INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




A solução de monitorização em tempo real para a indústria

JOSÉ MIGUEL BATISTA DE LEMOS
Junho de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

A solução de monitorização em tempo real para a indústria

José Miguel Batista de Lemos

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Piedade Carvalho

Porto, junho 2021

Dedicatória

Aos meus pais por sempre acreditarem em mim e nas minhas capacidades e por nunca me terem deixado desistir dos meus sonhos.

Resumo

Com a evolução da tecnologia e aumento da competitividade do mercado, alguns clientes da OEE Technologies passaram a optar por soluções concorrentes mais ágeis, escaláveis e com menores custos associados. Por consequente, o sistema de monitorização atual da empresa deixou de ser o mais eficiente e lucrativo.

Desta forma, era necessário procurar por novas soluções, algo que trouxesse um maior valor ao sistema de monitorização atual e que de certa forma corrigisse alguns dos problemas existentes. Nesse sentido, surgiu então a ideia de integrar uma nova funcionalidade que permitisse realizar uma monitorização em tempo real e em diferentes plataformas.

É neste contexto que surge então este projeto assente no desenvolvimento de um sistema de monitorização em tempo real que, baseado nos requisitos da empresa, calcula as métricas de disponibilidade, desempenho, qualidade e OEE, podendo ser facilmente incorporado na mesma.

Assim, espera-se que de uma forma prática e fácil, os utilizadores possam monitorizar a produção das suas empresas em tempo real, através de uma solução que surge como forma de apoio à análise do sistema produtivo e às decisões de gestão da produção.

Depois de desenvolvido e testado o projeto, concluiu-se que a solução desenvolvida acrescenta valor ao sistema de monitorização da empresa, pelo que ficam reunidas assim as condições para a sua integração.

Palavras-chave: monitorização em tempo real, métricas, disponibilidade, desempenho, qualidade, OEE e sistema produtivo

Abstract

The evolution of technology and the increase of market competitiveness, let some OEE Technologies customers choose more agile, scalable, and lower-cost competitive solutions. Consequently, the company's current monitoring system is no longer the most efficient and profitable choice.

Thus, it was necessary to look for new solutions, something that would bring greater value to the current monitoring system and that somehow could correct some of the existing problems. In this sense, then came the idea of integrating a new functionality that would allow the current system monitoring in real-time and on different platforms.

It is in this context that this project arises based on the development of a real-time monitoring system that based on the company's requirements calculates availability, performance, quality, and OEE metrics which can be easily incorporated into it.

Thus, it is expected that practically and easily, users can monitor the production of their companies in real-time through a solution that was developed to support the analysis of the production system and production management decisions.

After the project had been developed and tested it was concluded that the final solution adds value to the company's monitoring system, so that the conditions for its integration are met.

Keywords: real-time monitoring, metrics, availability, performance, quality, OEE and production system

Agradecimentos

Antes de mais quero agradecer a todos os que tornaram possível o desenvolvimento deste estágio.

Agradecer à empresa OEE Technologies, nomeadamente ao supervisor Jonas Jurjonas, pela oportunidade e confiança depositada em mim bem como todo o conhecimento e ajuda que me proporcionaram ao longo destes meses. Agradecer à família, nomeadamente os meus pais e o meu irmão, pelo apoio e conforto que me proporcionaram para conseguir desenvolver este projeto da melhor maneira e por fim, e não menos importante, à professora Piedade Carvalho pela paciência e disponibilidade sempre demonstrada para me ajudar em tudo que fui precisando para desenvolver este projeto.

Deixar também uma nota de agradecimento ao departamento de Erasmus que me aconselharam e tornaram esta experiência de estágio na Lituânia possível.

Conteúdo

Índice de Figuras

Índice de Tabelas

Índice de Excertos de Código

Acrónimos e Símbolos

Lista de Acrónimos

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
API	Application Programming Interface
APM	Application Performance Management
BD	Base de Dados
BMC	Business Model Canvas
CEO	Chef Executive Officer
CPU	Central Processing Unit
CSV	Comma Separated Values
ERP	Enterprise Resources Planning
FURPS	Functionality, Usability, Reliability, Performance e Supportability
GRASP	General Responsibility Assignment Software Patterns
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEEE	Instituto de Engenheiros Eletricistas e Eletrotécnicos
ISEP	Instituto Superior de Engenharia do Porto
JDBC	Java Database Connectivity
IoT	Internet of Things
IoS	Internet of Services
IP	Internet Protocol
JAR	Java ARchive
JSON	Javascript Object Notation
LAN	Local Area Network
MEI	Mestrado em Engenharia Informática
MVC	Model-View-Controller
NCD	New Concept Model
OEE	Overall Equipment Efficiency
PDF	Portable Document Format
SQL	Structured Query Language

SGBD	Sistema de Gestão de Base de Dados
SOLID	Single-responsiblity principle / Open-closed principle / Liskov substitution principle / Interface segregation principle / Dependency inversion principle
SSH	Secure Shell
TCP	Transmission Control Protocol
TMDEI	Tese / Dissertação / Estágio do Mestrado em Engenharia Informática do ISEP
TPM	Manutenção Produtiva Total
UC	Use Case
URI	Uniform Resource Identifier
XML	Extensive Markup Language

1 Introdução

O presente capítulo apresenta uma breve descrição do contexto em que este projeto surge, bem como uma explicação do problema da empresa OEE Technologies ao qual este pretende dar resposta. Além disso, são mencionados os objetivos que se pretendem atingir com o desenvolvimento deste sistema de monitorização e qual o valor que o mesmo trará para a empresa.

Para isso, é realizado um enquadramento, uma descrição dos objetivos, uma análise de valor e uma abordagem preconizada da solução que será desenvolvida. No final do capítulo é descrita a estrutura geral do documento.

1.1 Enquadramento

O presente projeto foi desenvolvido no âmbito da unidade curricular de Tese / Dissertação / Estágio (TMDEI) da área de especialização de Sistemas Gráficos e Multimédia, do Mestrado em Engenharia Informática (MEI) do Instituto Superior de Engenharia do Porto (ISEP).

Inserido num contexto de estágio curricular, esta dissertação foi realizada na empresa OEE Technologies, ao abrigo do plano Erasmus+, sob orientação da professora Piedade Carvalho e supervisionado pelo diretor executivo da empresa (CEO), Jonas Jurjonas.

No que diz respeito ao conteúdo, este documento descreve uma solução industrial de monitorização em tempo real que é analisada, explorada e desenvolvida ao longo do mesmo.

1.2 Empresa

Nascida em 2001, a OEE Technologies é uma empresa lituana, sediada em Vilnius, que presta serviços de consultoria a empresas dos mais diversos ramos. Desde o sector industrial da manufatura, passando pelo sector da saúde, finanças, entre outros, a OEE é um dos principais fornecedores de soluções para negócios na Lituânia.

A empresa presta serviços de consultoria aos seus clientes com software de planeamento e gestão de recursos empresariais (ERP). O seu principal objetivo passa por ajudar outras empresas a padronizar os seus negócios, construindo e desenvolvendo software que permita integrar, automatizar e gerir com segurança ambientes complexos, recorrendo a padrões de medição de produtividade OEE (OEE, 2021).

1.3 Problema

Com a evolução da tecnologia e aumento da competitividade do mercado, alguns clientes da OEE Technologies passaram a optar por soluções concorrentes mais ágeis, escaláveis e com menores custos associados. Com isso, para a OEE Technologies o sistema atual deixou de ser o mais eficiente e lucrativo. Os clientes passaram a exigir mais rapidez, um software de monitorização mais fácil e acessível de usar.

Atualmente o sistema apresenta diversos problemas, que dependendo do cliente têm uma maior ou menor gravidade, entre os quais se destacam:

- Informação relativa à monitorização alojada localmente;
- Serviço demasiado dispendioso;
- Sistema demasiado suscetível a erros;
- Interfaces pouco amigáveis para o utilizador.

Os três primeiros problemas apontados devem-se sobretudo ao facto da informação se encontrar alojada em servidores locais e em diferentes bases de dados que não comunicam diretamente entre si. Assim, os clientes têm de se deslocar à empresa para aceder e analisar os seus dados, ligando-se ao sistema através da rede local (LAN) o que torna o processo dispendioso não só em termos de tempo, como também financeiramente. Além disso, o facto dos dados se encontrarem distribuídos por diferentes bases de dados leva a que determinados serviços tenham elevados custos do ponto de vista operacional. Associado a isso, por vezes,

quando o sistema precisa de algum tipo de manutenção, torna-se demasiado lento, ficando mesmo inoperacional durante esse período, o que resulta uma vez mais em custos acrescidos para o cliente e para a empresa.

Relativamente ao último problema apontado, alguns clientes revelaram que apesar de agradados com o sistema de monitorização da OEE Technologies, tiveram alguma dificuldade em integrar e trabalhar com o mesmo, devido a uma interface demasiado complexa.

Deste modo, é agora necessário melhorar o sistema existente, integrando novas ferramentas de monitorização e análise dos dados em tempo real de modo aumentar a eficiência do serviço prestado.

1.4 Objetivo

Este projeto tem como principal objetivo o desenvolvimento de um sistema de monitorização da produção de empresas cliente em tempo real, como forma de apoio à análise do sistema produtivo e às decisões de gestão da produção. Foi visto na secção 1.3 que o sistema atual não serve, terá de ser corrigido, sendo que este projeto assume o objetivo de solucionar as lacunas mencionadas tornando todo o sistema mais eficiente.

Uma vez que se trata duma melhoria do sistema em vigor, com a integração dum novo software de monitorização acessível remotamente, tanto em web como em mobile, numa fase inicial o objetivo passa por perceber o sistema atual. É crucial ter em consideração todos os requisitos do sistema em vigor e entender quais deles permanecem dentro dos procedimentos padrão da OEE Technologies, prestando muita atenção à proteção e segurança de dados. Por outro lado, é preciso obter todo o conhecimento possível sobre as vantagens / desvantagens alcançadas por cada uma das possíveis soluções, a fim de estabelecer a melhor abordagem com as tecnologias mais adequadas, a fim de permitir a sua escalabilidade.

Relativamente à melhoria da interface do sistema, será marcada como um objetivo extra, que se for alcançado, tornará o sistema mais amigável para o utilizador, facilitando a sua experiência. Mas por enquanto, o foco principal passa pela integração do novo sistema de monitorização em tempo real, o que já por si só requer uma investigação e análise profunda.

1.5 Análise de valor

A solução aqui desenvolvida aponta para um aperfeiçoamento de software existente na empresa que na conjuntura atual não se encontra totalmente otimizado, causando transtornos para os desenvolvedores e sobretudo para os clientes.

Com a análise, redesenho e otimização do software com novas tecnologias, o serviço valoriza, o que acaba por ser do agrado dos clientes, traduzindo-se num reforço da sua confiança e satisfação para com os serviços da empresa. Isto acontece porque a monitorização se torna não só mais eficiente, como menos dispendiosa, resultando numa melhoria significativa do negócio de cada cliente. Já para a empresa, o valor é igualmente elevado, uma vez que vê a qualidade dos seus produtos não só aumentar como ganha também vantagem em relação a outros competidores.

De realçar que esta análise é realizada de forma mais extensiva no subcapítulo “Análise de Valor” do capítulo “Contexto e Estado da arte”.

1.6 Abordagem preconizada

Nos sistemas de informação, por norma há duas metodologias que são utilizadas no processo de pesquisa para definir a melhor maneira para se obter conhecimento sobre o assunto em estudo: a ciência do comportamento e a ciência do design. A primeira metodologia está associada à criação de teorias e tenta justificar um conceito a partir da compreensão que se tem da matéria, enquanto a segunda está associada à criação de artefactos e representa a forma prática de provar uma determinada solução com diferentes avaliações ou testes (Goldkuhl, 2016).

Nesse sentido, esta dissertação apresenta uma investigação centrada na resolução de um problema, bem como na resolução das preocupações existentes no contexto organizacional da OEE Technologies, seguindo a metodologia design science.

Assim, e de acordo a metodologia seguida, podem-se identificar 6 etapas diferentes:

1. **Identificação do problema e motivação:** Introduzida na secção 1.3 e aprofundada na secção 2.2, esta etapa ajuda a entender o problema da OEE Technologies relacionado

com facto do seu processo de monitorização em tempo real não se encontrar ao nível desejado, bem como as motivações que levaram ao surgimento deste projeto.

2. **Definição dos objetivos:** uma vez identificado o problema e esclarecidas as motivações para o resolver, são definidos nesta etapa os objetivos da solução. De um modo geral o objetivo passa pelo desenvolvimento de um sistema de monitorização em tempo real, que se encontra detalhado na secção 1.4.
3. **Design e desenvolvimento:** depois de definidos os objetivos, o projeto passa então a ser posto em prática. Assim, são definidos inicialmente, em conjunto com a empresa, os requisitos necessários para a concretização do projeto (Capítulo 3). De seguida, passa-se então para o desenho da solução pretendida através de diferentes tipos de diagramas e tabelas (Capítulo 4). Uma vez desenhada a solução, procede-se então à implementação da mesma no Capítulo 5.
4. **Demonstração:** assim que se obtém a solução final, tem de se demonstrar a eficácia e eficiência da mesma através de diferentes tipos de testes. Para isso desenvolver-se-á o Capítulo 6 com toda as demonstrações realizadas.
5. **Avaliação:** passando nos testes, inicia-se então o balanço final no Capítulo 7, onde se avalia a qualidade da solução obtida, comparando os objetivos estabelecidos no início com os resultados observados com a solução.
6. **Comunicação:** depois de todas as etapas concluídas com sucesso, e comprovada a qualidade e valor da solução, dá-se início à última etapa com a divulgação da sua utilidade e eficácia a outros pesquisadores e profissionais (Pello, 2018). Neste caso, pretende-se que a mesma seja divulgada para a empresa, uma vez que é uma parte interessada, servindo o Capítulo 8 para esse efeito.

1.7 Estrutura

Este documento é composto por seis capítulos, sendo eles a Introdução, Contexto e Estado da Arte, Avaliação de Soluções/Abordagens Existentes, Experimentação e Avaliação e, por fim, Conclusões e Trabalho Futuro.

O primeiro capítulo é formado por seis secções: Enquadramento, Empresa, Problema, Objetivo, Análise de Valor, Abordagem Preconizada e Estrutura. Nas primeiras quatro secções pretende-se que o leitor fique com uma noção geral do problema identificado e o seu contexto. De

seguida, apresenta-se um pequeno resumo da Análise de Valor da solução bem como a sua abordagem preconizada. Por fim, inclui-se a presente secção onde é apresentada a estrutura deste documento.

Já o segundo capítulo inicia-se com um aprofundamento do contexto e o problema, para que o utilizador compreenda a importância deste projeto para a empresa. Além disso, é elaborada uma Análise de Valor da solução concebida. De seguida, é realizado o Estado da Arte onde se faz um estudo de softwares que podem ser integrados com o software da empresa de forma a solucionar o problema aqui representado. Por fim é realizada uma comparação entre as soluções obtidas e é efetuada a seleção daquelas que serão integradas.

Apresenta também os capítulos 3 e 4 relativos à análise e desenho da solução, respetivamente, capítulos esses que contêm o domínio do problema (glossário, modelo de domínio), os requisitos funcionais e não funcionais do projeto realizado, assim como uma explicação detalhada da sua arquitetura. Também contém o Capítulo 5 em que é descrito com detalhe o enquadramento e implementação da solução.

O Capítulo 6 é dedicado aos testes que foram realizados ao sistema desenvolvido.

Em jeito de conclusão, o relatório termina com o Capítulo 7 das conclusões, onde se encontram os objetivos concretizados, uma apreciação final do projeto, as limitações e possíveis direções do trabalho no futuro.

2 Contexto e Estado da arte

2.1 Detalhes de Contexto e Problema

Este capítulo apresenta um enquadramento teórico do projeto desenvolvido com uma descrição detalhada da indústria 4.0. Além disso, é realizada uma análise pormenorizada do problema atual da empresa OEE Technologies relacionado com o seu sistema de monitorização, e que foi introduzido, de forma breve, na secção 1.3.

2.1.1 Contexto

As três primeiras revoluções industriais foram determinadas por grandes avanços tecnológicos, que permitiram o aparecimento de grandes invenções da história da humanidade como a máquina a vapor, a eletricidade e os computadores (Beke, Horvath, & Tak-acsne, 2020). Com o passar dos anos e com as constantes inovações tecnológicas a que se assistiram, a indústria passou por importantes e profundas mudanças. Associado a isso, assistiu-se ao desenvolvimento de uma nova realidade, com uma transformação tecnológica sem precedentes à qual a sociedade precisa a todo o custo de se adaptar (SAKURAI & ZUCHI, 2018).

Este desenvolvimento está de certo modo relacionada com a globalização, na qual as empresas pretendem obter vantagens competitivas e duradouras que as diferenciem na sua área de negócio. Com isso, novas estratégias surgem de modo a acompanhar o mercado e constantes melhorias nos processos internos das empresas são realizadas. O objetivo passa sempre por aperfeiçoar o nível do serviço prestado tanto para os clientes como para os fornecedores, no menor espaço de tempo, ao melhor custo possível e com o máximo de eficiência nos processos (de Oliveira & Simões, 2017).

E é com base neste contexto que assistimos atualmente à quarta revolução industrial, denominada como indústria 4.0, que se encontra em expansão e a transformar os modelos de gestão da indústria mundial (de Oliveira & Simões, 2017).

2.1.1.1 Indústria 4.0

Segundo (KAGERMANN, 2013), o termo Indústria 4.0 teve origem através de um projeto entre empresas, universidades e o governo alemão, sendo citado pela primeira vez durante a feira de Hannover na Alemanha, em 2011, num contexto de modernização das indústrias locais.

A essência desse novo modelo industrial assenta na forma de descrever a propensão da digitalização e a automação do ambiente de manufatura (OESTERREICH & TEUTEBERG, 2016). O fundamento básico passa pela conexão entre máquinas, sistemas e ativos, possibilitando às empresas a criação de redes inteligentes de modo a controlar os módulos de produção de forma autónoma (SILVEIRA, 2017).

O conceito da Indústria 4.0 é assim a combinação das conquistas tecnológicas dos últimos anos com a visão de um futuro com sistemas de produção inteligentes e automatizados, onde o mundo real é ligado ao virtual (ZAWADZKI & ŻYWICKI, 2016).

Para a implementação desta quarta revolução industrial há seis pilares que segundo a (Siemens) são essenciais:

- **Capacidade de operação em tempo real** - uma vez que a aquisição e tratamento de dados em tempo real é um fator que agiliza a tomada de decisões;

- **Virtualização** – através de uma cópia virtual das fábricas inteligentes, consegue-se assim o rastreio e monitorização das mesmas de forma remota;

- **Descentralização** – sendo as decisões tomadas pelo sistema ciber-físico como forma de atender às necessidades da produção em tempo real. Neste sistema, as máquinas geram informações que são enviadas para o software, tornando possível o controlo, monitorização e análise dos dados que são transferidos automaticamente entre o mundo físico (máquinas) e o ciber (softwares) (Pederneiras, 2019);

- **Orientação de Serviços** – através da utilização de arquiteturas de software orientadas a serviços aliadas ao conceito de Internet of Services (IoS);

- **Modularidade** – sendo a produção realizada de acordo com a demanda, acoplamento e desacoplamento dos seus módulos, o que permite alterar as tarefas das máquinas facilmente;

- **Interoperabilidade** - capacidade dos sistemas ciber-físicos (suportes de peças, postos de reunião e produtos), humanos e fábricas inteligentes em comunicarem entre si por intermédio da Internet das Coisas e da Internet.

A sustentar estes pilares da quarta revolução industrial, para a implementação e funcionamento da Indústria 4.0 surgem assim as seguintes tecnologias:

- **Internet of Things** – consiste na relação entre as coisas (produtos, serviços ou lugares) e as pessoas, por meio de plataformas e tecnologias conectadas (SCHWAB, A quarta revolução industrial., 2016). Conhecida também como internet das coisas ou IoT, a mesma permite interligar informações de toda a cadeia produtiva, recolhendo dados e informações através de sensores que se ligam às máquinas já conectadas à rede, possibilitando análises em tempo real de cada parte do processo de produção (Siemens);

- **Segurança cibernética** – passa por um conjunto de ações preventivas de modo a impedir os ataques cibernéticos em dispositivos. O seu objetivo passa por salvaguardar a confidencialidade, integridade e disponibilidade da informação através de por exemplo o uso de sistemas antivírus, criptografia e backups de toda a informação relevante. Uma vez que os problemas ou falhas podem comprometer todo um trabalho em desenvolvimento, a segurança é considerada a base para o sucesso de qualquer programa ou produto tecnológico (SILVEIRA, 2017);

- **Big Data e Analytics** - são estruturas de grandes volumes de dados, complexas e que utilizam abordagens inovadoras para a captura, análise e gestão de informações (SILVEIRA, 2017);

- **Computação em nuvem** - consiste numa base de dados (BD) que é acessível de qualquer parte do mundo, em milissegundos, por meio de dispositivos conectados à internet (RUBMANN, et al., 2015). Além disso, esta tecnologia garante mobilidade, escalabilidade e segurança aos processos através da utilização de servidores virtuais, o que permite ampliar as possibilidades de conexão entre sistemas (Siemens);

- **Robótica avançada** - robôs adaptáveis e flexíveis com perspetivas de no futuro tornarem a interação entre as máquinas e humanos uma realidade (SCHWAB, A quarta revolução industrial., 2016);

- **Inteligência artificial** - através dos dados recolhidos, armazenados e analisados, os algoritmos permitem que as máquinas aprendam da mesma forma que o cérebro humano aprende. Assim, tarefas que atualmente são feitas por pessoas (Siemens) poderão ser realizadas por máquinas, ajudando a reduzir custos, uma maior eficiência e até mesmo "computorizar" empregos (SCHWAB, A quarta revolução industrial., 2016);

- **Novos materiais** - mais leves, fortes, recicláveis e adaptáveis podem ser "inteligentes" com propriedades como autorreparação ou autolimpeza (SCHWAB, A quarta revolução industrial., 2016).

A Indústria 4.0 expõe assim um novo modelo de funcionamento industrial. Baseada na conectividade e num grande volume de dados geridos em tempo real, assume um compromisso na apresentação de resultados positivos na produção das indústrias. No entanto, apesar de todas as vantagens as empresas ainda se deparam com grandes dificuldades na sua implantação (Indústria, 2018).

2.1.2 Problema

As empresas geram, recolhem e armazenam uma grande quantidade de dados provenientes das muitas tarefas necessárias à operacionalização dos seus negócios. Porém, em algumas dessas organizações os dados não são mantidos numa única base de dados. A informação fica repartida por diferentes sistemas computacionais, cada qual alocado com uma função específica, em unidades de negócio diferentes e sem troca de dados entre os mesmos (Junior & Pires, 2010).

Associado a isso advêm grandes custos para armazenar, racionar e reformatar dados redundantes entre sistemas, atualizar e manter códigos obsoletos, assim como regular a comunicação entre esses recursos distintos numa tentativa de automatizar a transferência de dados (Junior & Pires, 2010). Nesse sentido, resumidamente pode se dizer que se os sistemas de uma determinada companhia são fragmentados, inevitavelmente o seu negócio também é fragmentado (DAVENPORT, 1998).

Por outro lado, a maioria dos sistemas e softwares capazes de monitorizar e gerir todos estes dados, devido à elevada quantidade de funcionalidades, são complexos e requerem uma instalação e configuração especializada por parte dos clientes (Stock, Stöhr, Rauschecker, & Bauernhan, 2014). No caso da OEE Technologies, como foi referido na secção 1.3, os clientes têm de se deslocar à empresa, configurar e conectar-se à rede local a partir dos seus computadores através de uma ligação LAN, com toda a complexidade associada a esse processo. Com isso, os custos e meios necessários para a instalação e manutenção desses sistemas, podem ser demasiado elevados para a dimensão de certas empresas. Nesses casos, o serviço prestado torna-se pouco rentável, não sendo obtida uma boa relação custo/benefício para o cliente (Stock, Stöhr, Rauschecker, & Bauernhan, 2014).

Face a todos estes problemas, é visível que as soluções ERP que a empresa oferece não são as mais eficientes e com maior valor para o cliente. Por conseguinte, uma perceção de baixo valor e elevados custos na implementação do um novo sistema pode causar uma certa resistência por partes dos utilizadores (Mahmud, Ramayah, & Kurnia, 2017). Isto implica que os vendedores de ERP, neste caso a OEE Technologies, produzam um sistema o mais simples, sustentável e flexível possível, que ofereça ferramentas eficientes e satisfaça as exigências dos utilizadores com o mínimo esforço e custo (Eid & Abba, 2017), o que atualmente, por todos os problemas mencionados, não acontece.

2.2 Análise de valor

Neste capítulo, é analisado o valor deste projeto, de modo a avaliar o seu potencial no contexto do problema. Aqui serão descritos diversos conceitos que englobam a criação de valor, incluindo o desenvolvimento do termo de inovação e o valor da solução.

O conceito de inovação no contexto empresarial, é um conceito geral que pode ser aplicado a produtos, serviços, esforços e políticas diferentes. Pode incluir novos produtos que atenderão melhor os clientes ou um novo programa que melhorará a comunicação interna de uma empresa, ajudando os funcionários a se comunicarem melhor sobre os projetos em que trabalham (Ward, 2020). Por norma, associado à inovação vem o conceito de valor. Segundo (Cole, 2015), o valor associado à inovação é um processo no qual uma empresa introduz novas tecnologias ou atualizações que são projetadas para alcançar a diferenciação do produto e baixos custos.

Nesta tese, a fim de conduzir a análise de valor, será usado o modelo proposto por (Koen P. , et al., 2001), denominado de New Concept Development Model (NCD).

2.2.1 New Concept Development Model (NCD)

O "*front-end*" é uma parte importante de um processo inovador bem-sucedido, cobrindo todas as suas etapas desde a estratégia, à geração da ideia até ao plano de negócio. A qualidade do trabalho realizado nesta fase é decisiva para o sucesso da inovação, pois é nesta etapa onde as oportunidades de inovação são identificadas, analisadas e trabalhadas (ESCHBERGER, 2018).
Neste contexto surge assim o modelo NCD, fornecendo uma ajuda importante para a compreender das atividades que ocorrem no "*front-end*" (Koen, Bertels, & Kleinschmidt, Managing the Front End of Innovation—Part II: Results from a Three-Year Study, 2014).

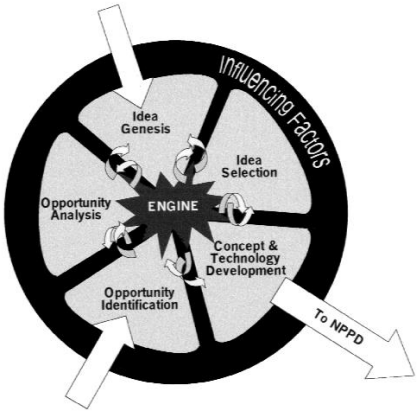

Figura 1 - O modelo NCD (Koen, Ajamian, & Burkart, 2001)

Como podemos observar na Figura 1, o modelo NCD divide-se em três áreas distintas: o motor, a roda e o aro. O motor, no centro do modelo, fornece potência para a inovação, fortalecendo os componentes essenciais para o sistema, tais como a visão, estratégia, recursos, equipas e colaboração. Já na parte interna tem-se a roda, onde (Koen P. , et al., 2001) define 5 elementos essenciais:

1. **Opportunity Identification** – é aqui que se identificam as oportunidades para a empresa normalmente impulsionadas pelos objetivos do seu negócio. A oportunidade pode dar uma direção inteiramente nova ao negócio ou melhorar o produto já existente. Também pode ser uma nova plataforma para o produto, um novo processo de fabrico, uma nova oferta de serviço ou uma nova abordagem de marketing ou vendas. As fontes

e métodos que uma empresa usa para identificar as oportunidades que deseja seguir são a essência deste elemento.

2. **Opportunity Analysis** – uma vez identificadas as oportunidades, são necessárias informações adicionais para as traduzir em negócios específicos e oportunidades tecnológicas, assim como para fazer avaliações precoces e frequentemente incertas das tecnologias e do mercado em geral. Este elemento pode fazer parte de um processo formal ou ocorrer iterativamente em reação às oportunidades identificadas. Tanto a inteligência competitiva como as análises das tendências são amplamente utilizadas neste elemento.
3. **Idea Genesis** – génesis é o nascimento, desenvolvimento e maturação de uma oportunidade numa ideia concreta. Através de um processo evolutivo, é nesta fase que as ideias são construídas, destruídas, combinadas, remodeladas, modificadas e atualizadas. A geração de ideias pode ser um processo formal, incluindo sessões de brainstorming que ajuda a organização a gerar ideias novas ou aperfeiçoar as existentes. Com este elemento espera-se por isso uma descrição mais concreta e desenvolvida da ideia ou do conceito de produto "detetado".
4. **Idea Selection** – uma vez desenvolvida a ideia, inicia-se então o processo de escolha. Na maior parte das empresas há inúmeras ideias de produtos e processos, pelo que o processo de escolha das ideias com maior valor comercial se torna bastante complicado. Nesta fase, projetam-se modelos que têm em consideração os riscos do mercado e da tecnologia, níveis de investimento, realidades competitivas, capacidades organizacionais e vantagens exclusivas, juntamente com os retornos financeiros. A seleção de ideias, como na análise de oportunidades, deve ser menos rigorosa, uma vez que muitas ideias devem crescer e avançar com menos certeza.
5. **Concept and Technology Development** – por fim, o elemento final do modelo envolve o desenvolvimento de um "business case" baseado em estimativas do potencial de mercado, necessidades do cliente, requisitos de investimento, avaliações de concorrentes, tecnologia desconhecida e risco geral do projeto.

Por fim, como terceiro e último elemento tem-se o aro, onde são incluídos os fatores ambientais que influenciam o motor e dão forma aos cinco elementos da atividade. Com isso se inclui a capacidade organizacional da empresa, ameaças da concorrência, clientes e tendências mundiais, mudanças regulatórias e a riqueza e força das ciências e tecnologia (Koen, Bertels, &

Kleinschmid, Managing the Front End of Innovation - Part I: Results From a Three-Year Study, 2014).

2.2.2 NCD aplicado à empresa

Uma vez percebido o New Concept Development Model, o projeto aqui relatado integra-se da seguinte forma com os elementos fundamentais do modelo:

- **Oportunity identification** – como referido anteriormente, a empresa encontra-se com diversos problemas. A fragmentação do seu sistema, a alocação de toda a sua informação apenas em servidores locais e a complexidade associada aos seus serviços fazem com que o sistema de monitorização não se encontre no nível desejado. Deste modo surge assim uma oportunidade para o melhorar.
- **Oportunity analysis** – uma vez identificada a oportunidade, é importante perceber se esta tem viabilidade para a empresa. Na conjuntura atual, a OEE tem sofrido um impacto negativo no seu negócio. A complexidade do sistema atual gera uma resistência por parte dos utilizadores, tornando-o pouco lucrativo e eficiente. Por isso, na perspetiva da empresa este projeto faz todo o sentido, uma vez que visa melhorar o sistema atual, gerando um produto final de maior valor que consequentemente trará um lucro maior.
- **Idea generation & enrichment** – assim que a oportunidade é definida e validada, passa-se então para a caracterização das ideias. Como o objetivo passa por aprimorar um produto já existente, é importante ter um conhecimento sobre o mesmo. Desse modo, realizou-se um estudo do sistema em vigor, de modo a o entender e perceber os problemas existentes. Posteriormente, analisaram-se diversas ferramentas e processos que enquadrados no sistema atual, os poderiam solucionar. Por fim, e já com uma ideia sobre as vantagens e desvantagens de cada uma das possíveis soluções, iniciou-se a preparação de um conjunto de processos e metodologias a serem aplicadas.
- **Idea selection** – depois da análise de diferentes soluções existentes no mercado, excluir-se-ão em conjunto com a empresa, todas aquelas que tiverem um custo elevado ou que forem incompatíveis com o sistema atual. Além disso, para a OEE é fundamental ter um controlo total de todos os seus processos e informação, pelo que a escolha de novas ferramentas e tecnologias é bastante seletiva. Desse modo, a preferência recai em softwares *open-source* de modo a salvaguardar a integridade, privacidade e

segurança de toda a informação. Relativamente às metodologias existentes, estas irão depender da solução escolhida.

- **Concept definition** – através da análise e melhoria do sistema atual surgem assim diversas vantagens tanto para a empresa como para o cliente. Com a correção dos problemas apontados anteriormente, obtém-se um produto final mais simples, eficiente e de maior valor. Deste modo, a OEE tem assim uma oportunidade para aumentar a sua produtividade, tanto do ponto de vista de processos como de negócio, através de um sistema de monitorização melhor e mais otimizado, e produtos de maior valor para oferecer ao cliente. Estas melhorias significarão não só a satisfação das empresas cliente face aos serviços prestados, como poderá resultar também na angariação de novos clientes.

2.2.3 Valor

Para medir o valor na prática, é essencial entender o que é valor do ponto de vista do negócio. O valor nos mercados financeiros corresponde ao valor em termos monetários dos benefícios técnicos, económicos, de serviço e sociais que os clientes obtêm em troca pelo preço que pagam. Assim, aumentar ou diminuir o preço de uma oferta de mercado não altera o valor que a oferta tem para o cliente. As perceções de valor ocorrem dentro de um determinado contexto. Mesmo quando não existem ofertas de mercado comparáveis, há sempre uma alternativa competitiva (Anderson & Naru, 1998).

Assim, podemos capturar a essência desta definição de valor a partir da seguinte equação:

$$(Valor_{fornecedor} - \text{Preço}_{fornecedor}) > (\text{Valor}_{\text{alternativa}} - \text{Preço}_{\text{alternativa}})$$

Avaliando os possíveis resultados do trabalho aqui descrito, encontram-se vários benefícios que oferecem valor tanto para o cliente como para a empresa. Com a implementação de um novo sistema de monitorização em tempo-real espera-se que os problemas do sistema atual sejam solucionados e se obtenha um serviço mais eficiente, mais rápido, satisfazendo melhor as necessidades dos clientes. No entanto, para usufruir do mesmo pode existir um custo associado que a empresa tem de suportar. Nesse custo pode ser incluído o investimento num novo software de monitorização e uma reformulação, em parte, das soluções existentes. Além disso, poderá haver custos associados ao processo de aprendizagem do novo software por parte dos trabalhadores, que irão variar consoante a complexidade do mesmo.

Pese embora o custo associado, é percetível que o novo sistema irá oferecer um maior valor para os clientes. Uma vez que o seu desenvolvimento vem corrigir um problema por eles apontado, esperando-se que traga assim um reforço da sua confiança associado à sua satisfação com estas intervenções. Consequentemente, trará também um maior valor para a empresa, não só porque haverá uma melhoria no produto já existente, mas também porque isso se refletirá num maior lucro para a mesma.

2.2.4 Valor para o cliente

As perceções que o cliente tem sobre a qualidade da oferta, interação pessoal, suporte de serviço e "know-how" do provedor são os principais benefícios que afetam positivamente o valor para o cliente. Por outro lado, as perceções relativas aos custos diretos, de aquisição e de operação são os sacrifícios essenciais que afetam negativamente o valor para o cliente (P. Blocke, 2011).

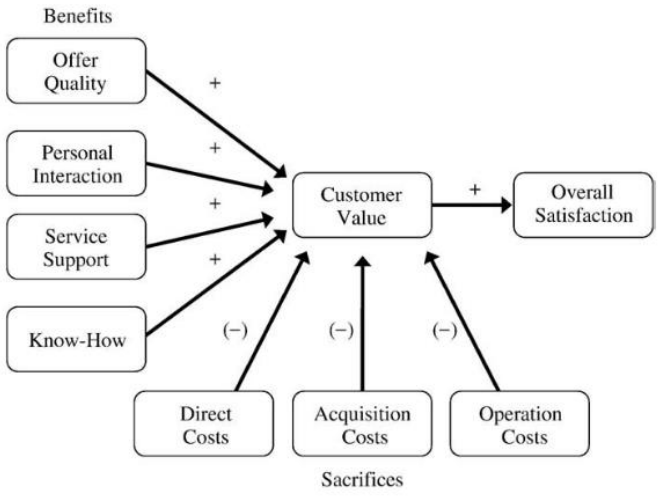

Figura 2 - Estrutura conceitual para medição do valor do cliente e dos indutores de valor nas relações de negócios (P. Blocke, 2011).

Com o novo sistema de monitorização em tempo-real, os clientes obtêm um serviço melhorado que trará maiores vantagens para os seus negócios. Os fluxos de trabalho tornam-se muito mais fluídos, tornando assim a experiência para os clientes muito mais agradável, estável e com menos falhas. Estes passam a ter acesso remotamente aos dados monitorizados em tempo-real sem precisar de se deslocar à empresa. Por outro lado, espera-se que o novo software seja mais simples e fácil de ser integrado e percebido por parte dos clientes.

Espera-se por isso que este impacto se traduza em confiança, sentimento de segurança e aumento do rendimento, oferecendo assim um maior valor para os seus negócios.

2.2.5 Valor percecionado

O conceito de valor percecionado possui diferentes interpretações dentro do contexto de marketing, que divergem entre diferentes autores.

Segundo (KOTLER, 1998), o valor percecionado é atribuído pelos clientes ao produto ou serviço, baseado na relação entre os benefícios previstos, segundo a perspetiva do consumidor, e os custos percebidos para sua aquisição comparativamente à concorrência.

Já para (ZEITHAML, 1988), consiste na avaliação total do consumidor sobre a utilidade de um produto, sustentada por perceções do que é recebido (benefícios) e do que é dado (sacrifícios).

Por outro lado, (WOODRUFF, 1997) adota o conceito de valor para o cliente como sendo a perceção do cliente sobre as preferências e as avaliações dos atributos do produto, do desempenho desses atributos e das consequências originadas pelo uso.

No contexto da OEE Technologies, o valor percecionado por parte de cada cliente em relação aos serviços prestados pela empresa varia consoante as suas necessidades, orçamento, tipo de negócio, entre outros fatores descritos anteriormente. No entanto, é certo que a compreensão do conceito de valor percebido e do seu papel como grande impulsionador da lealdade dos clientes influencia a lucratividade da empresa.

2.2.6 Proposta de Valor

A proposta de valor é um elemento do marketing responsável por destacar um negócio, posicionando-o para o seu público-alvo como melhor que a concorrência. Para o cliente entender realmente o serviço que pondera adquirir, é necessário esclarecer todas as questões pertinentes. Dependendo do produto, existem questões que surgem naturalmente e que são impreteríveis para o cliente na compreensão da finalidade do produto.

A proposta de valor é uma prática oriunda do Marketing, com o objetivo de dar ao cliente uma ideia clara, concisa e transparente de como um determinado negócio pode ser relevante para ele (Redator Rock Content, 2020).

Deste modo, este projeto propõe-se a ser a solução dos atuais problemas do sistema de monitorização da OEE Technologies. Para atingir este objetivo, é necessário identificar as

melhores soluções para resolver os problemas, bem como a melhor maneira de as integrar na empresa com a maior viabilidade financeira.

Para isso, tem de se analisar o sistema de forma aprofundada de modo a encontrar a melhor solução. Uma vez encontrada a solução, é uma questão de a desenvolver e implementar de modo a melhorar o sistema de monitorização da empresa.

A partir daí pode-se identificar diversas vantagens para os clientes. Por um lado, os clientes que já se encontravam satisfeitos com o sistema em vigor, vão sentir uma melhoria acentuada do mesmo. Esta melhoria reforça assim a sua satisfação não só para com o serviço, como também consequentemente para com a empresa.

Por outro lado, os clientes que apontaram alguns problemas, observam assim um interesse por parte da empresa em melhorar a sua experiência e o valor do seu produto.

No geral, é proposto um produto de maior valor que pode sobressair-se relativamente a outros competidores no mesmo mercado, garantindo também uma maior satisfação, confiança e lealdade dos clientes.

2.2.7 Canvas

O Canvas possibilita a análise de modelos sustentados de negócios através de um modelo composto por termos elementares como recursos-chave, clientes-alvo e canais de receita (Muhtaroglu, Demir, Obalı, & Girgin, 2013).

Desse modo, construiu-se o Business Model Canvas (BMC), representado na Figura 3, que contém os nove elementos que estruturam o conteúdo do negócio (Keane, Cormican, & Sheahan, 2018) da OEE.

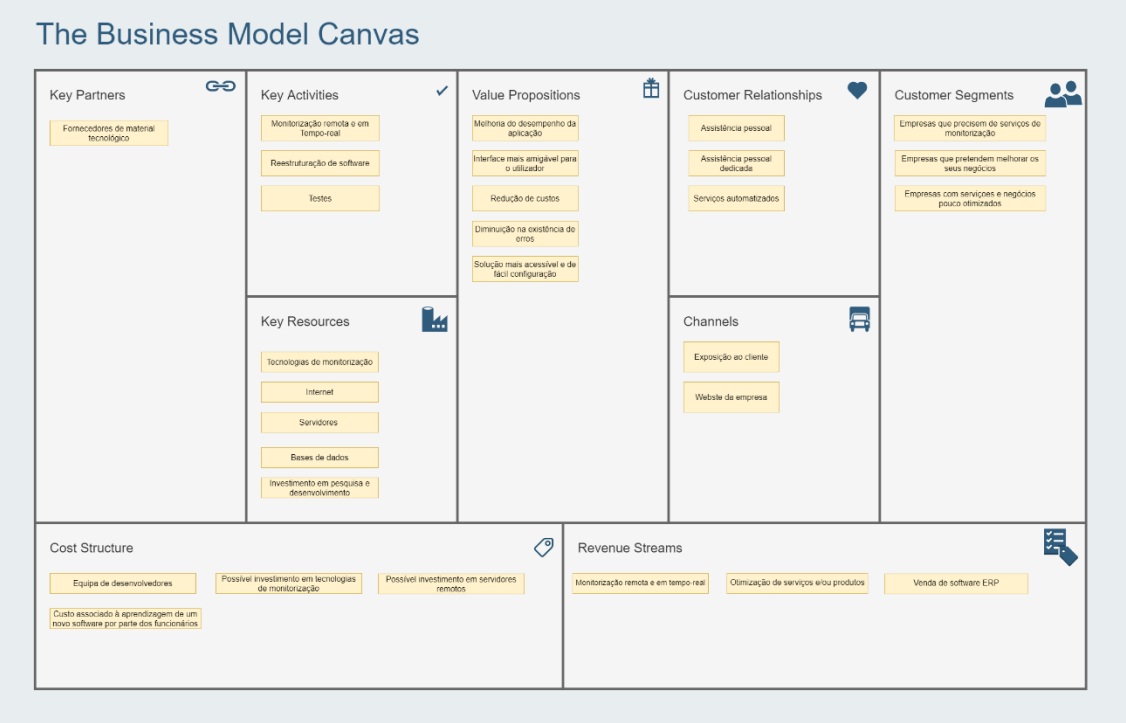

Figura 3 - Canvas Business Model

A partir da Figura 3, compreende-se assim o modelo de negócio da solução. É possível entender que a mesma pretende oferecer valor aos clientes através de uma melhoria do sistema atual, tornando o mesmo mais eficiente. Com isso, são esperados alguns custos, nomeadamente com a integração de novas tecnologias e com a aprendizagem das mesmas por parte dos desenvolvedores. É visível também que para a sua conceção estão envolvidos alguns parceiros chave para a empresa que disponibilizam o software necessário.

Por fim, para a entrega da solução definiram-se os canais de entrega bem como os clientes com interesse na solução. Deste modo, definiu-se que o produto será entregue a clientes da área da indústria, saúde e serviços através da plataforma web da empresa e através da exposição dos mesmos presencialmente. Em caso de problemas na sua implementação, serão disponibilizados também serviços de assistência que dependendo da necessidade podem ser feitos de forma automatizada ou pessoal.

2.3 Tecnologias usadas pela empresa

Nesta secção, encontram-se descritas as tecnologias usadas pela empresa OEE Technologies de modo a compreender melhor a sua arquitetura que é descrita em detalhe na secção 2.4.

2.3.1 Interface serial RS422

A interface RS422, funciona como um meio de comunicação ponto a ponto para conectar múltiplos sensores. Permite a conexão até um máximo de 10 recetores para um transmissor a uma distância máxima de 1200 metros (IPC2U).

2.3.2 Interface serial RS485

O RS485 é uma interface física entre o controlador local e os sensores. É uma interface simples e funciona como um meio de comunicação multiponto que permite conectar múltiplos recetores e transmissores de sinais. Através da mesma, é possível conectar até um máximo de 32 nós, a uma distância que pode chegar aos 1200 metros, podendo este número ser estendido através do uso de repetidores de sinal.

É importante referir também que todos os sensores que usem esta interface, terão parâmetros de comunicação em série iguais, como a taxa de transmissão, paridade, bits de início e parada. No entanto, são permitidos parâmetros diferentes de comunicação para diferentes sensores (Cisco, 2016).

2.3.3 Ethernet

De acordo com o Instituto de Engenheiros Eletricistas e Eletrotécnicos (IEEE), a Ethernet é definida como sendo um protocolo 802.3, servindo de base para a comunicação entre computadores e dispositivos através do uso do protocolo TCP/IP. Além disso, é a base sobre a qual muitas redes locais (LAN's) são construídas, possibilitando uma troca de dados segura e confiável entre computadores (Weis, 2020).

2.3.4 TCP/IP

O TCP/IP (Transmission Control Protocol/ Internet Protocol) representa um conjunto de protocolos de comunicação padrão da Internet que permitem que diversos equipamentos comuniquem entre si.

A Internet é uma rede comutada por pacotes, na qual as informações são divididas em pequenas partes, enviadas individualmente e em simultâneo por diferentes rotas, sendo posteriormente reconstruídas na extremidade recetora. Deste modo, o TCP é o componente que captura e reconstrói os pacotes de dados, enquanto o IP é responsável por garantir que os mesmos sejam enviados para o destino correto (Britannica, 2019).

Assim, a estrutura deste protocolo divide-se em cinco camadas: física, de acesso à rede, internet, transporte e aplicação. Cada uma destas camadas possui uma função específica e utiliza e presta serviços às camadas adjacentes através do uso de diferentes protocolos (FEUP).

2.3.5 Rede local LAN

Uma LAN é uma rede local que permite a comunicação entre computadores dentro de um mesmo espaço físico, conectados por um router. Este tipo de rede é bastante usado em empresas, escolas ou dentro de casa, permitindo a troca de informações e recursos entre diferentes dispositivos (Digital).

2.3.6 SSH

O SSH é um protocolo de rede que permite gerir remotamente os servidores através da internet. Através deste protocolo, o utilizador pode autenticar-se noutro computador através de uma rede protegida por criptografia. Desse modo, os dados, informações, documentos e arquivos são acedidos pelo utilizador através de uma comunicação criptografada entre o seu computador e os servidores que armazenam a informação (Andrey, 2019).

2.3.7 Socket

Um *Socket* é um ponto de destino de uma comunicação bidirecional entre dois programas em execução numa rede. Para o seu funcionamento, este é conectado a um endereço IP e a uma porta de acesso para que a camada TCP possa identificar a aplicação para a qual os dados são

enviados. Além disso, possui classes que são usadas para representar a conexão entre um programa cliente e um programa servidor. Através de um pacote java.net, são fornecidas duas classes, *Socket* e *ServerSocket*, que implementam o lado do cliente e o lado do servidor da conexão, respetivamente (Oracle, Lesson: All About Sockets, 2020).

2.3.8 Sistema Operativo Debian

O Debian é um sistema operacional *open-source* baseado em Linux, grátis, usado numa ampla variedade de dispositivos, incluindo laptops, desktops e servidores. Possui um amplo suporte de hardware, atualizações fáceis e suaves, e apresenta um software estável e robusto (Debian, 2021).

No contexto da empresa, Debian é o sistema operativo em vigor, sendo usado nos servidores da mesma e acessível localmente, a partir de outros computadores via LAN, ou remotamente via SSH.

2.3.9 Tomcat

O Tomcat atua como um servidor Web e um contentor de Servlet (classe java usada para estender os recursos dos servidores (Oracle, The Java EE 5 Tutorial, 2010)). O seu propósito passa por compilar aplicações em produção que processam inúmeros pedidos. Uma vez que providencia um ambiente de servidor Web HTTP para execução de código Java, assume assim um comportamento de servidor de teste onde os desenvolvedores podem gerir as suas aplicações em produção e testar o seu funcionamento (Souza, 2020).

2.3.10 MySQL

O MySQL é um sistema de gestão de bases de dados (SGBD) relacional, que utiliza uma linguagem baseada em consultas, denominada em inglês de structured query language (SQL). Também conhecida por ser multiutilizador e multitarefas, é a base de dados mais utilizada globalmente.

Este SGBD é totalmente *open-source* pelo que pode ser editado por qualquer utilizador de forma a atender as suas necessidades. Com suporte para aplicações que trabalham com grandes volumes de dados, possui ainda uma interface extremamente simples e compatível

com a maior parte dos sistemas operacionais. Além disso, apresenta diferentes tipos de tabela para o armazenamento dos dados, tendo em conta que cada tipo tem as suas próprias prioridades (velocidade, volume de dados, entre outros) (Teixeira, 2013).

Todos os dados da empresa encontram-se armazenados em bases de dados MySQL (MariaDB), podendo ser consultados através de um software cliente denominado HeidiSQL.

2.3.11 MariaDB

MariaDB é um software gratuito e open source de bases de dados relacionais, criado pelos mesmos desenvolvedores do MySQL, que fornece uma interface SQL para acesso e manipulação dos dados.

Este software transforma diferentes tipos de dados em informações estruturadas para uma vasta gama de aplicações. Além disso, é rápido, escalável e robusto, possuindo um ecossistema de mecanismos de armazenamento, plug-ins e outras ferramentas que o tornam muito versátil para uma ampla variedade de casos de uso (MariaDB, 2021).

2.3.12 HeidiSQL

O HeidiSQL é um software grátis e *open-source* que permite aceder e editar dados de diferentes bases de dados, tais como, MariaDB, MySQL, Microsoft SQL, PostgreSQL e SQLite. Trata-se por isso de um software cliente, com uma interface que permite a conexão com múltiplos servidores e bases de dados. Além disso, permite a criação e edição de tabelas, execução de triggers, views, entre outros (Becker, 2021).

2.3.13 Jaspersoft Studio

O Jaspersoft Studio é um software *open-source*, desenvolvido em Java, que permite a criação de relatórios em diferentes formatos, tais como, PDF, HTML, XLS, CSV e XML. Deste modo, é possível projetar e executar diferentes modelos, construir relatórios de consultas e escrever expressões complexas (Lanhellas, 2014). Com mais de 50 tipos de gráficos, mapas, tabelas, crosstabs e visualizações personalizadas, é possível também personalizar visualmente o layout do relatório (Jasper, 2021).

Aquando da criação dos relatórios, são gerados um ficheiro Jasper e outro XML. O primeiro contém os arquivos dos relatórios enquanto o segundo regista as dimensões, campos e características do mesmo.

Este software pode ainda ser integrado com diferentes bases de dados, como por exemplo MySQL, através de diferentes frameworks (Lanhellas, 2014).

2.3.14 Java

O Java é uma linguagem de programação baseada em classes e orientada a objetos, criada e comercializada pela Sun Microsystems desde 1995.

Derivada das linguagens C e C++, mas mais simples e fácil de usar, esta utiliza um modelo de programação orientado a objetos permitindo vincular os diferentes tipos de dados às suas operações.

Além disso, possui uma vasta gama de bibliotecas sendo usada para construir uma ampla variedade de aplicações (Code Institute, 2021).

2.3.15 Tecnologias de Suporte

Nesta secção serão mencionadas as diferentes tecnologias usadas de suporte ao desenvolvimento do projeto.

2.3.15.1 GIT

O Git é um sistema de controle de versões gratuito e *open-source* projetado para lidar com todo o tipo projetos com velocidade e eficiência. É uma das ferramentas mais usadas na gestão e controlo de versões em projetos, uma vez que permite manter um histórico das alterações que vão sendo realizadas. Além disso permite que vários desenvolvedores possam editar em simultâneo o mesmo projeto em dispositivos diferentes, bem como a criação de ramos (branches) e reversão de código.

A sua principal vantagem passa pelo facto de cada programador ter acesso a um repositório próprio local que pode, ou não, ser sincronizado com o repositório central/remoto, a fim de o modificar as vezes que quiserem (Git, 2021).

2.3.15.2 Gitlab

O Gitlab é uma plataforma de armazenamento de código-fonte através da qual os desenvolvedores contribuem em projetos privados ou públicos, os denominados open-source). A cada projeto que contém o código-fonte é dado o nome de repositório.

Além disso, esta plataforma faculta diversos recursos para o armazenamento do código fonte, tais como *pull request*, revisão de código, edição em linha, *forks* e clonagem de repositórios, e integrações com ferramentas de terceiros. Já para o controlo de versões, é utilizada a tecnologia GIT (Bertola, 2019).

2.4 Arquitetura atual da empresa

Como foi detalhado em capítulos anteriores, a OEE Technologies é uma empresa de consultoria que fornece soluções ERP a clientes de diferentes setores.

De forma a prestar os seus serviços, inicialmente é fundamental recolher toda a informação necessária relativamente ao negócio do cliente. De seguida, esses dados são enviados para um servidor, sendo posteriormente armazenados em bases de dados. Numa última etapa, esta informação é recolhida e tratada através do software da empresa, sendo criados relatórios que são enviados para o cliente.

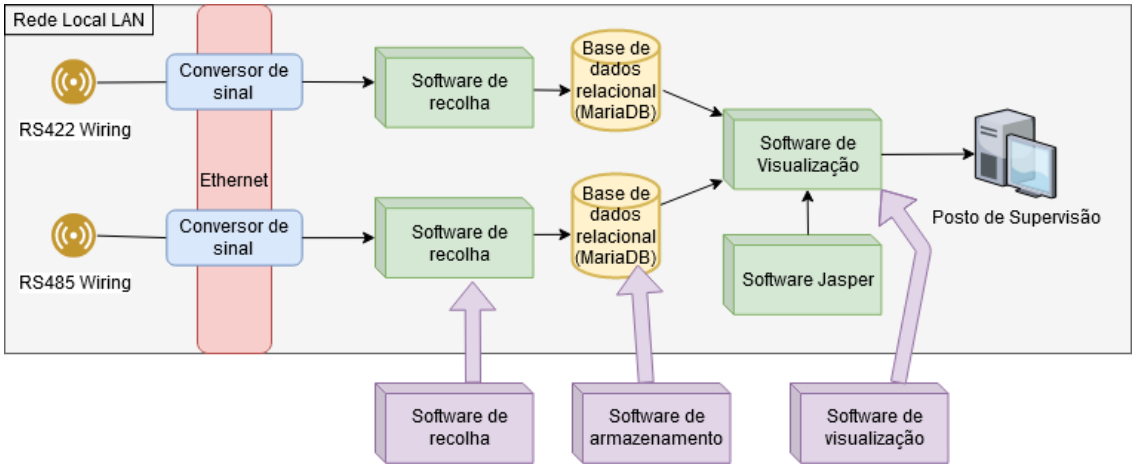

Figura 4 - Arquitetura do Sistema

Como é possível observar na Figura 4, a informação captada pelos sensores é enviada para aparelhos com interfaces em série RS422 e RS485 que estendem a rede através da Ethernet, eliminando assim a limitação do comprimento dos cabos na comunicação.

Em seguida, os dados são enviados através de *Sockets* para o software de captura da empresa. Por sua vez, este software envia os dados recolhidos para duas diferentes bases de dados MariaDB. Uma armazena os dados provenientes dos sensores conectados via RS422 enquanto a outra armazena os dados provenientes dos sensores conectados via RS485.

Consequentemente, a informação armazenada nas bases de dados é recolhida e tratada pelo software de visualização da empresa, escrito maioritariamente em Java, sendo responsável pelos cálculos OEE. Este encontra-se conectado às bases de dados através de uma API denominada por *Java Database Connectivity* (JDBC), que possui um conjunto de classes e interfaces escritas em Java que permitem a conexão.

A partir dos dados recolhidos, o software permite a criação de relatórios com a análise de toda a informação, através de gráficos, estatísticas e métricas OEE. Para esse efeito, este encontra-se integrado com o software Jasper que permite desenhar, criar e exportar relatórios com toda a informação relevante para o cliente, de modo que o utilizador possa analisar o seu negócio.

2.5 Estado da arte em soluções/abordagens existentes

Esta secção apresenta o estado da arte em soluções e abordagens existentes no contexto do problema. Nesse sentido, inicialmente são explicados conceitos importantes do negócio da empresa de modo a introduzir e apresentar as soluções encontradas.

2.5.1 ERP

Como referido em capítulos anteriores, a OEE Technologies é uma consultora que fornece soluções ERP a outras empresas.

O *Enterprice Resource Planning* (ERP) é um software de gestão que incorpora diferentes técnicas e conceitos que fazem parte da gestão de uma organização como um todo, com o intuito de otimizar a eficiência da empresa. Este tipo de software utilizado globalmente, é

produzido de modo a modular e automatizar os processos básicos de uma empresa, integrando e gerindo a informação das várias áreas organizacionais (Leon, 2007).

De acordo com um estudo recente da S&R, a Global ERP Solutions Market Insights, Forecast to 2025, o histórico de dados usados para a previsão do mercado de ERP apontam para um crescimento significativo do segmento até 2026 (Content, 2020).

Dentro dos fatores que impulsionam o desenvolvimento do mercado de ERP globalmente, incluem-se a necessidade das empresas em otimizar os seus processos e a obtenção de uma maior eficiência operacional de baixo custo (Content, 2020).

Neste sentido, de acordo com a (Microsoft) este software pode abranger diferentes área de negócio, nas quais se destacam:

- **Finanças** - através da oferta de soluções que permitem consultar dados financeiros em tempo real a qualquer momento e em qualquer local, automatizando as tarefas diárias e monitorizando o negócio.
- **Recursos humanos** - a partir de uma solução que possibilite a gestão dos dados da empresa e otimização da gestão de colaboradores, monitorização do seu desempenho e identificação dos problemas de recursos humanos antes que estes ocorram.
- **Fabrico** - melhoramento da comunicação na empresa, automatização de processos diários, satisfação das necessidades dos clientes e gestão dos recursos através do acesso a dados em tempo real. Esta solução otimiza ainda a gestão de projetos e custos, assim como o planeamento da produção.
- **Cadeia de fornecimento** - proporciona *dashboards* e *business intelligence* que permitem a gestão e controlo do inventário.

Conclui-se, portanto, que um ERP é um software que traz diversas vantagens para uma organização, como o aumento da eficácia e performance dos processos da empresa e um fácil acesso à informação, melhorando a gestão da mesma.

2.5.2 Monitorização

No seguimento das soluções ERP, encontra-se a monitorização. Do ponto de vista tecnológico, a monitorização passa por um conjunto de ferramentas e processos que permitem medir e gerir sistemas tecnológicos, gerando um maior valor para o negócio. Os dados gerados por um dado

sistema e/ou aplicações são assim traduzidos em informação útil podendo ser avaliados pelo utilizador. Deste modo, é possível garantir que uma empresa entrega o que os clientes desejam. Além disso, estes dados também fornecem informação tecnológica relevante, permitindo avaliar o funcionando e a qualidade dos serviços de uma empresa (Turnbull, 2018).

Embora os sistemas de planeamento de recursos empresariais possam controlar alguns procedimentos, a utilização do ERP para a monitorização de tarefas simplifica processos dentro da organização (Blink).

Através da monitorização dos processos é possível obter uma maior eficiência, redução de custos desnecessários, diminuição de erros na produção e favorecimento da lucratividade da empresa. Um dos aspetos essenciais da monitorização na indústria 4.0 é o fornecimento de informações precisas sobre todos os processos e produtos com falhas. Desse modo, problemas relacionados com a existência de itens danificados, escassez de matéria-prima ou outro tipo de problemas da fábrica são detetados mais facilmente, aumentando assim a qualidade dos produtos.

Por outro lado, a gestão eficiente e a monitorização de trabalhadores, máquinas e processos de trabalho permitem detetar falhas atempadamente de modo a agir mais rapidamente.

Com isso, a empresa torna-se assim mais eficiente e lucrativa, conseguindo reduzir os seus custos com uma gestão mais eficiente, com menos quebras de equipamentos, entre outros fatores que interfiram nos custos (Vedois, 2018).

2.5.3 Métricas

As métricas representam medidas do uso de recursos ou procedimentos que podem ser observados e captados num sistema. O seu uso como fonte de informação num sistema de monitorização, permite analisar diferentes ambientes e obter informações úteis para a empresa.

Assim, as métricas podem-se classificar em duas categorias: as de baixo nível e as de alto nível. As primeiras são geralmente provenientes do sistema operativo, enquanto as segundas são oriundas de aplicações executadas num dado elemento. Algumas podem ser apresentadas como uma relação entre o valor aproveitado face à capacidade total, enquanto outras representam uma média indicativa do estado de ocupação de um componente.

Geralmente, as que se encontram naturalmente expostas pelo sistema operacional e representam a utilização de recursos físicos são mais fáceis de captar, como por exemplo a utilização do disco, do CPU e da memória (Ellingwood, 2017).

Já as métricas de alto nível dependem das aplicações com as quais se relacionam. Por exemplo, os dados estatísticos referentes ao número de requisições de servidores ou ao número de *queries* de uma base de dados, dependem de uma interface disponível para a captura de dados. A exposição desses dados é designada por instrumentação (ÖZMEN, 2009).

É fundamental que aquando da captura dos dados seja registado o momento da sua ocorrência, uma vez que o tempo é a chave principal para operações de agregação e correlação. Por um lado, a agregação possibilita uma visão geral do sistema em diferentes granularidades, enquanto a correlação permite relacionar métricas e identificar padrões.

2.5.4 OEE – Eficiência Global dos Equipamentos

Na década de 1960, Seiichi Nakajima propôs um conjunto de métricas denominadas *Overall equipment efficiency* (OEE) que medem a eficiência global dos equipamentos. Tal como o nome indica, serve para medir a eficiência dos equipamentos numa indústria, identificando a parcela de tempo de produção que é realmente produtiva. Esta ferramenta foi desenvolvida para oferecer uma estrutura quantitativa à metodologia Manutenção Produtiva Total (TPM). Assim o objetivo passa por melhorar o processo produtivo, eliminando perdas ou reduzindo-as ao máximo (Salomão).

Deste modo, segundo (OEE, What is Overall Equipment Effectiveness?, 2020), este conjunto de métricas é o produto de 3 fatores:

- **Disponibilidade** - calcula as perdas geradas pela inoperatividade do equipamento e respetivas causas, nomeadamente a falta de trabalho, defeitos mecânicos ou elétricos, problemas com materiais, pessoas ou processos, entre outras.
- **Desempenho** – mede o impacto causado no negócio pela ineficiência da produção.
- **Qualidade** - avalia o prejuízo gerado pela produção de materiais com defeitos que não podem ser entregues ao cliente, ou que já foram vendidos.

Tendo em conta a métrica do processo TPM, o programa original da OEE tem como objetivo a redução de seis grandes perdas:

Tabela 1 - 6 grandes perdas e os indicadores OEE (OEE, Six Big Losses)

Overall Equipment Effectiveness	Recommended Six Big Losses	Traditional Six Big Losses
Availability Loss	Unplanned Stops	Equipment Failure
	Planned Stops	Setup and Adjustments
Performance Loss	Small Stops	Idling and Minor Stops
	Slow Cycles	Reduced Speed
Quality Loss	Production Rejects	Process Defects
	Startup Rejects	Reduced Yield
OEE	Fully Productive Time	Valuable Operating Time

Com base na Tabela 1 pode-se concluir que quando há falhas nos indicadores de disponibilidade, performance ou qualidade são gerados diferentes tipos de perdas. Todavia, através do uso das métricas OEE essas perdas são diminuídas, conseguindo-se assim um aumento da produtividade (OEE, Six Big Losses).

2.5.5 Cálculo das métricas OEE

Para o cálculo do OEE deve-se apenas considerar o tempo que é responsabilidade da equipa de produção, ou seja, o que corresponde ao horário normal de trabalho. Deste modo, tem-se o tempo que a equipa de produção precisa para produzir um determinado produto, sendo esta a base do cálculo do OEE (OEE, Como calcular o OEE?).

Para melhor compreensão deste processo, é preciso perceber como se calculam os diferentes fatores que compõem, pelo que seguida será demonstrado de que maneira cada fator é calculado.

2.5.5.1 Disponibilidade

Corresponde ao tempo em que o equipamento esteve a produzir face ao tempo total disponível para produção. Assim, há dois termos a ter em conta:

1. **Tempo de produção**: tempo em que a máquina se encontra a produzir;
2. **Tempo planeado**: tempo programado para produzir.

Com a relação entre estes termos, obtém-se a seguinte fórmula (Novida):

$$Disponibilidade = Tempo\ em\ produção / Tempo\ planeado$$

Como se pode observar, a disponibilidade está diretamente relacionada com a produção. Por isso, quanto menor for a disponibilidade, maior é o tempo em que o equipamento ficou parado. Por outro lado, se a disponibilidade for elevada, significa que o equipamento teve poucas paragens durante o tempo de produção.

2.5.5.2 Desempenho

Define a relação entre a velocidade expectável de produção e a velocidade efetiva da operação diária das máquinas. Ou seja, esta métrica mede a performance dos equipamentos.

Deste modo, há dois elementos a considerar:

1. **Tempo padrão**: é aquele em que o equipamento foi projetado para funcionar;
2. **Número de peças produzidas**: é o número de peças produzidas num dado intervalo de tempo;
3. **Tempo efetivo**: duração da realização de uma tarefa, considerando o tempo de produção e descartando os momentos de paragem.

Assim, a relação entre os termos traduz-se da seguinte forma (Novida):

$$Desempenho = tempo\ padrão * nº\ de\ peças\ /\ tempo\ efetivo$$

Assim, quanto menor for o tempo necessário para a realização de uma tarefa, maior será o desempenho. Conclui-se por isso que o tempo efetivo determinará o desempenho final da produção.

2.5.5.3 Qualidade

Qualquer empresa deve prezar pela qualidade dos seus processos e produtos, salvaguardando os padrões de qualidade pré-estabelecidos de modo a cumprir o compromisso para com os seus objetivos.

Desta forma, tem de se ter em consideração o número de produtos produzidos, quantos tiveram de ser corrigidos e quantos foram perdidos.

O cálculo da qualidade é assim realizado (Novida):

$$Qualidade = quantidade\ de\ produtos\ produzidos - (quantidade\ corrigida + quantidade\ perdida)\ /\ quantidade\ de\ produtos\ produzidos$$

Com base na análise à fórmula descrita em cima, é fácil concluir que a qualidade será tanto maior quanto menor for a quantidade de produtos defeituosos produzidos e o número de produtos perdidos.

2.5.5.4 OEE

Por fim, a partir do momento que se têm os fatores de disponibilidade, performance e qualidade, pode-se calcular o OEE. Para isso, basta realizar o seguinte produto (Novida):

$$OEE\ (\%) = Disponibilidade\ (\%) * Performance\ (\%) * Qualidade\ (\%)$$

Em suma, quanto maior for a disponibilidade, performance e qualidade, maior será o valor OEE, obtendo-se assim uma alta eficiência na produção.

2.5.6 Tipos de ferramentas

Após um estudo sobre o sistema da empresa e a definição das métricas a usar, foram identificadas três áreas de estudo: captura, armazenamento e visualização.

A partir dessas áreas podem-se definir três tipos de ferramentas:

- **Ferramentas de captura**: através de módulos de software capazes de captar as métricas provenientes dos sensores usados para a monitorização;
- **Ferramentas de armazenamento**: através de bases de dados que possibilitam o armazenamento de dados, de modo que fiquem classificados em função do tempo da recolha;
- **Ferramentas de visualização**: a partir de módulos de software que permitam a visualização das métricas e criação de gráficos e relatórios de forma remota, além de permitirem a configuração de alertas.

De forma a atingir os objetivos propostos por este projeto, foi realizado um levantamento de ferramentas disponíveis que se enquadram nas três categorias acima referidas. Para isso, foram formulados critérios para cada tipo de ferramenta, que são detalhados secção seguinte.

2.5.7 Critérios

De modo a fazer o levantamento do estado da arte, foram definidos previamente em conjunto com a empresa alguns critérios a aplicar na procura por diferentes soluções. Deste modo, foram tidas em consideração as funcionalidades de cada um dos módulos de software, os requisitos do sistema e o contexto em que a implementação será feita.

Para a sua concretização efetuou-se de novo a divisão em três tipos: captura, armazenamento e visualização.

De realçar que o levantamento do estado da arte para o módulo de software para captura foi realizado numa perspetiva de estudo para implementação futura, pelo que a solução escolhida não será implementada neste projeto.

2.5.7.1 Módulo de software para captura

O módulo de software responsável pela captura dos dados provenientes dos sensores deve ser:

- *Open-source*;
- Gratuito ou de baixo custo;
- Compatível com Linux;
- Possibilidade de captura das métricas selecionadas, seja nativamente ou por meio de plugins oficiais.
- Envio de métricas para ambientes externos (outra máquina), nativamente ou através de plugins oficiais.
- Compatibilidade com a ferramenta de armazenamento, de forma a evitar a necessidade de implementações para transporte e transformação de dados, o que pode prejudicar significativamente o desempenho da integração;
- Baixo número de dependências, de forma a facilitar a sua instalação e configuração.

2.5.7.2 Módulo de software para armazenamento

Por sua vez, o módulo para armazenamento deve possuir as seguintes características:

- *Open-source*;
- Gratuito ou de baixo custo;
- Compatível com Linux;
- Compatível com o Java;

- Dependências;
- Fácil de usar e com uma linguagem de consultas baseada em SQL;
- Conectividade a partir de ambientes externos à máquina hospedeira.

2.5.7.3 Módulo de software para visualização e análise

Por fim, o módulo para visualização deve atender os requisitos abaixo:

- *Open-source*;
- Gratuito ou de baixo custo;
- Compatível com diferentes sistemas operativos;
- Possibilidade de ser usado através de diferentes navegadores, a partir de qualquer plataforma e de forma remota;
- Permita visualizar as métricas em tempo real;
- Compatibilidade com o módulo de software que fornece os dados, evitando a necessidade de desenvolvimento de scripts para o transporte e transformação de dados, o que pode impactar negativamente no desempenho da integração.

Através do cumprimento destes critérios, prevê-se uma maior efetividade na pesquisa por módulos de software que possam ser incorporados na empresa.

2.5.8 Soluções encontradas

Com base nos requisitos definidos na secção anterior, fez-se então o levantamento das tecnologias por cada tipo.

2.5.8.1 Módulos de software para captura

Como foi visto anteriormente, estes módulos são responsáveis pela recolha dos dados provenientes dos sensores. As secções seguintes descrevem assim diferentes módulos encontrados para este propósito.

2.5.8.1.1 Collectd

O Collectd é um software Unix *daemon*, criado por Florian Forster em 2015, para recolha de estatísticas sobre o desempenho de um sistema ou de uma aplicação. Sendo um software daemon, este é executado em segundo plano, reunindo as principais métricas do sistema. A partir destas métricas, podem ser produzidas valiosas visualizações que fornecem informações sobre os problemas de um dado sistema (Logic, 2020).

Deste modo, segundo a (Logic, 2020) podem-se apontar as seguintes vantagens:

- ***open-source*** – sendo um software de código aberto e com uma presença sólida no mercado de captura de métricas do sistema, é de uso gratuito e está em constante desenvolvimento.
- **alto nível de portabilidade** – este software é orientado por plug-ins, não tendo dependências externas, de modo que pode ser executado em diferentes sistemas operacionais.
- **extensível** - o Collectd é altamente extensível, possuindo diferentes plugins com suporte para várias linguagens (como C, Perl, Java e Python), que permitem que as funcionalidades sejam aprimoradas para atender às necessidades dos utilizadores.
- **escalável** – possui a capacidade de monitorizar vários *hosts*, com uma gestão de recursos eficiente.

É também importante referir que o *daemon* só implementa a infraestrutura para filtragem e tratamento de dados, bem como algumas funções auxiliares. A captura, armazenamento e envio dos dados é desempenhada por plugins nativos, que podem ser geridos de acordo com a necessidade do utilizador, obedecendo a um intervalo configurado (Forster & Harl, 2021). Assim, e de acordo com (Forster & Harl, 2021), esses plugins podem classificar-se em 3 categorias:

- **Plugins do sistema operacional**: captam as métricas a nível do sistema operacional e não são multiplataforma;
- **Plugins de aplicações**: captam métricas de servidores Apache, Nginx, MySQL ou MariaDB, são multiplataforma, mas exigem alguma configuração do software;
- **Plugins para tarefas específicas**: executam scripts ou plugins de rastreio de pacotes SNMP na rede.

Já os plugins de escrita possuem duas classificações distintas:

- **Plugins de armazenamento**: armazenam dados capturados na própria máquina;
- **Plugins de envio**: armazenam os dados capturados, através da rede, em bases de dados PostgreSQL, MongoDB, entre outras. Relativamente à compatibilidade, com os plugins

de envio é possível enviar dados para ambientes externos como InfluxDB, Graphite e Prometheus.

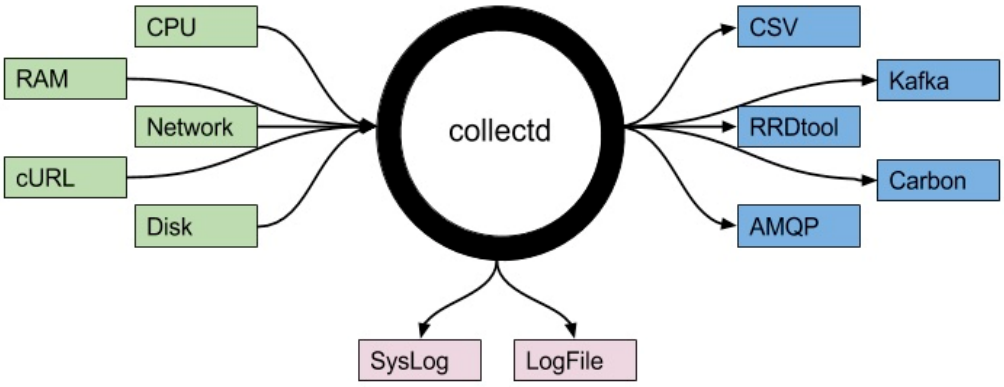

Figura 5 - Sistema de plugins do software Collectd (Pav, 2016)

Deste modo, é possível observar a partir da Figura 5 alguns dos plugins de captura e de armazenamento deste software. De realçar que apesar de não estar representado na figura, o Collectd possui ainda plugins para captura de informação de sensores, que no contexto da empresa pode ser importante.

Relativamente à instalação, é simples, podendo ser realizada através de gestores de pacotes dos sistemas operacionais, ou através do download e compilação do projeto a partir do GitHub (Forster & Harl, 2021). O mesmo se aplica aos plugins, pese embora o facto de alguns precisarem da instalação de algumas bibliotecas, que podem ser consultadas no site do desenvolvedor (Collectd, 2021).

2.5.8.1.2 Telegraph

O Telegraf é um software *open-source*, usado para captar, processar, agregar e escrever métricas e eventos de bases de dados, sistemas e sensores de IoT. Trata-se de um software escrito em Go, sendo assim um binário compilado e autónomo. Além disso, pode ser executado em qualquer sistema sem dependências externas, npm, pip, gem ou outras ferramentas de gestão de pacotes.

A sua arquitetura, baseia-se em plugins de fácil instalação e configuração, e com o mínimo impacto no sistema, o que facilita a sua integração por parte dos desenvolvedores, sendo assim facilmente estendível. Pode por isso ser integrado com diferentes módulos como Graphite, Influx, Prometheus, entre outros (Telegraf, Telegraf, 2021).

Sendo um agente orientado por plugins, podem-se destacar 4 diferentes tipos de plugins (Telegraf, Telegraf, 2021):

- **de entrada** - coletam métricas do sistema, serviços ou API's de terceiros;
- **processadores** - transformam, convertem e/ou filtram as métricas;
- **agregadores** - criam métricas agregadas (média, mínimo, máximo, quantis, entre outros);
- **de saída** - escrevem métricas para vários destinos.

Assim, tem-se um software bastante flexível, que pode ser facilmente integrado no sistema da empresa, sem obrigar a grandes alterações no mesmo.

2.5.8.1.3 NetData

A Netdata é um software de monitorização gratuito e *open-source* que permite a monitorização, visualização e solução de problemas para sistemas, serviços e aplicativos. É uma ferramenta que possibilita a recolha de dados, visualização de métricas, aumentando assim o desempenho do sistema (Netdata, 2021).

Nesse sentido, a Netdata recolhe informação de (Netdata, 2021):

- **Sistemas** - monitorização do CPU, memória, disco, rede, entre outras.
- **Contentores** - reúne métricas de agentes de contentores, recursos e os aplicativos que os mesmos executam.
- **Aplicações** - captura métricas ao segundo de servidores da web, bases de dados, *logs*, corretores de mensagens, ferramentas APM (Application Performance Management), servidores de e-mail, entre outras.

Figura 6 - Painel de visualização de métricas do Netdata (Lai, 2018)

No entanto, este software não é apenas agente como o Collectd ou Telegraf, forçando à instalação de base de dados de séries temporais, um visualizador de métricas (Figura 6) e um sistema de notificação (Netdata, 2021). Além disso, possui um número de dependências bastante considerável que implica a necessidade de considerar aspetos de compatibilidade de cada dependência, complicando assim muito a sua utilização.

2.5.8.2 Módulos de software de armazenamento

Depois de analisados os módulos de software de captura, a secção atual contém um levantamento de diferentes bases de dados para armazenamento dos dados coletados.

2.5.8.2.1 MariaDB

O MariaDB é o sistema de gestão de bases de dados atualmente em vigor na empresa, sendo o seu funcionamento explicado na secção 2.3.11.

2.5.8.2.2 MonetDB

O MonetDB é um sistema de gestão de bases de dados (SGBD) orientado à coluna. Sendo gratuito e *open-source*, apresenta-se como uma solução bastante eficiente para grandes volumes de dados. Para isso, apresenta um modelo de armazenamento baseado em fragmentação vertical, uma arquitetura moderna para execução de consultas ajustadas por CPU, índices automáticos e adaptativos, otimização de consultas em tempo de execução e uma arquitetura de software modular.

Além disso, utiliza uma linguagem SQL com suporte para chaves estrangeiras, permitindo também o uso de *joins*, *views*, *triggers* e *stored procedures*. É totalmente compatível com as

propriedades ACID, salvaguardando a atomicidade, consistência, isolamento e durabilidade das operações.

Por fim, oferece ainda suporte a um amplo espectro de interfaces de programação como JDBC, Java, PHP, Python, C / C ++, entre outras (MonetDB, 2021).

2.5.8.2.3 Prometheus

O Prometheus é um sistema *open-source* e gratuito de gestão de bases de dados (SGBD) de séries temporais e de monitorização. Implementado na linguagem Go, é compatível com servidores com o sistema operativo Linux e suporta as linguagens de programação .NET, C ++, Go, Haskell, Java, JavaScript (Node.js), Python e Ruby.

As suas funcionalidades passam essencialmente por captar e armazenar dados, permitir consultas através de queries, apresentar gráficos de análise e realizar alertas. Fornece ainda terminais de acesso aos seus dados através de diferentes API's por intermédio de ligações RESTful HTTP e JSON.

É também um sistema *pull-based*, o que significa que recolhe periodicamente as métricas captadas por uma aplicação externa.

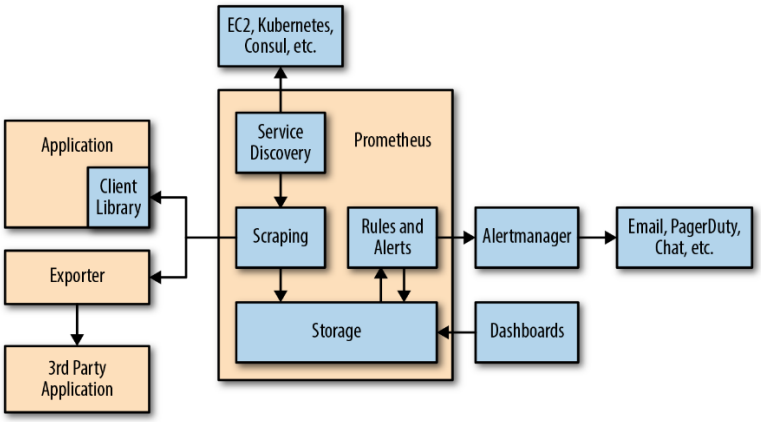

Figura 7 - Arquitetura Prometheus (Brazil, 2018)

A partir da Figura 7, é possível observar que o Prometheus usa uma técnica de *scraping* para captura de métricas. Para isso, é definida uma configuração com toda a informação necessária para a realizar, definindo os metadados a aplicar, a autenticação necessária para a conexão ou outras informações que definem como o *scrape* se realizará (Turnbull, 2018).

Por sua vez, os dados resultantes são capturados e armazenados localmente no servidor, podendo ser enviados para outros tipos de armazenamento externo ou bases de dados de série temporal. Além disso, é possível consultar e agregar os dados da série temporal e criar regras para registar consultas e agregações comumente usadas. Assim, é possível criar novas séries temporais a partir de séries temporais existentes (Turnbull, 2018).

Além disso, é possível observar na Figura 7 que o Prometheus também pode definir regras para alertas. Estes são critérios que podem ser configurados para acionar um alerta quando forem atendidos. No entanto, o servidor não vem com uma ferramenta de alerta embutida. Em vez disso, os alertas são enviados do servidor Prometheus para outro servidor, denominado *AlertManager*. Este pode gerir e enviar alertas para uma variedade de destinos, como por exemplo através do e-mail.

Já para a visualização das métricas, apesar de possuir uma *dashboard* que captura os dados da *storage*, esta é algo limitada, pelo que é recomendada a incorporação de um módulo de software de visualização.

2.5.8.2.4 InfluxDB

O InfluxDB é uma base de dados *open-source* gratuita, desenvolvida em Go, especificamente projetada para lidar com grandes volumes de dados de variadas fontes, com registo da data e hora produzidos por sensores, aplicações e infraestruturas (Influxdata, 2021). De acordo com o ranking da (DB-Engines, 2021), é a base de dados de séries temporais mais utilizada da atualidade.

Compatível com Linux, pode funcionar como uma solução independente ou ser usado para processar dados do software Graphite. Além disso, o InfluxDB foi desenvolvido para lidar com grandes volumes de dados com registo temporal produzidos por dispositivos IoT, sensores e soluções de automação residencial. Assim como o Prometheus, possui também a sua própria linguagem de consulta inspirada em SQL, denominada InfuxQL. Nesta linguagem, as consultas contínuas e as políticas de retenção apresentam procedimentos semelhantes aos de numa base de dados SQL. Os mesmos são especificados uma vez e executados de forma regular e automaticamente (Elsmore, 2020).

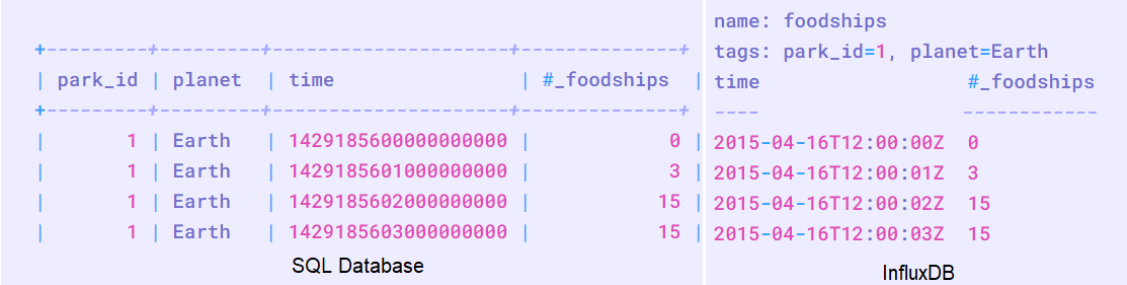

Figura 8 - Diferença entre uma tabela numa base de dados SQL e na InfuxDB (Editado de (InfluxData, InfluxDB compared to SQL databases, 2021)).

Todavia, há algumas diferenças a considerar. Por exemplo, o comando SQL JOIN não se encontra disponível para medições no InfluxDB. Além disso, e como é possível observar na Figura 8, o design de cada esquema é diferente, sendo cada medida tratada como uma tabela SQL, onde o índice primário é sempre predefinido para o tempo (InfluxData, InfluxDB compared to SQL databases, 2021).

2.5.8.2.5 Graphite

A Graphite é um software *open-source* e gratuito de armazenamento de dados numéricos de séries temporais que permite ainda renderizar gráficos com base nesses dados. Desenvolvido em Python, é compatível com os sistemas operativos Linux e Unix e suporta as linguagens Javascript (Node.js) e Python. Além disso, permite o acesso aos seus recursos através HTTP API ou via sockets (db-engines, 2021).

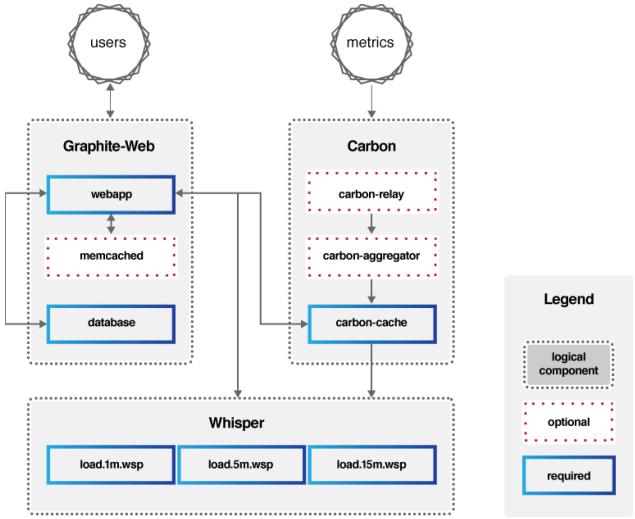

Figura 9 - Arquitetura Graphite (Graphite, 2021)

A partir da Figura 9, é possível observar que o software Graphite se divide em três componentes:

1. **Carbon** - serviço de alto desempenho que escuta dados de séries temporais;
2. **Whisper** - biblioteca de bases de dados simples para armazenar dados de séries temporais;
3. **Graphite-web** - Interface do utilizador e API do Graphite destinada a renderizar gráficos e painéis.

As métricas são alimentadas por meio do serviço Carbon, que grava os dados em bases de dados Whisper para armazenamento de longo prazo. Assim, os utilizadores interagem com a interface ou web API do Graphite, que por sua vez consulta o Carbon e o Whisper na procura dos dados necessários para construir os gráficos solicitados.

Por fim, a plataforma web oferece uma variedade de estilos e formatos de saída, incluindo imagens brutas, CSV, XML e JSON, permitindo que qualquer pessoa incorpore facilmente gráficos personalizados em outras páginas da web ou painéis (Graphite, 2021).

2.5.8.3 Módulos de software para visualização

Para finalizar o estado da arte, esta secção apresenta o levantamento e descrição de módulos de software que permitem visualizar e analisar os dados recolhidos dos sensores.

2.5.8.3.1 Grafana

Grafana é um software de visualização de métricas *open-source* que permite consultar, visualizar e enviar alertas sobre métricas e *logs*, independentemente de onde estejam armazenados.

Possui um modelo de fonte de dados facilmente conectável e suporte avançado, através dos seus plugins, para uma vasta gama de bases de dados de séries temporais, como Graphite, Prometheus, Elasticsearch, OpenTSDB e InfluxDB. Além disso, este software integra um suporte para fornecedores de monitorização de nuvem como Google Stackdriver, Amazon Cloudwatch, Microsoft Azure e bases de dados SQL como MySQL e Postgres (Grafana, Grafana, 2021).

Figura 10 - Exemplo duma Grafana Dashboard (Kumar, 2019)

Através dos dados consultados, como é possível observar na Figura 10, é possível construir gráficos customizados, tabelas, indicadores e *heatmaps*. Já o seu layout, apresenta uma ampla gama de opções de personalização, com editores de painel que permitem alterar as diferentes configurações de uma forma acessível (Grafana, The analytics platform for all your metrics, 2021).

2.5.8.3.2 Kibana

O Kibana é um software de *front-end,* gratuito e *open-source*, que fornece recursos de pesquisa e visualização de dados indexados ao cluster Elasticsearch. Uma vez que foi projetado para funcionar apenas com o cluster Elasticsearch, não oferece suporte a nenhum outro tipo de fonte de dados. No entanto, é bastante acessível e facilmente configurável, podendo ser instalado em Linux, Mac, Windows ou Docker (Elastic, 2021).

Figura 11 - Exemplo de uma Kibana Dashboard (Elastic, 2021)

Como é possível observar na Figura 11, o seu painel combina vários elementos visuais, nomeadamente vários tipos de gráficos que possibilitam visualizações analíticas em tempo real de grandes volumes de dados. Podem ainda ser adicionadas tabelas, histogramas e mapas dependendo da análise que se pretende efetuar (Elastic, 2021).

Relativamente à compatibilidade com módulos de software de armazenamento, possui alguns plugins oficiais para a receção de dados. Porém, dentro das opções analisadas, é apenas compatível com o Graphite (Elastic, 2021).

2.5.8.3.3 Chronograf

O Chronograf é um módulo de software *open-source* gratuito para visualização de dados. Produzido pela InfluxData, está otimizado para trabalhar com ferramentas de armazenamento de dados como a InfluxDB, não sendo compatível com outras fontes (InfluxData, Chronograf, 2021).

Figura 12 - Exemplo de uma Chronograf Dashboard (Chronograf, 2021)

Assim, este software apresenta-se como uma solução completa com mais de 20 painéis predefinidos que, como se pode observar na Figura 12, podem ser personalizados ou reconstruídos consoante as necessidades de cada utilizador.

Além disso, é a interface de utilizador para o Kapacitor, que é um mecanismo de processamento de dados nativo e pode processar dados de fluxo e lote do InfluxDB. Permite ainda criar e gerir alertas com uma interface amigável para o utilizador e oferece um visualizador de *logs* para visualizar, pesquisar, filtrar e analisar dados de log (InfluxData, Chronograf 1.7 documentation, 2021).

2.5.9 Avaliação das soluções

Considerando a finalidade das ferramentas selecionadas, foram analisados três softwares, cada um pertencente a um dos três tipos de ferramentas, através de critérios definidos na secção 2.5.7.

Começando pelos softwares de captura de métricas, foram analisadas três alternativas: Collectd, Telegraf e Netdata.

Tabela 2 - Análise dos critérios para a escolha do software de captura

Critérios	**Collectd**	**Telegraf**	**Netdata**
Open-source	Sim	Sim	Sim
Gratuito ou de baixo custo	Sim	Sim	Sim
Compatibilidade com Linux	Sim	Sim	Sim
Suporte às métricas	Sim	Sim	Sim
Conectividade com ambientes externos	Sim	Sim	Sim, mas com necessidade de configuração
Compatibilidade com software de armazenamento de dados	InfluxDB, Graphite, Prometheus	InfluxDB, Graphite, Prometheus	InfluxDB, Graphite, Prometheus
Dependências	Sim (dependências Perl)	Não	Sim (libuuid-devel, zlib-devel, libuv, entre outros)

A partir da Tabela 2, é possível observar que os softwares analisados estão de um modo geral de acordo com os critérios propostos. No entanto, há alguns que se destacam mais que outros. As principais diferenças detetadas prendem-se com o *footprint*, conectividade com ambientes externos e dependências.

Nesse sentido, o Netdata é o software com um maior *footprint*, uma vez que por padrão faz uso dos recursos computacionais para manter tanto o armazenamento como o ambiente de visualização das métricas. Este fator pode ser melhorado através de uma configuração manual do Netdata Agent. Em relação à conectividade com ambientes externos, todos os softwares oferecem essa possibilidade, mas o Netdata requer uma configuração manual do conetor responsável pela exportação. Relativamente às dependências, o Telegraf foi o único software

que não apresentou qualquer dependência, o que acaba por ser um fator importante uma vez que reduz a possibilidade de problemas de compatibilidade e depreciação de pacotes.

Deste modo, a ferramenta de captura selecionada foi o Telegraf. Além de não possuir dependências, possui um grande número de plugins que podem ser instalados a partir de repositórios padrão. Além disso, possui um *footprint* reduzido, e a configuração é simples, através de um único ficheiro *conf*.

Apesar das vantagens, e como explicado previamente no Capítulo 2.5.7, esta seleção não vai ser implementada neste projeto, ficando o estudo disponível apenas para consideração futura pela empresa.

Relativamente à escolha do software de armazenamento, foram analisadas cinco alternativas: MariaDB, MonetDB, Prometheus, InfluxDB e Graphite.

Tabela 3 - Análise dos critérios para a escolha do software de armazenamento

Critérios	**MariaDB**	**MonetDB**	**Prometheus**	**InfluxDB**	**Graphite**
Open-source	Sim	Sim	Sim	Sim	Sim
Gratuito ou de baixo custo	Sim	Sim	Sim	Sim	Sim
Compatibilidade com Linux	Sim	Sim	Sim	Sim	Sim
Dependências	Não	Não	Não	Não	Sim
Facilidade de uso	Sim (SQL)	Sim (SQL)	Sim (PromQL)	Sim (InfluxQL)	Não (Funções)
Conectividade com ambientes externos	Sim (JDBC)	Sim (JDBC)	Sim (API HTTP REST)	Sim (Line Protocol/HTTP)	Sim (HTTP API Sockets)

Dentro dos parâmetros analisados na Tabela 3, há dois fatores que influenciaram a decisão final, as dependências e a usabilidade.

O excesso de dependências é o principal ponto negativo do Graphite. Além disso, os três módulos que compõem o software (Whisper, Carbon e Graphite-web), possuem um

procedimento de instalação complexo. Isto, somado ao uso de funções que se aplicam a cada métrica em vez de uma linguagem própria de consulta, descartou esta hipótese.

Já o Prometheus e o InfluxDB, ao contrário do Graphite, não possuem dependências externas e revelaram ser fáceis de instalar. No entanto, e apesar de possuírem linguagens baseadas em SQL, apresentam diferenças na sua sintaxe que tornam o seu uso relativamente complexo. Deste modo, a sua escolha obrigaria a um período de aprendizagem por parte da empresa, tornando o seu processo de integração e manutenção mais complicado. Com isso, haveria custos associados, sendo por isso também descartadas.

Assim, restam o MariaDB e o MonetDB que de um modo geral revelaram ser as melhores opções. Sem dependências externas, com possibilidade de conexão com o Java através de uma configuração JDBC simples e usando a linguagem SQL, ambas as soluções cumpriram todos os requisitos pré-estabelecidos. Por isso, optou-se por implementar e testar as duas bases de dados de modo a averiguar a melhor solução.

Por último, analisaram-se três módulos de software (Grafana, Kibana e Chronograf) para visualização das métricas.

Tabela 4 - Análise dos critérios para a escolha do software de visualização

Critérios	**Grafana**	**Kibana**	**Chronograf**
Open-source	Sim	Sim	Sim
Compatibilidade com Linux e Windows	Sim	Sim	Sim
Gratuito ou de baixo custo	Sim	Sim	Sim
Acesso remoto e em multiplataformas	Sim	Sim	Sim
Customizável	Sim	Sim, mas limitada	Sim
Compatibilidade com o software que fornece os dados	Sim (Graphite, Prometheus, InfluxDB, MariaDB, MonetDB)	Não (ElasticSearch)	Sim (InfluxDB)

Mais uma vez, é visível a partir da Tabela 4 que os softwares analisados cumprem a maior parte dos requisitos especificados. Apesar disso, alguns critérios revelaram-se importantes para a decisão final, nomeadamente a compatibilidade com o módulo de software para armazenamento e o quão customizáveis são as ferramentas analíticas.

Deste modo, o Kibana revelou-se incompatível com o módulo para armazenamento escolhido, uma vez que só aceita como origem de dados softwares que são desenvolvidos pela Elastic, o seu desenvolvedor. Além disso, não permite enviar alertas de acordo com um conjunto de regras predefinidas, a partir de softwares externos. Já entre os outros dois softwares, ambos os revelaram ser bastante competentes, cumprindo todos os critérios estabelecidos. No entanto, a escolha recaiu no Grafana devido à compatibilidade com diversas origens de dados e a possibilidade de desenvolver os gráficos com diferentes linguagens de consulta. Possui diversos componentes visuais para análises de dados mais detalhadas e tem uma interface melhor do que o Chronograf.

3 Análise da Solução

Uma vez descrito detalhadamente o problema na secção 2.1.2, e analisados os principais trabalhos e tecnologias relacionados com o domínio do mesmo no Capítulo 2.5, procedeu-se então à análise da solução pretendida.

Para isso, efetuou-se um estudo detalhado do problema, definindo-se inicialmente os conceitos relevantes para o negócio num glossário e elaborando-se posteriormente uma análise e design da arquitetura e funcionalidades que serviram de base para a implementação da solução.

Em seguida definiram-se todas as funcionalidades pretendidas tendo por base os requisitos funcionais e não funcionais recolhidos.

Desta forma, este capítulo apresenta por isso secções dedicadas ao estudo aprofundado do domínio do problema (glossário do projeto e modelo de domínio), à especificação dos requisitos funcionais e não funcionais e à descrição detalhada dos casos de uso identificados.

3.1 Domínio do problema

Numa primeira abordagem realizou-se uma análise à estrutura arquitetural do sistema de monitorização usado pela empresa, bem como aos conceitos e boas práticas existentes, com vista a uma melhor compreensão do mesmo.

Foi igualmente necessário relacionar o enquadramento teórico relativo ao processo de monitorização com o que é realmente realizado, de forma a compreender as principais diferenças e conceitos que são utilizados especificamente no contexto do problema.

Após a descrição e análise de todos os dados recolhidos, procedeu-se à construção de um glossário do projeto, onde foram colocados os principais conceitos e uma breve explicação de cada um.

3.1.1 Glossário da aplicação em desenvolvimento

Com vista a uma melhor compreensão do funcionamento do projeto, foi realizado um glossário com os principais conceitos usados. Estes são parte integrante dos diagramas de análise ao longo deste capítulo, sendo usados também no desenho (Capítulo 4) e construção (Capítulo 5) da solução.

Tabela 5 – Glossário do domínio

ChangedState	Simboliza a mudança de estado dos sensores de ativo para desativado ou vice-versa.
ExpectedWorking	Representa o estado expectável dos sensores, ou seja, se deveriam estar ativos ou desativados num dado momento.
Defeito	Indica se uma peça foi produzida com ou sem anomalias.
Estado	Mostra se um sensor está ativo ou desativado.
HasDefect	Possui informação sobre os defeitos das peças produzidas.
Interface Gráfica	Apresenta os cálculos das métricas ao utilizador.
IsWorking	Contém informação sobre os sensores que estão ativos.
Máquina	É responsável por criar peças.
Peça	Produzida por uma máquina, pode ou não ter defeitos.
Sensor	Monitoriza uma máquina.
Servidor	Consulta o estado dos 16 sensores.

SensorsInfo	Objeto com a informação de uma leitura dos sensores (ExpectedWorking, HasDefect, IsWorking) e o Timestamp.
SensorsPeriodInfo	Objeto com a informação a inserir na base de dados. Estende a informação do SensorsInfo e calcula o ChangedState e o TimeSinceLastUpdate.
TimeSinceLastUpdate	Número de segundos entre a última leitura de dados do servidor e a leitura atual.
Timestamp	Valor único que representa a data e o tempo exatos em que a leitura atual está a ser feita.

3.1.2 Modelo de Domínio

Depois de elaborado o glossário com os conceitos de negócio, procedeu-se então à construção do modelo de domínio que pode ser observado na Figura 13.

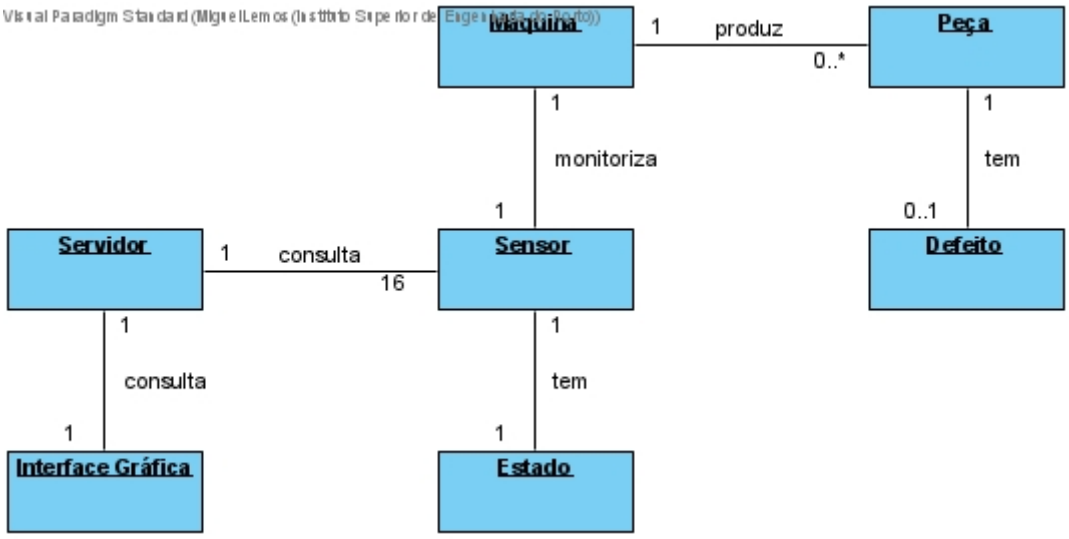

Figura 13 - Modelo de domínio

A partir do diagrama é percetível toda a lógica de negócio do projeto. No entanto, é importante referir que alguns destes conceitos não terão representação direta através de classes do domínio. Por questões de desempenho e de otimização de espaço usado na BD, que serão vistos mais à frente, a representação de cada conceito pode assumir a forma de atributo de classe, classe ou componente do sistema. Por isso, a inclusão de alguns conceitos no modelo de domínio acontece apenas para perceção da lógica de negócio.

3.2 Requisitos funcionais e não funcionais

Aquando do processo de análise e engenharia de requisitos, foi realizado um levantamento, em conjunto com a empresa, dos funcionais e não funcionais com o objetivo de satisfazer as funcionalidades previstas.

A partir do modelo FURPS+ capturaram-se e qualificaram-se requisitos com completude ao longo do projeto, dividindo-se em cinco dimensões – funcionalidade, usabilidade, confiabilidade, desempenho e suportabilidade. Relativamente ao "+", indica uma dimensão adicional que lida com restrições técnicas ao desenho, implementação, interface e físicas (engenheirando-software, 2015).

Com base nos requisitos obtidos, procedeu-se posteriormente à criação de diferentes casos de uso com vista à implementação das diferentes funcionalidades. A seguir encontram-se enumerados e organizados todos os requisitos, divididos de acordo com o modelo FURPS+, identificados durante o decurso da análise e implementação da aplicação.

3.2.1 Requisitos funcionais

Os requisitos funcionais são de extrema importância no desenvolvimento de aplicações, pois, sem eles não há funcionalidades nos sistemas (Canguçu, 2021).

Com isso, a sua presença neste sistema tem por base a restrição da apresentação das métricas para um dado um período de tempo. Estas restrições são dinâmicas e variam consoante o período que o utilizador estabelece.

- **Utilizador:** representativo da pessoa (cliente) que usa a aplicação.

Assim sendo, é apresentada de seguida a primeira dimensão do modelo FURPS+, onde se descrevem as funcionalidades destinadas ao utilizador.

Utilizador:

Acedendo ao sistema desenvolvido pode:

- Inicializar a captura dos dados provenientes dos sensores;
- Visualizar em tempo real na *dashboard* do Grafana, a variação do valor das métricas OEE calculadas a partir dos dados capturados;
- Selecionar diferentes períodos de tempo para a visualização das métricas.

É assim compreendido que existe apenas um utilizador comum para esta aplicação, pelo que os requisitos funcionais acima referidos têm de ser válidos para o mesmo.

3.2.2 Requisitos não funcionais

Os requisitos não funcionais de software são aqueles que não descrevem o que o sistema fará, mas sim de que modo o fará.

Designados também como atributos de qualidade, encontram-se relacionados com a utilização de um sistema em termos de usabilidade, desempenho, confiabilidade e de tecnologias envolvidas. Além disso, assumem um papel de extrema importância durante o desenvolvimento de um sistema, podendo ser usados como critérios de seleção na escolha de alternativas ao projeto, arquitetura e forma de implementação, sendo avaliados através de testes ou de forma subjetiva (Canguçu, 2021).

Deste modo, são apresentados em seguida os requisitos não funcionais acompanhando o modelo de FURPS+.

Usabilidade

Avalia da interface com o utilizador. Possui diversas subcategorias, entre elas: prevenção de erros, estética, design, ajudas, documentação, consistência e padrões.

- A interação entre os utilizadores e o sistema deve ser simples, intuitiva e inteiramente ajustada à ação em causa;
- Deve existir um tratamento específico para situações de funcionamento não esperado, impossibilitando que os erros sejam visualizados na interface com o utilizador;
- Sempre que pedido pelos utilizadores, a aplicação deve fornecer uma ajuda/explicação apropriada para a tarefa que o utilizador está a executar no momento.

Desempenho

Avalia as condições de desempenho do software, nomeadamente: tempo de resposta, consumo de memória, utilização da CPU, capacidade de carga e disponibilidade da aplicação.

- O cálculo das métricas deve ser inferior a 5 segundos para mostragem das métricas em tempo real;
- O sistema deve estar preparado para que o tempo de resposta seja sensivelmente o mesmo independentemente da carga existente.

Suportabilidade

Os requisitos de suportabilidade agrupam vários atributos, como: testabilidade, adaptabilidade, manutenção, compatibilidade, configurabilidade, instabilidade, escalabilidade entre outros.

- Deve ser escalável para o processo de adição de novas funcionalidades.

Restrições de design

Especifica ou restringe o processo de design do sistema.

- Adoção do processo de desenvolvimento de software iterativo e incremental;
- Boas práticas de design, nomeadamente padrões GRASP, SOLID e boas normas de codificação;
- Uso dos padrões *Model*, *View* e *Controller* (MVC), bem como *Repository* e Service.
- Deve ser usada uma base de dados relacional baseada em SQL para persistência dos dados.

Restrições de implementação

Distingue ou restringe o código ou a construção de um sistema através de: normas e linguagens de implementação, políticas de integridade de base de dados, limites de recursos e sistema operativo.

- A captura de dados do servidor deve ser feita através do uso de *Sockets*;
- O núcleo principal do software deve ser desenvolvido em programação orientada a objetos;
- Adoção do controle de versões (GIT);
- Testes unitários e de performance;
- Deve ser utilizado o Grafana para a parte da interface gráfica (cliente).

Restrições de interface

Especifica ou restringe as funcionalidades inerentes à interface entre o sistema e outros.

- O módulo de captura deve permitir a comunicação com o servidor através do uso de *Sockets*;
- Uso de interface JDBC por parte do módulo de software de captura para comunicação com a base de dados;
- O módulo de visualização deve comunicar com a base de dados através de *queries* SQL, em tempo real, para a mostragem das métricas de forma contínua.

3.3 Casos de uso

Uma vez identificados os requisitos funcionais e não funcionais especificados na secção 4.2, procedeu-se então à criação dos diferentes casos de uso representativos das diversas funcionalidades implementadas.

Na Figura 14 é possível observar a distribuição dos mesmos, tendo em conta os atores integrantes.

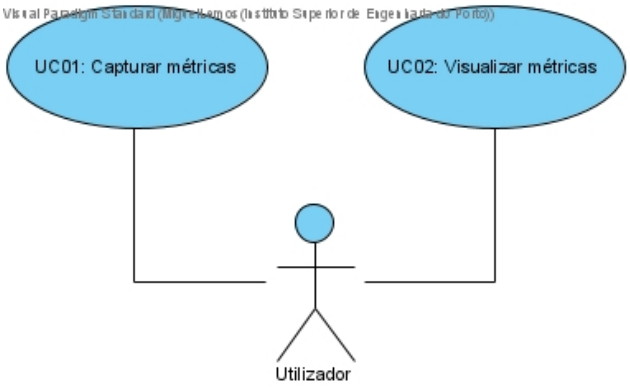

Figura 14 - Diagrama de casos de uso

3.3.1 UC01: Capturar métricas

De modo a permitir a visualização das métricas, é preciso previamente capturá-las. Assim, é necessária uma funcionalidade que inicie o módulo de captura, sendo esta apresentada pelo UC01.

Tabela 6 - UC01: Capturar métricas

Ator	Utilizador
Descrição	O caso de uso tem como objetivo a captura das métricas.
Pré-Condições	A API externa da empresa tem de estar a funciona.
Cenário principal	1 - O ator inicia o processo de captura de métricas no sistema. 2 – O sistema tenta estabelecer a ligação para capturar as métricas e informa do sucesso/insucesso da operação.
Cenário alternativo	2a. O sistema não consegue estabelecer a ligação. O ator volta a tentar iniciar o processo de captura (passo 1) 2b. O sistema não captura as métricas. O ator volta a tentar iniciar o processo de captura (passo 1)
Pós condições	É iniciada a captura das métricas.
Requisitos atendidos	1 (*Utilizador*)

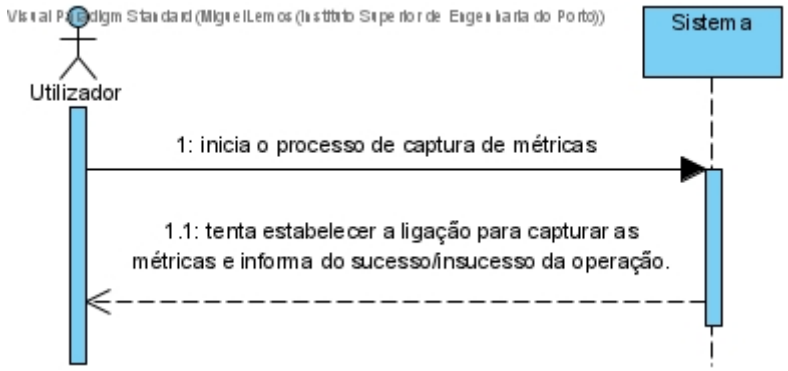

Figura 15 - Diagrama de Sequência do Sistema do UC01: Capturar métricas

A partir da Tabela 6, é possível observar que para o sistema iniciar com sucesso, necessário que a API externa da empresa esteja a funcionar, caso contrário é impossível estabelecer ligação e receber os dados. Em caso de insucesso, há dois cenários alternativos onde o utilizador é notificado sobre o insucesso da conexão ou sobre a inexistência de métricas para coletar. Em ambos os casos, o utilizador pode repetir de novo a operação e tentar reiniciar o sistema.

3.3.2 UC02: Visualizar métricas

Uma vez iniciada com sucesso a captura das métricas, o utilizador pode visualizá-las através de uma funcionalidade que as apresente ao utilizador, e que se encontra representada pelo UC02.

Tabela 7 - UC02: Visualizar métricas

Ator	Utilizador
Descrição	O caso de uso tem como objetivo a visualização das métricas.
Pré-Condições	A captura de métricas (UC01) tem de estar a ser realizada com sucesso.
Cenário principal	1 - O ator inicia o processo de visualização de métricas no sistema. 2 – O sistema disponibiliza uma opção para selecionar um período de tempo. 3 – O ator insere o período para o qual pretende visualizar as métricas. 4 – O sistema disponibiliza uma opção para selecionar uma frequência de atualização. 5 – O ator seleciona a frequência de atualização que pretende. 6 – O sistema apresenta as métricas.
Cenário alternativo	6a. O sistema não possui métricas para o período de tempo definido. O ator insere outro período de tempo. (passo 3) 6b. O sistema não apresenta métricas. O ator seleciona a frequência de atualização que pretende (passo 5)
Pós condições	São apresentadas as métricas.
Requisitos atendidos	2 (*Utilizador*)

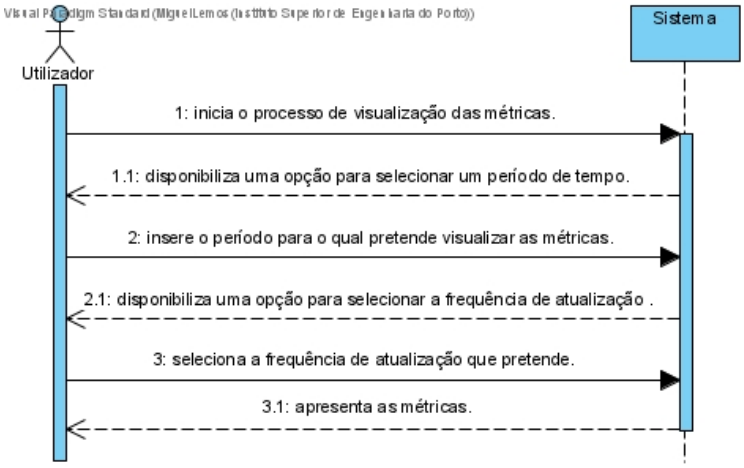

Figura 16 - Diagrama de Sequência do Sistema do UC02: Visualizar métricas

Da análise à Tabela 7, verifica-se que para o sistema disponibilizar as métricas, é necessário que a captura das mesmas (UC01) seja realizada com sucesso, caso contrário não há dados para mostrar.

Além disso, percebe-se também que é necessário selecionar um período de tempo e a frequência de atualização (consultas à base de dados) de modo a visualizar as métricas. Caso uma destas condições não se verifique, existem dois cenários alternativos. No primeiro caso (6a), se o período de tempo selecionado pelo utilizador não possuir dados, este pode inserir outro período de tempo e tentar de novo. No segundo caso (6b), se não tiver sido selecionada nenhuma frequência de atualização, o utilizador tem a possibilidade de selecionar uma das frequências disponibilizadas.

4 Desenho da Solução

Uma vez finalizada a fase de análise da solução que se encontra no Capítulo 3, procedeu-se então ao desenho da mesma.

Para isso, o presente capítulo expõe inicialmente uma abordagem teórica à arquitetura usada para o seu desenvolvimento, bem como os padrões de software usados. Além disso, são apresentados também os diagramas dos diferentes componentes e classes que a constituem, assim como os diagramas de sequência representativos dos diferentes casos de uso. Em seguida, é desenhada a arquitetura da base de dados e o painel de visualização do Grafana.

4.1 Arquitetura geral

De modo a perceber de que forma a solução será implementada, procedeu-se então ao desenho da mesma. No Capítulo 3, foram selecionadas quatro tecnologias para serem integradas em três partes diferentes (captura, armazenamento e visualização).

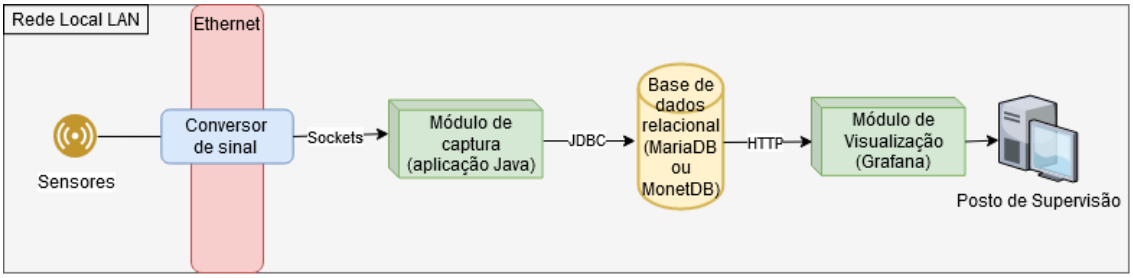

Figura 17 - Arquitetura da Solução

A partir da Figura 17, compreende-se que o módulo de captura desenvolvido em Java tem de comunicar com a API externa da empresa para recolher os dados dos sensores. Desta forma, é necessário o uso de *Sockets* para a realização da conexão através da rede.

De seguida, e depois de devidamente tratados os dados, é necessário persisti-los na base de dados. Uma vez que ambas as BD's utilizam a API JDBC para a comunicação com aplicações Java, é necessário integrar esta API no módulo de captura para estabelecer a comunicação com cada BD.

Por fim, o Grafana conecta-se com a base de dados através de um plugin que usa o protocolo HTTP. Assim que a ligação é estabelecida, através do uso de *queries* recolhe os dados que necessita para o cálculo das métricas e apresenta os resultados dos cálculos ao utilizador.

4.2 Arquitetura do módulo de captura

4.2.1 Arquitetura Model-View-Controller (MVC)

Em termos arquiteturais, a solução seguiu o modelo Model-View-Controller, representado na Figura 18, tornando o código mais modular e fácil de testar.

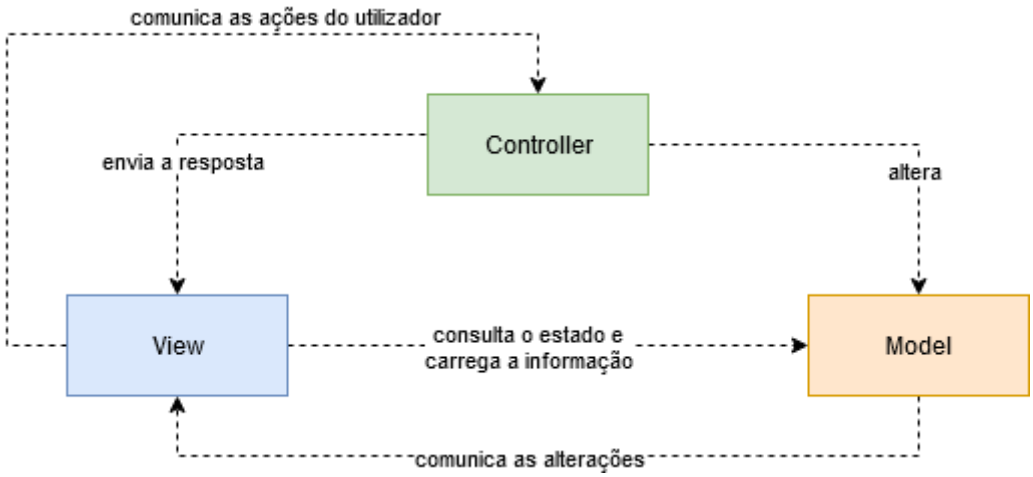

Figura 18 - Diagrama com a representação da arquitetura MVC

O modelo MVC é um padrão de software usado principalmente na camada referente à interface de interação de uma aplicação com o utilizador. O seu uso tem como objetivo organizar o código através da separação de diferentes conceitos de uma aplicação. Assim, tem-se:

- **Model** – representa a lógica de negócio através de um objeto com diferentes dados;
- **View** – é a interface com o utilizador, sendo responsável pela apresentação dos dados contidos na classe *Model*. A *View* sabe como deve aceder aos dados, mas não sabe o que estes significam ou o que o utilizador pode fazer para manipulá-los;
- **Controller** – atua como uma camada intermédia de controlo entre o *Model* e a *View*. A sua função passa por controlar o fluxo de dados no objeto modelo e atualizar a *View* sempre que os dados são alterados. O seu uso permite assim manter os conceitos de *View* e *Model* separados (Code Academy, s.d.).

4.2.2 Padrão Service

Para além da arquitetura MVC, foi usado também o padrão *Service* para acesso e tratamento dos dados provenientes dos sensores, que segue representado na Figura 19.

Figura 19 - Diagrama representativo do padrão Service

O padrão *Service* é uma abstração da lógica do domínio. Por definição, o uso deste modelo implica a criação de uma nova camada na aplicação, onde são estabelecidas diferentes operações e coordenadas respostas para as mesmas. Através da utilização desta prática, é possível centralizar a implementação de toda a lógica do negócio numa mesma camada, evitando duplicação de código e aumentando o encapsulamento (Fowler, 2002).

4.2.3 Padrão Repository

O padrão *Repository* consiste na criação de uma camada intermédia entre os objetos do domínio e as estruturas de armazenamento de dados, como se pode observar na Figura 20.

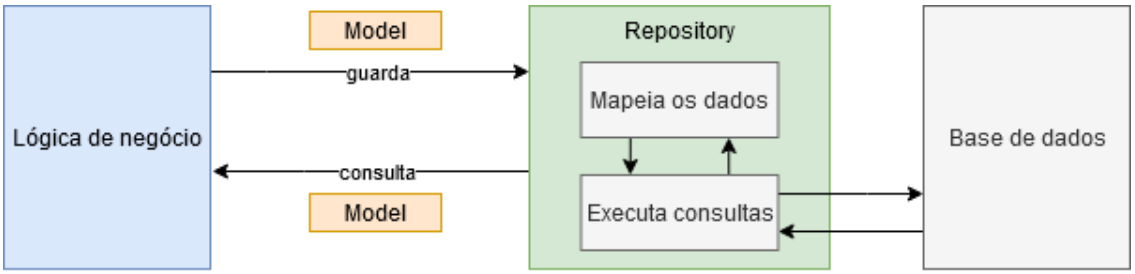

Figura 20 - Diagrama com a representação do padrão Repository

Através do uso desta prática, é possível diminuir o encapsulamento entre as classes, atribuindo aos repositórios a responsabilidade única para interagir diretamente com as bases de dados. Deste modo, as classes de domínio passam a aceder aos objetos armazenados a partir das interfaces fornecidas pelos repositórios, não precisando de implementar qualquer lógica de acesso à base de dados (Martinez, 2020).

4.2.4 Diagrama de Classes

De forma a perceber de que forma as diferentes classes e interfaces que constituem o projeto interagem entre si, desenhou-se o diagrama de classes presente na Figura 21.

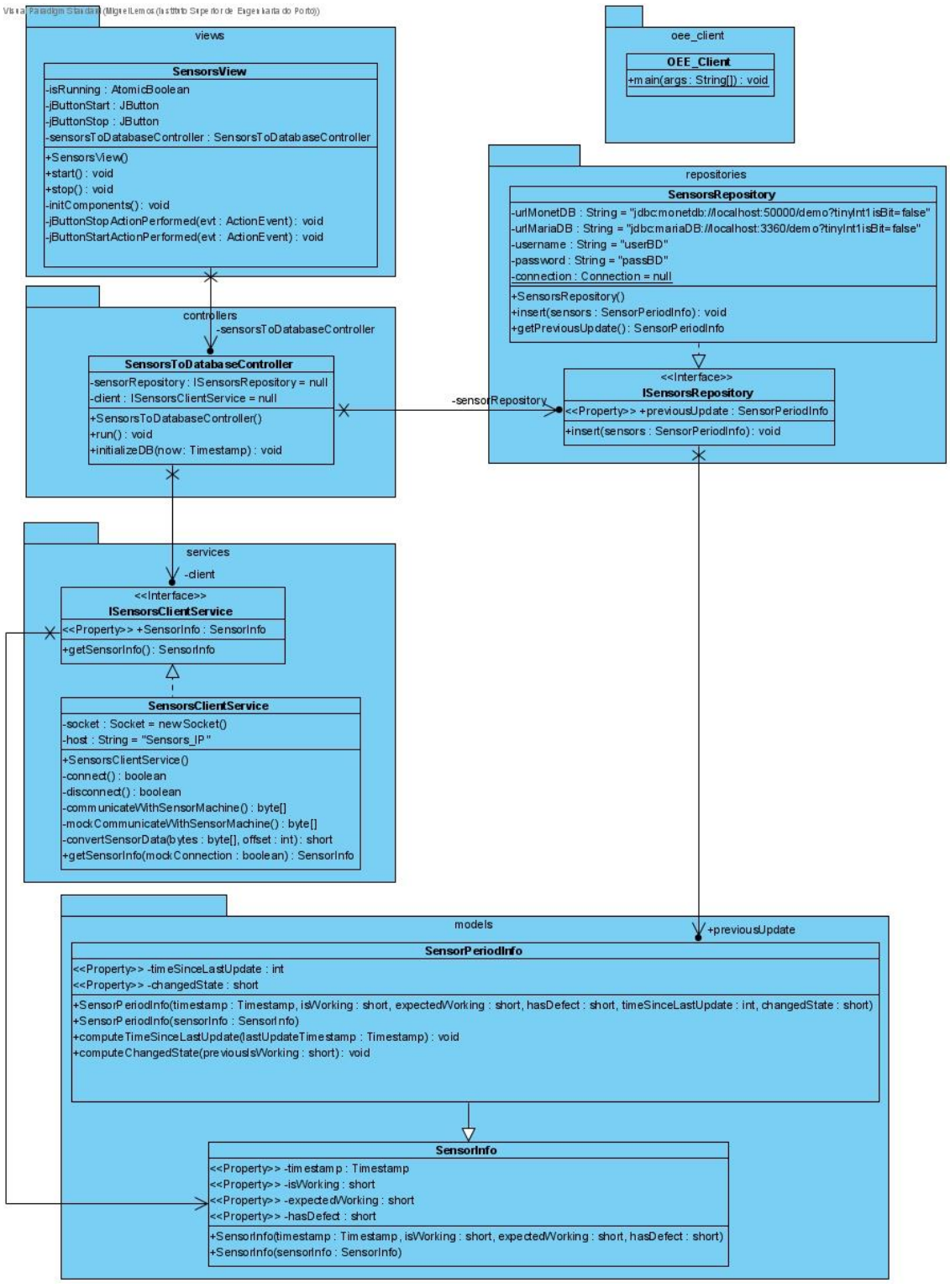

Figura 21 - Diagrama de classes

A partir do diagrama verifica-se então o uso da arquitetura MVC através do uso das classes *SensorsView*, *SensorsToDatabaseController* e as classes modelo *SensorInfo* e *SensorPeriodInfo*. A primeira recebe a informação recolhida dos sensores através do padrão *Service*. Já a segunda, herda as propriedades do *SensorInfo* e é inserida na base de dados através do padrão *Repository*.

4.3 Arquitetura da Base de Dados

Para a arquitetura da base de dados houve dois fatores que tiveram de ser considerados, o volume de dados e a rapidez das consultas.

Dado que o Grafana tem de disponibilizar as métricas em tempo real, é importante que o desempenho das *queries* para o cálculo das métricas seja sensivelmente o mesmo independentemente da carga existente.

Em certos casos, dependendo do período de monitorização, pode ser necessário armazenar um grande volume de dados. Por isso, é importante que o armazenamento dos dados seja o mais otimizado possível.

De modo a perceber quantas colunas seriam precisas e que dados seriam necessários armazenar foram analisadas as fórmulas de cálculo para cada métrica.

O cálculo da disponibilidade consiste na divisão do tempo em que a máquina se encontra a produzir pelo tempo programado para produção. A partir dos dados recolhidas dos sensores é possível saber diretamente se cada máquina se encontra em produção e se estava programada para produzir nesse instante. Desta forma, são necessárias duas colunas para guardar ambos os dados. Para contabilizar os períodos de tempo é preciso uma coluna extra que guarde os *Timestamps* a cada recolha de dados dos sensores e outra que guarde o período de tempo desde a última inserção na BD.

Assim, tem-se a versão inicial tabela da BD presente na Figura 22.

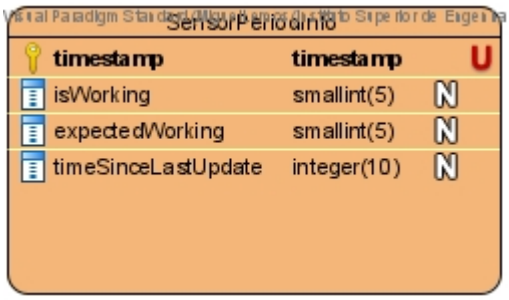

Figura 22 - Tabela da base de dados (versão inicial)

Já para o cálculo do desempenho é necessário multiplicar o tempo em que o equipamento foi projetado para funcionar pelo número de peças produzidas e dividir o resultado pelo tempo de produção. Com as colunas criadas para o cálculo da disponibilidade é possível obter o tempo de produção.

Para os restantes dados, uma vez que o tempo projetado é uma constante definida pelo cliente, este pode ser incorporado na *query* não sendo necessário o seu armazenamento. Já para o número de peças, apesar de poder ser calculado a partir das colunas existentes, optou-se pela criação de uma coluna nova. Apesar desta decisão aumentar a quantidade de dados armazenados, a alternativa passaria por incorporar este cálculo na *query*, o que traria um impacto extremamente negativo no desempenho da mesma. Como referido na secção 3.1.1, uma peça é produzida quando há uma alteração de estado do sensor de ligado (bit a 0) para desligado (bit a 1). Estas operações integradas na *query* implicariam a utilização de vários JOIN afetando a complexidade linear do algoritmo, tornando o mesmo cada vez mais ineficaz com o crescimento da base de dados.

Assim, criou-se uma nova coluna para guardar essas mudanças de estado como se pode observar na Figura 23.

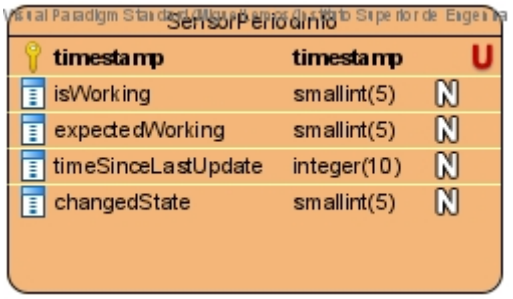

Figura 23 - Tabela da base de dados (versão intermédia)

Por fim, o cálculo da qualidade baseia-se na divisão entre o número de peças sem defeito e o número total de peças produzidas. Desta forma, não é necessária a criação de mais colunas. Através da coluna criada para as variações de estado é possível obter o número de peças, sendo o número de peças com defeito lido diretamente dos sensores.

A partir daí, obteve-se a versão final da tabela da BD na Figura 24.

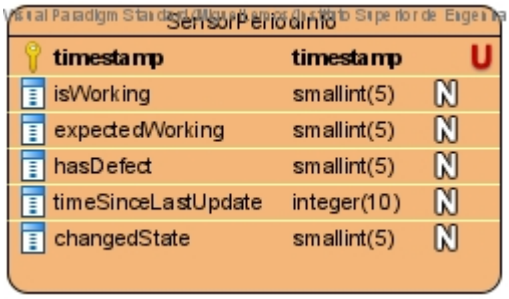

Figura 24 - Tabela da base de dados (versão final)

A partir do modelo é possível observar que a base de dados contém apenas uma tabela, denominada *Sensors*, com seis colunas, onde o *timestamp* é a chave primária.

Das colunas presentes na mesma, o *isWorking, expectedWorking e hasDefect* contêm dados recolhidos dos sensores. O primeiro indica o estado dos sensores (ligado ou desligado), o segundo refere o estado expectável (devia estar ligado ou desligado) e o terceiro indica se cada peça produzida tem defeito ou não.

Uma vez que a representação binária dos dados lidos contém a informação de 16 sensores para cada um dos três atributos, significa que se pode guardar a informação de cada atributo num *smallint*, ocupando assim o mínimo de espaço possível (16 bits) e otimizando o armazenamento.

Já o *timeSinceLastUpdate* e o *changedState,* são valores calculados pelo software de captura. O primeiro calcula a diferença de tempos entre a leitura atual e a anterior, enquanto o segundo é responsável pelo cálculo das mudanças de estado dos sensores. Estas operações, implicam a utilização de vários JOIN o que afeta a complexidade linear do algoritmo tornando o mesmo cada vez mais ineficaz com o crescimento da base de dados.

4.4 Diagramas de Componentes

A fase de desenvolvimento do design arquitetural deve seguir as boas práticas da engenharia, uma vez que caso seja necessário efetuar alterações futuras, estas terão um elevado impacto no funcionamento de todo o projeto.

Decidiu-se, portanto, usar o diagrama de componentes para representar a arquitetura usada pois facilita a compreensão e o desenho do sistema, explicitando claramente a comunicação

entre as partes envolvidas e permitindo assim uma melhor manutenção do design aquando da alteração dos requisitos.

De maneira a perceber a interação existente entre os diferentes componentes que constituem o sistema desenvolvido, procedeu-se então à elaboração de dois diagramas para o efeito, onde se apresentam uma vista geral e outra mais específica. A Figura 25, ilustra assim a vista geral da interação entre os diferentes componentes da aplicação.

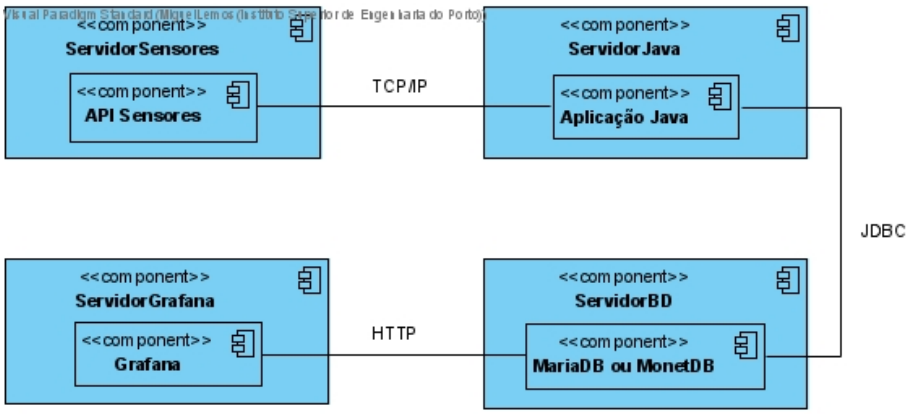

Figura 25 - Diagrama de componentes de alto nível

A partir da Figura 25, é possível observar que a aplicação Java comunica diretamente com duas API's diferentes. Por um lado, conecta-se à API da empresa através de uma ligação TCP/IP, a partir da qual consegue obter os dados dos sensores. Já para a comunicação com a base de dados, a aplicação utiliza a API *Java Database Connectivity* (JDBC) que possibilita a execução de consultas e inserções dos dados. Por fim, o Grafana coleta os dados da base de dados através de uma ligação HTTP.

Do ponto de vista de arquitetura da aplicação, procedeu-se ainda à elaboração de um diagrama de componentes mais específico que representasse as interações entre os componentes que constituem apenas o sistema desenvolvida. Para o efeito, surge a figura 26 que aplica o padrão MVC, descrito na secção 4.2.1.

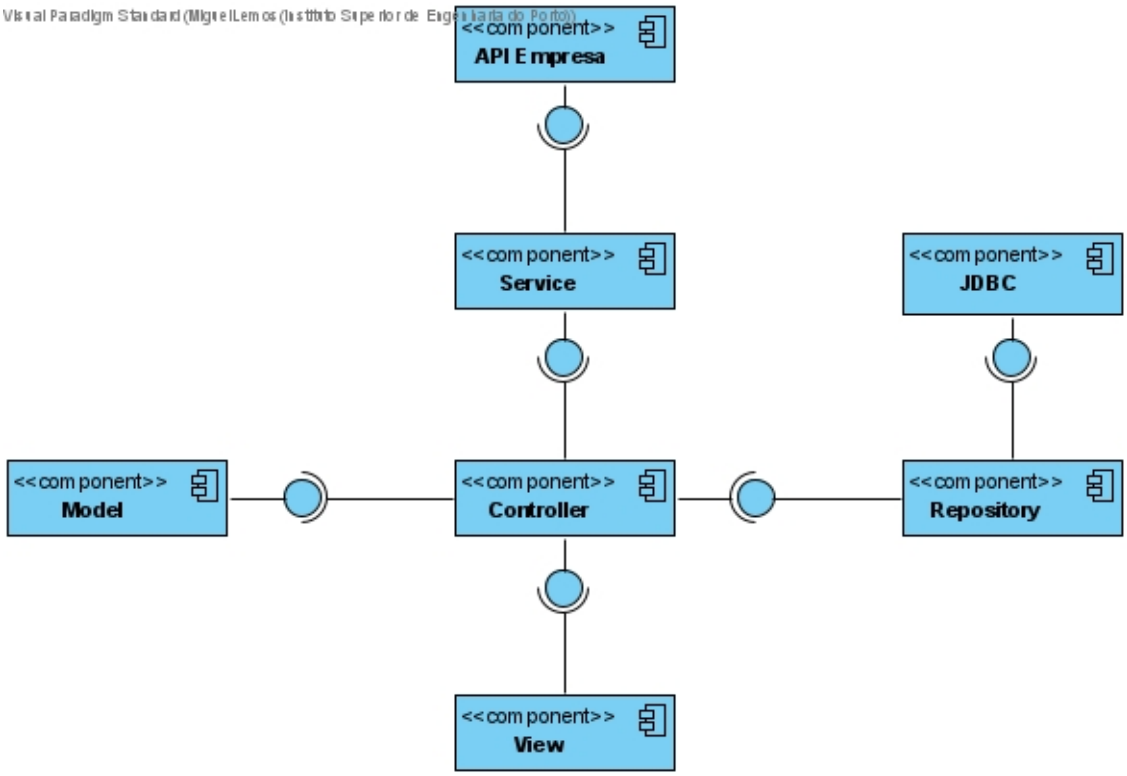

Figura 26 - Diagrama de componentes de baixo nível

Da análise à Figura 26, é possível observar que o módulo de captura é constituído por uma camada de apresentação intermediada por um componente *Controller*, que funciona como uma camada intermédia entre a View e o Model. O *Controller* comunica com a API da empresa através de um serviço que gere a comunicação através de Sockets. Por outro lado, a comunicação com a base de dados é intermediada pelo *Repository* que utiliza a interface JDBC para realizar as consultas.

Já o Model contém a lógica de negócio da aplicação e comunica com o componente *Controller* sendo manipulado pelo mesmo.

4.5 Diagramas de Sequência

Os diagramas de sequência são uma parte importante do desenho da solução, uma vez que ajudam a compreender melhor as necessidades de um novo sistema através da modelação lógica do mesmo. Deste modo, é possível observar a maneira como os objetos e componentes interagem entre si e planear e compreender as funcionalidades detalhadas de um cenário existente ou futuro.

A seguir encontram-se os Diagramas de Sequência relativos aos UC's que foram desenvolvidos para o projeto e uma explicação global de cada um.

4.5.1 UC01: Capturar métricas

A captura de métricas estabelece-se através da comunicação entre o módulo de captura desenvolvido em Java e a API externa da empresa. Para isso, após o utilizador iniciar o módulo de recolha, o *Controller* pede ao *Service* que inicie a conexão com a API.

De modo a perceber em detalhe de que forma é realizada a comunicação com a API da empresa, é preciso entender de que forma são usados os *Sockets*. Como foi explicado na secção 2.3.7, os *Sockets* permitem que a camada TCP possa identificar a aplicação para a qual os dados são enviados, uma vez que possuem informação sobre o endereço e porta do servidor a conectar.

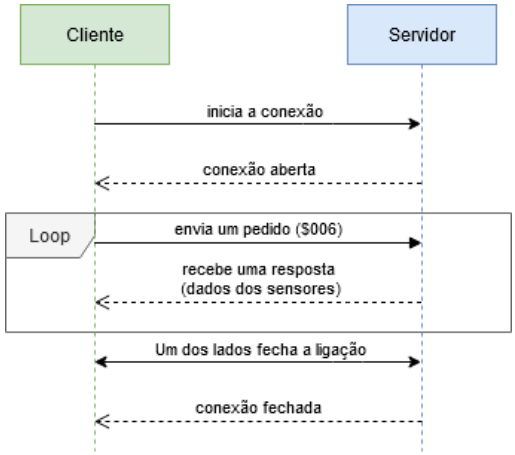

Figura 27 - Diagrama de sequência da comunicação cliente-servidor

Deste modo, é possível observar na Figura 27 que a comunicação com a API da empresa é iniciada pelo cliente através da conexão com o servidor. Para isso, o cliente usa o URI que contém a informação do endereço IP e porta de destino.

Uma vez bem-sucedida a conexão, a comunicação é feita de forma simétrica. Assim, o cliente envia vários pedidos ao servidor que vai respondendo às requisições com os dados que este pretende. Em caso de insucesso são devolvidas mensagens de erro. Assim, este processo vai-se repetindo até que um dos lados encerre a conexão.

Depois de estabelecida a conexão, o *Controller* converte e devolve a informação recolhida num objeto *SensorInfo*. A partir desse objeto, é criado o *SensorPeriodInfo* que herda a sua

informação e calcula a mudança de estado e o tempo desde a última atualização. Para isso, o *Repository* consulta a última inserção na base de dados e devolve para o *Controller* um objeto *SensorPeriodInfo* a partir do qual são realizados os cálculos. Por fim, o *Controller* envia esse objeto para o *Repository* que o insere na base de dados e informa do sucesso da operação.

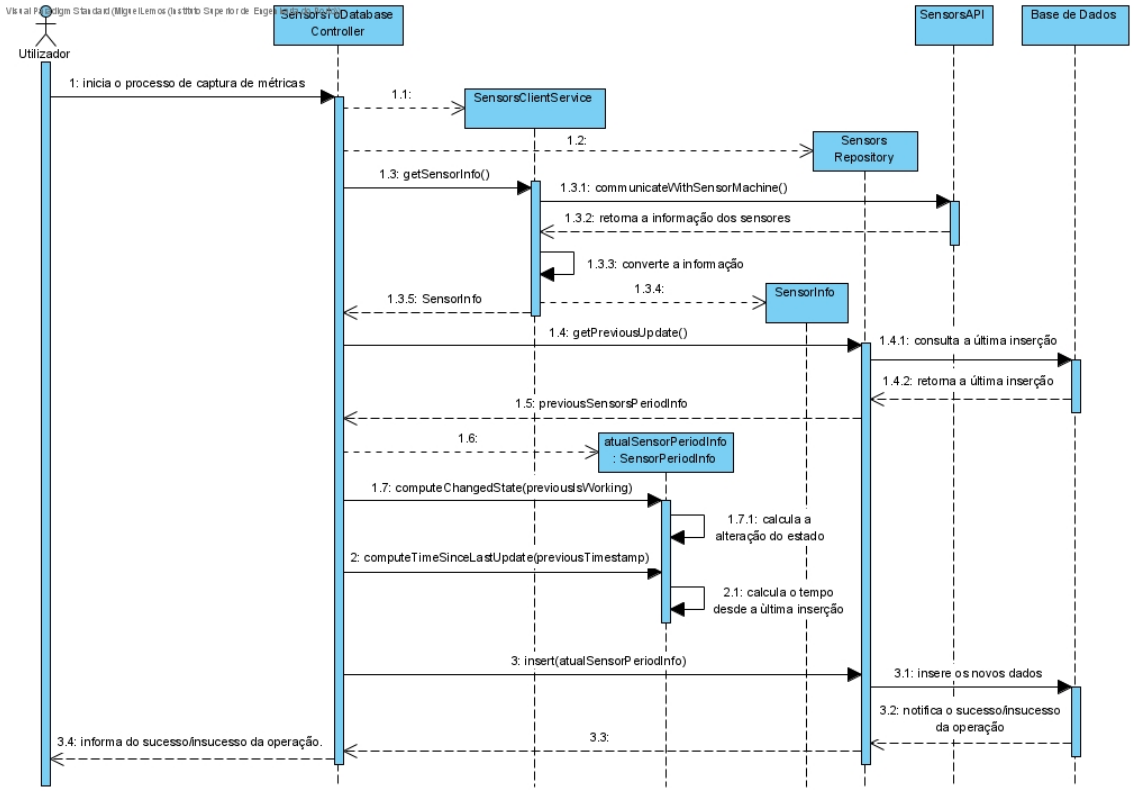

Figura 28 - Diagrama de sequência UC01: Capturar métricas

4.5.2 UC02: Visualizar métricas

Para a apresentação das métricas OEE, o Grafana realiza consultas SQL à base de dados na procura pela informação necessária para mostrar ao utilizador. Deste modo, a MariaDB assume assim um papel essencial para a concretização desse objetivo, sendo responsável pela realização de todos os cálculos necessários para cada consulta. Uma vez realizados os cálculos, a base de dados envia então as métricas para o Grafana que por sua vez as apresenta ao utilizador, como se pode observar na Figura 29.

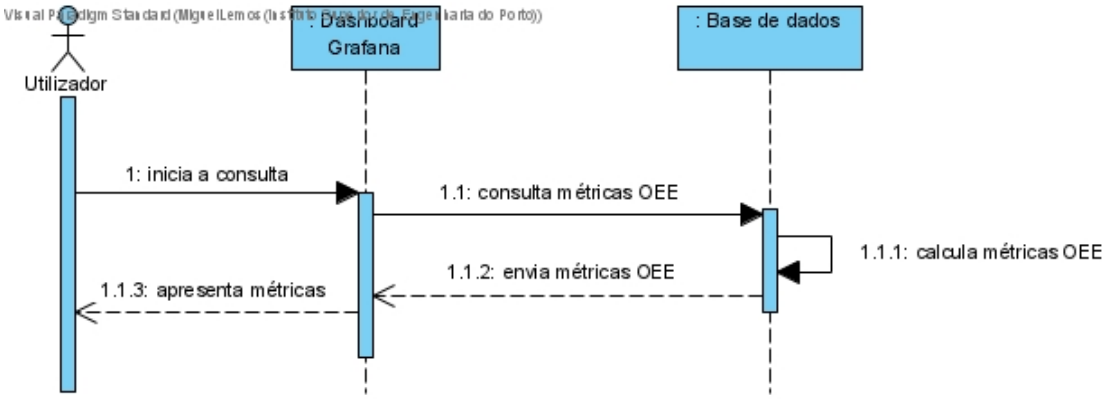

Figura 29 - Diagrama de sequência UC02: Visualizar métricas

5 Construção da Solução

Este capítulo é dedicado à apresentação dos detalhes relacionados com o enquadramento e implementação das soluções preconizadas no capítulo anterior. Assim, encontra-se uma explicação dos módulos de software integrados no desenvolvimento da solução, bem como uma descrição de cada funcionalidade implementada e respetivos resultados.

5.1 UC01: Capturar métricas

Para possibilitar a visualização das métricas procedeu-se à implementação do módulo de captura para recolher os dados dos sensores. Assim, começou-se por estabelecer uma comunicação com a API da empresa.

5.1.1 Comunicação com a API da empresa

De modo a efetuar a comunicação com a API da empresa desenvolveu-se o método representado no Excerto de Código 1.

```
private boolean connect(){
    try
    {
        socket.connect(new InetSocketAddress(host, 4660));
    }

    //Host not found
    catch (UnknownHostException e)
    {
        System.err.println("Unknown host: " + host + " with exception: "+ e);
        return false;
    }catch (IOException ex) {
        System.out.println("Error: "+ex);
        return false;
    }

    return true;
}
```

Excerto de Código 1 - Conexão com a API da empresa

Como é possível observar, a conexão é efetuada através da criação de *Sockets* que estabelecem a comunicação entre as duas partes. Aquando da sua criação, é necessário definir o endereço IP e a porta do servidor para que a ligação seja realizada. Caso não seja encontrado o endereço de servidor ou ocorra qualquer tipo de falha na comunicação, são tratadas as exceções e impressos os erros.

Uma vez estabelecida a conexão, o passo seguinte passa por enviar uma mensagem para a API da empresa de modo a obter os dados pretendidos. Assim, desenvolveu-se o Excerto de Código 2.

```
PrintWriter output = new PrintWriter(socket.getOutputStream(), true);

InputStream input = socket.getInputStream();

String request = "$006";
output.println( request );

byte[] bytes = new byte[12];
input.read(bytes, 0, 12);
```

Excerto de Código 2 - Comunicação com a API da empresa

A partir do excerto acima, verifica-se que são abertos dois canais no *Socket*, um para escrita e outro para leitura. O canal de escrita é usado para enviar um pedido ao servidor, sendo a resposta retornada a partir do canal de leitura. Neste caso, para a obtenção dos dados dos sensores, o pedido é realizado através da mensagem “$006”, sendo a resposta guardada num *array* de bytes com a informação pretendida.

Uma vez obtida a informação dos sensores, é então fechada a conexão.

```
private boolean disconnect(){
    try {
        socket.close();
    } catch (IOException ex) {
        System.out.println("Error close socket: "+ex);
        return false;
    }

    return true;
}
```

Excerto de Código 3 - Desconexão com a API da empresa

Como é possível observar a partir do Excerto de Código 3, a conexão é encerrada através do fecho do *Socket* a partir do método *close()*. Caso não seja possível fechar o *Socket*, é tratada a exceção e impresso o erro.

Uma vez explicados os passos necessários para a comunicação, é apresentado no Excerto de Código 4 o método que engloba o processo todo.

```
protected byte[] communicateWithSensorMachine(){

    if(!connect()){
        return null;
    }

    try {
        PrintWriter output = new PrintWriter(socket.getOutputStream(), true);

        InputStream input = socket.getInputStream();

        String request = "$006";
        output.println( request );

        byte[] bytes = new byte[12];
        input.read(bytes, 0, 12);

        disconnect();
        return bytes;

    }catch(IOException exception){
        System.out.println("Communication error: "+exception);
    }

    return null;
}
```

Excerto de Código 4 - Método global responsável pela comunicação com a API da empresa

A partir da análise ao código em cima obtém-se assim uma visão geral da recolha os dados dos sensores. Inicialmente é criada uma conexão com a API da empresa sendo depois iniciado o processo de comunicação para a receção dos dados pretendidos. Depois de recebida toda a informação é encerrada então a conexão com o servidor. Uma vez concluído este processo, é retornado um *array* de bytes com a toda a informação recebida.

5.1.2 Conversão dos dados recebidos

Assim que os dados dos sensores são recolhidos é necessário convertê-los para facilitar o seu uso. No entanto, é preciso primeiro entender de que maneira estes são recebidos.

Com a execução do método presente no Excerto de Código 4 obteve-se um *array* com 16 bytes, onde os 12 primeiros contêm toda a informação dos 16 sensores.

Dentro desses 12 bytes, os dados encontram-se divididos em 3 blocos de 4 bytes com informação relativa ao seu:

- estado atual (se estão ativos ou desativados);
- estado expectável (se deveriam estar ativos ou desativados);
- número de peças com defeito.

Para explicar de que forma a informação é alojada em cada byte construiu-se a Tabela 8 com um bloco de 4 bytes relativos ao estado dos sensores num dado momento.

Tabela 8 - Dados recolhidos dos sensores

Char []	F	A	4	3
Binário	0000 1111	0000 1010	0000 0100	0000 0011

Como é possível observar na tabela acima, apesar do valor lido representar o formato hexadecimal, este encontra-se alojado numa S*tring* composta por 4 *chars*. Desta forma, em vez de se ter um hexadecimal que ocuparia apenas 2 bytes, tem-se uma *String* que ocupa o dobro do espaço, gerando desperdício.

Uma vez que se pretende otimizar o armazenamento, e só são necessários 16 bits para armazenar os 16 sensores, procedeu-se então à conversão da *String* para hexadecimal com o objetivo de diminuir para metade o espaço ocupado por cada um dos três atributos.

Para isso, elaborou-se o método de conversão representado no Excerto de Código 5.

```
private short convertSensorData(byte[] bytes, int offset){
    String str_bytes = new String(bytes);

    final int hexaBase = 16;

    for (int i = offset; i < offset+4; i++) {
        bytes[i] = Byte.parseByte("" + str_bytes.charAt(i), hexaBase);
    }

    for (int i = offset; i < offset+4; i+=2) {
        bytes[i] <<= 4;
        bytes[i] |= bytes[i+1];
    }

    byte[] sensors = {bytes[offset], bytes[2 + offset]};
    ByteBuffer wrapped = ByteBuffer.wrap(sensors);
    return wrapped.getShort();
}
```

Excerto de Código 5 - Método que converte os dados provenientes dos sensores

A partir do código acima, verifica-se que o primeiro passo realizado foi a conversão dos 4 bytes recebidos para hexadecimal. Como a metade inicial de cada byte é zero, juntaram-se os bytes dois a dois utilizando operações binárias, tal como ilustrado na Tabela 9.

Tabela 9 - Lógica usada na conversão dos dados dos sensores

Valores recebidos	**'F'**	**'A'**	**'4'**	**'3'**
Hexadecimal	0x0F	0x0A	0x04	0x03
<<= 4	0xF0	0x0A	0x40	0x03
|(operação binária OR)	0xFA	0x0A	0x43	0x03
ByteBuffer	0xFA43			
getShort()	-1469			

Uma vez entendida toda a informação recebida dos sensores, elaborou-se então o seguinte método.

```
@Override
public SensorInfo getSensorInfo(){
    SensorInfo sensorInfo = null;
    byte[] response;
    Timestamp now;

    response = communicateWithSensorMachine();
    now = new Timestamp(System.currentTimeMillis());


    if(response.length > 0){
        short isWorking = convertSensorData(response, 0);
        short expectedWorking = convertSensorData(response, 4);
        short hasDefect = convertSensorData(response, 8);

        sensorInfo = new SensorInfo(now, isWorking, expectedWorking, hasDefect);
    }

    return sensorInfo;
}
```

Excerto de Código 6 - Método que devolve um objeto com a informação recolhida dos sensores

A partir do Excerto de Código 6, verifica-se que depois de recebidos os dados dos sensores, estes são enviados por parâmetro para o método de conversão. Além disso, é enviado também um offset, uma vez que o *array*, como foi referido em cima, possui dados com três tipos diferentes de informações. Assim, o método *convertSensorData* é chamado 3 vezes, sendo enviados em cada chamada 4 bytes com a informação respetiva.

Por fim, é construído e retornado um objeto *SensorInfo* com toda a informação da leitura devidamente tratada e otimizada.

5.1.3 Construção da lógica de negócio

Uma vez convertidos os dados recebidos dos sensores, passou-se então à implementação da lógica de negócio. Para isso, teve-se em consideração não só os dados convertidos no passo anterior como também a arquitetura da base de dados definida na secção 4.3.

Assim, foi criado um objeto *SensorsPeriodInfo* que para além de herdar as propriedades do objeto *SensorInfo* construído com a informação dos dados dos sensores, calcula o tempo desde a última inserção e a variação de estado. Este processo pode observar-se a partir do Excerto de Código 7.

```
SensorInfo sensorInfo = client.getSensorInfo(mockSensorsData);
SensorPeriodInfo sensorPeriodInfo = new SensorPeriodInfo(sensorInfo);
```

Excerto de Código 7 - Herança das propriedades do SensorInfo pelo SensorPeriodInfo

De forma a calcular o tempo desde a última inserção, é realizada uma consulta à base de dados à procura da última linha de dados inserida. Para isso, é usado o padrão *Repository*, onde o *Controller* acede ao método do *SensorsClientRepository* através de uma interface.

Por sua vez, o repositório consulta a base de dados através do método presente no Excerto de Código 8.

```
@Override
public SensorPeriodInfo getPreviousUpdate() {
    SensorPeriodInfo sensors = null;
    try {
        String query = "SELECT * FROM sensors.sensors ORDER BY Timestamp DESC LIMIT 1";

        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        if (rs.next()) {
            Timestamp timestamp = rs.getTimestamp("Timestamp");
            short isWorking = rs.getShort("IsWorking");
            short expectedWorking = rs.getShort("ExpectedWorking");
            short hasDefects = rs.getShort("HasDefect");
            int timeSinceLastUpdate = rs.getInt("TimeSinceLastUpdate");
            short changedState = rs.getShort("ChangedState");

            sensors = new SensorPeriodInfo(timestamp, isWorking, expectedWorking, hasDefects, timeSinceLastUpdate, changedState);
        }

    } catch (SQLException e) {
        System.out.println("Connection error " + e);
    }
    return sensors;
}
```

Excerto de Código 8 - Método de consulta da última inserção na base de dados

A partir da análise ao código em cima, é possível observar que a consulta à base de dados é realizada através da execução de uma *query* SQL definida numa *String*. Para isso, a biblioteca JDBC disponibiliza a interface *Statement* que define o método *executeQuery* para execução de consultas SQL. De seguida, é devolvido um *ResultSet* com o resultado da consulta sendo a sua informação usada para instanciar o objeto *SensorPeriodInfo* que é devolvido pela função.

A partir dos dados dos sensores desse objeto, o *Controller* pede ao *SensorPeriodInfo* atual que calcule a variação do estado em relação ao anterior.

```
public void computeChangedState(short previousIsWorking){
    changedState = (short) (this.getIsWorking() ^ previousIsWorking);
}
```

Excerto de Código 9 - Método que calcula a variação de estado dos sensores

Como é possível observar no excerto em cima, o *SensorPeriodInfo* calcula o *changedState* aplicando uma operação lógica XOR entre os dados dos seus sensores e os da última leitura. Para perceber o motivo do uso da operação XOR é necessário perceber o propósito do atributo *ChangedState*. Como referido no capítulo 4.3, é a partir deste atributo que é possível calcular o

número de peças produzidas, uma vez que a variação de estado, de ativo para desativado, corresponde à criação de uma peça.

De forma a exemplificar o processo criou-se a Tabela 10.

Tabela 10 - Lógica da operação XOR usada no cálculo da variação de estado

	Representação decimal	Representação binária
Leitura Anterior	65533	1111 1111 1111 1101
Leitura Atual	65535	1111 1111 1111 1111
Operação XOR	2	0000 0000 0000 0010

A partir da Tabela 10, é possível observar que entre a leitura anterior e a atual, apenas um dos 16 sensores, representados em binário, mudou de estado. Inicialmente, o sensor encontrava-se ligado (valor do bit a 0) passando a estar desligado na leitura atual (valor do bit a 1). Esta alteração significa que houve a produção de uma peça por parte da máquina conectada a esse sensor.

Desta forma, ao aplicar a operação lógica XOR entre os dois valores, obtém-se um resultado que nos permite saber a cada leitura, que sensores mudaram de estado, deixando esses bits a 1. A maneira como este valor é integrado nos cálculos das métricas, é explicado detalhadamente na secção 5.2.3 aquando da realização desses cálculos.

Por fim, uma vez calculada a mudança de estado, o *SensorPeriodInfo* calcula então o período de tempo desde a última leitura. Como referido na secção 4.3 aquando da definição do modelo de base de dados, este valor é essencial para o cálculo das métricas de disponibilidade e do desempenho.

```
public void computeTimeSinceLastUpdate(Timestamp lastUpdateTimestamp){
    if(lastUpdateTimestamp != null){
        timeSinceLastUpdate = (int) (getTimestamp().getTime() - lastUpdateTimestamp.getTime()) / 1000;
    }
}
```

Excerto de Código 10 - Método que calcula o tempo desde a última inserção

A partir do Excerto de Código 10, compreende-se então que para o cálculo do período de tempo desde a última leitura, são subtraídos aos milissegundos do *timestamp* da leitura atual os

milissegundos do *timestamp* da leitura anterior. Em seguida esse valor é divido por 1000 passando o resultado de milissegundos para segundos.

Todo o processo descrito ao longo deste capítulo encontra-se resumido no método seguinte.

```
public void run(boolean mockSensorsData) {
    SensorInfo sensorInfo = client.getSensorInfo(mockSensorsData);
    SensorPeriodInfo sensorPeriodInfo = new SensorPeriodInfo(sensorInfo);

    SensorPeriodInfo previousInfo = sensorRepository.getPreviousUpdate();

    short previousIsWorking = 0;
    Timestamp previousTimestamp = null;

    if (previousInfo != null) {
        previousIsWorking = previousInfo.getIsWorking();
        previousTimestamp = previousInfo.getTimestamp();
    }

    sensorPeriodInfo.computeChangedState(previousIsWorking);
    sensorPeriodInfo.computeTimeSinceLastUpdate(previousTimestamp);

    if (sensorPeriodInfo.getChangedState() != 0 && sensorPeriodInfo.getTimeSinceLastUpdate() >= 0) {
        sensorRepository.insert(sensorPeriodInfo);
    }
}
```

Excerto de Código 11 - Método com a lógica do negócio

Uma vez concluída toda a construção da lógica de negócio é realizada uma validação antes do objeto final ser inserido na base de dados. O objetivo passa por averiguar se ocorreram alterações nos estados dos sensores para que não haja a inserção de linhas na BD com a mesma informação.

5.1.4 Persistência dos dados na base de dados

Uma vez validado o objeto, este é então inserido na base de dados através do padrão *Repository*. Assim, o *Controller* usa uma interface para aceder ao método *insert* da classe *SensorsRepository* que fica responsável pelo tratamento dos dados de forma a proceder à sua inserção.

```java
@Override
public void insert(SensorPeriodInfo sensors) {
    if (sensors == null) {
        return;
    }

    try {
        String query =
        "INSERT INTO sensors.sensors (IsWorking, Timestamp, ExpectedWorking, TimeSinceLastUpdate, ChangedState, HasDefect) VALUES (?, ?, ?, ?, ?, ?)";

        PreparedStatement preparedStmt = connection.prepareStatement(query);
        preparedStmt.setShort(1, sensors.getIsWorking());
        preparedStmt.setTimestamp(2, sensors.getTimestamp());
        preparedStmt.setShort(3, sensors.getExpectedWorking());
        preparedStmt.setInt(4, sensors.getTimeSinceLastUpdate());
        preparedStmt.setShort(5, sensors.getChangedState());
        preparedStmt.setShort(6, sensors.getHasDefect());
        preparedStmt.execute();

    } catch (SQLException e) {
        System.out.println("Connection error " + e);
    }
}
```

Excerto de Código 12 - Método que insere a informação na base de dados

Para a inserção na base de dados, é usada a interface *Statement* disponibilizada pela biblioteca JDBC que define o método *prepareStatement* para execução do comando SQL. Uma vez criado o *Statement* com a *query* pretendida são passados os valores dos atributos do objeto *SensorPeriodInfo* pela ordem das colunas definida na *query*. Por fim, é executado o *Statement* sendo os dados inseridos na base de dados.

5.2 UC02: Visualizar métricas

Depois de recolhidos e armazenados os dados, é necessário implementar o módulo de visualização. Para isso começou-se por configurar o Grafana.

5.2.1 Configuração do Grafana

Para a comunicação com a base de dados, é necessário especificar os detalhes da conexão.

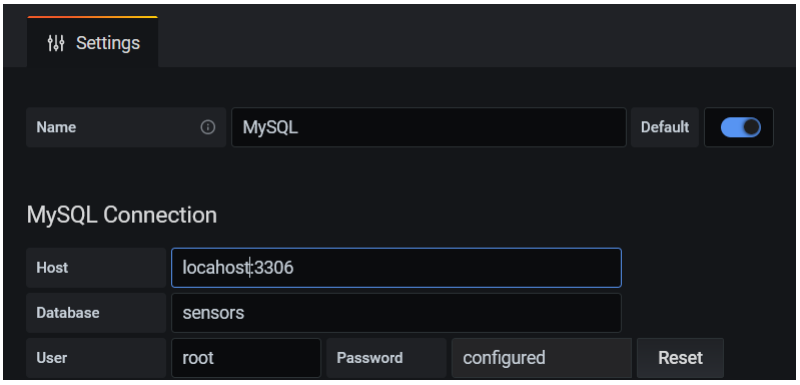

Figura 30 - Configuração do Grafana

Como é possível observar na Figura 30, é necessário indicar o endereço IP e a porta do servidor que aloja a BD. De seguida, escolheu-se a tabela onde os dados se encontram e por fim guardaram-se as configurações e estabeleceu-se a conexão.

Figura 31 - Configuração do Grafana: notificação de sucesso

5.2.2 Construção do painel de visualização do Grafana

Assim que a base de dados foi adicionada ao Grafana, passou-se então à construção do painel com vista à apresentação das métricas ao utilizador.

O primeiro passo consistiu na criação de um painel geral para a visualização das métricas.

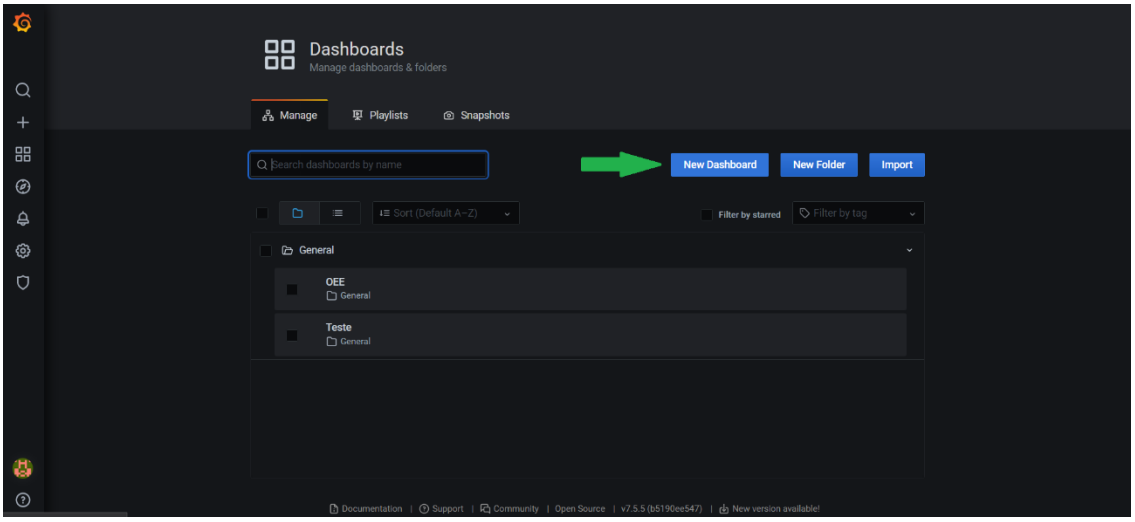

Figura 32 - Construção do painel do Grafana: passo 1

Depois de criado o painel, adicionaram-se subsecções para inserir diferentes gráficos para a visualização de cada métrica.

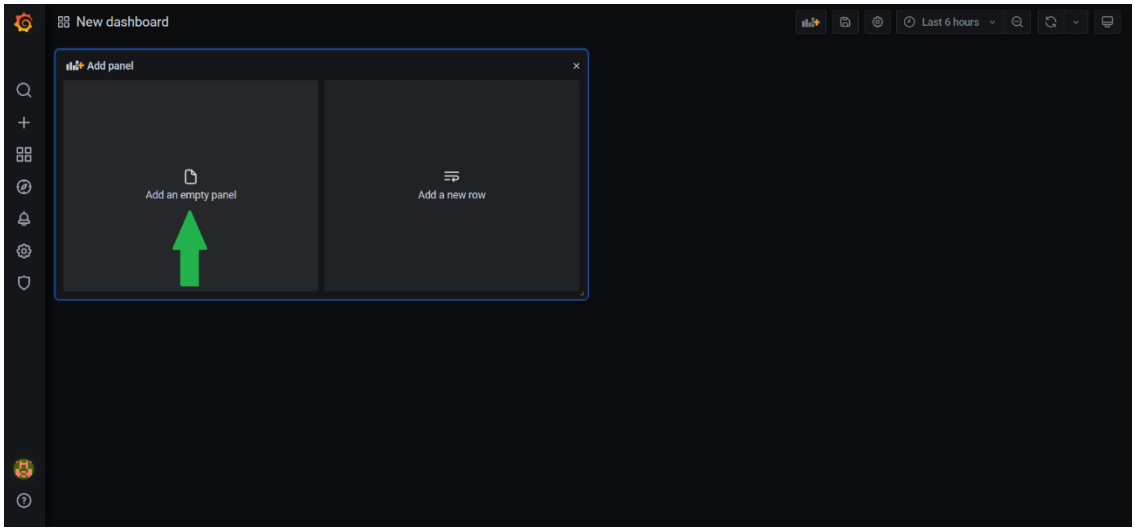

Figura 33 - Construção do painel do Grafana: passo 2

De seguida, atribuiu-se um nome a cada subsecção e optou-se pela seleção de gráficos de barras para a visualização de cada métrica.

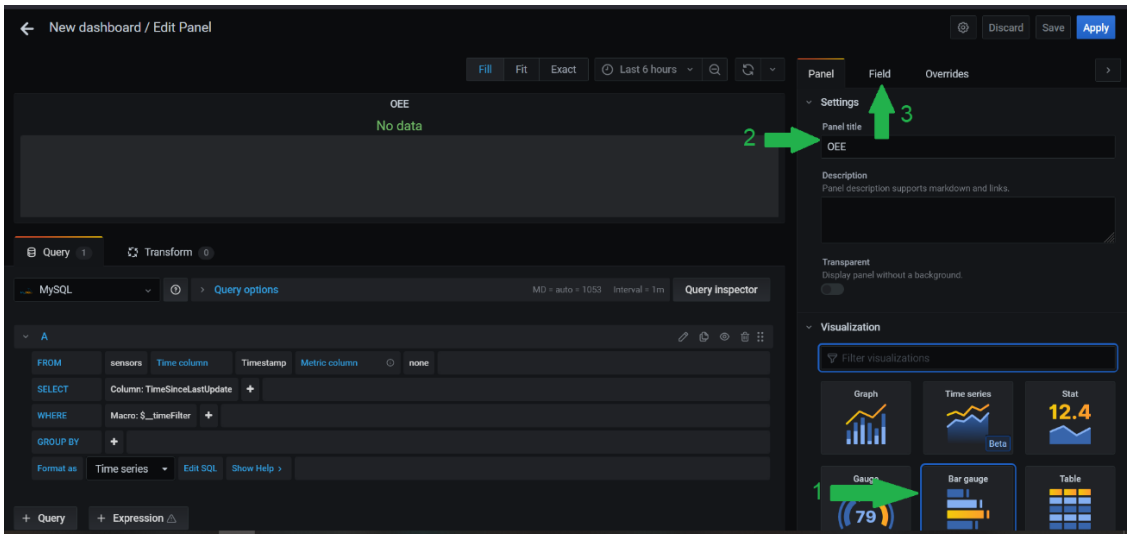

Figura 34 - Construção do painel do Grafana: passo 3

Depois de escolhidos os gráficos, uma vez que as métricas são calculadas em percentagem, definiu-se então a percentagem como unidade de mostragem. Além disso, adicionaram-se estilos às barras de mostragem e guardaram-se todas as alterações.

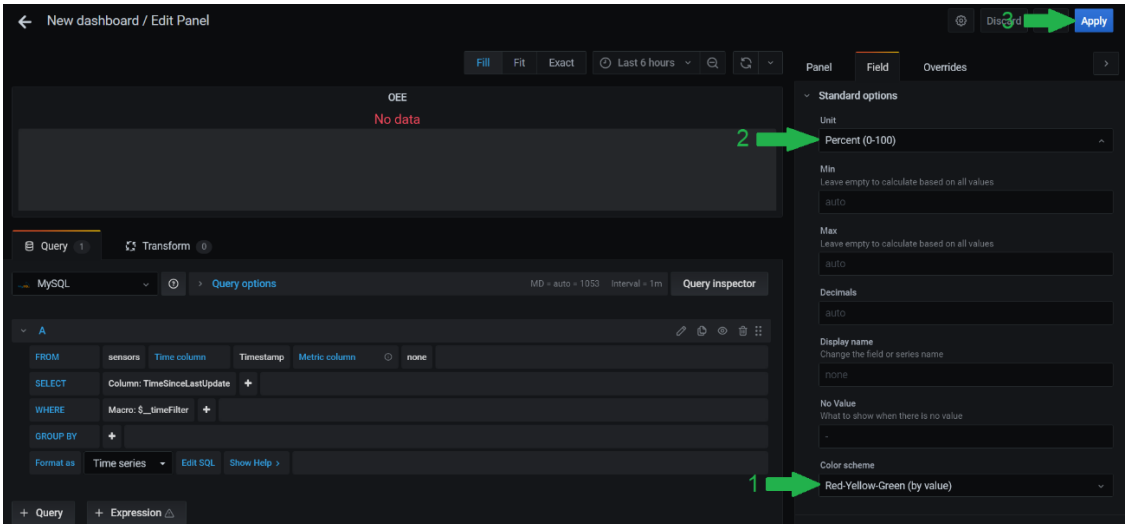

Figura 35 - Construção do painel do Grafana: passo 4

Por fim, obteve-se o painel presente na Figura 36.

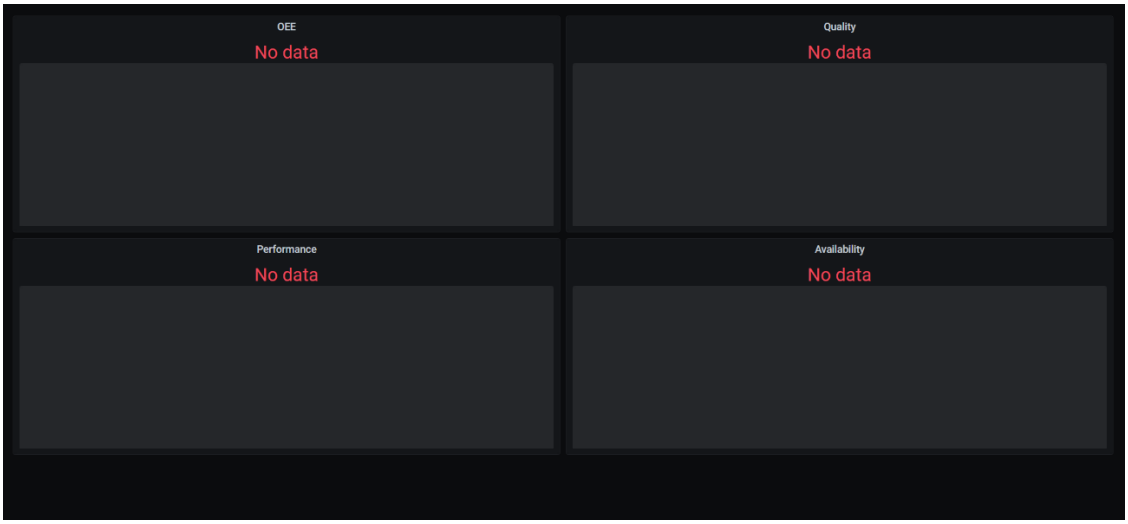

Figura 36 - Construção do painel do Grafana: passo 5

Uma vez que ainda não foram construídas as *queries* para o cálculo das métricas, os gráficos encontram-se para já sem informação.

5.2.3 Apresentação das métricas

Depois de construído o painel, elaboraram-se então as *queries* SQL para o cálculo e apresentação dos resultados de cada uma das métricas: Disponibilidade, Desempenho, Qualidade e OEE. Para a explicação dos cálculos realizados são usados neste capítulo excertos das *queries* que se encontram representadas na íntegra nos anexos.

5.2.3.1 Disponibilidade

Para o cálculo da disponibilidade, é necessário calcular o tempo em que a máquina se encontra a produzir e o tempo programado para produção. Para isso, é preciso saber quando é que os sensores se encontram ativos e também se era suposto estarem ativos nesse dado momento.

Através da coluna *IsWorking* da base de dados, é possível determinar o estado dos sensores, ou seja, saber se estes se encontram ativos. Já a partir da tabela *ExpectedActive*, consegue-se saber se era expectável o sensor estar ativo nesse dado momento. Em relação ao tempo, é dado através da coluna *TimeSinceLastUpdate*.

Assim sendo, elaborou-se então a *query* do Excerto de Código 13.

```
SELECT
    1 AS time_sec,
    COALESCE(Up1 / ExpUp1 * 100, 0) AS S1,
    COALESCE(Up2 / ExpUp2 * 100, 0) AS S2,
    ...
    ...
FROM
(
    SELECT
        SUM((ExpectedWorking&1 != 1) * TimeSinceLastUpdate) AS ExpUp1,
        SUM((IsWorking&1 != 1) * TimeSinceLastUpdate) AS Up1,

        SUM((ExpectedWorking&2 != 2) * TimeSinceLastUpdate) AS ExpUp2,
        SUM((IsWorking&2 != 2) * TimeSinceLastUpdate) AS Up2,
        ...
        ...
    FROM sensors
    WHERE $__timeFilter(Timestamp)
) availabilityMetrics
```

Excerto de Código 13 – Parte da query do cálculo da métrica da disponibilidade

A partir do excerto acima, verifica-se que para cada um dos 16 sensores, é calculada a soma dos tempos em que era expectável estar ativo bem como a soma dos tempos em que esteve a trabalhar.

De modo a separar o cálculo da métrica por sensor, é realizada a operação lógica AND com exponenciais de 2, que se encontra explicada na Tabela 11.

Tabela 11 - Lógica da query do cálculo da métrica da disponibilidade

ExpectedWorking	1111 1111 1111 1111
1º sensor = 2^0 = 1	0000 0000 0000 0001
ExpectedWorking&1	0000 0000 0000 0001
ExpectedWorking&1 != 1	FALSO

Como é possível observar a partir da tabela acima, através da operação lógica AND entre o *ExpectedWorking* e os exponenciais de 2, consegue-se saber o estado expectável de cada sensor. Neste caso, como se querem apenas os tempos em que era expectável estar ativo, são somados apenas aqueles em que o valor do bit nessa posição for diferente de 1. Para o tempo de produção (*IsWorking*), a lógica aplicada é exatamente a mesma.

5.2.3.2 Desempenho

Já para o cálculo do desempenho, é necessário calcular o tempo em que o equipamento foi projetado para funcionar, o número de peças produzidas num dado intervalo de tempo e o tempo de produção. Assim, é preciso determinar quando há mudanças no estado dos sensores e também se os sensores se encontram ativos nesse dado momento. Já o tempo projetado, é uma constante definida pelo cliente.

Desta forma, criou-se a seguinte *query*.

```
SELECT
    1 AS time_sec,
    COALESCE(1.9 * NumPieces1 / Up1 * 100, 0) AS S1,
    COALESCE(1.9 * NumPieces2 / Up2 * 100, 0) AS S2,
    (...)
    (...)
FROM
(
    SELECT
        SUM(1 & ChangedState & IsWorking) AS NumPieces1,
        SUM((IsWorking&1 != 1) * TimeSinceLastUpdate) AS Up1,

        SUM((2 & ChangedState & IsWorking) = 2) AS NumPieces2,
        SUM((IsWorking&2 != 2) * TimeSinceLastUpdate) AS Up2,
        (...)
        (...)
    FROM sensors
    WHERE $__timeFilter(Timestamp)
) performanceMetrics
```

Excerto de Código 14 – Parte da query do cálculo da métrica da performance

É possível observar no excerto acima, que para cada um dos 16 sensores, é calculada a métrica do desempenho, multiplicando o valor do tempo projetado para trabalhar pelo número de peças produzidas, dividindo o resultado pelo tempo em que esteve ativo e multiplicando por 100 para obter a percentagem. Esta fórmula usa o valor da soma do número de peças e da soma dos tempos em que esteve a trabalhar.

De modo a calcular o número de peças, é realizada a operação lógica AND entre os exponenciais de 2, a mudança de estado (*ChangedState*) e o estado dos sensores (*IsWorking*), que se encontra explicada na Tabela 12.

Tabela 12 - Lógica da query do cálculo da métrica da performance

Valor em binário do 1º sensor = 2^0 = 1	0000 0000 0000 0001
Valor em binário do *ChangedState*	1111 1111 1111 1111
IsWorking	1111 1111 1111 1110
1 & *ChangedState*	0000 0000 0000 0001
1 & *ChangedState* & *IsWorking*	0000 0000 0000 0000
(1 & *ChangedState* & *IsWorking*) = 1	FALSO

A partir da tabela, compreende-se que através da operação lógica AND entre os exponenciais de 2 e o *ChangedState*, consegue-se saber se houve variação de estado num dado sensor. Através da segunda operação lógica AND com o valor *IsWorking*, no sensor 1, como o bit é 0, conclui-se que o sensor continua ligado, sendo o resultado da operação lógica 0. Isto significa que o sensor ainda não acabou de produzir a peça. Se o valor do bit fosse 1, significaria que o sensor já se tinha desligado, ou seja, tinha acabado de produzir uma peça. Neste caso, o valor do bit seria 1 e a condição verdadeira. Para o tempo de produção (*IsWorking*), foi aplicada a mesma lógica aplicada aquando do cálculo da disponibilidade.

5.2.3.3 Qualidade

Para o cálculo do desempenho, é necessário calcular o número de peças sem defeito e o número de peças produzidas num dado intervalo de tempo. Assim, é preciso determinar quando há mudanças no estado dos sensores e perceber se as peças produzidas têm defeitos.

Com base nesses requisitos, elaborou-se a *query* seguinte.

```
SELECT
    1 AS time_sec,
    COALESCE(100 - NumPiecesDefect1 / NumPieces1 * 100, 0) AS S1,
    COALESCE(100 - NumPiecesDefect2 / NumPieces2 * 100, 0) AS S2,
    (...)
    (...)
FROM
(
    SELECT
        SUM(1 & ChangedState & IsWorking & HasDefect) AS NumPiecesDefect1,
        SUM((1  & ChangedState & IsWorking) = 1 ) AS NumPieces1,

        SUM((2 & ChangedState & IsWorking & HasDefect) = 2) AS NumPiecesDefect2,
        SUM((2  & ChangedState & IsWorking) = 2 ) AS NumPieces2,
        (...)
        (...)
    FROM sensors
    WHERE $__timeFilter(Timestamp)
) qualityMetrics
```

Excerto de Código 15 – Parte da query do cálculo da métrica da qualidade

Compreende-se a partir do excerto acima, que para cada um dos 16 sensores, é calculada a métrica da qualidade, através da divisão do número de peças produzidas sem defeitos pelo número total de peças, multiplicando por 100 para obter a percentagem. Esta fórmula usa o valor da soma do número de peças com defeito e da soma do número total de peças.

De modo a calcular o número de peças com defeito, é usada a operação lógica AND entre os exponenciais de 2, a mudança de estado (*ChangedState*) e o valor dos sensores com defeito (*HasDefect*), que se encontra detalhada na Tabela 13.

Tabela 13 - *Lógica da query do cálculo da métrica da qualidade*

1º sensor = 2^0 = 1	0000 0000 0000 0001
ChangedState	1111 1111 1111 1111
IsWorking	1111 1111 1111 1111
HasDefect	0000 0000 0000 0001
1 & *ChangedState*	0000 0000 0000 0001
1 & *ChangedState* & *IsWorking*	0000 0000 0000 0001
1 & *ChangedState* & *IsWorking* & *HasDefect*	0000 0000 0000 0001
(1 & *ChangedState* & *IsWorking* & *HasDefect*) = 1	VERDADEIRO

Através da operação lógica AND entre os exponenciais de 2, o *ChangedState* e o *IsWorking*, no caso do primeiro sensor representado na Tabela 13, o valor do bit passaria a 1, o que significa que o sensor teria acabado de produzir uma peça. Ao aplicar a última operação lógica AND com o *HasDefect*, o valor do bit mantém-se a 1 o que significa que a peça teria sido produzida com defeito. Caso o valor do bit do *HasDefect* fosse 0, a operação lógica daria 0, o que tornaria a condição falsa, não sendo por isso somada.

5.2.3.4 OEE

Por fim, para o cálculo das métrica OEE, foram utilizados os cálculos das três métricas anteriores: disponibilidade, desempenho e qualidade.

```
SELECT
    1 AS time_sec,
    COALESCE((Up1 / ExpUp1) * (1.9 * NumPieces1 /  Up1) * NumPiecesDefect1 / NumPieces1, 0) * 100 AS S1,
    COALESCE((Up2 / ExpUp2) * (1.9 * NumPieces2 /  Up2) * NumPiecesDefect2 / NumPieces2, 0) * 100 AS S2,
    (...)
    (...)
FROM
(
    SELECT
        SUM(1 & ChangedState & IsWorking & HasDefect) AS NumPiecesDefect1,
        SUM((1  & ChangedState & IsWorking) = 1 ) AS NumPieces1,
        SUM((ExpectedWorking&1 != 1) * TimeSinceLastUpdate) AS ExpUp1,
        SUM((IsWorking&1 != 1) * TimeSinceLastUpdate) AS Up1,

        SUM((2 & ChangedState & IsWorking & HasDefect) = 2) AS NumPiecesDefect2,
        SUM((2  & ChangedState & IsWorking) = 2 ) AS NumPieces2,
        SUM((ExpectedWorking&2 != 2) * TimeSinceLastUpdate) AS ExpUp2,
        SUM((IsWorking&2 != 2) * TimeSinceLastUpdate) AS Up2,
        (...)
        (...)
    FROM sensors
    WHERE $__timeFilter(Timestamp)
) oeeMetrics
```

Excerto de Código 16 – Parte da query do cálculo da métrica OEE

A partir do Excerto de Código 16, verifica-se que são multiplicados os valores das métricas de disponibilidade com as de desempenho e com as de qualidade, sendo depois o resultado multiplicado por 100 para obter a percentagem.

Os resultados de cada métrica, são obtidos através das mesmas operações lógicas realizadas para o cálculo de cada uma das métricas nas secções 5.2.3.1, 5.2.3.2 e 5.2.3.3.

Por fim, depois de desenvolvidas as *queries* e integradas no Grafana, são apresentadas as métricas em tempo real ao utilizador como se pode observar na Figura 37.

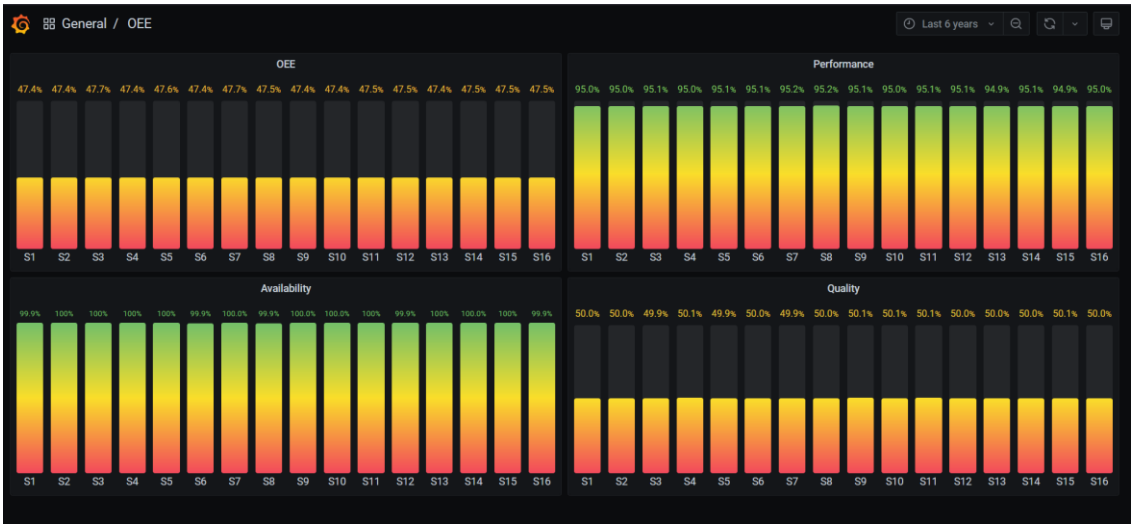

Figura 37 - Painel do Grafana com as métricas

Desta forma, através de cada um dos 16 sensores, é possível acompanhar em tempo real a disponibilidade, performance, qualidade e a eficácia geral de cada máquina (OEE) associada. Além disso, é possível definir diferentes períodos de tempo para o cálculo das métricas, oferecendo assim uma solução bastante intuitiva e fácil de usar.

6 Balanço

Aquando do Estado da Arte, o sistema de monitorização da OEE Technologies encontrava-se estagnado face ao aumento da competitividade do mercado. Os produtos que a empresa oferecia perdiam valor face à existência de soluções concorrentes mais completas e acessíveis.

O seu sistema limitava-se a gerar relatórios de monitorização para os clientes. Para os utilizadores menos acostumados com novas tecnologias, o processo de instalação e configuração era demasiado complexo e a interface gráfica pouco intuitiva.

Desta forma, a empresa necessitava de novas soluções, menos complexas e mais acessíveis, que acrescentassem valor ao sistema atual. E foi neste contexto que surgiu a ideia para a integração de uma funcionalidade que permitisse a monitorização em tempo real. A empresa pensou numa funcionalidade que fosse multiplataforma e que oferecesse aos clientes a possibilidade de monitorizar as suas empresas em tempo real.

Nesse sentido, este projeto surgiu para a concretização dessa ideia, baseando-se nos requisitos definidos pela OEE Technologies e oferecendo todas as funcionalidades pretendidas.

O estudo prévio do seu sistema de monitorização e o estabelecimento de critérios que tivessem em conta os requisitos do mesmo permitiram o desenvolvimento de uma solução compatível e de fácil integração.

Em termos de tecnologias usadas, as escolhas revelaram-se de um modo geral acertadas, nomeadamente o uso do Grafana para a apresentação das métricas. A plataforma revelou-se

bastante intuitiva e fácil de usar, sendo que a sua escolha recolheu uma opinião bastante positiva por parte da empresa.

Quanto às bases de dados usadas, ambas as soluções revelaram ser boas opções, não só pelos resultados de desempenho apresentados, mas também pelos estudos comparativos que permitiram fazer. A seleção da BD em vigor no sistema da empresa (MariaDB), em conjunto com o MonetDB permitiu estabelecer diferentes pontos comparativos do desempenho entre ambas. Os dados deste estudo podem assim ser usados pela empresa para análise de uma eventual mudança do seu sistema de armazenamento.

Além disso, através de toda uma análise exaustiva dos requisitos do sistema, conseguiu-se ainda arquiteturar a base de dados de forma a otimizar ao máximo o espaço ocupado pela mesma. Para isso, em muito contribuíram a análise dos dados recebidos, o estudo dos dados necessários para o cálculo das métricas e toda a conversão e otimização realizados pelo módulo de captura. Uma vez concluído o projeto, a empresa tem assim ao seu dispor todo um sistema de monitorização em tempo-real que acrescenta valor ao seu sistema.

7 Teste da Solução

É importante que o sistema desenvolvido seja escalável e robusto para suportar o volume de dados e a utilização desejadas pelo cliente. As interfaces têm de ser desenvolvidas de acordo com os requisitos e objetivos iniciais.

Deste modo, para avaliar a solução têm de ser realizados diferentes tipos de testes:

- testes unitários, para garantir o funcionamento do código a um nível mais atómico;
- testes de integração, que assegurem a correta interação dos vários sistemas da empresa (bases de dados, sensores);
- testes de performance, que garantam que o sistema desenvolvido tem um bom desempenho;
- teste de usabilidade, através de um inquérito que avalie a experiência de diferentes utilizadores ao usar o Grafana.

7.1 Testes unitários

De forma a garantir que todo o módulo de captura funciona conforme o esperado, foram então realizados diferentes testes unitários.

Inicialmente, começou-se por testar o método que converte a informação recolhida dos sensores, ilustrado no Excerto de Código 17, de forma a garantir que os dados eram devidamente convertidos.

```
@Test
public void convertSensorData() {
    System.out.println("getSensorInfo");
    SensorsClientService service = new SensorsClientService();

    byte[] bytes = new byte[]{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B'};
    short[] expResut = new short[]{(short) 0x0123, (short) 0x4567, (short) 0x89AB};
    int numBytesToRead = 4;
    for (int i = 0; i < expResut.length; i++) {
        assertEquals(expResut[i], service.convertSensorData(bytes, i * numBytesToRead));
    }
}
```

Excerto de Código 17 - Teste da conversão dos dados dos sensores

De seguida, através do teste presente no Excerto de Código 18 verificou-se se o objeto modelo era construído com toda a informação proveniente dos sensores, de forma a garantir que os dados guardados na BD estão corretos.

```
@Test
public void testGetSensorInfo() throws Exception{
    System.out.println("connect");

    byte[] bytes = new byte[]{'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B'};

    SensorsClientService sensorsClientService = spy(SensorsClientService.class);

    when(sensorsClientService.communicateWithSensorMachine()).thenReturn(bytes);

    SensorInfo sensorInfo = sensorsClientService.getSensorInfo();
    assertNotNull(sensorInfo);

    assertEquals((short) 0x0123, sensorInfo.getIsWorking());
    assertEquals((short) 0x4567, sensorInfo.getExpectedWorking());
    assertEquals((short) 0x89AB, sensorInfo.getHasDefect());
}
```

Excerto de Código 18 - Teste que verifica se o objeto SensorInfo foi contruído com os dados dos sensores

Uma vez testada a informação recebida dos sensores, achou-se relevante verificar os cálculos dos dados que são usados nos algoritmos das métricas, uma vez que é essencial que os mesmos estejam corretos para que não haja erros com prejuízo para o utilizador.

```
@Test
public void testComputeTimeSinceLastUpdate() {
    System.out.println("computeTimeSinceLastUpdate");
    Timestamp before = Timestamp.valueOf("2021-06-06 00:00:00");
    Timestamp after = Timestamp.valueOf("2021-06-06 00:02:05");
    int expectedDiff = 125;

    SensorInfo s = new SensorInfo(after, (short) 0, (short) 0, (short) 0);
    SensorPeriodInfo sp = new SensorPeriodInfo(s);

    sp.computeTimeSinceLastUpdate(before);

    assertEquals(expectedDiff, sp.getTimeSinceLastUpdate());
}
```

Excerto de Código 19 - Teste do cálculo do período de tempo desde a última inserção na BD

A partir do Excerto de Código 19 tem se então o primeiro teste que verifica se o período de tempo entre a última leitura de dados e a atual é bem calculado. Já no Excerto de Código 20, é testado o cálculo da variação do estado entre a leitura atual e a anterior.

```
@Test
public void testComputeChangedState() {
    System.out.println("computeChangedState");
    short before = 0x4FF5;
    short after  = 0x22FF;
    int expectedDiff = 0x6D0A;

    Timestamp timestamp = new Timestamp(0);
    SensorInfo s = new SensorInfo(timestamp, after, (short) 0, (short) 0);
    SensorPeriodInfo sp = new SensorPeriodInfo(s);

    sp.computeChangedState(before);

    assertEquals(expectedDiff, sp.getChangedState());
}
```

Excerto de Código 20 - Teste do cálculo da variação de estado dos sensores

Relativamente à cobertura geral do projeto, a mesma estabeleceu-se pelos 70%, como é possível comprovar pelo relatório de cobertura de testes realizado pela ferramenta do SonarQube para esse efeito e presente na Figura 38.

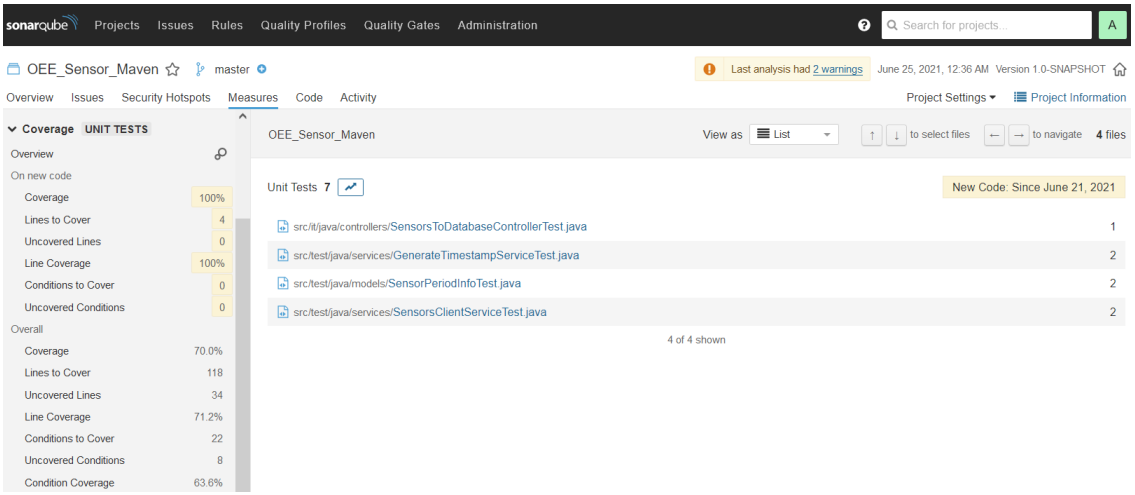

Figura 38 - Excerto do relatório de cobertura do código

De realçar que o teste de integração se encontra entre os testes unitários porque o SonarQube não tem métricas dedicadas para cada tipo de teste, convergindo assim todos os módulos de teste num único relatório.

7.2 Testes de integração

Uma vez concluídos os testes unitários, passou-se então à realização dos testes de integração. Nesta fase, combinaram-se os diferentes componentes que constituem o módulo de captura, de forma a averiguar o seu funcionamento como um todo.

Para isso, realizou-se o teste de integração presente no Excerto de Código 21 que combina todos os componentes do sistema.

```
@Test
public void testRun() {
    System.out.println("run");
    SensorPeriodInfo atualInfo = new SensorPeriodInfo(
                                                    Timestamp.valueOf("2021-09-18 23:23:26"),
                                                    (short) 1,
                                                    (short) 1,
                                                    (short) 1,
                                                    (short) 1,
                                                    (short) 1
                                                    );

    SensorsRepository sensorsRepository = spy(SensorsRepository.class);

    SensorsClientService sensorsClientService = spy(SensorsClientService.class);
    when(sensorsClientService.getSensorInfo()).thenReturn(atualInfo);

    SensorsToDatabaseController controller = new SensorsToDatabaseController();
    controller.setClient(sensorsClientService);
    controller.setSensorRepository(sensorsRepository);
    controller.run();
    SensorPeriodInfo result = sensorsRepository.getPreviousUpdate();

    assertEquals(atualInfo.getTimestamp(), result.getTimestamp());
    assertEquals(atualInfo.getIsWorking(), result.getIsWorking());
    assertEquals(atualInfo.getExpectedWorking(), result.getExpectedWorking());
    assertEquals(atualInfo.getHasDefect(), result.getHasDefect());
}
```

Excerto de Código 21 - Teste de integração

Para a sua concretização, simulou-se a interação com a API da empresa através da criação de Mocks sendo verificado no final se a informação recolhida correspondia à informação guardada na base de dados.

7.3 Testes de performance

O MariaDB é um sistema de base de dados orientado à linha, o que torna as inserções relativamente rápidas, mas obriga a que toda a tabela seja carregada em memória caso se pretenda fazer uma operação de agregação que utilize apenas uma coluna.

Como tal, torna-se interessante comparar o desempenho com um sistema de bases de dados orientado à coluna, como é o caso do MonetDB. Apesar desta arquitetura resultar em inserções mais lentas, como se verificou aquando da inserção de dados usados neste teste, a velocidade é ainda mais do que suficiente para corresponder às necessidades da aplicação. Assim, resolveu-se realizar testes de performance para comparar a performance de ambas as bases de dados.

Num ambiente de execução, é impossível ter um ambiente completamente isolado. A execução de várias medições resulta em variações temporais que podem ser explicadas por interrupções de outros programas no sistema, temperatura do CPU, entre outros.

Nas medições de performance é uma boa prática ignorar o impacto das outras aplicações bem como apresentar medições que possam ser replicadas no mesmo ambiente de teste (hardware e software).

Desta forma, há dois meios que comumente são usados em cálculos de performance, a média e a mediana. No entanto, para os testes de performance realizados neste projeto essas hipóteses foram rejeitadas. Por um lado, a média não serve, pois, basta haver um valor mais desnivelado para influenciar o seu cálculo. Já a mediana, sendo melhor métrica que a média, uma vez que ignora os *outliers* (valores que fogem ao padrão de uma amostra), continua a não apresentar o melhor tempo de execução alcançável.

Por isso, utilizou-se então o algoritmo K Best, que para além de eliminar os *outliers*, determina o melhor tempo alcançável.

Para a sua execução, foi necessário inicialmente preparar um ambiente de execução, sendo utilizado um computador portátil com as especificações presentes na Figura 39.

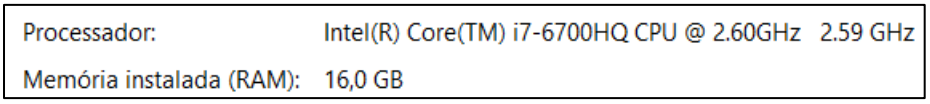

Figura 39 - Especificações do ambiente de execução

Uma vez preparado o ambiente de execução, e executado o algoritmo, construiu-se então o gráfico ilustrado na Figura 40.

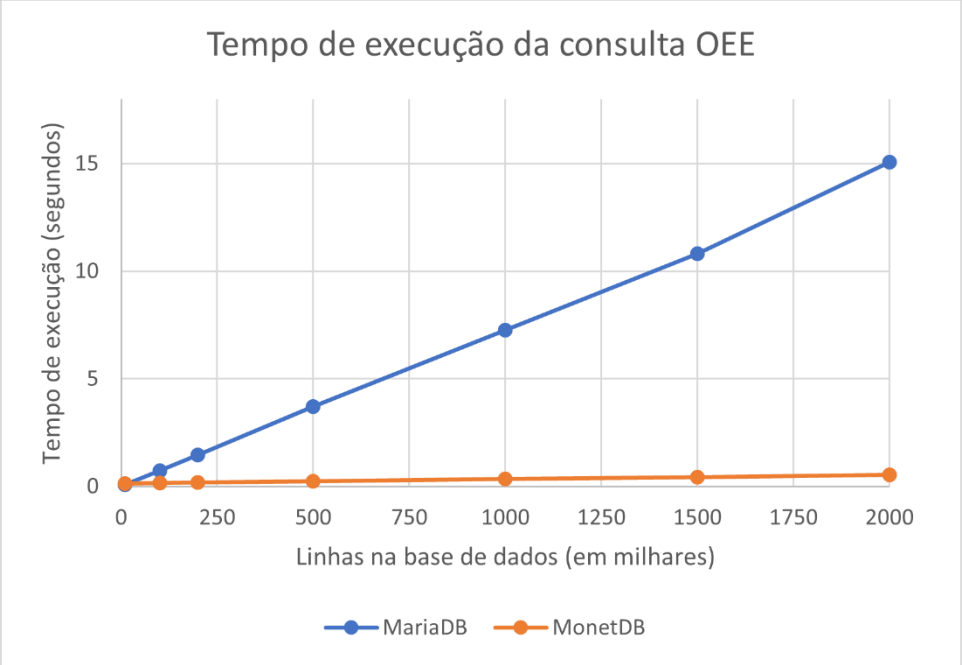

Figura 40 - Comparação do tempo de execução da consulta OEE

É possível verificar que apesar de o tempo de execução aumentar em ambas com o acumular de dados, este aumento é consideravelmente maior no MariaDB. Por isso, a partir deste gráfico é possível extrapolar com algum rigor o tempo de execução da *query* para qualquer volume de dados que caiba em memória.

Para além do gráfico da Figura 40, construiu-se também um gráfico para avaliar o *speedup.* Esta grandeza permite avaliar o desempenho relativo do MonetDB em relação ao MariaDB.

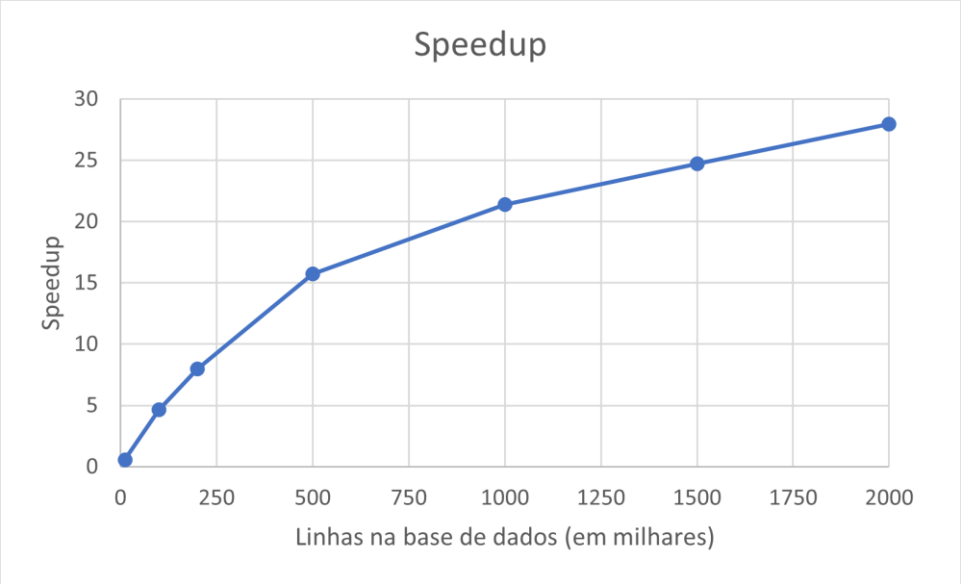

Figura 41 - Gráfico do Speedup

A partir da análise do gráfico da Figura 41, é possível verificar que quanto maior é o número de dados, maior é a vantagem do MonetDB em relação ao MariaDB, conseguindo ser cerca de 28 vezes mais rápido para um volume de dados de 2 milhões de linhas.

Desta forma, conclui-se que com o MonetDB as consultas podem ser realizadas sobre um volume potencialmente infinito de dados. Nesse sentido, quanto mais rápidas as consultas, maior o período que pode ser consultado em tempo razoável aquando da visualização das métricas OEE.

7.4 Teste de usabilidade

De forma a avaliar o sistema desenvolvido realizou-se um teste à usabilidade do Grafana para averiguar a dificuldade de uso do mesmo.

Assim, foi elaborado um inquérito na empresa onde se avaliou a eficácia, eficiência, rigor, desempenho e acessibilidade do sistema, bem como a satisfação geral dos utilizadores. Para a avaliação desses seis parâmetros foram formuladas seis questões. Em cada uma delas há cinco possibilidades de resposta que permitem identificar eventuais problemas e possibilidades de melhoria.

Para um teste de usabilidade, por mais que os inquiridos tenham visões diferentes, os seus comportamentos assemelham-se. Desta forma, a partir de um dado número de inquiridos os resultados tornam-se repetitivos (Vieira, 2019), daí a escolha de apenas doze programadores seniores da empresa. A sua escolha assenta no facto de terem um conhecimento profundo do sistema atual da empresa e das reais necessidades dos clientes, pelo que a sua opinião seria importante nesse sentido. Além disso, tratando-se da possível integração de uma nova solução, é relevante perceber as dificuldades que os mesmo teriam aquando do seu uso.

7.4.1 Eficácia

De forma a averiguar se os utilizadores conseguem utilizar todas as funcionalidades do sistema, é importante que o mesmo seja eficaz e facilmente compreendido. Por isso questionaram-se os utilizadores sobre o quão fácil foi a compreensão do mesmo.

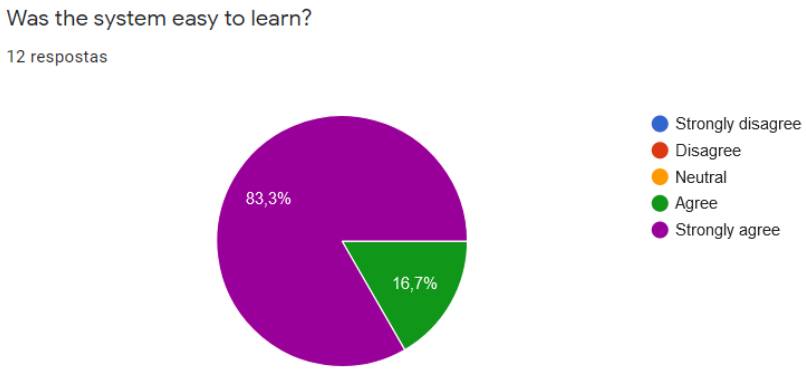

Figura 42 - Resultados do inquérito: 1 - Foi fácil compreender o modo de funcionamento do sistema?

A partir dos dados da Figura 42, compreende-se que todos os utilizadores denotaram extrema facilidade em compreender o sistema pelo que se conclui que o mesmo é eficaz.

7.4.2 Eficiência

Além de eficaz, é importante também que o sistema seja eficiente, de forma que os utilizadores consigam concluir as tarefas no menor período de tempo. Assim, é importante que o fluxo de navegação para a execução de uma dada tarefa seja o mais simples e curto possível. Além disso, importa que todo o desenho da interface seja organizado e pensado numa perspetiva de utilizador, usando botões com informação sugestiva para cada ação.

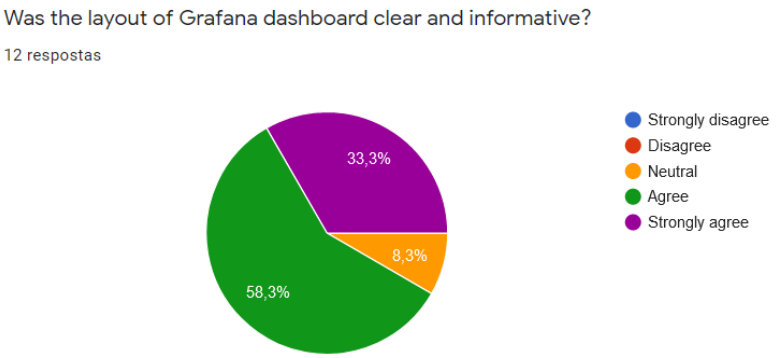

Figura 43 - Resultados do inquérito: 2 - O layout da dashboard do Grafana foi claro e informativo?

Com base nos resultados ilustrados Figura 43, verifica-se que 8.3% dos inquiridos revelaram alguma dificuldade em executar determinadas ações. Esta dificuldade pode ser justificada pelo facto de em alguns modelos de telemóvel a responsividade da *dashboard* do Grafana não ser a melhor. No entanto, e pese embora esse problema, para 58.3% dos inquiridos o *layout* da *dahsboard* foi claro e informativo.

7.4.3 Precisão

É relevante também que toda a informação disponibilizada pelo sistema esteja correta e atualizada para evitar incoerências e não induzir o utilizador em erro. Deste modo, recolheu-se a opinião dos mesmos em relação a este aspeto.

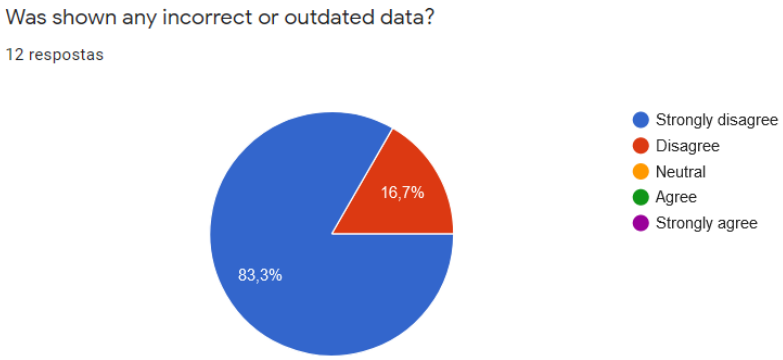

Figura 44 - Resultados do inquérito: 3 - Foi mostrada informação incorreta ou desatualizada?

A partir dos resultados obtidos na Figura 44, verifica-se que os inquiridos não encontraram qualquer incorreção na informação pelo que se pode constatar que o sistema é preciso.

7.4.4 Desempenho

Em relação ao desempenho, uma vez que se trata de um sistema de monitorização em tempo-real, é importante que a solução apresente as métricas de forma contínua e sem atrasos. Desta forma, um mau desempenho afeta diretamente a usabilidade do sistema. Como tal, procurou-se saber a opinião dos utilizadores sobre este aspeto.

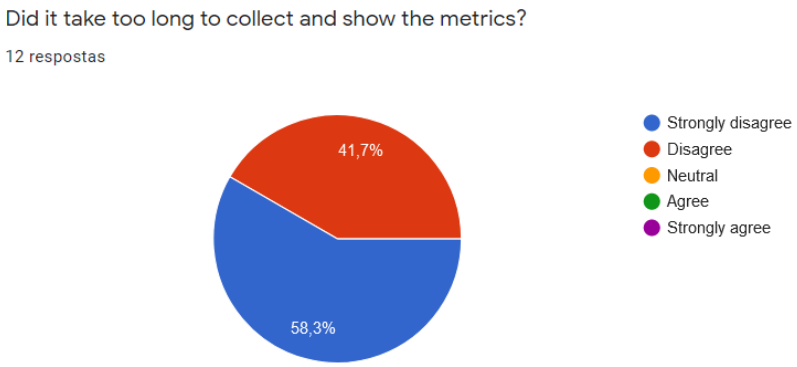

Figura 45 - Resultados do inquérito: 4 - A captura e visualização das métricas foi demorada?

Através da Figura 45, constata-se que de um modo geral os inquiridos não encontraram quaisquer atrasos significativos na apresentação das métricas.

7.4.5 Acessibilidade

De seguida, procurou-se saber se os inquiridos tiveram algum tipo de dificuldade no uso do sistema. Este aspeto tem extrema importância, uma vez que a existência de um sistema complexo poderia dificultar o seu uso e afetar negativamente a usabilidade da solução.

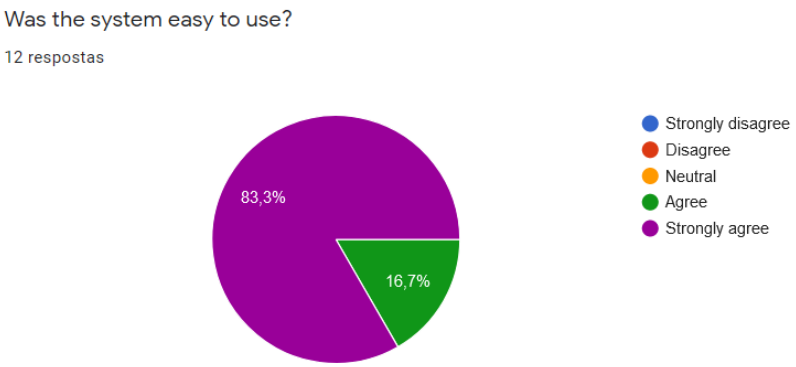

Figura 46 - Resultados do inquérito: 5 - O sistema foi fácil de usar?

No entanto, a partir dos dados recolhidos na Figura 46 verifica-se que o sistema foi considerado acessível por todos os inquiridos, pelo que o impacto que a acessibilidade terá na usabilidade, neste caso, será sempre positivo.

7.4.6 Satisfação geral

Por fim, recolheram-se as respostas dos utilizadores para averiguar a satisfação geral.

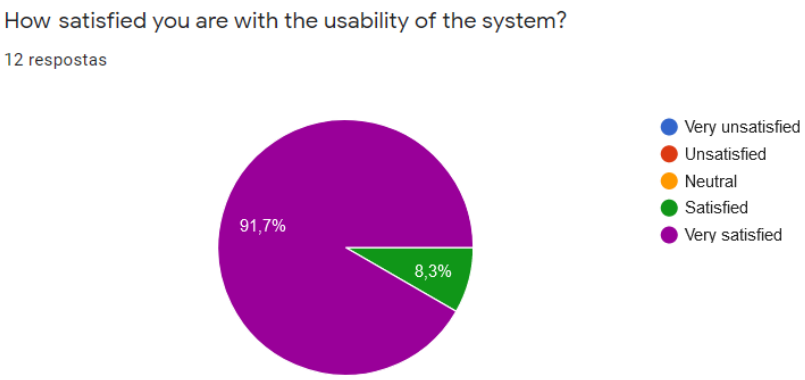

Figura 47 - Resultados do inquérito: 6 - Como avalia a sua satisfação em relação à usabilidade do sistema?

A partir da Figura 47, verifica-se que 91.7% dos inquiridos mostraram-se muito satisfeitos com a solução desenvolvida. Desta forma, confirmam-se os bons resultados obtidos nas questões anteriores. Conclui-se assim que os desenvolvedores não tiveram grandes dificuldades ou problemas no seu uso, mostraram-se bastante agradados no geral com a tecnologia, permitindo afirmar que a solução passou com distinção no teste de usabilidade realizado.

7.5 Avaliação da solução

Para a análise da qualidade do código desenvolvido foi utilizada a ferramenta SonarQube. Através desta ferramenta foi possível realizar revisões automáticas com análises estáticas de código de modo a detetar *bugs*, *code smells* (código com possíveis problemas futuros), vulnerabilidades de segurança e ainda verificar a quantidade de código testada. Todos os resultados da análise podem ser observados na Figura 48.

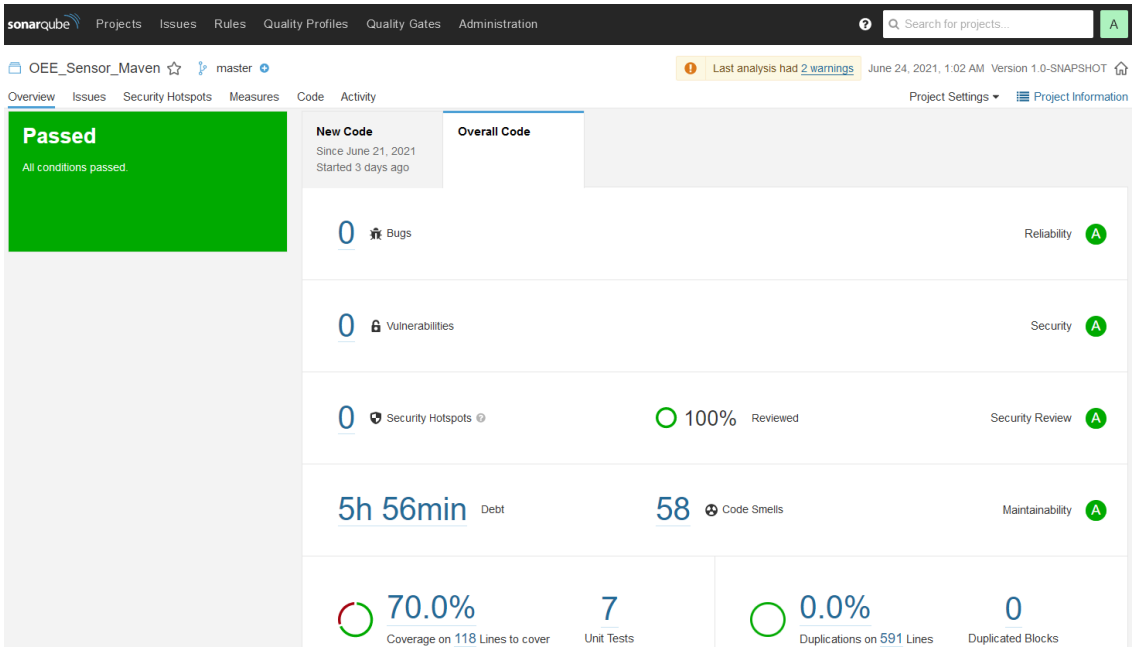

Figura 48 - Qualidade do código desenvolvido

A partir da análise realizada pelo Sonarqube, verifica-se que o módulo de captura passou nos testes de qualidade de código, apresentando qualidade A em todos os parâmetros. Por sua vez, a cobertura de testes, como já tinha sido vista na secção 6.1, é de 70% e a quantidade de código repetido é de 0%.

Conclui-se assim que o módulo desenvolvido apresenta uma alta fiabilidade e fácil manutenção, o que em muito é devida à aplicação de boas práticas e padrões de software.

8 Conclusão

De uma forma geral, é possível concluir-se que foi obtido um produto com grande utilidade e que foi de encontro às previsões pré-estabelecidas.

O sistema de monitorização em tempo real desenvolvido resolve assim uma lacuna na empresa, uma vez que permite acompanhar em tempo real as métricas OEE, em diferentes plataformas e em qualquer lugar.

Assim, no presente capítulo é descrito de uma maneira crítica os objetivos concretizados, as limitações e trabalho futuro, terminando com uma apreciação final sobre todo o trabalho desenvolvido.

8.1 Objetivos concretizados

O objetivo principal, definido na subsecção 1.4. tinha por base a criação de um sistema de monitorização intuitivo, que permitisse acompanhar em tempo real a partir de qualquer dispositivo e lugar, as diferentes métricas.

Após a conclusão do desenvolvimento da solução proposta, passou a estar disponível uma solução que permite visualizar as métricas em tempo real, sendo assim cumprido um dos requisitos principais.

Além disso é importante referir que todos os requisitos funcionais foram atingidos através da conclusão dos casos de uso propostos no Capítulo 3. Assim, foi possível criar toda a base necessária para o funcionamento do sistema podendo este, ser usado de forma acessível pelos utilizadores.

Relativamente aos requisitos não funcionais, grande parte dos mesmos foram concluídos com sucesso, como se pode observar na Tabela 14.

Tabela 14 - Requisitos não funcionais atingidos e não atingidos

A interface com o utilizador deve ser simples, intuitiva e inteiramente ajustada à ação em causa.	V
O sistema deve ser responsivo	V

Deve existir um tratamento específico para situações de funcionamento não esperado, impossibilitando que os erros sejam visualizados na interface com o utilizador.	V
Sempre que pedido pelos utilizadores, o sistema deve fornecer ajuda/explicação apropriada para a tarefa que o utilizador está a executar no momento.	V
O cálculo das métricas deve ser inferior a 5 segundos para a sua visualização em tempo real.	SM
O sistema deve estar preparado para que o tempo de resposta seja sensivelmente o mesmo independentemente da carga existente.	SM
Deve ser escalável para o processo de adição de novas funcionalidades.	V
Adoção do processo de desenvolvimento de software iterativo e incremental.	V
Boas práticas de design, nomeadamente padrões GRASP, SOLID e boas normas de codificação.	V
Deve ser usada uma base de dados relacional baseada em SQL para persistência dos dados.	V
A captura de dados do servidor deve ser feita através do uso de *Sockets*.	V
O núcleo principal do software deve ser desenvolvido em programação orientada a objetos.	V
Adoção do controlo de versões (GIT)	V

Realização de testes unitários e de performance.	V
Deve ser utilizado o Grafana para a parte da interface gráfica (cliente).	V
Uso de interface JDBC no módulo de captura para comunicação com a base de dados.	V
O sistema deve permitir visualizar as métricas em múltiplas plataformas	V
Deve cumprir o Regulamento Geral de Proteção de Dados (RGPD)	V
O layout da aplicação deve ter em atenção a organização dos gráficos para cada sensor e por cada métrica, entre outros aspetos relevantes.	V
Uso do padrão MVC.	V
Uso do padrão *Service* para a recolha dos dados dos sensores.	V
Uso do padrão *Repository* para acesso à base de dados.	V

Legenda: **V** - Requisito atingido, **X** - Requisito não atingido, **SM** – Requisito parcialmente atingido

Da Tabela 14, é possível concluir que apesar da maior parte dos requisitos não funcionais terem sido atingidos, houve dois que ficaram aquém.

Como foi verificado nos testes de performance (secção 6.3), com volumes considerados normais pela empresa (no máximo até 400 mil dados), a execução cumpre os requisitos estabelecidos em ambas as BD. No entanto, para volumes superiores, o tempo de processamento da MariaDB é maior afetando assim o seu desempenho.

Apesar de para a empresa não ser um problema, uma vez que segundo a mesma dificilmente serão usados volumes de dados dessa dimensão, é algo que pode ser melhorado, como se irá observar no capítulo seguinte.

8.2 Limitações e trabalho futuro

Como em todos os projetos, há sempre contratempos e impossibilidades que tornam o mesmo limitado. A escassez de tempo e a inexperiência em ambiente industrial dificultaram em algumas fases o seu desenvolvimento levando a atrasos e mudanças de perspetivas em termos arquiteturais.

No entanto foram sempre encontradas soluções e conseguiu-se um sistema de monitorização bastante coeso e que segue as boas práticas e padrões de software.

Relativamente às funcionalidades, as limitações prendem-se sobretudo pelo facto de o cálculo das métricas não apresentar um melhor desempenho para grandes volumes de dados. Apesar de não ser um problema na maior parte dos casos, é algo que ainda pode ser melhorado.

Relativamente a trabalho futuro, há uma grande margem para expandir e escalar as suas funcionalidades.

Num primeiro nível, seria importante diminuir as limitações referidas acima, melhorando o desempenho dos cálculos das métricas. Como se verificou na secção 6.3 aquando dos testes de performance, o MonetDB apresentou um desempenho muito superior ao MariaDB, demonstrando que as bases de dados orientadas por colunas são muito mais eficientes para o tipo de consultas pretendidas.

Desta forma, surgem assim duas sugestões de trabalho futuro. A primeira passa por configurar o MariaDB de modo que esta passe a ser orientada por colunas. Já segunda proposta, passa pela atualização manual do plugin do MonetDB de modo a ser compatível com as novas versões do Grafana.

De realçar que o sistema desenvolvido se encontra completamente preparado para o uso de cada uma das duas bases de dados, pelo que qualquer uma das mudanças sugeridas não afetará a integridade da aplicação.

Por fim, e já num segundo nível, poderiam ser adicionados mais gráficos para a visualização de outras métricas que a empresa achasse relevante e integrar o módulo de software Jasper no Grafana para a geração de relatórios.

8.3 Apreciação final

Apesar de todas as questões enunciadas, limitações ao projeto ou requisitos não concluídos, globalmente a solução cumpriu os objetivos a que se propôs. Por diversas vezes, ocorreram alterações de certos requisitos por parte da empresa, derivadas ao aparecimento de certos problemas, o que levou a um ajuste da direção do projeto, de acordo com as necessidades que foram aparecendo.

Assim, é possível concluir que a solução acaba por fazer todo o sentido no contexto da empresa, dado que foi feita e pensada para solucionar os problemas existentes e melhorar o sistema atual. Além disso, foi desenvolvida sob supervisão da mesma, havendo uma comunicação constante de modo que todos os requisitos fossem cumpridos.

As tecnologias usadas, como foi analisado no Capítulo 7, revelaram ser escolhas certeiras tendo por base todos os requisitos pré-estabelecidos e cumprindo os objetivos para os quais se destinavam.

Em termos de metodologia de desenvolvimento, levou-se em consideração as boas práticas de desenvolvimento, nomeadamente com um planeamento prévio do que ia ser feito, uma análise e design da solução, e consequente implementação e testes.

A nível pessoal, este projeto representou assim um enorme desafio a todos os níveis. O facto de ter sido realizado em ambiente industrial e num país diferente, tornou a experiência ainda mais única a todos os níveis, servindo de enriquecimento não só em termos de conhecimento, mas também revelando ser uma grande mais-valia noutros níveis. Assim, significou para o leitor um abrir de novos horizontes e um grande desenvolvimento das *soft-skills*, nomeadamente na melhoria da autonomia, organização e gestão do tempo.

Referências

Anderson, J. C., & Naru, J. A. (1998). BUSINESS MARKETING: UNDERSTAND WHAT CUSTOMERS VALUE. *Harvard Business Review, 98601*.

Andrey, L. (13 de Maio de 2019). *O Que é SSH e Como Funciona?* Obtido em 4 de Março de 2021, de https://www.weblink.com.br/blog/tecnologia/acesso-ssh-o-que-e/

Bartodziej, C. J. (2016). The concept Industry 4.0. Em *The concept Industry 4.0* (pp. 27-28). Springer Gabler, Wiesbaden. Obtido de https://link.springer.com/chapter/10.1007/978-3-658-16502-4_3#citeas

Becker, A. (2021). *HeidiSQL, Features List.* Obtido em 5 de Março de 2021, de https://www.heidisql.com/#featurelist

Beke, E., Horvath, R., & Tak-acsne, G. K. (Dezembro de 2020). Industry 4.0 and Current Competencies. *Naše gospo-darstvo/Our Economy, 66*(4), 64. doi:10.2478/ngoe-2020-0024

Bertola, F. (3 de Outubro de 2019). *Git, Github e Gitlab: o que são e principais diferenças.* Obtido em 14 de Março de 2021, de https://www.zup.com.br/blog/git-github-e-gitlab

Blank, S. (2013). Why the lean start-up changes everything. *harvard business review, 91*(5), 63-72. Obtido de https://hbr.org/2013/05/why-the-lean-start-up-changes-everything

Blink. (s.d.). *ERP: 5 Vantagens deste Software na Monitorização de Tarefas!* Obtido em 25 de Fevereiro de 2021, de https://www.blink-it.pt/erp-5-vantagens-monitorizacao-tarefas/

Brazil, B. (2018). What Is Prometheus? Em *Prometheus: Up & Running.* O'Reilly Media, Inc. Obtido em 10 de Março de 2021, de https://www.oreilly.com/library/view/prometheus-up/9781492034131/ch01.html

Britannica, T. E. (11 de Julho de 2019). *TCP/IP. Encyclopedia Britannica.* Obtido em 28 de Abril de 2021, de https://www.britannica.com/technology/TCP-IP

Canguçu, R. (25 de Fevereiro de 2021). *Codificar.* Obtido em 8 de Junho de 2021, de O que são Requisitos Funcionais e Requisitos Não Funcionais? : https://codificar.com.br/aplicativos/requisitos-funcionais-nao-funcionais/

Chronograf. (2021). *Create Chronograf dashboards.* Obtido em 10 de Março de 2021, de https://docs.influxdata.com/chronograf/v1.8/guides/create-a-dashboard/

Cisco. (Setembro de 2016). *Cisco Connected Assets 3.0.* Obtido em 7 de Março de 2021, de https://www.cisco.com/c/dam/en_us/solutions/industries/docs/connected-assets-design-guide.pdf

Code Academy. (s.d.). *MVC: Model, View, Controller*. Obtido em 10 de Junho de 2021, de https://www.codecademy.com/articles/mvc

Code Institute. (2021). *What is Java and why is it important?* Obtido em 19 de Junho de 2021, de Code Institute: https://codeinstitute.net/blog/what-is-java/

Cole, B. (Julho de 2015). *Value innovation.* Obtido em 12 de Março de 2021, de https://searchcio.techtarget.com/definition/value-innovation

Collectd. (2021). *Collectd Documentation.* Obtido em 27 de Fevereiro de 2021, de https://collectd.org/documentation.shtml

Content, R. R. (11 de Setembro de 2020). *Mercado de ERP mostra crescimento constante em todo o mundo.* Obtido em 24 de Fevereiro de 2021, de https://rockcontent.com/br/blog/mercado-de-erp-mostra-crescimento-constante-em-todo-o-mundo/

DAVENPORT, T. H. (Agosto de 1998). Putting the enterprise into the enterprise system. *Harvard Business Review*, pp. 121-131. Obtido em 20 de Fevereiro de 2021, de https://hbr.org/1998/07/putting-the-enterprise-into-the-enterprise-system

DB-Engines. (Fevereiro de 2021). *DB-Engines Ranking of Time Series DBMS.* Obtido em 27 de fevereiro de 2021, de https://db-engines.com/en/ranking/time+series+dbms

db-engines. (2021). *Graphite System Properties.* Obtido em 10 de Março de 2021, de https://db-engines.com/en/system/Graphite

de Oliveira, F. T., & Simões, W. L. (2017). *A INDÚSTRIA 4.0 E A PRODUÇÃO NO CONTEXTODOS ESTUDANTES DA ENGENHARIA.* Catalão: Universidade Federal de Goiás –Regional Catalão. Obtido em 15 de Fevereiro de 2021, de https://files.cercomp.ufg.br/weby/up/1012/o/Fernanda_Tha%C3%ADs_de_Oliveira.pdf

Debian. (2021). *Why debian.* Obtido em 04 de Março de 2021, de https://www.debian.org/intro/why_debian

Desjardins, J. (17 de Abril de 2019). *How much data is generated each day?* Obtido em 18 de Fevereiro de 2021, de https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-daycf4bddf29f/

Digital, H. (s.d.). *O que é uma rede LAN e uma rede WAN?* Obtido em 13 de Março de 2021, de https://helpdigitalti.com.br/o-que-e-uma-rede-lan-e-uma-rede-wan/

Eid, M. I., & Abba, H. I. (2017). User adaptation and ERP benefits: moderation analysis of user experience with ERP. *Kybernetes, 46*(3), 530–549. doi:10.1108/K-08-2015-0212

Elastic. (2021). *What is Kibana?* Obtido em 10 de Março de 2021, de https://www.elastic.co/what-is/kibana

Ellingwood, J. (5 de Dezembro de 2017). *An Introduction to Metrics, Monitoring, and Alerting.* Obtido em 25 de Fevereiro de 2021, de https://www.digitalocean.com/community/tutorials/an-introduction-to-metrics-monitoring-and-alerting

Elsevier, B.V. (2016). Tangible Industry 4.0: a scenario-based approach to learning for the future of production. Obtido de https://www.sciencedirect.com/science/article/pii/S2212827116301500

Elsmore, M. (23 de Março de 2020). *Prometheus vs. InfluxDB: A Monitoring Comparison.* Obtido em 27 de Fevereiro de 2021, de https://logz.io/blog/prometheus-influxdb/

engenheirando-software. (19 de Setembro de 2015). *Requisitos Arquiteturais*. Obtido em 8 de Junho de 2021, de http://engenheirando-software.github.io/2015/09/19/requisitos-arquiteturais/

ESCHBERGER, T. (27 de Abril de 2018). *The fuzzy front-end of the innovation process.* Obtido em 22 de Fevereiro de 2021, de https://www.lead-innovation.com/english-blog/fuzzy-front-end

FEUP. (s.d.). *Pequena história da Internet.* Obtido em 28 de Abril de 2021, de https://paginas.fe.up.pt/~mrs01003/TCP_IP.htm

Forster, F., & Harl, S. (2021). *CollectD.* Obtido em 27 de Fevereiro de 2021, de https://github.com/collectd/collectd

Fowler, M. (2002). *Service Layer*. Obtido em 11 de Junho de 2021, de https://java-design-patterns.com/patterns/service-layer/

Framework, B. D. (26 de Março de 2019). *"Where does 'Big Data' come from?* Obtido em 18 de Fevereiro de 2021, de https://www.bigdataframework.org/short-history-of-big-data/

Git. (2021). *Git - About.* Obtido em 14 de Março de 2021, de https://git-scm.com/about

Goldkuhl, G. (2016). Separation or unity? Behavioral science vs Design science. *AIS SIGPRAG, 12*, 1. Obtido em 15 de Fevereiro de 2021, de http://www.vits.org/publikationer/dokument/804.pdf

Grafana. (2021). *Grafana.* Obtido em 10 de Março de 2021, de https://grafana.com/oss/grafana/?plcmt=footer

Grafana. (2021). *The analytics platform for all your metrics.* Obtido em 10 de Março de 2021, de https://grafana.com/grafana/?plcmt=footer#visualize-content

Graphite. (2021). *Getting Started.* Obtido em 10 de Março de 2021, de https://graphiteapp.org/

Indústria, A. V. (15 de Agosto de 2018). *avozdaindustria.* Obtido em 18 de Fevereiro de 2021, de https://avozdaindustria.com.br/ind-stria-40-totvs/5-desafios-da-implanta-o-da-ind-stria-40-como-super-los

Influxdata. (2021). *Act in Time. Build on InfluxDB.* Obtido em 27 de Fevereiro de 2021, de https://www.influxdata.com/

InfluxData. (2021). *Chronograf.* Obtido em 10 de Março de 2021, de https://www.influxdata.com/time-series-platform/chronograf/

InfluxData. (2021). *Chronograf 1.7 documentation.* Obtido em 10 de Março de 2021, de https://docs.influxdata.com/chronograf/v1.7/

InfluxData. (2021). *InfluxDB compared to SQL databases.* Obtido em 27 de Fevereiro de 2021, de https://docs.influxdata.com/influxdb/v1.7/concepts/crosswalk/

IPC2U. (s.d.). *The main differences between RS-232, RS-422 and RS-485.* Obtido em 7 de Março de 2021, de https://ipc2u.com/articles/knowledge-base/the-main-differences-between-rs-232-rs-422-and-rs-485/

Isoton, M. J. (2019). *Integração de ferramentas para a coleta de métricas em servidores Linux.* Caxias do Sul: Universidade de Caxias do Sul. Obtido em 26 de Fevereiro de 2021, de https://repositorio.ucs.br/xmlui/bitstream/handle/11338/5950/TCC%20Michel%20J%c3%banior%20Isoton.pdf?sequence=1&isAllowed=y

Jasper. (2021). *Jaspersoft Studio.* Obtido em 6 de Março de 2021, de https://community.jaspersoft.com/project/jaspersoft-studio

Junior, J. B., & Pires, S. R. (2010). *SISTEMAS INTEGRADOS DE GESTÃO ERP E CLOUD COMPUTING: CARACTERÍSTICAS, VANTAGENS E DESAFIOS.* Simpoi. Obtido em 20 de Fevereiro de 2021, de http://www.unimep.br/~jocamarg/Joao_files/artigos/Joao_Camargo_1103.pdf

KAGERMANN, H. (2013). *Recommendations for implementing the strategic iniative Industrie.* acatech - National Academy of Science and Engineering. Obtido em 16 de Fevereiro de 2021, de https://www.din.de/blob/76902/e8cac883f42bf28536e7e8165993f1fd/recommendations-for-implementing-industry-4-0-data.pdf

Keane, S. F., Cormican, K. T., & Sheahan, J. N. (2018). Comparing how entrepreneurs and managers represent the elements of the business model canvas. *Journal of Business Venturing Insights, 9*, 65-74. Obtido de https://www.sciencedirect.com/science/article/pii/S235267341730094X#bib44

Koen, Ajamian, & Burkart. (2001). Providing clarity and a common language to the "Fuzzy front end". *Research Technology Management, 44*(2), 46-55. Obtido em 22 de Fevereiro de 2021, de https://www.researchgate.net/publication/233595627_Providing_Clarity_and_Common_Language_to_the_Fuzzy_Front_End

Koen, P. A., Bertels, H. M., & Kleinschmid, E. (2014). Managing the Front End of Innovation - Part I: Results From a Three-Year Study. *Taylor & Francis Online*, 34-43. Obtido em 22 de Fevereiro de 2021, de https://www.tandfonline.com/doi/pdf/10.5437/08956308X5702145?casa_token=TkrQJSN3A98AAAAA:T3g3IC7Zm8iMB_cYOG6pAU1pG5e4TUsRwHhjYx0GqBk4-o3Df5pT8FmXcA7RJtUYwuKom2H8fwSx

Koen, P. A., Bertels, H. M., & Kleinschmidt, E. J. (2014). Managing the Front End of Innovation—Part II: Results from a Three-Year Study. *Research-Technology Management, 57*(3), 25-35. Obtido em 22 de Fevereiro de 2021, de https://www.tandfonline.com/doi/abs/10.5437/08956308X5703199

Koen, P., Ajamian, G., Burkart, R., Clamen, A., Davidson, J., D'Amore, R., . . . Wagner, K. (2001). Providing Clarity and A Common Language to the "Fuzzy Front End". *Taylor & Francis Online, 44*(2), 46-55. Obtido de https://www.tandfonline.com/doi/abs/10.1080/08956308.2001.11671418

KOTLER, P. (1998). *Administração de marketing: análise, planejamento, implementação e controle.* São Paulo: Atlas. Obtido de http://www.geocities.ws/mba_marketing2001/v07-4art05.pdf

Kumar, N. (2019). *Creating your personal IoT/Utility Dashboard using Grafana, Influxdb & Telegraf on a Rasperry Pi.* Obtido em 10 de Março de 2021, de https://medium.com/@neelabhsingh/creating-your-personal-iot-utility-dashboard-using-grafana-influxdb-telegraf-on-a-rasperry-pi-ca78eb9d7ef0

Lai, B. H. (22 de Março de 2018). *Netdata: simple server monitoring.* Obtido em 27 de Fevereiro de 2021, de https://www.joyfulbikeshedding.com/blog/2018-03-22-netdata-simple-server-monitoring.html

Lanhellas, R. (2014). *JasperReport: Relatórios em Java com iReport.* Obtido em 6 de Março de 2021, de https://www.devmedia.com.br/jasperreport-relatorios-em-java-com-ireport/31075

Lasi, H., Fettke, P., Kemper, H. G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering, 6*(4), 239-242.

Leon, A. (2007). *Enterprise Resource Planning* (2nd ed.). New Deli: Tata McGraw-Hill Education Private Limited. Obtido em 24 de Fevereiro de 2021, de https://www.amazon.com/Enterprise-Resource-Planning-Alexis-Leon/dp/1259005917

Logic, S. (15 de Janeiro de 2020). *Tracking Systems Metrics with collectd.* Obtido em 27 de Fevereiro de 2021, de https://www.sumologic.com/blog/tracking-systems-metrics-collectd/

Mahmud, I., Ramayah, T., & Kurnia, S. (2017). To use or not to use : Modelling end user grumbling as user resistance in pre-implementation stage of enterprise resource planning system. *Information Systems, 69*, 164-179. doi:10.1016/j.is.2017.05.005

MAHMUD, R., KOTAGIRI, R., & BUYYA, R. (2018). Fog computing: A taxonomy, survey and futuredirections. *Internet of Everything*, 103-130. doi:10.1007/978-981-10-5861-5_5

Mai, H. D., Hoang, T. T., Tao, M. T., & Au, P. H. (2020). *Research and implementation of monitoring systems Prometheus and Graphana.* FPTU HCM. Obtido de http://googleauthensite02.insidehn.uni.fpt.edu.vn:82/ViewPDFOnline/document.php?loc=0&doc=783615700437438646884834429815748146 61

MariaDB. (2021). *About MariaDB Server.* Obtido em 13 de Março de 2021, de https://mariadb.org/about/#entry-header

Martinez, P. (2020). *Domain-Driven Design: Everything You Always Wanted to Know About it, But Were Afraid to Ask*. Obtido em 10 de Junho de 2021, de https://medium.com/ssense-tech/domain-driven-design-everything-you-always-wanted-to-know-about-it-but-were-afraid-to-ask-a85e7b74497a

Massa, L., & Tucci, C. L. (2014). *Business model innovation* (1st ed.). Oxford Handbook Online. Obtido de https://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199694945.001.0001/oxfordhb-9780199694945-e-002

Microsoft. (s.d.). *O que é o ERP e porque precisa dele?* Obtido em 25 de Fevereiro de 2021, de https://dynamics.microsoft.com/pt-pt/erp/what-is-erp/

MonetDB. (2021). *Column store features*. Obtido em 15 de Junho de 2021, de https://www.monetdb.org/content/column-store-features

Muhtaroglu, F. C., Demir, S., Obalı, M., & Girgin, C. (2013). Business Model Canvas Perspective on Big Data Applications. *IEEE International Conference on Big Data*, 33.

Netdata. (2021). *How collectors work.* Obtido em 27 de Fevereiro de 2021, de https://learn.netdata.cloud/docs/collect/how-collectors-work

Novida. (s.d.). *OEE – Como calcular e usar a métrica a seu favor!* Obtido em 26 de Fevereiro de 2021, de https://www.novida.com.br/blog/oee/

OEE. (2020). *What is Overall Equipment Effectiveness?* Obtido em 26 de Fevereiro de 2021, de https://www.oee.com/

OEE. (2021). *OEE*. Obtido em 12 de Fevereiro de 2021, de OEE: http://oee.lt/lt/

OEE. (s.d.). *Como calcular o OEE?* Obtido em 26 de Fevereiro de 2021, de https://www.oee.com.br/como-calcular-o-oee/

OEE. (s.d.). *Six Big Losses.* Obtido em 26 de Fevereiro de 2021, de https://www.oee.com/oee-six-big-losses.html

OESTERREICH, T. D., & TEUTEBERG, F. (2016). Understanding the implications of digitisation and automation in the context of Industry 4.0: A triangulationapproach and elements of a research agenda for the construction industry. *Computers in Industry, 83*, 121-139. doi:https://doi.org/10.1016/j.compind.2016.09.006

Oracle. (2010). *The Java EE 5 Tutorial.* Obtido em 5 de Março de 2021, de https://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html

Oracle. (2020). *Lesson: All About Sockets.* Obtido em 13 de Março de 2021, de https://docs.oracle.com/javase/tutorial/networking/sockets/index.html

Oracle. (s.d.). *Definition of Enterprise Resource Planning (ERP).* Obtido em 24 de Fevereiro de 2021, de https://recipp.ipp.pt/bitstream/10400.22/16442/1/DM_VanessaRamos_2020_MEGI.pdf

Osterwalder, A. (June de 2004). *The Business Model Ontology–A Proposition in A Design Science Approach.* University of Lausanne. Lausanne: University of Lausanne. Obtido de https://d1wqtxts1xzle7.cloudfront.net/30373644/thebusiness-model-ontology.pdf?1356515806=&response-content-disposition=inline%3B+filename%3DThe_Business_Model_Ontology_a_propositio.pdf&Expires=1612980424&Signature=Tw0WqBhYZV7HPshGOwHMjTXtRsDBCDMxsqeqy3UMb

ÖZMEN, A. (2009). An entropy-based algorithm for data elimination in time-driven software. *Journal of Systems and Software, 82*(5), 907-913. doi:https://doi.org/10.1109/DSN.2017.39

Öztürk, F., Kayar, A., & Vatansever, A. (2019). Advanced Manufacturing with Industry 4.0 Applications. Istambul.

P. Blocke, C. (2011). Modeling customer value perceptions in cross-cultural business markets. *Journal of Business Research, 64*, 533–540. Obtido de http://www.isihome.ir/freearticle/ISIHome.ir-22054.pdf

Pav. (21 de Dezembro de 2016). *Host monitoring with collectd.* Obtido em 27 de Fevereiro de 2021, de https://codeblog.dotsandbrackets.com/host-monitoring-with-collectd/

Pederneiras, G. (28 de Julho de 2019). *Sistema ciber físico na Indústria 4.0.* Obtido em 25 de Abril de 2021, de https://www.industria40.ind.br/artigo/18514-sistema-ciber-fisico-na-industria-40

Pello, R. (31 de Outubro de 2018). *medium.* Obtido em 15 de Fevereiro de 2021, de https://medium.com/@pello/design-science-research-a-summary-bb538a40f669

Redator Rock Content. (24 de November de 2020). *rockcontent.* Obtido em 10 de February de 2021, de https://rockcontent.com/br/blog/proposta-de-valor/

RUBMANN, M., LORENZ, M., GERBERT, P., WALDNER, M., JUSTUS, J., ENGEL, P., & HARNISCH, M. (Abril de 9 de 2015). *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries.* Obtido em 17 de Fevereiro de 2021, de bcg: https://www.bcg.com/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries

SAKURAI, R., & ZUCHI, J. D. (2018). AS REVOLUÇÕES INDUSTRIAIS ATÉ A INDUSTRIA 4.0. *Revista Interface Tecnológica, 15*(2), 480-491. doi:10.31510/infa.v15i2.386

Salomão, A. (s.d.). *OEE – Eficiência Global dos Equipamentos.* Obtido em 26 de Fevereiro de 2021, de https://qualyteam.com/pb/blog/oee-eficiencia-global-dos-equipamentos/

Santos, B. P., Alberto, A., Lima, T. D., & Charrua-Santos, F. M. (2018). INDUSTRY 4.0: CHALLENGES AND OPPORTUNITIES. *Revista Produção E Desenvolvimento, 4*(1), 111-124. Obtido de https://revistas.cefet-rj.br/index.php/producaoedesenvolvimento/article/view/316

SCHWAB, K. (2016). A quarta revolução industrial. Em Edipro (Ed.). São Paulo: World Economic Forum. Obtido em 17 de Fevereiro de 2021, de https://issuu.com/j00kun/docs/klaus_schwab_-_a_quarta_revolu__o_i

SCHWAB, K. (2016). A quarta revolução industrial. Em Edipro (Ed.). São Paulo: World Economic Forum. Obtido em 17 de Fevereiro de 2021, de https://issuu.com/j00kun/docs/klaus_schwab_-_a_quarta_revolu__o_i

SCHWAB, K. (2016). A quarta revolução industrial. Em Edipro (Ed.). São Paulo: World Economic Forum. Obtido em 17 de Fevereiro de 2021, de https://issuu.com/j00kun/docs/klaus_schwab_-_a_quarta_revolu__o_i

Siemens. (s.d.). Um guia prático sobre a Indústria 4.0. *Siemens Newsletter*. Obtido em 16 de Fevereiro de 2021, de https://new.siemens.com/br/pt/empresa/stories/industria/industria-4-0.html

SILVEIRA, C. B. (2017). O que é a Indústria 4.0 e como ela vai impactar o mundo. *Citisystems*. Obtido em 16 de Fevereiro de 2021, de https://www.citisystems.com.br/industria-4-0/

Souza, I. d. (17 de Novembro de 2020). *Saiba o que é Apache Tomcat e por que ele é usado.* Obtido em 5 de Março de 2021, de https://rockcontent.com/br/blog/tomcat/

Stock, D., Stöhr, M., Rauschecker, U., & Bauernhan, T. J. (2014). Cloud-based Platform to Facilitate Access to Manufacturing IT. *Procedia CIRP, 25*, 320-328. doi:10.1016/j.procir.2014.10.045

Teixeira, J. R. (8 de Novembro de 2013). *Introdução ao MySQL.* Obtido em 5 de Março de 2021, de https://www.devmedia.com.br/introducao-ao-mysql/27799

Telegraf. (2021). *Telegraf.* Obtido em 27 de Fevereiro de 2021, de https://www.influxdata.com/time-series-platform/telegraf/

Telegraf. (2021). *Telegraf.* Obtido em 27 de Fevereiro de 2021, de https://github.com/influxdata/telegraf

Thames, L., & Schaefer, D. (2016). Software-defined Cloud Manufacturing for Industry 4.0. Em *Software-defined Cloud Manufacturing for Industry 4.0* (pp. 12-17). Schaefer. Obtido de https://www.sciencedirect.com/science/article/pii/S2212827116307910

Trimi, S., & Berbegal-Mirabent, J. (2012). Business model innovation in entrepreneurship. *International Entrepreneurship and Management Journal, 8*, 449-465. Obtido de https://link.springer.com/article/10.1007%2Fs11365-012-0234-3

Turnbull, J. (2018). Introduction to Prometheus. Em *Monitoring with Prometheus* (pp. 49-51). Turnbull Press. Obtido em 10 de Março de 2021, de https://books.google.lt/books?hl=pt-PT&lr=&id=EtlfDwAAQBAJ&oi=fnd&pg=PA1&dq=prometheus+grafana&ots=569AHJ1f4d&sig=EQy5iMwAL68FXasLoAgJw8hkOiQ&redir_esc=y#v=onepage&q=prometheus%20grafana&f=false

Vedois. (6 de Junho de 2018). *IMPORTÂNCIA DO MONITORAMENTO DE PRODUÇÃO NA INDÚSTRIA 4.0.* Obtido em 25 de Fevereiro de 2021, de http://vedois.com.br/site/importancia-do-monitoramento-de-producao-na-industria-4-0/

Vieira, J. (26 de Março de 2019). *Teste de usabilidade: tudo o que você precisa saber!* Obtido em 28 de Junho de 2021, de Medium: https://medium.com/aela/teste-de-usabilidade-o-que-voc%C3%AA-precisa-saber-39a36343d9a6

Ward, S. (24 de Julho de 2020). *What Is Business Innovation?* Obtido em 12 de Março de 2021, de https://www.thebalancesmb.com/business-innovation-definition-2948310

Weis, O. (5 de Junho de 2020). *O que é Ethernet e Como Funciona.* Obtido em 28 de Abril de 2021, de https://www.net-usb.com/pt/usb-over-ethernet-system/what-is-ethernet/

WILLIG, A. (2008). *Recent and Emerging Topics in Wireless Recent and Emerging Topics in Wireless* (2nd ed.). Berlim: IEEE. Obtido de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.144.6170&rep=rep1&type=pdf

WOODRUFF, R. (1997). Customer Value: the next source for competitive advantage. *Journal of the Academy of Marketing Science, 25*(2), 139-153. Obtido de http://www.geocities.ws/mba_marketing2001/v07-4art05.pdf

Yigal, A. (7 de Abril de 2020). *Grafana vs. Kibana: The Key Differences to Know.* Obtido em 10 de Março de 2021, de https://logz.io/blog/grafana-vs-kibana/

ZAWADZKI, P., & ŻYWICKI, K. (2016). Smart product design and production control for effective mass customization in the Industry 4.0 concept. *Management and Production Engineering Review, 7*(3). doi:10.1515/mper-2016-0030

ZEITHAML, V. (1988). Consumer perceptions of price, quality and value: a means-end model and synthesis of evidence. *Journal of Marketing, 52*(3), 2-22. Obtido de http://www.geocities.ws/mba_marketing2001/v07-4art05.pdf

Anexos

Anexo A

Nesta secção encontram-se anexadas as *queries* usadas pelo Grafana para o cálculo das métricas.

```
SELECT
    1 AS time_sec,
    COALESCE(Up1 / ExpUp1 * 100, 0) AS S1,
    COALESCE(Up2 / ExpUp2 * 100, 0) AS S2,
    COALESCE(Up3 / ExpUp3 * 100, 0) AS S3,
    COALESCE(Up4 / ExpUp4 * 100, 0) AS S4,
    COALESCE(Up5 / ExpUp5 * 100, 0) AS S5,
    COALESCE(Up6 / ExpUp6 * 100, 0) AS S6,
    COALESCE(Up7 / ExpUp7 * 100, 0) AS S7,
    COALESCE(Up8 / ExpUp8 * 100, 0) AS S8,
    COALESCE(Up9 / ExpUp9 * 100, 0) AS S9,
    COALESCE(Up10 / ExpUp10 * 100, 0) AS S10,
    COALESCE(Up11 / ExpUp11 * 100, 0) AS S11,
    COALESCE(Up12 / ExpUp12 * 100, 0) AS S12,
    COALESCE(Up13 / ExpUp13 * 100, 0) AS S13,
    COALESCE(Up14 / ExpUp14 * 100, 0) AS S14,
    COALESCE(Up15 / ExpUp15 * 100, 0) AS S15,
    COALESCE(Up16 / ExpUp16 * 100, 0) AS S16
FROM
(
    SELECT
        SUM((ExpectedWorking&1 != 1) * TimeSinceLastUpdate) AS ExpUp1,
        SUM((IsWorking&1 != 1) * TimeSinceLastUpdate) AS Up1,

        SUM((ExpectedWorking&2 != 2) * TimeSinceLastUpdate) AS ExpUp2,
        SUM((IsWorking&2 != 2) * TimeSinceLastUpdate) AS Up2,

        SUM((ExpectedWorking&4 != 4) * TimeSinceLastUpdate) AS ExpUp3,
        SUM((IsWorking&4 != 4) * TimeSinceLastUpdate) AS Up3,

        SUM((ExpectedWorking&8 != 8) * TimeSinceLastUpdate) AS ExpUp4,
        SUM((IsWorking&8 != 8) * TimeSinceLastUpdate) AS Up4,

        SUM((ExpectedWorking&16 != 16) * TimeSinceLastUpdate) AS ExpUp5,
        SUM((IsWorking&16 != 16) * TimeSinceLastUpdate) AS Up5,

        SUM((ExpectedWorking&32 != 32) * TimeSinceLastUpdate) AS ExpUp6,
        SUM((IsWorking&32 != 32) * TimeSinceLastUpdate) AS Up6,

        SUM((ExpectedWorking&64 != 64) * TimeSinceLastUpdate) AS ExpUp7,
        SUM((IsWorking&64 != 64) * TimeSinceLastUpdate) AS Up7,

        SUM((ExpectedWorking&128 != 128) * TimeSinceLastUpdate) AS ExpUp8,
        SUM((IsWorking&128 != 128) * TimeSinceLastUpdate) AS Up8,

        SUM((ExpectedWorking&256 != 256) * TimeSinceLastUpdate) AS ExpUp9,
        SUM((IsWorking&256 != 256) * TimeSinceLastUpdate) AS Up9,

        SUM((ExpectedWorking&512 != 512) * TimeSinceLastUpdate) AS ExpUp10,
        SUM((IsWorking&512 != 512) * TimeSinceLastUpdate) AS Up10,

        SUM((ExpectedWorking&1024 != 1024) * TimeSinceLastUpdate) AS ExpUp11,
        SUM((IsWorking&1024 != 1024) * TimeSinceLastUpdate) AS Up11,

        SUM((ExpectedWorking&2048 != 2048) * TimeSinceLastUpdate) AS ExpUp12,
        SUM((IsWorking&2048 != 2048) * TimeSinceLastUpdate) AS Up12,

        SUM((ExpectedWorking&4096 != 4096) * TimeSinceLastUpdate) AS ExpUp13,
        SUM((IsWorking&4096 != 4096) * TimeSinceLastUpdate) AS Up13,

        SUM((ExpectedWorking&8192 != 8192) * TimeSinceLastUpdate) AS ExpUp14,
        SUM((IsWorking&8192 != 8192) * TimeSinceLastUpdate) AS Up14,

        SUM((ExpectedWorking&16384 != 16384) * TimeSinceLastUpdate) AS ExpUp15,
        SUM((IsWorking&16384 != 16384) * TimeSinceLastUpdate) AS Up15,

        SUM((ExpectedWorking&32768 != 32768) * TimeSinceLastUpdate) AS ExpUp16,
        SUM((IsWorking&32768 != 32768) * TimeSinceLastUpdate) AS Up16
    FROM sensors
    WHERE $__timeFilter(Timestamp)
) availabilityMetrics
```

Excerto de Código 22 - Query do cálculo da métrica da disponibilidade

```
SELECT
1 AS time_sec,
COALESCE(1.9*NumPieces1 / Up1 * 100, 0) AS S1,
COALESCE(1.9*NumPieces2 / Up2 * 100, 0) AS S2,
COALESCE(1.9*NumPieces3 / Up3 * 100, 0) AS S3,
COALESCE(1.9*NumPieces4 / Up4 * 100, 0) AS S4,
COALESCE(1.9*NumPieces5 / Up5 * 100, 0) AS S5,
COALESCE(1.9*NumPieces6 / Up6 * 100, 0) AS S6,
COALESCE(1.9*NumPieces7 / Up7 * 100, 0) AS S7,
COALESCE(1.9*NumPieces8 / Up8 * 100, 0) AS S8,
COALESCE(1.9*NumPieces9 / Up9 * 100, 0) AS S9,
COALESCE(1.9*NumPieces10 / Up10 * 100, 0) AS S10,
COALESCE(1.9*NumPieces11 / Up11 * 100, 0) AS S11,
COALESCE(1.9*NumPieces12 / Up12 * 100, 0) AS S12,
COALESCE(1.9*NumPieces13 / Up13 * 100, 0) AS S13,
COALESCE(1.9*NumPieces14 / Up14 * 100, 0) AS S14,
COALESCE(1.9*NumPieces15 / Up15 * 100, 0) AS S15,
COALESCE(1.9*NumPieces16 / Up16 * 100, 0) AS S16
FROM
(
    SELECT
        SUM(1 & ChangedState & IsWorking) AS NumPieces1,
        SUM((IsWorking&1 != 1) * TimeSinceLastUpdate) AS Up1,

        SUM((2 & ChangedState & IsWorking) = 2) AS NumPieces2,
        SUM((IsWorking&2 != 2) * TimeSinceLastUpdate) AS Up2,

        SUM((4 & ChangedState & IsWorking) = 4) AS NumPieces3,
        SUM((IsWorking&4 != 4) * TimeSinceLastUpdate) AS Up3,

        SUM((8 & ChangedState & IsWorking) = 8) AS NumPieces4,
        SUM((IsWorking&8 != 8) * TimeSinceLastUpdate) AS Up4,

        SUM((16 & ChangedState & IsWorking) = 16) AS NumPieces5,
        SUM((IsWorking&16 != 16) * TimeSinceLastUpdate) AS Up5,

        SUM((32 & ChangedState & IsWorking) = 32) AS NumPieces6,
        SUM((IsWorking&32 != 32) * TimeSinceLastUpdate) AS Up6,

        SUM((64 & ChangedState & IsWorking) = 64) AS NumPieces7,
        SUM((IsWorking&64 != 64) * TimeSinceLastUpdate) AS Up7,

        SUM((128 & ChangedState & IsWorking) = 128) AS NumPieces8,
        SUM((IsWorking&128 != 128) * TimeSinceLastUpdate) AS Up8,

        SUM((256 & ChangedState & IsWorking) = 256) AS NumPieces9,
        SUM((IsWorking&256 != 256) * TimeSinceLastUpdate) AS Up9,

        SUM((512 & ChangedState & IsWorking) = 512) AS NumPieces10,
        SUM((IsWorking&512 != 512) * TimeSinceLastUpdate) AS Up10,

        SUM((1024 & ChangedState & IsWorking) = 1024) AS NumPieces11,
        SUM((IsWorking&1024 != 1024) * TimeSinceLastUpdate) AS Up11,

        SUM((2048 & ChangedState & IsWorking) = 2048) AS NumPieces12,
        SUM((IsWorking&2048 != 2048) * TimeSinceLastUpdate) AS Up12,

        SUM((4096 & ChangedState & IsWorking) = 4096) AS NumPieces13,
        SUM((IsWorking&4096 != 4096) * TimeSinceLastUpdate) AS Up13,

        SUM((8192 & ChangedState & IsWorking) = 8192) AS NumPieces14,
        SUM((IsWorking&8192 != 8192) * TimeSinceLastUpdate) AS Up14,

        SUM((16384 & ChangedState & IsWorking) = 16384) AS NumPieces15,
        SUM((IsWorking&16384 != 16384) * TimeSinceLastUpdate) AS Up15,

        SUM((32768 & ChangedState & IsWorking) = 32768) AS NumPieces16,
        SUM((IsWorking&32768 != 32768) * TimeSinceLastUpdate) AS Up16
    FROM sensors
    WHERE $__timeFilter(Timestamp)
) performanceMetrics
```

Excerto de Código 23 - Query do cálculo da métrica do desempenho

```
SELECT
    1 AS time_sec,
    COALESCE(100 - NumPiecesDefect1 / NumPieces1 * 100, 0) AS S1,
    COALESCE(100 - NumPiecesDefect2 / NumPieces2 * 100, 0) AS S2,
    COALESCE(100 - NumPiecesDefect3 / NumPieces3 * 100, 0) AS S3,
    COALESCE(100 - NumPiecesDefect4 / NumPieces4 * 100, 0) AS S4,
    COALESCE(100 - NumPiecesDefect5 / NumPieces5 * 100, 0) AS S5,
    COALESCE(100 - NumPiecesDefect6 / NumPieces6 * 100, 0) AS S6,
    COALESCE(100 - NumPiecesDefect7 / NumPieces7 * 100, 0) AS S7,
    COALESCE(100 - NumPiecesDefect8 / NumPieces8 * 100, 0) AS S8,
    COALESCE(100 - NumPiecesDefect9 / NumPieces9 * 100, 0) AS S9,
    COALESCE(100 - NumPiecesDefect10 / NumPieces10 * 100, 0) AS S10,
    COALESCE(100 - NumPiecesDefect11 / NumPieces11 * 100, 0) AS S11,
    COALESCE(100 - NumPiecesDefect12 / NumPieces12 * 100, 0) AS S12,
    COALESCE(100 - NumPiecesDefect13 / NumPieces13 * 100, 0) AS S13,
    COALESCE(100 - NumPiecesDefect14 / NumPieces14 * 100, 0) AS S14,
    COALESCE(100 - NumPiecesDefect15 / NumPieces15 * 100, 0) AS S15,
    COALESCE(100 - NumPiecesDefect16 / NumPieces16 * 100, 0) AS S16
FROM
(
    SELECT
        SUM(1 & ChangedState & IsWorking & HasDefect) AS NumPiecesDefect1,
        SUM((1  & ChangedState & IsWorking) = 1 ) AS NumPieces1,

        SUM((2 & ChangedState & IsWorking & HasDefect) = 2) AS NumPiecesDefect2,
        SUM((2  & ChangedState & IsWorking) = 2 ) AS NumPieces2,

        SUM((4 & ChangedState & IsWorking & HasDefect) = 4) AS NumPiecesDefect3,
        SUM((4  & ChangedState & IsWorking) = 4 ) AS NumPieces3,

        SUM((8 & ChangedState & IsWorking & HasDefect) = 8) AS NumPiecesDefect4,
        SUM((8  & ChangedState & IsWorking) = 8 ) AS NumPieces4,

        SUM((16 & ChangedState & IsWorking & HasDefect) = 16) AS NumPiecesDefect5,
        SUM((16  & ChangedState & IsWorking) = 16 ) AS NumPieces5,

        SUM((32 & ChangedState & IsWorking & HasDefect) = 32) AS NumPiecesDefect6,
        SUM((32  & ChangedState & IsWorking) = 32 ) AS NumPieces6,

        SUM((64 & ChangedState & IsWorking & HasDefect) = 64) AS NumPiecesDefect7,
        SUM((64  & ChangedState & IsWorking) = 64 ) AS NumPieces7,

        SUM((128 & ChangedState & IsWorking & HasDefect) = 128) AS NumPiecesDefect8,
        SUM((128  & ChangedState & IsWorking) = 128 ) AS NumPieces8,

        SUM((256 & ChangedState & IsWorking & HasDefect) = 256) AS NumPiecesDefect9,
        SUM((256  & ChangedState & IsWorking) = 256 ) AS NumPieces9,

        SUM((512 & ChangedState & IsWorking & HasDefect) = 512) AS NumPiecesDefect10,
        SUM((512  & ChangedState & IsWorking) = 512 ) AS NumPieces10,

        SUM((1024 & ChangedState & IsWorking & HasDefect) = 1024) AS NumPiecesDefect11,
        SUM((1024  & ChangedState & IsWorking) = 1024 ) AS NumPieces11,

        SUM((2048 & ChangedState & IsWorking & HasDefect) = 2048) AS NumPiecesDefect12,
        SUM((2048  & ChangedState & IsWorking) = 2048 ) AS NumPieces12,

        SUM((4096 & ChangedState & IsWorking & HasDefect) = 4096) AS NumPiecesDefect13,
        SUM((4096  & ChangedState & IsWorking) = 4096 ) AS NumPieces13,

        SUM((8192 & ChangedState & IsWorking & HasDefect) = 8192) AS NumPiecesDefect14,
        SUM((8192  & ChangedState & IsWorking) = 8192 ) AS NumPieces14,

        SUM((16384 & ChangedState & IsWorking & HasDefect) = 16384) AS NumPiecesDefect15,
        SUM((16384  & ChangedState & IsWorking) = 16384 ) AS NumPieces15,

        SUM((32768 & ChangedState & IsWorking & HasDefect) = 32768) AS NumPiecesDefect16,
        SUM((32768  & ChangedState & IsWorking) = 32768 ) AS NumPieces16
    FROM sensors
    WHERE $__timeFilter(Timestamp)
) qualityMetrics
```

Excerto de Código 24 - Query do cálculo da métrica da qualidade

```
SELECT
    1 AS time_sec,
    COALESCE((Up1 / ExpUp1) * (1.9 * NumPieces1 /  Up1) * NumPiecesDefect1 / NumPieces1, 0) * 100 AS S1,
    COALESCE((Up2 / ExpUp2) * (1.9 * NumPieces2 /  Up2) * NumPiecesDefect2 / NumPieces2, 0) * 100 AS S2,
    COALESCE((Up3 / ExpUp3) * (1.9 * NumPieces3 /  Up3) * NumPiecesDefect3 / NumPieces3, 0) * 100 AS S3,
    COALESCE((Up4 / ExpUp4) * (1.9 * NumPieces4 /  Up4) * NumPiecesDefect4 / NumPieces4, 0) * 100 AS S4,
    COALESCE((Up5 / ExpUp5) * (1.9 * NumPieces5 /  Up5) * NumPiecesDefect5 / NumPieces5, 0) * 100 AS S5,
    COALESCE((Up6 / ExpUp6) * (1.9 * NumPieces6 /  Up6) * NumPiecesDefect6 / NumPieces6, 0) * 100 AS S6,
    COALESCE((Up7 / ExpUp7) * (1.9 * NumPieces7 /  Up7) * NumPiecesDefect7 / NumPieces7, 0) * 100 AS S7,
    COALESCE((Up8 / ExpUp8) * (1.9 * NumPieces8 /  Up8) * NumPiecesDefect8 / NumPieces8, 0) * 100 AS S8,
    COALESCE((Up9 / ExpUp9) * (1.9 * NumPieces9 /  Up9) * NumPiecesDefect9 / NumPieces9, 0) * 100 AS S9,
    COALESCE((Up10 / ExpUp10) * (1.9 * NumPieces10 /  Up10) * NumPiecesDefect10 / NumPieces10, 0) * 100 AS S10,
    COALESCE((Up11 / ExpUp11) * (1.9 * NumPieces11 /  Up11) * NumPiecesDefect11 / NumPieces11, 0) * 100 AS S11,
    COALESCE((Up12 / ExpUp12) * (1.9 * NumPieces12 /  Up12) * NumPiecesDefect12 / NumPieces12, 0) * 100 AS S12,
    COALESCE((Up13 / ExpUp13) * (1.9 * NumPieces13 /  Up13) * NumPiecesDefect13 / NumPieces13, 0) * 100 AS S13,
    COALESCE((Up14 / ExpUp14) * (1.9 * NumPieces14 /  Up14) * NumPiecesDefect14 / NumPieces14, 0) * 100 AS S14,
    COALESCE((Up15 / ExpUp15) * (1.9 * NumPieces15 /  Up15) * NumPiecesDefect15 / NumPieces15, 0) * 100 AS S15,
    COALESCE((Up16 / ExpUp16) * (1.9 * NumPieces16 /  Up16) * NumPiecesDefect16 / NumPieces16, 0) * 100 AS S16
FROM
(
    SELECT
        SUM(1 & ChangedState & IsWorking & HasDefect) AS NumPiecesDefect1,
        SUM((1  & ChangedState & IsWorking) = 1 ) AS NumPieces1,
        SUM((ExpectedWorking&1 != 1) * TimeSinceLastUpdate) AS ExpUp1,
        SUM((IsWorking&1 != 1) * TimeSinceLastUpdate) AS Up1,

        SUM((2 & ChangedState & IsWorking & HasDefect) = 2) AS NumPiecesDefect2,
        SUM((2  & ChangedState & IsWorking) = 2 ) AS NumPieces2,
        SUM((ExpectedWorking&2 != 2) * TimeSinceLastUpdate) AS ExpUp2,
        SUM((IsWorking&2 != 2) * TimeSinceLastUpdate) AS Up2,

        SUM((4 & ChangedState & IsWorking & HasDefect) = 4) AS NumPiecesDefect3,
        SUM((4  & ChangedState & IsWorking) = 4 ) AS NumPieces3,
        SUM((ExpectedWorking&4 != 4) * TimeSinceLastUpdate) AS ExpUp3,
        SUM((IsWorking&4 != 4) * TimeSinceLastUpdate) AS Up3,

        SUM((8 & ChangedState & IsWorking & HasDefect) = 8) AS NumPiecesDefect4,
        SUM((8  & ChangedState & IsWorking) = 8 ) AS NumPieces4,
        SUM((ExpectedWorking&8 != 8) * TimeSinceLastUpdate) AS ExpUp4,
        SUM((IsWorking&8 != 8) * TimeSinceLastUpdate) AS Up4,

        SUM((16 & ChangedState & IsWorking & HasDefect) = 16) AS NumPiecesDefect5,
        SUM((16  & ChangedState & IsWorking) = 16 ) AS NumPieces5,
        SUM((ExpectedWorking&16 != 16) * TimeSinceLastUpdate) AS ExpUp5,
        SUM((IsWorking&16 != 16) * TimeSinceLastUpdate) AS Up5,

        SUM((32 & ChangedState & IsWorking & HasDefect) = 32) AS NumPiecesDefect6,
        SUM((32  & ChangedState & IsWorking) = 32 ) AS NumPieces6,
        SUM((ExpectedWorking&32 != 32) * TimeSinceLastUpdate) AS ExpUp6,
        SUM((IsWorking&32 != 32) * TimeSinceLastUpdate) AS Up6,

        SUM((64 & ChangedState & IsWorking & HasDefect) = 64) AS NumPiecesDefect7,
        SUM((64  & ChangedState & IsWorking) = 64 ) AS NumPieces7,
        SUM((ExpectedWorking&64 != 64) * TimeSinceLastUpdate) AS ExpUp7,
        SUM((IsWorking&64 != 64) * TimeSinceLastUpdate) AS Up7,
        SUM((128 & ChangedState & IsWorking & HasDefect) = 128) AS NumPiecesDefect8,
        SUM((128  & ChangedState & IsWorking) = 128 ) AS NumPieces8,
        SUM((ExpectedWorking&128 != 128) * TimeSinceLastUpdate) AS ExpUp8,
        SUM((IsWorking&128 != 128) * TimeSinceLastUpdate) AS Up8,

        SUM((256 & ChangedState & IsWorking & HasDefect) = 256) AS NumPiecesDefect9,
        SUM((256  & ChangedState & IsWorking) = 256 ) AS NumPieces9,
        SUM((ExpectedWorking&256 != 256) * TimeSinceLastUpdate) AS ExpUp9,
        SUM((IsWorking&256 != 256) * TimeSinceLastUpdate) AS Up9,

        SUM((512 & ChangedState & IsWorking & HasDefect) = 512) AS NumPiecesDefect10,
        SUM((512  & ChangedState & IsWorking) = 512 ) AS NumPieces10,
        SUM((ExpectedWorking&512 != 512) * TimeSinceLastUpdate) AS ExpUp10,
        SUM((IsWorking&512 != 512) * TimeSinceLastUpdate) AS Up10,

        SUM((1024 & ChangedState & IsWorking & HasDefect) = 1024) AS NumPiecesDefect11,
        SUM((1024  & ChangedState & IsWorking) = 1024 ) AS NumPieces11,
        SUM((ExpectedWorking&1024 != 1024) * TimeSinceLastUpdate) AS ExpUp11,
        SUM((IsWorking&1024 != 1024) * TimeSinceLastUpdate) AS Up11,

        SUM((2048 & ChangedState & IsWorking & HasDefect) = 2048) AS NumPiecesDefect12,
        SUM((2048  & ChangedState & IsWorking) = 2048 ) AS NumPieces12,
        SUM((ExpectedWorking&2048 != 2048) * TimeSinceLastUpdate) AS ExpUp12,
        SUM((IsWorking&2048 != 2048) * TimeSinceLastUpdate) AS Up12,

        SUM((4096 & ChangedState & IsWorking & HasDefect) = 4096) AS NumPiecesDefect13,
        SUM((4096  & ChangedState & IsWorking) = 4096 ) AS NumPieces13,
        SUM((ExpectedWorking&4096 != 4096) * TimeSinceLastUpdate) AS ExpUp13,
        SUM((IsWorking&4096 != 4096) * TimeSinceLastUpdate) AS Up13,

        SUM((8192 & ChangedState & IsWorking & HasDefect) = 8192) AS NumPiecesDefect14,
        SUM((8192  & ChangedState & IsWorking) = 8192 ) AS NumPieces14,
        SUM((ExpectedWorking&8192 != 8192) * TimeSinceLastUpdate) AS ExpUp14,
        SUM((IsWorking&8192 != 8192) * TimeSinceLastUpdate) AS Up14,

        SUM((16384 & ChangedState & IsWorking & HasDefect) = 16384) AS NumPiecesDefect15,
        SUM((16384  & ChangedState & IsWorking) = 16384 ) AS NumPieces15,
        SUM((ExpectedWorking&16384 != 16384) * TimeSinceLastUpdate) AS ExpUp15,
        SUM((IsWorking&16384 != 16384) * TimeSinceLastUpdate) AS Up15,

        SUM((32768 & ChangedState & IsWorking & HasDefect) = 32768) AS NumPiecesDefect16,
        SUM((32768  & ChangedState & IsWorking) = 32768 ) AS NumPieces16,
        SUM((ExpectedWorking&32768 != 32768) * TimeSinceLastUpdate) AS ExpUp16,
        SUM((IsWorking&32768 != 32768) * TimeSinceLastUpdate) AS Up16
    FROM sensors
    WHERE $__timeFilter(Timestamp)
) oeeMetrics
```

Excerto de Código 25 - Query do cálculo da métrica OEE