**ВОПРОСЫ для подготовки к ГИА**

**по дисциплине «Интернет-программирование»**

**Направление подготовки: 09.03.03 – Прикладная информатика**

1. Определение MVP (Minimum Viable Product). Зачем нужен MVP

Минимально жизнеспособный продукт (англ. minimum viable product, MVP) — продукт, обладающий минимальными, но достаточными для удовлетворения первых потребителей функциями и продемонстрировать возможности ПО клиенту. Основная задача MVP — получение обратной связи для формирования гипотез дальнейшего развития продукта. Такой вариант часто используют стартапы.

1. Виды MVP (перечислислить, рассказать про каждый)

Какие бывают MVP:

* **Целевая страница**, где собрана подробная информация о будущем продукте и запланированных функциях. Здесь собираются емайл или другие контакты с заинтересованными пользователями. Так пополняется клиентская база, которой потом предлагают демо и другие варианты тестирования продукта через доступные каналы коммуникации.
* **Консьерж.** Пока продукта не существует, все функции выполняют люди. И общение с пользователями происходит не через интерфейс, а напрямую (мессенджеры, email и так далее). К примеру, если вместо сервиса бронирования отелей нужно обратиться к менеджеру.
* **Имитация продукта.** В этой версии есть интерфейс для пользователя, но функционал реализовывается через самих разработчиков, которые вручную выполняют запросы. То есть на этом этапе не работает автоматизация, которая может появиться в перспективе, но юзеры могут об этом и не знать.

Примеры: Uber, Amazon, Duolingo, Instagram

1. Определение CMS. В каких случаях следует использовать CMS.

Так появились CMS — системы управления контентом (англ. Content Management System).

Часто CMS называют «движок» сайта, но это неверно. CMS управляет только контентом, а «движок» — это система, на которой сайт работает. Некоторые системы управления контентом сайта — это два в одном: обеспечивают работу сайта и позволяют редактировать его содержимое. Например, Wordpress или Joomla.

Сайт, который работает на CMS, легко наполнять контентом любого вида, вносить исправления, отвечать на комментарии, подключать дополнительные модули, например, систему электронных платежей или форму обратной связи. В большинстве случаев дополнительные знания по программированию, дизайну или вёрстке не нужны. Применение CMS поможет владельцам сайтов любой сложности работать с контентом быстро и просто.

CMS системы лучше всего использовать для простых сайтов без сложной логики. Например, для строительной фирмы, или же какого-нибудь агенства. Одним словом, сайты визитки.

1. Перечислите виды CMS, их преимущества и недостатки.

Программы управления контентом могут быть разделены на несколько категорий:

* Самописные. Такую систему разрабатывают специально для задач клиента. Программисты с нуля пишут код для создания страниц, добавления видео, фото и любых функций. Обычно самописные CMS для сайтов разрабатывают крупные компании, если нужно, чтобы сайт выполнял какую-то нестандартную задачу. Например, Ozon работает на собственной CMS.

**Плюсы:** Можно реализовать любые идеи. Можно быстро внедрять любые новые технологии.

**Минусы:** Для разработки самописной CMS нужно много времени и большой бюджет. Скорее всего потребуется создание собственного отдела разработки, где несколько программистов

* Коробочные. Это уже готовая рабочая система, на которой можно создать сайт любой тематики — хоть для детского сада, хоть для нефтеперерабатывающего завода. В «коробке» уже есть необходимый функционал, например шаблоны оформления страниц сайта или варианты структуры блога.Коробочные CMS могут быть платными, как например 1C-Битрикс, или бесплатными как WordPress или Drupal.

**Плюсы:** В «коробке» уже заложены основные модули и шаблоны, которые нужны для создания сайта, их можно быстро развернуть и настроить.

Легко поменять внешний вид сайта, структуру, контент.

**Минусы:** Если нужно реализовать сложное решение, например интернет-магазин с десятками тысяч артикулов, то проще нанять специалиста, который настроит и запустит сайт

* Конструкторы. Конструктор — это интернет-платформа для создания сайтов, на которой всё уже продумано и готово: дизайн, размеры блоков и их расположение. Ничего не нужно скачивать и устанавливать. Конструкторы используют для несложных сайтов с небольшим количеством страниц.

**Плюсы:** Можно быстро создать и запустить сайт с минимальным функционалом, а после — его совершенствовать. Огромное количество шаблонов для всех сфер бизнеса. Дополнительные решения, которые можно внедрить за пару минут. Например, подключить чат-бота

**Минусы:** Зависимость от компании-разработчика. Если нужно будет перенести сайт на отдельный хостинг, придётся заплатить. Пользователь скован рамками шаблонов.

Пример Тильда или Викс

1. Для чего применяются системы контроля версий?

Система контроля версий – это система, записывающая изменения

в файл или набор файлов в течение времени и позволяющая вернуться позже к определенной версии. Можно отменить те или иные изменения файла, откатить его удаление, посмотреть кто что-то поменял. Как правило системы контроля версий применяются для хранения исходного кода, но это необязательно. Они могут применяться для хранения файлов совершенно любого типа.

Git — это распределённая система управления версиями (Version Control System). Она создана, чтобы помогать участникам команды сотрудничать наиболее эффективно:

* отслеживать изменения в исходном коде и других текстовых файлах в рамках совместного проекта;
* вносить правки и контролировать версии в ходе разработки;
* возвращаться к снимкам и находить неполадки.

1. Фреймворки и библиотеки, определения и отличия.

Фреймворк (англ. framework — «каркас, структура») — готовый набор инструментов, который помогает разработчику быстро создать продукт: сайт, приложение, интернет-магазин, CMS-систему.

Зачем надо?

**Ускорить разработку**. При использовании фреймворка программисту не нужно писать много кода с нуля. Он берёт готовый набор инструментов и подстраивает их под свой проект.

Если не использовать фреймворки, потребуется самостоятельно придумывать, как отобразить данные на экране, как и где их хранить, как получать. Это может занять много времени.

**Предотвратить ошибки.** Благодаря фреймворкам разработчик может избежать распространённых ошибок — как архитектурных, так и функциональных.

Пример:Angular, ASP.NET

Библиотека — набор методов, который упрощает труд разработчика. Как правило, библиотеки отвечают только за одну область: например, хранение данных или рендеринг — их вывод на страницу.

Пример: Prime NG для Angular

Библиотеки никак не влияют на архитектуру проекта. Фреймворки же как раз задают архитектуру и содержат инструменты из разных областей, таких как рендеринг, работа с базами данных, обработка запросов.

1. Что такое REST API. Для чего используют REST API

REST API — это архитектурный подход, который устанавливает ограничения для API: как они должны быть устроены и какие функции поддерживать. Это позволяет стандартизировать работу программных интерфейсов, сделать их более удобными и производительными.

HTTP (HyperText Transfer Protocol) — это протокол передачи данных, используемый во всемирной паутине для обмена информацией между клиентами и серверами.

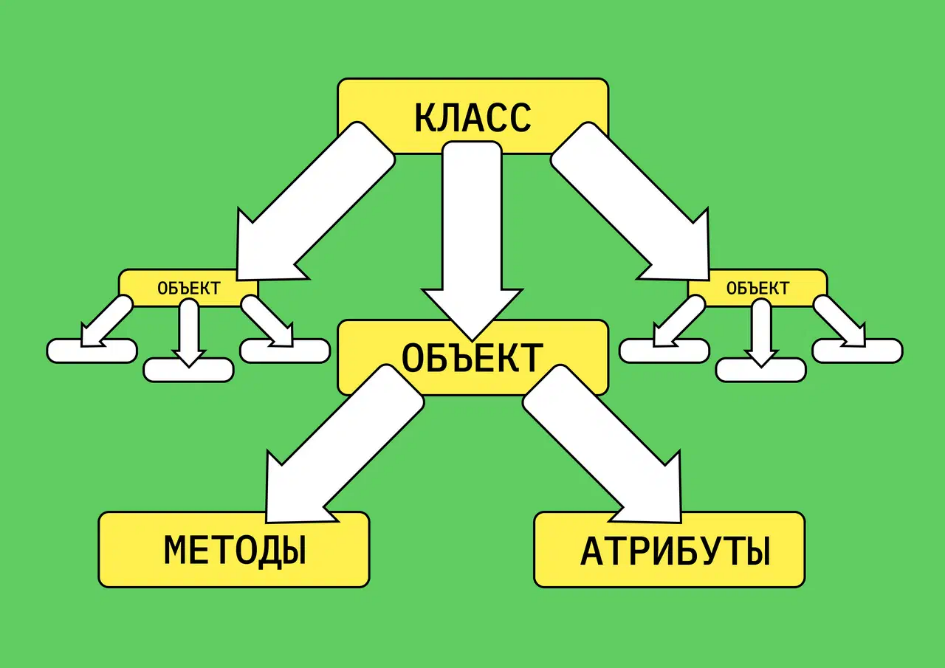
HTTP определяет методы запроса к ресурсу, которые используются для выполнения разных операций:

* GET-запрос используется для получения информации от сервера.
* PUT-запрос используется для обновления существующих данных на сервере
* DELETE-запрос используется для удаления указанного ресурса на сервере.
* POST-запрос для добавления данных

1. Что такое объектно-ориентированное программирование (ООП)?

Объектно-ориентированное программирование, или ООП — это одна из парадигм разработки. Парадигмой называют набор правил и критериев, которые соблюдают разработчики при написании кода.

Суть понятия объектно-ориентированного программирования в том, что все программы, написанные с применением этой парадигмы, состоят из объектов. Каждый объект — это определённая сущность со своими данными и набором доступных действий.



1. Какие языки программирования используются для формирования backend части сайта. От чего зависит выбор языка программирования в интернет-проекте.

Во frontend главное требование к языку — способность работать в браузере, в backend важны удобство, скорость и производительность. А еще — то, насколько язык подходит для конкретной задачи.

Универсальной технологии не существует: под каждую выбирается своя, со своими преимуществами и недостатками. Примеры языков: Go, Java, C#, Ruby, PHP

Выбор языка программирования для интернет-проекта зависит от следующих факторов:

* Требования к проекту. Необходимо определить требуемую функциональность, сложность и объём веб-приложения.
* Простота использования. Оцените, насколько легко изучать и использовать тот или иной язык.
* Масштабируемость. Если в будущем веб-приложение будет расти или справляться с возрастающими нагрузками, необходимо выбрать язык, способный поддерживать этот рост.
* Производительность. Выбирайте язык, обеспечивающий отличную скорость выполнения, эффективную обработку сложных задач и отзывчивость пользователей.
* Поддержка сообщества. Ищите языки с развитым сообществом разработчиков, обширными ресурсами и библиотеками.
* Совместимость с платформами. Оцените совместимость выбранного языка с предполагаемыми платформами, такими как web, mobile или desktop.

1. Методы получения данных с сервера на стороне клиента.

Основа передачи данных в вебе — это HTTP (HyperText Transfer Protocol), клиент-серверный протокол взаимодействия. Работает он следующим образом.

Клиент, обычно веб-браузер, инициирует запрос к серверу. В ответ он получает необходимый документ — HTML-страницу — или сообщение об ошибке, если запрос сформирован неправильно или документа нет на сервере.

* GET-запрос используется для получения информации от сервера.
* PUT-запрос используется для обновления существующих данных на сервере
* DELETE-запрос используется для удаления указанного ресурса на сервере.
* POST-запрос для добавления данных