

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Рязанский государственный радиотехнический университет имени В.Ф. Уткина
Факультет вычислительной техники
Кафедра электронных вычислительных машин

К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
д.т.н., профессор
_____ Б.В.Костров
« ____ » _____ 2020 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ
РАБОТА БАКАЛАВРА

на тему

«Разработка web-интерфейса SLA-сервера»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

ОПОП: Вычислительные машины, комплексы системы и сети

Студент	_____	(_____)
		(Фамилия И.О.)
Руководитель работы	_____	(_____)
		(Фамилия И.О.)
Руководитель ОПОП	_____	(_____)
		(Фамилия И.О.)

« ____ » _____ 2020 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Рязанский государственный радиотехнический университет имени В.Ф. Уткина
Факультет вычислительной техники
Кафедра электронных вычислительных машин

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
д.т.н., профессор
_____ Б.В.Костров
« ____ » _____ 2020 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра

Студенту Панину Павлу Геннадьевичу, группа 645
(фамилия, имя, отчество полностью, номер группы)

1. Тема работы: Разработка web-интерфейса SLA-сервера

2.Сроки сдачи студентом законченной работы __ июня 2020 г.

3. Руководитель работы Чичикин Вячеслав Алексеевич, к.т.н., доцент кафедры ЭВМ
РГРТУ имени В.Ф. Уткина
(фамилия, имя, отчество полностью, место работы, должность)

4.Исходные данные к работе ОС Windows 7 и выше, ОП не менее 4 ГБ, тактовая частота процессора не менее 2 GHz

5.Содержание расчетно-пояснительной записки (технико-экономическое обоснование темы, расчетная, экспериментальная часть и др. с расшифровкой задания по каждой части)
Введение

1. Постановка задачи

1.1 Описание предметной области

1.2 Определение этапов разработки

2. Технико-экономическое обоснование темы

2.1 Описание проблемы

2.2 Назначение разрабатываемого ПО

2.3 Характеристика основных функций и задач разрабатываемого ПО

2.4 Обзор существующих программных средств

3. Теоретическая часть

3.1 Системный анализ предметной области

3.2 Выбор средств разработки

4. Проектная часть

4.1 Проектирование пользовательского интерфейса

4.1.1 Разработка дерева форм

4.1.2 Разработка прототипа пользовательского интерфейса

4.2 Организация взаимодействия с сервером

4.3 Разработка пользовательского интерфейса

5. Разработка программной документации
5.1 Руководство системного программиста
5.2 Руководство пользователя
6. Тестирование программного обеспечения
Заключение
Список использованной литературы
Приложение А. Листинги основных программных модулей
Приложение Б. Основные методы HTTP API используемого сервера

6. Перечень графического материала (с точным указанием обязательных слайдов)
6.1 Описание проблемы – 3 слайда
6.2 Постановка задачи – 1 слайд
6.3 Обзор аналогичных средств – 1 слайд
6.4 Интерфейс пользователя – 4 слайда
6.5 Подведение итогов – 1 слайд
7. Консультант по работе _____

Дата выдачи задания «__» _____ 2020 г.

Руководитель работы _____

Задание принял к исполнению «__» _____ 2020 г.

Подпись студента _____

Примечания: 1. Задание оформляется на одном листе в двух экземплярах.
2. Лист задания включается в законченную работу после титульного листа и вместе с работой представляется в ГЭК.

РЕФЕРАТ

Данная выпускная квалификационная работа посвящена разработке WEB-интерфейса для работы со статистическими техническими данными в сфере сетевых технологий.

Пояснительная записка объёмом 112 страниц состоит из 6 частей, содержит 66 рисунков, 1 таблицу, 2 приложения.

В процессе выполнения работы был спроектирован, разработан и протестирован WEB-интерфейс, предназначенный для работы с внешним SLA-агентом в виде работающего сервера.

Область применения результативного ПО включает в себя администрирование сетей, а также сферу сетевых технологий.

В приложении А содержатся исходные коды основных разработанных программных элементов, в приложении Б – некоторые основные HTTP методы для работы с API сервера.

ABSTRACT

This final qualification work is devoted to the development of a WEB-interface for working with statistical technical data in the field of network technologies.

The explanatory note of 112 pages consists of 6 parts, contains 66 pictures, 1 table, 2 appendices.

In the course of the work, a WEB interface designed to work with an external SLA agent in the form of a working server was designed, developed and tested.

The scope of effective software includes network administration, as well as the field of network technologies.

Appendix A contains the source codes of the main developed software elements, and Appendix B contains some basic HTTP methods for working with the server API.

Содержание

ВВЕДЕНИЕ.....	8
1 Постановка задачи.....	9
1.1 Описание предметной области	9
1.2 Определение этапов разработки	11
2 Техничко-экономическое обоснование темы	13
2.1 Описание проблемы	13
2.2 Назначение разрабатываемого ПО	13
2.3 Характеристика основных функций и задач разрабатываемого ПО	14
2.4 Обзор существующих программных средств	14
3 Теоретическая часть	23
3.1 Системный анализ предметной области	24
3.2 Выбор средств разработки и языков программирования.....	26
4 Проектная часть	31
4.1 Проектирование пользовательского интерфейса.....	31
4.1.1 Разработка дерева форм	31
4.1.2 Разработка прототипа пользовательского интерфейса.....	34
4.2 Организация взаимодействия с сервером.....	38
4.3 Описание технологии разработки клиентской части веб-приложений с использованием фреймворка «Angular»	39
4.4 Разработка пользовательского интерфейса	41
4.4.1 Корневой модуль приложения	41
4.4.2 Компонент «Корневой».....	42
4.4.3 Компонент «Панель инструментов».....	42
4.4.4 Компонент «Подвал».....	45

4.4.5	Компонент «Авторизация»	46
4.4.6	Компонент «Основная информация»	49
4.4.7	Компонент «Поиск»	50
4.4.8	Компонент «Диалог с результатами»	54
4.4.9	Сервис «Общий»	56
4.4.10	Реализация мобильной версии	56
5	Разработка программной документации	58
5.1	Руководство системного программиста	58
5.1.1	Общие сведения о программе	58
5.1.2	Структура программы	58
5.1.3	Настройка программы	58
5.1.4	Проверка программы	58
5.1.5	Дополнительные возможности программы	59
5.1.6	Сообщения системному программисту	59
5.2	Руководство пользователя	59
5.2.1	Назначение и условия применения программы	59
5.2.2	Обращение к программе для запуска	60
5.2.3	Входные и выходные данные	60
5.2.4	Работа с программой	60
5.2.5	Сообщения оператору	68
6	Тестирование программного обеспечения	71
	ЗАКЛЮЧЕНИЕ	75
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	76
	ПРИЛОЖЕНИЕ А. ИСХОДНЫЕ КОДЫ ПРОГРАММНЫХ ЭЛЕМЕНТОВ... ..	77
	ПРИЛОЖЕНИЕ Б. ОСНОВНЫЕ МЕТОДЫ HTTP API СЕРВЕРА	111

ВВЕДЕНИЕ

В современном доме ни одна комната не обходится без устройства, подключенного к сети Internet. Разнообразие этих устройств велико: от систем домашней автоматизации быта до промышленного сетевого оборудования. Данные устройства были созданы ради обеспечения пользователям комфорта, безопасности и надёжности.

Однако не всё так просто. Для полноценного и бесперебойного функционирования систем связанных между собой устройств требуется не только грамотная первоначальная настройка каждого узла, но и периодическая диагностика сети.

Также в наши дни огромную долю рынка занимают услуги, обходя по популярности товары. И обусловлено это тем, что каждой купленной единице техники необходима поддержка, и если компания-изготовитель не предоставляет необходимые покупателю услуги, клиент вынужден искать на рынке провайдера требуемых услуг.

Именно поэтому задача предоставления клиенту не только качественного оборудования, но и достойного уровня оказываемых услуг является первоочерёдной для компаний-изготовителей всевозможного электронного оборудования - будь то гарантия на новый смартфон, подписка на платное ПО на новом ноутбуке, или же расширенная страховка на автомобиль.

Сфера рынка, затрагивающая отрасль различного сетевого оборудования, так же подчиняется законам мирового рынка, как и любая другая сфера. В данной работе будет поддаваться анализу именно сфера сетевых технологий оборудования и программного обеспечения, предоставляемых на данный момент ведущими мировыми лидерами.

1 Постановка задачи

1.1 Описание предметной области

В настоящее время каждый интернет-провайдер имеет возможность мониторинга линии связи до конечного устройства клиента. Другими словами, если клиент обратится с проблемой отсутствия интернета, оператор может дистанционно проверить, на каком именно участке линии произошёл обрыв, и только после этого отправить мастера на устранение проблемы.

В процессе совершенствования компании и борьбы за место на рынке, когда обеспечения качества сервиса до оборудования клиента уже недостаточно, провайдер ищет решение, которое позволило бы ему проверять состояние не только собственного оборудования доступа, но и оборудования, установленного у клиента. Ведь, по большому счету, не важно, в каком месте линии до ноутбука или телефона абонента проблема – важно есть ли у него Интернет, а если нет – провайдеру нужно как можно раньше узнать об этом и как можно быстрее решить вопрос.

В целях минимизации затрат на выезд мастеров к клиентам и оптимизации решения проблем прогрессивные компании находятся в поисках решений, позволяющих проводить диагностику локальных сетей дистанционно, и одним из этих решений является SLA-агент.

SLA-агент – механизм диагностики состояния сети на стороне конечного пользователя. Его задача заключается в периодической отправке статистических данных, собранных устройством с системных счетчиков, а также результатов проверки доступности заранее заданных узлов утилитами Ping и Traceroute.

После внедрения в сеть провайдера подобной системы компания получает инструмент, позволяющий удаленно и быстро решать различные проблемы, возникающие на роутерах пользователей. Например, теперь при обращении в техническую поддержку, оператор может незамедлительно получить информацию о состоянии роутера клиента, загруженности частотного диапазона Wi-Fi, проблемах с кабелем, соединяющим роутер и ПК клиента. По статистике, очень много жалоб

абонентов, оказывается, не вызваны проблемами в сети, а связаны с домашними условиями – или медленная скорость по Wi-Fi из-за совпадения частот с соседями, или роутер неправильно настроен, или неправильно обжат кабель.

Без системы мониторинга SLA в роутерах провайдер может провести только диагностику "последней мили" клиента, проверить целостность кабеля от своего оборудования доступа до CPE (роутера клиента) и оборудование своей сети. С внедрением агента зона возможностей расширяется и на роутер клиента. Больше нет необходимости высылать специалиста к клиенту домой для проверки работоспособности роутера и поиска причин возникновения проблем. Можно провести диагностику удаленно и сразу же, при обращении клиента, после чего выдать рекомендации по устранению или произвести ремонт мастером.

Благодаря подробной статистике провайдер может установить наличие проблем с ПК, ноутбуком или мобильными устройствами клиента. Например, по аномально завышенным показателям счетчикам broadcast или multicast кадров оператор может предположить, что на ПК клиента есть вирусы и порекомендовать ему провести проверку.

Собранные системой данные могут быть представлены за определенные временные отрезки, что позволяет проанализировать их и увидеть глобальные проблемы с сервисами, причиной которых могут стать как неверная конфигурация сети провайдера, так и нововведения или изменения в сети.

Более того, интернет-провайдер способен увидеть проблему и решить её до того, как с ней столкнутся его пользователи и обратятся в техническую поддержку.

Подводя итоги краткого описания преимуществ мониторинга и опираясь на статистику некоторых компаний, можно отметить, что система SLA на самом деле является удобной, позволяет экономить ресурсы и улучшить качество сервисов, предоставляемых сетевыми провайдерами.

1.2 Определение этапов разработки

Перед началом работы необходимо обозначить основные этапы разработки программного обеспечения:

1. Системный анализ предметной области. Достаточно глубокий анализ области применения программного средства позволит заранее сформировать требования к разрабатываемой программной среде, опираясь на недостатки и преимущества конкурентов на мировом рынке.
2. Определение назначения разрабатываемого программного обеспечения. Формирование списка предполагаемых пользователей и определение основных функциональных особенностей позволит наиболее точно составить требования к графическому интерфейсу.
3. Выбор средств для разработки. Стек web разработчика – важный набор инструментов, средств и утилит, от комбинации и степени освоения которых зависит качество разработанного ПО.
4. Проектирование пользовательского интерфейса. Первый шаг на пути к созданию пользовательского интерфейса – создание его графического прототипа с поведением, а также дерева форм. Это позволяет избежать неоднозначных разногласий в дальнейших этапах разработки.
5. Организация взаимодействия пользовательского интерфейса с сервером (front end и back end). «Нарисованному» пользовательскому интерфейсу необходимо добавить поведение с помощью языков программирования и реализовать связь пользователя с сервером.
6. Разработка пользовательского интерфейса. Необходимо отобразить спроектированный web интерфейс средствами выбранного стека технологий в web-браузере.
7. Разработка программной документации. Этот этап подразумевает разработку двух видов документации: руководства системного программиста и руководства пользователя.
8. Тестирование программного обеспечения. На данном этапе производится тестирование программы с целью проверки корректности

её работы с ввода разного вида данных, а также корректных получения, обработки и отображения необходимых данных с сервера.

2 Технико-экономическое обоснование темы

2.1 Описание проблемы

Рассмотрим пример с обычным среднестатистическим пользователем домашнего интернета и провайдером. При первом подключении клиента к сети провайдера мастер производит первоначальную настройку и, удостоверившись в работоспособности домашней сети и наличии стабильного соединения с глобальной сетью, покидает клиента, завершая свою работу. Первоначальная настройка выполнена, но в случае внезапного обрыва соединения с сетью клиент будет вынужден снова вызывать мастера для диагностики и устранения неполадок. В данной ситуации возникает вопрос о возможности дистанционной диагностики, так как если проблема на стороне клиента не является серьёзной, отправлять мастера к клиенту становится нецелесообразно по ряду причин.

Проблема дистанционной диагностики состояния локальных сетей в настоящее время становится всё более актуальной благодаря положительному росту как и количества сетевых узлов, так и конечных пользователей.

2.2 Назначение разрабатываемого ПО

Главное назначение данного программного обеспечения – улучшение качества сервиса обслуживания, а именно:

- максимальное сокращение времени между обращением клиента в техподдержку и решением проблемы;
- минимизация затрат всевозможных ресурсов на поиск необходимой информации на сервере.

В первую очередь разрабатываемое программное обеспечение предназначено для использования лицами, чья профессиональная деятельность включает в себя сетевой мониторинг. Этими лицами могут быть:

- системные администраторы;
- специалисты по сетевому мониторингу;
- иные лица, ответственные за работу сети в организации;

- пользователи, которым предоставлен доступ к мониторингу сети.

2.3 Характеристика основных функций и задач разрабатываемого ПО

Так как web-интерфейс предназначен для взаимодействия пользователя и сервера, ПО должно способствовать быстрому поиску необходимой информации на сервере и корректному отображению на экране пользователя, а именно двум основным аспектам:

- обработке и отправке введенных пользователем данных на сервер;
- обработке и отображению полученных с сервера данных.

Разрабатываемое программное обеспечение должно работать с такой информацией, как:

- техническая информация об устройствах, находящихся в системе;
- общее количество устройств в системе, их вид и параметры;
- отчёты о техническом состоянии обзореваемых устройств в разрезе времени;
- информация о пользователе, данные для его идентификации и авторизации.

2.4 Обзор существующих программных средств

На данный момент существует два варианта получения доступа к SLA администрированию: программные агенты могут быть встроены в сетевое оборудование заводом-изготовителем или приобретаться и устанавливаться отдельно.

Cisco IP SLA Monitor.

Cisco Systems в течение многих лет остается безоговорочным лидером в сегменте сетевого оборудования. Опрос показал, что пользователи отметили качество ее продукции, сервисного обслуживания клиентов и техподдержки партнеров.

Программный агент IP SLA (рисунок 2.1), встроенный в Cisco IOS маршрутизаторов Cisco Systems дает возможность измерять качество IP соединения в привязке к работе бизнес-критичных приложений, таких как VoIP, видеоконференцсвязь и критичные к задержкам данные. Cisco IP SLA Monitor (Service Assurance Agent) использует активный метод контроля. Генерируя тестовый трафик IP SLA Monitor обеспечивает измерение показателей производительности и качества сети. Маршрутизатор на одной стороне канала генерирует трафик с заданными параметрами, на второй стороне канала маршрутизатор выступает в роли ответчика.

Данное программное решение от компании Cisco обладает массой преимуществ, расширенной документацией на разных языках и оперативной поддержкой. Минус у данного решения один – высокая цена продукции.

Cisco занимает лидирующую позицию на мировом рынке и ведёт беспрестанную борьбу за удержание своего места среди конкурентов. Другими словами, за качество и грамотную поддержку интернационального масштаба приходится сильно переплачивать.

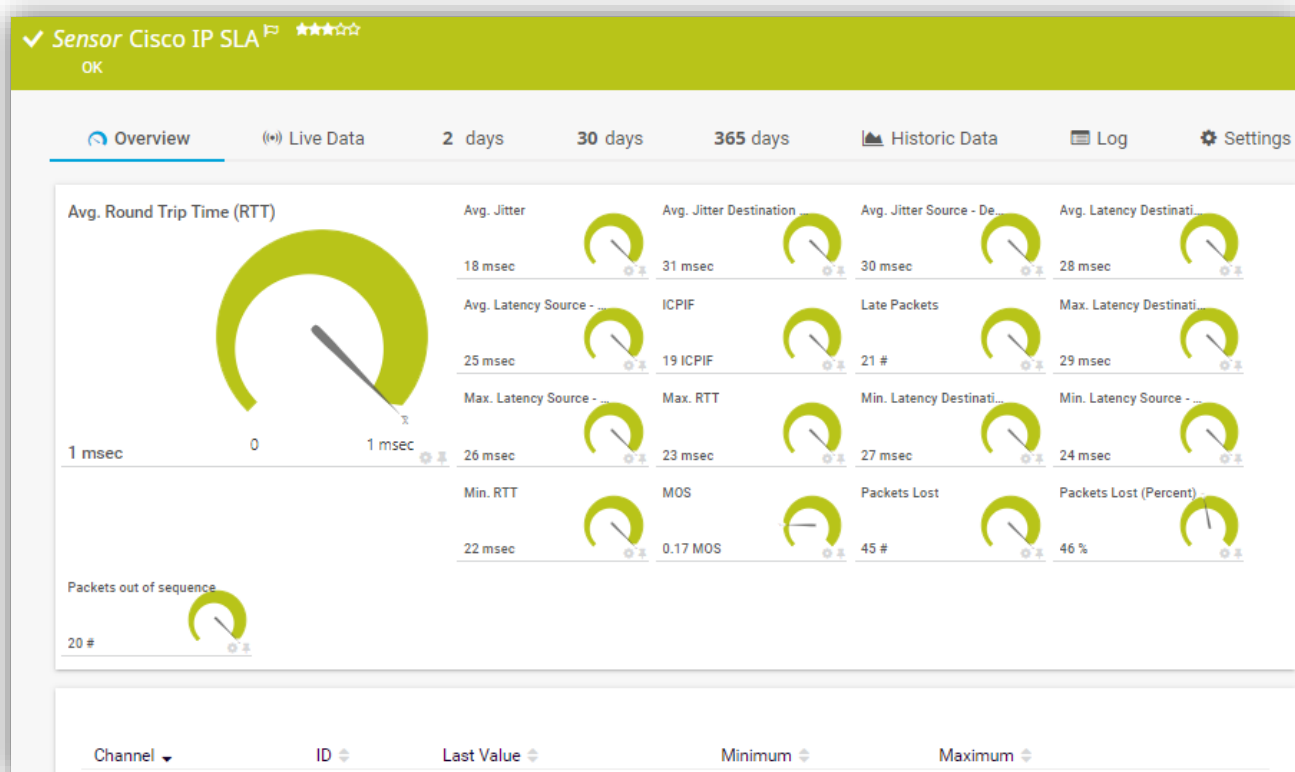


Рисунок 2.1 – Пользовательский интерфейс Cisco IP SPA Sensor

D-Link SLA-system.

Эффективное сетевое управление в оборудовании D-Link представлено как и встроенными системами программных средств, так и приобретаемыми опционально. При необходимости интернет-провайдеры могут заказать кастомизацию приобретаемых роутеров, в том числе размещение логотипа на корпусе устройства, индивидуальный дизайн упаковки, разработку встроенного ПО с учетом специфических требований заказчика или добавление необходимого функционала (например, SLA-агента для мониторинга работоспособности клиентских сетей). Подробная информация о программе кастомизации размещена на официальном сайте компании.

Для выполнения Соглашения об уровне качества обслуживания SLA (Service Level Agreement), провайдерам необходимо стремиться к сокращению среднего времени восстановления работоспособности устройства (Mean Time to Repair - MTTR) и повышению доступности услуг. Функционал Ethernet OAM способствует

решению этих проблем и позволяет провайдерам обеспечить наилучшее качество предоставляемых услуг. Коммутаторы передовых серий этой компании поддерживают стандартизированные функции OAM, включая IEEE 802.3ah, IEEE802.1ag и ITU-T Y.1731. Connectivity Fault Management (CFM) предоставляет функции наблюдения, поиска и устранения неисправностей в сетях Ethernet, позволяя контролировать соединение, изолировать проблемные участки сети и идентифицировать клиентов, к которым применялись ограничения в сети.

Компания D-Link является ведущим мировым производителем сетевого оборудования, предлагающим широкий набор решений для создания локальных сетей, построения беспроводных сетей и организации широкополосного доступа, передачи изображений и голоса по IP (VoIP). В 2012 году компания открыла в Российской Федерации собственное производство, сертифицированное в соответствии с требованиями ГОСТ. В Российской Федерации во многих городах открыты офисы компании и учебные центры D-Link.

Так как компания занимает лидирующие позиции в производстве сетевого оборудования именно потребительского класса и устройств для «умного дома», ценовая политика характеризуется средним значением цен на рынке. Среднестатистический пользователь желает получить продукт надлежащего качества по приемлемой цене и не имеет завышенных требований. Поэтому целевым потребителем данной компании по большей части является обычный рядовой пользователь домашнего интернета, который не имеет желания переплачивать за сверхвысокое качество и излишнюю надёжность.

Существующий SLA-агент от компании D-Link соответствует всем современным техническим требованиям, но разработан с использованием устаревших фреймворков и библиотек (стека технологий).

Пример одной из версий пользовательского интерфейса представлен на рисунке 2.2:

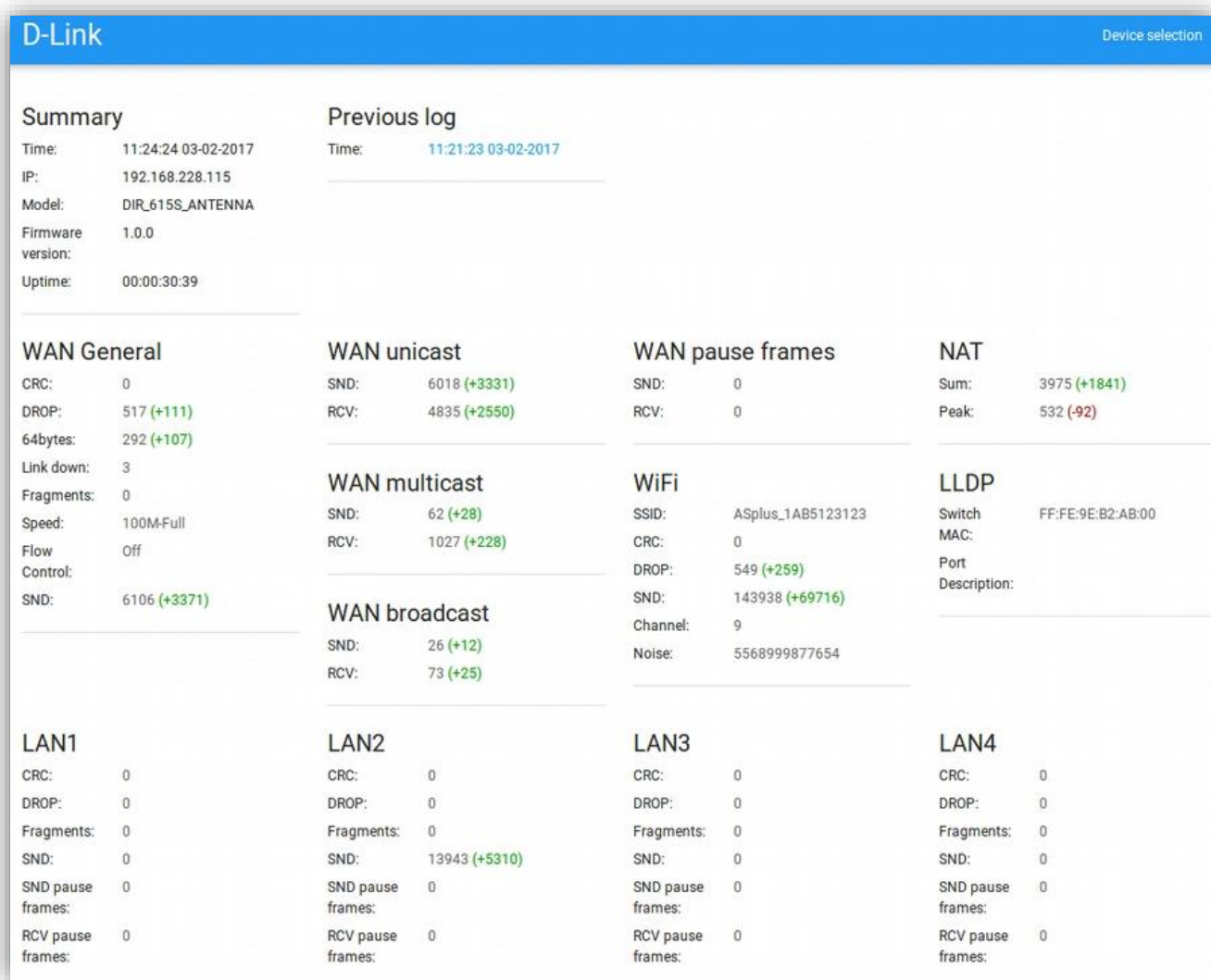


Рисунок 2.2 – Пример пользовательского интерфейса системы мониторинга D-Link SLA-system

PRTG Network Monitor.

PRTG – программное решение от компании Paessler AG, приобретаемое отдельно, оно не поставляется вместе с оборудованием и может быть интегрировано практически в любые системы.

PRTG проводит мониторинг выбранной IT инфраструктуры безостановочно и оповещает оператора о проблемах ещё до того, как пользователь с ними столкнётся.

Программный интерфейс представлен на рисунке 2.3 и рисунке 2.4



Рисунок 2.3 – Визуальный интерфейс PRTG Network Monitor

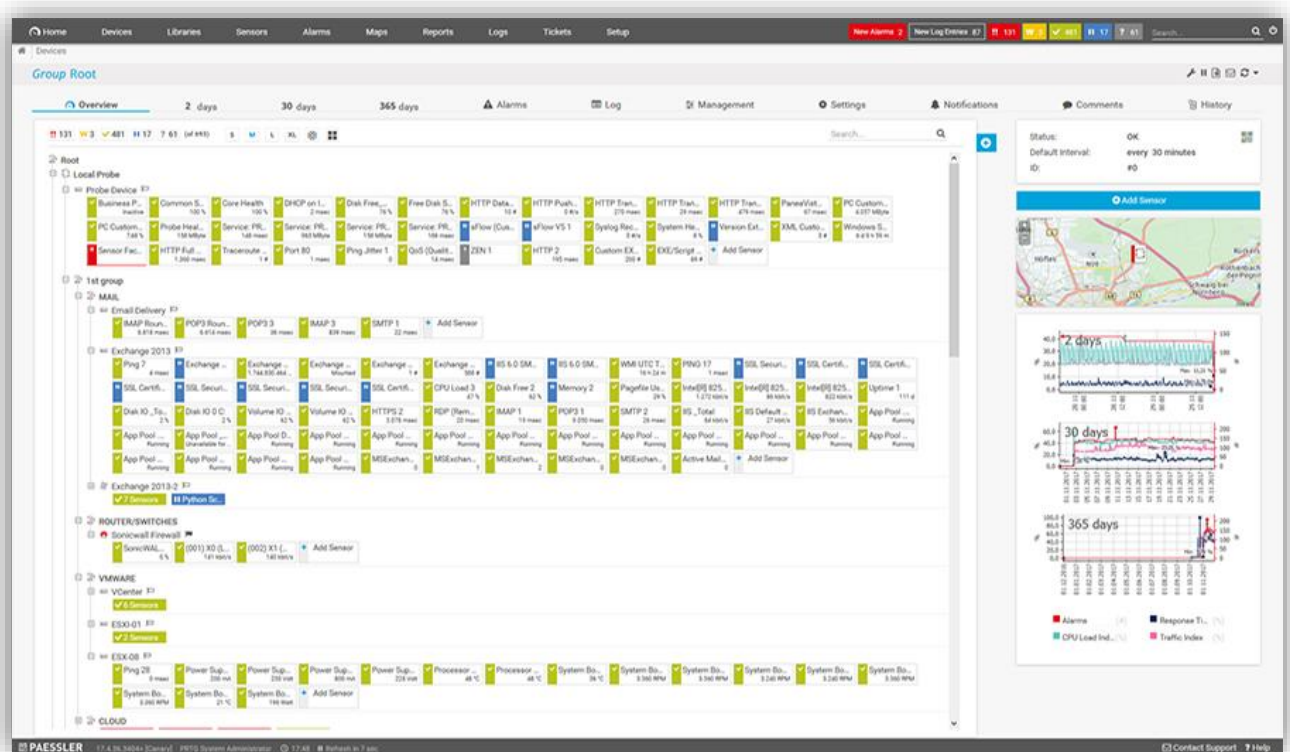


Рисунок 2.4 – Просмотр дерева групп в PRTG Network Monitor

Данное программное решение не отличается в техническом плане от своих аналогов, однако имеет гибкую систему подстройки под нужды заказчика. Цена приобретаемого решения зависит от конкретных требований заказчика, что позволяет снизить затраты на покупку продукта при малых масштабах сети компании, которую необходимо диагностировать удалённо. Также отличительной особенностью этой системы является отсутствие полноценной поддержки на Русском языке.

На рисунке 2.5 представлен прайс на услуги компании Paessler AG, который прямо пропорционален количеству сенсоров в системе мониторинга:

PRTG 500	PRTG 1000	PRTG 2500	PRTG 5000	PRTG XL1	PRTG PLUS
\$ 1,750 *	\$ 3,200 *	\$ 6,500 *	\$ 11,500 *	\$ 15,500 *	CUSTOM
Perpetual license	Perpetual license	Perpetual license	Perpetual license	Perpetual license	Subscription license
500 sensors	1,000 sensors	2,500 sensors	5,000 sensors	Unlimited sensors	Unlimited sensors
1 server installation	1 server installation	1 server installation	1 server installation	1 server installation	Unlimited server installations
BUY NOW	BUY NOW	BUY NOW	BUY NOW	BUY NOW	CONTACT SALES

Рисунок 2.5 – Цены на услуги компании Paessler AG

Подводя итоги касательно PRTG Network Monitor, можно сказать, что описываемая компания имеет внушительный опыт в данной сфере и умеет подстраиваться под нужды клиента. Пользовательский интерфейс аналогично соответствует всем требованиям, но решающим фактором при выборе системы SLA-мониторинга для компании не столь крупных масштабов будет являться цена приобретаемого продукта. В целях экономии материальных ресурсов и времени заказчику выгоднее будет купить оборудование с предустановленными системами мониторинга, чем покупать отдельно. Именно поэтому данное решение имеет два существенных недостатка: цена покупки и необходимость дополнительной установки на купленное стороннее оборудование.

SLAMON Online.

Компания SLAMON основана в России и предоставляет услуги мониторинга сетей в формате подписки с ежемесячной оплатой. Функционал системы соответствует высшим требованиям, так как SLAMON основан на платформе, включенной в Единый реестр средств измерений Белоруссии и России.

Предоставляемые компанией услуги и заявленные преимущества:

- мониторинг доступа в Интернет;
- мониторинг качества IP-телефонии;
- мониторинг доступности сайта;
- мониторинг корпоративных систем;
- контроль загрузки канала;
- приоритизация трафика;
- отчеты SLA;
- мобильное приложение;
- интеграция с сетевым оборудованием;
- интеграция с Service Desk;
- высочайшая точность измерений.

Данная компания специализируется на предоставлении своих услуг в формате подписки с ежемесячной оплатой за каждый установленный аппаратный и программный агент. Выбранное решение довольно специфично и подойдет не для каждой компании. Главная особенность, которую нужно учитывать – с ростом офисов и масштабов сети компании придется увеличивать месячную плату за предоставляемые услуги.

Спектр услуг, предоставляемый SLAMON Online, зависит от типа пакета оформленной подписки. Для получения доступа ко всем перечисленным выше услугам необходимо оплатить самый дорогой пакет.

Цены на услуги компании представлены на рисунке 2.6:

БАЗОВЫЙ	ОПТИМАЛЬНЫЙ	ПРЕМИУМ
₽ 1 150 /месяц за агента	₽ 1 900 /месяц за офис	₽ 2 450 /месяц за офис
ПОЗВОЛЯЕТ ВЫЯВИТЬ ПРОБЛЕМЫ С КАЧЕСТВОМ СЕРВИСОВ:	ПОЗВОЛЯЕТ ОПРЕДЕЛИТЬ ПРИЧИНУ ПРОБЛЕМ:	ПОЗВОЛЯЕТ ПОВЫСИТЬ ПРОИЗВОДИТЕЛЬНОСТЬ ПРИЛОЖЕНИЙ:
Мониторинг Интернет	Мониторинг IP-телефонии	Контроль загрузки канала*
Мониторинг web-сайта	Мониторинг Webex	Анализ трафика приложений*
Мониторинг Email	Пользовательские сценарии	Приоритизация приложений*
Мониторинг онлайн-сервисов	Измерение скорости доступа в Интернет	Интеграция с Service Desk
Не более 3-х офисов	Периодические отчеты SLA	Мобильное приложение
	Уведомления по Email	Уведомления по почте
	Мониторинг облачных сервисов	Помощь в настройке сервиса
Выбрать	Выбрать	Выбрать

Рисунок 2.6 – Стоимость услуг компании SLAMON

Для работы SLAMON необходимо установить агент в точке, из которой будет осуществляться мониторинг качества сервисов. Агенты бывают программные и аппаратные. Можно сделать вывод, что данное решение применимо в компаниях малого масштаба с отсутствующими предустановленными заводом-изготовителем агентами в аппаратных узлах сети. Если же сеть, мониторинг которой необходимо производить, значительно расширяется, плата за подписку на услуги сервиса уверенно возрастает. Данное средство имеет смысл внедрять исключительно в компании небольшого масштаба, руководители которых желают получить полный спектр услуг по настройке и поддержке мониторинга, при этом не вдаваясь в подробности реализации.

3 Теоретическая часть

В связи с тем, что вышеописанные аналоги обладают рядом перечисленных недостатков, тема данной ВКР является актуальной.

Так как технический прогресс не стоит на месте, пользователь изъявляет желание получать обновления полученного продукта и при этом не переплачивать за ненужные ему дополнительные функции и возможности в приобретённом им товаре.

Для пользователей сети Internet куда важнее получать обновления не только функционала программной продукции, но и визуального интерфейса.

Исходя из вышесказанного, можно сделать вывод, что перечисленные системы SLA-мониторинга не имеют недостатков в техническом плане, дело лишь в цене приобретаемого продукта, usability и информативности пользовательского интерфейса программного решения. Поэтому в качестве предлагаемого средства в рамках данной ВКР решено было представить собственную версию WEB-интерфейса SLA-мониторинга, работающую в связке с тестовым SLA-сервером, предоставленным компанией D-Link. Предлагаемая новая улучшенная версия графического интерфейса будет сочетать в себе преимущества всех перечисленных аналогов. Также новое программное решение будет разрабатываться с целью устранения недостатков, выявленных при анализе конкурентов на мировом рынке.

Произвести интеграцию разрабатываемого программного решения предлагается следующим образом. Программный продукт должен быть предустановлен на реализуемом компанией оборудовании перед продажей. Осуществление запуска и настройки системы SLA-мониторинга может производиться как и мастерами компании-изготовителя, так и средствами покупателя. Другими словами, программное оснащение должно иметь внятную и подробную документацию, а пользовательский интерфейс должен быть интуитивно понятным для целевого пользователя.

3.1 Системный анализ предметной области

Рассмотрим пример. При приобретении аппаратного обеспечения для организации малых масштабов или систем умного дома заказчик может не задаваться вопросом касательно внедрения системы дистанционного мониторинга в свою сеть. В дальнейшем, при вынужденном расширении внутренней сети предприятия, или системы умного дома, появится вопрос о необходимости SLA-мониторинга. В ситуации, при которой SLA-агенты были предустановлены на оборудовании заводом-изготовителем, клиенту нет нужды искать сторонних поставщиков требуемого решения, так как все необходимые программные средства уже имеются в каждом узле настроенной внутренней сети – остаётся лишь произвести активацию и настройку SLA-мониторинга.

Именно поэтому покупка аппаратного обеспечения для собственной сети с предустановленными системами SLA-мониторинга является вкладом заказчика в успешное будущее его компании, а разработка и поддержка WEB-интерфейса для этих систем – актуальной задачей в условиях современного мира. Далее будет рассмотрено более детально, в чём заключается суть данного вида сетевого мониторинга.

SLA-агент – механизм диагностики состояния сети на стороне конечного пользователя. Его задача заключается в периодической отправке статистических данных, собранных устройством с системных счетчиков, а также результатов проверки доступности заранее заданных узлов различными утилитами.

Другими словами, задача программного агента на каждом узле – периодическая отправка данных на сервер, а задача сервера – отправка обработанных данных пользователю по запросу последнего, которым может являться, например, системный администратор.

Нам предстоит организация взаимодействия с сервером, а значит, нужно разделить весь объём работы на два фронта – front end и back end. Выполнение данной ВКР подразумевает получение существующих на сервере данных помощью

web-интерфейса. Разрабатываемое ПО должно отвечать за корректный поиск уже существующих статистических данных на тестовом сервере с указанными пользователем параметрами – другими словами, связывать пользователя и сервер (рисунок 3.1):

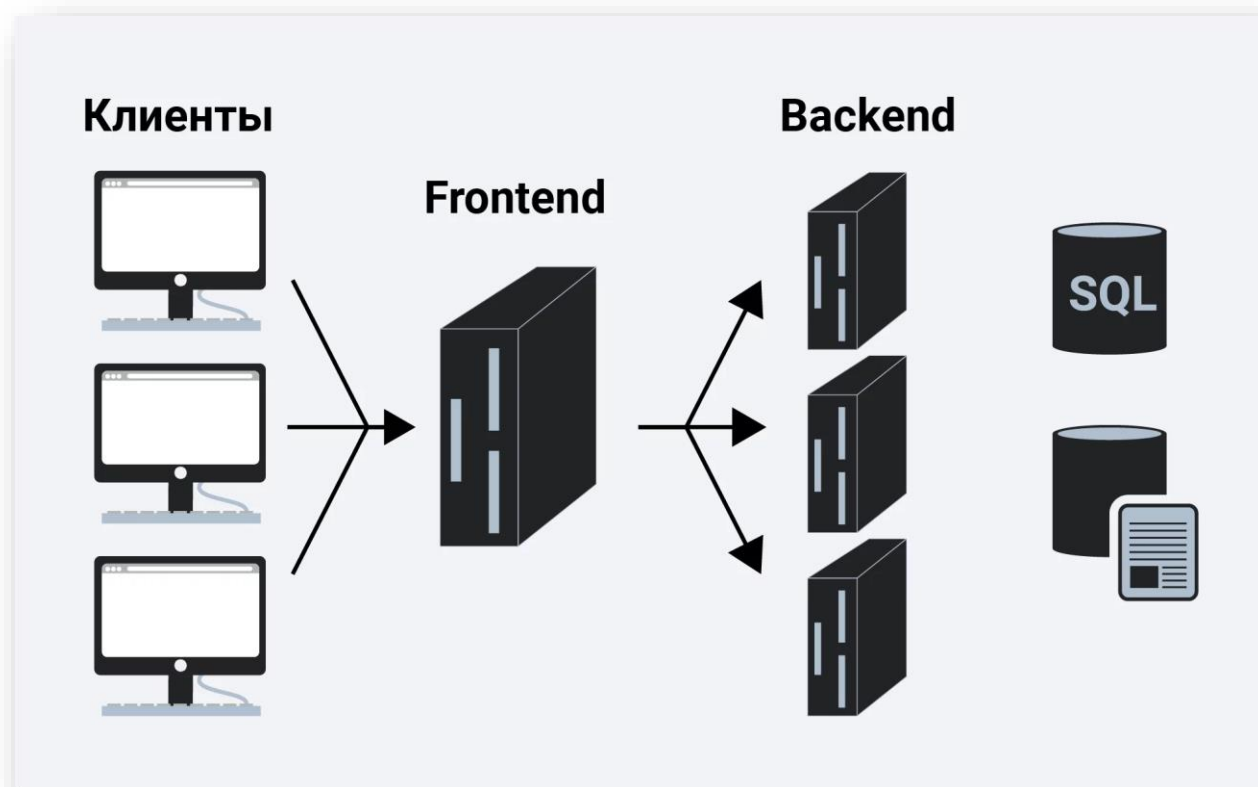


Рисунок 3.1 – Диаграмма фронтов разработки

В рамках данной ВКР взаимодействие с сервером будет выполняться по HTTP API, то есть, работа с сервером происходит в режиме чёрного ящика: разработчику frontend совершенно не обязательно знать внутреннее устройство сервера. Работа по API – это получение ожидаемых данных определённого типа в ответ на отправленные данные определённого типа, естественно, в формате JSON.

Подводя итоги тщательного анализа узкой предметной области, разрабатываемый web интерфейс должен обрабатывать, а затем отправлять введённые пользователем данные, затем получать ответ сервера, обрабатывать и отображать полученные данные.

3.2 Выбор средств разработки и языков программирования

Так как рассматриваемый существующий на данный момент Web-интерфейс SLA-сервера от компании D-Link разработан на фреймворке AngularJS, который, как известно, устарел и имеет ряд серьезных недостатков, выбор средств для разработки необходимо совершать исходя из разнообразия подходящих современных решений. Далее будет описан выбранный современный стек технологий web-разработчика.

В качестве инструмента для формирования, отправки и тестирования HTTP-запросов было принято решение использовать утилиту Postman. Преимущества данного программного решения:

- ПО поставляется бесплатно;
- интуитивно понятный пользовательский интерфейс;
- наличие встроенных гайдов по использованию;
- широкий спектр предоставляемых возможностей;
- гибкая настройка рабочего пространства;
- множество положительных отзывов;
- бесперебойная работа;
- минимальные требования к ресурсам ПК.

Данный продукт используются не только тестировщиками, занимающимися API в ключе автоматизации и тестирования. Утилита так же предоставляет интерес для разработчиков в плане написания и выявления ошибок в API. То есть, инструмент представляет собой полноценную IDE с возможностью тестирования API.

Доступно три версии ПО:

- Postman - бесплатно;
- Postman Pro - \$8 в месяц;
- Postman Enterprise - \$21 в месяц.

В рамках данной разработки будет достаточно набора возможностей, предоставляемых бесплатной версией.

Пользовательский интерфейс утилиты Postman представлен на рисунке 3.2:

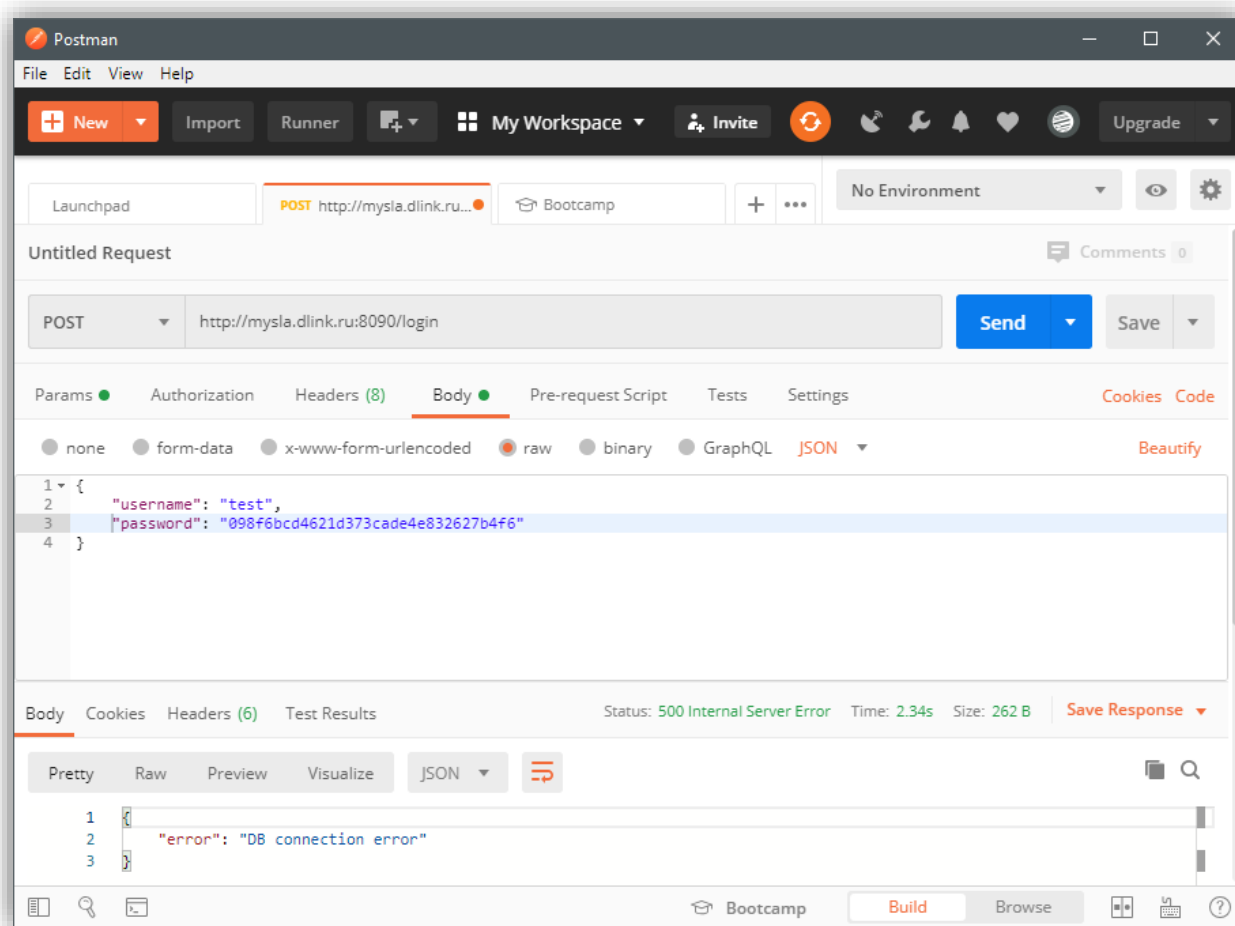


Рисунок 3.2 – Пользовательский интерфейс инструмента Postman

Так как цель разработки – создание именно WEB-интерфейса, необходим WEB-браузер для отладки и тестирования продукта. Очевидно, что для полноценного тестирования необходимо проверить полученное программное решение во всех существующих браузерах, чтобы исключить возможные несоответствия в отображении элементов с разных платформ и устройств. Но вести разработку и отладку сразу с нескольких браузеров в рамках данной работы не является целесообразным.

В качестве основного интернет-браузера был выбран Google Chrome, так как большинство пользователей пользуются именно им. Опираясь на статистику, в декабре 2019 года 66.64% запросов с персональных компьютеров к WEB-сервисам осуществлялось с помощью Chrome, а с мобильных устройств - 62.08%.

Главной и отличительной особенностью браузера от Google на данный момент является наличие собственного магазина расширений, его название – Chrome Web Store. В маркете присутствует множество утилит для помощи разработчику и автоматизации в создании, отладки и тестировании программных продуктов.

Стоит упомянуть и встроенный инструмент разработчика под названием Chrome DevTools, который позволяет редактировать страницы «на лету» и быстро диагностировать проблемы, и эта помощь является незаменимой для ускорения разработки сайтов и сервисов, улучшения их качества. Основные функции, предоставляемые данным набором инструментов:

- просмотр и изменение DOM - объектной модели документа;
- просмотр и изменение CSS текущих страниц;
- отладка JavaScript;
- просмотр сообщений и запуск JavaScript в консоли;
- оптимизация скорости WEB-сайта;
- исследование активности сети.

Функция просмотра контента Device Mode, которая позволяет увидеть разрабатываемую страницу такой, какой она будет отображаться на других устройствах. Есть возможность настроить разрешение, DPI, коэффициент соотношения пикселей, масштаб, устанавливаемый по умолчанию мобильным браузером и прочие параметры просмотра контента.

В качестве среды разработки был выбран редактор кода Microsoft Visual Code в наборе с множеством расширений, которые в свою очередь превращают текстовый редактор в полноценную IDE. Преимущества установленных расширений, а также самого VS Code:

- поддержка Angular;
- отладчик;
- поддержка утилит для удобного тестирования;
- система подсветки синтаксиса и помощи в написании кода;
- гибкая система настроек;
- интеграция с системами контроля версий.

В качестве фреймворка для разработки клиентской части был выбран Angular. Angular позволяет создавать так называемые «Одностраничные приложения» или SPA (веб-приложения или веб-сайты, использующие единственный HTML-документ как оболочку для всех веб-страниц и организующие взаимодействие с пользователем через динамически подгружаемые HTML, CSS и JavaScript). Преимущества Angular:

- инструменты разработчика (CLI);
- единая структура проекта;
- поддержка TypeScript;
- dependency injection;
- шаблоны, основанные на расширении HTML;
- кроссбраузерная поддержка HTTP, WebSockets, Service Workers;
- динамический роутинг;
- material design - библиотека компонентов пользовательского интерфейса.

Одним из основных минусов Angular является высоких порог вхождения из-за Observable (RxJS) и Dependency Injection.

В качестве UI библиотеки была выбрана библиотека Angular Material [8].

Разрабатываемому WEB-интерфейсу требуется среда JavaScript runtime, в качестве которой была выбрана Node.js.

В качестве ПО для проектирования пользовательского интерфейса, создания прототипов, концептов, макетов, графических эскизов и прочего был выбран инструмент Adobe XD - удобное векторное средство для дизайна взаимодействия с пользователем.

В данном разделе были выбраны и описаны средства разработки и языки программирования, выделены их ключевые особенности.

4 Проектная часть

4.1 Проектирование пользовательского интерфейса

4.1.1 Разработка дерева форм

В процессе изучения аналогов в плане выявления их недостатков и преимуществ, а также тщательного анализа предметной области спроектировано дерево форм, которое будет являться основным логическим каркасом для проектирования и создания пользовательского интерфейса и поведения пользователя, или, если использовать общепринятые термины – UX/UI. Спроектированное дерево форм представлено на рисунке 4.1.



Рисунок 4.1 – Дерево форм

Авторизация – маленькая формочка с полями ввода логина, пароля и кнопкой входа.

Страница поиска устройств – набор панелей расширений с функциями поиска, его настройки, вывода результатов, а также с панелью инструментов, на которой находится кнопка переключения между двумя основными режимами работы ПО – поиском отчётов и поиском устройств. Также на этой странице имеется панель с краткой сводкой основной информации.

Панель поиска имеет два поля для ввода IP и MAC адреса, а также две кнопки для поиска по IP и поиска по MAC.

Панель фильтров поиска позволяет указать требуемые параметры для поиска по IP или по MAC – такие как:

- DROP;
- SRC;
- Fragments;
- LinkDown;
- 64bytes;
- LessFromThisDate;
- MoreFromThisDate;
- DeviceModel;
- DeviceSoftware;
- SerialNumber.

Следует отметить, что для полей ввода даты предусмотрен DatePicker – удобный формат ввода требуемой даты.

На панели результатов поиска выводится список устройств, найденных по заданным параметрам.

Диалог со списком устройств – вывод краткой информации о каждой группе устройств, группировка производится по моделям устройств.

Для каждого устройства в диалоге возможно открытие формы выбора логов для просмотра – выбирается дата, время найденных логов, и в списке можно выбрать нужные логи, также можно перейти на страницу Daily Info.

Страница просмотра логов выводит детальную информацию о конкретном устройстве в определённый момент времени – сугубо технические данные узкой специализированной направленности.

Страница поиска отчётов позволяет искать отчёты по датам, выводя список всех отчётов в заданный день с указанием времени, устройства и другой технической информации. Также страница содержит несколько панелей для упрощения процедуры поиска и вывода полученной информации.

Панель отчётов текущей даты автоматически производит поиск по отчётам на текущий день и выводит результат при открытии страницы.

Панель поиска отчётов по дате позволяет изменить текущую дату на произвольную для поиска отчётов.

Результаты поиска выводятся на панели поиска, с помощью которой можно выбрать необходимый отчёт формата Daily Info.

Daily Info – набор технических узкоспециализированных данных, которые необходимы для корректной диагностики сети. Данные выводятся в удобном для пользователя формате.

Спроектированное дерево форм не является точной инструкцией по реализации графического интерфейса. WEB-разработчик имеет право внесения корректировок в карту приложения (дерево форм), однако требуется предварительная консультация с дизайнером проекта. Как показывает практика, многие существенные правки вносятся в процессе создания HTML-каркаса пользовательского интерфейса.

4.1.2 Разработка прототипа пользовательского интерфейса

Для разработки выбрана тёмная цветовая палитра, которую можно назвать «тёмная тема». Выбор тёмных оттенков обуславливается экономией электроэнергии на пользовательских мониторах, если матрица дисплея имеет распределённую подсветку и каждый пиксель подсвечивается отдельно. Также отсутствие ярких белых крупных элементов позволяет существенно снизить нагрузку на глаза пользователя, что позволяет избежать переутомления глазных мышц, сухости и прочего дискомфорта при просмотре содержимого страниц приложения. Так как в разработке WEB-интерфейса файлы стилей .CSS являются подключаемыми, в дальнейшем возможна реализация возможности переключения между светлыми и тёмными темами непосредственно во время пользования данным приложением.

Цветовая палитра фонов элементов интерфейса будет представлять градации серого цвета в тёмных его оттенках, а палитра границ и форм объектов – градации более светлых тонов серого цвета. Для максимальной читабельности текст будет белым. В целях придания уникальности данному цветовому решению выбран акцентный цвет #0087a9. При необходимости увеличения информативности пользовательского интерфейса могут понадобиться другие цвета, отличные от градаций серого, в таком случае вывод набора новых цветов должен основываться на указанном основном акцентном цвете программного решения.

Так как основной библиотекой компонентов графического интерфейса в данном проекте является Angular Material, то и прототип интерфейса, и сам интерфейс будут в стиле Material Design. Основные преимущества стилистики:

- user-friendly интерфейс – интуитивно понятный внешний вид;
- достаточная информативность иконок при умеренной лаконичности – большинство действий достаточно описать иконками без использования текстовых подсказок и прочее.

Прототипы (макеты, эскизы) пользовательского интерфейса, представленные далее, не являются точными инструкциями к разработке. Прототипы позволяют

представить примерное расположение элементов на странице, их группировку и зависимости, однако при непосредственно разработке WEB-интерфейса, создании его каркаса, стилей, а также изменение UX/UI после завершённого прототипирования допускается внесение корректировок в уже законченные прототипы. Другими словами, WEB-разработчик имеет право вносить правки в UX/UI при условии проведения консультаций с дизайнером, при этом сами прототипы могут оставаться без изменений.

Так как разработка интерфейса производится с уже существующим API, формат данных, их тип и структура predetermined, поэтому некоторые элементы и формы интерфейса будут проектироваться и корректироваться на стадии разработки, непосредственно в момент расширения функционала, появления необходимости отрисовки полученных данных, а также точного понимания, какие именно данные нужно отображать.

Далее представлены разработанные прототипы пользовательского интерфейса, а именно страниц приложения. Разработка концепта базируется в первую очередь на главных правилах построения графического дизайна и «поведении пользователя». Также разработка макетов велась с целью исключения недостатков и взятия в оборот достоинств, выявленных при системном анализе предметной области и обзоре конкурентов на мировом рынке.

На рисунке 4.2 изображён эскиз страницы авторизации. С точки зрения эргономичности UX/UI решено не перегружать пользовательский интерфейс информацией. При открытии приложения перед оператором находятся два текстовых поля и две кнопки. Текстовые поля предназначены для ввода логина и пароля, а кнопки для скрытия/раскрытия пароля и авторизации.

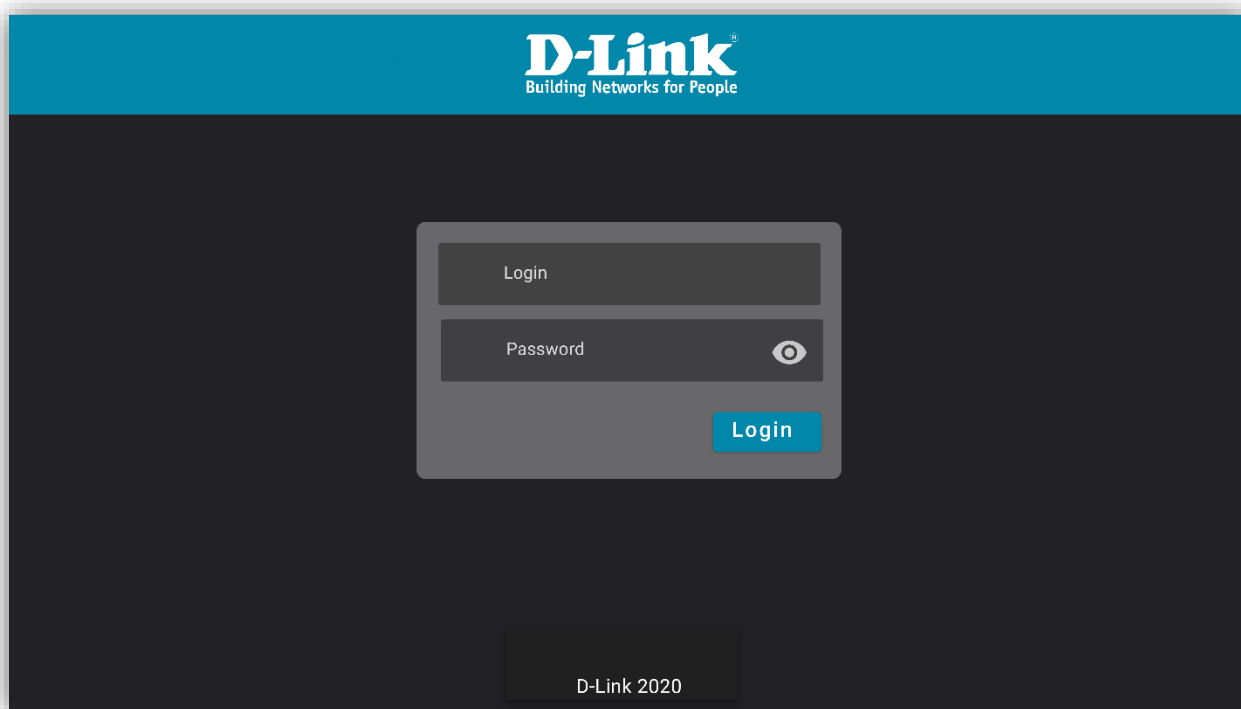


Рисунок 4.2 – Эскиз страницы авторизации

Макет страницы поиска представлен на рисунке 4.3. На панели инструментов верхней части экрана появляются кнопки в правой части, а логотип переезжает в левую часть. В теле страницы расположены две карточки: поиск по двум основным параметрам (IP и MAC адреса) и фильтры для этих видов поиска. Кнопки поиска имеются в каждом поле поиска, чтобы разделить виды поиска между собой.

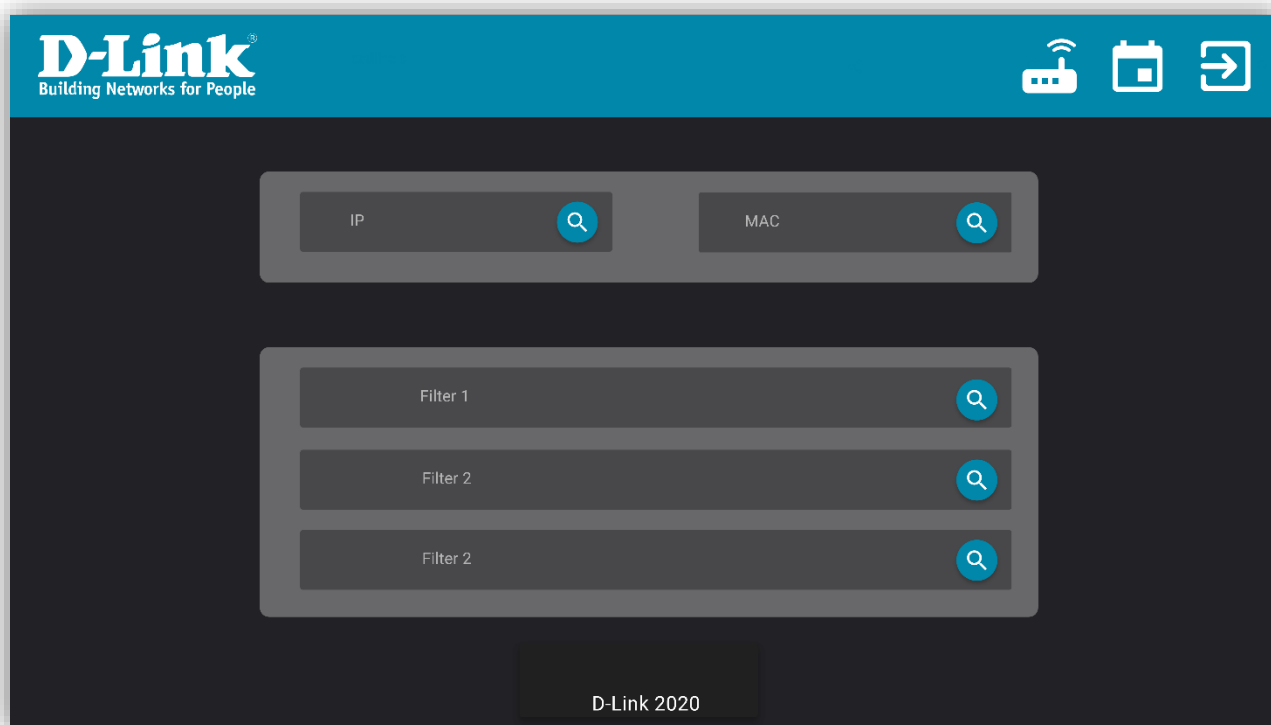


Рисунок 4.3 – Макет страницы поиска устройств

Производить выдачу результатов поисковых запросов пользователя предлагается в формате таблиц, где каждый столбец будет соответствовать определённому параметру, а каждая строчка (за исключением заголовочной) – соответствовать конкретному устройству. Также на каждой строчке предлагается отрисовывать кнопку для открытия детальной информации для каждого устройства. Прототип страницы выдачи результатов поиска представлен на рисунке 4.4:

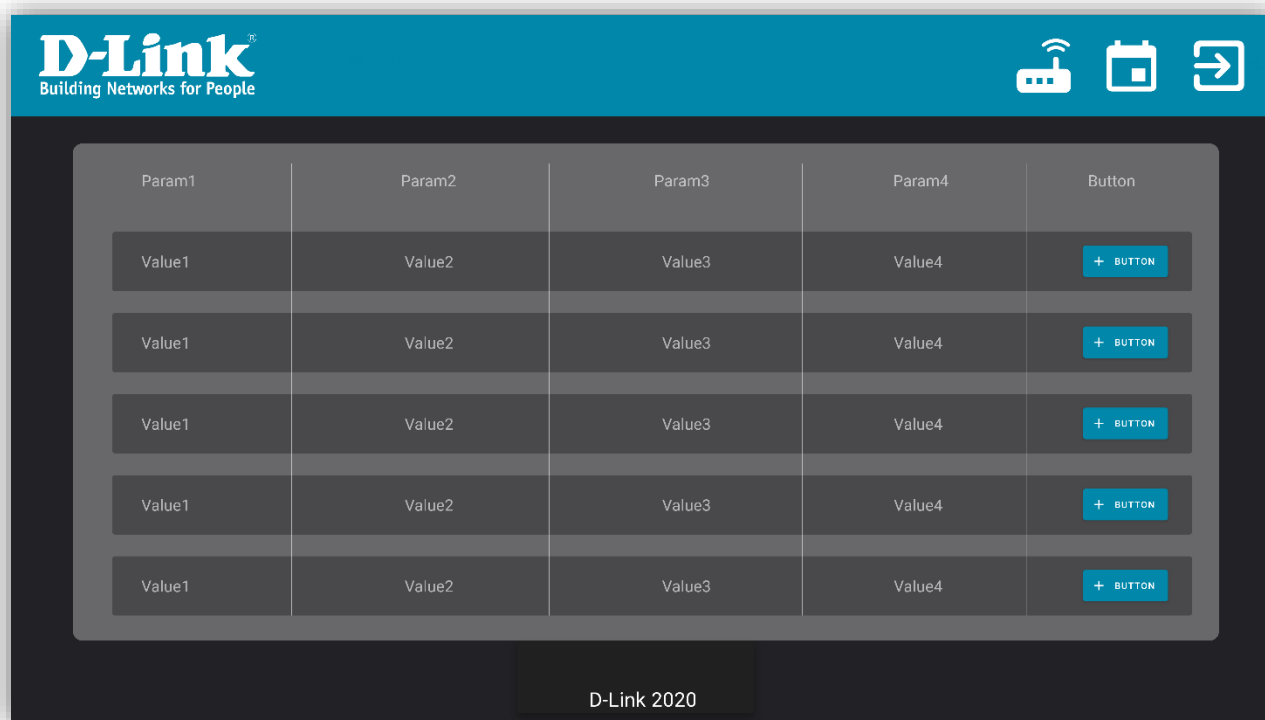


Рисунок 4.4 – Прототип таблицы с результатами поиска

4.2 Организация взаимодействия с сервером

Для решения поставленной задачи, отладки запросов и проверки работоспособности ПО принято решение использовать SLA-сервер по адресу <http://mysla.dlink.ru>. Тестирование работоспособности будет проводиться с помощью отладочной учетной записи. Разрабатываемый WEB-интерфейс может взаимодействовать с HTTP API, доступным онлайн по адресу <http://mysla.dlink.ru:8090>.

Следует отметить, что в связи с особенностями политики CORS сервер запрещает подключение к нему с других доменов, вследствие чего было принято решение использовать проху-сервер для добавления заголовков «Access-Control-Allow-Origin» в каждый запрос. Прокси-сервер работает таким образом, что принимает запрос от пользователей и перенаправляет их на указанный сервер, и полученные данные от сервера перенаправляет обратно пользователю. Исходные коды сервера можно найти в репозитории проекта [9]. Развёртка сервера была произведена на хостинг для разработчиков Heroku [10], вследствие чего доступ к

этому проху может быть осуществлён с любого устройства, подключенного к сети Internet [11].

Аутентификация осуществляется с помощью вызова метода POST /login с полезной нагрузкой {username: username, password: md5.createHash(password)}, который возвращает объект с полями:

- token - токен доступа;
- permission - права доступа;
- user - имя пользователя.

Поле token сохраняется и прикрепляется к каждому последующему запросу в заголовке запроса «Token».

В Приложении Б представлены некоторые методы, доступные для вызова на вышеуказанном тестовом API.

4.3 Описание технологии разработки клиентской части веб-приложений с использованием фреймворка «Angular»

Angular – фреймворк, написанный на TypeScript, или платформа, созданная для разработки одностраничных web-приложений с использованием таких языков, как TypeScript и HTML. Фреймворк реализует различные функции в виде набора TypeScript библиотек.

Архитектура приложения, построенного на Angular опирается на некоторые фундаментальные концепции. Основоположными блоками для "стройки" можно назвать NgModules (модули Angular), которые предоставляют контекст компиляции для компонентов. Модули Angular собирают связанный код в функциональные наборы; Angular-разработка определяется набором модулей. В разработке всегда имеется по крайней мере корневой модуль.

NgModule определяется классом с декоратором @NgModule(). Декоратор @NgModule() – это функция, которая принимает один объект метаданных, свойства

которого описывают модуль. Далее представлены свойства, которые можно назвать наиболее важными:

- `declarations`: компоненты, директивы и пайпы, которые принадлежат этому NgModule;
- `exports`: подмножество объявлений, которые должны быть видны и использоваться в шаблонах компонентов других NgModules;
- `imports`: другие модули, чьи экспортированные классы необходимы шаблонам компонентов, объявленным в этом NgModule;
- `providers`: создатели сервисов, которые этот NgModule вносит в глобальный набор сервисов; они становятся доступными во всех частях приложения;
- `bootstrap`: основное представление приложения, называемое корневым компонентом, в котором размещены все остальные представления приложения. Только корневой NgModule должен устанавливать свойство `bootstrap`.

Компонент контролирует участок экрана, называемый представлением. Внутри класса компонента определяется его логика – что он делает для поддержки представления. Класс взаимодействует с представлением через API свойств и методов. Вид компонента определяется его сопутствующим шаблоном. Шаблон – это форма HTML, которая сообщает Angular, как визуализировать компонент. Иерархия представлений может включать представления от компонентов в одном и том же NgModule, но она также может включать представления от компонентов, определенных в разных NgModules.

Сервис – это широкая категория, охватывающая любое значение, функцию или особенность, которая нужна приложению. Сервис обычно является классом с узконаправленной, четко определенной целью. Его задача - выполнять что-то определённое, конкретное. Фреймворк отличает компоненты от сервисов для повышения модульности и возможности повторного использования. Дело в том, что работа каждого компонента заключается в том, чтобы обеспечить взаимодействие с

front-end и ничего более. Компонент должен представлять свойства и методы для привязки данных, чтобы быть посредником между представлением (отображаемым шаблоном) и логикой приложения (которая часто включает в себя некоторое представление о модели).

В данном разделе были описаны основные принципы работы с фреймворком Angular и его основные части.

4.4 Разработка пользовательского интерфейса

Тестовый сервер принимает и отправляет данные в формате JSON. Преимущество HTTP API – передача данных в удобном для восприятия человеком виде, а также удобном для пост-обработки этих данных для их последующего использования.

Основная задача разрабатываемого ПО – сбор введённых пользователем данных, обработка в JSON, отправка на сервер, получение с сервера JSON, повторная обработка для отображения пользователю.

Для ускорения процесса разработки и улучшения usability пользовательского интерфейса во всём проекте использована библиотека Angular Material. Для соблюдения выбранного цветового решения было принято решение заменить в одной из prebuilt-themes акцентные цвета на собственный цвет и корректировать стили конкретных элементов под свои нужды.

Исходные коды всех разработанных элементов приведены в Приложении А.

Ссылка на репозиторий проекта на GitHub, или архив с последним успешным билдом могут быть предоставлены по требованию.

4.4.1 Корневой модуль приложения

Так как разработка является демонстрационной, весь код приложения размещён в одном корневом модуле. Функционал пользовательского интерфейса разделён на несколько логических блоков и распределён по компонентам и сервисам.

4.4.2 Компонент «Корневой»

Данный компонент несёт в себе невидимый для пользователя функционал, так как главное предназначение компонента – хранение констант, необходимых для корректной работы приложения и создание HTML-каркаса, который предопределяет расположение всех остальных дочерних компонентов на экране пользователя и передачу в них значений констант, хранящихся в корневом компоненте.

Корневой компонент не имеет вычислений в коде или сложной логики, его основная задача – хранение и передача другим компонентам значений констант, передача, получение и хранение значений переменных, а также скрывание/показ других дочерних компонентов.

4.4.3 Компонент «Панель инструментов»

Панель инструментов – полоска акцентного цвета в верхней части экрана, которая обычно содержит логотип и несколько кнопок для быстрого доступа к основным разделам приложения.

Toolbar имеет динамический набор возможностей взаимодействия – элементы, кнопки и список возможных действий изменяется в зависимости от страницы, на которой находится пользователь. В начале работы с приложением, на странице авторизации, панель не содержит кнопок, только логотип провайдера системы SLA мониторинга, расположенный по центру экрана (рисунок 4.5):

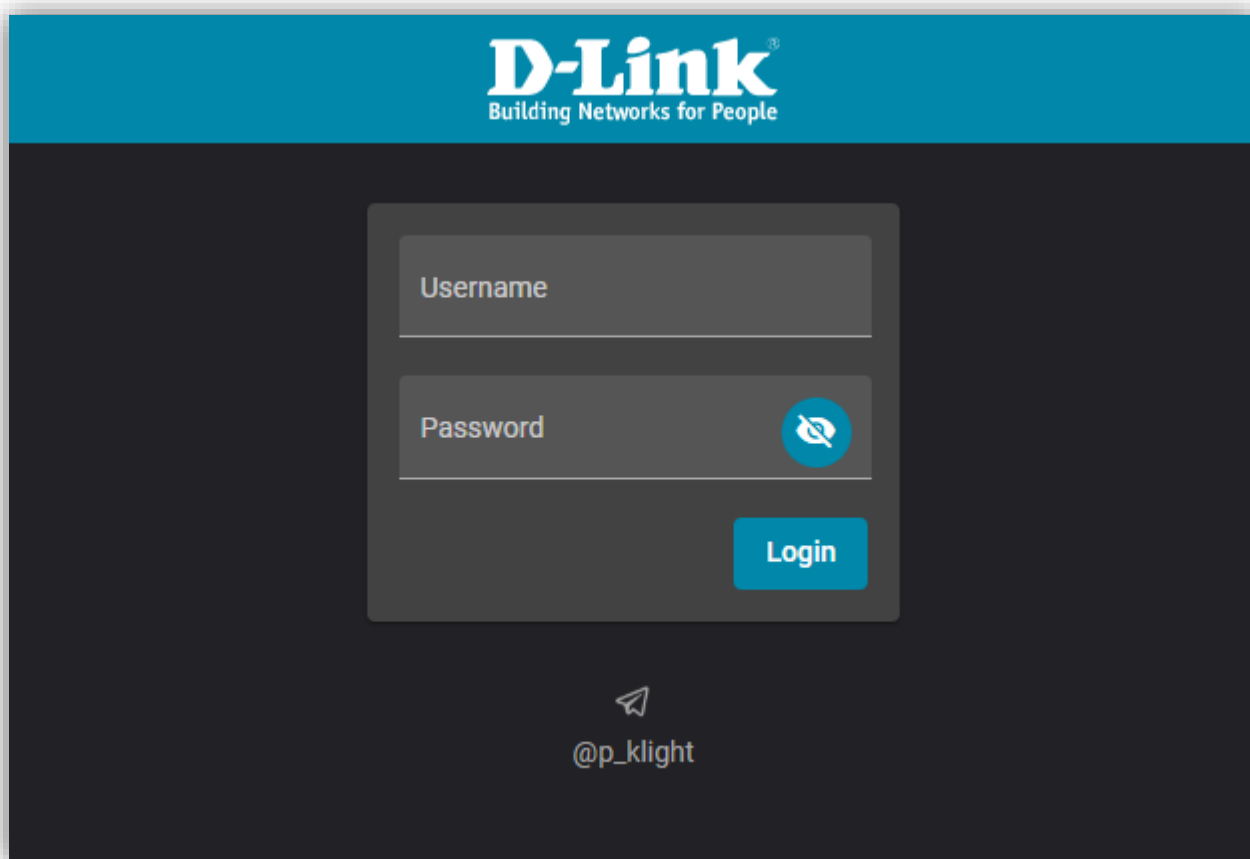


Рисунок 4.5 – Toolbar на странице авторизации

Сразу после успешной авторизации логотип на панели инструментов смещается в левый край, а в правой части панели появляются кнопки. Эти кнопки отвечают за переход между двумя основными режимами работы приложения – поиск по устройствам и по датам, а также присутствует кнопка выхода из системы (рисунок 4.6). Все 3 кнопки оснащены интерактивными подсказками – при наведении курсора мыши всплывает сообщение с описанием действия, принадлежащего кнопке (рисунок 4.7):

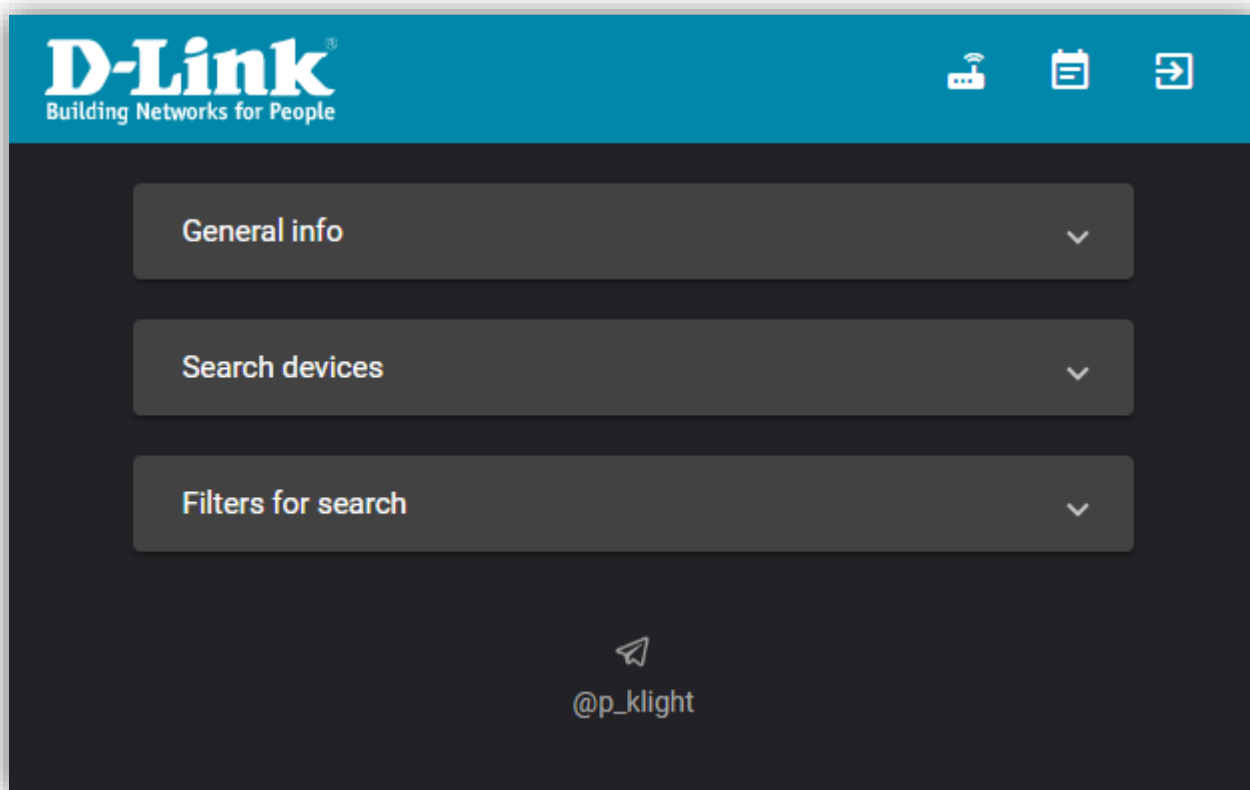


Рисунок 4.6 – Toolbar после успешной авторизации пользователя

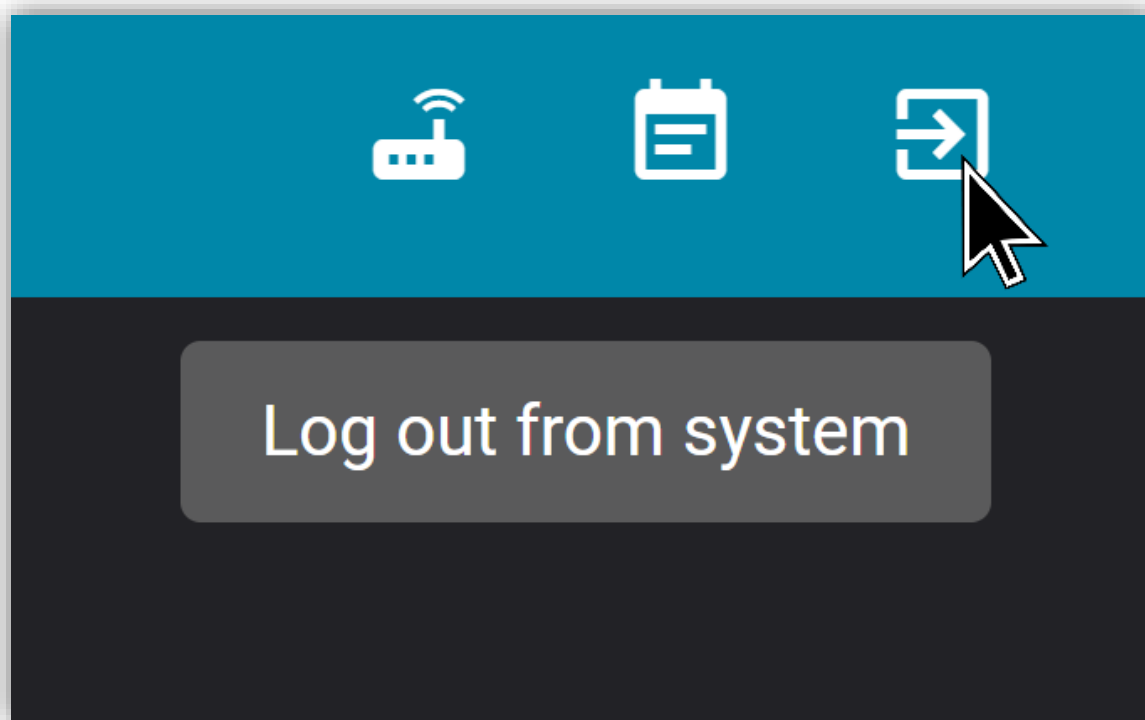


Рисунок 4.7 – Всплывающая подсказка

Визуальное поведение рассматриваемого компонента рассмотрено, далее приведено описание логики.

Компонент панели инструментов получает от корневого компонента значение переменной, которая сигнализирует об успешной авторизации, и когда значение переменной становится истинным, форма логина закрывается и открывается главная страница (страница поиска), и вместе с этим на панели инструментов логотип смещается в левый край, а с правого появляются кнопки.

4.4.4 Компонент «Подвал»

Данный компонент имеет единственное предназначение – отрисовку Footer, или подвала. Footer – сектор, или блок с информацией, который находится внизу экрана. Обычно он прикреплен или к нижней границе окна, либо к последнему (самому нижнему) элементу на странице.

Чаще всего в «подвале» размещают контактную информацию, настройки языковой локали приложения и прочие параметры, которые должны быть на каждой странице, но не акцентировать на себе слишком много внимания и не быть первым объектом, на который обращает внимание пользователь.

В данной разработке в процессе системного анализа предметной области и проектирования пользовательского интерфейса не было выявлено информации, которая должна присутствовать на каждой странице, но при этом обращаться на себя внимание в последнюю очередь. Однако создание Footer является перспективным вкладом в будущее проекта – при появлении информации для этого блока останется только вставить её в раздел. Чтобы не оставлять Footer пустым, в него были помещены контакты разработчика и логотип, информирующий о способе связи с ним. Данная информация не несёт в себе никакой смысловой нагрузки и носит сугубо демонстрационный характер, дабы проиллюстрировать пользователю работоспособность и возможный функционал. Разработанный Footer проиллюстрирован на рисунке 4.8:

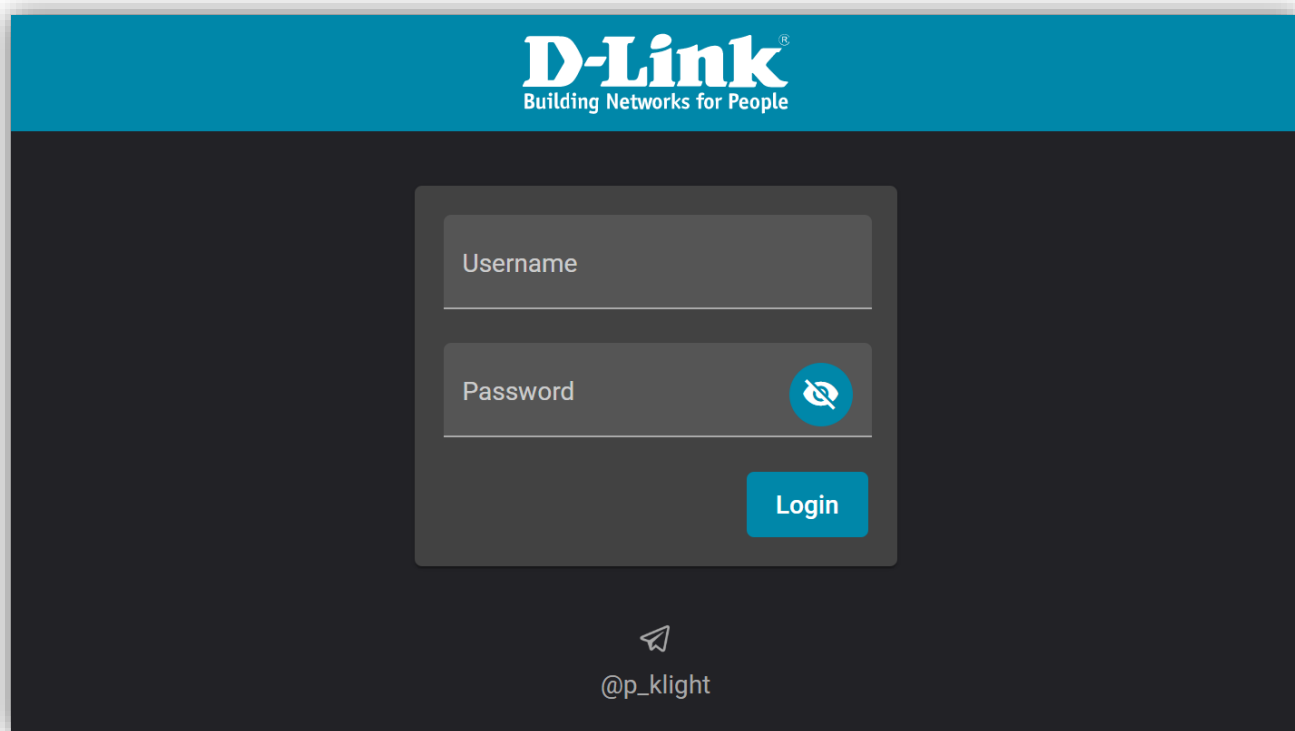


Рисунок 4.8 – Footer на странице авторизации

4.4.5 Компонент «Авторизация»

Данный компонент предназначен для работы с процедурой авторизации. Присутствует форма с двумя полями ввода текста – логин и пароль – и сама кнопка «Login» для осуществления процедуры входа. Внешний вид разработанного компонента на странице авторизации (в масштабе для повышения читабельности) представлен на рисунке 4.9:

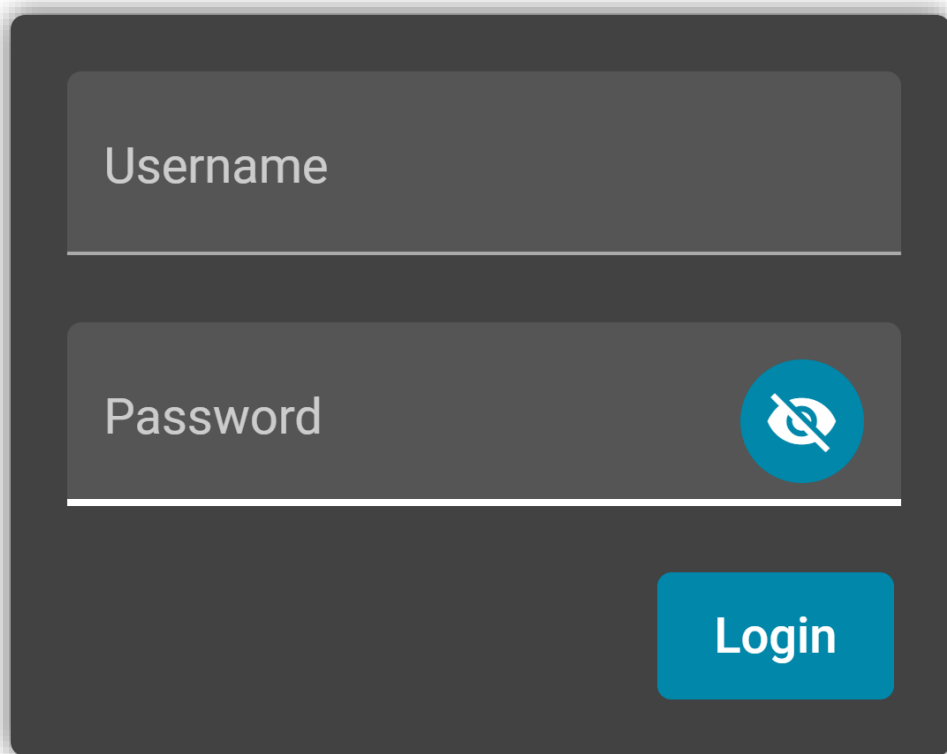
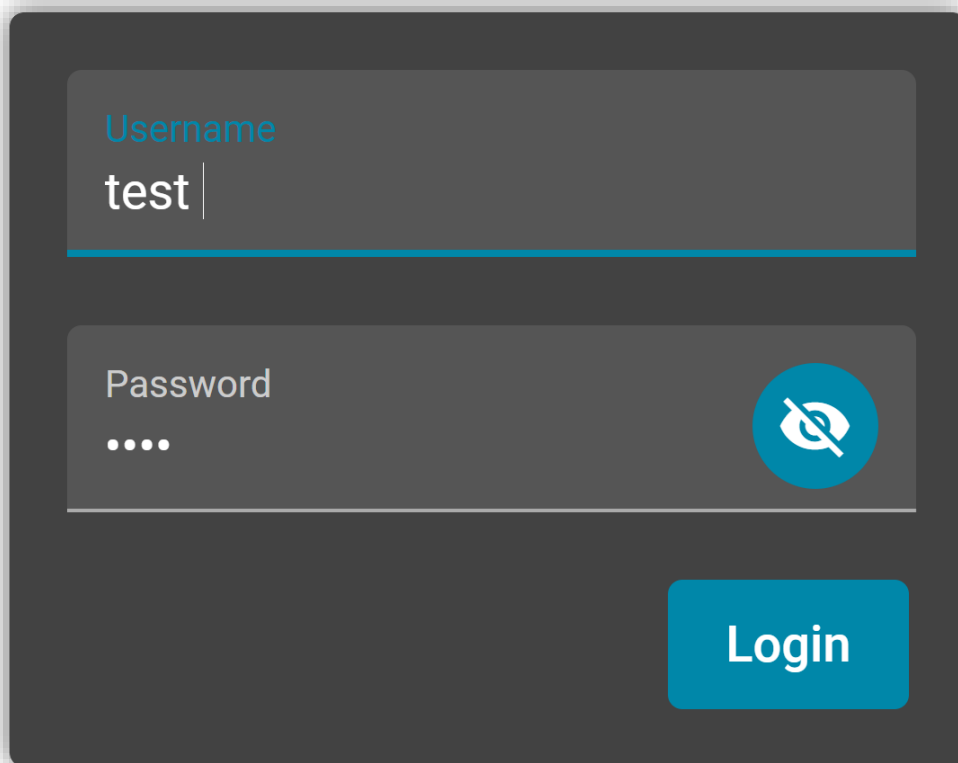
The image shows a dark gray login form component. It has two input fields: 'Username' and 'Password'. The 'Password' field has a blue circular icon with a white eye and a diagonal line through it, indicating a toggle for password visibility. Below the fields is a blue 'Login' button with white text.

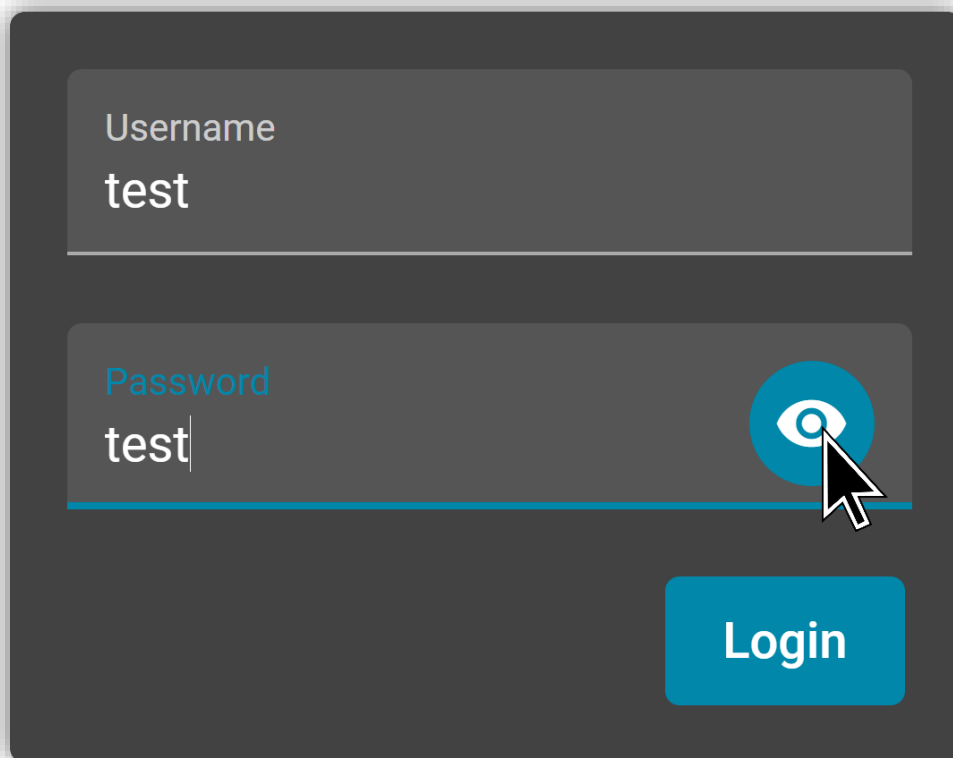
Рисунок 4.9 – Компонент «форма авторизации»

При помощи компонентов библиотеки Angular Material каждое поле содержит название, которое смещается из центра поля при попытке ввода текста (рисунок 4.10), кнопкой скрытия логина (рисунок 4.11) и анимацией процесса авторизации (рисунок 4.12), при котором нажатая кнопка начала процесса авторизации пропадает:



A login form with a dark gray background. It contains two input fields and a button. The first field is labeled "Username" in blue text and contains the text "test". The second field is labeled "Password" in gray text and contains four dots. To the right of the password field is a blue circular icon with a white eye and a diagonal line through it, indicating that the password is hidden. At the bottom right is a blue button with the text "Login" in white.

Рисунок 4.10 – Смещение названия поля при заполнении



A login form with a dark gray background. It contains two input fields and a button. The first field is labeled "Username" in gray text and contains the text "test". The second field is labeled "Password" in blue text and contains the text "test". To the right of the password field is a blue circular icon with a white eye, indicating that the password is visible. A black mouse cursor is pointing at the icon. At the bottom right is a blue button with the text "Login" in white.

Рисунок 4.11 – Кнопка отображения пароля

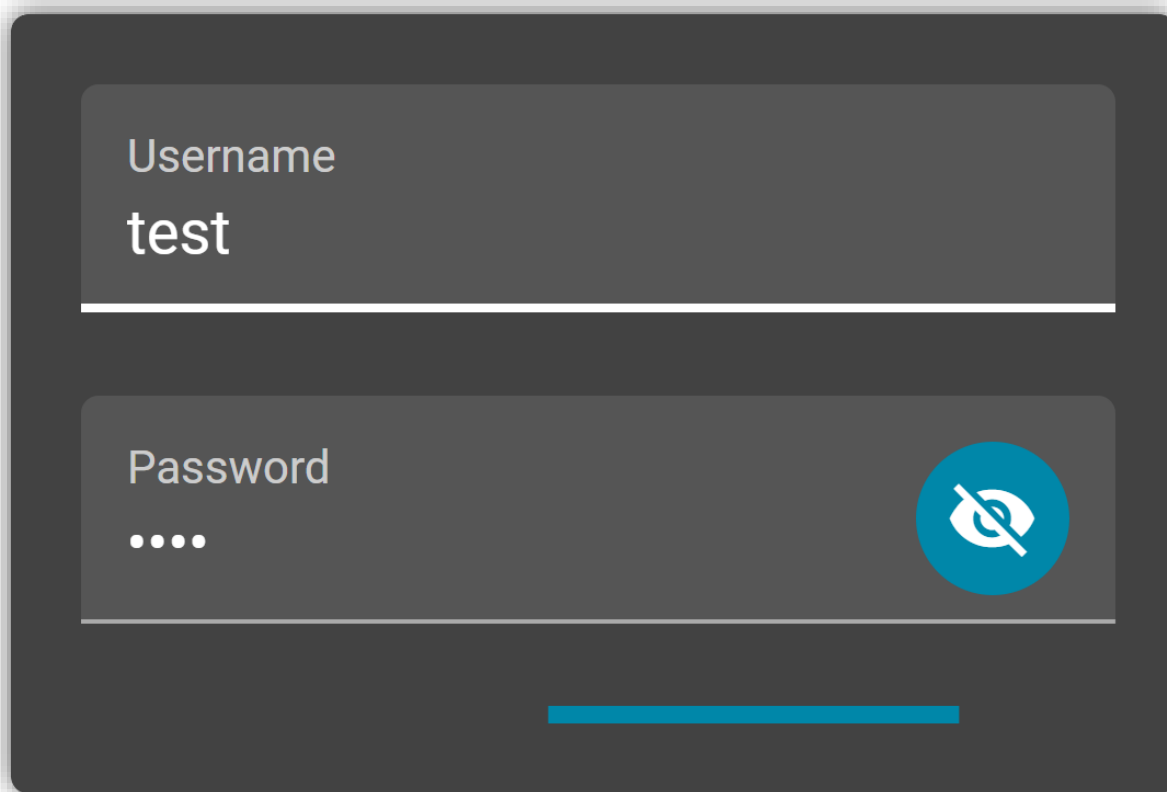


Рисунок 4.12 – Процесс авторизации

В плане выполняемого кода компонент авторизации выполняет следующие действия:

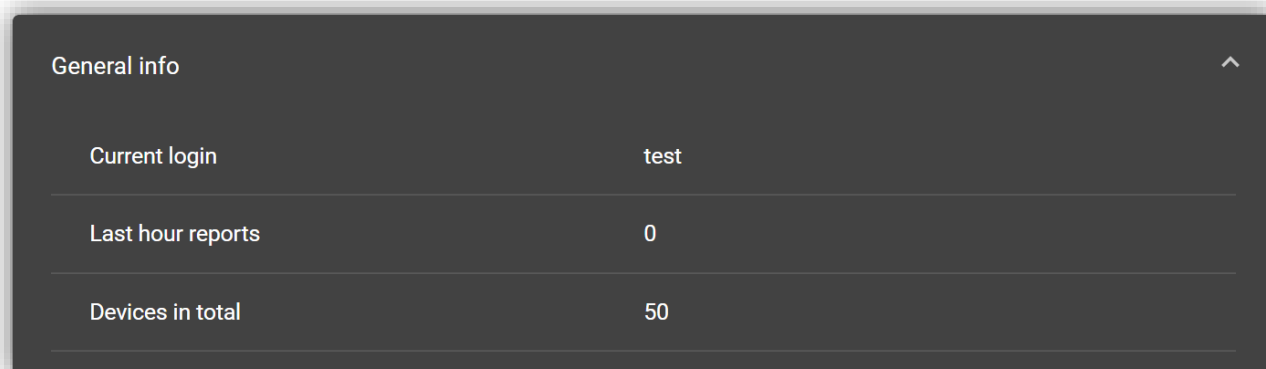
- принимает через поля логин и пароль;
- генерирует md5 hash пароля;
- отправляет на сервер запрос на авторизацию;
- рассматривает полученный ответ от сервера и в случае получения токена авторизации передаёт в главный компонент текущий токен авторизации и логин пользователя.

4.4.6 Компонент «Основная информация»

Компонент находится в верху страницы поиска по устройствам, представляет собой панель расширения и выводит основную информацию о системе (рисунок 4.13):

- логин текущего пользователя;

- количество отчётов за последний час;
- общее количество устройств в системе.



The image shows a dark-themed UI panel titled 'General info' with an upward arrow icon in the top right corner. The panel contains a table with three rows of system statistics.

Current login	test
Last hour reports	0
Devices in total	50

Рисунок 4.13 – Вывод краткой информационной сводки

Текущий логин предоставляет компонент авторизации, а значения для остальных двух строк запрашиваются с сервера и отображаются в случае успешного их получения.

4.4.7 Компонент «Поиск»

Визуальная часть состоит из трёх составляющих панелей расширения:

- поиск по IP и MAC;
- указание фильтров поиска;
- результаты поиска.

Каждое поле этого компонента имеет смещаемый заголовок во время ввода данных. Все поля, требующих определённого формата данных, имеют placeholder с подсказками.

По умолчанию панель с результатами скрыта и отображается только в случае наличия результатов поиска. Вид компонента поиска по умолчанию представлен на рисунке 4.14:

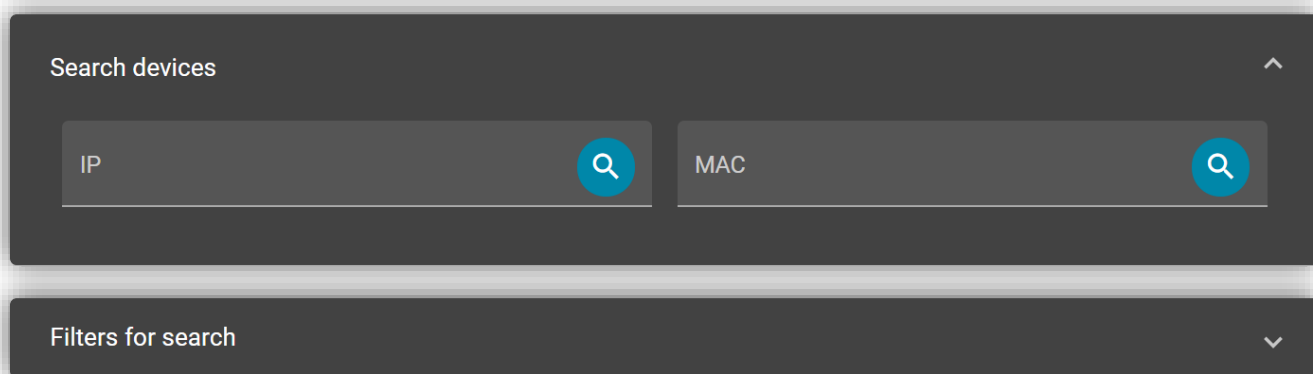


Рисунок 4.14 – Компонент поиска. Вид по умолчанию

Каждое из двух полей обладает подсказками для формата вводимых данных. Подсказки появляются при установке курсора в поле (рисунок 4.15):

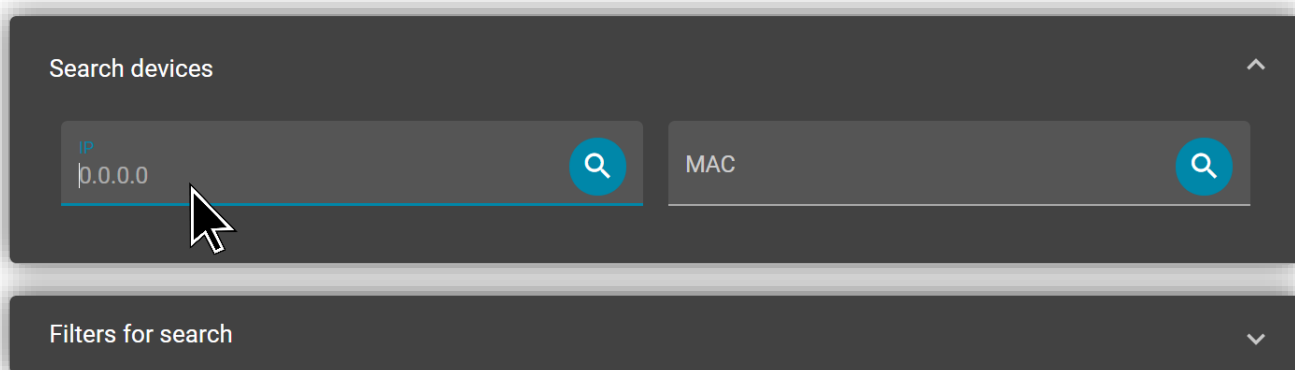


Рисунок 4.15 – Подсказки формата данных для ввода

Предусмотрено две кнопки поиска, чтобы разграничивать поиск по IP и поиск по MAC. Присутствует анимация загрузки на каждой кнопке для упрощённого восприятия процесса пользователем (рисунок 4.16):

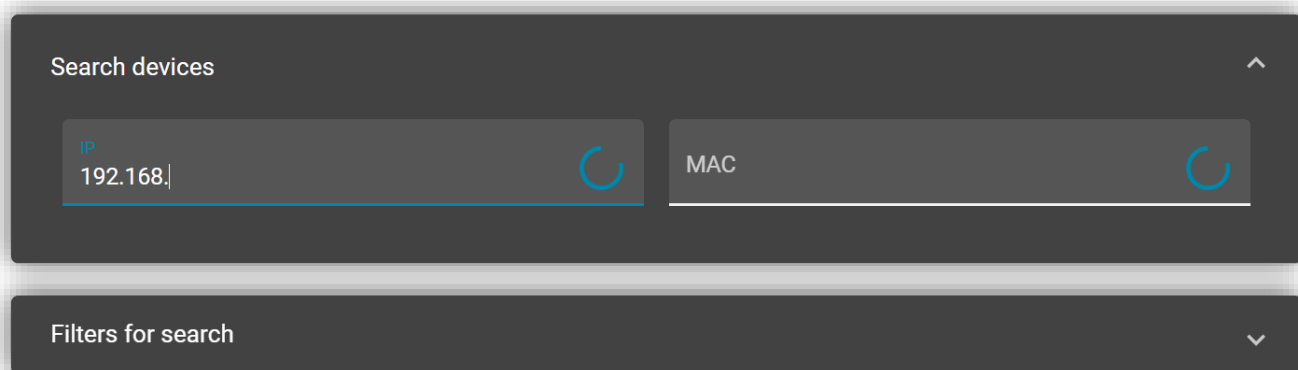


Рисунок 4.16 – Анимация процесса поиска

По умолчанию панель с фильтрами скрыта. Внешний вид развёрнутой панели представлен на рисунке 4.17:

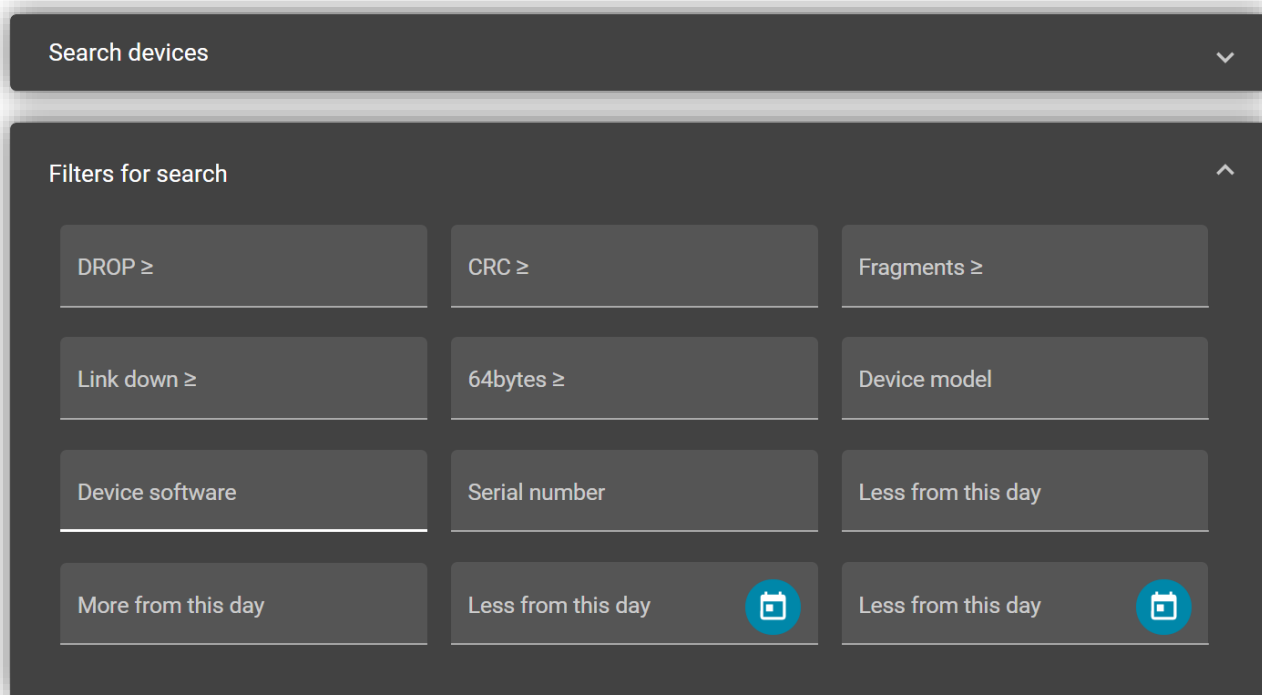


Рисунок 4.17 – Панель фильтров поиска

В полях с вводом даты реализован DatePicker в полноэкранном режиме (touch UI), позволяющий пользователю быстро выбрать дату. Перед пользователем появляется интерактивное окно выбора даты, при этом сама страница остаётся на фоне и затемняется (рисунок 4.18, без масштабирования элементов интерфейса):

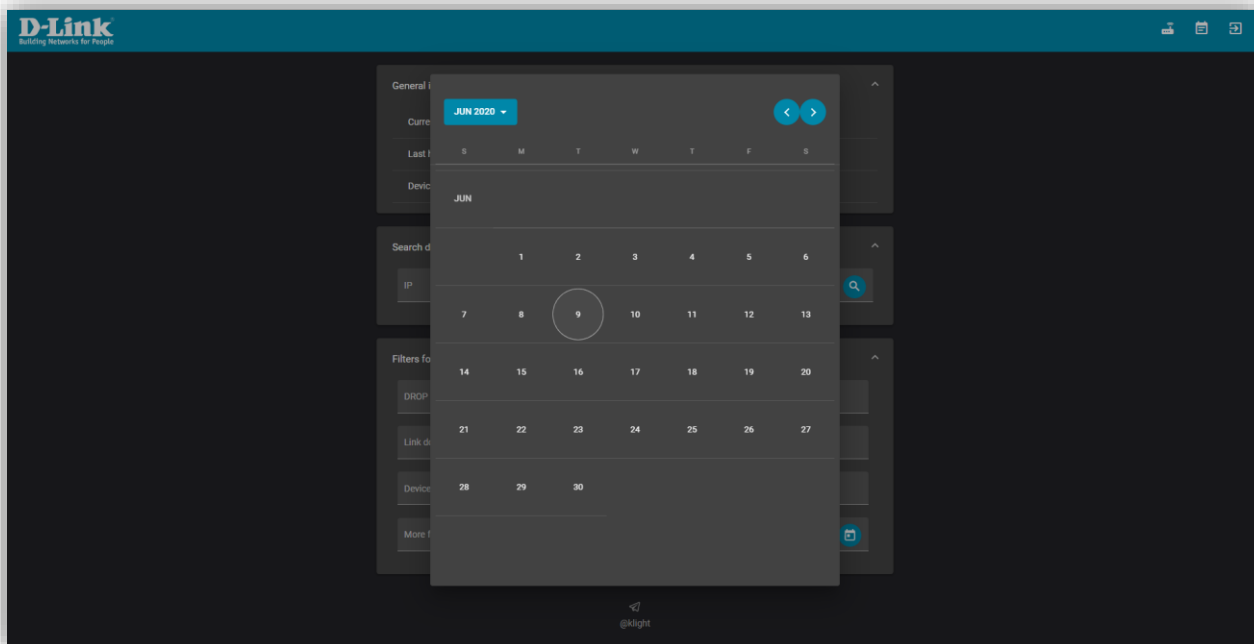


Рисунок 4.18 – Полноэкранное всплывающее окно выбора даты

В случае успешного поиска появляется панель с результатами поиска снизу панели фильтров, в случае неуспешного – сообщение об ошибке (либо средствами браузера, либо форматом уведомления в нижней части страницы), более подробно получение сообщений описано в пункте 5.2.5. Внешний вид панели с результатами поиска представлен на рисунке 4.19:

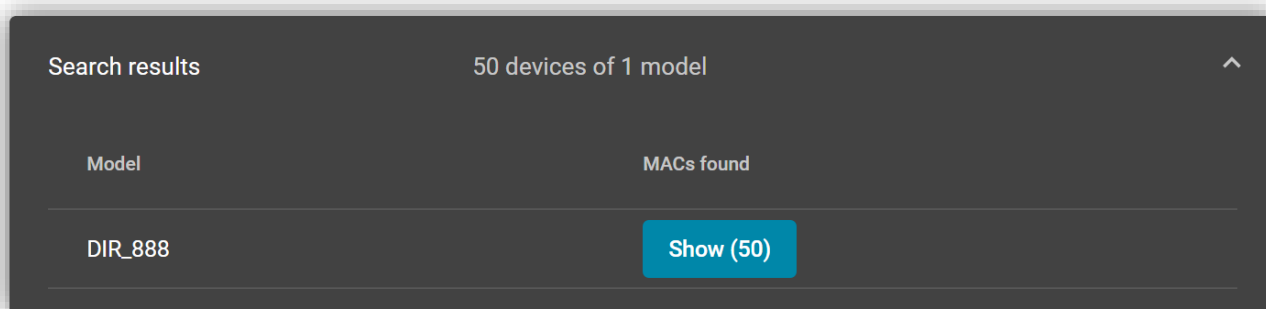


Рисунок 4.19 – Панель с результатами поиска

Описание каждого элемента данной панели представлено в пункте 5.2.4 настоящей работы.

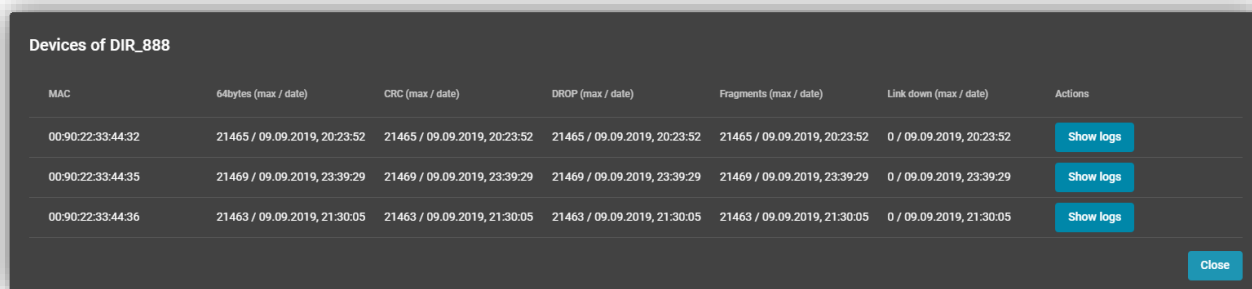
Компонент поиска принимает от главного компонента текущий токен. Принимая данные из полей ввода, формирует запрос на поиск и отправляет на сервер, прикрепляя токен. Далее обрабатывает полученные с сервера данные и выводит в удобном для пользователя виде:

- список полученных результатов делится на группы, названием которых становится модель устройства;
- в середине заголовка панели выводится общее количество найденных устройств и количество групп;
- в первом столбце таблицы выводится наименование группы (название модели);
- в заголовке второго столбца таблицы указан тип поиска;
- кнопка во втором столбце таблицы открывает диалог с более детальной информацией, на кнопке указано количество устройств в группе.

Кнопки, реализованные в панели с результатами, открывают компонент «Диалог с результатами» и передают ему необходимые для отображения данные через сервис «Общий».

4.4.8 Компонент «Диалог с результатами»

Данный компонент вызывается и открывается в качестве диалога (рисунок 4.20). Принимая необходимый набор данных через сервис «Общий», происходит обработка данных для их отображения.



MAC	64bytes (max / date)	CRC (max / date)	DROP (max / date)	Fragments (max / date)	Link down (max / date)	Actions
00:90:22:33:44:32	21465 / 09.09.2019, 20:23:52	21465 / 09.09.2019, 20:23:52	21465 / 09.09.2019, 20:23:52	21465 / 09.09.2019, 20:23:52	0 / 09.09.2019, 20:23:52	Show logs
00:90:22:33:44:35	21469 / 09.09.2019, 23:39:29	21469 / 09.09.2019, 23:39:29	21469 / 09.09.2019, 23:39:29	21469 / 09.09.2019, 23:39:29	0 / 09.09.2019, 23:39:29	Show logs
00:90:22:33:44:36	21463 / 09.09.2019, 21:30:05	21463 / 09.09.2019, 21:30:05	21463 / 09.09.2019, 21:30:05	21463 / 09.09.2019, 21:30:05	0 / 09.09.2019, 21:30:05	Show logs

Рисунок 4.20 – Диалог с результатами поиска, фильтрация по модели

Диалог открывается аналогично DatePicker, затемняя основное окно на фоне и занимает максимальный объем свободного пространства в окне браузера (рисунок 4.21):

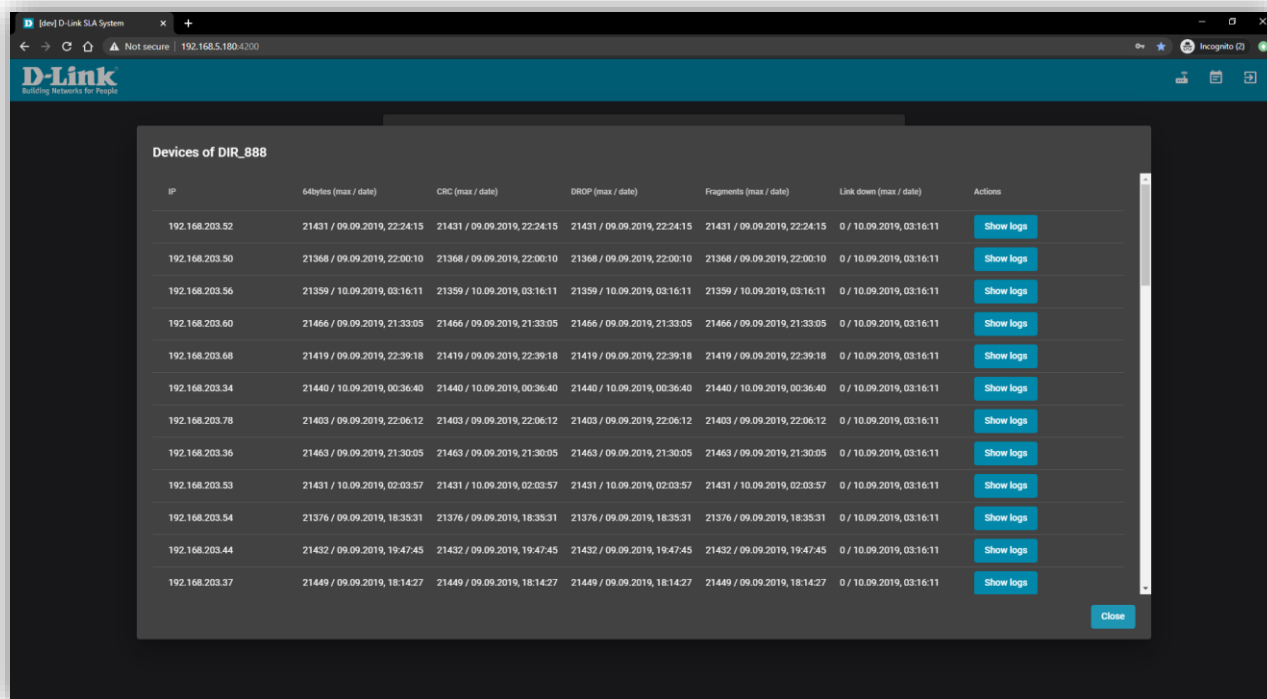


Рисунок 4.21 – Расположение диалога относительно окна браузера

Основная информация в данном диалоге:

- в заголовке отображается название модели;
- в заголовке первого столбца – тип идентификатора;
- столбцы со второго по шестой содержат техническую информацию, фильтрация которой осуществляется с помощью панели фильтров в компоненте поиска. Информация в этих столбцах обработана и представлена в удобном для восприятия человеком формате;
- в последнем столбце расположены кнопки(кнопка) взаимодействия. По умолчанию это кнопка открытия логов конкретного устройства.

4.4.9 Сервис «Общий»

С помощью сервиса компоненты обмениваются данными, а именно имеют «общие» переменные, изменения значений которых влекут за собой изменения значений переменных в других компонентах.

Данный сервис позволяет обмениваться компонентам информацией, необходимой для корректного отображения диалога с результатами и фильтрацией по группам (моделям) устройств.

4.4.10 Реализация мобильной версии

Для того, чтобы систему можно было использовать на мобильных устройствах, во время верстки страниц приложения применялась техника адаптивной верстки – дизайн, который подстраивается (адаптируется) под размер окна, в том числе может происходить перестройка блоков с одного места на другое, или их замена блоками, отображаемыми только при определенном разрешении. Адаптивная верстка пришла на смену идеи создания специальных мобильных версий сайта, «живущих» на отдельных поддоменах. Так, например, поля для ввода IP и MAC, которые в основной версии расположены в ряд, в мобильной версии располагаются в столбец, а панели на странице поиска растягиваются на всю ширину экрана. Пример страниц мобильной версии изображен на рисунке 4.22:

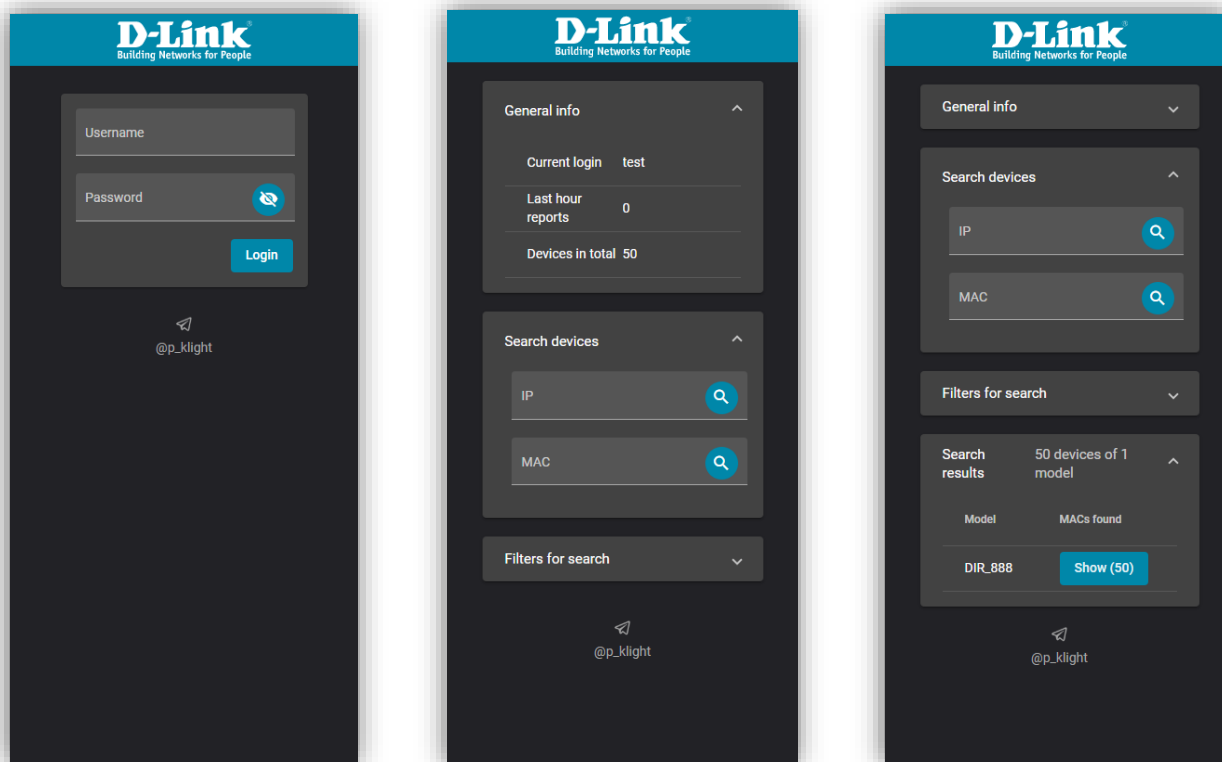


Рисунок 4.22 - – Скриншоты некоторых страниц в мобильной версии системы

В данном разделе была описана реализация программной системы, приведены скриншоты пользовательского интерфейса, описаны механизмы работы с основными страницами веб-приложения.

5 Разработка программной документации

5.1 Руководство системного программиста

5.1.1 Общие сведения о программе

Так как ПО необходима JavaScript runtime, в качестве которой выступает Node.js, скомпилированную версию проекта необходимо размещать на сервере, или хостинге, которым может являться либо localhost, либо домен в мировой сети.

Приложение разработано с использованием таких языков, как TypeScript, HTML, CSS. Для создания приложения использовался Visual Studio Code с набором необходимых инструментов для разработки и утилитой командной строки Angular CLI.

5.1.2 Структура программы

Все файлы, находящиеся в папке с результатом компиляции, необходимы для корректной работы приложения. Приложение взаимодействует с Back End посредством HTTP-запросов.

Перед тем как скомпилировать проект в первый раз, необходимо выполнить консольную команду «npm install» для того, чтобы установить зависимости проекта.

Список зависимостей проекта указаны в *package.json* в корневой директории исходных кодов проекта.

5.1.3 Настройка программы

Программа не требует дополнительной настройки.

В случае, если адрес HTTP сервера или его API будет изменён, ПО требуется скомпилировать и произвести развёртывание заново, предварительно сменив значения констант в исходных кодах программы.

5.1.4 Проверка программы

Запуск программы можно считать успешным, если при открытии приложения отображается форма ввода логина и пароля.

В ходе тестирования была подтверждена корректная работа программы.

5.1.5 Дополнительные возможности программы

Помимо функций, указанных в техническом задании, WEB-интерфейс имеет адаптивную вёрстку, при которой контент на просматриваемой странице автоматически адаптируется под параметры окна, вследствие чего повышается usability, пропадает необходимость перезагрузки страниц приложения при смене размеров окна и, самое главное, UI не будет иметь двух версий – для десктопных браузеров и для телефонов.

5.1.6 Сообщения системному программисту

В программе не предусмотрены сообщения системному программисту.

5.2 Руководство пользователя

5.2.1 Назначение и условия применения программы

Программа предназначена для SLA мониторинга оборудования в обслуживаемой сети устройств, для просмотра статистики о каждом устройстве, детальной информации и отчётов в разрезе времени.

Рассматривая функционал более детально, ПО создано для поиска систематизированных статистических данных на SLA сервере, их обработки и удобного отображения, а также для формирования и отправки на сервер запросов на поиск данных.

Также о назначении программы и о её функционале было упомянуто в пунктах 2.2 и 2.3.

Для запуска приложения подойдет любой современный веб-браузер, например: Chrome 83, IE 11 или Edge 44, минимальные системные требования приложения будут соответствовать минимальным системным требованиям браузера. При открытии главной страницы системы объём загруженных скриптов приблизительно равен 1 Мб. Полный размер приложения при загрузке всех модулей приблизительно составляет 3 Мб. При первой загрузке какого-либо компонента приложения он кэшируется в браузере и дальнейшее обращение к этому компоненту загружает минимальное количество данных с сервера.

5.2.2 Обращение к программе для запуска

Для того чтобы начать работу с системой, необходимо открыть веб-браузер и перейти по адресу, выделенному системе.

Если ПО было запущено на локальном ПК командой «npm start» или «ng serve», то стандартный адрес обращения к программе - **http://localhost:4200/**.

Если ПО развёрнуто не на текущем локальном ПК, то необходимо перейти либо по выданному системным администратором адресу, либо по адресу, указанному в параметрах «npm start» или «ng serve».

5.2.3 Входные и выходные данные

В качестве входных данных используются:

- данные авторизации (логин и пароль);
- основные данные для поиска устройств (IP и MAC);
- многочисленные фильтры для поиска устройств;
- данные для поиска отчётов, такие как дата, время и идентификаторы устройств.

Выходные данные:

- таблицы с информацией;
- списки устройств;
- графики с техническими данными;
- информация об устройствах;
- отчёты.

5.2.4 Работа с программой

Некоторые приведённые в данном пункте скриншоты, изображения, рисунки и примеры были подвергнуты масштабированию ввиду повышения читабельности настоящей пояснительной записки к данной работе, так как минимальное разрешение монитора, при котором использование ПО считается комфортабельным, превышает ширину печатной строки листа А4, на котором должен уместиться

скриншот. Более того, благодаря адаптивной вёрстке web-приложения оператор имеет возможность менять размеры окна web-браузера в разумных пределах, а также менять разрешение экрана без потери функциональности. Более подробно про адаптивную вёрстку изложено в пункте 4.4.10 настоящей работы.

Если пользователь открывает приложение в первый раз или его токен авторизации невалидный (отсутствует или истек срок действия), то он попадает на страницу авторизации (рисунок 5.1).

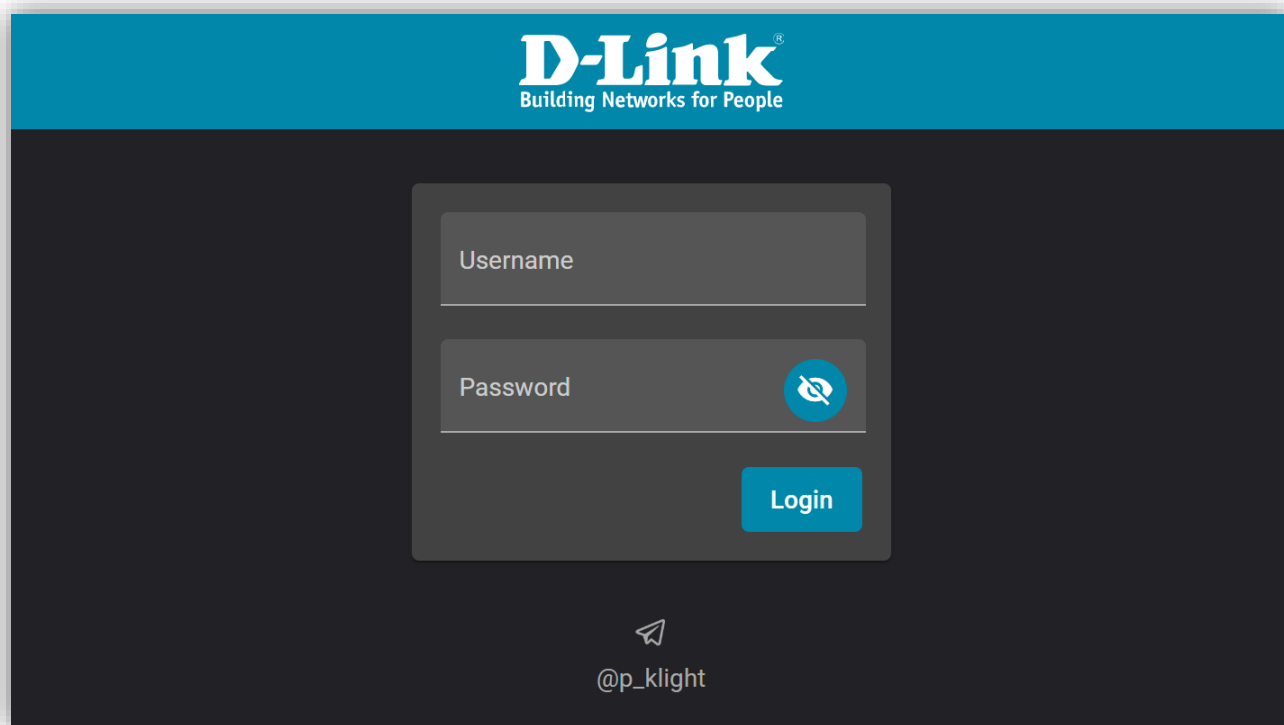


Рисунок 5.1 – Страница авторизации

Для продолжения работы с системой пользователю необходимо ввести логин и пароль в соответствующие поля ввода, затем нажать кнопку входа «Login». В целях удостоверения корректного введения логина имеется кнопка отмены скрытия вводимых данных в поле ввода пароля. Если авторизация прошла успешно, то пользователь будет перенаправлен на главную страницу, которой является страница поиска. Большинство кнопок системы, изображенных в виде иконок, имеет подсказки, посмотреть которые можно наведя курсор мыши на соответствующую

кнопку. Более подробно о подсказках изложено в пункте 5.2.5, а пример всплывающей подсказки приведён на рисунке 4.7 настоящей работы.

После успешной авторизации пользователя форма авторизации пропадает, а в правой части экрана появляются три кнопки (рисунок 5.2):

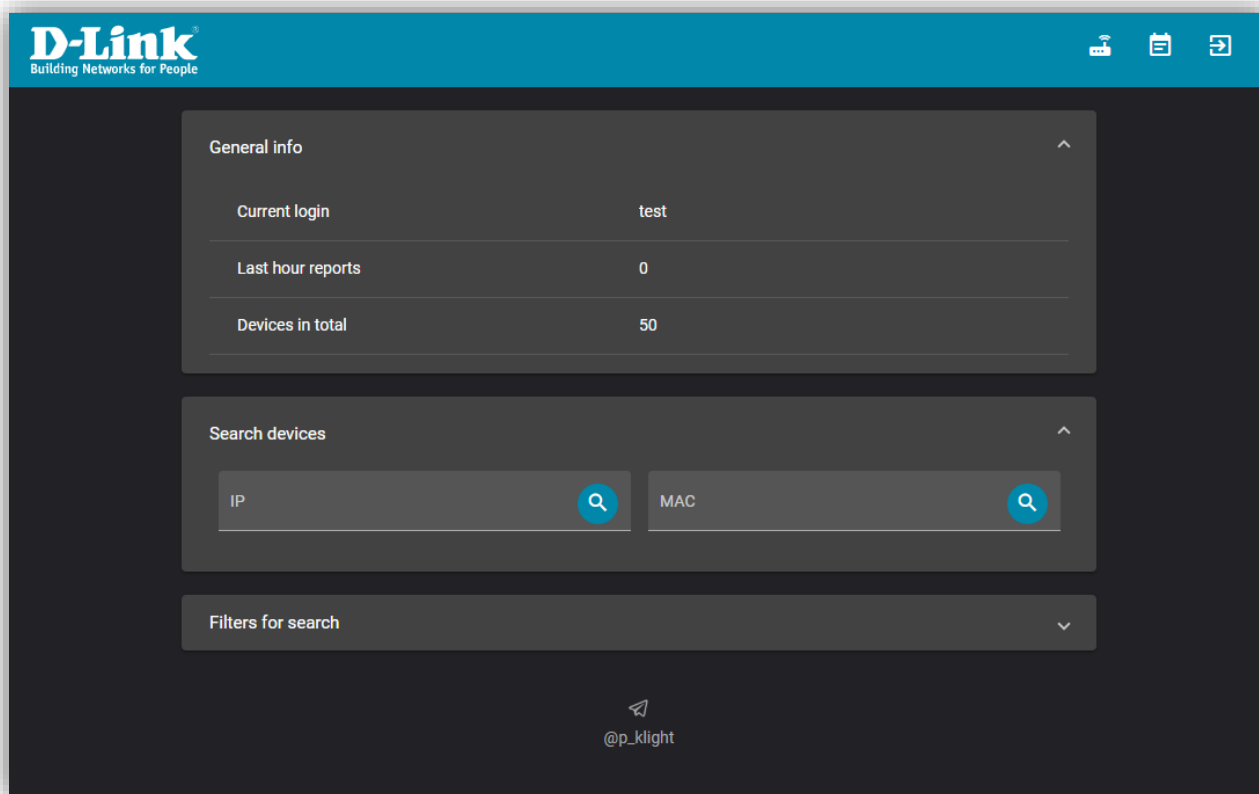


Рисунок 5.2 – Страница поиска, открывающаяся после успешной регистрации

Представленные на рисунке 5.2 кнопки отвечают за три основных действия, которые должны быть всегда в зоне видимости и быстрого доступа для авторизованного пользователя (слева направо):

- кнопка перехода на страницу поиска по устройствам, она же главная страница. Кнопка становится доступной для нажатия только при нахождении на странице, отличной от страницы поиска по устройствам;
- кнопка открытия поиска отчётов по датам. Становится доступной при нахождении на любой другой странице;

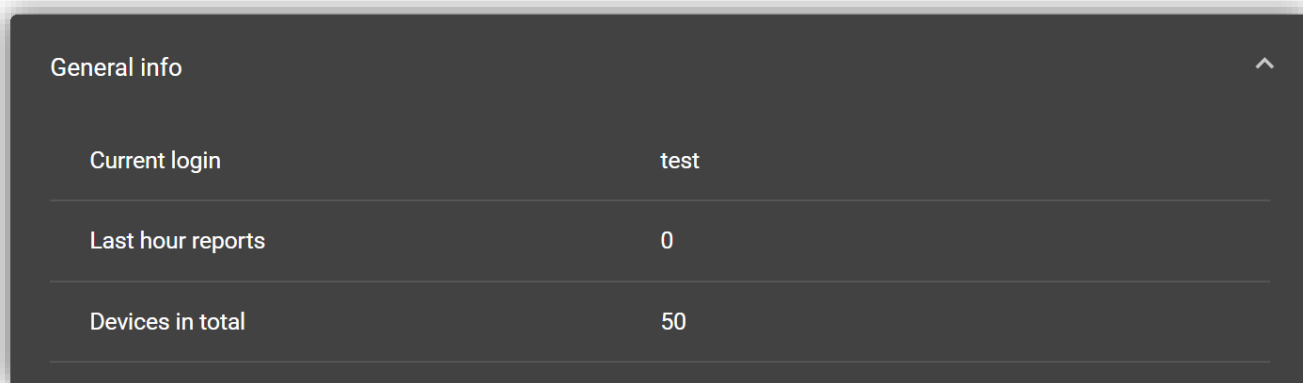
- кнопка выхода из системы. Возвращает пользователя на страницу авторизации.

Все три кнопки имеют всплывающие подсказки при наведении курсора, пример такой подсказки представлен на рисунке 4.7 настоящей работы.

В теле главной страницы представлены панели расширения. Некоторые из них открыты по умолчанию (рисунок 5.2), а некоторые появляются только после выполнения определённых действий. Каждую панель можно свернуть, или развернуть для удобства группировки информации на экране. Свернуть/развернуть панель можно нажатием на заголовок панели.

Первая панель расширения – «General info», или же «Основная информация» (рисунок 5.3). В теле панели представлена краткая сводка:

- текущий логин пользователя;
- количество отчётов за последний час (отсчитывается с момента загрузки страницы);
- общее количество устройств в системе.



General info		^
Current login	test	
Last hour reports	0	
Devices in total	50	

Рисунок 5.3 - Панель расширения «General info»

Сразу после панели со сводкой основной информации идёт панель поиска устройств по IP или MAC. Так как это два разных вида поиска, предусмотрено две кнопки (по одной в каждом поле) для осуществления каждого вида поиска отдельно.

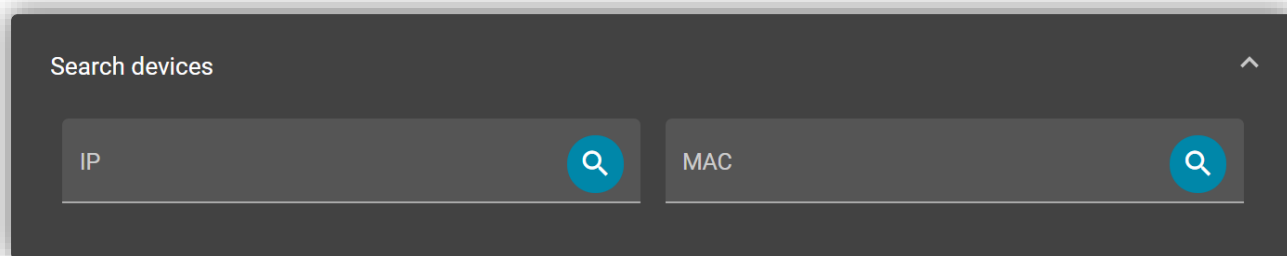


Рисунок 5.4 – Панель поиска по IP или MAC

В случае, если требуется указать дополнительные параметры поиска, пользователь может указать необходимые фильтры. По умолчанию панель с фильтрами свёрнута (рисунок 5.5) и разворачивается с помощью нажатия на заголовок панели (рисунок 5.6):

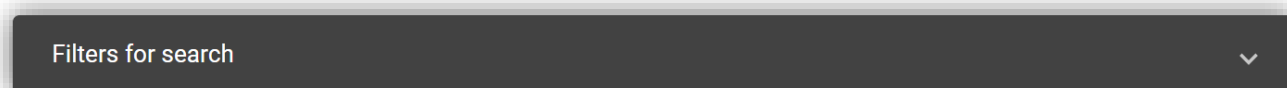


Рисунок 5.5 – Свёрнутая панель фильтров поиска

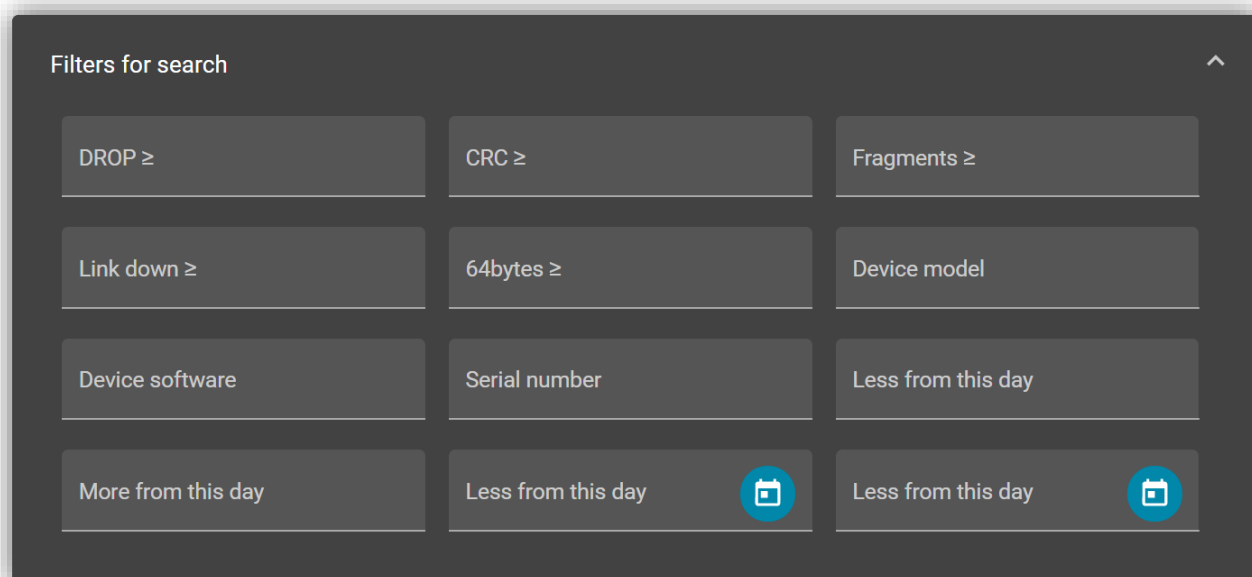


Рисунок 5.6 – Панель фильтров поиска

Чтобы быстро ввести необходимую дату в последние два поля, имеется кнопка с иконкой календаря, позволяющая открыть интерактивный календарь. (рисунок 5.7):

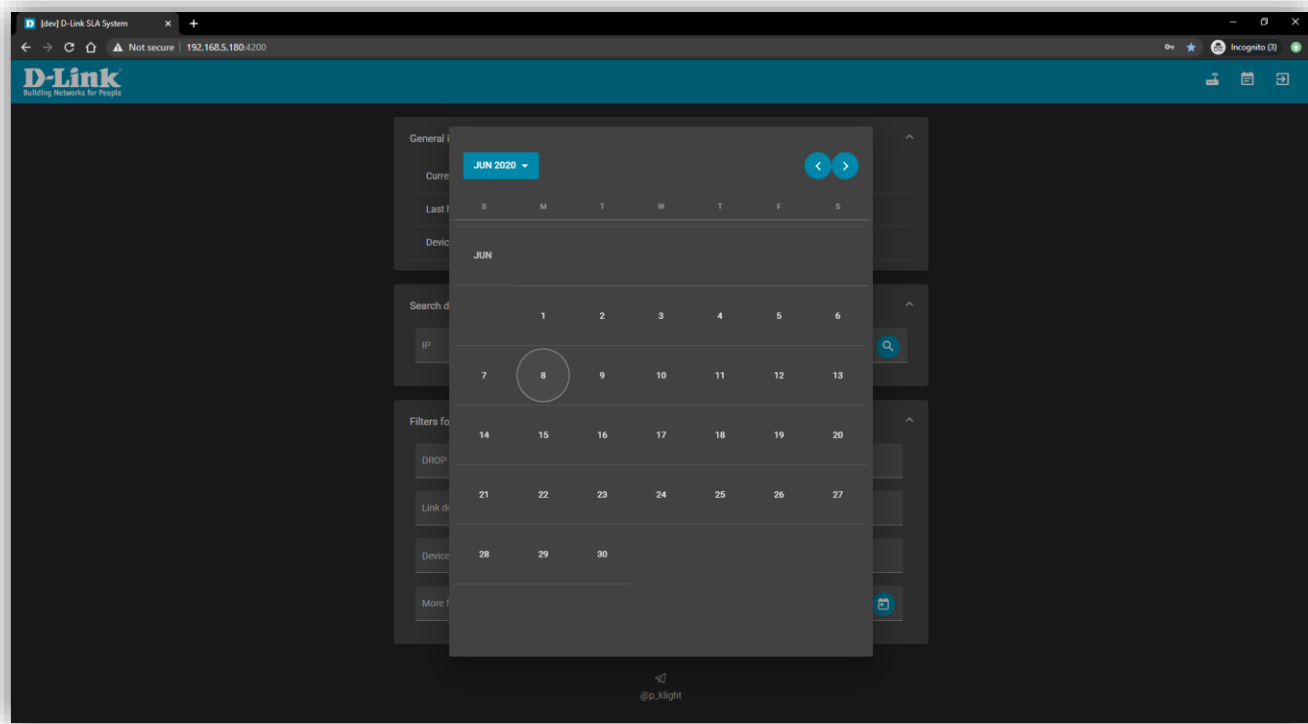


Рисунок 5.7 – Полноэкранный инструмент выбора даты.

После ввода всех параметров поиска следует нажать на кнопку поиска и дождаться окончания загрузки. Если результат поиска оказался нулевым, система уведомит оператора с помощью сообщений (уведомлений). Более подробно получение сообщений описано в пункте 5.2.5. Если результат поиска ненулевой, на экране появляется панель с результатами поиска, представленная на рисунке 5.8:

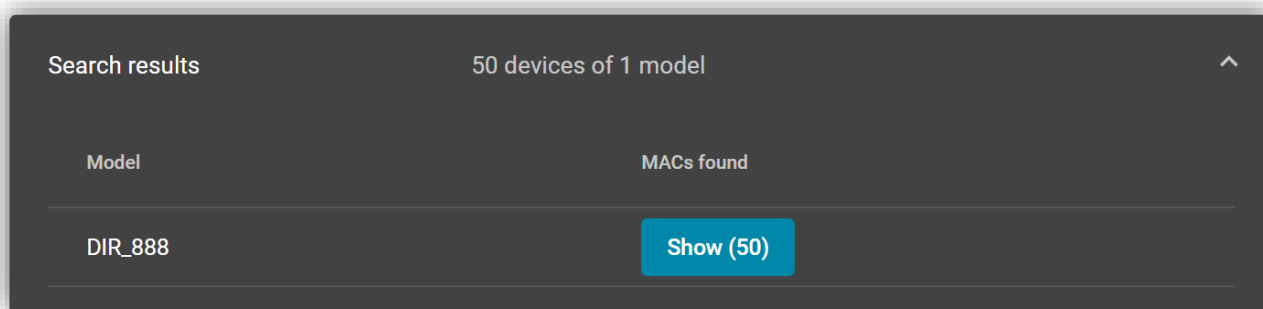


Рисунок 5.8 – Панель с результатами поиска

Основная информация на панели (рисунок 5.9):

1. Количество найденных устройств;
2. Количество наименований различных моделей в найденных устройствах;
3. Название текущей модели групп устройств, 1 строка = 1 модель;
4. Тип поиска (IP или MAC);
5. Кнопка открытия таблицы с результатами, на которой указано количество устройств.

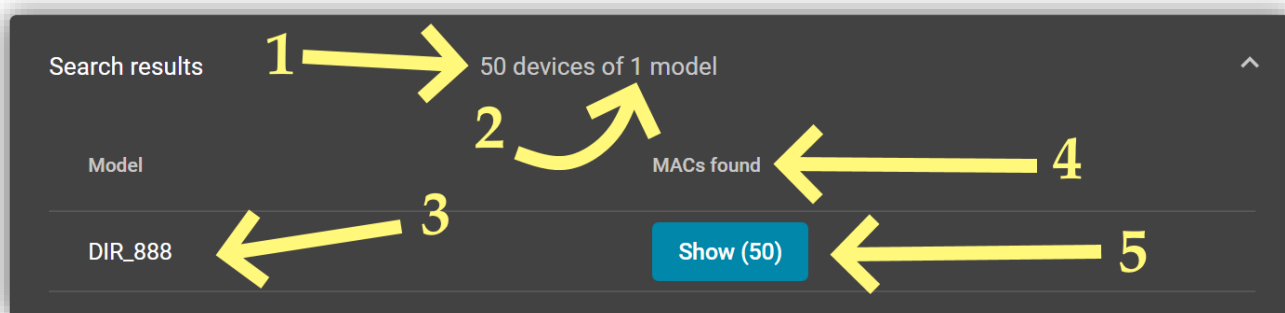


Рисунок 5.9 – Схема расположения данных на панели с результатами поиска

После нажатия на кнопку открытия таблицы на экране пользователя открывается в полный экран таблица с результатами поиска для конкретной модели устройств (рисунок 5.10):

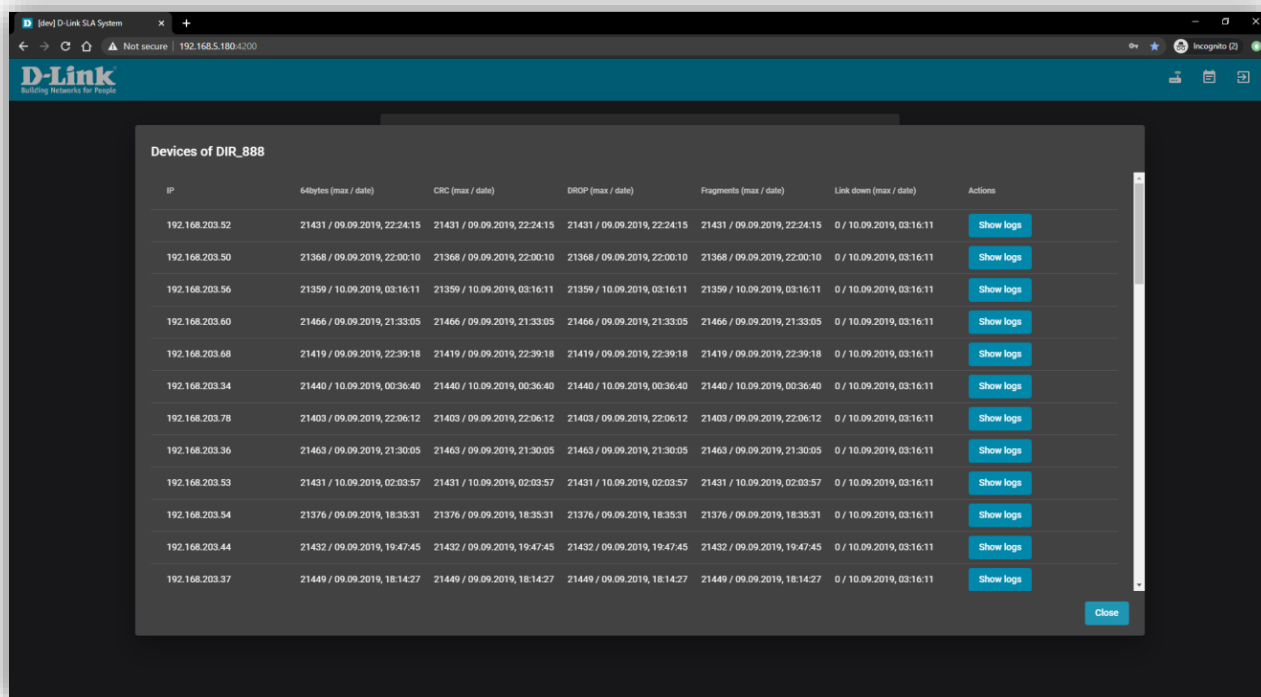


Рисунок 5.10 – Полноэкранный режим просмотра полученной информации

Описание информации в таблице (рисунок 5.11):

1. Наименование группы устройств (по модели);
2. Тип идентификатора (IP или MAC) и сам идентификатор;
3. 64bytes (максимальное значение / дата);
4. CRC (максимальное значение / дата);
5. DROP (максимальное значение / дата);
6. Fragments (максимальное значение / дата);
7. Link down (максимальное значение / дата);
8. Столбец с кнопками взаимодействия с устройством (например, просмотр логов);
9. Кнопка закрытия окна просмотра.

MAC	64bytes (max / date)	CRC (max / date)	DROP (max / date)	Fragments (max / date)	Link down (max / date)	Actions
00:90:22:33:44:32	21465 / 09.09.2019, 20:23:52	21465 / 09.09.2019, 20:23:52	21465 / 09.09.2019, 20:23:52	21465 / 09.09.2019, 20:23:52	0 / 09.09.2019, 20:23:52	Show logs
00:90:22:33:44:35	21469 / 09.09.2019, 23:39:29	21469 / 09.09.2019, 23:39:29	21469 / 09.09.2019, 23:39:29	21469 / 09.09.2019, 23:39:29	0 / 09.09.2019, 23:39:29	Show logs
00:90:22:33:44:36	21463 / 09.09.2019, 21:30:05	21463 / 09.09.2019, 21:30:05	21463 / 09.09.2019, 21:30:05	21463 / 09.09.2019, 21:30:05	0 / 09.09.2019, 21:30:05	Show logs

Рисунок 5.11 – Схема расположения данных в окне с результатами

Следующие далее окна выбора даты логов и страницы с логами не нуждаются в дополнительном описании, так как интерфейс этих разделов является интуитивно понятным и аналогичен описанному ранее.

Предполагается, что целевой пользователь данного ПО будет достаточно глубоко погружён в предметную область SLA мониторинга. Без предварительного знакомства с основными терминами, параметрами поиска, характеристиками устройств и внутренними принципами работы SLA сервисов пользование данным ПО не предоставляется возможным. Целевой оператор должен обладать определённым набором знаний в конкретной предметной области для обращения с ПО, в противном случае результат взаимодействия с программным обеспечением будет нулевым.

5.2.5 Сообщения оператору

В процессе работы с системой оператор, или, другими словами, пользователь может получать два типа сообщений: уведомления и подтверждение действия.

Так как ПО отправляет данные на сервер всего один раз – во время авторизации, а всё остальное время получает данные с сервера, подтверждения действий не требуются при выполняемых операциях, так как не оставляют изменений на сервере. Подтверждение может требоваться в ситуациях, которые требуют особого внимания – например, подтверждение при выходе из системы (рисунок 5.12)

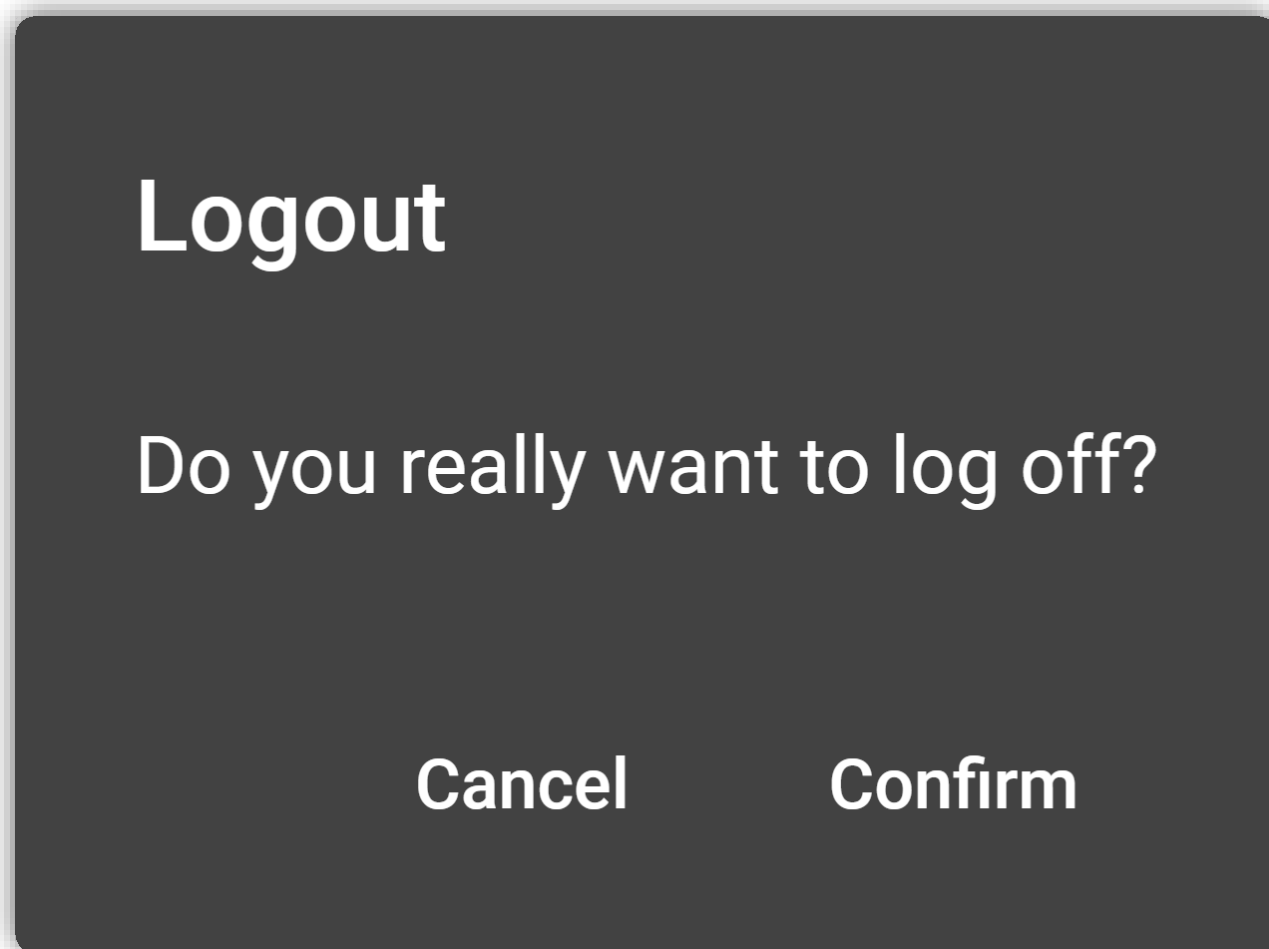


Рисунок 5.12 – Подтверждение выхода из системы.

В системе предусмотрены уведомления пользователя о результатах поиска, при которых нет совпадений указанных фильтров с данными на сервере. Другими словами, «По вашему запросу ничего не найдено» (рисунок 5.13):

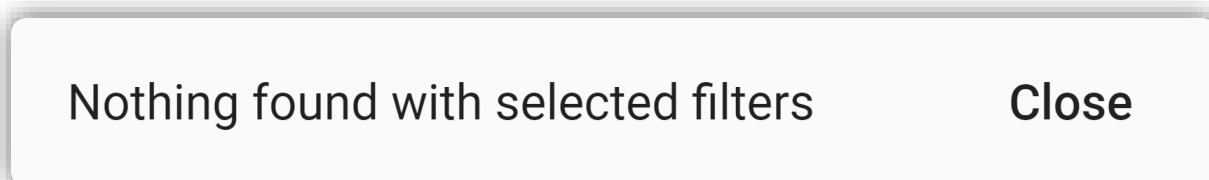


Рисунок 5.13 – Пример уведомления о нулевых результатах поиска

Пользователь может скрыть уведомление самостоятельно, либо оно будет скрыто автоматически через короткое время.

В некоторых версиях ПО сообщения пользователю реализованы стандартными средствами браузера. Визуальное оформление может отличаться, но текст будет одинаков в окнах сообщения любого браузера. На рисунке 5.14 представлен пример такого сообщения оператору в браузере Google Chrome с использованием тёмной темы:

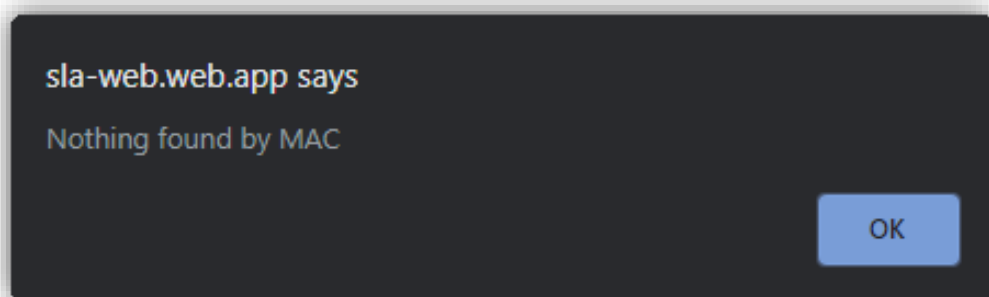


Рисунок 5.14 – Сообщения пользователю стандартными средствами браузера

Ещё один вид взаимодействия с пользователем, который можно отнести к сообщениям оператору – подсказки при наведении на определённые объекты WEB-интерфейса. Пример таких всплывающих подсказок приведён на рисунке 4.7 настоящей работы.

6 Тестирование программного обеспечения

Одним из ключевых этапов в разработки любого программного обеспечения является его тестирование – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Тестирование ПО будет осуществляться с помощью сценариев тестирования (Test Case) – минимальной единицы тестирования, которая состоит из набора необходимых в реализации шагов, ожидаемого результата и его соответствия полученному. Необходимо исследовать поведение программы в зависимости от данных, поступающих на её входы и действий, совершаемых пользователем. Подробное описание каждого Test Case и результат прохождения описан в таблице 1:

Таблица 1 – Сценарии тестирования

Номер	Описание	Сценарий	Ожидаемый результат	Статус
1	Проверка открытия главной страницы после успешной авторизации.	1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти».	Открытие главной страницы	Успешно

Номер	Описание	Сценарий	Ожидаемый результат	Статус
2	Проверка повторного входа в систему после выхода из системы	1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Дождаться загрузки главной страницы. 6. Нажать кнопку «Выйти» 7. Повторить пункты 3,4	Открытие главной страницы	Успешно
3	Проверка сворачивания панели расширения	1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Дождаться загрузки главной страницы. 6. Нажать на заголовок любой представленной на странице панели.	Выбранная панель сворачивается	Успешно

Номер	Описание	Сценарий	Ожидаемый результат	Статус
4	Проверка корректного отображения информации на панели «General info»	1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Дождаться загрузки главной страницы.	Во втором столбце таблицы будут отображаться ненулевые значения, полученные с сервера	Успешно
5	Проверка успешного поиска устройств	1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Дождаться загрузки главной страницы. 6. Ввести в поле IP или MAC значение существующих идентификаторов устройств 7. Нажать на кнопку поиска в этом же поле	Снизу от панели фильтров появляется панель с результатами поиска по устройствам	Успешно

Номер	Описание	Сценарий	Ожидаемый результат	Статус
6	Проверка отсутствия результатов поиска	1. Запустить браузер в режиме «инкогнито». 2. Перейти по адресу приложения. 3. В форму авторизации ввести пару логин-пароль существующего пользователя. 4. Нажать кнопку «Войти». 5. Дождаться загрузки главной страницы. 6. Ввести в поле IP или MAC значение несуществующих идентификаторов устройств. 7. Нажать на кнопку поиска в этом же поле.	Панель с результатами поиска пропадёт (если отображалась до этого), появится сообщение об ошибке	Успешно

Протестировав разработанную систему, можно сделать вывод о ее работоспособности и о корректном выполнении предъявленных требований к системе.

ЗАКЛЮЧЕНИЕ

Сетевые технологии – важная сфера деятельности, которую можно назвать жизненно необходимой в условиях современного мира. Практически ни одна компания не обходится без пользования сетью Internet, телефонных звонков, систем видеонаблюдения и прочего, а ведь работоспособность этих сложных систем поддерживается сетевыми администраторами и сетевиками – людьми, ответственными за бесперебойное функционирование всех систем коммуникаций.

В ходе данной работы были рассмотрены основные проблемы, присущие этой области, исследованы варианты их решения от ведущих мировых компаний. Анализ предметной области позволил выявить существенные недостатки, основные преимущества и сформировать требования к разработанному ПО. Стек выбранных технологий позволил сократить временные и ресурсные затраты на осуществление разработки, при этом позволяя ей оставаться актуальной.

Полученное на выходе web-приложение полностью удовлетворяет всем потребностям пользователя, на которого ориентирована разработка, решает все поставленные задачи и является актуальным современным программным средством.

В ходе тестирования были доказаны:

- работоспособность;
- достоверность;
- отказоустойчивость;
- адаптивность;
- кроссплатформенность.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Богданова, Е.А. Технологии защиты информации в компьютерных сетях. Межсетевые экраны и интернет-маршрутизаторы: учебное пособие / Е.А. Богданова и др. – М.: Национальный Открытый Университет «ИНТУИТ», 2013. – 743 с.
2. Jimmy Desai Service Level Agreements. A Legal and Practical Guide – IT GOVERNANCE PUBLISHING, 2010. – 122 с.
3. John Lee K., Ron Ben-Natan Integrating Service Level Agreements - John Wiley & Sons Limited, 2002. - 466 с.
4. Хабрахабр [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru>
5. Stack Overflow [Электронный ресурс]. – Режим доступа: <https://stackoverflow.com>
6. Blogspot [Электронный ресурс]. – Режим доступа: <https://blogspot.com>
7. PAESSLER [Электронный ресурс]. – Режим доступа: <https://www.paessler.com/>
8. Angular Material UI component library [Электронный ресурс]. – Режим доступа: <https://material.angular.io/>
9. GitHub – Avendattor / CORS Anywhere [Электронный ресурс]. – Режим доступа: <https://github.com/Avendattor/cors-anywhere>
10. Cloud Application Platform | Heroku [Электронный ресурс]. – Режим доступа: <https://www.heroku.com/>
11. API that enables cross-origin requests to anywhere [Электронный ресурс]. – Режим доступа: <https://stark-depths-37590.herokuapp.com/>

ПРИЛОЖЕНИЕ А. ИСХОДНЫЕ КОДЫ ПРОГРАММНЫХ ЭЛЕМЕНТОВ

index.html

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>[dev] D-Link SLA System</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="./assets/favicon.ico">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons&display=block"
rel="stylesheet">
  <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500&display=swa
p" rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="styleshe
et">
</head>

<body class="mat-typography">
  <app-root></app-root>
</body>

</html>
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule, HttpClientJsonpModule } from '@angular/common/http';
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatFormFieldModule } from '@angular/material/form-field';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatTableModule } from '@angular/material/table';
import { MatButtonModule } from '@angular/material/button';
import { MatExpansionModule } from '@angular/material/expansion';
import { MatInputModule } from '@angular/material/input';
import { MatCardModule } from '@angular/material/card';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatIconModule } from '@angular/material/icon';
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { MatProgressBarModule } from '@angular/material/progress-bar';
import { MatNativeDateModule } from '@angular/material/core';
import { MatDialogModule } from '@angular/material/dialog';

import { AppRoutingModuleModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { TopBarComponent } from './top-bar/top-bar.component';
import { LoginFormComponent } from './login-form/login-form.component';
import { FooterComponent } from './footer/footer.component';
import { MainTopComponent } from './main-top/main-top.component';
```

```
import { SearchComponent } from './search/search.component';
import { SharedService } from './search/shared/shared.service';
import { ResultDialogComponent } from './search/result-dialog/result-
dialog.component';
```

```
@NgModule({
  declarations: [
    AppComponent,
    TopBarComponent,
    LoginFormComponent,
    FooterComponent,
    MainTopComponent,
    SearchComponent,
    ResultDialogComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    HttpClientModule,
    HttpClientModule,
    MatDatepickerModule,
    MatFormFieldModule,
    BrowserAnimationsModule,
    MatTableModule,
    MatButtonModule,
    MatExpansionModule,
    MatInputModule,
    MatCardModule,
    MatToolbarModule,
    MatIconModule,
    MatProgressSpinnerModule,
    MatProgressBarModule,
    MatNativeDateModule,
    MatDialogModule,
  ],
  providers: [SharedService],
  bootstrap: [AppComponent]
})
export class AppModule {

}
```

app.component.ts

```
import { Input, Component } from '@angular/core';
import { HttpClient, HttpHeaders, HttpClientModule } from '@a
ngular/common/http';
import { catchError } from 'rxjs/operators';
```

```

import { MatDatepickerModule } from '@angular/material/datepicker';

// import { Md5 } from 'angular-md5'

import { LoginFormComponent } from './login-form/login-form.component';
//import { LoginFormComponent } from './login-form/login-form.component';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  title = 'SLA';

  isDebugActive = false;

  // // using proxy cause of CORS policy restrictions

  // // free, 200 requests per 60 minutes limit
  // proxyURL = "https://cors-anywhere.herokuapp.com/";

  // // personally builded & hosted, no limit
  proxyURL = "https://stark-depths-37590.herokuapp.com/";

  testURL = "https://postman-echo.com/post/";
  currentSLAURL = "http://mysla.dlink.ru/api/";

  firmwareURL = "info/firmware";

  currentToken: string;
  isTokenReceived: boolean = false;

  receivedJSON: string = '';

  curLogin: string = '';

  getFirmware: object;
  isFirmwareDataReceived: boolean;
  firmwareData;

  // is Authorization complete
  isAuth: boolean;
  // login-form hide after Auth
  hideLogin = false;

```

```

// gets data from login component
receiveToken($event) {
    this.currentToken = $event;
    //console.log(this.currentToken);
    if (this.currentToken == $event || this.currentToken != '') {
        this.isTokenReceived = true;
    }
}

receiveUsername($event) {
    this.curLogin = $event;
}

receiveIsAuth($event) {
    this.isAuth = $event;

    if (this.isAuth = true && this.isTokenReceived == true && this.curLogin != '') {
        this.loadMainPage();
    }
}

hideLoginForm() {
    this.hideLogin = !this.hideLogin;
}

constructor(private http: HttpClient) { }

private getFirmwareData() {
    //console.log(`CurToken = ` + this.currentToken);
    this.http.get(this.proxyURL + this.currentSLAURL + this.firmwareURL, {
        headers: new HttpHeaders({
            'Content-Type': 'application/json',
            'token': this.currentToken
        })
    }).toPromise().then((data: any) => {
        //console.log(data);
        // console.log(data.token);
        this.firmwareData = data;
        // console.log(this.currentToken);
        // this.receivedJSON = JSON.stringify(data.receivedJSON);
        if (this.firmwareData != '') {
            this.isFirmwareDataReceived = true;
        }
    });
}

```



```

}

// hides login-form and loads some data
private loadMainPage() {

    this.hideLoginForm();
    //this.getFirmwareData();
}
}

```

app.component.html

```

<div>
    <app-top-bar></app-top-bar>
</div>
<div *ngIf="!hideLogin">
    <app-login-
form (sendTokenEvent)="receiveToken($event)" (loadMainPageEvent)="receiveIsAuth($eve
nt)"
    (sendUsernameEvent)="receiveUsername($event)" [currentSLAURL]="currentSLAURL" [p
roxyURL]="proxyURL">
    </app-login-form>

</div>
<div *ngIf="hideLogin">

    <div>
        <app-main-
top [isAuth]="isAuth" [currentToken]="currentToken" [currentSLAURL]="currentSLAURL"
[proxyURL]="proxyURL"
        [curLogin]="curLogin">
            <div>
                <h6>
                    current login: "{{ curLogin }}"
                </h6>
            </div>
        </app-main-top>
    </div>
    <div *ngIf="hideLogin">
        <app-
search [isAuth]="isAuth" [currentToken]="currentToken" [currentSLAURL]="currentSLAUR
L" [proxyURL]="proxyURL">
        </app-search>
    </div>
    <div *ngIf="isDebugActive">
        <div *ngIf="hideLogin" class="center-wide">
            <h3>
                [debug]
            </h3>
            <p>
                Is token received: "{{ isTokenReceived }}"
            </p>

```

```

        <p>
            Firmware data Received: "{{ isFirmwareDataReceived }}"
        </p>
    </div>
</div>

</div>
<div>
    <app-footer></app-footer>
</div>

```

top-bar.component.ts

```

import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-top-bar',
  templateUrl: './top-bar.component.html',
  styleUrls: ['./top-bar.component.css']
})
export class TopBarComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}

```

top-bar.component.html

```

<div class="blue-dlink">

</div>

<mat-toolbar color="primary">
    <div id="dlink-logo">

    </div>
</mat-toolbar>

```

top-bar.component.css

```

#dlink-logo{
    background: transparent url(../../assets/logo_white.svg) no-
repeat scroll center / auto 45px;
    height: inherit;
    width: 150px;
    margin: auto;
    -webkit-box-align: center;
}

```

```

.top-bar{
  width: 100%;
  background-color: #0087A9;
  height: 70px;
}

.blue-dlink{
  width: 100%;
  background-color: #0087A9 !important;
  box-shadow:
    0 2px 2px 0 rgba(0,0,0,0.14),
    0 3px 1px -2px rgba(0,0,0,0.12),
    0 1px 5px 0 rgba(0,0,0,0.2);
}

```

login-form.component.ts

```

import { Component, OnInit, EventEmitter, Output, Input } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Md5 } from 'ts-md5/dist/md5';
import { HttpClient, HttpHeaders } from '@angular/common/http';

@Component({
  selector: 'app-login-form',
  templateUrl: './login-form.component.html',
  styleUrls: ['./login-form.component.css']
})

export class LoginFormComponent implements OnInit {

  inputUserName: string = '';
  inputPassword: string;
  hashedPassword;
  hidelogin = true;

  loginURL = "login";
  @Input() proxyURL: string = '';
  @Input() currentSLAURL: string = '';

  postLoginData: object;

  currentToken: string = '';

  isTokenReceived: boolean;
  loginProcess = false;

  // is Authorization complete
  isAuth = false;

```

```

// loginForm: FormGroup;

// @Output() loginEvent = new EventEmitter<any>();
// @Output() passwordEvent = new EventEmitter<any>();
@Output() sendTokenEvent = new EventEmitter<any>();
@Output() loadMainPageEvent = new EventEmitter<any>();
@Output() sendUsernameEvent = new EventEmitter<any>();

constructor(
  // private formBuilder: FormBuilder,
  private http: HttpClient
) { }

ngOnInit(): void { }

inputTestLogin(inputLogin, inputPassword) {
  this.inputUserName = inputLogin;
  this.inputPassword = inputPassword;
}

usernameInput(event: any) {
  const value = event.target.value
  this.inputUserName = value
  // console.log(this.inputUserName)
}

passwordInput(event: any) {
  const value = event.target.value
  this.inputPassword = value
  // console.log(this.inputPassword)
}

login() {
  this.loginProcess = true;
  // this.loginEvent.emit([this.inputUserName, this.inputPassword]);

  this.hashedPassword = Md5.hashStr(this.inputPassword);
  this.postLoginData = {
    username: this.inputUserName,
    password: this.hashedPassword
  };

  if (this.currentToken == '') {
    this.authToken();
  }
}

```

```

    // else {}

    // if (this.currentToken != '') {}
}

loadMainPage() {
    this.sendTokenEvent.emit(this.currentToken);
    this.sendUsernameEvent.emit(this.inputUserName);
    this.loadMainPageEvent.emit(this.isAuth);

}

// posts login data to server, gets token, loads main page
private authToken() {
    this.http.post(this.proxyURL + this.currentSLAURL + this.loginURL, this.postLoginData, {
        headers: new HttpHeaders({
            'Content-Type': 'application/json',
        })
    }).toPromise().then((data: any) => {
        // console.log(data);
        // console.log(data.token);

        // console.log(this.currentToken);
        // this.receivedJSON = JSON.stringify(data.receivedJSON);

        if (this.currentToken == data.token) {
            this.isTokenReceived = true;
            this.isAuth = true;
        }
        if (this.currentToken != data.token) {
            this.currentToken = data.token;
            this.isTokenReceived = true;
            this.isAuth = true;
        }

        this.loadMainPage();
    });
}
}

```

login-form.component.html

```

<div class="pad30 widthauto login-div">
    <mat-card>

```

```

<div>
  <mat-form-field appearance="fill" class="widthauto w236pxim">
    <mat-label>Username</mat-label>
    <input matInput (input)="usernameInput($event)">
    <!-- <button mat-icon-
button matSuffix (click)="inputTestLogin('test','test')">
      <mat-icon>account_circle</mat-icon>
    </button> -->
  </mat-form-field>
</div>
<div>
  <mat-form-field appearance="fill">
    <mat-label>Password</mat-label>
    <input matInput [type]="hidelogin ? 'password' : 'text'" (input)="pa
sswordInput($event)">
    <button mat-icon-
button matSuffix (click)="hidelogin = !hidelogin" [attr.aria-
label]=" 'Hide password' "
      [attr.aria-pressed]="hidelogin">
    <mat-icon>{{hidelogin ? 'visibility_off' : 'visibility'}}</mat-
icon>
    </button>

  </mat-form-field>
</div>
<div *ngIf="loginProcess">
  <mat-progress-bar mode="indeterminate"></mat-progress-bar>
</div>

<div class="right">
  <button *ngIf="!loginProcess" (click)="logIn()" mat-
button color="primary">
    <!-- <span *ngIf="loginProcess"></span> -->
    Login
  </button>
</div>
</mat-card>
</div>

```

login-form.component.css

```

.centerLogin {
  margin: auto;
  width: 90%;
  max-width: 400px;
  background-color: #444446;

  border: 1px #cccccc;

  padding: 25px;

```

```

margin-top: 20px;

margin-bottom: 20px;

border-radius: 4px;
box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.14),
            0 3px 1px -2px rgba(0, 0, 0, 0.12), 0 1px 5px 0 rgba(0, 0, 0, 0.2);
padding-bottom: 9px;
}

label{
    color: #fff!important;
}

/* input{
    background-color: #444444 !important;
    color: #fff!important;
} */

.w236pxim {
    width:236px !important;
}

```

main-top.component.ts

```

import { Component, OnInit, Input } from '@angular/core';
import { HttpClient, HttpHeaders, HttpClientModule, HttpClientModule } from '@angular/common/http';
import { catchError } from 'rxjs/operators';

export interface GeneralTable {
    name: string;
    data: any;
}

var GENERAL_TABLE_DATA: GeneralTable[] = [
    { name: 'Current login', data: '' },
    { name: 'Last hour reports', data: '' },
    { name: 'Devices in total', data: '' },
];

@Component({
    selector: 'app-main-top',
    templateUrl: './main-top.component.html',
    styleUrls: ['./main-top.component.css']
})
export class MainTopComponent implements OnInit {

```

```

receivedJSON: object;
// stringifiedJSON;
totalDevices: string = '';
totalDevicesURL = "/info/totalDevices";

// data for main table
generalTableDataToShow;
displayedColumnsGeneralInfo: string[] = [
    'name',
    'data'
];

timestamp: number;
reportsLastHour = '';
reportsLastHourURL: string = "logs/lasthour/";

// is Authorization complete
@Input() isAuth: boolean = false;

@Input() curLogin: string = '';

// data for HTTP GET
@Input() currentToken;
@Input() proxyURL = '';
@Input() currentSLAURL = '';

panelOpenState = false;

getTotalDevices() {
    if (this.isAuth == true) {
        //console.log(this.currentToken);
        this.http.get(this.proxyURL + this.currentSLAURL + this.totalDevicesURL, {
            headers: new HttpHeaders({
                'Content-Type': 'application/json',
                'token': this.currentToken
            })
        }).toPromise().then((data: any) => {
            //console.log("TD");
            this.totalDevices = data.length;
            // this.stringifiedJSON = JSON.stringify(data.receivedJSON);
            GENERAL_TABLE_DATA[2].data = data.length;
        });
    }
}

getLastHourReports() {
    if (this.isAuth == true) {
        this.makeTimeStamp();
        this.http.get(this.proxyURL + this.currentSLAURL + this.reportsLastHourURL + t
his.timestamp, {
            headers: new HttpHeaders({

```



```

        'Content-Type': 'application/json',
        'token': this.currentToken
    })
    }).toPromise().then((data: any) => {
        this.reportsLastHour = data;
        // this.stringifiedJSON = JSON.stringify(data.receivedJSON);
        GENERAL_TABLE_DATA[1].data = data;
    });
    }
}

makeTimeStamp() {
    var currentDate = new Date();
    this.timestamp = currentDate.getTime();
}

constructor(private http: HttpClient) { }

fillGeneralTable() {
    GENERAL_TABLE_DATA = [
        { name: 'Current login', data: this.curLogin },
        { name: 'Last hour reports', data: this.reportsLastHour },
        { name: 'Devices in total', data: this.totalDevices },
    ];

    this.generalTableDataToShow = GENERAL_TABLE_DATA;
}

ngOnInit(): void {
    if (this.isAuth == true) {
        this.getTotalDevices();
        this.getLastHourReports();
        this.fillGeneralTable();
    }
}

}

```

main-top.component.html

```

<div class="center-wide">

    <mat-expansion-panel [expanded]="true">
        <mat-expansion-panel-header>
            <mat-panel-title>
                General info
            </mat-panel-title>

```

```

        <mat-panel-description>

        </mat-panel-description>
    </mat-expansion-panel-header>

    <table mat-table [dataSource]="generalTableDataToShow" class="generalTable">
        ...

        <!-- first Column -->
        <ng-container matColumnDef="name" class="w150px">
            <!-- <mat-header-cell *matHeaderCellDef> Model </mat-header-cell> --
>
            <mat-
cell *matCellDef="let element" class="w150px"> {{element.name}} </mat-cell>
            </ng-container>

            <!-- second Column -->
            <ng-container matColumnDef="data" class="widthauto">
                <!-- <mat-header-cell *matHeaderCellDef> IPs found </mat-header-
cell> -->
                <mat-cell *matCellDef="let element"> {{element.data}} </mat-cell>
            </ng-container>

            <!-- <mat-header-
row *matHeaderRowDef="displayedColumnsGeneralInfo"></mat-header-row> -->
            <mat-
row *matRowDef="let row; columns: displayedColumnsGeneralInfo;"></mat-row>

        </table>

    </mat-expansion-panel>

</div>

```

search.component.ts

```

import { SharedService } from '../search/shared/shared.service';
import { ResultDialogComponent } from '../result-dialog/result-dialog.component';

import { Component, OnInit, Input } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatNativeDateModule } from '@angular/material/core';
import { group } from '@angular/animations';
import { MatTableModule } from '@angular/material/table';
import { MatDialog, MatDialogModule, MAT_DIALOG_DATA, MatDialogConfig } from '@angular/material/dialog';

export interface tableIPMACSearch {

```

```

    Model: string;
    IPs: number;
    List: string;
}

const TEST_TABLE_DATA: tableIPMACSearch[] = [
  { Model: 'DIR-8', IPs: 10, List: "param1" },
  { Model: 'DIR-77', IPs: 20, List: "param2" },
  { Model: 'DIR-666', IPs: 5, List: "param3" },
];

var TABLE_DATA: tableIPMACSearch[] = [];

@Component({
  selector: 'app-search',
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.css']
})

export class SearchComponent implements OnInit {

  inputToSearchByIP: string = null;
  inputToSearchByMAC: string;

  currentSearchType: string;
  areResultsFound: boolean = false;

  models;

  IPMACFilterSearch: object = [
    null,
    null,
    null,
    null,
    null,
    null,
    null,
    null,
    null,
    null,
    null
  ];

  receivedIPMACSearchJSON;
  parsedIPMACSearchJSON: object;
  // data for table with results
  tableDataToShow;
  displayedColumns: string[] = [
    'Model',
    'IPs',
    // 'List'

```

```

];
// is Authorization complete
@Input() isAuth: boolean;

// data for HTTP GET
@Input() currentToken;
@Input() proxyURL;
@Input() currentSLAURL;
searchByIPURL = "ips/";
searchByMACURL = "macs/";

// search type: IP or MAC, needed for results table
currentResultsType: string = '';

// indicates search process (for spinner)
searchProgress: boolean = false;

// shared data for search results dialog
searchResultsForDialogMAC;

templateDialogJSON = {
  allResults: "",
  groupName: "",
  resultsType: "",
};

constructor(
  private http: HttpClient,
  public dialog: MatDialog,
  private sharedService: SharedService,
) { }

ngOnInit(): void {
  this.sharedService.sharedSearchResultsJSON.subscribe(message => this.searchResultsForDialogMAC = message);
}

searchByIPInput(event: any) {
  const value = event.target.value
  this.inputToSearchByIP = value
  //console.log(this.inputToSearchByIP)
}

searchByMACInput(event: any) {
  const value = event.target.value
  this.inputToSearchByMAC = value
  //console.log(this.inputToSearchByMAC)
}

// searchFilterInput(event: any) {

```

```

//  const drop = event.target.value[0]
//  this.IPMACFilterSearch[0] = drop
// }

SearchFilterDROP(event: any) {
  this.IPMACFilterSearch[0] = event.target.value
}

SearchFilterCRC(event: any) {
  this.IPMACFilterSearch[1] = event.target.value
}

SearchFilterFragments(event: any) {
  this.IPMACFilterSearch[2] = event.target.value
}

SearchFilterLinkDown(event: any) {
  this.IPMACFilterSearch[3] = event.target.value
}

SearchFilter64bytes(event: any) {
  this.IPMACFilterSearch[4] = event.target.value
}

SearchFilterDeviceModel(event: any) {
  this.IPMACFilterSearch[7] = event.target.value
}

SearchFilterDeviceSoftware(event: any) {
  this.IPMACFilterSearch[8] = event.target.value
}

SearchFilterSerialNumber(event: any) {
  this.IPMACFilterSearch[9] = event.target.value
}

SearchFilterLFTD(event: any) {
  this.IPMACFilterSearch[5] = event.target.value
}

SearchFilterMFTD(event: any) {
  this.IPMACFilterSearch[6] = event.target.value
}

formJSONforIPMACSearch() {
  return {
    "drop": this.IPMACFilterSearch[0],
    "crc": this.IPMACFilterSearch[1],
    "fragments": this.IPMACFilterSearch[2],
    "linkDown": this.IPMACFilterSearch[3],
    "64bytes": this.IPMACFilterSearch[4],
  }
}

```

```

        "ltDate": this.IPMACFilterSearch[5],
        "gtDate": this.IPMACFilterSearch[6],
        "deviceModel": this.IPMACFilterSearch[7],
        "deviceSoftware": this.IPMACFilterSearch[8],
        "serialNumber": this.IPMACFilterSearch[9]
    }
}

searchByIP() {
    // console.log(`Token: `+this.currentToken)
    if (this.isAuth == true) {
        this.currentSearchType = 'IP';
        this.searchProgress = true;

        // search for all results if field is empty
        if (this.inputToSearchByIP == null || this.inputToSearchByIP == "") {
            this.searchByMAC()
        }
        else {
            var toSearchByIP = this.inputToSearchByIP;
            this.http.post(this.proxyURL + this.currentSLAURL + this.searchByIPURL + toSearchByIP, this.formJSONforIPMACSearch(), {
                headers: new HttpHeaders({
                    'Content-Type': 'application/json',
                    'token': this.currentToken
                })
            }).toPromise().then((data: any) => {
                var receivedIPSearchJSON = data;
                if (receivedIPSearchJSON.length == 0) {
                    alert("Nothing found by IP");
                    this.searchProgress = false;
                }
                else if (receivedIPSearchJSON.length >= 0) {
                    this.showSearchResults(receivedIPSearchJSON);
                    this.areResultsAreFound(true, 0);
                    // alert(receivedIPSearchJSON.length + " Results found");
                }
            });
        }
    }
}

searchByMAC() {
    if (this.isAuth == true) {
        this.searchProgress = true;
        this.currentSearchType = 'MAC';
        // search for all results if field is empty
        if (this.inputToSearchByMAC == null || this.inputToSearchByMAC == "") {
            var toSearchByMAC = "all";
        }
        else {

```

```

        toSearchByMAC = this.inputToSearchByMAC;
    }

    this.http.post(this.proxyURL + this.currentSLAURL + this.searchByMACURL + toSearchByMAC, this.formJSONforIPMACSearch(), {
        headers: new HttpHeaders({
            'Content-Type': 'application/json',
            'token': this.currentToken
        })
    }).toPromise().then((data: any) => {
        var receivedMACSearchJSON = data;
        if (receivedMACSearchJSON.length == 0) {
            alert("Nothing found by MAC");
            this.searchProgress = false;
            this.areResultsAreFound(false, 100);
        }
        else if (receivedMACSearchJSON.length >= 0) {
            this.showSearchResults(receivedMACSearchJSON);
            this.areResultsAreFound(true, 1);
        }
    });
}

areResultsAreFound(areFound: boolean, type: number) {
    if (areFound == true) {
        switch (type) {

            // IP search
            case 0:
                this.currentResultsType = 'IP';
                break;

            // MAC search
            case 1:
                this.currentResultsType = 'MAC';
                break;

            default:
                break;
        }
        this.areResultsFound = true;
        this.searchProgress = false;
    }
    else if (areFound == false) {
        this.areResultsFound = false;
    }
}

showSearchResults(receivedIPMACSearchJSON) {
    this.parsedIPMACSearchJSON = this.parseReceivedData(receivedIPMACSearchJSON);
}

```

```

this.receivedIPMACSearchJSON = receivedIPMACSearchJSON;
// console.log(this.parsedIPSearchJSON);

this.models = receivedIPMACSearchJSON.reduce((models, parsedArrayItem, index, array) => {
    if (!models.find(model => parsedArrayItem.model.toString() === model)) {
        models.push(parsedArrayItem.model.toString())
    }
    return models;
}, [])

// console.log(models);
// console.log("Models total: " + models.length);

this.prepareTableDataToShow(receivedIPMACSearchJSON, this.models);
}

prepareTableDataToShow(receivedData: any, models: any) {

    // // for test
    // this.tableDataToShow = TEST_TABLE_DATA;

    // for prod
    for (var i = 0, lenModels = models.length; i < lenModels; i++) {

        var numberOfIPs: number = 0;

        // counting devices (IPs) quantity
        for (var j = 0, len = receivedData.length; j < len; j++) {
            if (receivedData[i].model == models[i]) {
                numberOfIPs++;
            }
        }

        // filling one table row
        TABLE_DATA[i] = {
            Model: models[i],
            IPs: numberOfIPs,
            List: models[i]
        }

        numberOfIPs = 0;
    }
    this.tableDataToShow = TABLE_DATA;

    // necessary for table update on each search
    TABLE_DATA = []
}

parseReceivedData(dataToParse: any) {
    var dataStringified = JSON.stringify(dataToParse);

```



```

    var parsedData = JSON.parse(dataStringified);
    return parsedData;
}

clearAllFields() {
    //console.log("clearAllFields()");
    this.inputToSearchByIP = undefined;
    //console.log(this.inputToSearchByIP);
    this.inputToSearchByMAC = undefined;
}

deviceSearch() {

}

showDevicesList(modelsGroupToShow) {
    // console.log(modelsGroupToShow);
    var JSON = this.makeJSONForDialog(this.receivedIPMACSearchJSON, modelsGroupToShow);

    this.openResultDialog(JSON);
}

makeJSONForDialog(dataToProcessing, group) {
    var proceedJSON = this.templateDialogJSON;

    // var proceedJSON = dataToProcessing;

    // for (var i = 0, len = dataToProcessing.length; i < len; i++) {
    //     if (dataToProcessing[i].model = group ) {

    //     }
    // }

    proceedJSON.allResults = dataToProcessing;
    proceedJSON.resultsType = this.currentResultsType;
    proceedJSON.groupName = group;

    return proceedJSON;
}

openResultDialog(dataToShow) {

    this.sendDataToDialog(dataToShow);
    // this.searchResultsForDialogMAC = dataToShow;

    const dialogRef = this.dialog.open(ResultDialogComponent);

    dialogRef.afterClosed().subscribe(result => {
        // console.log(`Dialog result: ${result}`);
    });
}

```

```

}

sendDataToDialog(dataToSend) {
  this.sharedService.updateJSON(dataToSend);
}

generateEndingOfTheWord(dataToProcess) {
  if (dataToProcess.length == 1 || dataToProcess.length == -1) {
    return '';
  }

  else {
    return 's';
  }
}
}

```

search.component.html

```

<div class="center-wide">
  <div class="mat-exp-panel">

    <mat-expansion-panel [expanded]="true">
      <mat-expansion-panel-header>
        <mat-panel-title>
          Search devices
        </mat-panel-title>
        <mat-panel-description></mat-panel-description>
      </mat-expansion-panel-header>

      <mat-form-field appearance="fill" class="widthauto ip-mac-search">
        <mat-label>IP</mat-label>
        <input matInput maxlength="15" placeholder="0.0.0.0" (input)="search
ByIPInput($event)">

        <button *ngIf="!(searchProgress)" mat-icon-
button matSuffix (click)="searchByIP()">
          <mat-icon>search</mat-icon>
        </button>

        <button *ngIf="searchProgress" mat-icon-
button disabled class="transparent" matSuffix >
          <mat-icon>
            <mat-spinner diameter="28"></mat-spinner>
          </mat-icon>
        </button>

      </mat-form-field>

```

```

<mat-form-field appearance="fill" class="widthauto ip-mac-search">
  <mat-label>MAC</mat-label>
  <input matInput maxLength="17" placeholder="00:00:00:00:00:00" (input)="searchByMACInput($event)">

  <button *ngIf="!(searchProgress)" mat-icon-
button matSuffix (click)="searchByMAC()">
    <mat-icon>search</mat-icon>
  </button>

  <button *ngIf="searchProgress" mat-icon-
button disabled class="transparent" matSuffix >
    <mat-icon>
      <mat-spinner diameter="28"></mat-spinner>
    </mat-icon>
  </button>

</mat-form-field>

<!-- <button (click)="searchByMACInput('all')" mat-
button color="primary" class="widthauto ip-mac-search-filter">
  Show all devices
</button> -->

</mat-expansion-panel>
</div>

<div class="mat-exp-panel">
  <mat-expansion-panel>
    <mat-expansion-panel-header>
      <mat-panel-title>
        Filters for search
      </mat-panel-title>
      <mat-panel-description></mat-panel-description>
    </mat-expansion-panel-header>
    <!-- place for filter inputs -->
    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
      <mat-label>DROP ></mat-label>
      <input matInput (input)="SearchFilterDROP($event)">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
      <mat-label>CRC ></mat-label>
      <input matInput (input)="SearchFilterCRC($event)">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">

```

```

        <mat-label>Fragments ≥</mat-label>
        <input matInput (input)="SearchFilterFragments($event)">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
        <mat-label>Link down ≥</mat-label>
        <input matInput (input)="SearchFilterLinkDown($event)">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
        <mat-label>64bytes ≥</mat-label>
        <input matInput (input)="SearchFilter64bytes($event)">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
        <mat-label>Device model</mat-label>
        <input matInput (input)="SearchFilterDeviceModel($event)">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
        <mat-label>Device software</mat-label>
        <input matInput (input)="SearchFilterDeviceSoftware($event)">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
        <mat-label>Serial number</mat-label>
        <input matInput (input)="SearchFilterSerialNumber($event)">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
        <mat-label>Less from this day</mat-label>
        <input matInput (input)="SearchFilterLFTD($event)" placeholder="YYYY
-MM-DD">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
        <mat-label>More from this day</mat-label>
        <input matInput (input)="SearchFilterMFTD($event)" placeholder="YYYY
-MM-DD">
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
        <mat-label>Less from this day</mat-label>

```

```

        <input matInput [matDatepicker]="pickerLFTD" (input)="SearchFilterLF
TD($event)"
            placeholder="YYYY-MM-DD">
        <mat-datepicker-toggle matSuffix [for]="pickerLFTD"></mat-
datepicker-toggle>
        <mat-datepicker touchUi #pickerLFTD></mat-datepicker>
    </mat-form-field>

    <mat-form-field appearance="fill" class="widthauto ip-mac-search-
filter">
        <mat-label>Less from this day</mat-label>
        <input matInput [matDatepicker]="pickerMFTD" (input)="SearchFilterLF
TD($event)"
            placeholder="YYYY-MM-DD">
        <mat-datepicker-toggle matSuffix [for]="pickerMFTD"></mat-
datepicker-toggle>
        <mat-datepicker touchUi #pickerMFTD></mat-datepicker>
    </mat-form-field>

</mat-expansion-panel>
</div>

<div class="mat-exp-panel">
    <mat-expansion-panel *ngIf="areResultsFound" [expanded]="true">
        <mat-expansion-panel-header>
            <mat-panel-title>
                Search results
            </mat-panel-title>
            <mat-panel-description>
                {{receivedIPMACSearchJSON.length}} device{{generateEndingOfTheWo
rd(receivedIPMACSearchJSON)}} of {{models.length}} model{{generateEndingOfTheWord(mo
dels)}}
            </mat-panel-description>
        </mat-expansion-panel-header>

        <table mat-table [dataSource]="tableDataToShow">

            <!-- Model Column -->
            <ng-container matColumnDef="Model">
                <mat-header-cell *matHeaderCellDef> Model </mat-header-cell>
                <mat-cell *matCellDef="let element"> {{element.Model}} </mat-
cell>
            </ng-container>

            <!-- number of devices Column -->
            <ng-container matColumnDef="IPs">
                <mat-header-
cell *matHeaderCellDef> {{currentResultsType}}s found </mat-header-cell>
                <mat-cell *matCellDef="let element">

```

```

        <a>
            <button (click)="showDevicesList(element.Model)" mat-
button color="primary">
                Show ({{element.IPs}})
            </button>
        </a>
    </mat-cell>
</ng-container>

<mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-
row>

<mat-row *matRowDef="let row; columns: displayedColumns;"></mat-row>

</table>
</mat-expansion-panel>
</div>

```

search.component.css

```

.btn-fit{
    margin: auto !important;
    align-self: center;
}

.searchFormIPMAC{

    width: calc(100% - 100px);
}

.inputSearchIPMAC{
    width: 100%;
}

table {
    width: 100%;
}

```

result-dialog.component.ts

```

import { SharedService } from "../shared/shared.service";

import { Component, OnInit } from '@angular/core';
import { Identifiers } from '@angular/compiler';
import { stringify } from 'querystring';

```

```

export interface tableDialog {
    type;
    bytes64: string;
    crc: string;
    drop: string;
    fragments: string;
    link_down: string;
    action: {
        resultType;
        device;
        macs;
        logs;
    };
}

var dialogTableData: tableDialog[] = [];

@Component({
    selector: 'app-result-dialog',
    templateUrl: './result-dialog.component.html',
    styleUrls: ['./result-dialog.component.css']
})
export class ResultDialogComponent implements OnInit {

    currentLocale: string = "ru-RU";

    // shared data for serach results dialog
    sharedData;
    sharedDataStringified;
    processedData = {
        id: "",
        bytes64: "",
        crc: "",
        drop: "",
        fragments: "",
        link_down: "",
        groupName: "",
        resultsType: "",
    }

    tableDialogDataToShow;

    displayedColumns: string[] = [
        'type',
        'bytes64',
        'crc',
        'drop',
        'fragments',
        'link_down',
        'action',
    ];
}

```

```

constructor(
    private sharedService: SharedService,
) { }

ngOnInit() {
    this.sharedService.sharedSearchResultsJSON.subscribe(results => this.sharedData
= results);
    this.prepareTable();
}

public timestampToDate(unix, locale) {
    var date = new Date(unix).toLocaleString(locale);
    return date;
}

generateStringValueDate(value, date) {
    var string: string = value + " / " + this.timestampToDate(date, this.currentLoca
le);
    return string;
}

prepareTable() {
    this.sharedDataStringified = JSON.stringify(this.sharedData);

    for (var i = 0, len = this.sharedData.allResults.length; i < len; i++) {
        if (this.sharedData.allResults[i].model[0] == this.sharedData.groupName) {

            var processedData = {
                type: this.sharedData.allResults[i]._id,
                bytes64: this.generateStringValueDate(this.sharedData.allResults[i]["64byt
es"].value, this.sharedData.allResults[i]["64bytes"].time), //[0]["64bytes"]
                crc: this.generateStringValueDate(this.sharedData.allResults[i].crc.value,
this.sharedData.allResults[i].crc.time),
                drop: this.generateStringValueDate(this.sharedData.allResults[i].drop.valu
e, this.sharedData.allResults[i].drop.time),
                fragments: this.generateStringValueDate(this.sharedData.allResults[i].frag
ments.value, this.sharedData.allResults[i].fragments.time),
                link_down: this.generateStringValueDate(this.sharedData.allResults[i].link
_down.value, this.sharedData.allResults[i].link_down.time),
                action: {
                    resultType: this.sharedData.resultsType,
                    device: this.sharedData.allResults[i]._id
                }
            }

            dialogTableData[i] = {
                type: processedData.type,
                bytes64: processedData.bytes64,
                crc: processedData.crc,
                drop: processedData.drop,

```



```

        fragments: processedData.fragments,
        link_down: processedData.link_down,
        action: {
            resultType: processedData.action.resultType,
            device: processedData.action.device,
            macs: '',
            logs: '',
        }
    }
}

this.processedData.groupName = this.sharedData.groupName;

this.tableDialogDataToShow = dialogTableData;
dialogTableData = [];

// this.sharedService.updateJSON({});
}

showLogs(deviceToShow) {

}

}

```

result-dialog.component.html

```

<h2 mat-dialog-title>Devices of {{sharedData.groupName}}</h2>
<mat-dialog-content class="mat-typography">
    <!-- <pre>{{sharedDataStringified}}</pre> -->
    <table mat-table [dataSource]="tableDialogDataToShow" class="generalTable">

        <!-- Column -->
        <ng-container matColumnDef="type">
            <mat-header-cell *matHeaderCellDef> {{sharedData.resultsType}} </mat-
header-cell>
            <mat-cell *matCellDef="let element"> {{element.type}} </mat-cell>
        </ng-container>

        <!-- Column -->
        <ng-container matColumnDef="bytes64">
            <mat-header-cell *matHeaderCellDef>64bytes (max / date)</mat-header-
cell>
            <mat-cell *matCellDef="let element"> {{element.bytes64}} </mat-cell>
        </ng-container>

        <!-- Column -->
        <ng-container matColumnDef="crc">
            <mat-header-cell *matHeaderCellDef>CRC (max / date)</mat-header-cell>

```

```

        <mat-cell *matCellDef="let element"> {{element.crc}} </mat-cell>
    </ng-container>

    <!-- Column -->
    <ng-container matColumnDef="drop">
        <mat-header-cell *matHeaderCellDef>DROP (max / date)</mat-header-cell>
        <mat-cell *matCellDef="let element"> {{element.drop}} </mat-cell>
    </ng-container>

    <!-- Column -->
    <ng-container matColumnDef="fragments">
        <mat-header-cell *matHeaderCellDef>Fragments (max / date)</mat-header-
cell>
        <mat-cell *matCellDef="let element"> {{element.fragments}} </mat-cell>
    </ng-container>

    <!-- Column -->
    <ng-container matColumnDef="link_down">
        <mat-header-cell *matHeaderCellDef>Link down (max / date)</mat-header-
cell>
        <mat-cell *matCellDef="let element"> {{element.link_down}} </mat-cell>
    </ng-container>

    <!-- Column -->
    <ng-container matColumnDef="action">
        <mat-header-cell *matHeaderCellDef>Actions</mat-header-cell>
        <mat-cell *matCellDef="let element">
            <button mat-
button (click)="showLogs(element.action.logs)">Show logs</button>
        </mat-cell>
    </ng-container>

    <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
    <mat-row *matRowDef="let row; columns: displayedColumns;"></mat-row>

</table>
</mat-dialog-content>

<mat-dialog-actions align="end">
    <!-- <button mat-button (click)="prepareTable()">Show table</button> -->
    <button mat-button [mat-dialog-close]="true" cdkFocusInitial>Close</button>
</mat-dialog-actions>

```

result-dialog.component.html

```

.mat-column-action {
    width: fit-content !important;
}

```

```

.mat-footer-row,
.mat-header-row,
.mat-row {
    display: inline-flex;
    min-width: 100%;
}

```

shared.service.ts

```

import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

@Injectable()
export class SharedService {

    private searchResultsForDialogMACJSON = new BehaviorSubject(
        {

        }
    );

    sharedSearchResultsJSON = this.searchResultsForDialogMACJSON.asObservable();

    constructor() { }

    updateJSON(JSON: object) {
        this.searchResultsForDialogMACJSON.next(JSON)
    }

}

```

styles.css

```

/* You can add global styles to this file, and also import other style files */

@import url("https://fonts.googleapis.com/css2?family=Roboto&display=swap");
@import "./purple-green_custom.css";

body {
    font-family: "Roboto", sans-serif;
    margin: 0px;
    background-color: #222226;
}

:root {
    --primary: #0087a9;
    --blue: #0087a9;
}

.center-wide {

```

```

    margin: auto;
    width: 80%;
    max-width: 790px;
    margin-top: 20px;
    /* margin-bottom: 20px; */
}

.right {
    text-align: right;
}

pre {
    background-color: #444446 !important;
    color: #fff !important;
}

.div-button {
    align-content: stretch;
}

.btn-primary {
    color: #fff;
    background-color: #0087a9;
    border-color: #0087a9;
    margin-left: 15px;
    /* margin-right: 5px;
    margin-right: 15px; */
    position: relative;
}

.div-between-buttons {
    margin-left: auto;
    margin-right: auto;
    position: relative;
    height: 38px;
}

.generalTable {
    width: 100%;
    /* background-color: #444446 !important; */
    color: #fff !important;
}

td {
    color: rgba(255, 255, 255, 0.8);
}

td,
th {
    padding: 5px 5px;
    display: table-cell;
}

```

```

    text-align: left;
    vertical-align: middle;
    border-radius: 2px;
    border-color: rgb(236, 236, 236);
}

tr.last {
    border: none;
}

tr {
    border-bottom: 1px solid rgba(255, 255, 255, 0.12);
}

.float-right {
    float: right;
}

html,
body {
    height: 100%;
}

body {
    margin: 0;
    font-family: Roboto, "Helvetica Neue", sans-serif;
}

.pad30 {
    padding: 30px;
}

.widthauto {
    width: auto;
}

.login-div {
    margin: auto;
    width: 266px;
    max-width: 98%;
}

.w150px {
    width: 150px;
}

.mat-form-field.ip-mac-search-filter {
    width: 100%;
    max-width: 230px;
    margin-left: 7.5px;
    margin-right: 7.5px;
}

```

```
.mat-form-field.ip-mac-search {  
  width: 100%;  
  max-width: 356px;  
  margin-left: 7.5px;  
  margin-right: 7.5px;  
}  
  
div.center-wide + div.center-wide {  
  padding: 25px;  
}  
  
.mat-exp-panel {  
  padding-bottom: 20px;  
}  
  
.transparent {  
  color: transparent;  
  background: transparent;  
}
```

ПРИЛОЖЕНИЕ Б. ОСНОВНЫЕ МЕТОДЫ HTTP API СЕРВЕРА

- GET /info/prev/:mac/:timestamp - информация о данном устройстве из лога, предшествующего заданному моменту времени;
- GET /info/:mac/:timestamp - информация о данном устройстве в заданный момент времени;
- GET /info/totalDevices - общее количество устройств в базе;
- GET /info/firmware - общий список устройств с названием модели, версией прошивки и MAC-адресом;
- GET /activeday/:mac - информация об активных днях устройства;
- GET /logs/lasthour/:timestamp - количество отчетов пришедших в последний час;
- GET /logs/info/prev/:mac/:timestamp - информация о предыдущем логе;
- GET /logs/info/next/:mac/:timestamp - информация о следующем логе;
- GET /logs/timestamps/:mac/:day/:month/:year - список timestamp'ов, в которые прилетали логи от устройства за указанный день;
- GET /logs/:mac/:day/:month/:year - список логов, прилетевших от устройства за указанный день;
- GET /events/info/prev/:mac/:timestamp - информация о предыдущем событии;
- GET /events/timestamps/:mac/:day/:month/:year - список timestamp'ов, в которые прилетали логи событий от устройства за указанный день;
- GET /ips - список всех IP-адресов в базе;
- GET /macs - список всех MAC-адресов в базе;
- GET /macs/ip/:ip - список MAC-адресов по заданным IP;

- GET /macs/avail/:timestamp - список MAC-адресов, приславших логи после заданного момента времени;
- GET /timerange/:mac - минимальный и максимальный timestamp для данного MAC-адреса.

Получение информации по дням:

- GET /wan/:mac/:day/:month/:year;
- GET /lan/:port/:mac/:day/:month/:year;
- GET /summary/:mac/:day/:month/:year;
- GET /wifi/:freq/:mac/:day/:month/:year;
- GET /info/:mac/:day/:month/:year;
- GET /wifi/clients/:mac/:day/:month/:year;
- GET /system/:mac/:day/:month/:year.