Licenciatura em Sistemas e Tecnologias da Informação

Dashboard: Monitoramento de API's com Gráfana

Projeto Final de Licenciatura GSC

Elaborado por Igor Rosa Costa

Aluno nº 20172125

Orientador: Professor Doutor Mário Macedo

Barcarena

dezembro de 2020

Atlântica Instituto Universitário

Licenciatura em Gestão de Sistemas e Computação

O autor é o único responsável pelas ideias expressas neste relatório

Agradecimentos

O alcançar desta etapa não teria sido possível sem a colaboração, auxílio, carinho e dedicação por parte de várias pessoas ao longo de todo o percurso da minha formação. Por esta mesma razão, não quero deixar passar esta oportunidade para agradecer a todos aqueles que, direta ou indiretamente, contribuíram para o meu sucesso e a minha chegada até aqui.

Agradeço primeiramente a todos os professores que tanto me ensinaram no decorrer da minha Licenciatura, especialmente ao Mário Macedo, que também contribuiu para a realização deste trabalho.

A todos os meus colegas de curso, pela ajuda e companhia nas inúmeras horas de estudos que tivemos juntos.

À minha mãe, Ana Paula Rosa da Paz, pela educação e sacrifício ao desempenhar o papel difícil de mãe. Obrigado por tudo.

A minha irmã, Iasmim Rosa Costa, pelo apoio ao longo desta trajetória.

Agradeço ao meu Gestor na NOS, que me permitiu utilizar um projeto executado internamente como material para o meu trabalho de final de curso.

Por último, mas não menos importante, agradeço aos meus cães Mickey e Minie, por ter sido uma fantástica companhia durante os meus estudos e realização deste trabalho de final de curso,

O meu muito obrigado.

Resumo

Dashboard: NOS API’s Dashboard

O presente projeto tem o intuito de disponibilizar um *dashboard*, que siga todos os requisitos para boas práticas de design e funcionalidade, permitindo também monitorar API’s em tempo real.

O *Dashboard* irá ser alimentado por um banco de dados MySQL fornecido pela NOS com os status atualizados de cada API.

A primeira parte deste trabalho contém as informações que motivaram o desenvolvimento deste projeto, tema, justificativa e os objetivos.

Enquanto que a segunda parte irá fornecer informações sobre o desenvolvimento do projeto, apresentando a Arquitetura e Protótipo do Sistema.

Por fim, o trabalho será finalizado com as considerações finais sobre o desenvolvimento do NOS API’s Dashboard.

Palavras-chave: Grafana, Banco de dados, Alarmística. API’s

Abstract

NOS API's Dashboard

This project aims to provide a dashboard, which follows all the requirements for good design and functionality practices, also allowing to monitor API's in real time.

This Dashboard will be powered by a MySQL database provided by the company NOS with the updated status of each API.

The first part of this work contains the information that motivated the development of this project. The theme, justification, and objectives.

The second part contains information about the development of the project, presenting the System Architecture and Prototype.

The final part contains the final considerations on the development of NOS API's Dashboard.

Keywords: Grafana, Database, Alerts. API's

Índice

Índice de figuras

Índice de tabelas

Lista de abreviaturas e siglas

API – Application Programming Interface

GQM - Goal-Question-Measurement

SGBD – Sistema de gerenciamento de banco de dados

BD – Banco de dados

SO – Sistema Operativo

SQL - Structured Query Language

ER – Engenharia de Requisitos

1. Introdução

1.1 Contexto e Motivação

A empresa NOS, uma das líderes nas telecomunicações no mercado português, em seu contínuo investimento em modernização e investimento em tecnologia de ponta, tem tido um crescimento enorme na utilização de *API's* para gestão dos serviços e também para facilitar o suporte e formação de novos funcionários. Dessa forma, é possível gerir plataformas com menos conhecimento através do uso das *API's*. Como consequência a esse crescimento, surge então a necessidade de haver um monitoramento sobre essas *API's* para certificar que os serviços estão sempre operacionais, e caso deixe de funcionar, as equipes consigam recuperar o serviço mesmo antes que o cliente final tenha a perceção da sua falha.

O Gestor, responsável pelo departamento das ferramentas de monitoria, propôs então um projeto no qual fosse criado um *dashboard* com uma interface gráfica limpa, onde fosse possível monitorar várias *API's* de forma estruturada, além de gerar alarmística sempre que venha a ficar indisponível, tornando assim os processos mais eficientes.

1.2 Definição do Problema

Uma vez que as *API's* são muito importantes para a continuidade dos serviços da NOS, surgiu a necessidade de ter uma monitoria extensiva sobre as mesmas para que caso venha a ficar indisponível, seja logo tomado as devidas intervenções para recuperar. Dentre as principais carências apontadas pelas equipes envolvidas no uso das API's, são destacados:

1) Necessidade de um *dashboard* que fosse visível para diversas equipes;
2) Ferramenta capaz de monitorar em real-time;
3) Equipe tem dificuldade de centralizar a monitoração das API's; e

4) Necessidade de tomar medidas imediatas, assim que a API fique indisponível.

1.3 Objetivos de Investigação

Considerando o contexto supracitado, definiram-se os seguintes objetivos de investigação (*RG – Research Goals*) de modo a resolver os problemas identificados:

- RG 1 - Efetuar a revisão da literatura de modo a compreender os principais aspetos técnicos para o desenvolvimento de um *Dashboard* que respeite todas as normas de boa prática.
- RG 2 – Elaborar a especificação de requisitos da aplicação NOS API's Dashboard;
- RG 3 – Definir a Arquitetura do Sistema;
- RG 4 – Definir os principais Casos de Uso;
- RG 5 – Definir o Modelo de Dados; e,
- RG 6 – Desenvolver e testar o protótipo aplicacional NOS API's Dashboard.

1.4 Método de Investigação

O método de investigação abordado no desenvolvimento deste projeto foi *Design Science Research* (DSR), com a finalidade de definir uma solução ao problema definido no capítulo anterior.

De acordo com, Hevner e Chatterjee, o conjunto de possíveis soluções de qualquer problema é específico como todos os meios possíveis que satisfazem todas as condições finais consistentes com as leis identificadas. Quando estes podem ser formulados adequadamente e colocados matematicamente, a pesquisa de operações padrão técnicas podem ser usadas para determinar uma solução ideal para o fim especificado (Hevner & Chatterjee, 2010).

De acordo com, Peffers, este modelo de processo consiste em seis atividades em uma sequência nominal (Peffers et al., 2006).

1. **Identificação do problema e motivação** – Definir a motivação do problema e justificar o valor da solução. Como a definição do problema será usada para desenvolver uma solução eficaz. Justificar o valor de uma solução motiva o pesquisador e o público da pesquisa a buscar a solução e aceitar os resultados e ajuda a entender o raciocínio associado ao entendimento do problema. O recurso necessário para esta atividade inclui o conhecimento do estado do problema e a importância de sua solução.
2. **Objetivos de uma solução –** Ao definir o problema, é necessário apontar os objetivos para solucionar o problema. Essa solução pode ser quantitativa, se a solução for melhor que a vigente, ou qualitativos, caso seja uma solução inédita para um problema. Desta forma, a solução deve ser apresentada de forma lógica e racional de acordo com o problema.
3. **Design e desenvolvimento –** Para criar uma solução, é necessário definir os aspetos necessários para a construção do artefacto, definindo amplamente modelos, métodos ou instâncias. Essa atividade inclui determinar a funcionalidade desejada do artefacto e sua arquitetura, após isso deverá então desenvolver o artefacto real. Para isso é necessário utilizar o conhecimento da teoria para ser possível criar uma solução.
4. **Demonstração –** Necessário demonstrar a eficácia do artefacto para resolver o problema. Isso pode envolver o uso em experimentações, simulações, estudo de caso, prova ou outra atividade apropriada.
5. **Avaliação –** Necessário analisar e medir o quão bem o artefacto suporta uma solução para o problema. Essa atividade envolve a comparação dos objetivos de uma solução com resultados reais observados do uso do artefacto na demonstração. Esta avaliação pode conter orçamentos ou itens produzidos com satisfação pesquisadas, comentários de clientes ou simulações. No final desta

atividade, os pesquisadores podem decidir se devem voltar a etapa 3 para melhorar a eficácia do artefacto ou continuar.

6. **Comunicação –** É importante comunicar a importância do problema, o artefacto, sua utilidade e novidade, o rigor do design e sua eficácia a pesquisadores e outros públicos relevantes, como profissionais, quando apropriado.
7. **Demonstração –** Demonstrar a eficiência do artefacto, no intuito de resolver o problema identificado. Podendo abranger em forma de experimentação, simulação, prova, estudo de caso ou outra atividade apropriada. Nessa etapa, é necessário aplicar todo o conhecimento eficaz do artefacto para a resolução do problema.

1.5 Resultados Alcançados

Os resultados esperados foram alcançados, permitindo assim elaborar uma *dashboard* capas de monitorar API's em tempo-real e solucionar os problemas de investigação em causa.

1.6 Estrutura do Documento

O presente documento é composto por seis capítulos que se iniciam com este em que se enquadra a investigação, apresentando o contexto e motivação, definição do problema, objetivos da investigação e os resultados alcançados;

O segundo capítulo, fornecerá a revisão da literatura apresentando os principais conceitos e tecnologias abordados neste projeto;

O terceiro capítulo, irá demonstrar a especificação de requisitos, casos de uso do projeto, requisitos funcionais e não funcionais e o software de suporte para o projeto;

O quarto capítulo, descreve o protótipo do sistema;

No quinto capítulo, serão exibidos o protótipo e os testes executados assim como a descrição de todo o cenário;

Por fim, no sexto capítulo, é apresentado as conclusões da investigação.

2.Revisão da Literatura

2.1 Introdução

Com o crescimento das tecnologias a necessidade de ferramentas para gerência dos dados e plataformas principalmente em uma Telecom onde se lida com vários tipos de tecnologias, torna se então imprescindível monitorar o desempenho das plataformas. Neste capítulo irá ser apresentado a revisão da literatura, tendo como objetivo os pontos importantes para esta investigação, abordando os tipos e requisitos para a criação de um *Dashboard* e também bancos de dados.

2.2 Dashboard

Cada vez mais surge a necessitam de partilhar dados importantes dos serviços e desempenho de uma forma fácil e sucinta. Para isso, são empregadas diversas ferramentas que possibilitam um fácil acesso aos dados, de forma a poderem ser analisados e melhor compreendidos, várias empresas utilizam relatórios, gráficos e *dashboards*. Atualmente é indispensável a utilização de ferramentas para poder analisar o desempenho, e até mesmo como auxílio à tomada de decisão.

2.2.1 Conceito

O termo *dashboard*, dentre suas muitas tipificações, pode trazer em mente a ideia do painel de instrumentos, como por exemplo de um carro, onde é possível encontrar vários

componentes que contêm informações muito importantes sobre o funcionamento e estado do veículo de forma fácil de compreensão (Malik, 2005).

Figura 1- (Simulado DETRAN 2020 - O mais seguro e atualizado, sem data)

No entanto, os *dashboards* apresentam uma infinidade de possibilidades além de um mero painel, pois é uma ferramenta que fornece um meio interativo e centralizado de monitorar, medir, analisar e extrair análises de dados relevantes, de diferentes conjuntos de dados em áreas-chave, enquanto exibe informações de forma interativa, intuitiva e visual. Portanto, o termo empregue na atualidade possui um novo significado, através da monitoria dos sistemas em tempo real através de gráficos, alertas e relatórios (Malik, 2005).

Este é um recurso personalizável cujo objetivo é atender às necessidades específicas de um departamento e de uma empresa. Dessa forma, é oferecido aos usuários uma visão abrangente de departamentos, metas, iniciativas, processos ou projetos internos de sua empresa. Eles são medidos por meio de indicadores-chave de desempenho (KPIs), que permitem a avaliação da saúde de um negócio, ajudando-o a promover o crescimento e a melhoria.

O *dashboard* se tornou uma ferramenta crucial para a tomada de decisão pois permite juntar vários dados de forma simples e sucinta, que conduz a tomada de decisões (Caldeira 2014). Sem a sua existência, as empresas teriam que analisar uma vasta série de dados desestruturados, que além de ser ineficiente e demorado, abre uma margem maior para erros humanos. Por este motivo, *dashboards* são vistos como um meio ao qual as informações podem ser resumidas e prontamente comunicadas aos tomadores de decisão (Pauwels et al., 2009)

2.2.2 Tipos de Dashboard

Quanto aos tipos de *dashboard*, é necessário ter em consideração o tipo de informação que se pretende, e a quem se destina, pois os *dashboards* podem ser elaborados com diversos gráficos e para diferentes funcionalidades. É bastante importante enfatizar que um bom *dashboard* deve conter informações personalizadas para o seu destinatário, pois nem sempre o que um utilizador necessita é importante para o outro. Como por exemplo os gráficos de desempenho de uma máquina Linux não irão ser importante para um funcionário de Markting, no entanto poderão ser cruciais para um departamento de IT.

De acordo com (F et al., 2012, p.) e (Eckerson, 2012) os *dashboards* são divididos em três tipos:

- **Operacional** – tem como principal objetivo fornecer informações detalhadas, simples e intuitiva. Sendo assim, desempenha a função de um painel operacional, que enfatiza o monitoramento mais do que analise e gerenciamento. Este tipo de *dashboard* é destinado principalmente para os funcionários com contato direto com os clientes, pois apresenta informações em tempo real do dia a dia das entidades.
- **Táticos** - Os painéis táticos são responsáveis por monitorar processos e projetos departamentais, e tende a ser utilizado mais por gerentes e analistas de negócios.

Esses *dashboards* são utilizados para comparar desempenho real de uma seção com o desempenho esperado ou resultados do último período, normalmente são atualizados diariamente ou semanalmente.

- **Estratégico** – geralmente atualizam as informações de forma diária ou semanal, tende a enfatizar mais a análise do que o monitoramento ou gerenciamento. Este tipo de *dashboard* é mais utilizado por administradores das entidades, permitindo analisar e traçar objetivos. Este tipo de *dashboard* permite analisar a evolução de uma instituição relativo aos seus objetivos.

Porém para (Malik, 2005) os *dashboards* devem ser classificados de acordo com o tipo de utilização.

- **Painéis de desempenho corporativo** – apresentam dados de várias divisões e segmentos de negócios e fornecem uma visão global da empresa, este tipo de *dashboard* são mais direcionados para os gerentes.
- **Painéis divididos** – são mais focados nos departamentos, devem apenas apresentar as informações pertinentes para cada departamento, por exemplo uma *dashboard* apenas para o Marketing, outra que contenha somente informações pertinentes para os recursos humanos entre outros departamentos.
- **Painéis de monitoramento de processos / atividades –** são criados com o intuito de monitorar processos e atividades do negócio da empresa, ou seja, sua função é ser analisada e evitar os problemas antes que aconteçam. Como por exemplo gestão do espaço em disco de um equipamento, antes que o mesmo fique sem espaço.
- **Painéis de aplicação –** são geralmente criados no intuito de receber aplicativos personalizados para fornecer métricas específicas definidas no aplicativo. Como por exemplo uma *dashboard* de gestão de viagens.
- **Painéis de clientes –** contém métricas relevantes para o cliente da organização.
- **Painéis de fornecedores –** tem o intuito de facilitar a interação do fornecedor e a organização.

Na seguinte tabela é possível analisar 3 tipos de *dashboards* referidas por (F et al., 2012) & (Eckerson, 2012), onde permite comparar as diferenças entre as mesmas.

Tabela 1 - Performance Dashboards: Measuring, Monitoring, and Managing Your Business adaptado de (Eckerson, 2012)

	Operacional	**Tático**	**Estratégico**
Foco	Monitorar operações	Otimizar processos	Executar estrategias
Ênfase	Monitorar	Analisar	Colaboração
Users	Supervisores	Managers	Executivos
Foco	Operações	Departamentos	Empresa
Informação	Detalhada	Detalhada/Sumarizada	Sumarizada
Atualizado	Varias vezes ao dia	Diário/Semanal	Mensal/Treimestral
Tipo de Painel	"Dashboard"	"BI Portal"	"Scorecard"

2.2.3 Boas práticas para o design de um *Dashboard*

Para a criação de um *dashboard,* é necessário levar em consideração vários fatores levando em consideração o tipo de dados que será tratado, a quem é direcionado. Como é mencionado por (Janes et al., 2013) idealmente, deverá ser criado um *dashboard* que seja útil, e que vá de encontro com as necessidades do usuário, em vez de criar uma ferramenta com vários gráficos desnecessários, e que acabam por criar poluição visual.

Como é mencionado por (Few, sem data) um painel é uma exibição visual das informações mais importantes necessárias para obter um ou mais objetivos, consolidado e organizado em uma única tela. Por isso, é imprescindível adequar às necessidades aos diferentes tipos de *dashboards*, dos quais foram demonstrados no ponto anterior. Ao adequar a informação às necessidades do usuário, sua interpretação se torna mais fidedigna e eficiente. Como mencionado acima, o usuário ao focar na informação que lhe é cara, não se perde na poluição visual que pode lhe induzir ao erro, além de dificultar a interpretação.

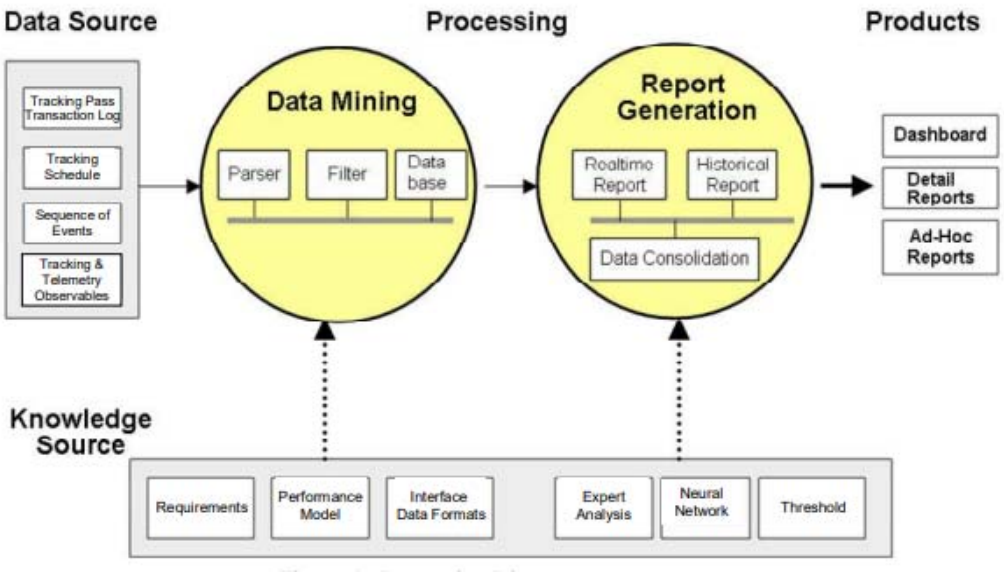


Figura 2 - Diagrama de processo extraído de (Pham & Liao, 2008).

Já no século 19 eram utilizadas as cores para indicar o nível de urgência apresentado em um *dashboard*, neste caso nos carros. (Janes et al., 2013)

- **Vermelho** – normalmente é um indicativo de que o problema é grave e que é necessário tomar medidas imediatas, estes tipos de problemas podem pôr em causa a segurança e o funcionamento.
- **Amarelo –** normalmente indica que deverá tomar medidas breves, como por exemplo quando liga o indicativo do combustível
- **Verde –** é meramente informativo, indicando que a funcionalidade foi ligada.

Figura 3 - Indicadores de um veículo extraído de (Janes et al., 2013).

Uma vez que os *dashboards* são muito ligados a apresentação de dados em forma gráfica, o utilizador espera sempre uma ferramenta com um visual apelativo. De acordo com (Malik, 2005) & (Ivanov et al., 2018) um *dashboard* com um layout esteticamente cuidado e apelativo, é muito mais eficiente do que um com cores e espaços limitados.

É referido por (Malik, 2005) os seguintes elementos chave para o design de um *dashboard*:

- Painel com gráficos e cores
- Selecionar os tipos de gráficos apropriados
- Utilizar animações apenas relevantes
- Posicionamento correto dos elementos do *dashboard*

De acordo com (Janes et al., 2013) para escolher os dados de forma correta para a criação de um *dashboard* é criado um modelo de medição. Para ser escolhidos os dados de forma correta, iniciará por medir os objetivos, que tipos de questões deverão ser respondidas e que tipo de dados serão coletados para poder responder a essas questões. Para atingir esse objetivo é adotado um método GQM + Abordagem estratégica, que é baseado no GQM (Goal-Question-Measurement).

O modelo GQM é definido em três níveis:

- **Nível Conceitual** – nível onde é identificado os objetivos a alcançar relativo aos processos, recursos ou produtos.
- **Nível Operacional** – nível onde são feitas questões que ajudam a compreender como atingir as metas.
- **Nível Quantitativo** – nível onde é identificado as métricas necessárias para responder as questões.

Os dados em um *dashboard* devem sempre respeitar a ordem de importância, principalmente se os dados forem em tempo real, os pontos mais críticos devem sempre se sobressair ao foco do usuário, através das cores ou posicionamento, caso seja em colunas, aparecer sempre no topo em posição de destaque.

2.2.4 Tipos de Gráficos

Um aspeto fundamental de um *dashboard* bem-sucedido passa pela escolha correta de seus gráficos. A análise rápida dos dados é um ponto fulcral de usabilidade dessa ferramenta, e a combinação de elementos projetados ao utilizador é o que irá garantir coesão e rápida compreensão.

Atualmente, está disponível uma ampla gama de opções de gráficos, muito mais complexos que os de estilo básico, como o de barra, linha e pizza, tão comumente usado quando os recursos digitais eram limitados. No momento atual é possível inserir estilos e tipos de gráficos alternativos, múltiplas escalas e visualizações de dimensão e opções alternativas de apresentação (Bauer, 2005). Entretanto, para melhor categorizá-los, Bauer tipifica os gráficos em quatro áreas principais:

- **Básico** - são os principais gráficos, como barras horizontais, colunas, pizza, área e linha, e seu principal objetivo é a comparação de série de dados singulares e múltiplos. Esta lista pode ser consideravelmente expandida quando são

adicionados valores alternativos, como as variações agrupadas, empilhadas, emparelhadas e progressivas.

- **Diferença** - são empregues onde o diferencial entre duas séries de dados é de maior relevância do que o valor absoluto. Isso pode ser expresso como uma comparação direta entre várias séries de dados (intervalo de alteração) ou como uma comparação com uma linha de base de referência (barra de desvio).
- **Distribuição** - gráficos únicos que traçam o perfil da distribuição dos dados por meio de gráficos de barras de frequência (histograma); descritores métricos como mediana, média e percentis (*box-and-whisker*); e indicadores de movimento diário da atividade do mercado de ações (HLC - *high-low-close*).
- **Especializado** - Coleção de gráficos que enfocam áreas de negócios específicas que incluem planejamento (gráficos de bolhas), análise competitiva (radar), perfis financeiros (gráfico em mosaico) e plotagem tridimensional (trilinear). Gráficos especializados podem ser facilmente expandidos para incluir exemplos mais focados em funções de negócios e fatores específicos da indústria.

Ao se pensar no design de um *dashboard* é importante escolher o estilo de gráfico apropriado para atender às necessidades específicas do aplicativo de negócios. Isso requer que vários critérios-chave, como foco da análise, tempo contínuo, número de séries de dados e métricas de dados, todos sejam incorporados ao processo de seleção de gráfico. Isso requer que vários critérios-chave, como foco da análise, continuidade do tempo, número de séries e métricas desses dados, para que assim, todos sejam incorporados ao processo de seleção de gráfico.

- **Foco de análise** - Como os dados serão analisados é um critério importante para identificar o tipo de gráfico apropriado. As séries de dados podem ser comparadas entre si (por exemplo, barra de agrupamento, barra emparelhada ou barra de mudança) ou a uma linha de base de referência (por exemplo, barra de desvio, mostrador ou barra circular). Os diagramas de dispersão enfocam a relação entre

as séries de dados (ou variáveis), enquanto os gráficos *box-and-whisker* e os histogramas traçam o perfil da distribuição dos dados.

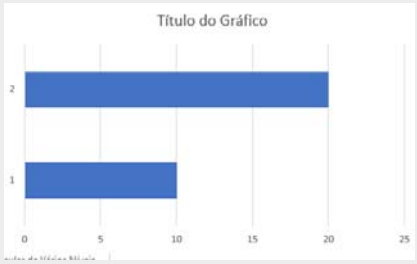
- **Continuidade do tempo -** O elemento tempo é um aspeto chave ao escolher o tipo de gráfico, e sua escolha aqui é bastante simples. A série de dados é capturada em um ponto específico do tempo, ou em vários pontos, em decorrência a um espaço de tempo mais amplo. O principal potencial dos gráficos de pizza, bolha, barra empilhada e mosaico é sua capacidade de visualizar todos os dados detalhados de uma única fatia do tempo. Lado a lado, também permitem que dois prazos alternativos sejam exibidos e comparados. Como alternativa, os histogramas de gráficos de dispersão e gráficos de séries temporais se destacam na análise de dados em vários períodos de tempo.
- **Número de séries de dados** - Uma compensação distinta normalmente existe entre o contínuo de tempo e o número de séries de dados que podem ser exibidas. Para uma fatia no tempo, várias séries de dados podem ser facilmente exibidas por meio de gráficos de barra, pizza, área, radar e bolha. Gráfico de intervalo e gráficos de barra emparelhados capturam de forma sucinta a diferença entre duas séries de dados. Múltiplos períodos de tempo, com pelo menos três ou mais séries de dados, requerem gráficos especializados, como bolha e dispersão
- **Métricas de dados** - Também é importante considerar qual formato de dados é mais apropriado para exibir as tendências de negócios. Embora a maioria dos dados originais seja linear por natureza, os dados derivados, como proporções, índices, percentagens, logaritmos e cumulativos, podem frequentemente ser mais perspicazes como KPIs. Por exemplo, em situações em que existem diferenças graves nas taxas de crescimento da linha de negócios ou no tamanho real dos negócios, as escalas logarítmicas atenuam o problema de escala e fornecem uma leitura mais consistente da situação dos negócios.

É também preciso considerar que podem ser criados mais impacto e poder de análise se combinando vários tipos de gráficos. Por exemplo, uma barra empilhada e um gráfico de linha podem ser combinados para ilustrar não apenas a receita de vendas do ano atual por

linha de negócios, mas também as vendas totais em comparação com uma média móvel mensal tendencial ou totais do ano anterior. Em vez de apenas dados pontuais, essa combinação de gráficos enquadra adequadamente os dados de vendas atuais, adicionando algum nível de contexto. Entretanto, é preciso ter cuidado, pois a seleção do gráfico errado (ou a combinação errada de gráficos) pode ser mais prejudicial do que a omissão do gráfico, se uma mensagem errada for transmitida.

Abaixo serão listados os principais exemplos de gráficos comumente usados em dashboards:

Tabela 2 - Tipos de gráficos

Gráficos de barras Eles são uma boa opção para resumir dados que são subdivididos em categorias que serão comparadas, assim realçando tendências, ou para fazer generalizações de dados rapidamente.	 Figura 4 - Gráfico de Barras
Gráficos de colunas Também conhecidos como gráficos de barras verticais, devido sua semelhança tanto em sua aparência, quanto empregabilidade. São úteis para comparar dados discretos ou mostrar tendências ao longo do tempo. É composto por marcadores de dados verticais de forma que facilite comparar valores individuais.	 Figura 5 - Gráfico coluna

Gráficos de linha Neste tipo de gráfico os dados estão organizados em linhas, e tende a apresentar dados contínuos ao longo do tempo, por isso, seu eixo X é regularmente dotado de uma unidade de tempo (horas, meses, anos, etc). Dessa forma, é comumente usado para monitorar tendências ao longo do tempo, como estoque ou preços de ações, ou até mesmo temperaturas nos processos de produção.	 Figura 6 - Gráfico de linha

Gráficos de Setores: circulares e em anel

Conhecido popularmente como gráfico de pizza, devido a sua composição, que exibe vários pontos de dados de forma circular. Cada seção do gráfico é dividida em segmentos pela relação com a circunferência. Os gráficos de pizza são eficazes para exibir a percentagem de um todo, relacionando ou categorizando por grupos segmentados. No entanto, sua capacidade como uma ferramenta de visualização é limitada, pois única série de dados

Deverá ser utilizado esse tipo de dados nos casos em que:

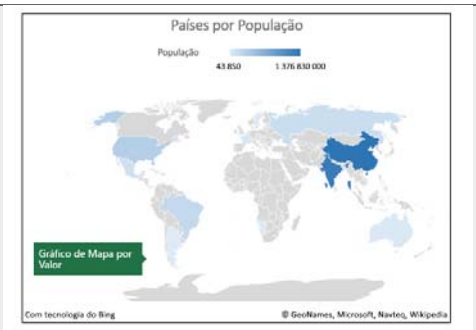
1. Tiver mais que um dado
2. Não exista dados com valores negativos
3. Não ter muitos dados com zero, o ideal é não ter nenhum
4. Não ter mais que sete categorias, pois irá gerar poluição visual

Figura 7 - Gráfico circular

Figura 8 - Gráfico de Circular

Figura 9 - Gráfico de Anel

Gráfico medidor (Gauge) Também conhecidos como gráficos de mostrador ou gráficos de velocímetro, usam agulhas para mostrar as informações como uma leitura em um mostrador. Nesse tipo de gráfico, o valor é apontado por uma agulha, e lido em relação ao intervalo de dados demonstrado em relação ao eixo do mesmo. Seu uso é principalmente em dashboards executivas comparar valores entre um pequeno número de variáveis usando várias agulhas no mesmo medidor ou usando vários medidores. Um gráfico de medidor consiste em um eixo de medidor (que contém o intervalo de dados, intervalos de cores e marcadores de intervalos), agulhas e um ponto de pivô central.	 Figura 10 - Gráfico Medidor
Gráficos de radar Os gráficos de radar integram vários eixos em uma única figura radial. Este gráfico compara os valores agregados repartidos em vários pontos. Nesse caso, os dados são coordenados ao longo de um eixo separado que começa no centro do gráfico.	 Figura 11 - Gráfico de radar

Gráfico de mapa Este gráfico é utilizado para comparar valores entre áreas geográficas.	 Figura 12 - Gráfico de mapa extraído (Tipos de gráficos disponíveis no Office, sem data)

2.3 Bancos de dados

Como é mencionado por (Suehring, 2002),(Garcia-Molina et al., sem data) & (Rob & Coronel, sem data) os bancos de dados são necessários em todos os tipos de negócios, pois são utilizados para manter registos internos, apresentar dados a clientes e dar suporte a processos comerciais. Esta ferramenta também está no centro de muitas investigações científicas.

Os bancos de dados derivam de um corpo de conhecimento e tecnologia que veio a ser desenvolvido ao longo de várias décadas, e está incorporado em um software denominado Sistema de Gerenciamento de Banco de Dados (SGBD) ferramenta utilizada para criar e gerir grandes quantidades de dados, de forma segura.

De acordo com (Rob & Coronel, sem data) os softwares de gestão de banco de dados, são ferramentas que estão entre o usuário final e o banco de dados. Estes SGBD deverão receber solicitações de aplicações que podem ser desenvolvidas em diversos tipos de

linguagem de programação como *frontend,* para que o utilizador consiga aceder aos dados sem a necessidade de comunicar diretamente com o SGBD.

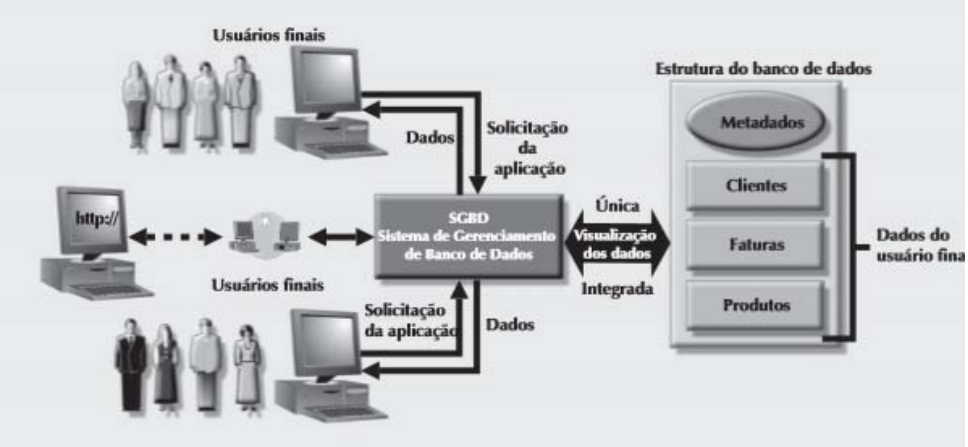


Figura 13 - O SGBD gerência a interação entre o usuário final e o banco de dados extraído (Rob & Coronel, sem data)

2.4 Engenharia de Requisitos

A Engenharia de requisitos (ER) é um processo de engenharia de software, que é utilizada para traçar as necessidades do novo sistema, permitindo assim desenvolver todas as necessidades para que o cliente fique satisfeito. Esta análise é feita durante todo o ciclo de vida do projeto. Caso o levantamento dos requisitos funcionais e não funcionais sejam feitos de forma incorreta, isso poderá levar com que o projeto não fique alinhado de acordo com as necessidades do cliente.

2.4.1 Requisitos Funcionais

Os requisitos funcionais são responsáveis por materializar as características esperadas que a interface consiga transpor. Os requisitos funcionais são muito importantes no desenvolvimento do aplicativo, pois sem ele o aplicativo fica sem funcionalidades. (Sommerville, 2016)

2.4.2 Requisitos Não Funcionais

Requisitos não funcionais descrevem a forma como o sistema deve funcionar, estes requisitos são definidos por características que impõe limites do sistema, por exemplo espaço, sistema operativo entre outros. Estes requisitos são virados para o funcionamento do software. (Sommerville, 2016)

3. Especificação de Requisitos NOS API's Dashboard

Neste tópico irá ser abordado os requisitos necessários para a elaboração do NOS API's Dashboard

3.1 Introdução

Este tópico tem como objetivo identificar as especificações de requisitos do sistema necessário para implementação da NOS API's Dashboard. Este projeto tem como propósito a criação de uma *dashboard* capaz de disponibilizar status de API's colapsadas em Tree.

3.2 Casos de Uso

A Figura 14 demonstra um Diagrama de Casos de Uso, que descreve de forma objetiva as funcionalidades do sistema e as interações entre os atores.

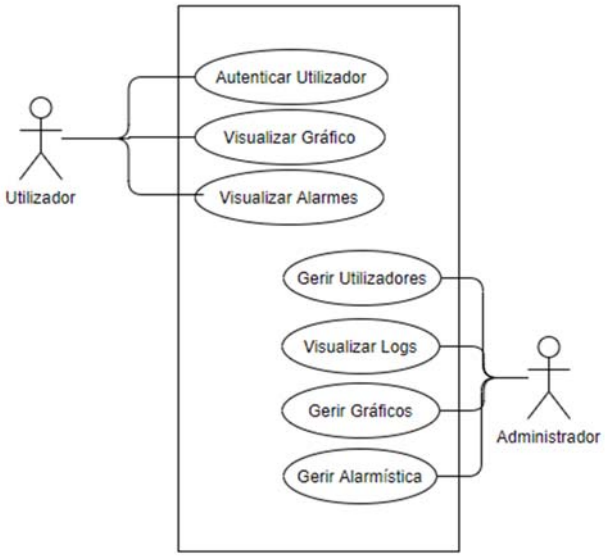


Figura 14 - Diagrama de Casos de Uso

A Tabela 3 demonstra os casos de uso presentes no sistema, contendo também uma breve descrição.

Tabela 3 - Descrição Casos de Uso

Nome	Ator	Descrição
Autenticar na Página	Cliente Visualizador	*Login* na plataforma com *Email ou Username* e palavra-passe
Visualizar Gráfico	Cliente Visualizador	Gráficos que disponibilizam informações sobre as API's
Visualizar Alarmes	Cliente Visualizador	Tabela com os status das API's
Gerir Utilizadores	Administrador	Possibilidade de adicionar, remover e configurar permissões dos utilizadores.
Visualizar logs	Administrador	Validar logs da plataforma

Gerir gráficos	Administrador	Capacidade de adicionar, modificar ou remover gráficos
Gerir alarmística	Administrador	Capacidade de adicionar, modificar ou remover alarmes

3.3 Requisitos do Sistema

3.3.1 Requisitos do Sistema

Neste tópico será definido de uma forma técnica e mais detalhada os requisitos funcionais para o projeto, conforme a Tabela 4 descreve.

Tabela 4 - Tabela Requisitos Funcionais

Código	Classificação	Ator	Descrição da Função dos Requisitos Funcionais
RF01	Essencial	*DataSource* Plugin	O Plugin de *DataSource* deve ser capaz de se conectar a banco de dados *Mysql*
RF02	Essencial	Plugin de *Dashboard*	Deve ser capaz de tratar os dados recebidos da tabela "status" para poder gerar informação de disponibilidade.
RF03	Essencial	Plugin de *Dashboard*	Deve ser capaz de gerar alarmística de acordo com o dado que recebe da tabela "status"

RF04	Essencial	Plugin de *Dashboard*	Deve ser capaz de tratar os dados em tempo real e gerar os alarmes necessários.
RF05	Importante	*Grafana*	Plataforma deverá ter utilizadores com privilégios diferentes, em que apenas o administrador seja capaz de adicionar, remover ou editar gráficos e alarmes.

3.3.2. Requisitos Não Funcionais

Neste tópico será definido de uma forma técnica e mais detalhada os requisitos não funcionais para o projeto, conforme a Tabela 5 descreve:

Tabela 5 - Requisitos Não Funcionais

Código	Classificação	Tipo	Descrição da Função dos Requisitos não Funcionais
RNF01	Essencial	Usabilidade	A *Dashboard* deve ser simples e de fácil compreensão, mesmo para utilizadores iniciantes.
RNF02	Essencial	Usabilidade	Os alarmes devem ser de simples compreensão e claro para a perceção de qual API se encontra indisponível.
RNF03	Essencial	Segurança	Os dados que são coletados entre o banco de dados e a *Dashboard* devem

			ser protegidos pois existem *APIs* que contêm informações sensíveis.
RNF04	Desejável	Usabilidade	Os utilizadores conseguirem publicar as suas *API's* na *Dashboard* sem a necessidade da intervenção do administrador da plataforma.

3.4 Diagrama de atividade

A Figura 15 demonstra a atividade que deve ser encadeada ao validar o estado das API's, caso tenha alguma API *Offline* é então necessário dar início a recuperação da mesma e voltar a validar até que nenhuma API se encontre mais *offline.*

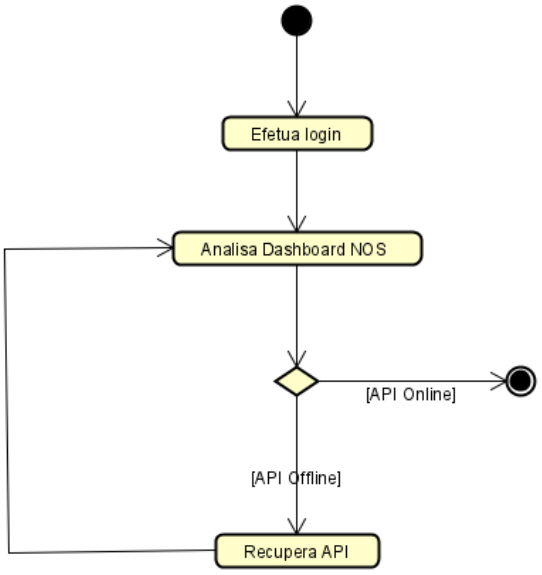


Figura 15 - Diagrama de Atividade

3.5 Software de suporte

Tendo em conta os requisitos funcionais deste projeto, optou-se por desenvolver a dashboard em Grafana, um software *OpenSource* que não tem nenhum custo e que vai de encontro com os requisitos. Está plataforma irá ser instalada em Docker sobre um servidor com CentOS 7.

Foi escolhido o Docker para este projeto, pois o *Grafana* disponibiliza uma imagem oficial que se encontra já otimizada. O Docker tem uma grande vantagem relativo as máquinas virtuais, pois os recursos do sistema são utilizados de forma mais eficiente, é utilizado muito menos memória, é possível iniciar e interromper uma aplicação muito mais rapidamente e é mais compacto a nível de armazenamento. Para além dessas razões este projeto irá estar apenas sobre uma máquina o que impossibilita o uso de Kubernetes que funciona apenas em cluster.

O projeto irá ser feito em CentOS 7 pois é o SO utilizado pela NOS em grande parte dos seus servidores.

4. Modelo e Arquitetura do Sistema

4.1 Arquitetura do Sistema

Para este projeto foi utilizado as tecnologias *Docker, Grafana, Mariadb e CentOS.*

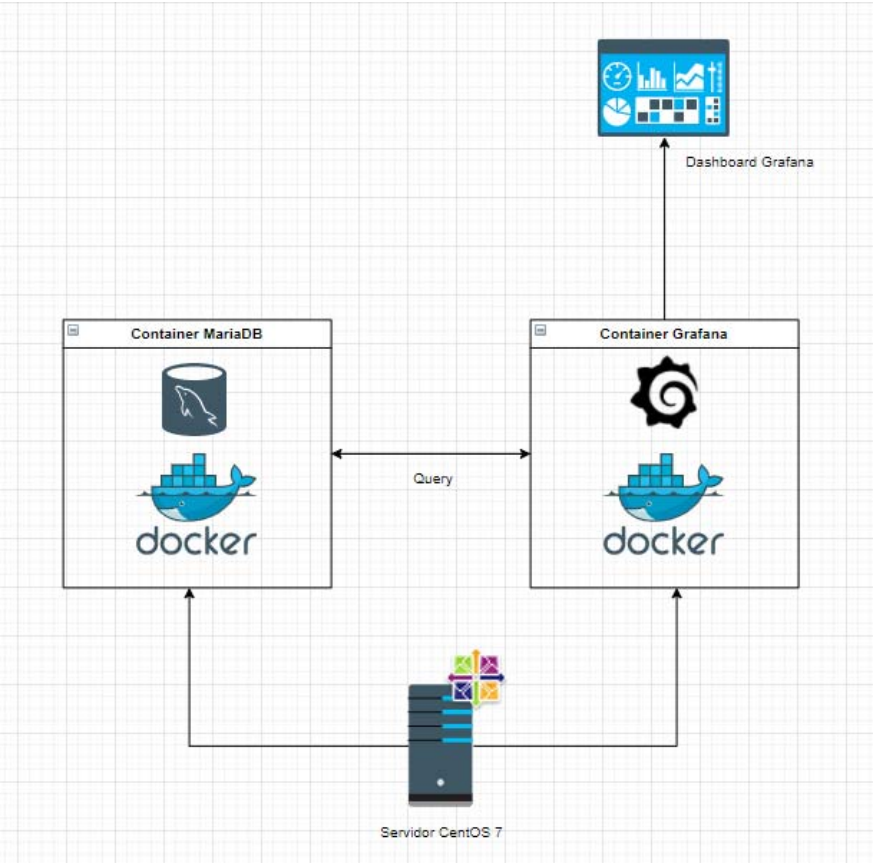


Figura 16 - Arquitetura do Sistema

A Figura 16 ilustra como o sistema se encontra repartido, o servidor onde se encontra instalado a aplicação contem um SO CentOS 7, dentro deste servidor se encontra instanciado dois containers, um com MariaDB e outro com o *Grafana,* A Aplicação do *Grafana* tem acesso ao banco de dados para efetuar query’s e trazer os dados para a Dashboard.

4.2 Arquitetura de Componentes

Conforme é ilustrado na Figura 17 os dados são coletados através do uso de *API's* HTTP que é disponibilizada pelo *Grafana.*

A cada vez que a página faz *refresh* é utilizado uma API do *Grafana* que envia a query configurada para o *dashboard* direto ao banco de dados, trazendo então o resultado da query como output.

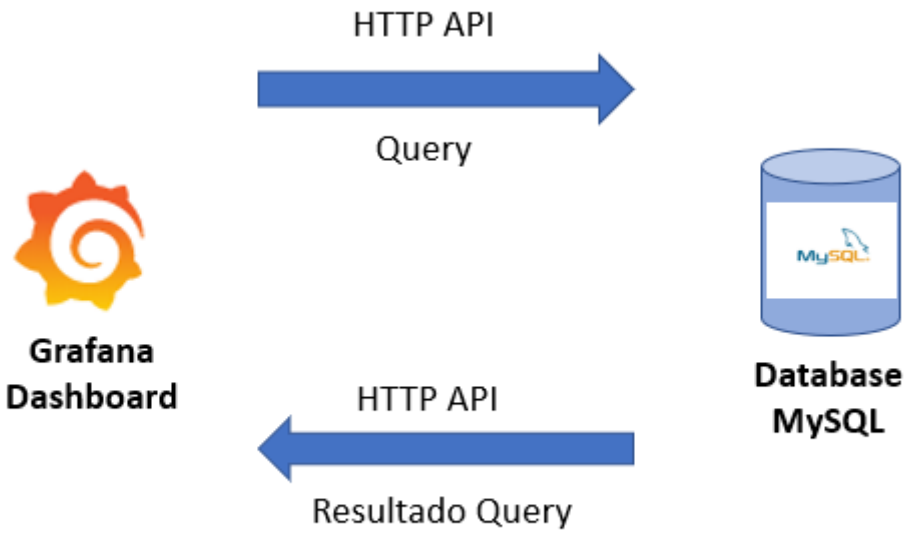


Figura 17 - Arquitetura de Componentes NOS API Dashboard

4.3 Entidade para representação de API's

O banco de dados para esta aplicação é disponibilizado pela NOS, servindo apenas para atualizar o status das API para que o *Grafana* possa se alimentar dos dados.

Este banco de dados é alimentado com os dados produzidos pelo *Jmeter*, são criados ficheiros com os outputs dos testes, após isso é então filtrado através de um *ShellScript* que preenche os dados de acordo com as colunas da tabela API.

Todo este procedimento de coleta de dados das API's é efetuado pela NOS entregando assim os valores finais para serem trabalhados no *dashboard*.

Esta BD é composta apenas por uma tabela.

A Entidade API contém os seguintes atributos:

- "id" – Atributo candidato (Chave primaria)
- "name" – Atributo Composto (Nome das *API's*)
- "status" – Atributo Identificador (Estado da API)

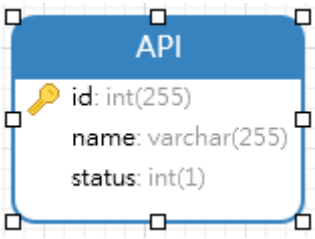


Figura 18 - Entidade API

Field	Type	Null	Key	Default	Extra
id	int(255)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
status	varchar(1)	NO		NULL	

Figura 19 - Descrição da tabela

5. Protótipo NOS API's Dashboard

5.1 Descrição Geral

O NOS API's Dashboard é um plugin criado para reproduzir o status das API's, estas API's têm impacto tanto nos serviços internos como externos, algumas dessas API's são responsáveis por executar comandos em equipamentos de clientes externos por exemplo modem e box ou por vezes em servidores internos, entre outras funcionalidades que as API's vêm incrementando ao negócio da empresa.

Os estados das API's são armazenados em uma BD MariaDB. Estas informações são utilizadas para gerar uma *TreeView* onde irá conter todas as API's agrupadas de acordo com o seu status, possibilitando que a supervisão consiga identificar log as que se encontram indisponíveis.

Neste Grupo irá ser demonstrado a ferramenta *Grafana* e a sua utilização de acordo com o caso de uso. Para demonstrar a interface, foi utilizado imagens do protótipo.

Este protótipo foi instalado sobre o sistema operativo CentOS 7, uma vez que a ferramenta será alocada em um servidor com este sistema operativo.

5.6 Interface Administrador

5.6.1 Configuração plugin Data Sources

Para que o *Grafana* tenha acesso aos dados que estão sendo guardados na BD é necessário configurar uma conexão na página do Grafana, é disponibilizado um *plugin* de *data sources* que permite configurar a conexão preenchendo o inquérito ilustrado na Figura 20

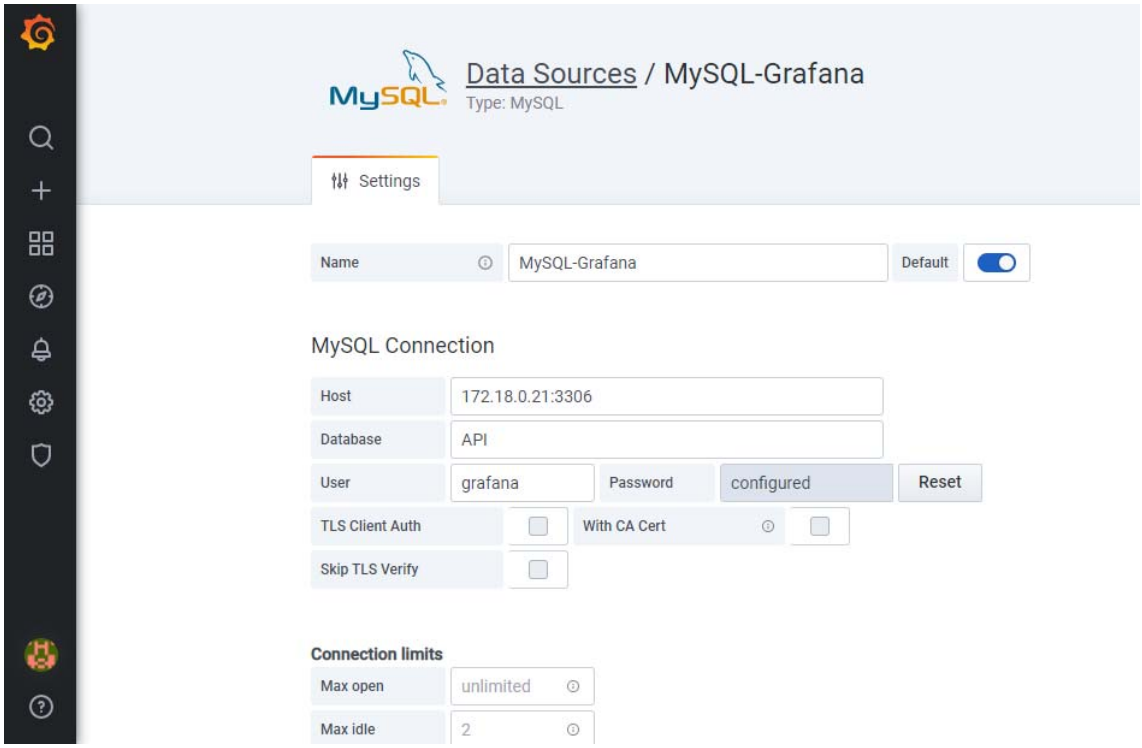


Figura 20 - Configuração do plugin data *source*

- Host – Deverá inserir o IP da máquina onde se encontra o banco de dados e a porta onde o serviço do MySQL está sendo executado;
- Database – É preciso preencher com o nome do banco de dados;
- User – É preciso preencher com o utilizador que foi criado no banco para que o Grafana possa se conectar e aceder aos dados;
- Password – É a senha configurada para acesso ao banco, em conformidade com o utilizador indicado no ponto atrás;
- Os outros dados devem ser preenchidos mais por questão de performance ou limitação do banco de dados, caso não tenha nenhuma limitação poderá deixar em branco.

5.7 Desenvolvimento do Plugin TreeView

Para dar início ao desenvolvimento do Plugin é necessário aceder ao repositório onde se encontra os plugins do *grafana* e criar um projeto usando o comando (*npx create-react-app "Nome Pasta"*).

Após fazer este procedimento irá ficar disponível o plugin na interface do *Grafana*, é possível então criar um dashboard, especificar a query que irá trazer os dados e selecionar o plugin que desejamos, como é possível verificar nas imagens abaixo.

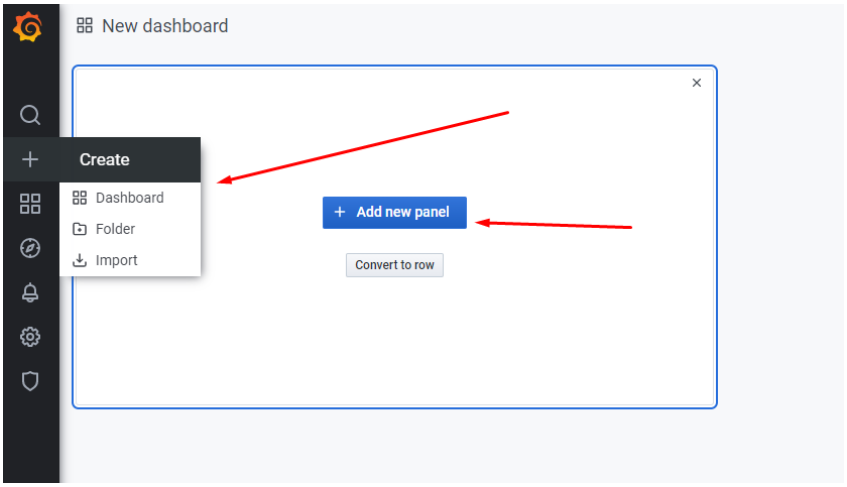


Figura 21 - Criação de Dashboard

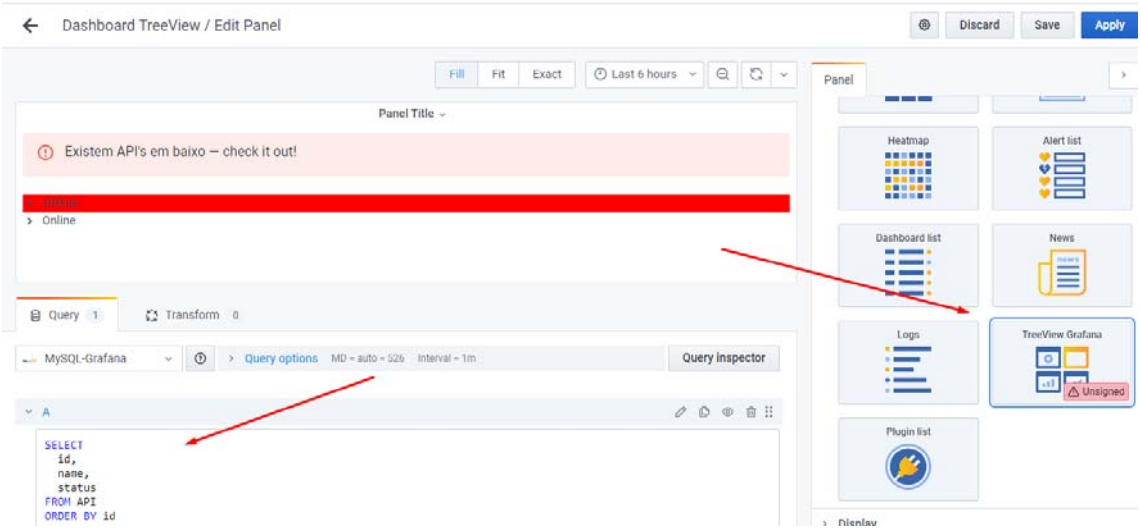


Figura 22 - Opções da Dashboard

Para o desenvolvimento deste *dashboard* foi necessário desenvolver um plugin em *Typscript* e *React*, pois até a data não existe nenhum plugin de TreeView disponível em G*rafana*, apesar de já haver este requisito aos desenvolvedores.

Foi utilizado as seguintes livrarias para possibilitar o tratamento de dados e disponibilizar na interface, Figura 23.

```
grafana-Treeview > src > TS TreePanel.tsx > ...
1   import React from 'react';
2   import { PanelProps } from '@grafana/data';
3   import { SimpleOptions } from 'types';
4   import { TreeView, TreeItem } from '@material-ui/lab';
5   import ExpandMoreIcon from '@material-ui/icons/ExpandMore';
6   import ChevronRightIcon from '@material-ui/icons/ChevronRight';
7   import Alert from '@material-ui/lab/Alert';
8
```

Figura 23 - Importes

O importe do "PanelProps" é responsável por permitir aceder a API de *datasource* do *grafana* permitindo assim trazer os dados do banco SQL. Os dados desta API são salvos em arrays como é possível ver na Figura 24.

```
let id = this.props.data.series[0].fields[0].values.toArray();
var apiname = this.props.data.series[0].fields[1].values.toArray();
var std = this.props.data.series[0].fields[2].values.toArray();
```

Figura 24 - Array de dados

Através do ".map" é possível ler o array de "id" criando assim o "apiid" que é usado para percorrer a casa dos *arrays* trazendo as informações do nome da API e o seu status. Toda está coleta é feita utilizando o (Id – 1) das API's para poder inicializar a 0. Na Figura 25 é possível verificar o tratamento dos dados através da utilização o "*if*" e salvando os dados dentro de um array contendo as API's *online* e *offline.*

```
let statonline = id.map((apiid: any) => {
  let arrayOnline: object[] = [];
  if (std[apiid - 1] === 1) {
    arrayOnline.push([apiname[apiid - 1]]);
  }
  return arrayOnline;
});
let statffline = id.map((apiid: any) => {
  let arrayOffline: object[] = [];
  if (std[apiid - 1] === 2) {
    arrayOffline.push([apiname[apiid - 1]]);
  }
  return arrayOffline;
});
```

Figura 25 - Array Status

Após conter os dados todos separados é então utilizado a livraria "@material-ui" que permite a construção da *Tree.* A Figura 26 e Figura 27 é a forma que os "*TreeItems*" são criados

```
});
let splitOnline = id.map((apiid: number) => {
  let nodeid = [2 + apiid];
  return <TreeItem nodeId={nodeid.toString()} label={statonline[apiid - 1]} />;
});
let splitOffline = id.map((apiid: number) => {
  let nodeid = [2 + apiid];
  return <TreeItem nodeId={nodeid.toString()} label={statffline[apiid - 1]} />;
});
```

Figura 26 - TreeItems

```
return (
  <div>
    <Alert severity="error">
      <h5>Existem API's em baixo – check it out! {statffline.length} </h5>
    </Alert>
    <br></br>
    <TreeView defaultCollapseIcon={<ExpandMoreIcon />} defaultExpandIcon={<ChevronRightIcon />}>
      <TreeItem nodeId="2" label="Offline" style={{ backgroundColor: 'red' }}>
        {splitOffline}
      </TreeItem>
      <TreeItem nodeId="1" label="Online">
        {splitOnline}
      </TreeItem>
    </TreeView>
  </div>
);
} else {
  return (
    <div style={{ fontSize: '6px' }}>
      <Alert severity="success">
        <strong>
          <h5>Todas as API's se encontram Online {statffline.length} </h5>
        </strong>
      </Alert>
      <br></br>
      <TreeView defaultCollapseIcon={<ExpandMoreIcon />} defaultExpandIcon={<ChevronRightIcon />}>
        <TreeItem nodeId="2" label="Offline">
          {splitOffline}
        </TreeItem>
        <TreeItem nodeId="1" label="Online">
          {splitOnline}
        </TreeItem>
      </TreeView>
    </div>
```

Figura 27 – TreeItem

5.8 *Dashboard* Final

Nas imagens a baixo consta o resultado do protótipo onde é possível verificar o estado final do *dashboard* com um exemplar de cada estado.

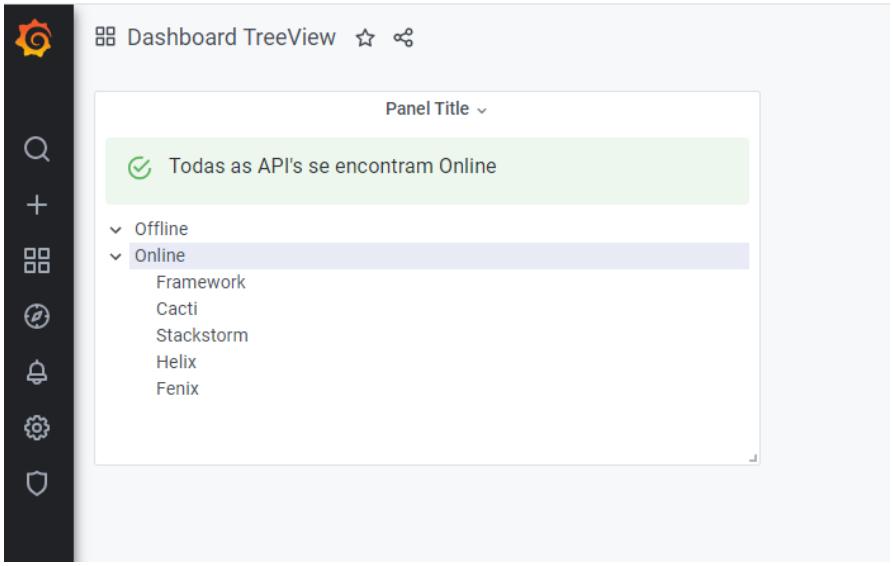


Figura 28 - API's Online

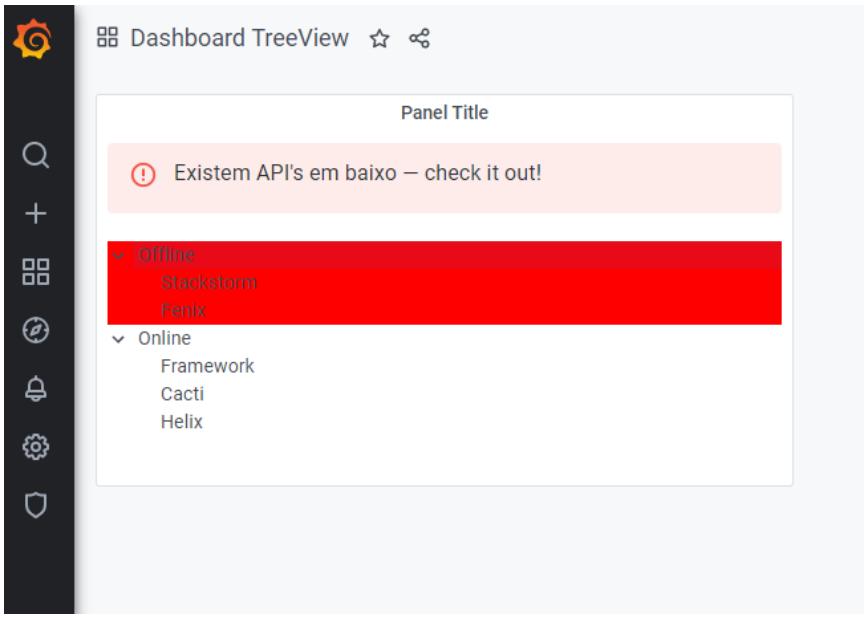


Figura 29 - API's Offline

6. Conclusão

"Uma vez que as API's são muito importantes para a continuidade dos serviços da NOS, surgiu a necessidade de ter uma monitoria extensiva sobre as mesmas para que caso venha a ficar indisponível, seja logo tomado as devidas intervenções para recuperar.

5) *Necessidade de um dashboard que fosse visível para várias equipes;*
6) *Ferramenta capaz de monitorar em real-time;*
7) *Equipe tem dificuldade de centralizar a monitoração das API's; e*
8) *Necessidade de tomar medidas imediatas assim que a API fique indisponível."*

Foram definidos os seguintes objetivos de investigação (RGs):

- RG 1 - Efetuar a revisão da literatura de modo a compreender os principais aspetos técnicos para o desenvolvimento de um *Dashboard* que respeite todas as normas de boa prática.
- RG 2 – Elaborar a especificação de requisitos da aplicação NOS API's Dashboard;
- RG 3 – Definir a Arquitetura do Sistema;
- RG 4 – Definir os principais Casos de Uso;
- RG 5 – Definir o Modelo de Dados; e,
- RG 6 – Desenvolver e testar o protótipo aplicacional NOS API's Dashboard.

Deste modo que os RGs foram feitos nos seguintes capítulos:

RG 1, foi realizado a revisão da literatura no Capítulo 2;

RG 2, foram definidos os requisitos funcionais e não funcionais da aplicação, como se demonstrou no Capítulo 3

RG 3, foi definido a Arquitetura do sistema por meio de Grafana no Capítulo 4

RG 4, foram ilustrados os Casos de Uso do Sistema, como demonstrado no Capítulo 3

RG 5, O modelo de dados necessário ao Sistema, foi apresentado no Capítulo 4

RG 6, O NOS API’s Dashboard foi realizado, como é possível verificar no Capítulo 5

Por este motivo, como conclusão preliminar, considera-se que o problema identificado foi resolvido e os objetivos de investigação cumpridos.

Como trabalho futuro, pretende-se que os dados estejam divididos entre serviços e infraestrutura, que tenha a funcionalidade de tentar recuperar a API direto na *dashboard* e se caso não conseguir então, enviar um alarme para o TeMIP, plataforma de alarmística da HP.

Bibliografia

Brath, R., & Peters, M. (2004). *Dashboard Design: Why Design is Important*. 4.

Eckerson, W. W. (Ed.). (2012). *Performance Dashboards: Measuring, Monitoring, and Managing Your Business* (1.ª ed.). Wiley. https://doi.org/10.1002/9781119199984

Few, S. (2007). *Dashboard Confusion*. 4.

Garcia-Molina, H., Ullman, J. D., & Widom, J. (2001). *Database Systems: The Complete Book*. 20.

GQM_texto.pdf. (2009). Obtido 19 de Junho de 2020, de https://www.cin.ufpe.br/~scbs/metricas/seminarios/GQM_texto.pdf

Hevner, A., & Chatterjee, S. (2010). Design Science Research in Information Systems. Em A. Hevner & S. Chatterjee, *Design Research in Information Systems* (Vol. 22, pp. 9–22). Springer US. https://doi.org/10.1007/978-1-4419-5653-8_2

Ivanov, V., Pischulin, V., Rogers, A., Succi, G., Yi, J., & Zorin, V. (2018). Design and validation of precooked developer dashboards. *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*, 821–826. https://doi.org/10.1145/3236024.3275530

Janes, A., Sillitti, A., & Succi, G. (2013). *Effective Dashboard Design*. *26*(1), 9.

Malik, S. (2005). *Enterprise dashboards: Design and best practices for IT*. Wiley.

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., & Hui, W. (2006). *THE DESIGN SCIENCE RESEARCH PROCESS: A MODEL FOR PRODUCING AND PRESENTING INFORMATION SYSTEMS RESEARCH*. 24.

Pham, T., & Liao, J. (2008, Maio 12). DSN Performance Dashboards. *SpaceOps 2008 Conference*. SpaceOps 2008 Conference, Heidelberg, Germany. https://doi.org/10.2514/6.2008-3494

Rob, P., & Coronel, C. (2010). *SISTEMAS DE BANCO DE DADOS*. 16.

Simulado DETRAN 2020—O mais seguro e atualizado. Obtido 19 de Junho de 2020, de https://www.aprovadetran.com.br

Suehring, S. (2002). *MySQL bible*. Wiley Pub.

Tipos de gráficos disponíveis no Office. Obtido 19 de Junho de 2020, de https://support.microsoft.com/pt-pt/office/tipos-de-gráficos-disponíveis-no-office-a6187218-807e-4103-9e0a-27cdb19afb90

F, A. I., Edwinah, A., & E, O. M. (2012). Use-of-Dashboard: A Vital Moderator of Sales Force Competence Management and Marketing Performance Relationship. *Information and Knowledge Management*, *2*(5), 30–39.

Pauwels, K., Ambler, T., Clark, B. H., LaPointe, P., Reibstein, D., Skiera, B., Wierenga, B., & Wiesel, T. (2009). Dashboards as a Service: Why, What, How, and What Research Is Needed? *Journal of Service Research*, *12*(2), 175–189. https://doi.org/10.1177/1094670509344213

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., & Hui, W. (2006). *THE DESIGN SCIENCE RESEARCH PROCESS: A MODEL FOR PRODUCING AND PRESENTING INFORMATION SYSTEMS RESEARCH*. 24.

Sommerville, I. (2016). *Software engineering* (Tenth edition, global edition). Pearson.

Caldeira, J. (2014). Monitorização da Performance Organizacional. Actual.

Anexo

A. Preparação do Sistema Operativo.

Para o desenvolvimento deste protótipo é necessário instalar uma ISO contendo o sistema operativo CentOS7 em uma VM, após ter o SO instalado é então necessário instalar apenas o Docker pois todos os outros componentes se encontram instanciados dentro dos containers.

Após conter o Docker instalado é necessário dar enable no serviço para que sempre que o servidor reinicie o docker seja também levantado (**systemctl enable docker.service**), após iniciar o serviço e dar *enable*, é então necessário criar pastar compartilhadas para facilitar o export e import de dados dos containers, para isso é utilizado o comando (***docker volume create*** **'nome'**), neste protótipo foram criados dois volumes partilhados, um com acesso a pasta de plugins do *Grafana* e outro ao ficheiro SQL da BD, como indicado na imagem a baixo.

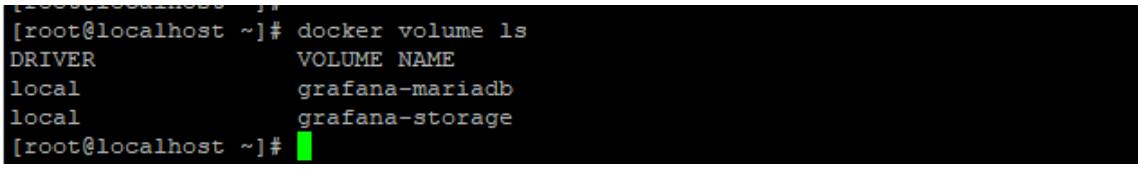


Figura 30 - Volumes criados

Para aceder a estas pastas partilhadas basta fazer um *cd* para */var/lib/Docker/volumes*, como é possível verificar na Figura 31.

```
[root@localhost docker]# df -h
Sist.fichs               Tama  Ocup Livre Uso% Montado em
devtmpfs                 475M     0  475M   0% /dev
tmpfs                    487M     0  487M   0% /dev/shm
tmpfs                    487M   27M  460M   6% /run
tmpfs                    487M     0  487M   0% /sys/fs/cgroup
/dev/mapper/centos-root   17G  7,0G   11G  41% /
/dev/sda1               1014M  167M  848M  17% /boot
tmpfs                     98M     0   98M   0% /run/user/0
overlay                   17G  7,0G   11G  41% /var/lib/docker/overlay2/3820ba69672e9e9f41
shm                       64M     0   64M   0% /var/lib/docker/containers/3270a38c824fd063
overlay                   17G  7,0G   11G  41% /var/lib/docker/overlay2/7b6126dc6a69c0566a
shm                       64M     0   64M   0% /var/lib/docker/containers/65cf206cb6f4c431
[root@localhost docker]# cd /var/lib/docker/
[root@localhost docker]# ls
containers  image  network  overlay2  plugins  swarm  tmp  trust  volumes
[root@localhost docker]#
```

Figura 31 - Localização dos volumes

Por ultimo é necessário criar uma rede para que os containers consigam se comunicar e sempre que o SO reinicie não altere o IP dos mesmos, para isso é necessário utilizar o seguinte comando (***docker network create --subnet='IP/MASK' 'Nome da rede'***), para este projeto foi criado a rede 172.18.0.0/16 com nome grafana (***docker network create --subnet=172.18.0.0/16 grafana***), esta configuração pode ser vista através do comando (***docker network ls***) de acordo com a Figura 32 e para validar as configurações da mesma é necessário utilizar o comando (***docker inspect 'nome'***).

```
[root@localhost docker]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
e2bff3688630        bridge              bridge              local
243bdc0bbe9c        grafana             bridge              local
9a721b23a776        host                host                local
373eb6664f9f        none                null                local
[root@localhost docker]#
```

Figura 32 - Docker Network

B. Configuração do container MariaDB

Para este protótipo foi utilizado um container do banco de dados MariaDB para armazenar os dados dos status das API's, para instanciar a imagem do MariaDB foi utilizado o seguinte comando (***docker run --network grafana --ip 172.18.0.21 --name mariadb -p***

"3306":"3306" -e MYSQL_ROOT_PASSWORD=grafana -v grafana-mariadb:/var/lib/mysql -d mariadb).

- **--network "nome"** - indica em qual rede esta imagem irá ser instanciada;
- **--ip "IP"** - indica qual será o IP do container;
- **--name** - indica qual o nome do container;
- **-p** - indica em qual Porta o serviço irá funcionar;
- **-e** - serve para passar dados nas variáveis de ambiente, neste caso foi utilizado para configurar a senha do MariaDB;
- **-v** - indica quais as pastas que irão ser compartilhadas entre o docker e o SO;
- No final indicar a imagem a ser instanciada.

Após instanciar a imagem do MariaDB é então necessário iniciar o container através do comando "**docker start mariadb**".

Após iniciar o container é então possível aceder ao mesmo para iniciar as configurações e criações do banco de dados e das tabelas. Para aceder ao container do MariaDB é preciso usar o comando (***docker exec -it mariadb bash***) este container já contém todos os serviços necessário para que o banco de dados funcione sem a necessidade de configurar novos serviços.

Necessário então aceder ao banco de dados usando (mysql -u "user" -p) para poder dar início as configurações necessárias.

1. **Criar o utilizador para aceder a BD:** *CREATE USER 'grafana' IDENTIFIED BY 'grafana';*
2. **Criar o banco de dados Figura 32:** *create database API;*
3. **Criar permissões para o utilizador:** *grant all privileges on API.* to grafana@'172.18.0.20' identified by 'grafana';*
4. **Atualizar os privilégios***: FLUSH PRIVILEGES;*

5. **Aceder a BD:** *USE API;*
6. **Criar a tabela para poder receber os dados:** *Create table API (id int(255) NOT NULL PRIMARY KEY AUTO_INCREMENT,name varchar(255) NOT NULL,status int(1) NOT NULL).*

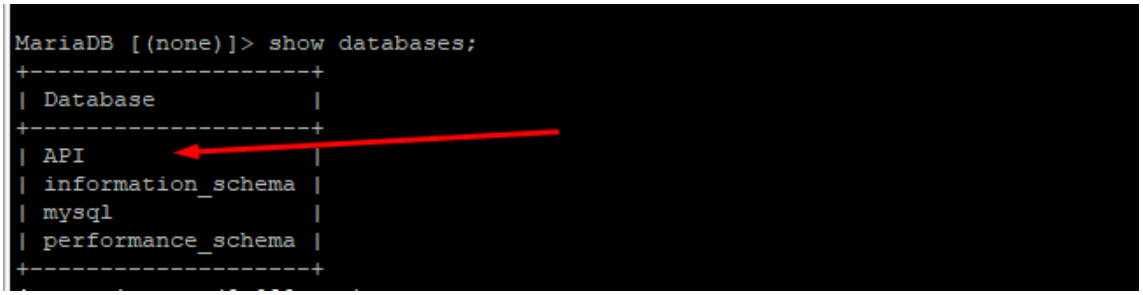


Figura 33 - Databases

A tabela API tem o campo "id" como auto incremental pois irá ser importante para manipular os dados para desenvolver o Plugin, Figura 34.

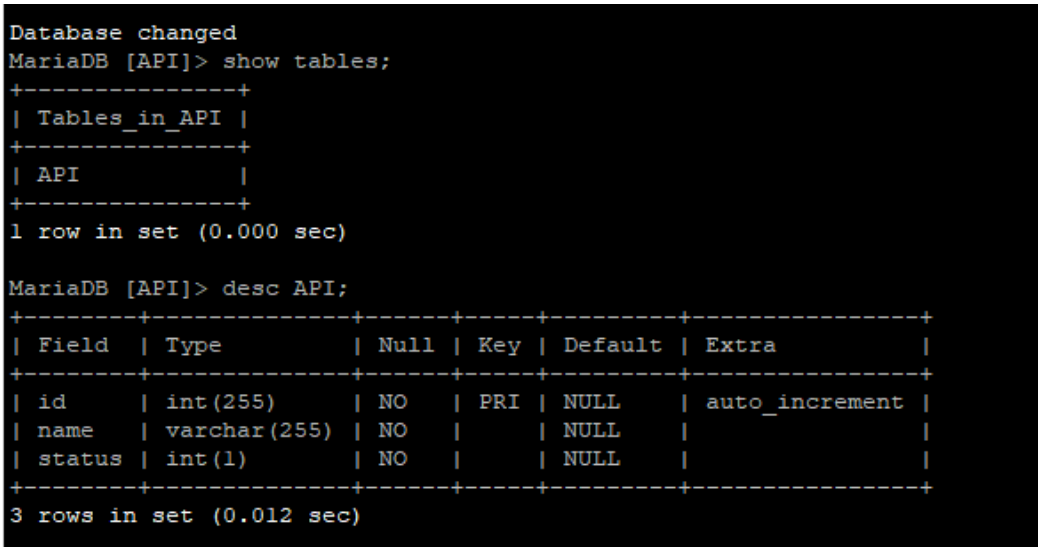


Figura 34 - Tabelas e Descrição

C. Configuração do container Grafana

A imagem para ser instanciada neste projeto é preciso conter o sistema operativo ubuntu, pois o desenvolvimento do plugin em *React* e *Typscript* necessitam do *Nodejs* e *Yarn*. Para isso foi então necessário recorrer a imagem "grafana/grafana:master-ubuntu" do repositório do docker. A imagem foi instanciada utilizando o seguinte comando (**docker run --network grafana --ip 172.18.0.20 --user 472 -d -p 3000:3000 --name grafana -v grafana-storage:/var/lib/grafana/plugins grafana/grafana:master-ubuntu**).

- **--network "nome"** - indica em qual rede esta imagem irá ser instanciada;
- **--ip "IP"** - indica qual será o IP do container;
- **--user** - utilizador root do container, neste caso é necessário o 472 pois a partir da versão 7.3 este é o utilizador que pertence ao grupo root;
- **-d** - faz com que o processo corra em background, pois se não utilizar este comando, ao fechar o terminal o container desliga;
- **-p** - indica em qual Porta o serviço irá funcionar;
- **--name** - indica qual o nome do container;
- **-v** - indica quais as pastas que irão ser compartilhadas entre o docker e o SO;
- No final indicar a imagem a ser instanciada.

Assim que a imagem se encontra instanciada é então necessário iniciar o container e conectar ao mesmo com o utilizador root (**docker exec -u root -it grafana bash**).

Para dar início a instalação do Nodejs e Yarn foram os seguintes passos:

1. **Atualizar o repositório do ubuntu:** *apt updat;*
2. **Instalar o Nodejs:** *apt install nodejs;*
3. **Instalar yarn:** *sudo yum install yarn;*
4. **Instalar o npm:** *apt install npm;*
5. **Fazer download do nvm:**

curl -sL https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh -o install_nvm.sh;

6. **Instalar o nvm:** *bash install_nvm.sh.*

Após instalar todos os pacotes que são necessários é então preciso utilizar o comando nvm para utilizar a versão desejada do Nodejs.

D. Interface com o Utilizador

Página de *Login*

Para iniciar a sessão no NOS API's Dashboard, o utilizador necessita efetuar a autenticação utilizando credenciais que sejam válidas, ou seja, endereço de *Email* ou *Username* que tenham sido cadastrados no sistema por um Administrador da ferramenta.

Na Figura 35 é possível verificar a página de login da ferramenta.

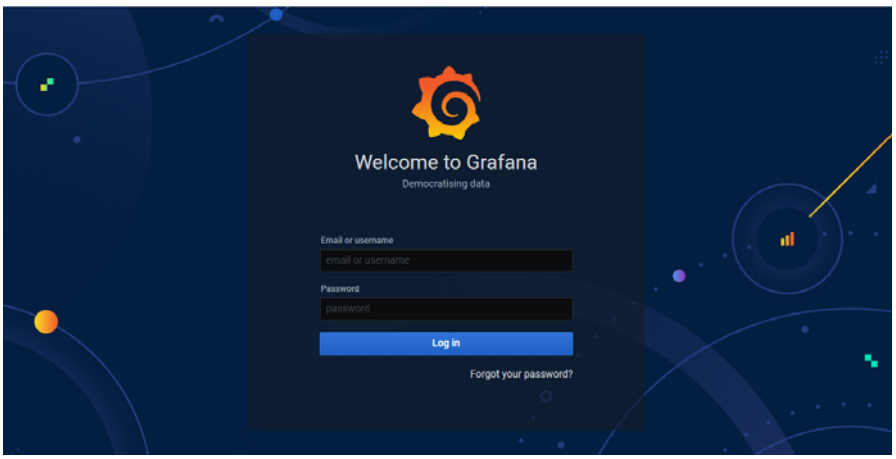


Figura 35 - Página de Autenticação NOS API's Dashboard

Caso o utilizador insira informações incorretas aparece uma caixa de diálogo informando que o *username* ou a senha está incorreto, não especificando qual por motivos de segurança para que caso haja uma invasão, dificulte que o invasor identifique se o *username* se encontra cadastrado na plataforma.

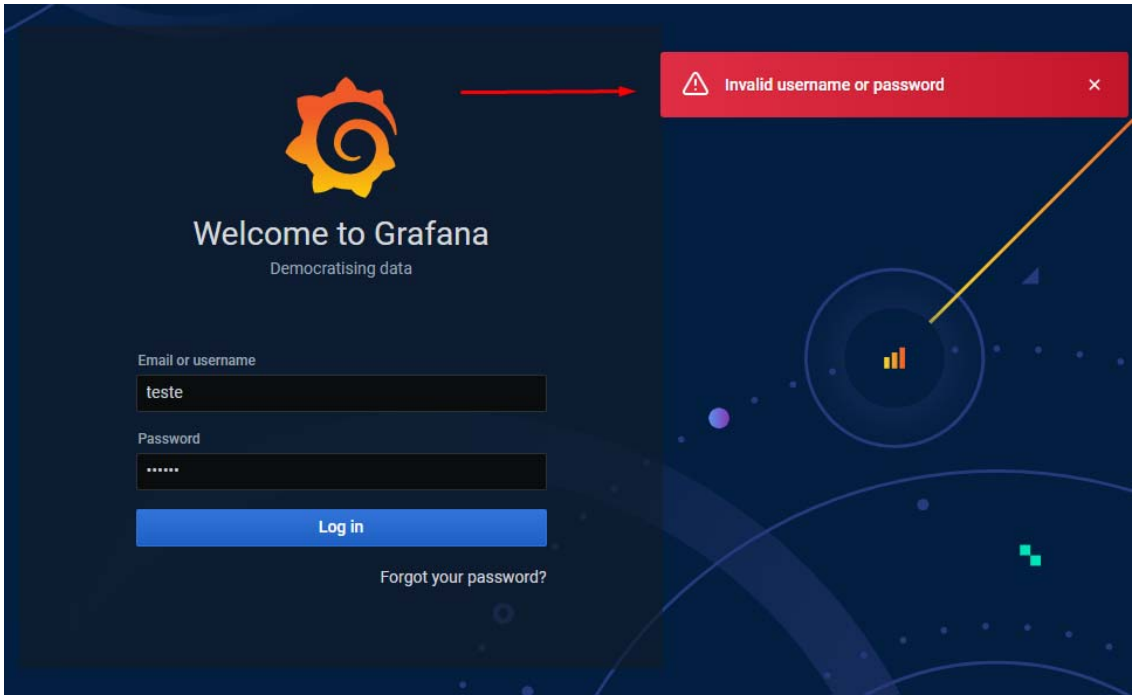


Figura 36 - Autenticação com dados inválidos

Reset Password

O utilizador caso tenha esquecido a senha de acesso, poderá utilizar a opção de "Forgot your password?", esta opção permite que o utilizador solicite o reset da palavra-passe.

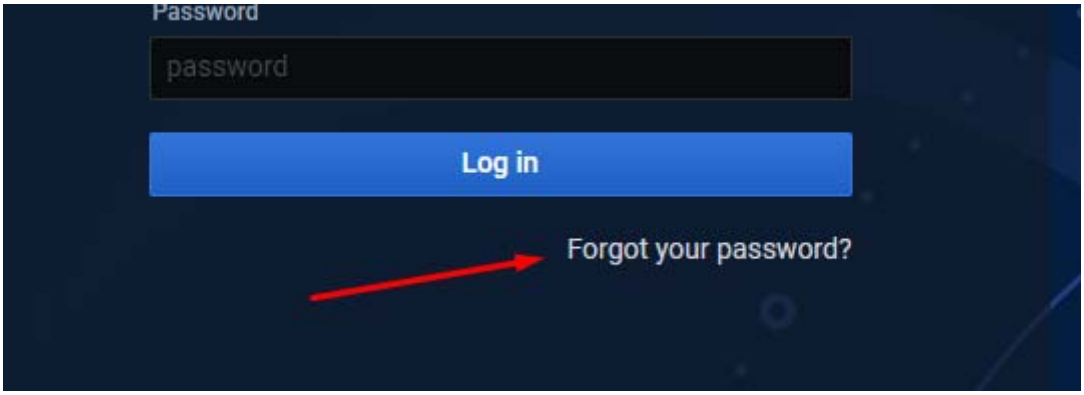


Figura 37 - Botão para reset da senha

Após clicar na opção "Forgot your password?" irá ser reencaminhado para uma página que contém um formulário onde solicita o email do utilizador para que seja enviado um link de restart da senha.

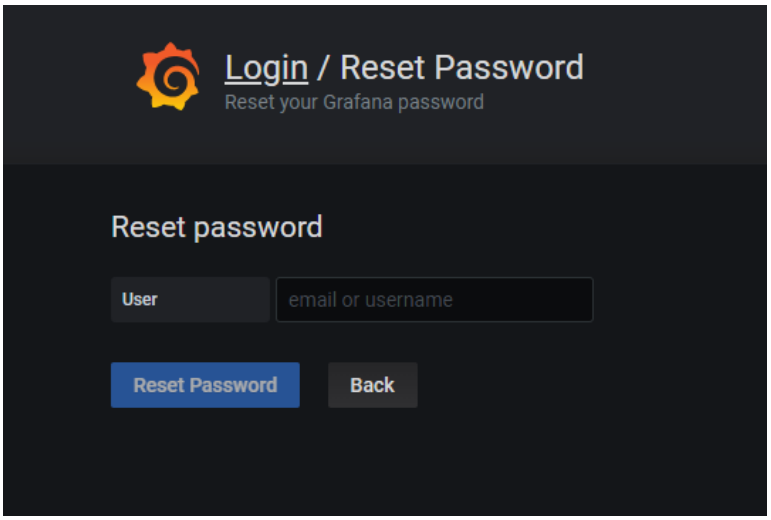


Figura 38 - Formulário de reset da palavra-passe

Página inicial do utilizador

Após efetuar o login utilizando dados validos, o utilizador terá de selecionar o *dashboard* clicando no canto superior esquerdo, onde irá ser reencaminhado para uma página que consta todas as opções de *dashboards* disponíveis.

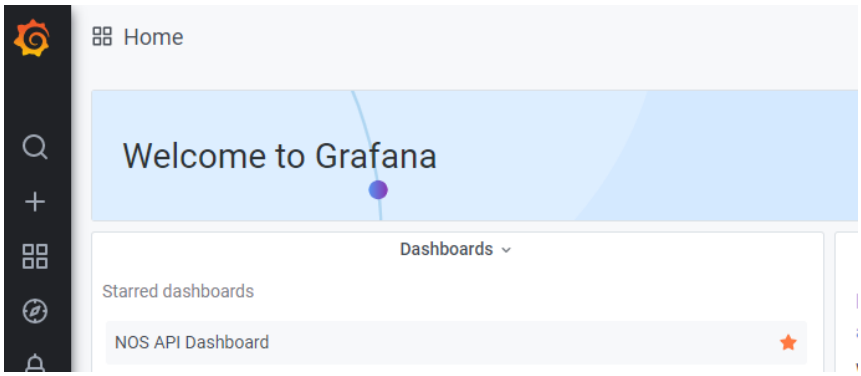


Figura 39 - Botão para visualizar dashboards

Após clicar na opção "Home" irá aparecer a seguinte página.

Figura 40 - Lista de Dashboards