



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

## **ПРАКТИЧЕСКАЯ РАБОТА №3**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Студент группы ИКБО-02-20**

**Тарарина Ольга  
Владимировна**

---

(подпись студента)

**Руководитель практической работы**

**преподаватель Волков М.Ю.**

---

(подпись руководителя)

Работа представлена

«\_\_\_»\_\_\_\_\_ 2022 г.

Допущен к работе

«\_\_\_»\_\_\_\_\_ 2022 г.

Москва 2022

## СОДЕРЖАНИЕ

1. Цель работы .....	3
2. Ход работы.....	3
3. Ответы на вопросы к практической работе.....	9
3.1. Сервер и клиент.....	<b>Ошибка! Закладка не определена.</b>
3.2. База данных.....	<b>Ошибка! Закладка не определена.</b>
3.3. API. ....	<b>Ошибка! Закладка не определена.</b>
3.4. Сервис, отличия от сервера.....	<b>Ошибка! Закладка не определена.</b>
3.5. Архитектура клиент-сервер. ....	<b>Ошибка! Закладка не определена.</b>
3.6. Виды сервисов.....	<b>Ошибка! Закладка не определена.</b>
3.7. Масштабируемость. ....	<b>Ошибка! Закладка не определена.</b>
3.8. Протоколы передачи данных.....	<b>Ошибка! Закладка не определена.</b>
3.9. Тонкий и толстый клиенты. ....	<b>Ошибка! Закладка не определена.</b>
3.10. Паттерн MVC: общие тезисы. ....	<b>Ошибка! Закладка не определена.</b>
3.11. Паттерн MVC: Model-View-Presenter. ....	<b>Ошибка! Закладка не определена.</b>
3.12. Паттерн MVC: Model-View-View Model. ....	<b>Ошибка! Закладка не определена.</b>
3.13. Паттерн MVC: Model-View-Controller.....	<b>Ошибка! Закладка не определена.</b>
3.14. Docker: общие тезисы и определения. ....	<b>Ошибка! Закладка не определена.</b>
3.15. Dockerfile.....	<b>Ошибка! Закладка не определена.</b>
3.16. Docker Compose.....	<b>Ошибка! Закладка не определена.</b>
3.17. LAMP.....	<b>Ошибка! Закладка не определена.</b>
4. Ссылка на удаленный репозиторий проекта .....	19
ЗАКЛЮЧЕНИЕ .....	<b>Ошибка! Закладка не определена.</b>
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ .....	19

## 1. Цель работы

В задании предлагается создать сложную серверную конфигурацию, состоящую из связки apache+nginx+php+База данных. Возможно использование связки apache+php как единый компонент. В данной конфигурации предполагается создание как минимум 3 элементов(контейнеров) или использование как основы серверной конфигурации, созданной в практической работе №1. В этой конфигурации предполагается акселерированное проксирование без кэширования.

Предполагается, что сервер nginx будет отображать статический контент, а apache динамический и в связке мы получим быстроедействие и эффективную систему.

## 2. Ход работы

Для облегчения работы с рекомендуемыми инструментами используются предоставленные скрипт инициализации БД для СУБД MYSQL и скрипт генерации тестовой страницы вместе с оформлением на языке PHP. Данные файлы были помещены в папку “php” и создан файл “Dockerfile” (рис. 2.1).

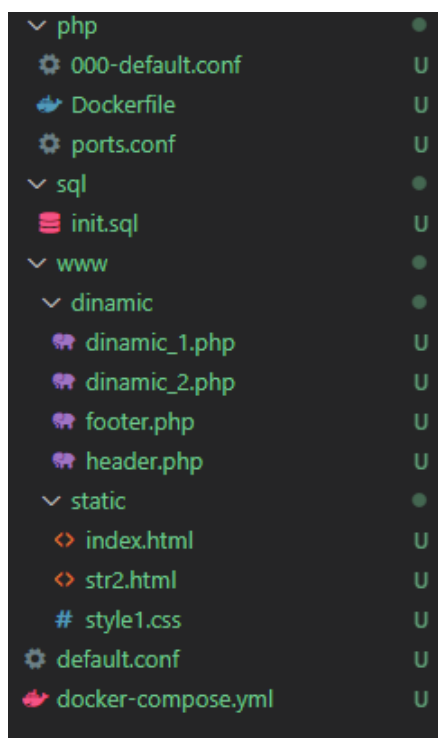


Рис. 2.1 Файловая структура

Для совместной работы контейнера сервера и контейнера базы данных был создан файл “docker-compose.yml” (рис 2.3). Для контейнера базы данных используется готовый образ MariaDB.

```
version: '3' # версия Docker

services: # Контейнеры, которые запускаем

  datab: # Имя контейнера
    image: mariadb:latest # Образ бд
    restart: always # Поведение при крахе
    volumes: # Связываем папку с устройства с папкой на виртуальной машине
      - "./sql:/docker-entrypoint-initdb.d"
    environment:
      MARIADB_ROOT_PASSWORD: password

  php: # Имя контейнера
    build: # Процесс билдинга
      ./php
    hostname: apache.php
    volumes:
      - ./www/dinamic:/var/www/dinamic
      - ./php/ports.conf:/etc/apache2/ports.conf
      - ./php/000-default.conf:/etc/apache2/sites-available/000-default.conf
    depends_on: # Зависимость, после чего запускаем
      - datab

  nginx: # Имя контейнера
    image: nginx:latest
    restart: always # Поведение при крахе
    ports:
      - 8080:8080
    volumes:
      - ./www/static:/usr/share/nginx/
      - ./default.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - datab # запустится после базы данных.
```

Листинг 2.2 – Файл docker-compose.yml

Сделано 2 статических страницы (листинг 2.3 и листинг 2.5)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="author" content="Тарарина Ольга">
  <meta name="description" content="practic2-2">
  <meta name="keywords" content="PCЧП, php, HTML, CSS">
  <link rel="stylesheet" href="style1.css">
  <title>Главная</title>
```

```

</head>
<body>
  <header>
    <a id="logo-ref" href="#"></a>
    <nav>
      <a href="http://localhost:8080/">Главная</a>
      <a href="http://localhost:8080/str2.html">Раздел общий</a>
      <a href="http://localhost:8080/dinamic_1">Раздел 1</a>
      <a href="http://localhost:8080/dinamic_2">Раздел 2</a>
    </nav>
  </header>
  <section>
    <div class="content-container">
      <div class="text-container">
        <h1>Ассортимент</h1>
        <p class="subtitle">
          В небольших хозяйственных магазинах и гипермаркетах товаров
          для дома и ремонта представлены товары как для общестроительных работ, так и для
          декорирования помещений. В зависимости от специализации конкретной торговой сети
          и размера магазина там представлены крепёж, стальные изделия, ручные и
          электроинструменты, абразивы, водопроводные, электрические и сантехнические
          изделия, химические материалы, краски и лаки, моющие средства, садовые товары.
        </p>
      </div>
      <a href="str2.html"></a>
    </div>
  </section>
  <footer>2022© Student MIREA. ALL RIGHTS RESERVED</footer>
</body>
</html>

```

Листинг 2.3 Статическая главная страница



Рис. 2.4 Статическая главная страница

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="author" content="Тарарина Ольга">
  <meta name="description" content="practic2-2">
  <meta name="keywords" content="PCЧП, php, HTML, CSS">
  <link rel="stylesheet" href="style1.css">
  <title>Общее</title>
</head>
<body>
  <header>
    <a id="logo-ref" href="#"></a>
    <nav>
      <a href="http://localhost:8080/">Главная</a>
      <a href="http://localhost:8080/str2.html">Раздел общий</a>
      <a href="http://localhost:8080/dinamic_1">Раздел 1</a>
      <a href="http://localhost:8080/dinamic_2">Раздел 2</a>
    </nav>
  </header>
  <section id="home">
    <a href="http://localhost:8080/dinamic_1"><div class="gradient-
card">Строительные материалы</div></a>
    <a href="http://localhost:8080/dinamic_2"><div class="gradient-
card">Товары для дома и сада</div></a>
  </section>
  <footer>2022© Student MIREA. ALL RIGHTS RESERVED</footer>
</body>
</html>

```

Листинг 2.5 Статический общий раздел

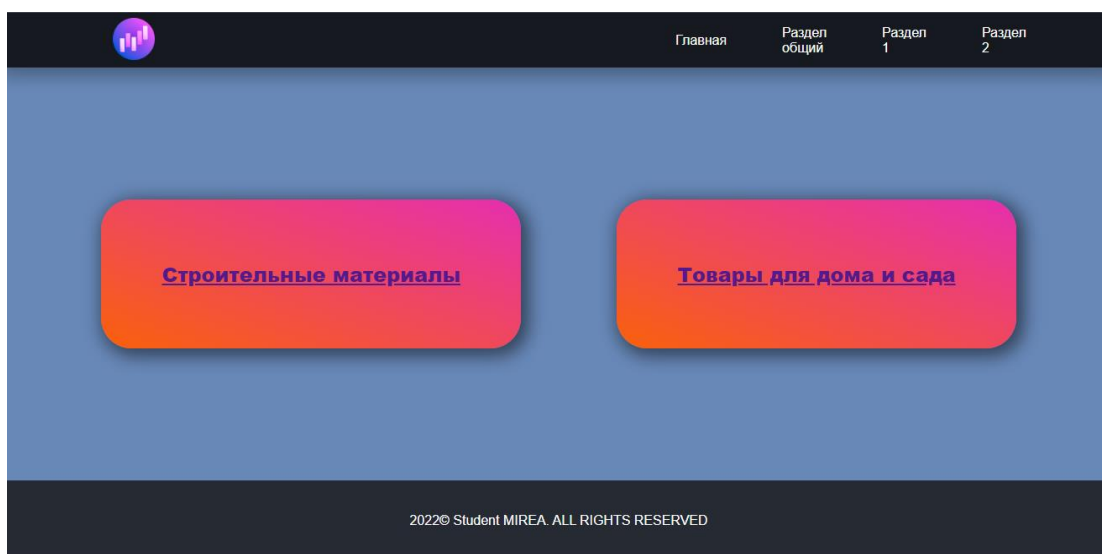


Рис. 2.6 Статический общий раздел

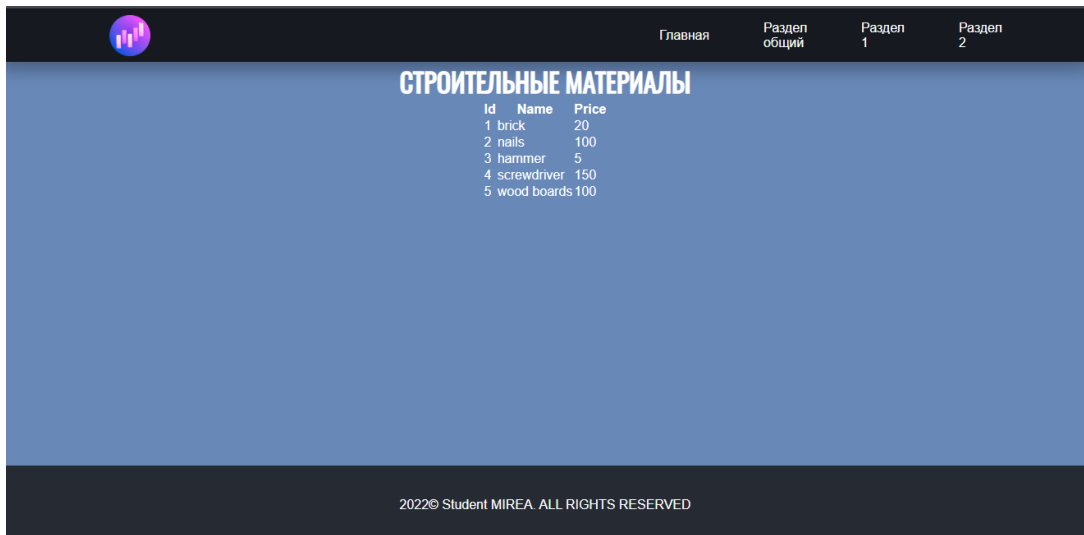
И 2 динамические страницы, берущие таблицы из базы данных(листинг 2.7 и листинг 2.9)

```
<!-- хедер (header.php) -->
<?php
    $title = "Динамическая страница 1";
    include ("header.php");
?>

<h3>Строительные материалы</h3>
<table>
    <tr><th>Id</th><th>Name</th><th>Price</th></tr>
<?php
$mysqli = new mysqli("datab", "user", "password", "appDB");
$result = $mysqli->query("SELECT * FROM materials");
foreach ($result as $row){
    echo
    "<tr><td>{$row['ID']}</td><td>{$row['name']}</td><td>{$row['price']}</td></tr>";
}
?>
</table>

<!-- футер (footer.php) -->
<?php include ("footer.php");?>
```

Листинг 2.7 Динамический 1 раздел



Id	Name	Price
1	brick	20
2	nails	100
3	hammer	5
4	screwdriver	150
5	wood boards	100

Рис. 2.8 Динамический 1 раздел

```
<!-- хедер (header.php) -->
<?php
    $title = "Динамическая страница 2";
    include ("header.php");
?>

<h3>Товары для дома и сада</h3>
<table>
```

```

        <tr><th>Id</th><th>Name</th><th>Price</th></tr>
<?php
$mysqli = new mysqli("datab", "user", "password", "appDB");
$result = $mysqli->query("SELECT * FROM home");
foreach ($result as $row){
    echo
    "<tr><td>{$row['ID']}</td><td>{$row['name']}</td><td>{$row['price']}</td></tr>";
}
?>
</table>

<!-- футер (footer.php) -->
<?php include ("footer.php");?>

```

Листинг 2.9 Динамический 2 раздел

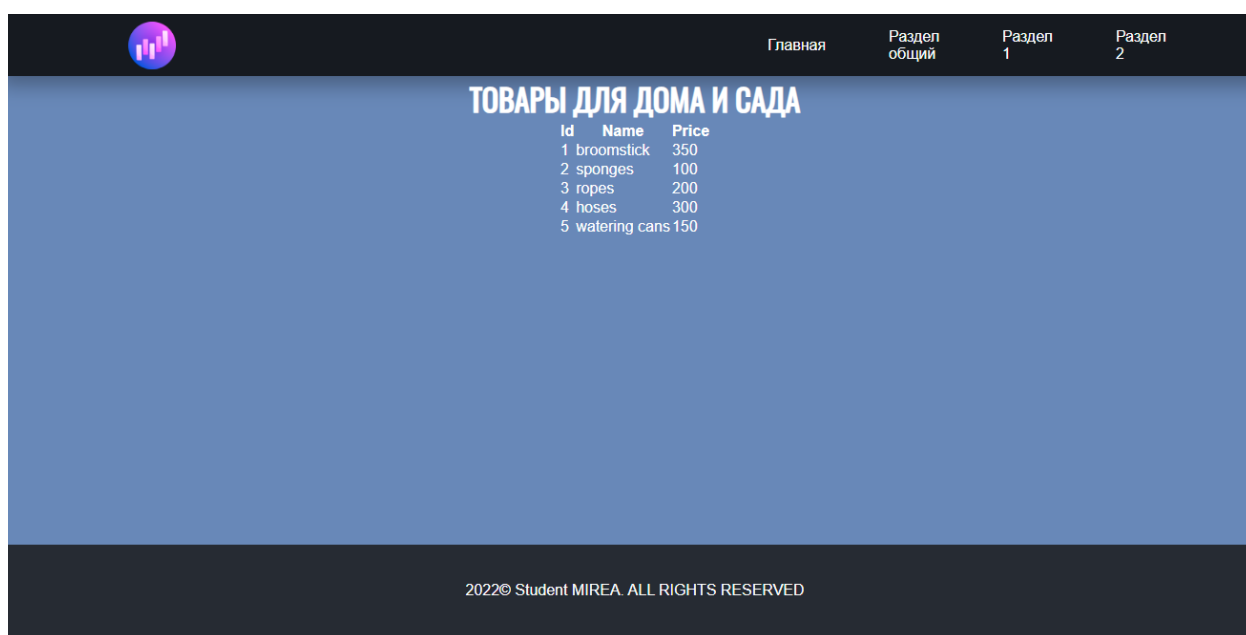


Рис. 2.10 Динамический 2 раздел

## ВЫВОДЫ

Была создана конфигурация серверного программного обеспечения с сервисом для статических и динамических страниц. Это сделано с помощью nginx и apache, где первый обрабатывает входящие от пользователя запросы и является обратным прокси-сервером, который также обрабатывает статические страницы, а второй обрабатывает запросы nginx, когда запрашивается динамическая страница.



### **3. Ответы на вопросы к практической работе**

#### **3.1. 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-запросы от веб-браузеров, выступающих в качестве и выдачи им в ответ соответствующих ответов (как правило это HTTP-ответы, включая HTML-страницы) а также файлы изображений, медиа-потoki, файлы других типов и различные данными. Важно заметить, что, по сути, веб-сервер — это информационная система, так как его функционирование невозможно как без специального программного обеспечения, так и соответствующей аппаратной части, т.е. компьютера.

#### **3.2. Что такое сервер приложения и чем он отличается от веб-сервера?**

Сервер приложений - это программа сервера, работающая в распределенной сети и обеспечивающая среду выполнения для прикладных программ. Сервер приложений представляет собой основной компонент среды выполнения во всех конфигурациях - на нем выполняются приложения.

Основное отличие веб-сервера от сервера приложений заключается в том, что веб-сервер предназначен для обслуживания статических страниц, например HTML и CSS, тогда как сервер приложений отвечает за генерацию динамического содержимого путём выполнения кода на стороне сервера

#### **3.3. Кратко опишите историю развития интернета в рамках развития веб-серверов.**

Традиционно принято считать, что современный интернет начала развиваться в конце 80-ых и начале 90-ых годов. В то время интернет-сайты были просто хранилищем документов, которые имели специальную разметку. В частности, использовался язык разметки SGML. SGML (Standard Generalized

Markup Language) — это стандартный обобщенный язык разметки. Только в 1999 году при смене протокола HTTP/1.1 на более совершенный HTTP/2 произошли некоторые кардинальные изменения, которые были необходимы для развития сети.

В течение продолжительного времени веб-сервер оставался самодостаточным инструментом реализации полноценного сайта. По мере роста сети Интернет, возможности статического HTML перестали удовлетворять интернет-сообщество. В качестве примера можно привести ситуацию, когда добавляются новые страницы. Таким образом даже сайты, созданные в 90-ых годах были ощутимо менее удобными чем современные.

### **3.4. Кратко опишите протокол HTTP.**

Для создания ссылок между документами и ссылок между документом и связанными файлами, был предложен специальный язык гипертекстовой разметки – HTML (HyperText Markup Language). Для доступа к таким документам посредством сети – протокол HTTP (HyperText Transfer Protocol). Не смотря на развитие интернета и модификацию, многие принципы, заложенные в эти языки существенно, не изменились до настоящего времени.

Протокол HTTP реализован по клиент-серверной технологии и работает по принципу запрос-ответ без сохранения состояния. Целью запроса служит некий ресурс, который определяется единым идентификатором ресурса – URI (Uniform Resource Identifier), HTTP использует одну из разновидностей URI – URL (Uniform Resource Locator) – универсальный указатель ресурса, который помимо сведений о ресурсе определяет также его физическое местоположение. В процессе работы HTTP-сервера получает запрос клиента, обрабатывает его и либо выдает ему запрашиваемый ресурс, либо сообщает, что это сделать невозможно.

### **3.5. Опишите механизм взаимодействия HTTP-сервера, HTTP-клиента и пользователя.**

Пользователь посредством HTTP-клиента (чаще всего это браузер) запрашивает у HTTP-сервера некий URL, сервер проверяет и отдает соответствующий этому URL-файл, обычно это HTML-страница. Запрашиваемый документ (в нашем примере это HTML-страница) может, в свою очередь, ссылаться на связанные ресурсы, например, изображения, файлы данных, документы. Для отображения таких связанных ресурсов на странице, клиентом (браузером) последовательно запрашиваются у сервера все связанные документы. После получения всех запрашиваемых ресурсов клиент (браузер) обработает их в соответствии с кодом HTML-документа и выдаст пользователю готовую страницу.

### **3.6. 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-прежнему не способны к самостоятельной генерации контента.

### **3.7. Опишите технологию SSI.**

SSI (от английского Server Side Includes – включение на стороне сервера) – это, как уже было сказано ранее, 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.

### **3.8. Что такое система управления контентом?**

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.

### **3.9. Верно ли, что сервер приложения умеет работать с протоколом HTTP?**

Помимо описанных достоинств, технология SSI дает возможность добавления на веб-страницы дополнительное динамическое содержимое, такое, например, как актуальная дата на странице.

Несмотря на ограничения SSI активно применяют и в настоящее время, в первую очередь из-за простоты и низкой ресурсоемкости создаваемых страниц. SSI, как правило, применяют там, где предполагается вставка статического контента в код.

### **3.10. 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.

### **3.11. 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.

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.

### **3.12. 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).

### **3.13. Что такое FastCGI?**

FastCGI представляет собой клиент-серверный протокол для взаимодействия веб-сервера и сервера приложений, обеспечивающий высокую производительность и безопасность. 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.

### **3.14. Назовите основные отличия CGI от FastCGI.**

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.

### **3.15. Что такое менеджер процессов?**

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.

### **3.16. Что такое PHP-FPM?**

Из внешних менеджеров для FastCGI-процессов применяются PHP-FPM и spawn-fcgi. 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

### **3.17. 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.

### **3.18. Что такое Lighttpd?**

Веб-сервер, библиотека, разрабатываемый с расчётом на скорость и защищённость, а также соответствие стандартам. Это свободное программное обеспечение, распространяемое по лицензии BSD. lighttpd работает в Linux и других Unix-подобных операционных системах, а также в Microsoft Windows.

### **3.19. 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

### **3.20. Опишите механизм взаимодействия серверов с использованием FastCGI.**

В то время как CGI-программы взаимодействуют с сервером через STDIN и STDOUT запущенного CGI-процесса, FastCGI-процессы используют Unix Domain Sockets или TCP/IP для связи с сервером. Это даёт следующее преимущество над обычными CGI-программами: FastCGI-программы могут быть запущены не только на этом же сервере, но и где угодно в сети. Также возможна обработка запросов несколькими FastCGI-процессами, работающими параллельно.

### **3.21. Опишите процесс выбора встроенного или внешнего менеджера процессов.**

Выбирая между встроенным менеджером и внешним, надо оценить ситуацию и выбирать именно тот инструмент, который наиболее подходит запросам. Например, создавая простой сервер для нескольких сайтов на типовых движках применение внешнего менеджера будет явно излишним. Данный подход хорош именно тем, что можно, как из конструктора, собрать именно то, что нужно для решения конкретной задачи.

### **3.22. Что такое интерфейс шлюза?**

Шлюзы позволяют развязать между собой среды веб-сервера и вебприложения, позволяя использовать любые сочетания не обращая внимание на возможную несовместимость. Неважно, поддерживает ли конкретный вебсервер конкретную технологию или скриптовый язык, главное, чтобы он умел работать с нужным типом шлюза.

### **3.23. Что такое SCGI?**

SCGI (Simple Common Gateway Interface) – простой общий интерфейс шлюза – разработан как альтернатива CGI и во многом аналогичен FastCGI, но более прост в реализации. Все, что применимо к FastGCI, справедливо и для SCGI.

### **3.24. Что такое PCGI**

PCGI (Perl Common Gateway Interface) - библиотека Perl для работы с интерфейсом CGI, долгое время являлась основным вариантом работы с Perl-приложениями через CGI, отличается хорошей производительностью при скромных потребностях в ресурсах и неплохой защиты от перегрузки.

### **3.25. Что такое PSGI?**

PSGI (Perl Web Server Gateway Interface) – технология взаимодействия вебсервера и сервера приложений для Perl. Если PCGI представляет собой инструмент для работы с классическим CGI-интерфейсом, то PSGI более напоминает FastCGI. PSGI-сервер представляет среду для выполнения Perl-приложений которая постоянно запущена в качестве службы и может взаимодействовать с вебсервером через TCP/IP или UNIX-сокеты и предоставляет Perl-приложениям те же преимущества, что и FastCGI.

### **3.26. Что такое WSGI?**

WSGI (Web Server Gateway Interface) – предназначен для взаимодействия веб-сервера с сервером приложений для программ, написанных на языке Python.

### **3.27. Опишите механизм взаимодействия серверов Apache и PHP.**

Пользователи Apache не нуждаются в знании этого, но для разработчика программ, понимание модулей и API модуля Apache является необходимым знанием при работе с Apache. Большинство, но не все модули, связаны с различными аспектами обработки HTTP запроса. Преимущество модульного подхода состоит в том, что он позволяет фокусировать модуль на



специфическую задачу, игнорируя при этом другие аспекты HTTP, не касающиеся данной задачи.

### **3.28. Опишите преимущества веб-сервера Apache.**

К преимуществам HTTP сервера Apache, необходимо отнести высокий уровень надежности, гибкие настройки, свободный доступ к программе. В частности, к нему можно подключать большое количество внешних модулей, систем управления базами данных и т.п. Apache поддерживает интернетпротокол IPv6. Также к достоинствам Apache, необходимо отнести регулярно выпускаемые обновления и патчи (заплатки), которые позволяют быстро устранить различные проблемы с работой или безопасностью. Удобство и легкость настройки этого программного обеспечения, делают его одним из самых популярных вариантов как для начинающих, так и для опытных веб-мастеров.

### **3.29. Опишите недостатки веб-сервера Apache.**

Наиболее вескими недостатками являются:

1. Возможные проблемы с производительностью на высоконагруженных сайтах с большим трафиком, «тяжелым» контентом или приложениями, которые требуют высоких вычислительных мощностей.
2. Большое количество параметров настройки может привести к возникновению уязвимостей, из-за невнимательности при конфигурации.
3. Существует вероятность того, что в модули от независимых разработчиков будет внедрен вредоносный код.

### **3.30. Опишите архитектуру веб-сервера Apache.**

В архитектуру Apache входит: простое ядро, платформо-зависимый уровень (APR), и модули. Любое приложение для Apache - даже простейшее, обслуживающее "дефолтную" страницу Apache "It worked" - использует несколько модулей.

### **3.31. Опишите функции ядра веб-сервера Apache.**

Ядро HTTP сервера Apache разрабатывается фондом Apache Software Foundation, который поддерживает огромное количество разработчиков по всему миру. Его основными функциями являются:

1. Передача данных по HTTP.
2. Обработка файлов.
3. Загрузка и поддержка модулей.

Сервер может функционировать без дополнительных модулей, однако в этом случае, его возможности крайне ограничены.

### **3.32. Опишите конфигурацию веб-сервера Apache.**

Конфигурацию Apache, можно разделить на три основных уровня:

1. Конфигурация сервера.
2. Конфигурация виртуального хоста.
3. Конфигурация уровней каталога.

Все настройки осуществляются при помощи специальных текстовых файлов, со своим собственным языком, который основан на блоках директив. С их помощью, могут быть изменены практически все параметры ядра. Большинство модулей, имеет свои собственные параметры. Также они достаточно часто используют для своей работы настроечные файлы операционных систем.

Ряд настроек можно задавать с помощью параметров командной строки.

### **3.33. Что такое URI, URL и чем они различаются.**

URI является либо URL, либо URN, либо одновременно обоими.

URL — это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса. А URN — это URI, который только идентифицирует ресурс в определённом пространстве имён (и, соответственно, в определённом контексте), но не указывает его местонахождение.

#### **4. Ссылка на удаленный репозиторий проекта**

Полный код проекта можно найти по ссылке:

<https://github.com/01ga2001/docker>

#### **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. [Разработка серверных частей интернет-ресурсов](#) [Электронный ресурс].

Дата обращения 23.09.2022

2. [Руководство по РНР](#)

Дата обращения 23.09.2022

3. 3-я лекция по предмету

Дата проведения 08.10.2022