Оглавление

Ответы на вопросы к парктической работе 1

1. Сервер и клиент.

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.

Клиент — это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу.

2. База данных.

База данных — это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категорировать.

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, отличия от сервера.

Сервис - легко заменяемый компонент сервисно-ориентированной архитектуры со стандартизированными интерфейсами - это технология для взаимодействия между системами.

Веб-сервис(веб-служба) - идентифицируемая уникальным веб-адресом

(URL-адресом) программная система со стандартизированными интерфейсами — это сервер реализующий http протокол.

5. Архитектура клиент-сервер.

Данная модель — это идея разделения системы или приложения на отдельные задачи, размещаемые на различных платформах для большей эффективности.

Клиент представляет собой программу представления данных, которая для их получения посылает запросы серверу, который в свою очередь может делать запрос к базе данных, обрабатывает данные и возвращает их к клиенту. Возможны случаи разделение обработки данных, когда часть работы сервера в виде обработки данных выполняет клиент.

6. Виды сервисов.

Виды сервисов:

- Сервер приложений (англ. application server) — это программная платформа (фреймворк), предназначенная для эффективного исполнения процедур (программ, скриптов), на которых построены приложения.
- Веб-серверы - подвид серверов приложений.
- Серверы баз данных используются для обработки запросов (СУБД).
- Файл-сервер хранит информацию в виде файлов и предоставляет пользователям доступ к ней.
- Прокси-сервер - промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника (Веб-прокси служащий для кэширования данных. , Обратный прокси - межсетевой экран).
- Файрволы (брандмауэры).
- Почтовые серверы.

7. Масштабируемость.

Масштабируемость - способность работать с увеличенной нагрузкой путем наращивания ресурсов без фундаментальной перестройки архитектуры и/или модели реализации при добавлении ресурсов (вертикальная, горизонтальная).

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.

В общей классификации протоколы делятся на низкоуровневые, протоколы верхнего уровня и протоколы промежуточного уровня.

Распределение протоколов по уровням модели OSI

	TCP/IP	OSI	
7	Прикладной	Прикладной	HTTP, SMTP, SNMP, FTP, Telnet, SSH, SCP, SMB, NFS, RTSP, BGP
6		Представления	TLS, SSL
5		Сеансовый	RPC, NetBIOS, PPTP, L2TP
4	Транспортный	Транспортный	TCP, UDP, GRE
3	Сетевой	Сетевой	IP, ICMP, IGMP, OSPF, RIP, IPX
2	Канальный	Канальный	Ethernet, Token ring, HDLC, PPP, X.25, Frame relay, ISDN

9. Тонкий и толстый клиенты.

При применении толстого клиента полная функциональность приложения обеспечивается вне зависимости от сервера. Тонкий клиент же в отличие от толстого только отображает данные, принятые от сервера.

10. Паттерн MVC: общие тезисы.

Первая часть данного паттерна это модель (Model). Это представление содержания функциональной бизнес-логики приложения, не зависит от остальных частей продукта.

Представление (View) это есть отображение данных, получаемых от модели. Никакого влияния на модель представление оказать не может.

Третьим компонентом системы является контроллер. Данный компонент является неким буфером между моделью и представлением. Обобщенно он управляет представлением на основе изменения модели.

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

12. Паттерн MVC: Model-View-View Model.

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.

14. Docker: общие тезисы и определения.

Подобно виртуальной машине докер запускает свои процессы в собственной, заранее настроенной операционной системе. Но при этом все процессы докера работают на физическом host-сервере, деля все процессоры и всю доступную память со всеми другими процессами, запущенными в host-системе. Подход, используемый Docker, находится посередине между запуском всего на физическом сервере и полной виртуализацией, предлагаемой виртуальными машинами. Этот подход называется контейнеризацией.

15. Dockerfile.

Чтобы создавать свои собственные образы нужен специальный скрипт

16. Docker Compose.

Когда идет работа с несколькими контейнерами, то требуется механизм их объединения и оркестровки. Таким инструментом является Docker Compose. Это средство для решения задач развертывания проектов. Docker Compose используется для одновременного управления несколькими контейнерами, входящими в состав приложения.

17. LAMP.

Для полноценной работоспособности конфигурации нужны: операционная система, Веб-сервер, язык программирования и База данных. Из всего этого следует идея технологии LAMP — акроним, обозначающий набор (комплекс) серверного программного обеспечения, широко используемый в интернете. LAMP назван по первым буквам входящих в его состав компонентов:

- Linux — операционная система Linux;

- Apache — веб-сервер;
- MariaDB / MySQL — СУБД;
- PHP — язык программирования, используемый для создания веб-приложений (помимо PHP могут подразумеваться другие языки, такие как Perl и Python).

Ответы на вопросы к практической работе 2

- ***Конфигурационный файл php.ini***

Файл конфигурации (php.ini) считывается при запуске PHP. Для версий серверных модулей PHP это происходит только один раз при запуске веб-сервера. Для CGI и CLI версий это происходит при каждом вызове.

- ***Как написать простой скрипт на php.***

Необходимо создать файл с расширением *.php* в корневом каталоге веб-сервера (*DOCUMENT_ROOT*) и запишите в него классическую стркутуру HTML. Для того чтобы использовать PHP необходимо заключить код внутри тега <?PHP?>.

- ***Основные правила, связанные с переменными в php.***

Правила для переменных PHP:

- Переменная начинается с знака $, за которым следует имя переменной
- Имя переменной должно начинаться с буквы или символа подчеркивания
- Имя переменной не может начинаться с числа
- Имя переменной может содержать только буквенно-цифровые символы и знаки подчеркивания (a-z, 0-9 и _)

4. 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)

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

- boolval — Возвращает логическое значение переменной
- debug_zval_dump — Выводит строковое представление внутренней структуры zval
- doubleval — Псевдоним floatval
- empty — Проверяет, пуста ли переменная
- floatval — Возвращает значение переменной в виде числа с плавающей точкой
- get_debug_type — Возвращает имя типа переменной в виде, подходящем для отладки
- get_defined_vars — Возвращает массив всех определённых переменных
- get_resource_id — Возвращает целочисленный идентификатор для данного ресурса
- get_resource_type — Возвращает тип ресурса
- gettype — Возвращает тип переменной
- intval — Возвращает целое значение переменной
- is_array — Определяет, является ли переменная массивом
- is_bool — Проверяет, является ли переменная булевой
- is_callable — Проверяет, что значение может быть вызвано как функция в текущей области видимости
- is_countable — Проверить, что содержимое переменной является счётным значением
- is_double — Псевдоним is_float

- is_float — Проверяет, является ли переменная числом с плавающей точкой
- is_int — Проверяет, является ли переменная целым числом
- is_integer — Псевдоним is_int
- is_iterable — Проверяет, является ли переменная итерируемой
- is_long — Псевдоним is_int
- is_null — Проверяет, является ли значение переменной равным null
- is_numeric — Проверяет, является ли переменная числом или строкой, содержащей число
- is_object — Проверяет, является ли переменная объектом
- is_real — Псевдоним is_float
- is_resource — Проверяет, является ли переменная ресурсом
- is_scalar — Проверяет, является ли переменная скалярным значением
- is_string — Проверяет, является ли переменная строкой
- isset — Определяет, была ли установлена переменная значением, отличным от null
- print_r — Выводит удобочитаемую информацию о переменной
- serialize — Генерирует пригодное для хранения представление переменной
- settype — Задаёт тип переменной
- strval — Возвращает строковое значение переменной
- unserialize — Создаёт PHP-значение из хранимого представления
- unset — Удаляет переменную
- var_dump — Выводит информацию о переменной
- var_export — Выводит или возвращает интерпретируемое строковое представление переменной

6. ***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

7. 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.

8. 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. Если вы записываете $a = 5, вы присваиваете 5 переменной *$a*. 5, очевидно, имеет значение 5 или, другими словами, 5 - это выражение со значением 5 (в данном случае 5 - это целочисленная константа).

9. 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*.

10. 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")

11. ***Оператор присваивания в php***.

Базовый оператор присваивания обозначается как "=". На первый взгляд может показаться, что это оператор "равно". На самом деле это не так. В действительности оператор присваивания означает, что левый операнд получает значение правого выражения, (то есть устанавливается значением).

Результатом выполнения оператора присваивания является само присвоенное значение. Таким образом, результат выполнения "$a = 3" будет равен 3.

12. Операторы сравнения в php.

Пример	Название	Результат
$a == $b	Равно	**true** если *$a* равно *$b* после преобразования типов.
$a === $b	Тождественно равно	**true** если *$a* равно *$b* и имеет тот же тип.
$a != $b	Не равно	**true** если *$a* не равно *$b* после преобразования типов.
$a <> $b	Не равно	**true** если *$a* не равно *$b* после преобразования типов.
$a !== $b	Тождественно не равно	**true** если *$a* не равно *$b*, или они разных типов.
$a < $b	Меньше	**true** если *$a* строго меньше *$b*.
$a > $b	Больше	**true** если *$a* строго больше *$b*.
$a <= $b	Меньше или равно	**true** если *$a* меньше или равно *$b*.
$a >= $b	Больше или равно	**true** если *$a* больше или равно *$b*.
$a <=> $b	Космический корабль (spaceship)	Число типа int меньше, больше или равное нулю, когда *$a* соответственно меньше, больше или равно *$b*.

13. Логические операторы в php.

Пример	Название	Результат
$a and $b	И	**true**, если и *$a*, и *$b* **true**.
$a or $b	Или	**true**, если или *$a*, или *$b* **true**.
$a xor $b	Исключающее или	**true**, если *$a*, или *$b* **true**, но не оба.
! $a	Отрицание	**true**, если *$a* не **true**.
$a && $b	И	**true**, если и *$a*, и *$b* **true**.
$a || $b	Или	**true**, если или *$a*, или *$b* **true**.

Ответы на вопросы 3

1) 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.

2) Что такое сервер приложения и чем он отличается от веб-сервера?— Веб-сервер включает только веб-контейнер. В то время как сервер приложений включает в себя веб-контейнер, а также контейнер EJB. 2. Веб-сервер полезен или приспособлен для статического контента.

3) Кратко опишите историю развития интернета в рамках развития вебсерверов. — 29 октября 1969 UCLA's Network Measurement Center, Stanford Research Institute (SRI), University of California-Santa Barbara и University of Utah устанавливают ноды, ставшие первыми известными публичными серверами. Первое Интернет-сообщение "LO" было попыткой студента Чарльза Клайна "войти"(login) на компьютер SRI из Университета.

4) 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.

5) Опишите механизм взаимодействия HTTP-сервера, HTTP-клиента и пользователя.– HTTP — это клиент-серверный протокол, то есть запросы отправляются какой-то одной стороной — участником обмена (user-agent) (либо прокси вместо него). Чаще всего в качестве участника выступает веб-браузер, но им может быть кто угодно, например, робот, путешествующий по Сети для пополнения и обновления данных индексации веб-страниц для поисковых систем.
Каждый запрос (англ. request) отправляется серверу, который обрабатывает его и возвращает ответ (англ. response). Между этими запросами и ответами как правило существуют многочисленные посредники, называемые прокси, которые выполняют различные операции и работают как шлюзы или кэш, например.

6) Опишите цели и задачи веб-сервера. Главная задача веб сервера принимать HTTP-запросы от пользователей, обрабатывать их, переводить в цифровой компьютерный код. Затем выдавать HTTP-ответы, преобразуя их из миллионов нолей и единичек в изображения, медиа-потоки, буквы, HTML страницы..

7) Опишите технологию SSI. технология позволяющая удобно «собирать» веб-страницы из частей, вставлять в них результаты выполнения CGI-скриптов и придавать страницам прочие элементы динамики.

8) Что такое система управления контентом? сайта, так называемый «движок». Как работают и для чего применяются ЦМС.

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

10) Что такое CGI? стандарт интерфейса, используемого внешней программой для связи с веб-сервером.

11) Как работает система с использованием интерфейс шлюза - CGI? приложение выполняет все необходимые операции и формирует результаты в виде HTML. Сформированный гипертекст возвращается веб-серверу через стандартный поток вывода (stdout)

12) Назовите достоинства и недостатки CGI. FastCGI снимает множество ограничений CGI-программ. Недостаток CGI-программ в том, что они должны быть перезапущены веб-сервером при каждом запросе

13) Что такое FastCGI? - Интерфейс FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения

14) Назовите основные отличия CGI от FastCGI. В отличие от CGI, который создает новый процесс для каждого веб-запроса, FastCGI использует непрерывный процесс для

обработки серии веб-запросов, которые управляются диспетчером процессов FastCGI, а не веб-сервером.

15) Что такое менеджер процессов? Его суть состоит в организации рабочих процессов с учетом деления на специальные зоны ответственности.

16) Что такое PHP-FPM? является альтернативной реализацией PHP FastCGI с несколькими дополнительными возможностями, обычно используемыми для высоконагруженных сайтов.

17) Что такое Spawn-fcgi? Разделение привилегий без необходимости suid-исполняемого файла или запуска сервера с привилегиями root.

18) Что такое Lighttpd? веб-сервер, разрабатываемый с расчётом на скорость и защищённость, а также соответствие стандартам.

19) Что такое chroot окружение? Chroot-окружение – это системный вызов, который временно перемещает root каталог в новую папку.

20) Опишите механизм взаимодействия серверов с использованием FastCGI. Интерфейс FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения, дальнейшее развитие технологии CGI. По сравнению с CGI является более производительным и безопасным.

PHP интерпретатор запускается как независимый сервер, обрабатывающий входящие запросы на исполнение PHP скриптов по протоколу FastCGI, что позволяет ему работать с любым веб-сервером, поддерживающим этот протокол

более производительный и безопасный

вместо того чтобы создавать новые процессы для каждого нового запроса, использует постоянно запущенные процессы для обработки множества запросов

использует Unix Domain Sockets или TCP/IP для связи с сервером

могут быть запущены не только на этом же сервере, но и где угодно в сети

возможна обработка запросов несколькими FastCGI-процессами, работающими параллельно

в кластере должен находиться только FastCGI-процесс, а не целый веб-сервер

обеспечивает дополнительную безопасность, такую как, например, запуск FastCGI-процесса под учётной записью пользователя, отличного от пользователя веб-сервера, а также может находиться в chroot'e, отличном от chroot'a веб-сервера

может быть использован в любом языке, поддерживающем сокеты

21) Опишите процесс выбора встроенного или внешнего менеджера процессов. Зависит от конфигурации сервера, если существует один сервер, который просто является прокси для остальных, то лучше встроенный

22) Что такое интерфейс шлюза? этот сетевой стандарт позволяет Web-серверу запускать любую программу и передавать Web-браузеру данные в виде текстовой или двоичной (графической, звуковой) информации.

23) Что такое SCGI? протокол по взаимодействию приложений с веб серверами, разработанный как альтернатива Common Gateway Interface

24) Что такое PCGI? PCGI — библиотека к языку программирования Perl для работы с интерфейсом CGI

25) Что такое PSGI? PSGI или Perl Web Server Gateway Interface - спецификация, предназначенная для отделения среды веб-сервера от кода веб-фреймворка

26) Что такое WSGI? WSGI — стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером, например Apache.

27)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.

28)Опишите преимущества веб-сервера Apache.

• Высокий уровень надежности

• Гибкие настройки
• Свободный доступ к программе
• Регулярные обновления и патчи
• Удобство и легкость настройки

29)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

30)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.

31)Опишите функции ядра веб-сервера Apache.

Основные функции ядра:

• Передача данных по HTTP
• Обработка файлов
• Загрузка и поддержка модулей

32)Опишите конфигурацию веб-сервера Apache.

Конфигурацию Apache можно разделить на три основных уровня:

• Конфигурация сервера
• Конфигурация виртуального хоста
• Конфигурация уровней каталога

33)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

Ответы на вопросы 4

- Что такое HTTP-запрос?

HTTP запросы — это сообщения, отправляемые клиентом, чтобы инициировать реакцию со стороны сервера.

HTTP-запрос состоит из трех элементов:

- стартовой строки, которая задает параметры запроса или ответа
- заголовка, который описывает сведения о передаче и другую служебную информацию
- тело (его не всегда можно встретить в структуре). Обычно в нем как раз лежат передаваемые данные. От заголовка тело отделяется пустой строкой
- Опишите существующие HTTP-запросы.

Для разграничения действий с ресурсами на уровне HTTP-методов и были придуманы следующие варианты:

GET — запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

HEAD — запрашивает ресурс так же, как и метод GET, но без тела ответа.

POST — используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

PUT — заменяет все текущие представления ресурса данными запроса.

DELETE — удаляет указанный ресурс.

CONNECT — устанавливает "туннель" к серверу, определённому по ресурсу.

OPTIONS — используется для описания параметров соединения с ресурсом.

TRACE — выполняет вызов возвращаемого тестового сообщения с ресурса.

PATCH — используется для частичного изменения ресурса.

- Опишите обработку запроса на PHP. Что нужно использовать, как вычленить параметры запроса?

Любой запрос клиента к серверу должен начинаться с указания метода. Метод сообщает о цели запроса клиента. Протокол HTTP поддерживает достаточно много методов, но реально используются только три: POST , GET и HEAD . Метод GET позволяет получить любые данные, идентифицированные с помощью URL в запросе ресурса. Если URL указывает на программу, то возвращается результат работы программы, а не ее текст (если, конечно, текст не есть результат ее работы). Дополнительная информация, необходимая для обработки запроса, встраивается в сам запрос (в адресную строку). При использовании метода GET в поле тела ресурса возвращается собственно затребованная информация (текст HTML-документа, например).

- Опишите создание HTML-форм на PHP.

PHP содержит множество средств для работы с формами. Это позволяет очень просто решать типичные задачи, которые часто возникают в веб-программировании:

- Регистрация и аутентификация пользователя;
- Отправка комментариев на форумах и социальных сетях;
- Оформление заказов.

Общий принцип действия функционала: сначала пользователь регистрируется через соответствующую форму. После заполнения всех полей данные отправляются для обработки на сервер, где заносятся в таблицу БД. При следующем заходе на ресурс юзер вводит указанные при регистрации логин и пароль. Их правильность проверяется путем выборки данных из таблицы. Если оба значения указаны правильно, то пользователь попадает на страницу

приветствия. Иначе выдается сообщение о неправильном вводе пароля и логина.

- Что такое API?

Популярный термин API (англ. Application Programming Interface — программный интерфейс приложения) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

Все эти коммуникации происходят с помощью функций, классов, методов, структур, а иногда констант одной программы, к которым могут обращаться другие. Это основной принцип работы API.

- Опишите API как средство интеграции приложений.

Интеграция API – это соединение между двумя или более приложениями через их API (интерфейсы прикладного программирования), которые позволяют системам обмениваться источниками данных. Интеграция API позволяет управлять процессами во многих секторах и уровнях организации, обеспечивая синхронизацию данных, повышая производительность и увеличивая прибыль.

- Что такое Web API?

Web API или Web Service API –это интерфейс обработки приложений между веб-сервером и веб-браузером. Все веб-сервисы являются API, но не все API являются веб-сервисами. REST API – это особый тип Web API, в котором используется стандартный архитектурный стиль, описанный выше.

- Приведите пример API.

На практике API могут использоваться для связи практически любых процессов. Вот несколько распространенных примеров использования API:

- Обмен информацией о рейсах между авиакомпаниями и туристическими сайтами

- Использование Google Maps в приложении для совместных поездок (райдшеринга)
- Создание виртуальных собеседников в службе обмена сообщениями
- Встраивание видеоклипов с YouTube на веб-странице
- Автоматизация рабочих процессов в программных инструментах для B2B-сектора

- Что такое REST?

REST (от англ. Representational State Transfer — «передача репрезентативного состояния» или «передача „самоописываемого“ состояния») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Другими словами, REST — это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать.

- Как организована передача данных в архитектуре REST?

В REST архитектуре клиенты отправляют на сервер запросы для получения или модификации данных, а сервера отправляют клиентам ответы на их запросы.

- Как организована работа REST?

Как было сказано выше, REST определяет, как компоненты распределенной системы должны взаимодействовать друг с другом. В общем случае этот происходит посредством запросов-ответов. Компоненту, которая отправляет запрос называют клиентом; компоненту, которая обрабатывает запрос и отправляет клиенту ответ, называют сервером. Запросы и ответы, чаще всего, отправляются по протоколу HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста»). Как правило сервер — это некое веб-приложение. Клиентом же может быть не то чтобы что угодно, но довольно

многое. Например, мобильное приложение, которое запрашивает у сервера данные. Либо браузер, который отправляет запросы с веб-страницы на сервер для загрузки данных. Приложение А может запрашивать данные у приложения Б. Тогда А является клиентом по отношению к Б, а Б — сервером по отношению к А. Одновременно с этим, А может обрабатывать запросы от В, Г, Д и т.д. В таком случае, приложение А является одновременно и сервером, и клиентом. Все зависит от контекста. Однозначно одно: компонента которая шлет запрос — это клиент. Компонента, которая принимает, обрабатывает и отвечает на запрос — сервер.

- Что такое SOAP?

SOAP (от англ. Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Первоначально SOAP предназначался в основном для реализации удалённого вызова процедур (RPC). Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур.

- Чем SOAP отличается от REST?

REST был создан для решения проблем SOAP. Поэтому у него более гибкая архитектура. Он состоит только из простых рекомендаций и позволяет разработчикам реализовывать рекомендации по-своему. Он допускает различные форматы сообщений, такие как HTML, JSON, XML и простой текст, в то время как SOAP допускает только XML. REST также является более легкой архитектурой, поэтому веб-сервисы RESTful имеют более высокую производительность.

- Для чего нужен SOAP-процессор?

SOAP позволяет разработчикам вызывать процессы, запущенные в разных операционных системах (таких как Windows, macOS и Linux), для аутентификации, авторизации и обмена данными с использованием

расширяемого языка разметки (XML). Поскольку веб-протоколы, такие как HTTP, установлены и работают практически во всех операционных системах, SOAP позволяет клиентам вызывать веб-службы и получать ответы независимо от языка и платформы.

- Опишите общую структуру SOAP-сообщения.

SOAP-сообщение – это обычный XML-документ, содержащий следующие элементы:

- Конверт
- Заголовок
- Тело
- Неисправность

- Что такое и что содержит Конверт (SOAP Envelope)?

Конверт SOAP аналогичен конверту обычного письма. Он содержит информацию о письме, которое будет шифровано в основном разделе SOAP, включая данные о получателе и отправителе, а также информация о самом сообщении. Например, заголовок конверта SOAP может указывать на то, как должно обрабатываться сообщение.

- Что такое и что содержит Заголовок SOAP (SOAP Header)?

Заголовок – содержит любые необязательные атрибуты сообщения, используемые при обработке сообщения, либо в промежуточной точке, либо в конечной конечной точке. Это необязательный элемент.

- Что такое и что содержит Тело SOAP (SOAP Body)?

Тело – содержит данные XML, содержащие отправляемое сообщение. Это обязательный элемент.

- Опишите SOAP-сообщение с вложением.

Неисправность – необязательный элемент неисправности, который предоставляет информацию об ошибках, возникающих при обработке сообщения.

- Что такое graphql?

GraphQL — это язык запросов для. API - интерфейсов и среда, в которой они выполняются. С помощью GraphQL можно получить данные из API и передать их в приложение (от сервера к клиенту). Официальная документация. GraphQL есть только на английском языке, на русский язык пока ещё не переведена.

- Что такое Распознаватели (resolvers) в graphql?

Резолверы — это функции, которые запускаются каждый раз, когда запрос запрашивает поле. Когда реализация GraphQL получает запрос, она выполняет резолвер для каждого поля. Если резолвер возвращает поле типа Object, то GraphQL запускает резолвер-функцию этого поля. Когда все резолверы возвращают скалярные значения, цепочка замыкается, и запрос получает готовый JSON-результат.

- Из чего состоит экосистема graphql, что нужно, чтобы использовать данную технологию?

GraphQL используется для работы с данными в «вашем приложении», а не «в вашей базе данных». Дело в том, что GraphQL — это система, независимая от источников данных, то есть, для организации её работы неважно — где именно хранятся данные.

- Что такое валидация данных и для чего она нужна?

Валидация данных (англ. Data validation) — это процесс проверки данных различных типов по критериям корректности и полезности для конкретного применения. Валидация данных проводится, как правило, после

выполнения операций ETL и для подтверждения корректности результатов работы моделей машинного обучения (предиктов).

- Где и когда выполнять валидацию данных?

Для валидации требуется доступ к недоступной части состояния системы. Это особенно характерно для проверки данных, вводимых человеком через графический интерфейс пользователя. Современные приложения часто построены с использованием многоуровневой архитектуры, которая предполагает, что реализация пользовательского интерфейса выделена в презентационный слой, а для проверки требуется доступ к другим слоям, вплоть до слоя базы данных.

Валидация требует полностью повторить логику обработки. Как уже отмечено двумя абзацами выше, при многослойной архитектуре приложения пользовательский интерфейс обычно выделяется в специальный презентационный слой, а логика обработки данных находится на другом слое. И бывают такие ситуации, когда для валидации нужно практически полностью выполнить эту обработку, потому не существует более короткого способа понять, завершится она успехом или нет.

- Как выполнять валидацию данных?

Посимвольная проверка. Как правило такие проверки выполняются в пользовательском интерфейсе, по мере ввода данных. Но не только. Например, лексический анализатор компилятора тоже выявляет недопустимые символы непосредственно в процессе чтения компилируемого файла. Поэтому такие проверки можно условно назвать «лексическими».

Проверка отдельных значений. Для пользовательского интерфейса это проверка значения в отдельном поле, причём выполняться она может как по мере ввода (проверяется то неполное значение, которое введено к настоящему моменту), так и после завершения ввода, когда поле теряет фокус. Для программного интерфейса (API) это проверка одного из параметров,

переданных в вызываемую процедуру. Для данных, получаемых из файла, это проверка какого-то прочитанного фрагмента файла. Такие проверки, опять-таки по аналогии с компиляторной терминологией, можно назвать «синтаксическими».

Совокупность входных значений. Можно предположить, что в программу сначала передаются какие-то данные, после чего подаётся некоторый сигнал, который инициирует их обработку. Например, пользователь ввёл данные в форму или в несколько форм (в так называемом «визарде») и наконец нажал кнопку «OK». В этот момент можно выполнить так называемые «семантические» проверки, нацеленные на валидацию не только отдельных значений, но и взаимосвязей между ними, взаимных ограничений.

Проверка состояния системы после обработки данных. Наконец, есть последний способ, к которому можно прибегнуть, если валидацию непосредственно входных данных выполнить не удаётся — можно попытаться их обработать, но оставить возможность вернуть всё к исходному состоянию. Такой механизм часто называется транзакционным.

- Приведите пример с поэтапной валидацией данных.

Рассмотрим пример розничного продавца, который собирает данные о своих магазинах, но не может создать надлежащую проверку почтового индекса. Надзор может затруднить использование данных для получения информации и бизнес-аналитики. Если почтовый индекс не введен или введен неправильно, может возникнуть несколько проблем.

В некоторых картографических программах может быть сложно определить местоположение хранилища. Почтовый индекс магазина также поможет получить представление о районе, в котором расположен магазин. Без проверки данных почтового индекса более вероятно, что данные потеряют ценность. Это приведет к дополнительным расходам, если потребуется восстановить данные или ввести почтовый индекс вручную.

Простым решением этой проблемы было бы установить флажок, гарантирующий ввод действительного почтового индекса. Решением может быть выпадающее меню или форма автоматического заполнения, которая позволяет пользователю выбрать почтовый индекс из списка действительных кодов. Такой тип проверки данных называется проверкой кода или проверкой кода.

- Что такое запрос и мутация в graphql и чем они отличаются?

В GraphQL есть только два типа операций, которые вы можете выполнять: запросы и мутации.

В то время как мы используем запросы для извлечения данных, мы используем мутации для изменения данных на стороне сервера.

Если запросы в GraphQL эквивалентны вызовам GET в REST, то мутации представляют собой методы изменения состояния в REST (например, DELETE, PUT, PATCH и т.д.).