Содержание

[ВВЕДЕНИЕ 3](#_Toc42331422)

[1 Постановка задачи 4](#_Toc42331423)

[1.1 Описание предметной области 4](#_Toc42331424)

[1.2 Определение этапов разработки 6](#_Toc42331425)

[2 Технико-экономическое обоснование темы 8](#_Toc42331426)

[2.1 Описание проблемы 8](#_Toc42331427)

[2.2 Назначение разрабатываемого ПО 8](#_Toc42331428)

[2.3 Характеристика основных функций и задач разрабатываемого ПО 9](#_Toc42331429)

[2.4 Обзор существующих программных средств 9](#_Toc42331430)

[3 Теоретическая часть 18](#_Toc42331431)

[3.1 Системный анализ предметной области 19](#_Toc42331432)

[3.2 Выбор средств разработки и языков программирования. 21](#_Toc42331433)

[4 Проектная часть 25](#_Toc42331434)

[4.1 Проектирование пользовательского интерфейса 25](#_Toc42331435)

[4.1.1 Разработка дерева форм 25](#_Toc42331436)

[4.1.2 Разработка прототипа пользовательского интерфейса 27](#_Toc42331437)

[4.2 Организация взаимодействия с сервером 28](#_Toc42331438)

[4.3 Описание технологии разработки клиентской части веб-приложений с использованием фреймворка «Angular» 29](#_Toc42331439)

[4.4 Разработка пользовательского интерфейса 31](#_Toc42331440)

[4.4.1 Корневой модуль приложения 31](#_Toc42331441)

[4.4.2 Компонент «Корневой» 31](#_Toc42331442)

[4.4.3 Компонент «Панель инструментов» 32](#_Toc42331443)

[4.4.4 Компонент «Подвал» 32](#_Toc42331444)

[4.4.5 Компонент «Авторизация» 32](#_Toc42331445)

[4.4.6 Компонент «Основная информация» 32](#_Toc42331446)

[4.4.7 Компонент «Поиск» 32](#_Toc42331447)

[4.4.8 Компонент «Диалог с результатами» 32](#_Toc42331448)

[4.4.9 Сервис «Общий» 32](#_Toc42331449)

[4.4.10 Реализация мобильной версии 32](#_Toc42331450)

[5 Разработка программной документации 34](#_Toc42331451)

[5.1 Руководство системного программиста 34](#_Toc42331452)

[5.2 Руководство пользователя 34](#_Toc42331453)

[6 Тестирование программного обеспечения 35](#_Toc42331454)

[ЗАКЛЮЧЕНИЕ 36](#_Toc42331455)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 37](#_Toc42331456)

[ПРИЛОЖЕНИЕ А. ЛИСТИНГИ ОСНОВНЫХ ПРОГРАММНЫХ МОДУЛЕЙ 38](#_Toc42331457)

[ПРИЛОЖЕНИЕ Б. ОСНОВНЫЕ МЕТОДЫ HTTP API СЕРВЕРА 72](#_Toc42331458)

# 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.

# Постановка задачи

## 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 сэкономить ресурсы и улучшить качество сервисов, предоставляемых сетевыми провайдерами.

## 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, наличие поведения не обязательно.
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.

# Технико-экономическое обоснование темы

## 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.

## 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.

## 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 введённых 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 обозреваемых устройств в разрезе времени;
* информация о пользователе, данные для его идентификации и авторизации.

## 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 2.4.1), 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.

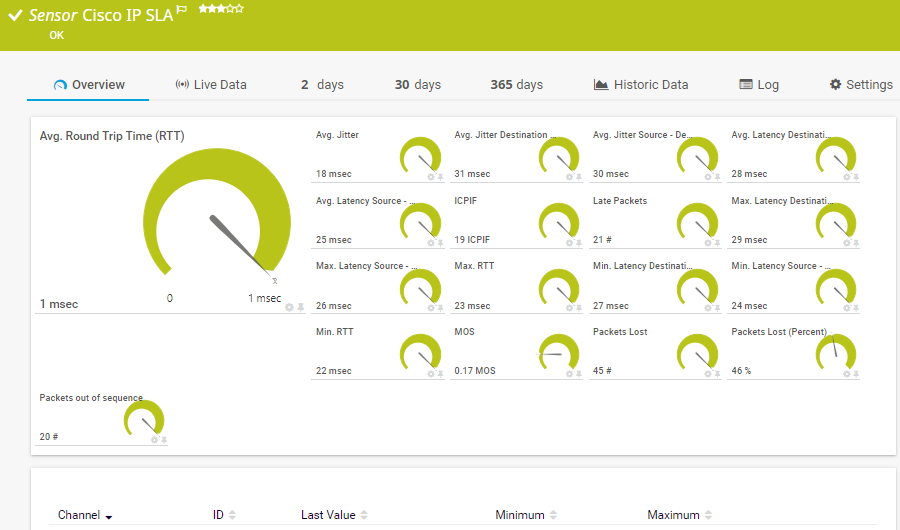


Рисунок . – Пользовательский интерфейс Cisco IP SPA Sensor

D-Link SLA-system. Эффективное сетевое управление в оборудованиях 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 чётких требований. Поэтому целевым потребителем данной компании по большей части является обычный рядовой пользователь домашнего интернета, который не имеет желания переплачивать за сверхвысокое качество, излишнюю надёжность и пожизненную гарантию.

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 2.2.1:

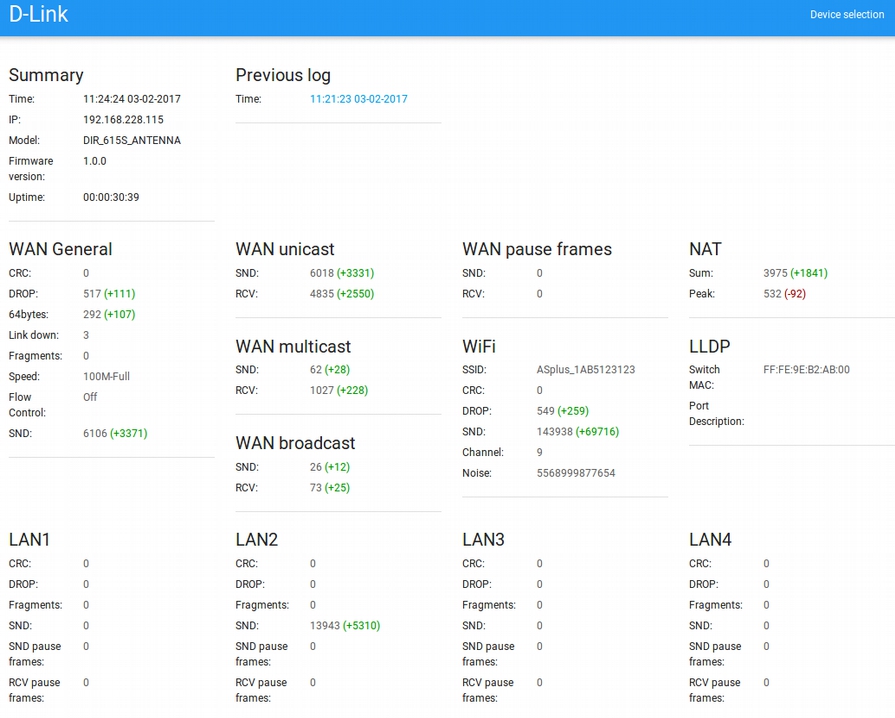


Рисунок . – Пример пользовательского интерфейса системы мониторинга D-Link SLA-system

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 2.3.1 и рисунке 2.3.2:

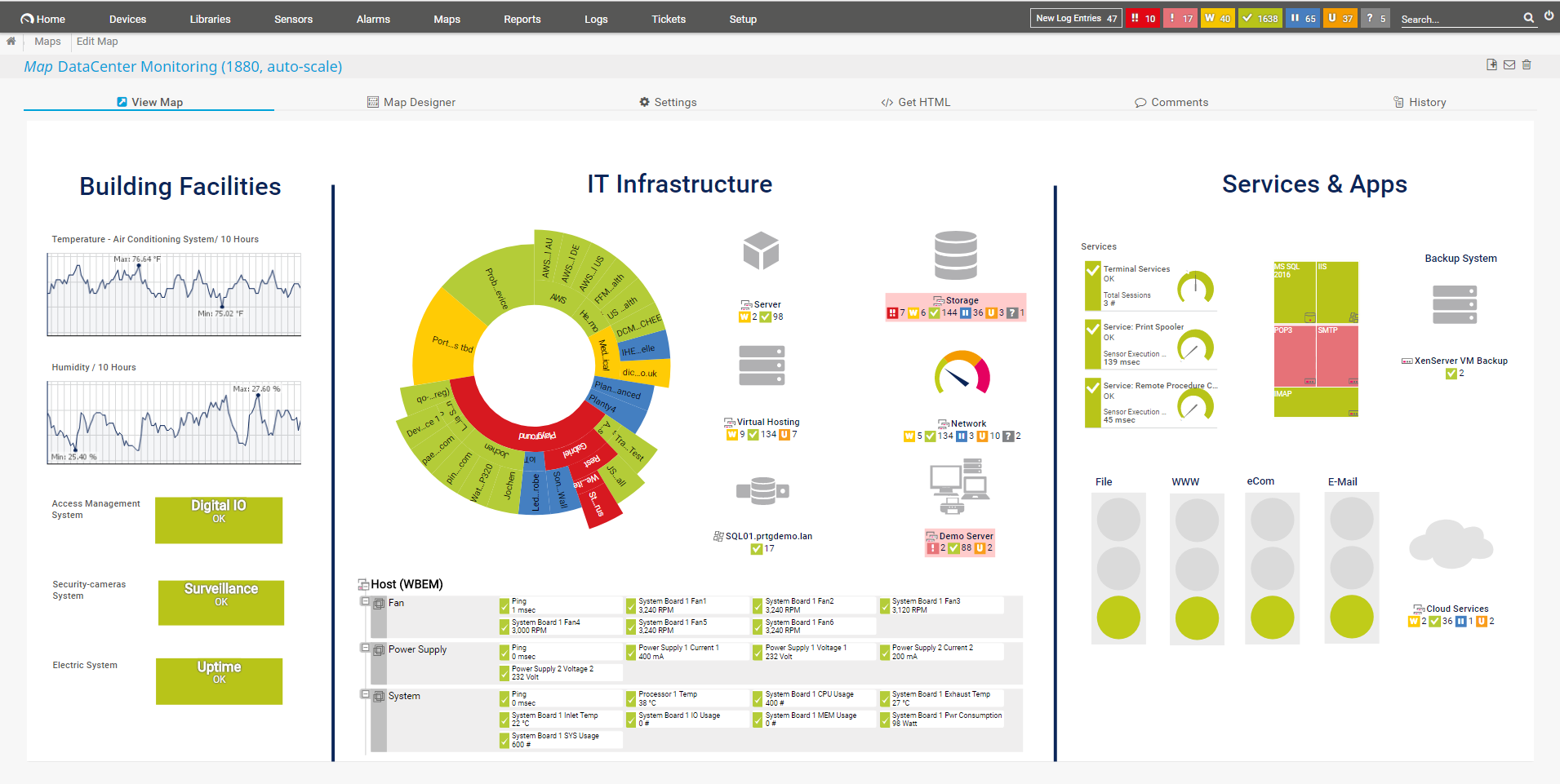


Рисунок . – Визуальный интерфейс PRTG Network Monitor

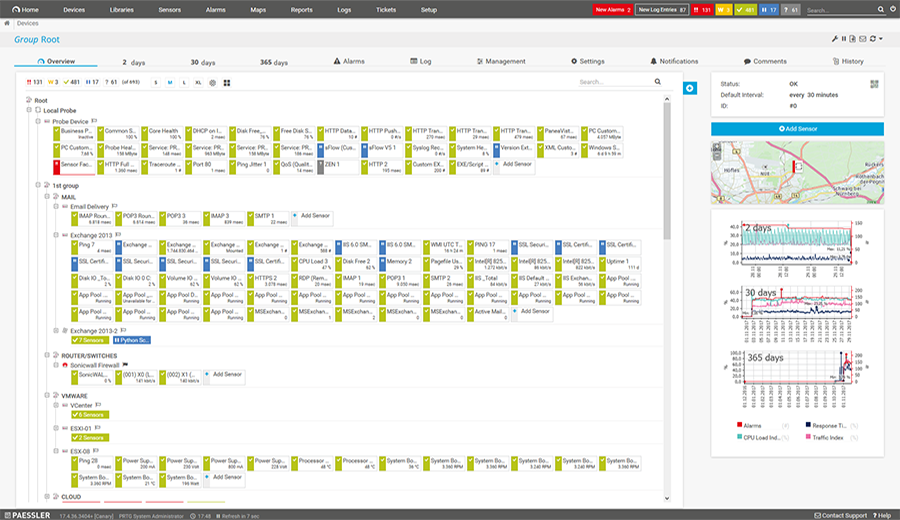


Рисунок . – Просмотр дерева групп в PRTG Network Monitor

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

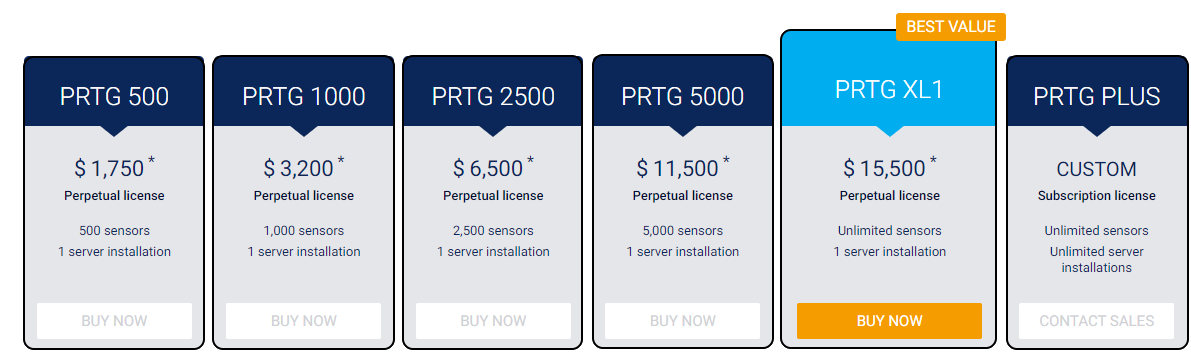


Рисунок . – Цены на услуги компании Paessler AG

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 помесячной 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 помесячной оплатой за каждый установленный аппаратный и программный агент. Выбранное решение довольно специфично и подойдёт не для каждой компании. Главная особенность, которую нужно учитывать – с ростом офисов и масштабов сети компании придётся 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 2.4.1:

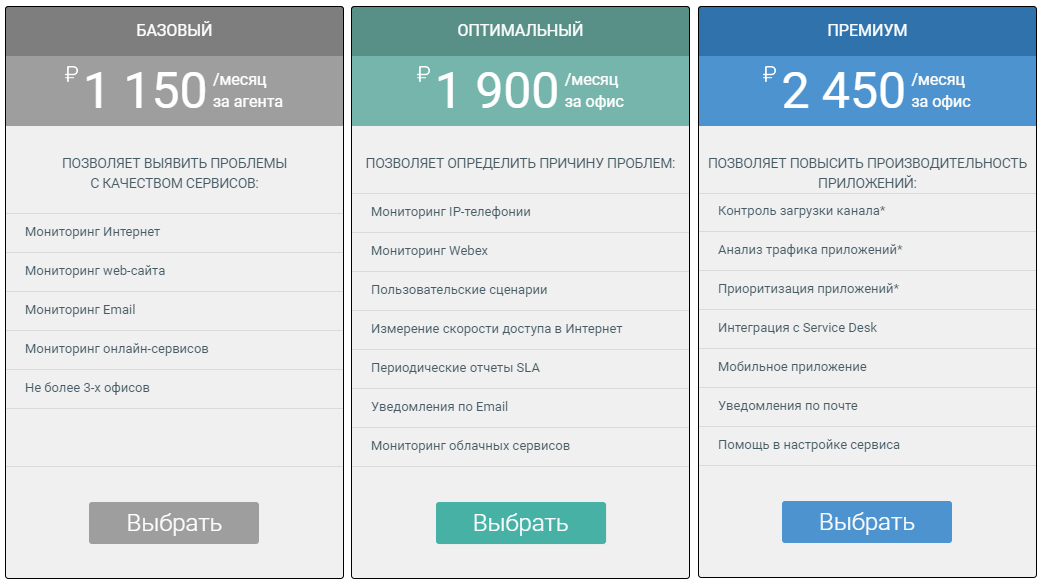


Рисунок . – Стоимость услуг компании SLAMON

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.

# 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 и в информативности пользовательского интерфейса программного решения. Поэтому в качестве предлагаемого средства в рамках данной ВКР решено было представить собственную версию WEB-интерфейса SLA-мониторинга в связке с рабочим SLA-сервером. 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.

## 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 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, 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 сервере с указанными пользователем параметрами – другими словами, связывать пользователя и сервер (рисунок 3.1.1):

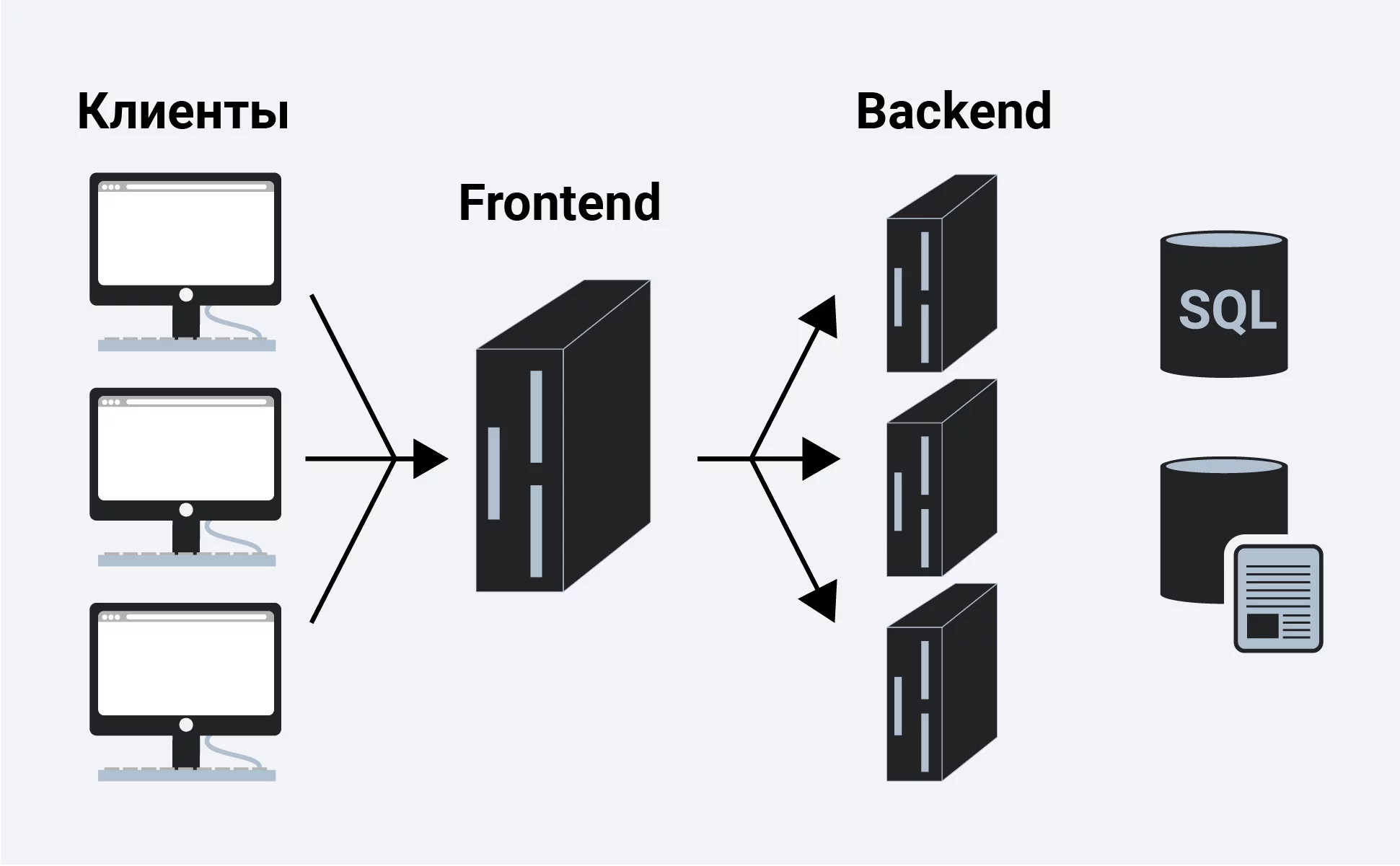


Рисунок . – Диаграмма фронтов разработки

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

## 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 приято 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 4.1.1:

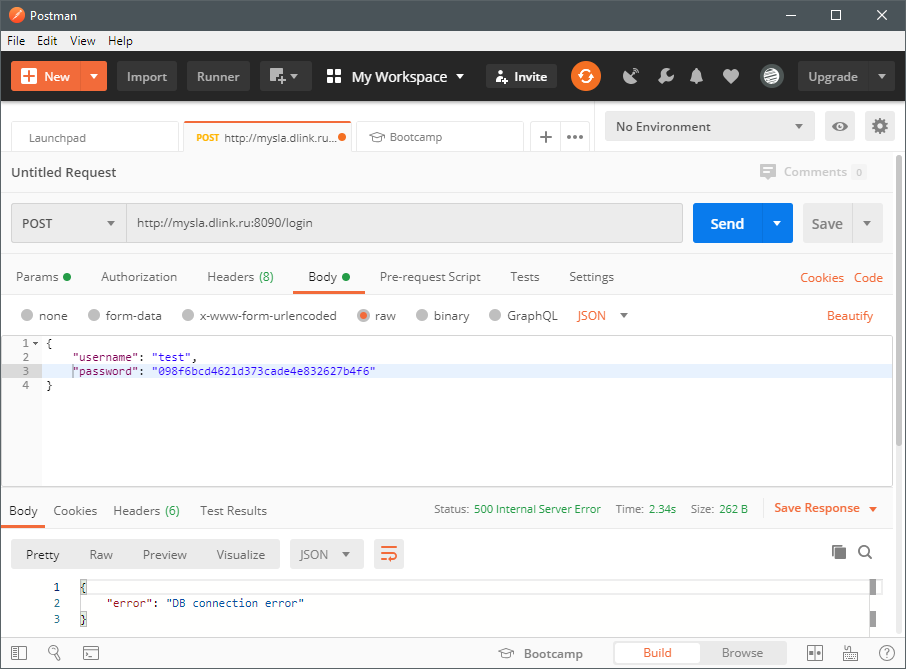


Рисунок . – Пользовательский интерфейс инструмента Postman

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 c 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].

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

# Проектная часть

## Проектирование пользовательского интерфейса

### 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 . – 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.

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.

### 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.

## 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 proxy-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 proxy может быть осуществлён с любого устройства, подключенного к сети Internet [11].

Аутентификация осуществляется с помощью вызова метода POST /login c 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.

## 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 NgModule.

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 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.

## 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.

### 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.

### 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 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.

### Компонент «Панель инструментов»

### Компонент «Подвал»

### Компонент «Авторизация»

### Компонент «Основная информация»

### Компонент «Поиск»

### Компонент «Диалог с результатами»

### Сервис «Общий»

### 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 экрана, 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..

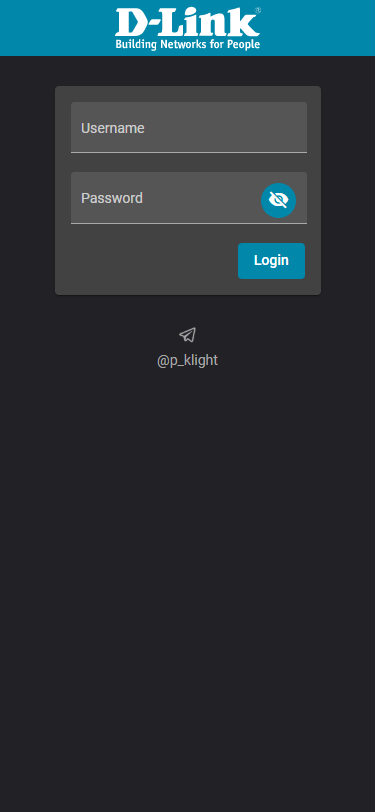
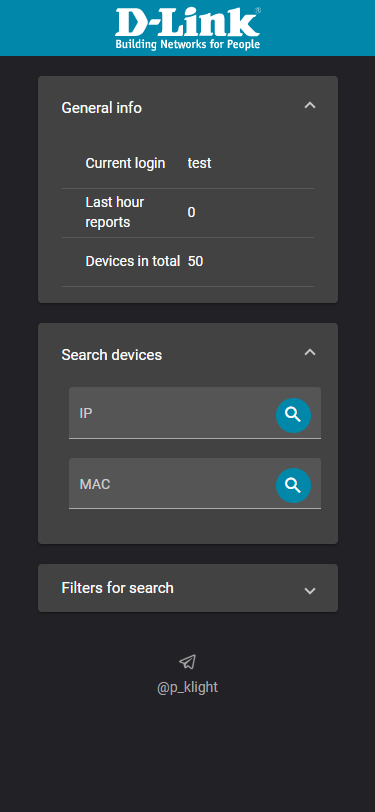
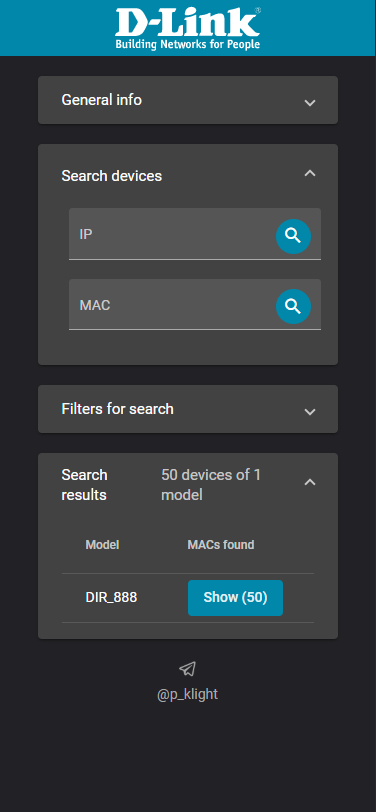
  

Рисунок . - – Скриншоты некоторых страниц в мобильной версии системы

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

# Разработка программной документации

## Руководство системного программиста

## Руководство пользователя

# Тестирование программного обеспечения

# 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.

В ходе тестирования были доказаны:

* работоспособность;
* достоверность;
* отказоустойчивость;
* адаптивность;
* кросплатформенность.

# 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 c.
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. ЛИСТИНГИ ОСНОВНЫХ ПРОГРАММНЫХ МОДУЛЕЙ

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=swap" rel="stylesheet">

  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

</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 { AppRoutingModule } 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,

    HttpClientJsonpModule,

    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, HttpClientJsonpModule, HttpClientModule } from '@angular/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($event)" (loadMainPageEvent)="receiveIsAuth($event)"

    (sendUsernameEvent)="receiveUsername($event)" [currentSLAURL]="currentSLAURL" [proxyURL]="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]="currentSLAURL" [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.сss

#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)="passwordInput($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: #444446 !important;

    color: #fff!important;

} \*/

.w236pxim {

    width:236px !important;

}

main-top.component.ts

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

import { HttpClient, HttpHeaders, HttpClientJsonpModule, 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 + this.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

  ];

  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 serach 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(dataToProccess) {

    if (dataToProccess.length == 1 || dataToProccess.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)="searchByIPInput($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)="SearchFilterLFTD($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)="SearchFilterLFTD($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{{generateEndingOfTheWord(receivedIPMACSearchJSON)}} of {{models.length}} model{{generateEndingOfTheWord(models)}}

                </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.currentLocale);

    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]["64bytes"].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.value, this.sharedData.allResults[i].drop.time),

          fragments: this.generateStringValueDate(this.sharedData.allResults[i].fragments.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.