Статические сайты, как правило, используются на сайтах визитках, либо для документации. Для чего то более серьезного уже нужен динамический сайт.

Что его характеризует?

Данные отделены от логики работы сайта и находятся в хранилище. Например, если на сайте есть раздел со статьями, то для вывода статьи существует ровно один обработчик (код, который отвечает за этот вывод), а данные, при этом, зависят от ссылки по которому открыта статья. То есть, в динамических сайтах однотипные страницы генерируются из одного места с подстановкой разных данных. В случае статического сайта, каждая ссылка была бы представлена физическим файлом с вбитым в него контентом.

- *# Пример шаблона для генерации топика на хекслете (haml + ruby)*

.card

.card-header

%**div**

= link\_to user\_path(@topic.creator.username), **class**: '**no**-**color**' **do**

%**b**= @topic.creator

.card-block

%**div**

.hexlet-topic-content.hexlet-content-container

%**p**= markdown2html(@topic.title)

.mt-4

- @topic.comments.each **do** |comment|

%**div**

%**hr**.mb-3.hexlet-dashed

.row

.col-md-10

%**b**= comment.creator

%**div**

.hexlet-topic-content.hexlet-content-container

%**p**= markdown2html(comment.body)

При создании сайта, обычно, делают специальный административный интерфейс, через который можно модифицировать эти данные. А операции по манипулированию сущностями называют CRUD, что расшифровывается как Create, Read, Update и Delete. Другими словами для управления пользователями в админке (жаргон) есть соответствующий круд. Кстати распространено мнение что большинство веб-разработчиков занимается, по большей части, созданием крудов.

Самыми распространенными решениями, для хранения данных, являются реляционные базы данных, такие как: MySQL и PostgreSQL. Их использование подразумевает знание SQL, языка, который позволят манипулировать этими данными внутри базы данных и получать их наружу.

*-- Пример sql запроса*

**SELECT** **id**, **name** **from** 'users' **where** state = "active" **ORDER** **BY** **id** **DESC** **LIMIT** 20;

Для описания логики сайта и генерации HTML используется один из серверных языков программирования. Теоретически, для создания сайтов, можно использовать почти любой язык, но так сложилось, что только некоторые из них популярны для веба:

* PHP
* JavaScript (и его производные Elm, TypeScript, ClojureScript)
* Ruby
* Python
* Erlang/Elixir
* Go
* Clojure
* Java/Kotlin/Scala
* C#

И хотя, перечисленные выше языки, могут сильно отличаться друг от друга, принципы по которым строятся веб-прилжения в них - совпадают.

Еще раз подчеркну то, что выполнение кода происходит только на сервере и скрыто от глаз пользователей. Все что видно из браузера, это HTML документ пришедший от сервера.

Ниже я попробую классифицировать способы разработки веб-приложений:

Конструкторы

Несмотря на то, что этот способ не требует программирования, его нельзя не упомянуть. На рынке представлено десятки конструкторов для создания сайтов без программирования, особенно популярны такие решения в e-commerce (интернет-магазины). Например [ecwid](https://www.ecwid.com/) или [setup.ru](http://setup.ru/?r=fyrnoanp).

CMS

Content Management System - это программное решение, которое позволяет собрать сайт из уже готовых блоков. Расширяется такая система только с помощью плагинов, которых довольно много у популярных систем. В случае необходимости можно дописать и свое. Некоторые из подобных систем платные, другие бесплатные. Например, Wordpress относится к бесплатным, при этом является одной из самых качественных и популярных CMS в мире.

Такие системы особенно распространены в PHP среде. Существуют даже [рейтинги](http://www.ratingruneta.ru/cms/) по которым можно ориентироваться. Этот рейтинг создан в первую очередь для бизнесменов, но его так же полезно посмотреть для того чтобы примерно оценить состояние дел на рынке.

Хотя использование CMS выглядит очень заманчивым, но так же как и shared хостинг, для любой, более менее серьезной, системы CMS будет больше мешать чем помогать. Чаще их используют когда надо дешево и сам сайт представляет из себя нечто типичное, например, каталог продуктов. Системы аналогичные [booking.com](https://booking.com/) или [Яндекс](https://yandex.ru/) невозможно построить на базе CMS.

Фреймворки

Основной способ разработки, используемый профессиональными разработчиками. Фреймворк - это каркас, который создан для программистов и предоставляет готовые решения для типичных задач веб-разработки, например маршрутизацию, интеграцию с хранилищем, шаблонизацию и многое другое. Фреймворки не навязывают конкретную структуру базы данных (в отличие от CMS), более того они вообще не требуют ее наличия. С другой стороны, у хороших фреймворков такое количество дополнений, что сайт с не самой простой логикой и возможностями можно запрограммировать (почти что собрать) за очень короткий срок.

Фреймворк, который больше других повлиял на мир веб-разработки, называется Ruby on Rails. Другой Ruby фреймворк Sinatra, стал основателем направления "микрофреймворков", которые теперь водятся в большом количестве в любом языке программирования. Они все похожи как братья близнецы, знаете принципы работы одного, легко сможете ориентироваться в остальных, даже на других языках.

require 'sinatra'

get '/frank-says' do

'Put this in your pipe & smoke it!'

**end**

Чистый язык (Самописное решение)

Такое встречается разве что в PHP. На самом деле нет ни одной причины (кроме отсутствия квалификации), по которой стоит выбирать разрабатывание сайта без использования, как минимум, фреймворков. А любой разговор о производительности должен начинаться только после прочтения [http://optimization.guide](http://optimization.guide/)

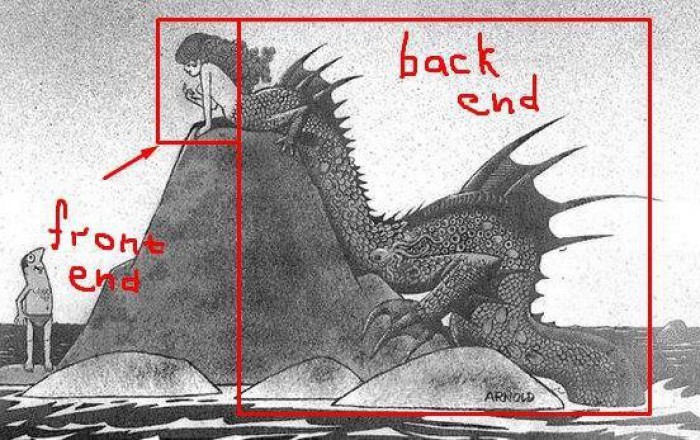
Веб-сервер

Какой бы способ разработки вы не выбрали, есть один элемент без которого обойтись нельзя, это веб-сервер. Веб-сервером называется специальная программа, которая принимает входящие http(s) запросы, например, из браузера, запускает ваш код на выполнение и возвращает сформированный ответ. Веб-сервер может возвращать не только html страницы, но так же и любые ресурсы, такие как архивы, рисунки, видео. Самым популярным решением на сегодняшний день является [nginx](https://www.nginx.com/).

Понимание работы веб-серверов крайне важно для любого веб-разработчика, она включает в себя знание операционных систем, сетей и протоколов.

Сервисы

Backend разработка не ограничивается только самим сайтом, более того, сайт это всего лишь вершина айсберга. Любой более менее серьезный проект, под капотом представляет из себя множество подсистем (говорят сервисы или микросервисы). Возьмите например сайт [booking.com](http://www.booking.com/), мировой лидер по бронированию отелей. Посмотрите на него внимательно, пройдитесь по ссылкам и попробуйте прикинуть, сколько программистов работает внутри. Думаю что цифра вас удивит, программистов в букинге более 800 человек. Отдельная команда занимается подсистемой нотификаций (емейлы, факсы), другая биллингом, третья разрабатывает Backend для мобильного приложения, четвертая собственно мобильное приложение, а скорее всего мобильных команд несколько, каждая под свою платформу.



Как правило в подобных проектах используются всевозможные технологии, какие только можно вообразить. Часть внутренних команд не имеет никакого отношения к вебу, хотя весь продукт

ТЕСТЫ

Что из себя представляет динамический сайт в типичной конфигурации?

Веб-сервер + код + база данных

Можно ли всегда однозначно определить является ли сайт статическим или динамическим?

Нет, ведь браузер получает уже готовый html