Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

РАЗРАБОТКА HTTP-СЕРВЕРА

БГУИР КР 1-40 02 01 201 ПЗ

Студент: Д. Г. Альховик

Руководитель: Е. Д. Игнатьев

Минск 2018

Оглавление

[ВВЕДЕНИЕ 5](#_Toc6619609)

[1 ОБЗОР ИСТОЧНИКОВ 6](#_Toc6619610)

[1.1 Архитектура «Клиент-сервер» 6](#_Toc6619611)

[1.2 Обзор протокола HTTP 7](#_Toc6619612)

[1.3 Сервлеты 8](#_Toc6619613)

[1.4 Контейнер сервлетов Apache Tomcat 8](#_Toc6619614)

[1.5 Система управления базами данных MongoDB 8](#_Toc6619615)

[1.6 Обзор аналогов 9](#_Toc6619616)

[1.7 Постановка задачи 10](#_Toc6619617)

[2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ 11](#_Toc6619618)

[3 ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ 13](#_Toc6619619)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 16](#_Toc6619620)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 20](#_Toc6619621)

[6 ТЕСТИРОВАНИЕ 21](#_Toc6619622)

[ЗАКЛЮЧЕНИЕ 26](#_Toc6619623)

[**ПРИЛОЖЕНИЕ А** 28](#_Toc6619624)

[**ПРИЛОЖЕНИЕ Б** 29](#_Toc6619625)

[**ПРИЛОЖЕНИЕ В** 30](#_Toc6619626)

# ВВЕДЕНИЕ

Во все времена информация была и остаётся одним из основных жизненно необходимых ресурсов человека. Она является основным безопасным способом передачи опыта от одного человека другому, а, следовательно, получая и обрабатывая большие объемы информации человек получает знания необходимые ему для жизнедеятельности и самосовершенствования.

С течением времени человечество накопило очень много сведений об окружающем нас мире, но сохранить и осмыслить столько данных человек не способен. Тогда возникает вопрос: как не потерять весь тот опыт, который человечество накапливало веками?

Выходом стало фиксирование полученной информации во всевозможных хранилищах, таких как книги. Однако и книга не всегда удобна для использования из-за объёмных размеров и трудности быстрого поиска необходимых сведений. Тогда на помощь пришли вычислительные машины, которые способны хранить огромные объёмы информации при помощи баз данных.

Хотя работа с базами данных не всегда является удобной напрямую, так как иногда необходим определенный осмысленный поиск среди хранящийся информации, либо же некоторые данные и вовсе не предоставляют ценности без соответствующей обработки и т.д.

Отличным решением для данной проблемы станет использование технологии «клиент-сервер». В данном случае это удобно, так как пользователь может доступным для него образом отправить запрос на сервер, который сумеет грамотно его обработать и выполнить, чтобы удовлетворить потребность клиента. А дабы работа клиента с сервером могла быть оперативной из любой точки земного шара следует использовать «всемирную сеть» как способ передачи запросов.

Целью данного курсового проекта будет разработка сервера, позволяющего сохранять, обрабатывать и возвращать необходимые данные по запросу пользователя и работающего по протоколу http(HyperText Transfer Protocol).

Таким образом в результате должны получить готовый http-сервер, способный работать с несколькими типами клиентов для хранения и обработки их информации на удалённом носителе.

В соответствии с поставленной целью определены следующие задачи:

* Организация обработки запроса пользователя
* Преобразование данных для их хранения
* Запись данных в базу данных
* Чтение данных из базы данных
* Организация работы по протоколу http

# ОБЗОР ИСТОЧНИКОВ

Для решения поставленных задач необходимо определить нужные компоненты для создания http-сервер. Для создания связи между клиентом и сервером нужно использовать соответствующую сетевую архитектуру. Для её реализации и построения корректной работы с базами данных также необходимо использовать подходящий для этого язык программирования.

Важно также выбрать базу данных, которая позволила бы хранить различные типы данных для более универсального использования сервера. Связь между протоколом и сервером воспользуемся стандартными http-протоколами.

## Архитектура «Клиент-сервер»

«Клиент-сервер» — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине).

Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами.

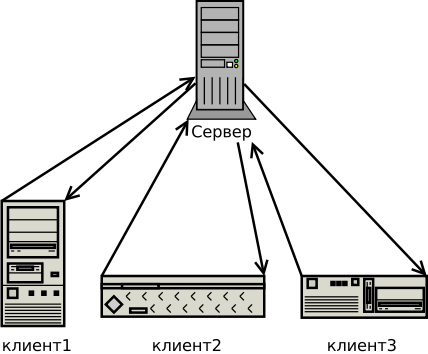


Рисунок 1.1 – Сетевая архитектура «Клиент-сервер»

Преимуществами данного подхода являются:

* Отсутствие дублирования кода программы-сервера программами-клиентами.
* Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
* Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.

## Обзор протокола HTTP

HyperText Transfer Protocol – это протокол передачи данных прикладного уровня, предназначенный передачи произвольных данных. Основой HTTP является технология «клиент-сервер».

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д.

Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI.

## Сервлеты

Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ.

Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения веб-серверов. Для таких приложений технология Java Servlet определяет HTTP-специфичные сервлет классы. Пакеты javax.servlet и javax.servlet.http обеспечивают интерфейсы и классы для создания сервлетов.

Описать работу сервлета можно так: на сервер приходит запрос от клиента, запрос содержит внутри себя URL и параметры. Сервер имеет специальный конфигурационный файл, который ему сообщает о том, какой сервлет надо выполнить в случае прихода определенного URL. Сервлет выполняется (там вы можете использовать параметры) и создает HTML-страницу, которая отсылается клиенту.

Сервер по сути является контейнером (теперь уже не визуальных компонентов), который загружает сервлеты, выполняет их, вызывая определенные методы и получив от них результат, отправляет его клиенту.

## Контейнер сервлетов Apache Tomcat

Tomcat (в старых версиях — Catalina) — контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов, спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java.

Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish.

Для данного проекта нам подойдёт такой компонент как Coyote, который поддерживает протокол HTTP 1.1 для веб-серверов или контейнера приложений. Coyote прослушивает входящие соединения на определённом TCP порту сервера, пересылает запросы в механизм Tomcat для обработки запросов и отправляет ответ назад запрашивающему клиенту.

## Система управления базами данных MongoDB

MongoDB — документоориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++.

Система поддерживает ad-hoc-запросы: они могут возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск по регулярным выражениям. Также можно настроить запрос на возвращение случайного набора результатов.

Данная система подходит нам больше всего, так как она не хранит информацию в таблицах, что удобно нам, потому что мы не знаем изначально, в каком виде информация будет поступать нам на сервер. Это даёт нам возможность хранить объекты целиком, не конвертируя содержимое данных.

## Обзор аналогов

Сервис разрабатывается для сохранения нужной нам информации на удаленном сервере по заданному заголовку, что позволяет нам, не имея логина и пароля, а лишь зная нужную комбинацию символов, получать сохраненную ранее информацию или записывать новую.

Важной особенностью также является то, что это не будет какое-то приложение, которое необходимо скачивать и устанавливать на своём устройстве, так как всё взаимодействие с пользователем происходит через страницу интернет-браузера по заданному адресу. Это даёт широкий спектр возможностей и преимуществ перед другими аналогами, так как делает сервис легко запускаемым с любого устройства, поддерживающего работу браузера, и доступным для любого пользователя.

Основными аналогами данного сервиса является Listik, который является сервисом онлайн заметок для хранения страниц браузера, нужной информации или файлов. Из приложений можно выделить такие крупные проекты как Simplenote и Evernote.

1.6.1 Онлайн сервис заметок Listik

Listik – онлайн сервис заметок, разработанный компанией Sinta в 2013 году, который предназначен для общего использования в быту. Довольно простой интерфейс в сочетании с приятным дизайном делает Listik отличным сервисом для хранения своих мыслей, небольших фалов и скриншотов браузера.

В сервисе каждый пользователь имеет свой кабинет, в котором он может работать и составлять свои планы или хранить нужные данные.

Из минусов можно заметить лишь отсутствие совместной работы пользователей, что не дает использовать данный сервис для работы в компании.

1.6.2 Приложение Simpleton

Приложение позволяет синхронизировать записи между всеми устройствами без оформления подписки. Пользователь имеет возможность обмениваться заметками с другими и вести с ними совместную работу. Интерфейс Simplenote очень прост и удобен в управлении.

Программа поддерживается всеми популярными операционными системами. Ключевой особенностью Simplenote является исключительная скорость и эффективность работы.

1.6.3 Приложение Evernote

Evernote позволяет создавать как простые, так и сложные рабочие процессы, используя комбинацию заметок и тегов, чтобы все было организовано.

Одной из лучших возможностей для сбора исследований является расширение Web Clipper (поддерживается в Chrome, Firefox и Safari), которое позволяет сэкономить целые веб-страницы, включая текст, изображения и PDF-файлы, одним щелчком мыши. Доступ к заметкам можно получить на ноутбуках, мобильных устройствах. Другие функции включают возможность установки напоминаний, представления стилей PowerPoint и объединение их друг с другом.

Недавние обновление включает новые таблицы и интеграцию с Siri для тех, кто использует версию iOS.

## Постановка задачи

На основе контейнера сервлетов Tomcat создать сервер, способный обрабатывать запросы клиента, выполнять соответствующие операции с соблюдением протокола HTTP.

Среди основного функционала необходимо создать простой и понятный набор инструментов, помогающий связать запросы пользователя и работу с базой данных MongoDB.

Среди дополнительных задач, которые следовало бы реализовать: создание личных кабинетов, расширить возможности хранения различных типов данных и расширить диапазон выполняемых запросов.

# СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описана структурная схема работы приложения. Сама структурная схема представлена в приложении А.

После изучения теоретических аспектов разрабатываемого программного модуля, появилась возможность разбить систему на отдельные функциональные модули, что позволяет отойти от проектирования монолитного стиля проектирования приложения и перейти к модульности, что позволит разбить приложение на заменяемые блоки, которые облегчат разработку и обновление приложения.

Каждый модуль обладает необходимым функционалом для решения одной или нескольких задач. Модули взаимодействуют с помощью различных данных между собой позволяя приложению работать.

Работа программы разбита на четыре основных блоков:

* блок пользовательского ввода информации,
* блок обработки запроса,
* блок работы с сервером,
* блок работы с базой данных.
  1. Блок пользовательского ввода информации

Блок пользовательского ввода информации содержит функции для получения от пользователя необходимых данных, преобразования этих данных в соответствующий запрос и передачи этого запроса во второй блоке, блоке обработки запроса, поэтому связь между ними односторонняя.

1.2 Блок обработки запроса

Блок обработки запроса служит для преобразования запроса, пришедшего из блока пользовательского ввода информации в соответствующий ряд команд для выполнения блока работы с сервером.

1.3 Блок работы с сервером

Блок работы с сервером при получении определенных команд отвечает за переброску команд в блок работы с базой данных для корректного их выполнения. Также этот блок обрабатывает результаты выполнения блока работы с базой данных и передаче их соответствующим функциям для отображения в правильном виде полученных данных, поэтому он обладает широким спектром возможностей и является основным звеном связывающим и согласующем работу базы данных и пользователя.

1.4 Блок работы с базой данных

Блок работы с базой данных отвечает за корректную запись и удаления требуемой по запросу информации. Этот блок также может отправлять данные в блок работы с сервером, в которых будет находится результаты выполнения поставленных задач или запрашиваемой информации. Более сложной частью работы данного блока также является изменение уже существующих данных в базе, так как сочетает в себе грамотную работу по поиску и замены определённого поля в элементе.

# ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Рассмотрим подробно функционирование программы. Для этого проведем анализ основных модулей программы и рассмотрим их зависимости. А также проанализируем все классы, которые входят в состав кода программы, и рассмотрим назначение всех методов, свойств и переменных класса.

Условно программный модуль можно разделить на следующие части:

* блок обработвки запросов;
* блок работы с базой данных.

3.1 Блок обработки запросов

Java — сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре с помощью виртуальной Java-машины. Данный язык удобен для проекта, так как он хорошо совместим с Tomcat и имеет библиотеки для работы с сервлетами.

Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ. По сути на сервер приходит запрос от клиента, который содержит внутри себя URL и параметры, благодаря которым сообщается, какой именно сервлет необходимо выполнить. В итоге можно сказать, что сервер является контейнером, который грамотно выбирает сервлеты, вызывая определённые методы и получив от них результат, отправляет его клиенту.

3.1.1 Объект Controller наследуется от HttpServlet:

Методы:

– String getBody(HttpServletRequest req) – функция для получения тела при запросах и преобразования её в строку.

– List<String> parseBodyToList(String string) – функция для преобразования строки, полученной от функции getBody(HttpServletRequest req), в список для удобства работы с переданными данными.

– boolean isValid(char symbol) – функция для определения разрешенных символов, передаваемых в теле запроса, используется функцией parseBodyToList(String string).

– void doGet(HttpServletRequest req, HttpServletResponse resp) – функция для обработки Get-запросов.

– void doPost(HttpServletRequest req, HttpServletResponse resp) – функция для обработки Post-запросов.

– void doPut(HttpServletRequest req, HttpServletResponse resp) – функция для обработки Put-запросов.

– void doDelete(HttpServletRequest req, HttpServletResponse resp) – функция для обработки Delete-запросов.

Именно этот класс отвечает за выбор принятие запроса и содержит методы для его обработки.

3.2 Блок работы с базой данных

Для улучшения читаемости кода и соблюдения single responsibility principle создадим 2 отдельных класса: один для хранения данных во время обработки запроса и второй для непосредственной работы с базой данных. Для этого используется ряд библиотек, предоставленных компанией MongoDB Inc., для работы с базой данных MongoDB в среде Java.

3.2.1 Объект Note:

Поля:

–String HEADER – константное значение, необходимое для создания соответствующего поля в базе данных.

–String INF – константное значение, необходимое для создания соответствующего поля в базе данных.

-String header – необходима для хранения заголовка.

-String inf – необходима для хранения информации, прикреплённой к данному заголовку.

Методы:

– Note(String header, String inf) – конструктор, в котором инициализируются поля класса.

– void setHeader(String header) – функция для изменения значения поля header.

– void setInf(String inf) – функция для изменения значения поля inf.

– String getHeader() – функция для получения значения поля header.

– String getInf() – функция для получения значения поля inf.

3.2.2 Объект MongoWorkNote:

Поля:

– Mongo mongo – необходимо для хранения значений базы данных hostman и port.

– DB db – необходимо для хранения имени подключаемой базы данных.

– DBCollection collection – необходимо для хранения имени коллекции в базе данных.

Методы:

– MongoWorkNote() – конструктор, в котором инициализируются поля класса.

– String getByHeader(String searchHeader) – функция для возврата содержимого закрепленного за переменной searchHeader.

– void add(Note note) – функция для записи объекта Note в базу данных.

– Note get(String searchHeader) – функция для получения объекта Note из базы данных.

– void update(Note note) – функция для обновления содержимого объекта Note в базе данных.

– void delete(Note note) – функция для удаления объекта Note из базы данных.

Методы данного класса могут создавать пользовательские Exceptions как AlreadyHasHeaderException, NoHeaderEcxeption, NoInformationException, NoSuchElementInDBException, ParameterException, которые обрабатываются при их появлении в методах класс Controller.

# РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

В данном разделе рассмотрим часть основных алгоритмов:

1. Алгоритм обработки тела запроса;
2. Алгоритм получения содержимого элемента базы данных по заданному заголовку;
3. Алгоритм записи элемента в базу данных.

4.1 Алгоритм обработки тела запроса

При обработки запросов Put, Delete и Post нужные параметры предаются не через параметры запросы, а через его тело. Однако у переменной HttpServletRequest req отсутствует метод для получения тела запроса, поэтому появляется необходимость в его создании.

Тело запроса можно получить используя следующий метод:

private String getBody(HttpServletRequest req) throws IOException {

StringBuilder buffer = new StringBuilder();

BufferedReader reader = req.getReader();

String line;

while ((line = reader.readLine()) != null) {

buffer.append(line);

buffer.append('\0');

}

String data = buffer.toString();

return data;

}

Здесь с помощью метода интерфейса ServletRequest можно получить тело запроса, представленного в виде BufferedReader. Данный метод приводит данный тип к нужному нам типу String.

После чего необходимо разделить данную строку, так как в ней все параметры записаны в неподходящем для работы виде (name1=value1&name2=value2). Для дальнейшего использования параметров будет удобно преобразовать данную строку в список строк, так как нужное нам значения поля в таком случае окажется следующем после имени искомого поля.

private List<String> parseBodyToList(String string){

char[] stringChar=string.toCharArray();

List<String> list=new ArrayList<String>();

int i=0;

while(i<stringChar.length){

StringBuilder buffer=new StringBuilder();

while (isValid(stringChar[i])){

if(stringChar[i]=='%'){

buffer.append(' ');

i+=3;

continue;

}

buffer.append(stringChar[i]);

i++;

}

String string1=buffer.toString();

list.add(string1);

i++;

}

return list;

}

Данный метод разделяет все записанные строки, состоящие за символов разрешённых для параметров и определённых методом isValid(char symbol) символов, и помещает их в список.

После чего работа с данным списком осуществляется следующем образом:

private String getFromBodyListByName(List<String> list, String name) throws ParameterException {

if(!list.contains(name)){

throw new ParameterException();

}

int index=0;

index=list.indexOf(name);

index++;

String result=list.get(index);

return result;

}

Здесь проверяется наличие искомого нами параметра и, если такой имеется, возвращается его значение, которое хранится следующем элементом в списке. Если же такого значения нет создается соответствующее исключение.

4.2 Алгоритм получения содержимого элемента базы данных по заданному заголовку

Для получения содержимого воспользуемся данным методом, также использующий библиотеки MongoDB:

public String getByHeader(String searchHeader) throws NoHeaderEcxeption, NoSuchElementInDBException {

if (searchHeader == null || searchHeader.isEmpty()) {

throw new NoHeaderEcxeption();

}

String result = null;

BasicDBObject query = new BasicDBObject();

query.put(Note.HEADER, searchHeader);

DBObject findElement = collection.findOne(query);

if (findElement == null) {

throw new NoSuchElementInDBException();

}

result = String.valueOf(findElement.get(Note.INF));

return result;

}

Данный метод основывается на методе DBCollection.findOne(BasicDBObject ob), который возвращает объект из базы данных с заданным параметром через передаваемый ему объект. В случае отсутствия элемента в базе данных создаётся NoSuchElementInDBException(). После чего с помощью метода DBCollection.get(Sting field) возвращается содержимое поля INF, которое в последствии преобразуется в нужный нам тип String.

4.3 Алгоритм записи элемента в базу данных

Для записи элементов в базу данных используются следующий метод:

public void add(Note note) throws AlreadyHasHeaderException, NoHeaderEcxeption {

try{

String string=getByHeader(note.getHeader());

if(string!=null) {

throw new AlreadyHasHeaderException();

}

}catch (NoSuchElementInDBException e) {

}

String jSONstring=new Gson().toJson(note);

BasicDBObject dbObject= (BasicDBObject) JSON.parse(jSONstring);

collection.insert(dbObject);

}

В данном методе блок try-catch предназначен для проверки отсутствия элемента с таким же заголовком, так как для нашего сервиса каждый заголовок уникален. Для данной проверки используется метод getByHeader(String searchHeader), который возвращает содержимое элемента с заданным заголовком, если такой имеется, или NoSuchElementInDBException при его отсутствии.

Для записи передаваемого объекта в коллекцию необходимо создать элемент самой коллекции, содержащий те же поля, что и передаваемый объект. Самым простым способом для создания такого объекта является представить передаваемый объект в JSON формате, так как MongoDB имеет ряд функций для работы с этим форматом. Также использование такого способа записи позволяет переопределять данный метод для передачи объектов любого пользовательского типа. После создания объекта запись в базу данных осуществляется с помощью метода DBCollection.insert(BasicBDObject ob).

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Поскольку в данной работе задачей стояло написание сервера, что является лишь частью полноценного сервиса, спроектированного на основе архитектуру «клиент-сервер», пользователем в данном случае также будет является разработчик, разрабатывающий клиентскую часть сервиса.

Для корректной работы данного сервера нужно учесть следующие условия:

- использование верного URL адреса для запросов;

- использование параметров запроса при передаче значений при Get-запросе;

- использование тела запроса при передаче значений при Post-, Put- и Delete-запросе;

- наличие соответствующих методов, способных обрабатывать и принимать объекты в JSON формате;

- рекомендуется наличие класса для хранения объекта возвращаемого при Get-запросе;

- возможность обработки сообщений и статусов об успешном выполнении запроса, не требующего возврата объекта;

- возможность обработки возвращаемых ошибок при неверном выполнении запроса или при неверно переданных параметрах.

При соблюдении требуемых условий дополнительной установки каких-либо компонентов не требуется.

# ТЕСТИРОВАНИЕ

Для тестирования сервиса воспользуемся следящими ПО:

1. MongoDB Compass Community (MongoDB Inc.) – инструмент графического интерфейса для отображения информации о базе данных MongoDB и выполнения запросов.
2. Postman (Postman Inc.) – приложение для тестирования API.

Проведём тестирование некоторых запросов.

|  |  |  |  |
| --- | --- | --- | --- |
| Вид запроса | Действие | Ожидаемый результат | Успешность тестирования |
| 1 | 2 | 3 | 4 |
| GET | Отправка запроса и значений в параметре запроса | Искомый объект в JSON формате или сообщения об ошибке | да |
| POST | Отправка запроса и значений в теле запроса | Сообщение об успешном добавлении элемента или ошибке | да |
| PUT | Отправка запроса и значений в теле запроса | Сообщение об успешном изменении элемента или ошибке | да |
| DELETE | Отправка запроса и значений в теле запроса | Сообщение об успешном удалении элемента или ошибке | да |

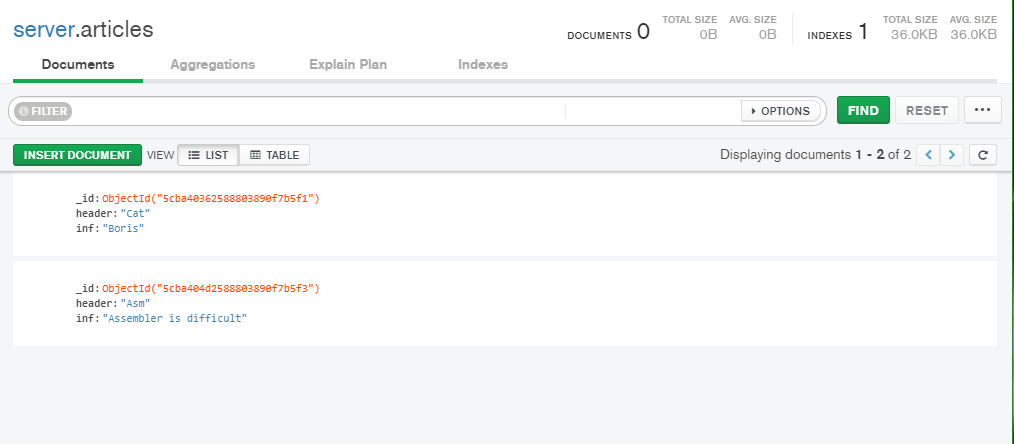


Рисунок 1 – состояние базы данных до проведения тестов

6.1 Результат тестирования GET-запроса.

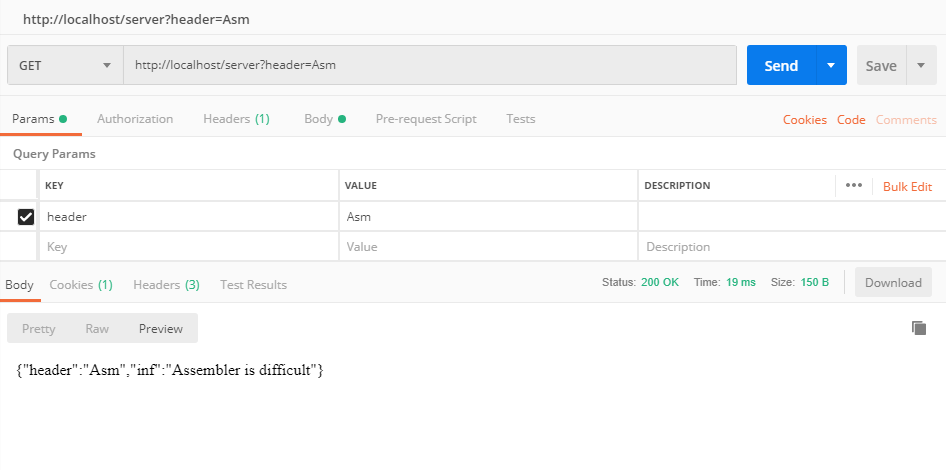


Рисунок 2 – результат выполнения запроса

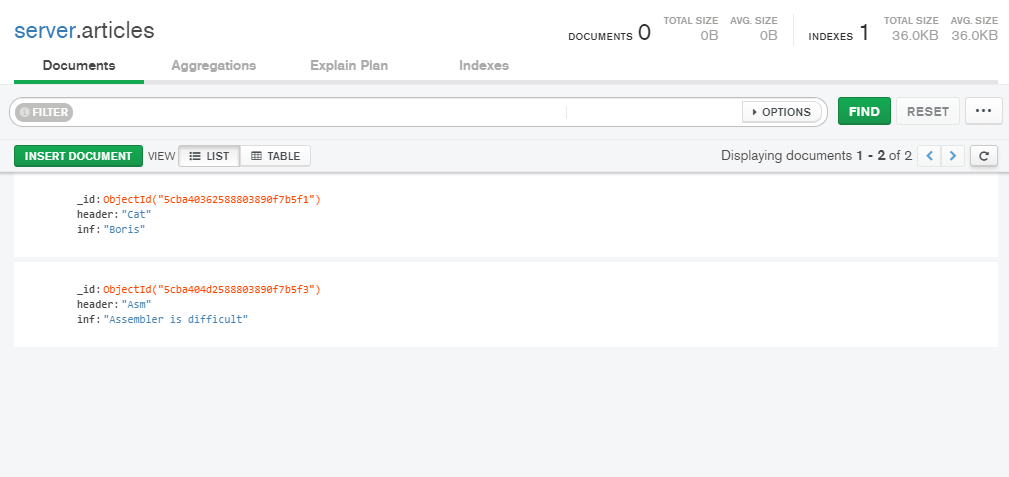


Рисунок 3 – состояние базы данных

6.2 Результат тестирования POST-запроса.

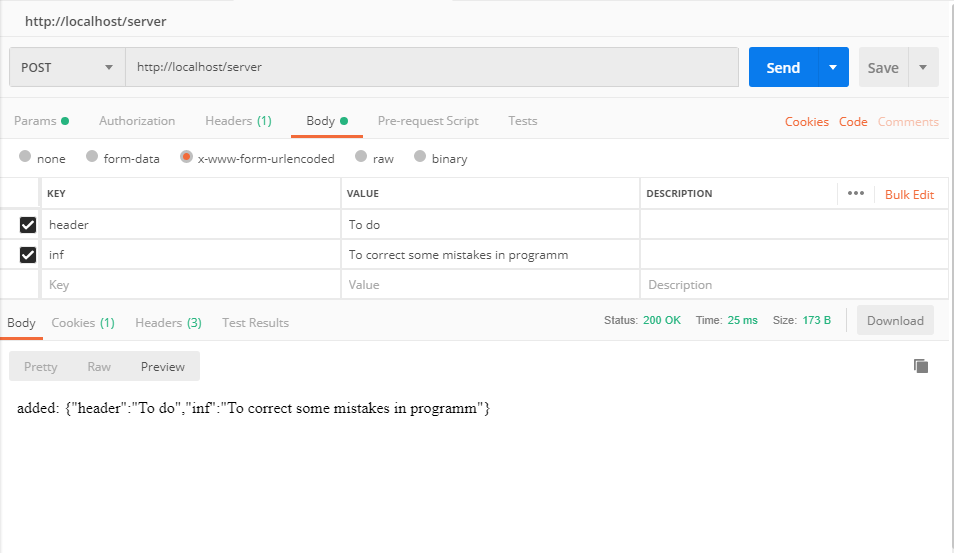


Рисунок 4 – результат выполнения запроса

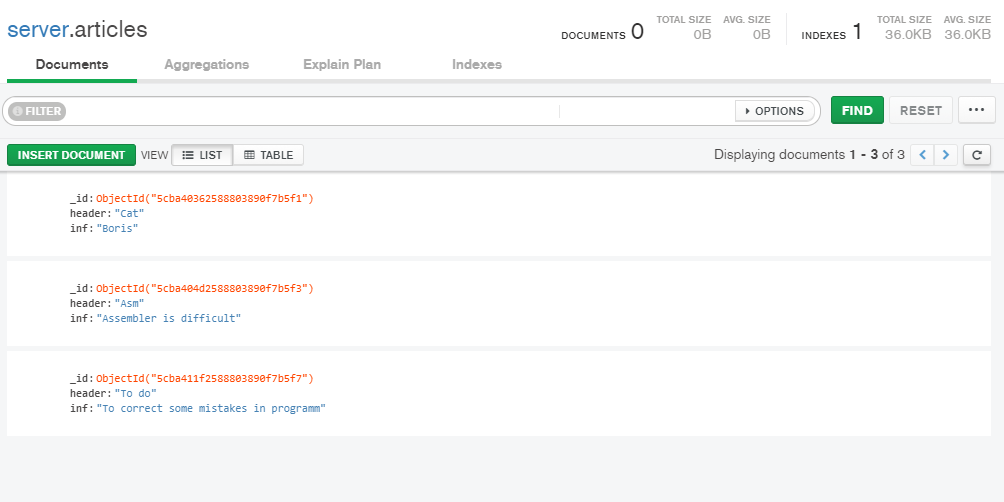


Рисунок 5 – состояние базы данных

6.3 Результат тестирования PUT-запроса.

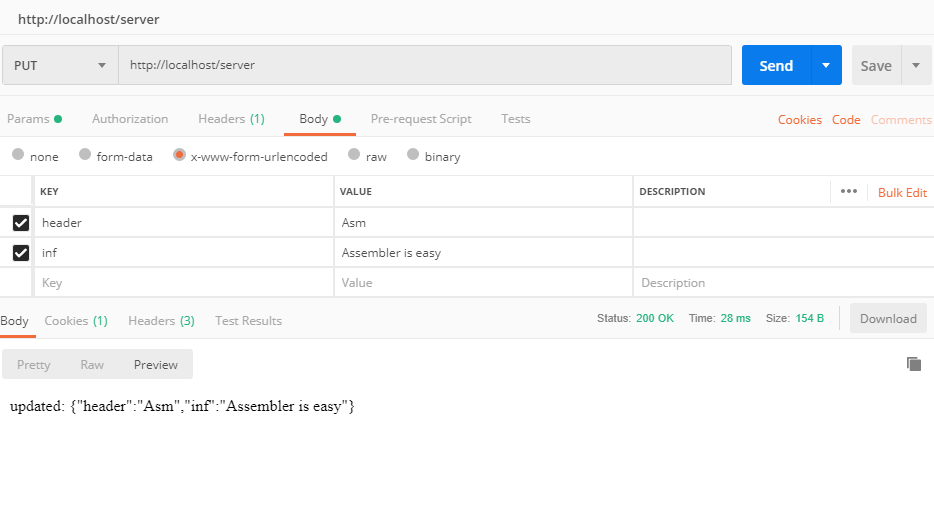


Рисунок 6 – результат выполнения запроса

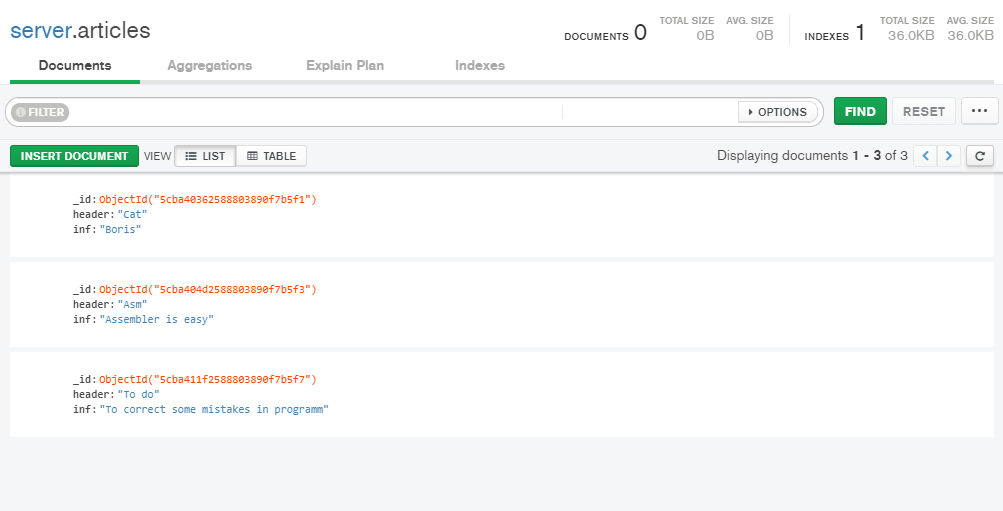


Рисунок 7 – состояние базы данных

6.4 Результат тестирования DELETE-запроса.

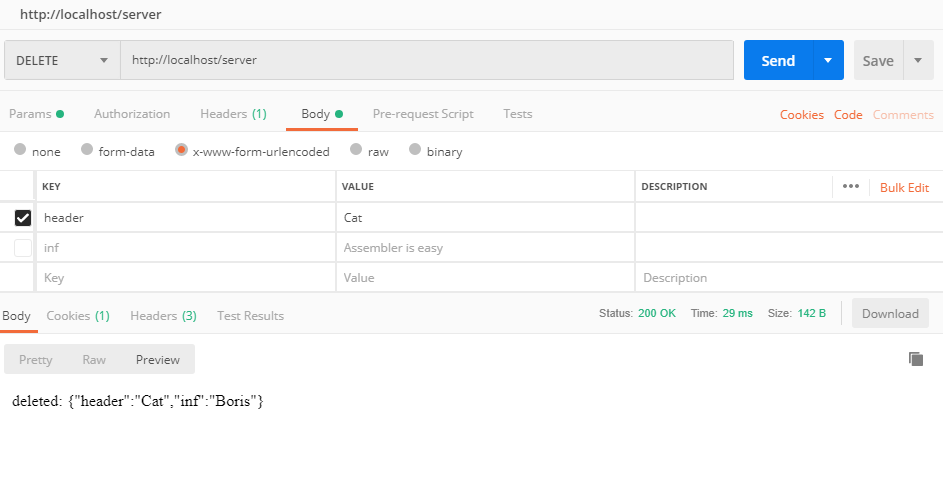


Рисунок 8 – результат выполнения запроса

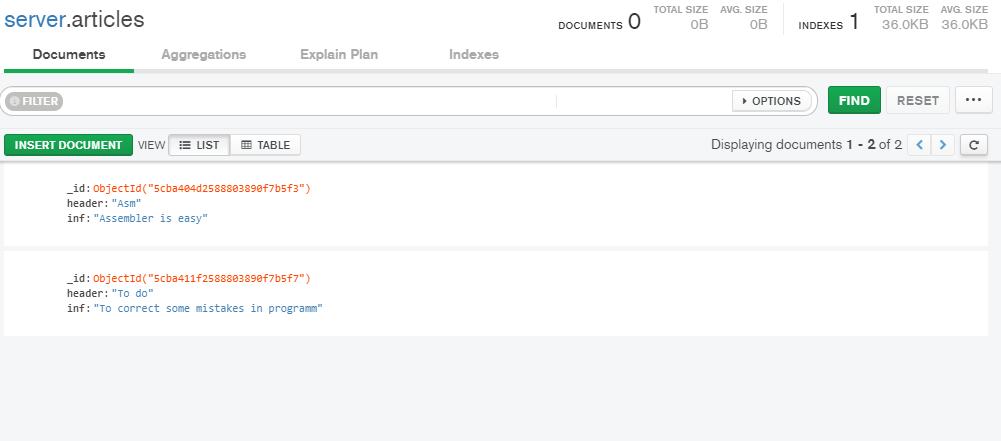


Рисунок 9 – состояние базы данных

# ЗАКЛЮЧЕНИЕ

Во время выполнения курсовой работы были выполнены все поставленные задачи.

Был разработан сервер с помощью сервлетов на языке программирования Java, который по запросу пользователя способен правильно обрабатывать, переданный ему запрос, и внести изменения в базу данных MongoDB.

Данный сервер имеет четыре вида запроса для добавления, удаления, изменения и получение элемента из базы данных. Также сервер возвращает советующие ошибки при некорректном выполнении запроса или сообщения об успешном его выполнении, что позволяет пользователю быть уверенным в работе заданной операции.

Использование современных технологий и методологий разработки программного обеспечения позволило разработать программный продукт, который в дальнейшем возможно будет без особых усилий поддерживать и расширять.

Результаты тестирования показывали корректное выполнение всех поставленных задач, что говорит о том, что данный продукт не только реализует все поставленные перед ним задачи, но и имеет высокую стабильность работы.

Использование объектно-ориентированного подхода программирования дало возможность удобной реализации разработки проекта. Но самым важным преимуществом подхода в данной работе является возможность дальнейшего развития данного продукта.

Применение данного сервера может является основой для разработки различных приложений и сервисов, которым необходимо хранить и обрабатывать данные на удалённом устройстве. Поэтому данный продукт имеет смысл развивать и расширять в дальнейшем.

Подводя итог, был разработан http-сервер, работающий на разных устройствах и корректно работающий с данными ему запросами, однако несмотря на проделанную работу, данный продукт ещё не раскрыл весь свой потенциал. Более детальная проработка запросов, увеличение возможности обработки большего их количества, добавления новых операций работы с базой данных и адаптация приложения для большего количества пользовательских объектов позволит создать мощный компонент архитектуры «клиент-сервер», при использовании которого возможна реализация более крупномасштабных проектов.

СПИСОК ЛИТЕРАТУРЫ

[1] Гордеев А.В., Молчанов А.Ю. Системное программное обеспечение. Учебник для ВУЗов. Спб., Питер, 2003

[2] Таненбаум Э. Современные операционные системы. Спб, Питер, 2004

[3] Шилдт, Герберт Java 8. Руководство для начинающих / Герберт Шилдт. - М.: Вильямс, 2015. - 720 c.

[4] Эккель, Брюс Философия Java / Брюс Эккель. - М.: Питер, 2016. - **809** c.

[5] И.Н. Блинов, В.С. Романчик, Java. Методы программирования, издательство «Четыре четверти», 2013. — 896 с.

[6] Гупта Арун Java EE 7. Основы; Вильямс - М., 2014. - 336 c.

[7] Редько, В.Н. Базы данных и информационные системы / В.Н. Редько, И.А. Бассараб. - М.: Знание, **2004**. **- 240** c.

# **ПРИЛОЖЕНИЕ А**

(обязательное)

Схема структурная

# **ПРИЛОЖЕНИЕ Б**

(обязательное)

Диаграмма классов

# **ПРИЛОЖЕНИЕ В**

(обязательное)

Схема алгоритма