Введение

Необходимо разработать веб-интерфейс SLA-сервера. Веб-интерфейс SLA-сервера представляет собой веб-приложение, позволяющее производить диагностику состояния сети путем анализа данных за различные промежутки времени о состоянии данной сети, полученные с датчиков устройств, что избавляет от ручной работы по сбору необходимых данных.

1 Техническое задание на разработку web-интерфейса SLA-сервера

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.

Web-интерфейс SLA-сервера от компании D-Link разработан на фреймворке AngularJS, который уже устарел и имеет ряд серьезных недостатков. Необходимо разработать свою версию web-интерфейса с использованием следующего поколения фреймворка Angular. В качестве backend'а можно использовать любую подходящую технологию и язык.

Для решения поставленной задачи развернут тестовый SLA-сервер по адресу http://mysla.dlink.ru. Доступна учетная запись test/test. HTTP API доступен онлайн по адресу http://mysla.dlink.ru:8090. Выполнение задания предполагает активное использование инструментов разработки браузера для получения информации о HTTP API.

Доступны следующие HTTP API методы:

– 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.

Аутентификация:

Вызов метода POST /login с полезной нагрузкой {username: username, password: md5.createHash(password)} возвращает объект с полями:

– token – токен доступа;

– permission – права доступа;

– user – имя пользователя.

Поле token следует сохранить и прикреплять к каждому последующему запросу в заголовке запроса «Token».

2 Анализ тестового SLA-сервера

Предоставленный тестовый SLA-сервер представляет собой одностраничное веб-приложение (SPA – Single-Page Application), разработанное на фреймворке AngularJS. Страница «Выбор устройства» тестового сервера представлена на рисунке 1.

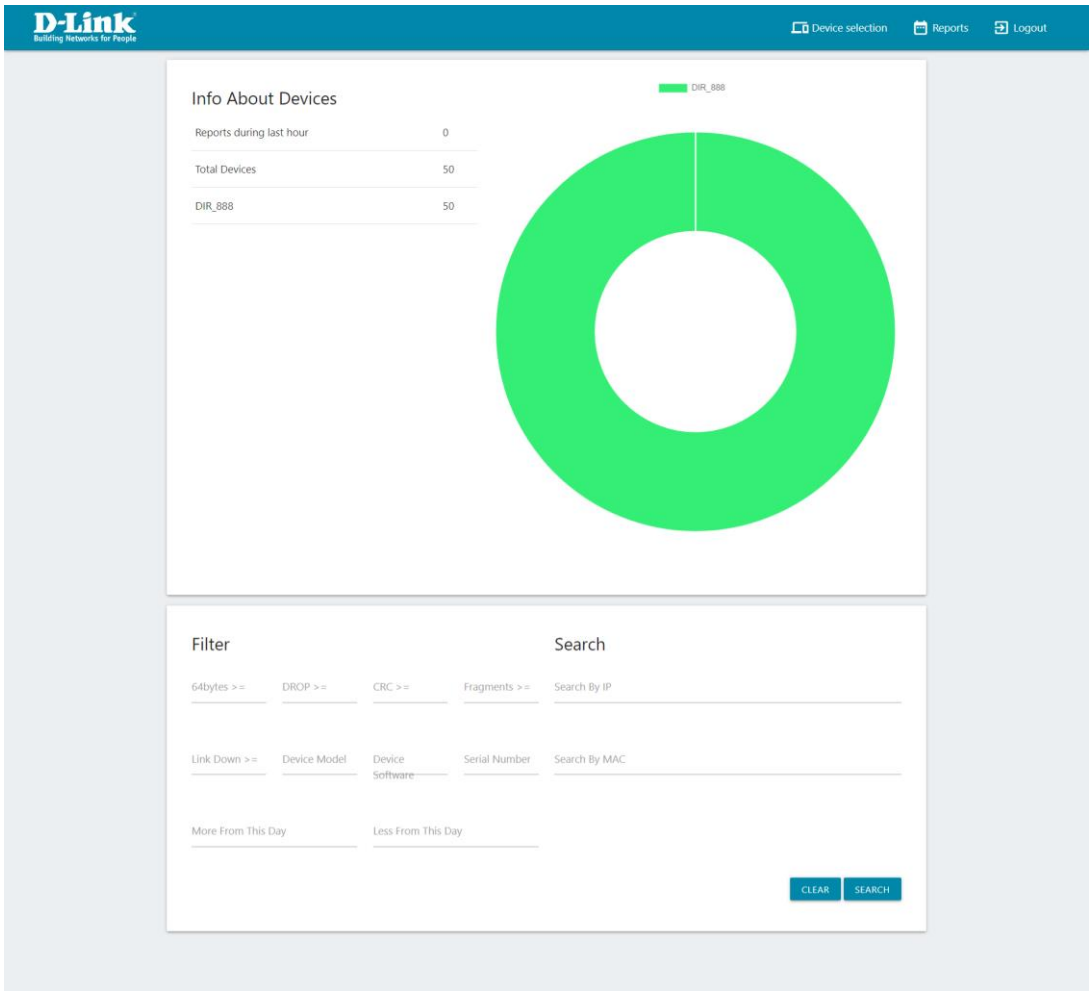

Рисунок 1 – Страница «Выбор устройства»

Тестовый SLA-сервер предоставляет пользователю следующие возможности:

– получение информации о общем количестве устройств в системе;

– поиск устройств по IP или MAC-адресу с дополнительной фильтрацией по параметрам;

– получение общего списка отчетов за указанный день;

– получение списка отчетов по выбранному устройству за указанный день;

– получение подробной информации об устройстве в заданный момент времени;

– получение статистики выбранного устройства по всем отчетам за указанный день.

Интерфейс тестового SLA-сервера выполнен в стиле Google Material – одного из самых популярных дизайнов программного обеспечения.

С помощью средств разработки браузера Google Chrome были проанализированы HTTP запросы, выполняемые web-интерфейсом для получения данных с API-сервера. Процесс анализа HTTP-запросов представлен на рисунке 2.

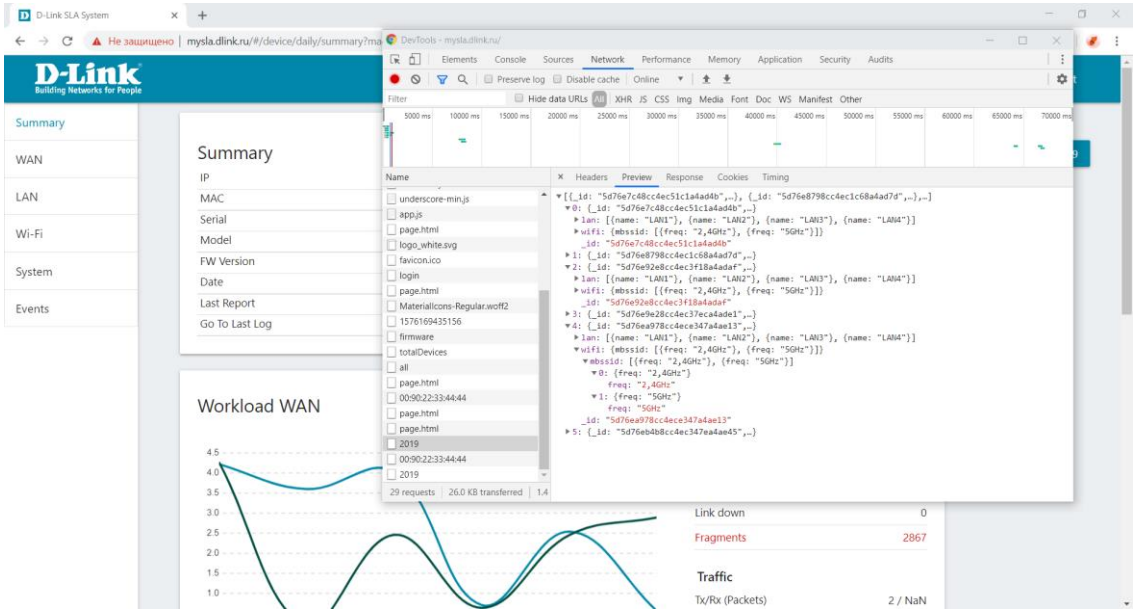

Рисунок 2 – Анализ HTTP-запросов

В ходе данного анализа были дополнительно получены API-методы для поиска устройств с фильтрацией по параметрам.

3 Анализ и тестирование HTTP API

Работа с HTTP API тестового сервера производилась посредством программы Postman. Postman – программное обеспечение для тестирования HTTP API приложений, позволяющее делать HTTP запросы и производить анализ ответов сервера, составлять документацию по данному API. Окно программы Postman изображено на рисунке 3.

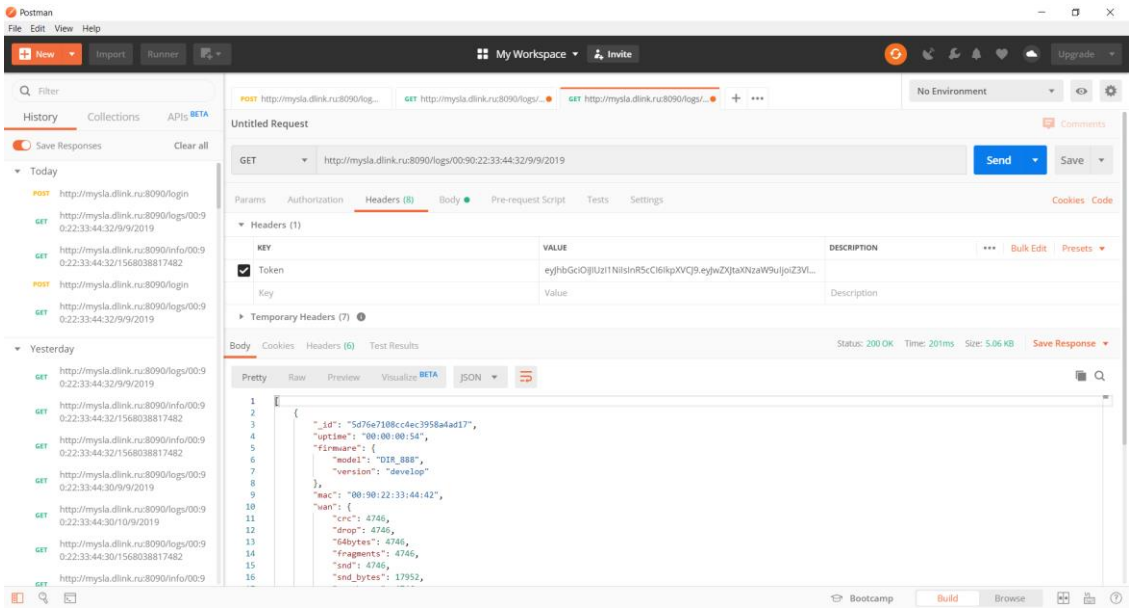

Рисунок 3 – Postman

В ходе тестирования API было выяснено, что в зависимости от кода ответа сервер возвращает различные форматы данных:

– 200 OK (запрос выполнен успешно) – application/json;

– 401 Unauthorized (не хватает действительных учетных данных для целевого ресурса) – text/plain;

– 404 Not Found (запрашиваемый ресурс не найден) – application/json.

Подробное описание форматов данных:

– application/json – JSON (JavaScript Object Notation); формат обмена данными, основанный на формате объектов языка программирования JavaScript;

– text/plain – текстовые данные.

Пример ответа сервера на запрос информации о устройстве в заданный момент времени представлен на рисунке 4 (некоторые вложенные объекты свернуты).

Рисунок 4 – JSON-объект с информацией о устройстве

4 Разработка базы данных

В ходе анализа и тестирования тестового SLA-сервера и HTTP API были выявлены структуры данных, необходимые для разработки реляционной базы данных (БД).

В качестве базы данных была выбрана MySQL 5.6, так как имеет все необходимые для хранения и обработки данных функции и имеет поддержку управления через web-приложение «phpMyAdmin», позволяющим производить администрирование БД через веб-браузер, что избавляет от необходимости наличия системы управления базой данных (СУБД) на компьютере клиента.

Разработка БД производилась в программе «DataGrip 2019.3». DataGrip – программное обеспечение, разработанное компанией JetBrains, предоставляющее единый интерфейс управления реляционными базами данных, имеет консоль запросов, удобную навигацию, а также умное автодополнение кода, его анализ и подсказки. Интерфейс программы представлен на рисунке 5.

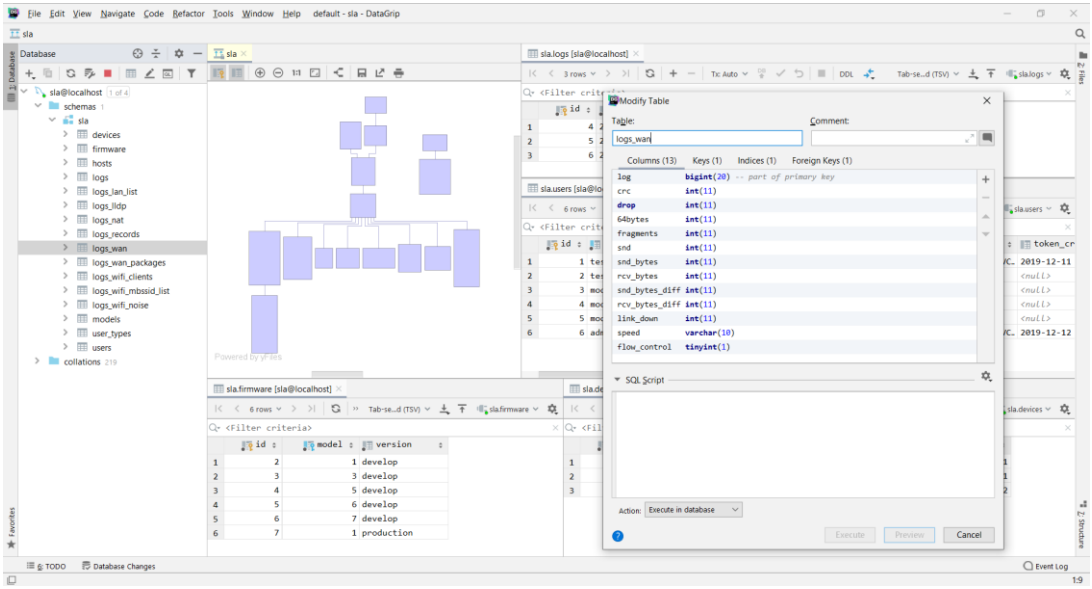

Рисунок 5 – Интерфейс программы DataGrip

Разработанная база данных имеет 16 таблиц:

1 models – модели устройств:

2 firmware – программное обеспечение устройств;

3 devices – информация о устройствах;

4 hosts – информация о главных устройствах системы;

5 logs – системный журнал с записями статистики устройств в указанные моменты времени;

6 logs_wan – информация о WAN в указанный момент времени;

7 logs_wan_packages – информация о передаваемых пакетах по WAN в указанный момент времени;

8 logs_nat – информация о NAT в указанное время;

9 logs_lldp – информация о LLDP в указанное время;

10 logs_lan_list – информация о LAN устройства в заданное время;

11 logs_wifi_mbssid_list – подробная информация о используемых каналах Wi-Fi в указанное время;

12 logs_wifi_clients – информация о клиентах Wi-Fi в указанное время;

13 logs_wifi_noise – информация о мощностях шумов на Wi-Fi каналах устройства в указанный момент времени;

14 logs_records – информация с результатами работы программ Ping и Traceroute в указанное время;

15 user_types – типы пользователей с разграничением прав доступа;

16 users – список пользователей.

Схема данных представлена на рисунке 6.

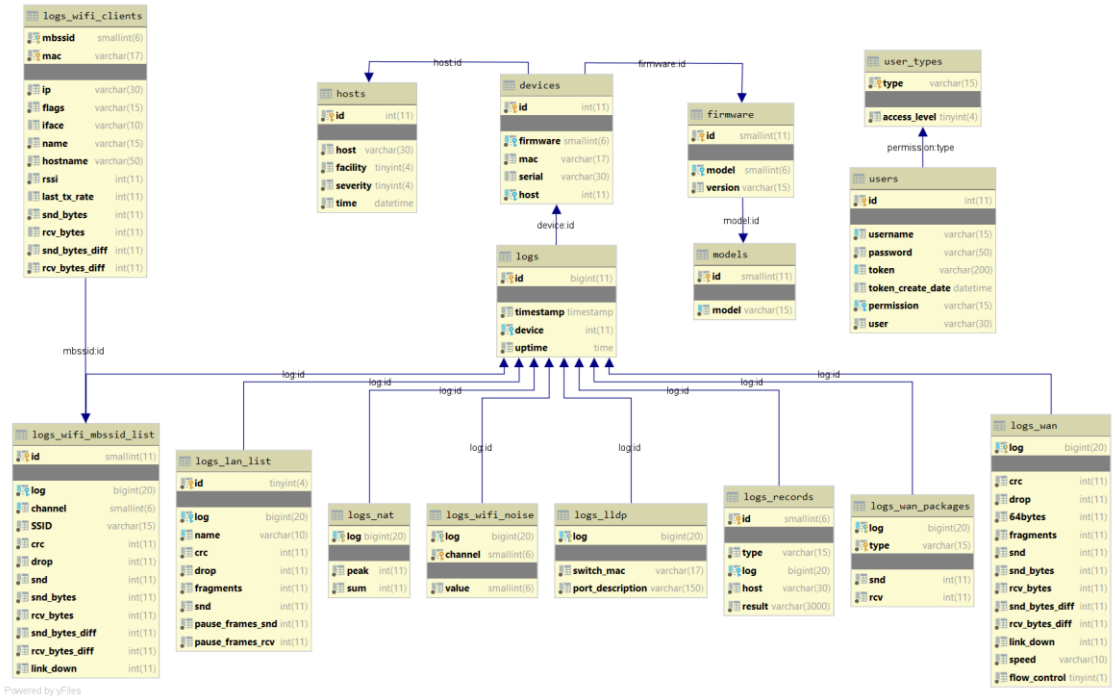

Рисунок 6 – Схема данных

5 Разработка и тестирование серверной части

Из-за ограничений системы безопасности доступ к данным с тестового HTTP API-сервера не может быть предоставлен приложению напрямую. Для решения данной проблемы был разработан прокси-сервер (сервер-посредник), который принимает запросы от приложения, передает их целевому серверу, получает ответ и возвращает его приложению. Данная схема передачи данных изображена на рисунке 7.

Рисунок 7 – Схема передачи данных

Для разработки прокси-сервера была использована платформа Node.js. Node.js – активно развивающаяся программная платформа, основанная на JavaScript движке V8, позволяющая языку программирования JavaScript взаимодействовать с устройствами ввода-вывода. Также Node.js позволяет JavaScript прослушивать порты устройства и отвечать на запросы, что позволяет реализовать веб-сервер.

В качестве основы веб-сервера был выбран фреймворк Koa.js, позволяющий легко обрабатывать HTTP запросы и отвечать на них.

Для передачи запросов целевому серверу использовалась библиотека Request, позволяющая производить HTTP запросы на указанные адреса.

Для реализации проксирования был разработан промежуточный обработчик (middleware) запросов для Koa.js. Исходный код промежуточного обработчика представлен на рисунке 8.

```
const rp = require('request-promise-native');

function proxy(url) {
  return async (ctx) => {
    try {
      ctx.body = await rp({
        method: ctx.method,
        uri: `${url}${ctx.url}`,
        headers: ctx.headers,
        body: ctx.request.body,
        json: true,
      });
    } catch (err) {
      if (!err.response) {
        ctx.status = 500;
        ctx.body = 'Internal destination server error';
        return;
      }
      ctx.status = err.response.statusCode;
      ctx.body = err.response.body;
    }
  };
}

module.exports = proxy;
```

Рисунок 8 – Исходный код промежуточного обработчика

В случае отказа целевого сервера данный промежуточный обработчик сообщит об этом ответом с кодом 500 Internal Server Error (Внутренняя ошибка сервера) с сообщением «Internal destination server error» («Внутренняя ошибка целевого сервера»).

Работа прокси-сервера была протестирована в программе Postman. Тестирование прокси-сервера показано на рисунке 9.

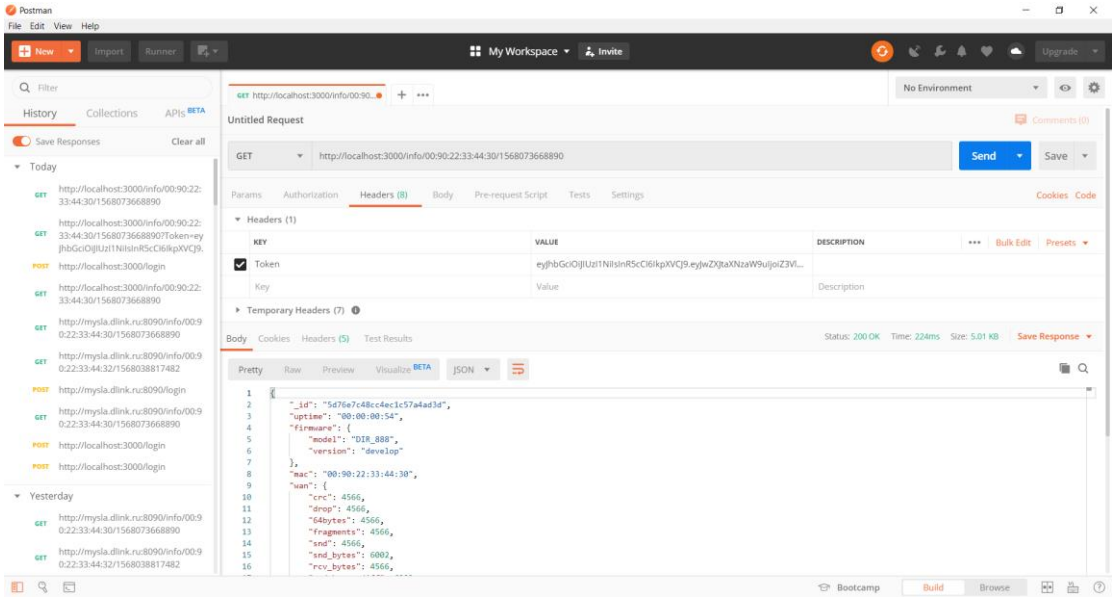

Рисунок 9 – Тестирование прокси-сервера

В ходе тестирования ошибок не обнаружено.

6 Разработка клиентской части

Web-клиент SLA-сервера был разработан при помощи фреймворка Angular 8. Angular – это открытая и свободная платформа для разработки веб-приложений, написанная на языке TypeScript, разрабатываемая командой из компании Google, а также сообществом разработчиков из различных компаний. TypeScript – язык программирования, разрабатываемый компанией Microsoft, являющийся расширением языка программирования JavaScript, в который компилируется в итоге. В качестве языка для написания каскадных таблиц стилей использовался SCSS (Sassy CSS) – это метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS кода и упрощения файлов каскадных таблиц стилей.

Кроме этого использовались дополнительный модули:

– Angular Material – библиотека готовых элементов для разработки интерфейсов в стиле Google Material.

– Bootstrap – набор готовых инструментов для разработки веб-сайтов.

– Moment.js – JavaScript-библиотека для удобной работы с датами.

– Chart.js – JavaScript-библиотека для вывода информации в виде диаграмм.

– TS-MD5 – TypeScript-библиотека для хеширования данных с использованием 128-битного алгоритма MD5.

Дизайн web-клиента выполнен в стиле Google Material, так как он хорошо знаком подавляющему большинству пользователей и имеет приятный вид.

Для разработки клиентской части на Angular использовалась интегрированная среда разработки (IDE) «WebStorm 2019.2.3», разработанная компанией «JetBrains». Данная IDE имеет лучшие инструменты для разработки проектов на JavaScript и TypeScript, а также имеет хорошую поддержку работы с фреймворком Angular. На рисунке 10 представлена данная IDE.

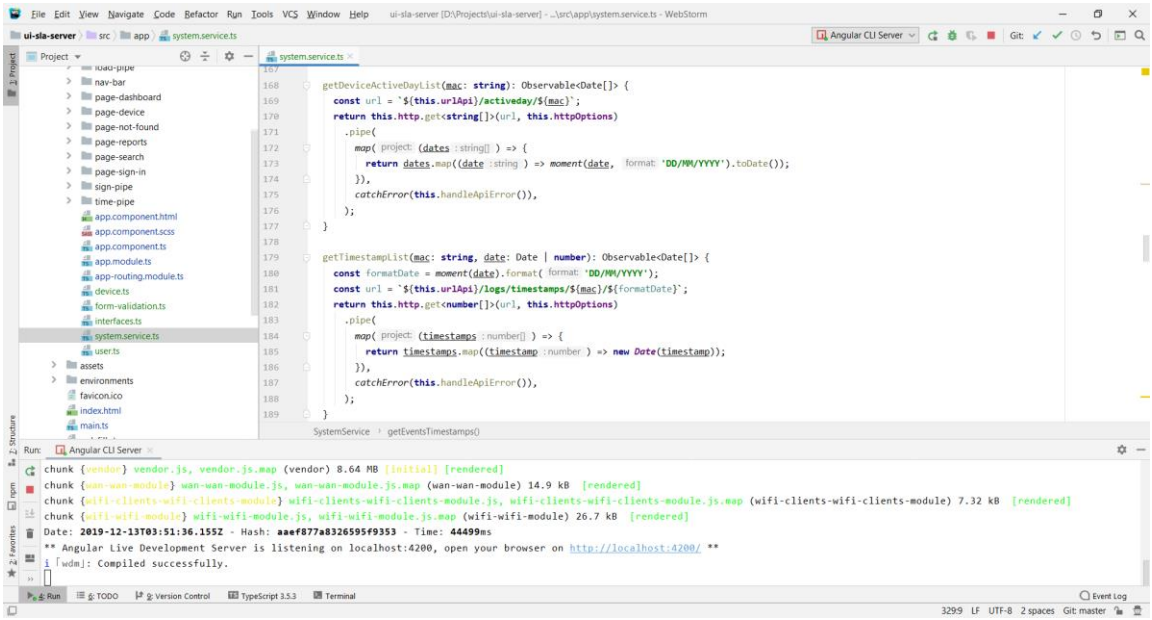

Рисунок 10 – IDE «WebStorm 2019.2.3»

Для создания проекта и работы над ним использовалась консольная утилита @angular/cli. Чтобы установить @angular/cli, требуется npm – менеджер пакетов, входящий в состав Node.js. Для установки используется следующая команда: «npm install --global @angular/cli». Процесс установки показан на рисунке 11.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> npm install --global @angular/cli
C:\Users\wdhap\AppData\Roaming\npm\ng -> C:\Users\wdhap\AppData\Roaming\npm\node_modules\@angular\cli\bin\ng

> @angular/cli@8.3.20 postinstall C:\Users\wdhap\AppData\Roaming\npm\node_modules\@angular\cli
> node ./bin/postinstall/script.js

+ @angular/cli@8.3.20
added 3 packages from 2 contributors, removed 1 package and updated 24 packages in 28.236s
```

Рисунок 11 – Установка @angular/cli

Для создания проекта использовалась команда «ng new ui-sla-server», где «ui-sla-server» – название проекта. Процесс создания проекта показан на рисунке 12.

```
PS D:\Temp> ng new ui-sla-server
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? SCSS   [ https://sass-lang.com/documentation/syntax#scss
 ]
CREATE ui-sla-server/angular.json (3737 bytes)
CREATE ui-sla-server/package.json (1299 bytes)
CREATE ui-sla-server/README.md (1029 bytes)
CREATE ui-sla-server/tsconfig.json (543 bytes)
CREATE ui-sla-server/tslint.json (1953 bytes)
CREATE ui-sla-server/.editorconfig (246 bytes)
CREATE ui-sla-server/.gitignore (631 bytes)
CREATE ui-sla-server/browserslist (429 bytes)
CREATE ui-sla-server/karma.conf.js (1025 bytes)
CREATE ui-sla-server/tsconfig.app.json (270 bytes)
CREATE ui-sla-server/tsconfig.spec.json (270 bytes)
CREATE ui-sla-server/src/favicon.ico (948 bytes)
CREATE ui-sla-server/src/index.html (297 bytes)
CREATE ui-sla-server/src/main.ts (372 bytes)
CREATE ui-sla-server/src/polyfills.ts (2838 bytes)
CREATE ui-sla-server/src/styles.scss (80 bytes)
CREATE ui-sla-server/src/test.ts (642 bytes)
CREATE ui-sla-server/src/assets/.gitkeep (0 bytes)
CREATE ui-sla-server/src/environments/environment.prod.ts (51 bytes)
CREATE ui-sla-server/src/environments/environment.ts (662 bytes)
CREATE ui-sla-server/src/app/app-routing.module.ts (246 bytes)
CREATE ui-sla-server/src/app/app.module.ts (393 bytes)
CREATE ui-sla-server/src/app/app.component.html (25530 bytes)
CREATE ui-sla-server/src/app/app.component.spec.ts (1119 bytes)
CREATE ui-sla-server/src/app/app.component.ts (218 bytes)
CREATE ui-sla-server/src/app/app.component.scss (0 bytes)
CREATE ui-sla-server/e2e/protractor.conf.js (808 bytes)
CREATE ui-sla-server/e2e/tsconfig.json (214 bytes)
CREATE ui-sla-server/e2e/src/app.e2e-spec.ts (646 bytes)
CREATE ui-sla-server/e2e/src/app.po.ts (262 bytes)
npm WARN deprecated core-js@2.6.11: core-js@<3 is no longer maintained and not recommended for usage due to the number o
f issues. Please, upgrade your dependencies to the actual version of core-js@3.
```

Рисунок 12 – Создание проекта Angular

Проект Angular имеет структуру файлов, показанную на рисунке 13.

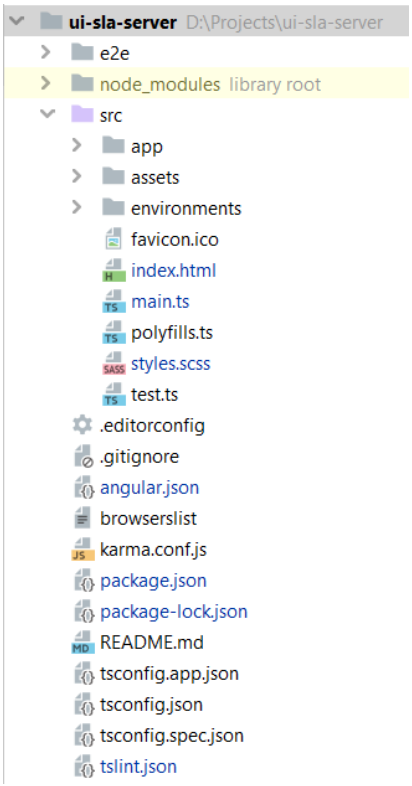

Рисунок 13 – Структура проекта Angular

В директории app находятся основные файлы проекта – модули и компоненты этих модулей. Модульная система обеспечивает расширяемость проекта и упрощает дальнейшую поддержку приложения.

Для создания модулей и компонентов используется @angular/cli.

Для создания модулей/компонентов используется команда следующего вида: «ng generate [module | component] name», где name – имя создаваемого модуля/компонента.

Компоненты и модули состоят из файлов трех типов:

– .html – шаблон модуля/компонента на языке гипертекстовой разметки HTML с расширением его возможностей путем добавления шаблонных выражений от Angular (пример шаблона приведен на рисунке 14);

– .scss – файлы SCSS стилей (пример приведен на рисунке 15);

– .ts – файлы TypeScript, содержащие регистрацию, описание и логику модуля/компонента (пример компонента показан на рисунке 16).

app.component.html

```
<ngx-loading-bar color="#fff" [includeSpinner]="false"></ngx-loading-bar>
<app-nav-bar></app-nav-bar>
<p class="text-center my-5 text-secondary" *ngIf="isLoading || isTrySignIn">
  Please wait...
</p>
<router-outlet *ngIf="!isLoading && !isTrySignIn"></router-outlet>
```

Рисунок 14 – Шаблон корневого компонента приложения

styles.scss

```scss
@mixin set-grid($param, $start, $end) {
  grid-#{$param}-start: $start;
  grid-#{$param}-end: $end;
}

@mixin set-column($start, $end) {
  @include set-grid('column', $start, $end);
}

@mixin set-row($start, $end) {
  @include set-grid('row', $start, $end);
}

html, body {
  height: 100%;
}

body {
  margin: 0;
  font-family: Roboto, "Helvetica Neue", sans-serif;
  background-color: #f8f8f8;
}

mat-card {
  mat-card-header.card-header-with-action {
    display: flex;
    justify-content: space-between;
    align-items: center;

    .card-title-action {
      margin-bottom: 12px;
    }
```

Рисунок 15 – Файл глобальных стилей проекта

page-sign-in.component.ts

```ts
import { Component } from '@angular/core';
import { SystemService } from '../system.service';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Router } from '@angular/router';
import { FormControl, FormGroup, FormGroupDirective, NgForm, Validators } from '@angular/forms';
import { ErrorStateMatcher } from '@angular/material';

export class MyErrorStateMatcher implements ErrorStateMatcher {
  isErrorState(control: FormControl | null, form: FormGroupDirective | NgForm | null): boolean {
    const isSubmitted = form && form.submitted;
    return !!(control && control.invalid && (control.dirty || control.touched || isSubmitted));
  }
}

@Component({
  selector: 'app-page-sign-in',
  templateUrl: './page-sign-in.component.html',
  styleUrls: ['./page-sign-in.component.scss'],
})
export class PageSignInComponent {
  isLoad = false;
  signInForm = new FormGroup( controls: {
    username: new FormControl( formState: '', Validators.required),
    password: new FormControl( formState: '', Validators.required),
  });
  matcher = new MyErrorStateMatcher();

  constructor(
    private router: Router,
    private system: SystemService,
    private snackBar: MatSnackBar,
  ) {}
```

Рисунок 16 – TypeScript-файл компонента формы входа

Для создания навигации в проекте используется настройка маршрутизации. Файл с маршрутизацией относительно корня проекта представлен на рисунке 17.

app-routing.module.ts

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  { path: '', redirectTo: 'dashboard', pathMatch: 'full' },
  { path: 'sign-in', loadChildren: () => import('./page-sign-in/page-sign-in.module').then(m => m.PageSignInModule) },
  { path: 'dashboard', loadChildren: () => import('./page-dashboard/page-dashboard.module').then(m => m.PageDashboardModule) },
  { path: 'reports', loadChildren: () => import('./page-reports/page-reports.module').then(m => m.PageReportsModule) },
  { path: 'device', loadChildren: () => import('./page-device/page-device.module').then(m => m.PageDeviceModule) },
  { path: 'search', loadChildren: () => import('./page-search/page-search.module').then(m => m.PageSearchModule) },
  { path: '**', loadChildren: () => import('./page-not-found/page-not-found.module').then(m => m.PageNotFoundModule) },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
export class AppRoutingModule {}
```

Рисунок 17 – Описание маршрутизации относительно корня проекта

В ходе работы над проектом было разработано 23 модуля, 33 компонента и написана разметка 28 шаблонов.

7 Тестирование клиентской части

Тестирование работоспособности веб-приложения было проведено в трех браузерах:

– Microsoft Edge;
– Mozilla Firefox;
– Google Chrome.

Тестирование в браузере Microsoft Edge представлено на рисунке 18.

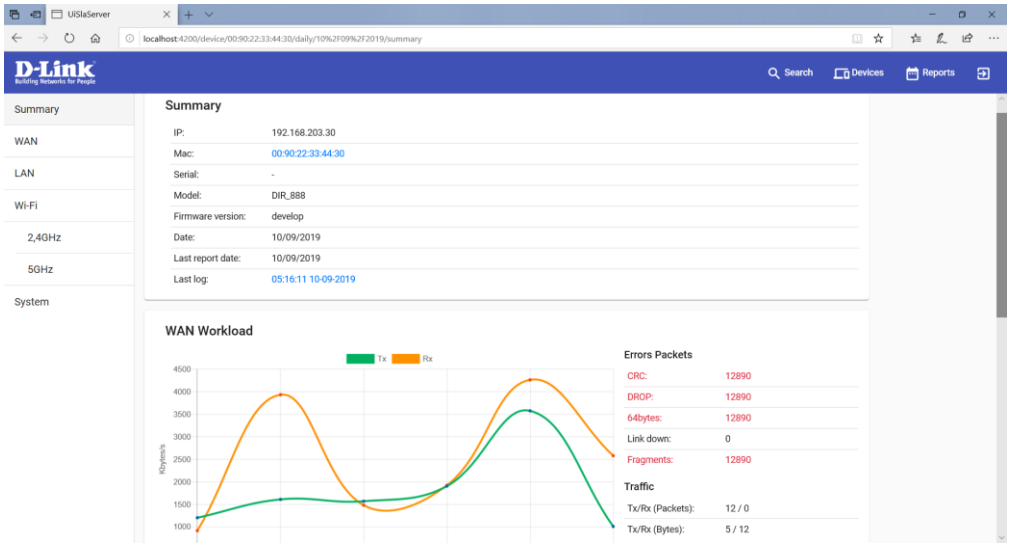

Рисунок 18 – Тестирование в браузере Microsoft Edge

Тестирование в браузере Mozilla Firefox показано на рисунке 19.

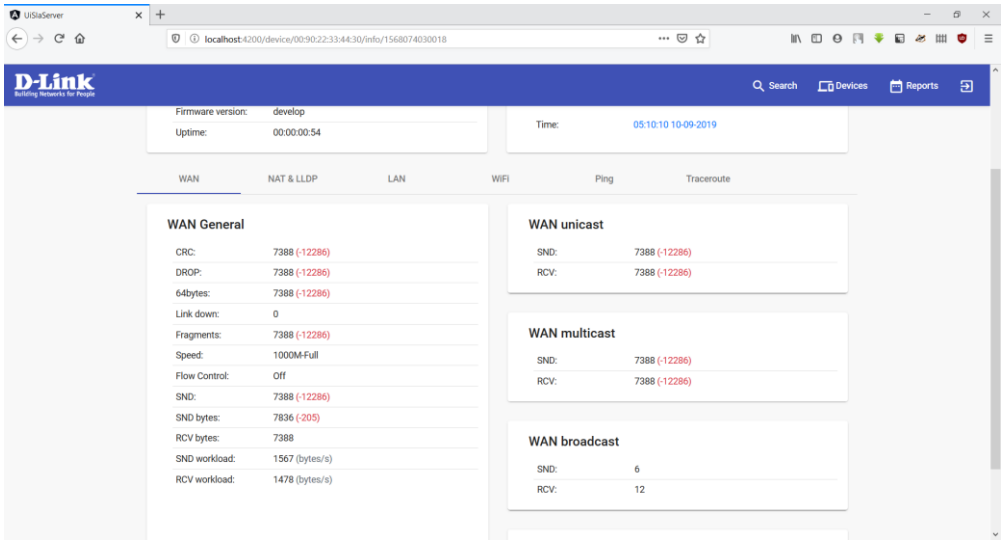

Рисунок 19 – Тестирование в браузере Mozilla Firefox

Тестирование в браузере Google Chrome изображено на рисунке 20.

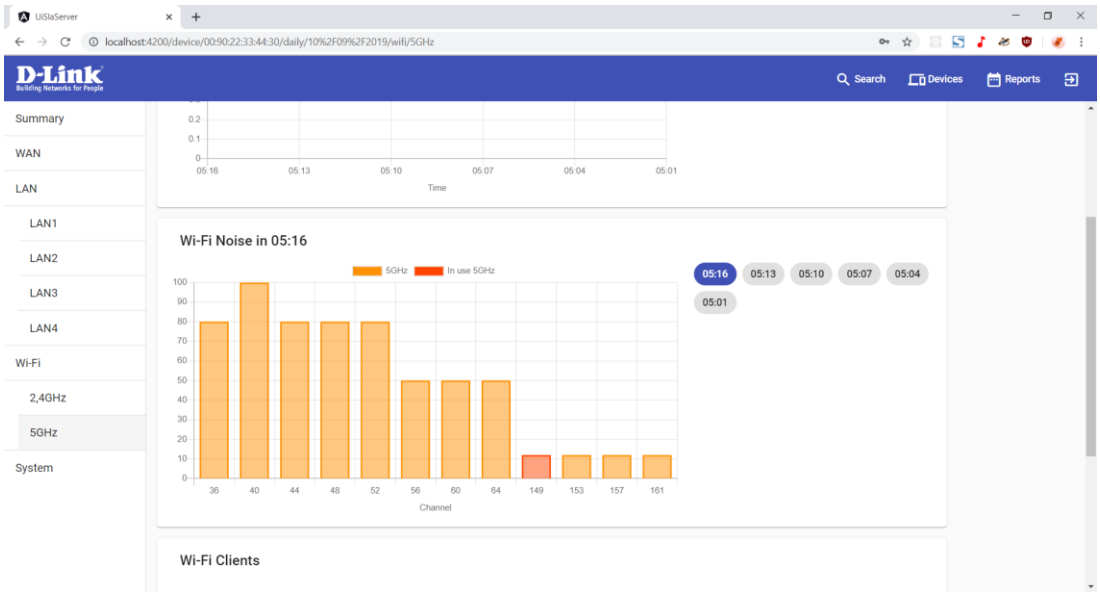

Рисунок 20 – Тестирование в браузере Google Chrome

В ходе тестирование ни в одном из браузеров ошибок выявлено не было.

8 Итоговый результат

При запуске приложения открывается форма входа. Данная форма имеет 2 поля: «Username» и «Password», обязательных для заполнения. В случае ввода некорректных или неверных данных будет выведено соответствующее уведомление. На рисунке 21 представлена попытка входа без ввода имени пользователя.

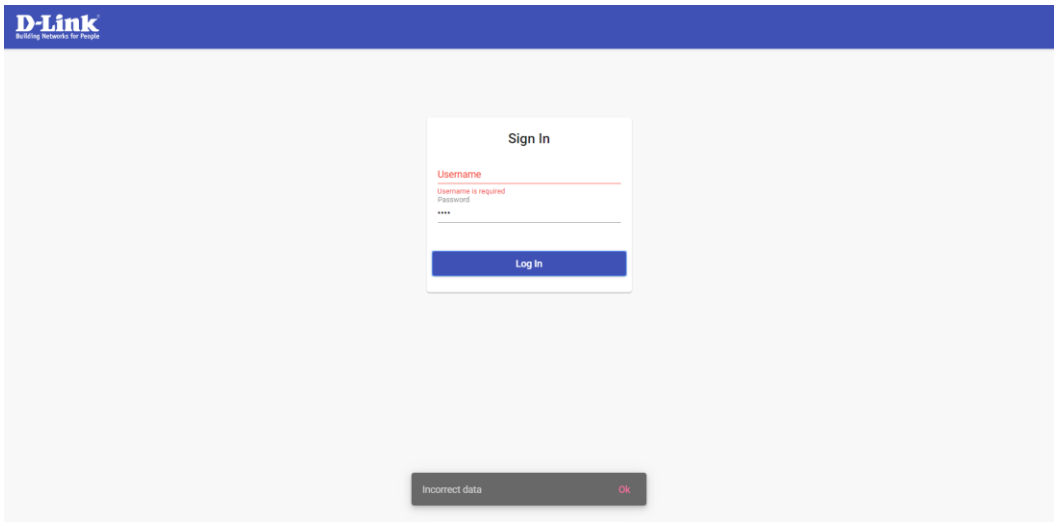

Рисунок 21 – Форма входа

После успешной авторизации будет открыта страница «Devices» («Устройства»), изображенная на рисунке 22.

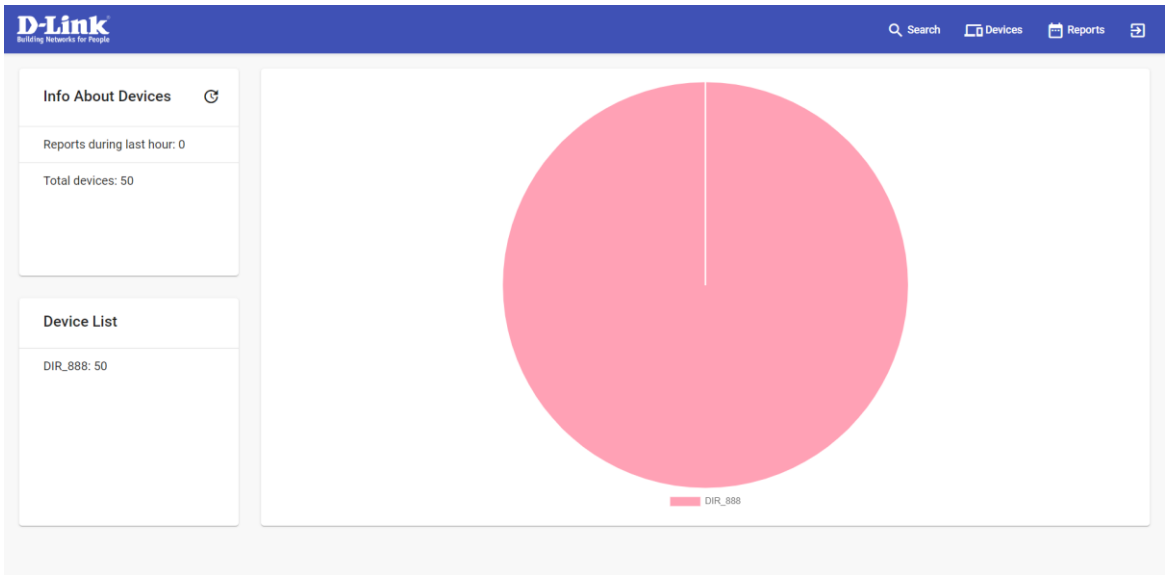

Рисунок 22 – Страница «Устройства»

На данной странице расположено 3 карточки:

– «Info About Devices» («Информация о устройствах») – содержащая информацию о количестве отчетов, сделанных за последний час и общее количество устройств в системе.

– «Device List» («Список устройств») – содержит список моделей устройств и количество устройств данной модели в системе.

– Карточка с круговой диаграммой наглядно показывает процентное соотношение количества устройств определенной модели в системе.

Если выбрать интересующую модель устройства из списка или на диаграмме, то откроется диалоговое окно, содержащее список устройств данной модели (рисунок 23). Кнопка «Log list» («Список логов») напротив устройства позволяет перейти на страницу со списком записей статистики для текущего устройства (рисунок 24).

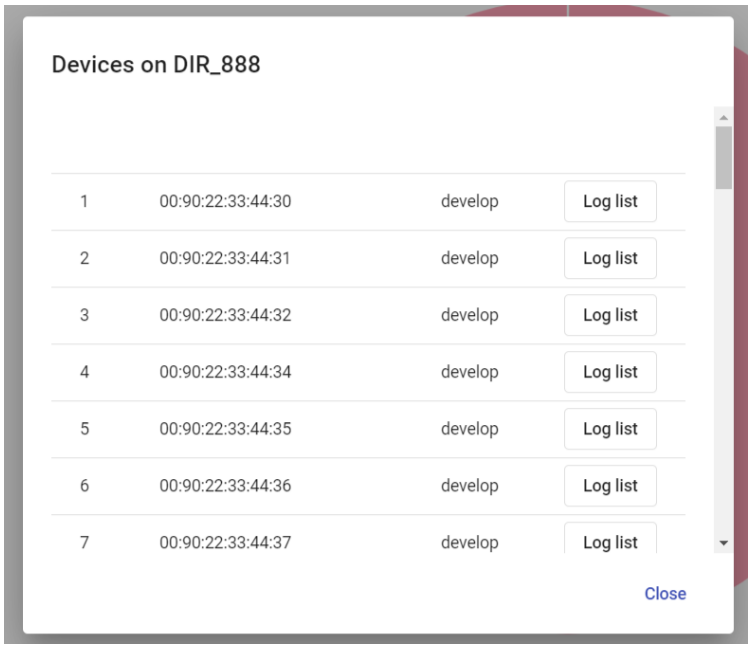

Рисунок 23 – Список устройств

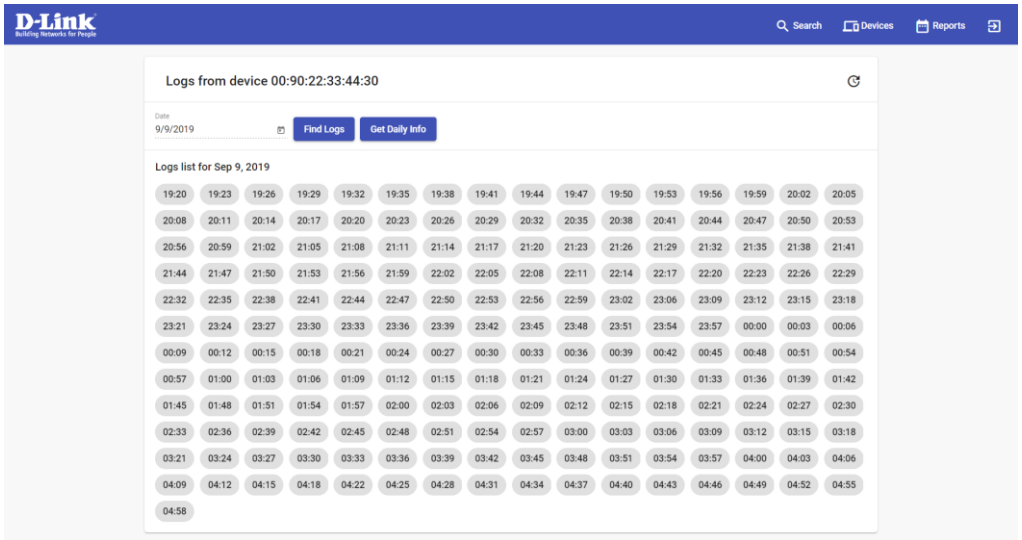

Рисунок 24 – Страница со списком логов устройства

На странице со списком записей находится форма с полем выбора даты (можно выбрать только ту дату, на которую есть записи), кнопкой поиска записей («Find Logs») и кнопкой «Get Daily Info», позволяющей получить статистику за весь день. Кнопка поиска записей загружает временные метки всех записей за этот день и выводит их в виде ряда фишек. При выборе фишки будет показана подробная информация по текущему устройству на заданный момент времени. Страница с подробной информацией изображена на рисунке 25.

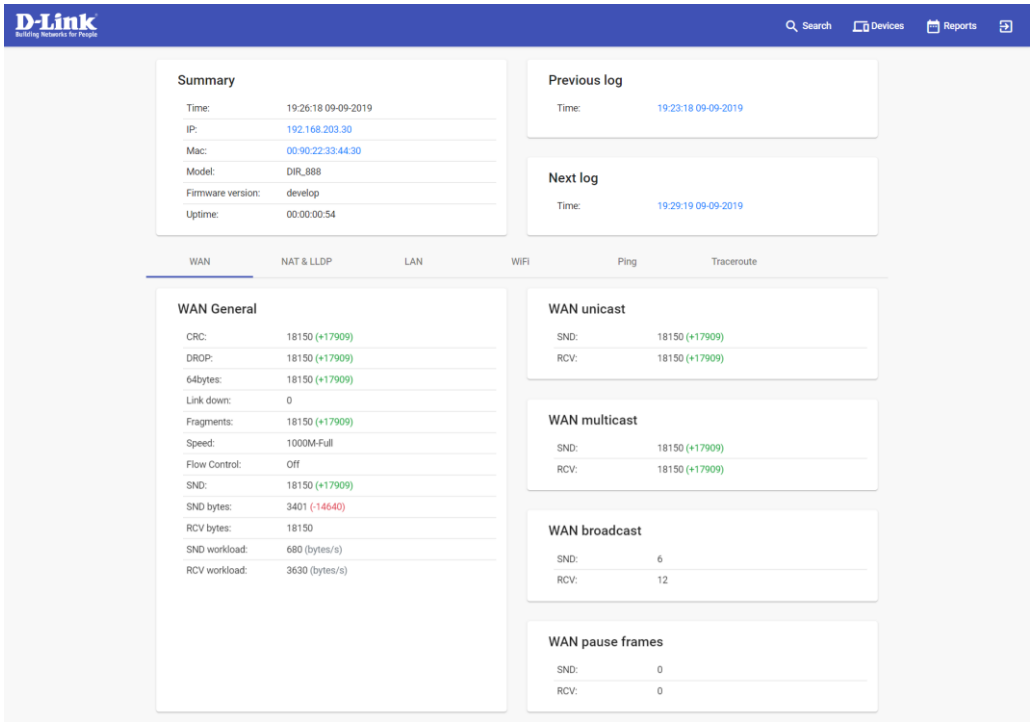

Рисунок 25 – Подробная информация о устройстве на указанный момент времени

Данная страница имеет общую информацию о записи (карточка «Summary»), а также позволяет перейти к предыдущей или следующей записи при их наличии (карточки «Previous log» и «Next log»). Ниже общей информации имеются вкладки с подробной информацией по указанной категории. В случае наличия предыдущей записи в подробной информации будет указана разница текущих и предыдущих показаний. В приложении А представлено содержимое всех вкладок.

Если на странице со списком логов устройства нажать на кнопку получения статистики за весь день, то будет открыта страница с общей информацией за указанный день, представленная на рисунке 26.

Рисунок 26 – Информация о устройстве за весь день.

Данная страница содержит информацию о устройстве, ссылку на последний лог на текущую дату и статистику о рабочей нагрузке WAN (рисунок 27) и шумах и каналах Wi-Fi (рисунок 28). Также слева имеется навигация для просмотра статистики по отдельным параметрам.

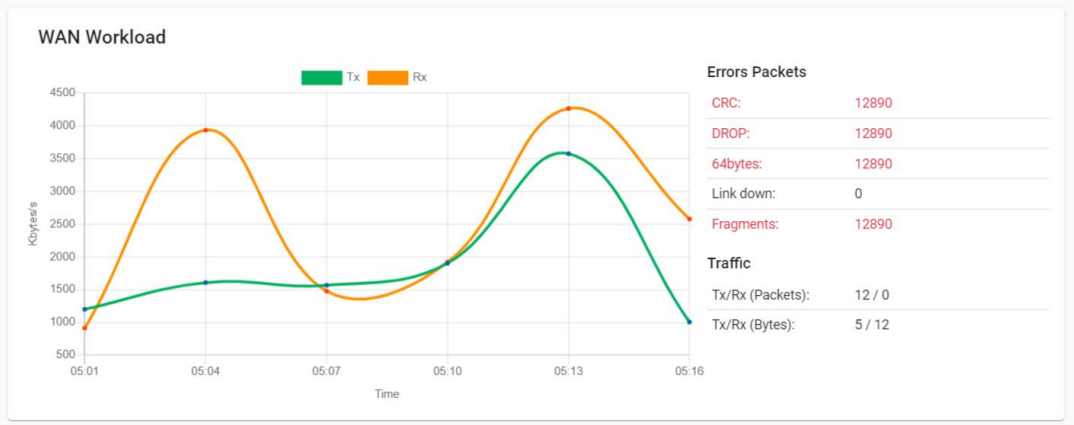

Рисунок 27 – Рабочая нагрузка WAN

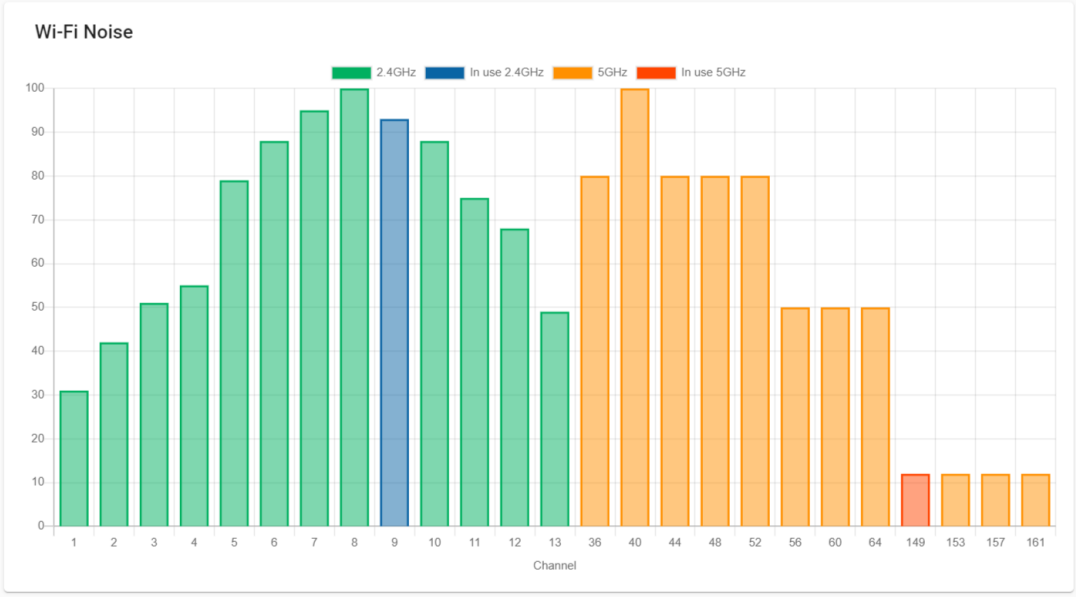

Рисунок 28 – Информация о шумах и каналах Wi-Fi

В разделе WAN представлена статистика о использовании глобальной сети: рабочая нагрузка и потеря пакетов. Данный раздел изображен на рисунке 29.

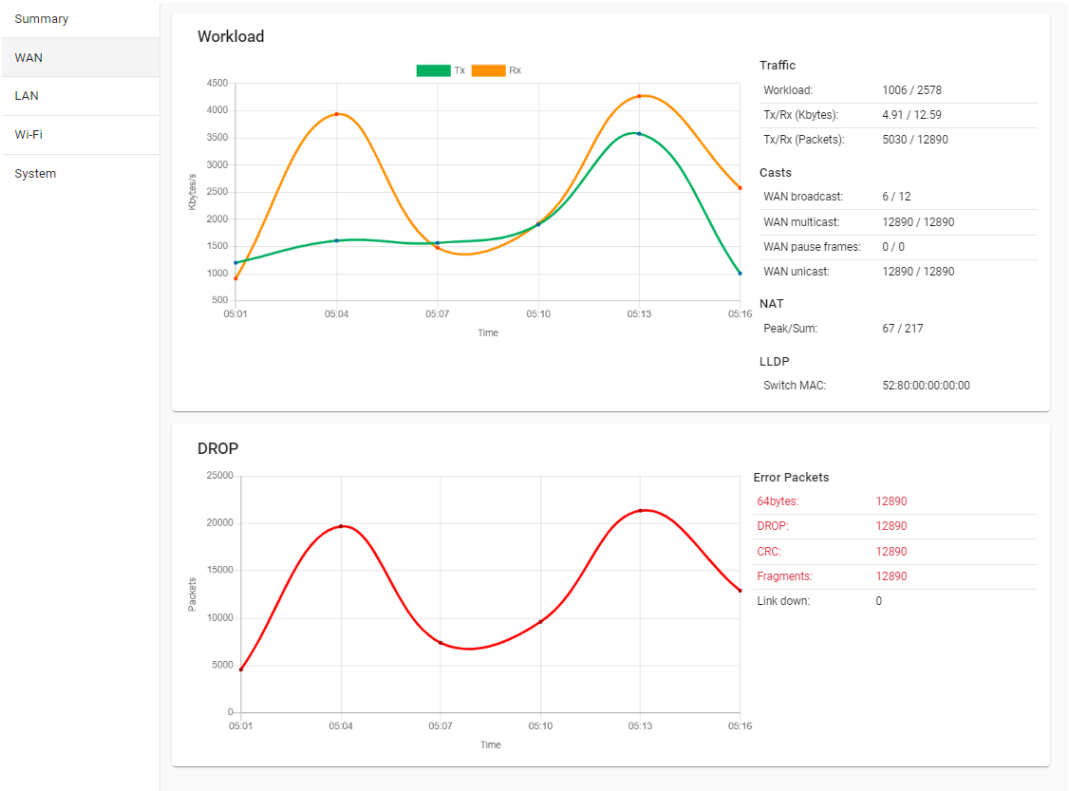

Рисунок 29 – Раздел WAN

Раздел LAN содержит подразделы на каждый из имеющихся разъемов. В данных подразделах представлена статистика по потере пакетов в локальной сети (рисунок 30).

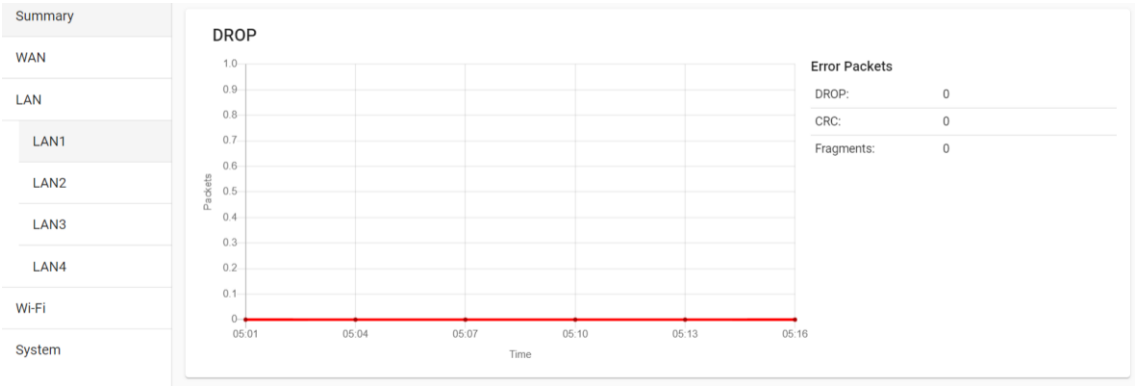

Рисунок 30 – Раздел LAN

Раздел Wi-Fi разбит на подразделы по частоте, поддерживаемых текущим устройством. Данный раздел содержит статистику по выбранной частоте: рабочую нагрузку, уровень шума на каналах за указанное время, количество клиентов в указанное время. Данный раздел показан на рисунке 31.

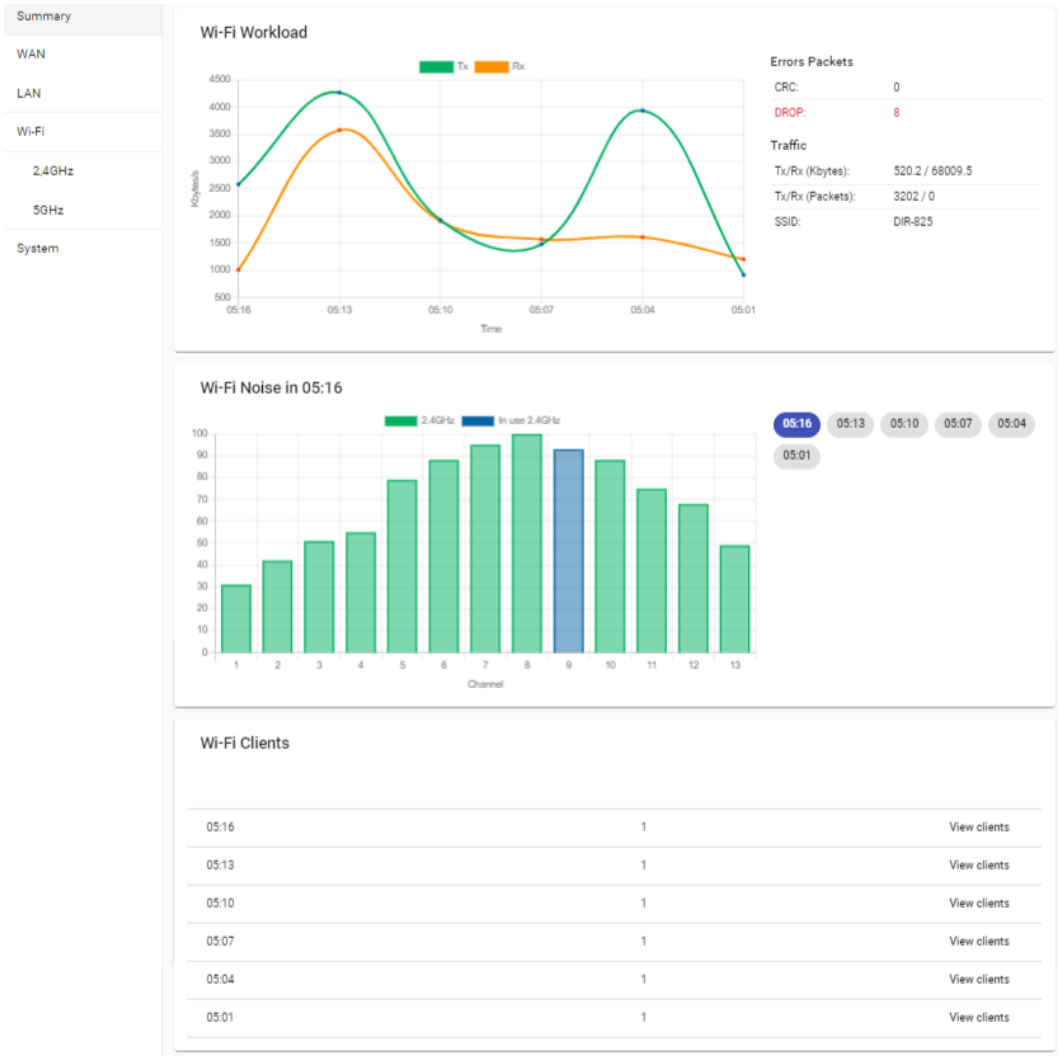

Рисунок 31 – Раздел Wi-Fi

Можно увидеть подробную информацию о клиентах нажав на кнопку «View clients» («Показать клиентов») (рисунок 32).

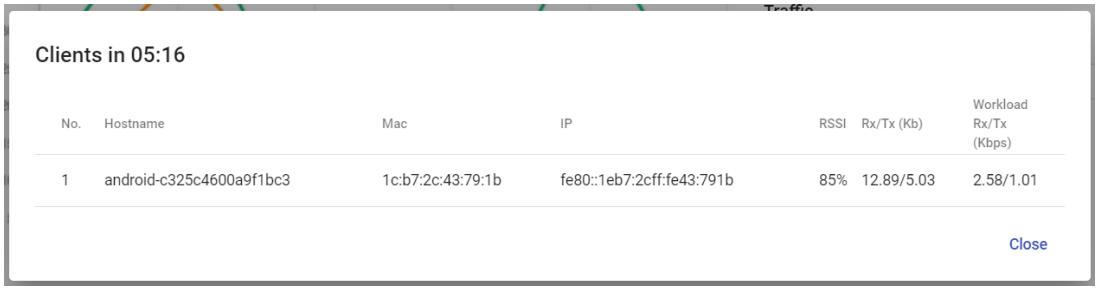

Рисунок 32 – Список клиентов в указанное время

Раздел System содержит список меток времени, в которые использовался центральный процессор, в виде фишек (рисунок 33). При выборе фишки будет открыта подробная информация о устройстве на данный момент времени.

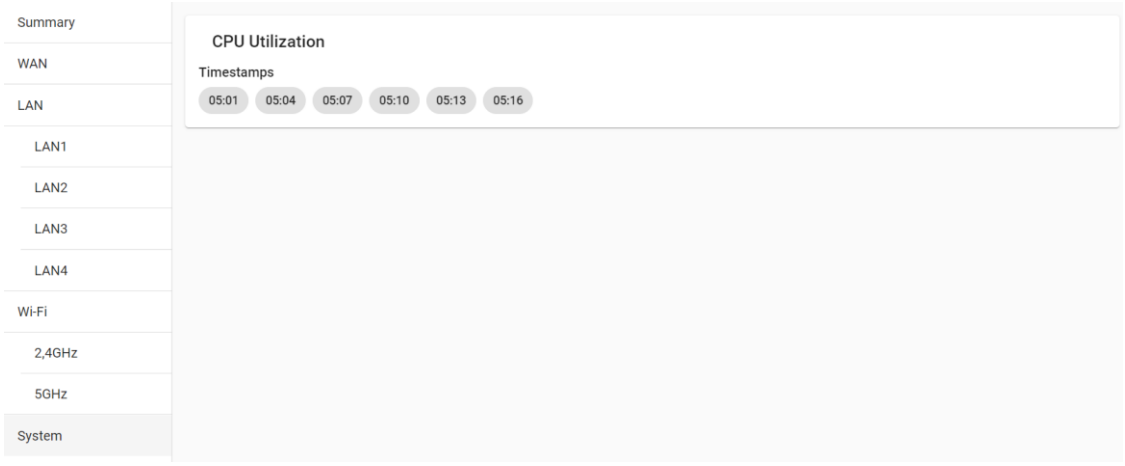

Рисунок 33 – Раздел System

При переходе на страницу «Reports» («Отчеты») в панели навигации, будет открыта страница со списком всех отчетов за указанный день. Данная страница показана на рисунке 34. По умолчанию берется текущая дата.

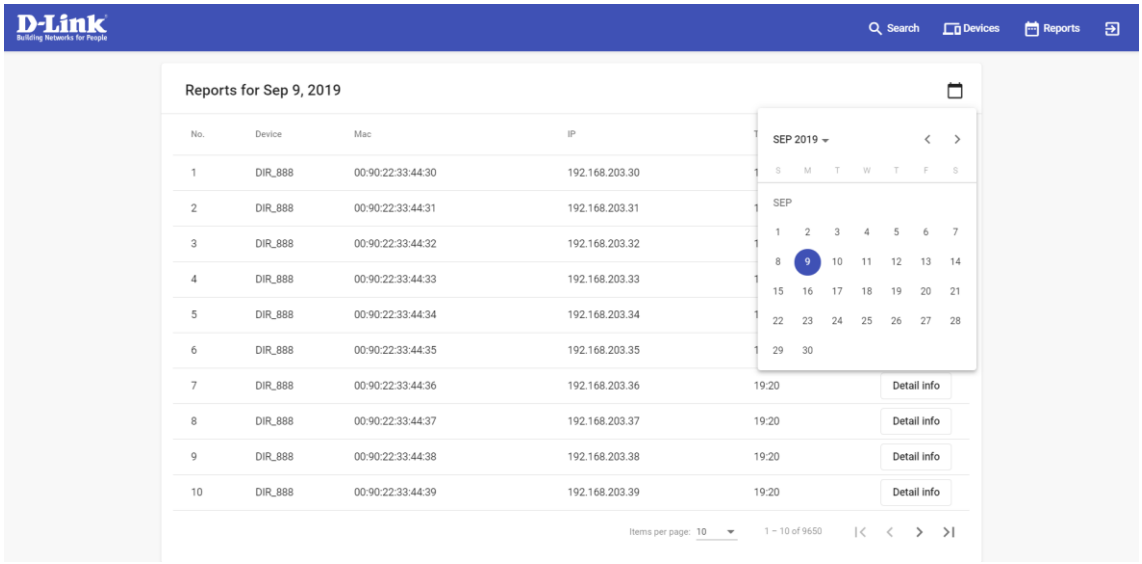

Рисунок 34 – Страница «Отчеты»

Список выведен в виде таблицы с информацией о устройстве, на которое составлен отчет, и меткой времени создания отчета. Список имеет пагинацию. Также существует возможность выбрать требуемое количество записей, выводимых на одной странице.

При нажатии на кнопку «Detail info» («Подробная информация») будет открыта страница с подробной информации о выбранном устройстве в указанный момент времени.

При переходе на страницу «Search» («Поиск») будет открыта страница поиска устройств, представленная на рисунке 35.

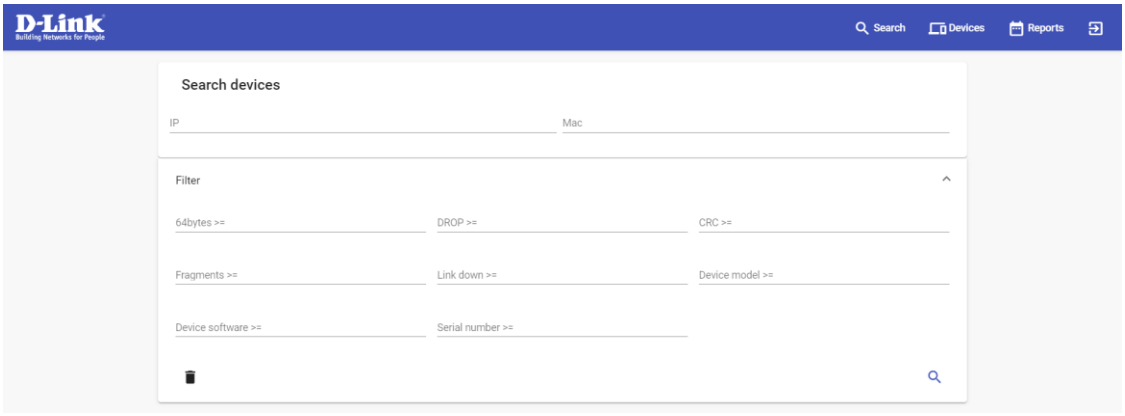

Рисунок 35 – Страница поиска устройств

На данной странице можно производить поиск устройств по IP и/или MAC-адресу и, при необходимости, добавлять фильтры для поиска.

Входные данные проверяются на корректность, а также при вводе IP или MAC-адреса система будет подсказывать подходящие значения, имеющиеся в системе.

В случае поиска с пустыми значениями адресов будут выведены все устройства, имеющиеся в системе (рисунок 36).

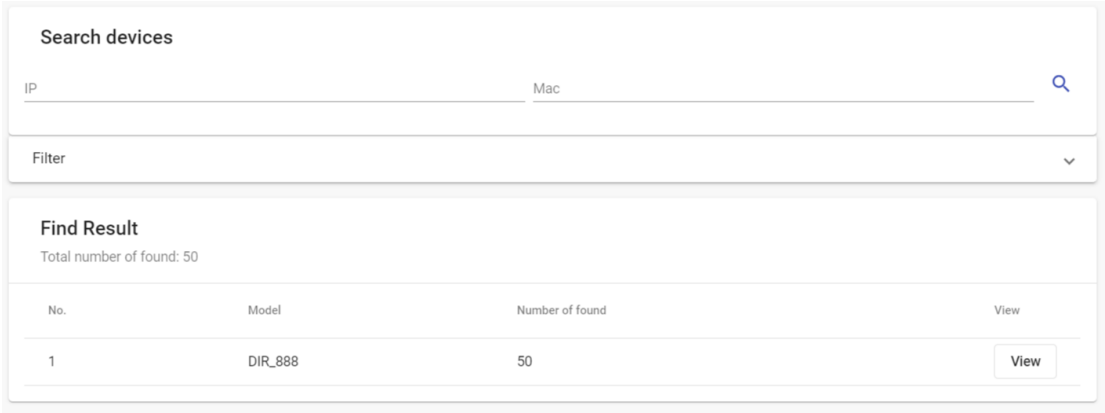

Рисунок 36 – Поиск с пустыми значениями адресов

Результаты группируются по модели устройства. Количество найденных устройств данной модели указано в столбце «Number of found». Чтобы посмотреть список устройств, необходимо нажать на кнопку «View» («Показать») у соответствующей записи. Окно со списком устройств показано на рисунке 37.

Рисунок 37 – Список найденных устройств выбранной модели

В данном списке представлена информация о устройствах и возможность перейти на список логов устройства.

Если поиск производится только по MAC-адресу, то будут выведены устройства, имеющие указанный MAC-адрес.

В случае, если поиск производился по IP адресу, то результаты поиска будут выведены в виде IP-адресов (рисунок 38) со списком устройств, имеющих данный IP-адрес, просмотреть которые можно нажав на кнопку «Devices» соответствующей группы (рисунок 39).

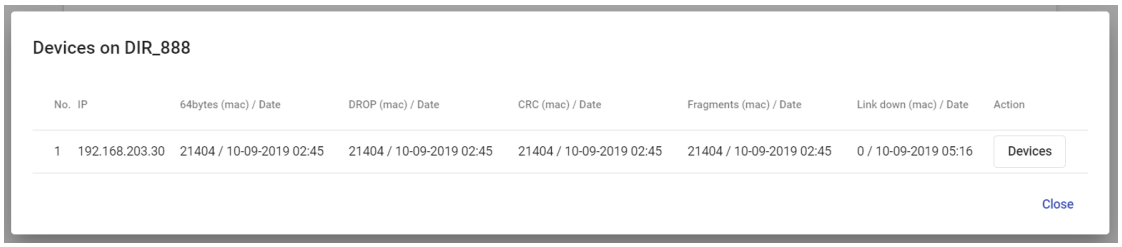

Рисунок 38 – Результаты поиска по IP-адресу

Рисунок 39 – Список устройств, имеющих данный IP адрес

При выборе устройства будет осуществлен переход на страницу со списком логов данного устройства.

Если дополнительно с IP-адресом в поиске указывался MAC-адрес, то внутри групп будет произведена дополнительная фильтрация устройств по указанному MAC-адресу.

При нажатии на иконку выхода в панели навигации, будет осуществлен выход из системы и перенаправление на форму входа.

Заключение

В результате выполнения задания был разработан веб-интерфейс SLA-сервера, позволяющий производить диагностику состояния сети путем анализа данных за различные промежутки времени о состоянии данной сети, полученные с датчиков устройств, что избавляет от ручной работы по сбору необходимых данных.

Также был разработан прокси-сервер для обхода системы безопасности и получения данных с HTTP API-сервера.

Кроме этого была разработана реляционная база данных в системе управления базами данных MySQL 5.6.

Список использованных источников

1 Янг А., Мек Б., Кантелон М. Node.js в действии. 2-е издание. Издательский дом «Питер», 2018 г. – 432 с.

2 Симпсон К. ES6 и не только. O'Reilly, 2017 г. – 336 с.

3 Кириченко А., Хрусталев А. HTML5+CSS3. Основы современного WEB-дизайна. НиТ, 2019 г. – 352 с.

4 Холмс С. Стек MEAN. Mongo, Express, Angular, Node. Издательский дом «Питер», 2017 г. – 496 с.

5 Дакетт Дж. HTML и CSS. Разработка и дизайн веб-сайтов. Эксмо, 2019 г. – 480 с.

6 https://learn.javascript.ru

7 https://angular.io

8 https://material.angular.io

9 https://rxjs.dev

10 https://www.chartjs.org/

Приложение А

Подробная информация о устройстве на указанный момент времени

На рисунке А1 показана подробная информация о записи и состоянии WAN.

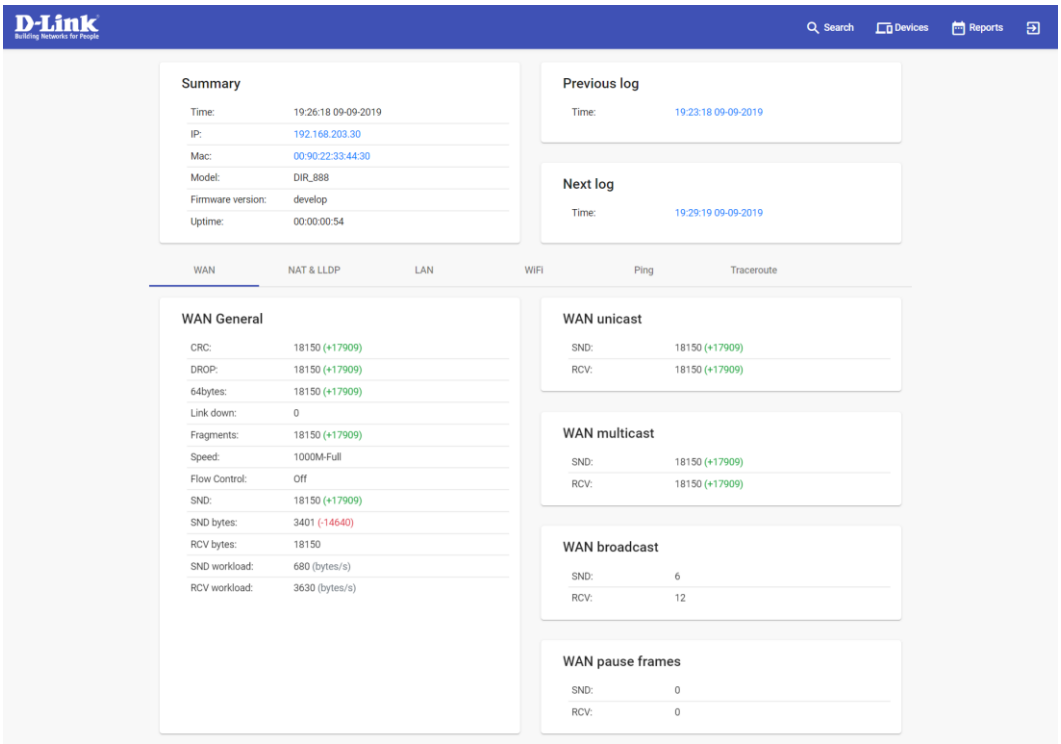

Рисунок А1 – Вкладка WAN

Подробная информация о NAT и LLDP устройства представлена на рисунке А2.

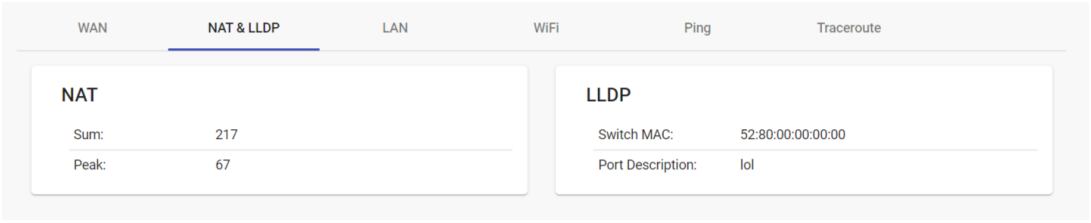

Рисунок А2 – Вкладка NAT & LLDP

Подробная информация о LAN показана на рисунке А3.

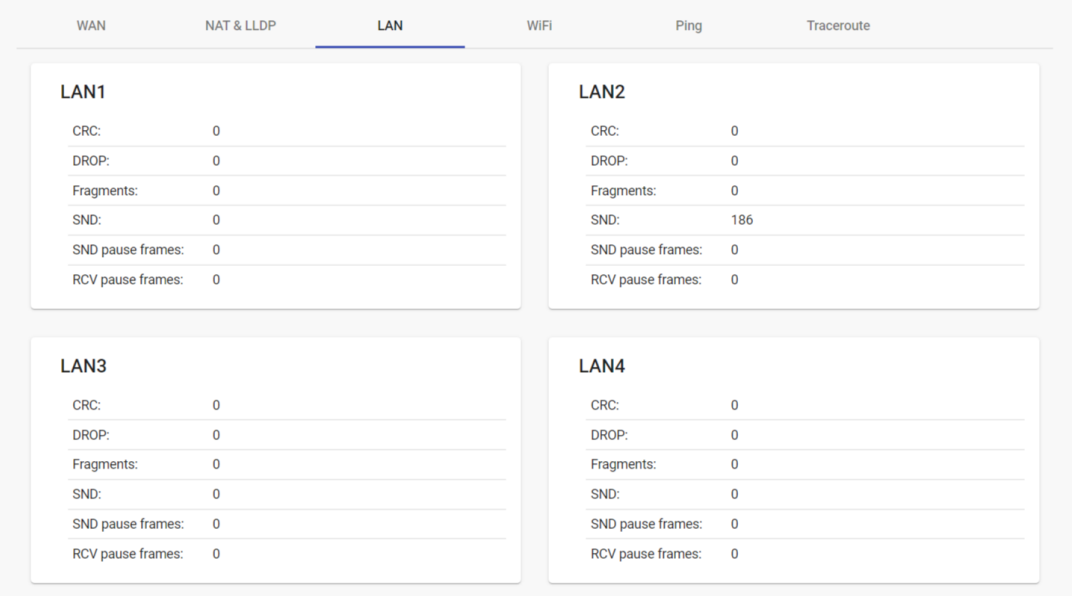

Рисунок А3 – Вкладка LAN

Информация о уровнях шума каналов Wi-Fi, используемых каналах и клиентах Wi-Fi представлена на рисунке А4.

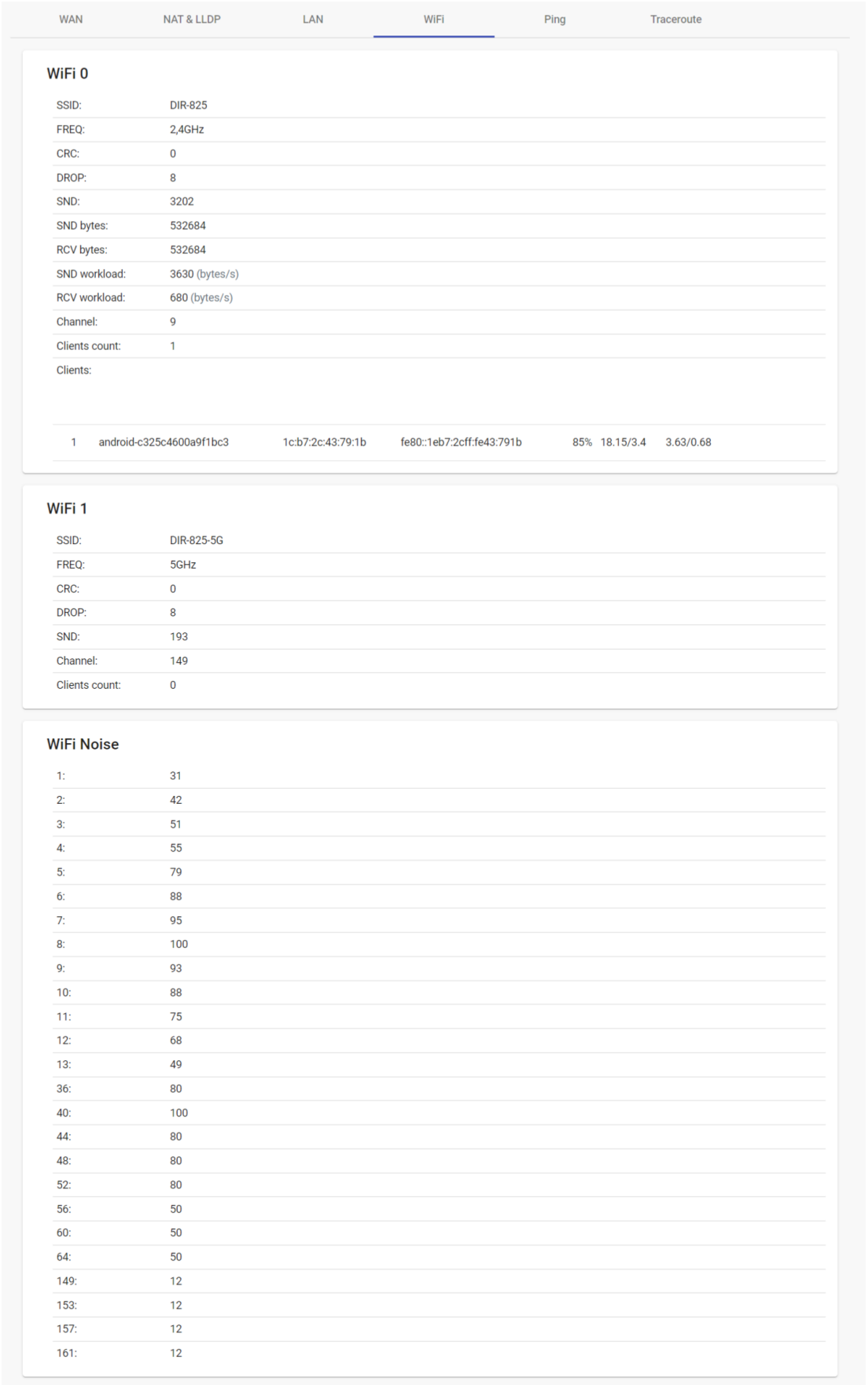

Рисунок А4 – Вкладка WiFi

Результат работы программы Ping представлен на рисунке А5.

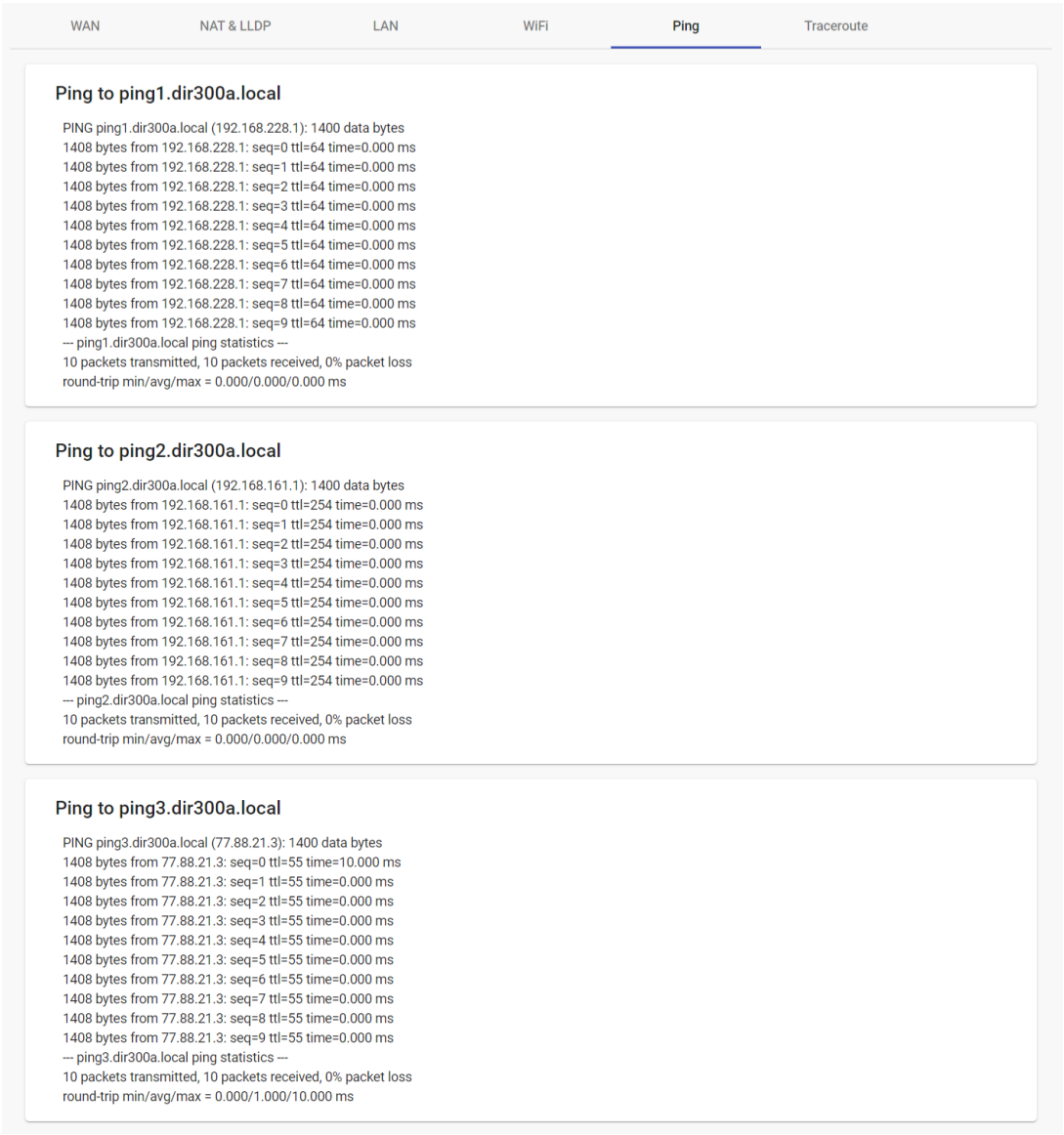

Рисунок А5 – Вкладка Ping

На рисунке А6 показан результат работы программы Traceroute.

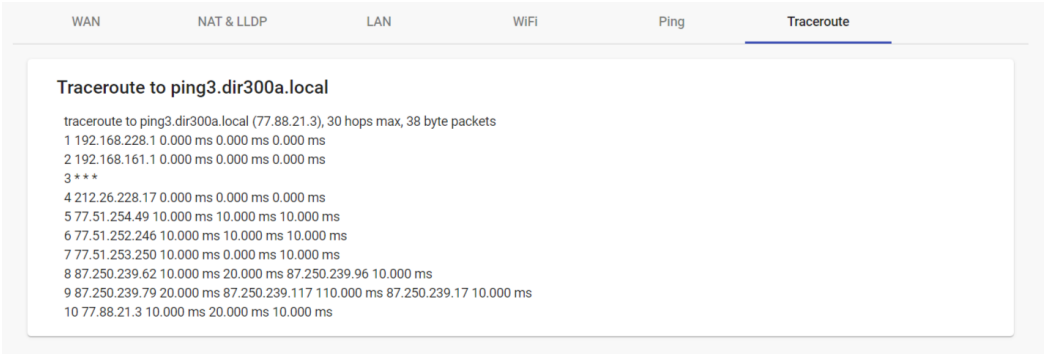

Рисунок А6 – Вкладка Traceroute