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 ин- формации о абитуриентах»

Выполнил:

студент группы ПИб-31 направления подготовки

09.03.03 Прикладная информатика Полежаев Егор Вадимович

Проверил:

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

Пермь – 2023

Содержание

[Постановка задачи на проектирование 3](#_Toc125566314)

[Анализ технологий проектирования 4](#_Toc125566315)

[Реализация функционала информационной системы 9](#_Toc125566316)

[Заключение 14](#_Toc125566317)

[Список источников 15](#_Toc125566318)

[ПРИЛОЖЕНИЕ А 17](#_Toc125566319)

[ПРИЛОЖЕНИЕ Б 22](#_Toc125566320)

# Постановка задачи на проектирование

Веб-сервис реализует бизнес-процесс который посредством HTML-запроса выводит определенные данные в определенном порядке.

Архитектура веб-сервиса должна соответствовать принципам REST и обязательно содержать API-документацию с помощью которой пользователь и производит свою работу.

Из JSON или CVS файлов находящихся на сервере(или локальном компьютере) на страницу сайта с помощью HTML-запроса выводятся содержащиеся данные, и при необходимости, сортируются в указанном порядке(возрастания или убывания).

В данном действии есть место автоматизации так как:  
- Во-первых данные которые хранятся в .json и .csv файлах содержатся не во всех компьютерах мира по умолчанию, однако их содержимое должно быть доступно всем пользователям. Веб-сервис позволяет выставить это содержимое в сеть что бы ей могли пользоваться все у кого есть подключению к нему, без скачиваний и регистрации.

- Во-вторых данные помимо вывода могут быть отсортированы для удобства восприятия, например, сортировка лучших по рейтингу может быстро показать у кого больше очков этого рейтинга или сразу выделить топ 10 лучших.

# Анализ технологий проектирования

Сведения о формате CSV

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

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

Сведения о формате JSON

Данные JSON (нотация объектов JavaScript) представлены в виде пар "ключ — значение" в частично структурированном формате. JSON часто сравнивают с XML, так как они могут хранить данные в иерархическом формате. Дочерние данные в них представлены вместе с родительскими. Оба формата являются самоописывающими и удобочитаемыми для пользователя, однако документы JSON, как правило, намного меньше. Поэтому их чаще используют при сетевом обмене данными, особенно с появлением веб-служб на базе REST.

Файлы в формате JSON имеют несколько преимуществ по сравнению с CSV-файлами.

JSON поддерживает иерархические структуры, упрощая хранение связанных данных в одном документе и представление сложных связей.

Большинство языков предоставляют упрощенные библиотеки сериализации JSON или встроенную поддержку десериализации JSON в объекты.

JSON поддерживает списки объектов, помогая избежать беспорядочных преобразований списков в реляционную модель данных.

JSON — это широко используемый формат файлов для баз данных NoSQL, таких как MongoDB, Couchbase и Azure Cosmos DB.

Так как множество данных передачи уже находится в формате JSON, большинство веб-языков изначально поддерживают работу с JSON или используют внешние библиотеки для сериализации и десериализации данных JSON. Благодаря этой поддержке JSON используют в логических форматах посредством представления структуры данных, форматах обмена для горячих данных и хранилищах для холодных данных.

Многие модули обработки пакетных и потоковых данных изначально поддерживают сериализацию и десериализацию JSON. Несмотря на то что данные, содержащиеся в документах JSON, в конечном итоге могут быть сохранены в более оптимизированных по производительности форматах, таких как Parquet или Avro, они служат в качестве необработанных исходных данных, что очень важно для повторной обработки данных (при необходимости).

Использование форматов CSV и JSON

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

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

Python

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

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

Судите сами — «Питону» удалось:

— захватить часть рынка web-разработки;

— обосноваться на рынке десктопных приложений;

— стать лидером в сфере машинного обучения.

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

NODE JS

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

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

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

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

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

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

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

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

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

REST

REST (от англ. Representational State Transfer — «передача репрезентативного состояния» или «передача „самоописываемого“ состояния») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Другими словами, REST — это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы; прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC.

В интернете вызов удалённой процедуры может представлять собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

В отличие от веб-сервисов (веб-служб) на основе SOAP, не существует «официального» стандарта для RESTful веб-API. Дело в том, что REST является архитектурным стилем, в то время как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful-реализаций используют такие стандарты, как HTTP, URL, JSON и, реже, XML.

Вследствии того что основным языком программирования был выбран Node.js вполне логично для хранения данных использовать формат .json как наиболее удобный, читабельный.

# Реализация функционала информационной системы

Структуры данных имеют представленны в двух видах:  
CSV-файлы и JSON-файлы. И первых и вторых файлов 7 штук в сумме всего 14. Расположенны и CSV и JSON-файлы в разных директориях и в одном случае директория вкладывается в другую.

HTTP запросы представленны на главной странице API документации всего имеется 5 команд:

1. “/“

2. “/type”

3. “/type/filename.type”

4. “/type/filename.type/?rating=desc”

5. “/type/filename.type /?id&group&rating&name/?rating=desc”

Каждая последующая помимо выполнения функционала старой команды выполняет ряд новых задач и каждую из них следует описать для дальшейшего понимания процессов:

1.Если сервис получает корневой запрос '/', то возвращает документацию API этого сервиса. Документация пода- ётся 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 сле- дует определиться с необходимыми полями author, version, types\_of\_files, request\_formats и установить их значениями. Этот объект (листинг 1) следует заполнить самостоятельно за- ранее 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=direct3": "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, напри- мер, '/json', то возвращает список файлов соответствующего типа из некоторой публичной директории public/ на веб-сер- висе 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.

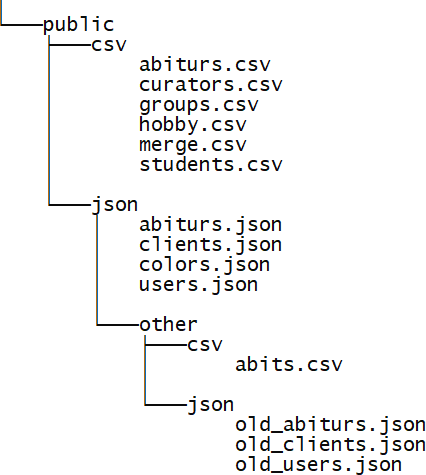


Рисунок 2. Пример расположения файлов в публичной директории веб-сервиса.

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 дирек- ториях. При получении запроса с именем фала сервис сам мо- жет найти относительный путь к указанному файлу только по его имени с помощью рекурсивной функции перебора дирек- торий

4.Сервис должен обрабатывать и более сложные за- просы. Если в запросе указаны тип, имя файла и названия по- лей с соответствующими направлениями сортировки, напри- мер, '/json/abiturs.csv/?rating=desc&name=asc', где до знака '=' указывается название поля, а после него  одно из возможных направлений для сортировки (asc по возраста- нию или desc по убыванию), то сервис возвращает выборку из массива объектов (включая все существующие поля объ- ектов), отсортированную по указанным направлениям. В дан- ном примере rating=desc&name=asc указано сортировать по полю рейтинг по убыванию и затем по полю имя по возраста- нию. Требуется самостоятельно реализовать метод сорти- ровки по нескольким полям, не используя специализирован- ные 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  },  ...  ] |

Стоит кратко упомянуть о важных классах и методах на которых завязана большая часть работы:

1. get\_files\_filter() в файле module.js

const get\_files\_filter = (dir, ext) => {

const rd = require('fs').readdirSync;

const st = require('fs').statSync;

const path = require('path');

const rec = (dir) => {

let items = rd(dir, 'utf-8');

for (let item of items) {

let path\_item = path.join(dir, item.toString());

if (st(path\_item).isDirectory()) {

rec(path\_item, ext);

}

else {

if (path\_item.slice(-3) == ext) { name\_files.push(path\_item);}

else if (path\_item.slice(-4) == ext) {name\_files.push(path\_item);}

}

}

}

let name\_files = [];

rec(dir);

return name\_files;

}

Будет использована в app.js для отдельного вывода файлов.

Производит отделение файлов в папке public основываясь на её расширении.

Проверяя последние 3 или 4 символа она сверяет их с введеной HTTP командой и в соответствии выдает нужный итог.

2. Класс WorkData, comparator и orderBy() в файле library.js

class WorkData {

get json() {

return this.\_json;

}

constructor(file\_name) {

this.\_json = require(file\_name);

}

comparator(a, b, fields, directs) {

let dict = {'asc': +1, 'desc': -1};

let d = dict[directs[0]];

let f = fields[0];

if ((fields.length === 1) || (a[f] !== b[f])) {

return d \* (a[f] > b[f] ? +1 : -1);

}

return this.comparator(a, b, fields.slice(1,), directs.slice(1,));

}

orderBy(fields, directs) {

return this.\_json.sort((a, b) => this.comparator(a, b, fields, directs));

}

}

module.exports = {

WorkData

}

Будет использован в app.js для сортировки.

Компаратор хранит переменные и их изменения, сравнивает их состояния. В то же время рекурсивная функция сортирует и подает новые данные в комнаратор для сравнения.

При вызове в app.js заранее так же будут созданы пару переменных которые хранят параметры для подачи в класс.

# Заключение

По итогу курсового проекта был получен простейший веб-сервис с функционалом позволяющий открывать и читать данные файлов обычным пользователям с помощью обычных HTTP запросов которые передаются на удаленный или локальный сервер.

Были использованы оптимальные инструменты разработки(Node.js, VS Code, и т.д.) и задействованна архитектура разработки REST.

Инструменты удобны, просты, но не лишены своей функциональности по сравнению со своими аналогами(например Python/Django и Visual Studio).  
К архитектуре разработки можно быстро привыкнуть из-за интуитивно понятной логики в связи с чем работа на ней не составит труда.

В перспективе первое за что надо взяться для доработки веб-сервиса это его дизайн и визуализация. Хоть и минималистичный дизайн проще и удобнее воспринимать(по крайней мере мне), но многим будущим пользователям которые будут пользоваться веб-сервисам может не понравится это и они заранее сочтут проект нестоящим их внимания или вовсе полностью неработающим из-за чего просто не запустят его и не введут ни одной команды.

# Список источников

1. Прохоренок, Н. А. JavaScript и Node.js для веб-разработчи- ков / Н. А. Прохоренок, В. А. Дронов.  СПб: БХВ-Петербург, 2022.  768 с.
2. Полуэктова, Н. Р. Разработка веб-приложений: учебное по- собие для вузов / Н. Р. Полуэктова.  Москва: Издательство Юрайт, 2022.  204 с.  (Высшее образование).  ISBN 978-5- 534-13715-6.  Текст: электронный // Образовательная плат- форма Юрайт [сайт].  URL: <https://urait.ru/bcode/496682>(дата обращения: 27.11.2022).
3. Молинаро Э. SQL/ Сборник рецептов.  2-ое изд.: пер. с англ. / Э. Молинаро, Р. де Граф.  СПб: БХВ-Петербург, 2022.

 592 с.

1. Маркин, А.В. Программирование на SQL: учебное пособие для среднего профессионального образования / А.В. Маркин. Москва: Издательство Юрайт, 2022.  435 с.  (Профессио- нальное образование).  ISBN 978-5-534-11093-7.  Текст : электронный // Образовательная платформа Юрайт [сайт].  URL: <https://urait.ru/bcode/495666> (дата обращения: 27.11.2022).
2. 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 плат- форма Юрайт [сайт].  URL: <https://urait.ru/bcode/491029>(дата обращения: 27.11.2022).
3. 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> (дата обращения: 27.11.2022).

1. Лаврищева, Е. М. Программная инженерия. Парадигмы, технологии и CASE-средства: учебник для вузов / Е. М. Лаври- щева.  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> (дата обращения: 27.11.2022).
2. Малов, А. В. Концепции современного программирования: учебное пособие для вузов / А. В. Малов, С. В. Родионов.  Москва: Издательство Юрайт, 2022.  96 с.  (Высшее образо- вание).  ISBN 978-5-534-14911-1.  Текст : электронный // Образовательная платформа Юрайт [сайт].  URL: <https://urait.ru/bcode/485436>(дата обращения: 27.11.2022).
3. Электронный каталог библиотеки Пермского ГАТУ: базы данных, содержащие сведения обо всех видах литера- туры, поступающей в фонд Научной библиотеки Пермского ГАТУ. – URL: <https://pgsha.ru/generalinfo/library/webirbis/>.
4. Электронная библиотека / Пермский государственный аграрнотехнологический университет имени академика Д. Н. Прянишникова. – URL: <https://pgsha.ru/generalinfo/library/elib/>.

ПРИЛОЖЕНИЕ А

**Техническое задание**

наименование организации – разработчика ТЗ на АС

УТВЕРЖДАЮ

Руководитель

(должность, наименование предприятия – заказчика АС)

Личная подпись Расшифровка подписи

(печать)

Дата

УТВЕРЖДАЮ

Руководитель

(должность, наименование предприятия – разработчик АС)

Личная подпись Расшифровка подписи

(печать)

Дата

наименование вида АС

наименование объекта автоматизации

сокращённое наименование АС

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

На листах

Действует с

2021

1. **Общие сведения**
   1. **Наименование системы**

Полное наименование разрабатываемой системы – «Данные по абитуриентам».

Краткое наименование – «ADA»

* 1. **Наименование заказчика и исполнителя** Организация: ФГБОУ ВО Пермский ГАТУ. Адрес: ул. Петропавловская, 23.

Телефон: +7 (342) 217-90-66;

Исполнитель: Полежаев Егор Вадимович

* 1. **Плановые сроки начала и окончания работ**

Дата начала работ: 01.09.2022

Дата окончания работ: 27.01.2023

1. **Назначение и цели создания системы**

К целям создания веб-сервиса «ADA» можно отнести:

лей;

* улучшение оперативного взаимодействия и интеграция моду-
* автоматизировать чтение и хранение данных в файлах Достижение целей приведёт к следующим положительным ре-

зультатам:

* + пользователи смогут оперировать информацией посредством быстрых команд для изучения файлов хранящихся в директориях сервера

1. **Характеристика объекта автоматизации**

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

1. **Требования к системе**

Общие требования к веб-сервису «Данные по абитуриентам» являются:

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

1. **Требования к системе**

Общие требования к веб-сервису «Данные по абитуриентам» являются:

* надёжность и работоспособность;
* интуитивно понятный интерфейс;
* лицензионная чистота – применение средств в рамках общего лицензионного соглашения касательно корпоративного портала;
* соблюдение информационной безопасности и разграничение прав доступа к данным.
  1. **Требования к способам и средствам связи для информацион- ного обмена между компонентами**

Для обеспечения информационного обмена компоненты подси- стемы должны взаимодействовать с объединённой информационной базой данных. Благодаря хранению данных в различных схемах и фор- матах данных (csv, json, xml) веб-сервис может объединить эти данные представив их как единый информационный поток.

* + 1. **Перспективы развития, модернизация системы**

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

* + 1. **Требования к квалификации персонала и режиму его работы**

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

* + 1. **Требования к надёжности технических средств и программ- ного обеспечения**

Надёжность по отношению к техническим средствам должна обеспечиваться использованием в системе средств повышенной отка- зоустойчивости и их резервированием, а также дублированием носите- лей информационных банков данных.

Надёжность программного комплекса обеспечивается использо- ванием сертифицированных операционных систем, общесистемных программных средств и инструментальных программных систем, ис- пользуемых при разработке программного обеспечения. Само про- граммное обеспечение должно обеспечивать защиту от некорректных действий пользователей и ошибочных исходных данных.

* + 1. **Требования к безопасности**

Разрабатываемый информационный веб-сервис должен обеспе- чивать безопасный доступ к данным, предотвращая несанкционирован- ный доступ или модифицирование данных. Модуль аутентификации подключаемых информационных систем должен обеспечивать защи- щённый доступ ко всему программному интерфейсу приложения.

Также при разработке модуля необходимо соблюдать разграни- чение прав на чтение и публикацию информации.

* + 1. **Требования по эргономике и технической эстетике**

Программный интерфейс сервиса должен соответствовать про- мышленным стандартам на размещение и обработку данных в форма- тах  csv, json, xml.

* + 1. **Требования к программному обеспечению**

При проектировании веб-сервиса необходимо эффективно ис- пользовать в качестве серверного окружения  программную плат- форму Node.js, для хранения структурированных данных формат csv, а для хранения иерархически организованных данных формат json или xml.

* + 1. **Требования к техническому обеспечению**

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

* + - * процессор – 2х Intel Xeon 3.7 ГГц;
      * оперативная память – 4 ГБ;
      * дисковая система – 0.25 ТБ;
      * 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-сервиса отдел внед- рения должен обеспечить выполнение следующих работ:

* определить подразделение и ответственных должностных лиц для внедрения сервиса;
* обеспечить пользователей сервиса кратким руководством, ко- торое поможет быстрее освоить программный интерфейс доступа к данным;
* провести опытную эксплуатацию веб-сервиса;
* подготовить отчёт о проделанной работе.

# ПРИЛОЖЕНИЕ Б

**Листинг кода “about.js”**

{

"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=direct3": "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"

}

}

**Листинг кода “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: // http://localhost:3000

about(response);

break;

case 2: // http://localhost:3000/json или http://localhost:3000/csv

let files = get\_files\_filter('./public/', args[1]);

response.write(files.join('\n'));

break;

case 3: // http://localhost:3000/json/abiturs.json

let wd = new WorkData(`./public/${args[1]}/${args[2]}`);

response.write(JSON.stringify(wd.json));

break;

case 4: // http://localhost:3000/json/abiturs.json/?rating=desc&name=asc

params = args[3].slice(1, args[3].length).split('&')

let fields = []

let directs = []

for(i in params){

let x = params[i].split('=')

fields.push(x[0])

directs.push(x[1])

}

let wd\_order = new WorkData(`./public/${args[1]}/${args[2]}`).orderBy(fields, directs);

response.write(JSON.stringify(wd\_order));

break;

default:

response.statusCode = 404;

response.write('Запрос ошибочный...');

break;

}

} catch (error) {

console.error(error);

}

response.end();

}

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);

}

server.on("request", callback);

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

**Листинг кода “library.js”**

class WorkData {

get json() {

return this.\_json;

}

constructor(file\_name) {

this.\_json = require(file\_name);

}

comparator(a, b, fields, directs) {

let dict = {'asc': +1, 'desc': -1};

let d = dict[directs[0]];

let f = fields[0];

if ((fields.length === 1) || (a[f] !== b[f])) {

return d \* (a[f] > b[f] ? +1 : -1);

}

return this.comparator(a, b, fields.slice(1,), directs.slice(1,));

}

orderBy(fields, directs) {

return this.\_json.sort((a, b) => this.comparator(a, b, fields, directs));

}

}

module.exports = {

WorkData

}

**Листинг кода “module.js”**

const { arch } = require('os');

const about = (res) => {

let about = require('./about.json');

res.write(JSON.stringify(about, null, 4));

}

const get\_files\_filter = (dir, ext) => {

const rd = require('fs').readdirSync;

const st = require('fs').statSync;

const path = require('path');

const rec = (dir) => {

let items = rd(dir, 'utf-8');

for (let item of items) {

let path\_item = path.join(dir, item.toString());

if (st(path\_item).isDirectory()) {

rec(path\_item, ext);

}

else {

if (path\_item.slice(-3) == ext) { name\_files.push(path\_item); }

else if (path\_item.slice(-4) == ext) { name\_files.push(path\_item); }

}

}

}

let name\_files = [];

rec(dir);

return name\_files;

}

module.exports = {

about,

get\_files\_filter

}

**Листинг файла “abiturs.json”**

[

{

"id": "1",

"lastName": "Мишкин",

"rating": "217",

"gender": "1",

"birthDate": "2002-08-23",

"city": "Кунгур"

},

{

"id": "2",

"lastName": "Бортич",

"rating": "224",

"gender": "0",

"birthDate": "2002-06-03",

"city": "Лысьва"

},

{

"id": "3",

"lastName": "Деревянко",

"rating": "182",

"gender": "0",

"birthDate": "2002-02-20",

"city": "Оса"

},

{

"id": "4",

"lastName": "Столбова",

"rating": "194",

"gender": "0",

"birthDate": "2003-01-22",

"city": "Кунгур"

},

{

"id": "5",

"lastName": "Хомич",

"rating": "205",

"gender": "0",

"birthDate": "2002-04-23",

"city": "Кунгур"

},

{

"id": "6",

"lastName": "Круглов",

"rating": "191",

"gender": "1",

"birthDate": "2002-04-23",

"city": "Березники"

},

{

"id": "7",

"lastName": "Иванов",

"rating": "192",

"gender": "1",

"birthDate": "2002-05-17",

"city": "Кунгур"

},

{

"id": "8",

"lastName": "Петров",

"rating": "191",

"gender": "1",

"birthDate": "2002-11-25",

"city": "Кунгур"

},

{

"id": "9",

"lastName": "Сидоров",

"rating": "196",

"gender": "1",

"birthDate": "2004-01-20",

"city": "Кунгур"

},

{

"id": "10",

"lastName": "Капустин",

"rating": "196",

"gender": "1",

"birthDate": "2002-06-04",

"city": "Кунгур"

},

{

"id": "11",

"lastName": "Томатова",

"rating": "201",

"gender": "0",

"birthDate": "2003-04-16",

"city": "Кунгур"

},

{

"id": "12",

"lastName": "Ежова",

"rating": "214",

"gender": "0",

"birthDate": "2001-10-07",

"city": "Лысьва"

},

{

"id": "13",

"lastName": "Микова",

"rating": "222",

"gender": "0",

"birthDate": "2001-10-07",

"city": "Пермь"

},

{

"id": "14",

"lastName": "Мамин",

"rating": "199",

"gender": "1",

"birthDate": "2001-11-10",

"city": "Пермь"

},

{

"id": "15",

"lastName": "Комов",

"rating": "195",

"gender": "1",

"birthDate": "2002-01-17",

"city": "Пермь"

}

]