерживает несколько парадигм программирования, включая структурированное (особенно процедурное), объектно-ориентированное и функциональное программирование. Его часто описывают как "включенный в батарейки" язык из-за его обширной стандартной библиотеки.

Гвидо ван Россум начал работать над Python в конце 1980-х годов как преемник языка программирования ABC и впервые выпустил его в 1991 году как Python 0.9.0. Python 2.0 был выпущен в 2000 году и представил новые функции, такие как понимание списков, сбор мусора с обнаружением циклов, подсчет ссылок и поддержка Unicode. Python 3.0, выпущенный в 2008 году, был основной редакцией, которая не полностью обратно совместима с более ранними версиями. Python 2 был прекращен с версией 2.7.18 в 2020 году.

Python неизменно считается одним из самых популярных языков программирования.

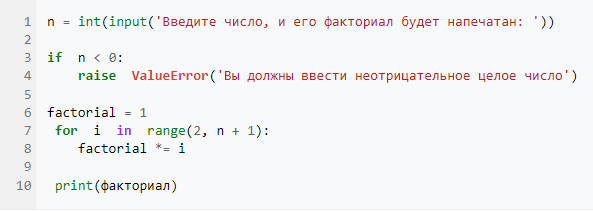


Рисунок 4 - Программа для вычисления факториала положительного целого числа

Описание паттерна MVC

MVC (Model-View-Controller) — это фундаментальный паттерн, который нашел применение во многих технологиях, дал развитие новым технологиям и каждый день облегчает жизнь разработчикам.

Впервые паттерн MVC появился в языке SmallTalk. Разработчики должны были придумать архитектурное решение, которое позволяло бы отделить графический интерфейс от бизнес логики, а бизнес логику от данных. Таким образом, в классическом варианте, MVC состоит из трех частей, которые и дали ему название. Рассмотрим их:

1. Модель

Под Моделью, обычно понимается часть содержащая в себе функциональную бизнес-логику приложения. Модель должна быть полностью независима от остальных частей продукта. Модельный слой ничего не должен знать об элементах дизайна, и каким образом он будет отображаться. Достигается результат, позволяющий менять представление данных, то как они отображаются, не трогая саму Модель.

Модель обладает следующими признаками:

* Модель — это бизнес-логика приложения;
* Модель обладает знаниями о себе самой и не знает о контроллерах и представлениях;
* Для некоторых проектов модель — это просто слой данных (DAO, база данных, XML-файл);
* Для других проектов модель — это менеджер базы данных, набор объектов или просто логика приложения;

1. Представление (View)

В обязанности Представления входит отображение данных полученных от Модели. Однако, представление не может напрямую влиять на модель. Можно говорить, что представление обладает доступом «только на чтение» к данным.

Представление обладает следующими признаками:

* В представлении реализуется отображение данных, которые получаются от модели любым способом;
* В некоторых случаях, представление может иметь код, который реализует некоторую бизнес-логику.

Примеры представления: HTML-страница, WPF форма, Windows Form.

1. Контроллер

Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

Контроллер обладает следующими признаками:

* Совокупность обработчиков событий представления;
* Принимает события от представления;
* Отдаёт событие на обработку в модель;
* Возвращает пользователю обновлённое представление;
* Выбирает какое именно представление должно быть отображено в ответ.

# Реализация функционала информационной системы

Описание методов для обработки данных

Метод fs.appendFileSync()

Метод fs.appendFileSync() используется для синхронного добавления заданных данных в файл. Новый файл создается, если он не существует. Необязательный параметр options может использоваться для изменения поведения операции.

Синтаксис:

fs.appendFileSync (path, data, options)

Пример:

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

Метод csvjson

Метод csvjson — это комплексный парсер nodejs для преобразования csv в json или массивы столбцов.

Синтаксис:

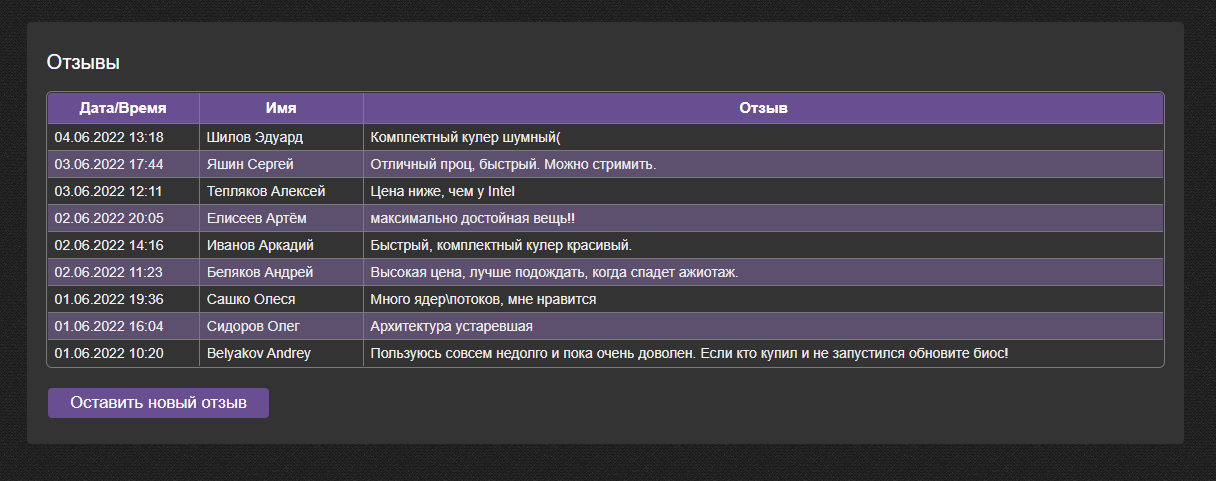
csvjson.toObject(data, options)

Пример:

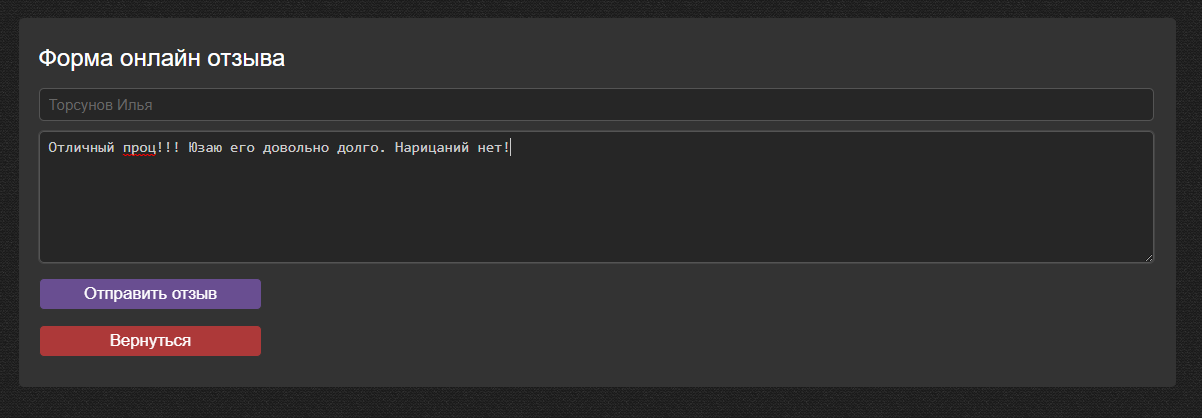
csvjson.toObject(textCSV, { delimiter: del })

Интерфейс пользователя

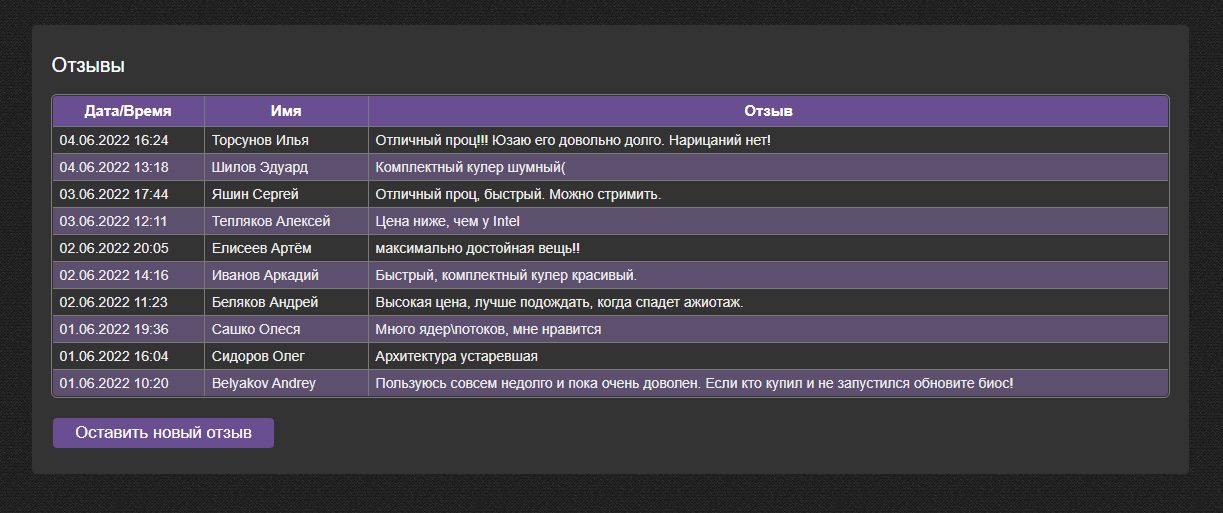
Первоначальная страница с отзывами:



Чтобы оставить свой отзыв нажимаем «Отправить новый отзыв» и нас перекидывает в форму написания отзыва:



Здесь пишем свой отзыв и нажимаем кнопку «Отправить отзыв», нас перекидывает, и он появляется в списке всех отзывов:



# Заключение

В рамках данного проекта было разработано web-приложение «Отзывы о продукте», а также были изучены: программная платформа NodeJS, технологии csv и json, а также паттерн MVC.

Благодаря паттерну MVC мы разбили приложение на 3 модуля: models, views и private. Что в итоге даёт нам в будущем легко изменять определённую часть приложения, не затрагивая другие элементы.

Разработанное web-приложение даёт владельцу интернет-магазина отслеживать какой товар как продаётся и получать обратную связь по продуктам.

# 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. <https://www.geeksforgeeks.org/node-js-fs-appendfilesync-function/>
10. <https://ru.wikipedia.org/wiki/Node.js>

# ПРИЛОЖЕНИЕ А

**Техническое задание**

1. Общие сведения
2. Наименование системы

Полное наименование разрабатываемой системы – web-приложение "Отзывы о продукте".

Краткое наименование – web-приложение "Отзывы о продукте".

1. Наименование заказчика и исполнителя

Организация: ФГБОУ ВО Пермский ГАТУ.

Адрес: ул. Петропавловская, 23.

Телефон: +7 (342) 217-90-66;

Исполнитель: Торсунов Илья Павлович.

1. Плановые сроки начала и окончания работ

Дата начала работ: 25.05.2022.

Дата окончания работ: 07.06.2022.

1. Назначение и цели создания системы

К целям создания web-приложения «Отзывы о продукте» можно отнести:

* улучшение взаимодействия покупателя и магазина;
* автоматизировать создание отзывов клиентов.

Достижение целей приведёт к следующим положительным результатам:

* покупатель сможет выбрать хороший товар;
* товар будет лучше продаваться, если на нём будет много хороших отзывов;
* магазин будет получать хорошую прибыль с продаж;

1. Характеристика объекта автоматизации

Объектом автоматизации является книга отзывов и предложений.

1. Требования к системе

Общие требования к web-приложению «Отзывы о продукте» являются:

* надёжность и работоспособность;
* интуитивно понятный интерфейс;
* простота и эффективность;
* 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 web-приложение «Отзывы о продукте» может объединить эти данные представив их как единый информационный поток.

1. Перспективы развития, модернизация системы

Дальнейшим развитием web-приложения «Отзывы о продукте» может быть объединение иных подсистем интернет-магазина.

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 «Отзывы о продукте» должно обеспечивать безопасный доступ к данным, предотвращая несанкционированный доступ или модифицирование данных.

1. Требования по эргономике и технической эстетике

Web-приложение должно иметь удобный и интуитивно понятный графический пользовательский интерфейс.

Пользовательский интерфейс модуля также должен аккомпанировать цветовой гамме и общему стилю интернет-магазина.

1. Требования к программному обеспечению

При проектировании web-приложения «Отзывы о продукте» необходимо эффективно использовать различные фреймворки и платформы. На данный момент в проекте в качестве серверного окружения используется программная платформа Node.js, а для хранения данных применяется CSV-файл.

1. Требования к техническому обеспечению

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

* процессор – Intel Xeon 3.7 ГГц;
* оперативная память – 4 ГБ;
* дисковая система – 1 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 web-приложения «Отзывы о продукте» 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 web-приложение;
* провести опытную эксплуатацию web-приложения «Отзывы о продукте».

# ПРИЛОЖЕНИЕ Б

**Листинг кода приложения – app.js**

// 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`);

});

Модули

models/model-feed.js

const md\_feed = {

title: "Оставить отзыв",

heading: "Форма онлайн отзыва",

button: "Отправить отзыв"

};

module.exports = { md\_feed }

models/model-index.js

const md\_index = {

title: "Отзывы о продукте",

heading: "Отзывы",

button: "Оставить новый отзыв",

feeds: []

}

module.exports = { md\_index }

config.json

{

"debug": {

"port": 3001,

"hostname": "localhost"

},

}

routes/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) => {

let nodeDate = require('date-and-time');

let date\_time = nodeDate.format(new Date(), 'DD.MM.YYYY HH:mm');

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

return `${date\_time}|${nameUser}|${feed}\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); // находим по имени в шаблоне

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

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

});

module.exports = router;

routes/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');

md\_index.feeds = md\_index.feeds

.sort((a,b) => a.date\_time>b.date\_time? -1: +1);

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

});

module.exports = router;

views/feed.ejs

<!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">

<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>

views/index.ejs

<!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">

<h3><%=heading%></h3>

<table>

<tr>

<th>Дата/Время</th>

<th>Имя</th>

<th>Отзыв</th>

</tr>

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

<tr>

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

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

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

</tr>

<%} %>

</table>

<br>

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

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

</form>

</div>

</body>

</html>

feed.css

body {

background: url(/public/bg.png) repeat;

background-color: #1a1a1a;

font: normal normal 14px/20px Arial, Georgia, serif;

}

h2 {

color: #fff;

font-size: 24px;

font-weight: lighter;

}

h3 {

color: #fff;

font-size: 20px;

font-weight: lighter;

}

h4 {

color: #fff;

font-size: 16px;

font-weight: lighter;

}

.feed {

color: #eee;

font-size: 14px;

font-weight: normal;

}

table {

width: 100%;

border-collapse: collapse;

border-radius: 5px;

overflow: hidden;

box-shadow: 0 0 0 1px #7a7a7a;

border: 1px solid #333;

}

td, th {

border: 1px solid #7a7a7a;

padding: 3px 7px 3px 7px;

color: #eee;

}

th {

text-align: center;

padding: 5px;

background-color: #694e91;

color: #fff;

font-size: 15px;

}

tr:nth-child(odd) {

background-color: #5d4f6e;

}

/\* Форма обратной связи \*/

#inline {

margin-left:30px;

width:60%;

margin:0 auto;

background:#333;

padding:10px 20px;

border-radius:5px;

}

.txt {

display:inline-block;

background: #262626;

color:#676767;

width:98%;

margin-bottom:10px;

border:1px solid #545454;

padding:5px 9px;

font-size:15px!important;

line-height:1.4em;

border-radius:5px;

}

.txtarea {

display:inline-block;

background: #262626;

color:#676767;

resize: vertical;

width:98%;

height:120px;

margin-bottom:10px;

border:1px solid #545454;

padding:5px 9px;

font-size:15px!important;

line-height:1.4em;

border-radius:5px;

}

.txt:focus, .txt:hover, .txtarea:focus, .txtarea:hover {

outline: 1px solid #444;

color: #ccc;

}

input.error, textarea.error {

border-color:#973D3D;

border-style:solid;

background:#F0BEBE;

color:#A35959;

}

input.error:focus, textarea.error:focus {

border-color:#973D3D;

color:#A35959;

}

#send {

color:#fff;

display:block;

cursor:pointer;

padding:5px 11px;

font-size:1.2em;

border:1px solid #333;

border-radius:5px;

background:#694e91;

width:20%;

margin-bottom:15px;

}

#send:hover {

background:#8d65c7;

}

#back {

color:#fff;

display:block;

cursor:pointer;

padding:5px 11px;

font-size:1.2em;

border:1px solid #333;

border-radius:5px;

background:#ad3939;

width:20%;

margin-bottom:20px;

}

#back:hover {

background:#cb5151;

}

/\* Форма обратной связи \*/

## **ПРИЛОЖЕНИЕ B**

**РЕЦЕНЗИЯ**

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

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

доцент кафедры ИТиПИ, к.т.н., доцент

Беляков Андрей Юрьевич