|  |
| --- |
| Курсовая работа  **РАЗРАБОТКА И РЕАЛИЗАЦИЯ GUI ИНТЕРФЕЙСА ИНТЕРАКТИВНОГО МАГАЗИНА КОМПЬЮТЕРНОЙ ТЕХНИКИ** |
| Студент гр. ИС-17 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата  Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.Г. Деркаченко  подпись, дата  Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_\_\_».  Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_  подпись Фамилия И.О.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_  подпись Фамилия И.О. |
|  |

Витебск, 2020

**СОДЕРЖАНИЕ**

Введение

1. Теоретическая часть
   1. GUI интерфейс
   2. Библиотека React
   3. Выбор языка программирования
   4. Препроцессор SASS
   5. Обоснование выбора среды разработки

1. Практическая часть

2.1 Создание GUI интерфейса

2.2 Создание компонента ContactInformation

2.3 Создание компонента Header

2.4 Создание компонента Main

2.5 Создание компонента OurProduction

2.6 Создание компонента DeliveryInformation

2.7 Создание компонента Partners

2.8 Создание компонента Info

2.9 Создание компонента Footer

Заключение

Список использованных источников

Содержание электронного носителя

**Введение**

Каждый день миллионы людей активно используют свои мобильные устройства. Они печатают, кликают, листают, перетаскивают и меняют размер изображений на своих сенсорных дисплеях. Естественно так было не всегда.

В 70-х годах компания Xerox создала первый ***графический пользовательский интерфейс* GUI**. Основная идея пользовательского интерфейса «*что видишь на экране, то и получишь*».

Инженеры в Xerox пытались таким способом упростить подготовку документов к печати.

Вскоре после этого Apple позаимствовала эту идею в обмен на некоторое количество акций своей новой фирмы. В 1984 году свет увидел новый Macintosh – компьютер для пользователей, с помощью которого люди могли выполнять какие-то повседневные задачи.

Идея оказалась настолько успешной, что в очень скором времени ее использовали в компании Microsoft, создав свою собственную платформу Windows.А появление [Windows 95](http://www.iphones.ru/iNotes/30596) без преувеличения сотворило революцию в сфере ПК.

С течением времени появились и другие устройства призванные облегчить жизнь пользователей. Например, музыкальные плееры. На смену CD и MP3 плеерам пришел iPod ознаменовав тем самым доступность и портативность музыки в цифровом формате.

Примерно в это же время начали появляться сотовые телефоны. Первые их модели были похожи на простые стационарные телефоны. С развитием технологий сотовые телефоны начали оснащать ЖК-дисплеями. Вскоре их владельцы смогли отправлять текстовые сообщения по сотовой сети.

Через некоторое время такие разработки как компьютер, сотовый телефон и MP3 стали вполне самодостаточными, состоявшимися продуктами. В Apple воспользовались этим и объединили их в одном устройстве iPhone.

 На фоне этого, iPhone оказался просто таки революционным продуктом, в своей сфере, с его емкостным сенсорным дисплеем, с поддержкой multitach. Другие телефоны пока еще были оснащены пластиковыми резистивными дисплеями, и их пользователям, чтобы реализовать тот или иной жест, приходилось сильно надавливать на экран. Емкостный же дисплей реагирует на электроимпульсы в человеческом теле, и в сильных нажатиях просто нет необходимости.

Однако отображение контента все еще желало лучшего. Все изменилось с выходом Retina-дисплей. Этот дисплей обладает высокой плотностью пикселей, благодаря чему текст и изображения стали видны намного четче – будто на странице глянцевого журнала.

Основная суть GUI (графический пользовательский интерфейс) использовать вещи которые уже знакомы и понятны пользователям, для того что бы научить их обращаться с новыми вещами.

Возьмем в качестве примера приложение калькулятор. В этом графическом интерфейсе можно уловить схожесть со старым привычным калькулятором.

Интерфейс пользователя и интерфейс аппаратной части гармонично сосуществуют, достигнув симбиоза. Лучшее ПО сливается с прекрасно проработанными аппаратными компонентами, и они вместе создают само устройство. Увидев его, пользователь уже не задумывается над тем, как оно работает. В мыслях его сразу формируются картинки того, как его можно использовать.

**Теоретическая часть.**

* 1. **Библиотека React**

React – это open source библиотека компонентов пользовательского интерфейса (UI-компонентов). Данная библиотека была создана компанией Facebook. UI-компоненты, создаются на языке JavaScript. Этот подход, называется *созданием составного пользовательского интерфейса*. Если просто, то React создан для упрощения создания пользовательских интерфейсов. Он разбивает пользовательский интерфейс на компоненты. Это так же естественно, как разбивать JavaScript код на функции. Функция - это блок кода, который я могу переиспользовать в разных частях приложения. Точно так же и UI компонент – это блок пользовательского интерфейса с каким-то внешним видом, с каким-то поведением которые можно переиспользовать в разных частях приложения. И точно так же как хорошая функция которая должна быть маленькая и независимая, для того что бы ее можно было легко обновлять и тестировать, хороший UI компонент тоже должен быть компактным и независимым от всего остального приложения что бы его можно было легко обновлять и тестировать.

React построен на идее UI компонентов. Он дает инструменты для того что бы эффективно создавать UI компоненты, и затем строить из них приложение.

UI компоненты используются уже давно. Однако у React есть две сильные стороны, которые отличают его от предыдущих альтернатив.

* React использует язык JSX.

JSX –это язык расширения который позволяет комбинировать JS код с разметкой которая похожа на HTML. Это позволяет более существенно писать UI логику.

* Это алгоритм который React использует внутри себя для того что бы понимать какие именно части вэб-сайта нужно обновить. К примеру пользователь кликает на какой-нибудь компонент и этот компонент изменяет свой внешний вид. Или приходят новые данные с сервера и нужно обновить эти данные на UI.

Так вот reconciliation algorithm, который работает внутри React, делает достаточно сложных вещей для того что бы найти именно те блоки на странице которые требуют обновления и обновить только их, не трогая те блоки которые не изменились. Именно этот подход делает React приложения достаточно быстрыми.

За последние несколько лет библиотека приобрела невероятную популярность и является обязательным инструментом для многих команд и разработчиков, создающих динамические пользовательские интерфейсы.

Появление React в корне изменило обстановку в веб-программировании: сообщество React положило начало многих подходов, понятий и паттернов проектирования.

React — свежий взгляд на разработку компонентов пользовательского интерфейса. Это новое поколение библиотек презентационного уровня. В сочетании с моделью и библиотекой маршрутизации React может заменить Angular, Backbone или Ember в технологическом стеке веб-программирования и мобильной разработки.

Разработчики всегда ищут более эффективные способы разработки пользовательских интерфейсов. Большая часть коммерческой ценности лежит в пользовательских интерфейсах, серверная часть-всего лишь ресурс.

Времена, когда для обычной отладки кода frontend части приходилось набивать вызовами alert() прошло. С развитием интернет технологий развивается и сложность построения сайтов. Сайты превратились в веб-приложения со сложными интерфейсами, бизнес логикой и уровнями данных, которые требуют постоянных изменений и обновлений. Часто в реальном времени.

Для решения проблем, с построением сложных пользовательских интерфейсов было написано много библиотек, паттернов, пакетных менеджеров, которые я буду использовать в курсовом проекте.

Ядро React – это система, автоматически обрабатывающая пользовательский интерфейс, когда изменяются данные в его компонентах. Они вносят изменения, которые помогут перехватить, используя так называемые методы жизненного цикла. Системы React поддерживающие обновление пользовательского интерфейса, значительно упрощают создание надежных модульных компонентов, применяемых приложением.

React в первую очередь библиотека а не framework. Разница в том, что в framework есть масса других аспектов кроме самого UI. Это работа с сервером, валидация, управление глобальным состоянием, Unit test и масса других аспектов. В крупных UI framework-ах вроде Angular, Ember, Aurelia все эти аспекты покрыты из коробки.

React в свою очередь отвечает только за UI компоненты. Но это не страшно т.к. есть масса других библиотек, которые покрывают эти аспекты и которые отлично работают с React. Как раз в этом и заключается гибкость React. Не понравилось как работает библиотека HTTP запросов. Нет проблем, заменю ее на другую. С другими библиотеками дела обстоят точно так же.

Для интерпретации и компиляции кода с последующим выведением контента в браузер, весь код проходит через виртуально объектную модель документа DOM. DOM - это программный интерфейс, который позволяет JavaScript программам взаимодействовать с различными типами документов (HTML, XML, SVG).

Как работает DOM? Она чем-то схожа с другим миром программного обеспечения-трехмерными играми. В таких играх иногда используется рендеринг, который работает примерно так: получить информацию с игрового сервера, отправить его в игровой мир (визуальное представление которое видит пользователь), установить какие изменения стоит внести в этот мир, и тогда графическая карта определит какой минимум изменений. Однако преимущества этого подхода заключается в том, что нужны только ресурсы для работы с инкрементными изменениями, внести которые, как правило, намного быстрее, чем если бы пришлось обновлять все. Таким способом React пытается наиболее эффективным способом обновлять пользовательский интерфейс.

React создает и поддерживает виртуальную модель DOM в памяти, а средство рендеринга обрабатывает обновление в DOM браузера на основе изменений. React может выполнять интеллектуальные обновления и работать только с измененными частями данных, поскольку способна использовать *эвристическую сверку* для вычисления того, какие части DOM в памяти требуют изменений в фактической модели DOM.

* 1. **Выбор языка программирования**

Подавляющее большинство веб-сайтов используют JavaScript, и все современные веб-браузеры – для настольных компьютеров, игровых приставок, электронных планшетов и смартфонов – включают интерпретатор JavaScript, что делает JavaScript самым широко применимым языком программирования. JavaScript входит в тройку технологий, которые должен знать любой разработчик: язык разметки HTML, позволяющий определять содержимое веб-страниц, язык стилей CSS, позволяющий определять внешний вид веб страниц, и язык программирования JavaScript, позволяющий определять поведение веб-страниц.

JavaScript является высокоуровневым, динамическим, нетипизированным и интерпретируемым языком программирования, который хорошо подходит для программирования в объектно-ориентированном и функциональном стилях.

Ядро языка JavaScript определяет минимальный прикладной интерфейс для работы с текстом, массивами, датами и регулярными выражениями, но в нем отсутствуют операции ввода-вывода. Ввод и вывод (а также более сложные возможности, такие как сетевые взаимодействия, сохранение данных и работа с графикой) перекладываются на «окружающую среду», куда встраивается JavaScript. Обычно роль окружающей среды играет веб-браузер.

С помощью JS можно управлять HTML-элементами, которые определяют содержимое вэб-страниц, а так же каскадными таблицами стилей, которые определяют представление содержимого.

JS позволяет не только управлять содержимым и оформлением HTML – документов в браузерах, но и определять поведение этих документов с помощью обработчиков событий. Обработчик событий – это функция JS, которая регистрируется в браузере и вызывается браузером, когда возникает событие определенного типа. Таким событием может быть щелчок мышью или нажатие клавиши, или движение пальцами на экране смартфона.

Прикладной интерфейс, написанный на чистом JS, является достаточно сложным и испытывает проблемы с совместимостью между браузерами. По этим причинам большинство программистов предпочитают использовать клиентские библиотеки или фреймворки, упрощающие программирование. Наиболее популярна из этих библиотек – библиотека React которую я буду использовать в данном курсовом проекте.

* 1. **Препроцессор SASS**