Backend разработка—Введение в веб-разработку

Бэкенд-разработчик (с англ. back-end (дословно «задняя часть») developer; варианты названия профессии: backend-программист, девелопер или просто backend) — это специалист, который занимается программно-административной частью веб-приложения, внутренним содержанием системы, серверными технологиями — базой данных, архитектурой, программной логикой. Обычно бэкенды сами администрируют свои системы, если же эта обязанность возложена на отдельного специалиста — DevOps, тогда backend-программист может сконцентрироваться на написании непосредственно кода.

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 которой 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 — **Create**, **Read**, **Update** и **Delete**. Другими словами, для управления статьями 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 позволяет 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-приложения 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 представлены десятки конструкторов для создания сайтов без программирования, особенно популярны такие решения в e-commerce (интернет-магазины). Например, ecwid или setup.ru.

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 создать свой плагин. 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 среде PHP. Существуют даже рейтинги, по которым можно ориентироваться. Этот рейтинг создан в первую очередь для бизнесменов, но его также 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 сложной системы, CMS будет больше мешать, чем помогать. Чаще их используют для типовых решений, например, каталога продуктов или сайта о компании. Системы, аналогичные booking.com или Яндекс невозможно построить на базе CMS.

Фреймворки

Основной способ разработки, используемый профессиональными разработчиками. Фреймворк — это каркас, который создан для программистов. Он 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 её наличия. С другой стороны, у хороших фреймворков такое количество дополнений, что сайт с не самой простой логикой и возможностями можно запрограммировать (почти 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.

```
# app.rb
require 'sinatra'

# get это функция соответствующая глаголу GET протокола HTTP
get '/frank-says' do
  'Put this in your pipe & smoke it!'
end
$ ruby app.rb
== Sinatra (v1.4.8) has taken the stage on 4567 for development with backup from WEBrick

$ curl http://localhost:4567/frank-says
Put this in your pipe & smoke it!
```

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 optimization.guide.

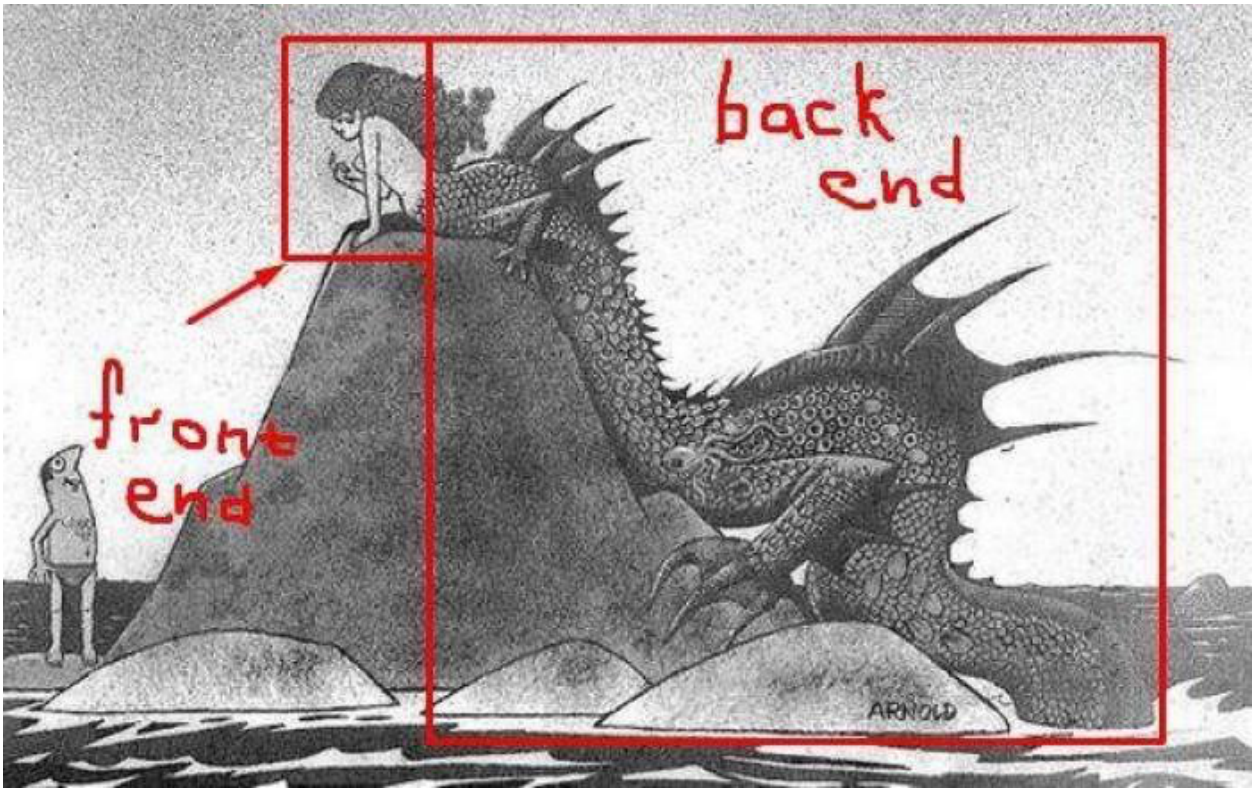
Веб-сервер

Какой бы способ разработки вы ни выбрали, есть один элемент, без которого обойтись нельзя: 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 также и другие ресурсы, такие как архивы, рисунки, видео. Одним из самых популярных решений на сегодняшний день является nginx.

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

Сервисы

Backend разработка не ограничивается только самим сайтом. Сайт — это всего лишь вершина айсберга. Любой более-менее серьезный проект под капотом представляет из себя множество подсистем (говорят, «сервисы» или «микросервисы»). Возьмите, к примеру, сайт 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 больше 800. Отдельная команда занимается подсистемой нотификаций (email'ы, факсы), другая — биллингом, третья разрабатывает Backend для мобильного приложения, четвертая, собственно, мобильное приложение. И, скорее всего, мобильных команд несколько — каждая под свою платформу.



Как правило, в подобных проектах используются самые разные технологии. Часть внутренних команд не имеет никакого отношения к вебу, хотя весь продукт представляет из себя в первую очередь веб-сайт.

Основные инструменты бэкенд-разработчика — серверные языки программирования. В целом в работе специалист использует разные инструменты:

- языки PHP, Python, Ruby, Java, C#, Node.js (программная платформа);
- дополнительно к Node.js полезно изучить Express (библиотека для взаимодействия платформы Node.js с сервером) и Mongo DB (базу данных для получения и хранения информации);
- в качестве дополнительных средств применяются фреймворки Laravel, Symfony, CodeIgniter, Django, Flask, Ruby on Rails, Spring, Express.
- для хранения данных используются MySQL, PostgreSQL, SQLite.

Таким образом, круг задач бэкенд-разработчика выглядит так:

- разработка модели предметной области (домена);
- разработка платформы и основного функционала, то есть бизнес-логики;
- разработка разработка безопасных приложений, поддерживающих пользовательский интерфейс;
- настройка серверов (боевого, тестового и рабочего) и программ по мониторингу их состояний;
- использование системы контроля версий (это общее требование для всех программистов);
- настройка баз данных, создание моделей предметной области и взаимодействий между ними;
- настройка процессов непрерывной интеграции и поставки.

Профессиональные знания и навыки

В первую очередь бэкенд-разработчику понадобится знание хотя бы одного языка программирования: Go, PHP, Python, Ruby, Java, C#. К этой базе должны прилагаться:

- умение писать быстрый, красивый и правильный код;
- знание популярных веб-фрейморков;
- умение проектировать базы данных и оптимизировать запросы;
- знание современных парадигм программирования;
- знание паттернов проектирования;
- понимание устройств веб-сервисов, интерфейсов;
- английский язык для чтения технической документации.