



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования

**«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»**

Кафедра «Автоматизированные Системы Управления»

КУРСОВОЙ ПРОЕКТ

по дисциплине:

«Управление жизненным циклом объектов транспортного комплекса»

на тему:

«Создание ИЭТР по управлению сетевым оборудованием и устранению
технических неполадок»

Выполнил:

Учебная группа: 4ВБИТС

ФИО: Евдакимов Н. В.

Руководитель курсового проекта:

ФИО: Юрчик П.Ф.

Подпись _____

«____» _____ 20____ г.

Москва, 2026 г.

Содержание

1. Введение	2
2. Актуальность темы	3
3. Описание предметной области.	5
4. Цель работы	6
5. Задачи для достижения цели работы	8
6. Метод реализации ИЭТР	9
7. Обоснование выбора метода реализации	10
8. Ход работы	12
8.1 Создание интерактивного диалога.	12
8.1.1 Блок-схемы	12
8.1.2 Реализация диалога	14
8.2 Создание документации	18
8.2.1 Известные ошибки	18
8.2.2 Справка	18
8.2.3 Регламентные работы	18
8.3 Настройка Docusaurus	19
8.4 Сборка в Docker	19
8.5 Размещение в интернете	20
8.6 Итоговая проверка	21
9. Заключение	22

1 Введение

При разработке любой современной информационной системы неизбежно возникает потребность в создании документации для конечного пользователя. Для удобства обращения современные документации также представляют собой интерактивные информационные системы, позволяющие оперативно определить проблему и переключить пользователя в необходимый раздел.

Большинство современных Интерактивных Электронных Технических Руководств(ИЭТР) реализуются на стеках веб-технологий, что позволяет обеспечить универсальный и простой доступ, избавившись от сложностей, связанных с версионированием и актуализации данных, а также с необходимостью клиентов в хранении собственных копий документации.

2 Актуальность темы

Современные предприятия и организации невозможно представить без развитой сетевой инфраструктуры. От её стабильной работы зависит доступ к данным, корпоративным сервисам и интернету. Любой сбой в сети мгновенно приводит к финансовым потерям, остановке бизнес-процессов и снижению общих результатов работы. Поэтому поддержка сетевой инфраструктуры становится одной из ключевых задач ИТ-подразделений.

Сети постоянно усложняются. В них входят десятки и сотни устройств от разных производителей, применяются технологии виртуализации и облачные сервисы. Обслуживающему персоналу всё труднее держать в памяти все нюансы настройки и поиска неисправностей. Традиционные бумажные руководства и статичные электронные документы не справляются с этой нагрузкой. В них сложно быстро найти нужную информацию, они быстро устаревают и не дают пошаговых инструкций с учётом реальной конфигурации оборудования.

Возникает потребность в новом инструменте, который объединит всю техническую документацию по сети в единую интерактивную среду. Именно такой инструмент называется интерактивным электронным техническим руководством, или ИЭТР. Разработка ИЭТР для системы поддержки сетевой инфраструктуры решает сразу несколько острых проблем. Снижается время диагностики и устранения аварий, так как специалист получает не просто описание, а логически выстроенный алгоритм действий. Минимизируются ошибки при настройке оборудования, ведь система может содержать интерактивные проверки и подсказки, привязанные к конкретной модели устройства. Упрощается ввод в должность новых сотрудников, поскольку всё знание о сети структурировано и доступно в одном месте.

Актуальность работы также продиктована общемировым трендом на цифровую трансформацию технического обслуживания. Создание ИЭТР

для сетевой инфраструктуры — это шаг к построению единой базы знаний, которая развивается вместе с сетью. Такой проект не только повышает надёжность и скорость восстановления сервисов, но и формирует культуру управления сетевыми активами, что напрямую влияет на конкурентоспособность бизнеса в условиях цифровой экономики.

3 Описание предметной области

Предметная область данной работы охватывает процессы эксплуатации и поддержания работоспособности корпоративной сетевой инфраструктуры. Современная сеть предприятия представляет собой сложный комплекс оборудования, в который входят маршрутизаторы, коммутаторы, серверы и точки беспроводного доступа. Бесперебойная работа всех этих устройств напрямую зависит от скорости реакции обслуживающего персонала на инциденты и от глубины понимания инженерами архитектуры сети. Основной проблемой классического подхода к обслуживанию является разрозненность информации. Техническая документация часто существует в виде бумажных руководств или статичных файлов, которые неудобно использовать прямо на месте аварии. Инженеры теряют время на поиск нужной схемы подключения, инструкции по настройке протокола или алгоритма поиска неисправности, что увеличивает время простоя бизнес-процессов.

В центре этой предметной области находится понятие интерактивного электронного технического руководства, которое приходит на смену пассивным справочникам. Такая система позволяет структурировать гигабайты технических данных и представить их в виде базы знаний с мгновенным доступом. В отличие от простого набора документов, ИЭТР для сетевой инфраструктуры связывает описание методов поддержки с текущей конфигурацией оборудования. Инженер видит не просто текст, а интерактивную логическую схему. Это превращает сложный процесс ремонта в четко регламентированную процедуру, где риск ошибки из-за человеческого фактора сводится к минимуму.

4 Цель работы

Целью данной курсовой работы является разработка интерактивного электронного технического руководства для поддержки сетевой инфраструктуры предприятия. Современная сетевая среда включает множество устройств, сложные топологии и постоянно обновляемое программное обеспечение. Техническая документация при этом часто хранится в разрозненных файлах и папках, что сильно затрудняет оперативный поиск нужных инструкций в критический момент. ИЭТР призвано решить эту проблему, объединив все материалы по сетям, методам их поддержки и настройки в единую интерактивную систему.

Основное назначение руководства — обеспечить системных администраторов и специалистов технической поддержки удобным инструментом для быстрого доступа к актуальной информации. В ходе работы необходимо создать логичную структуру, которая позволит пошагово выполнять настройку оборудования, проводить диагностику неисправностей и устранять типовые проблемы. Важной задачей становится не просто хранение текстовых инструкций, а реализация интерактивных элементов. К ним относятся встроенные проверочные списки, контекстные подсказки и перекрестные ссылки на связанные разделы. Это напрямую сократит время восстановления работоспособности сети и снизит риск ошибок, вызванных человеческим фактором.

Разработка ИЭТР также направлена на стандартизацию процедур обслуживания. Когда все сотрудники следуют единым утверждённым алгоритмам, повышается общая надёжность инфраструктуры. Руководство должно служить и учебной базой для новых работников, помогая им быстрее освоить внутренние регламенты и особенности сетевой архитектуры. Таким образом, конечная цель заключается в создании практичного программного продукта. Он преобразует множество разрозненных документов

в централизованный ресурс технической поддержки, делая процессы эксплуатации и администрирования сети более прозрачными, управляемыми и эффективными.

5 Задачи для достижения цели работы

Для успешного выполнения работы необходимо развернуть статический сайт с реализованной на ней ИЭТР, которая включает в себя необходимые методические указания, регламентные работы, а также интерактивный диалог поиска неисправностей.

Разработка включает в себя следующие пункты:

- Выбор фреймворков, предназначенных для создания сетевой документации
- Разработка интерактивного диалога
- Создание и заполнение информации по регламентным работам, известным проблемам и специфике технологии
- Размещение ресурса в сети "Интернет"

6 Метод реализации ИЭТР

В качестве основного инструмента для создания интерактивного электронного технического руководства был выбран генератор статических сайтов Docusaurus. Это решение позволило сосредоточиться на содержании документации, а не на программировании интерфейса. Docusaurus работает с файлами в формате Markdown, что делает написание и редактирование текстов очень простым и доступным даже для специалистов без глубоких знаний веб-разработки. Весь материал по настройке сетевого оборудования, регламентам обслуживания и типовым процедурам поиска неисправностей был подготовлен именно в таком виде.

Структура сайта формируется из логически связанных папок с документами, что позволило естественным образом разделить описание архитектуры сети, инструкции по первоначальной настройке, справочник команд и руководство по устранению аварий.

Итоговый ресурс предполагается к размещению в качестве веб-ресурса во внутренней сети предприятия. Такой подход гарантирует, что специалисты всегда обращаются к самой актуальной версии документации. Для развёртывания не требуется сложное серверное программное обеспечение, достаточно обычного веб-сервера. Сборка проекта генерирует набор статических файлов, которые очень быстро загружаются и не нагружают канал связи. Это оказалось критически важным, так как в случае частичной деградации сети само руководство должно оставаться легкодоступным и помогать восстановить работоспособность инфраструктуры.

7 Обоснование выбора метода реализации

Выбор в пользу генератора документации Docusaurus был сделан прежде всего из-за его простоты и доступности для инженерного состава. Специалистам по сетям не требуется изучать языки программирования или сложные системы управления контентом, чтобы вести документацию. Все материалы пишутся в легковесном формате Markdown, который интуитивно понятен и близок к обычному тексту. Это позволило максимально быстро наполнить руководство реальными инструкциями по настройке коммутаторов, схемами VLAN и регламентами замены оборудования, не отвлекаясь на технические детали оформления.

В то же время простота фреймворка не ограничила интерактивные возможности, необходимые для полноценного ИЭТР. Поскольку Docusaurus построен на базе React, в любое руководство можно встраивать собственные динамические элементы. Это решение дало возможность создать в рамках документации интерактивные диалоговые подсказки и пошаговые мастера диагностики неисправностей. Например, инженер может прямо на странице пройти опросный алгоритм, который задаст уточняющие вопросы о симптомах сбоя и предложит конкретную процедуру восстановления. Такая функциональность переводит ресурс из разряда статичного справочника в категорию экспертной системы, ведущей диалог с пользователем.

Размещение итогового продукта как веб-ресурса во внутренней сети также повлияло на выбор инструмента. Docusaurus генерирует готовый набор статических файлов, что гарантирует очень быструю загрузку страниц и минимальную нагрузку на каналы связи. Это критически важно, когда к руководству обращаются в момент аварии или нестабильной работы сети. Кроме того, развёртывание не требует серверных баз данных или специального программного обеспечения, достаточно обычного веб-сервера. Сопровождение такого ресурса сводится к простой замене файлов при об-

новлении контента, а централизованное хранение исключает путаницу с версиями инструкций у разных сотрудников. Сочетание лёгкости написания документов, мощной интерактивности на базе React и надёжности статического веб-сайта сделало данный метод реализации оптимальным для создания системы поддержки сетевой инфраструктуры.

8 Ход работы

8.1 Создание интерактивного диалога

8.1.1 Блок-схемы

Ниже приведены блок-схемы, иллюстрирующие общую схему интерактивного диалога.

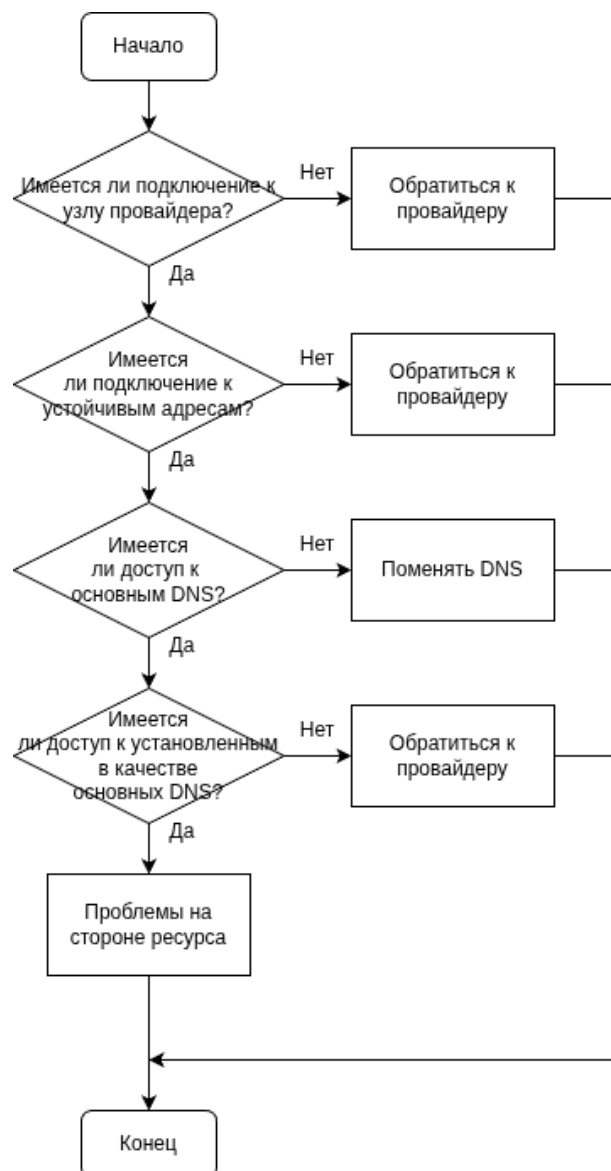


Рис. 1: Ошибки подключения к Интернету при исправном функционировании локальной сети

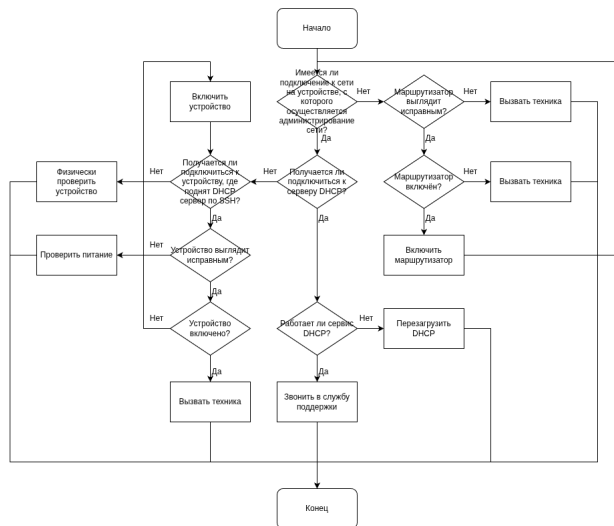


Рис. 2: Ошибки подключения к локальной сети, не локализованные на одном устройстве

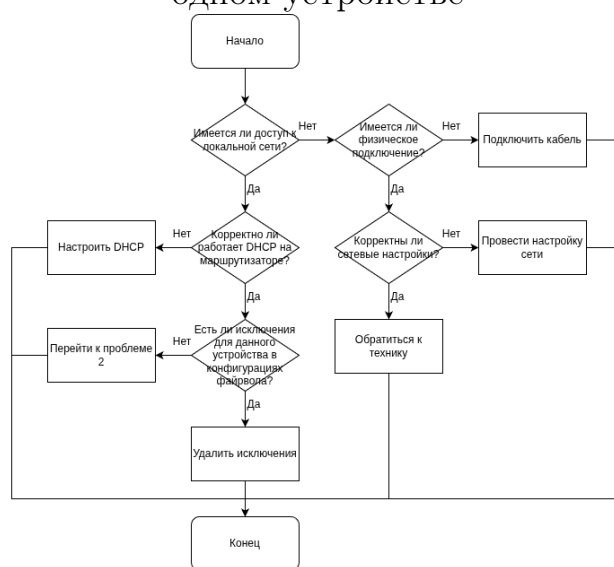


Рис. 3: Ошибки подключения к Интернету при исправном функционировании локальной сети

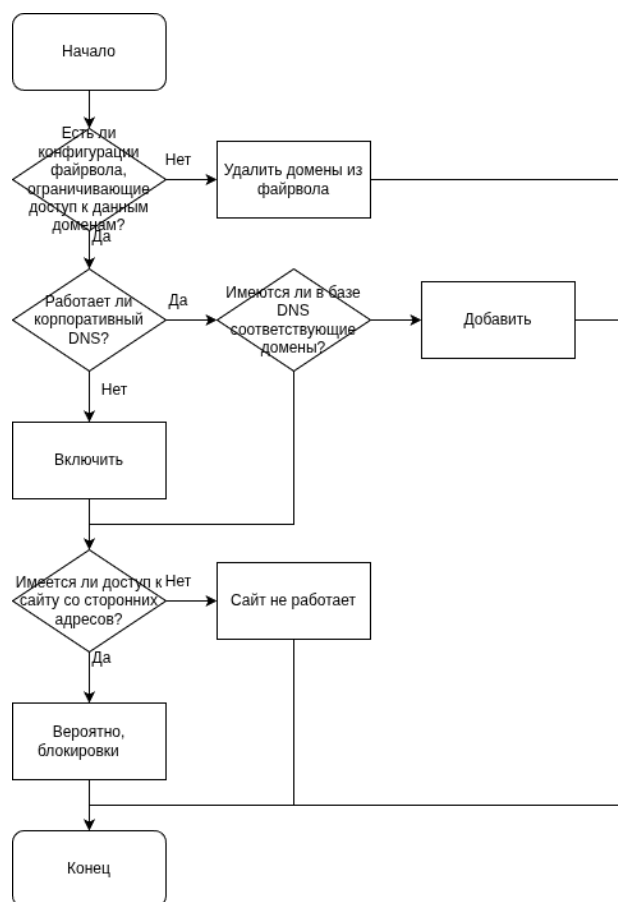


Рис. 4: Ошибки подключения к сайтам при исправной работе интернета

8.1.2 Реализация диалога

Для реализации диалога в системе Docusaurus было решено использовать встроенный функционал и создать небольшое React приложение.

При создании React приложения решено следовать классической парадигме программирования MVC, где роль Модели(Model) играет объект, реализующий паттерн Snapshot, в который сведены основные данные о вариантах развития диалога; роль Отображения(View) берёт на себя сам Docusaurus, предоставляя основные стили и интерфейс для диалога; роль Контроллера(Controller) играет несколько асинхронных функций, обрабатывающих нажатия на кнопки.

Разделение уровня данных и уровня реализации осуществляется посредством выделения содержания объекта Snapshot в отдельный json файл.

Структура модели Модель представляет из себя объект типа "хэш-таблица содержащий в качестве ключа тэг объекта-элемента, а в качестве значения - сам объект-элемент

Объект-элемент представляет собой любой объект, реализующий следующий интерфейс TypeScript:

```
1 interface Option {  
2     options?: Option[];  
3     text?: string;  
4     goto?: string;  
5     alert?: string;  
6 }
```

1. Поле options представляет собой варианты ответа на вопрос - такие же объекты Option
2. Поле text представляет собой текст вопроса или ответа
3. Поле goto представляет собой тэг или ссылку для безусловного перехода к следующему Options или к выходу из диалога
4. Поле alert представляет собой сообщение выводящееся во всплывающем окне. Если alert нулевое, сообщение не выводится

Безусловным слабым местом подобной схемы является смешение содержания вопросов, ответов на них и необходимых действий. Такое смешение требует от разработчика внимательного подхода при составлении схемы, так как поведение программы при попытке перейти из вопроса непосредственно в вопрос не определено - переход от вопроса к вопросу всегда требует промежуточного элемента с goto, так что дочерние элементы Option в options в строгом смысле не являются полноценными, так как не предполагают наличия своих полей options.

Тем не менее, подобный вариант был выбран предпочтительным по причине, что схема с разделением вопросов и ответов потребовала бы ощу-

тимо более алгоритмически сложной реализации Контроллера и привела бы к меньшей гибкости системы за счёт существования минимум двух различных классов с похожим функционалом.

Структура контроллера Контроллер представляет собой React приложение, которое получает и парсит json объект модели в хэш-таблицу с названием `allOptions`.

Контроллер по умолчанию выбирает Option с тэгом `start` и назначает его в поле `currentKey`.

Контроллер рассматривает каждый соответствующий ключу Option и в случае, если он не содержит `goto`, выводит элементы поля `options` в качестве кнопок.

Наиболее существенным элементом контроллера является следующая анонимная функция, вызываемая при нажатии кнопок:

```
1  const handleAnswerClick = (option: Option) => {
2    if (option.alert) {
3      alert(option.alert);
4    }
5    if (option.goto) {
6      if (option.goto.startsWith("/")) {
7        window.location.assign(option.goto);
8        return;
9      }
10     setCurrentKey(option.goto);
11     handleAnswerClick(allOptions[option.goto]);
12   }
13   };
```

Функция выпускает `alert` в случае наличия `и`, в случае наличия `goto`, переходит либо на нужную страницу(начинается с `/`), либо меняет текущий ключ и запускает рекурсивное выполнение.

Очевидными слабостями данной реализации являются два момента:

1. Рекурсия сама по себе

2. Возможность мягкой блокировки(softlock) в случае отсутствия у option полей options и goto

Наличие этих слабостей объясняется тем, что любые возможные потенциальные проблемы связаны с наличием явных ошибок при составлении основного заранее определённого json файла, что по принципу единой ответственности переводит вопрос стабильности программы диалога в этой части к ведению QA и связанных инструментов проверки корректности файлов конфигурации:

1. Рекурсия в сколь-либо реалистичных сценариях может привести к проблеме переполнения стека лишь в случае нахождения цикла безусловного перехода - ситуации, когда существует замкнутая на себя посредством goto цепочка элементов Option.
2. Мягкая блокировка возникает лишь в случае наличия такого Option, который одновременно не содержит как поля options, так и goto.

Обе эти ошибки могут возникнуть исключительно в процессе разработки и могут быть отслежены алгоритмически за полиномиальное время, а потому соответствующие проверки должны быть включены в программу тестирования, а не в рантайм приложения.

Структура отображения Основную реализацию отображения диалога берёт на себя, как это было отмечено ранее, фреймворк Docusaurus. Он подключается за счёт внедрения результата работы React приложения в качестве тела в производный HTML тэг `<Layout></Layout>`. Кнопки имеют стандартный для Docusaurus CSS класс `"button button-secondary button-lg"`

8.2 Создание документации

Документация в рамках Docusaurus ведётся в формате Markdown(*.md) - современном текстовом формате, совмещающим человекочитаемость, характерную для обычных текстовых документов, и набор упрощённых(в сравнении с LaTeX или HTML) тегов форматирования и гипертекстовой разметки. Этот формат надёжно закрепился в нише копирайтинга и поддерживается широким кругом ресурсов, включающим большинство социальных сетей и мессенджеров.

Для сложных задач Markdown поддерживает стандартные HTML теги, так как его спецификация подразумевает явное преобразование Markdown файлов в XHTML.

Также существует расширяющий возможности классического Markdown формат Markdown + JSX(*.mdx), представляющий собой zip архив с характерной структурой директорий и позволяющий инкапсулировать Markdown документы со всеми зависимостями - фото, видео и т.д.

8.2.1 Известные ошибки

Известные ошибки представляют собой конечные точки интерактивного диалога - те страницы, на которые осуществляется переход при разрешении диалога.

8.2.2 Справка

Включает в себя основную информацию по сетям

8.2.3 Регламентные работы

Включает в себя таблицу, содержащую описание основных регламентных работ, и подробную информацию по самим работам.

8.3 Настройка Docusaurus

Docusaurus в основном контролируется через два файла конфигураций в формате JS (*.js): `docusaurus.config.js` и `sidebars.js`.

`docusaurus.config.js` содержит основную часть конфигурации Docusaurus в формате объекта `Config`.

Наиболее существенными для проекта являются следующие поля:

`presets/themeConfig/navbar/items`, содержащий конфигурацию шапки проекта, включая основные разделы.

`presets/themeConfig/footer`, содержащий конфигурации нижнего континула проекта.

`sidebars.js` содержит описание основных разделов сайта, а именно структуру с подобными полями:

```
1 stuffSidebar: [{ type: "autogenerated", dirName: "stuff" }]
```

MD страницы размещаются в директориях `docs` и `blog`, доступ к ним осуществляется по пути относительно базовых директорий без указания расширения

JS/TS страницы, в частности, разработанный интерактивный диалог, размещаются в директории `src/pages`. Там же находится базовая конфигурация сайта.

Домашняя страница находится в `src/components/HomepageFeature`

Все статические файлы(изображения, pdf-документы, архивы и т.д.) находятся в директории `static`

8.4 Сборка в Docker

Для удобства размещения рекомендуется использовать средства контейнеризации вроде Docker, позволяющие минимизировать до разумного минимума зависимость работы приложения от среды.

```
1 FROM node:26-alpine AS builder
```

```

2 WORKDIR /app
3 COPY package.json package-lock.json* yarn.lock* ./
4 RUN if [ -f yarn.lock ]; then yarn install --frozen-lockfile; \
5     elif [ -f package-lock.json ]; then npm ci; \
6     else npm install; fi
7 COPY . .
8 RUN npm run build
9
10 ### STAGE 2 ###
11
12 FROM nginx:stable-alpine
13 COPY --from=builder /app/build /usr/share/nginx/html
14 EXPOSE 80
15 CMD ["nginx", "-g", "daemon off;"]

```

Сборка проекта осуществляется в два этапа: сначала создаётся сборочная среда на базе NodeJS, которая собирает статический сайт, а следом результат сборки перемещается под обычный Nginx.

Сборка и запуск контейнера осуществляются посредством следующих двух команд:

```

1 docker build -t docusaurus-site .
2 docker run -p 8080:80 docusaurus-site

```

8.5 Размещение в интернете

Несмотря на то, что проект предполагается к размещению в локальных сетях, ничто не мешает разместить его в сети "Интернет".

Поскольку данный раздел выходит за рамки настоящего курсового проекта, вопросы, связанные с арендой и настройкой VPS, регистрацией домена и настройкой сервера, опущены, так как в сущности являются стандартными процедурами.

Единственный существенный момент относится к вопросу размещения сайта на одном сервере с другими ресурсами и оформления необходимых TLS сертификатов. Для упрощения решения данной задачи был

сделан выбор в пользу Caddy - современного открытого веб-сервера, который автоматизирует получение и пролонгирование сертификатов Let's Encrypt.

8.6 Итоговая проверка

В результате мы имеем ИЭТР. Ниже приложены снимки экрана

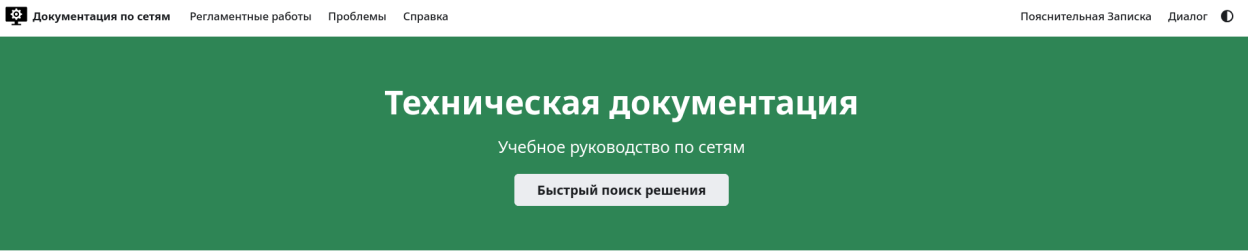


Рис. 5: Стартовая страница

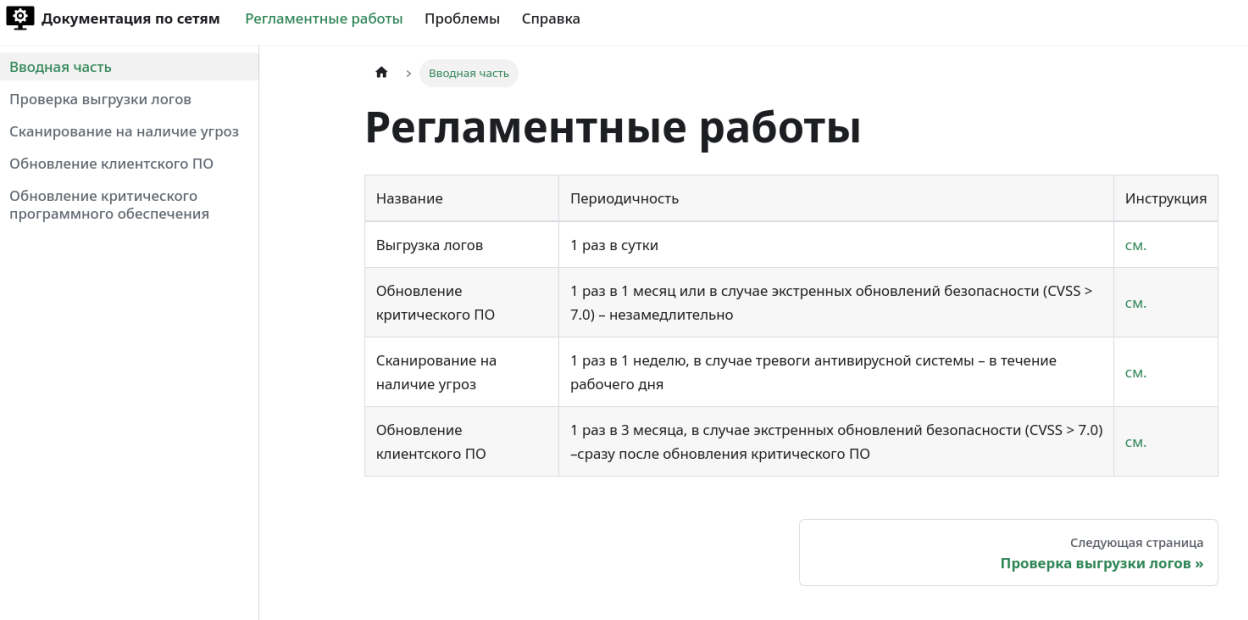


Рис. 6: Регламентные работы

Какой вариант вы выбираете?

Нет интернета везде

Проблемы с локальной сетью

Проблема на одном устройстве

Интернет есть, но сайты не грузятся

Рис. 7: Диалог

9 Заключение

В ходе выполнения курсового проекта была достигнута основная цель - разработана полноценная интерактивная электронная техническая документация для поддержки сетевой инфраструктуры. Полученное решение объединило в себе подробные справочные материалы, практические руководства по настройке сетей и встроенный диалоговый помощник. Благодаря такому подходу пользователь получает не просто статичный набор инструкций, а живую систему, способную в реальном времени подсказывать дальнейшие шаги при решении типовых задач.

Основой для построения документации послужил генератор статических сайтов Docusaurus. Его использование позволило структурировать большой объём технической информации, обеспечить быстрый поиск и удобную навигацию между разделами. Вся документация автоматически адаптируется под любые устройства, что делает её доступной как с рабочего места инженера, так и с мобильного телефона во время выезда на объект. Версионирование материалов даёт возможность отслеживать изменения в регламентах обслуживания сетевого оборудования и при необходимости возвращаться к более ранним редакциям инструкций.

Ключевым интерактивным элементом системы стал диалоговый модуль, написанный на React. Он представляет собой встраиваемый компонент, который реагирует на запросы пользователя и выдаёт актуальные рекомендации, основываясь на содержании документации. С помощью этого диалога инженер может быстро уточнить параметры команды, найти описание неисправности или получить пошаговый план настройки конкретного сервиса. Интеграция помощника напрямую в структуру руководства сокращает время поиска информации и снижает вероятность ошибок при настройке сетевых узлов.

Для удобства развёртывания и переноса система была упакована в

Docker-контейнер. Такой подход исключает зависимость от окружения сервера и гарантирует, что проект запустится одинаково стабильно на любой платформе. Образ контейнера содержит всё необходимое для работы, поэтому внедрение документации в инфраструктуру организации происходит за считанные минуты. Контейнеризация также упрощает масштабирование: при росте числа пользователей достаточно запустить несколько экземпляров сервиса.

Финальным этапом стало размещение готового продукта в сети Интернет. Для обеспечения безопасного доступа был выбран веб-сервер Caddy, который автоматически управляет SSL-сертификатами и настраивает защищённое соединение по протоколу HTTPS. Администратору не требуется вручную продлевать сертификаты или прописывать сложные правила маршрутизации, что заметно снижает порог входа для обслуживания всей системы. Таким образом, документация доступна сотрудникам из любой точки мира, а передаваемые данные остаются надёжно защищёнными.

Практическая значимость выполненной работы состоит в том, что созданный прототип можно быстро адаптировать под нужды конкретного предприятия. Его структура легко расширяется: достаточно дополнить документацию новыми статьями в формате Markdown, и диалоговый помощник сразу начнёт учитывать эти сведения. В перспективе система может развиваться в сторону полноценной базы знаний с элементами искусственного интеллекта для прогнозирования отказов и автоматического формирования планов профилактики сетевого оборудования. Все поставленные задачи были решены, а готовый продукт полностью соответствует современным требованиям к интерактивным электронным техническим руководствам.