Создание полноценного приложения React.js + PHP

Сергей Кундрюков
Full-stack JavaScript разработчик
«Техносерв Консалтинг»

Что будет на лекции?

1. Что такое «приложение»
2. Учимся абстрагироваться
3. Что такое SPA
4. Из чего состоят веб-приложения
5. Как не запутаться
6. Смотрим код реального приложения

Что такое «приложение»

Приложение – это программа выполняющая определенные задачи и, обычно, взаимодействующая с пользователем или другими программами.

Что такое «приложение»

Приложения бывают:

1. Консольные
2. Десктопные
3. Серверные
4. Веб-приложения
5. Мобильные приложения
6. VR-приложения и др.

Что такое «приложение»

Какие задачи ложатся на приложения?

1. Развлечения
2. Облегчение жизни
3. Упрощение рутины
4. Оптимизация бизнес-процессов
5. Обучение
6. И др.

Учимся абстрагироваться

Любое полноценное приложение – это сложный механизм взаимодействия разного кода.

Чтобы упросить себе работу – нужно уметь абстрагироваться!

Что это означает?

Учимся абстрагироваться

При создании большого приложение даже опытному программисту бывает не под силу держать всё в голове.

Поэтому, сперва нужно ответить на вопрос:
«Что должно делать приложение?»

А уже потом – *«как оно должно быть реализовано?»*

Учимся абстрагироваться

Для этого нужно постепенно погружаться от бОльшего к мЕньшему, разбивая сложную задачу на всё меньшие и меньшие кусочки.

Какой профит?

Писать очень сложный код или решать очень сложные повседневные задачи становится легко!

Что такое SPA?

SPA (Single Page Application) – одностраничное приложение.

Это такой вид приложения, которое работает не перезагружая страницу.

Что такое SPA?

Большинство сайтов работают так:

1. Вы зашли на сайт – загрузилась страница
2. Вы нажали на ссылку – страница перезагрузилась с нуля
3. Нажали ещё одну – см. п. 2

Что такое SPA?

Чем это плохо:

1. Большая нагрузка на сервер
2. Тяжелее масштабировать проекты
3. Дольше загрузка ценной информации для пользователя

Поэтому так популярны стали SPA!

Из чего состоят веб-приложения?

Приложение может быть построено множеством разных способов и все они будут работать по разному.

Я опишу тот подход, который сейчас наиболее актуален и который ценится в большинстве команд.

Из чего состоят веб-приложения?

Наши приложения должны уметь:

1. Где-то хранить данные
2. Как-то обрабатывать данные
3. Выводить информацию на экран
4. Взаимодействовать с пользователем
5. Обмениваться данными внутри приложения

Из чего состоят веб-приложения?

Хорошей практикой является разделять серверную часть (Back-end) и пользовательскую часть (Front-end).

Как показывает практика – разделённые бек и фронт удобнее развивать и поддерживать, исправлять ошибки, внедрять новые фичи и т.п.

Из чего состоят веб-приложения?

Бекенд должен уметь:

1. Хранить данные
2. Обрабатывать данные
3. «Слушать» запросы
4. Авторизовывать пользователей
5. И т.п.

Фронтенд должен уметь:

1. Получать/отдавать данные
2. Взаимодействовать с пользователем
3. Рисовать странички
4. Облегчать работу сервера

Из чего состоят веб-приложения?

Это позволит:

1. Удешевить разработку
2. Сократить расходы на оборудовании
3. Увеличить скорость и гибкость приложений

Back-end. Хранение данных

Что хранить?

1. Тексты/разделы
2. Данные пользователей
3. Файлы
4. Сессии пользователей
5. Прочую информацию

Как хранить?

1. В файлах JSON/XML
2. В оперативной памяти
3. Базы данных

Back-end. Обработка данных

Например, можно получить данные их формы обратной связи, создать заказ, сохранить его в БД и сказать пользователю, что всё хорошо (или нет).

Или изменить входящие данные, вернуть в другом виде.

Back-end. Слушать запросы

Общение сервера и клиента происходит в режиме «запрос-ответ». Это позволяет нам обмениваться данными с помощью протокола HTTP.

Существуют разные запросы, основные это:
GET, POST, PUT, PATH, DELETE (об этом позже)

Back-end. Авторизация пользователей

Всё общение клиента и сервера идёт с помощью запросов.
Мы можем отследить на сервере от какого пользователя приходят запросы, чтобы позволить ему получать закрытую от всех информацию.

Например, личный кабинет или админ-панель сайта.

Front-end. Получать/отдавать данные

Попав на ваш сайт пользователь может получать в ответ почти пустую страницу, а Фронтенд нарисует её в процессе работы программы.

Так же, фронтенд может анализировать действия пользователей и передавать информацию на сервер, например, запрос «загрузить ещё новости в ВК».

Front-end. Взаимодействие с юзером

Попав на сайт, вы незаметно провоцируете события. Клики мыши, скролл, движение мышкой, время пребывания на сайте.

Фронтенд должен уметь обрабатывать эти действия.

Front-end. Рисование страничек

Наше приложение не перезагружает страниц. Это значит – оно рисует их само!

Фронденд должен уметь шаблонизировать данные, например, получать 100 товаров в виде текста с сервера и рисовать 100 карточек этих товаров, удобных для пользователя.

Front-end. Облегчение работы сервера

Очень важный аспект качественного фронтенда. Если ваши странички полностью перезагружаются, это значит, что страничка полностью удаляется из памяти и запрашивает с сервера её целиком.

SPA запрашивает часть странички или данные о части странички и умеет рисовать их само.

Как не запутаться?

На первый взгляд, прозвучало много новой информации, при чём без конкретики.

Я специально не даю детали реализации, чтобы суть была понятна даже самым не опытным нашим товарищам.

Как не запутаться?

Всё описанное выше требует реализации.
Для этого есть уже устоявшиеся подходы и работающие инструменты.

Я постараюсь описать популярные подходы, чтобы вы могли реализовать своё приложение.

Как не запутаться?

Обмен данными происходит с помощью HTTP-запросов на нашего бекенда URL.
Запросы бывают:

1. GET – получить
2. POST – отправлять
3. PUT – положить
4. DELETE – удалить

Как не запутаться?

Чтобы они работали, сервер необходимо научить различать их и обрабатывать.

Запросы делаются на такие URL, которые несут какой-то смысл в своих названиях.

Как не запутаться?

Допустим, у нас есть в приложении товары, машины и пользователи. Мы можем получить список каждой сущности с помощью запросов.

Запросы должны идти на URL вида:
backend.ru/v1/api/products – работа с товарами
backend.ru/v1/api/cars – работа с машинами
backend.ru/v1/api/users – работа с пользователями

Как не запутаться?

Чтобы получать не список, а конкретный товар, их принято получать по ID так:

backend.ru/v1/api/products/:id

Здесь :id – динамично подставляемое значение нужного нам товара

Как не запутаться?

При запросах могут возникать ошибки. Их принято разделать по кодам. Вот самые популярные из них:

1. 400 – не правильный запрос
2. 404 – не найдено
3. 500 – ошибка сервера

4** – ошибка клиента

5** – ошибки сервера

Как не запутаться?

Такой подход называется REST API. Подходов существует множество, но при REST мы знаем, что если мы делаем GET на /products/ - то мы получаем список товаров.

Если DELETE на /products/13 – удаляем товар с ID 13

Как не запутаться?

Следующий подход – это Routing (Маршрутиризация). Это возможность обрабатывать маршруты (url) по разному.

На сервере – это работа с разными сущностями, на клиенте – показ разных страничек с разными данными, например - товарами.

Как не запутаться?

Шаблонизация – возможность один раз написать HTML-разметку, а потом подставлять в неё разные данные (тексты, фото).

Так же, шаблонизация позволяет рисовать множество однотипных объектов (карточки товаров, слайды презентации).

Как не запутаться?

Шаблонизация – возможность один раз написать HTML-разметку, а потом подставлять в неё разные данные (тексты, фото).

Так же, шаблонизация позволяет рисовать множество однотипных объектов (карточки товаров, слайды презентации).

Как не запутаться?

Это всё!

Всё, что нужно, чтобы сделать полноценное веб-приложение.

Всё это реализовать нам помогут React.js и PHP.

Смотрим код реального приложения

В работе мы будем использовать:

Back-end

1. PHP 5.6
2. Веб-сервер Apache 2.4
3. JSON для хранения данных

Front-end

1. Node.js
2. React.js
3. react-router-dom
4. create-react-app

Смотрим код реального приложения

Всё необходимое для реализации Бекенда можно получить, арендовав обычный хостинг (для выгрузки приложения в интернет), либо скачать пакет OpenServer.

Тогда настройки займут минимум времени.

Смотрим код реального приложения

Для реализации Фронтенда, мы будем использовать уникальный инструмент, который за нас скачает всё необходимое, чтобы начать создавать React-приложение.

Для этого нужно установить Node.js с официального сайта, потом выполнить команду:
npm i –g create-react-app

Смотрим код реального приложения

Теперь у нас установлено консольконе приложение, которое настраивает всё необходимое и позволяет нам просто кодить.

Чтобы создать приложение, нужно выполнить в консоли команду *create-react-app app-name*

Смотрим код реального приложения

Установка окружения занимает массу времени, в конце в папке, где вы открыли консоль, будет создана папка app-name (обязательно через тире), куда и будет развёрнута заготовка приложения.

Она включает в себя WebPack, Babel, React.js и много всего, что руками устанавливать лень.

Смотрим код реального приложения

Если вы скачали готовое приложение с гитхаба, то создавать новое приложение нет необходимости.

Нужно будет в папке приложения выполнить команду *npm i* – и приложение само установит нужные модули.

Смотрим код реального приложения

Перейдём к самому коду!

WebDev Summit 2018

Спасибо за внимание!

github.com/serp-ya

vk.com/kosinsky

facebook.com/Sergei.Kundryukov