*Допущен к защите: Дипломный проект защищен:*

*Старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись)* *«\_\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_года*

*«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ год Зав.учебной частью \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись)*

**ДИПЛОМНИЙ ПРОЕКТ**

***КА «Шаг» СП -2811.10/2016***

*Тема* Универсальный движок блога

*(на русском языке)*

*Тема:* Universal blog engine

*(на английском языке)*

* ***Слушатель*** *\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись) (дата) (Ф.И.О на русском языке)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Ф.И.О на английском языке)*

***Руководитель***  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(рекомендуемая оценка)*

***Экзаменнационная комиссия по защите дипломных проектов:***

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись) (дата) (Ф.И.О)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись) (дата) (Ф.И.О)*

**ЗАДАНИЕ И КАЛЕНДАРНЫЙ ПЛАН**

по выполнению дипломной работы

|  |  |  |
| --- | --- | --- |
| Студентом группы |  |  |
| Мучник Илья Михайлович | | |
| *(Фамилия, Имя, Отчество)* | | |

|  |  |
| --- | --- |
| Тема работы: | Универсальный движок блога |

Утверждена приказом директора филиала от \_\_\_\_\_\_\_\_\_\_\_\_\_ № \_\_\_\_

Срок сдачи студентом законченной работы:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Объем дипломной работы: 50-80 машинописных страниц.

Перечень прилагаемого материала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Организации, на базе которых выполняется дипломная работа: |
| а) |
| б) |
| в) |

**Календарный план выполнения дипломной работы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование этапа работ | Срок выполнения | | Оценка выполненного этапа работы | Роспись руководителя |
| по плану | фактически |
| Написание введения | 01.02.19 |  |  |  |
| Написание основной части | 04.02.19 |  |  |  |
| Оформление дипломной работы | 05.02.19 |  |  |  |
| Получение отзыва руководителя | 09.02.19 |  |  |  |
| Подготовка доклада и наглядных пособий | 10.02.19 |  |  |  |
| Защита дипломной работы | 12.02.19 |  |  |  |

УТВЕРЖДАЮ

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_ 200\_\_ г.

Ст. преподаватель\_\_\_\_\_\_\_\_\_\_ Задание принял к исполнению:

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись студента)

* **ВВЕДЕНИЕ**

Этот дипломный проект представляет из себя заготовку для создания большинства типичных сайтов. Идеей для написания подобного движка являются тестовые задания которые я получал от различных фирм.

На тестовые задания часто дается ограниченное время в которое очень тяжело вложиться.И ко мне пришла мысль сделать заготовку от которой можно отталкиваться.

В ходе выполнения дипломного проекта мною было изучено очень много новых востребованных технологий и закреплены знания по ранее изученым.

**ОБЩАЯ ЧАСТЬ**

**1 ВЫБОР ИНСТРУМЕНТОВ РАЗРАБОТКИ**

**1.1 Выбор IDE**

**IntelliJ IDEA** -cамая умная и удобная среда разработки для JavaScript, включающая поддержку всех последних технологий и фреймворков. IntelliJ IDEA предоставляет инструменты для продуктивной работы и идеально подходит для создания коммерческих, мобильных и веб-приложений.

**Преимущества:**

* Умное автодополнение, инструменты для анализа качества кода, удобная навигация, расширенные рефакторинги и форматирование для Java, Groovy, Scala, HTML, CSS, JavaScript, CoffeeScript, ActionScript, LESS, XML и многих других языков.
* Поддержка всех популярных фреймворков и платформ, включая Node.js, React,Java EE, Spring Framework, Grails, Play Framework, GWT, Struts, js,AngularJS, Android, Flex, AIR Mobile и многих других.
* Интеграция с серверами приложений, включая Tomcat, TomEE, GlassFish, JBoss, WebLogic, WebSphere, Geronimo, Resin, Jetty и Virgo.
* Инструменты для работы с базами данных и SQL и NOSQL файлами, включая удобный клиент и редактор для схемы базы данных.
* Интеграция с коммерческими системами управления версиями Perforce, Team Foundation Server, ClearCase, Visual SourceSafe.
* Инструменты для запуска тестов и анализа покрытия кода, включая поддержку всех популярных фреймворков для тестирования.

**1.2 Выбор браузера**

**Google chrome**

Успех браузера связан не столько со всемирной известностью компании, сколько с многочисленными преимуществами обозревателя по сравнению с конкурентами.

* Быстрая работа. Браузер быстро открывается даже на слабых устройствах, без задержек открывает новые вкладки и оперативно подгружает содержимое страниц.
* Высокий уровень безопасности. Разработчики постоянно актуализируют списки потенциально опасных сайтов, которые могут навредить работе операционной системы и привести к утечке личных данных пользователя. Кроме того, реализация схемы «одна вкладка – один процесс» препятствует отслеживанию деятельности на других страницах, в случае, если просматриваемый сайт оказался небезопасен. При сохранении исполняемых файлов, хром браузер дополнительно запросит подтверждения, блокируя возможность скрытого скачивания вредоносного ПО.
* Стабильность. Гугл браузер крайне редко зависает и вылетает. Каждая вкладка реализована как отдельный процесс, что позволяет при неполадках с одной из них, не терять данные других.
* Минималистичный интуитивный интерфейс. Все функциональные элементы расположены компактно, не съедая пространство для отображения страниц.
* Огромный выбор плагинов и расширений. Большая их часть предназначена для упрощения работы в интернете, но есть огромное количество бесплатных аддонов, заменяющих полноценные программы для просмотра и редактирований различных файлов.
* Наличие омнибокса. Адресная строка совмещена с поисковой. Помимо этого, во время ввода запроса Хром выдаёт подсказки, сформированные на основе популярных поисковых ключей и ранее посещённых страниц.
* Режим «инкогнито». Анонимный режим удобный для работы на чужих устройствах. Не сохраняет историю просмотренных страниц и файлы куки.
* Автоматический перевод текста. Нет нужды копировать непонятные куски текста и вставлять их в онлайн-переводчик. Работает со всеми языками.
* Удобная синхронизация с гугл-аккаунтом. С её помощью трудно потерять пароли и закладки, пользуясь разными устройствами.
* Наличие инструментария для разработчиков и веб-дизайнеров. Специальные возможности позволяют просматривать коды элементов веб-страниц, контролировать задачи, запущенные в обозревателе и проводить разнообразные диагностики процессов.Продуманное сочетание функций поиска и навигации для андроид устройств. Покажет интересующий адрес на карте и составит оптимальный маршрут до него.

**1.3 Дополнительное программное обеспечение**

**1.3.1 POSTMAN**

Postman стал популярен в качестве инструмента для автоматизации тестирования API, благодаря своему расширению для браузера, которое предназначено для проверки API. Postman можно установить, как в виде расширения для браузера, так и в качестве desktop-приложения для операционных систем Mac, Linux и Windows. Его используют не только тестировщики, которые занимаются автоматизацией тестирования API, но он так же полезен разработчикам для создания и проверки API. По факту, его можно рассматривать, как среду для разработки и тестирования API

**1.3.2 MongoDB Compass**

Возможности Compass

* Просматривать, добавлять и удалять базы данных и коллекции
* Использовать все функции CRUD для работы c документами
* Выполнять и оптимизировать запросы с помощью графического интерфейса
* Визуализировать геопространственые данные и использовать их в запросах
* Работать с индексами
* Расширять возможности Compass, используя дополнительные плагины или даже, создавать собственные
* Построение соединяющих связей с помощью интуитивно-понятного графического интерфейса

**2 ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ**

**JavaScript**

JavaScript – язык программирования, являющийся прототипно-ориентированным. Он отражает язык ECMAScript, чьим прототипом изначально и являлся. Первая вариация появилась ещё в 1995 году и с тех пор постоянно совершенствовалась, пока не пришла к нынешнему виду.

Чаще этот язык используется в разработке приложений и браузерах с целью придания им интерактивности и «живости».

**Широкий выбор настроек и постоянное совершенствование языка**

Одно из ключевых преимуществ данного программного продукта – это поддержка практически всеми известными и самыми популярными браузерами. Выделим другие плюсы JavaScript:

- понятен для пользователей, не являющихся профессиональными программистами;  
- прямое подключение скриптов к HTML коду;  
- возможность запуска программ в браузере и на сервере;  
- широкий выбор полезных функциональных настроек.

Язык программирования JavaScript постоянно совершенствуется, разработчик выпускает обновления, в которые добавляются новые функции. Обновленные версии учитывают недоработки прошлых продуктов и пожелания пользователей. С приложением можно взаимодействовать через обычные текстовые редакторы, включая продукты Microsoft Office. Научиться работать с JavaScript можно самостоятельно, используя специальные учебные пособия (их можно найти в Сети), практикуясь и советуясь с программистами.

JavaScript является безопасным продуктом, изначально создаваемым для работы с браузерами, но со временем язык программирования значительно расширил возможности для использования.

**Стандартные возможности популярного языка программирования**

Область применения js очень широкая и мало чем ограничивается: приложения, браузеры, редакторы и пр.

Выделим некоторые стандартные возможности продукта:

- удаление старых и создание новых HTML-тегов;  
- демонстрация элементов, изменение их стилей;  
- работа с действиями пользователей: клики, перемещения курсора и пр.

Во многом возможности языка программирования зависят от того, в каком окружении он запущен, а также от опыта самого пользователя в работе с данным продуктом. С помощью JavaScript посылаются запросы на сервер, есть возможность загружать данные без необходимости перезагрузки браузера (или страницы).

Ключевая идея использования JS заключается в том, чтобы изменять значения параметров HTML и свойств страницы. Например, можно поменять расцветку фона страницы в браузере или изменить картинку, которая загружена на странице, выдать окно-предупреждение и т.д.

Возможностей много, освоить их все можно только при наличии практики. Язык продолжает развиваться и дополняться новыми опциями, которые расширяют его функциональность и область применения.

**3 ВЫБОР АРХИТЕКТУРЫ**

**3.1 RestApi**

(REpresentational State Transfer) — это архитектура, т.е. принципы построения распределенных гипермедиа систем, того что другими словами называется World Wide Web, включая универсальные способы обработки и передачи состояний ресурсов по HTTP

Автор идеи и термина [Рой Филдинг](https://en.wikipedia.org/wiki/Roy_Fielding) 2000г.

REST на сегодняшний день практически вытеснил все остальные подходы, в том числе дизайн основанный на [SOAP](https://ru.wikipedia.org/wiki/SOAP) и [WSDL](https://ru.wikipedia.org/wiki/WSDL)

# **Что нам дает REST подход**

* Масштабируемости взаимодействия компонентов системы (приложения)
* Общность интерфейсов
* Независимое внедрение компонентов
* Промежуточные компоненты, снижающие задержку, усиливающие безопасность

# **Когда использовать REST?**

• Когда есть ограничение пропускной способности соединения

• Если необходимо кэшировать запросы

• Если система предполагает значительное масштабирование

• В сервисах, использующих AJAX

# **Преимущества REST:**

* Отсутствие дополнительных внутренних прослоек, что означает передачу данных в том же виде, что и сами данные. Т.е. данные не оборачиваются в [XML](https://ru.wikipedia.org/wiki/XML), как это делает [SOAP](https://ru.wikipedia.org/wiki/SOAP) и [XML-RPC](https://ru.wikipedia.org/wiki/XML-RPC), не используется [AMF](https://ru.wikipedia.org/wiki/Action_Message_Format), как это делает Flash и т.д. Просто отдаются сами данные.
* Каждая единица информации (**ресурс**) однозначно определяется URL — это значит, что URL по сути является первичным ключом для единицы данных. Причем совершенно не имеет значения, в каком формате находятся данные по адресу — это может быть и HTML, и jpeg, и документ Microsoft Word.
* Как происходит управление информацией ресурса — это целиком и полностью основывается на протоколе передачи данных. Наиболее распространенный протокол конечно же [HTTP](https://ru.wikipedia.org/wiki/HTTP). Для HTTP действие над данными задается с помощью методов : GET (получить), PUT (добавить, заменить), POST (добавить, изменить, удалить), DELETE (удалить). Таким образом, действия [CRUD](https://ru.wikipedia.org/wiki/CRUD)(Create-Read-Update-Delete) могут выполняться как со всеми 4-мя методами, так и только с помощью GET и POST.

**3.2 Single Page Application**

Одностраничные приложения позволяют имитировать работу десктоп приложений. Архитектура устроена таким образом, что при переходе на новую страницу, обновляется только часть контента. Таким образом, нет необходимости повторно загружать одни и те же элементы. Это очень удобно для разработчиков и пользователей. Для разработки SPA используется один из самых популярных языков программирования - javascript. Небольшое веб приложение можно сделать с библиотекой jQuery. Но, сразу стоит отметить, что jQuery очень плохо подходит для разработки крупных проектов. Наша компания, Merehead, рекомендует использовать более мощные технологии для разработки SPA. Для этих целей хорошо подойдет React.js, Angular.js, Vue.js и другие фреймворки/библиотеки. Их архитектура позволяет разрабатывать гибкие веб приложения. Более того на базе фреймоврков можно строить мобильные приложения с повторным использованием когда. Такие возможности дает Rreact Native и Ionic. Основные преимущества Single Page Application: Высокая скорость. Так как SPA не обновляет всю страницу, а только нужную часть, это существенно повышает скорость работы. Высокая скорость разработки. Готовые библиотеки и фреймворки дают мощные инструменты для разработки веб приложений. Над проектом могут параллельно работать back-end и front-end разработчики. Благодаря четкому разделение они не будут мешать друг другу. Мобильные приложения. SPA позволяет легко разработать мобильное приложение на основе готового кода. При всех своих достоинствах, Single Page Application имеет некоторые недостатки, которые сдерживают рост популярности: Плохая SEO оптимизация. SPA работает на основе javascript и загружает информацию по запросу со стороны клиента. Поисковые системы с трудом могут имитировать данное поведение. Потому большинство страниц попросту недоступны для сканирования поисковыми ботами. Не активный javascript. Некоторые пользователи отключают javascript в своих браузерах, а без него ваше приложение не будет работать. Низкий уровень безопасности. JavaScript имеет низкий уровень безопасности, но если использовать современные фреймворки, они могу сделать ваше веб приложение безопасным. Но стоит обратить внимание, что использование jQuery может существенно понизить безопасность вашего проекта. Одностраничные веб приложения хорошо подходят для разработки динамических платформ, с небольшим объемом данных. Кроме того, если Вам потребуется в будущем построить мобильное приложение, SPA отлично подойдет как основа. Основным недостатком, который сдерживает стремительный рост популярности SPA это плохая SEO оптимизация.

**4 ВЫБОР СТЕКА ТЕХНОЛОГИЙ**

**4.1 Node.js**

Один из важнейших плюсов платформы – это асинхронность в сочетании с событийным подходом. В отличие от потокового, событийно-ориентированное программирование основано на каких-то внешних действиях. Т.е. выполнение программы зависит от действий пользователя или поступившего в программу сетевого пакета и т.д.

Такое решение упрощает программирование интерактивных приложений, т.е. основанных на работе с использованием ввода-вывода. Это могут быть онлайн-чаты или игровые приложения, красочные интерактивные веб-сервисы или всевозможные калькуляторы, рейтинги, голосования и т.д.

Кроме того, к преимуществам платформы относят:

1. **Простой и широко известный JavaScript**. Конечно, платформа предполагает собственные инструменты и особенности, например, здесь нет браузерных API, coockie или DOM, зато присутствуют собственные библиотеки и другие интересные решения. Но в основном используются возможности и синтаксис всем привычного JavaScript.
2. **Богатая стандартная библиотека**. Платформа изначально обладала широким набором возможностей, а в новых версиях библиотека пополняется и улучшается.
3. **Огромное внешних библиотек и готовых модулей**. Использование пакетного менеджера NPM позволяет постоянно развивать экосистему Node. Сегодня число опенсорсных пакетов в нем перевалило за цифру 500 тысяч и постоянно растет.
4. **Движок V8**. JavaScript создавался как интерпретируемый скриптовый язык. Но процесс его интерпретации не столь быстрый и простой, как хотелось бы. При этом язык развивается, он давно стал полноценным, на JavaScript можно писать большие программы. А потому наличие компилятора стало не просто плюсом, но – необходимостью.

Над постоянным улучшением платформы работают тысячи заинтересованных программистов из разных стран. Проект распространяется с открытым кодом, а потому развивается огромными темпами. Координирует работу фонд Node.js.

**4.2 Express.js**

Для этого фреймворка существует большое количество подробных инструкции и описаний, которые составлены разработчиками, проверившими его эффективность на практике. Поэтому именно с *Express*рекомендуется начинать работу, если вы намерены научиться создавать приложения на платформе Node.js.

*Согласитесь, гораздо разумнее воспользоваться накопленным и проверенным опытом, чем заново изобретать велосипед.*

Основная особенность этого фреймворка заключается в том, что для *Express* характерен **небольшой объем базового функционала**. Все остальные нужные вам функции нужно будет добирать за счет внешних модулей. По сути, *Express* в чистом виде – это сервер и у него может не быть ни одного модуля.

Благодаря такому минимализму разработчик изначально получает в свое распоряжение **легкий** **и быстрый инструмент**, который он может расширять и развивать.

При этом немаловажно, что выбор модулей для *Express* не связан ни с какими ограничениями: ни с количественными, ни с функциональными.

В результате, этот фреймворк обеспечивает разработчику возможность решать любые задачи, не ограничивая его при этом в выборе средств.

С одной стороны, не может не радовать тот факт, что отсутствие готовых универсальных решений  фактически означает, что каждое **создаваемое приложение будет уникальным.**

С другой стороны, разработчику нужно самостоятельно отбирать и организовывать модули, а это предполагает большой объем работы и соответственно, требует от разработчика больше времени и усилий.

**4.3 React.js**

React это библиотека для создания пользовательских интерфейсов. Одной из ее отличительных особенностей является возможность использовать JSX, язык программирования с близким к HTML синтаксисом, который компилируется в JavaScript. Разработчики могут добиваться высокой производительности приложений с помощью Virtual DOM. C React вы можете создавать [изоморфные приложения](https://xbsoftware.ru/blog/react-seo-izomorphnost-react-odnostrannoe-prilozhenie/), которые помогут вам избавиться от неприятной ситуации, когда пользователь с нетерпением ожидает, когда же наконец завершится загрузка данных и на экране его компьютера наконец появится что-то помимо анимации загрузки. Созданные компоненты могут быть с легкостью изменены и использованы заново в новых проектах. Высокий процент переиспользования кода повышает покрываемость тестами, что, в свою очередь, приводит к более высокому уровню контроля качества. Используя [React Native](https://facebook.github.io/react-native/) мобильные приложения для Android и iOS, используя опыт JavaScript и React разработки.

Это были технические особенности, которые могут послужить пищей для размышлений для разработчиков. Теперь давайте перейдем к следующему вопросу.

Какую пользу из React может извлечь заказчик?  
Итак, давайте разбираться:

* Virtual DOM может повысить производительность высоконагруженных приложений, что может снизить вероятность возникновения возможных неудобств и улучшает пользовательский опыт;
* Использование изоморфного подхода помогает производить рендеринг страниц быстрее, тем самым позволяя пользователям чувствовать себя более комфортно во время работы с вашим приложением. Поисковые системы индексируют такие страницы лучше. Поскольку один и тот же код может быть использован как в клиентской, так и в серверной части приложения, нет необходимости в дублировании одного и того же функционала. В результате время разработки и затраты снижаются;
* Благодаря переиспользованию кода стало гораздо проще создавать мобильные приложения. Код, который был написан во время создания сайта, может быть снова использован для создания мобильного приложения. Если вы планируете использовать не только сайт, но и мобильное приложение, нет необходимости нанимать две большие команды разработчиков.

Теперь давайте более подробно рассмотрим, за счет чего достигаются вышеописанные выгоды. Конечно, одной статьи будет недостаточно, чтобы описать все возможности данной библиотеки. Но мы сконцентрируемся на самых важных из них чтобы помочь вам понять, могут ли React разработчики решать проблемы с помощью:

1. Улучшения пользовательского опыта ваших сайтов и приложений
2. Увеличения скорости разработки
3. Использования наиболее трендовых технологий разработки

**4.4 Redux**

Redux мастерски справляется со сложными взаимодействиями состояний, которые трудно передать с помощью состояния компонента React. По сути, это система передачи сообщений, которая встречается и в объектно-ориентированном программировании, но она не встроена непосредственно в язык, а реализована в виде библиотеки. Подобно ООП, Redux переводит контроль от вызывающего объекта к получателю — интерфейс не управляет состоянием напрямую, а передает ему сообщение для обработки.

В этом плане хранилище в Redux — это объект, редюсеры — это обработчики методов, а действия — это сообщения. Вызов store.dispatch({ type:"foo", payload:"bar" }) равносилен store.send(:foo, "bar") в Ruby. Middleware используется почти таким же образом, как в аспектно-ориентированном программировании (например, before\_action в Rails), а с помощью connect в react-redux осуществляется внедрение зависимости.

**Какие преимущества даёт такой подход?**

* Благодаря инверсии контроля, упомянутой выше, исчезает необходимость обновлять пользовательский интерфейс, как только меняется имплементация смены состояний. Добавлять такие сложные функции, как логирование, отмена действия или даже time-travel debugging, становится проще простого. Интеграционные тесты сводятся лишь к тому, чтобы проверить, отправляется ли правильное действие, а для всего остального достаточно юнит-тестов.
* Состояние компонентов в React слишком неповоротливо для работы со сквозной функциональностью, которая затрагивает многие модули приложения, как, например, информация о пользователе или оповещения. Как раз для этого в Redux есть дерево состояний, независимое от пользовательского интерфейса. К тому же, при обработкe состояния вне интерфейса легче поддерживать постоянство, ведь сериализация в localStorage или URL проводится в единственном месте.
* Редюсеры дают огромную свободу в работе с действиями, которые можно сочетать, отправлять одновременно и даже обрабатывать в стиле method\_missing.

**4.5 Mongoose**

Главное преимущество - абстракция над чистым монго.

Многие разработчики, которые приходят из типов баз данных SQL, чувствуют себя очень неудобно, работая с динамическими коллекциями, которые не имеют определенной структуры. Поэтому **Схемы** в первую очередь помогают с этим.   
Кроме того, он реализует проверку и другие опрятные функции, чтобы убедиться, что ваша схема согласована при вставке/обновлении/поиске документов из коллекций.

Он также создает абстракцию **модели, из-за** которой упрощается работа с ней, поэтому похоже, что вы работаете с объектами, а не с чистыми данными.

Есть много других полезных свойств, таких как **промежуточное ПО**, **плагины**, **население**, **проверка**. Пожалуйста, проверьте [документы mongoose](http://mongoosejs.com/docs/guide.html) для получения дополнительной информации:

Лично я предпочитаю чистое монго, поскольку он более согласуется с официальными документами 10gen mongo и не создает абстракции, которые всегда стоят определенных ограничений и правил, которые вам придется соблюдать.

**5 ВЫБОР БАЗЫ ДАННЫХ**

Как мы знаем, любая реляционная БД имеет стандартную схему, которая показывает количество таблиц и связи между ними. В **MongoDB** такой схемы с связи между таблицами нет.

Ниже приведены основные преимущества MongoDB:

* Отсутствие схемы  
  Данная БД основана на коллекциях различных документов. Количество полей, содержание и размер этих документов может отличаться. Т.е. различные сущности не должны быть идентичны по структуре.
* Крайне понятная структура каждого объекта.
* Легко масштабируется
* Для хранения используемых в данный момент данных используется внутренняя память, что позволяет получать более быстрый доступ.
* Данные хранятся в виде JSON документов
* MongoDB поддерживает динамические запросы документов (document-based query)
* Отсутствие сложных JOIN запросов
* Нет необходимости маппинга объектов приложения в объекты БД

Подведя итого можно сказать, что MongoDB является достаточно неплохим решением, если мы имеем дело с Big Data.

**6 Библиотеки**

**6.1 Серверная часть**

Bcrypt - для шифрования и сравнения паролей

Jsonwebtoken - для создания токенов

Jwt-decode - декодирование токенов

Uuid - создание hash

Morgan - для логирования запросов

Multer - для работы с файлами

**6.2 Клиенская часть**

Axios — для работы с Ajax без jQuery

Jwt-decode - декодирование токенов

Lodash — для удобной работы с коллекциями

Validator — для валидации

Prop-types — для проверки типов

**7 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

Подобных заготовок с применением подобного стека и данной архитектуры мной не было обнаружено.

**ПРЕИМУЩЕСТВА:**

* Производительность (за счёт использования кэша);
* Масштабируемость;
* Портативность компонентов;
* Лёгкость внесения изменений;
* Возможность использовать разные клиентские приложения(веб,мобильные,десктопные);
* Скорость за счет V8 , VirtualDom и noSQL базы данных
* Является базой для выполнения тестовых заданий

**СПИСОК ИСТОЧНИКОВ**

* <https://wikipedia.org/>
* <https://stackoverflow.com/>
* [httpsHYPERLINK "https://msdn.microsoft.com/ru-ru/"://HYPERLINK "https://msdn.microsoft.com/ru-ru/"msdnHYPERLINK "https://msdn.microsoft.com/ru-ru/".HYPERLINK "https://msdn.microsoft.com/ru-ru/"microsoftHYPERLINK "https://msdn.microsoft.com/ru-ru/".HYPERLINK "https://msdn.microsoft.com/ru-ru/"comHYPERLINK "https://msdn.microsoft.com/ru-ru/"/](https://msdn.microsoft.com/ru-ru/)
* <https://react.semantic-ui.com/>
* <https://metanit.com/>
* <https://reactjs.org/>
* <https://nodejs.org/uk/docs/>