Введение

Бережное отношение к нашему здоровью это путь к полноценной жизни. Очень важно сохранить его и не сокрушаться в будущем. Необходимо закладывать привычку заботиться о себе с самого детства. Как много маленьких пациентов не хотят идти к врачу, не хотят сдавать анализы или со слезами делают уколы и прививки. А поход к зубному врачу для многих детей и родителей это испытание.

В хорошем приложении должны быть такие неотъемлемые части как:

1. удобство записи к врачам и отмены посещения. Возможностью самостоятельного выбора способа напоминания о предстоящем визите (по смс, по эл. Почте, на вотсап или в телеграмм)

2. истории лечения в ЛК с датами, фамилией врача, с результами исследований, при необходимости с фотографиями. Здесь обязательно продумать безопасность.

3. отзывы о клинике и врачах

В современном мире информационных технологий здравоохранение играет ключевую роль в обеспечении качественного и доступного медицинского обслуживания. В этом контексте, разработка современного веб-приложения для стоматологической клиники представляет собой актуальную задачу, направленную на оптимизацию процессов в медицинской сфере.

Важность

Разработка такого приложения позволит существенно улучшить качество обслуживания пациентов, уменьшить временные затраты на административные процессы и повысить эффективность работы медицинского персонала. Важной составляющей данного проекта является создание удобного и интуитивно понятного интерфейса, способствующего максимальному комфорту как для пациентов, так и для сотрудников клиники.

Возможности

В рамках данного дипломного проекта планируется использование современных технологий в сфере фронтенд-разработки. Применение HTML и SCSS позволит создать высококачественный и адаптивный пользовательский интерфейс, а фреймворк Vue.js обеспечит плавную и динамичную работу приложения, что существенно повысит его функциональность и привлекательность для пользователей. Объединение этих технологий позволит создать мощный инструмент для управления информацией в стоматологической клинике, что в свою очередь приведет к повышению качества медицинских услуг и улучшению общего опыта взаимодействия с клиентами.

Актуальность

С каждым годом уровень цифровизации во всех областях возрастает, и важность внедрения современных информационных технологий становится все более очевидной. Веб-приложение для стоматологической клиники открывает новые перспективы в управлении данными пациентов, организации записи на прием и обеспечении комфортного взаимодействия между персоналом клиники и пациентами.

Важность фронтенда.

Фронтенд - это лицо любого веб-приложения, интерфейс, через который пользователи взаимодействуют с системой. Он играет решающую роль в формировании первого впечатления от приложения, а также в том, насколько удобно и эффективно пользователи смогут взаимодействовать с функционалом.

Виды веб-приложений

SPA (Single Page Application) — одностраничное интерактивное приложение, которое не требует перезагрузки основной страницы во время своей работы. В отличие от традиционных веб-приложений, они работают быстрее, так как вся логика приложения реализуется в браузере, а не на сервере.

Плюсы:

Удобный интерфейс. Пользователю проще ориентироваться на одной странице, когда есть только один вариант движения — вниз. Никакие дополнительные ссылки не отвлекут его внимание и не позволят запутаться внутри программы.

Высокая скорость. Во время работы, приложение не обновляет всю страницу, а лишь подгружает нужный пользователю контент. Поэтому, оно работает гораздо быстрее многостраничных сайтов — запрос на сервер делается один раз, после чего информация на странице просто дополняется.

Кэширование данных. Точно также работает и кэш — достаточного обратиться к серверу один раз и он сохранит все данные пользователя. Теперь клиент сможет работать в приложении даже в автономном режиме.

Минусы:

Недостаточная SEO-оптимизация. У одностраничных приложений всего один URL-адрес, который не меняется в процессе работы, в отличие от многостраничников. Поэтому разработчик может привязать к ней только один поисковый запрос. Такие сайты трудно оптимизировать под поисковые системы из-за того, что дополнительно подгруженную информацию поисковый бот просто не найдет.

Плохая безопасность. Одностраничные приложения больше подвержены атаке хакеров, которые крадут данные пользователей, с помощью межсайтовых сценариев. Они внедряют их со стороны клиента и легко могут получить конфиденциальную информацию. У MPA и PWA, которые мы рассмотрим ниже, безопасность настроена гораздо лучше.

MPA (Multi Page Application) — многостраничное веб-приложение. Оно позволяет пользователю переходить между отдельными страницами при совершении запроса. У таких ресурсов, как правило, более сложный интерфейс и большое количество дополнительных ссылок. При каждом новом запросе, например, при оплате продукта, у пользователя откроется новая страница.

Плюсы:

Отличное SEO. В отличие от SPA, у MPA каждая страница имеет свой URL-адрес, что дает возможность отлично контролировать SEO-запросы пользователей — на каждую из них, разработчик может добавить свой уникальный ключ и метатеги.

Неограниченная масштабируемость. MPA не ограничивает вас по количеству создания страниц и информации, которую нужно на ней разместить. В отличие от SPA, тут можно использовать множество функций на одной странице. Поэтому, если ваша задача — больше возможностей для клиента, выбирайте MPA.

Минусы:

Низкая скорость. При каждом взаимодействии с приложением, сервер перезагружает и обновляет все ресурсы — HTML, CSS, скрипты и все данные страницы. Поэтому скорость ожидания ответа на запрос, увеличивается, что негативно может повлиять на производительность приложения.

Более сложная разработка. При создании такого приложения, требуется полноценная разработка серверной и фронтальной части программы и использование фреймворков для внешнего и внутреннего интерфейса. Все это занимает гораздо больше времени, в сравнении с созданием SPA.

Как сделать сайт более быстрым. Для создания быстрого веб-приложения, следует учитывать ряд основополагающих принципов и применять оптимизационные техники: Оптимизация изображений и мультимедийных файлов: Используйте сжатие изображений и видео без потери качества. Избегайте загрузки изображений большого разрешения, если они не требуются в таком формате. Ленивая загрузка контента: Загружайте контент только когда он необходим пользователю. Например, изображения за пределами видимой области могут загружаться только при прокрутке. Минимизация HTTP-запросов: Объединяйте файлы CSS и JavaScript, чтобы сократить количество запросов к серверу. Используйте CSS-спрайты для сокращения числа запросов к изображениям. Кеширование ресурсов: Используйте HTTP-кэширование для временного хранения ресурсов на стороне клиента. Это позволяет браузеру использовать ранее загруженные данные вместо их повторной загрузки. Минимизация и объединение CSS и JavaScript файлов: Удалите избыточные пробелы, комментарии и лишние символы из файлов. Объедините несколько файлов в один, чтобы уменьшить количество запросов. Использование CDN (Content Delivery Network): Используйте сети доставки контента для хранения и распространения ресурсов (например, библиотек JavaScript, стилей) на сервера распределенные по всему миру. Это ускорит загрузку ресурсов. Оптимизация серверной части: Поддерживайте эффективный хостинг, используйте кэширование на сервере, уменьшите время обработки запросов. Использование асинхронных запросов: Используйте асинхронные запросы для загрузки данных и контента на страницу, чтобы не блокировать остальной контент во время загрузки. Использование шрифтов локально: Если используются веб-шрифты, предпочтительно загружать их с сервера вместо подключения через сторонние сервисы, что может привести к дополнительной задержке. Регулярные тесты и анализ производительности: Регулярно тестируйте производительность своего приложения с помощью инструментов, таких как Google PageSpeed Insights, и исправляйте выявленные узкие места. Использование сжатия GZIP/Deflate: Настройте сервер так, чтобы он отправлял данные сжатыми, что уменьшит объем передаваемой информации. Оптимизация базы данных: Оптимизируйте запросы к базе данных и используйте индексы для ускорения поиска данных. Соблюдение этих принципов и техник поможет создать быстрое и отзывчивое веб-приложение для стоматологической клиники, что повысит удовлетворенность пользователей и эффективность работы персонала.

Для создания переходов между страницами буду использовать библиотеку Vue Router.

Vue Router - это официальная библиотека маршрутизации для фреймворка Vue.js. Она позволяет создавать одностраничные приложения (SPA) и управлять навигацией между разными видами (views/pages) или компонентами вашего веб-приложения.

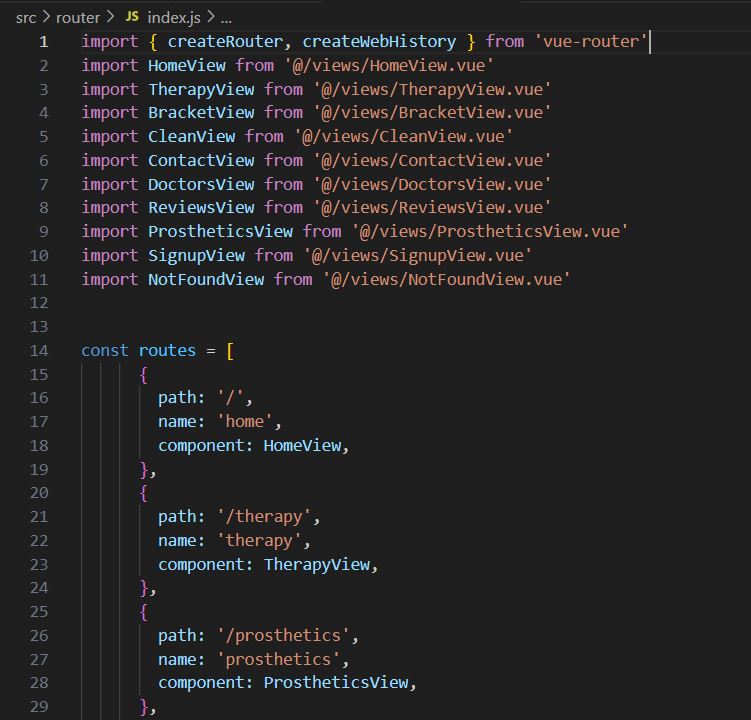
Vue Router предоставляет возможность определять маршруты, которые соотносятся с определенными URL-адресами. Каждый маршрут связывается с определенным компонентом Vue, который отображается при переходе по данному URL.

Для его подключение необходимо:

1. Установить библиотеку одной из команд : **npm install vue-router** или **yarn add vue-router.**
2. Импорт и подключение Vue Router в main.js



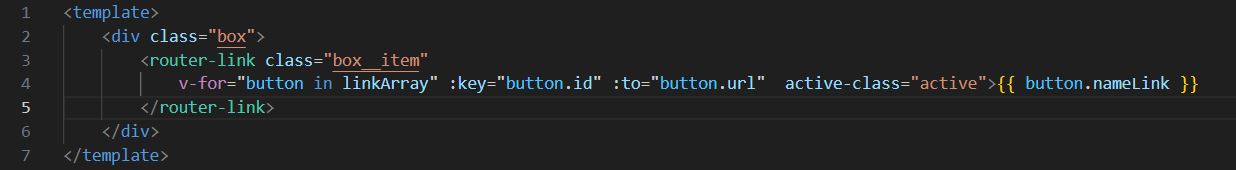
1. В папке router => index.js импортируется роутер. В этом же файле мы создаем переменную router, в которой будут храниться объекты – странички нашего приложения (адрес ссылки, название страницы и компонент).



Для странички NotFound путь будет выглядеть так:



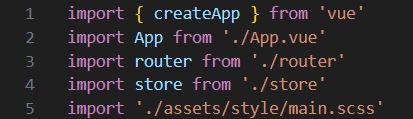
1. Создаем компонент NavComp, который будет отвечать за переход по страничкам.



С помощью компонента router-link мы отображаем пользователю ссылку, cвойство to определяет URL для перехода, так же добавляем active-class, который будет отвечать за стилистику активной кнопки. Затем в главном компоненте App.vue подключаем компонент router- view, который отвечает за отображение, соответствующего компонента.

Подключение стилей.

В своей работе я буду использовать препроцессоры, с их помощью можно структурировать стили путем создания отдельных файлов с переменными и миксинами и импорта их в главный файл стиля main.scss, который в свою очередь импортируется в main.js:



Так как у каждого компонента своя область видимости, чтобы использовать переменные и миксины в отдельных компонентах необходимо сделать импорт файла с переменными в сам компонент.



@font-face – позволяет использовать свои собственные шрифты, не зависящие от наличия шрифтов на устройстве пользователя. Это полезно, когда нужно использовать специфические шрифты, которые не доступны по умолчанию. Рекомендуется предоставлять шрифты в нескольких форматах для лучшей поддержки различными браузерами. Форматы woff и woff2 часто используются для веба, так как обеспечивают хорошее сжатие и поддержку большинства современных браузеров.

Компоненты, отвечающие за страницы моего приложения будут храниться в отдельной папке views, в то же время, другие компоненты , которые могут быть частью отдельных страниц, будут храниться в папке components, что позволит сохранять порядок в структуре. Например, на странице «Наши врачи» в зависимости от того куда нажимает пользователь подгружаются разные компоненты, отвечающие за отображение информации о каждом докторе в отдельности.



Подключение API.

**API** (англ. Application Programming Interface — программный интерфейс приложения) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными. Интерфейс — это граница между двумя функциональными системами, на которой происходит их взаимодействие и обмен информацией. При этом процессы внутри каждой из систем скрыты друг от друга.

За последние несколько лет API среди веб-проектов получили довольно большой спрос. Сегодня сложно представить работу веб-разработчиков без такого инструмента как API, которые также стали очень эффективным.

В своей работе я буду подключать Google Map. Для этого надо получить на официальном сайте API ключ.

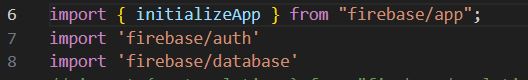
В дипломной работе в качетсве БД буду использовать платформу “Firebase”

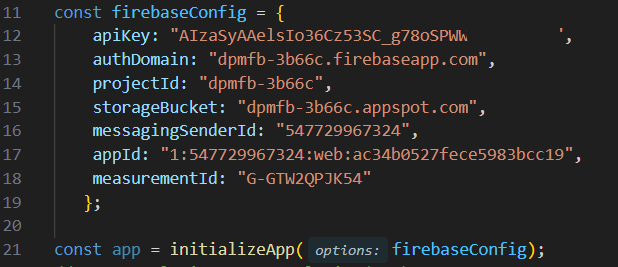
Firebase - это облачная платформа, разработанная Google, которая предоставляет разработчикам инструменты для создания приложений. Она позволяет создавать высококачественные приложения для веба, мобильных устройств и других платформ, предоставляя готовые инструменты для разработки и развертывания приложений без необходимости управления инфраструктурой.

Firebase включает в себя:

* Firebase Realtime Database: Облачная база данных реального времени.
* Firebase Authentication: Механизм аутентификации и управления пользователями.
* Firebase Cloud Messaging: Сервис для отправки уведомлений на устройства пользователей.
* Firebase Hosting: Облачный хостинг для статического и динамического контента.
* Другие компоненты (Firebase Analytics, Firebase Functions и т.д.), которые могут быть релевантными для конкретного исследования.

Для подключение Firebase необходимо сделать соответствующие импорты в main.js:





Сделать!:

1. ЛК,

2. Подключить БД

3. API: гугл карты, погода

4. Возможность онлайн записи

4.1 подключение к crm

4.2 автоматическая отправка письма или сообщения о записи

4.3 Автоматическое напоминание о записи за сутки

Изображения взяты из источника:

<a href="https://www.freepik.com/free-vector/set-teeth-treatment-hygiene-with-prosthesis\_5686262.htm#query=svg%20illustration%20tooth&position=14&from\_view=search&track=ais">Image by studiogstock</a> on Freepik