

Geekbrains

**Разработка Web-сайта для Дизайн-бюро**

**c использованием  JavaScript-библиотеки React**

Программа:

Цифровые Профессии

Специлизация

Frontend-программист.

Пешкова Инна Валентиновна

Воронеж

2024

Оглавление

[Почему дизайн-бюро 2](#_Toc167474149)

[ГЛАВА 1ФУНКЦИИ WEB-САЙТА В СФЕРЕ УСЛУГ 4](#_Toc167474150)

[1.1 Анализ веб-сайтов российского рынка индустрии дизайна 4](#_Toc167474151)

[1.2 Сайт-визитка 7](#_Toc167474152)

[1.3 Обоснование выбора программного обеспечения 10](#_Toc167474153)

[ГЛАВА 2. ОСНОВНАЯ ЧАСТЬ ПРОЕТА 12](#_Toc167474154)

[2.1 Техническое задание 12](#_Toc167474155)

[2.1.1 Название организации-заказчика. 12](#_Toc167474156)

[2.1.5 Плановые сроки начала и окончания работ 12](#_Toc167474157)

[2.2 Характеристика области применения продукта. 12](#_Toc167474158)

[2.2.1 Процесс и структура продукта разработки. 12](#_Toc167474159)

[2.2.2 Требования к продукту в целом. 13](#_Toc167474160)

[2.2.3 Программного обеспечения, используемое для реализации. 14](#_Toc167474161)

[2.3 Требования к интерфейсу сайта. 14](#_Toc167474162)

[2.4 Разработка сайта Дизайн-бюро «Надежда». 15](#_Toc167474163)

[2.4.1 Дизайн-сайта. 15](#_Toc167474164)

[2.4.2 Разработка структуры программы 20](#_Toc167474165)

[2.4.3 Обоснование метода разработки структуры программы 21](#_Toc167474166)

[2.4.4 Алгоритм системы web-сайта. 23](#_Toc167474167)

[2.4.5 Используемые технологии в создании сайта. 24](#_Toc167474168)

[2.5 Реализация проекта. 25](#_Toc167474169)

[2.5.1 Начало реализации сайта на React. 25](#_Toc167474170)

[2.5.2 Реализация компонента Header. 26](#_Toc167474171)

[2.5.3 Реализация компонента отображения списка услуг на сайте. 26](#_Toc167474172)

[2.5.3 Реализация компонента Portfolio. 27](#_Toc167474173)

[2.5.4 Создание слайдера на странице сайта . 28](#_Toc167474174)

[2.5.5 Pеализация блока Cost. 29](#_Toc167474175)

[2.5.6 Создание калькулятора проекта. 29](#_Toc167474176)

[2.5.7 Создание Footer проекта. 31](#_Toc167474177)

[ЗАКЛЮЧЕНИЕ 33](#_Toc167474178)

[Список используемой литературы 34](#_Toc167474179)

[ПРИЛОЖЕНИЕ 35](#_Toc167474184)

**Введение**

В современную эпоху, известную как «информационный век», информация стала наиболее важным и незаменимым ресурсом. Успех в современном бизнесе зависит от владения и умелой передачи соответствующей информации клиентам своевременным, эффективным и прозрачным образом.

Интернет — обширный источник информации, позволяющий быстро обмениваться текстом, звуком, изображениями и видео. Он привлекает растущую аудиторию, что делает его главной целью для рекламодателей. Учитывая обширную потенциальную клиентскую базу, Интернет является важной платформой для продажи товаров и услуг. Поскольку сектор услуг продолжает расширяться, онлайн-торговля и реклама приобретают большое значение, особенно в таких регионах, как Россия, где большую роль играют большие расстояния и ограничения во времени. Компании в значительной степени полагаются на самопрезентацию и присутствие в Интернете для привлечения новых клиентов и рынков, часто через веб-сайты.

Для бизнеса Интернет – это современная рекламная и одновременно площадка для общения, которая обеспечивает приток клиентов.

Веб-сайт компании отражает ее современное присутствие и является основным источником информации для потенциальных клиентов и общественных деятелей.

Почему дизайн-бюро

**Актуальность данной разработки** обусловлена востребованностью качественного сайта для начинающего свою деятельность дизайн-бюро. Способ заявить о себе, рассказать о своей деятельности, привлекая аудиторию из потенциальных клиентов посредством web-сайта, отличается низкими затратами и охватом большой целевой аудитории.

**Объектом разработки** является сайт Дизайн-Бюро «Надежда».

**Цель** настоящей работы: разработка web-сайта, предназначенного для ознакомления потенциальных клиентов с деятельностью компании и спектром предоставляемых услуг, а так же проведение первоначального анкетирования клиента.

В соответствии с поставленной целью в работе определены следующие

задачи:

- изучить современные технологии по разработки сайтов;

- определить цели, идеи, для web-сайта;

- разработать техническую концепцию сайта (структуру);

- разработать техническое задание на создание web-сайта;

- разработать программный продукт для организации, который позволит увеличить клиентскую базу.

Результатом работы является дизайн-проект сайта. Программный продукт сайт дизайн-бюро.

**План работы:**

Анализ веб-сайтов российского рынка индустрии дизайна

Определение типа Web-сайта

Анализ деятельности дизайн-бюро.

Определение необходимых технологий при выполнении проекта.

Разработка дизайна-сайта.

Написание сайта согласно отобранным технологиям.

ГЛАВА 1. ФУНКЦИИ WEB-САЙТА В СФЕРЕ УСЛУГ

1.1 Анализ веб-сайтов российского рынка индустрии дизайна

Деятельность любого предприятия сегодня нуждается в создании информационных систем (ИС) и может обусловливаться как частичной, так и полной необходимостью автоматизации или модернизации процессов общения с потенциальным клиентом. Для созданием информационной системы необходимо решить следующие вопросы:

Для каких именно целей необходима разработка системы?

К какому времени необходимо создать разработку?

Какие материальные затраты необходимы от проектирования до функционала?

Сегодня много способов коммерческого подхода к Интернету. В интернете можно рекламировать свои услуги, доносить нунную информацию до пользователей, либо продавать услуги или товары. Цифровая комерция в Интернете имеет огромный потенциал и открывает реальные перспективы коммерции. В современных ИС можно выделить два основных направления использования Интернет в бизнесе: технологии Интернет для бизнеса и бизнес в Интернет-пространстве.

Интернет для бизнеса используется, с самого начала зарождения Интернета. Любому бизнесу необходимы информационное сопровождение своих бизнес-процессов, а также информационное взаимодействие в режиме On-line с филиалами в других городах и странах, с клиентами, с поставщиками. Информационное сопровождение бизнеса это– надежная и недорогая система. В свое время компании, которые первыми стали использовать электронную почту и телеконференции, получили конкурентное преимущество. Компании которые первыми стали обзаводиться сайтами, которые очень быстро стали представлять «лицо» компании в бизнесе, сильно опережали своих конкурентов. Со временем сайты стали становиться одним из инструментов управления бизнесом.

Подход Бизнес в Интернете основан на понимании того, что современный Интернет - это информационное виртуальное пространство, которое доступно любому пользователю сети в любое время в любой точке Земли. Современные технологии Интернета позволяет пользователям, не выходя с работы или дома, делать покупки в Интернет-магазинах, оплачивать услуги, играть на бирже, получать образование, повышать культурный уровень. Для любой компании , это реальная возможность «продвигать» бизнес через Интернет-маркетинг, продажи, оказание услуг, реклама. В связи с этим сформировались два понятия: электронный бизнес и электронная коммерция.

Электронная коммерция - это вид бизнеса, при котором взаимодействие (транзакции) между участниками коммерческих сделок происходит с помощью информационных технологий

**Виды деятельности электронной коммерции заключаются в :**

- Покупка и продажа товаров онлайн

- Онлайн-продажа билетов

- Онлайн-платежи

- Уплата разных налогов

- Программное обеспечение для онлайн-бухгалтерского учета

- Онлайн-поддержка клиентов

Электронную коммерцию в настоящее время принято разделять на ряд направлений, основными из которых считаются: «бизнес – бизнес), «бизнес – потребитель», «потребитель – бизнес», «потребитель – потребитель.

Электронный бизнес –это выполнение всех видов предпринимательской деятельности через Интернет.

Сюда входят такие виды деятельности:

- Покупка товаров

- Обучение клиентов

- Информирование клиентов

- продажа продукции или услуг

В электронном бизнесе можно выделить четыре слоя:

-Телекоммуникационные компании и производители программного обеспечения компьютерного и сетевого оборудования (Интернет-инфраструктура),

- Интернет-провайдеры обеспечивают транзакции в сети, и являются владельцами каналов связи ( Интернет-услуги)

-Информационные посредники (службы, консультационные и обслуживающие компании, обеспечивающие создание сайтов и web-страниц.

- Владельцы поисковых систем, создатели баз данных.

В современном бизнесе многое зависит от самопрезентации компании, ее позиционировании на рынке оказываемых услуг и способности искать новых клиентов и рынки сбыта. Наличие своего сайта это один из имиджевых инструментов, так их как маркетинговых. Web-витрины (Front Office) нужны в сети Интернет торговым компаниям для привлечения возможных покупателей к продуктам и услугам данных компаний.

Для привлечения возможных клиентов к услугам данного дизайн-бюро создается web-сайт, одно из современных средств передачи информации, коммуникативное средство, и, наконец, рекламный продукт, дающий большие возможности в области поиска и привлечения клиентов. Еще одним значимым доводом в пользу создания сайта является то, что сайт – это актуальное, современное средство предоставления информации. Наличие собственного сайта в наше время является правилом хорошего тона и залогом успеха в развитии любого бизнеса.

В настоящее время существует бесчисленное множество сайтов. Все сайты очень разноплановые и отличаются друг от друга по большому количеству параметров.

По типам предоставляемых сервисов Web-сайты можно разделить на коммерческие и некоммерческие. Основной аудиторией коммерческого сайта являются действительные и потенциальные клиенты.

Исходя из функций и свойств сайтов их можно подразделить на:

- информационные сайты,

- визитки,

- электронные магазины,

- корпоративные представительствами,

-системами управления предприятием,

-порталами.

Информационный сайт - это сайт, который обязан содержать исчерпывающую информацию о деятельности компании. Сайты этого типа, как правило, содержат множество информации о своей деятельности, а также используют : связь с клиентами путем опросов, голосования, рассылки. Это сайты некоммерческого типа, хотя и призваны привлечь клиентов.

Цель создания информационного сайта – представить некую информацию в Интернете, сервис и поддержка клиентов, оказание информационных услуг.

Рассмотрим подробнее характеристики информационного сайта:

– предоставление подробной, исчерпывающей информации пользователю;

– объемное количество сервисов: опросы, голосование, чат, призванные сделать сайт интерактивным;

– в зависимости от специфики информационного наполнения, может быть как креативным, так и строго деловым;

– несложная система навигации;

– требует частого обновления;

– поддержка обеспечивается дополнительными усилиями, как правило, командой людей;

– целевая аудитория, случайные посе тители;

Основными задачами информационного сайта являются:

– дать пользователю максимальную и исчерпывающую информацию;

– обеспечить запоминаемость у потенциальных клиентов и посетителей;

– легкость по весу, чтобы у посетителя не было проблем с загрузкой

– оперативное обновление, чтобы обеспечить интерес посетителей.

1.2 Сайт-визитка

Сайт-визитка – это несложный сайт, содержащий общую информацию о компании и роде, оказываемых услуг.

Цель создания сайта-визитки – представить компанию в Интернете. Свойства сайта-визитки:

1)общее назначение сайта-визитки – предоставление общей информации пользователю;

2)характеристика и основные элементы – наличие координат, контакты ключевых людей компании, род деятельности и предоставляемые услуги;

3) количество страниц – 1-3;

4)тип и характеристика дизайна – несложный деловой дизайн;

5)система навигации – максимально удобная и простая система на вигации;

6)частота и необходимость обновления – обновление происходит нечасто, по мере необходимости;

7)кем производится обновление – поддержка либо через фирму- Исполнителя, либо с помощью системы управления сайтом;

8)кем являются посетители – в основном, целевая аудитория, люди, которые ищут определенный товар, услугу;

9)наличие баннеров – баннеры на сайте-представительстве счита ются плохим тоном;

10)проводимая рекламная компания – индексация в поисковых сис темах, регистрация в каталогах, обмен ссылками с дружескими сайтами.

Задачи сайта-визитки:

1)дать пользователю общее представление о компании;

2)максимально заинтересовать и спровоцировать покупку или иное действие;

3)привлечение дополнительных клиентов;

4)обеспечить запоминаемость у потенциальных клиентов и посети телей.

В своем дипломном проекте остановимся именно на сайте-визитке.

Анализ деятельности дизайн-бюро «Надежда»

Основные задачи дизайн-бюро «Надежда» состоят в сле дующем:

­– формирование и корректное представление сведений о предоставляемых услугах;

– формирование круга возможных клиентов;

– непосредственная работа с клиентами (прием заявок, пожеланий);

– формирование портфолио салона;

– формирование сведений о заявках клиентов и сроках их выполнения;

– продвижение дизайн- салона.

Данный комплекс задач характеризуется большим объемом оперативной информации. Использование информационных систем (ИС) (сайта салона) существенно облегчает работу с клиентами, повышает ее оперативность. Использование интернет сайта должно повысить уровень узнаваемости салона, способствовать привлечению новых клиентов и информировать о предоставляемых услугах потенциальных клиентов.

Именно поэтому и возникает необходимость использования ИС для решения данного комплекса задач.

Интернет-сайт салона позволит улучшить следующие процессы:

– поиск клиентов;

– продвижения предоставляемых услуг;

–оперативная связь с потенциальным клиентом.

Развитие российского рынка индустрии дизайна, стремительно набирает темпы. Роль дизайн-салонов в нем огромна. Web- сайт салона превращает почти любую компанию в современный бизнес.

Для анализа рассмотрим web-сайт конкурента: дизайн-студия «Apriori Design», расположенный по адресу: https://apriori.design/dekor/ (рисунок 1).

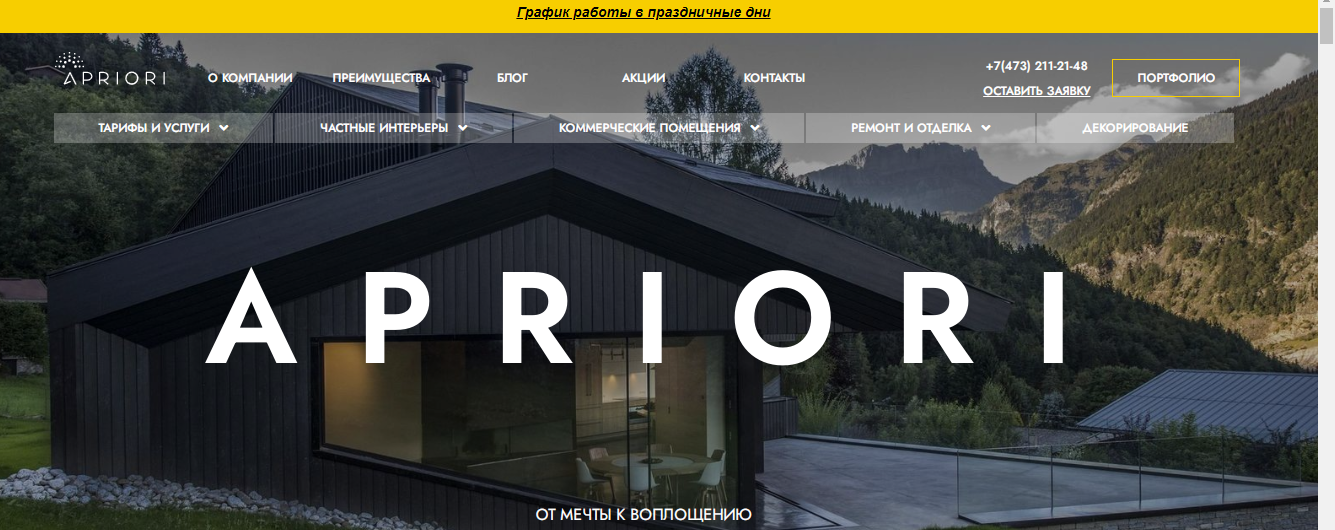


Рисунок 1 – Главная страница сайта SPA-центра красоты и здоровья «Янус»

Достоинства: представлен полный объем информации об услугах и технологиях, сайт достаточно функционален.

Недостатки: непривлекательное оформление, мало имеется фотоматериала портфолио, мелкий трудночитаемый текст.

Следующий web-сайт конкурента: расположенный по адресу:  [https://emihome.ru/](http://spacockteil.ru/) (рисунок 2



Рисунок 2 – Главная страница сайта

Достоинства: на сайте удобная навигация, представлен полный объем информации об услугах, о салоне, в полной мере предоставлен фотоматериал о сотрудниках и интерьере, указана вся необходимая контактная информация.

Недостатки: разная скорость загрузки страниц сайта.

Следующий web-сайт конкурента: Студия архитектуры и дизайна «ElDeco», расположенный по адресу:  [https://eldeco.ru/](%20https://eldeco.ru/%20) (рисунок 3).

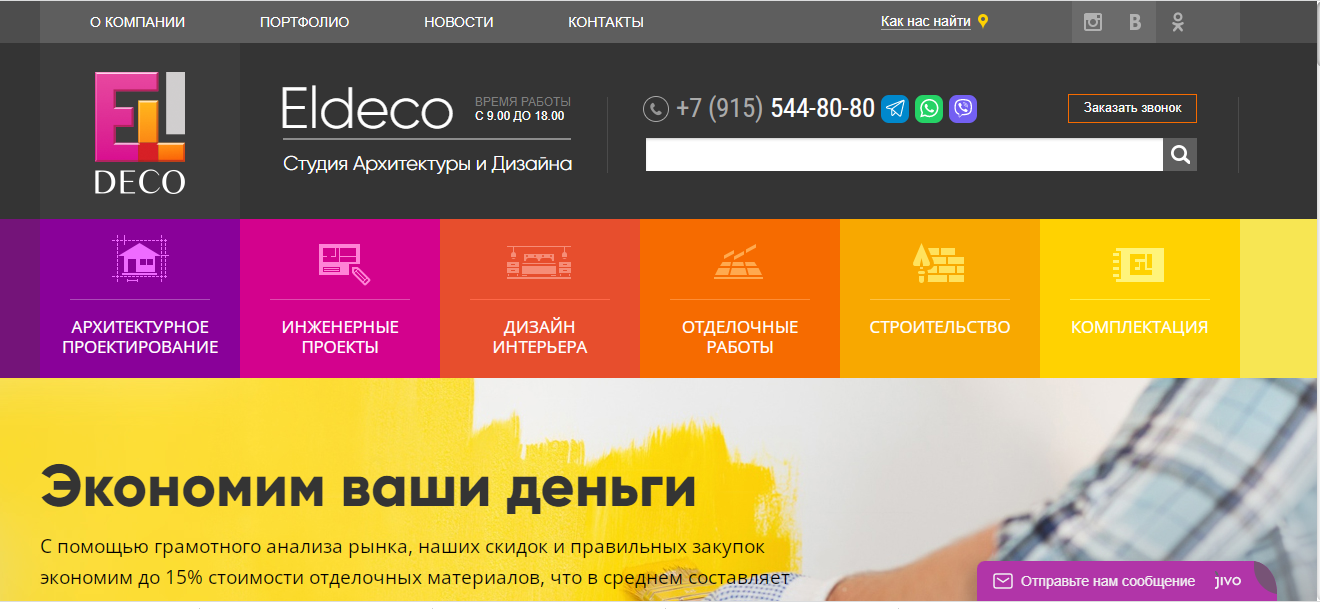


Рисунок 3 – Главная страница Студия архитектуры и дизайна «ElDeco».

Достоинства: привлекательная flash-анимация, представлен полный объем информации об услугах и технологиях, сайт достаточно функционален.

Недостатки: flash-анимация постоянная, что отвлекает от поиска нужной информации.

1.3 Обоснование выбора программного обеспечения

Программное обеспечение принято подразделять на системное, прикладное и инструментальное, а по способу распространения и использования - на закрытое, открытое и свободное.

Разработка моей информационной системы велась при помощи:

HTML – стандартный язык разметки документов в сети Интернет. Большинство web-страниц создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа, в удобной для человека форме;

CSS (каскадные таблицы стилей)– формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство оформления внешнего вида web-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам;

JS - скриптовый язык программирования общего назначения, интенсивно применяемый для разработки web-приложений и является одним из лидеров среди языков программирования, применяющихся для создания динамических web-сайтов;

**React** —библиотека JavaScript для создания пользовательских интерфейсов.  Эта библиотека действительно может упростить жизнь разработчикам . С её помощью можно построить интерфейс из отдельных компонентов, которые легко поддерживать. Она добавляет удобный слой абстракции, избавляя от необходимости работать с DOM напрямую.

React — это уже не новая библиотека, поэтому она отлично протестирована, регулярно поддерживается и стабильно обновляется, а переход к новым версиям проходит максимально гладко. Благодаря сообществу у React хорошо проработанная документация и большой опыт, накопленный в статьях, курсах и конференциях. Это значительно облегчает не только изучение библиотеки новичками, но и поиск ответов на всевозможные вопросы в процессе работы. На Гитхабе можно найти готовые React-компоненты почти на все случаи жизни. А если их нет, но есть нужные независимые библиотеки, то можно поискать интеграцию либо сделать её самостоятельно. В React-сообществе со временем сформировались определённые подходы и соглашения по организации проекта, кода, решению частых проблем. Для разработчиков это значит, что нужно тратить меньше времени на обсуждение некоторых договорённостей — можно использовать готовые и уже устоявшиеся.

Состав команды. Даже если делали все сами, напишите, какие роли вы выполняли на проекте. Хороший пример: Участвовала в командном проекте в роли Java-разработчика. Также на проекте были задействованы: проджект-менеджер, дизайнер и тестировщик. ⠀

Для данного проекта необходимы следующие человеческие ресурсы:

-проджект-менеджер;

-Фронтенд-разработчик;

-дизайнер;

-тестировщик.

ГЛАВА 2. ОСНОВНАЯ ЧАСТЬ ПРОЕТА

2.1 Техническое задание

2.1.1 Название организации-заказчика.

Дизайн-Бюро «Надежда».

Название продукта разработки.

Проектирование и разработка сайта для дизайн-бюро.

Назначение продукта.

Функциональное и эксплуатационное назначение программы привлечение новых клиентов и их информирование об основной деятельности, а также применение сайта как инструмента для более удобной коммуникации между заказчиком и исполнителем .

Необходимо, чтобы пользователь в результате посещения сайта: увидел – основной список услуг, портфолио работ, контакты, смог рассчитать примерную стоимость своего заказа. Почувствовал - доступность и удобство пользования. Связался с разработчиком

2.1.5 Плановые сроки начала и окончания работ**.**

Начало 10 января 2024года, окончание 20 мая 2024 года.

2.2 Характеристика области применения продукта.

2.2.1 Процесс и структура, в которых предполагается использование продукта разработки.

Дипломная работа должна быть выполнена в несколько этапов:

-постановка задачи.

-разработка технического задания.

-проектирование сайта.

-отладка программы.

-разработка программной документации.

-оформление пояснительной записки.

Защита дипломной работы.

2.2.2 Требования к продукту в целом.

Программный продукт должен быть разработан на языке высокого уровня программирования.

Программа должна обеспечивать:

-удобный интерфейс;

-лёгкость в использовании;

Аппаратные требования: Для создания и использования программы

необходимы следующие минимальные аппаратные требования:

-Процессор с частотой 2 ГГц

-Оперативная память – 4ГБ

-Жесткий диск - 20 Гб

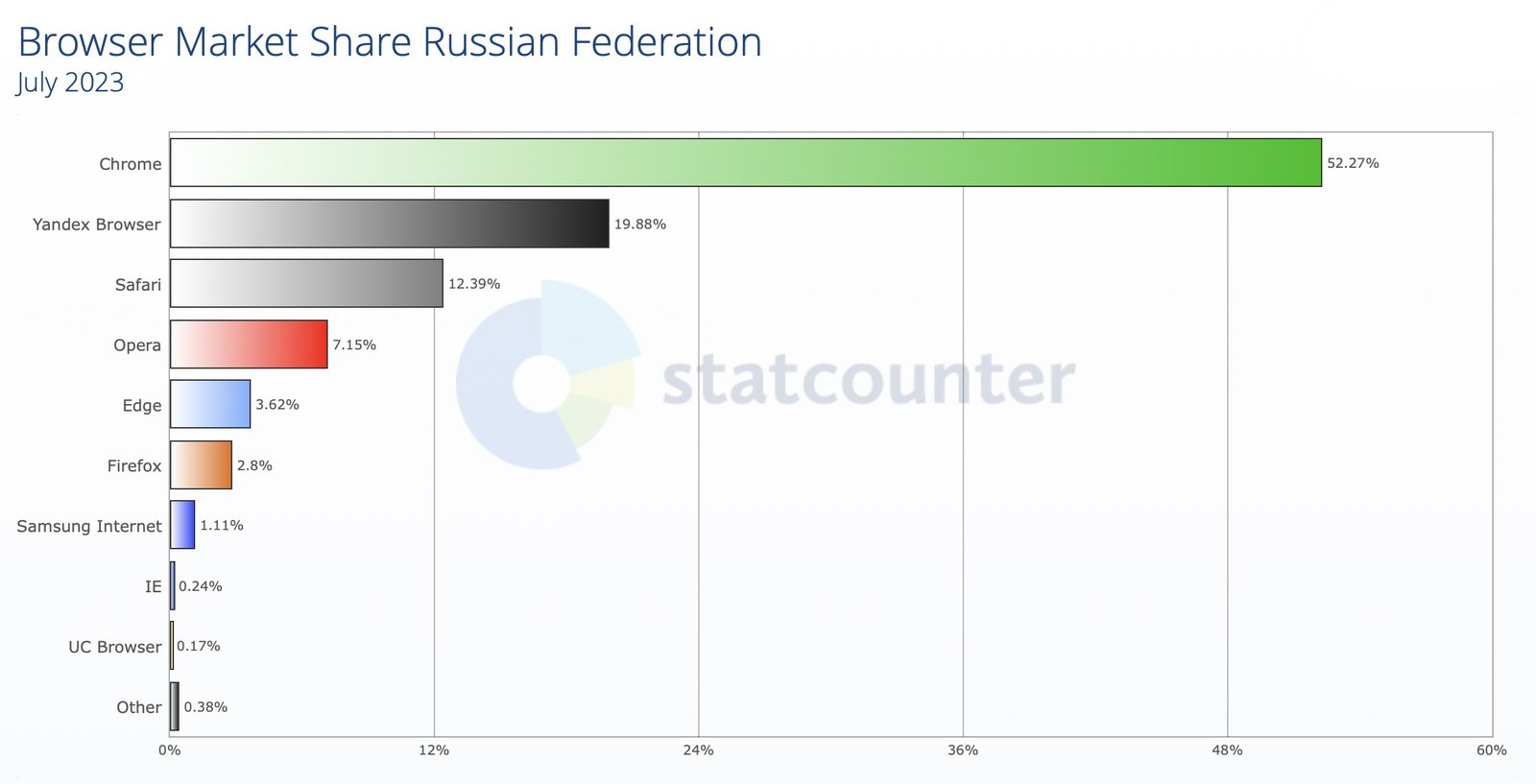
-Видеокарта 1ГБ

Для создания ПО и использование его в дальнейшем необходимо следующее программное обеспечение:

Windows 10 или аналог других ПО.

Разрабатываемый сайт в ориентирован на потребителей данной услуги в России. Для этого необходимо знать самые популярные браузеры России.

По данным иcследованиям компании Hi-Tech на момент июля 2023 года самым популярным браузером в РФ является Google Chrome — каждый второй гражданин пользуется этой программой. На втором месте оказался отечественный «Яндекс Браузер» с 19,88% рынка. Третье место принадлежит Safari (12,39%).



Поэтому при разработки проекта будем использовать Google Chrome /

Рисунок 4 – Иcследованиям компании Hi-Tech».

2.2.3 Программного обеспечения, используемое для реализации.

Для реализации проекта выбраны:Visual Studio Code, графический редактор Figma, растровый графический редактор Adobe **Photoshop**, [JavaScript](https://ru.wikipedia.org/wiki/JavaScript)-[библиотека](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript) с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для разработки [пользовательских интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F)- **React.**

2.3 Требования к интерфейсу сайта.

Оформление сайта должно быть хорошо читаемое. Желательное цветовое решение следующее: спокойная бежево-коричневая цветовая гамма с теплыми оттенками. Стиль сайта -классический с обязательными изображениями интерьеров. На сайте обязательно должен присутствовать Swiper.

Структура разделов сайта:

Главная страница сайта должна содержать максимально полезной информации для клиента:

- информация о компании

- описание услуг

-портфолио работ

-информация о стоимости услуг

-контактные данные

Описание страницы

**На главной странице должно быть:** Меню, Приветствие, Рассказ о деятельности, Каталог работ, Калькулятор стоимости Контактные данные компании.

**Страница о разработчике должна содержать:** Фото дизайнера**,**  ее образование, достижения, награды.

**Swiper**: должен вызываться при клике по картинке проекта представленной на главной странице и содержать коллекцию фото из проекта.

**Калькулятор**: должен представлять выбор категории разработки, поле для ввода метража и поле для вывода результата:

**Услуги :** блок с описанием услуг компании, условиях работы.

**Обратная связь:** блок располагается внизу сайта с информации о контактах и формах связи с разработчиком (адрес, email, ссылка связи в контакте и телеграмм канале.

2.4 Разработка сайта Дизайн-бюро «Надежда».

2.4.1 Дизайн-сайта.

**Разработка** сайта начинается с дизайна. Для того, чтобы пользователи сразу оценили по достоинству дизайн, он должен быть лаконичным, с приятной для глаза цветовой гамме , хорошо читаемым шрифтом. Дизайн сайта будет создаваться в онлайн-редакторе Figma. В этом редакторе очень удобно проектировать интерфейсы, создавать макеты сайтов, мобильных приложений, презентации, иллюстрации, логотипы и анимацию. Этим редактором в основном пользуются дизайнеры, но продакт-менеджерам и разработчикам тоже необходимо разбираться в программе. Так участники проекта будут лучше понимать друг друга и работать над продуктом. Figma предоставляет возможности смотреть макет 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. Редактор открывается через браузер, устанавливать дополнительные приложения не нужно. Это удобно, когда нужно показать макет клиенту или коллегам. Достаточно отправить ссылку и настроить права доступа. Заказчику сайта нет необходимости объяснять словами, как должно работать приложение или сайт, можно создать его прототип — модель будущего продукта. 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.

Подготовка прототипа:

1. Делим главную страницу на блоки и структурируем их.
2. Прорисовываем элементы интерфейса.
3. Определить цветовое решение
4. Посмотреть результат в режиме презентации.

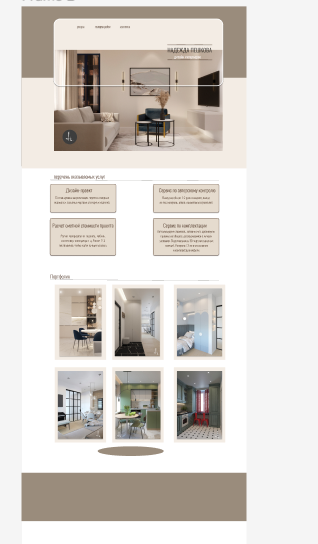


Рисунок5 -Дизайн макет. Главная страница.

Работа с системой начинается с запуска Интернет обозревателя и перехода на хост web-сервера. После перехода открывается главная страница сайта, которая изображена на рисунке 5.

На главной странице в шапке сайта фоном служит изображение интерьера. Отображаются ссылки и кнопки для навигации по сайту для посетителей. Так же на главной странице располагается логотип дизайн бюро «Надежда», который так же является ссылкой на главную страницу сайта.

Справа в «шапке» находится меню сайта – ссылки для навигации по сайту. На рисунке №6 показано данное меню.

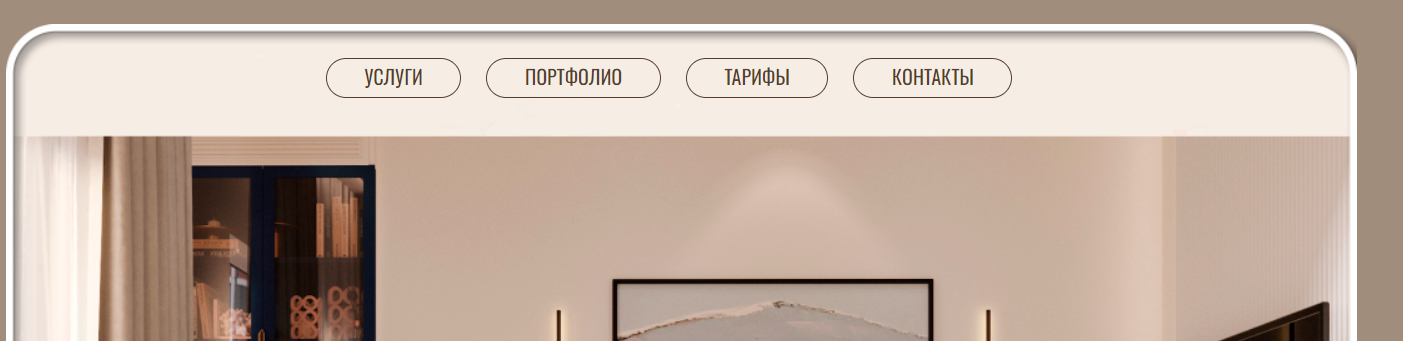


Рисунок №6- Меню главной страницы.

При клике по любой из ссылок пользователь попадает на информацию с соответствующей названию информацией. При нажатии на кнопку меню сайта «Услуги» пользователь попадает на информацию, в которой размещены карточки об услугах дизайн-бюро (рисунок 7).



Рисунок №7-Карточки с услугами дизайн-бюро.

При нажатии на кнопку меню сайта «Портфолио» пользователь попадает на блок, в котором представлены референсы проектов. Каждый референс является одной из карточек проекта. Картинка с изображением интерьера одновременно являются ссылками для вызова галереи с референсами проекта (рисунок 8,9)

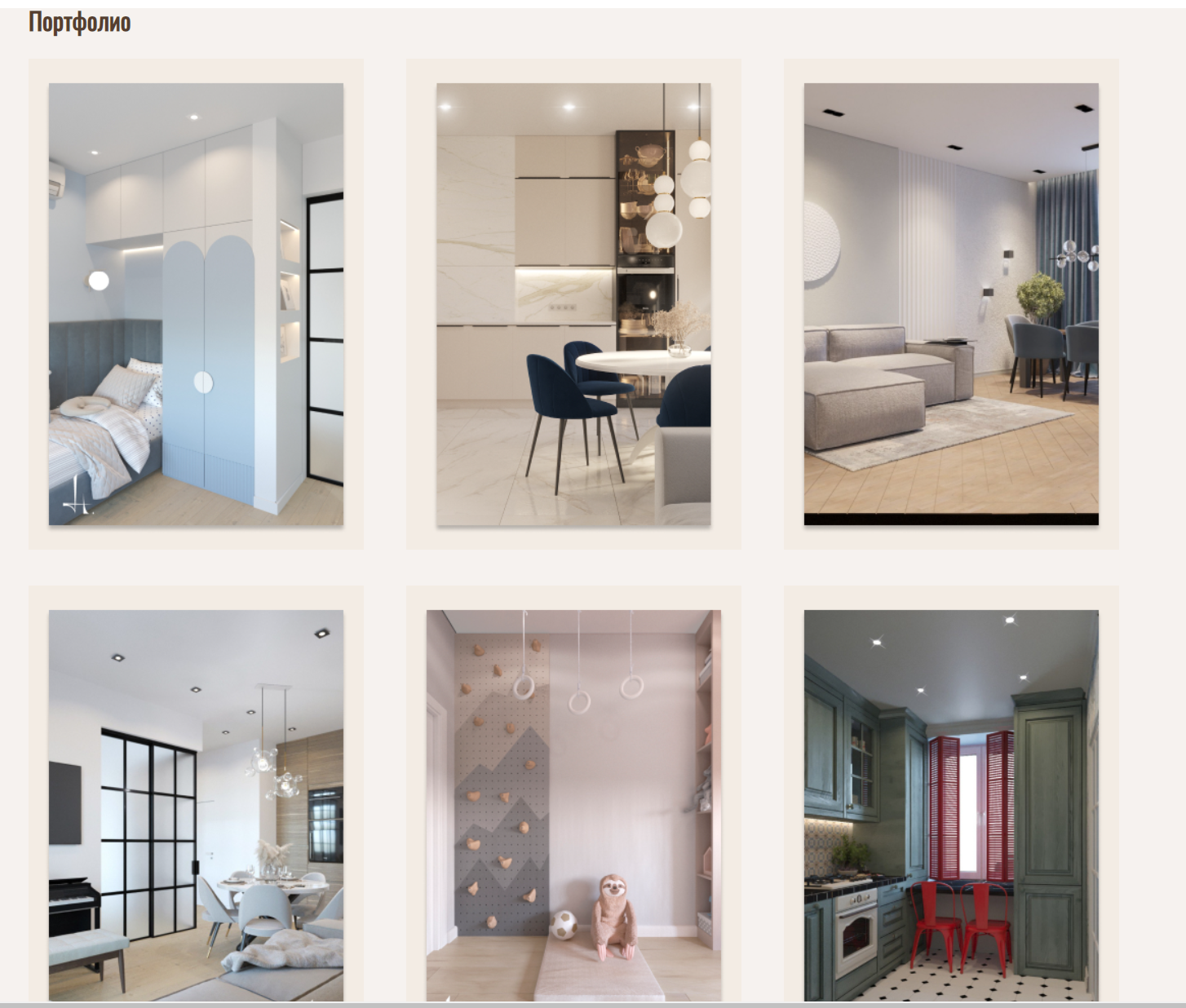


Рисунок №8-Блок с референсами проекта.

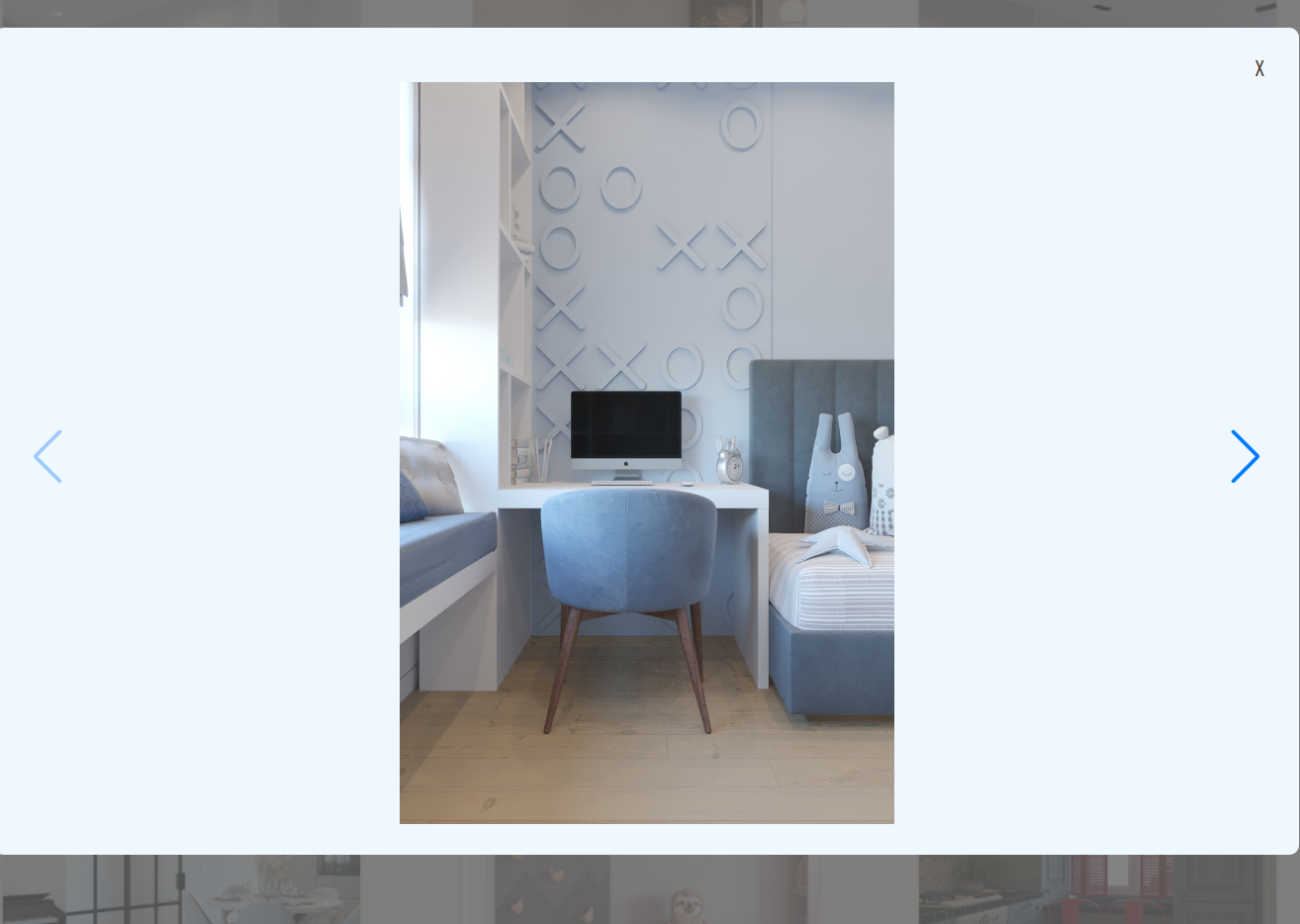


Рисунок №9- Модальное окно с галерей работ

Следующий пункт меню сайта «Тарифы». Пользователь видит краткое описание раздела и прейскурант цен на предоставляемые услуги, может ознакомиться с предлагаемыми услугами оплаты. Также в этом блоке представлена кнопка для вызова калькулятора расчета проекта (рисунок № 10,11)

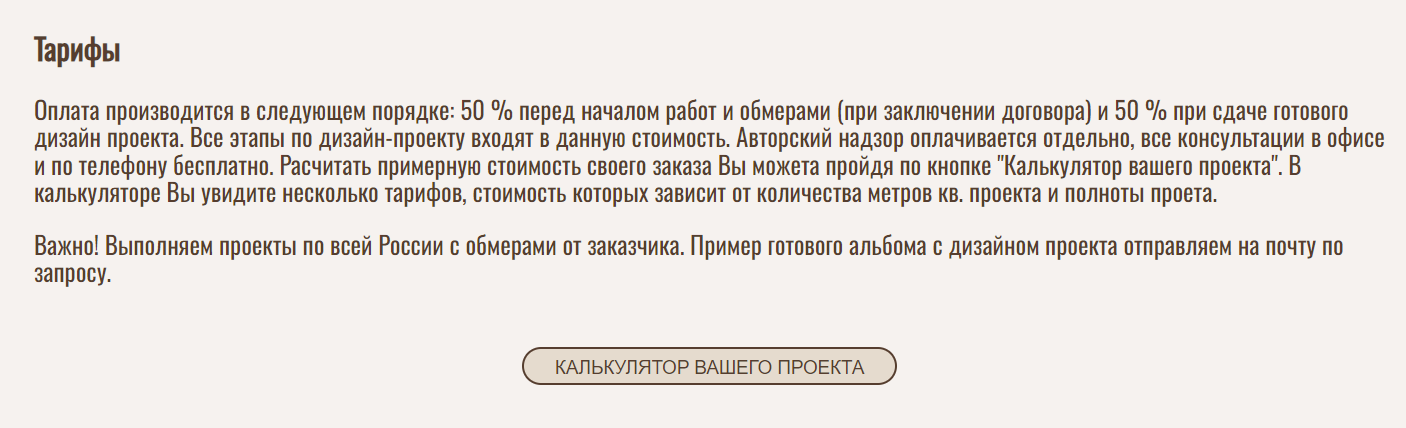


Рисунок №10-Блок тарифы с кнопкой вызова калькулятора проекта.

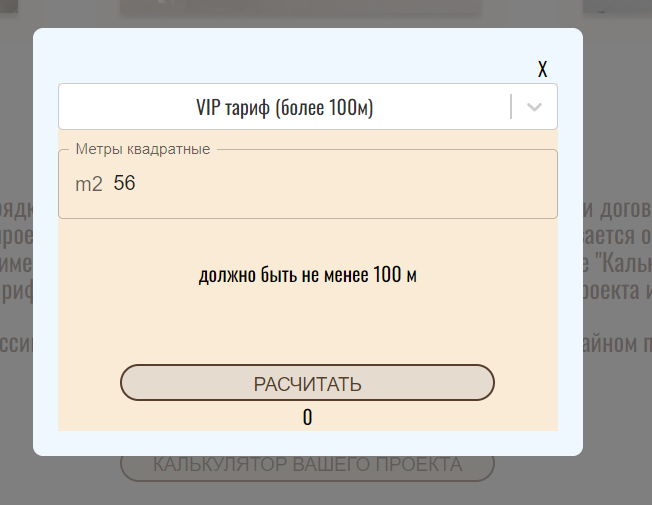


Рисунок №11-Калькулятор проекта.

В подвале будет содержаться информация о дизайнере интерьеров, профилях в социальных сетях и контактной информации. Иконки социальных сетей будут отображаться с помощью переданного массива icons. Пользователь сможет перейти по ссылкам на социальные сети и связаться с дизайнером.

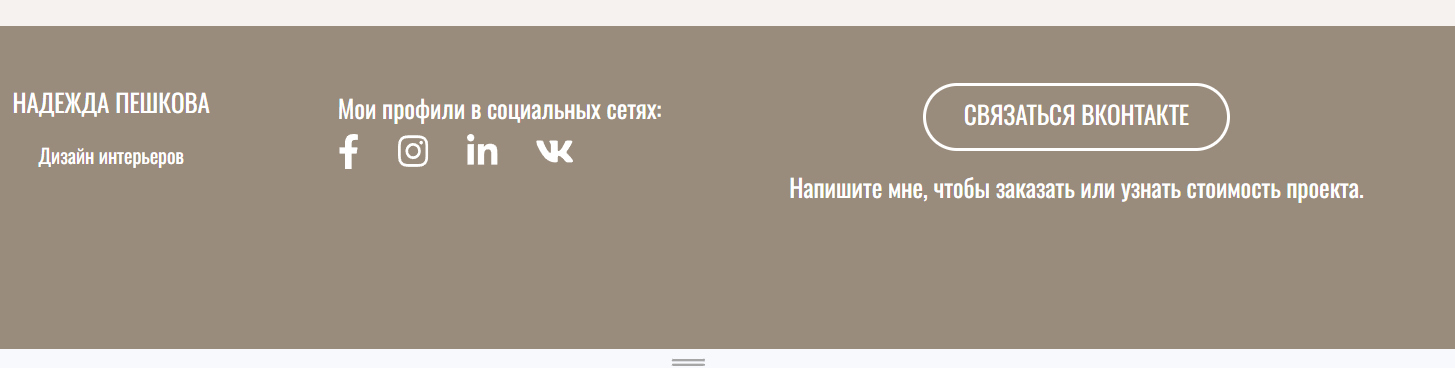


Рисунок №12-Подвал сайта

2.4.2 Разработка структуры программы

Разработка программы, будет являться сложной системой, поэтому необходимо принять меры для ее упрощения. Для этого сайт разрабатывают по частям, которые называются программными компонентами. Это приведет к тому, что каждый компонент программируется, компилируется и отлаживается отдельно от других модулей, и тем самым, физически разделен с другими компонентами программы. Более того, каждый разработанный программный компонент включается в состав основного блока. Таким образом, программный компонент может рассматриваться и как средство борьбы со сложностью программ, (т.е. как средство многократного использования, который с легкостью может быть изменен в дальнейшем).

Основаниями к выделению того или иного модуля в проектируемой системе служит совокупность функций, которые можно реализовать в рамках одного модуля. Например: блок header, основной блок, блок footer.

2.4.3 Обоснование метода разработки структуры программы

В качестве модульной структуры программы принято использовать древовидную структуру, включая деревья со сросшимися ветвями. В узлах такого дерева размещаются программные модули, а направленные стрелки показывают статическую подчиненность модулей, т.е. каждая дуга показывает, что в интерфейсе модуля, из которого она исходит, имеется ссылка на модуль, в который она входит. Другими словами, каждый модуль может обращаться к подчиненным ему модулям, т.е. выражается через эти модули. При этом модульная структура программы, в конечном счете, должна включать и совокупность спецификаций модулей, образующих эту программу.

В процессе разработки программы ее модульная структура может по-разному формироваться и использоваться для определения порядка программирования и отладки модулей, указанных в этой структуре. Обычно рассматривают два метода: восходящей разработки и нисходящей разработки.

Метод восходящей разработки заключается в следующем. Сначала строится модульная структура программы в виде дерева. Затем поочередно программируются модули программы, начиная с модулей самого нижнего уровня (листья дерева модульной структуры программы), в таком порядке, чтобы для каждого программируемого модуля были уже запрограммированы все модули, к которым он может обращаться. После того, как все модули программы запрограммированы, производится их поочередное тестирование и отладка в принципе в таком же (восходящем) порядке, в каком велось их программирование. На первый взгляд такой порядок разработки программы кажется вполне естественным: каждый модуль при программировании выражается через уже запрограммированные непосредственно подчиненные модули, а при тестировании использует уже отлаженные модули. Однако, современная технология не рекомендует такой порядок разработки программы. Во-первых, для программирования какого-либо модуля совсем не требуется текстов используемых им модулей - для этого достаточно, чтобы каждый используемый модуль был лишь специфицирован (в объеме, позволяющем построить правильное обращение к нему), а для тестирования его возможно (и даже, как мы покажем ниже, полезно) используемые модули заменять их имитаторами (заглушками). Во-вторых, каждая программа в какой-то степени подчиняется некоторым внутренним для нее, но глобальным для ее модулей соображениям (принципам реализации, предположениям, структурам данных и т.п.), что определяет ее концептуальную целостность и формируется в процессе ее разработки. При восходящей разработке эта глобальная информация для модулей нижних уровней еще не ясна в полном объеме, поэтому очень часто приходится их перепрограммировать, когда при программировании других модулей производится существенное уточнение этой глобальной информации (например, изменяется глобальная структура данных). В-третьих, при восходящем тестировании для каждого модуля (кроме головного) приходится создавать ведущую программу (модуль), которая должна подготовить для тестируемого модуля необходимое состояние информационной среды и произвести требуемое обращение к нему. Это приводит к большому объему "отладочного" программирования и в то же время не дает никакой гарантии, что тестирование модулей производилось именно в тех условиях, в которых они будут выполняться в рабочей программе .

Метод нисходящей разработки заключается в следующем. Как и в предыдущем методе сначала строится модульная структура программы в виде дерева. Затем поочередно программируются модули программы, начиная с модуля самого верхнего уровня (головного), переходя к программированию какого-либо другого модуля только в том случае, если уже запрограммирован модуль, который к нему обращается. При таком порядке разработки программы вся необходимая глобальная информация формируется своевременно, т.е. ликвидируется весьма неприятный источник просчетов при программировании модулей. Существенно облегчается и тестирование модулей, производимое при нисходящем тестировании программы. Первым тестируется головной модуль программы, который представляет всю тестируемую программу и поэтому тестируется при "естественном" состоянии информационной среды, при котором начинает выполняться эта программа. При этом все модули, к которым может обращаться головной, заменяются на их имитаторы. Каждый имитатор модуля представляется весьма простым программным фрагментом, сигнализирующим, в основном, о самом факте обращения к имитируемому модулю с необходимой для правильной работы программы обработкой значений его входных параметров (иногда с их распечаткой) и с выдачей, если это необходимо, заранее запасенного подходящего результата. После завершения тестирования и отладки головного и любого последующего модуля производится переход к тестированию одного из модулей, которые в данный момент представлены имитаторами, если таковые имеются. Для этого имитатор выбранного для тестирования модуля заменяется на сам этот модуль, и добавляются имитаторы тех модулей, к которым может обращаться выбранный для тестирования модуль. При этом каждый такой модуль будет тестироваться при "естественных" состояниях информационной среды, возникающих к моменту обращения к этому модулю при выполнении тестируемой программы. Таким образом, большой объем "отладочного" программирования заменяется программированием достаточно простых имитаторов используемых в программе модулей. Кроме того, имитаторы удобно использовать для подыгрывания процессу подбора тестов путем задания нужных результатов, выдаваемых имитаторами.

Некоторым недостатком нисходящей разработки, приводящим к определенным затруднениям при ее применении, является необходимость абстрагироваться от базовых возможностей используемого языка программирования, выдумывая абстрактные операции, которые позже нужно будет реализовать с помощью выделенных в программе модулей. Однако способность к таким абстракциям представляется необходимым условием разработки больших программных средств, поэтому ее предпочтение в своей работе я отдаю именно нисходящей разработке [9].

В выбранном мною методе нисходящей разработки модульная древовидная структуру программы должна разрабатываться до начала программирования модулей. На самом деле это делать не обязательно: так существуют подходы к разработке программ, при которых модульная структура формируется в процессе программирования модулей. Но для реализации проектируемого сайта нисходящей разработки вполне достаточно.

2.4.4 Алгоритм системы web-сайта.

Разрабатывая код для сайта будем рассчитывать на то, что он будет реализована в инструментальной системе браузера. Инструментальные системы работают по объектно-ориентированной концепции, при которой пользователю представлено главное окно сайта и его объекты, которые обладают свойствами и событиями. Для событий определяются последовательности действий, называемые обработчиками событий. В качестве событий имеются следующие:

- нажатие на какую-либо кнопку, ссылку;

- переход со страницы на страницу;

- выбор из списка;

- пролистывание галереи.

В качестве объектов, для которых будут проектироваться обработчики событий будут выступать: модальное окно, главное меню сайта, пункты главного меню, окно галереи, окно калькулятора.

Из вышеизложенного следует, что проектирование будет связанного с событиями объектов.

2.4.5 Используемые технологии в создании сайта.

В своей работе над дипломом я остановила свой выбор на JavaScript-библиотеке-React. В проектеReact позволяет выполнить несколько задач:

- показывает на странице компонент интерфейса;

- основывается на идее декларативного программирования, а не на императивном подходе;

-разработчик только описывает, как ведёт себя интерфейс в зависимости от данных и событий.

-позволяет использовать массу фреймворков, достаточн, только их подключить к проекту.

У React масса преимуществ:

- уeниверсальность. Ни одна другая JS-библиотека не работает в таком количестве окружений и на таком количестве устройств.

- гибкость. Он позволяет:

-менять инструменты разработки;

-выбирать, библиотеки например, для контроля за переходами между экранами и хранения данных;

-сделать приложение полностью на React или использовать его только для одной фишки.

-Читаемость. В React приложение разбито на компоненты, поэтому в 9 случаях из 10 при поломке видно, в какой части кода беда:

- разработчик видит, что сломалось;

-идёт в компонент и устраняет проблему для всех идентичных компонентов.

Сегодня разработчики используют React для разных интерфейсов, например:

-во многих пользовательских и внутренних сервисах Яндекса;

-для ленты сайтов и приложений Facebook и Instagram;

-в онлайн-кинотеатре Кинопоиска; на Пикабу;

-в мессенджере WhatsApp; и множестве других интерфейсов.

React существует много лет и за это время:

-успели найти решения для всех типичных проблем;

-придумали тысячи инструментов, которые делают жизнь разработчика проще;

-сформировали вокруг библиотеки гигантскую и постоянно растущую экосистему.

[2.5 Реализация](file:///E:\Downloads%20Trash\Дипломный_проект_Пример,_программист.docx#_Toc132887944) проекта.

2.5.1 Начало реализации сайта на React.

Наш проект начинаем с установки Node.js. Это кроссплатформенная среда для разработки клиентских приложений, в основе которой лежит язык программирования JavaScript.У Node.js открытый код, а построен он на движке Chrome V8, что позволяет переводить JavaScript в машинный код. С помощью этого инструмента можно запускать написанные программы из командной строки прямо на компьютере. Есть возможность установить его на сервер. Также платформу можно назвать событийно-ориентированным I/O фреймворком. Node.js реагирует на действия пользователей, например, в чатах и играх. А технические особенности делают его удобным и легковесным. Скачиваем установщик Node.js с официального сайта (https://nodejs.org) и следуем инструкциям по установке.

1. Создание нового проекта: Открываем командную строку или терминал и переходим в папку, где хотим создать свой проект. Затем выполняем следующую команду для создания нового проекта React:

npx create-react-app <<имя нашего проекта>>;

2. Переход в папку проекта: После создания проекта переходим в его папку с помощью команды:

cd <<имя нашего проекта>>;

3. Запуск сервера разработки: Выполняем следующую команду: npm startю

Это запустит сервер разработки и откроет ваш проект в браузере по адресу http://localhost:3000.

4. Редактирование компонентов: Открываем интегрированную среду разработки и переходим в папку "src" проекта. В этой находитсяфайл "App.js", который является основным компонентом нашего приложения. Он будет редактироваться и создавать новые компоненты для своего сайта;

5. Создание компонентов: Создаем новые файлы с расширением ".jsx" или ".js" в папке "src" и определяем в них свои компоненты

6. Импорт и использование компонентов: Импортируем свои компоненты в файле "App.js" и используем их внутри компонента "App". (Приложение 1. Код "App.js");

7. Добавление стилей: Cтили к своим компонентам будем добавлять используя CSS. Создавая файлы стилей с расширением ".css" и импортируйте их в свои компоненты.

8. Добавление функциональности: На протяжении создания проекта будем добавлять функциональность к вашим компонентам, используя JavaScript или библиотеки и фреймворки, импортируя необходимые модули и используя их в своих компонентах.

2.5.2 Реализация компонента Header.

Компонент **Header** в нашем проекте будет отображать шапку сайта. Внутри компонента Header возвращаем JSX разметку. Внешний div элемент имеет класс "header\_fon" , задаем ему фоновое изображение с помощью CSS свойства background-image. Вложенный div элемент с классом "asdasdasd" устанавливает фоновое изображение, позиционирование фона, а также высоту. Внутри этого div элемента происходит отображение кнопок навигации с помощью метода map(). Для каждого объекта в массиве navigations создается компонент NavigationButton с передачей ему текста и ссылки из объекта «navigations». Также создается константа navigations, которая содержит массив объектов с текстом и ссылками для кнопок навигации. Этот массив будет использоваться для создания кнопок навигации. Результатом выполнения будет отображение шапки сайта с фоновыми изображениями и кнопками навигации. Каждая кнопка будет содержать текст и ссылку, указанные в массиве navigations. (Приложение 2. Код "Header.jsx");

Сами кнопки навигации будут переиспользованы в проекте не однократно, в следствии чего создаем отдельный компонент NavigationButton для кнопки и будем его использовать в своем проекте. (Приложение 3. Код " NavigationButton.jsx");

2.5.3 Реализация компонента отображения списка услуг на сайте.

Компонент Services, отображает список услуг на сайте. Внутри компонента Services возвращается JSX разметка. Список услуг на сайте представляет собой карточки с описанием услуги. Для этого в проекте будем использовать массив. Задаем массив константой cards, который содержит массив объектов с текстом и описанием услуг. Далее определяем функцию convertCardToView, которая принимает объект карточки и возвращает компонент Card (Приложение 1. Код "Card.jsx").с передачей ему текста и описания услуги из объекта. Для каждого объекта в массиве cards вызывается функция convertCardToView. Внешний блок элемента имеет класс "conteiner" и задает ширину, цвет фона, отступы и выравнивание текста. ложенный h2 элемент с классом "serviceTitle" задает заголовок для списка услуг. Div элемент с классом "cardsContainer" использует CSS Grid для создания сетки из карточек услуг. Результатом будет отображение списка услуг на сайте в виде карточек. Каждая карточка будет содержать текст и описание услуги, указанные в массиве cards. (Приложение 4. Код "Services.jsx").

2.5.3 Реализация компонента Portfolio.

Компонент Portfolio будет отображать страницу с карточками дизайна интерьеров. Для этого определяем функциональный компонент Portfolio, который возвращает JSX разметку. Внешний div элемент имеет класс "conteiner" и задает ширину, цвет фона, отступы и выравнивание компонентов на странице. Внутри компонента вкладываем еще один компонент WorkCards -который отображает галерею работ на главной странице. Этот компонент будет отображать фото карточек работ и обрабатывать клики на них. Результатом выполнения этого кода будет отображение страницы портфолио на сайте. Страница будет содержать заголовок "Портфолио" и галерею работ, которая будет отображать фото карточек работ. При клике на фото карточки будет открываться диалоговое окно с галереей референсов из проекта . (Приложение 5. Код “Portfolio”.jsx", “WordCards”.jsx").

Галерею работ будем вызывать в модальном окне, которое можно будет переиспользовать в проекте несколько раз. Данный код представляет собой компонент Dialog, который отображает диалоговое окно на сайте. Для этого

определяем функциональный компонент Dialog, который будет принимать в качестве props следующие значения:

- open: логическое значение, указывающее, открыто ли диалоговое окно или нет;

- onCancel: функция, вызываемая при клике на кнопку закрытия диалогового окна;

- children: содержимое диалогового окна.

Вложенный элемент с классом "cancel" будет содержать кнопку закрытия диалогового окна. При клике на эту кнопку будет вызываться функция onCancel. Вложенный элемент props.children будет отображать содержимое диалогового окна. Результатом диалогового окна на сайте должно быть следующее: если значение props.open равно true. Диалоговое окно будет открыто и содержать кнопку закрытия, при клике на которую будет вызываться функция props.onCancel, а также содержимое, переданное в props.children. Если значение props.open равно false, диалоговое окно не будет отображаться. (Приложение 6. Код "Dialog.jsx").

2.5.4 Создание слайдера на странице сайта .

Создадим компонент PortfolioDetalis, который будет в модальном окне отображать слайдер с изображениями на сайте. Компонент который будет принимает в качестве props следующие значения:

- id: идентификатор карточки, по которой кликнули;

- children: содержимое компонента.

Создаем состояние state с помощью хука useState. Изначально его состояние будет пустое.

Хуки это нововведение в React 16.8, которое позволяет использовать состояние и другие возможности React без написания классов. Это функции, с помощью которых мы можем «подцепиться» к состоянию и методам жизненного цикла React из функциональных компонентов. Благодаря хукам легко повторно использовать логику состояния в разных компонентах. Хуки позволяют разбить один компонент на маленькие функции по их назначению (например, подписке или загрузке данных), а не на основе методов жизненного цикла. Они упрощают организацию кода и позволяют повторно его использовать, избавляют от необходимости понимать, как работает this в JavaScript, поведение которого отличается от большинства языков. Хуки дают новый мощный способ повторного использования кода в компонентах.

Создадим еще один хук useEffect. Внутри которого будет происходит фильтрация массива imageArrays по полю "groop". Если значение props.id не пустое, то создается новый массив selectionArray, в который попадают только те элементы из imageArrays, у которых значение поля "groop" равно значению props.id. Затем новому массиву присвоим состоянию state. Далее будем использовать компонент Swiper из библиотеки swiper/react для отображения слайдера. Swiper JS - это библиотека для создания слайдеров на веб-страницах. Для того, чтобы ее использовать ее необходимо подключить к ее проекту: $ npm install swiper. Затем обязательно импортируем данную библиотеку в наш компонент: import { Swiper, SwiperSlide } from "swiper/react". После устанавливаем настройки слайдера: navigation (добавление кнопок навигации), modules=[Navigation] (подключение модуля навигации), className=style.swiperConteiner (задание класса для контейнера слайдера).

Внутри Swiper будем использовать метод map для отображения каждого элемента из состояния state в виде слайда. Каждый слайд представляет собой компонент SwiperSlide, внутри которого находится изображение с src =value.link и alt=value.alt. Результатом выполнения этого кода будет отображение слайдера на сайте. Слайдер будет содержать изображения из массива imageArrays, которые были отфильтрованы по значению props.id. Каждое изображение будет отображаться в виде слайда внутри компонента Swiper. (Приложение 7. Код “PortfolioDetalis”.jsx").

2.5.5 Pеализация блока Cost.

Компонент Cost, отображает блок "Тарифы" на сайте. Функциональный компонент Cost определяется следующим образом:

- создается состояние state с помощью хука useState. Изначально состояние его функции openCalculator равно false;

- создается еще одна функция toggleCalculator внутри которой происходит изменение состояния openCalculator с помощью функции setOpenCalculator. Если предыдущее значение состояния было false, то теперь оно становится true, и наоборот;

- JSX-разметка компонента Cost в озвращается следующим образом:

- внутри компонента Dialog (опять переиспользуем этот компонент для модального окна) отображается диалоговое окно с калькулятором, если значение openCalculator равно true. При нажатии на кнопку "Отмена" вызывается функция toggleCalculator.

- затем следует блок "Тарифы" с заголовком "Тарифы" и двумя абзацами текста.

- внутри блока "Тарифы" определяем кнопку "Калькулятор вашего проекта", при клике на которую вызывается функция toggleCalculator.

Результатом работы этого компонента будет отображение блока "Тарифы" на сайте. Внутри блока будет текст с описанием тарифов и кнопка "Калькулятор вашего проекта". При клике на кнопку откроется диалоговое окно с калькулятором. При повторном клике на кнопку диалоговое окно закроется. (Приложение 8. Код "Сost.jsx").

2.5.6 Создание калькулятора проекта.

Калькулятор проекта представляет собой компонент CalculatorForm, который отображает калькулятор на сайте. В функциональном компоненте CalculatorForm создается состояние state с помощью хука useState, изначально состояние его переменной result равно 0. Используем еще один хук useForm , с помощью которого создается объект form с несколькими свойствами и методами:

- register используется для регистрации полей ввода формы.

- control используется для управления компонентами формы, особенно для контроля значений полей ввода.

- formState содержит информацию о состоянии формы, включая ошибки валидации.

- handleSubmit вызывается при отправке формы и выполняет указанную функцию onSubmit.

- watch используется для отслеживания изменений значений полей ввода формы.

- reset используется для сброса значений полей ввода формы.

Функция формы onSubmit вызывается при отправке формы и сбрасывает значения полей ввода. Переменные squareMetr и tarif используются для отслеживания значений полей "Метры квадратные" и "Тариф" соответственно.

Функция onCalculate вызывается при клике на кнопку "Расчитать". Внутри функции происходит проверка условия: если выбран тариф "VIP" и количество метров меньше 100, то расчет не производится. В противном случае, результатом расчета является произведение количества метров на выбранный тариф. Затем значения полей ввода сбрасываются. Внутри формы отображаются поле выбора тарифа с использованием компонента ReactSelect. Затем отображается поле ввода "Метры квадратные" с использованием компонента Input. Если есть ошибки валидации для поля "Метры квадратные", они отображаются. Под полем ввода отображается кнопка "Расчитать". Под кнопкой отображается результат расчета в виде суммы в рублях.

Результатом будет отображение калькулятора на сайте. Пользователь может выбрать тариф и ввести количество метров. При нажатии на кнопку "Расчитать" будет произведен расчет и отображен результат в виде суммы в рублях. При использовании в компоненте объекта form будет использоваться Material-UI. Это крупная библиотека, спомошью которой разработчики создают красивый дизайн без использования СSS. В ней находятся React-компоненты, которые дают возможность программисту быстро создать устойчивые формы. Компоненты Material-UI работают изолированно. **Они самодостаточны**и внедрят только те стили, которые им нужны для отображения. Они не зависят от каких-либо глобальных стилей. Устанавливаем данную библиотеку в наш проект npm install@mui/material@emotion/react @emotion/styled. Затем импортируем компонент библиотеки в компонент сайта. import InputAdornment from "@mui/material/InputAdornment"

(Приложение 9. Код " CalculatorForm.jsx").

Для калькулятора будем использовать элементs Input, поэтому создадим отдельный компонент Input. Этот компонент также будем создавать с помощью использования компонент TextField из библиотеки Material-UI для отображения поля ввода на сайте. Внутри компонента возвращается JSX-разметка, которая содержит компонент TextField. TextField принимает несколько пропсов:

- label: используется для отображения подписи к полю ввода.

- variant: указывает стиль поля ввода (в данном случае "outlined").

- value: значение поля ввода.

- onChange: функция, которая вызывается при изменении значения поля ввода.

- type: тип поля ввода (например, "text" или "number").

- placeholder: текст-подсказка, отображаемый в поле ввода до того, как пользователь введет данные.

- name: имя поля ввода.

- InputProps: дополнительные атрибуты для настройки компонента TextField.

В отображеное поля ввода на сайте пользователь может ввести данные и при изменении значения будет вызываться функция onChange. Подпись к полю ввода, стиль и другие атрибуты можно настроить с помощью передачи соответствующих пропсов при использовании компонента Input. (Приложение 9. Код "Input.jsx").

2.5.7 Создание Footer проекта.

Компонент Footer представляет собой отображения подвала (footer) на сайте. Внутри компонента возвращаем JSX-разметка, которая содержит HTML-элементы для отображения подвала.

Подвал состоит из нескольких блоков:

- Блок с классом "conteiner" является контейнером для остальных элементов подвала.

- Блок с классом "footerRow" содержит строки подвала.

- Блок с классом "footerCol" содержит информацию о копирайте и дизайне интерьеров.

- Блок с классом "footerIcons" содержит информацию о профилях в социальных сетях.

- Блок с классом "footerContacts" содержит контактную информацию и кнопку для связи. В блоке "footerIcons" используется массив icons для отображения иконок социальных сетей. Массив содержит объекты, каждый из которых представляет одну иконку и ссылку на соответствующий профиль. С помощью метода map происходит итерация по массиву, и для каждого объекта создается тег <a> с соответствующей ссылкой и <img> с иконкой. Затем полученные элементы добавляются в переменную iconsComponent. После этого отображаем переменную iconsComponent в блоке "footerIconsRow".

В подвале будет содержаться информация о копирайте, дизайне интерьеров, профилях в социальных сетях и контактной информации. Иконки социальных сетей будут отображаться с помощью переданного массива icons. Пользователь сможет перейти по ссылкам на социальные сети и связаться с дизайнером. (Приложение 10. Код "Footer.jsx").

ЗАКЛЮЧЕНИЕ

В любой фирме, как большой, так и маленькой, возникает проблема продвижении услуг. Средствами HTML, CSS, REACT был разработан web-сайт, который позволяет:

-информировать клиентов о предоставляемых услугах;

-привлекать новых клиентов;

-продвигать бренд фирмы.

В процессе создания сайта для дизайнера было выполнено множество задач, начиная от разработки уникального дизайна и продумывания пользовательского опыта, и заканчивая разработкой функциональности и оптимизацией для улучшения производительности. Созданный сайт предоставляет дизайнеру платформу для представления своих работ и привлечения новых клиентов. Он имеет современный и эстетически приятный дизайн, который подчеркивает творческий подход дизайнера и привлекает внимание посетителей.

Функциональность сайта включает в себя портфолио с возможностью просмотра работ в высоком качестве, контактную информацию для связи с дизайнером, а также форму обратной связи для отправки запросов на сотрудничество или получения дополнительной информации. Сайт был разработан с использованием современных технологий и инструментов, что обеспечивает его отзывчивость и отличную производительность на различных устройствах и браузерах. В целом, созданный сайт дает дизайнеру возможность эффективно представить свои работы и привлечь новых клиентов, а также обеспечивает удобство и позитивный пользовательский опыт для посетителей. Использование разработанного программного продукта упрощает процессы продвижения предоставляемых услуг.

В ходе работы над дипломным проектом я овладела:

-языком разметки HTML;

-языком декорирования и описания внешнего вида документа (веб-страницы) СSS;

- языком программирования **JavaScript;**

  -научилась пользоваться библиотекой REACT.

В данный момент Сайт находится в тестовом режиме и в стадии наполнения контентом.

Список используемой литературы

1. Джон Дакетт JAVASCRIPT И JQUERY интерактивная веб-разработка- Москва Издательство «Э» 2017-640с.
2. Современный учебник JavaScript:  офиц. сайт. <https://learn.javascript.ru>.
3. Редакция Дока- поддерживает [Яндекс Практикум](https://practicum.yandex.ru/programming-upskilling/?utm_source=pr&utm_medium=content&utm_campaign=pr_content_programming-upskilling_doka) — сервис онлайн-образования офиц. сайт http: //doka.guide
4. Документация REACT: офиц. сайт. http: ://ru.react.js.org/docs/getting-started.html/

ПРИЛОЖЕНИЕ

**Приложение №1.Код "App.js"**

import "./App.css";

import { Header } from "./components/Header/Header";

import Portfolio from "./components/Portfolio/Portfolio";

import Services from "./components/Services/Services";

import Cost from "./components/Сost/Сost";

import Footer from "./components/Footer/Footer";

function App() {

  return (

    <div className="App">

      <Header />

      <Services />

      <Portfolio />

      <Cost />

      <Footer />

    </div>

  );

}

export default App;

**Приложение2 .Код "Header.jsx"**

import { NavigationButton } from "../shared/NavigationButton/NavigationButton";

import "./Header.css";

export const Header = () => {

  //  Массив для создания кнопок навигации

  const navigations = [

    { text: "Услуги", link: "#deals" },

    { text: "Портфолио", link: "#portfolio" },

    { text: "Тарифы", link: "#tarifs" },

    { text: "Контакты", link: "#contacts" },

  ];

  return (

    <div className="header\_fon">

      <div className="asdasdasd">

        <div className="aaa">

          {navigations.map((navigation) => (

            <NavigationButton text={navigation.text} link={navigation.link} />// Используется образ навигационной кнопки

          ))}

        </div>

      </div>

    </div>

  );

};

**Стили СSS**

.header\_fon{

    background-image: url("./../shared/assets/header\_fon.png");

    width: 100vw;

    align-items: center;

   padding-top: 30px;

}

.asdasdasd{

margin: 0 auto;

align-items: center;

background-image: url("./../shared/assets/picture-background.png");

    background-repeat: no-repeat;

    background-position: center ;

    height: 75vh;

}

**Приложение 3. Код " NavigationButton.jsx"**

import "./NavigationButton.css"

// Создание образа навигационной кнопки сайта

export const NavigationButton =({text, link})=>{

    return <a href={link} className="head-button">

    {text}

</a>

}

**Стили СSS**

.head-button{

    display: inline-block;

    margin-right: 20px;

    height: 30px;

    padding-left: 30px;

    padding-right: 30px;

    border: 1px solid #563E2F;

    border-radius: 60px;

    color: #563e2f;

    line-height: 28px;

    font-weight: 600px;

    font-size: 16px;

    text-transform: uppercase;

    text-decoration: none;

    transition: all 0.2s ease-in;

    margin-top: 30px;

}

/\* поведение кнопок при наведении курсора\*/

.head-button:hover{

    background-color: #9A8C7C;

    color: #fefdfc;

    border: 1px solid #9A8C7C;

}

**Приложение №4. Код "Services.jsx"**

import React from "react";

import styles from "./Services.module.css";

import Card from './Card/Card'

const Services = () =>{

    const cards = [

        {text:"Дизайн-проект", post:"ТЗ, планировка, визуализации, чертежи, сводные ведомости, эскизные чертежи столярных изделий."},

        {text:"Сервис по авторскому контролю", post:"Выезд на объект 1-2 раза в неделю, выезд на тех. контроль, ответы на вопросы строителей.."},

        {text:"Расчет сметной стоимости проекта", post:"Расчет материалов на  отделку,  мебель, сантехнику, электрикуи т. д. Расчет 2-3 поставщиков, чтобы найти лучшие условия."},

        {text:"Сервис по комплектации", post:"Актуализируем стоимость, готовим счет, организуем    приемку на объекте, договариваемся о лучших  условиях. Подготавливаем 3D чертежи заказных позиций. Контроль ТЗ по изготавлению и комплектации мебели."}

    ]

    const convertCardToView = (card)=><Card text={card.text} post={card.post}/>

    return(

        <div className={styles.conteiner}>

            <h2 className={styles.serviceTitle} id="deals">Перечень оказываемых услуг</h2>

            <div className={styles.cardsContainer}>

                { cards.map(convertCardToView)}

            </div>

         </div>

    )

}

export default Services;

**Стили СSS**

.conteiner{

    width: 1081px;

    background-color: #f6f2ef;

    padding-top: 0;

    padding-bottom: 38px;

    color: #563E2F;

    text-align: start;

}

.portfolioTitle {

    font-size: 30px;

    line-height: 17px;

    color: #563E2F;

}

.cardsContainer {

    display: grid;

    grid-template-columns: repeat(2, 1fr);

    grid-template-rows: repeat(2, 1fr);

    grid-gap: 50px;

}

**Приложение 5.Код "Card.jsx"**

import React  from "react";

import styles from "./Card.module.css";

const Card = (props) =>{

    return (

        <div className={styles.card} >

        <h3 className={styles.cardText}>{props.text}</h3>

        <p className={styles.cardPost}>{props.post}</p>

    </div>

    )

}

export default Card;

**Приложение 6. Код “Portfolio”.jsx", “WordCards”.jsx"**

import React from "react";

import styles from "./Portfolio.module.css";

import WorkCards from "./WorkCards/WorkCards";

const Portfolio = () => {

return ( <div className={styles.conteiner}>

<h2 className={styles.portfolioTitle} id='portfolio'>Портфолио</h2>

<WorkCards/>

</div>)

}

export default Portfolio;import style from "./WorkCards.module.css";

import PortfolioDetalis from "../../Slider/PortfolioDetalis";

import React, { useState } from "react";

import Dialog from "../../shared/Dialog/Dialog";

const WorkCards = (props) => {

const [state, setState] = useState(null);

const [openDialog, setOpenDialog] = useState(false)

function setPortfolioId(id) {

setState(id);

setOpenDialog(true)

}

function onDialogCancel(){

setOpenDialog(false)

}

const workPiсtures = [

{ image: "/work/order1.jpg", id: "order1" },

{ image: "/work/order2.jpg", id: "order2" },

{ image: "/work/order3.jpg", id: "order3" },

{ image: "/work/order4.jpg", id: "order4" },

{ image: "/work/order5.jpg", id: "order5" },

{ image: "/work/order6.jpg", id: "order6" },

]; const workCardsComponents = workPiсtures.map((picture) => {

return (

<div>

<img src={picture.image} alt="" onClick={() => setPortfolioId(picture.id)} />

</div>

);

});

return (

<div>

<div className={style.conteinerPicture}>

{workCardsComponents}

</div>

<Dialog open={openDialog} onCancel={onDialogCancel}>{state && <PortfolioDetalis id={state} />} </Dialog>

</div>

);

};

export default WorkCards;

import React from "react";

import style from "./Dialog.module.css";

//диалоговое окно для использования в проекте (в Slider и CalculatorForm)

const Dialog = (props) => {

return (

props.open && (

<div className={style.dialogContainer}>

<div className={style.dialogContent}>

<div className={style.cancel}>

<span onClick={props.onCancel}>X</span>

</div>

{props.children}

</div>

</div>

)

);

};

**Приложение 7. Код "Dialog.jsx**

import React from "react";

import style from "./Dialog.module.css";

//диалоговое окно для использования в проекте (в Slider и CalculatorForm)

const Dialog = (props) => {

  return (

    props.open && (

      <div className={style.dialogContainer}>

        <div className={style.dialogContent}>

          <div className={style.cancel}>

            <span onClick={props.onCancel}>X</span>

          </div>

          {props.children}

        </div>

      </div>

    )

  );

};

import { useEffect, useState } from "react";

import { Swiper, SwiperSlide } from "swiper/react";

import { Navigation } from "swiper/modules";

import "swiper/css";

import "swiper/css/navigation";

import style from "./PortfolioDetalis.module.css";

const imageArrays = [

{ link: "work/order1/children1.jpg", alt: "Img", groop: "order1" },

{ link: "work/order3/flat9.jpg", alt: "Img", groop: "order3" },

…..];

const PortfolioDetalis = (props) => {

const [state, setState] = useState([]);

useEffect(() => {

if (props.id) {

const selectionArray = imageArrays.filter(

(item) => item.groop === props.id

);

setState(selectionArray);

}

}, [props.id]);

return (

<div className={style.PortfolioDetalisConteiner}>

<Swiper

navigation

modules={[Navigation]}

className={style.swiperConteiner}

>

{state.map((value) => (

<SwiperSlide className={style.pictureWork}>

<img src={value.link} alt={value.alt} />

</SwiperSlide>

))}

</Swiper>

</div>

);

};

**Приложение 8. Код "Cost.jsx"**

import React, { useState } from "react";

import style from "./Cost.module.css";

import MyButton from "../shared/Button/MyButton";

import Dialog from "../shared/Dialog/Dialog";

import CalculatorForm from "../CalculatorForm/CalculatorForm";

const Cost = () => {

  const [openCalculator, setOpenCalculator] = useState(false);

  // функция для открытия  диалогового окна с калькулятором

  function toggleCalculator() {

    setOpenCalculator((prev) => !prev);

  }

  return (

    <>

      {/\* // диалоговое окно с  калькулятором\*/}

      <Dialog open={openCalculator} onCancel={toggleCalculator}>

        <CalculatorForm />

      </Dialog>

      {/\* Блок "Тариф" на странице сайта \*/}

      <div className={style.costWork}>

        <h2 className={style.costTitle} id='tarifs'>Тарифы</h2>

        <p>

          Оплата производится в следующем порядке: 50 % перед началом работ и

          обмерами (при заключении договора) и 50 % при сдаче готового дизайн

          проекта. Все этапы по дизайн-проекту входят в данную стоимость.

          Авторский надзор оплачивается отдельно, все консультации в офисе и по

          телефону бесплатно. Расчитать примерную стоимость своего заказа Вы можета

          пройдя по кнопке "Калькулятор вашего проекта". В калькуляторе Вы увидите несколько

           тарифов, стоимость которых зависит от количества метров кв. проекта и полноты проета.

        </p>

        <p>

          Важно! Выполняем проекты по всей России с обмерами от заказчика.

          Пример готового альбома с дизайном проекта отправляем на почту по

          запросу.

        </p>

        <div className={style.calculator}>

          {/\* Переиспользованная кнопка проекта в данном контексте вызывает диалоговое окно \*/}

          <MyButton

            onClick={toggleCalculator}

            className={style.calculatorButton}

          >

            Калькулятор вашего проекта

          </MyButton>

        </div>

      </div>

    </>

  );

};

**Приложение 9. Код " CalculatorForm.jsx".**

import React, { useState } from "react";

import Input from "../shared/Input/Input";

import InputAdornment from "@mui/material/InputAdornment";

import { useForm } from "react-hook-form";

import styles from "./CalculatorForm.module.css";

import { Controller } from "react-hook-form";

import ReactSelect from "react-select";

import MyButton from "../shared/Button/MyButton";

const options = [

  { value: 1400, label: "VIP тариф (более 100м)" },

  { value: 1600, label: "Стандарт" },

  { value: 500, label: "План" },

];

const getValue = (value) => {

  return value ? options.find((option) => option.value === value) : "";

};

const CalculatorForm = (props) => {

   const [result, setResult] = useState(0);

  const {

    register,

    control,

    formState: {

           errors, //свойство проверки на ошибки

    },

    handleSubmit,

    watch,

    reset,

  } = useForm({

    mode: "onBlur",

    defaultValues:{

      tarif: 1600,

      squareMetr: null

    }

  });

  const onSubmit = (data) => {

    reset();

  };

  const squareMetr = watch("squareMetr");

  const tarif = watch("tarif");

    function onCalculate() {

      if (tarif === 1400 && squareMetr < 100) {

        return ; // расчет не производится

      }

      setResult(squareMetr \* tarif);

        reset();

    }

  return (

    <form className={styles.calculationForm} onSubmit={handleSubmit(onSubmit)}>

       <Controller

          control={control}

          name="tarif"

          render={({ field: { onChange, value }, fieldState: { error } }) => (

            <div>

              <ReactSelect

                placeholder="Taриф"

                options={options}

                value={getValue(value)}

                onChange={(newValue) => onChange(newValue.value)}

                              />

              {error && <div style={{ color: "red" }}>{error?.message}</div>}

            </div>

          )}

          />

      <Input

        className={styles.calculationInput}

        label="Метры квадратные"

        sx={{ m: 1, width: "25ch" }}

        type="number"

        placeholder="Введите количество метров"

        InputProps={{

          ...register("squareMetr", {

            required: "Поле обязательно к заполнению", //проверка на заполнение формы

            min: {

              // проверка на введенное кол-во метров

              value: tarif === 1400 ? 100 : 0,

              message: "должно быть не менее 100 м",

            },

          }),

          startAdornment: <InputAdornment position="start">m2</InputAdornment>,

        }}

      />

      <div>

        {errors?.squareMetr && <p>{errors?.squareMetr?.message || "Error!"}</p>}

      </div>

      <div>

        <MyButton

          variant="outlined"

          onClick={onCalculate}

         >

          Расчитать

        </MyButton>

<div className={styles.calculationResult}> <span >{`${result} `}рублей </span> </div>

       </div>

    </form>

  );

};

**Приложение 10. Код "Input.jsx"**

import React from "react";

import { TextField } from "@mui/material";

// создание образа Input - из библиотеки material ui

// атрибуты передаем через пропсы из CalculatorForm

const Input = (props) => {

  return (

      <TextField

        label={props.label}

        variant="outlined"

        value={props.value}

        onChange={props.onChange}

        type={props.type}

        placeholder={props.placeholder}

        name={props.name}

        InputProps={props.InputProps}

      />

  );

};

**Приложение 11. Код "Footer.jsx".**

import React from "react";

import styles from "./Footer.module.css"

const Footer = () => {

// Массив иконок для использования их в footer cайта

    const icons = [

        {icon:"/footerIcons/facebook.svg", link:"https://google.com"},

        {icon:"/footerIcons/instagram.svg"},

        {icon:"/footerIcons/linkedin.svg"},

        {icon:"/footerIcons/vk.svg"},

    ]

// Map массива

     const iconsComponent = icons.map(({icon, link}) =>{

        return(

            <a href={link} ><img src={icon} alt=""/></a>

        );

     })

  return (

    <div className={styles.footer}>

      <div className={styles.conteiner}>

        <div className={styles.footerRow}>

          <div className={ styles.footerCol}>

            <div className={styles.footerCopyrightName} id='contacts'> НАДЕЖДА ПЕШКОВА</div>

            <p>Дизайн интерьеров</p>

          </div>

          <div className={styles.footerIcons}>

            <p>Мои профили в социальных сетях:</p>

            <div className={styles.footerIconsRow}>

              {/\* определение местоположения икон массива \*/}

               {iconsComponent}

            </div>

          </div>

          <div className={styles.footerContacts}>

            <a href="https://vk.com/architect\_peshkova" className={styles.footerButton}>

              Связаться Вконтакте

            </a>

            <p>Напишите мне, чтобы заказать или узнать стоимость проекта.</p>

          </div>

        </div>

      </div>

    </div>

  );

};