Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Пермский государственный аграрно-технологический университет
имени академика Д. Н. Прянишникова»

Беляков А.Ю.

ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА

Методические рекомендации для прохождения практики

Пермь
ФГБОУ ВО Пермский ГАТУ
2023

УДК 004.43
ББК 32.973-018.1
Б 448

Рецензенты:

Е.А. Муратова, к.э.н., доцент, заведующий кафедрой информационных технологий и программной инженерии (ФГБОУ ВО Пермский ГАТУ)

А.А. Зорин, к.т.н., доцент кафедры информационных технологий и программной инженерии (ФГБОУ ВО Пермский ГАТУ)

Б 448 Беляков А.Ю.

Проектно-технологическая практика: методические рекомендации для прохождения практики / авт. А.Ю. Беляков; М-во науки и высшего образования РФ, федеральное гос. бюджетное образов. учреждение высшего образования «Пермский гос. аграрно-технолог. университет им. акад. Д.Н. Прянишникова». – Пермь: ФГБОУ ВО Пермский ГАТУ, 2023. –31 с.

В методических рекомендациях для прохождения проектно-технологической практики представлена постановка задачи на проектирование информационной системы, сформулированы требования к технической реализации проекта и к оформлению отчёта о практике.

Методические рекомендации предназначены для обучающихся очной и заочной форм обучения по направлениям подготовки 09.03.03 Прикладная информатика и 09.03.04 Программная инженерия.

УДК 004.43
ББК 32.937-018.1

Рекомендованы к изданию методической комиссией факультета экономики и информационных технологий ФГБОУ ВО Пермский ГАТУ, протокол №4 от 6 декабря 2022 г.

Содержание

Введение

Методические рекомендации предназначены для помощи в выполнении проектно-технологической практики по направлениям подготовки 09.03.03 Прикладная информатика, направленность (профиль) «Прикладная информатика в экономике» и 09.03.04 Программная инженерия, направленность (профиль) «Разработка программно-информационных систем». Существенную роль в освоении профессиональных компетенций играет отработка умений самостоятельного выполнения технических проектов автоматизации бизнес-процессов, что вызывает потребность в разработке методических рекомендаций.

Целью данных методических рекомендаций по проектно-технологической практике является оказание помощи обучающимся в подготовке технического задания на проектирование информационной системы, разработке и тестировании функционала информационной системы.

В методических рекомендациях по проектно-технологической практике представлены постановка задачи на проектирование веб-сервиса, особенности разработки информационной системы, включая описание используемых технологий, архитектуру приложения, интерфейс пользователя. Методические рекомендации содержат описание результатов проектно-технологической практики, перечень рекомендуемой основной и дополнительной литературы, базы данных и информационно-справочные и поисковые системы.

В рамках выполнения задач проектно-технологической практики будут закреплены следующие комплексные умения и практические навыки по выполнению программной реализации:

– проектировать и разрабатывать архитектуру web-сервиса;

– получать и расшифровывать запрос от клиента;
– обрабатывать запросы и отправлять ответы клиенту;
– осуществлять выборку, добавление, изменение или удаление данных;
– обрабатывать структурированные или иерархические файлы с данными – csv, json, xml;
– запускать методы по обработке данных синхронно или асинхронно.

1. Постановка задачи на практику

Проектно-технологическая практика посвящена рассмотрению методики автоматизации бизнес-процессов предприятия и выполнению этапов, связанных с разработкой программной реализации.

Последовательность автоматизации бизнес-процессов предприятия можно разбить на следующие самостоятельные этапы:

– исследование предприятия;
– выбор неавтоматизированного или слабо автоматизированного процесс;
– составление модели процесса «как-есть» и «как-будет»;
– подготовка Технического Задания на разработку информационной системы;
– разработка прототипа и проведение апробации;
– разработка программной реализации;
– осуществление тестирования;
– внедрение технического решения;
– сопровождение и доработки технического решения.

Из приведённой последовательности этапов в выполнении практики будут задействованы:

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

Таким образом, в рамках проектно-технологической практики предстоит разработать и апробировать техническое решение в виде web-сервиса для автоматизации бизнес-процесса предприятия.

В техническое решение будут интегрированы такие вопросы проектирования и программирования как: архитектура web-сервиса, объектно-ориентированное программирование, организация обмена данными между клиентом и сервером, авторизация пользователя на сервере, передача параметров запроса через http-протокол, способы работы со структурированными и иерархическими данными.

2. Анализ бизнес-процесса

В рамках проектно-технологической практики рассмотрим бизнес-процесс сбора информации по определённому вопросу для компании. В качестве собираемой информации могут быть курсы валют, динамика изменения криптовалют, статистика по продажам определённой продукции, данные по ценообразованию определённого вида продукции, статистика по погодным условиям для выбранных населённых пунктов, статистика матчей в чемпионате по определённому виду спорта, различного рода тематические рейтинги, публикуемые на страницах сайтов.

Периодический сбор и сохранение такого вида информации – это довольно рутинная операция. В случае исполнения этой задачи сотрудником вручную можно получить ряд ошибок, неточностей и опечаток при сборе и структурировании

данных. Кроме того, рационально вести сбор данных однообразно, в одном и том же формате вне зависимости от предпочтений исполнителя, назначенного сотрудника.

Таким образом, задача автоматизации сбора, обработки и хранения данных является актуальной. Разрабатываемая информационная система должна не только автоматически собирать данные и сохранять их в структурированном формате, но и обеспечивать интерфейс взаимодействия пользователя с собранными данными. Пользователь без авторизации может просматривать данные, фильтровать, сортировать и делать выборку по переделённым полям. Пользователь с авторизацией получает дополнительные возможности по добавлению, изменению и удалению хранящихся на сервере данных.

Информационная система должна быть реализована в виде веб-сервиса (веб-службы), который дежурит в режиме ожидания запроса от пользователя. В случае получения http-запроса веб-сервис просыпается, распознаёт детали запроса от пользователя, опознаёт пользователя (при необходимости), и приступает к обработке данных.

В качестве запроса на получение новой информации может быть принята команда на запуск парсера, сбор определённых данных с ранее назначенных сайтов или сервисов (например, данные о текущей погоды в указанном списке городов) и сохранение в структурированном виде полученных данных на сервере.

В качестве запроса на получение клиентом данных может быть принята команда на выборку, фильтрацию и сортировку уже хранимых на сервере данных.

В качестве запроса на обработку данных может быть принята команда на добавление, изменение или удаление хранимых на сервере данных.

На практике можно выделить четыре различных способа организации программного интерфейса приложения API (Application Programming Interface) в зависимости от поддерживаемого способа обмена данными:

1. SOAP API (Simple Object Access Protocol, то есть простой протокол доступа к объектам) – информационная система доступа к объектам, в которой клиент и сервер обмениваются сообщениями посредством XML-файлов.

2. RPC API (Remote Procedure Call, то есть удалённый вызов процедур) – организация системы удаленного вызова процедур, при котором клиент запускает функцию на стороне сервера, а сервер отправляет результат обратно клиенту.

3. Websocket API (пер. сетевая розетка) – способ организации постоянного соединения с двусторонним взаимодействием клиента и сервера в выделенном процессе, при котором не только клиент может отправлять запросы к серверу, но и сервер может отправлять сообщения обратного вызова клиенту.

4. REST API (Representational State Transfer, т.е. передача репрезентативного состояния) – способ организации работы с данными, при котором клиент отправляет http-запросы на сервер в виде определённым образом организованной строки, а сервер разбирает запрос пользователя и на основе него запускает функцию обработки данных и возвращает сформированный ответ (репрезентативное состояние данных) обратно клиенту в одном из доступных форматов, как правило, json-строку.

3. Проектирование информационной системы

В рамках проектно-технологической практики в качестве программной реализации (прототипа) требуется разработать

web-сервис "*Работа с данными*". Разработку информационной системы ***можно вести на любом, подходящем для такого рода задач языке программирования*** (C#, Java, Python, php, JavaScript). В рамках проектно-технологической практики следует обосновать сделанный выбор платформы для разработки, языка программирования и структур хранения данных.

В рамках методических рекомендаций будут приводиться примеры реализации с использованием языка программирования JavaScript, платформы Node.js и сопутствующего стека технологий:

– JavaScript – мультипарадигменный язык программирования;

– Nodejs – платформа для разработки серверных приложений на языке JavaScript;

– Axios – это асинхронная JavaScript-библиотека, предназначенная для создания http-клиента;

– Cheerio – это JavaScript-библиотека, аналог jQuery для работы на платформе Node.js, предназначенная для работы с элементами DOM html-страницы;

– csv – способ представления табличных данных;

– json – способ представления иерархически организованных данных;

– xml – способ представления иерархически организованных данных;

– re (regular expressions) – формальный язык обработки текстовой информации, для поиска и проверки комбинаций символов в тексте, основанный на использовании метасимволов;

– lodash – функциональная библиотека;

– ramda – библиотека для обработки данных с отсутствием побочных эффектов, гарантией неизменяемости данных пользователя и каррированием функций.

Рассмотрим ***архитектуру веб-сервиса.***

В данном проекте требуется разработать именно серверное приложение, работающее по протоколу HTTP и основанное на архитектурном стиле REST, представляющем собой согласованный набор требований и ограничений, определяющий элементы информационной системы.

При правильном проектировании информационная система, построенная на основе REST-архитектуры, обладает следующими свойствами:

– производительность;

– масштабируемость.

Ограничения REST-архитектуры предлагают разработчикам использовать HTTP-методы явно в соответствии с определением протокола. Этот основной принцип проектирования REST устанавливает однозначное соответствие между операциями create, read, update и delete (CRUD) и HTTP-методами:

– для создания ресурса на сервере используется POST;

– для извлечения ресурса используется GET;

– для изменения состояния ресурса или его обновления используется PUT;

– для удаления ресурса используется DELETE.

Перечислим некоторые требования к архитектуре REST-сервиса:

1. Архитектура должна базироваться на клиент-серверной модели приложения (рис. 1).

2. Не следует тратить ресурсы на хранение состояния работы с клиентом. Все запросы от клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для формирования ответа.

3. Кэширование. Ответы сервера должны оформляться как некэшируемые с целью предотвращения получения клиентами устаревших или неверных данных в ответ на последующие запросы.

4. Стандартизация интерфейса доступа к функциям сервиса.

5. Слои (в рамках выполнения практики этот пункт не требуется). Применение промежуточных серверов способно повысить масштабируемость за счет балансировки нагрузки и распределенного кэширования. Промежуточные узлы также могут подчиняться политике безопасности с целью обеспечению конфиденциальности информации.

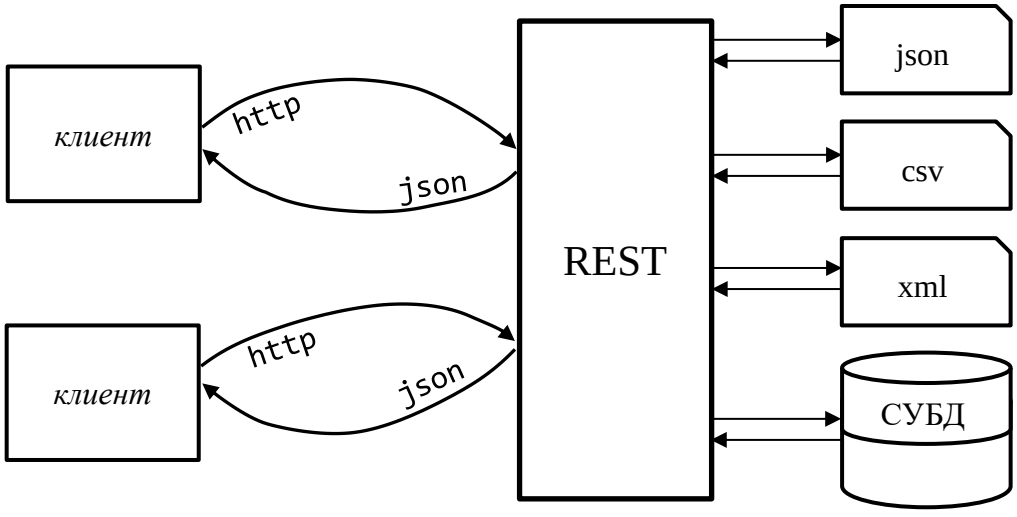

Рисунок 1. Архитектура веб-сервиса.

Информационная система, разработанная в рамках описанных выше ограничений, приобретает такие желательные

свойства как производительность, масштабируемость, простота разработки и последующей модификации, переносимость и надежность.

В задачи данной практики ***не входит*** реализация работы с какой-либо СУБД. Следует организовать хранение структурированной информации в файлах *.csv, а иерархической в *.json или *.xml.

Краткая формулировка требуемого функционала веб-сервиса.

Итак, при проектировании требуется разработать web-сервис, который принимает http-запрос и возвращает массив объектов в формате json-строки.

Программный интерфейс сервиса должен выполнять, как минимум, следующие варианты запросов:

– выдавать API-документацию сервиса;

– выдавать список доступных к просмотру клиентом файлов;

– возвращать массив объектов из указанного клиентом файла;

– возвращать выборку из массива объектов, включая все существующие поля объектов, отсортированную по указанным в запросе направлениям;

– возвращать выборку из массива объектов, включая только выбранные в запросе поля объектов, отсортированную по указанным в запросе направлениям.

Опишем подробнее каждый из требуемых видов запросов.

1. Если сервис получает корневой запрос `'/'`, то возвращает документацию API этого сервиса. Документация подаётся в виде json-строки, включая формат запросов к сервису и

список доступных типов файлов (на выбор – csv, json, xml). Так как формат json подразумевает хранение объекта, то следует определиться с необходимыми полями – author, version, types_of_files, request_formats и установить их значениями. Этот объект (листинг 1) следует заполнить самостоятельно заранее в соответствии с функционалом сервиса и разместить в директории с приложением.

Листинг 1.

Пример json-строки с документацией.

```
{
    "author": "Ilon Mask",
    "version": "1.0.3",
    "types_of_files": "csv,json",
    "request_formats": {
        "/": "API",
        "/type": "найти все файлы по указанному расширению, например, /json",
        "/type/filename.type": "получить список объектов из указанного файла,
         например, /json/users.json",
        "/type/filename.type/?field=direct": "получить список объектов из
         указанного файла с сортировкой по указанным полям, например,
         /json/abiturs.csv/?rating=desc&name=asc",
        "/type/filename.type/?field1&field2&field3/?field2=direct2&field3=dir
         ect3": "получить список объектов из указанного файла, оставив в них
         только указанные поля, с сортировкой по указанным полям, например,
         /json/abiturs.csv/?id&group&rating&name/?rating=desc&name=asc"
    }
}
```

2. Если сервис получает запрос первого уровня, например, `'/json'`, то возвращает список файлов соответствующего типа из некоторой публичной директории `public/` на веб-сервисе и всех иерархически вложенных в неё директорий (рис.2). Для простоты рассмотрения сделайте так, чтобы имена файлов в разных папках не повторялись. Требуется самостоятельно реализовать рекурсивный поиск всех файлов такого типа в иерархически вложенных директориях сервиса.

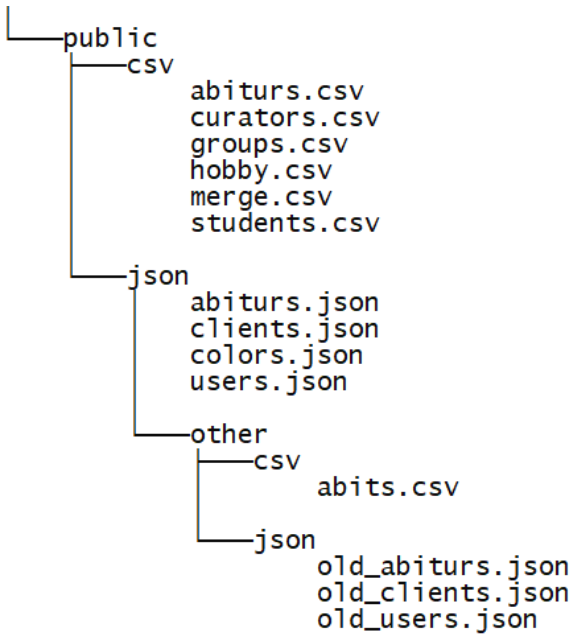

Рисунок 2. Пример расположения файлов в публичной директории веб-сервиса.

3. Если в запросе указаны тип и имя файла, например, `'/json/users.json'`, то сервис возвращает массив объектов из этого файла в формате json. Следует уточнить, что мы решаем упрощенную задачу, в рамках которой условились, что имена файлов не повторяются, даже, если они лежат в разных директориях. При получении запроса с именем фала сервис сам может найти относительный путь к указанному файлу только по его имени с помощью рекурсивной функции перебора директорий.

4. Сервис должен обрабатывать и более сложные запросы. Если в запросе указаны тип, имя файла и названия полей с соответствующими направлениями сортировки, например, `'/json/abiturs.csv/?rating=desc&name=asc'`, где до знака `'='` указывается название поля, а после него – одно из возможных направлений для сортировки (asc – по возрастанию или desc – по убыванию), то сервис возвращает выборку из массива объектов (***включая все существующие поля объ-***

ектов), отсортированную по указанным направлениям. В данном примере `rating=desc&name=asc` указано сортировать по полю рейтинг по убыванию и затем по полю имя по возрастанию. Требуется самостоятельно реализовать метод сортировки по нескольким полям, не используя специализированные библиотеки lodash или ramda.

Для определённости сравните два возможных варианта запроса с указанием типа и файла и указанием типа, файла, выборки полей и направлений сортировки (таб.1).

Таблица 1.

Сравнение результатов выборки.

/json/tiobe.json	/json/tiobe.json/?lang&rat/?rat=desc

[
  {
  "id": "9",
  "lang": "Assembly language",
  "rat": "1.87%"
  },
  {
  "id": "2",
  "lang": "C",
  "rat": "16.56%"
  },
  {
  "id": "5",
  "lang": "C#",
  "rat": "4.92%"
  },
  {
  "id": "3",
  "lang": "C++",
  "rat": "11.94%"
  },
  {
  "id": "13",
  "lang": "Classic Visual Basic",
  "rat": "1.15%"
  },
...
]
	[
	  {
	  "lang": "Python",
	  "rat": 16.66
	  },
	  {
	  "lang": "C",
	  "rat": 16.56
	  },
	  {
	  "lang": "C++",
	  "rat": 11.94
	  },
	  {
	  "lang": "Java",
	  "rat": 11.82
	  },
	  {
	  "lang": "C#",
	  "rat": 4.92
	  },
	  {
	  "lang": "Visual Basic",
	  "rat": 3.94
	  },
	...
	]

5. Если в запросе указаны тип, имя файла, названия полей и в самом конце запроса дополнительно названия полей с соответствующими направлениями сортировки, например, `'/json/abiturs.csv/?id&group&rating&name/?rating=desc&name=asc'`, где до знака `'='` указывается название поля, а после

него – одно из возможных направлений для сортировки (asc – по возрастанию или desc – по убыванию), то сервис возвращает выборку из массива объектов ***только по выбранным в запросе полям***. Например, в указанным выше запросе можно выделить какие именно поля выбирать из объектов (`?id&group&rating&name`) – `id, group, rating, name` и как именно сортировать (`rating=desc&name=asc`) – по полю рейтинг по убыванию и затем по полю имя по возрастанию. Требуется самостоятельно реализовать метод сортировки по нескольким полям, не используя специализированные библиотеки lodash или ramda.

4. Пример программной реализации

Рассмотрим прототип веб-сервиса с минимально достаточным набором файлов и папок:

– about.json – объект с документацией сервиса (листинг 1),

– `app.js` – основное приложение для запуска сервиса (Листинг 2),

– `module.js` – вспомогательный модуль, в котором расположены дополнительные функции получения списка всех файлов, вывода json-строки с документацией сервиса и другие, по необходимости (Листинг 3),

– `library.js` – библиотека классов, в которой расположен один класс для выполнения задач по выборке и сортировке данных из указанного файла (Листинг 4),

– `public/` – директория для размещения файлов во структурированными данными форматов csv или json (рис. 2).

Объект с документацией сервиса отдельно рассматривать не будем, так как его предназначение и содержание обсудили выше (Листинг 1).

Самый важный для веб-сервиса файл – это приложение app.js (Листинг 2), которое решает ряд ключевых задач:

– создаёт объект сервера http.createServer();

– назначает порт port для «прослушки» входящих запросов;

– включает цикл «прослушки» входящих запросов – server.listen();

– назначает запускаемую функцию callback, при получении события request – server.on("request", callback);

– осуществляет маршрутизацию приложения в зависимости от вида запроса с проверкой на корректность запроса – select_case().

Листинг 2.

Программный код приложения app.js.

```
const http = require("http");
const { about, get_files_filter } = require('./module');
let { WorkData } = require('./library');

const server = http.createServer();
const port = 3000;

let select_case = (args, response) => {
    try {
        switch (args.length) {
            case 1:
                about(response);
                break;
            case 2:
                let files =  get_files_filter('./', args[1]);
                response.write(files.join('\n'));
                break;
            case 3:
                let wd = new WorkData(`./public/${args[1]}/${args[2]}`);
                response.write(JSON.stringify(wd.json, null, 4));
                break;
            case 4:
                let wd_sort = new WorkData(`./public/${args[1]}/${args[2]}`);
                response.write(JSON.stringify(wd_sort.json, null, 4));
                break;
            default:
                response.statusCode = 404;
```

```
                response.write('Запрос ошибочный...');
                break;
        }
    } catch (error) {
        console.error(error);
    }
}

let callback = (request, response) => {
    let args = request.url.split('/');
    response.setHeader('Content-Type', 'text/plain; charset=utf-8');
    if (args[args.length-1] === '') args.pop();
    select_case(args, response);
    response.end();
}

server.on("request", callback);
server.listen(port, () => console.log(`localhost:${port}`));
```

Подключаемый к основному приложению вспомогательный модуль module.js содержит такие дополнительный функции как получение и вывод документации об API сервиса и рекурсивный поиск всех файлов по фильтру из запроса пользователя (Листинг 3).

Листинг 3.

Модуль с дополнительными функциями module.js.

```
const about = (res) => {
    let about = require('./about.json');
    res.write(JSON.stringify(about, null, 4));
}

const get_files_filter = (dir, ext) => {
    const rec = (dir) => {
        // тут самостоятельно пропишите
        // программный код рекурсивной функции
        // которая будет собирать все файлы по фильтру
        // рекурсивная функция должна включать
        // шаг рекурсии – для обработки поддиректории
        // точку останова – для добавления файла в массив name_files
        // добавлять в массив, если правильное расширение файла
    }
    let name_files = [];
    rec(dir);
    return name_files;
}

module.exports = {
    about,
    get_files_filter
}
```

И, наконец, библиотека классов library.js (Листинг 4) предназначена для хранения класса WorkData, включающего:

– приватное поле для хранения массива объектов _json,

– свойства json,

– конструктора для инициализации объекта, решающего задачу чтения данных из файла,

– основного метода orderBy, обеспечивающего представление данных по запросу пользователя.

Листинг 4.

Библиотека классов library.js.

```
class WorkData {
    _json; // данные считанные из файла
    get json() {
        return this._json;
    }
    constructor(file_name) {
        this._json = require(file_name);
    }
    orderBy(fields, directs) {
        // тут добавить рекурсивную сортировку по параметрам
        // fields – это массив с полями объектов, по которым следует
        // организовать сортировка, поля представлены в строковом виде
        // например, fields = [‘rat’, ‘name’];
        // directs – это массив направлений для сортировки
        // по соответствующим полям, направления можно указывать
        // строками ‘asc’ или ’desc’,
        // например, directs = [‘desc’,’asc’]
        return this._json;
    }
}

module.exports = {
    WorkData
}
```

Это минимальный набор файлов, классов и методов для обеспечения работоспособности веб-сервиса, реализующего обработку данных в ответ на запрос пользователя. Часть из представленного функционала предстоит разработать в рамках проектно-технологической практики самостоятельно.

5. Подготовка и защита отчёта о практике

Этапы выполнения практики:

- получение задания на выполнение практики;
- обсуждение этапов проектирования и порядка разработки информационной системы;
- подготовка технического задания на проектирование информационной системы;
- разработка структуры для хранения данных для информационной системы;
- проектирование модулей информационной системы;
- тестирование и доработка;
- составление отчёта о выполнении практики;
- защита проектно-технологической практики.

Рекомендуемый объём работы, содержание Отчёта о выполнении практики и постраничное описание отражены в таблице 2.

Таблица 2.

Название пункта и рекомендуемое содержание	Рекомендуемый объем, кол-во страниц
Титульный лист	1
Содержание	1
1. Постановка задачи на проектирование информационной системы *Рекомендуется кратко описать бизнес процесс, требуемые информационные потоки и обосновать необходимость проведения автоматизации.*	1
2. Анализ технологий проектирования *– обзор форматов для хранения данных csv-файлы, json-файлы, xml-файлы;* *– обзор языков программирования (например, C#, Python, Node.js);* *– краткое описание архитектуры веб-сервиса, при этом следует уделить внимание обоснованию выбора форматов для хранения данных и технологий для проектирования веб-сервиса.*	4-8
3. Реализация функционала информационной системы	6-10

– описание используемых структур для хранения данных (csv, json, xml); *– описание форматов http-запросов к сервису;* *– описание методов для обработки данных при выполнении запросов на чтение, добавление, изменение и удаление записей;* *– авторизация (при наличии);* *– описание отдельных, наиболее значимых, классов и методов программной реализации с Листингами кода.*	
Заключение *описание что сделано, выводы об эффективности используемых технологий, о разработанной функциональности сервиса и перспективы дальнейшей совершенствования.*	1
Список литературы	1
Приложение А. Техническое задание	5
Приложение Б. Программный код	5-10
Рецензия *рецензию после сдачи отчёта заполнит преподаватель*	1

Заключение

Работая над технической реализацией информационной системы в рамках проектно-технологической практики, используя представленные в методических рекомендациях материалы обучающийся дополняет и систематизирует знания, полученные им в рамках контактной работы с преподавателем, что позволит ему полностью освоить компетенции, предусмотренные при прохождении проектно-технологической практики.

В рамках организации проектно-технологической практики обучающихся должны реализовываться следующие принципы:

– принцип интерактивности обучения (обеспечение интерактивного диалога и обратной связи, которая позволяет осуществлять контроль и коррекцию действий студента);

– принцип развития интеллектуального потенциала (формирование алгоритмического, наглядно-образного, теоретического стилей мышления, умений принимать оптимальные или вариативные решения в сложной ситуации, умений обрабатывать информацию);

– принцип обеспечения целостности и непрерывности дидактического цикла обучения (предоставление возможности выполнения всех звеньев дидактического цикла в пределах темы).

В процессе выполнения задач практики обучающийся должен активно использовать электронные библиотечные системы, электронные поисковые системы и электронные периодические справочники. Кроме того, обучающийся должен регулярно использовать Интернет-ресурсы, находящиеся в свободном доступе.

Перечень рекомендуемой литературы

Основная:

1. Прохоренок, Н. А. JavaScript и Node.js для веб-разработчиков / Н. А. Прохоренок, В. А. Дронов. – СПб: БХВ-Петербург, 2022. – 768 с.

2. Полуэктова, Н. Р. Разработка веб-приложений: учебное пособие для вузов / Н. Р. Полуэктова. – Москва: Издательство Юрайт, 2022. – 204 с. – (Высшее образование). – ISBN 978-5-534-13715-6. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/496682

3. Молинаро Э. SQL/ Сборник рецептов. – 2-ое изд.: пер. с англ. / Э. Молинаро, Р. де Граф. – СПб: БХВ-Петербург, 2022. – 592 с.

Дополнительная:

1. Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем: учебник для вузов / Е. М. Лаврищева. – 2-е изд., испр. и доп. – Москва: Издательство Юрайт, 2022. – 432 с. – (Высшее образование). – ISBN 978-5-534-07604-2. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/491029

2. Гниденко, И. Г. Технологии и методы программирования: учебное пособие для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. – Москва: Издательство Юрайт, 2022. – 235 с. – (Высшее образование). – ISBN 978-5-534-02816-4. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/489920

3. Лаврищева, Е. М. Программная инженерия. Парадигмы, технологии и CASE-средства: учебник для вузов / Е. М. Лаврищева. – 2-е изд., испр. – Москва: Издательство Юрайт, 2022. – 280 с. – (Высшее образование). – ISBN 978-5-534-01056-5. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/491048

4. Малов, А. В. Концепции современного программирования: учебное пособие для вузов / А. В. Малов, С. В. Родионов. – Москва: Издательство Юрайт, 2022. – 96 с. – (Высшее образование). – ISBN 978-5-534-14911-1. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/485436

Базы данных, информационно-справочные и поисковые системы

1. Электронный каталог библиотеки Пермского ГАТУ: базы данных, содержащие сведения обо всех видах литературы, поступающей в фонд Научной библиотеки Пермского ГАТУ. – URL: https://pgsha.ru/generalinfo/library/webirbis/.
2. Электронная библиотека / Пермский государственный аграрнотехнологический университет имени академика Д. Н. Прянишникова. – URL: https://pgsha.ru/generalinfo/library/elib/.
3. ConsultantPlus (КонсультантПлюс) : компьютерная справочно-правовая система. – URL: https://www.consultant.ru/. – Режим доступа: для авторизированных пользователей. – Доступ из корпусов ПГАТУ.
4. eLIBRARY.RU : научная электронная библиотека. – URL: https://elibrary.ru/defaultx.asp. – Режим доступа: для зарегистрированных пользователей.
5. Polpred.com (Полпред.ком) : электронно-библиотечная система. – URL: https://polpred.com/news. – Режим доступа: для зарегистрированных пользователей.
6. IPRSMART : электронно-библиотечная система. – URL: https://www.iprbookshop.ru/. – Режим доступа: для зарегистрированных пользователей.
7. Гребенникон : электронная библиотека. – URL: https://grebennikon.ru/. – Режим доступа: для зарегистрированных пользователей.
8. Руконт : национальный цифровой ресурс : межотраслевая электронная библиотека. – URL: https://lib.rucont.ru/search. – Режим доступа: для зарегистрированных пользователей.
9. Лань : электронно-библиотечная система. – URL:

https://e.lanbook.com/. – Режим доступа: для зарегистрированных пользователей.
10. Юрайт : электронно-библиотечная система. – URL: https://urait.ru/. – Режим доступа: для зарегистрированных пользователей.
11. Сетевая электронная библиотека (СЭБ). – URL: https://e.lanbook.com/. – Режим доступа: для зарегистрированных пользователей.
12. Электронные информационные ресурсы ФГБНУ ЦНСХБ. – Режим доступа: для авторизированных пользователей. – Доступ из интернет-зала главного корпуса.
13. **Перечень открытых интернет-ресурсов**:
Интуит - Открытый университет:
– Курс «Введение в стандарты WEB»: https://intuit.ru/studies/professional_skill_improvements/1432/info
– Курс «Web-технологии»: https://intuit.ru/studies/professional_skill_improvements/1252/info

Приложение 1

_____

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 _____

202__

1. Общие сведения

1.1 Наименование системы

Полное наименование разрабатываемой системы – «Название Вашего сервиса» *(например, «Данные по абитуриентам»)*.

Краткое наименование – «Краткое название Вашего приложения» *(например, «AbiData»)*.

1.2 Наименование заказчика и исполнителя

Организация: ФГБОУ ВО Пермский ГАТУ.

Адрес: ул. Петропавловская, 23.

Телефон: +7 (342) 217-90-66;

Исполнитель: Ваше Фамилия Имя Отчество.

1.3 Плановые сроки начала и окончания работ

Дата начала работ: __.__.20__. *(день начала практики)*

Дата окончания работ: __.__.20__. *(день окончания практики)*

2. Назначение и цели создания системы

К целям создания веб-сервиса «AbiData» можно отнести *(дальше самостоятельно укажите пункты, которые характеризуют Ваш сервис)*:

– улучшить оперативное взаимодействие и интеграцию модулей;

– автоматизировать опрос сотрудников и студентов университета.

Достижение целей приведёт к следующим положительным результатам:

– руководитель сможет быстрее оценивать оперативную картину видя назначенные мероприятия и поручения в едином потоке информации;

– благодаря возможности комментировать информационную публикацию в модуле «AbiData» отправители избавятся от необходимости получать обратную связь от исполнителей или участников мероприятий через модуль «Сообщения» или «Чат» каждый раз при проведении мероприятия или назначении поручения;

– кураторам или старостам будет удобнее информировать свои группы студентов при помощи информационных публикаций или проводить опросы;

– пользователи портала смогут видеть информационные публикации в отсортированном в порядке убывания даты.

3. Характеристика объекта автоматизации

Объектом автоматизации является подсистема организационной коммуникации интернет-сайта по продажам продукции.

4. Требования к системе

Общие требования к веб-сервису «XXXXX XXXXXX» 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.

4.1 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 и форматах данных (csv, json, xml) веб-сервис может объединить эти данные представив их как единый информационный поток.

4.1.1 Перспективы развития, модернизация системы

Дальнейшим развитием веб-сервиса может быть изменение структуры для хранения данных и перенос всего функционала с синхронного способа взаимодействия на асинхронный.

4.1.2 Требования к квалификации персонала и режиму его работы

Для веб-сервиса не предусматривается визуальных режим отображения даннх, нет графического интерфейса и нет необходимости определять роли пользователей. Разрабатываемый веб-сервис рассчитан исключительно на взаимодействие с информационными системами по протоколу http.

4.1.3 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.

4.1.4 Требования к безопасности

Разрабатываемый информационный веб-сервис должен обеспечивать безопасный доступ к данным, предотвращая несанкционированный доступ или модифицирование данных. Модуль аутентификации подключаемых информационных систем должен обеспечивать защищённый доступ ко всему программному интерфейсу приложения.

Также при разработке модуля необходимо соблюдать разграничение прав на чтение и публикацию информации.

4.1.5 Требования по эргономике и технической эстетике

Программный интерфейс сервиса должен соответствовать промышленным стандартам на размещение и обработку данных в форматах – csv, json, xml.

4.1.6 Требования к программному обеспечению

При проектировании веб-сервиса необходимо эффективно использовать в качестве серверного окружения – программную платформу Node.js, для хранения структурированных данных формат csv, а для хранения иерархически организованных данных формат json или xml.

4.1.7 Требования к техническому обеспечению

Техническое обеспечение системы должно быть рассчитано на одновременное обращение к сервису до ста клиентов (информационных систем) с запросами по http-протоколу. Максимально допустимые технические характеристики сервера:

- процессор – 2x Intel Xeon 3.7 ГГц;
- оперативная память – 4 ГБ;
- дисковая система – 0.25 ТБ;
- сетевой адаптер – 1 Гб/с.

5. 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.

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

При подготовке к вводу в эксплуатацию веб-сервиса отдел внедрения должен обеспечить выполнение следующих работ:

– определить подразделение и ответственных должностных лиц для внедрения сервиса;

– обеспечить пользователей сервиса кратким руководством, которое поможет быстрее освоить программный интерфейс доступа к данным;

– провести опытную эксплуатацию веб-сервиса;

– подготовить отчёт о проделанной работе.

Приложение 2

Министерство науки и высшего образования 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
О ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

на тему: «Разработка REST-сервиса, предоставляющего доступ к информации о *ТУТ НАЗВАНИЕ НА ВАШ ВЫБОР*»

Выполнил:
студент группы ПИб-хххх
направления подготовки
09.03.03 Прикладная информатика
Иванов Иван Иванович

Проверил:
доцент кафедры ИТиПИ, к.т.н., доцент
Беляков Андрей Юрьевич

Пермь – 20__