**Раздел 2. Обзор библиотек, онлайн-сервисов и инструментов необходимых для создания сайта предприятия**

**О Figma.**

Figma — онлайн-редактор, в котором удобно проектировать интерфейсы, создавать макеты сайтов, мобильных приложений, презентации, иллюстрации, логотипы и анимацию. В основном инструментом пользуются дизайнеры, но продакт-менеджерам и разработчикам тоже полезно разбираться в программе. Так участникам проекта будет проще понимать друг друга и работать над продуктом.

Особенности программы:

* Макет можно смотреть и редактировать онлайн, изменения видны сразу всем участникам. Менеджеру или разработчику не нужно дёргать дизайнера и спрашивать, что происходит с проектом. Они заходят в программу и видят последние изменения.
* Открывается через браузер, устанавливать дополнительные приложения не нужно. Это удобно, когда нужно показать макет клиенту или коллегам. Достаточно отправить ссылку и настроить права доступа.
* В платном тарифе есть дополнительные инструменты для командной работы. Например, голосовой чат — участники проекта могут созвониться в редакторе, обсудить текущий макет и поправить его. Не нужно включать демонстрацию экрана и переходить между разными программами.

Конечно, менеджеру по продукту не требуется владеть программой так же хорошо, как дизайнеру. Но основные инструменты графического редактора всё же стоит изучить: это ускорит работу над сайтом или приложением.

[Прототипы](https://education.yandex.ru/journal/chto-takoe-figma#prototipy).

Чтобы не объяснять словами, как должно работать приложение или сайт, можно создать его прототип — модель будущего продукта. Добавить переходы между экранами, анимацию, выпадающее меню — так разработчику или клиенту будет проще представить, как выглядит макет на реальном устройстве. Также это поможет отследить ошибки в сценарии и исправить их до того, как проект уйдёт в разработку.

Интерфейс.

Интерфейс — это внешний вид продукта: кнопки, формы, иконки. Их можно создавать прямо в фигме и добавлять эффекты: создавать анимацию для баннеров, делать кнопки кликабельными.

**О среде разработки Visual Studio Code.**

Visual Studio Code – это продукт от компании Майкрософт. Инструмент, предназначенный для верстальщиков и разработчиков. Один из самых популярных в соответствующей нише.

Представляет собой редактор кода от Microsoft, выступающий «облегченной» интерпретацией VisualStudio. С помощью него можно не только заниматься написанием приложений. Visual Studio Code поддерживает большое количество плагинов, которые позволят «разогнать» редактор до полноценной среды программирования.

Подходит для работы на операционных системах:

* Windows;
* MacOS;
* Linux.

Распространяется на бесплатной основе, благодаря чему набрал огромную популярность. Работает одинаково хорошо как на стареньких устройствах, так и на современных компьютерах.

Microsoft Studio Code – это редактор, который больше подходит новичкам за счет своей первоначальной «облегченности». Данный продукт может использоваться и продвинутыми разработчиками при создании достаточно сложных кодов.

Visual Studio C – это редактор, поддерживающий множество языков. Он подойдет для кроссплатформенной разработки. Активно используется веб-программистами. Поддерживает такие языки как:

* JavaScript;
* HTML;
* CSS;
* PHP;
* Go;
* Ruby;
* Python;
* C#;
* TypeScript.

Рассматриваемый инструмент позволяет работать почти со всеми современными языками программирования. Он также поддерживает подключение расширений и фреймворков наиболее известных – React JS, Vue.js, LESS, SCSS.

Теоретически Visual Studio Code может использоваться для любых целей при разработке программных продуктов. На практике ситуация обстоит иначе. Обычно редактор применяется тогда, когда для создания программы не требуются мощности полноценной IDE.

Отличительной чертой Visual Studio Code является то, что он легко расширяется. Чтобы добавить к стандартным опциям новые, достаточно скачать и установить плагин/дополнение с официального встроенного каталога. Все инструменты, которые поддерживает Визуал Студия Код, как и сам редактор, распространяются совершенно бесплатно.

Visual Studio Code – это приложение, которое умеет многое. Оно поддерживает:

1. Написание и редактирование кода приложения. Это – основное предназначение бесплатного продукта от MS. Пользователи смогут не только написать, но и отредактировать, а также сохранить код. Редактор оснащен функциями корректировки. Он автоматически расставляет в коде отступы для повышения его читаемости. Предусматривает автодополнение, что помогает ускорить разработку программы, особенно на первых порах. Visual Studio Code – это продукт, в котором имеется целая система помощи редактирования исходного кода. Она называется IntelliSense.
2. Автоматическое редактирование. Полезная функция, позволяющая автоматизировать исправление и корректировку исходного кода приложения. Реализовывается через специальные плагины и расширения. С их помощью в Visual Studio Code будут исправлены некоторые незначительные ошибки: неправильные отступы, именование, несоответствие стиля и так далее. Для корректировки сложных неполадок не расширения не подойдут – система не поймет, как все исправить. Но для исправления небольших недочетов у VS Code специальные возможности есть. И работают они очень хорошо.
3. Быструю навигацию. У Visual Studio Code есть строка поиска с поддержкой регулярных выражений. А еще – автодополнение. Также редактор позволяет выделить код и закомментировать его при помощи простого сочетания клавиш.
4. Горячие клавиши. Используются для быстрого применения функций без предварительного входа в меню редактора. Можно воспользоваться встроенными сочетаниями Visual Studio Code или настроить их самостоятельно.
5. Контроль версий. Визуал Студия позволяет сразу из редактора провести откат, добавить комментарий, отправить коммит в удаленный репозиторий. В консоль для этого заходить не придется, как и набирать те или иные команды – все осуществляется через встроенный визуальный интерфейс.
6. Установку дополнений. Это – ключевая особенность Visual Studio Code. Из редактора можно перейти в каталог дополнение и расширений, подобрать необходимый плагин и установить его в несколько кликов. Пример – поддержка различных языков, автоматическое исправление, конфигураторы. Данная опция дает возможность настройки редактора под нужды каждого конкретного разработчика.
7. Отладчик. Он используется для языка JavaScript и базирующихся на нем технологий. Пример – TypeScript. Для остальных языков разработки встроенных отладчиков нет – они устанавливаются в виде плагинов. Интерактивная [отладка](https://otus.ru/journal/otladka-prilozheniya-chto-dolzhen-pomnit-razrabotchik/) пошагово помогает выполнять код, а также на каждом этапе отслеживать изменение данных.

К преимуществам Visual Studio Code относят следующие моменты:

1. Простое освоение. Редактор оснащен интуитивно понятным интерфейсом, а также возможностью настройки. Единственная проблема для новичков – это горячие клавиши, которые организованы необычным образом.
2. Небольшой вес.
3. Кроссплатформенность.
4. Наличие online версии. В случае ее использования для написания программного кода не придется вообще ничего устанавливать. Visual Studio Online работает через браузер. Расположен редактор [здесь](https://vscode.dev/).
5. Поддержка разных языков. Опция активируется за счет установки плагинов.
6. Гибкость.
7. Данная технология позволяет работать со строками (lines), а также приводить исходный код к единому стилю в несколько кликов. Ей легко пользоваться, но недостатки у VS Code тоже есть.

К минусам Visual Studio Code относят:

1. Ограниченную функциональность, особенно в сравнении с полноценной IDE от Майкрософт.
2. Медленную работу, особенно на старых устройствах. Связано это с тем, что редактор написан на Election.

**О HTML и CSS.**

HTML (HyperText Markup Language) представляет язык разметки гипертекста, используемый преимущественно для создания документов в сети интернет. HTML начал свой путь в начале 90-х годов как примитивный язык для создания веб-страниц, и в настоящий момент уже трудно представить себе интернет без HTML. Подавляющее большинство сайтов так или иначе используют HTML.

В 2014 году официально была завершена работа над новым стандартом - HTML5, который фактически произвел революцию, привнеся в HTML много нового.

Что именно привнес HTML5?

* HTML5 определяет новый алгоритм парсинга для создания структуры DOM
* добавление новых элементов и тегов, как например, элементы video, audio и ряд других
* переопределение правил и семантики уже существовавших элементов HTML

Фактически с добавлением новых функций HTML5 стал не просто новой версией языка разметки для создания веб-страниц, но и фактически платформой для создания приложений, а область его использования вышла далеко за пределы веб-среды интернет: HTML5 применяется также для создания мобильных приложений под Android, iOS, Windows Mobile и даже для создания десктопных приложений для обычных компьютеров (в частности, в ОС Windows 8/8.1/10).

В итоге, как правило, HTML 5 применяется преимущественно в двух значениях:

* HTML 5 как обновленный язык разметки гипертекста, некоторое развитие предыдущей версии HTML 4
* HTML 5 как мощная платформа для создания веб-приложений, которая включает не только непосредственно язык разметки гипертекста, обновленный HTML, но и язык программирования JavaScript и каскадные таблицы стилей CSS 3.

Кто отвечает за развитие HTML5? Этим занимается World Wide Web Consortium (сокращенно W3C - Консорциум Всемирной Паутины) - независимая международная организация, которая определяет стандарт HTML5 в виде спецификаций. Текущую полную спецификацию на английском языке можно посмотреть по адресу <https://www.w3.org/TR/html5/>. И надо отметить, что организация продолжает работать над HTML5, выпуская обновления к спецификации.

Поддержка браузерами

Надо отметить, что между спецификацией HTML5 и использованием этой технологии в веб-браузерах всегда был разрыв. Большинство браузеров стало внедрять стандарты HTML5 еще до их официальной публикации. И к текущему моменту большинство последних версий браузеров поддерживают большинство функциональностей HTML5 (Google Chrome, Firefox, Opera, Internet Explorer 11, Microsoft Edge). В то же время многие старые браузеры, как например, Internet Explorer 8 и более младшие версии, не поддерживают стандарты, а IE 9, 10 поддерживает лишь частично.

При этом даже те браузеры, которые в целом поддерживают стандарты, могут не поддерживать какие-то отдельные функции. И это тоже надо учитывать в работе. Но в целом с поддержкой данной технологии довольно хорошая ситуация.

Для проверки поддержки HTML5 конкретным браузером можно использовать специальный сервис [http://html5test.com](http://html5test.com/).

**О JavaScript.**

JavaScript появился в 1995 году как способ программирования веб-страниц в браузере Netscape Navigator. С тех пор язык был принят во всех остальных ведущих графических браузерах. Это сделало возможным применение современных веб-приложений - приложений, с которыми можно взаимодействовать напрямую, не перезагружая страницу при каждом действии.

JavaScript также используется на более традиционных сайтах для выполнения различных интерактивных «умных» действий.

Важно отметить, что JavaScript никогда не имел ничего общего с языком программирования под названием Java. Похожее название было продиктовано не здравым смыслом, а маркетинговыми соображениями. Когда появился JavaScript, язык Jаvа активно продавался и приобретал популярность. Кто­то решил, что будет хорошей идеей бесплатно прокатиться на волне чужого успеха. Теперь приходится за это расплачиваться.

После того как JavaScript прижился за пределами Netscape, был написан стандартный документ. Он оговаривал, как должен работать JavaScript, чтобы различные программы, утверждающие, что поддерживают JavaScript, в действительности имели в виду один и тот же язык. Это так называемый стандарт ECMAScript, по названию организации Ecma International, которая выполнила стандартизацию. На практике термины ECMAScript и JavaScript взаимозаменяемы - представляют собой два названия одного и того же языка.

Идея не типизированности JavaScript в том, чтобы облегчить программирование нa JavaScript для новичков. В действительности это главным образом затрудняет поиск проблем в программах, поскольку система не указывает вам на них.

Однако у такой гибкости есть свои преимущества. Она оставляет место для множества приемов, которые невозможны в более жестких языках, и, как мы увидим (например, в главе 10), ее можно использовать для преодоления отдельных недостатков JavaScript.

JavaScript претерпел несколько версий. Версия ECMAScript 3 широко поддерживалась во времена, кoгдa JavaScript только находился на пути к своему господству, примерно между 2000 и 2010 годами. В это время велась работа над амбициозной версией 4, в которой планировался ряд радикальных улучшений и расширений языка. Такое радикальное изменение живого, широко используемого языка оказалось политически трудным, и в 2008 году работы над версией 4 были прекращены, что привело к появлению в 2009 году гораздо менее амбициозной версии 5, которая внесла лишь отдельные бесспорные улучшения. Затем в 2015 году вышла версия 6, серьезное обновление, включающее в себя некоторые идеи, запланированные для версии 4. С тех пор каждый год появляются небольшие обновления.

Тот факт, что язык развивается, означает, что браузеры должны постоянно идти с ними в ногу или они могут не поддерживать все функции. Разработчики языка стараются не вносить никаких изменений, которые могли бы нарушить работоспособность существующих программ, поэтому старые программы на новых браузерах по-прежнему будут работать.

Браузеры не единственные платформы, где применяется JavaScript. Некоторые базы данных, такие как MongoDB и CouchDB, используют JavaScript в качестве языка сценариев и запросов. Ряд платформ для программирования настольных ПК и серверов, в частности проект Node.js, предоставляют среду для программирования нa JavaScript вне браузера. [5]

**О TypeScript.**

TypeScript — это язык будущих веб- и мобильных приложений, проектов NodeJS и IoT (систем интернет-управления устройствами). Он позволяет создавать более безопасные программы, обеспечивать их документацией, полезной и вам, и будущим инженерам, поддерживает безболезненный рефакторинг, а также избавляет от необходимости проводить половину модульных, или юнит-тестов.

Программы — это файлы, содержащие прописанный вами текст. Специальная программа — компилятор — считывает и преобразует ваш текст в абстрактное синтаксическое дерево (АСД). Оно представляет собой структуру данных, игнорирующую пустые области, комментарии и ваше ценное мнение о пробелах или табуляции. Затем компилятор преобразует АСД в низкоуровневую форму — байт-код, который можно запустить в среде выполнения и получить результат. Итак, когда вы запускаете программу, фактически вы просите среду выполнения считать байт-код, сгенерированный компилятором на основе АСД, полученного из исходного кода. Детали этого процесса могут отличаться, но для большинства языков он выглядит так:

1. Программа преобразуется в АСД.

2. АСД компилируется в байт-код.

3. Байт-код считывается средой выполнения.

Особенность TypeScript в том, что вместо компиляции прямо в байт-код он компилирует в код JavaScript. Затем вы просто запускаете его в браузере с NodeJS.

В проверке типов заключена магия TypeScript. С ее помощью компилятор (TSC) убеждается, что программа работает так, как вы ожидаете.

Итак, если мы добавим проверку типов и преобразование в JavaScript, то процесс компиляции TypeScript будет выглядеть примерно так (рис. 1):

|  |  |
| --- | --- |
| TypeScript | 1. Код на TS -> TypeScript АСД  2. АСД проверяется проверкой типов  3. TypeScript АСД -> Код на JS |
| JavaScript | 4. Код на JS -> JavaScript АСД  5. АСД -> Байт-код  6. Байт-код оценивается по времени выполнения |

**Рис. 1** Компиляция и запуск TypeScript.

Шаги 1-3 производятся компилятором, а шаги 4-6 — средой выполнения JavaScript, находящейся в вашем браузере, или NodeJS, или любым другим JavaScript-движком.

В течение всего процесса шаги 1-2 используют типы программы, а шаг 3 уже этого не делает. Стоит еще раз повториться: когда TSC компилирует код в JavaScript, он не будет смотреть на типы. Это означает, что типы никогда не смогут повлиять на сгенерированный вывод и будут использованы только для проверки типов. Эта особенность позволяет безопасно с ними экспериментировать — обновлять и улучшать их без риска сломать приложение.JavaScript-компиляторы и среды выполнения, как правило, представляют собой единую программу, называемую движком. Будучи программистом, с ним вы и будете взаимодействовать. Так работают V8 (движок, лежащий в основе NodeJS, Chrome и Opera), SpiderMonkey (Firefox), JSCore (Safari) и Chakra (Edge). Именно поэтому JavaScript и называют интерпретируемым языком.[6]

**О React.**

React — популярная библиотека, используемая для создания пользовательских интерфейсов. Она была создана в Facebook с целью решить ряд проблем, связанных с крупномасштабными сайтами, управляемыми данными. В момент выпуска библиотеки в 2013 году этот проект воспринимался с некоторой долей скептицизма, поскольку соглашения по React довольно необычны.

React действительно является небольшой библиотекой, поставляемой без того набора инструментов, который может понадобиться для создания приложений. Благодаря React при помощи JavaScript создается код, похожий на HTML.

И разумеется, все эти теги требуют предварительной обработки перед запуском в браузере. А для этого, скорее всего, понадобится встроенный инструмент, такой как Webpack.

Если вы, как и мы, читали эту статью, то, возможно, были впечатлены промисами, связанными с новой библиотекой JavaScript, способной решить все проблемы с DOM (объектной моделью документа); с библиотекой, которая всегда будет проста в применении и никогда не причинит неудобств.

А потом начали возникать вопросы: как можно будет преобразовать этот JSX? Как загрузить данные? Куда девать CSS? Что такое декларативное программирование?

В каждом направлении возникало еще больше вопросов о том, как внедрить эту библиотеку в повседневную работу. С каждым разговором появлялась новая терминология, возникали новые технические приемы и все больше вопросов.

Во-первых, стоит заметить, что библиотека невелика по размеру и используется только для одной части всей работы. Она не содержит инструментария, ожидаемого от традиционного фреймворка JavaScript. Основные решения о том, какими средствами экосистемы воспользоваться, принимают разработчики. Кроме того, постоянно появляются новые наборы инструментов, а старые отходят на второй план. При таком количестве библиотек совершенно не удивляет ощущение, что за ними всеми невозможно угнаться.

React пока не утратил своей новизны. Он дошел до стадии стабильности основных функциональных свойств, но даже они еще могут измениться. В следующие версии библиотеки будет включено средство Fiber — новая реализация основного алгоритма React, целью которого является увеличение скорости выдачи изображения на экран. Выстраивать предположения о том, насколько оно повлияет на React-разработчиков, пока рано, но оно, несомненно, отразится на скорости, с которой приложения выводят на экран изображение и обновляют его.

Многие из этих изменений связаны с целевыми устройствами. В данной книге рассматриваются способы разработки одностраничных веб-приложений с помощью React, но не нужно предполагать, что браузеры являются единственным местом, где могут работать такие приложения. Средство React Native, выпущенное в 2015 году, позволяет пользоваться преимуществами React-приложений в программах, предназначенных для работы под iOS и Android. Хотя об этом еще рано говорить, но уже появился фреймворк React VR для создания интерактивных приложений с поддержкой виртуальной реальности как способ разработки 360-градусных интерфейсов с использованием React и JavaScript. [7]

**О Next.js.**

За последние несколько лет веб-разработка сильно изменилась. До появления современных фреймворков JavaScript создание динамических веб-приложений было сложной задачей, и для того, чтобы заставить их работать должным образом, требовалось множество различных библиотек и конфигураций.

Angular, React, Vue и все другие фреймворки позволили сети развиваться очень быстро и принесли с собой несколько очень инновационных идей для разработки веб-интерфейсов.

React, в частности, был создан Джорданом Уолком в Facebook и находился под сильным влиянием библиотеки XHP Hack, которая позволила разработчикам Facebook на PHP и Hack создавать повторно используемые компоненты для интерфейса своих приложений. Библиотека JavaScript стала доступна с открытым исходным кодом в 2013 году и навсегда изменила то, как мы создаем веб-сайты, веб-приложения, нативные приложения (позже с React Native) и даже возможности виртуальной реальности (с React VR). В результате React быстро стала одной из самых любимых и популярных библиотек JavaScript, и миллионы веб-сайтов используют ее в рабочей среде для самых разных целей.

Была только одна проблема: по умолчанию React запускается на стороне клиента (это означает, что он запускается в веб-браузере), поэтому веб-приложение, полностью написанное с использованием этой библиотеки, может негативно повлиять на поисковую оптимизацию (SEO) и начальную производительность загрузки, поскольку для корректного отображения на экране требуется некоторое время. Фактически, чтобы отобразить полное веб-приложение, браузер должен был загрузить весь пакет приложений, проанализировать его содержимое, затем запустить его и отобразить результат в браузере, что могло занять до нескольких секунд (для очень больших приложений).

Многие компании и разработчики начали изучать, как предварительно отрисовать приложение на сервере, позволяя браузеру отображать отрисованное приложение React в виде обычного HTML, делая его интерактивным, как только бандл JavaScript будет передан клиенту.

Затем Vercel придумал Next.js, который, как оказалось, изменил правила игры. С момента своего первого выпуска фреймворк предоставил множество инновационных функций "из коробки", таких как автоматическое разделение кода, рендеринг на стороне сервера, системы маршрутизации на основе файлов, предварительная выборка маршрутов и так далее. Next.js показал, насколько простым должно быть написание универсальных веб-приложений, позволяя разработчикам писать повторно используемый код как для клиентской, так и для серверной части и упрощая реализацию очень сложных задач (таких как разделение кода и рендеринг на стороне сервера).

Сегодня Next.js предоставляет множество новых функций, таких как:

1. Статическая генерация сайтов

2. Инкрементная статическая генерация

3. Поддержка встроенного TypeScript

4. Автоматическое заполнение полей

5. Оптимизация изображений

6. Поддержка интернационализации

7. Анализ производительности

Сегодня Next.js используется в производстве такими компаниями высшего уровня, как Netflix, Twitch, Tik Tok, Hulu, Nike, Rubber, Elastic и многими другими. Если вам интересно, вы можете ознакомиться с полным списком по ссылке [https://nextjs.org/showcase.[8](https://nextjs.org/showcase.%5b8)]

**О Node.js.**

У Node много общего с другими популярными веб-серверами, такими как разработанный Microsoft веб-сервер Internet Information Services (IIS) или Apache. Но интереснее узнать, в чем его отличия. Так что начнем с этого.

Подход Node к веб-серверам чрезвычайно минималистичен. В отличие от IIS или Apache, для освоения которых могут понадобиться годы, Node легок в установке и настройке. Это не значит, что настройка серверов Node на максимальную производительность в условиях промышленной эксплуатации тривиальна, просто конфигурационные опции проще и яснее.

Другое базовое различие между Node и более традиционными веб-серверами — однопоточность Node. Сначала это может показаться шагом назад, но впоследствии становится ясно, что это гениальная идея. Однопоточность чрезвычайно упрощает задачу написания веб-приложений, а если вам требуется производительность многопоточного приложения, можете просто запустить больше экземпляров Node и, в сущности, получить преимущества многопоточности. Дальновидный читатель, вероятно, посчитает это каким-то шаманством. В конце концов, разве реализация многопоточности с помощью серверного параллелизма (в противоположность параллелизму приложений) просто не перемещает сложность в другое место вместо ее устранения? Возможно, но мой опыт говорит о том, что сложность оказывается перемещенной именно туда, где она и должна быть. Более того, с ростом популярности облачных вычислений и рассмотрения серверов как обычных товаров этот подход становится более разумным. IIS и Apache, конечно, мощные веб-серверы, разработанные для того, чтобы выжимать из современного сильнейшего аппаратного обеспечения производительность до последней капли. Это, однако, имеет свою цену: чтобы добиться такой производительности, для их установки и настройки работникам необходима высокая квалификация.

Если говорить о способе написания приложений, то приложения Node больше похожи на приложения PHP или Ruby, чем на приложения .NET или Java. Движок JavaScript, используемый Node (V8, разработанный компанией Google), не только компилирует JavaScript во внутренний машинный код (подобно C или C++), но и делает это прозрачным образом, так что с точки зрения пользователя код ведет себя как чистый интерпретируемый язык программирования. Отсутствие отдельного шага компиляции уменьшает сложность обслуживания и развертывания: достаточно просто обновить файл JavaScript, и ваши изменения автоматически станут доступны.

Другое захватывающее достоинство приложений Node — невероятная независимость Node от платформы. Это не первая и не единственная платформонезависимая серверная технология, но независимо от платформы важнее предлагаемое разнообразие платформ, чем сам факт ее наличия. Например, вы можете запустить приложение .NET на сервере под управлением Linux с помощью Mono, но это очень нелегкая задача ввиду разнородности документации и системной несовместимости. Аналогично можете выполнять PHP-приложения на сервере под управлением Windows, но их настройка обычно не так проста, как на машине с Linux. В то же время Node элементарно устанавливается на всех основных операционных системах (Windows, OS X и Linux) и облегчает совместную работу. Для команд веб-разработчиков смесь PC и компьютеров Macintosh вполне обычна. Определенные платформы, такие как .NET, задают непростые задачи разработчикам и дизайнерам клиентской части приложений, часто использующим компьютеры Macintosh, что серьезно сказывается на совместной работе и производительности труда. Сама идея возможности подключения работающего сервера на любой операционной системе за считаные минуты (или даже секунды!) — мечта, ставшая реальностью.

В сердцевине стека экосисткмы, конечно, находится Node. Это программное обеспечение, которое позволяет выполнять JavaScript-код на сервере без участия браузера, что, в свою очередь, позволяет использовать фреймворки, написанные на JavaScript, такие как Express. Все веб-приложения, кроме самых простых, потребуют базы данных, и существуют базы данных, которые лучше других подходят экосистеме Node.

Ничего удивительного, что имеются интерфейсы для всех ведущих реляционных баз данных (MySQL, MariaDB, PostgreSQL, Oracle, SQL Server): было бы глупо пренебрегать этими признанными китами. Однако наступление эпохи разработки под Node дало толчок новому подходу к базам данных — появлению так называемых NoSQL-баз данных. Не всегда полезно давать чему-либо определение через то, чем оно не является, так что я добавлю, что эти NoSQL-базы данных корректнее было бы называть документоориентированными базами данных или базами данных типа «ключ — значение». Они реализуют более простой с понятийной точки зрения подход к хранению данных. Таких баз данных множество, но MongoDB — одна из лидеров, и именно ее мы будем использовать в книге. [9]

**О Express.js.**

Сайт Express характеризует Express как «минималистичный и гибкий фреймворк для веб-приложений Node.js, обеспечивающий набор возможностей для построения веб- и мобильных приложений».

Достоинства Express:

1. Минималистичный. Один из наиболее привлекательных аспектов Express. Много раз разработчики фреймворков забывали, что лучше меньше, да лучше. Философия Express заключается в обеспечении минимальной прослойки между вами и сервером. Это не говорит о слабой надежности фреймворка или недостаточном количестве его полезных возможностей. Просто он в меньшей степени становится у вас на пути, позволяя более полно выражать свои идеи. Express предоставляет вам минималистичный фреймворк, а вы можете добавлять необходимый функционал в разных его частях, заменяя то, что вам никогда не понадобится. Это глоток свежего воздуха, ведь многие фреймворки создают раздутые, непонятные и сложные проекты еще до того, как вы написали хотя бы одну строчку кода. Зачастую первой задачей становится отсечение ненужного функционала или замена того, который не соответствует требованиям. Express практикует другой подход, позволяя вам добавлять то, что нужно, там, где нужно.

Гибкий. Механизм действия Express очень прост: он принимает HTTP-запрос от клиента (которым может быть браузер, мобильное устройство, другой сервер, десктопное приложение — одним словом, все, что использует HTTP) и возвращает HTTP-ответ. Этот базовый шаблон описывает практически все, связанное с Интернетом, что делает приложения Express чрезвычайно гибкими.

Фреймворк для веб-приложений. Наверное, более точным описанием будет «серверная часть фреймворка для веб-приложений». На сегодня под «фреймворками для веб-приложений» обычно имеются в виду фреймворки одностраничных приложений, такие как React, Angular или Vue. Тем не менее, за исключением небольшой группы отдельных приложений, большинство веб-приложений должно обмениваться данными и взаимодействовать с другими сервисами. Обычно это происходит через веб-API, который можно рассматривать как серверный компонент фреймворка веб-приложений. Обратите внимание, что построение всего приложения с рендерингом только на стороне сервера по-прежнему возможно (а иногда и желательно). В этом случае Express вполне может быть фреймворком всего веб-приложения!

Быстрый. Express, став одним из лучших фреймворков для платформы Node.js, привлек много внимания со стороны больших компаний с высокопроизводительными и сильно нагруженными веб-сайтами. Это вынудило команду по разработке Express сконцентрироваться на производительности, и теперь Express — лидер в этом сегменте.

Некатегоричный. Одними из отличительных особенностей экосистемы JavaScript являются ее размер и разнообразие. Ввиду того что Express зачастую занимает центральную позицию в разработке веб-приложений в Node.js, существуют сотни (если не тысячи) страниц сообщества, где подробно рассматриваются приложения Express. Разработчики Express, в свою очередь, предоставили чрезвычайно гибкую систему разработки промежуточного ПО, которая упрощает использование выбранных вами компонентов при создании приложения. В процессе разработки можно заметить, что Express отключает встроенные компоненты в пользу настраиваемого промежуточного ПО. [9]

**О Nest.js.**

Существует огромное количество доступных веб-фреймворков, а с появлением Node.js их стало еще больше. JavaScript-фреймворки очень быстро входят в моду и выходят из нее по мере развития веб-технологий. Nest.js это хорошая отправная точка для многих разработчиков, которые хотят использовать современный веб-фреймворк, потому что он использует язык, который очень похож на язык JavaScript, наиболее часто используемый в Интернете на сегодняшний день. Многих разработчиков обучали программированию с использованием таких языков, как Java или C/C++, которые являются строгими языками, поэтому использование JavaScript может быть немного неудобным, и в нем легко допускать ошибки из-за отсутствия безопасности типов. Nest.js используется TypeScript, который является золотой серединой. Это язык, который обеспечивает простоту и мощь JavaScript при обеспечении безопасности типов других языков, к которым вы, возможно, привыкли. Безопасность типа в Nest.js доступен только во время компиляции, поскольку Nest.js сервер скомпилирован в Node.js Экспресс-сервер, работающий на JavaScript. Однако это по-прежнему является важным преимуществом, поскольку позволяет вам лучше разрабатывать программы без ошибок перед выполнением.

Node.js имеет богатую экосистему пакетов в NPM (Node Package Manager). С более чем 350 000 пакетов, это крупнейший в мире реестр пакетов. Фреймворк Nest.js создан поверх фреймворка Express, что позволяет получить доступ ко всем его возможностям при разработке приложений. Одно из самых больших преимуществ Node.js — это огромный репозиторий модулей, которые можно установить и сразу использовать, вместо того чтобы писать свои собственные. Некоторые из этих модулей уже включены в платформу Nest, например, @nestjs/mongoose, которая использует библиотеку NPM mongoose. До 2009 года JavaScript был в основном интерфейсным языком, но после выпуска Node.js в 2009 году он стимулировал разработку многих проектов на JavaScript и TypeScript, таких как Angular, React и Vue. Angular послужил источником вдохновения для разработки Nest.js, потому что они оба используют «Систему модулей и компонентов», позволяющую использовать их повторно. Они также очень хорошо сочетаются друг с другом, поскольку Nest предоставляет возможность запускать универсальный сервер для обслуживания предварительно отрендеренных веб-страниц Angular, чтобы ускорить доставку веб-сайта, используя упомянутый выше серверный рендеринг (SSR). [10]

**О PostgreSQL.**

На сегодняшний день существует большое количество различных систем управления базами данных - СУБД, от коммерческих до открытых, от реляционных до новомодных NoSQL и аналогичных.

Одним из лидеров направления СУБД является PostgreSQL и ее различные ответвления, о некоторых из которых мы рассмотрим подробнее.

Редакции PostgreSQL:

1. Классический PostgreSQL - является «ванильная» сборка от PGDG, PostgreSQL Global Development Group. PostgreSQL создана на основе некоммерческой СУБД Postgres, разработанной как open-source проект в Калифорнийском университете в Беркли. Название расшифровывалось как «Post Ingres», и при создании Postgres были применены многие ранние наработки БД Ingres;

2. Российский Postgres Professional - сборка Postgres Pro это российская коммерческая СУБД, разработанная компанией Postgres Professional с использованием свободно-распространяемой СУБД PostgreSQL. Но при этом классическая СУБД значительно переработана для соответствия требованиям корпоративных заказчиков и российских регуляторов. Также Postgres Pro входит в реестр российского ПО и имеет действующий сертификат ФСТЭК.;

3. 2nd Quadrant Postgres-XL - для тяжелых и высоконагруженных систем в семействе PostgreSQL есть версия Postgres-XL (2nd Quadrant Postgres-XL), которая отличается от PostgreSQL иной философией и целями развития, в рамках которых стабильность, корректность и производительность ставятся выше функциональности. Данная редакция направлена на поддержку функций PostgreSQL при распределении рабочей нагрузки по кластеру и нескольких серверов БД. Само название "Postgres-XL" означает "Расширяемая решетка". Для выполнения этих задач Postgres-XL включает в себя дополнительные средства для повышения производительности и безопасности, такие как MPP-параллелизм и расширенная модель разграничения доступа, и охватывает область применения обработки больших объёмов данных, в то время как PostgreSQL в основном нацелен на OLTP (Online Transaction Processing).

В основе работы СУБД PostgreSQL лежит серверный процесс базы данных, выполняемый на одном сервере. Доступ из приложений к данным базы PostgreSQL производится с помощью специального процесса базы данных. То есть клиентские программы не могут получать самостоятельный доступ к данным даже в том случае, если они функционируют на том же ПК, на котором осуществляется серверный процесс.

У каждого серверного процесса есть своя локальная память. В ней находится кэш каталога (часто используемая информация о базе данных), планы запросов, рабочее пространство для выполнения запросов и другое.

Экземпляр СУБД работает с несколькими базами данных. Эти базы данных называются кластером. Хранение данных на диске организовано с помощью табличных пространств. Табличное пространство указывает расположение данных (каталог на файловой системе). [11]