1. Сервер и клиент

Сервер и клиент (рабочая станция) могут иметь одинаковую аппаратную конфигурацию, так как различаются лишь по участию в своей работе человека.

2. База данных

Это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категорировать. Базы данных функционируют под управлением систем управления базами данных (сокращенно СУБД).

3. API

API (Application Programming Interface - прикладной программный интерфейс) - набор функций и подпрограмм, обеспечивающий взаимодействие клиентов и серверов. API (в клиент-сервере) - описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

4. Сервис, отличие от сервера

Сервер (программное обеспечение) - программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определённым ресурсам или услугам.

Сервер (аппаратное обеспечение) - выделенный или специализированный компьютер для выполнения сервисного программного обеспечения без непосредственного участия человека.

Сервис - легко заменяемый компонент сервисно-ориентированной архитектуры со стандартизированными интерфейсами.

5. Архитектура клиент-сервер

Архитектура клиент-сервер — это сетевое окружение, в котором управление данными осуществляется на серверном узле, а другим узлам (клиентам) предоставляется доступ к данным. К достоинствам клиент-серверной архитектуры можно отнести: ● основная нагрузка ложится на сервер(а), в системе имеется одна (или несколько) мощная(ых) серверная(ых) конфигурация часто сложной конструкции и множество “слабых” машин клиентов, что снижает общую стоимость всей системы; ● данные находятся в безопасности, так как сервер дает клиенту только требуемую выборку данных для клиента, основываясь на его уровне доступа; ● разграничение полномочий между клиентом и сервером; ● кроссплатформенность - реализаций клиента может быть сколько угодно: от веб-браузера до приложений мобильных платформ; ● нет дублирования сервисов на каждом клиенте, основная обработка данных лежит на сервере.

6. Виды сервисов

Существует великое множество возможных сервисов, как самостоятельных, так и в составе приложений. Рассмотрим возможные виды сервисов:

● Серверы приложений.

Сервер приложений (англ. application server) — это программная платформа (фреймворк), предназначенная для эффективного исполнения процедур (программ, скриптов), на которых построены приложения.

● Веб-серверы.

Являются подвидом серверов приложений. Изначально предоставляли доступ к гипертекстовым документам по протоколу HTTP. Сейчас поддерживают расширенные возможности, в частности, передачу произвольных данных.

● Серверы баз данных.

Серверы баз данных используются для обработки запросов. На сервере находится СУБД для управления БД и ответов на запросы.

● Файл-серверы.

Файл-сервер хранит информацию в виде файлов и предоставляет пользователям доступ к ней. Как правило, файл-сервер обеспечивает и определенный уровень защиты от несанкционированного доступа.

● Прокси-сервер.

Прокси-сервер (от англ. proxy - представитель, уполномоченный; часто просто прокси, сервер-посредник) - промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника. Существует несколько видов прокси-серверов:

● Веб-прокси — широкий класс прокси-серверов, служащий для кэширования данных.

● Обратный прокси — прокси-сервер, который, в отличие от вебпрокси, ретранслирует запросы клиентов из внешней сети на один или несколько серверов, логически расположенных во внутренней сети. Часто используется для балансировки сетевой нагрузки между несколькими веб-серверами и повышения их безопасности, играя при этом роль межсетевого экрана на прикладном уровне.

● Файрволы (брандмауэры).

Межсетевые экраны, анализирующие и фильтрующие проходящий сетевой трафик, с целью обеспечения безопасности сети.

● Почтовые серверы.

Предоставляют услуги по отправке и получению электронных почтовых сообщений.

7. Масштабируемость

Вертикальная масштабируемость(англ. scaling up) представляет собой увеличение производительности компонентов серверной системы в интересах повышения производительности всей системы. Следует понимать, что система ограничена объемом памяти, скоростными характеристиками и т.д. Данный метод не снимает нагрузку на всю систему, но является самым простым. Ярким примером является увеличение оперативной памяти, установка более мощного процессора.

Горизонтальная масштабируемость(англ. scaling out) представляет собой как разбиение системы на более мелкие структурные компоненты и разнесение их, так и увеличение количества компонентов, параллельно выполняющих одну и ту же функцию. Данный метод часто требует изменение программного обеспечения для налаживания взаимодействия между новыми компонентами. Частым примером является добавление еще одного сервера тех же характеристик к существующему.

8. Протоколы передачи данных

Протокол передачи данных - набор определенных правил или соглашений интерфейса логического уровня, который определяет обмен данными между различными программами. Эти правила задают единообразный способ передачи сообщений и обработки ошибок.

9. Тонкий и толстый клиенты

При классификации компонентов архитектуры клиент-сервер существует понятия “толстый” и “тонкий” клиент. При применении толстого клиента полная функциональность приложения обеспечивается вне зависимости от сервера. В данном случае сервер чаще всего выступает в роли хранилища информации, а вся логика приложения, как и механизм отображения данных располагаются и выполняются на клиенте. Даже при отсутствии соединения с сервером работа ведется с локальными копиями

данных, а при возобновлении соединения происходит синхронизация данных.

Тонким клиентом называют компьютеры и программы, функционирующие в терминальной или серверной сети. Множество задач по обработке данных осуществляются на главных компьютерах, к которым присоединено приложение и компьютер. Тонкий клиент же в отличие от толстого только отображает данные, принятые от сервера. Вся логика приложения выполняется на более производительном сервере, что не требует клиентских мощностей, кроме хорошего и стабильного канала связи. К сожалению, любой сбой на сервере и в канале связи влечет “падение” всего приложения.

10. Паттерн MVC: общие тезисы

Паттерн MVC - фундаментальный паттерн (шаблон), нашедший свое применение повсеместно, не только в веб-разработке. Название паттерну дают первые буквы его основных компонентов: Model View Controller

Существуют разные виды MVC-паттерна, различающихся 3 компонентом системы. Наиболее распространенными видами являются:

- Model-View-Presenter;
- Model-View-View Model;
- Model-View-Controller.

11. Паттерн MVC: Model-View-Presenter

Особенностью паттерна Model-View-Presenter является то, что он позволяет создавать абстракцию представления. Для реализации данного метода выделяется интерфейс представления. А презентер получает ссылку на реализацию интерфейса, подписывается на события представления и по запросу меняет модель.

Признаки подхода с использованием презентера:

● двусторонняя коммуникация с представлением;

● представление взаимодействует напрямую с презентером, путем вызова соответствующих функций или событий экземпляра презентера;

● презентер взаимодействует с View путем использования специального интерфейса, реализованного представлением;

● одному презентеру соотвествует одно отображение.

12. Паттерн MVC: Model-View-View Model

Особенностью паттерна Model-View-View Model является связывание элементов представления со свойствами и событиями View-модели.

Признаками данного подхода являются:

● Двусторонняя коммуникация с представлением.

● View-модель — это абстракция представления. Означает, что свойства представления совпадают со свойствами View-модели / модели.

● View-модель не имеет ссылки на интерфейс представления (IView). Изменение состояния View-модели автоматически изменяет представление и наоборот, поскольку используется механизм связывания данных (Bindings).

● Одному экземпляру View-модели соответствует одно отображение.

13. Паттерн MVC: Model-View-Controller

Особенностью паттерна Model-View-Controller является то, что контроллер и представление зависят от модели, но при этом сама модель не зависит от двух других компонентов.

Признаками данного подхода являются:

● Контроллер определяет, какое представление должно быть отображено в требуемый момент. Если рассматривать применение для разработки вебприложений, то контроллер управляет запросами пользователя. Его основная функция — вызывать и координировать действие

необходимых ресурсов и объектов, нужных для выполнения действий, задаваемых пользователем. Обычно контроллер вызывает соответствующую модель для задачи и выбирает подходящий вид.

● События представления могут повлиять только на контроллер. Контроллер может повлиять на модель и определить другое представление.

● Возможно несколько представлений только для одного контроллера. Данный вариант чаще всего используется при разработке веб-приложений.

14. Docker: общие тезисы и определения

Docker — это платформа, которая позволяет упаковать в контейнер приложение со всем окружением и зависимостями, а затем доставить и запустить его в целевой системе.

Приложение, упакованное в контейнер, изолируется от операционной системы и других приложений. Поэтому разработчики могут не задумываться, в каком окружении будет работать их приложение, а инженеры по эксплуатации — единообразно запускать приложения и меньше заботиться о системных зависимостях.

В поставку Docker входят следующие компоненты:

Docker host — это операционная система, на которую устанавливают Docker и на которой он работает.

Docker daemon — служба, которая управляет Docker-объектами: сетями, хранилищами, образами и контейнерами.

Docker client — консольный клиент, при помощи которого пользователи взаимодействуют с Docker daemon и отправляют ему команды, создают контейнеры и управляют ими.

Docker image — это неизменяемый образ, из которого разворачивается контейнер.

Docker container — развёрнутое и запущенное приложение.

Docker Registry — репозиторий, в котором хранятся образы.

Dockerfile — файл-инструкция для сборки образа.

Docker Compose — инструмент для управления несколькими контейнерами. Он позволяет создавать контейнеры и задавать их конфигурацию.

Docker Desktop — GUI-клиент, который распространяется по GPL. Бесплатная версия работает на Windows, macOS, а с недавних пор и на Linux. Это очень удобный клиент, который отображает все сущности Docker и позволяет запустить однонодовый Kubernetes для компьютера.

15. Dockerfile

Dockerfile — это текстовый файл с инструкциями, необходимыми для создания образа контейнера. Эти инструкции включают идентификацию существующего образа, используемого в качестве основы, команды, выполняемые в процессе создания образа, и команду, которая будет выполняться при развертывании новых экземпляров этого образа контейнера.

16. Docker Compose

Docker-compose — это плагин для докера, который позволяет запускать множество контейнеров одновременно и маршрутизировать потоки данных между ними.

17. LAMP

Docker-LAMP - это набор образов docker, которые включают базовое изображение phusion, а также стек LAMP (Apache, MySQL и PHP) в одном удобном пакете.

18. Конфигурационный файл php.ini

Файл php.ini это конфигурационный файл с подавляющим большинством настроек PHP. Файл php.ini загружается вместе с локальным сервером или загружается вместе с дистрибутивом PHP при самостоятельной сборке локального сервера.

19.Как написать простой скрипт на php

```
<?php
    echo “Hello World!”;
>
```

20.Основные правила, связанные с переменными в php

Переменные в PHP представлены знаком доллара с последующим именем переменной. Имя переменной чувствительно к регистру. Правильное имя переменной должно начинаться с буквы или символа подчёркивания и состоять из букв, цифр и символов подчёркивания в любом количестве.

21.Основные типы данных в php

В PHP есть десять базовых типов данных:

- bool (логический тип)
- int (целые числа)
- float (дробные числа)
- string (строки)
- array (массивы)
- object (объекты)
- callable (функции)
- mixed (любой тип)
- resource (ресурсы)
- null (отсутствие значения)

22.Какие существуют функции для работы с переменными в php вне зависимости от типа данных

Функции, определяемые пользователем, встроенные функции, анонимные функции, стрелочные функции, Callback-функции.

23.Предопределенные переменные в php

- Суперглобальные переменные — Встроенные переменные, которые всегда доступны во всех областях

- $GLOBALS — Ссылки на все переменные глобальной области видимости
- $_SERVER — Информация о сервере и среде исполнения
- $_GET — Переменные HTTP GET
- $_POST — Переменные HTTP POST
- $_FILES — Переменные файлов, загруженных по HTTP
- $_REQUEST — Переменные HTTP-запроса
- $_SESSION — Переменные сессии
- $_ENV — Переменные окружения
- $_COOKIE — HTTP Cookies
- $php_errormsg — Предыдущее сообщение об ошибке
- $http_response_header — Заголовки ответов HTTP
- $argc — Количество аргументов, переданных скрипту
- $argv — Массив переданных скрипту аргументов

24.Переменные переменных в php

Иногда бывает удобно иметь переменными имена переменных. То есть, имя переменной, которое может быть определено и изменено динамически. Переменная переменной берет значение переменной и рассматривает его как имя переменной.

25.Выражения в php

Выражения - это самые важные строительные элементы PHP. Почти всё, что вы пишете в PHP, является выражением. Самое простое и точное определение выражения - "все что угодно, имеющее значение". Основными

формами выражений являются константы и переменные. Точка с запятой обозначает конец выражения.

26.Арифметические операторы в php

Арифметические операции

Пример	Название	Результат
+$a	Идентичность	Конвертация $a в int или float, что более подходит.
-$a	Отрицание	Смена знака $a.
$a + $b	Сложение	Сумма $a и $b.
$a - $b	Вычитание	Разность $a и $b.
$a * $b	Умножение	Произведение $a и $b.
$a / $b	Деление	Частное от деления $a на $b.
$a % $b	Деление по модулю	Целочисленный остаток от деления $a на $b.
$a ** $b	Возведение в степень	Возведение $a в степень $b.

27. Битовые операции в php

Побитовые операторы

Пример	Название	Результат
$a & $b	И	Устанавливаются только те биты, которые установлены и в $a, и в $b.
$a | $b	Или	Устанавливаются те биты, которые установлены в $a или в $b.
$a ^ $b	Исключающее или	Устанавливаются только те биты, которые установлены либо только в $a, либо только в $b, но не в обоих одновременно.
~ $a	Отрицание	Устанавливаются те биты, которые не установлены в $a, и наоборот.
$a << $b	Сдвиг влево	Все биты переменной $a сдвигаются на $b позиций влево (каждая позиция подразумевает "умножение на 2")
$a >> $b	Сдвиг вправо	Все биты переменной $a сдвигаются на $b позиций вправо (каждая позиция подразумевает "деление на 2")

28. Оператор присваивания в php

Базовый оператор присваивания обозначается как "=". Оператор присваивания означает, что левый операнд получает значение правого выражения, (то есть устанавливается значением). Результатом выполнения оператора присваивания является само присвоенное значение.

29. Операторы сравнения в php

Операторы сравнения:

- Оператор == (равенства)
- Оператор === (идентичности)
- Оператор != или <> (неравенства)
- Оператор !== (неидентичности)
- Оператор > (больше)
- Оператор < (меньше)
- Оператор >= (больше или равно)
- Оператор <= (меньше или равно)

30. Логические операторы в php

Пример	Название	Результат
$a and $b	Логическое И	TRUE, если оба значения и $a, и $b равны TRUE.
$a or $b	Логическое ИЛИ	TRUE, если оба значения равны TRUE или хотя бы одно из значений $a или $b равно TRUE.
$a xor $b	Исключающее ИЛИ	TRUE, если одно из значений $a или $b равно TRUE, но не оба сразу.
! $a	Логическое НЕ	TRUE, если значение $a не равно TRUE.
$a && $b	Логическое И	TRUE, если оба значения и $a, и $b равны TRUE.
$a || $b	Логическое ИЛИ	TRUE, если оба значения равны TRUE или хотя бы одно из значений $a или $b равно TRUE.

31. Условная конструкция в php

Конструкция if (условие) проверяет истинность некоторого условия, и если оно окажется истинным, то выполняется блок выражений, стоящих после if. Если же условие ложно, то есть равно false, тогда блок if не выполняется. Блок else содержит инструкции, которые выполняются, если условие после if ложно, то есть равно false. Конструкция elseif вводит дополнительные условия в программу.

```php
$a = 5;
if($a>0){
    echo "Переменная а больше нуля";
}
elseif($a < 0){
    echo "Переменная а меньше нуля";
}
else{
    echo "Переменная а равна нулю";
}
```

32. Циклы в php

PHP поддерживает три вида циклов:

- Цикл с предусловием (while)
- Цикл с постусловием (do-while)
- Цикл со счетчиком (for)
- Специальный цикл перебора массивов (foreach)

33. Конструкции switch и match в php

```php
<?php
// Оператор switch:

switch ($i) {
    case 0:
        echo "i равно 0";
        break;
    case 1:
        echo "i равно 1";
        break;
    case 2:
        echo "i равно 2";
        break;
}
```

```php
<?php
$food = 'cake';
$return_value = match ($food) {
    'apple' => 'На столе лежит яблоко',
    'banana' => 'На столе лежит банан',
    'cake' => 'На столе стоит торт',
};
```

34. Include и require в php

Конструкция include предназначена для включения файлов в код сценария PHP во время исполнения сценария PHP. В отличие от конструкции require конструкция include позволяет включать файлы в код PHP скрипта во время выполнения сценария. Конструкция require позволяет включать файлы в PHP сценарий до выполнения сценария PHP.

35. Функции в php

Функции в php:

- Функции, определяемые пользователем
- Встроенные функции
- Анонимные функции

- Стрелочные функции
- Callback-функции как объекты первого класса

36. Что такое веб-сервер?

Веб-сервер — это сервер, принимающий HTTP-запросы от клиентов, обычно веб браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными. Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер.

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

Сервер приложений — это программная платформа, предназначенная для эффективного исполнения процедур (программ, скриптов), на которых построены приложения. Сервер приложений действует как набор компонентов, доступных разработчику программного обеспечения через API, определённый самой платформой.

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 путем выполнения кода на стороне сервера, например, JSP, EJB и т. п.

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

От статических web-страниц, трудных в изменении, мы пришли в web-серверам, способным работать с динамическими данными.

39. Кратко опишите протокол HTTP.

HTTP — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

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

Клиент инициирует взаимодействие с сервером и посылает запрос, содержащий: метод доступа, адрес URI (Uniform Resource Identifier, универсальный идентификатор ресурса), версию протокола, сообщение с информацией о типе передаваемых данных, информацией о клиенте, пославшем запрос, и, возможно, с содержательной частью (телом) сообщения.

Ответ сервера содержит: строку состояния, в которую входит версия протокола и код возврата (успех или ошибка), сообщение, в которое входит информация сервера, метаинформация (т.е. информация о содержании сообщения) и его тело.

41. Опишите цели и задачи веб-сервера.

Основная цель и задача веб-сервера - обработка HTTP-запросов и возврат пользователю их результатов.

42. Опишите технологию SSI.

SSI (Server Side Includes — включения на стороне сервера) - язык для динамической «сборки» веб-страниц на сервере из отдельных составных частей и выдачи клиенту полученного HTML документа.

43. Что такое система управления контентом?

Система управления содержимым — информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления содержимым, иначе — контентом.

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

Сервер приложений не умеет работать с протоколом HTTP и обрабатывать пользовательские запросы, так как это задача веб-сервера. Чтобы обеспечить их взаимодействие был разработан общий интерфейс шлюза - CGI (Common Gateway Interface).

45. Что такое CGI?

CGI — стандарт интерфейса, используемого внешней программой для связи с веб-сервером.

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

Обобщенный алгоритм работы через CGI можно представить в следующем виде: клиент запрашивает CGI-приложение по его URI; Веб-сервер принимает запрос и устанавливает переменные окружения, через них приложению передаются данные и служебная информация; Веб-сервер перенаправляет запросы через стандартный поток ввода (stdin) на вход вызываемой программы; CGI-приложение выполняет все необходимые операции и формирует результаты в виде HTML; Сформированный гипертекст возвращается веб-серверу через стандартный поток вывода

(stdout). Сообщения об ошибках передаются через stderr; Веб-сервер передает результаты запроса клиенту.

47. Назовите достоинства и недостатки CGI.

Достоинства: процесс CGI скрипта не зависит от Веб-сервера и, в случае падения, никак не отразится на работе последнего; может быть написан на любом языке программирования; поддерживается большинством Веб-серверов.

Недостатки: самым большим недостатком этой технологии являются повышенные требования к производительности веб-сервера. Дело в том, что каждое обращение к CGI-приложению вызывает порождение нового процесса, со всеми вытекающими отсюда накладными расходами. Если же приложение написано с ошибками, то возможна ситуация, когда оно, например, зациклится. Браузер прервет соединение по истечении тайм-аута, но на серверной стороне процесс будет продолжаться, пока администратор не снимет его принудительно.

48. Что такое FastCGI?

Интерфейс FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения, дальнейшее развитие технологии CGI.

49. Назовите основные отличия CGI от FastCGI.

В отличие от CGI, в FastCGI для каждого скрипта не запускается отдельный процесс, благодаря чему меньше расходуются ресурсы.

50. Что такое менеджер процессов?

Менеджер процессов отвечает за создание новых процессов в системе и за управление основными ресурсами, связанными с процессом. Все эти услуги предоставляются посредством сообщений. Так, например, если процесс хочет породить новый процесс, он делает это, посылая сообщение с указанием атрибутов создаваемого процесса. Обратите внимание, что т.к. сообщения передаются по сети, вы можете легко создать процесс на другом узле сети, послав сообщение Менеджеру процессов на этом узле

51. Что такое PHP-FPM?

php-fpm — это fastcgi process manager. Он представляет из себя отдельную службу, которая работает независимо от какого-либо веб сервера. Он может сам принимать запросы от веб сервера через unix-сокет или через сетевое соединение. Другими словами, вы можете держать сайты на одном сервере, а php-скрипты исполнять на другом.

52. Что такое Spawn-fcgi?

Spawn-fcgi используется для запуска удаленных и локальных FastCGI процессов.

53. Что такое Lighttpd?

lighttpd — веб-сервер, разрабатываемый с расчетом на скорость и защищенность, 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.

54. Что такое chroot окружение?

Chroot окружение — способ запуска программ, системный вызов и просто команда, позволяющая изменить корневой каталог в системе.

55. 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.

56. Опишите процесс выбора встроенного или внешнего менеджера процессов.

Надежнее встроенный менеджер.

57. Что такое интерфейс шлюза?

Аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной).

58. 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.

59. 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.

60. Что такое PSGI?

PSGI (Perl Web Server Gateway Interface) — технология взаимодействия веб-сервера 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 веб-сервером через TCP/IP или UNIX-сокеты и предоставляет Perl приложениям те же преимущества, что и FastCGI.

61. Что такое WSGI?

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

62. Опишите механизм взаимодействия серверов Apache и PHP.

Apache обычно обслуживает файлы, извлекая файл и отправляя поток вниз по HTTP-соединению. Однако с PHP Apache извлекает файл, передает его в двоичный файл PHP и отправляет выход поток из команды вниз по HTTP-соединению.

63. Опишите преимущества веб-сервера Apache.

• Высокий уровень надежности

• Гибкие настройки

• Свободный доступ к программе

• Регулярные обновления и патчи

• Удобство и легкость настройки

64. Опишите недостатки веб-сервера Apache.

• Проблемы с производительностью на высоконагруженных сайтах

• Большое кол-во параметров настройки может привести к уязвимости в конфигурации

• Некоторая вероятность наличия вредоносного кода в модулях от независимых разработчиков

65. Опишите архитектуру веб-сервера Apache.

Ядро Apache включает в себя основные функциональные возможности, такие как обработка конфигурационных файлов, протокол HTTP и система загрузки модулей.

Система конфигурации Apache, основанная на текстовых конфигурационных файлах.

Apache имеет встроенный механизм виртуальных хостов. Он позволяет полноценно обслуживать на одном IP-адресе множество сайтов (доменных имён), отображая для каждого из них собственное содержимое.

66. Опишите функции ядра веб-сервера Apache.

Основные функции ядра:

• Передача данных по HTTP

• Обработка файлов

• Загрузка и поддержка модулей

67. Опишите конфигурацию веб-сервера Apache.

Конфигурацию Apache можно разделить на три основных уровня:

• Конфигурация сервера

• Конфигурация виртуального хоста

• Конфигурация уровней каталога

68. Что такое URI, URL и чем они различаются.

URI – имя и адрес ресурса в сети, включает в себя URL и URN

URL – адрес ресурса в сети, определяет местонахождение и способ обращения к нему

URN – имя ресурса в сети, определяет только название ресурса, но не говорит как к нему подключиться