МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Институт (Филиал) № 8 «Компьютерные науки и прикладная математика» **Кафедра** 806

Группа М8О-411Б-19 **Направление подготовки** 02.03.02 «Фундаментальная информатика и информационные технологии»

Профиль Физика

Квалификация: бакалавр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему «Разработка клиентской части web-приложения onmp.ru на React»

Автор ВКРБ:	Сергеев Владислав Алексеевич	(_____)
Руководитель:	Пивоваров Дмитрий Евгеньевич	(_____)
Консультант:	—	(_____)
Консультант:	—	(_____)
Рецензент:	—	(_____)

К защите допустить

Заведующий кафедрой № 806 Крылов Сергей Сергеевич (_____)

____ мая 2023 года

Москва 2023

РЕФЕРАТ

Выпускная квалификационная работа бакалавра состоит из 41 страницы, 10 рисунков, 6 использованных источников, 1 приложения.

WEB-СЕРВИС, ВРАЧИ, ОНМП, КАРТА ВЫЗОВА, КЛИЕНТСКАЯ ЧАСТЬ, REACT.JS

Объектом разработки является Web-сервис для врачей из отделения неотложной медицинской помощи, позволяющий в электронной форме заполнять карты вызова.

С целью достижения поставленной задачи был выполнен анализ функциональных требований, связанных с разработкой сервиса, а также проведено исследование методов, используемых для заполнения карт вызова.

Данное решение было реализовано на языке JavaScript с использованием следующих библиотек и инструментов:

- React.js является основной библиотекой для разработки клиентской части web-сервиса.
- Axios предоставляет удобный и гибкий способ взаимодействия с серверной частью.
- Redux является библиотекой для управления состоянием приложения.

Основным результатом работы, полученным в результате разработки, является функциональная и эффективная система, заменяющая ручную бумажную работу медицинских работников. Реализованный веб-сервис позволяет медицинским работникам легко заполнять, хранить и осуществлять поиск карт вызова.

Полученные результаты разработки предоставляют немаловажное значение для отделения неотложной помощи, так как позволяют существенно улучшить эффективность и точность работы медицинских работников, а также повысить качество обслуживания пациентов.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей выпускной квалификационной работе бакалавра применяют следующие термины с соответствующими определениями:

Веб-сервис — это программное обеспечение, которое предоставляет функциональность и данные через сеть Интернет. Он позволяет взаимодействовать с удаленными компонентами или приложениями посредством сетевых протоколов, обычно HTTP

Интерфейс — это точка взаимодействия между двумя или более сущностями, позволяющая им обмениваться информацией или взаимодействовать друг с другом. В контексте разработки программного обеспечения, интерфейс определяет способы взаимодействия пользователя с системой или между различными компонентами системы

Клиентская часть — это часть программного приложения, которая выполняется на стороне клиента, то есть на компьютере пользователя или на устройстве пользователя, таком как веб-браузер или мобильное устройство. Клиентская часть отвечает за представление данных и взаимодействие с пользователем

Фреймворк — это набор программных инструментов, библиотек и правил, предназначенных для разработки приложений. Фреймворк предоставляет основу и структуру для создания приложений, определяя общую архитектуру, стандарты и методологии разработки

API — это набор правил и соглашений, которые определяют, как различные компоненты программного обеспечения должны взаимодействовать друг с другом. API определяет методы и форматы данных, которые могут использоваться для обмена информацией и выполнения определенных операций

Frontend — это термин, который используется в веб-разработке для обозначения части проекта, которая отвечает за создание и отображение пользовательского интерфейса. Он включает в себя все элементы, которые видит пользователь и с которыми он взаимодействует при использовании веб-приложений или сайтов

JavaScript — это высокоуровневый, интерпретируемый язык программирования, который широко используется для создания динамических веб-страниц. Он предоставляет возможности для разработки

интерактивных элементов на веб-странице и взаимодействия с пользователем

HTTP-запросы — это средство взаимодействия между клиентскими приложениями и серверами в сети Интернет. Протокол HTTP определяет стандартные методы и форматы запросов, которые клиенты могут отправлять серверам для получения данных или выполнения определенных операций

UX-дизайн — относится к процессу проектирования и улучшения взаимодействия пользователей с продуктом или сервисом, с целью обеспечения максимально положительного и удовлетворительного опыта пользователя. Он фокусируется на создании продуктов, которые легко и интуитивно понятны, удобны в использовании и соответствуют потребностям и ожиданиям пользователей

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей выпускной квалификационной работе бакалавра применяют следующие сокращения и обозначения:

ОНМП — Отдел Неотложной Медицинской Помощи

API — Application Programming Interface

FSD — Feature-Sliced Design

UX-Design — User Experience Design

ВВЕДЕНИЕ

Актуальность данной работы связана с необходимостью оптимизации процесса работы врачей из отделения неотложной медицинской помощи, особенно тех, которые выезжают на вызовы с использованием машин скорой помощи. В настоящее время эти медицинские специалисты заполняют бумажные карты вызова во время осмотра пациента на дому. Однако, такой подход требует ношения с собой больших объемов бумажной документации, что требует больших затрат времени и ресурсов на их заполнение, обработку и хранение.

Переход к электронному заполнению карт вызова имеет ряд значительных преимуществ. Прежде всего, медицинским работникам будет значительно удобнее носить с собой меньшее количество объемной макулатуры, так как электронная форма позволит им быстро и удобно заполнять карты вызова с использованием мобильных устройств, которые они всегда имеют под рукой. Это значительно повысит мобильность и гибкость работы медицинского персонала, позволяя им быстро и эффективно заполнять и обрабатывать данные.

Второе преимущество заключается в возможности хранения электронных карт вызова в интернете. Вместо физического хранения и поиска бумажных документов, медицинские работники смогут сохранять все карты вызова в электронном виде, обеспечивая быстрый и удобный доступ к ним из любого места. Это также позволит легко распечатывать необходимые карты вызова в случае необходимости, например, для передачи другим специалистам или для ведения медицинской документации.

Основной целью данной работы является разработка клиентской части web-сервиса с использованием технологии React.js. Это позволит создать современный и интуитивно понятный пользовательский интерфейс, который облегчит заполнение карт вызова, ускорит процесс ввода данных и повысит эффективность работы медицинских работников.

Был проведен анализ требований к функциональности и пользовательскому интерфейсу web-сервиса, разработана архитектура решения, реализованы необходимые компоненты и проведено тестирование системы. Также была осуществлена интеграция клиентской части с серверной частью системы для обеспечения полной функциональности и

взаимодействия с данными.

В рамках данной исследовательской работы можно определить следующие задачи:

- Анализ требований к клиентской части web-сервиса: Эта задача включает в себя изучение требований, предъявляемых заказчиком к функциональности и дизайну клиентской части web-сервиса. Необходимо провести тщательный анализ требований, чтобы полноценно понять ожидания и потребности заказчика.
- Проектирование клиентской части: включает разработку детального плана и архитектуры клиентской части web-сервиса. Эта задача включает определение структуры компонентов, взаимодействия между ними, выбор подходящих библиотек и фреймворков для разработки, а также определение способа организации данных и обработки событий.
- Разработка функционала авторизации и аутентификации: Данный функционал позволит медицинским работникам входить в систему с использованием уникальных учетных данных. Каждому пользователю будет предоставлен доступ к рабочей среде, где они смогут создавать и использовать собственные шаблоны, а также просматривать и заполнять свои собственные карты вызова.
- Создание интерфейса для заполнения и сохранения карт вызова: Разработка пользовательского интерфейса, позволяющего медицинским работникам вводить информацию о состоянии здоровья пациента, тем самым заполняя карту вызова. Данные должны быть легко вводимы и интуитивно понятны, чтобы ускорить процесс заполнения.
- Реализация функционала хранения и организации карт вызова в папках: Готовые, Незавершенные, Архив, Шаблоны. Для организации и удобного отображения карт вызова. Это позволит медицинским работникам быстро найти и просмотреть необходимые карты вызова в соответствующих категориях.
- Разработка функционала создания и использования шаблонов: Реализация возможности создания шаблонов для быстрого использования частично заполненных карт вызова. Создавая шаблон, пользователь заполняет только некоторые поля карты,

чтобы в дальнейшем использовать данный шаблон как основу для карты вызова.

- Реализация Функционала поиска, группировки и сортировки карт вызова: Предоставление возможности быстрого поиска нужной информации на основе различных параметров, таких как дата, название карты. Кроме того, группировка и сортировка карт вызова позволят легко ориентироваться в большом объеме данных, что значительно упростит процесс поиска нужной информации.
- Тестирование и отладка системы: Проведение тестирования для проверки функциональности и корректности работы системы. Исправление ошибок и устранение неполадок для обеспечения стабильной работы web-сервиса.

В данной работе разработка клиентской части web-сервиса осуществляется с использованием React.js - популярной JavaScript-библиотеки для построения пользовательских интерфейсов. React.js предоставляет эффективные инструменты и подходы для создания интерактивных веб-приложений с удобной обработкой данных и переиспользуемыми компонентами. Данная библиотека базируется на концепции компонентного подхода, где пользовательский интерфейс разбивается на независимые компоненты, каждый из которых может иметь свою логику и состояние. Компоненты могут быть переиспользованы, что упрощает разработку и обслуживание приложения.

Дополнительным инструментом к React был выбран Redux. Это библиотека JavaScript, которая предоставляет инструменты для управления состоянием приложения. Она является популярным выбором для разработки сложных и масштабируемых приложений, особенно тех, которые имеют большое количество взаимосвязанных данных.

Redux основывается на концепции однонаправленного потока данных и централизованного хранилища состояния. Вместо того чтобы хранить состояние внутри компонентов, Redux предлагает вынести состояние в отдельное хранилище. Компоненты могут получать доступ к состоянию из хранилища и обновлять его, используя специальные функции.

В целом, совместное использование React и Redux позволяет создавать масштабируемые, предсказуемые и легко поддерживаемые приложения с удобным управлением состоянием [1].

Помимо этого, был выбрана библиотека Axios, которая предоставляет удобный интерфейс для выполнения HTTP-запросов из браузера. Она позволяет взаимодействовать с внешними API, отправлять запросы на сервер и обрабатывать полученные ответы.

Главная цель Axios - упростить и сделать более эффективной работу с HTTP-запросами. Она предоставляет простые методы для отправки различных типов запросов, таких как GET, POST, PUT, DELETE и другие. Кроме того, Axios предоставляет возможности для установки заголовков запроса, обработки ошибок, использования интерсепторов и других функций.

Разработка данного сервиса основывалась на архитектурной методологии Feature-Sliced Design, которая помогает структурировать код и организовать проект на основе функциональных возможностей. Она призвана улучшить масштабируемость, переиспользуемость и поддерживаемость кодовой базы. Основная идея данной методологии заключается в том, что проект разбивается на слои, каждый из которых имеет свою ”зону ответственности”. Каждый модуль содержит все необходимые компоненты, состояние, стили и логику, связанные с этой функциональностью. Это позволяет разработчикам работать над каждой фичей независимо друг от друга.

Результатом данной работы будет функциональный web-сервис, предоставляющий возможность электронного заполнения и хранения карт вызова для медицинских работников из отделения неотложной помощи. Ожидается, что данное решение значительно улучшит процесс работы медицинских учреждений, сократит временные затраты на бумажную работу и повысит качество медицинской помощи пациентам.

Таким образом, данная дипломная работа имеет практическую значимость и является важным шагом в совершенствовании сферы оказания неотложной медицинской помощи, предлагая инновационное решение, которое отсутствует на рынке.

1 ПОТРЕБНОСТЬ В РАЗРАБОТКЕ

1.1 Анализ текущих методов заполнения и хранения карт вызова

В настоящее время, при вызове скорой помощи, медики используют методические пособия и бланки карт вызова, которые хранятся в объемной папке.

Метод заполнения карт вызова имеет свои особенности. В зависимости от состояния пациента медики заполняют различные разделы карты, начиная от основной информации о пациенте и заканчивая деталями о его состоянии, симптомах и проведенных медицинских процедурах.

Для правильного заполнения карт вызова медицинскими работниками требуется знание определенных протоколов и стандартов. Для этого они обычно имеют с собой несколько методических пособий, содержащих инструкции по заполнению различных разделов карты вызова в зависимости от состояния пациента. Это усложняет процесс и требует дополнительного времени и усилий. Помимо этого, медицинские работники заполняют карты вызова в процессе осмотра пациента. Это требует их присутствия и внимания к пациенту, а также необходимости записывать информацию вручную.

Хранение карт вызова также имеет свои особенности. Обычно эти документы хранятся в специальных папках или ящиках машины скорой помощи, доступных только медицинским работникам, которые после окончания смены передаются отделению в медицинском учреждении для формирования отчетности. Это создает дополнительный вес и занимает место в машине скорой помощи, а также создаёт проблемы с доступом к предыдущим записям и усложняет поиск и извлечение информации о конкретных пациентах или случаях.

Однако, существуют и недостатки текущих методов заполнения и хранения карт вызова. Например, при использовании бумажных бланков возможны ошибки в заполнении или потеря документов. Кроме того, доступ к картам вызова может быть ограничен, что затрудняет своевременное и эффективное оказание медицинской помощи.

В связи с этим, возникает потребность в новых технологиях, которые позволят ускорить процесс заполнения карт вызова, уменьшить количество ошибок и обеспечить безопасное хранение этих документов. Таким образом

разработка web-сервиса, позволяющего заполнять карты вызова в электронной форме, представляет собой актуальное и необходимое решение для улучшения и оптимизации процесса работы медицинских работников из отделения неотложной медицинской помощи.

В целом, анализ текущих методов заполнения и хранения карт вызова показал, что они имеют свои преимущества и недостатки. Но в современном мире, где ценится скорость и качество медицинской помощи, возникает потребность в новых технологиях, которые помогут повысить эффективность работы медицинских работников и обеспечить быстрое и точное оказание помощи пациентам.

1.2 Описание проблем и ограничений существующих решений

Анализируя существующие решения, такие как сайт-прототип заполнения карт вызова onmp.ru, представленный на рисунке 1,

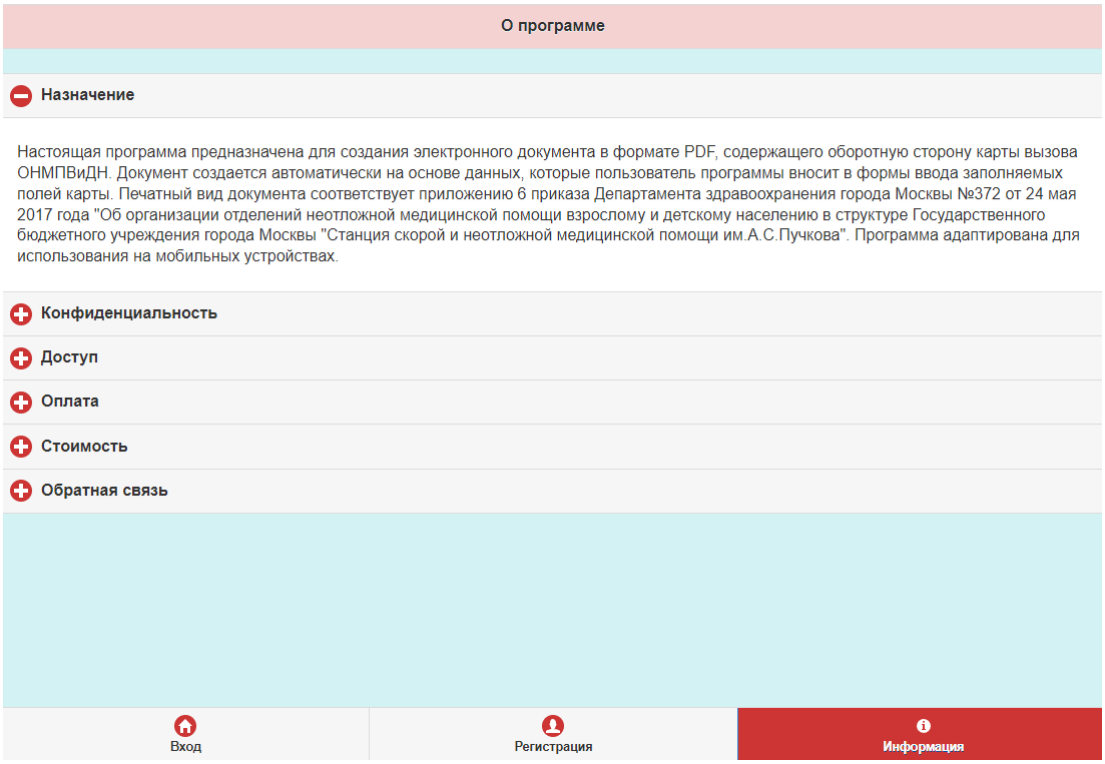

Рисунок 1 – Сайт-прототип onmp.ru

можно выделить несколько проблем и ограничений, которые ограничивают его удобство использования и эффективность:

Язык программирования PHP: Использование PHP для разработки сайта может ограничивать его гибкость и масштабируемость. PHP является языком

серверной разработки, и его использование может усложнить внесение изменений и расширение функциональности клиентской части.

Неудобный UX-дизайн: Отсутствие интуитивного и удобного UX-дизайна на сайте может создавать сложности при навигации между разделами, а также затруднять быстрый доступ к нужной информации. Неочевидные переходы и отсутствие удобной навигации могут снижать эффективность работы мед.работников и увеличивать время заполнения карт вызова.

Отсутствие промежуточного сохранения: Наличие функции промежуточного сохранения карт вызова является важным аспектом для удобства пользователей. Отсутствие такой функции на сайте может приводить к потере данных в случае прерывания работы или случайного закрытия страницы.

Ограниченные возможности навигации по карте: Отсутствие быстрой навигации по карте вызова может затруднять поиск и редактирование определенных разделов или полей, особенно в случае больших и сложных карт.

Учитывая эти проблемы и ограничения существующего решения, разработка нового web-сервиса для заполнения карт вызова с использованием React.js имеет потенциал для устранения данных проблем и предоставления более удобного и эффективного инструмента для медицинских работников из отделения неотложной помощи.

1.3 Обзор преимуществ перехода на электронную форму заполнения карт вызова

Переход на электронную форму заполнения и хранения карт вызова имеет несколько преимуществ.

- Использование электронной формы позволяет существенно сократить потребление бумаги и уменьшить негативное влияние на окружающую среду. Это способствует экологической устойчивости и снижению расходов на закупку и хранение бумажных материалов.
- Электронная форма позволяет быстро и удобно заполнять карты вызова, что ведет к повышению эффективности работы медицинских работников. Заполнение осуществляется путем выбора и ввода данных в соответствующие поля, что сокращает

время, затрачиваемое на ручное заполнение бумажных форм.

- Электронная форма обеспечивает легкий доступ к заполненным картам вызова из любого места с подключением к Интернету. Это позволяет медицинскому персоналу быстро получать необходимую информацию о пациентах, это особенно полезно для медицинских работников, которые находятся в пути, например, в машине скорой помощи. Они могут легко получить доступ к системе и заполнить карты вызова на своих устройствах. Также электронна форма облегчает анализ и обработку данных для статистической отчетности и исследований. Помимо этого, медицинский персонал может получить доступ к предыдущим записям о состоянии пациента и процедурах, которые ему проводились, что помогает улучшить качество медицинской помощи и сделать более точный диагноз.
- При использовании электронной формы можно внедрить функциональность автоматического заполнения и проверки данных. Например, система может предложить автозаполнение некоторых полей на основе предыдущих записей или данных из других систем, а также проводить проверку на наличие ошибок или недостаточных данных.
- Электронные карты вызова могут быть сохранены в централизованной базе данных, что облегчает их архивирование и поиск. Можно использовать различные фильтры и категории для организации и классификации карт по статусу (готовые, незавершенные, архив) или другим параметрам, упрощая их управление и обработку.
- В электронной форме можно создавать шаблоны часто используемых или стандартных карт вызова. Это позволяет медицинскому персоналу быстро заполнять карты, используя предварительно сохраненные данные, что повышает производительность и снижает вероятность ошибок.

Переход на электронную форму заполнения карт вызова значительно сокращает бумажную работу, повышает эффективность и точность процесса заполнения, улучшает доступность и хранение данных, а также облегчает анализ и управление информацией. Это современный и инновационный

подход, который может значительно улучшить работу мед.работников и повысить качество оказываемой медицинской помощи.

2 АНАЛИЗ ТРЕБОВАНИЙ ЗАКАЗЧИКА К ФУНКЦИОНАЛЬНОСТИ И ДИЗАЙНУ

2.1 Описание требований к интерфейсу заполнения карт вызова

Эффективное и надежное заполнение карт вызова является важным процессом для медицинских работников, поскольку это напрямую влияет на качество медицинской помощи. Интерфейс, который будет использоваться для заполнения карт вызова, должен быть специально разработан, чтобы обеспечить максимальную эффективность и удобство использования.

Для удовлетворения этих требований заказчиком было предложено несколько критериев, которым должен соответствовать медицинский интерфейс:

- Интерфейс должен быть легким в освоении и использовании даже для пользователей с минимальными навыками работы с компьютером. Он должен предоставлять понятные и логично расположенные элементы управления, а также ясные инструкции для заполнения каждого поля.
- Интерфейс должен быть организован логически, с ясной и понятной структурой разделов и подразделов, отражающих последовательность заполнения карты вызова. Навигация должна быть интуитивной и обеспечивать быстрый переход между разделами, чтобы медицинскому работнику было удобно перемещаться по карте вызова.
- Интерфейс должен предоставлять функциональность сохранения заполненной информации, чтобы медицинским работникам не приходилось вводить одни и те же данные снова и снова при каждом новом вызове. Также желательно наличие функции автозаполнения полей на основе предыдущих записей или шаблонов.
- Интерфейс должен быть адаптивным и оптимизированным для работы на различных устройствах, включая компьютеры, планшеты и смартфоны. Это обеспечит удобство использования сервиса как в офисной среде, так и на мобильных платформах, когда медицинские работники находятся на вызове.

– Интерфейс должен предоставлять возможность интеграции с дополнительными функциональностями, которые помогут медицинским работникам в процессе заполнения карты вызова. Например, он может предоставлять доступ к алгоритмам оказания помощи, таблицам дифференциальной диагностики, калькуляторам дозировок и справочникам, которые помогут в принятии решений и заполнении необходимых полей.

2.2 Рассмотрение требований к организации папок и шаблонов карт вызова

При организации папок и шаблонов карт вызова, необходимо учитывать множество требований для обеспечения максимальной эффективности и удобства использования.

В первую очередь, необходимо установить четыре папки хранения карт вызова: Готовые, Незавершенные, Архив и Шаблоны.

В папку Готовые попадают карты после того, как пользователь заполнит все необходимые поля на карте вызова и нажмет кнопку ”Сохранить”. Это позволяет пользователю быстро найти уже готовые карты вызова и не тратить время на поиск незавершенных карт.

В папку Незавершенные попадают карты, когда пользователь возвращается в каталог карт, закрывает вкладку или не нажимает кнопку ”Сохранить”. При этом, уже заполненные данные сохраняются в процессе заполнения в базу данных, чтобы пользователь мог продолжить заполнение карты вызова с того места, где он остановился. Это позволяет избежать потери уже заполненных данных и повторного заполнения всех полей заново.

В папку Архив попадают карты после того, как они были распечатаны. Это гарантирует сохранность уже готовых карт вызова, которые могут понадобиться в будущем для анализа статистики и других целей.

Папка Шаблоны предоставляет дополнительный функционал создания шаблона с частичным заполнением полей карты вызова. Это позволяет пользователям создавать базовые шаблоны карт вызова, которые можно использовать в будущем как основу для новых карт вызова. Это значительно ускоряет процесс заполнения карт вызова, так как многие поля уже будут заполнены заранее.

Такая организация папок и шаблонов карт вызова обеспечит удобство

хранения, поиска и использования карт, а также улучшит процесс работы медицинского персонала, обеспечивая быстрый доступ к необходимым данным и возможность повторного использования шаблонов.

Дополнительные требования к функциональности организации папок и шаблонов карт вызова:

Сортировка по дате:

- Возможность сортировки карт вызова внутри каждой папки по дате их создания или последнего изменения.
- Медицинским работникам предоставляется возможность выбора порядка сортировки (по возрастанию или убыванию даты).

Сортировка по названию карты:

- Возможность сортировки карт вызова внутри каждой папки по их названию.
- Медицинским работникам предоставляется возможность выбора порядка сортировки (по алфавиту, по возрастанию или убыванию).

Группировка:

- Возможность группировки карт вызова по определенным параметрам, например, по дате и названию карты.
- Медицинским работникам предоставляется возможность выбора группировки и просмотра карт вызова внутри каждой группы.

Реализация сортировки и группировки в каждой папке улучшит удобство работы с картами вызова, позволит пользователю быстро находить нужную информацию и эффективно управлять списками карт. Это значительно повысит эффективность работы и улучшит пользовательский опыт.

3 ТЕХНОЛОГИЧЕСКИЙ СТЕК И АРХИТЕКТУРНАЯ МЕТОДОЛОГИЯ

3.1 Обоснование выбора React.js в качестве основной технологии разработки

3.1.1 Обзор существующих технологий и фреймворков

При разработке клиентской части web-сервисов существует несколько популярных технологий и фреймворков, которые предоставляют разработчикам средства для создания интерактивных и отзывчивых пользовательских интерфейсов. В данном разделе мы рассмотрим три из них: Angular, Vue.js и React.js.

Angular - это фреймворк, разработанный компанией Google. Он использует язык программирования TypeScript и предоставляет набор инструментов для создания масштабируемых приложений [2].

Основные особенности Angular:

- компонентная архитектура: Angular основан на компонентах, которые являются основными строительными блоками приложения. Компоненты позволяют создавать переиспользуемые элементы интерфейса и упрощают управление состоянием;
- двустороннее связывание данных: Angular предлагает двустороннее связывание данных, что позволяет автоматически синхронизировать данные между моделью и представлением;
- мощный набор инструментов: Angular предоставляет широкий спектр инструментов для разработки, таких как маршрутизация, формы, валидация, анимации и многое другое;
- строгая типизация: TypeScript, язык программирования, используемый Angular, предлагает статическую типизацию, что улучшает безопасность и облегчает разработку сложных приложений;

Vue.js - это прогрессивный JavaScript-фреймворк с открытым исходным кодом. Он создан для разработки пользовательских интерфейсов и может быть поэтапно внедрен в существующие проекты [3].

Основные особенности Vue.js:

- простота и гибкость: Vue.js предлагает простой и интуитивно понятный API для создания компонентов и управления состоянием.

Он также обладает гибкостью, позволяющей использовать только необходимые части фреймворка;

- реактивность: Vue.js использует систему реактивности, которая автоматически обновляет пользовательский интерфейс при изменении данных;
- однофайловые компоненты: Vue.js позволяет создавать компоненты с помощью одного файла, включающего шаблон, скрипт и стили. Это делает разработку и поддержку компонентов более организованной и удобной;
- экосистема и сообщество: Vue.js имеет активное сообщество разработчиков и обширную экосистему плагинов и инструментов, которые упрощают разработку и расширение функциональности;

React - это JavaScript-библиотека, разработанная Meta (Компания признана экстремистской и запрещенной в России). Она используется для создания пользовательских интерфейсов и часто применяется в разработке одностраничных приложений [4].

Основные особенности React:

- компонентный подход: React основан на компонентах, которые позволяют создавать переиспользуемые элементы интерфейса. Компоненты в React обладают своим состоянием и жизненным циклом, что упрощает разработку и поддержку приложений;
- виртуальный DOM: React использует виртуальный DOM, который позволяет эффективно обновлять только измененные части пользовательского интерфейса, что повышает производительность;
- односторонний поток данных: В React данные передаются через props от родительских компонентов к дочерним. Это делает управление состоянием более предсказуемым и упрощает отладку;
- большое сообщество и экосистема: React имеет широкое сообщество разработчиков и богатую экосистему инструментов, библиотек и компонентов, которые облегчают разработку и ускоряют процесс создания приложений;

3.1.2 Преимущества React.js

При выборе основной технологии разработки для клиентской части web-сервиса было принято решение использовать React.js. Рассмотрим основные

аргументы и преимущества, которые оказали влияние на это решение:

- React.js является одной из самых популярных JavaScript-библиотек для создания пользовательских интерфейсов. Он активно используется множеством компаний и разработчиков по всему миру. Большое сообщество разработчиков React способствует обмену знаниями, наличию документации, обновлениям и поддержке.
- React.js предлагает гибкую и модульную архитектуру разработки. Он позволяет создавать компоненты, которые могут быть переиспользованы и управляться независимо друг от друга. Это упрощает разработку сложных пользовательских интерфейсов и облегчает сопровождение кода.
- React.js использует виртуальный DOM, который является эффективным механизмом обновления только измененных частей пользовательского интерфейса. Это приводит к улучшенной производительности и отзывчивости приложения, особенно при работе с большими объемами данных.
- React.js обладает богатой экосистемой инструментов, библиотек и компонентов, которые могут быть использованы для ускорения разработки. Наличие таких инструментов, как Redux для управления состоянием, React Router для маршрутизации, Styled Components для стилизации и многих других, делает разработку более эффективной и продуктивной.
- React.js обеспечивает хорошую тестируемость кода благодаря своей компонентной архитектуре. Тестирование компонентов React может быть проведено с использованием различных инструментов и библиотек, таких как Jest или Enzyme. Это позволяет обеспечить высокую степень надежности и качества разрабатываемого приложения.
- React.js поддерживается активным сообществом разработчиков и командой Facebook. Это гарантирует наличие обновлений, исправлений ошибок и новых возможностей. Кроме того, Facebook активно продвигает и разрабатывает новые инструменты и библиотеки в рамках экосистемы React.

Исходя из этих преимуществ и аргументов, выбор React.js в качестве

основной технологии разработки для клиентской части web-сервиса является обоснованным и позволит нам создать современное, масштабируемое и эффективное приложение.

3.2 Описание используемых инструментов

3.2.1 React Router

React Router является мощным инструментом для управления маршрутизацией веб-приложений на платформе React. Он предоставляет удобные и гибкие средства для определения и управления различными маршрутами приложения, позволяя пользователям перемещаться между различными страницами и взаимодействовать с контентом.

Основной концепцией в React Router является использование компонента <Route>. Каждый <Route> определяет отдельный маршрут и указывает, какой компонент должен быть отображен при соответствии данному маршруту. Например, <Route path=”/calculator”component=Calculator /> означает, что при обращении к маршруту ”/calculator”будет отображаться компонент Calculator.

Для создания ссылок между страницами используется компонент <Link>. Он позволяет создавать кликабельные элементы, которые перенаправляют пользователя на другие маршруты приложения. <Link to=”/calculator»Calculator</Link> создает ссылку на маршрут ”/calculator при нажатии на которую пользователь будет перенаправлен на страницу Calculator.

Для работы с динамическими параметрами в URL, такими как идентификаторы или другие переменные, React Router предлагает использовать параметризованные маршруты. Например, <Route path=”/users/:id”component=User /> определяет маршрут, включающий динамический параметр ”:id”. Значение этого параметра может быть извлечено в компоненте User с помощью хука useParams().

React Router также предоставляет механизм вложенных маршрутов. Это позволяет создавать иерархическую структуру маршрутов, где определенные компоненты могут быть вложены в другие компоненты и иметь свои собственные подмаршруты. Это особенно полезно для создания сложных макетов приложений с различными уровнями вложенности.

Помимо основных компонентов, React Router также предлагает другие полезные возможности, такие как редиректы, защита маршрутов с помощью приватных роутов, анимации переходов между страницами и многое другое.

Одним из главных преимуществ React Router является его простота использования и интеграция с экосистемой React. Он предоставляет удобные и понятные API, что делает его доступным для разработчиков всех уровней опыта. Благодаря модульной архитектуре React Router, компоненты и маршруты могут быть легко переиспользованы и поддерживаемыми, что способствует быстрой разработке и поддержке приложения.

В заключение, React Router - это мощный инструмент, который значительно облегчает управление маршрутизацией веб-приложений на платформе React. Он предоставляет удобные и гибкие средства для определения маршрутов, создания ссылок и обработки динамических параметров. Благодаря своей простоте использования и интеграции с экосистемой React, React Router является популярным выбором для разработки масштабируемых и гибких веб-приложений.

3.2.2 Redux

Redux - это популярная библиотека управления состоянием для JavaScript-приложений, в том числе для приложений, созданных с использованием фреймворка React. Она предоставляет эффективные средства для организации и управления состоянием приложения, делая его предсказуемым и легко поддерживаемым.

Основной концепцией Redux является единственный источник истины - единое хранилище данных, которое содержит всю информацию, необходимую для работы приложения. Вся информация хранится в виде неизменяемого объекта, который называется ”состояние”. Состояние приложения не может быть изменено напрямую, а только через специальные функции, называемые ”редьюсеры”.

Редьюсеры - это чистые функции, которые принимают текущее состояние и действие, и возвращают новое состояние на основе этих данных. Они определяют, как должно измениться состояние приложения в ответ на определенные действия. Redux обеспечивает простой и предсказуемый поток данных, где состояние всегда обновляется с помощью редьюсеров.

Один из ключевых элементов Redux - это диспетчер, который является

интерфейсом для отправки действий в хранилище. Действия - это простые объекты, которые описывают, что произошло в приложении. Диспетчер принимает действия и передает их в редьюсеры, которые в свою очередь обновляют состояние приложения.

Для удобного использования Redux в приложении, React-компоненты могут подписываться на изменения состояния и получать необходимые им данные из хранилища. Это достигается с помощью специального компонента высшего порядка (Higher-Order Component) или с помощью хука useSelector в React Redux.

Кроме основных концепций, Redux предлагает ряд дополнительных возможностей для упрощения разработки. Например, с помощью middleware можно добавить дополнительные функциональности к обработке действий, такие как асинхронные запросы или логирование. Redux также обеспечивает инструменты для отладки, позволяющие просматривать историю действий и состояния приложения во время разработки.

Одним из главных преимуществ Redux является его способность обрабатывать сложные состояния приложения и поддерживать их расширение в будущем. Он способствует предсказуемости и тестируемости кода, а также упрощает сопровождение и отладку приложения.

В заключение, Redux является мощным инструментом для управления состоянием в JavaScript-приложениях, особенно в сочетании с React. Он предоставляет эффективные средства для организации и обновления состояния, делая приложение более предсказуемым и легко поддерживаемым. Redux позволяет создавать масштабируемые и гибкие приложения, которые легко расширять и изменять в будущем.

3.2.3 Axios

Axios - это популярная библиотека для выполнения HTTP-запросов из JavaScript-приложений. Она предоставляет удобный и гибкий интерфейс для взаимодействия с сервером, обеспечивая простоту в использовании и мощные функциональные возможности.

Одной из главных причин популярности Axios является его простота в настройке и использовании. Для отправки HTTP-запроса с помощью Axios необходимо всего несколько строк кода. Библиотека предоставляет функции для различных типов запросов, таких как GET, POST, PUT, DELETE, и др., а

также для отправки данных в формате JSON, форм-данных или в виде файлов. Это делает процесс взаимодействия с сервером интуитивно понятным и удобным для разработчиков.

Одной из ключевых особенностей Axios является поддержка Promise API. Он возвращает Promise-объект, что позволяет использовать синтаксис async/await для управления асинхронными операциями. Это упрощает обработку ответов от сервера и выполнение последовательных запросов.

Axios также обладает мощными возможностями по обработке ошибок и перехвату запросов. Он предоставляет механизмы для обработки различных HTTP-статусов, таких как успешный ответ, ошибка сервера или отсутствие сетевого подключения. Разработчики могут определить собственные обработчики ошибок и выполнить соответствующие действия в зависимости от ситуации.

Еще одним преимуществом Axios является его способность автоматически преобразовывать данные в различные форматы. Он может автоматически распознавать и парсить данные в формате JSON, XML или FormData. Это позволяет сократить объем кода и упростить процесс обработки данных.

Кроме того, Axios поддерживает возможность создания интерсепторов, которые позволяют изменять и модифицировать запросы и ответы перед их отправкой или обработкой. Это дает возможность добавить дополнительную логику, например, для авторизации, логирования или обработки заголовков.

Axios также интегрируется хорошо с другими библиотеками и фреймворками, такими как React или Vue. Он предоставляет дополнительные функциональные возможности, такие как отмена запросов, отслеживание прогресса загрузки, установка времени ожидания и т.д.

В целом, Axios является мощным инструментом для выполнения HTTP-запросов в JavaScript-приложениях. Он предоставляет простой и понятный интерфейс, гибкие функциональные возможности, а также надежность и стабильность. С помощью Axios разработчики могут легко взаимодействовать с сервером и обрабатывать данные, делая их приложения более эффективными и функциональными.

3.3 Архитектурная методология Feature-Sliced Design

3.3.1 Описание выбранной архитектурной методологии

Рассмотрим архитектурную методологию Feature-Sliced Design. В официальной документации сказано: ”Feature-Sliced Design (FSD) — это архитектурная методология для проектирования Frontend-приложений. Проще говоря, это свод правил и соглашений по организации кода. Главная цель методологии — сделать проект понятным и структурированным, особенно в условиях регулярного изменения требований бизнеса.”[5]

Из этого можно сделать вывод, что данная методолгия акцентирует внимание на организации кода вокруг функциональных возможностей (фич) приложения. FSD стремится к высокой модульности, повторному использованию кода и улучшению поддерживаемости проекта.

Проект делится на 6 слоев (layers), где каждый слой состоит из слайсов (slices) и каждый слайс состоит из сегментов (segments). Рассмотрим подробнее каждый из них:

Слои (layers):

- Shared: Этот слой содержит переиспользуемый код, который не зависит от специфики приложения или бизнес-логики. В нем могут находиться универсальные библиотеки, компоненты пользовательского интерфейса (UI), внешние библиотеки и API.
- Entities: В этом слое находятся бизнес-сущности, такие как пользователи, продукты, заказы и другие основные объекты, которые являются основой функциональности приложения.
- Features: Фичи представляют собой взаимодействия с пользователем, действия, которые приносят бизнес-ценность. Каждая фича включает в себя свою логику, компоненты UI, стейт-менеджмент и взаимодействие с другими слоями.
- Widgets: Этот слой представляет собой композиционный слой, который соединяет сущности и фичи в самостоятельные блоки. Он может содержать компоненты, которые могут быть повторно использованы в разных фичах.
- Pages: Слой страниц используется для сборки полноценных страниц из сущностей, фич и виджетов. Здесь происходит композиция

компонентов и логики для создания конечного пользовательского интерфейса.

- Processes: Этот слой является устаревшим и необязательным. Он содержит сложные сценарии, которые покрывают несколько страниц и включают более сложную логику, такую как авторизация или другие бизнес-процессы.
- App: Этот слой содержит настройки, стили и провайдеры, необходимые для всего приложения.

Слайсы (slices):

Каждый слой разделен на слайсы, которые группируют логически связанные модули. Слайсы помогают организовать код по предметной области и обеспечивают низкую связность и высокое зацепление между модулями. Слайсы не могут использовать другие слайсы на том же слое, что способствует упрощению навигации и пониманию кодовой базы.

Сегменты (segments):

Каждый слайс состоит из сегментов, которые отвечают за технические аспекты функциональности. Некоторые распространенные сегменты включают:

- UI: Содержит компоненты пользовательского интерфейса, отвечающие за отображение данных и взаимодействие с пользователем.
- Model (store, actions): Содержит код для управления состоянием фичи, такой как хранилище данных (store) и действия (actions) для изменения состояния.
- API: Отвечает за взаимодействие с внешними сервисами или API для получения или отправки данных.
- Lib (utils/hooks): Содержит вспомогательные функции, утилиты и пользовательские хуки, которые могут быть использованы внутри фичи.

Представление декомпозиции проекта можно увидеть на рисунке 2

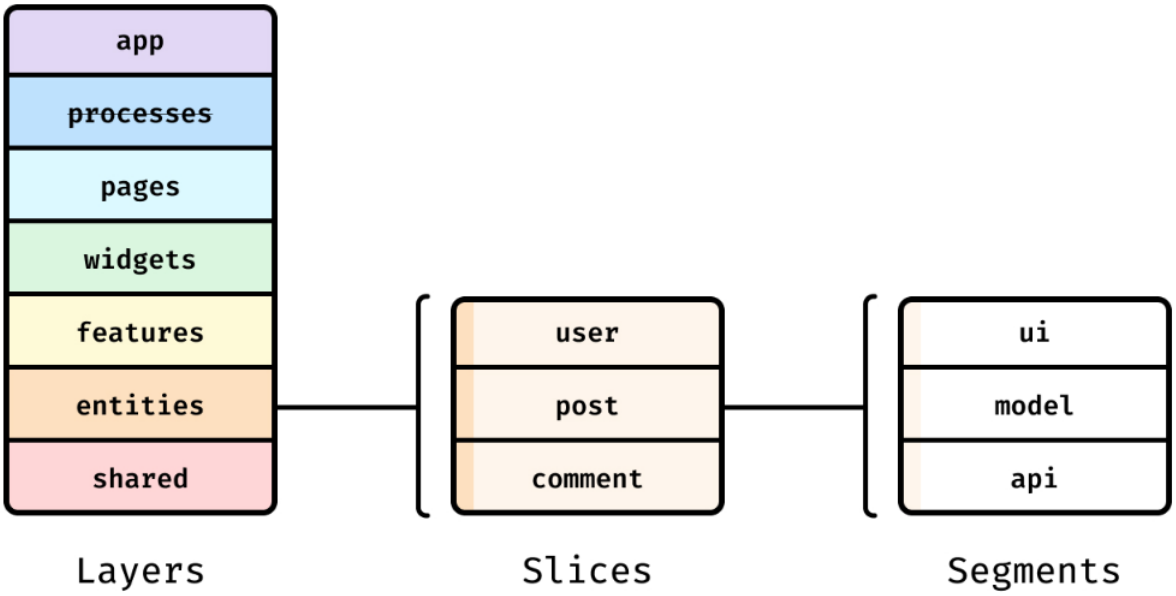

Рисунок 2 – Разделение проекта на слои, слайсы и сегменты

Данная методология обладает рядом преимуществ, которые делают его эффективным подходом для разработки фронтенд-проектов:

- Архитектура FSD сосредоточена на бизнес-функциональности и разделяет код по функциональным возможностям. Это позволяет лучше структурировать и организовать код, делая бизнес-логику явной и понятной для разработчиков.
- Методология обеспечивает гибкость и адаптивность архитектуры. Компоненты могут быть легко заменены или добавлены в соответствии с новыми требованиями или условиями проекта. Это позволяет проекту быть более масштабируемым и способствует более простому внесению изменений.
- Архитектура FSD состоит из доменных модулей, таких как слои, слайсы и сегменты, что делает ее относительно простой для изучения и понимания. Разработчики могут быстро ориентироваться в структуре проекта и находить нужные модули, что способствует более эффективному сотрудничеству в команде.
- Благодаря структурированному подходу FSD, каждый модуль может быть независимо модифицирован или переписан без сайд-эффектов на другие модули. Это помогает управлять техническим долгом проекта и делает его более гибким для будущих изменений и развития.

– FSD находит баланс между принципом ”Don’t Repeat Yourself”(DRY) и локальной кастомизацией. Архитектура позволяет переиспользовать компоненты и модули, сохраняя при этом возможность их локальной настройки и кастомизации для конкретных потребностей проекта.

3.3.2 Применение выбранной методологии в разработке

Разработка по данной методологии начинается с определения и организации слоев. Каждый слой представляет собой логическую группировку модулей с определенными функциональными обязанностями. Слои включают shared, entities, features, widgets, pages, processes и app.

Рассмотрим алгоритм декомпозиции проекта на указанные слои по зонам ответственности. Когда разрабатывается новый модуль, разработчику необходимо определить к какому слою он должен относится. Представленные вопросы помогают отнести нужный модуль к нужному слою:

– Shared: Это не относится к Бизнес-Логике и является общим переиспользуемым служебным кодом?
– Entities: Это относится непосредственно к бизнес-сущности?
– Features: Это относится к действию пользователя, представляющему бизнес-ценность?
– Widgets: Это самостоятельный и полноценный блок страницы с конкрытными действиями?
– Pages: Это относится к конкретной странице/экрану?
– Processes: Это относится к конкретному юзкейсу, протекающему через несколько страниц?
– App: Это общая инициализирующая логика приложения?

Рассмотрим каждый слой, отнеся к нему какой-нибудь модуль:

Shared: здесь располагаются переиспользуемые компоненты, например: кнопки, тексты, иконки, спиннеры, настройки безопасности, инстансы Axios, хуки, хелперы для работы со Store и т.д. Иными словами, компоненты без привязки к бизнес-логике. Пример представлен на рисунке 3

Рисунок 3 – Слой Shared

Entities: например карточка поста в соц-сети, на которой представленны только общие части, специфичные для всех постов. Функциональностей по типу лайков, вспомогательных кнопок, кнопки поделится и т.д., здесь не представлены. Они представлены в виде слотов, куда функциональности добавляются уже на уровне выше. Пример представлен на рисунке 4

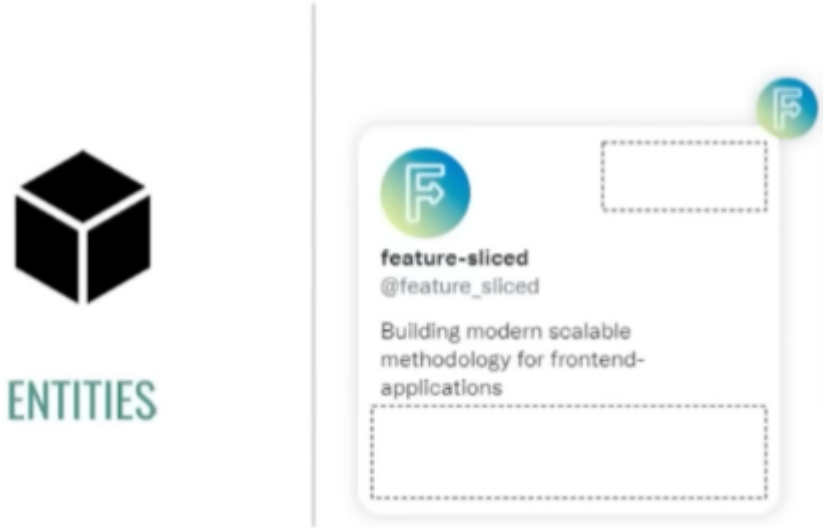

Рисунок 4 – Слой Entities

Features: здесь уже появляются функциональности, которые несут в себе какой-то бизнес-функционал, который приводит к какому-то результату, например: лайки, кнопка подписаться и т.д. Причем одна фича должна решать одну задачу. Пример представлен на рисунке 5

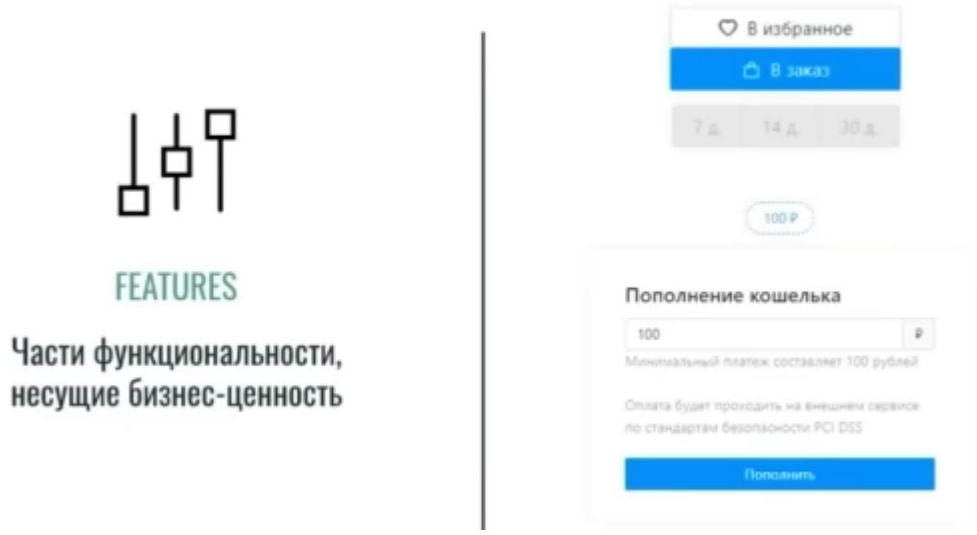

Рисунок 5 – Слой Features

Widgets: это комбинация entities, которые содержат ”пустые слоты”и в эти слоты подставляются необходимые фичи (features). В данном случае, виджеты - это самостоятельные смысловые блоки, комбинирующие нижние слои. Полноценный пост, который можно оставить в группе - это уже виджет, причем один виджет может называть userPost, а другой groupPost, потому что с точки зрения функционала и работы с Backend они скорее всего будут разными. Пример представлен на рисунке 6

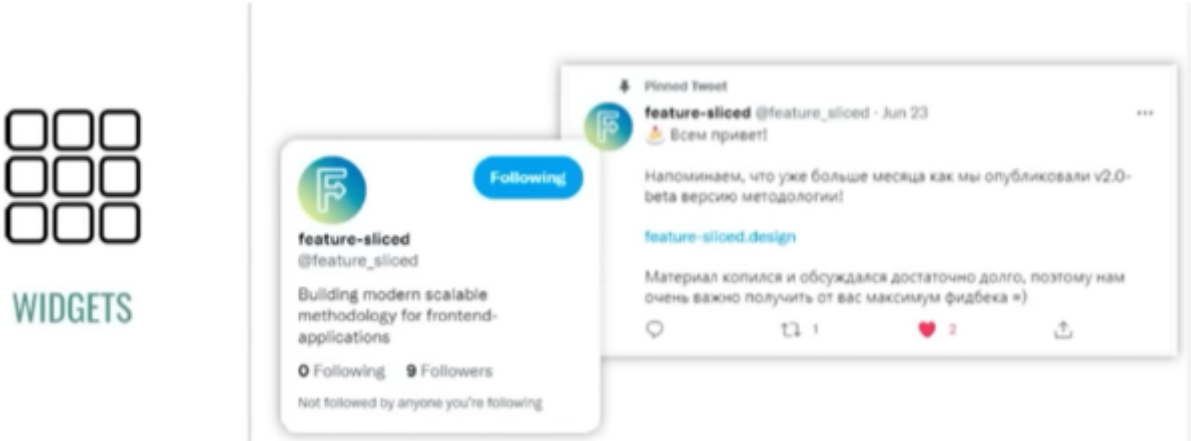

Рисунок 6 – Слой Widgets

Pages: страница собирается непосредственно из виджетов. Непосредственно в данном слои находится минимум бизнес-логики, запросов и прочего, все необходимое выносится на слои ниже. Пример представлен на рисунке 7

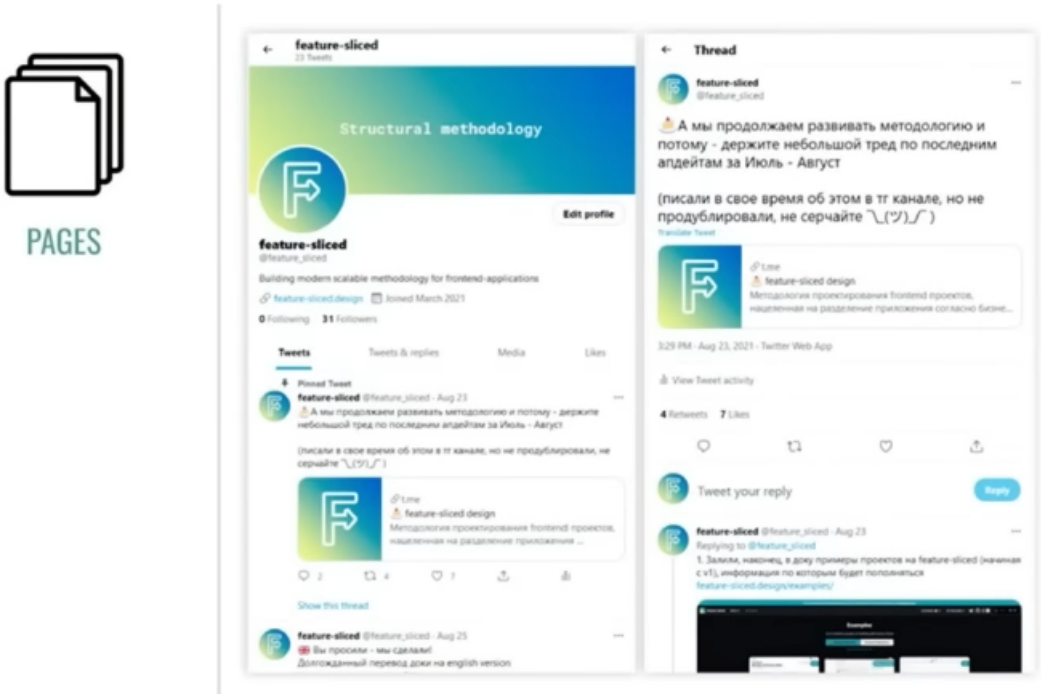

Рисунок 7 – Слой Pages

Processes: в нем хранится какой-то сложный набор действий, который протекает через несколько страниц. Например авторизация в несколько страниц.

App: данный слой хранит в себе инициализирующую логику приложения (Инициализирующая точка входа в приложение). Здесь хранятся все роутинг, провайдеры, глобальные стили, глобальные типы и все что не будет никогда и ни при каких условиях использоваться где-то ниже

4 РЕАЛИЗАЦИЯ КЛИЕНТСКОЙ ЧАСТИ

4.1 Реализация функционала заполнения и сохранения карт вызова

Функционал заполнения и сохранения карт вызова представляет собой важную часть медицинского приложения, которое позволяет медицинским работникам эффективно вести медицинскую документацию и сохранять данные о пациентах. Перед реализацией функционала заполнения, необходимо подробно рассмотреть обратную сторону карты вызова, рисунок А.1 .Для удобства навигации и промежуточного сохранения заполненных полей карта вызова должна быть разделена на несколько разделов.

Первым является раздел - ”Жалобы и Анамнез предназначен для заполнения данных о жалобах пациента и его медицинском анамнезе. Здесь медицинский работник может внести подробности, симптомы и предыдущую медицинскую историю пациента.

Следующий раздел - ”Объективно предназначен для ввода объективных данных, полученных в результате осмотра пациента. Здесь могут быть указаны физические параметры, результаты осмотра органов и систем пациента.

Затем идут разделы, посвященные различным системам органов, такие как ”Органы Дыхания ”Органы кровообращения ”Органы Пищеварения ”Нервная Система ”Костная Система”. В каждом из этих разделов можно внести подробности о соответствующей системе органов пациента, включая симптомы, результаты исследований и т.д.

В разделе ”Status Localis”заполняются основные данные о пациенте, такие как вес, пол, рост, возраст. Эти данные могут быть переданы модулям ”Калькулятор дозировок”и ”Таблицы Дифференциальной диагностики”через инструмент Redux, чтобы обеспечить удобную работу с этими данными в указанных модулях.

В разделе ”Данные инструментальных исследований”может быть реализована функция загрузки фотографии снимка ЭКГ. После загрузки фотографии она может быть отправлена на сервер для обработки модулем ”Острые патологии по снимку ЭКГ”. Это позволит медицинскому персоналу быстро анализировать и диагностировать патологии на основе снимков ЭКГ.

В разделе ”Оказанная помощь, ее эффект и рекомендации”необходимо указать информацию о предоставленной медицинской помощи, ее результаты и рекомендации для пациента.

И, наконец, раздел ”Расходные материалы”предназначен для внесения информации о расходных материалах, использованных при оказании медицинской помощи.

Каждый раздел имеет множество полей, которые заполняются медицинским работником в соответствии с конкретной ситуацией и состоянием пациента. Для обеспечения более быстрого заполнения карты вызова необходимо проанализировать заполненные поля уже заполненных карт и выявить часто встречающиеся ответы. На основе этого анализа необходимо реализовать дополнительный функционал быстрых ответов, который предложит предварительно заполненные часто используемые значения для данных полей, что значительно ускорит процесс заполнения карты вызова. Промежуточный результат разработки заполнения карты вызова можно увидеть на рисунке 8

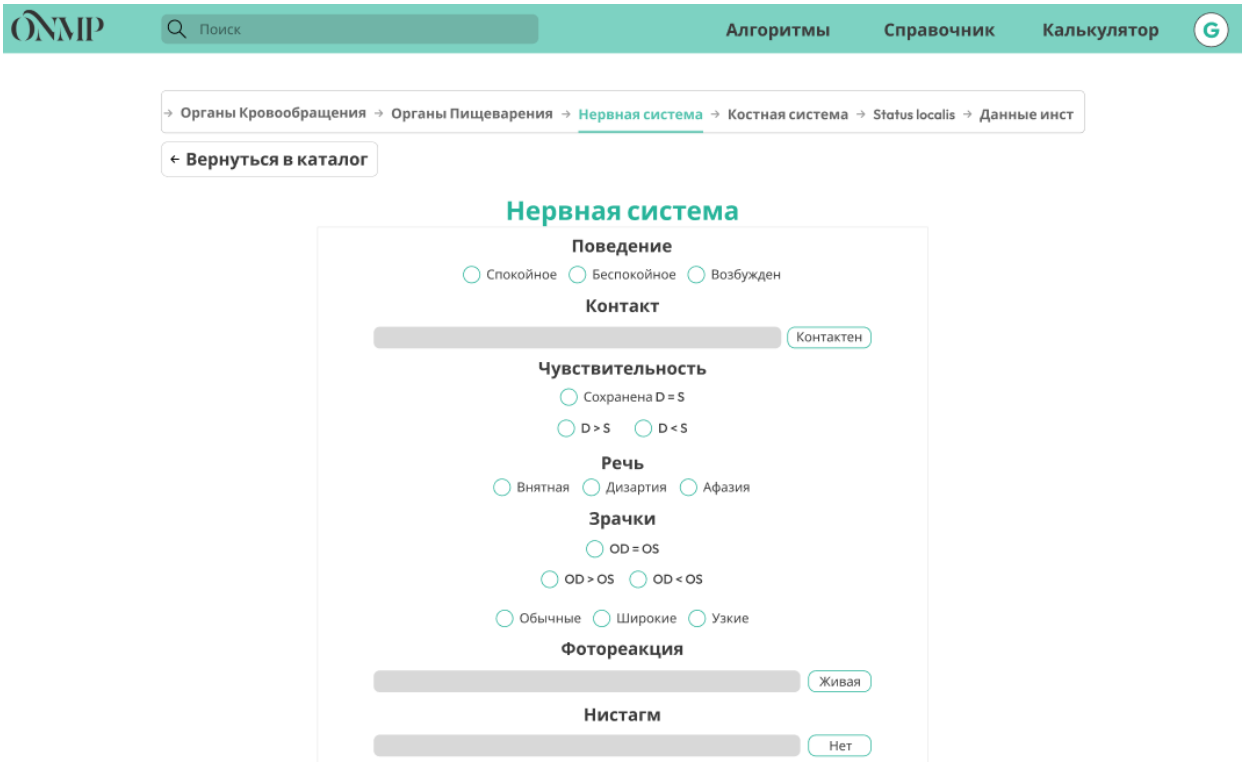

Рисунок 8 – Страница заполнения карты вызова

В результате реализации функционала заполнения и сохранения карт вызова в соответствии с описанными разделами, медицинские работники

смогут эффективно и точно вести медицинскую документацию, сохранять данные о пациентах и облегчить себе работу при последующих визитах или обращениях пациентов.

4.2 Разработка функционала организации папок и использования шаблонов

Функционал организации папок и использования шаблонов для карт вызова обеспечивает максимальную эффективность и удобство использования. Разработка этого функционала включает создание четырех папок: Готовые, Незавершенные, Архив и Шаблоны.

В папку ”Готовые”попадают карты вызова после того, как пользователь успешно заполнил все необходимые поля и нажал кнопку ”Сохранить”. Такая организация позволяет пользователям легко находить готовые карты вызова и быстро получать доступ к информации о каждом вызове.

В папку ”Незавершенные”попадают карты вызова, когда пользователь не завершает их заполнение, например, закрывает вкладку или не нажимает кнопку ”Сохранить”. Однако уже введенные данные сохраняются в базе данных в процессе заполнения, чтобы пользователь мог возобновить работу с того места, где он остановился. Это обеспечивает сохранность введенных данных и позволяет избежать необходимости повторного заполнения всех полей.

В папку ”Архив”перемещаются карты вызова после их распечатки или выполнения других действий, указанных пользователем. Архивирование карт вызова обеспечивает сохранность уже готовых карт, которые могут быть полезными для последующего анализа статистики, отчетов и других целей.

Папка ”Шаблоны”предоставляет возможность создания базовых шаблонов карт вызова с частичным заполнением полей. Пользователи могут создавать шаблоны с предварительно заполненными данными, которые могут быть использованы в будущем как основа для создания новых карт вызова. Это существенно ускоряет процесс заполнения карт, поскольку многие поля уже будут заполнены заранее, и пользователю останется только внести дополнительную информацию. В результате, пользователи смогут более эффективно использовать время при заполнении карт вызова и уменьшить вероятность ошибок. Промежуточный результат разработки списка карт можно увидеть на рисунке 9

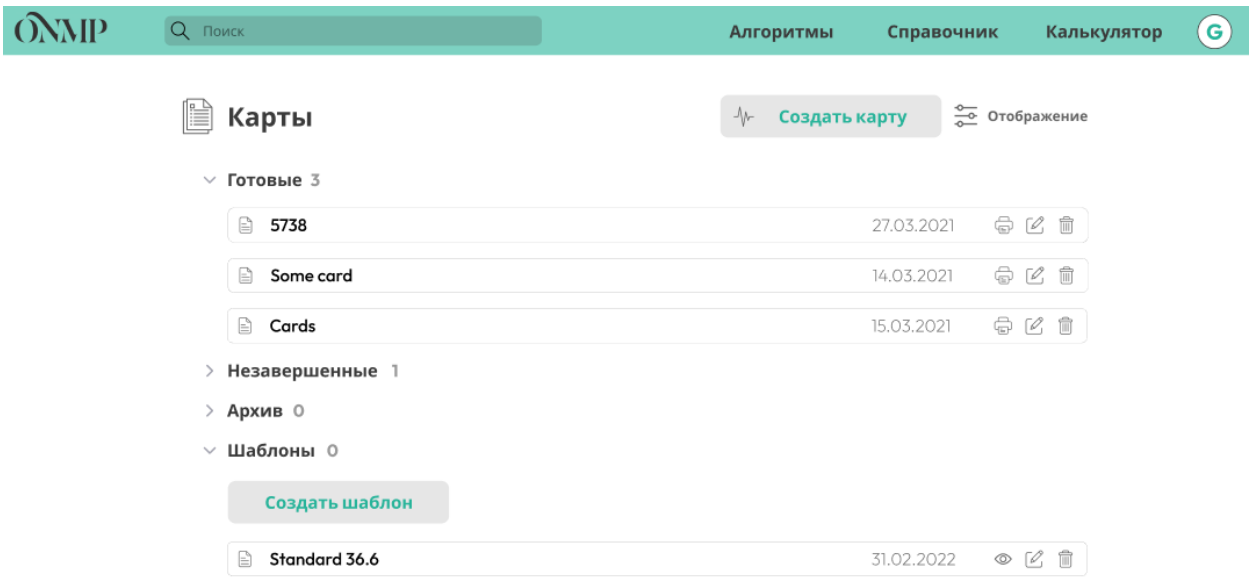

Рисунок 9 – Страница со списком созданных карт

Реализация функционала организации папок и шаблонов обеспечивает структурированный подход к хранению и управлению картами вызова, улучшает доступность и обработку данных, а также повышает эффективность работы с ними. Это существенно упрощает процесс работы медицинского персонала и повышает общую производительность системы.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были достигнуты следующие результаты:

В ходе разработки клиентской части web-сервиса был реализован функционал, позволяющий пользователям заполнять карты вызова с учетом всех необходимых данных и сохранять их в соответствующих папках. Были созданы разделы для удобства навигации внутри карты и промежуточного сохранения заполненных полей. Также был разработан функционал организации папок и использования шаблонов, позволяющий пользователям быстро находить готовые карты вызова, восстанавливать незавершенные и архивировать уже выполненные карты. Внедрение данного функционала значительно упрощает и ускоряет процесс работы медицинского персонала, обеспечивая сохранность и доступность данных.

Оценивая полноту решений поставленных задач, можно сделать вывод о том, что в разработанной клиентской части web-сервиса были учтены основные требования к функциональности и удобству использования. Пользователи получили возможность удобно заполнять и сохранять карты вызова, а также организовывать их хранение с помощью папок и шаблонов. Разделение карт на разные разделы облегчает навигацию и повышает эффективность работы с системой.

В разработанной системе реализован функционал передачи данных модулям ”Калькулятор дозировок”и ”Таблицы Дифференциальной диагностики”на основе заполненных полей ”status localis”. Это обеспечивает более точные расчеты и диагностические рекомендации, основываясь на данных, введенных пользователем.

Рекомендации по использованию результатов работы включают в себя обучение персонала по использованию разработанной системы, создание документации с подробным описанием функционала и инструкциями по его использованию, а также обеспечение технической поддержки для пользователей системы. Дальнейшее развитие системы может включать расширение функционала, добавление новых модулей и интеграцию с другими системами здравоохранения.

Оценка технико-экономической эффективности внедрения разработанной системы показывает потенциал снижения временных затрат на заполнение и сохранение карт вызова. Автоматизация процесса заполнения и

возможность использования шаблонов значительно повышают эффективность работы медицинского персонала. Кроме того, система обеспечивает удобство хранения и поиск готовых карт вызова, что способствует повышению производительности и снижению риска потери информации.

Сравнивая выполненную работу с лучшими достижениями в области, можно сделать вывод, что разработанная клиентская часть web-сервиса обладает современным и удобным пользовательским интерфейсом, а также предоставляет широкий функционал для заполнения, сохранения и организации карт вызова. Она соответствует современным стандартам разработки клиентских приложений и предоставляет удобные инструменты для работы медицинского персонала.

В целом, выполненная работа представляет собой значимый вклад в область разработки систем здравоохранения и обладает высоким уровнем научно-технического достижения. Разработанный функционал позволяет эффективно заполнять и сохранять карты вызова, повышая производительность и качество оказываемой медицинской помощи. Дальнейшее развитие и совершенствование системы могут вносить еще больший вклад в область здравоохранения и улучшать условия работы медицинских специалистов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Banks A.*, *Porcello E.* Learning React: Functional Web Development with React and Redux. — O'Reilly Media, 2017.

2. *Seshadri S.* Angular: Up and Running: Learning Angular, Step by Step. — O'Reilly Media, 2018.

3. *Camden R.*, *Gurney C.*, *Kirkbride P.* Front-End Development Projects with Vue.js: Learn to build scalable web applications and dynamic user interfaces with Vue 2. — Packt Publishing, 2022.

4. *Stefanov S.*, *Вильчинский Н.* React.js. Быстрый старт. — O'Reilly Media (Питер), 2017.

5. *Chelyadinov L.* Feature-Sliced Design - Архитектурная методология для фронтенд проектов. — 2018. — URL: https://feature-sliced.design/ru/docs (дата обращения 25.03.2023).

6. *Департамент здравоохранения города Москвы*. Приказ №372. — 2017. — URL: https://www.medprofsouz.ru/media/userfiles/files/prikaz%20dzm%20ot%2024_05_2017%20%E2%84%96%20372.pdf (дата обращения 14.11.2022).

ПРИЛОЖЕНИЕ А

Карта вызова ОНМПВ и ДН

Представленная карта карта вызова на рисунке А.1 является приложением №6 к приказу Департамента здравоохранения города Москвы от 24 мая 2017 года №372[6].

ЖАЛОБЫ _____

АНАМНЕЗ (в т.ч. – эпид., аллерг., гинекол. по показаниям) _____

ОБЪЕКТИВНО: общее состояние *(удовл., ср. тяжести, тяжелое, терминальное).* Сознание: *ясное, оглушение, сопор, кома - глубина по шк. Глазго.*
Положение *активное, пассивное, вынужденное:* _____
Кожные покровы: *Сухие, влажные, обычной окраски, бледные, гиперемия, цианоз, желтушность* _____
Сыпь_____ Зев_____ Миндалины _____
Лимфоузлы_____ Пролежни_____ Отеки_____ t°C_____
Органы дыхания: ЧДД_____ в мин., одышка *экспираторная, инспираторная, смешанная. Патологическое дыхание*_____
Аускультативно: *везикулярное, жесткое, бронхиальное, пуэрильное, ослаблено, отсутствует в*_____
Хрипы *сухие (свистящие, жужжащие) в* _____
Влажные (мелко-, средне-, крупнопузырчатые) в _____
Крепитация, шум трения плевры над _____
Перкуторный звук легочный, тимпанический, коробочный, притуплённый, тупой над _____
Кашель сухой, влажный, лающий. Мокрота _____
Органы кровообращения: пульс_____ в мин., *ритмичный, аритмичный,* наполнение_____ ЧСС_____ в мин.
дефицит пульса_____ АД_____ привычное_____ максимальное_____ мм рт. ст.
Тоны сердца *звучные, приглушены, глухие.* Шум *систолический, диастолический на* _____
*проводится*_____ *Шум трения перикарда. Акцент_____тона на* _____
Органы пищеварения. Язык *сухой, влажный, обложен* _____
Живот: форма_____ мягкий, напряжен в _____
болезненный в_____. Положительные симптомы *(Образцова, Ровзинга, Ситковского, Ортнера, Мерфи, Мэйо-Робсона, Щеткина-Блюмберга)* _____
Перистальтика_____ Печень_____ Селезенка _____
Рвота (частота)_____ Стул (консистенция, частота) _____
Нервная система: Поведение спокойное, беспокойное, возбужден. Контакт _____
Чувствительность_____ Речь *(внятная, дизартрия, афазия)*_____
Зрачки OD OS, обычные, широкие, узкие. Фотореакция_____ Нистагм _____
*Асимметрия лица*_____. Менингеальные симптомы *(ригидность затылочных мышц, Кернига, Брудзинского)*_____. Очаговые симптомы _____

Координаторные пробы _____
Мочеполовая система _____

Симптом поколачивания _____
Костная система (большой родничок, патологическая подвижность ОДА, крепитация, деформация костей/швов) _____

Status localis _____

Данные инструментальных исследований (ЭКГ, глюкометрия, пульсоксиметрия и пр.)

Оказанная помощь и ее эффект (в т.ч. результаты инстр. иссл. в динамике) _____

Рекомендации: _____

Расходные материалы:_____

Сигнальная карта_____ **Актив поликлиники**_____

Дата и номер наряда_____ **Подпись**_____ **Карту проверил** _____

Рисунок А.1 – Карта вызова ОНМПВ и ДН