РАЗДЕЛ 1 ПОДБОР ПОДХОДЯЩИХ ТЕХНОЛОГИЙ

Этот раздел содержит выбор подходящих технологий и выбор среды разработки.

1.1 Выбор технологи

Самое основное что должно быть на сайте, это HTML и CSS.

HTML «язык гипертекстовой разметки» - стандартизированный язык программирования документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML).

CSS - формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XML.

Для данной задачи, а именно создание сервиса для отдачи HTML и CSS файлов может подойти любой язык программирования (далее ЯП), но есть основные критерии которыми должен придерживаться каждый разработчик для выбора оптимального языка, а именно:

* объём данных по языку (материалы для изучения, подводные камни и прочие вспомогательные знания);
* легкость в использовании;
* легкость в переносе на новое оборудование;
* возможность использования на сервере для использования бесплатного хостинга.

Учитывая критериев выше, основными языками для выбора становятся:

* JavaScript;
* Java;
* C#;
* Python;
* PHP.

Мой выбор пал на JavaScript так как он единственный язык, который можно легко использовать и в браузере, а не только в серверной части и из-за динамической типизацией для реализации прототипа будет меньше потрачено времени на разработку самого прототипа

1.2 Технологии системы с использованием JavaScript

Одна из наиболее сложных проблем при написании систем, общающихся по сети - обработка ввода и вывода. Чтение и запись данных в сеть и из сети, на диск, и другие устройства. Перемещение данных требует времени, и грамотное планирование этих действий может сильно повлиять на время отклика системы для пользователя или сетевых запросов.

1.2.1 Node.js

В традиционном методе обработки ввода и вывода принято, что функция, к примеру, readFile, начинает читать файл и возвращается только когда файл полностью прочитан. Это называется синхронным вводом-выводом (synchronous I/O, input/output).

Node был задуман с целью облегчить и упростить использование асинхронного I/O. Такой интерфейс позволяет скрипту продолжать работу, пока интерфейс делает свою, и вызывает функцию обратного вызова по окончанию работы. Таким образом в Node работает весь I/O.

JavaScript легко вписывается в систему типа Node. Это один из немногих языков, в которые не встроена система I/O. Поэтому JavaScript легко встраивается в довольно эксцентричный подход к I/O в Node и в результате

непорождает две разных системы ввода и вывода. В 2009 году при разработке Node люди уже использовали I/O в браузере, основанный на обратных вызовах, поэтому сообщество вокруг языка было привычно к асинхронному стилю программирования.

1.2.2 Асинхронность

В синхронном окружении очевидным способом решения задачи будет сделать запросы последовательно. У этого метода есть минус – второй запрос начнётся только после окончания первого. Общее время будет не меньше, чем сумма времени на обработку двух запросов. Это неэффективное использование компьютера, который большую часть времени будет простаивать, пока происходит передача данных по сети.

Решение проблемы в синхронной системе – запуск дополнительных потоков контроля исполнения программы Второй поток может запустить второй запрос, и затем оба потока будут ждать возврата результата, после чего они заново будут синхронизированы для сведения работы в один результат.

1.2.3 Модули

Кроме нескольких упомянутых переменных, вроде console и process, Node держит мало функционала в глобальной области видимости. Для доступа к остальным встроенным возможностям вам надо обращаться к системе модулей.

Система встроена в Node и используется для загрузки всего, от встроенных модулей и скачанных библиотек до файлов, являющихся частями вашей программы.

При вызове require Node нужно преобразовать заданную строку в имя файла. Пути, начинающиеся с «/», «./»или «../», преобразуются в пути относительно текущего. «./» означает текущую директорию, «../» – директорию выше, а «/» – корневую директорию файловой системы. Если вы запросите «./world/world» из файла /home/marijn/elife/run.js, Node попробует

загрузить файл /home/marijn/elife/world/world.js. Расширение .js можно опускать.

Когда передаётся строка, которая не выглядит как относительный или абсолютный путь, то предполагается, что это либо встроенный модуль, или модуль, установленный в директории node\_modules. К примеру, require(«fs») выдаст вам встроенный модуль для работы с файловой системой, а require («elife») попробует загрузить библиотеку из node\_modules/elife/.

1.2.4 NPM

NPM, вскользь упомянутый в главе 10, это онлайн-хранилище модулей JavaScript, многие из которых написаны специально для Node. Когда вы ставите Node на компьютер, вы получаете программу npm, которая даёт удобный интерфейс к этому хранилищу.

После запуска npm install NPM создаст директорию node\_modules. Внутри неё будет директория figlet, содержащий библиотеку. Когда мы запускаем node и вызываем require(«figlet»), библиотека загружается и мы можем вызвать её метод text, чтобы вывести большие красивые буквы.

Что интересно, вместо простого возврата строки, в которой содержатся большие буквы, figlet.text принимает функцию для обратного вызова, которой он передаёт результат. Также он передаёт туда ещё один аргумент, error, который в случае ошибки будет содержать объект error, а в случае успеха – null.

Такой принцип работы принят в Node. Для создания букв figlet должен прочесть файл с диска, содержащий буквы. Чтение файла – асинхронная операция в Node, поэтому figlet.text не может вернуть результат немедленно. Асинхронность заразительна – любая функция, вызывающая асинхронную, сама становится асинхронной.

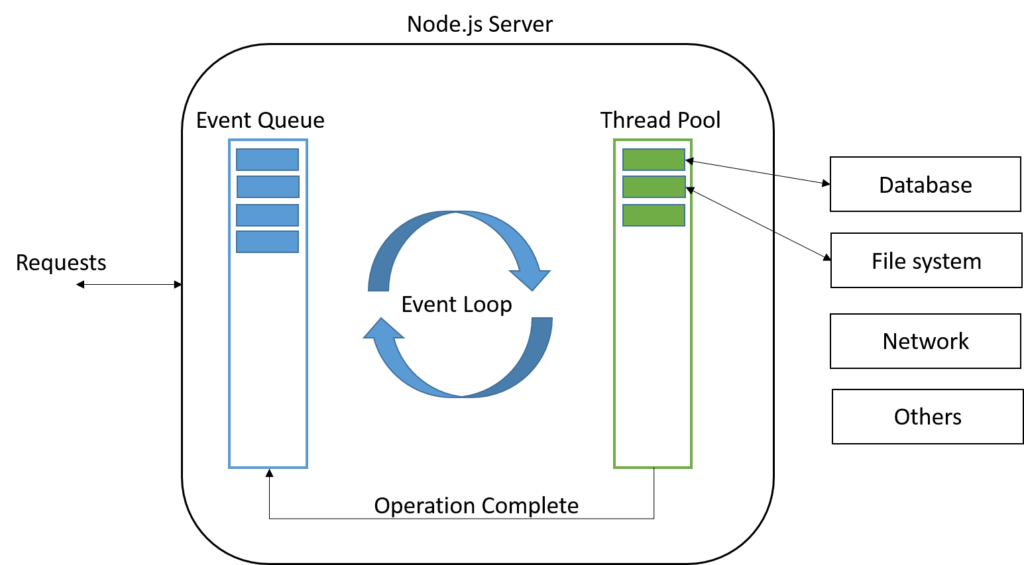


Рис. 1.1 – Структура EventLoop

NPM – это больше, чем просто npm install. Он читает файлы package.json, содержащие информацию в формате JSON про программу или библиотеку, в частности, на каких библиотеках она основана. Выполнение npm install в директории, содержащей такой файл, автоматически приводит к установке всех зависимостей, и в свою очередь их зависимостей. Также инструмент npm используется для размещения библиотек в онлайновом хранилище NPM, чтобы другие люди могли их находить, скачивать и использовать.

1.2.5 Потоки

Потоки с возможностью записи – популярная концепция в интерфейсах Node. У всех потоков есть метод write, которому можно передать строку или объект Buffer. Метод end закрывает поток, а при наличии аргумента, выведет перед закрытием кусочек данных. Обоим методам можно задать функцию обратного вызова через дополнительный аргумент, которую они вызовут по окончанию записи или закрытию потока.

Возможно создать поток, показывающий на файл, при помощи функции fs.createWriteStream. Затем можно использовать метод write для записи в файл по кусочкам, а не целиком, как в fs.writeFile.

Потоки с возможностью чтения будут чуть сложнее. Как переменная request, переданная функции для обратного вызова в сервере HTTP, так и переменная response, переданная в HTTP-клиенте, являются потоками с возможностью чтения. (Сервер читает запрос и потом пишет ответы, а клиент пишет запрос и читает ответа). Чтение из потока осуществляется через обработчики событий, а не через методы.

У объектов, создающих события в Node, есть метод on, схожий с методом браузера addEventListener. Вы даёте ему имя события и функцию, и он регистрирует эту функцию, чтоб её вызвали сразу, когда произойдёт событие.

У потоков с возможностью чтения есть события «data» и «end». Первое происходит при поступлении данных, второе – по окончанию. Эта модель подходит к потоковым данным, которые можно сразу обработать, даже если получен не весь документ. Файл можно прочесть в виде потока через fs.createReadStream.

После получения знаний, которые описаны выше, выходит, что Node – отличная простая система, позволяющая запускать JavaScript вне браузера. Изначально она разрабатывалась для работы по сети, чтобы играть роль узла в сети. Но она позволяет делать много всего, и если вы наслаждаетесь программированием на JavaScript, автоматизация ежедневных задач с Node работает отлично.

NPM предоставляет библиотеки для всего, что вам может прийти в голову, и она позволяет скачивать и устанавливать их простой командой. Node также поставляется с набором встроенных модулей, включая «fs» для работы с файловой системой, и «http» для запуска HTTP серверов и создания HTTP запросов.

Весь ввод и вывод в Node делается асинхронно, если только вы не используете явно синхронный вариант функции, например, fs.readFileSync. Вы предоставляете функции обратного вызова, а Node их вызывает в нужное время, когда операции I/O заканчивают работу.

1.2.6. База данных

SQL - базы данных с открытым кодом.

MySQL и PostgreSQL - самые известные представители этой группы - оптимальный выбор для большинства задач. Обе поддерживают кластеризацию, есть примеры использования в больших проектах и даже миграции с одной на другую в больших проектах. Пожалуй, основным минусом для нас является ручной шардинг и, как следствие, отсутствие автоматической ребалансировки кластера.

В нашей системе в качестве ключа шардинга - параметра, по которому определяется на каком сервере кластера хранить элемент данных, - естественно выбрать организацию (группу пользователей). Однако некоторые организации остаются маленькими - 1-2 пользователя, а другие по мере работы в сервисе вырастают до десятков тысяч пользователей. Распределение нагрузки по такому ключу рано или поздно приведет к переполнению одних серверов в кластере и недозагруженности других. В этот момент потребуется ребалансировка - то есть разделение ноды кластера на две. Эту работу сложно делать на работающем 24x7 кластере без потери надежности.  
NoSQL - базы данных

Модное в 2000-е годы движение NoSQL сейчас переживает период зрелости. Все игроки хорошо известны и обладают своими сторонниками. Созданные при бурном росте интернета, эти БД развивались для соответствующих задач, например, для [хранения и обработки миллиардов неструктурированных документов](https://medium.com/@markobonaci/the-history-of-hadoop-68984a11704). Многие решения [декларируют](https://mariadb.org/eventually-consistent-databases-state-of-the-art/) «eventual consistency», что означает отказ от строгого «C» в CAP-теореме. Мы не можем терять данные клиентов, поэтому для нас такой компромисс неприемлем.

Некоторые NoSQL - решения снижают доступность («A») и декларируют «CP», например, Cassandra. Это подходит для наших задач, однако мы были удивлены [отсутствием row-level consistency](http://datanerds.io/post/cassandra-no-row-consistency/): две совпавших по времени записи в разные колонки одной строки могут привести к порче данных. И хотя такого уровня глюков не ожидаешь от БД, вокруг этой

проблемы можно найти обходной путь (например, модифицировать строки только целиком), и мы взяли Cassandra.

Облачные базы данных.

Про эту категорию можно написать отдельный обзор. У каждого из основных PaaS-игроков (Amazon, Google и Microsoft) есть 6-8 разных предложений для хранения структурированных данных (и еще много сервисов для хранения BLOBS). Под любой тип нагрузки можно подобрать готовое решение.

Лучше отказались от облачных хранилищ по соображениям хранения персональных данных. Обычно клиенты находятся в разных странах, а ни один сервис не предлагает хранение ПДн во всех странах мира в соответствии с локальным законодательством. Другой причиной была сильная зависимость от конкретного вендора - вы не можете взять их технологию и развернуть на своем железе. Если появится желание уйти от вендора (при повышении цен или снижении надежности), проект миграции может быть очень долгим. [У Dropbox ушло более 2 лет](https://www.wired.com/2016/03/epic-story-dropboxs-exodus-amazon-cloud-empire/)на переезд из облака Amazon в собственное хранилище.

Подходящая база данных. Почти идеальный вариант будет NoSQL-база данных из-за того, что это прототип и скорее всего тип данных будет очень часто меняться как и сами данные.

Одна из популярных баз данных является MongoDB и ORM к ней для Node.JS это Mongoose, по которой есть отличная документация [https://mongoosejs.com/docs/api.html](https://mongoosejs.com/docs/api.html#Document)

1.3 Выбор основного фреймворка для клиентской части приложения

Фреймворки JavaScript развиваются очень быстрыми темпами, поэтому сегодня у нас есть часто обновляемые версии Angular и ReactJS, а также версии нового игрока на этом рынке  - Vue.js.

Проанализировав количество открытых вакансий во всем мире, которые требуют конкретных знаний определенного фреймворка. В качестве источника взяв сайт Indeed.com можно получить следующие данные среди более 60 000 вакансий.

Рис. 1.2 - Графическое соотношение вакансий

Беря во внимание полученные данные, можно выделить основные преимуществами и недостатки каждого фронт-енд фреймворка и, таким образом, выбрать наиболее подходящий из них для своих нужд.

Плюсы и минусы Angular 5

Angular  - это такой супергеройский фреймворк MVVM JavaScript, основанный в 2009 году, который идеально подходит для создания интерактивных веб-приложений.

Преимущества Angular 5:

* новые функции, такие как улучшенный RXJS, ускоренная компиляция (менее 3 секунд) и новый лаунчер HttpClient;
* подробная документация, которая позволяет каждому разработчику получить всю необходимую информацию без обращения за помощью к коллегам. Тем не менее, обучение требует немалого времени;
* двусторонняя привязка данных, которая обеспечивает превосходное поведение приложения, что минимизирует риски возможных ошибок;
* MVVM (Model-View-ViewModel), которая позволяет разработчикам раздельно работать в одном разделе приложения с использованием одного и того же набора данных;
* внедрение зависимостей функций связанных с компонентами с модулями и модульности в целом.

Недостатки Angular 5:

* сложный синтаксис, который исходит от первой версии Angular. Тем не менее, Angular 5 использует TypeScript 2.4, который изучить не так уж сложно;
* проблемы с миграцией, которые могут возникнуть при переходе от старой версии к новой;
* компании, которые используют Angular 5: Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase.

Плюсы и минусы ReactJS

ReactJS  -  это библиотека JavaScript, исходный код которой был открыт Facebook в 2013 году. Этот фреймворк отлично подходит для создания огромных веб-приложений, где данные могут меняться на регулярной основе.

Преимущества ReactJS:

* легок в изучении. React гораздо легче учится ввиду простоты его синтаксиса. Инженеры просто должны вспомнить свои навыки написания HTML и всё на этом. Нет нужды в глубоком изучении TypeScript, как в случае с Angular;
* высокий уровень гибкости и максимальная отзывчивость;
* виртуальная DOM (document object model), которая позволяет упорядочивать документы форматов HTML, XHTML или XML в дерево, которое лучше всего подходит веб-браузерам для анализа различных элементов веб-приложения;
* в сочетании с ES6/7 ReactJS может легко работать при высоких нагрузках;
* связывание данных от больших к меньшим. Это означает такой поток данных, при котором дочерние элементы не могут влиять на родительские данные;
* 100%-ая JavaScript-библиотека с открытым исходным кодом, которая получает множество ежедневных обновлений и улучшений в соответствии с отзывами разработчиков по всему миру;
* невероятно легкий вес, так как данные, которые выполняются на стороне пользователя, могут легко быть представлены на стороне сервера в то же самое время;
* миграция между версиями, как правило, очень проста. Также Facebook предоставляет «codemods» для автоматизации большей части этого процесса.

Недостатки ReactJS:

* нехватка официальной документации. Сверхбыстрая разработка ReactJS не оставляет места для правильной документации, которая сейчас немного хаотична, так как многие разработчики вносят в неё индивидуальные изменения без какого-либо систематического подхода;
* ReactJS не имеет чёткой цели. Это означает, что разработчики, иногда, имеют слишком большой выбор;
* долгое время для освоения. React JS требует глубокого понимания того, как интегрировать пользовательский интерфейс в структуру MVC.

Компании, которые используют ReactJS: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft.

Плюсы и минусы Vue.js

Vue.js  -  это фреймовк JavaScript, запущенный в 2013 году, который идеально подходит для создания адаптируемых пользовательских интерфейсов и сложных одностраничных приложений.

Преимущества Vue.js:

* усиленный HTML. Это означает, что Vue.js имеет много схожих с Angular характеристик. Это может помочь оптимизировать обработку HTML-блоков с использованием разных компонентов;
* подробная документация. Vue.js имеет очень хорошую документацию, которая может увеличить скорость обучения разработчиков и сэкономить много времени на разработку приложения с использованием базовых знаний HTML и JavaScript;
* адаптивность. Vue.js обеспечивает быстрый период перехода от других фреймворков к Vue.js в виду его сходства с Angular и React с точки зрения дизайна и архитектуры;
* восхитительная интеграция. Vue.js можно использовать как для создания одностраничных приложений, так и для более сложных веб-интерфейсов приложений. Самое важное, что небольшие интерактивные части можно легко интегрировать в существующую инфраструктуру, не оказывая при этом отрицательного влияния на всю систему;
* большое масштабирование. Vue.js помогает разрабатывать довольно крупные шаблоны для многократного использования, которые можно разработать без траты огромного количества времени в виду простой структуры;
* крошечный размер. Vue.js может весить около 20 КБ и при этом сохранять свою свою скорость и гибкость, что позволяет достичь гораздо более высокой производительности, по сравнению с другими фреймворками.

Недостатки Vue.js:

* риск чрезмерной гибкости. Иногда у Vue.js могут возникать проблемы при интеграции в огромные проекты, а опыта о возможных решениях до сих пор нет. Но они обязательно появятся в ближайшее время.

Компании, которые используют Vue.js: Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab and Laracasts, Adobe, Behance, Codeship, Reuters.

В итоге можно понять, что для настоящего инженера нет существенной разницы в том, какой фреймворк выбрать, так как для обучения новому требуется всего немного времени. Каждый фреймворк имеет свои плюсы и минусы. Это значит, что для каждого отдельного случая ваш выбор фреймовка может отличаться.

Но так как Vue имеет самый низкий порог входа, был выбран Vue.

1.4 Выбор основного фреймворка для серверной части приложения

Фреймворки Node.js пользуются огромным спросом на рынке, а в 2019 году они приобрели еще больше функций и преимуществ. На рынке доступно множество ведущих языков программирования, однако лучшие фреймворки Node.js 2019 года кардинально изменили процесс разработки.

Но прежде чем перейти к подробному обсуждению, разберемся в том, что такое Фреймворк Node

Node.js  -  это кросс-платформенная среда выполнения JavaScript с открытым исходным кодом, которая выполняет код JavaScript вне браузера.

В основном JavaScript используется для создания сценариев на стороне клиента, которые встроены в HTML-код страницы веб-сайта и запускаются движком JavaScript в браузере.

С помощью Node.js разработчики могут использовать JavaScript для получения инструментов командной строки. На стороне сервера он запускает необходимые сценарии для обработки динамического содержимого веб-страницы, прежде чем она будет доступна в браузере пользователя. Таким образом, Node.js олицетворяет собой концепцию «JavaScript везде», позволяя разрабатывать веб-приложения на одном языке, как для сценариев со стороны сервера, так и со стороны клиента.

Основные преимущества Фреймворков Node.js

Такие особенности, как лучшая производительность, функциональность, высокая скорость и масштабируемость, делают Node.js

выбором номер один для разработки приложений корпоративного уровня для крупных компаний.

С Node.js можно использовать один и тот же язык как для фронтэнда, так и для бэкэнда. Это избавит вас от необходимости изучения и применения новых ЯП.

Некоторые из основных преимуществ:

* функции с высокой скоростью;
* поддержка потоковой передачи данных;
* работа в режиме реального времени;
* имеют решения для всех запросов к базе данных;
* простота в написании кода;
* открытый исходный код;
* кросс-платформенность;
* работа с прокси-сервером;
* высокая производительность;
* решение проблем синхронизации;
* дружелюбное сообщество.

Рассмотрим ведущие фреймворки Node.js, которые будут популярны как в 2019 году, так и в ближайшие годы:

1. [AdonisJs](https://adonisjs.com/)



Рис. 1.4 - AdonisJs

AdonisJs  - один из самых популярных фреймворков Node.js, работающий на всех основных операционных системах. Он обладает статической экосистемой для написания серверных веб-приложений. Таким образом, можно выбрать подходящий пакет, ориентируясь на определенные бизнес-потребности.

Особенности AdonisJs:

* поддержка ORM, состоящего из баз данных SQL;
* эффективное создание SQL-запроса на основе активной записи;
* легкий в освоении конструктор запросов для создания простых и быстрых запросов;
* обеспечение поддержки баз данных No-SQL, таких как MongoDB.

2. [Express.js](https://expressjs.com/)

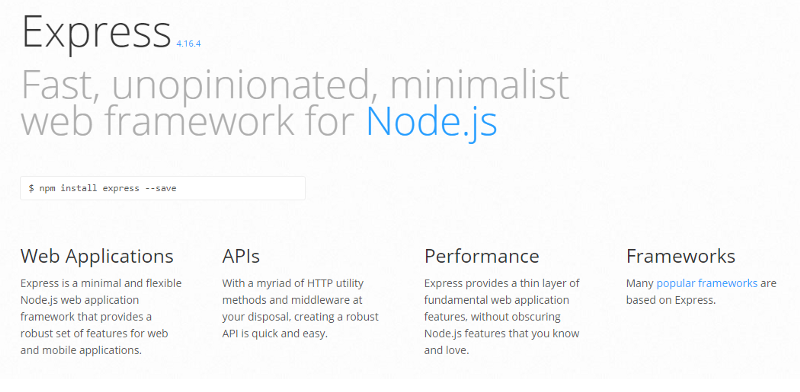


Рис. 1.5 - Express.js

Express.js  -  это самый простой и быстрый фреймворк Node.js, который используется в качестве промежуточного обработчика для управления серверами и маршрутами.

Express.js подходит для разработки простых приложений, которые могут обрабатывать несколько запросов одновременно и опираются на возможности технологии Express.

Особенности Express.js:

* полностью настраиваемый;
* низкая кривая обучаемости;
* ориентирован на браузер.

3. [Hapi.js](https://hapijs.com/)

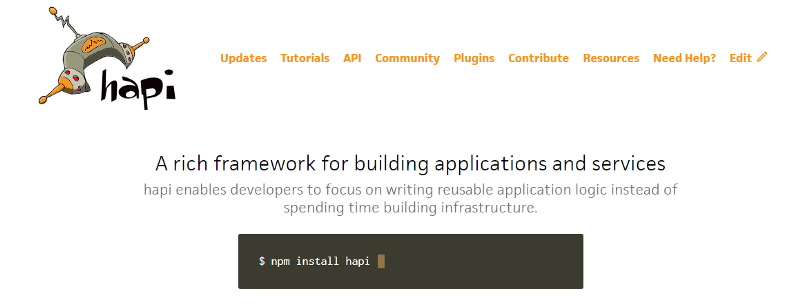


Рис. 1.12 - Hapi.js

Hapi.js  -  лучший веб-фреймворк Node.js, который используется для разработки интерфейсов прикладных программ. Благодаря сильной системе плагинов Hapi.js, можно полностью управлять процессом разработки.

Особенности Hapi.js

* мощная валидация входных данных;
* функциональность;
* реализация кэширования;
* улучшенная система обработки ошибок.

4. [Meteor.js](https://www.meteor.com/)

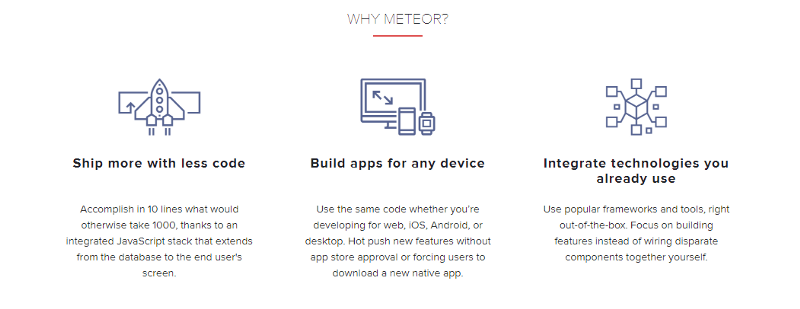


Рис. 1.13 - Meteor.js

Meteor.js используется для создания современных веб- и мобильных приложений и представляет собой full-stack платформу JavaScript. Одна из самых важных особенностей Meteor.js  - обеспечение обновления в режиме реального времени, означающее, что все изменения в сети будут мгновенно обновлены в шаблоне.

Особенности Meteor.js:

* возможность управления крупными проектами;
* большое и организованное сообщество по разработке документации;
* использование стека данных Facebook GraphQL;
* легок в понимании для большинства разработчиков.

5. [Sails.js](https://sailsjs.com/)

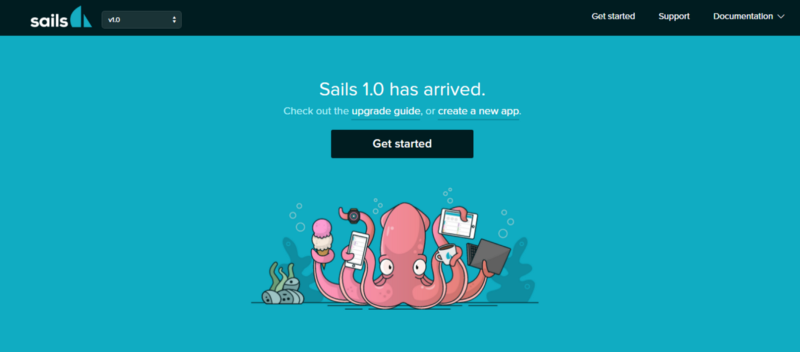


Рис. 1.14 - Sails.js

Sails.js  -  еще один популярный фреймворк Node.js, который используется для разработки пользовательских приложений Node.js корпоративного уровня. Sails.js состоит из data-driven API-интерфейсов, подключенных к масштабируемой сервис-ориентированной архитектуре.

Особенности Sails.js:

* Нет необходимости в дополнительной маршрутизации;
* Совместимость внешнего интерфейса с различными технологиями внешнего интерфейса;
* Чистая поддержка протоколов WebSocket;
* Совместимость со всеми базами данных.

6. [Koa.js](https://koajs.com/)

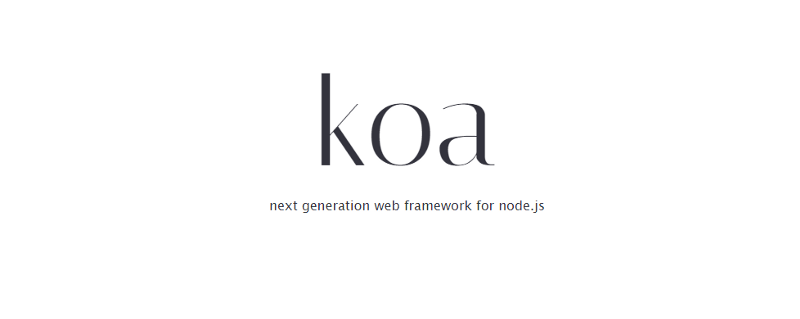


Рис. 1.15 - Koa.js

Команда создателей Express.js также выступила в качестве разработчиков Koa.js. Целью разработки послужило восполнение недостатков Express.js. Благодаря уникальности сценария и методов, Koa может работать в разных браузерах. Также он работает без использования обратных вызовов и обладает мощной системой обработки ошибок.

Особенности Koa.js:

* использование необходимых генераторов для управления и обработки обратных вызовов;
* эффективные процедуры по обработке ошибок;
* блоки на основе компонентов;
* каскадная модель промежуточного ПО и отсутствие ада обратных вызовов.

7. [LoopBack.js](https://loopback.io/)

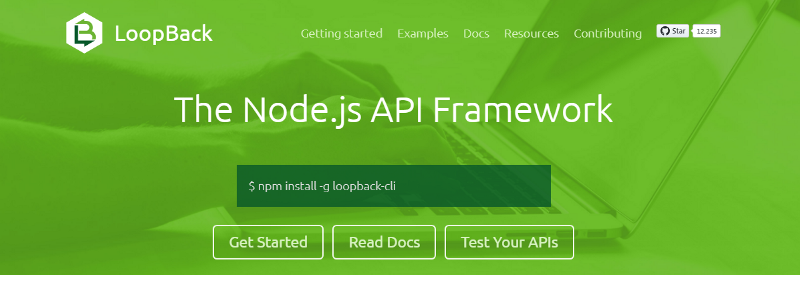


Рис. 1.16 - LoopBack.js.

LoopBack.js  -  еще один широко распространенный фреймворк Node.js с удобным интерфейсом командной строки и динамическим API explorer. С его помощью можно создавать различные модели в зависимости от требуемой схемы (или даже при отсутствии схемы). Он обладает хорошей совместимостью с различными службами REST и типами баз данных, включая MySQL, MongoDB, Oracle, Postgres и многие другие.

Особенности LoopBack.js:

* Быстрое создание динамических сквозных REST API;
* Лучшее соединение между различными устройствами и браузерами;
* Улучшенная корреляция между различными типами данных и сервисами;
* Использование средств разработки Android, iOS и Angular для создания клиентских приложений;
* Работает как в on-premises, так и в облаке.

8. [Derby.js](https://derbyjs.com/)

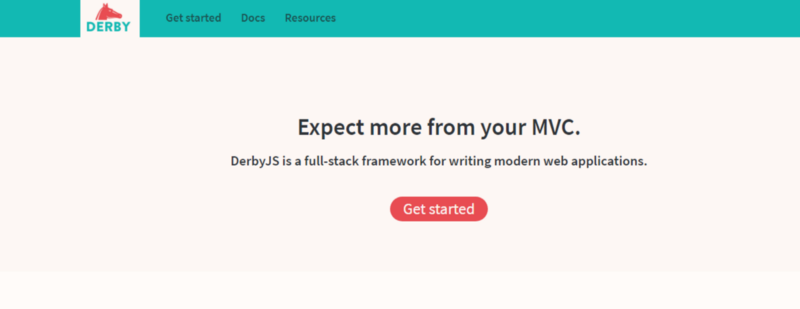


Рис. 1.17 - Derby.js

Derby.js обеспечивает бесперебойную синхронизацию данных между сервером и клиентом. Derby.js хорошо известен как full-stack фреймворк Node.js для написания современных веб-приложений. С его помощью можно создавать настраиваемый код и разрабатывать высокоэффективные веб-приложения. Благодаря своим уникальным характеристикам, Derby.js набирает популярность в 2019 году.

Особенности Derby.js:

* шаблон MVC как для клиентской, так и для серверной стороны;
* идеально подходит для разработки мобильных и веб-приложений;
* использование серверного рендеринга для быстрой загрузки страниц, HTML-шаблонов и поддержки поисковых систем.

9. Total.js



Рис. 1.18 - Total.js

Total.js требует минимального обслуживания и обеспечивает высокую производительность и безупречное масштабирование. Команда Total.js старается соответствовать требованиям пользователей, чтобы сделать платформу удобной и привлекательной для пользователей по всему миру. По этой причине, у Total.js есть все шансы набрать популярность в ближайшие годы.

Особенности Total.js:

* Архитектура программного обеспечения по схеме: модель-представление-контроллер;
* Расширяющийся и асинхронный фреймворк;
* Обеспечивает полную поддержку механизма маршрутизации RESTful;
* Полная поддержка протоколов websocket и потоковой передачи мультимедиа.

10. [Nest.js](https://nestjs.com/)

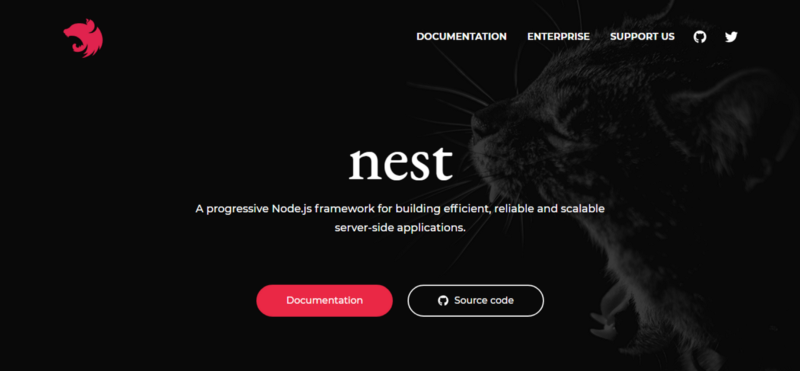


Рис. 1.19 - Nest.js

Nest.js  -  один из типов фреймворков Node.js, который используется для разработки профессиональных и масштабируемых серверных приложений Node.js. Он был написан на TypeScript, но использует JavaScript. Поскольку он разработан с помощью TypeScript, то сочетает в себе элементы

объектно-ориентированного программирования, функционального программирования и функционального реактивного программирования.

Особенности Nest.js:

* Архитектура приложений «из коробки»;
* Легкость в создании тестируемых и масштабируемых приложений высокого качества;
* Генерация приложений Nest.js с помощью Nest CLI.

Как выбрать фреймворк Node.

Выбрать один из множества фреймворков Node.js  -  достаточно трудное решение. Оно зависит исключительно от требований проекта и бизнеса. Фреймворки Node.js обладают различными особенностями, начиная от скорости, кривой обучаемости, структуры написания кода, гибкости, конфигурации и т. д.

В современном цифровом мире происходит широкое распространение технологий. Возрастает уровень конкуренции между различными технологиями и фреймворками. На рынке доступно огромное количество фреймворков Node.js, нужно лишь выбрать соответствующий именно вашим требованиям.

Взглянув на процентное соотношения использования различных фреймоворков серверной стороны, можно сказать то что одним из лидеров является Koa.js, он и будет выбран для серверной стороны приложения.

1.5 Дополнительные технологии для Vue.

Сам по себе Vue дает только реактивность данных на сайте, но роуты, хранение данных, мета информацию дают отдельные модули, который почти всегда будет установлены вместе с ним, а это:

* «lodash»: это библиотека JavaScript, которая предоставляет вспомогательные функции для общих задач программирования с использованием парадигмы функционального программирования;
* «register-service-worker»: Скрипт для упрощения регистрации работника сервиса с помощью хуков для общих событий;
* «mockjs»: это генератор данных для моделирования, который помогает внешнему интерфейсу разрабатывать и создавать прототипы отдельно от внутреннего процесса и сокращает монотонность, особенно при написании автоматических тестов;
* «vue-cal»: Полный календарь Vue JS, без зависимостей;
* «vue-router»: официальный маршрутизатор для Vue.js. Он глубоко интегрируется с ядром Vue.js, что позволяет без проблем создавать одностраничные приложения с Vue.js.

Особенности включают в себя:

* «vuex»: это шаблон управления состоянием + библиотека для приложений Vue.js. Он служит централизованным хранилищем для всех компонентов приложения с правилами, гарантирующими, что состояние может быть изменено только предсказуемым образом;
* «webpack»: это статический модульный сборщик для приложений на JavaScript;
* «axios»: клиент на основе обещаний для браузера и node.js.

1.6 Дополнительные технологии для Koa.js.

* «koa/cors»: настраивает заголовок CORS Access-Control-Allow-Origin;
* «aws-sdk»: пакет AWS SDK для JavaScript позволяет получать прямой доступ к сервисам AWS из кода JavaScript, исполняемого в браузере. Храните данные приложений в Amazon DynamoDB и сохраняйте файлы пользователей в Amazon S3;
* «axios»: клиент на основе обещаний для браузера и node.js;
* «bcryptjs»: оптимизированый bcrypt в JavaScript с нулевыми зависимостями. Совместим с привязкой bcrypt C ++ к node.js, а также работает в браузере;
* «cross-env»: делает так, что вы можете иметь одну команду, не беспокоясь о настройке или правильном использовании переменной среды для платформы;
* «joi»: позволяет создавать чертежи или схемы для объектов JavaScript (объекта, хранящего информацию), чтобы обеспечить проверку ключевой информации;
* «jsonwebtoken»: позволяет использовать абстракции для JWT авторизации;
* «koa-body»: полнофункциональное промежуточное ПО анализатора тела;
* «koa»: поддерживает многочастные, urlencoded и json тела запросов;
* «koa-combine-routers»: удобное промежуточное ПО для составления нескольких экземпляров коа-router;
* «koa-cookie»: промежуточное программное обеспечение парсера cookies для коа;
* «koa-helmet»: это обертка для шлема для работы с Коа. Он предоставляет важные заголовки безопасности, чтобы сделать ваше приложение более безопасным по умолчанию;
* «koa-logger»: логирование запросов для koa;
* «koa-morgan»: использование логов от запросов для koa для хранения в разных представлениях;
* «koa-router»: маршрутизация внутри запросов koa;
* «koa-static»: коа статический файл, обслуживающий промежуточное ПО, оболочка для коа-send;
* «koa2-history-api-fallback»: fallback для api;
* «koa2-swagger-ui»: позволяет разместить пользовательский интерфейс swagger в указанный каталог из приложения koa;
* «mongoose»: мongoose предоставляет простое решение на основе схем для моделирования данных вашего приложения