ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современные музеи стремятся к активному использованию цифровых технологий для привлечения посетителей, расширения аудитории и создания интерактивных образовательных сред. С развитием информационных технологий музеи сталкиваются с необходимостью адаптировать свои методы работы под современные реалии. Традиционные методы привлечения посетителей не всегда эффективны в условиях современного общества. Онлайн-ресурсы и веб-сайты могут значительно расширить аудиторию музея, предоставив возможность ознакомиться с ним.

Народный музей истории школы имени Александра Федоровича Тягачева является единственным музеем при школе в Дмитрове, который могут посетить не только школьники в образовательных целях, но и жители города и туристы. Данный музей уже имеет свой веб-сайт, однако проблема заключается в том, что его обслуживание закончилось, большая часть функционала не работает, информация не актуальна, а дизайн морально устарел. В рамках внутреннего рейтинга средних общеобразовательных школ проводятся разные конкурсы, которые повышают рейтинг школы. Одним из плюсов является богатая история школы, чем и может похвастаться СОШ №1 благодаря своему музею, однако перед руководством стоит задача обновить веб-сайт музея.

Актуальность обусловлена необходимостью создания нового информационного веб-сайта, поскольку это позволит повысить рейтинг школы. Веб-сайт предоставит возможность посетить музей виртуально, информировать о новых событиях и достижениях музея, а также расширит аудиторию.

Целью данной дипломной работы является разработка веб-сайта школьного музея с использованием современных технологий веб-разработки.

Для достижения этой цели перед нами стоят следующие задачи:

1. Изучение существующих веб-сайтов музеев и анализ их функциональных возможностей.

2. Определение требований и функциональных возможностей веб-сайта школьного музея.

3. Разработка дизайна и структуры веб-сайта с учетом практик веб-дизайна и пользовательского опыта.

4. Реализация функционала веб-сайта, позволяющего ознакомиться с новостями и информацией о мероприятиях, записываться на экскурсии, а также проводить виртуальную экскурсию.

5. Тестирование и анализ эффективности разработанного веб-сайта.

6. Размещение готовой версии веб-сайта на сервере.

Объектом исследования является веб-сайт музея, а предметом исследования – разработка и внедрение веб-сайта музея.

Теоретико-методологические основы работы:

1. Принципы веб-разработки. Изучение основных принципов разработки веб-приложений, включая выбор языков программирования, фреймворков и инструментов для создания сайта школьного музея.

2. Теория дизайна пользовательского опыта. Анализ теоретических основ дизайна пользовательского интерфейса и пользовательского опыта с целью создания удобного и интуитивно понятного веб-сайта для различных категорий пользователей.

3. Современные тенденции в веб-дизайне. Изучение современных трендов и подходов в веб-дизайне, включая адаптивный дизайн, анимацию, интерактивность и прочие элементы, которые могут быть использованы для улучшения пользовательского опыта.

4. Требования к доступности и удобству использования. Изучение требований и рекомендаций по созданию доступных веб-сайтов для пользователей с ограниченными возможностями.

5. Безопасность веб-приложений. Изучение основных принципов обеспечения безопасности веб-приложений, включая защиту от взлома, защиту персональных данных пользователей и обеспечение конфиденциальности информации.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ ВЕБ-САЙТОВ

1.1 Теоретические основы веб-сайтов

Веб-сайт (или сайт) – это коллекция веб-страниц, связанных между собой гиперссылками и размещенных на веб-сервере. Он является частью Всемирной Паутины (World Wide Web) и предоставляет доступ к различным видам информации, услугам или товарам через интернет.

Всемирная паутина, или система гипертекстовых документов World Wide Web (WWW) является одним из важнейших сервисов, благодаря которому пользователи могут быстро находить и использовать информацию, хранящуюся на множестве различных компьютеров в любой части Интернета.

Интернет – это коммуникационная сеть и всемирная система объединённых компьютерных сетей для хранения и передачи информации. Интернет построен на базе стека протоколов. На основе Интернета работает Всемирная паутина и множество других систем передачи данных.

Интернет можно представить как набор связанных между собой сетей компьютеров. В этой сети действуют компании, которые называются поставщиками услуг Интернета (ISP). Они имеют свои серверы и каналы для передачи данных различных уровней. Большие провайдеры предоставляют свои ресурсы региональным провайдерам, которые, в свою очередь, предоставляют доступ к Интернету конечным пользователям - как частным лицам, так и организациям.

Проблема в том, что компьютеры в Интернете могут различаться по аппаратным и программным характеристикам, имеют разное программное обеспечение и разные способы подключения к сети. Из-за этого Интернет становится сложной сетью, и задача обеспечения стабильного взаимодействия между компьютерами становится сложной.

Для решения этой проблемы был разработан многоуровневый подход. Он предполагает, что взаимодействие между любыми двумя компьютерами в Интернете можно рассматривать как последовательное применение

определенного набора технологий. Каждый уровень этого процесса выполняет определенную подзадачу взаимодействия, а затем передает результат следующему уровню. Идея состоит в том, чтобы каждый уровень обеспечивал не только интерфейс с соседними уровнями, но и с аналогичным уровнем на другом компьютере. Этот набор технологий, который позволяет двум компьютерам "понимать" друг друга, называется стеком протоколов Интернета.

Интернет-протоколы

Базовый уровень сети Интернет работает с использованием двух основных протоколов Internet Protocol (IP) и Transmission Control Protocol (TCP), на которые часто ссылаются совместно, как TCP/IP. Они позволяют передавать данные в виде специально оформленных пакетов ограниченного размера.

Эта модель представляет собой четыре уровня для взаимодействия компьютеров:

1. Прикладной уровень занимается обработкой информации, которую отправляет или принимает программа на компьютере.

К протоколам прикладного уровня относят:

- HTTP (Hypertext Transfer Protocol) – используется для передачи гипертекстовых документов веб-сервером браузеру пользователя. Этот протокол позволяет просматривать веб-страницы и обмениваться данными между веб-сервером и клиентским браузером.
- FTP (File Transfer Protocol) – предназначен для передачи файлов между компьютерами в сети. FTP позволяет загружать и скачивать файлы с удаленных серверов, а также управлять файлами на удаленном сервере.
- SMTP (Simple Mail Transfer Protocol) – используется для отправки электронной почты от клиента к серверу почты и между серверами почты. SMTP обеспечивает доставку электронной почты через Интернет.
- POP3 (Post Office Protocol version 3) и IMAP (Internet Message Access Protocol) – протоколы, используемые для получения электронной

почты с сервера почты на клиентское устройство. POP3 загружает сообщения с сервера и удаляет их, тогда как IMAP позволяет оставлять копии сообщений на сервере.

- DNS (Domain Name System) – протокол, используемый для преобразования доменных имен в IP-адреса и наоборот. DNS обеспечивает разрешение имен и управление доменными именами в сети Интернет. данными с другими устройствами в сети.

2. Транспортный уровень отвечает за связь между компьютерами и обработку передачи данных.

К протоколам транспортного уровня относят:

- TCP (Transmission Control Protocol) – обеспечивает надежную передачу данных путем установления соединения между отправителем и получателем. TCP гарантирует доставку данных в правильной последовательности и обнаружение потерь или повреждений данных.
- UDP (User Datagram Protocol) – предоставляет без соединения, ненадежную доставку данных. UDP используется там, где требуется более быстрая передача данных и возможность передачи потока данных без установления соединения и контроля ошибок.

3. Сетевой (межсетевой) уровень обеспечивает взаимодействие между различными сетями.

К протоколам сетевого уровня относят:

- IP (Internet Protocol) – это основной протокол сетевого уровня, который управляет маршрутизацией и адресацией данных в Интернете. Он присваивает каждому устройству уникальный IP-адрес и определяет, как данные должны быть направлены к своему конечному пункту назначения.

4. Канальный уровень определяет методы связи внутри одной сети.

К протоколам канального уровня относят:

- Ethernet – это один из самых распространенных протоколов канального уровня, используемый в проводных локальных сетях (LAN). Он

определяет способы доступа к среде передачи данных, формат кадров данных и методы обнаружения ошибок.

- Wi-Fi (802.11) – это набор стандартов беспроводной связи, которые определяют методы доступа к беспроводной среде и передачу данных между устройствами в беспроводной локальной сети (WLAN).
- Bluetooth – это протокол беспроводной связи, используемый для обмена данными между близлежащими устройствами, такими как смартфоны, планшеты и персональные компьютеры.

Адрес в Интернете

Каждое устройство, подключенное к сети, имеет свой уникальный IP-адрес, который используется для идентификации и обмена данными с другими устройствами в сети.

IP-адрес состоит из четырех чисел, разделенных точками, например: 192.168.0.1. Каждое число в этой записи может быть от 0 до 255, что позволяет существовать около 4,3 миллиардов уникальных комбинаций IP-адресов.

Существует две версии IP-адресов: IPv4 (Internet Protocol version 4) и IPv6 (Internet Protocol version 6). IPv4 является более распространенным и используется сегодня, но из-за ограниченного количества доступных адресов (из-за роста количества подключенных к Интернету устройств) постепенно внедряется IPv6, который предоставляет намного большее количество уникальных адресов.

Порт – это числовой идентификатор, который определяет конкретный сетевой интерфейс или процесс на устройстве. Порты используются для управления передачей данных между программными приложениями на различных устройствах. Каждый порт связан с определенным сетевым протоколом и используется для определенного типа обмена данными.

Сокет – это абстрактный конечный пункт взаимодействия между процессами на разных устройствах в компьютерной сети. Сокет представляет собой комбинацию IP-адреса и порта. Он используется для установления соединения между клиентским и серверным приложениями, а также для

передачи данных между ними. Сокеты могут быть как потоковыми (TCP), так и датаграммными (UDP), в зависимости от типа передачи данных.

Таким образом, IP-адрес и порт вместе идентифицируют уникальный адрес устройства и его конкретное приложение в сети, а сокет используется для установления связи между устройствами и передачи данных между ними.

Но обычным людям было бы неудобно ориентироваться в IP-адресах. Для удобства людей были придуманы доменные имена — это уникальные имена, которые используются для идентификации компьютера или ресурса в сети интернет. Доменное имя служит для замены числового IP-адреса компьютера более удобным для запоминания символьным именем. Например, доменное имя "example.com" может соответствовать IP-адресу "192.0.2.1". Доменные имена используются в URL-адресах веб-сайтов, в адресах электронной почты и в других приложениях сети интернет.

URL (Uniform Resource Locator) – это адрес, который используется для указания местонахождения ресурса в сети Интернет. Он определяет протокол доступа к ресурсу (например, HTTP, HTTPS, FTP), доменное имя (или IP-адрес) сервера, на котором расположен ресурс, и путь к конкретному файлу или странице на этом сервере. Например, в URL "https://www.example.com/index.html" протокол доступа – HTTPS, доменное имя – www.example.com, а путь к файлу – /files/index.html. URL используется в веб-браузерах для открытия веб-страниц, в программных приложениях для доступа к ресурсам в Интернете и в других ситуациях, когда требуется указать местонахождение веб-ресурса.

Рисунок 1.1 – Пример URL

Протокол HTTP

HTTP (Hypertext Transfer Protocol) – это протокол передачи гипертекста, который используется для передачи данных между веб-серверами и

клиентскими устройствами, такими как веб-браузеры. Он является основным протоколом, используемым во всемирной паутине (World Wide Web) для передачи веб-страниц, изображений, видео и других мультимедийных данных.

Принцип работы HTTP основан на модели запрос-ответ: клиент отправляет HTTP-запрос на сервер, а сервер отвечает на него, предоставляя запрошенные данные. Запрос может содержать различные методы, такие как GET (получение данных), POST (отправка данных на сервер), PUT (обновление данных на сервере) и DELETE (удаление данных на сервере).

Хотя HTTP широко используется для передачи веб-контента, он может быть небезопасным для передачи конфиденциальной информации, поэтому часто используется расширение HTTPS, которое обеспечивает шифрование и безопасность данных.

Архитектура

Архитектура сайта – это структура и организация веб-сайта, включающая в себя размещение контента, пользовательский интерфейс, взаимодействие с базой данных и другие компоненты.

Веб-сайт обычно работает по модели клиент-сервер, где браузер клиента запрашивает ресурсы у сервера, который отправляет запрошенные данные обратно на клиент.

Фронтенд. Фронтенд включает в себя пользовательский интерфейс и весь контент, который видит пользователь на веб-странице. Это HTML, CSS и JavaScript, которые определяют структуру, стиль и поведение веб-страницы.

HTML (HyperText Markup Language) – это язык разметки документов, используемый для создания веб-страниц. Он представляет собой набор тегов и атрибутов, которые определяют структуру и содержание веб-документа.

CSS (Cascading Style Sheets) – каскадные таблицы стилей, используемые для оформления внешнего вида веб-страниц. Они определяют стилизацию элементов HTML, таких как цвета, шрифты, размеры, расположение и другие аспекты визуального представления.

JavaScript – это высокоуровневый язык программирования, который используется для создания интерактивных и динамических веб-страниц. Он широко применяется как на стороне клиента (в браузере пользователя), так и на стороне сервера (с помощью платформы Node.js).

Непосредственно вся страница, что видит пользователь, состоит из тегов HTML, подключенных стилей и скриптов.

Бэкенд. Бэкенд – это серверная сторона веб-сайта, которая обрабатывает запросы от клиентов, взаимодействует с базой данных, обеспечивает безопасность и управляет бизнес-логикой. Он может быть написан на различных языках программирования, таких как Python, Java, PHP и другие.

База данных. База данных используется для хранения и организации данных, таких как пользовательская информация, контент сайта, настройки и т. д. Различные системы управления базами данных (СУБД), такие как MySQL, PostgreSQL, MongoDB, используются для управления данными на сервере.

Клиентская часть и серверная взаимодействует друг с другом через API – набор правил и инструкций, которые определяют, как различные программные компоненты могут взаимодействовать между собой. Он обеспечивает способ связи между различными приложениями, сервисами или компонентами программного обеспечения.

API часто использует HTTP для обеспечения коммуникации между клиентом и сервером. Клиент отправляет запросы HTTP на сервер, а сервер отвечает на них, предоставляя запрошенные данные или выполняя запрошенные операции.

Данные, возвращаемые по API, часто форматируются в JSON (JavaScript Object Notation) – формат обмена данными, основанный на синтаксисе JavaScript, который используется для передачи структурированных данных между приложениями. JSON представляет данные в виде пар "ключ-значение" и может быть легко интерпретирован как человеком, так и компьютером. Поскольку JSON предоставляет простой и удобный способ представления

структурированных данных, JSON-данные могут быть легко переданы по сети в сообщениях HTTP.

Резюмируя, HTTP используется для отправки запросов и получения ответов между клиентом и сервером, API определяет, какие запросы и ответы могут быть отправлены, а JSON используется для форматирования данных, которые передаются между ними.

1.2 Способы разработки и проектирования веб-сайтов

Существует несколько способов разработки и проектирования сайтов, которые могут варьироваться в зависимости от целей проекта, доступных ресурсов и предпочтений разработчика.

1. Ручное создание с использованием HTML/CSS/JavaScript.

Этот метод подразумевает создание веб-страниц и веб-сайта вручную с использованием языков разметки HTML для структуры, CSS для внешнего оформления и JavaScript для интерактивных элементов.

Подходит для небольших проектов или для разработчиков, имеющих хорошее понимание языков веб-разработки.

Преимущества:

- Полный контроль. При ручном создании веб-сайта вы имеете полный контроль над каждым аспектом его разработки, включая дизайн, функциональность и производительность.
- Уникальный дизайн. Вы можете создать уникальный и индивидуальный дизайн, не ограничиваясь готовыми шаблонами или темами, что позволяет вашему сайту выделяться среди конкурентов.
- Оптимизация производительности. Ручное создание позволяет оптимизировать производительность вашего сайта, учитывая все технические аспекты, такие как минимизация и сжатие файлов, оптимизация изображений и т.д.
- Более гибкая адаптация. Вы можете легко адаптировать сайт под различные устройства и разрешения экрана, что важно для создания

адаптивного дизайна и обеспечения удобства использования на мобильных устройствах.

- Более эффективная оптимизация SEO. При ручной разработке вы можете более эффективно оптимизировать сайт для поисковых систем, включая управление мета-тегами, структурированные данные и другие факторы, влияющие на ранжирование.

Недостатки:

- Требует времени и усилий. Ручное создание веб-сайта требует значительных временных и трудовых затрат, особенно если вы не имеете достаточного опыта или знаний в веб-разработке.
- Сложность обновлений и поддержки. После создания сайта может потребоваться значительное время и усилия на его обновление, поддержку и решение проблем, особенно при необходимости внесения изменений в код.
- Возможные ошибки и баги. При ручной разработке сайта существует риск допущения ошибок и багов в коде, особенно при отсутствии должной экспертизы или тестирования.
- Ограниченная масштабируемость. Ручное создание может ограничить масштабируемость вашего проекта, особенно если требуется быстрое расширение функциональности или добавление новых возможностей.
- Высокие затраты на разработку и поддержку. Ручное создание сайта может быть затратным в плане времени, трудозатрат и финансов, особенно если требуется найм опытных специалистов для разработки и поддержки.

2. Использование CMS (Content Management System).

CMS (система управления контентом) – это программное обеспечение, которое позволяет управлять созданием, редактированием и управлением контентом на веб-сайте без необходимости знания программирования. Она предоставляет удобный интерфейс для добавления, изменения и удаления различных элементов контента, таких как текст, изображения, видео, а также управления структурой сайта и его функциональностью.

CMS обеспечивает возможность создания различных типов веб-сайтов, начиная от блогов и корпоративных сайтов, заканчивая интернет-магазинами и порталами. Они обладают гибкими настройками и расширяемым функционалом, что позволяет адаптировать сайт под различные потребности. CMS предоставляет удобный интерфейс для добавления, редактирования и управления контентом на сайте. Пользователи могут легко создавать и публиковать новые статьи, страницы, изображения и другие элементы контента без необходимости знания HTML или CSS.

Многие CMS предоставляют возможность управления пользователями и определения их ролей и прав доступа. Это позволяет организовать работу команды и обеспечить безопасность сайта.

Также CMS поддерживают систему плагинов и модулей, которые позволяют добавлять дополнительные функции и возможности на сайт. Это позволяет расширить функциональность сайта без необходимости изменения его основного кода.

Преимущества использования CMS:

- Простота использования. Большинство CMS предоставляют удобный интерфейс, который позволяет пользователям без специальных знаний веб-разработки легко создавать, редактировать и управлять контентом на сайте.
- Бесплатный доступ. Почти все CMS (кроме 1С-Битрикс) изначально бесплатны, а кроме того, в сети существует множество готовых шаблонов сайтов под них (особенно под WordPress).
- Гибкость и расширяемость. CMS обладают разнообразными возможностями настройки и расширения функциональности. Они позволяют адаптировать сайт под конкретные потребности и масштабировать его с ростом бизнеса.
- Большое сообщество и поддержка. Популярные CMS имеют огромное сообщество пользователей и разработчиков, что обеспечивает

доступ к множеству документации, форумов поддержки, тем и плагинов для расширения функциональности сайта.

- Управление контентом. CMS предоставляют удобные средства управления контентом, позволяя добавлять, редактировать и удалять статьи, изображения, видео и другие элементы контента без необходимости внесения изменений в код сайта.

Недостатки CMS:

- Ограничения в дизайне. Некоторые CMS могут иметь ограниченные возможности кастомизации дизайна, особенно если использовать готовые шаблоны. Это может ограничить возможности для создания уникального и индивидуального дизайна сайта.
- Производительность. Некоторые CMS могут иметь недостаточно оптимизированный код или тяжелые процессы обработки, что может негативно сказываться на производительности сайта, особенно при большом объеме контента или высоких нагрузках.
- Уязвимость сайта. Самый важный минус любой широко распространенной CMS. Даже сайты, сделанные на конструкторах порой лучше защищены от взлома, чем те, которые находятся под управлением WordPress и других CMS. Даже платная 1С-Битрикс признана далеко не самой надежной и защищенной от атак и проникновения;
- Требования к знаниям. Разработка сайта на CMS уже требует от клиента базовых знаний по верстке и программированию (в основном, на языке PHP), что уже добавляет сложностей при создании сайта. Хотя, стоит заметить, что знания могут потребоваться тогда, когда клиент хочет добавить в имеющийся шаблон новый функционал или создать полностью с нуля собственный проект.

CMS предоставляют удобное решение для создания различных типов веб-сайтов с разнообразным функционалом и контентом, особенно в случаях, когда требуется быстрое развертывание и легкое управление веб-сайтом без глубоких технических знаний.

Примеры CMS: WordPress, Joomla, Drupal, Magento и Shopify.

3. Конструкторы сайтов.

Конструктор сайтов – специальный сервис, который позволяет создавать сайты без знания языков программирования.

Конструкторы сайтов обычно предоставляют простой и интуитивно понятный интерфейс, который позволяет пользователям создавать сайты методом "перетащи и брось", без необходимости писать код. Они предлагают широкий выбор готовых шаблонов, которые можно использовать в качестве основы для создания сайта. Пользователи могут настраивать дизайн, выбирая цвета, шрифты, макеты, легко добавлять текст, изображения, видео, формы обратной связи и другие элементы на страницы сайта.

Преимущества использования конструкторов:

- Простота использования. Конструкторы сайтов предоставляют интуитивно понятный интерфейс, который позволяет даже начинающим пользователям легко создавать и редактировать веб-сайты без необходимости знания HTML, CSS или JavaScript.
- Быстрое создание сайта. Благодаря предварительно созданным шаблонам и готовым элементам, конструкторы сайтов позволяют быстро создать веб-сайт всего за несколько часов, что особенно удобно для небольших проектов или временных сайтов.
- Низкая стоимость. Почти все конструкторы изначально бесплатны, а стоимость подключаемых модулей очень низкая.
- Готовые функциональные решения. Многие конструкторы сайтов предлагают готовые модули и функции, такие как формы обратной связи, галереи изображений, блоги, магазины и др., что позволяет расширить функционал сайта без необходимости программирования.
- Адаптивный дизайн. Многие конструкторы сайтов автоматически создают адаптивные сайты, которые корректно отображаются на различных устройствах, что повышает удобство использования сайта для посетителей.

Недостатки конструкторов сайтов:

- Ограниченные возможности кастомизации. В сравнении с созданием сайта с нуля или с использованием CMS, конструкторы сайтов могут иметь ограниченные возможности кастомизации, что может ограничить возможности для реализации уникальных дизайнов или функционала.
- Тяжеловесность сайта. Сайт, сделанный на конструкторе всегда будет загружаться дольше аналогичного сайта, сделанного на CMS или разработанного самостоятельно. Объясняется это тем, что конструктор содержит в себе огромное количество программного кода, который не относится к Вашему сайту, но необходим для построения его итогового внешнего вида.
- Домен. При использовании бесплатных аккаунтов в конструкторах сайтов, клиентам предоставляется возможность размещения своего сайта только на домене третьего уровня или выше, например, sitename.constructor.com. Однако такие домены обычно не вызывают большого доверия у пользователей интернета. Домены второго уровня (например, sitename.com) стоят дороже, чем при их приобретении у регистраторов доменов напрямую.
- Ограниченный функционал. Некоторые конструкторы сайтов могут предлагать ограниченный функционал или не поддерживать определенные требования вашего проекта, что может оказать влияние на его развитие и расширение в будущем.
- Ограничения по SEO. Некоторые конструкторы сайтов могут иметь ограниченные возможности по оптимизации для поисковых систем, что может сказаться на видимости и поисковом рейтинге вашего сайта в поисковых результатах.

Конструкторы сайтов могут быть полезны во многих случаях, особенно если у человека нет технических навыков веб-разработки или нет времени на создание сайта с нуля.

Конструктор подойдёт для создания простых лендингов или визиток. Для создания серьезного веб-сайта, крупного проекта блочный конструктор не

подойдет. Он просто не сможет реализовать весь функционал, который должен быть на ресурсе такого уровня.

Примеры конструкторов: Webflow, Wix, Squarespace.

4. Индивидуальное разработка с использованием фронтенд и бэкенд фреймворков или библиотек.

Этот подход подразумевает использование специализированных фронтенд и бэкенд фреймворков (например, React.js/Vue.js/Angular.js для фронтенда и Node.js/Django/Express.js для бэкенда) для создания кастомизированных и сложных веб-приложений.

Подходит для больших проектов или проектов, требующих высокой степени гибкости и функциональности.

Преимущества:

- Эффективность разработки. Использование готовых фронтенд и бэкенд фреймворков позволяет ускорить процесс разработки за счет готовых компонентов, модулей и инструментов, что экономит время и ресурсы.
- Стандартизация. Фреймворки предоставляют стандартизированные методы разработки, что облегчает совместную работу команды разработчиков и обеспечивает согласованность кода.
- Большая гибкость. Готовые фронтенд и бэкенд фреймворки обеспечивают гибкость и масштабируемость проекта. Они позволяют легко адаптировать и расширять функциональность приложения с помощью готовых расширений и плагинов.
- Современные технологии. Фронтенд и бэкенд фреймворки часто используют современные технологии и лучшие практики разработки, что обеспечивает высокую производительность, безопасность и масштабируемость вашего веб-приложения.
- Комплексные решения. Фронтенд и бэкенд фреймворки предлагают комплексные решения для различных аспектов разработки, включая маршрутизацию, управление состоянием, аутентификацию, авторизацию, базы данных и многое другое.

Недостатки:

- Сложность обучения. Для работы с фреймворками нужно знать JavaScript, HTML, CSS, а также сам фреймворк. Некоторые фреймворки могут иметь высокий порог входа, особенно для новичков в разработке, что может требовать дополнительного времени и усилий для изучения.

1.3 Формирование требований к разрабатываемому веб-сайту

Требования к сайтам могут сильно варьироваться в зависимости от типа бизнеса, целей сайта и потребностей аудитории. Существуют общие требования, которые часто учитываются при создании сайтов:

1. Адаптивный дизайн.

Это означает, что дизайн сайта должен быть разработан таким образом, чтобы автоматически адаптироваться к различным размерам экранов, на которых пользователи просматривают сайт, включая мобильные устройства, планшеты, ноутбуки и настольные компьютеры.

Адаптивный дизайн включает в себя использование гибких сеток, медиазапросов и других техник, позволяющих оптимизировать отображение и удобство использования сайта на разных устройствах.

2. Безопасность.

Это одно из наиболее важных требований для любого сайта. Оно включает в себя меры для защиты сайта от различных видов кибератак, таких как взломы, утечки данных, атаки на инфраструктуру и другие.

Меры безопасности могут включать использование SSL-шифрования для защиты передаваемых данных, регулярные аудиты безопасности, обновление программного обеспечения, реализацию контроля доступа и аутентификации, а также защиту от вредоносных программ.

3. Быстрая загрузка.

Это требование связано с оптимизацией производительности сайта, чтобы он был быстро доступен и загружался на устройствах пользователей.

Оптимизация скорости загрузки может включать в себя сжатие изображений и видео, минимизацию запросов к серверу, использование кэширования и CDN, оптимизацию кода и т. д.

4. SEO-оптимизация.

SEO (Search Engine Optimization) – это процесс оптимизации сайта для улучшения его позиций в результатах поисковых систем, что помогает привлечь больше органического трафика. SEO-оптимизация включает в себя различные практики, такие как оптимизация контента для ключевых слов, правильное использование метатегов, уникальные заголовки страниц, внутреннее и внешнее ссылочное строительство и другие.

5. Интуитивная навигация.

Навигация по сайту должна быть понятной и легкой для использования, чтобы пользователи могли быстро и легко находить нужную информацию.

Интуитивная навигация включает в себя использование четкой структуры сайта, наглядных меню, правильного оформления ссылок и кнопок, а также логическое расположение контента.

6. Поддержка мобильных устройств.

Сайт должен быть доступен и удобен для использования на различных мобильных устройствах.

Поддержка мобильных устройств включает в себя адаптивный дизайн, оптимизацию для сенсорных устройств, использование каскадных меню и другие техники, чтобы обеспечить удобство использования сайта на мобильных устройствах.

7. Качественный контент.

Это означает, что контент сайта должен быть информативным, релевантным и привлекательным для целевой аудитории.

Контент может включать в себя тексты, изображения, видео, аудио и другие форматы, которые представляют ценность для пользователей и помогают им решить свои задачи или проблемы.

8. Визуальное привлечение.

Это требование связано с дизайном сайта, который должен быть привлекательным, современным и соответствующим бренду.

Визуальное привлечение включает в себя использование красочных и выразительных изображений, приятных цветовых схем, читабельных шрифтов и эффектных анимаций, чтобы привлечь внимание пользователей.

9. Интерактивность.

Это требование предполагает наличие интерактивных элементов на сайте, которые позволяют пользователям взаимодействовать с контентом и друг с другом.

Интерактивность может включать в себя формы обратной связи, комментарии, форумы, голосования, анимации, слайдеры и другие элементы, которые делают сайт более динамичным и увлекательным для пользователей.

10. Поддержка.

Это требование предполагает наличие механизмов связи с пользователями для предоставления им помощи и поддержки при необходимости. Поддержка может включать в себя контактные формы, онлайн-чаты, техническую поддержку по электронной почте или телефону, а также раздел FAQ (часто задаваемые вопросы) для ответов на типичные вопросы пользователей.

11. Соответствие законодательству.

Это требование связано с соблюдением законов и нормативных актов в области защиты данных, авторских прав, прав потребителей и других правовых норм. Это включает в себя разработку политики конфиденциальности, соблюдение правил GDPR (Общего регламента по защите данных), использование лицензированных материалов и другие меры для защиты прав и интересов пользователей.

ГЛАВА 2. АНАЛИЗ ПРЕДМЕТА ИССЛЕДОВАНИЯ

2.1 Характеристика и анализ деятельности СОШ №1

МОУ Дмитровская средняя общеобразовательная школа № 1 имени В.И. Кузнецова – одно из ведущих общеобразовательных учреждений района (входит в пятёрку лучших школ района). В 1876 году началась история школы № 1, бывшей городской женской прогимназии. Школа имеет статус общеобразовательного учреждения и аккредитацию на проведение основных образовательных программ для учащихся начальной и средней школы.

Школа реализует основную общеобразовательную программу, ориентированную на формирование учащихся как грамотных и образованных граждан. Важнейшими аспектами деятельности школы являются следующие: работа с одаренными и способными детьми, их поиск, выявление и развитие, гражданско-патриотическое воспитание учащихся.

Учебные помещения оборудованы современной мебелью и техникой. Школа имеет компьютерные классы с доступом в интернет, библиотеку с обширным фондом книг и журналов, спортивный зал и площадку на улице для занятий физкультурой и спортом, актовый зал для проведения мероприятий, школьный музей.

Школа активно поддерживает внеклассную и внешкольную деятельность. Ежегодно проводятся культурные мероприятия, концерты, спортивные соревнования, экскурсии и тематические мероприятия. В школе действуют различные кружки и секции по интересам учащихся.

На базе Народного музея истории школы им. А.Ф. Тягачева работает районный Клуб краеведов и руководителей школьных музеев. Школьный музей неоднократный победитель областных конкурсов. Он создает условия для работы по патриотическому воспитанию детей на примере старших поколений.

Музей истории школы имени Тягачева является уникальным учреждением, сохраняющим историю и традиции школы. Данный музей является единственным музеем при школе в Дмитрове. Он был основан в 1964

году и с тех пор является центром культурного и образовательного наследия для учеников, преподавателей и гостей школы. Экспозиция музея включает в себя широкий спектр материалов, начиная от документов, свидетельствующих о первых шагах школы, и заканчивая современными достижениями и наградами учащихся, фотографии, учебные пособия, артефакты, персональные вещи и письма, относящиеся к истории школы и не только.

Музей предлагает образовательные программы для учащихся школы и туристов с целью ознакомления с историей учебного заведения, уважения к историческому наследию, гражданско-патриотического воспитания. Экскурсии, мастер-классы и лекции проводятся для учащихся и гостей разных возрастов.

В данный момент школа участвует в историческом конкурсе. В настоящее время тема патриотизма и истории родной страны как никогда актуальна, поэтому наличие музея с уникальными экспонатами времен ВОВ является привилегией и возможностью победить в конкурсе и поднять рейтинг школы. Однако конкурс проходит дистанционно, поэтому чтобы заявить о музее необходим красивый и удобный веб-сайт, который отразит деятельность музея и позволит посетить музей онлайн. Бонусом будет привлечение внимания новых посетителей, учеников и их родителей к истории школы.

Веб-сайт народного музея им. А.Ф. Тягачева уже существует, но были выявлены проблемы:

1. Веб-сайт больше не поддерживается.
2. Устаревший дизайн не соответствует нынешним стандартам.
3. Виртуальный гид не работает.
4. Карта расположения музея не работает.
5. Информация на сайте не актуальна.
6. При открытии сайта на мобильных устройствах дизайн не подстраивается под расширение экрана пользователя, нет адаптивного дизайна.

Руководство школы дало задание на разработку нового веб-сайта, который соответствует требованиям:

1. Современный дизайн, интуитивно понятный пользователю.
2. Адаптивный дизайн под все устройства.
3. Доступность сайта: пользователи без мышек смогут управлять сайтом с помощью клавиатуры, скринридер должен озвучивать все элементы сайта.
4. Сайт должен иметь такие страницы, как: главная страница с историей музея, новости, документы, контакты, гид.
5. У новостей должны быть заголовок, текст новости, фото, ссылки, файлы.
6. Иметь возможность администратору сайта выполнить вход.
7. Администратор должен иметь возможность добавлять новости, редактировать их и удалять, прикреплять файлы и фото и соответственно удалять их.
8. Администратор должен иметь возможность добавлять и удалять документы.
9. Пользователь должен иметь возможность оставить заявку на посещение музея.
10. Веб-сайт должен быть безопасным, пароль администратора никто не может перехватить.

2.2 Анализ средств разработки веб-сайтов

Было принято решение писать веб-сайт самостоятельно, без использования конструкторов и CMS, т.к. это позволит сделать сайт уникальным в плане дизайна, безопасным, оптимизированным и доступным. Исходя из требований, простой сайт-визитка не подойдет, так как нужно, чтоб пользователь и администратор взаимодействовали с контентом. На чистом JavaScript, HTML, CSS это очень ресурсозатратно и энергоемко, поэтому более оправданным решением является написание сайта с помощью фреймворков.

Итак, нужно определиться с фреймворком, языком программирования для бэкенда, базой данных и средой разработки.

Фреймворков для создания пользовательских интерфейсов очень много, поэтому рассматриваться будут самые популярные, т.к. у них большие сообщества, а следовательно, и лучше поддержка.

1. React

React – это одна из самых популярных JavaScript библиотек для создания пользовательских интерфейсов, которую создали разработчики Facebook. Библиотеку начали использовать на сайте этой социальной сети в 2011 году. А в 2013 году Facebook открыл исходный код React. Его используют как крупные компании, такие как Facebook, Instagram, Airbnb, Netflix, так и небольшие стартапы и индивидуальные разработчики.

С помощью React разработчики создают веб-приложения, которые обновляют содержимое страницы без необходимости перезагрузки. Благодаря этому, сайты моментально реагируют на действия пользователя, такие как заполнение форм, применение фильтров, добавление товаров в корзину и т. д.

React используется для отрисовки компонентов пользовательского интерфейса и может полностью управлять фронтендом. В этом случае, для управления маршрутизацией и состоянием, разработчики могут использовать дополнительные библиотеки, такие как Redux и React Router.

Преимущества:

- Быстрая разработка. React основан на компонентах, что делает код модульным, понятным и легко переиспользуемым. Их легко создавать в рамках этой экосистемы. Программистам не нужно с нуля продумывать логику, достаточно только описать состояние. Один раз созданный компонент можно использовать во многих частях проекта. Все это ускоряет разработку и оптимизирует затраты труда разработчика.
- Виртуальный DOM. Использование виртуального DOM позволяет React эффективно обновлять только те части страницы, которые действительно изменились, что повышает производительность приложений.

- Односторонний поток данных: React следует принципу "одностороннего потока данных", что упрощает управление состоянием приложения и делает его более предсказуемым.
- Расширяемость. Существует множество дополнительных библиотек и инструментов, таких как Redux, React Router, Material-UI, которые расширяют функциональность React и помогают разработчикам создавать более сложные приложения.
- Активное сообщество. React имеет большое и активное сообщество разработчиков, которые делятся опытом, создают инструменты и библиотеки, и поддерживают развитие экосистемы React. Новичкам будет легко найти ответы на свои вопросы или попросить помощи
- Документация. React имеет понятную документацию, которая поддерживается разработчиками.

Недостатки:

- Зависимость от обновлений. React —это постоянно развивающаяся библиотека. Это означает, что каждое очередное обновление может “ломать” какие-то части кода и требовать изменений в работе. Особые сложности могут возникнуть при наличии большой кодовой базы и отсутствии времени для отслеживания последних обновлений.
- Смешение представления и логики по умолчанию. В React по умолчанию отсутствует четкое разделение между логикой компонента и его представлением. Вместо этого в компоненте содержится функция render, которая возвращает JSX – синтаксис. Это может быть недостатком, так как это может привести к ситуации, когда логика и код представления перемешаны внутри компонента, что затрудняет его понимание и поддержку.
- Зависимость от сторонних библиотек. Поскольку React является лишь библиотекой, он не поставляется с официальными инструментами для выполнения стандартных задач веб-приложений, таких как управление маршрутизацией, выполнение HTTP-запросов и прочее. Это имеет как свои плюсы, так и минусы.

2. Angular.

Angular – это фреймворк, разработанный и поддерживаемый Google, который предоставляет широкий спектр инструментов и функций для создания масштабируемых и производительных веб-приложений.

Angular используется для создания различных видов веб-приложений, включая корпоративные панели управления, электронные коммерции, административные панели, CRM-системы и другие приложения с высокой степенью интерактивности и сложности.

Преимущества:

- Компонентная архитектура. Angular основан на компонентной архитектуре, что делает код модульным, понятным и легко переиспользуемым. Это значительно упрощает задачу разработчиков. Они могут обновлять отдельные функциональные части приложения, не беспокоясь о возможных негативных воздействиях на работу других компонентов. Кроме того, компоненты могут быть повторно использованы в различных частях приложения.

Разработчики создают компонент один раз и могут использовать его в различных участках приложения. В результате, компонентная архитектура способствует повышению понятности кода, что значительно облегчает процесс вхождения новых разработчиков в проект.

- Мощные инструменты. Фреймворк предоставляет широкий спектр инструментов, таких как Angular CLI, Angular Material, Angular Forms, которые упрощают и ускоряют процесс разработки. Angular предлагает широкий спектр встроенных инструментов и библиотек для разработки, что делает процесс создания сложных веб-приложений более удобным. Это означает, что разработчикам не нужно тратить время на поиск и интеграцию сторонних библиотек.

Вместо этого, они могут воспользоваться предоставленными инструментами для разработки, тестирования и развертывания приложений.

Среди этих инструментов CLI (интерфейс командной строки), инструменты для работы с формами, маршрутизацией, HTTP-запросами и многие другие.

- Межплатформенность. Angular предоставляет возможность создавать приложения для различных платформ, включая веб, мобильные и десктопные. Это обеспечивает гибкость в разработке и упрощает процесс создания приложений, так как разработчики могут использовать одни и те же навыки и инструменты для разработки приложений для различных платформ.

Недостатки:

- Перегруженность и сложность. Angular, как мощный и гибкий фреймворк, может быть слишком тяжелым для небольших проектов или простых веб-страниц. В таких случаях более подходящими альтернативами могут быть React или Vue.js.

Кроме того, Angular предоставляет множество встроенных возможностей и настроек, что может сделать его использование сложным, особенно для новичков. Но в то же время, это также обеспечивает гибкость и масштабируемость, необходимые для разработки сложных веб-приложений.

- Производительность. Хотя Angular был оптимизирован для повышения производительности, в некоторых случаях он может оказаться менее эффективным по сравнению с некоторыми другими фреймворками, такими как React. Однако при правильной оптимизации и использовании функций, таких как ленивая загрузка и дифференциальная загрузка, Angular может обеспечить отличную производительность для большинства веб-приложений.

- TypeScript. Несмотря на то, что TypeScript имеет множество преимуществ, его использование может стать преградой для некоторых разработчиков, особенно для тех, кто не знаком с этим языком программирования. Это особенно актуально для разработчиков JavaScript, которые не привыкли к строгой типизации.

3. Vue.js

Vue.js – это открытый исходный код фреймворк, который предоставляет инструменты для создания интерактивных пользовательских интерфейсов и одностраничных приложений. Vue создан бывшим инженером Google – Эваном Ю.

Известные сайты и приложения на Vue: Zoom, GitLab, Wizzair.

Преимущества:

- Простота изучения. Vue.js обладает простым и интуитивно понятным синтаксисом, который делает его легким для изучения и использования даже для начинающих разработчиков. Он предоставляет прямой и понятный подход к созданию интерфейсов.
- Гибкость. Фреймворк предоставляет гибкие инструменты для организации кода и компонентов, позволяя разработчикам создавать приложения любой сложности. Vue.js не навязывает строгую архитектуру приложения, позволяя выбирать наиболее подходящие решения для конкретных задач.
- Крошечный размер. Vue.js весит около 20 КБ, сохраняя при этом свою скорость и гибкость, что позволяет достичь гораздо лучшей производительности по сравнению с другими платформами.
- Производительность. Vue.js обеспечивает высокую производительность благодаря эффективному виртуальному DOM и системе реактивности. Он обновляет только те части страницы, которые действительно изменились, что улучшает быстродействие приложений.
- Экосистема. Вокруг Vue.js сформировалась развитая экосистема, включающая в себя множество библиотек, инструментов и ресурсов для разработки, тестирования и управления состоянием приложений. Это делает разработку с использованием Vue.js более удобной и эффективной.
- Подробная документация. Vue.js имеет очень подробную документацию, которая может ускорить процесс обучения для разработчиков и сэкономить много времени на разработку приложения, используя только базовые знания HTML и JavaScript.

Недостатки:

- Меньшее сообщество. В сравнении с другими популярными фреймворками, такими как React и Angular, у Vue.js может быть меньшее сообщество разработчиков и меньше ресурсов для обучения и поддержки. Это может усложнить поиск решений для определенных проблем и получение помощи в сообществе.
- Меньшая масштабируемость. Несмотря на то, что Vue.js предоставляет гибкие инструменты для создания приложений любой сложности, у него, может быть, ограничения в масштабируемости для крупных проектов по сравнению с другими фреймворками. Это может быть связано с отсутствием стандартных решений для управления состоянием или маршрутизацией в больших приложениях.

Резюмируя, Vue имеет низкий порог вхождения. В документации говорится, что разработчику нужен вечер, чтобы разобраться с основами фреймворка. Но чтобы писать сложные приложения, придётся потратить время и детальнее изучить Vue.

У React тоже невысокий порог вхождения. Чтобы приступить к работе, достаточно сделать импорт библиотеки и написать несколько строк кода. Но для создания сложных приложений придётся хорошенько изучить React, в том числе освоить синтаксис JSX.

Angular – более комплексный в изучении фреймворк. Чтобы начать работу, придётся изучить множество концепций и TypeScript. Но, с другой стороны, Angular предлагает всё необходимое для работы «из коробки», поэтому разработчику не придётся «изобретать велосипед», создавая очередной API-модуль.

Что касается выбора языка программирования для бэкенда, то он зависит от множества факторов: опыт, личные предпочтения, требования проекта, экосистема и используемые инструменты. К тому же, языков на которых можно написать бэкенд для сайта очень много, а у этих языков еще и множество фреймворков, поэтому рассмотрим их вкратце.

1. JavaScript (Node.js). JavaScript с использованием среды выполнения Node.js является одним из наиболее распространенных выборов для разработки бэкенда. Node.js обеспечивает высокую производительность и масштабируемость, а также широкий выбор библиотек и фреймворков для разработки.

Из преимуществ: высокая производительность и масштабируемость благодаря асинхронной природе и эффективному механизму обработки событий в Node.js; большое сообщество разработчиков и обширный выбор библиотек и фреймворков для разработки бэкенда, таких как Express.js, Nest.js и Koa.js.; широкое использование JavaScript как языка на фронтенде и бэкенде, что упрощает совместную разработку и поддержку кода.

Недостатки: возможные проблемы с блокировкой потоков в некоторых случаях; не такая хорошая поддержка многопоточности, как в некоторых других языках; для крупных проектов не подойдет.

2. Python – это популярный язык программирования с широким спектром инструментов и фреймворков для создания веб-приложений. Django и Flask являются двумя наиболее популярными фреймворками для разработки веб-приложений на Python.

Преимущества: простой и читаемый синтаксис, который делает Python очень популярным среди начинающих и опытных разработчиков; большое количество готовых библиотек и фреймворков для решения различных задач, таких как Django, Flask, FastAPI и aiohttp; широкое использование Python в области науки о данных и машинного обучения, что делает его привлекательным для разработки приложений с искусственным интеллектом и аналитикой данных.

Недостатки: не самая высокая производительность по сравнению с некоторыми другими языками, такими как C++ или Go; ограниченная поддержка асинхронных операций в стандартной библиотеке Python, хотя есть альтернативные библиотеки и фреймворки для асинхронного программирования.

3. Java. Java – это надежный и мощный язык программирования, который широко используется для создания корпоративных веб-приложений. Среди популярных фреймворков для веб-разработки на Java можно выделить Spring и Hibernate.

Преимущества: высокая производительность и стабильность; широко используется для корпоративных веб-приложений; богатая экосистема инструментов и фреймворков, таких как Spring и Hibernate; хорошая поддержка многопоточности.

Недостатки: больше кода для достижения тех же результатов по сравнению с некоторыми другими языками; большой объем памяти, необходимый для выполнения Java-приложений.

4. PHP – это язык программирования, который широко используется для создания динамических веб-сайтов и приложений. Фреймворки, такие как Laravel и Symfony, обеспечивают структурированную и эффективную разработку веб-приложений на PHP.

Преимущества: широко используется для создания динамических веб-сайтов и приложений; множество фреймворков, таких как Laravel и Symfony, обеспечивают быструю и эффективную разработку; большое сообщество разработчиков и обширная документация.

Недостатки: возможны проблемы с производительностью и масштабируемостью в больших проектах; некоторые проблемы с безопасностью из-за устаревших или небезопасных функций.

5. Ruby. Ruby часто используется совместно с фреймворком Ruby on Rails для быстрой и эффективной разработки веб-приложений. Ruby on Rails предоставляет множество готовых решений и сокращает время разработки.

Преимущества: простой и удобный синтаксис, что делает разработку быстрой и приятной; фреймворк Ruby on Rails ускоряет процесс разработки и предоставляет множество готовых решений; множество гемов (библиотек) для реализации различных функциональностей.

Недостатки: не так высокая производительность по сравнению с некоторыми другими языками; меньшая стабильность фреймворка Ruby on Rails по сравнению с некоторыми альтернативами.

6. Go, также известный как Golang, является открытым исходным кодом, компилируемым, статически типизированным языком программирования, разработанным внутри Google для написания бэкенда своих сервисов.

Преимущества: очень высокая производительность и эффективное использование ресурсов благодаря компиляции в машинный код; простой и понятный синтаксис, что делает Go привлекательным для новичков и опытных разработчиков; мощная стандартная библиотека и обширный набор инструментов для разработки сетевых приложений и микросервисов.

Недостатки: относительно молодой язык, поэтому меньше готовых библиотек и фреймворков по сравнению с другими языками; меньшее количество разработчиков, чем у более установленных языков, таких как Java или Python.

7. C++ – это язык программирования, который широко используется для разработки высокопроизводительных приложений. Может использоваться в качестве языка для написания бэкенда для веб-сайтов со сложной логикой, чтоб оптимизировать вычисления. Фреймворки для бэкенда веб-сайтов: Drogon, WebToolkit.

Преимущества: очень высокая производительность и полный контроль над ресурсами памяти и процессорным временем; широкое применение в различных областях, таких как разработка игр, системное программирование, встраиваемые системы и т.д.; низкоуровневый доступ к аппаратному обеспечению и возможность оптимизации кода для конкретных платформ.

Недостатки: сложный и объемный синтаксис, который может сделать разработку более трудоемкой и подверженной ошибкам; нет встроенной поддержки сборки мусора, что требует внимательного управления памятью и может привести к утечкам памяти и ошибкам в работе программы; низкий

уровень абстракции может увеличить время разработки по сравнению с более высокоуровневыми языками, такими как Python или Ruby.

Для разработки веб-сайтов используются различные типы баз данных в зависимости от потребностей проекта. Выбор базы данных зависит от требований к проекту, типа данных, масштабируемости, производительности и других факторов.

Реляционные базы данных (SQL). Классический тип баз данных, который использует язык структурированных запросов (SQL) для управления данными. Примеры включают MySQL, PostgreSQL, SQLite, Microsoft SQL Server и Oracle. Реляционные базы данных хорошо подходят для проектов, требующих жесткой структуры данных и обеспечивающих соответствие ACID (атомарность, согласованность, изоляция, долговечность).

Нереляционные базы данных (NoSQL). Этот тип баз данных используется для работы с неструктурированными или полуструктурированными данными. Примеры включают MongoDB, Cassandra, Redis и Couchbase. NoSQL базы данных хорошо подходят для проектов, где требуется масштабируемость, гибкость схемы данных и высокая производительность при обработке больших объемов данных.

Графовые базы данных. Этот тип баз данных специализируется на хранении и обработке данных в виде графов. Примеры включают Neo4j, Amazon Neptune и Microsoft Azure Cosmos DB. Графовые базы данных подходят для проектов, где важны связи и отношения между данными.

Временные базы данных. Этот тип баз данных специализируется на хранении временных данных или данных с временным характером. Примеры включают InfluxDB и TimescaleDB. Временные базы данных используются в проектах, где требуется анализ временных рядов, мониторинг, журналирование и другие временно-ориентированные операции.

Среды разработки для веб-сайтов (IDE) представляют собой интегрированные наборы инструментов, которые облегчают создание,

отладку и тестирование веб-приложений. Вот несколько популярных сред разработки для веб-сайтов:

1. Visual Studio Code (VS Code). Это легкий и мощный редактор кода, разработанный Microsoft. VS Code поддерживает множество языков программирования и фреймворков, а также имеет обширную систему расширений для настройки под конкретные нужды разработчика.

2. Atom – это другой распространенный редактор кода, созданный GitHub. Он также имеет обширную библиотеку плагинов и тем для настройки внешнего вида и функциональности.

3. Sublime Text – это быстрый и гибкий редактор кода, который предлагает множество функций, таких как автоматическое завершение кода, множественный курсор и пакетный контроль.

4. WebStorm – это полнофункциональная интегрированная среда разработки (IDE), специализирующаяся на разработке веб-приложений. Он предоставляет широкий спектр инструментов для работы с HTML, CSS, JavaScript и другими веб-технологиями.

5. Brackets – это редактор кода, разработанный Adobe, который специализируется на веб-разработке. Он обеспечивает интеграцию с препроцессорами CSS, автоматическое обновление браузера и другие полезные функции.

6. Eclipse – это мощная IDE, изначально созданная для разработки на Java, но которая также поддерживает веб-разработку с помощью плагинов, таких как Eclipse Web Tools Platform (WTP) и Eclipse JavaScript Development Tools (JSDT).

2.3 Описание выбранной среды разработки

Для написания клиентской части я выбрала библиотеку React по нескольким причинам. Во-первых, я знаю синтаксис HTML, CSS и JavaScript, поэтому синтаксис JSX не будет для меня сложным. Во-вторых, низкий порог входа в отличие от фреймворков, где есть четкие правила и указания как нужно

писать код, в React нет четких предписаний, поэтому у разработчика есть полная свобода действий, не нужно досконально изучать все инструменты. В-третьих, привлекает большое сообщество, в случае возникновения вопросов будет большая вероятность найти ответ.

Как работает React:

React использует декларативный подход в программировании. Если в императивном программировании необходимо задать каждое действие, то в декларативном достаточно описания конечного результата и реакции на действия. Библиотека позволяет оптимизировать процесс разработки за счет того, что программисту не нужно описывать подробности. React.js самостоятельно будет обновлять элементы, ориентируясь на условия. Задача программиста — грамотно описать их. Использование этого подхода позволяет создавать приложения на React максимально быстро.

Основным элементом в данной библиотеке являются компоненты, представляющие собой отдельные составные части страницы. Каждый компонент в React является инкапсулированным, содержащим в себе все необходимые данные и методы для своей работы.

Использование инкапсуляции позволяет хранить состояние элементов внутри самих них, делая их изолированными и автономными. Такой подход React обеспечивает возможность повторного использования элементов. Один и тот же элемент может быть использован на других страницах или в других разделах сайта. Преимущества этого подхода очевидны: разработчику не нужно заново создавать код для элемента, а отсутствие сложных зависимостей между ними упрощает процесс отладки.

Рендер страницы.

Каждая веб-страница имеет свою уникальную структуру, известную как DOM (Document Object Model) или объектная модель документа. Она состоит из HTML-тегов, CSS для стилей и связанного с ним JavaScript-кода.

Библиотека React обладает специальным механизмом управления отображением всех компонентов на странице, то есть созданием визуального

представления элементов. Для этого React использует кэширование копии DOM-дерева. Эта копия изменяется быстрее оригинала, что позволяет приложению обновлять страницу так же быстро. Например, если страница должна быть обновлена после определенного действия, React работает не с реальной структурой DOM, а с копией, что позволяет пользователю быстрее увидеть результат. Благодаря этому в приложениях, созданных на React, изменения отображаются почти мгновенно, без необходимости долгого ожидания.

Этот процесс основан на частичном обновлении DOM. В памяти хранятся две версии DOM: старая и новая. При необходимости React сравнивает их и обновляет только ту часть, в которой произошли изменения, вместо полной перерисовки DOM. Это позволяет оптимизировать процесс загрузки страницы.

Хуки.

Ранее в React для управления состоянием компонентов использовались классы, в которых определялись методы для управления состоянием. В настоящее время вместо этого широко используются хуки. Хуки — это функции, которые могут быть привязаны к состоянию компонента или его методам, подобно крючкам.

Использование хуков позволяет избежать необходимости создания классовых компонентов. Вместо этого функции выполняются автоматически при изменении состояния.

В React предоставляются как предварительно созданные хуки, так и возможность создания собственных хуков разработчиком.

JSX.

React внедрил собственное расширение для JavaScript называемое JSX. Оно позволяет описывать элементы HTML в виде кода на JavaScript. Это дает возможность разработчикам изменять структуру DOM с помощью знакомого им JavaScript.

На практике это означает, что программист может написать компонент, используя синтаксис HTML, который затем преобразуется в объект JavaScript с соответствующими свойствами и методами.

Для бэкенда я выбрала Node.js и фреймворк Express.js, так как мой веб-сайт не крупный, а из пользователей, которые могут манипулировать данными, только администратор.

Node.js – это среда выполнения JavaScript на стороне сервера, которая позволяет разработчикам создавать масштабируемые и эффективные серверные приложения. Он основан на событийно-ориентированной архитектуре и работает на движке V8 JavaScript от Google.

Express.js же это минималистичный и гибкий веб-фреймворк для Node.js, который предоставляет множество удобных функций и инструментов для разработки веб-приложений и API. Express упрощает создание маршрутов, обработку HTTP-запросов, управление сессиями, обработку ошибок и многое другое.

Вместе Node.js и Express.js образуют стек технологий для создания серверных приложений. Node.js обеспечивает среду выполнения для JavaScript, а Express.js предоставляет удобный и эффективный способ организации приложения и работы с HTTP-запросами и ответами.

В качестве места для хранения всех данных я выбрала базу данных PostgreSQL, так как мне важна безопасность и стабильность, а также она хорошо работает в связке с Node.js, потому что это довольно популярное решение.

PostgreSQL (или просто Postgres) — это объектно-реляционная система управления базами данных (СУБД), которая широко используется в различных типах приложений, включая веб-сайты.

PostgreSQL является проектом с открытым исходным кодом, лицензированным по лицензии PostgreSQL, которая позволяет использовать, изменять и распространять его бесплатно. Это обеспечивает свободный

доступ к коду и поощряет активное сообщество разработчиков, способствуя развитию и улучшению СУБД.

Postgres обладает широким набором функций и возможностей, включая поддержку сложных типов данных полнотекстовый поиск, географические запросы, оконные функции, а также расширения и пользовательские функции, что делает его гибким и мощным инструментом для различных видов приложений.

Обладает хорошей производительностью и масштабируемостью, позволяя эффективно обрабатывать как небольшие, так и крупные объемы данных. Также обеспечивает высокий уровень безопасности и целостности данных. Он поддерживает различные механизмы аутентификации и авторизации, включая ролевую модель доступа, SSL-шифрование, разделение прав доступа и т. д. Кроме того, PostgreSQL имеет встроенные механизмы для обеспечения целостности данных, такие как ограничения целостности, транзакции и механизмы восстановления после сбоев.

PostgreSQL доступен на различных платформах, включая Linux, Windows, macOS и другие.

Таким образом, образовался стек технологий PERN – PostgreSQL, Express.js, React, Node.js. У этого стека есть название, так как это довольно популярный стек и многие используют именно эти инструменты для создания сайта.

Но так как JavaScript динамический язык программирования, а PostgreSQL хранит в себе данные строго типизированные, то нужна ORM, чтобы преобразовывать несовместимые типы моделей в ООП между хранилищем данных и объектами программирования. Выбор пал на Sequalize. Это инструмент для организации взаимодействия между платформой Node.js и реляционными базами данных без использования специального языка запросов SQL. Он связывает базу данных PostgresSQL с объектами JavaScript, создавая виртуальную объектную базу данных.

Теперь имея стек технологий, можно представить архитектуру веб-сайта (рисунок 2.1).

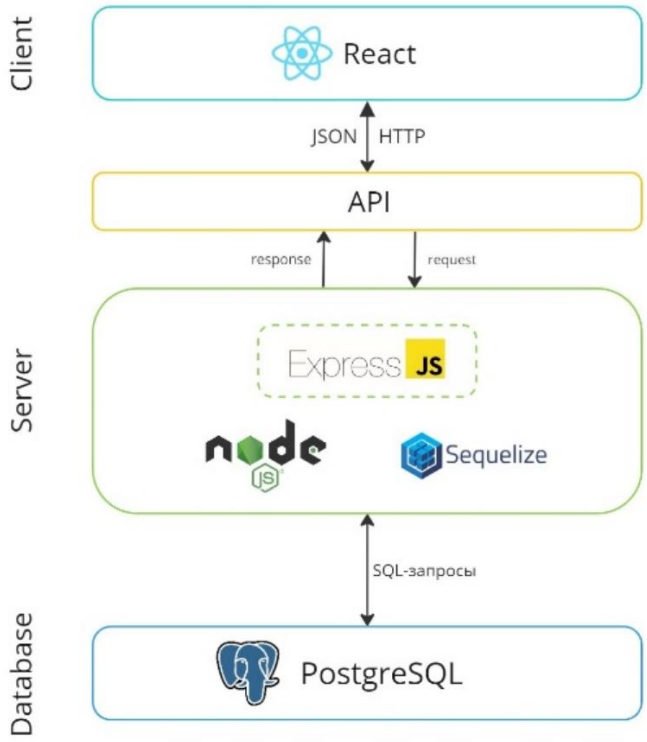

Рисунок 2.1 – Архитектура сайта

Клиент общается с сервером с помощью API. Клиент посылает HTTP запросы, а API определяет, по каким эндпоинтам можно обращаться клиенту к серверу. Сервер обрабатывает запрос от клиента, обращается к базе данных. Сервер и база данных обмениваются данными с помощью SQL-запросов. После манипуляции с данными клиент получает обратно ответ от сервера.

В качестве среды разработки выбрана Visual Studio Code, так как эта среда разработки бесплатная, популярная и предоставляет огромное количество расширений и настроек для разработчика, имеет понятный привлекательный интерфейс (рисунок 2.2).

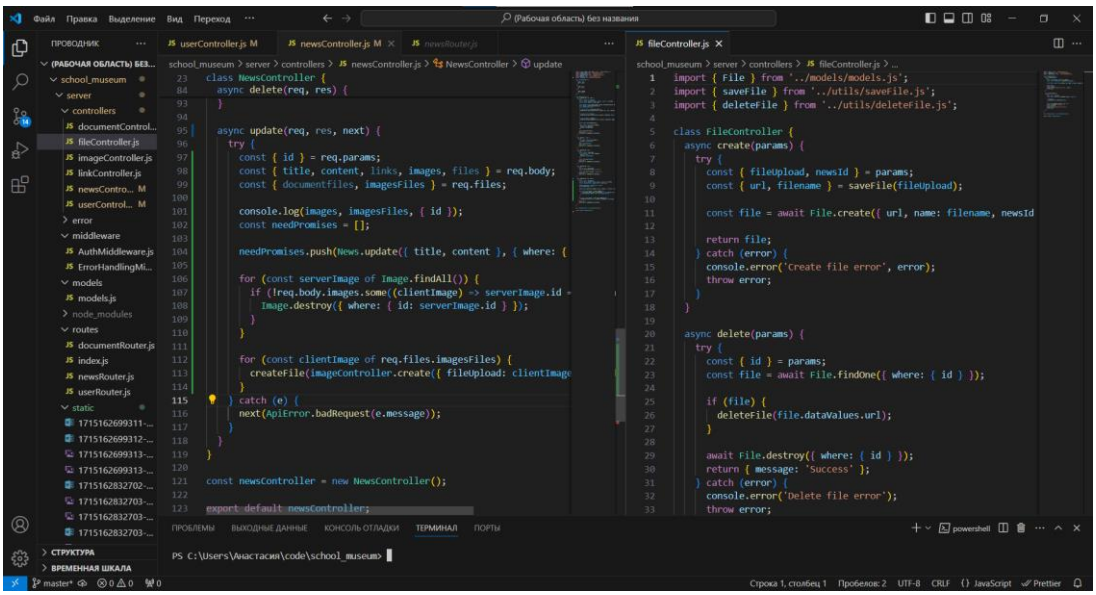

Рисунок 2.2 – Интерфейс среды разработки VS Code

VS Code предоставляет поддержку для множества языков программирования, включая такие языки как: JavaScript, HTML, CSS, PHP, Go, Ruby, Python, C#, TypeScript. Это делает его универсальным инструментом для разработки различных типов приложений.

В VS Code доступны множество функций редактора кода, включая подсветку синтаксиса, автодополнение, быстрые фиксы, отладчик, рефакторинг, интеграцию с Git и многое другое. Эти функции помогают улучшить производительность и эффективность работы разработчиков.

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

VS Code поддерживается на различных операционных системах, включая Windows, macOS и Linux, что делает его доступным для использования на большинстве популярных компьютеров.

Данная среда разработки имеет большое и активное сообщество пользователей и разработчиков, которые обмениваются опытом, создают расширения, предлагают улучшения и обеспечивают поддержку новичков.

Для отслеживания изменений в своем коде, создания различных версий файлов и возвращения к предыдущим состояниям проекта при необходимости будет использоваться Git.

Git – это система контроля версий, которая используется для отслеживания изменений в исходном коде и координации работы над проектами программного обеспечения.

Git сохраняет историю всех изменений в проекте, включая кто, когда и что изменил. Это позволяет легко отслеживать прогресс разработки, а также возвращаться к предыдущим версиям при необходимости. Также он дает возможность хранить репозиторий как локально на компьютере разработчика, так и удаленно на сервере. Это обеспечивает резервное копирование данных и обмен изменениями между разными участниками проекта, позволяя нескольким разработчикам работать над одним и тем же проектом, а затем сливать свои изменения в единую версию. Это синхронизирует изменения и помогает избежать конфликтов при совместной работе. Git поддерживает концепцию ветвления, которая позволяет разработчикам создавать отдельные ветки кода для работы над конкретными функциями или исправлениями ошибок, не затрагивая основную ветку проекта. Затем эти изменения могут быть объединены обратно в основную ветку.

2.4 Проектирование схемы

Для дальнейшей работы была спроектирована ER-диаграмма базы данных для веб-сайта школьного музея (рисунок 2.3).

Сущность «user» хранит в себе уникальный идентификатор(«id»), атрибуты «email» для хранения электронной почты пользователя и «password» для хранения пароля пользователя.

Сущность «document» представляет собой документы, которые администратор сайта будет загружать на сайт. В ней хранятся уникальный идентификатор(«id»), ссылка на документ(«url»), т.к. загружаемые документы будут храниться локально в папке, чтобы не перегружать базу данных, имя

файла(«name») для корректного отображения на сайте и внешний ключ «user_id» для связи с таблицей «user» один ко многим.

В сущности «news» будут храниться новости, которые загружает администратор сайта. Имеется уникальный идентификатор(«id»), а также внешний ключ, который связывает сущность «news» с «user» связью один ко многим. Из остальных атрибутов: заголовок новости(«title»), дата публикации(«date_publication»), текст новости(«content»). У новости также могут быть ссылки, картинки и файлы. Они представляют собой отдельные сущности, которые связаны с сущностью «news» с помощью внешних ключей связью один ко многим.

Сущность «image» — это фото новости. Из атрибутов уникальный идентификатор(«id»), ссылка на фото в локальной папке(«url»), внешний ключ(«news_id»).

Сущность «links» - ссылки новости. Из атрибутов уникальный идентификатор(«id»), сама ссылка(«url») и внешний ключ(«news_id»).

Сущность «files» - файлы новости. Из атрибутов уникальный идентификатор(«id»), ссылка на файл в локальной папке(«url»), имя файла для корректного отображения на сайте(«name»), внешний ключ(«news_id»).

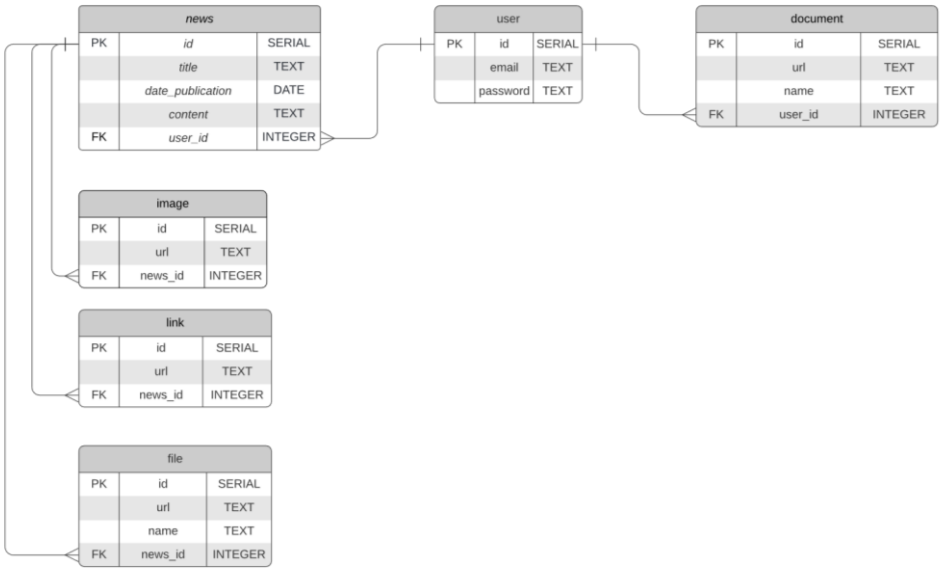

Рисунок 2.3 – ER-диаграмма школьного музея

Также для удобства был создан прототип сайта в Figma(рисунок 2.4).

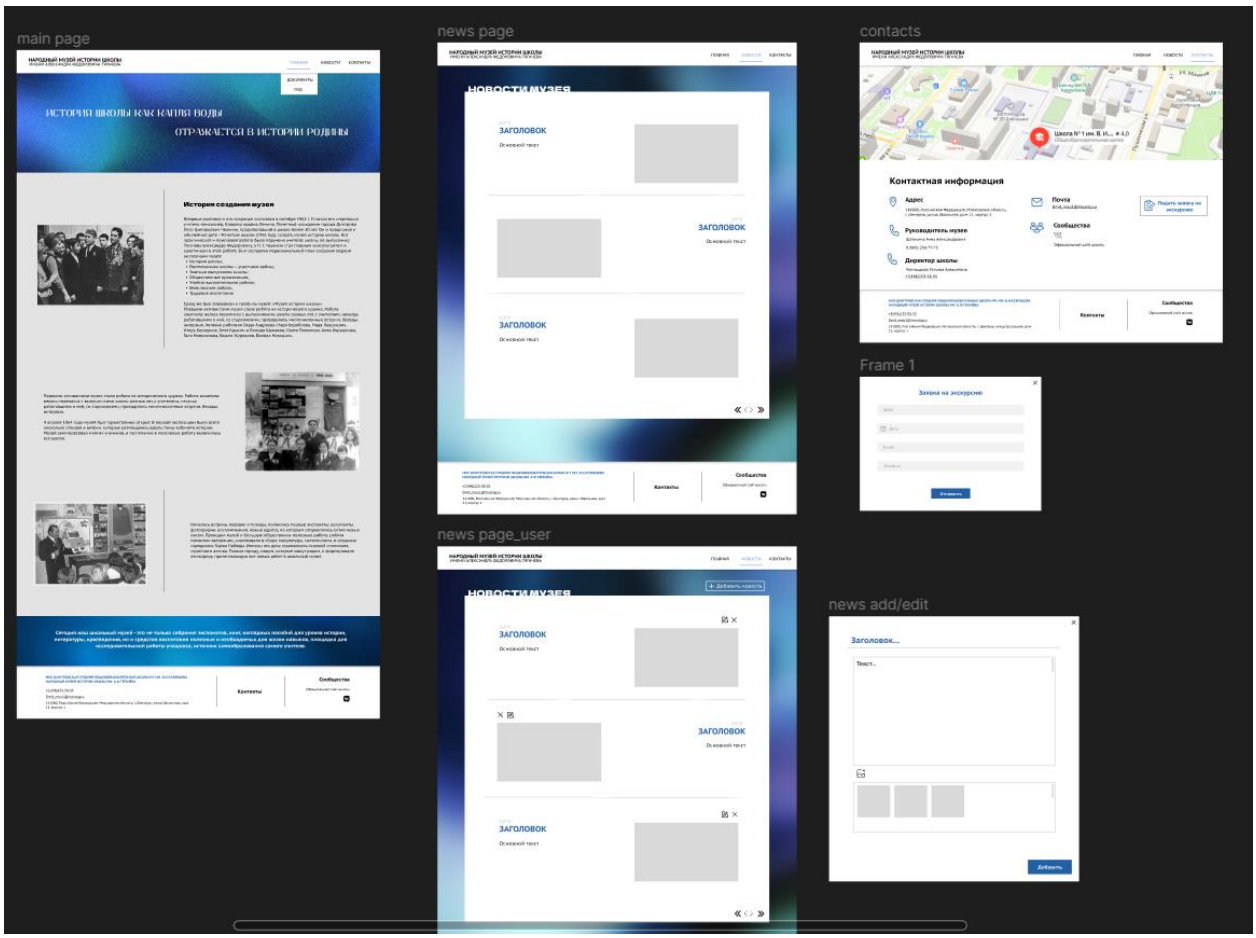

Рисунок 2.4 – Прототип сайта

ГЛАВА 3. РАЗРАБОТКА И ТЕСТИРОВАНИЕ ВЕБ-САЙТА

3.1 Установка и настройка необходимого программного обеспечения

JavaScript— мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript (стандарт ECMA-262[2]). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам[8].

Интерне́т[1] (англ. Internet, от англ. Internet protocol — досл. «межсетевой протокол», по первому протоколу передачи данных, который объединил отдельные компьютерные сети во всемирную сеть, «сеть сетей») — коммуникационная сеть и всемирная система объединённых компьютерных сетей для хранения и передачи информации[2].

Раньше упоминался как Всемирная сеть и Глобальная сеть, а также просто Сеть. Построена на базе стека протоколов TCP/IP[3]. На основе Интернета работает Всемирная паутина (World Wide Web, WWW) и множество других систем передачи данных. К началу 2023 года число пользователей достигло 5,16 млрд человек, что составляет более 50 % от всех жителей планеты Земля. Во многом это было обусловлено широким распространением сотовых сетей с доступом в Интернет стандартов 3G, 4G и 5G, развитием социальных сетей и удешевлением стоимости интернет-трафика[4][5].

Суперуспешное развитие Интернета во многом объясняется также тем, что во второй половине 2010-х годов мировая Сеть фактически стала полномасштабной заменой всем классическим инструментам получения информации, связи и коммуникации. Все «классические» СМИ — телевидение, радио и печатные издания — имеют полноценные онлайн-версии, кроме того, существует безграничное множество интернет-СМИ и

блог-платформ, соединяющих все признаки различных форм коммуникации, делая контент более «живым» и менее зависимым от штампов. На сегодняшний день самыми популярными интернет-ресурсами являются социальные сети (Facebook, Instagram, Twitter), мессенджеры (WhatsApp, Viber, Telegram), а также энциклопедия Википедия и видеохостинг YouTube, который для многих пользователей стал «новым телевидением», «заменой телевидению» и т. д.[6][7][8]

React-разработка заключается в описании того, что нужно вывести на страницу (а не в составлении инструкций для браузера, посвящённых тому, как это делать). Это, кроме прочего, означает значительное сокращение объёмов шаблонного кода.

В составе Angular, с другой стороны, есть средства командной строки, которые генерируют шаблонный код компонентов. Не кажется ли это немного не тем, чего можно ждать от современных инструментов разработки интерфейсов? Фактически, речь идёт о том, что в Angular так много шаблонного кода, что для того, чтобы его генерировать, даже создано специальное средство.

В React, приступая к разработке, просто начинают писать код. Тут нет шаблонного кода компонентов, который нужно как-то генерировать. Конечно, перед разработкой нужна некоторая подготовка, но, когда дело доходит до компонентов, их можно описывать в виде чистых функций.

3.2 Программная реализация модуля и логики веб-сайта

Перед Бренданом Эйхом, нанятым в компанию Netscape 4 апреля 1995 года[16], была поставлена задача внедрить язык программирования Scheme или что-то похожее в браузер Netscape. Поскольку требования были размыты, Эйха перевели в группу, ответственную за серверные продукты, где он проработал месяц, занимаясь улучшением протокола HTTP[16]. В мае разработчик был переброшен обратно, в команду, занимающуюся клиентской частью (браузером), где он немедленно начал разрабатывать концепцию нового языка программирования. Менеджмент разработки браузера, включая

Тома Пакина (англ. Tom Paquin), Михаэля Тоя[англ.], Рика Шелла (англ. Rick Schell), был убеждён, что Netscape должен поддерживать язык программирования, встраиваемый в HTML-код страницы[17].

Марк Андрессен

Билл Джой

Помимо Брендана Эйха, в разработке участвовали[16] сооснователь[18] Netscape Communications Марк Андрессен и сооснователь Sun Microsystems Билл Джой: чтобы успеть закончить работы над языком к релизу браузера, компании заключили соглашение о сотрудничестве в разработке[19]. Они ставили перед собой цель обеспечить «язык для склеивания» составляющих частей веб-ресурса: изображений, плагинов, Java-апплетов, который был бы удобен для веб-дизайнеров и программистов, не обладающих высокой квалификацией[16].

Первоначально по предложению Марка Андрессена[20] язык был назван Mocha[21][22][23], был реализован Бренданом Эйхом в течение десяти дней и впервые был включен в пре-альфу версию Netscape 2[20]. Затем он был переименован в LiveScript[23][24] и предназначался как для программирования на стороне клиента, так и для программирования на стороне сервера (там он должен был называться LiveWire)[19]. На синтаксис оказали влияние языки Си и Java, и, поскольку Java в то время было модным словом[16][19], 4 декабря 1995 года LiveScript переименовали в JavaScript[25], получив соответствующую лицензию у Sun. Анонс JavaScript со стороны представителей Netscape и Sun состоялся накануне выпуска второй бета-версии Netscape Navigator[16]. В нём декларируется, что 28 лидирующих ИТ-компаний выразили намерение использовать в своих будущих продуктах JavaScript как объектный скриптовый язык с открытым стандартом[26].

В 1996 году компания Microsoft выпустила аналог языка JavaScript, названный JScript. Анонсирован этот язык был 18 июля 1996 года[27]. Первым браузером, поддерживающим эту реализацию, был Internet Explorer 3.0.

По инициативе компании Netscape[28][29] была проведена стандартизация языка ассоциацией ECMA. Стандартизированная версия имеет название ECMAScript, описывается стандартом ECMA-262. Первой версии спецификации соответствовал JavaScript версии 1.1, а также языки JScript и ScriptEasy[11][19].

React (иногда React.js или ReactJS) — JavaScript-библиотека[4] с открытым исходным кодом для разработки пользовательских интерфейсов.

React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций[5][6][7].

React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость разработки, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL.

TCP/IP — сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю. В модели предполагается прохождение информации через четыре уровня, каждый из которых описывается правилом (протоколом передачи). Наборы правил, решающих задачу по передаче данных, составляют стек протоколов передачи данных, на которых базируется Интернет[1][2]. Название TCP/IP происходит из двух важнейших протоколов семейства — Transmission Control Protocol (TCP) и Internet Protocol (IP), которые были первыми разработаны и описаны в данном стандарте. Также изредка упоминается как модель DOD (Department of Defense)[3] в связи с

историческим происхождением от сети ARPANET из 1970-х годов (под управлением DARPA, Министерства обороны США[4]).

Набор интернет-протоколов обеспечивает сквозную передачу данных, определяющую, как данные должны пакетироваться, обрабатываться, передаваться, маршрутизироваться и приниматься. Эта функциональность организована в четыре слоя абстракции, которые классифицируют все связанные протоколы в соответствии с объёмом задействованных сетей. От самого низкого до самого высокого уровня — это уровень связи, содержащий методы связи для данных, которые остаются в пределах одного сегмента сети; интернет-уровень, обеспечивающий межсетевое взаимодействие между независимыми сетями; транспортный уровень, обрабатывающий связь между хостами; и прикладной уровень, который обеспечивает обмен данными между процессами для приложений.

Прикладной уровень

На прикладном уровне (Application layer) работает большинство сетевых приложений.

Эти программы имеют свои собственные протоколы обмена информацией, например, Интернет-браузер для протокола HTTP, ftp-клиент для протокола FTP (передача файлов), почтовая программа для протокола SMTP (электронная почта), SSH (безопасное соединение с удалённой машиной), DNS (преобразование символьных имён в IP-адреса) и многие другие.

В массе своей эти протоколы работают поверх TCP или UDP и привязаны к определённому порту, например:

HTTP на TCP-порт 80 или 8080,
FTP на TCP-порт 20 (для передачи данных) и 21 (для управляющих команд),
SSH на TCP-порт 22,
запросы DNS на порт UDP (реже TCP) 53,

обновление маршрутов по протоколу RIP на UDP-порт 520.
Эти порты определены Агентством по выделению имён и уникальных параметров протоколов (IANA).

К этому уровню относятся: Echo, Finger, Gopher, HTTP, HTTPS, IMAP, IMAPS, IRC, NNTP, NTP, POP3, POPS, QOTD, RTSP, SNMP, SSH, Telnet, XDMCP.

Транспортный уровень
Протоколы транспортного уровня (Transport layer) могут решать проблему негарантированной доставки сообщений («дошло ли сообщение до адресата?»), а также гарантировать правильную последовательность прихода данных. В стеке TCP/IP транспортные протоколы определяют, для какого именно приложения предназначены эти данные.

Протоколы автоматической маршрутизации, логически представленные на этом уровне (поскольку работают поверх IP), на самом деле являются частью протоколов сетевого уровня; например OSPF (IP-идентификатор 89).

TCP (IP-идентификатор 6) — «гарантированный» транспортный механизм с предварительным установлением соединения, предоставляющий приложению надёжный поток данных, дающий уверенность в безошибочности получаемых данных, перезапрашивающий данные в случае потери и устраняющий дублирование данных. TCP позволяет регулировать нагрузку на сеть, а также уменьшать время ожидания данных при передаче на большие расстояния. Более того, TCP гарантирует, что полученные данные были отправлены точно в такой же последовательности. В этом его главное отличие от UDP.

UDP (IP-идентификатор 17) протокол передачи датаграмм без установления соединения. Также его называют протоколом «ненадёжной» передачи, в смысле невозможности удостовериться в доставке сообщения адресату, а

также возможного перемешивания пакетов. В приложениях, требующих гарантированной передачи данных, используется протокол TCP.

UDP обычно используется в таких приложениях, как потоковое видео и компьютерные игры, где допускается потеря пакетов, а повторный запрос затруднён или не оправдан, либо в приложениях вида запрос-ответ (например, запросы к DNS), где создание соединения занимает больше ресурсов, чем повторная отправка.

И TCP, и UDP используют для определения протокола верхнего уровня число, называемое портом.

См. также: Список портов TCP и UDP

Сетевой (межсетевой) уровень

Межсетевой уровень (Network layer) изначально разработан для передачи данных из одной сети в другую. На этом уровне работают маршрутизаторы, которые перенаправляют пакеты в нужную сеть путём расчёта адреса сети по маске сети. Примерами такого протокола является X.25 и IPC в сети ARPANET.

С развитием концепции глобальной сети в уровень были внесены дополнительные возможности по передаче из любой сети в любую сеть, независимо от протоколов нижнего уровня, а также возможность запрашивать данные от удалённой стороны, например в протоколе ICMP (используется для передачи диагностической информации IP-соединения) и IGMP (используется для управления multicast-потоками).

ICMP и IGMP расположены над IP и должны попасть на следующий — транспортный — уровень, но функционально являются протоколами сетевого уровня, и поэтому их невозможно вписать в модель OSI.

Пакеты сетевого протокола IP могут содержать код, указывающий, какой именно протокол следующего уровня нужно использовать, чтобы извлечь данные из пакета. Это число — уникальный IP-номер протокола. ICMP и IGMP имеют номера, соответственно, 1 и 2.

К этому уровню относятся: DVMRP, ICMP, IGMP, MARS, PIM, RIP, RIP2, RSVP

Канальный уровень

Основная статья: Канальный уровень (TCP/IP)

Канальный уровень (англ. Link layer) описывает способ кодирования данных для передачи пакета данных на физическом уровне (то есть специальные последовательности бит, определяющих начало и конец пакета данных, а также обеспечивающие помехоустойчивость). Ethernet, например, в полях заголовка пакета содержит указание того, какой машине или машинам в сети предназначен этот пакет.

Примеры протоколов канального уровня — Ethernet, IEEE 802.11 WLAN, SLIP, Token Ring, ATM и MPLS.

PPP не совсем вписывается в такое определение, поэтому обычно описывается в виде пары протоколов HDLC/SDLC.

MPLS занимает промежуточное положение между канальным и сетевым уровнем и, строго говоря, его нельзя отнести ни к одному из них.

Канальный уровень иногда разделяют на 2 подуровня — LLC и MAC.

Кроме того, канальный уровень описывает среду передачи данных (будь то коаксиальный кабель, витая пара, оптическое волокно или радиоканал), физические характеристики такой среды и принцип передачи данных (разделение каналов, модуляцию, амплитуду сигналов, частоту сигналов, способ синхронизации передачи, время ожидания ответа и максимальное расстояние).

При проектировании стека протоколов на канальном уровне рассматривают помехоустойчивое кодирование — позволяющие обнаруживать и исправлять ошибки в данных вследствие воздействия шумов и помех на канал связи.
Развитием архитектуры Интернета и протоколов в модели TCP/IP занимается открытое международное сообщество проектировщиков IETF.
Git — это набор консольных утилит, которые отслеживают и фиксируют изменения в файлах (чаще всего речь идет об исходном коде программ, но вы можете использовать его для любых файлов на ваш вкус). Изначально Git был создан Линусом Торвальдсом при разработке ядра Linux. Однако инструмент так понравился разработчикам, что в последствии, он получил широкое распространение и его стали использовать в других проектах. С его помощью вы можете сравнивать, анализировать, редактировать, сливать изменения и возвращаться назад к последнему сохранению. Этот процесс называется контролем версий.

Для чего он нужен? Ну во-первых, чтобы отследить изменения, произошедшие с проектом, со временем. Проще говоря, мы можем посмотреть как менялись файлы программы, на всех этапах разработки и при необходимости вернуться назад и что-то отредактировать. Часто бывают ситуации, когда, во вполне себе работающий код, вам нужно внести определенные правки или улучшить какой-то функционал, по желанию заказчика. Однако после внедрения нововведений, вы с ужасом понимаете, что все сломалось. У вас начинается судорожно дергаться глаз, а в воздухе повисает немой вопрос: “Что делать?”

Без системы контроля версий, вам надо было бы долго напряженно просматривать код, чтобы понять как было до того, как все перестало работать. С Гитом же, все что нужно сделать - это откатиться на коммит назад.

Во-вторых он чрезвычайно полезен при одновременной работе нескольких специалистов, над одним проектом. Без Гита случится коллапс, когда разработчики, скопировав весь код из главной папки и сделав с ним задуманное, попытаются одновременно вернуть весь код обратно.

Git является распределенным, то есть не зависит от одного центрального сервера, на котором хранятся файлы. Вместо этого он работает полностью локально, сохраняя данные в директориях на жестком диске, которые называются репозиторием. Тем не менее, вы можете хранить копию репозитория онлайн, это сильно облегчает работу над одним проектом для нескольких людей. Для этого используются сайты вроде github и bitbucket.

3.3 Тестирование и публикация веб-сайта

Виды сайтов

Информационные сайты

Информационные сайты предназначены для предоставления пользователям контента по определенной тематике. Они часто включают статьи, новости, руководства и другую полезную информацию. Основная цель таких сайтов – информировать и обучать пользователей.

Примеры:

Новостные порталы (CNN, BBC)

Энциклопедии и справочники (Wikipedia, Britannica)

Блоги и тематические ресурсы (Medium, Habrahabr)

Корпоративные сайты

Корпоративные сайты представляют компании или организации в интернете. Они содержат информацию о компании, ее продуктах и услугах, контактные данные и новости. Такие сайты часто включают разделы для инвесторов и пресс-службы.

Примеры:

Официальные сайты компаний (Apple, Google)

Сайты малых и средних предприятий

Интернет-магазины

Интернет-магазины предназначены для продажи товаров и услуг онлайн. Они включают каталоги продукции, функционал корзины и оформления заказа, а также системы оплаты и доставки.

Примеры:

Гиганты электронной коммерции (Amazon, Alibaba)

Специализированные магазины (Etsy, Shopify)

Социальные сети

Социальные сети предоставляют платформы для общения и взаимодействия между пользователями. Они включают профили пользователей, ленты новостей, группы по интересам и системы обмена сообщениями.

Примеры:

Популярные социальные сети (Facebook, Instagram, Twitter)

Профессиональные сети (LinkedIn)

Форумы и сообщества

Форумы и сообщества предназначены для обсуждения различных тем и обмена мнениями между пользователями. Они структурированы в виде категорий и тем, где пользователи могут создавать и комментировать посты.

Примеры:

Традиционные форумы (Reddit, Stack Overflow)

Сообщества по интересам (Quora, DeviantArt)

Портфолио

Сайты-портфолио предназначены для демонстрации работ и достижений отдельных лиц или организаций. Они часто используются творческими профессионалами, такими как дизайнеры, фотографы, писатели и артисты, для представления своих работ потенциальным клиентам или работодателям.

Примеры:

Портфолио художников (Behance, Dribbble)

Личные сайты и блоги (например, портфолио писателей)

Образовательные сайты

Образовательные сайты предоставляют пользователям доступ к учебным материалам и курсам. Они могут включать видеоуроки, статьи, тесты и форумы для обсуждения.

Примеры:

Платформы онлайн-обучения (Coursera, edX)

Сайты университетов и школ

Развлекательные сайты

Развлекательные сайты предлагают пользователям развлекательный контент, такой как видео, музыка, игры и статьи на различные темы.

Примеры:

Стриминговые сервисы (YouTube, Netflix)

Игровые сайты (Twitch, Miniclip)

Сайты услуг

Сайты услуг предоставляют платформы для бронирования и заказа различных услуг, таких как путешествия, жилье, доставка еды и так далее.

Примеры:

Туристические сайты (Booking.com, Airbnb)

Сайты доставки (Uber Eats, DoorDash)

Лендинги

Лендинги, или посадочные страницы, обычно создаются для конкретных маркетинговых кампаний. Они предназначены для конверсии посетителей в потенциальных клиентов через формы регистрации, подписки или покупки.

Примеры:

Промо-страницы продуктов и услуг

Страницы подписки на рассылки

Заключение

Разнообразие видов сайтов отражает широкий спектр задач и потребностей, которые они удовлетворяют. От информационных порталов до интернет-магазинов и социальных сетей – каждый тип сайта играет свою уникальную роль в цифровом мире, предоставляя пользователям необходимую информацию, услуги и возможности для взаимодействия.

React.js: Введение и Основные Концепции

React.js – это популярная библиотека JavaScript, созданная для построения пользовательских интерфейсов, особенно одностраничных приложений. Разработанный в Facebook, React изменил подход к созданию веб-приложений благодаря своей скорости, гибкости и простоте в использовании. В этом обзоре рассмотрим ключевые аспекты React.js, его основные концепции и как его использовать для разработки современных веб-приложений.

Основные Концепции React.js

Компоненты

React построен на компонентах. Компоненты – это независимые, переиспользуемые части интерфейса. Они могут быть простыми (например, кнопка) или сложными (например, форма регистрации). Компоненты могут быть функциональными или классовыми.

Виртуальный DOM

React использует виртуальный DOM для оптимизации производительности. Вместо прямых манипуляций с DOM, React создает его легковесную копию – виртуальный DOM. Когда состояние компонента меняется, React обновляет виртуальный DOM, а затем сравнивает его с реальным DOM (используя алгоритм диффинга) и обновляет только измененные элементы.

Состояние и свойства

Компоненты могут иметь состояние (state) и свойства (props).

Свойства (props) передаются компонентам и неизменны (read-only). Они позволяют передавать данные от родительского компонента к дочернему.

Состояние (state) – это данные, которые могут изменяться внутри компонента. Изменение состояния вызывает повторный рендер компонента.

Жизненный цикл компонентов

Классовые компоненты в React имеют методы жизненного цикла, которые вызываются на разных этапах их существования.

Монтирование: методы вызываются при добавлении компонента в DOM.

constructor()

componentDidMount()

Обновление: методы вызываются при обновлении состояния или свойств компонента.

shouldComponentUpdate()

componentDidUpdate()

Размонтирование: метод вызывается перед удалением компонента из DOM.

componentWillUnmount()

Управление состоянием

Для управления состоянием в больших приложениях часто используют библиотеки, такие как Redux или MobX. В последние годы популярность набирает React Context API и React Hooks для управления состоянием без необходимости в сторонних библиотеках.

React Hooks

Введенные в React 16.8, хуки позволяют использовать состояние и другие функции React в функциональных компонентах.

useState – для управления состоянием.

useEffect – для управления побочными эффектами.

Работа с формами

React делает работу с формами удобной и гибкой. Состояние формы можно контролировать с помощью состояния компонента.

Роутинг

Для управления навигацией в React-приложениях используется библиотека react-router-dom. Она позволяет легко организовать маршрутизацию в приложении.

React.js – это мощная и гибкая библиотека для создания пользовательских интерфейсов. Благодаря компонентному подходу, виртуальному DOM, управлению состоянием и множеству вспомогательных библиотек, React позволяет создавать масштабируемые и производительные веб-приложения. С правильным пониманием его основных концепций и особенностей, разработчики могут эффективно использовать React для создания современных и интерактивных веб-интерфейсов.

JavaScript и Node.js являются фундаментальными технологиями для веб-разработки. JavaScript – это язык программирования, который выполняется в браузере и позволяет создавать интерактивные веб-страницы. Node.js, с другой стороны, предоставляет среду выполнения JavaScript на сервере, что открывает новые возможности для создания серверных приложений.

JavaScript: Основные Концепции

Введение

JavaScript (JS) – это язык программирования, изначально созданный для выполнения в браузерах, что позволило создавать интерактивные и динамичные веб-страницы. Со временем JavaScript эволюционировал и стал одним из самых популярных языков для разработки как фронтенд, так и бекенд приложений.

Основные Возможности

Переменные и Типы Данных

В JavaScript существует несколько типов данных: примитивные (числа, строки, булевы значения, null, undefined) и сложные (объекты, массивы).

Функции

Функции – это блоки кода, которые выполняют определенные задачи. В JavaScript функции могут быть объявлены различными способами: через ключевое слово function, как функциональные выражения или стрелочные функции.

Объектно-Ориентированное Программирование (ООП)

JavaScript поддерживает ООП через прототипное наследование и классы.

Асинхронное Программирование

JavaScript поддерживает асинхронные операции через колбэки, промисы и async/await.

Node.js: Основные Концепции

Введение

Node.js – это среда выполнения JavaScript, построенная на движке V8 от Google Chrome. Node.js позволяет запускать JavaScript на сервере, что делает его мощным инструментом для разработки серверных приложений.

Основные Возможности

Модульная Система

Node.js использует CommonJS для организации кода в модули. Каждый файл в Node.js считается модулем.

HTTP Сервер

Node.js предоставляет встроенный модуль http, который позволяет создавать веб-серверы.

Асинхронные I/O Операции

Node.js отлично подходит для асинхронных операций ввода-вывода благодаря своей неблокирующей архитектуре.

NPM (Node Package Manager)

NPM – это пакетный менеджер для Node.js, который позволяет устанавливать и управлять зависимостями.

JavaScript и Node.js представляют собой мощные инструменты для веб-разработки. JavaScript позволяет создавать динамичные и интерактивные пользовательские интерфейсы, а Node.js расширяет возможности JavaScript на серверную сторону, позволяя создавать масштабируемые и эффективные серверные приложения. Эти технологии вместе образуют прочную основу для разработки современных веб-приложений.

JavaScript (JS) и Node.js – это две ключевые технологии в сфере веб-разработки. JS является одним из самых популярных языков программирования, широко используемым для создания интерактивных веб-страниц и веб-приложений. Node.js, с другой стороны, является средой выполнения JavaScript на сервере, позволяя разработчикам создавать мощные и масштабируемые серверные приложения. Давайте более подробно рассмотрим каждую из этих технологий.

ЗАКЛЮЧЕНИЕ

В результате выпускной квалификационной работы был разработан веб-сайт школьного музея. Для достижения этой цели были выполнены следующие задачи:

1. Изучены теоретические материалы по разработке веб-сайтов.
2. Рассмотрены основные теоретические аспекты, необходимые для разработки сайта.
3. Рассмотрены способы разработки веб-сайтов.
4. Сформированы основные требования к веб-сайтам.
5. Была проанализирована организация, для которой изготавливается продукт, выявлены проблемы, выдвинуты требования к продукту.
6. Был совершен обзор всех средств разработки.
7. Выбраны средства разработки исходя из требований организации.
8. Выбранные средства были подробно описаны.
9. Спроектирована ER-диаграмма для веб-сайта.
10. Спроектирован прототип сайта.
11. Разработана архитектура веб-сайта, включающая клиентскую и серверную части. Использованы современные технологии и фреймворки для обеспечения надежности и масштабируемости системы.
12. Разработан и реализован интуитивно понятный пользовательский интерфейс, адаптированный для различных устройств.
13. Были проведены всесторонние тестирования веб-сайта для выявления и устранения ошибок.
14. Веб-сайт был опубликован на хостинг и готов к использованию.

Веб-сайт был оснащен необходимым функционалом, включая:

- Возможность просмотра информации о музее
- Раздел новостей
- Раздел документов
- Форма обратной связи и заявок на экскурсии.
- Гид по музею

- Авторизация и администрирование для удобного управления контентом.

Для администратора реализованы такие возможности как:

- Удаление новостей
- Создание новостей с выбором фотографий и файлов
- Обновление новостей – возможность поменять заголовок, текст новости, выбрать новые фотографии и файлов или удалить старые.
- Выбор документов
- Удаление документов

Для обеспечения доступности сайта реализовано управление через клавиатуру без мыши, соблюдена семантика, а также для пользователей смартфонов сделан адаптивный дизайн.

На основании проделанной работы и анализа полученных результатов были предложены рекомендации по дальнейшему развитию и расширению функционала веб-сайта. В частности, можно добавить страницу «Галерея» с фотографиями экспонатов музея и их подробным описанием. Также можно добавить календарь мероприятий музея. Можно улучшить страницу виртуального гида, так как при разработке столкнулись с некоторой сложностью – для загрузки панорамы на Яндекс карты выдвинуты требования, которые обычный телефон покрыть не может, для этого нужна специальная камера с углом обзора в 360 градусов. Позже вместе с руководством школы можно внедрить панораму вместо видео.

В итоге, созданный веб-сайт для школьного музея полностью соответствует поставленным требованиям. Он предоставляет пользователям удобный и современный способ ознакомления с музеем, а также упрощает взаимодействие между посетителями и администрацией музея. Внедрение данного веб-сайта позволит повысить интерес к школьному музею, улучшить информированность посетителей и способствовать развитию культурного образования, повысить внутренний рейтинг школы.

СПИСОК ИСТОЧНИКОВ

1. Полуэктова, Н. Р. Разработка веб-приложений : учебное пособие для вузов / Н. Р. Полуэктова. — 2-е изд. — Москва : Издательство Юрайт, 2024. — 204 с. — (Высшее образование). — ISBN 978-5-534-18645-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 22 — URL: https://urait.ru/bcode/545238/p.22 (дата обращения: 15.04.2024).

2. Тузовский, А. Ф. Проектирование и разработка web-приложений : учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2024. — 219 с. — (Высшее образование). — ISBN 978-5-534-16300-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 31 — URL: https://urait.ru/bcode/537106/p.31 (дата обращения: 09.05.2024).

3. Сысолетин, Е. Г. Разработка интернет-приложений : учебное пособие для вузов / Е. Г. Сысолетин, С. Д. Ростунцев ; под научной редакцией Л. Г. Доросинского. — Москва : Издательство Юрайт, 2023. — 90 с. — (Высшее образование). — ISBN 978-5-9916-9975-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 24 — URL: https://urait.ru/bcode/514303/p.24 (дата обращения: 17.04.2024).

4. Полуэктова, Н. Р. Разработка веб-приложений : учебное пособие для среднего профессионального образования / Н. Р. Полуэктова. — 2-е изд. — Москва : Издательство Юрайт, 2024. — 204 с. — (Профессиональное образование). — ISBN 978-5-534-18644-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 21 — URL: https://urait.ru/bcode/545237/p.21 (дата обращения: 16.04.2024).

5. Кузенкова, Г. В. WEB-технологии. Разработка сайтов : учебное пособие / Г. В. Кузенкова. — Нижний Новгород : ННГУ им. Н. И. Лобачевского, 2020. — 50 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/144688 (дата обращения: 18.04.2024). — Режим доступа: для авториз. пользователей.

6. Леон, У. Разработка веб-приложения GraphQL с React, Node.js и Neo4j / У. Леон ; перевод с английского А. Н. Киселева. — Москва : ДМК

Пресс, 2023. — 262 с. — ISBN 978-5-93700-185-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/314975 (дата обращения: 20.04.2024). — Режим доступа: для авториз. пользователей.

7. Хекслет · Библиотека React: особенности, перспективы, ситуация на рынке труда [Электронный ресурс]. – Режим доступа: https://ru.hexlet.io/blog/posts/biblioteka-react-review-article (дата обращения: 25.04.2022);

8. Хабр: Самые популярные языки программирования бэкенда: для чего они подходят лучше всего и какие компании их используют [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/skillbox/articles/534684/ (дата обращения: 23.04.2022);

9. Хабр: Сравнение SQL- и NoSQL-баз данных [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/ruvds/articles/727474/ (дата обращения: 20.04.2022);

10. Техкульт: Сравнение Javascript-фреймворков React, Angular, Vue.js и Svelte [Электронный ресурс]. – Режим доступа: https://www.techcult.ru/soft/11493-js-frameworks-2022 (дата обращения: 19.04.2022);