# Описание

*„Simple Network Management Protocol (SNMP) е протокол за управление на мрежи, част от стека от протоколи TCP/IP. Състои се от набор от стандарти за управление на мрежата, протокол от приложния слой, схема на база данни (****MIB****) и набор от обекти (****variable****) за пренос на данни.“*

Приложението дава възможност за добавяне на SNMP конфигурации (host, port, OID and community) и събирането на информация за конкретния **variable.** На определен период (**20 секунди**) се пращат заявки за взимането на информация за конкретния endpoint и информацията се пази за постоянно на сървъра (в момента **csv** файл). Събраната информация се визуализира във вид на **line chart**.

! Системата позволява визуализацията на информацията и във вид на **pie chart** но само ако са добавени групирания (ще бъде разяснено подробно в следващите секции).

Като допълнителна функционалност потребителите имат възможност да пращат **One Time Only Request** за да видят каква информация се съдържа в „**subtree-то**“ на конкретният **node**.

# Server (NodeJS)

Additional NPM packages:

* ***express (***[***https://expressjs.com/***](https://expressjs.com/)***)***
* ***fast-csv (***[***https://github.com/C2FO/fast-csv***](https://github.com/C2FO/fast-csv)***)***
* ***snmp-native (***[***https://www.npmjs.com/package/snmp-native***](https://www.npmjs.com/package/snmp-native)***)***

API

* **GET** '/snmpEndpoints' – Връща всички запазени SNMP endpoints.
* **POST** '/snmpEndpoints' - Добавя нов SNMP endpoint.

**Body Model - SNMPEndpoint**

* **POST** '/snmpEndpoint/data' - Взима данните събрани за определен SNMP endpoint

**Body Model - SNMPEndpoint**

* **POST** '/snmpEndpoints/test' - One time only взимане на данните които връща конкретна конфигурация.

**Body Model - SNMPEndpoint**

class **SNMPEndpoint** {

id?: number;

friendlyName: string;

description: string;

oid: string;

host: string;

port: number;

community: string;

status: **Status**;

supportGrouping: boolean;

groupingMatch: **GroupMatch**[];

groupingBetween: **GroupBetween**[];

}

class **Status** {

static Active = {

id: 1,

name: 'Active'

};

static Deactivated = {

id: 2,

name: 'Deactivated'

};

static Deleted = {

id: 3,

name: 'Deleted'

};

}

class **GroupMatch** {

original: string;

result: string;

}

class **GroupBetween** {

from: number;

to: number;

result: string;

}

На следващата снимка е показан **snmpClient.js** в който се съдържа логиката за пращане на **SNMP query**.

# 

# Client (Angular 5)

Additional NPM packages:

* ***Highcharts (***[***https://www.npmjs.com/package/highcharts***](https://www.npmjs.com/package/highcharts)***)***

Функционалности:

1 Добавяне на нов SNMP Endpoint конфигурация: **friendlyName**, **description**, **oid**, **host**, **port**, **community**

* 1. Ако **supportGrouping** е сетнато на **false** -> няма групиране и резултатите ще бъдат показвани само във вид на **line chart** спрямо датата.
  2. Ако **supportGrouping** е сетнато на **true** -> има две възможни групирания:

1.2.1 **Matching** където трябва да сетнете **original value** и **group**. Подходящо е когато знаете точно какви резултати очаквате.

Например: **original: 200** – **group: “Good”**

1.2.2 **Between** където трябва да сетнете диапазон (числов) **from/to** и **group** на очакваните резултати. Подходящ е когато имате някаквиranges от резултати които искате да групиране.

Например: **from: 200 to 500 – group “Good”**

Ако имаме сетнати групи, до **line chart-а** се появява **pie chart** който показва разположение по **%** на отделните групи.

**П.С. По подразбиране всички responses които НЕ влизат в нито една група влизат в default-на група N/A.**

Ако кликнете върху отделно парче от **pie chart-а** под графиките се показва таблица с response-те които са влезнали в тази група. Полезно е когато искате детайлна информация за всяка група по отделно.

1. **One time only SNMP Query**. Имате възможност да пратите еднократно query към съответният endpoint.

# Testing

Възможни сценарии за тестване на приложението:

1. **Mock** на **snmpClient.js** което ще ни даде възможност да решаваме какви резултати получаваме от самите **SNMP requests**.

2. **Mock** на **snmpRepository.js** което ще ни даде възможност когато взимаме данните, да не връщаме реалните данни а да връщаме наши данни. Или да тестваме дали данните които получаваме са правилни.

3. Закоментиране на **listener.start();** в **server.js** което ще значи, че **НЯМА** да пращаме **SNMP requests**. Следователно можем да **mock-нем** информацията която имаме, а именно - **db.csv** файла.

4. **Mock** на данните в **SNMPService** във front end-a. Така можем да тестваме front end-a без въобще да се вълнуваме дали имаме server.