МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий (ИТ)

Кафедра Инструментального и Прикладного Программного Обеспечения (ИиППО)

ПРАКТИЧЕСКАЯ РАБОТА №4

по дисциплине

«Разработка серверных частей интернет ресурсов»

Тема: **«Реализация взаимодействия клиента и сервера с использованием технологии API»**

Студент группы ИКБО-13-20 Лянной А.П.

_____

(подпись студента)

Принял руководитель работы Волков М.Ю.

_____

(подпись руководителя)

Практическая работа выполнена «___»_____2022 г.

Зачтено «___»_____2022 г.

Москва 2022 г.

ОГЛАВЛЕНИЕ

1. Цель работы

Предполагается реализация интерфейса прикладного программирования для доступа к некоторым данным по парадигме REST. Предполагается реализация серверной части обработки запросов и тестирование данного интерфейса с использованием программы Postman. Для реализации данного сервиса предлагается использовать серверную конфигурацию, модернизированную в течение первых трех практических работ. Важной частью данной практической работы является сохранение функциональности реализованной в практической работе №3. То есть интерфейс предлагается создать уже в существующем веб-приложении. Также предполагается использование темы практической работы №3 для продолжения модернизирования собственной системы. Изменение темы согласовывается отдельно с преподавателем. Хранение данных предполагается уже в существующей базе данных.

Технические требования к реализации интерфейса:

1. Доступ как минимум к 2 независимым сущностям.

2. Реализация как минимум операций группы CRUD (создание, чтение, обновление, удаление). Приветствуется реализация дополнительной функциональности.

3. Тестирование всех функциональных возможностей созданного интерфейса с использованием программы Postman.

2. Ход работы

Спецификация API:

Существуют две сущности – users(name, surname) и admins(name, actual, pass).

GET и DELETE запросы к конкретным сущностям выглядят следующим образом: /lapi/сущность?id=номер.

PUT и POST запрос имеет строку запроса /lapi/сущность и JSON тело записи.

Для users:

PUT: {“name”:”имя”,”surname”:”фамилия”}

POST: {“id”:”номер”,“name”:”имя”,”surname”:”фамилия”}

Для admins:

PUT: {“name”:”имя”,”actual”:”пароль”,”pass”:”шифр”}

POST: {“id”:”номер”,“name”:”имя”,”actual”:”пароль”,”pass”:”шифр”}

PUT – вставка новой записи в соответствующую таблицу. PUT запрос включает в себя json вставляемой сущности. Примеры верных запросов:

URI: /lapi/users/

BODY: {“name”:”Andrej”,”surname”:”Lyannoi”}

URI: lapi/admins/

BODY:{“name”:”Andrej”,”actual”:”Parol”,”pass”:”$apr1password”}

GET – получение одной или всех записей из соответствующей таблицы. GET запрос включает в себя id запрошенной сущности, либо пустоту. Примеры верных запросов:

/lapi/users/ - вернёт всех пользователей

/lapi/admins?id=1 – вернёт админа с id = 1

POST – обновление одной записи в соответствующей таблице. POST запрос включает в себя id обновляемой сущности и новый json. Пример верного запроса:

URI: /lapi/users/

BODY: {“id”,”1”,"name":"ayy","surname":"lmao"}

обновит пользователя с id = 1, изменив имя на “ayy”, а фамилию на “lmao”

DELETE – удаление одной или всех записей из соответствующей таблицы. DELETE запрос включает в себя id запрошенной сущности, либо пустоту. Примеры верных запросов:

/lapi/users/ - удалит всех пользователей

/lapi/admins?id=1 – удалит админа с id = 1

Для работы с api следует настроить проксирование и редиректы (листинг 1).

Листинг 1 – Проксирование nginx и редиректы apache

```
//docker-compose.yml

//Модификация default.conf (nginx)

location ~ /lapi.* {
     proxy_pass   http://apache;
}


//Модификация Dockerfile (apache)

RUN a2enmod rewrite

//.htaccess (редирект при api)

RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . api.php
```

API создаётся по шаблону для всех допустимых сущностей (users, admins) и запросов (PUT, GET, POST, DELETE). Приведём для демонстрации реализацию запроса GET для сущности USERS (листинг 2) и пару его успешных тестов в Postman (рисунок 1).

Листинг 2 – Фрагмент api.php

```
<?php
$query = urldecode($_SERVER['REQUEST_URI']);
$len = strlen($query);
$slash = 6;
for ($slash; ($query[$slash] != '/') && ($slash < $len);
++$slash);
$entity = substr($query, 6, $slash - 6);
switch ($_SERVER['REQUEST_METHOD'])
{
...
     case "GET":
     {
          $id = $_GET["id"];
          switch ($entity)
          {
               case "users":
               {
                    if ($id == "")
                    {
                         $mysqli = new mysqli("db", "user",
"password", "appDB");
                         $result = $mysqli->query("SELECT *
FROM users");
                         if ($result->num_rows == 0)
                         {
                              http_response_code(404);
                              echo "API ERROR: Record does not
exist";
                              break;
                         }
                         $json = "";
                         foreach ($result as $row)
                              $json                          =
$json.",".json_encode($row);
                         $json[0] = '[';
                         $json .= "]";
                         echo $json;
                         break;
                    }
                    $mysqli   =   new   mysqli("db",   "user",
"password", "appDB");
                    $result = $mysqli->query("SELECT * FROM
users WHERE ID = ".$id);
                    if ($result->num_rows == 0)
                    {
                         http_response_code(404);
                         echo "API ERROR: Record does not
exist";
                         break;
                    }
```

```
                    foreach ($result as $row)
                         echo json_encode($row);
                    break;
               }
...
```

Результаты вышеописанных действий приведены на рисунках ниже.

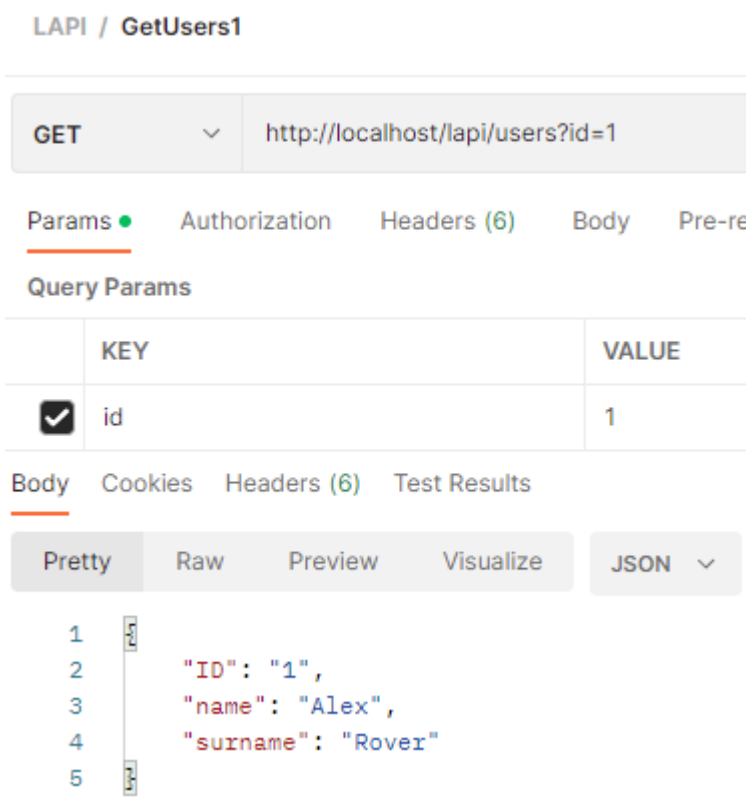

Рисунок 1 – Запросы и их результаты в Postman

3. Вывод

Реализован интерфейс прикладного программирования для доступа к некоторым данным по парадигме REST. Реализована серверная часть обработки запросов и тестирование данного интерфейса с использованием программы Postman.

4. Ответы на вопросы к практической работе

1. Что такое веб-сервер?

По сути, термин «**веб-сервер**» – некое подобие черного ящика. Веб-сервер в процессе своей работы обрабатывает запросы браузера и выдает в ответ веб-страницы. Работа веб-сервера сводится к приему HTTP-запросов от веб-браузеров, выступающих в качестве посредников и выдачи им в ответ соответствующих ответов (как правило это HTTP-ответы, включая HTML-страницы), а также файлы изображений, медиа-потоки, файлы других типов и различные данными.

2. Что такое сервер приложения и чем он отличается от веб-сервера?

Сервер приложения является интерпретатором скриптовых языков, с помощью которых работают написанные на них веб-приложения. Следует особо подчеркнуть, что сервер приложений не работает с протоколом HTTP и не обрабатывает пользовательские запросы. Все эти действия по-прежнему выполняет веб-сервер.

3. Кратко опишите историю развития интернета в рамках развития веб-серверов.

Apache был написан в начале 1995 года и считается, что его имя восходит к шуточному названию «a patchy [server]» (с англ. — «в заплатках»), так как он устранял ошибки популярного тогда сервера Всемирной паутины NCSA HTTPd 1.3. В дальнейшем, с версии 2.x, сервер был переписан заново и теперь не содержит кода NCSA. На данный момент разработка ведётся в ветке 2.4, а в версиях 1.3, 2.0 и 2.2 производятся лишь исправления ошибок безопасности. На текущий момент последняя версия ветки 2.4 — 2.4.46 (5 августа 2020), для первой версии это 1.3.42.

Веб-сервер Apache разрабатывается и поддерживается открытым сообществом разработчиков под эгидой Apache Software Foundation и включён во многие программные продукты, среди которых СУБД Oracle и IBM WebSphere.

С апреля 1996 и до настоящего времени является самым популярным HTTP-сервером в Интернете.

Nginx − веб-сервер и почтовый прокси-сервер. Игорь Сысоев начал разработку в 2002 году. Осенью 2004 года вышел первый публично доступный релиз. С июля 2011 работа над nginx продолжается в рамках компании Nginx. Nginx позиционируется как простой, быстрый и надежный сервер, не перегруженный функциями. Применение nginx целесообразно прежде всего для статических веб-сайтов и как обратного прокси-сервера перед динамическими сайтами.

4. Кратко опишите протокол HTTP.

Структура HTTP-протокола включает в себя сообщение и соединение. HTTP-соединение − это виртуальный канал транспортного уровня, установленный с целью связи, а HTTP-сообщение − это модуль связи, состоящий из структурированный последовательности байтов. Сообщения в свою очередь делятся на HTTP-запросы (request) и на HTTP-ответы (response). Сообщение состоит из 3 основных частей: строки состояния, которая определяет тип сообщения, заголовка сообщения, состоящего из одного и более полей для передачи служебной информации и тела сообщения, которое содержит HTTP-объекты. Под HTTP-объектом понимается метаинформация в форме полей заголовка объекта и содержания в форме тела объекта.

5. Опишите механизм взаимодействия HTTP-сервера, HTTP-клиента и пользователя.

Пользователь посредством HTTP-клиента (чаще всего это браузер) запрашивает у HTTP-сервера некий URL. В процессе работы HTTP-сервера получает запрос клиента, обрабатывает его и либо выдает ему запрашиваемый ресурс, либо сообщает, что это сделать невозможно. После получения всех запрашиваемых ресурсов клиент (браузер) обработает их в соответствии с кодом HTML-документа и выдаст пользователю готовую страницу.

6. Опишите цели и задачи веб-сервера.

Основные цели и задачи такого веб-сервера сводятся к обработке HTTP-запросов клиента и возвращении пользователю результатов этой обработки. Заметим, что Веб-сервер не способен к самостоятельной генерации контента, он может обрабатывать только статическое содержимое. Несмотря на большие возможности современных серверов, они по по-прежнему не способны к самостоятельной генерации контента.

7. Опишите технологию SSI.

SSI (от английского Server Side Includes – включение на стороне сервера) – язык, разработанный для динамического создания и «сборки» веб-страниц на сервере из отдельных составных частей и выдачи клиенту полученного HTML-документа. Использование языка SSI дает возможность в код страницы инкапсулировать содержимое различных файлов. В этом случае, внесение дополнительных повторяющихся элементов, такие, например, как шапка (header), подвал (footer), меню и другие осуществляется в специальные обособленные файлы. Затем созданные таким образом файлы просто подключаются при окончательной сборке страницы.

8. Что такое система управления контентом?

CMS (Content Management System) − система управления содержимым или иначе – система управления контентом. Системы управления контентом представляют собой самостоятельные информационные системы, которые используются для обеспечения и организации совместного процесса создания, редактирования и управления содержимым веб-ресурсов.

9. Верно ли, что сервер приложения умеет работать с протоколом HTTP?

Следует особо подчеркнуть, что сервер приложений не работает с протоколом HTTP и не обрабатывает пользовательские запросы. Все эти действия по-прежнему выполняет веб-сервер.

10. Что такое CGI?

CGI (от английского Common Gateway Interface) или интерфейс общего шлюза, представляет собой стандарт интерфейса, используемого внешней программой для связи с веб-сервером.

11. Как работает система с использованием интерфейс шлюза - CGI?

Для передачи данных используются стандартные потоки ввода-вывода, от веб-сервера к CGI-приложению данные передаются через поток STDIN, принимаются обратно через поток STDOUT, а для передачи сообщений об ошибках используется поток STDERR. Рассмотрим процесс работы такой системы подробнее. Получив запрос от браузера пользователя, веб-сервер определяет, что запрошено динамическое содержимое и формирует специальный запрос, который через интерфейс CGI передается веб-приложению. При его получении приложение запускается и выполняет запрос, результатом которого служит HTML-код динамически сформированной страницы, который передается назад веб-серверу, после чего приложение завершает свою работу.

12. Назовите достоинства и недостатки CGI.

К достоинствам CGI можно отнести языковую и архитектурную независимость: CGI-приложение может быть написано на любом языке и одинаково хорошо работать с любым веб-сервером. Учитывая простоту и открытость стандарта, это привело к активному развитию веб-приложений.

Однако, кроме достоинств, CGI обладает и существенными недостатками. Основной из них − высокие накладные расходы на запуск и остановку процесса, что влечет за собой повышенные требования к аппаратным ресурсам и невысокую производительность. Использование стандартных потоков ввода-вывода ограничивает возможности масштабирования и обеспечения высокой доступности, так как требует, чтобы веб-сервер и сервер приложений находились в пределах одной операционной системы (ОС). В настоящее время интерфейс CGI вытеснен более современными технологиями и практически не встречается на просторах интернета.

13. Что такое FastCGI?

Основной целью разработки данной технологии было повышение производительности CGI. Являясь ее дальнейшим развитием, FastCGI представляет собой клиент-серверный протокол для взаимодействия веб-сервера и сервера приложений, обеспечивающий высокую производительность и безопасность.

14. Назовите основные отличия CGI от FastCGI.

FastCGI устраняет основную проблему CGI − повторный запуск процесса веб-приложения на каждый запрос, FastCGI процессы запущены постоянно, что позволяет существенно экономить время и ресурсы. Для передачи данных вместо стандартных потоков используются UNIX-сокеты или TCP/IP, что позволяет размещать веб-сервер и сервера приложений на разных хостах, таким образом обеспечивая масштабирование и/или высокую доступность системы.

Также мы можем запустить на одном компьютере несколько FastCGI процессов, которые могут обрабатывать запросы параллельно, либо иметь различные настройки или версии скриптового языка. Например, можно одновременно иметь несколько версий PHP для разных сайтов, направляя их запросы разным FastCGI-процессам.

15. Что такое менеджер процессов?

Менеджер процессов является посредником между веб-сервером и серверами приложений. Это несколько усложняет схему, так как настраивать и сопровождать приходится большее количество служб, но в то же время открывает более широкие возможности, позволяя настроить каждый элемент сервера исключительно для решения своих задач.

16. Что такое PHP-FPM?

PHP-FPM первоначально был набором патчей к PHP от Андрея Нигматулина, решавший ряд вопросов управления FastCGI-процессами, начиная с версии 5.3 является частью проекта и входит в поставку PHP. PHP-FPM умеет динамически управлять количеством процессов PHP в зависимости от нагрузки, перезагружать пулы без потери запросов, аварийно перезапускать сбойные процессы и представляет собой менеджер с расширенным функционалом.

17. Что такое Spawn-fcgi?

Spawn-fcgi является частью проекта, но в состав одноименного веб-сервера не входит, по умолчанию Lighttpd использует собственный, более простой, менеджер процессов. Разработчики рекомендуют использовать его в случаях, когда вам нужно управлять FastCGI-процессами, расположенными на другом хосте, либо требуются расширенные настройки безопасности.

18. Что такое Lighttpd?

lighttpd (также «lighty», «лайти») — веб-сервер, разрабатываемый с расчётом на скорость и защищённость, а также соответствие стандартам. Проект lighttpd начался со стремления автора реализовать веб-сервер, который мог бы выдержать одновременно 10 тысяч соединений. lighttpd использует так называемую асинхронную обработку сетевых соединений. Благодаря этому загруженность сервера (в отличие от Apache) при доступе к файлам на диске не зависит от количества текущих соединений.

В lighttpd возможно использование особых системных вызовов для повышения производительности при передаче файлов. При этом задействуются не стандартные системные интерфейсы, а специфичные для платформы вызовы ядра операционной системы, и смена контекста CPU сводится к минимуму.

19. Что такое chroot окружение?

Chroot-окружение – это системный вызов, который временно перемещает root каталог в новую папку. Как правило, root каталог находится в «/». Но при помощи chroot можно задать другой каталог, который будет служить как root-каталог в окружении chroot. Любые приложения, которые запускаются внутри изолированного окружения, в принципе не могут взаимодействовать с остальной операционной системой. Кроме того, нерутовый пользователь (non-root), помещённый в chroot-окружение, не сможет перемещаться по иерархии каталогов.

20. Опишите механизм взаимодействия серверов с использованием FastCGI.

Для управления FastCGI-процессами и распределением нагрузки служат менеджеры процессов, они могут быть как частью веб-сервера, так и отдельными приложениями. Популярные веб-сервера Apache и Lighttpd имеют встроенные менеджеры FastCGI-процессов, в то время как Nginx требует для своей работы с FastCGI внешний менеджер.

5. Ссылка на удалённый репозиторий проекта

https://github.com/Kvadr0n/NOT-JAVA

6. Список использованной литературы

1. nginx: документация [Электронный ресурс] – Режим доступа: https://nginx.org/ru/docs/, свободный;

2. Apache HTTP Server Documentation [Электронный ресурс] – Режим доступа: https://httpd.apache.org/docs/, свободный.