

**FACULDADE DE TECNOLOGIA SENAI JARAGUÁ DO SUL  
GRADUAÇÃO TECNOLÓGICA EM SISTEMAS PARA INTERNET**

**AGIL IT  
SISTEMA GERENCIADOR DE ORDENS DE MANUTENÇÃO DE  
EQUIPAMENTO**

JULIO CESAR THOMAZELLI JUNIOR  
LUCAS MATHEUS DA SILVA GONÇALVES  
MÁRCIO HENRIQUE MEIER

**JARAGUÁ DO SUL, SC  
2020**

**JULIO CESAR THOMAZELLI JUNIOR  
LUCAS MATHEUS DA SILVA GONÇALVES  
MÁRCIO HENRIQUE MEIER**

**AGIL IT  
SISTEMA GERENCIADOR DE ORDENS DE MANUTENÇÃO DE  
EQUIPAMENTO**

Trabalho de Conclusão de Curso apresentado à  
Faculdade de Tecnologia SENAI Jaraguá do Sul  
como requisito parcial para obtenção do título de  
Tecnólogo em Sistemas para Internet

Professora Orientadora:  
Msc. Tathiana Duarte do Amarante

**JARAGUÁ DO SUL, SC  
2020**

## LISTA DE FIGURAS

Figura 2.1 –Planejamento de Projeto . . . . .	12
Figura 2.2 –Exemplo de modelo conceitual especificado em UML . . . . .	13
Figura 3.1 –Cenário Atual . . . . .	15
Figura 3.2 –Cenário Agil.It . . . . .	17
Figura 5.1 –Fluxo do AGIL.IT . . . . .	22
Figura 5.2 –Caso de Uso do AGIL.IT . . . . .	23
Figura 5.3 –Entidade de Relacionamento . . . . .	24
Figura 6.1 –Cronograma de Aprendizagem: Terceiro Semestre . . . . .	25
Figura 6.2 –Cronograma de Aprendizagem: Quarto Semestre . . . . .	26
Figura 6.3 –Cronograma de Aprendizagem: Quinto Semestre . . . . .	26
Figura 6.4 –Cronograma de Aprendizagem: Sexto Semestre . . . . .	26
Figura 6.5 –Diagrama de Causa e Efeito . . . . .	28
Figura 6.6 –EAP AGIL.IT . . . . .	30
Figura 6.7 –Fluxo e Processos AGIL.IT . . . . .	32
Figura 6.8 –Cadastro de Usuários . . . . .	34
Figura 6.9 –Monitor de Ordem de Manutenção: Cards . . . . .	35
Figura 6.10 Checklist de Segurança . . . . .	36
Figura 6.11 Monitor . . . . .	37
Figura 6.12 Notificações . . . . .	38
Figura 6.13 Ordem de Manutenção . . . . .	39
Figura 8.1 –Critério Geral de Apresentação . . . . .	59
Figura 8.2 –Critérios Gerais . . . . .	59

## **LISTA DE ABREVIATURAS E SIGLAS**

MVP	Minimum Viable Product
API	Application Programming Interface
ORM	Object-Relational Mapping
OM	Ordem de Manutenção
ERP	Software de gerenciamento de empresas
SAP	ERP alemão conhecido mundialmente
WEB	Internet
UC	Unidade Curricular

# SUMÁRIO

<b>1</b>	<b>Introdução</b>	<b>7</b>
1.1	Problema	7
1.2	Objetivos Gerais	8
1.3	Objetivos Específicos	8
1.4	Justificativa	8
1.5	Método de Trabalho	9
1.6	Organização do Trabalho	9
1.7	Glossário	9
<b>2</b>	<b>Fundamentação Teórica</b>	<b>10</b>
2.1	Sistemas Computacionais	10
2.2	Computação Verde	10
2.3	Planejamento e Gerenciamento de Projetos de Software	11
2.4	UML	12
2.5	SCRUM	13
2.6	GIT	13
2.7	ORM	13
<b>3</b>	<b>Descrição Geral do Sistema</b>	<b>14</b>
3.1	Descrição do Problema	14
3.1.1	Cenário Atual	14
3.1.2	Cenário Com Agil.It	15
3.2	Principais Envolvidos e suas Características	16
3.2.1	Usuários do Sistema	16
3.2.2	Desenvolvedores do Sistema	16
3.3	Regras de Negócio	17
<b>4</b>	<b>Pesquisa de Anterioridade</b>	<b>19</b>
4.1	Produttivo	19
4.2	SoftByte	19
4.3	ProdWin	19
4.4	Diferenciais Agil.It	19
<b>5</b>	<b>Requisitos do Sistema</b>	<b>20</b>
5.1	Requisitos Funcionais	20
5.1.1	Cadastrros	20
5.1.2	Consultas	20
5.1.3	Funcionalidades	21
5.2	Requisitos Não Funcionais	21

5.2.1	Funcionalidades . . . . .	21
5.3	Fluxograma do Sistema Desenvolvido . . . . .	21
5.4	Diagrama de Caso de Uso do Sistema . . . . .	22
5.5	Entidade de Relacionamento do Banco de Dados . . . . .	23
<b>6</b>	<b>Planejamento do Sistema . . . . .</b>	<b>25</b>
6.1	Cronograma do Projeto . . . . .	25
6.2	Análise de riscos . . . . .	26
6.2.1	Diagrama de Causa e Efeito . . . . .	27
6.3	PMBOK . . . . .	28
6.3.1	Estrutura Analítica do Projeto . . . . .	28
6.3.2	Fluxo de Processos . . . . .	31
6.4	Protótipo . . . . .	33
6.5	Aplicação WEB . . . . .	33
6.5.1	Cadastro de Usuário . . . . .	34
6.5.2	Monitor de Ordem de Manutenção: Cards . . . . .	35
6.5.3	Checklist de Segurança . . . . .	36
6.6	Aplicação Mobile . . . . .	36
6.6.1	Monitor . . . . .	36
6.6.2	Central de Notificações . . . . .	38
6.6.3	Ordem de Manutenção . . . . .	39
<b>7</b>	<b>Implementação . . . . .</b>	<b>40</b>
7.1	Tecnologias . . . . .	40
7.1.1	Aplicação WEB . . . . .	40
7.1.2	Aplicação Mobile . . . . .	40
7.1.3	Aplicação do Servidor . . . . .	41
7.2	Códigos Desenvolvidos . . . . .	41
7.2.1	WEB . . . . .	41
7.2.1.1	Componentes . . . . .	41
7.2.1.2	Cookies . . . . .	42
7.2.1.3	Helpers . . . . .	43
7.2.2	Mobile . . . . .	47
7.2.2.1	Ações da Ordem de Manutenção . . . . .	47
7.2.2.2	Dados Armazenados no Local Storage . . . . .	49
7.2.3	Servidor . . . . .	51
7.2.3.1	Mapeamento do Banco de Dados com TypeOrm . . . . .	51
7.2.3.2	Validação de Objetos com o Class Validator . . . . .	53
7.2.3.3	TypeOrm: Query as Function . . . . .	54
7.2.3.4	API . . . . .	55
7.2.3.5	Estratégia de Exclusão de Dados . . . . .	57

<b>8 Testes de Usabilidade</b>	<b>58</b>
8.0.1 Elaboração dos Testes Aplicados	58
8.0.2 Discussão dos Resultados	58
<b>9 Considerações e Trabalhos Futuros</b>	<b>61</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS</b>	<b>62</b>

# 1 INTRODUÇÃO

A faculdade SENAI (Serviço Nacional de Aprendizagem Industrial)<sup>1</sup> visa sistematizar os conhecimentos adquiridos pelos estudantes durante o decorrer do curso, como também, oferecer vivência prática-profissional mediante a aplicação dos conhecimentos em situações reais. Além disso, busca propiciar ao estudante o contato com o universo acadêmico da iniciação científica, através da implantação do *Projeto Integrador*, que tem exatamente essas características em locais que está sendo implantado. Portanto o Senai está em parceria com a empresa Duas Rodas desenvolvendo este projeto.

A empresa Duas Rodas possui sua sede em Jaraguá do Sul, Santa Catarina e tem como principal ramo a indústria de alimentos<sup>2</sup>. Para a execução de suas tarefas diárias, seus funcionários contam com o auxílio direto de máquinas de pequeno, médio e grande porte, e sabe-se que esses equipamentos necessitam de manutenção preventiva, preditiva e corretiva de tempos em tempos. Para tanto, se faz necessário ajustes adequados e rápidos para que não haja baixa produtividade e consequentemente uma redução em seus proveitos.

Esses equipamentos precisam passar por uma série de cuidados e estar em bom estado para o uso dos seus colaboradores. Portanto, é de extrema importância para empresa Duas Rodas ter um controle e um bom fluxo de manutenções desses equipamentos, de modo a não haver uma inatividade desnecessária, e quando necessário a manutenção, esta seja rápida e eficiente.

Entendendo o cenário acima, este trabalho se propõe a melhorar o processo de manutenção de equipamentos da empresa Duas Rodas, através de um sistema para gerenciamento de ordens de manutenção que irá linkar a parte administrativa com o operacional, fazendo com que o administrador possa ter uma comunicação rápida com o manutentor e visualizar em tempo real o andamento da manutenção. Sendo assim, este processo ficará mais eficiente e totalmente digital, fazendo com que a empresa reduza a quantidade de papel impresso, gerando assim uma economia e tornando-a mais sustentável.

## 1.1 Problema

O processo de manutenção de equipamentos de um parque fabril necessita de um acompanhamento constante das tarefas que serão executadas. No cenário da empresa Duas Rodas, o acompanhamento das ordens de manutenção ocorre de forma impressa, o que torna difícil a gestão da mesma, pois necessita que as informações sejam transportadas fisicamente entre os colaboradores, gestores e técnicos. No final da manutenção é preciso que seja digitado ao ERP SAP todo o processo executado pelos manutentores nos equipamentos.

---

<sup>1</sup> <<http://sc.senai.br/pt-br/faculdade-senai-jaragua-do-sul>>

<sup>2</sup> <<https://www.duasrodas.com/>>

## 1.2 Objetivos Gerais

O presente estudo visa aperfeiçoar os processos administrativos no que diz respeito às ordens de manutenção feitas pela empresa Duas Rodas, através de um sistema digital que irá facilitar a execução da manutenção de equipamentos e a gestão do processo, desde a abertura de uma ordem até o seu encerramento.

## 1.3 Objetivos Específicos

- Identificar referências bibliográficas voltadas a softwares e gestão de manutenção.
- Elaborar a análise dos requisitos e a prototipação do sistema.
- Desenvolver o sistema.
- Analisar e validar o desenvolvimento do sistema.
- Testes necessários.
- Apresentar o sistema desenvolvido conforme o problema proposto.

## 1.4 Justificativa

O desenvolvimento do projeto irá tornar o processo de manutenção de equipamentos da empresa Duas Rodas mais eficiente, através de um sistema de gerenciamento de manutenção que irá interligar toda a parte administrativa ao setor operacional, deixando o processo mais fluído, irá também reduzir os gastos com papéis e o tempo demandado para a gestão do processo.

O ciclo de vida da informação, é a mudança no valor da informação com o decorrer do tempo. Quando os dados são criados, muitas vezes possuem seu valor mais alto e são usados com frequência. Porém, conforme o tempo passa, os dados não digitais se perdem com mais facilidade e tem menos valor para a organização. As empresas modernas precisam que seus dados estejam protegidos, íntegros e disponíveis em tempo integral. Com isso, os sistemas digitais podem fornecer a otimização apropriada de armazenamento, uma política eficaz de gerenciamento de dados necessária para dar suporte e potencializar os benefícios da empresa, explica (SOMASUNDARAM, 2009).

As empresas não devem apenas beneficiar os proprietários, mas toda a sociedade, especialmente as classes mais penalizadas. Não basta somente a responsabilidade social, pois a comunidade deve ser pensada junto a interface com a natureza, da qual é um subsistema. Com isso, se introduziu a responsabilidade socioambiental, programas que tem por objetivo diminuir a pressão que a atividade produtiva e industrialista faz sobre o meio ambiente. As inovações tecnológicas ajudam neste propósito sem mudar o rumo do crescimento e desenvolvimento que implica a dominação da natureza, de acordo com (BOFF, 2017).

Exposto isso, o sistema proposto objetiva a diminuição do consumo excessivo de papel no processo de manutenção de equipamentos, deixando a empresa Duas Rodas mais sustentável e preparada para as tendências futuras envolvendo o meio ambiente.

## 1.5 Método de Trabalho

Para o desenvolvimento do projeto foi utilizado a metodologia SCRUM no formato MVP (Produto Mínimo Viável).

O Scrum, criado em 1993 por Ken Schwaber e Jeff Sutherland, tem a origem de seu nome no “jogo de rúgbi e se refere à maneira como um time trabalha junto para avançar com a bola no campo. Alinhamento cuidado, unidade de propósito, clareza de objetivo, tudo se unindo (ROCHA, 2015). A metodologia SCRUM consiste em quebrar o sistema em várias partes pequenas e fazer entregas a cada ciclo, que normalmente possuem de 1 a 2 semanas. Enquanto o formato MVP prega o desenvolvimento de algo com o menor investimento possível, a fim da validação da ideia ou conceito utilizado.

## 1.6 Organização do Trabalho

O desenvolvimento do trabalho será composto pela fundamentação teórica, descrição geral do sistema, pesquisa de anterioridade, requisitos do sistema, planejamento do sistema e por fim a implementação do sistema.

## 1.7 Glossário

SCRUM: Metodologia ágil de desenvolvimento de projetos.

MVP: Produto com o mínimo valor possível, visado para validação da ideia do projeto.

API: Interface para comunicação entre diferentes aplicações.

ORM: Tecnologia que auxilia o gerenciamento do banco de dados através da modelagens de classes.

Express: Tecnologia que abstrai requisições web.

Sequelize: Biblioteca de ORM para bancos relacionais, incluindo SQL Server.

Feedback: Retorno a um acontecimento.

Software: Programa de computador.

UC: Unidade Curricular.

EAP: Estrutura Analítica do Projeto.

PMBOK: Conhecimento em Gerenciamento de Projetos.

## 2 FUNDAMENTAÇÃO TEÓRICA

### 2.1 Sistemas Computacionais

Os sistemas computacionais tomaram conta da sociedade atual, o uso de tecnologias computacionais vem tendo muito espaço na comunidade, seja fazendo uso de um computador pessoal, smartphone ou um tablet. Sendo assim, tais dispositivos foram criado essencialmente para satisfazer as necessidades das empresas de forma a garantir e resolver seus problemas organizacionais, ou seja a criação de uma ferramenta que possa armazenar e cruzar dados e informações sem que sejam necessárias pilhas de papéis. Estes sistemas garantem às empresas a possibilidade de atingir mercados em locais mais distantes. (MATTIOLI, 2020).

Destaca-se, que o sistema da informação é um tipo especializado de sistema, que pode ser definido de várias formas. Com isso, pode-se entender que sistemas são séries de elementos ou componentes inter-relacionados que coletam entrada de dados, manipulando-os, processando-os, disseminando a saída dos dados e informações, fornecendo assim um mecanismo de *feedback*. (STAIR, 2008).

Atualmente a palavra **sistema** é mal utilizada, usa-se de forma indiscriminada e sem qualquer tipo de fundamento, ou ainda, é usada para expressar determinadas situações dentro de um software, principalmente nos meios empresariais conforme explica (ROSSINI, 2006).

Diante disso, é importante conceituar dados, sendo estes como fatos básicos, por exemplo, o nome e a quantidade de horas trabalhadas em uma semana de um funcionário, quantidade de peça em estoque ou pedidos. Importante mencionar que as informações são compostas por um conjunto de fatos organizados de modo a terem valor adicional, além do valor dos fatos propriamente ditos. Portanto, quando dados são organizados ou alcançados de maneira significativa, se transformam em informações. (STAIR, 2008).

Conforme informações acima dispostas, apesar do termo sistema estar muito difundido e utilizado muitas vezes de forma leviana, seu significado é bastante preciso. Um sistema é um conjunto de elementos que trabalham de forma integrada a atingir uma ou mais finalidades. Para que o sistema funcione corretamente, é necessário transformar dados em informações de forma que seus objetivos sejam alcançados, desde a finalidade única de cada elemento até a totalidade das funcionalidades integradas do mesmo. Com base nisso, na sequência será abordado o assunto referente ao gerenciamento de ordem de manutenção.

### 2.2 Computação Verde

Computação ou TI Verde é um conjunto de iniciativas de investimento na implantação, uso e gerenciamento a fim de minimizar o impacto negativo ambiental na área de tecnologia da informação. (QUEIRÓS, 2020).

O investimento de capital em computação verde é composto por três dimensões: es-

trutural, humano e relacional. O capital estrutural refere-se a infraestrutura da TI que engloba *hardware*, *software*, redes e tecnologia da informação. O capital humano é a capacidade e experiência dos profissionais de TI a respeito de conservação de energia em tecnologia e desenvolvimento pessoal com capacidades em TI verde, obtida através de treinamento e estudos. Por último, o capital relacional envolve o gerenciamento da TI verde e a relação das organizações com seus parceiros e usuários, implementando conceitos de proteção ambiental em produtos e serviços. (QUEIRÓS, 2020).

Com isso, a adoção das práticas de TI verde propicia uma operação mais sustentável às organizações, gerando economia com energia, papel, água, transporte, espaço físico, manutenção e descarte, proporcionando assim valor tanto para a organização quanto para a sociedade (MOURA, 2017).

## 2.3 Planejamento e Gerenciamento de Projetos de Software

Um projeto é um empreendimento temporário que objetiva criar um produto, resultado ou serviço único, que no caso de projeto de software é o sistema em funcionamento (COUTO, 2018).

Os projetos de software apresentam particularidades principalmente por desenvolver produtos incompreensíveis e pela dificuldade do seu gerenciamento, além da falta de comunicação entre os gerentes/desenvolvedores e os clientes/usuários (PRADO, 1999). As etapas de desenvolvimento seguem um ciclo de vida com fases próprias, tais como a especificação de requisitos, análise, projeto, implementação, testes e implantação (MOURA, 2018).

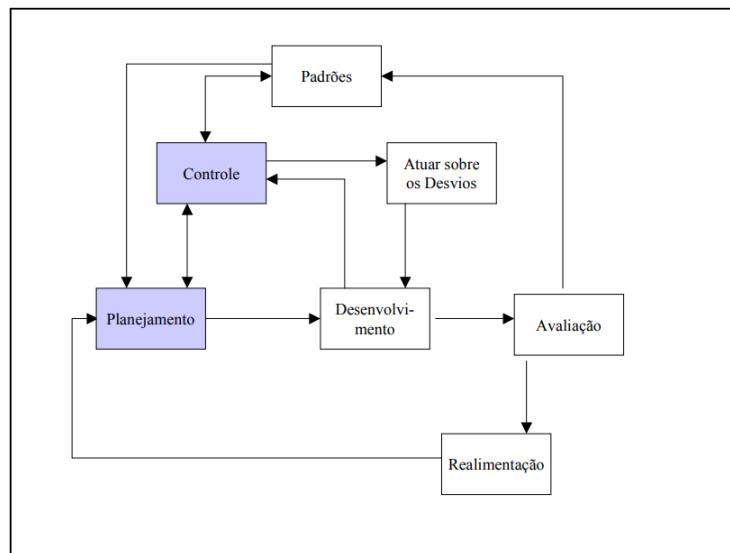
A competitividade e os avanços tecnológicos provocaram um aumento na exigência de qualidade e na complexidade de decisões administrativas. Para obter êxito em um projeto é necessário planejar e gerenciar com eficiência a execução de diversas atividades independentes, pois o grau de risco e incerteza quanto ao sucesso do projeto são elevados (HAUFE, 2001).

Para gerir um projeto, é possível utilizar modelos mais prescritivos como o PMBoK, mais adaptativos como o Scrum ou híbridos. Os modelos prescritivos possuem uma estrutura formal de elementos do processo como atividades e tarefas, um fluxo de trabalho que descreve como cada um destes elementos deve ocorrer e como eles são relacionados. A busca pela estrutura e a ordem é uma característica importante deste tipo de modelo, que normalmente são compostos por *frameworks* como PMBoK, PRINCE2 e IPMA(COUTO, 2018). Enquanto os modelos prescritivos presam principalmente pela metodologia e pelo fluxo dos processos, modelos adaptativos como os oriundos do manifesto ágil, são focados na interação entre os usuários no processo de construção do produto através de uma abordagem iterativa e adaptativa(COUTO, 2018). Os métodos ágeis possuem valores fundamentais, que foram definidos no manifesto ágil, de 2001:

- Indivíduos e interação entre eles mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Software em funcionamento mais que documentação abrangente;

- Responder a mudanças mais que seguir um plano.

Figura 2.1: Planejamento de Projeto



Fonte: (HAUFE, 2001)

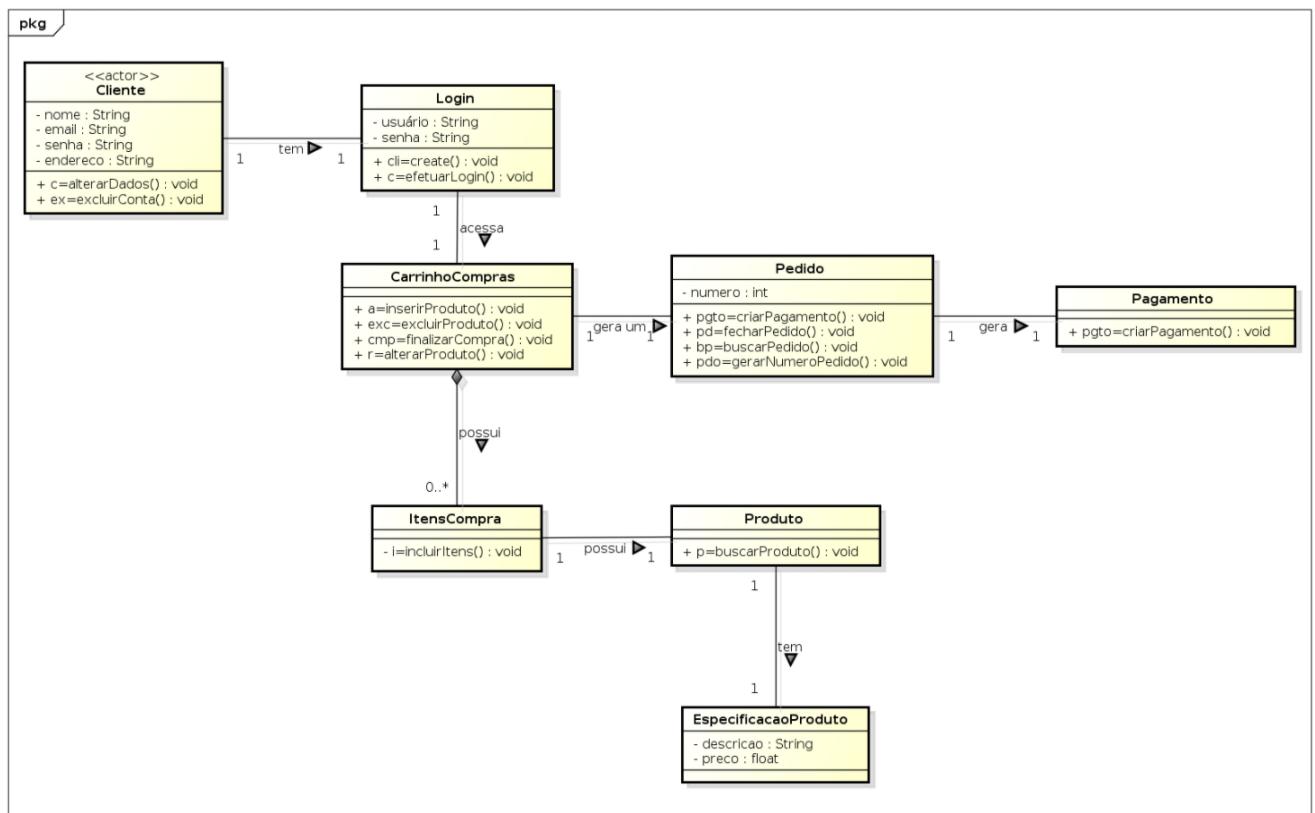
A figura 2.1 mostra o ciclo de gerenciamento de projetos de software. O controle atua tanto no planejamento quanto no desenvolvimento, proporcionando desvios que originam ações corretivas. Entretanto, a avaliação ao final do projeto, reabastece o planejamento com novos projetos e ideias, prosseguindo assim em ciclo indeterminado. Já os padrões são definidos a partir dos controles e das avaliações para controlar e planejar o decorrer do projeto.

## 2.4 UML

A UML (*Unified Modeling Language*) foi desenvolvida com o propósito de fornecer a arquitetos e engenheiros, ferramentas para análise, design e implementação de software, bem como oferecer suporte à modelagem de negócio (GASPARINI, 2018). A UML é dividido em dois grupos: Estrutural que permite representar a parte estática do sistema e Comportamental que permite representar objetos dinâmicos no sistema (SILVA, 2018).

A figura 2.2 representa um modelo de diagrama de classes que aborda uma experiência em um e-commerce. Um cliente pode fazer o login na plataforma, inserir, excluir e alterar produtos e finalizar a compra. Cada item do carrinho possui o produto e a quantidade e o produto possui preço e descrição. Após finalizar o carrinho a plataforma gera um pedido que gera um pagamento.

Figura 2.2: Exemplo de modelo conceitual especificado em UML



Fonte: (PRIMO, 2014)

## 2.5 SCRUM

## 2.6 GIT

## 2.7 ORM

### 3 DESCRIÇÃO GERAL DO SISTEMA

O Software de gestão empresarial utilizado pela empresa Duas Rodas atualmente é o SAP, nele é controlado todo fluxo operacional e de funcionamento da empresa. O Agil.It trabalhará em conjunto com o SAP no módulo de manutenção de equipamentos, atuando como meio digitalizado e fazendo ponte com o SAP onde irá integrar informações prévias e competentes às ordens de manutenção.

Dividido em duas aplicações independentes, o sistema é capaz de trabalhar de forma autônoma, podendo receber e enviar dados via API ou realizar cadastros manualmente na aplicação WEB. Enquanto no aplicativo será possível acompanhar as ordens, fazer apontamentos e realizar breves consultas.

#### 3.1 Descrição do Problema

A gestão do processo de manutenção de máquinas é feita de forma manual. Desta forma há bastante retrabalho para repassar todos os dados ao sistema, outra preocupação é o gasto com folhas de papel. Em adendo, não há uma boa análise dos problemas comuns, equipamentos mais problemáticos, eficiência dos técnicos, entre outras.

A gestão do processo de manutenção de máquinas da empresa é feita de forma manual. Sendo assim, há bastante retrabalho por parte dos administradores e líderes do setor em questão. O tempo consumido pelos colaboradores para a correção dos problemas não permite tempo hábil para fazer a análise dos defeitos ocorridos, sendo assim existe uma dificuldade em realizar métricas desde equipamentos problemáticos até ineficiência dos técnicos.

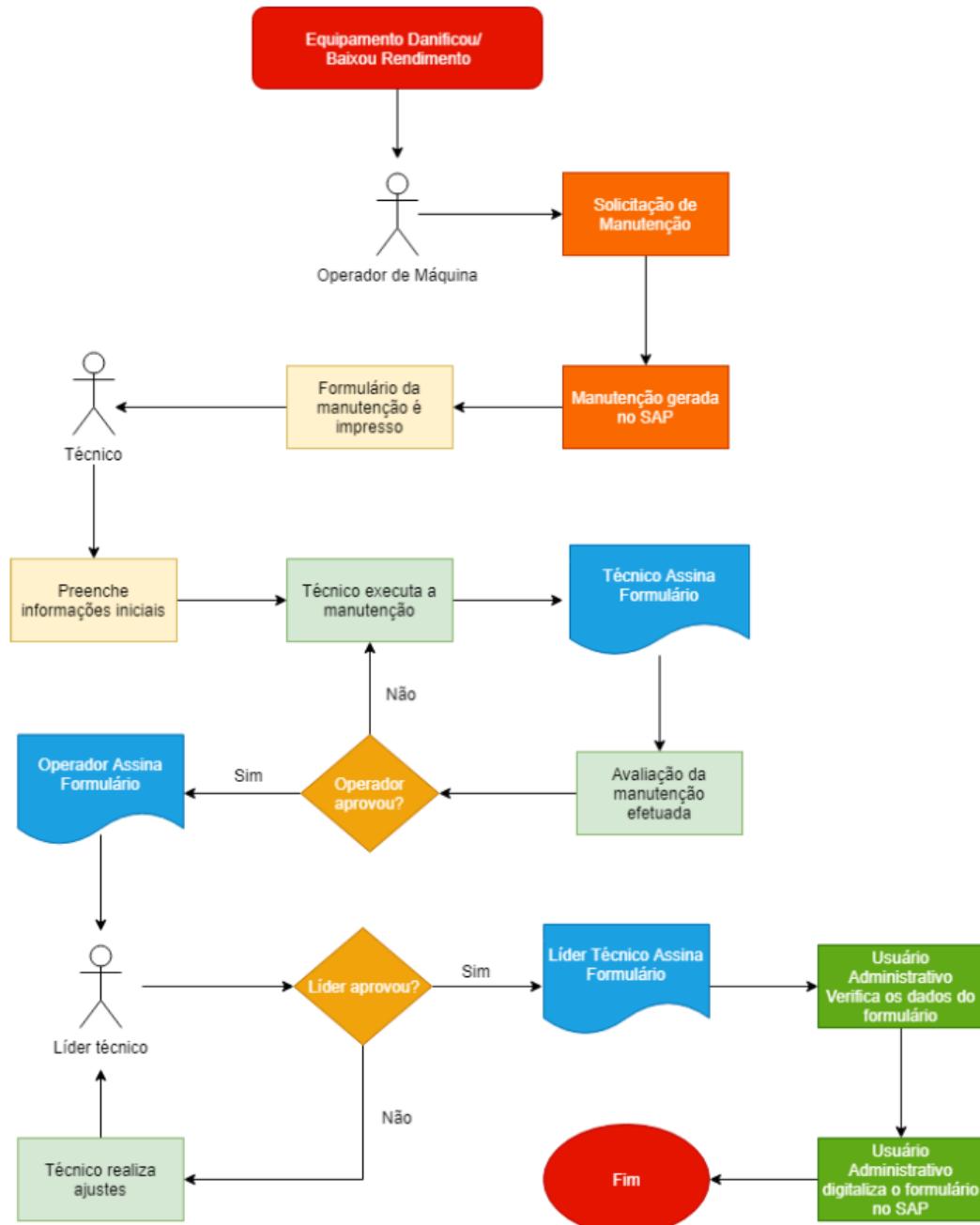
Outra preocupação que objetiva a automatização das ordens de manutenção é o consumo excessivo de papel, pois a tendência das empresas é serem cada vez mais sustentáveis e portanto, ao digitalizar o processo de ordem de manutenção, a empresa Duas Rodas poderá ter maiores índices de sustentabilidade e alcançar um maior destaque entre as demais empresas.

##### 3.1.1 Cenário Atual

No processo de manutenção de equipamentos, a empresa Duas Rodas emite uma ordem de manutenção pelo SAP. Após a emissão, é impresso um formulário e entregue ao técnico elencado para a realização da manutenção. O técnico então preenche os campos básicos do formulário e depois preenche os campos que detalham as operações realizadas, bem como os componentes utilizados. Ao finalizar a manutenção, são colhidas 3 assinaturas, a do operador da máquina, a do técnico da manutenção e a do líder de manutenção, sendo que cada uma das partes pode negar a assinatura e solicitar alterações na manutenção realizada. Com todas as assinaturas rubricadas, o formulário é enviado a um funcionário administrativo que digitaliza as informações preenchidas no SAP. Além de gerar um retrabalho imenso, há um problema com a caligrafia dos manutentores, o que dificulta a digitalização dos dados. Com o volume atual de

manutenções, é inviável manter o fluxo dessa forma. Com isso, entra o projeto Agil It. Atuando como meio digitalizador, faz a ponte entre a manutenção na fábrica e o sistema SAP, fazendo com que não haja mais retrabalho e eliminando totalmente os papéis utilizados nos formulários.

Figura 3.1: Cenário Atual



Fonte: Os Autores(2020)

### 3.1.2 Cenário Com Agil.It

No processo de manutenção de equipamentos com a Agil.It, a ordem de manutenção será gerada no SAP e os dados serão integrados ao sistema Agil.It. Ao ser integrado, o técnico de manutenção receberá uma notificação informando que tem uma nova OM e ele poderá

visualizá-la em seu monitor no aplicativo mobile ou na aplicação WEB. Ele poderá então iniciar a manutenção e fazer os apontamentos das operações e componentes. Após finalizar a manutenção, o técnico assinará a ordem de manutenção. Após isso, o operador do equipamento será notificado e a OM ficará disponível para sua avaliação, podendo este aceitar ou recusar de acordo com as realizações do trabalho. Caso recuse, o operador deve informar o motivo da rejeição ao sistema que então, irá reabrir a OM e solicitar ao técnico para verificar. Caso o operador aceite a execução do trabalho de manutenção, o mesmo deverá assinar a OM e com isto, o sistema notificará o líder técnico, sendo o trabalho deste, averiguar a execução da tarefa, o mesmo também poderá rejeitar e solicitar alterações. Após concluir, ele deve assinar a OM. Com as 3 assinaturas colhidas, o sistema irá notificar o usuário administrador e ele poderá liberar a OM para integração com o SAP.

## 3.2 Principais Envolvidos e suas Características

### 3.2.1 Usuários do Sistema

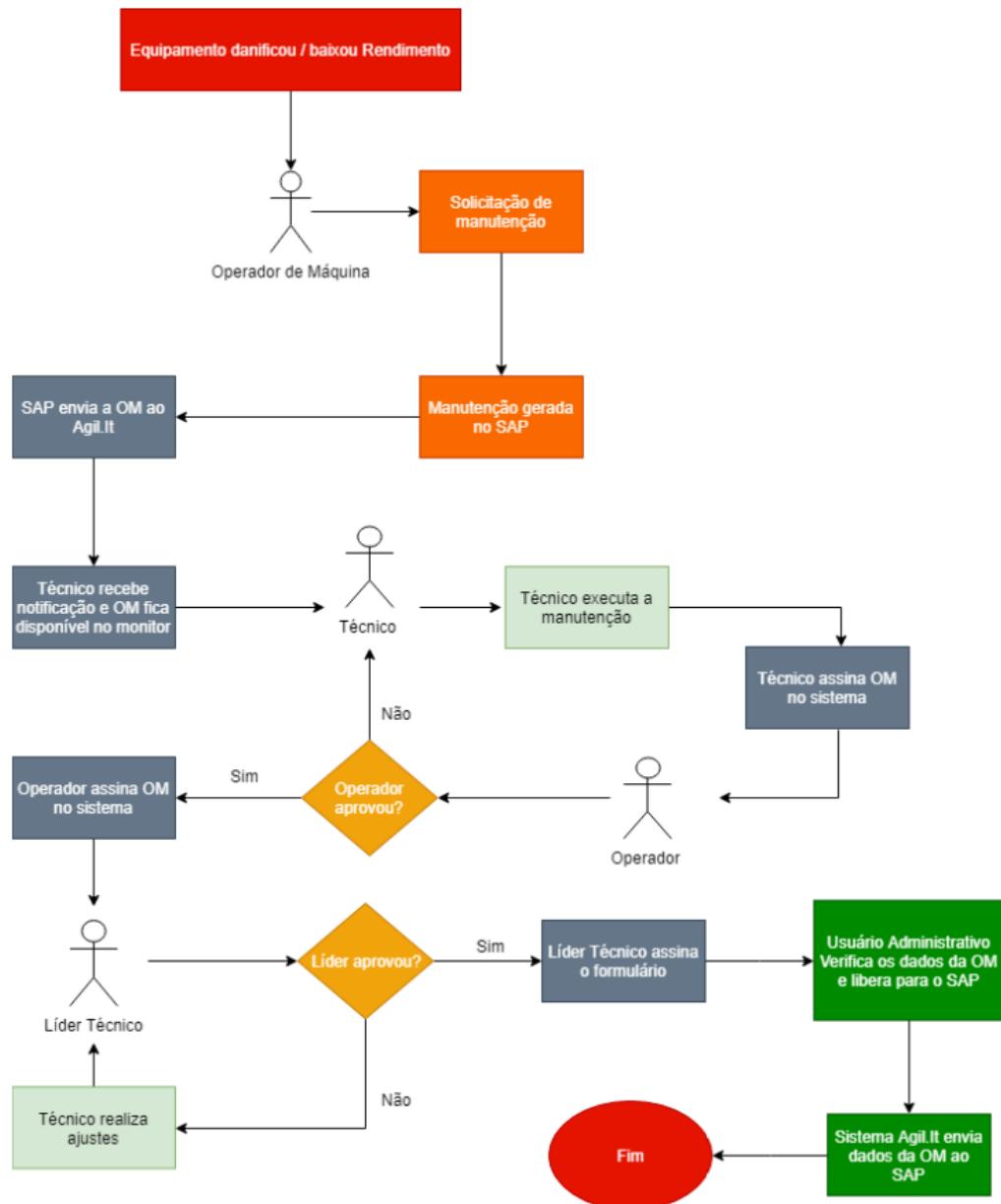
O Sistema contemplará 4 tipos de usuários, o administrador, o manutentor, o chefe de manutenção e o operário. Cada usuário terá acessos e funções diferentes no sistema. Todos terão acesso tanto a parte web da aplicação quanto a parte mobile (aplicativo).

- O usuário administrador será responsável pelos cadastros gerais, poderá consultar e finalizar ordens de manutenção.
- O manutentor terá a visão de ordens de manutenção pendentes para ele, podendo iniciar ou pausar alguma a qualquer momento. Depois de finalizada, poderá realizar a assinatura da ordem.
- O chefe de manutenção receberá requisições de ordens de manutenção feitas por operadores e irá cadastrá-las. Poderá distribuir as ordens aos manutentores, realizar consultas gerais e assinar as ordens.
- O operador, que poderá requisitar uma ordem de manutenção ao administrador, acompanhar as ordens solicitadas por ele e assinar a ordem após a conclusão.

### 3.2.2 Desenvolvedores do Sistema

Desenvolvedor	Atividade
Julio	Aplicativo Mobile
Lucas	Aplicativo Web
Márcio	Back End / Integração

Figura 3.2: Cenário Agil.It



Fonte: os autores (2020)

### 3.3 Regras de Negócio

No contexto da Engenharia de Software as Regras de Negócios são tratadas como alguns Requisitos de Software, pois sem elas, o software não existiria. Regras de negócios são premissas e restrições aplicadas a uma operação comercial de uma empresa, que precisam ser atendidas para que o negócio funcione da maneira esperada (CRERIE, 2008).

Segundo (PÁDUA, 2001) Regras de negócios são mais do que declarações sobre campos de dados ou implementação do sistema, elas definem tarefas dos atores da organização, os serviços que a organização dispõem e os recursos necessários para apoiar os processos do negócio.

Já (KILOV; SIMMONDS, 1998) são mais enfáticos, para eles uma regra de negócio deve ser objetiva e definida em termos de notações matemáticas de proposições, mostrando que precisão é essencial quando formula-se as regras de negócio. Uma proposição é um fato ou estado observável de negócio envolvendo uma ou mais entidades pelo qual é possível afirmar ou negar a veracidade dessas entidades.

Desta forma as regras de Negocio da empresa de Alimentos Duas Rodas seriam:

- Os administradores, líderes de manutenção e líderes de setor poderão criar ordens de manutenção;
- Apenas o administrador poderá finalizar ordens de manutenção;
- Os cadastros só serão possíveis na versão web;
- Cada ordem de manutenção deve conter uma assinatura do manutentor, uma do operador ou líder de setor e uma do administrador;
- Cada ordem de manutenção só pode ser finalizada após ter as 3 assinaturas;
- Somente administradores terão acesso a tela de pendência de assinaturas;
- Cada manutentor só pode ter uma ordem iniciada por vez;
- Manutentores podem pausar ordens de manutenção;
- Manutentores podem ter várias ordens pausadas ao mesmo tempo;
- Pode haver mais de um manutentor por ordem de manutenção;
- Uma ordem deve ter um manutentor principal;

## 4 PESQUISA DE ANTERIORIDADE

Durante a análise de requisitos, foram procurados outros sistemas que fazem esse processo de gerenciamento de ordem de manutenção para nos ajudar e entender quais são os problemas mais comuns desse ramo e quais as principais funcionalidades que o mercado oferece.

### 4.1 Produttivo

*Produttivo* é um sistema que tem vários módulos e diferentes aplicações. Um deles é o de Ordem de Serviço de Manutenção. Utilizando o Produttivo, um supervisor planeja e acompanha as atividades de manutenção utilizando o sistema web. Os manutentores, através de um smartphone ou tablet, alimenta dados ao programa de acordo com os defeitos identificados, bem como a solução abordada. Na finalização da ordem de serviço de manutenção, o responsável realiza a assinatura digital. (PRODUTTIVO, 2019)

### 4.2 SoftByte

*Visual Machine*, da empresa SoftByte tem um fluxo bem mais simples quando comparado à Produttivo. Porém, possui algumas ferramentas interessantes, como o agendamento de manutenções preventivas e preditivas e possui um sistema de controle de investimentos em manutenção em cada um dos equipamentos. (SOFTBYTE, 2019)

### 4.3 ProdWin

*ProdWin Pw-1* em seu módulo de manutenção prove controle do tempo gasto e dos materiais utilizados nas manutenções. Também dispõe de agendamento das manutenções, podendo relacioná-las a um técnico e ser acompanhadas em tempo real. O seu sistema de manutenção preditiva adiciona automaticamente a máquina na tela de alerta de manutenções, quando aproxima-se de seu limite operacional, conforme as informações previamente cadastradas. (PRODWIN, 2019)

### 4.4 Diferenciais Agil.It

Analizando os softwares encontrados, será implementado para o Agil.It o diferencial de ter integração entre sistemas, uma central de notificação e ser modelado a partir das necessidades da empresa Duas Rodas.

## 5 REQUISITOS DO SISTEMA

Os requisitos são capacidades que devem ser atendidas ou possuídas por um sistema para resolver um problema ou atingir um objetivo. O conjunto de todos os requisitos que formam a base para o desenvolvimento subsequente de um software (VAZQUEZ; SIMÕES, 2016). Neles são definidos como serão os comportamentos do sistema e seus fluxos. Eles foram abordados em dois segmentos: Os Requisitos funcionais que são fluxos do sistema e os requisitos não funcionais que são necessários para a utilização do sistema.

### 5.1 Requisitos Funcionais

Conforme (REZENDE, 2005), requisitos funcionais são as necessidades descritas pelo cliente, onde a equipe do projeto analisa e especifica as funções, desempenho, interfaces e restrições, conforme as fases das metodologias aplicadas. Sua finalidade é distinguir as dependências do sistema para que o mesmo funcione de acordo com os requisitos informados pelo cliente.

[Add mais referencias](#)

#### 5.1.1 Cadastros

- Usuários;
- Setores;
- Máquinas;
- Unidades de medida;
- Peças;
- Tipos de estoques;
- Estoque de peças;
- Ordem de manutenção.

#### 5.1.2 Consultas

- Máquinas;
- Estoque de peças;
- Ordem de manutenção;
- Login;
- Logoff.

### 5.1.3 Funcionalidades

- Monitor de ordens em aberto;
- Solicitação de abertura de ordem de manutenção;
- Central de notificação.

## 5.2 Requisitos Não Funcionais

(TORONTO, 2005) relata que o propósito dos requisitos não funcionais é descrever as qualidades requeridas para um sistema, como sua usabilidade e seu desempenho. Em sistemas alguns destes requisitos podem determinar tecnologias ou algoritmos específicos a serem utilizados, garantindo compatibilidade com sistemas existentes (MARTINS, 2007)

[Add mais uma referencia de RNF](#)

### 5.2.1 Funcionalidades

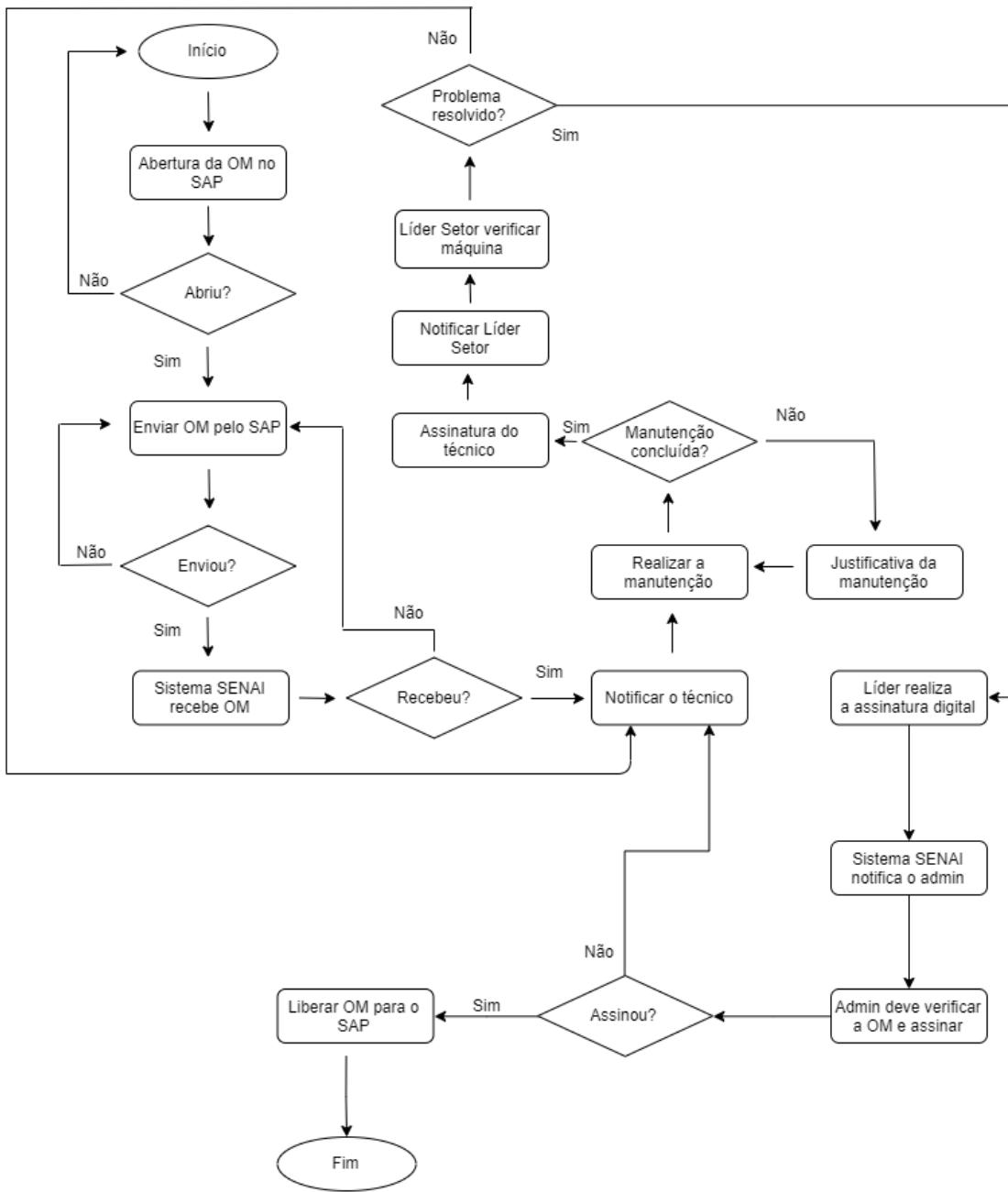
- Sistema desenvolvido para web;
- Utilizar o banco MS Sql Server;
- Usuários devem ter acesso à computadores e/ou dispositivos móveis.

## 5.3 Fluxograma do Sistema Desenvolvido

Segundo (GASQUES, 2002) fluxograma é uma representação gráfica das tarefas de um determinado processo, sendo de forma sequencial a execução delas. (HURT, 2014) relata que um fluxograma fornece um panorama de alto nível sobre um sistema de informação.

[Add mais referencias](#)

Figura 5.1: Fluxo do AGIL.IT



Fonte: os autores (2020)

Na Figura 5.1 é possível verificar todo o fluxo do sistema onde começa e termina com a integração do SAP. O sistema irá receber a OM do SAP e notificar o técnico responsável pela execução do serviço. Após o técnico realizar a manutenção, os usuários (técnico, e líder do setor) realizam as assinaturas e o usuário administrador analisa e libera a OM para a integração.

## 5.4 Diagrama de Caso de Uso do Sistema

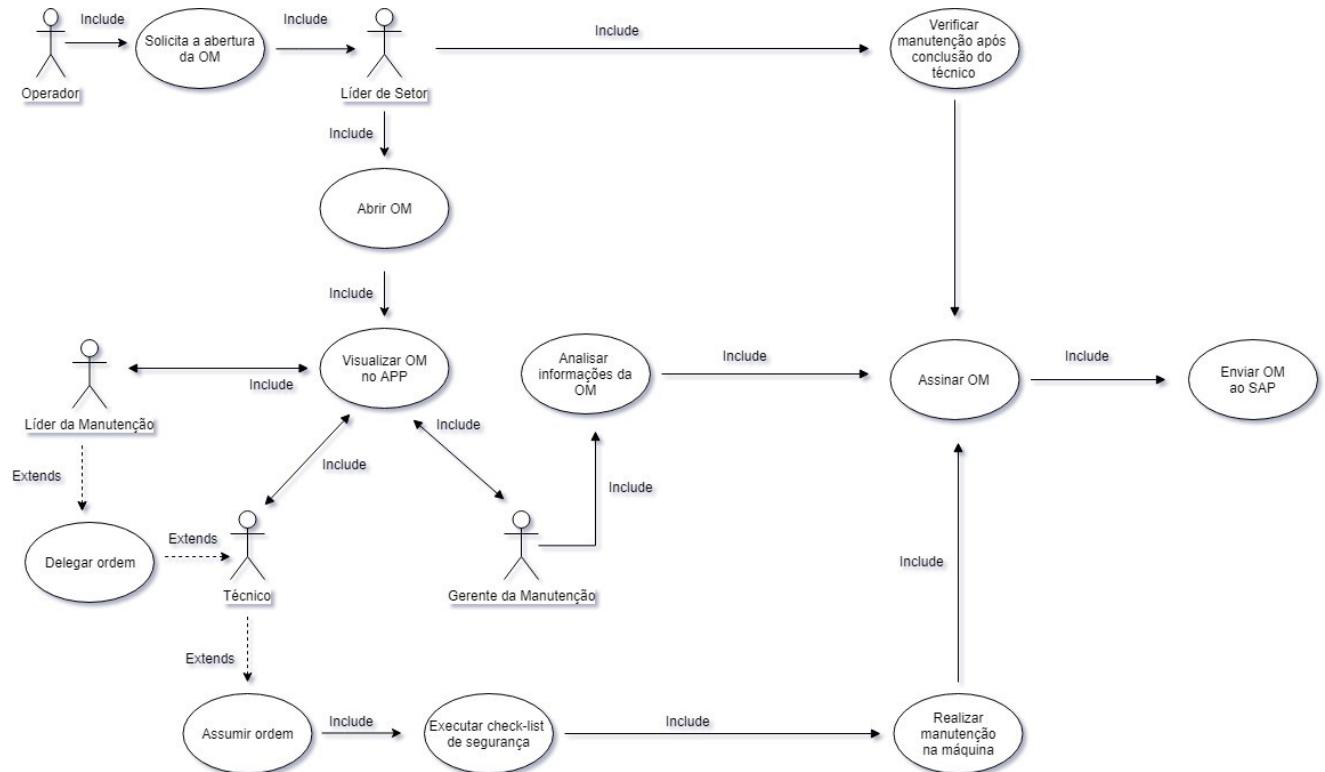
Diagramas dos casos de uso são técnicas utilizadas para captar os requisitos funcionais de um sistema, descrevem as interações entre usuários de um sistema e o próprio sistema,

fornecendo uma narrativa sobre como ele é utilizado (FOWLER, 2005).

Para (CARNIELLO, 2003) seu objetivo principal consiste em definir o comportamento de um sistema, sem revelar sua estrutura interna.

**Colocar mais uma referencia com definição de aplicação**

Figura 5.2: Caso de Uso do AGIL.IT



Fonte: os autores (2020)

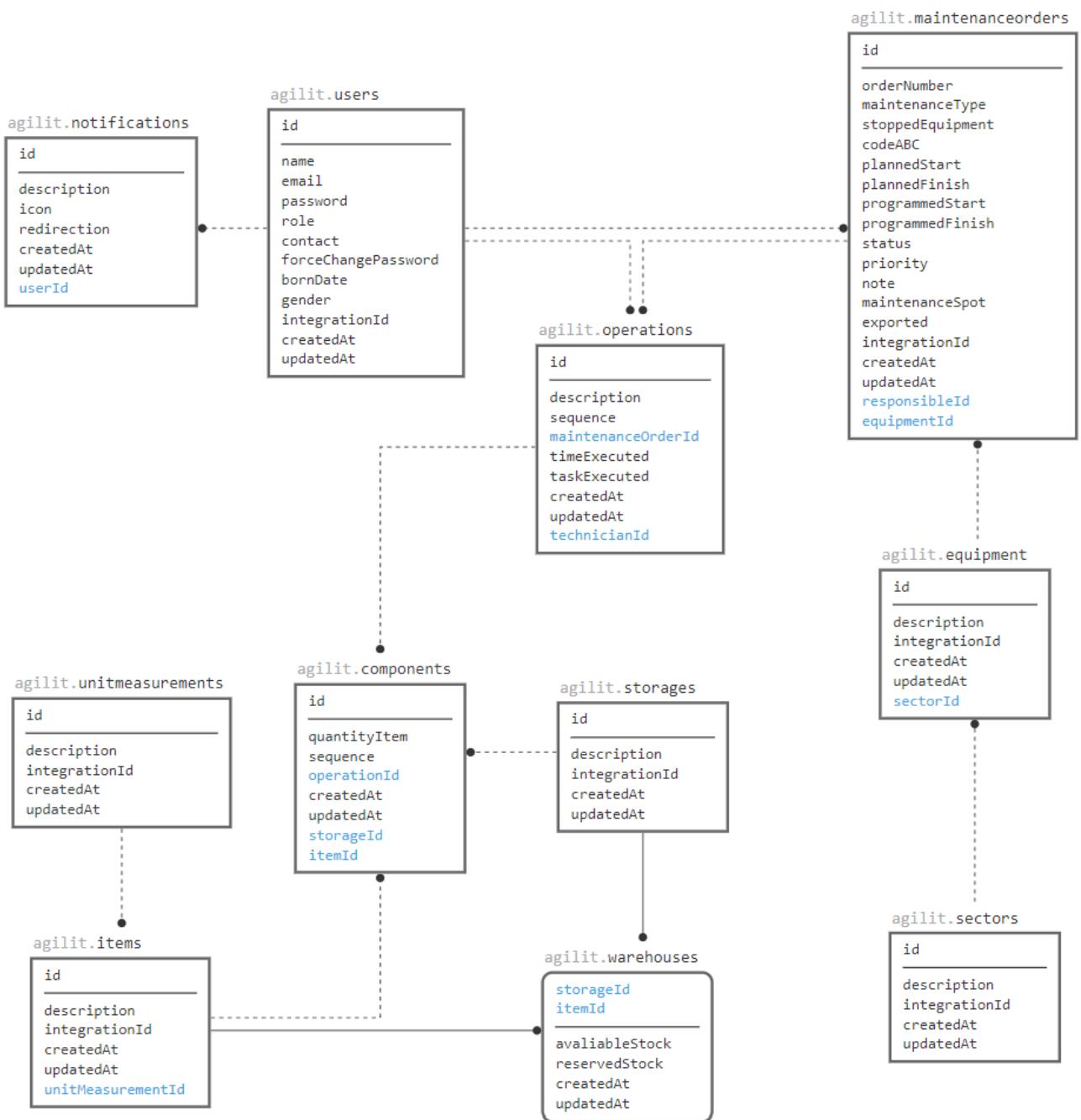
Na Figura 5.2 é possível verificar quais ações cada ator desempenhará no sistema. Os operadores de máquinas e os líderes de manutenção verificam a manutenção feita e assinam a OM. O técnico realizará a manutenção e verificará as peças necessárias para realizar a assistência ao equipamento e se as mesmas possuem estoque disponível. E o administrador realizará cadastros e encerará as ordens de manutenção.

## 5.5 Entidade de Relacionamento do Banco de Dados

Planejar todas as etapas, dedicando a atenção especialmente ao projeto de estruturação do banco de dados. Com a estrutura pronta, a facilidade para dar manutenção no sistema fica muito mais fácil. Utilizar a entidade de relacionamento, tem como objetivo, obter uma descrição de forma abstrata dos dados que serão armazenados no banco de dados (ROCHA; TERRA, 2010).

**Add mais referencias aqui também.**

Figura 5.3: Entidade de Relacionamento



Fonte: os autores (2020)

A Figura 5.3 mostra toda a entidade de relacionamento do banco de dados, onde cada tabela representa uma estrutura de dados no banco de dados e as ligações entre elas demonstram relacionamentos que essas tabelas possuem entre si.

## 6 PLANEJAMENTO DO SISTEMA

O planejamento do sistema é crucial para o sucesso do projeto. Nele deve-se estipular cronogramas de desenvolvimento e fazer a análise de risco para calcular a probabilidade e impacto de problemas que possam ocorrer durante o desenvolvimento do projeto. Um bom planejamento também deve ter uma análise de projeto que envolva recursos de diversas naturezas bem como humanos, capitais, técnicos e tecnológicos. Outro ponto importante é a prototipação do sistema para ter uma avaliação inicial com os *stakeholders* e ter uma base a ser seguida durante o desenvolvimento.

### 6.1 Cronograma do Projeto

Todo bom projeto deve ter um cronograma e um planejamento de ação, baseado nisso, foi elaborado dois cronogramas: um apresentando todo o conteúdo aprendido no semestre e outro montando plano de ação para implementação do projeto.

As figuras a seguir, são cronogramas que demonstram a trajetória do desenvolvimento do projeto. Divididos por semestre, os cronogramas listam as atividades aprendidas nas UCs, com todas as partes teóricas e práticas onde o objetivo principal é implementar todo conhecimento adquirido em prol de um projeto único: A unificação de todas as UCs do curso de Sistemas para Internet.

Figura 6.1: Cronograma de Aprendizagem: Terceiro Semestre

	02/2019	03/2019	04/2019	05/2019	06/2019	07/2019
Análise de Requisitos de Sistema						
Desenvolvimento de Canvas						
Apresentação do Projeto Duas Rodas						
Desenvolvimento Mobile						
Desenvolvimento de Casos de Uso						
Desenvolvimento de Modelo de Classes						
Desenvolvimento de APIs						
Desenvolvimento WEB						
Mapeamento de Regras de Negócio						
Mapeamento de Riscos do Projeto						
Desenvolvimento de Protótipos						

Figura 6.2: Cronograma de Aprendizagem: Quarto Semestre

	07/2019	08/2019	09/2019	10/2019	11/2019	12/2019
Análise de Requisitos de Sistema						
Desenvolvimento de Canvas						
Apresentação do Projeto Duas Rodas						
Desenvolvimento Mobile						
Desenvolvimento de Casos de Uso						
Desenvolvimento de Modelo de Classes						
Desenvolvimento de APIs						
Desenvolvimento WEB						
Mapeamento de Regras de Negócio						
Mapeamento de Riscos do Projeto						
Desenvolvimento de Protótipos						

Figura 6.3: Cronograma de Aprendizagem: Quinto Semestre

	02/2020	03/2020	04/2020	05/2020	06/2020	07/2020
Características e Aplicações do Software						
Análise de Processos						
Diagrama dos Fluxos de Dados						
Validação de Requisitos						
Estruturação do Sistemas						
Arquitetura de Objetos Distribuídos						
Padronização de Desenvolvimento						
Objetos e Classes de Objetos						
Desenvolvimento Baseado em Componente						
Apresentação das Informações						
5S						
Teste de Usuário						
Teste para Detecção de Defeitos						
Medições e Métricas das Funcionalidades do Software						
Técnicas de Recuperação do Software em Quedas						

Figura 6.4: Cronograma de Aprendizagem: Sexto Semestre

	07/2020	08/2020	09/2020	10/2020	11/2020
Desenvolvimento de Regras de Negócio					
Desenvolvimento de Melhorias de Usabilidade					
Desenvolvimento de Melhorias na Segurança do Sistema					
Desenvolvimento de Integração					
Testes de Interface					
Desenvolvimento do TCC					

## 6.2 Análise de riscos

A análise de riscos tem como objetivo identificar os possíveis problemas durante e após o desenvolvimento do projeto a fim de elaborar um plano de ação para solucionar rapidamente o problema de fato (SCHMITZ; ALENCAR, 2012).

Segundo (FILHO, 2003), um grande volume de dados publicados aponta para os riscos que correm os projetos de software executados sem a utilização de processos adequados. Um

levantamento publicado, a partir de uma base de dados de 4.000 projetos, constatou a ocorrência frequente dos seguintes problemas:

- 70% dos projetos de grandes aplicativos sofrem de instabilidade dos requisitos. Os requisitos crescem tipicamente cerca de 1% ao mês, atingindo níveis de mais de 25% de inchaço ao final do projeto.
- Pelo menos 50% dos projetos são executados com níveis de produtividade abaixo do normal.
- Pelo menos 25% do software de prateleira e 50% dos produtos feitos por encomenda apresentam níveis de defeitos superiores ao razoável.
- Produtos feitos sob pressão de prazos podem quadruplicar o número de defeitos.
- Pelo menos 50% dos grandes projetos de software estouraram seu orçamento e seu prazo.

Sendo assim, foi identificado os fatores de risco, no qual o projeto em questão possa estar exposto. Nela, faz-se uma análise do impacto e probabilidade de fatores prejudiciais ao projeto, conforme tabela 6.1 abaixo.

Tabela 6.1: Tabela de Riscos Agil.it

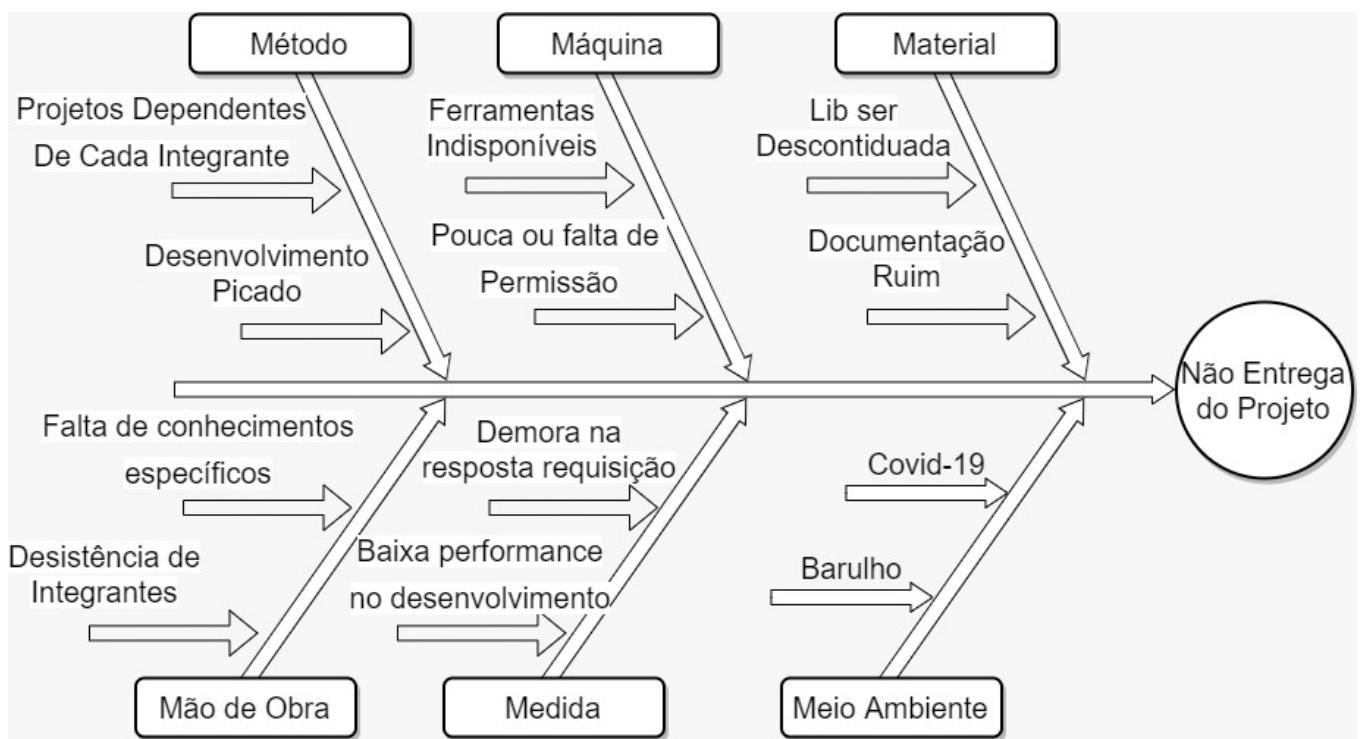
Riscos	Probabilidade	Impacto
Mudança de escopo	90%	2
Entrega no prazo	70%	3
Integração com SAP	70%	2
Implantação na empresa	60%	2
Coneção com o banco de dados	60%	3
Aceitação da usabilidade do sistema	50%	2
Usuários inexperientes	40%	2
Mudanças na tecnologia	20%	3
Segurança dos dados	15%	2
Coneção com a rede	10%	2
Falta de profissionais	5%	3

Na tabela 6.1 estão mapeados os principais riscos identificados para o projeto Agil.it. Nela, a probabilidade indica a chance do risco ocorrer, as cores acompanham a porcentagem da mesma, sendo 70% ou mais a cor vermelha, entre 40% e 69% a cor amarela e de 0% a 39% a cor verde e o impacto é uma escala de um a três (1-3) do quanto o risco pode afetar a conclusão e entrega do projeto. A seguir será abordado o diagrama de causa e efeito.

### 6.2.1 Diagrama de Causa e Efeito

[Descrever Diagrama de Causa e Efeito com referencias](#)

Figura 6.5: Diagrama de Causa e Efeito



Fonte: os autores (2020)

### 6.3 PMBOK

É uma metodologia de gerenciamento de projeto internacionalmente reconhecida, essas práticas podem auxiliar na resolução dos recursos humanos, capitais, tecnológicos e técnicos. Além disso, utilizando PMBOK é possível gerir melhor o andamento do projeto e de forma mais coordenada. Segundo (VARGAS, 2018) o PMBOK tem conhecimentos já comprovados que são amplamente utilizados, assim como o conhecimento de práticas mais inovadoras e avançadas. Dentre as técnicas de planejamento, pode-se utilizar para estudos e desenvolvimento a EAP - Estrutura Analítica de Projetos, descrita a seguir.

#### 6.3.1 Estrutura Analítica do Projeto

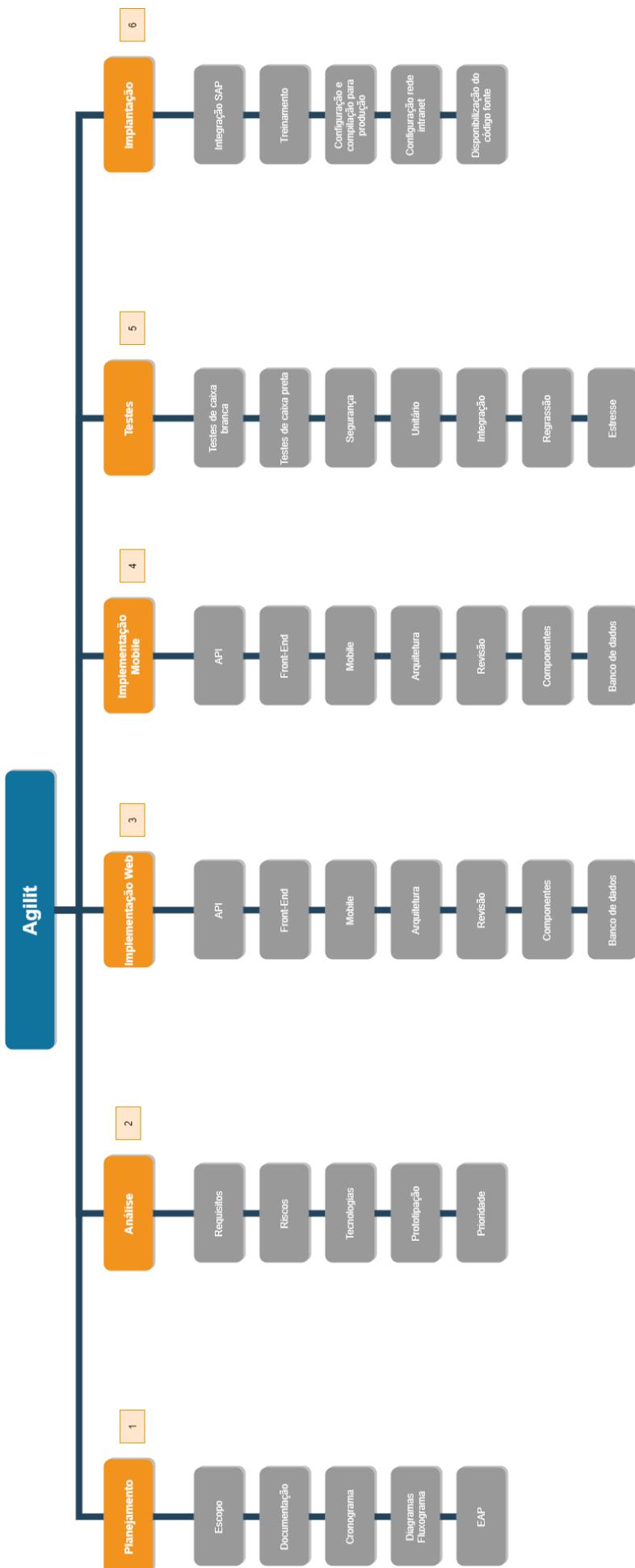
A Estrutura Analítica do Projeto (EAP) é a divisão estruturada de trabalho do projeto dividido em faixas gerenciadas cuja a sua totalidade significa em um entregável ao projeto final.

Segundo (INSTITUTE, 2018) o detalhamento da EAP deve chegar até o nível do pacote de trabalho, nível mais baixo na EAP, que é o ponto no qual o custo e o cronograma do trabalho podem ser estimados de forma confiável. Porém o nível de detalhamento desse pacote varia de acordo com a complexidade de cada projeto. (KERZNER, 2017) defende que a EAP deve ser composta por até três níveis pois se for detalhado com demasia o custo com o gerenciamento serão também excessivos.

Para o projeto Agil-it foi decidido utilizar 6 faixas de entregáveis:

- Planejamento;
- Análise;
- Implementação Web;
- Implementação Mobile;
- Testes;
- Implantação.

Figura 6.6: EAP AGIL.IT



Fonte: os autores (2020)

A figura 6.6 mostra a estrutura e o desenvolvimento que cada faixa requer no projeto. Ela não segue uma ordem cronológica, portanto não é necessário finalizar uma faixa para começar outra, muitas vezes elas são desenvolvidas em conjunto para uma melhor utilização do tempo de projeto.

### 6.3.2 Fluxo de Processos

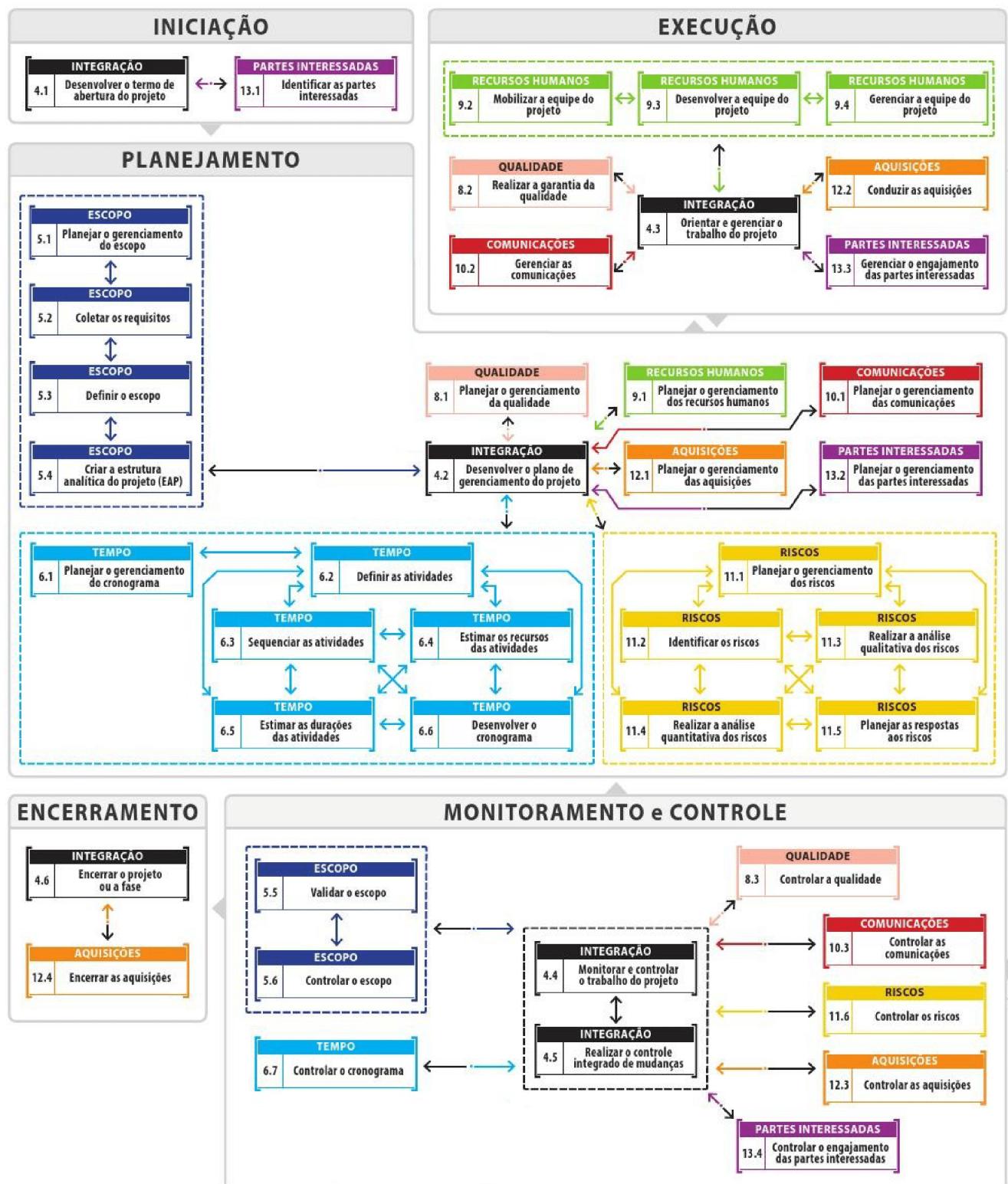
O fluxo de processo é subdividido em cinco fases principais sobre o fluxo do projeto, iniciação sendo a primeira seguindo de planejamento, execução, monitoramento, controle e finalização. Para (VARGAS, 2018) um projeto é desenvolvido a partir de uma idéia, em seguida parte para um plano e após isso é executado e concluído.

Para a definição dos processos, foi utilizado os seguintes recursos:

- Partes Interessadas;
- Escopo;
- Tempo;
- Recursos Humanos;
- Aquisições;
- Qualidade;
- Comunicações;
- Riscos.

A figura abaixo contém todos os fluxos desenhados para o projeto.

Figura 6.7: Fluxo e Processos AGIL.IT



Fonte: os autores (2020)

**Iniciação** - Esta é a fase inicial do projeto, nela é determinado a necessidade do projeto, com seus objetivos e justificativas, então é realizado os documentos iniciais onde as melhores

estratégias são identificadas e selecionadas.

**Planejamento** - Fase onde deve ser detalhado minuciosamente tudo aquilo que será realizado no projeto, desde cronogramas, interdependências entre atividades, alocação dos recursos envolvidos, análise de custos etc. Esta parte é muito importante pois a execução do projeto será em cima destas atividades para que sejam executadas sem dificuldades e imprevistos.

**Execução** - Nesta fase tudo é onde tudo que foi planejado anteriormente se torne realidade, tendo que ser executado e realizado conforme planejado, qualquer erro cometidas nas fases anteriores fica evidente durante essa fase.

**Monitoramento e Controle** - Esta fase acontece paralelamente às demais fases do projeto, acompanhando e controlando aquilo que está sendo realizado no projeto como um todo, podendo propor ações corretivas e preventivas no menor espaço de tempo possível após a detecção de erros.

## 6.4 Protótipo

A prototipação é uma etapa de suma importância no desenvolvimento de projeto de software. Além de melhorar a produtividade da equipe, ela facilita o entendimento dos requisitos do sistema e permite a apresentação de conceitos e funcionalidades da aplicação de modo simplificado. Nesse trabalho foi utilizado a prototipação visual cujo ênfase se aplica a estética e usabilidade. Nesse tipo de protótipo é possível identificar o layout e a identidade visual da aplicação. (DEXTRA, 2019)

Protótipos podem ser gerados de acordo com as seguintes categorias (COYETTE, 2004): protótipos em baixa fidelidade que focam na interação, em componentes de interface e na estrutura geral do sistema; protótipos em alta fidelidade que produzem uma imagem real do sistema; protótipos executáveis que produzem o código em uma linguagem de programação, focando em navegação, mas sem ainda levar em consideração as regras de negócio. Cada categoria serve para um propósito específico: protótipos em baixa fidelidade são úteis para demonstrar aos usuários quais atividades o sistema atende e as possibilidades de navegação no sistema, assim como para proporcionar uma visão geral do sistema. Protótipos em alta fidelidade são úteis para demonstrar padrões e guias de estilo. Protótipos executáveis são úteis para demonstrar navegação e testar o uso da interface (ROSEMBERG, 2008). Seguindo as definições, o projeto desenvolvido utiliza os protótipos de baixa fidelidade

## 6.5 Aplicação WEB

A aplicação web tem como foco o gerenciamento de toda a aplicação envolvendo consultas e cadastros gerais do sistema. Apesar desse foco acentuado à gestão, é possível desempenhar todos os papéis dentro da aplicação web.

### 6.5.1 Cadastro de Usuário

Figura 6.8: Cadastro de Usuários

Cadastro de Usuário

Código do usuário

Nome do usuário

Perfil usuário

Senha

Email

Contato

Data Nascimento

Gênero

Masculino

Feminino

Cancelar

Salvar

Fonte: os autores (2020)

A tela 6.8 é utilizada para cadastrar usuários no sistema. Caso queira atualizar um registro basta colocar o código dele no primeiro campo ou pesquisar no ícone de lupa. Para cadastrar um novo basta deixar o primeiro campo vazio.

## 6.5.2 Monitor de Ordem de Manutenção: Cards

Figura 6.9: Monitor de Ordem de Manutenção: Cards



Lucas Gonçalves  
Administrador

Crachá: 005415871997  
Função: Líder de Manutenção

---

🔧 Ordem Manutenção

📝 Cadastros

📄 Relatórios

⚙️ Configurações

✖️ Sair | 🔔 Notificações

### Monitor de Ordens de Manutenção

Status
Ordens Abertas

Data
de
01/01/2019

Data
até
01/02/2019

Prioridade
TODOS

Listar
Limpar

Exibir em Lista >

**OM - 2445492/DJ0449**

**Tipo:** Preventiva

**Equipamento:** DHA03005/007

**Prioridade:** Baixa

**Abertura:** 06/07/2019 ás 14h27

**OM - 2445492/DJ0449**

**Tipo:** Corretiva

**Equipamento:** DHA03005/007

**Prioridade:** Alta

**Abertura:** 06/07/2019 ás 14h27

**OM - 2445492/DJ0449**

**Tipo:** Rota

**Equipamento:** DHA03005/007

**Prioridade:** Média

**Abertura:** 06/07/2019 ás 14h27

**OM - 2445492/DJ0449**

**Tipo:** Preventiva

**Equipamento:** DHA03005/007

**Prioridade:** Média

**Abertura:** 06/07/2019 ás 14h27

**OM - 2445492/DJ0449**

**Tipo:** Preventiva

**Equipamento:** DHA03005/007

**Prioridade:** Urgente

**Abertura:** 06/07/2019 ás 14h27

**OM - 2445492/DJ0449**

**Tipo:** Corretiva

**Equipamento:** DHA03005/007

**Prioridade:** Urgente

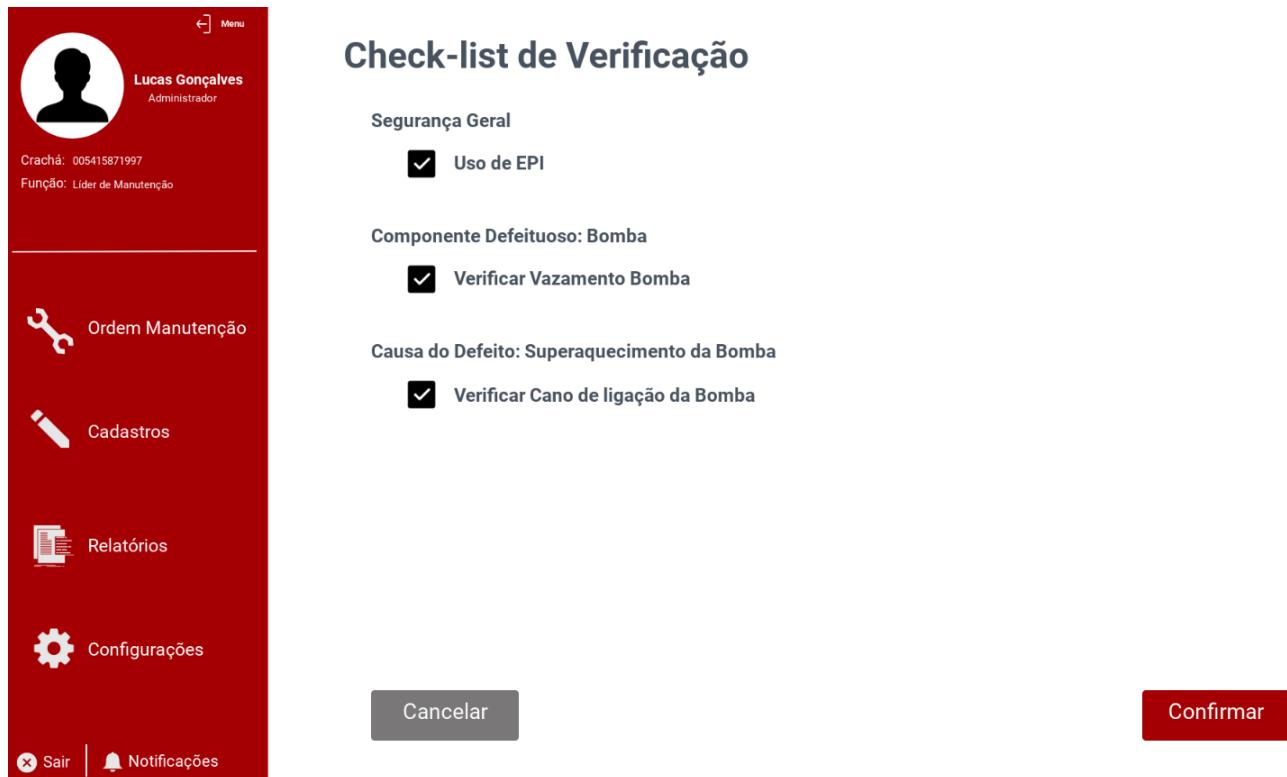
**Abertura:** 06/07/2019 ás 14h27

Fonte: os autores (2020)

A tela 6.9 permite a consulta rápida e dinâmica das ordens de manutenção. Nela você pode aplicar os filtros de acordo com as necessidades e será listada em forma de cartões, eles darão acesso à uma tela de detalhamento de ordem de manutenção.

### 6.5.3 Checklist de Segurança

Figura 6.10: Checklist de Segurança



Fonte: os autores (2020)

Na tela 6.10 o manutentor irá marcar a lista de segurança antes de iniciar a ordem de manutenção. Essa listá será gerada dinamicamente de acordo com a parametrização de segurança cadastrado no sistema.

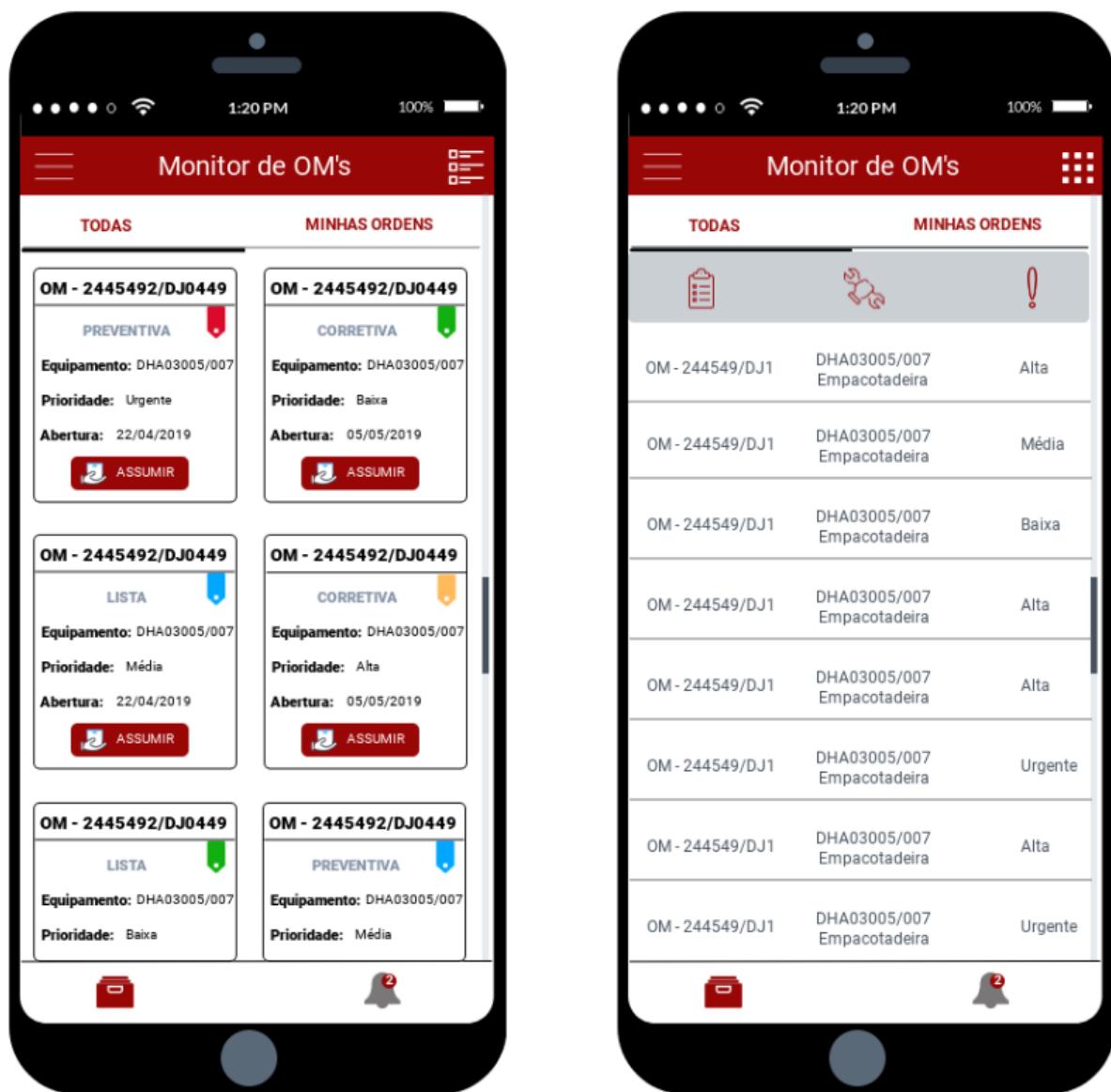
## 6.6 Aplicação Mobile

A aplicação Mobile é um ponto estratégico do produto, pois sua mobilidade permite com que os técnicos possam atuar na manutenção e realizar anotações e apontamentos no sistema através de um smartphone ou tablet.

### 6.6.1 Monitor

Na figura 6.11 é possível verificar o monitor do manutentor. Nesse monitor, o manutentor consegue rapidamente visualizar as OM's pendentes e seus respectivos status através das bandeiras indicadas no card. As vermelhas indicam que a OM tem uma prioridade emergente, as amarelas têm prioridade alta, as azuis têm prioridade média e as verdes possuem uma prioridade baixa.

Figura 6.11: Monitor



Fonte: os autores (2020)

## 6.6.2 Central de Notificações

Figura 6.12: Notificações

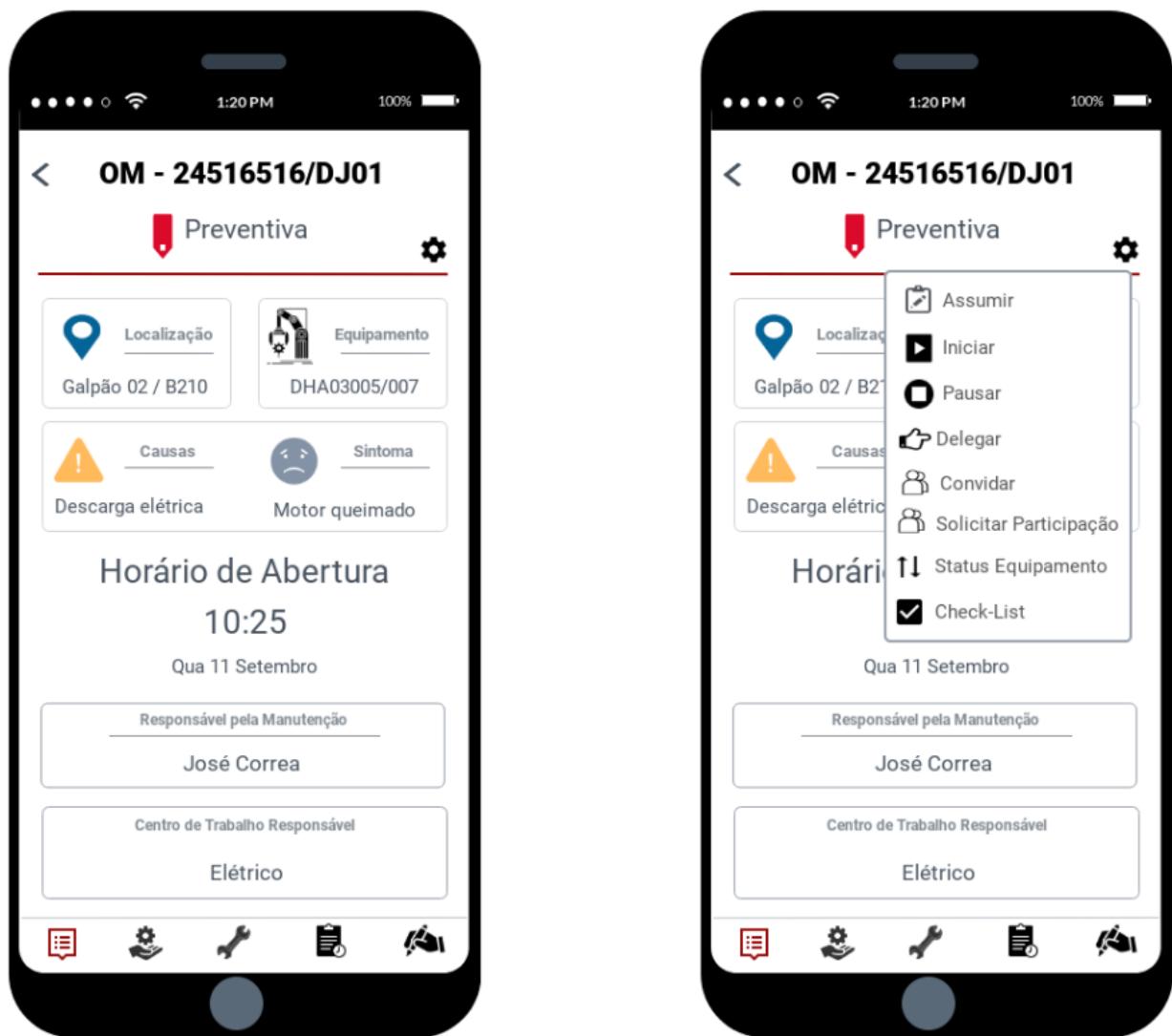


Fonte: os autores (2020)

Na tela 6.12 é possível verificar notificações do usuário autenticado no sistema.

### 6.6.3 Ordem de Manutenção

Figura 6.13: Ordem de Manutenção



Fonte: os autores (2020)

Na tela 6.13 será possível acompanhar o andamento de uma ordem de manutenção preventiva e corretiva, ver informações referente à ordem e executar ações nela, como alterar status, adicionar operações e realizar assinaturas.

## 7 IMPLEMENTAÇÃO

**PROFESSORA, POR FAVOR VERIFICAR COMO PODEMOS ESCREVER OS NOMES DE REGRAS E NOME DE CLASSES, SE ELES VÃO FICAR ENTRE ASPAS OU COMO DEVEMOS FAZER**

Para o desenvolvimento do projeto, houveram uma série de reuniões onde o objetivo era delimitar e estabelecer alguns requisitos para a implementação do projeto, que de suma relevância para o andamento e funcionamento dos setores a serem implantados.

Para o desenvolvimento foi estabelecido uma média de 30 dias de implementação direta no desenvolvimento do projeto, considerando que a data prevista para a entrega final do projeto será em 2020/2.

### 7.1 Tecnologias

Para o desenvolvimento do sistemas, foram divididas em três aplicações: uma visando a parte WEB, outra visando dispositivos móveis com os sistemas Android e IOS e para efetuar a comunicação entre os sistemas e o banco dados, foi elaborado uma aplicação de servidor cujo valida as requisições e efetua a comunicação entre as aplicações visíveis e o banco de dados.

#### 7.1.1 Aplicação WEB

Para o desenvolvimento foi utilizada as tecnologias pré requisitadas pela empresa Duas Rodas, sendo elas HTML5, SCSS e como complemento, utilizamos a biblioteca ReactJS.

**HTML5** - É uma linguagem de marcação utilizada para desenvolvimento Web, esta nova versão traz consigo importantes mudanças quanto ao papel do HTML no mundo da Web, através de novas funcionalidades como semântica e acessibilidade.

**SCSS** - É uma folha de estilo interpretada e compilada em Cascading style sheets, ao ser interpretado é criado blocos de códigos de regras CSS. Resumidamente é utilizada para fazer o Design do sistema, com cores personalizadas, etc.

**ReactJs** - O React é uma biblioteca JavaScript de código aberto com foco em criar interfaces de usuário em páginas web.

#### 7.1.2 Aplicação Mobile

Utilizando as tecnologias Ionic 4 e TypeScript esta aplicação é desenvolvida paralelamente com as demais aplicações, tem como objetivo a mobilidade, facilidade de acesso e interação podendo ser utilizado em qualquer lugar dentro da empresa.

**TypeScript** - TypeScript é um superconjunto de JavaScript desenvolvido pela Microsoft que adiciona tipagem e alguns outros recursos a linguagem. É utilizada somente pelos desen-

volvedores, pois auxilia na construção do código fonte. Seu código é transpilado para JavaScript, no final das contas tudo escrito em TypeScript vira JavaScript.

**Ionic 4** - O Ionic é um framework open source para desenvolvimento de aplicativos móveis multiplataforma.

### 7.1.3 Aplicação do Servidor

A aplicação do servidor é o canal centralizador das demais plataformas. Ela tem acesso único e exclusivo ao banco de dados. É responsável por todas as validações e executa regras de negócio. O servidor foi desenvolvido em Typescript e utiliza a TypeOrm para integração com o banco de dados, para a API utiliza o Express.

**TypeOrm** - TypeOrm é uma ORM para Typescript, utiliza injeção de dependências para modelar as classes e transformar em tabelas.

**Express** - Express é uma biblioteca que abstrai a camada HTTP e gerencia requisições recebidas pelo protocolo.

## 7.2 Códigos Desenvolvidos

Nesta etapa será apresentado alguns trechos dos códigos implementados para ilustrar o funcionamento e o entendimento de partes específicas do código fonte, onde os mesmos contém regras complexas e de suma importância ao leitor.

### 7.2.1 WEB

#### 7.2.1.1 Componentes

A plataforma web da Agil.It tem como padrão componentizar os elementos exibidos no front-end, para isso, utiliza-se a biblioteca React-md, com componentes acessíveis por propriedades (parâmetros) e totalmente personalizáveis. Os estilos podem ser configurados tanto em tempo de compilação quanto em tempo de execução pelas variáveis SCSS configuráveis e pelo uso de CSS. A biblioteca deve ser instalada no projeto e importada no código fonte conforme é mostrado a seguir.

---

#### Listing 7.1 Importando o componente de Ícone do react-md

---

```
1 import React, { PureComponent } from 'react';
2 import { FontIcon } from 'react-md';

4 export class C_Icon extends React.Component {
5   constructor(props) {
6     super(props);
7   }

9   render() {
10     return (
11       <FontIcon
```

```

12      className={this.props.className}
13      primary={this.props.primary}
14      forceFontSize={!!this.props.iconSize}
15      secondary={this.props.secondary}
16      forceSize={this.props.iconSize}
17      label={this.props.label}
18      style={this.props.style}
19      onClick={this.props.action}
20      disabled={this.props.disabled}
21      name={this.props.name}
22    >
23      {this.props.icon}
24    </FontIcon>
25  );
26 }
27 }
```

---

### 7.2.1.2 Cookies

Ao efetuar o login na plataforma, as informações do usuário, exceto sua senha, são identificadas através da biblioteca ‘universal-cookie’, essas informações são manipuladas para ocultar telas de determinados usuários de acordo com o seu perfil ou exibir alguma tela adicional de controle e análise, caso o perfil logado seja de um administrador. Outro exemplo seria impedir/permitir que tal usuário possa executar determinada ação ocultando elementos na tela de acordo com seu perfil. Com isso, o controle de informações e ações que o sistema oferece se torna muito mais fácil.

Após cada requisição feita ao servidor, o mesmo retorna um novo token renovado e então atualiza o token e usuário do cookie.

---

#### **Listing 7.2** Gravar dados no cookie

```

1 setToken(token) {
2   const user = JWTHelper.decomposeJwt(token)
3
4   this.cookies.set('token', token, { path: '/' })
5   this.cookies.set('user', user, { path: '/' })
6 }
```

---

Para recuperar os valores do cookies basta instanciar os cookies e utilizar o método get.

---

#### **Listing 7.3** Recuperar dados do cookie

```

1 class App extends Component {
2   constructor() {
3     super()
4
5     this.cookies = new Cookies();
6
7     this.state = {
8       token: this.cookies.get('token'),
9       user: this.cookies.get('user'),
```

```

10      };
11  };
12 }

```

---

Para remover um dados, basta utilizar o método remove.

#### **Listing 7.4 Recuperar dados do cookie**

```

1 onLogout() {
2   this.cookies.remove('token');
3   this.setState({ token: undefined })
4 }

```

---

#### 7.2.1.3 Helpers

Com o objetivo de agilizar o desenvolvimento do projeto, foram implementadas diversas classes de “Ajuda” como “DateHelper.js”, “MaintenanceOrderHelpers.js”, “searchModel.js”.

DateHelper é responsável por manipular e retornar as datas no formato desejado, possui funções que podem ser chamadas em qualquer classe para auxiliar o time de desenvolvimento no ganho de tempo. Como trabalhar com data é trabalhoso, segue um exemplo de como deixar um pouco mais simples.

#### **Listing 7.5 Função que formata a data em dia/mês/ano - horas/minutos**

```

1 static formatDateTime(inputDate) {
2   var date = this.getDate(inputDate);
3
4   var day = StringHelper.JustifyLeft(date.getDate(), 2, 0);
5   var month = StringHelper.JustifyLeft(date.getMonth() + 1, 2, 0);
6   var year = date.getFullYear();
7   var hour = date.getHours();
8   var minutes = date.getMinutes();
9
10  return `${day}/${month}/${year} às ${hour}h${minutes}`;
11 }

```

---

Executando a função em um componente.

#### **Listing 7.6 Utilizando a função de formatação de data hora**

```

1 <div style={{ margin: 10, position: "absolute", right: 0 }}>
2   <C_ToolTip
3     position="left"
4     tooltip={
5       <div>
6         <div>Aberto em:</div>
7         <div> {DateHelper.formatDateTime(order.openedDate)} </div>
8       </div>
9     }
10    >
11    <C_Icon

```

```

12     style={{cursor:"pointer", fontSize: 30, color:"#3177e8"}}
13     icon="access_time"
14   />
15 </C_ToolTip>
16 </div>

```

---

A “MaintenanceOrderHelper” é responsável por auxiliar na manipulação dos dados referentes as ordens de manutenção, possui funções que fazem a tradução de algumas propriedades dos objetos que são salvos em inglês no servidor.

Utilizando a função de tradução do status da Ordem na classe de Ordem de Manutenção.

#### **Listing 7.7 Utilizando a tradução de termos da ordem de manutenção**

```

1 <span style={{ fontSize: 18, marginLeft: 5 }}>{HelperOM.translate("status"
, order.orderStatus)}</span>

```

---

Função que traduz o status da Ordem.

#### **Listing 7.8 Função de tradução**

```

1 static translate(prop, value) {
2   let props;
4
4   if (prop == "priority") props = this.getPriority();
5   else if (prop == "status") props = this.getStatus();
6   else if (prop == "color") props = this.getColorPriority();
7   else if (prop == "layout") props = this.getLayoutType();
8   else return value;
10
10  return props[value];
11 }
12
13 static getStatus() {
14   return {
15     created: "Aberta",
16     assumed: "Assumida",
17     started: "Iniciada",
18     paused: "Pausada",
19     stopped: "Parada",
20     canceled: "Cancelada",
21     "signature-pending": "Assinatura Pendente",
22     signatured: "Assinada",
23     finished: "Finalizada",
24     "no_status": "Sem Status",
25   };
26 }

```

---

Ainda no “MaintenanceOrderHelper” existem funções que ordenam as ordens por prioridade e data de abertura na tela de monitoramento.

Utilizando a função na classe Dashboard ao dar um get da lista de ordens.

---

**Listing 7.9** Ordenando pedidos após a consulta no servidor
 

---

```

1 const response = await this.provider.getList(sendData);
2 let orderList = [];

4 if (response.success) {
5   orderList = response.data;
6 }

8 const sortedOrderList = HelperOM.sortOrders(orderList);
9 this.setState({ orders: sortedOrderList , showLoading: false });

```

---

Função que irá fazer a ordenação da lista de ordens de manutenção. A ordenação avalia 2 campos: prioridade e data de abertura. As ordens serão ordenadas das urgentes às com baixa prioridade e como um segundo fator será a data de abertura, sendo as mais antigas mostradas primeiro.

---

**Listing 7.10** Funções responsáveis pela ordenação
 

---

```

1 static sortOrders(list) {
2   var ordenatedPriority = this.ordenatedPriority();

4   return list.sort((a, b) => {
5     if (ordenatedPriority[a.priority] > ordenatedPriority[b.priority])
6       return -1;
7     else if (ordenatedPriority[a.priority] < ordenatedPriority[b.priority]
8       ]) return 1;

9     var dateTimeA = DateHelper.getDate(a.openedDate).getTime();
10    var dateTimeB = DateHelper.getDate(b.openedDate).getTime();

11    if (dateTimeA > dateTimeB) return 1;
12    else if (dateTimeA == dateTimeB) return 0;
13    else return -1;
14  });
15 }

17 static ordenatedPriority() {
18   return {
19     urgent: 3,
20     high: 2,
21     medium: 1,
22     low: 0,
23   };
24 }

```

---

O “SearchModel.js” contém funções pré-configuradas das colunas que serão exibidas nas telas de crud afim de uma melhor visualização e organização dos dados exibidos. A configuração é composta por um *array* de objetos que possui três propriedades: name, property e defaultValue. A propriedade name será o cabeçalho da coluna do GRID, property é a propriedade da listagem que será renderizado na coluna e defaultValue é o valor que será apresentado caso não haja dados para a property na listagem.

---

**Listing 7.11 Definição da configuração para consulta de dados**

```
1 const machineTypeColumns = () => [
2   {
3     name: "ID",
4     property: "id",
5     defaultValue: "Sem ID",
6   },
7   {
8     name: "Descrição",
9     property: "description",
10    defaultValue: "Sem Descrição",
11  },
12];

14 const equipmentColumns = () => [
15   {
16     name: "Código",
17     property: "code",
18     defaultValue: "Sem Código",
19   },
20   {
21     name: "Descrição",
22     property: "description",
23     defaultValue: "Sem Descrição",
24   },
25   {
26     name: "Tipo Máquina",
27     property: "machineType.description",
28     defaultValue: "Sem Tipo de Máquina",
29   },
30];
```

---

Utilizando as funções na classe Machine, cadastro de Equipamentos.

---

**Listing 7.12 Tela para cadastro de equipamento**

```
1 class CreateMachine extends Component {

3   constructor(props) {
4     super(props);

6     this.state = {
7       visible: true,
8       autocomplete: '',
9       machineType: '',
10      fields: {},
11      list: [],
12      machineTypeList: [],
13      equipmentColumns: equipmentColumns(),
14      machineTypeColumns: machineTypeColumns(),
15    };
16  };
17}
```

---

Ao obter as colunas em um State, os dados são enviados ao nosso componente de “C\_AutoComplete” que possui um ícone de lupa onde o usuário pode clicar para visualizar todos os equipamentos cadastrados.

---

#### **Listing 7.13 Componente C\_AutoComplete**

```

1  <C_AutoComplete
2    id="id"
3    name="id"
4    value={this.state.autocomplete}
5    label={"Equipamento"}
6    rightIcon={"search"}
7    list={this.state.list}
8    onChange={this.onChange}
9    searchColumns={this.state.equipmentColumns}
10   />

```

---

Função dentro do componente “C\_AutoComplete” que exibe a tabela de consulta de acordo com as colunas e lista de dados enviados por parâmetro.

---

#### **Listing 7.14 Componente C\_AutoComplete**

```

1  onClickIcon() {
2    if (!Array.isArray(this.props.searchColumns)) return;
3
4    confirmAlert({
5      customUI: ({ onClose }) => (
6        <C_SearchTable
7          columns={this.props.searchColumns}
8          content={this.props.list}
9          onClick={this.tableSelected}
10         extraFunction={onClose}
11       />
12     )
13   });
14 }

```

---

## 7.2.2 Mobile

### 7.2.2.1 Ações da Ordem de Manutenção

A estrutura da classe "AgilitActionUtils.ts" é feita com o intuito de reutilizar códigos, sendo que cada tipo de OM deve ser assinada da mesma forma que qualquer outra, mudando apenas seu atributo identificador, a seguir seguem algumas características importantes desta classe.

A primeira característica desta classe é de forma centralizada realizar as ações de uma determinada OM, recebendo como parâmetro para qual OM deverá ser feita a ação e a ação propriamente dita.

---

#### **Listing 7.15 Alterar situação da ordem de manutenção**

```

1 public async changeStatus(orderID, orderStatus : AgilitOrderStatus){
2   return this.restOrder.orderActions(orderID, orderStatus);
3 }
4
5 public orderActions(orderID : number, agilitOrderStatus :
6   AgilitOrderStatus){
7   const orderStatus : any = {
8     orderStatus: agilitOrderStatus
9   }
10
11   if (agilitOrderStatus == AgilitOrderStatus.ASSUMED){
12     let userInfo : any = AgilitStorageUtils.getDataJSON(
13       AgilitStorageTypes.USERDATA);
14
15     orderStatus.userId = userInfo.id;
16   }
17
18   this.http.url = this.http.getBaseUrl() + this.restAction + '/' + orderID
19     + '/status';
20   return ProviderHelper.put(this.http, orderStatus);
21 }
22

```

Outra característica muito importante desta classe é realizar as assinaturas de uma determinada OM, sendo passada como parâmetro a OM respectivamente e a senha de assinatura. Ao realizar uma assinatura a classe irá realizar algumas validações da OM a fim de verificar se está de acordo para enviar ao servidor, com isso a promessa deve ser resolvida ou rejeitada.

### **Listing 7.16 Assinar a ordem de manutenção**

```

1 public async signOrder(order, assignSignaturePassword): Promise<any>{
2   return new Promise(async (resolve, reject) => {
3     this.resolvePromise = resolve;
4     this.rejectPromise = reject;
5
6     try {
7       if (!this.validateOrder(order, assignSignaturePassword)){
8         return;
9     }
10
11     const user = AgilitStorageUtils.getDataJSON(
12       AgilitStorageTypes.USERDATA);
13
14     if (AgilitUtils.isNullOrUndefined(user) ||
15       AgilitUtils.isNullOrUndefined(user.id) || user.id == ''){
16       this.rejectPromise('Dados de usuário inválido!');
17       return;
18     }
19
20     await this.restOrder.orderAssignment(order.id, user.id).then(
21       (response: any) => {
22         if (AgilitUtils.isNullOrUndefined(response)){
23           return;
24         }
25
26         this.resolvePromise(response);
27       }
28     );
29   }
30 }
31

```

```

25      }
26      ).catch(
27      error => {
28          this.rejectPromise(error);
29      }
30  );
31  } catch (error) {
32      this.rejectPromise(error);
33  }
34 });
35 }

```

---

Para que uma OM seja assinada com sucesso a classe deve conter as validações dos estados atuais da OM a ser assinada, por exemplo, se o usuário realizar a assinatura de uma OM com status cancelada esta implementação tem de validar a mesma.

### **Listing 7.17 Validar estados da OM**

```

1 private validateOrder(order, password) : boolean{
2     if (password != AgilitStorageUtils.getData(AgilitStorageTypes.PASSWORD))
3     {
4         this.rejectPromise('Senha incorreta!');
5     }
6
7     if (order.orderStatus == AgilitOrderStatus.SIGNATURED){
8         this.rejectPromise('OM já está assinada!');
9         return;
10    }
11
12    if (order.orderStatus == AgilitOrderStatus.FINISHED){
13        this.rejectPromise('OM já está assinada!');
14        return;
15    }
16
17    if (order.orderStatus == AgilitOrderStatus.CANCELED){
18        this.rejectPromise('OM está cancelada!');
19        return;
20    }
21
22    return true;
23 }

```

---

#### **7.2.2.2 Dados Armazenados no Local Storage**

Algumas informações por exemplo como login, token de acesso devem ficar salvas em memória para posteriormente serem utilizadas novamente, com isso desenvolvemos a classe chamada "AgilitStorageUtils.ts" contendo regras, limpezas específicas dos dados e separação dos tipos dos dados que estão sendo armazenados e buscados.

### **Listing 7.18 Tipos de dados armazenados**

```
1 export enum AgilitStorageTypes{
2     USERDATA = 'user',
3     USERNAME = 'username',
4     PASSWORD = 'password',
5     TOKEN    = 'token'
6 }
```

Método responsável por fazer a limpeza de um dado específico da memória do navegador, conforme seu tipo.

#### **Listing 7.19** Limpeza específico de cada dado

```
1 public static clearSpecificData(agilitStorageType : AgilitStorageTypes){
2     window.localStorage.removeItem(agilitStorageType);
3 }
```

Buscando dados de um tipo específico armazenado na memória do navegador.

#### **Listing 7.20** Buscar em formato de STRING

```
1 public static getData(agilitStorageType : AgilitStorageTypes){
2     return window.localStorage.getItem(agilitStorageType);
3 }
```

Buscando dados de um tipo específico armazenado e convertendo para objeto.

#### **Listing 7.21** Buscar em formato JSON

```
1 public static getDataJSON(agilitStorageType : AgilitStorageTypes){
2     return JSON.parse(window.localStorage.getItem(agilitStorageType));
3 }
```

Este método faz uma verificação do tipo do dado que está sendo inserido na memória do navegador, o método aceita entradas de string e objeto, pois o mesmo normaliza para string automaticamente.

#### **Listing 7.22** Inserindo dados na memória do navegador

```
1 public static setData(agilitStorageType : AgilitStorageTypes, data :
2     Object|Array<any>|string){
3     if (AgilitUtils.isNullOrUndefined(data)){
4         return;
5     }
6     if (typeof data === 'object'){
7         window.localStorage.setItem(agilitStorageType, JSON.stringify(data));
8         return;
9     }
10    if (typeof data === 'string'){
11        window.localStorage.setItem(agilitStorageType, data);
12    }
13 }
```

```
14 }
```

---

### 7.2.3 Servidor

#### 7.2.3.1 Mapeamento do Banco de Dados com TypeOrm

A TypeOrm trabalha com injecção de dependências, então em cada propriedade tem um *decorator* que vai definir o mapeamento dela. Para gerar uma *primary key* com auto increment, basta usar a injecção "PrimaryGeneratedColumn", para uma coluna simples, apenas "column", para data de criação do registro, "CreateDateColumn" e para data de atualização do registro "UpdateDateColumn". Com a TypeOrm é possível ter classes abstratas para abranger propriedades em comum no banco de dados, para tanto foi criado a classe baseClass que irá conter todos os campos em comum de todas as tabelas do banco de dados.

---

#### Listing 7.23 BaseClass: Mapeamento de propriedades de tabelas do banco de dados

```
1 export abstract class BaseClass {  
  
3   @PrimaryGeneratedColumn()  
4   public id: number | undefined = undefined;  
  
6   @Column({ default: '' })  
7   public integrationID: string = '';  
  
9   @CreateDateColumn()  
10  public createdAt: Date | undefined = undefined;  
  
12  @Column()  
13  public createdBy: number | undefined = undefined;  
  
15  @UpdateDateColumn()  
16  public updatedAt: Date | undefined = undefined;  
  
18  @Column()  
19  public updatedBy: number | undefined = undefined;  
  
21  @Column({  
22    type: Boolean,  
23    default: false  
24  })  
25  public deleted: boolean = false;  
  
27  constructor() {  
28  }  
  
30 }
```

---

Para facilitar as tabelas genéricas de CRUD, onde têm apenas os dados base e o campo de descrição, foi criado a classe abstrata crudClass.

---

#### Listing 7.24 CrudClass: Mapeamento de propriedades para cruds genéricos

---

```

1 export abstract class CrudClass extends BaseClass {
2
3   @Column({nullable: false})
4   @IsNotEmpty({
5     message: 'Descrição: Campo obrigatório.'
6   })
7   public description: string = '';
8
9   constructor() {
10    super();
11  }
12}

```

---

E por fim, a aplicação de uma classe que irá gerar de fato uma tabela no banco de dados.

### **Listing 7.25** Mapeamento de propriedades da tabela sector

```

1 @Entity('sector')
2 export class Sector extends CrudClass {
3
4   constructor() {
5    super();
6  }
7
8 }

```

---

No primeiro parâmetro da injeção "Entity"definimos o nome da tabela, e como estendemos a classe do CrudClass que estende, subsequentemente da classe BaseClass a tabela terá todas as colunas definidas nas injeções de dependências.

Para fazer relações entre as tabelas existem alguns *decorators* específicos: "ManyToOne"para relações muito para um, "ManyToMany"para relações muitos para muitos, "OneToOne"para relações um para um e "OneToMany"para relações um para muitos.

### **Listing 7.26** Mapeamento de propriedades da tabela área de installation\_area

```

1 @Entity("installation_area")
2 export class InstallationArea extends CrudClass {
3
4   @ManyToOne(type => Sector, sector => sector.id, { nullable: false })
5   @JoinColumn()
6   public sector: Sector = undefined;
7
8   constructor() {
9    super();
10  }
11
12 }

```

---

A TypeOrm tenta fazer a relação automaticamente pelo nome dos campos, porém, se for uma relação mais complexa é possível usar o *decorator* "JoinColumn"para definir qual coluna se

relaciona com qual coluna e explicitar em qual das tabelas deve ficar a *foreign key*.

---

### **Listing 7.27** Mapeamento de propriedades da tabela maintenance\_worker

```

1 @Entity('maintenance_worker')
2 export class MaintenanceWorker extends BaseClass {

4   @ManyToOne(type => User, user => user.id)
5   @JoinColumn()
6   public user : User;

8   @ManyToOne(type => MaintenanceOrder, maintenanceOrder =>
9     maintenanceOrder.id, { cascade: false })
10  @JoinColumn()
11  public maintenanceOrder : MaintenanceOrder | undefined = undefined;

12  @Column()
13  public isMain: boolean = false;

15  @Column()
16  public isActive: boolean = false;

18  @OneToMany(type => WorkerRequest, workerRequest =>
19    workerRequest.maintenanceWorker, { cascade: false })
20  public workerRequest: Array<WorkerRequest>;
21
22  @OneToMany(type => WorkedTime, workedTime =>
23    workedTime.maintenanceWorker, { cascade: false })
24  public workedTime: Array<WorkedTime>;
25 }

```

---

#### 7.2.3.2 Validação de Objetos com o Class Validator

Para validações de campos da tabela utilizamos outra biblioteca com injeção de dependências que funciona muito bem quando mesclada com a TypeOrm, o Class Validator. Com ele podemos utilizar, por exemplo o *decorator* “*IsNotEmpty*” na propriedade “*description*”. Esse *decorator* irá validar se a propriedade está vazia ou não, caso esteja, irá retornar o erro “Descrição: Campo obrigatório.” conforme definição no próprio *decorator*.

---

### **Listing 7.28** Definição de validação de propriedades com o Class Validator

```

1 @Column({ nullable: false })
2 @IsNotEmpty({
3   message: 'Descrição: Campo Obrigatório.',
4 })
5 public description: string = '';

```

---

E para executar a validação conforme as injeções feitas, é preciso importar a função “*validate*” do class validator e executar passando uma instância do objeto.

---

### **Listing 7.29** Validação de Objetos com o Class Validator

```

1  async validate(entity: Entity) : Promise<any> {
2    const errors = await validate(entity);
3
4    if (errors.length === 0) {
5      return undefined
6    }
7
8    let errorList = []
9
10   errors.forEach(error => {
11     let constraints = error.constraints
12
13     for (const key in constraints) {
14       if (constraints.hasOwnProperty(key)) {
15         errorList.push(constraints[key])
16       }
17     }
18   });
19
20   return errorList
21 }

```

O retorno vem em forma de array e traz diversas informações a respeito dos erros ocorridos, no caso da API, é mostrado apenas a mensagem de erro em si, então a listagem é filtrada para mostrar apenas os erros.

### 7.2.3.3 TypeOrm: Query as Function

A TypeOrm traz algumas facilidades na interação com o banco de dados, uma delas é a possibilidade de montar query através de funções.

A função “all” faz a busca de lista dos dados de uma tabela específica no banco de dados.

---

#### **Listing 7.30** TypeOrm: Obter lista de registro

```

1  async all(request: Request, response: Response, next: NextFunction) {
2
3    const where = {
4      deleted: false,
5      ...this.getWhereConditions(request.params, request.query,
6        this.getRepositoryEntity().create()),
7      ...this.getCustomWheresList(),
8    }
9
10   const { skip, take } = request.headers;
11
12   return this.getRepositoryEntity().find(<any>{
13     relations: this.includes(),
14     where,
15     skip,
16     take,
17   });

```

Já a função “one” busca somente os dados de um filtro específico.

---

**Listing 7.31** TypeOrm: Obter um registro

```
1 async one(request: Request, response: Response, next: NextFunction) {  
2   return this.get(request.params.id);  
3 }
```

---

### 7.2.3.4 API

Na API é utilizado o Express para gerenciar as requisições feitas pelo front, mobile e integração de sistemas externos. Foi adicionado os *plugins* Cors para lidar com o CORS, BodyParser para converter JSON e o Helmet para segurança da aplicação.

No script de inicialização do servidor é estabelecido conexão com o banco de dados, instanciado a aplicação Express e instalado seus *plugins*,

---

**Listing 7.32** Script de inicialização do servidor

```
1 createConnection().then(async connection => {  
2  
3   // create express app  
4   const app = express();  
5   app.use(bodyParser.json());  
6   app.use(cors({exposedHeaders: 'token'}));  
7   app.use(helmet());  
8  
9   // Get all system routes  
10  let routes = new Routes();  
11  
12  routes.getCollections().forEach((collection: Collection) => {  
13    collection.getRoutes().forEach((route: Route) => {  
14      // Register each route  
15      route.registerRoute(app)  
16    });  
17  })  
18  
19  
20  app.listen(4000);  
21  
22 }).catch(error => console.log(error));
```

---

O roteamento é composto por várias coleções de rotas, no qual cada coleção tem um conjunto de rotas que é composto pela url que a rota responderá, os métodos que aceitará e a função que será invocada. Por fim, o método “registerRoute” irá registrar a rota da coleção ao express. Esses métodos que as rotas respondem ficam no package controller que fazem a ponta entre receber a requisição, processar os dados recebidos e responder o que foi solicitado, seja cadastrar, atualizar, excluir ou consultar um ou mais registros. Da mesma forma que foi feita com os models, foi desenvolvido um “CrudController” que é uma classe que utiliza *Generics* que para cada instancia recebe o valor da entidade do Crud e irá se adequar conforme o tipo fornecido.

Essa classe também recebe no construtor o acesso ao repositório que será gravado, no caso, a tabela do banco de dados.

---

**Listing 7.33** CrudController

```
1 export class CrudController<Entity> {  
2  
3     private repositoryEntity : Repository<Entity>;  
4  
5     constructor(repositoryEntity: Repository<Entity>) {  
6         this.repositoryEntity = repositoryEntity;  
7     }  
8 }
```

---

Todas as rotas são protegidas, com exceção da rota de login. Essa proteção é feita através de um *middleware* colocado ao registrar a rota no express, esse *middleware* vai capturar os cabeçalhos “token” para autenticação de sessão e “authorization” para usuários de integração, caso nenhum dos cabeçalhos seja fornecido a api irá retornar o status “401 - Unauthorized”. Caso seja passado o cabeçalho “authorization” o *middleware* irá verificar se o token é válido e se está na base de dados de usuários de integração. Se passado “token”, o sistema irá decompor e validar o *JWT* utilizando a biblioteca jsonwebtoken, Caso a validação do *JWT* informado falhe, seja por não estar com a assinatura válida da aplicação ou por ter expirado o tempo da utilização, retorna um erro para a API informando que o token não é válido. Caso a validação passe, o token é regerado com expiração renovada para 5 horas e é adicionado ao header da resposta. Lembrando que, o token pode ser obtido através da rota de Login.

---

**Listing 7.34** Função para validação de autenticação da API

```
1 export const checkJwt = (request: Request, response: Response, next: NextFunction) => {  
2  
3     const authorization = <string>request.headers["authorization"];  
4  
5     if (authorization) { // The Authorization was passed in so now we  
6         validate it  
7         this.checkIntegration(authorization)  
8             .then(valid => {  
9                 if (valid === true) {  
10                     return next();  
11                 } else {  
12                     return response.status(401).send();  
13                 }  
14             }).catch(err => {  
15                 return response.status(500).send({  
16                     "success": false,  
17                     "error": err.message  
18                 });  
19             })  
20     } else {  
21         //Get the jwt token from the request's header  
22         const token = <string>request.headers["token"];  
23         let jwtPayload;
```

```
24     //Try to validate the token and get data
25     try {
26         jwtPayload = <any>jwt.verify(token, JWT.jwtSecret);
27         response.locals.jwtPayload = jwtPayload;
28     } catch (error) {
29         //If token is not valid, respond with unauthorized
30         return response.status(401).send();
31     }
32
33     //set to response's header the new token
34     response.append('token', new UserController().generateJwtToken(<User>
35                     jwtPayload));
36
37     next();
38 }
39 };
```

### 7.2.3.5 Estratégia de Exclusão de Dados

A estratégia para exclusão de dados adotado no projeto é não deletar os registros mas alterar a coluna “deleted” para true. Desta maneira é possível rapidamente desfazer uma exclusão errada e também não ocorrerá problemas com histórico caso as entidades sejam deletadas.

---

#### Listing 7.35 Método para exclusão de registro

---

```
1 async removeEntity(entity: Entity) {
2     entity["deleted"] = true;
3
4     return this.getRepositoryEntity().save(entity)
5 }
```

---

## 8 TESTES DE USABILIDADE

### DEFINIR USABILIDADE

Uma das definições mais encontradas na literatura é a de que a usabilidade é parte de um projeto mais amplo, que aponta para o desenvolvimento de métodos e técnicas que podem incorporar considerações de ergonomia dentro do processo de design e avaliação da interface homem/computador (BASTIEN; SCAPIN, 1993).

Finalizar com uma Referência essa introdução.

Para que serve a usabilidade? 3 - Referências

Fechar com a importância da aplicação dela no projeto Integrador.

### 8.0.1 Elaboração dos Testes Aplicados

A usabilidade é uma qualidade de uso, ou seja, ela é definida ou medida para um determinado contexto no qual um sistema é operado. Assim, um sistema pode proporcionar boa usabilidade para um usuário experiente, mas péssima para um iniciante, ou vice-versa; ou ainda, pode ser fácil operar se o sistema for usado esporadicamente, mas difícil se for utilizado frequentemente (CYBIS, 2003).

Para isso, estudos como os de (BASTIEN; SCAPIN, 1993), apresentam regras e recomendações para que os sistemas computacionais sejam elaborados de modo a facilitar sua aprendizagem e uso, proporcionando usabilidade.

Para a avaliação do projeto, foi elaborado uma metodologia de verificação de alguns requisitos visando a análise e a usabilidade padrão mínimo para a implantação e utilização do web/mobile. Para isso foi criado algumas situações de usabilidade segundo alguns trabalhos científicos como a de (SILVA, 2016), onde verifica-se alguns critérios importantes que foram formulados para a aplicação das avaliações do desenvolvimento dos projetos Integradores. As avaliações aconteceram de forma remota com uma amostragem de aproximadamente 19 usuários, incluindo os colaboradores da empresa Duas Rodas.

Para a elaboração dos resultados, foram realizadas as médias de todas as avaliações e unificadas graficamente, buscando descrever-las o mais transparente possível. Para isso, foi criado 3 módulos de avaliação que são eles: Critério Geral de Apresentação, Critérios Gerais e Erros comuns, onde respectivamente encontra-se como características de layout, características de desenvolvimento e erros que podem ocorrer no projeto.

### 8.0.2 Discussão dos Resultados

Verificou-se no Gráfico 8.1, de Critérios Gerias de apresentação do projeto Agil-it que o percentual de avaliação, um aproveitamento de 84,5% e média de todos os resultados avaliados

de 8,5, considerando os critérios de apresentação dos Layouts, onde as notas foram de 1-10 avaliando cada quesito, como observa-se no gráfico.

Figura 8.1: Critério Geral de Apresentação



Para a análise dos Critérios Gerais, observa-se no gráfico 8.2 que o foco principal foi no desenvolvimento e desempenho do software considerando características fundamentais para o sucesso da usabilidade e da aprovação do Usuário. Neste quesito, a avaliação ficou em um aproveitamento de 88,2% com a média de 8,8 dos quesitos avaliados.

Figura 8.2: Critérios Gerais



Quanto a avaliação dos Erros Comuns, foi avaliado com aplica-se ou não este quesito, e observa-se na tabela XXI que, o software de gerenciamento do Agil.It, chegou a um percentual médio de aprovação de 92%, sendo assim, considera-se ótimo o percentual obtido na avaliação, na etapa do projeto.

<b>Erros comuns de Usabilidade</b>	
1	Ícones e Menus ambíguos
2	Linguagens que permitem apenas movimentos direcionados de forma única
3	Limite de entrada e manipulação
4	Limite de seleção e destaque
5	Sequência não clara de passos
6	Mais passos para gerenciar a interface do que para realizar tarefas
7	Links complexos entre/com aplicações
8	Confirmações e Feedbacks inadequados
9	Pouca inteligência e antecipação por conta do sistema
10	Mensagem de erro, tutoriais ajuda e documentação inadequadas
11	Poluição Visual
12	Má Organização da Informação
13	Componentes Incompreensíveis
14	Distrações irritantes
15	Navegação Ineficiente
16	Sobrecarga de Informações

## 9 CONSIDERAÇÕES E TRABALHOS FUTUROS

Com a análise de requisitos executada no semestre anterior, conseguimos dar continuidade com o desenvolvimento das aplicações com base neles propostos. Também será melhorada a parte documental e adicionado funcionalidades específicas necessárias para a entrega final do projeto. Desta maneira, será de suma importância a participação direta dos envolvidos para o desenvolvimento competente e satisfatório para a execução e implantação do Projeto Integrador.

## REFERÊNCIAS BIBLIOGRÁFICAS

- BASTIEN, J. C.; SCAPIN, D. L. Ergonomic criteria for the evaluation of human-computer interfaces. 1993.
- BOFF, L. **Sustentabilidade: o que é - o que não é**. Editora Vozes, 2017. ISBN 9788532656100. Disponível em: <<https://books.google.com.br/books?id=px46DwAAQBAJ>>.
- CARNIELLO, A. **Teste Basead na estrutura de casos de uso**. [S.I.: s.n.], 2003.
- COUTO, J. M. C. **Técnicas de visualização de dados em gerenciamento de projetos de desenvolvimento de software: proposta de extensão do PMBOK**. 108 p. Monografia (Pós Graduação) — Pontifícia Universidade Católica do Rio Grande do Sul, Rio Grande do Sul, 2018.
- COYETTE, A. et al. Sketchxml: towards a multi-agent design tool for sketching user interfaces based on usixml. In: **Proceedings of the 3rd annual conference on Task models and diagrams**. [S.I.: s.n.], 2004. p. 75–82.
- CRERIE, R.; BAIÃO, F. A.; SANTORO, F. M. Identificacao de regras de negocio utilizando mineração de processos. In: ACM. **Companion Proceedings of the XIV Brazilian Symposium on Multimedia and the Web**. [S.I.], 2008. p. 241–246.
- CYBIS, W. d. A. Engenharia de usabilidade: uma abordagem ergonômica. **Florianópolis: Labiutil**, 2003.
- DEXTRA. **Prototipação e sua importância no desenvolvimento de software**. 2019. Disponível em: <<https://dextra.com.br/pt/prototipacao-e-sua-importancia-no-desenvolvimento-de-software/>>.
- FILHO, W. de P. P. **Engenharia de software**. [S.I.]: LTC, 2003. v. 2.
- FOWLER, M. **UML Essencial**. [S.I.: s.n.], 2005. 104–105 p.
- GASPARINI, B. C. et al. Driv-uml: visualização de não-conformidades arquiteturais em uml no contexto da modernização dirigida a arquitetura. Universidade Federal de São Carlos, 2018.
- GASQUES, J. **O desafio do dízimo**. [S.I.: s.n.], 2002.
- HAUFE, M. I. **Estimativa da produtividade no desenvolvimento de software**. 108 p. Monografia (Mestrado) — Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, 2001.
- HURT, R. L. **Sistemas de informações contábeis**. [S.I.]: AMGH Editora Ltda, 2014.
- INSTITUTE, P. **A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide-Sixth Edition / Agile Practice Guide Bundle (BRAZILIAN PORTUGUESE)**. Project Management Institute, 2018. ISBN 9781628255270. Disponível em: <<https://books.google.com.br/books?id=fOlfdwAAQBAJ>>.
- KERZNER, H. **Project Management: A Systems Approach to Planning, Scheduling, and Controlling**. Wiley, 2017. ISBN 9781119165378. Disponível em: <<https://books.google.com.br/books?id=VNExDgAAQBAJ>>.
- KILOV, H.; SIMMONDS, I. Business rules: from business specification to design. In: BOSCH, J.; MITCHELL, S. (Ed.). **Object-Oriented Technologies**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 188–194. ISBN 978-3-540-69687-2.

- MARTINS, J. C. C. **Técnicas para gerenciamento de projetos de software.** [S.I.]: Brasport Livros e Multimidia Ltda, 2007.
- MATTIOLI, A. **Sistemas computacionais.** [S.I.]: Editora Pearson, 2020.
- MOURA, T. C. e. E. F. R. Condicionantes de sucesso em projetos de software e sua influência nos resultados. **Revista Gestão e Tecnologia**, v. 18, n. 1, p. 61–87, 2018.
- MOURA, T. M. **Análise da Implementação de Práticas de TI Verde em um Instituto Federal de Educação, Ciência e Tecnologia.** 85 p. Monografia (Mestrado) — Universidade Federal de Pernambuco, Pernambuco, 2017.
- PÁDUA, S. I. D. de. Investigação do processo de desenvolvimento de software a partir da modelagem organizacional, enfatizando regras do negócio. 2001.
- PRADO, D. **Gerência de Projetos em Tecnologia da Informação.** [S.I.]: Editora INDG, 1999. ISBN 9788586948176.
- PRIMO, M. O. **Diagrama de Classe de Projeto.** 2014.
- PRODUTTIVO. 2019. Disponível em: <<https://produttivo.com.br>>.
- PRODWIN. 2019. Disponível em: <<https://prodwin.com.br>>.
- QUEIRÓS, R. C. C.; MÉXAS, M. P.; DRUMOND, G. M. Tecnologia da informação verde nas organizações: uma visão estratégica. **Sistemas e Gestão**, v. 15, n. 2, p. 103–112, ago. 2020. Disponível em: <<https://revistasg.emnuvens.com.br/sg/article/view/1629>>.
- REZENDE, D. A. **Engenharia de Software e Sistemas de Informação.** [S.I.: s.n.], 2005. v. 3.
- ROCHA, F. G.; SABINO, R. F.; ACIPRESTE, R. H. L. A metodologia scrum como mobilizadora da prática pedagógica: um olhar sobre a engenharia de software. **FEES 2015**, p. 13, 2015.
- ROCHA, H.; TERRA, R. TerraER: Uma ferramenta voltada ao ensino do modelo de entidade-relacionamento. In: **VI Escola Regional de Banco de Dados (ERBD).** [S.I.: s.n.], 2010. p. 1–4.
- ROSEMBERG, C. et al. Prototipação de software e design participativo: uma experiência do atlântico. **IHC**, v. 8, p. 312–315, 2008.
- ROSSINI, A. M. Administração de sistemas de informação e a gestão do conhecimento. 2006.
- SCHMITZ, E.; ALENCAR, A. **Análise de Risco em Gerência de Projetos - 3ª Edição.** BRASPORT, 2012. ISBN 9788574525426. Disponível em: <<https://books.google.com.br/books?id=ubZ7YHtWZ08C>>.
- SILVA, J. P. S. d. Sasml: a uml based domain specific modeling language for self adaptive systems conceptual modeling. 2018.
- SILVA, V. M.; BARBOSA, R. d. M.; ADAMATTI, D. F. Princípios de usabilidade e a importância do usuário no projeto de interfaces. **sistema**, v. 15, p. 18, 2016.
- SOFTBYTE. 2019. Disponível em: <<http://softbyte.com.br>>.
- SOMASUNDARAM, G.; SHRIVASTAVA, A.; SERVICES, E. **Armazenamento e Gerenciamento de Informações: Como armazenar, gerenciar e proteger informações digitais.** Boekman, 2009. ISBN 9788577807642. Disponível em: <<https://books.google.com.br/books?id=d8uCfC46hwsC>>.

- STAIR, R. M. Princípios de sistemas de informação: uma abordagem gerencial. 2008.
- TORONTO, I. I. de Análise de Negócios de. **O guia para o corpo de conhecimento de análise de negócios.** [S.l.: s.n.], 2005.
- VARGAS, R. V. **Manual Prático do Plano de Projeto, Utilizando o PMBOK Guide.** [S.l.: s.n.], 2018.
- VAZQUEZ, C.; SIMÕES, G. **Engenharia de Requisitos: software orientado ao negócio.** BRASPORT, 2016. ISBN 9788574527901. Disponível em: <<https://books.google.com.br/books?id=gA7kDAAAQBAJ>>.