universidade de aveiro deti departamento de electrónica, telecomunicações e informática

**41488 – Projeto Industrial**

# Developer guide para Aplicação Externa

| | |
|---|---|
| **Nome do projeto:** | Repetidor LoRa |
| **Empresa:** | Wavecom - Soluções Rádio, SA |
| **Membros da equipa:** | Contacto principal: Carolina Pinho, carolinapinho@ua.pt, 912714948<br>Outros membros: Francisco Oliveira, franciscojno@ua.pt, 913349971<br>Miguel Amorim, miguelamorim@ua.pt, 966415488<br>Ana Vicente, ana.vic@ua.pt, 917716729<br>Henrique Chaves, henriquechaves09@ua.pt, 967127978<br>Gonçalo Ferreira, gdferreira99@ua.pt, 968010870 |
| **Data:** | 28 de junho de 2021 |
| **Supervisor:** | Rui Manuel Escadas |

## Conceitos

**Chirspstack** → serviço online de servidor LoRa disponibilizado pela empresa com API integrada e de acesso recorrendo a JWT token's.

Em primeiro lugar começou-se por analisar as referências disponibilizadas em relação à API [1], de seguida consoante o objetivo do projeto: apresentar uma **whitelist** → tabela de dispositivos conectados ao servidor, encontrou-se pedidos da API que retornam os end-devices e gateways conectadas ao servidor.

Um dos problemas da primeira implementação da API pelos query's era o tempo limite de 48 horas do token, então só no dia 8 de março de 2021, descobriu-se como obter um token que não é limitado a 48 horas, da seguinte forma:

- No **Application Server** na aba **API keys**;
- Criar uma nova key que gera um **token** uma única vez e com "*tempo ilimitado de uso*".

## API & Node-Red

Até então o desenvolvimento dos pedidos para API foram efetuados ao converter os query's de linguagem *curl* para ficheiros php que tinham de ser alocados num webhost free. Com o estudo das possíveis soluções para o projeto, chegou-se à conclusão que a aplicação externa, necessita de conseguir passar as informações de *join request's* e *join accept's*, pelo que através dos ficheiros alocados fora do **Node-RED** não seria possível alterar os parâmetros dos query's da API.

Então teve de se encontrar uma forma de tratar a informação e os query's na ferramenta da aplicação externa [2] → **Node-RED**. Esta solução passa por:

- Identificar os query's do ChirpStack Application Server REST API necessários para essas tarefas;
- Obter o pedido curl do query;
- Com essas informações criar um http request para o url do curl;
- Os headers do curl são enviados por uma função no campo msg.headers.

### Exemplo: Get Users List

Este é o exemplo mais simples encontrado que serve apenas para demonstrar o processo escolhido, pelo que para as funções fundamentais apenas se alterou alguns campos explicados a seguir.

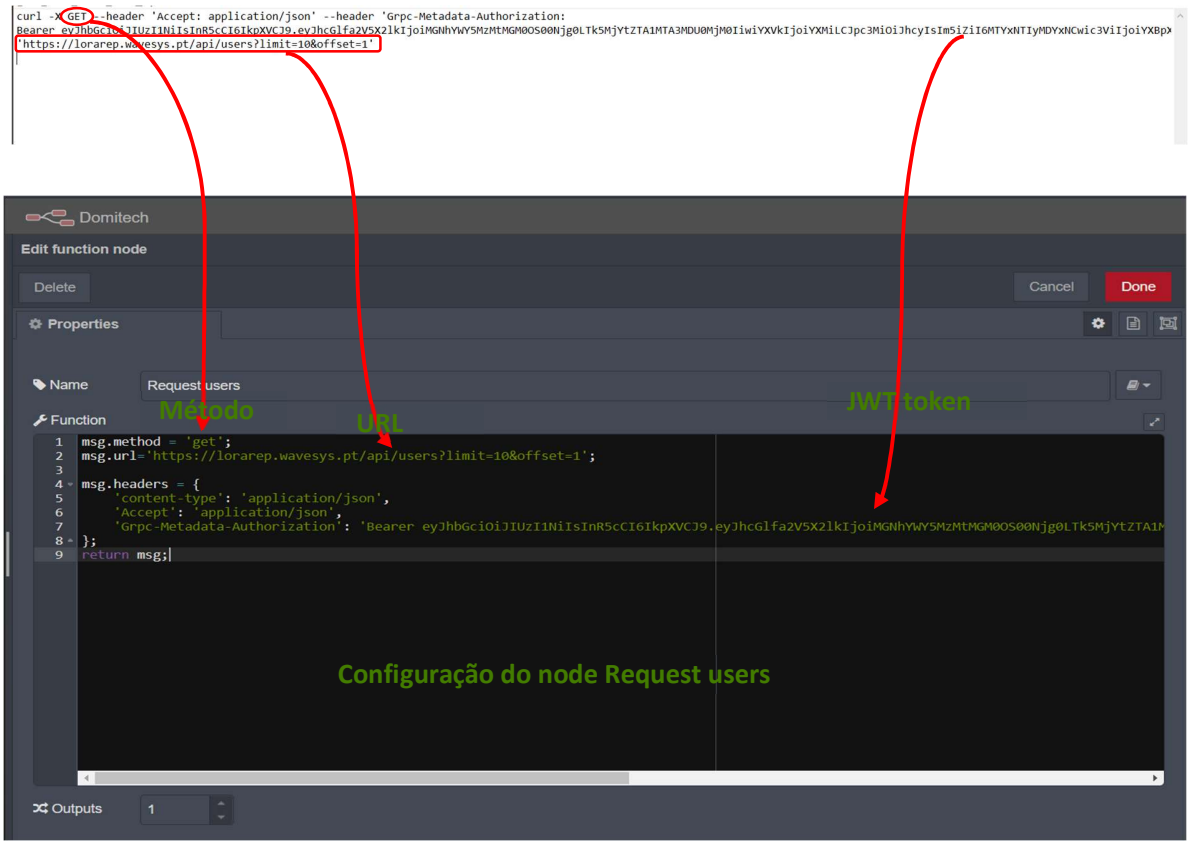


*Figura 1 - Configuração da função request API (Request users da Figura 2)*

A Figura 1 representa a configuração do node mais importante para implementação dos query's da API no Node-Red. Conclui-se que o campo de **msg.headers** não se altera para nenhum dos pedidos, então sempre que se quer configurar um novo query é só preciso ter atenção ao url novo, ao método especificado e se existe informações para o campo payload (no exemplo da Figura 1 não era necessário).

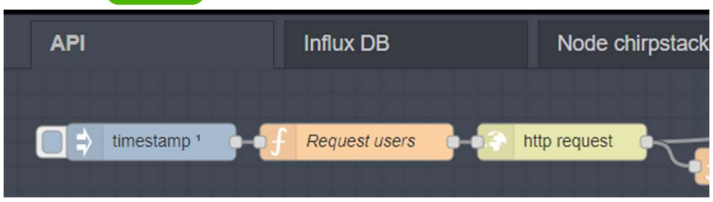


*Figura 2 - Nodes necessários para converter pedido curl para Node-Red*

O nó do meio da Figura 2 corresponde à configuração apresentada na Figura 1 e representa um nó function do **Node-RED** que permite ao nó seguinte (http request) fazer o request API ao servidor. O nó http request necessita de ser configurado com alguns detalhes como na Figura 3, pelo que em cada nó http request é necessário dar enable no TLS configuration.

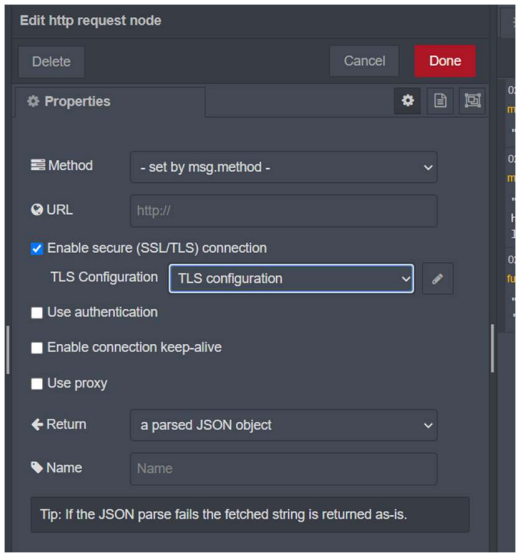


*Figura 3 - Configuração do nó http request*

Para além de se poder obter estas informações como foi descrito acima, também se descobriu que o **Node-RED** tem uma integração com a API do **Chirpstack** através de nodes prontos para esse efeito, tal como apresentado abaixo:

- **ChirpStack Users:** criar, editar, atualizar, eliminar e enumerar todos os utilizadores.
- **ChirpStack Devices:** criar, editar, atualizar, eliminar e enumerar todos os dispositivos.
- **ChirpStack Applications:** criar, editar, atualizar, eliminar e enumerar todos as aplicações.
- **ChirpStack Gateways:** criar, editar, atualizar, eliminar e enumerar todos as ***gateways***.
- **ChirpStack Organizations:** criar, editar, atualizar, eliminar e enumerar todos as organizações.
- **ChirpStack Device Profiles:** criar, editar, atualizar, eliminar e enumerar todos os perfis de dispositivos (***Device Profiles***).
- **ChirpStack network server requests:** criar um pedido de MAC (***MAC request***).

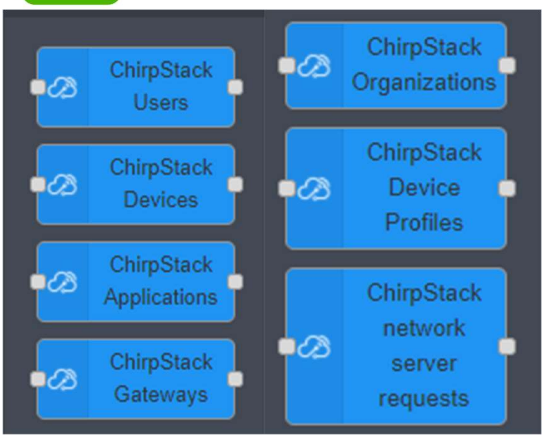


*Figura 4 - Nós para a integração direta entre o NR e Chirpstak*

No produto final apenas se recorreu à primeira hipótese (não usar os nós de **Chirpstack**), pois ao longo do desenvolvimento do trabalho deparámos-mos com algumas falhas nos nós apresentados e, também, o grupo ficou mais à vontade para alterar e programar os nós das funções que permitiam fazer os *request's* por HTTP.

## Interface Node-Red

De modo a apresentar as informações da aplicação externa e permitir ao utilizador interagir com os serviços que se quer disponibilizar começou-se por desenvolver uma interface gráfica no dashboard do **Node-RED** permitindo ao utilizador:

- Observar uma tabela com os utilizadores do serviço, outra com a informação dos dispositivos conectados ao servidor e gráficos de monitorização.
- Adicionar dispositivos ao servidor através de nome e/ou devEUI, e, consequentemente, ativá-los através da mesma interface sem ter de recorrer ao servidor **Chirpstack**.

Como explicado no documento principal deste relatório encontrou-se algumas limitações relativamente ao dashboard desta ferramenta, nomeadamente: a filtragem das tabelas não ser intuitiva e a escala temporal dos gráficos não permitir um ajuste em tempo real.

Como solução a ideia é utilizar outra ferramenta como o **Grafana** para poder apresentar essas informações de forma mais interativa e intuitiva. Antes disso, teve de se pensar como colocar as informações tratadas no **NR** disponíveis para outros serviços e, claramente, que a solução passa por utilizar alguma base de dados.

## Integrar InfluxDB com Node-red

Após algumas pesquisas, a base de dados escolhida foi a **InfluxDB** [3] a qual ninguém do grupo conhecia, mas trazia grandes vantagens, pois o **Chirpstack** tem uma integração direta com a mesma e o **NR** tem bibliotecas que permitem a leitura e escrita nas tabelas de uma base de dados **InfluxDB**.

O primeiro passo foi instalar a base dados no servidor da empresa de modo a alocar a informação de desejada e estar acessível em qualquer ponto, depois passou por instalar a biblioteca na ferramenta **Node-RED** e testar os comandos mais simples. Um dos exemplos conseguidos foi obter a informação de alguns pacotes que chegavam à gateway e apresentar numa tabela:

As últimas configurações já com o **Node-RED** podem ser encontradas na Figura 5 e o resultado final na Figura 6.

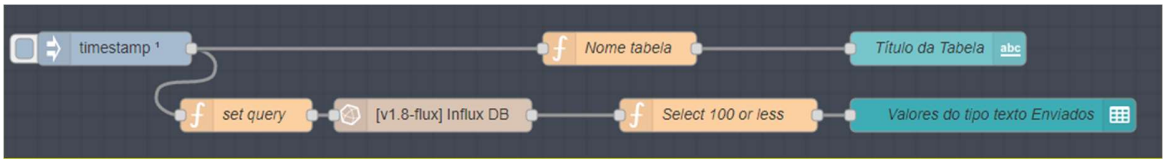


*Figura 5 - Conjunto de nós para apresentar os últimos 100 valores da InfluxDB numa tabela*

Apesar do grupo depois ter concluído que não havia necessidade de representar o conteúdo dos pacotes, mas sim monitorizar os mesmos, foi desenvolvida uma tabela que apresenta as mensagens dos dispositivos, neste caso "testes" do dispositivo Lo-pi4 com identificador 20919a4f045ff0b5, como se pode observar na Figura 6.

**Valores recebidos de tipo texto**

| ID | Device (dev_Eui) | Message type string |
|---|---|---|
| 1 | 20919a4f045ff0b5 | testes |
| 2 | 20919a4f045ff0b5 | testes |
| 3 | 20919a4f045ff0b5 | testes |
| 4 | 20919a4f045ff0b5 | testes |
| 5 | 20919a4f045ff0b5 | testes |
| 6 | 20919a4f045ff0b5 | testes |
| 7 | 20919a4f045ff0b5 | testes |
| 8 | 20919a4f045ff0b5 | testes |
| 9 | 20919a4f045ff0b5 | testes |
| 10 | 20919a4f045ff0b5 | testes |
| 11 | 20919a4f045ff0b5 | testes |

*Figura 6 - Tabela da interface*

## Integrar o sistema com Grafana

Neste ponto tem-se um servidor onde é possível tratar a informação que chega do servidor LoRA e guardar a informação numa base de dados, agora é hora de instalar e ficar à vontade a trabalhar na ferramenta nova que vai apresentar as informações da base dados.

Tal como para a base de dados, começou-se por instalar a ferramenta **Grafana** [4] no servidor disponibilizado pela empresa, de seguida testar comandos simples para aceder a uma tabela da base de dados e apresentar nesta nova ferramenta.

Com todas estas etapas concluídas, é preciso perceber que informação do **Node-RED** é preciso guardar na base de dados e como apresentar no **Grafana**, assim sendo, a disposição da base de dados é apresentada no documento principal e aqui será apresentada como se colocou essa informação na base de dados e de que forma se implementou no **Grafana** a interface final.

1. Guardar informação de todos os dispositivos e gateway's na **InfluxDB** através do **NR**

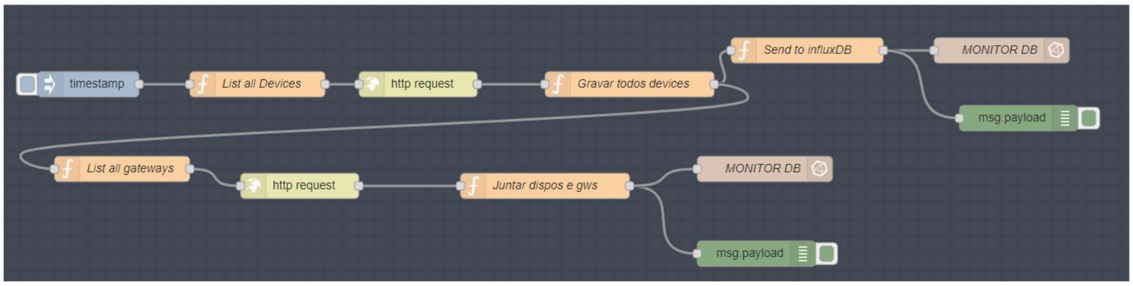


*Figura 7 - Nós que permitem obter a informação dos dispositivos e gateway's do servidor e enviar para a base de dados*

Com a informação toda numa tabela da base de dados **InfluxDB** é possível através da implementação de um query no **Grafana** aceder a essa informação e apresentar numa tabela da seguinte forma:

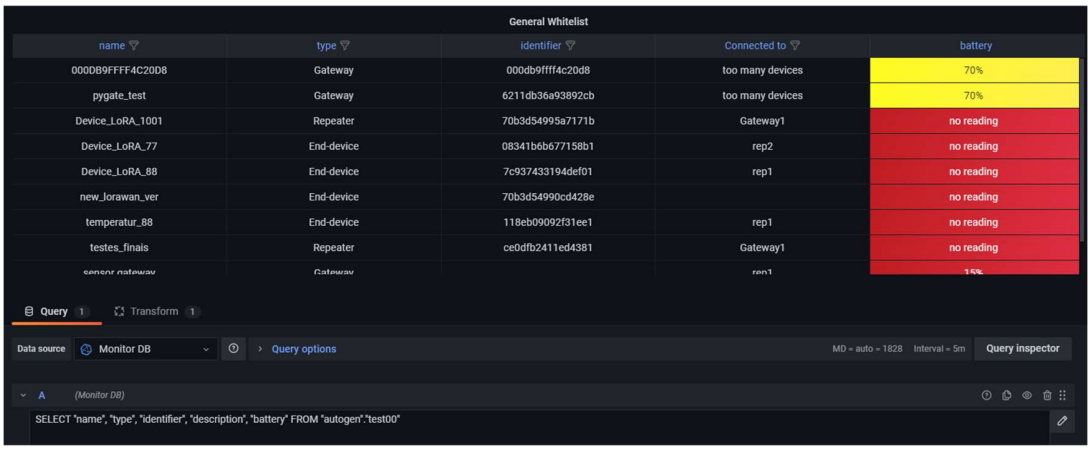


*Figura 8 - Implementação da tabela geral no Grafana*

2. Simular a informação de monitorização e enviar para a base dados

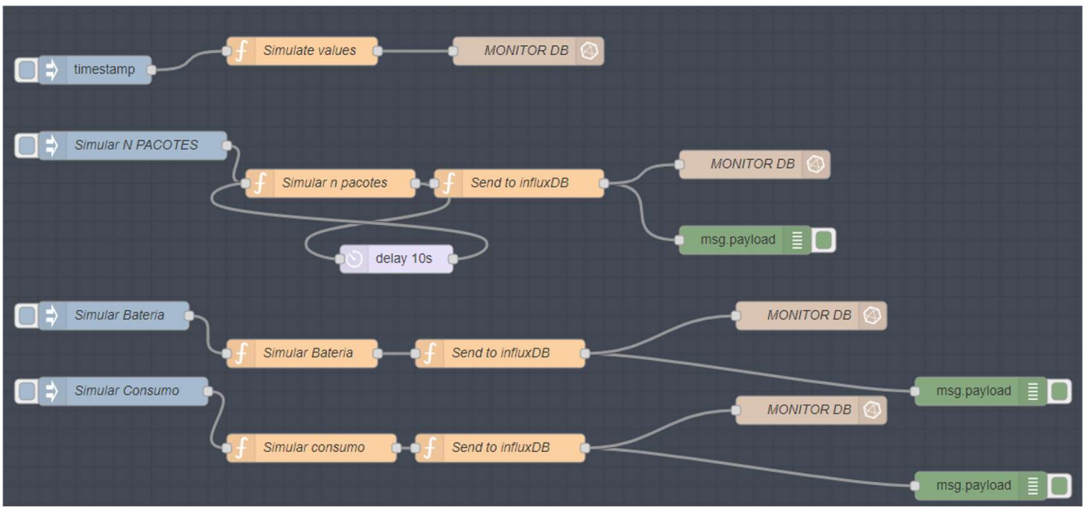


*Figura 9 - Nós que permitem guardar a informação simulada acerca da monitorização na base de dados*

Como não foi possível obter esta informação diretamente do projeto por não ter sido concluído simulou-se os valores, de modo que se alguém continuar com o desenvolvimento consiga rapidamente ajustar os parâmetros sem muito trabalho.

Desta forma, com a informação do número de pacotes enviados/recebidos, da bateria e consumo dos dispositivos alocada toda na base de dados é possível desenvolver dois gráficos temporais que apresentam essa informação e que permitem ao utilizador ajustar a escala temporal e qual o repetidor deseja observar.

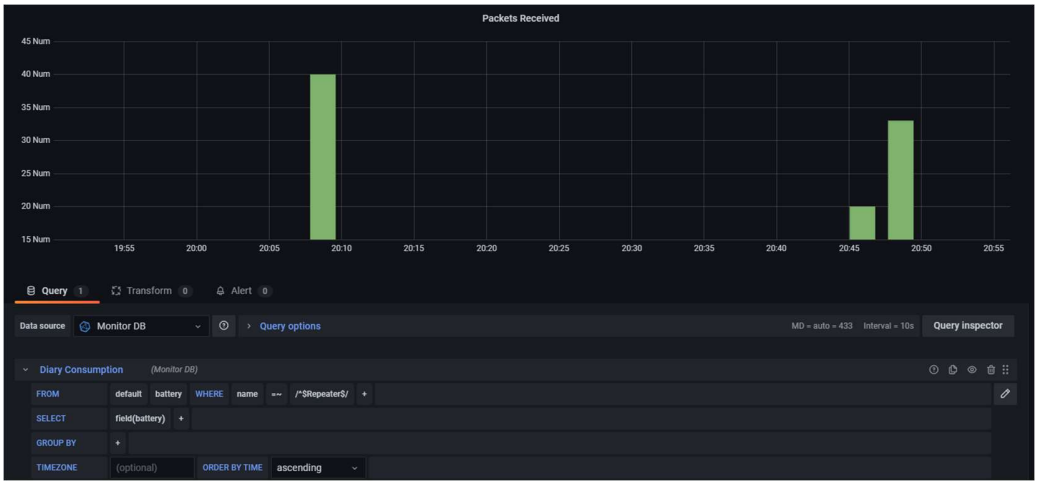


*Figura 10 - Demonstração da implementação do query que obtém informação de um determinado repetidor*

3. Monitorização individual de cada repetidor

Para concluir este objetivo, no **Node-RED** ao receber a informação de todos os dispositivos separou-se a mesma consoante a descrição de cada um, de modo que ao aceder à base de dados no **Grafana** fosse possível filtrar a mesma através de um capo. Na Figura 11 é possível observar o conjunto de nós que recebem a informação dos dispositivos conectados ao servidor LoRa: end-devices e repetidores, de seguida separa-os consoante o tipo e a cada end-device junta uma descrição que representa se está conectado diretamente à gateway ou a qual repetidor se liga para chegar ao servidor. Por fim, essa informação é enviada para uma tabela da **InfluxDB**.

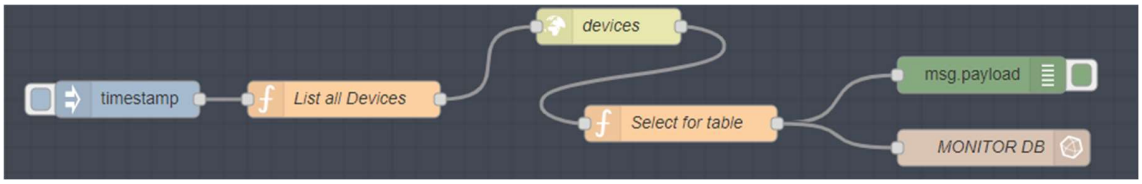


*Figura 11 - Nós que permitem enviar a informação individual dos repetidores para a base de dados*

Da mesma forma, que nos outros exemplos com a informação na base de dados acede-se e apresenta-se da maneira apresentada na Figura 12 as informações em tabelas e gráficos.

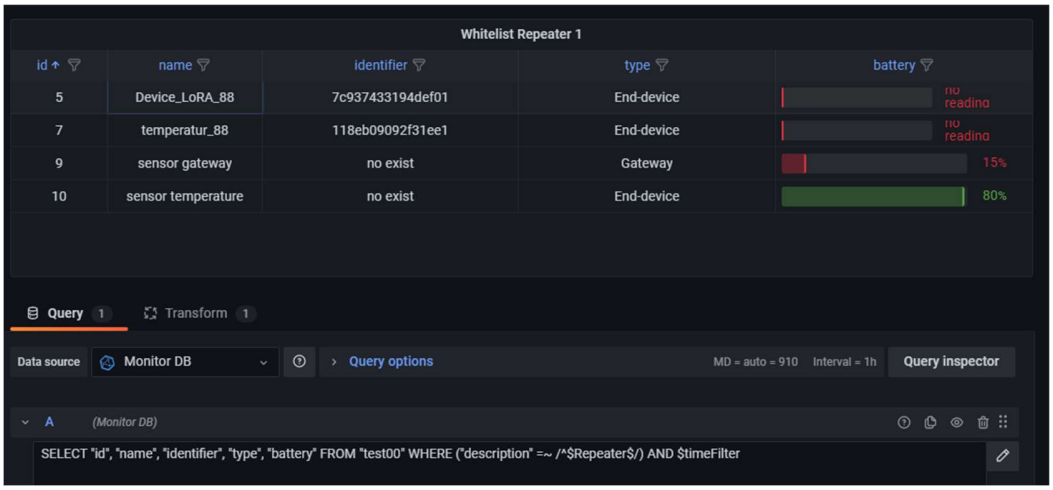


*Figura 12 - Implementação no Grafana da whitelist individual*

Criou-se variáveis no dashboard para o utilizador ter a possibilidade de escolher na caixa 1 da Figura 13, as informações que pretende alterar. Essa variável corresponde à informação que deve estar na coluna da descrição da tabela para cada repetidor.

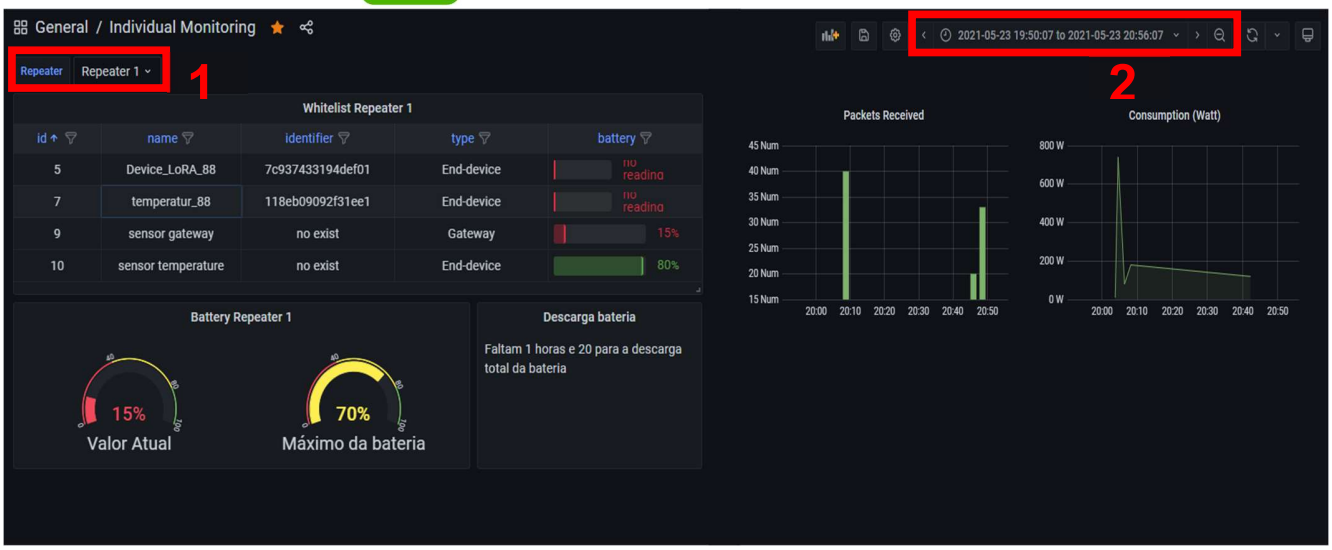


*Figura 13 - Versão final da interface implementada em Grafana*

Tendo em conta isto, o diagrama da arquitetura da base de dados é o seguinte (Figura 14):

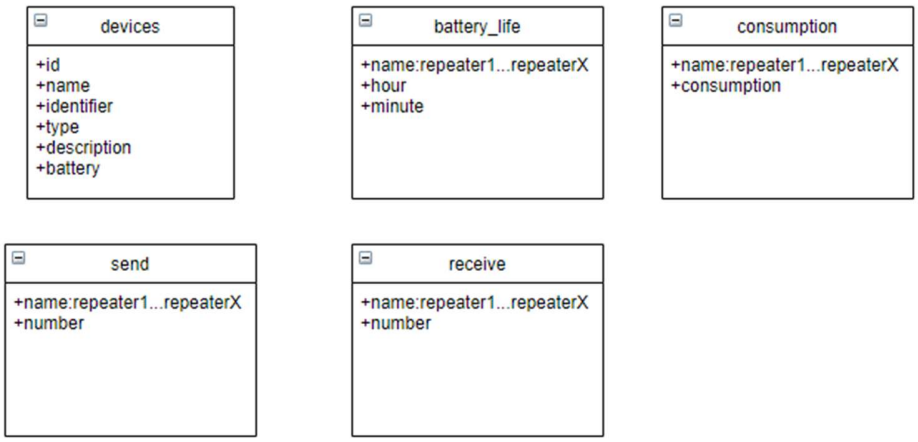


*Figura 14 - Arquitetura da base de dados na InfluxDB*

Programou-se um conjunto de nós no NR que permitem obter as informações dos dispositivos conectados ao servidor numa determinada aplicação e os gateways conectadas ao servidor. Para a descrição dos mesmos definiu-se uma estrutura onde *x* é um número inteiro que representa um determinado repetidor. Então, na descrição dos dispositivos conectados ao servidor obtém-se:

- Quando a descrição do dispositivo corresponde a rep-x, é um dispositivo que comunica com o servidor através do repetidor x, logo na tabela:
  - Type=end-device
  - Description=repx
- Quando a descrição do dispositivo corresponde a repeater-x representa o repetidor x do servidor, logo na tabela:

  - Type=repeater
  - Description=repeaterx
- Se a descrição for diferente destas considera-se que é um dispositivo que se conecta ao servidor diretamente através do gateway.

Assim sendo, ao fazer a leitura das tabelas da base de dados no Grafana é possível diferenciar o tipo de dispositivo e criar a whitelist para cada um dos repetidores, associando a descrição repeater-x ao número do repetidor em análise e dispostivos com a descrição rep-x são adicionados à whitelist correspondente.

Como trabalho futuro, o objetivo é desenvolver na aba de monitorização geral uma forma de apresentar tanto informação relativa à monitorização de consumo e de pacotes média dos repetidores como também acerca do LoRa traffic para várias situações.

Com a leitura deste documento fica-se a perceber como foi implementada a aplicação externa do projeto, as suas funcionalidades e a sua versão final.

**Referências:**

[1] → Referências disponíveis no website da ferramenta **Chirpstack** acerca do funcionamento dos serviços API da mesma. Disponível em: https://www.chirpstack.io/application-server/api/ [Visitado em 24/12/2020].

[2] → Blog online que nos permitiu perceber como fazer a conversão entre os pedidos curl do serviço API para os pedidos http do **NR**. Disponível em: https://stevesnoderedguide.com/node-red-http-request-node-beginners [Visitado em 04/01/2021].

[3] → Referência das informações relativas à base de dados utlizado time series **InfluxDB**. Disponível em: https://docs.influxdata.com/influxdb/v2.0/query-data/get-started/ [Visitado em 04/02/2021].

[4] → Referência que permitiu estudar uma forma de interligar a **InfluxDB** com o **Grafana**. Disponível em: InfluxDB data source | Grafana Labs [Visitado em 15/02/2021].