



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

André Luiz Nunes Amaru

**Desenvolvimento de metodologia e implementação de um softbot para uma
planta industrial didática – o Caso do Laboratório de Automação e Informática
Industrial da UFSC**

Florianópolis
2022

André Luiz Nunes Amaru

Desenvolvimento de metodologia e implementação de um softbot para uma planta industrial didática – o Caso do Laboratório de Automação e Informática Industrial da UFSC

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.

Orientador: Prof. Ricardo José Rabelo

Coorientador: Prof. Saulo Popov Zambiasi

Florianópolis

2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Amaru, André Luiz Nunes

Desenvolvimento de metodologia e implementação de um softbot para uma planta industrial didática - o Caso do Laboratório de Automação e Informática Industrial da UFSC / André Luiz Nunes Amaru ; orientador, Ricardo José Rabelo, coorientador, Saulo Popov Zambiasi, 2022.

78 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2022.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Arisa Nest. 3. Softbot. 4. MongoDB. 5. Python. I. Rabelo, Ricardo José. II. Zambiasi, Saulo Popov. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

André Luiz Nunes Amaru

Desenvolvimento de metodologia e implementação de um softbot para uma planta industrial didática – o Caso do Laboratório de Automação e Informática Industrial da UFSC

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 13 de dezembro de 2022.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Ricardo José Rabelo, Dr.
Orientador
UFSC/CTC/DAS

Prof. João Carlos Espíndola Ferreira, Dr.
Avaliador
UFSC/CTC/EMC

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

AGRADECIMENTOS

Aos meus pais Marcial Amaru Gómez e Rosangela Mendes Nunes Amarú, e a minha irmã Victória Nunes Amaru, por sempre me darem suporte e incentivo em todas as fases da minha vida.

À Universidade Federal de Santa Catarina e especialmente ao Departamento de Automação e Sistemas, pelas oportunidades oferecidas ao longo da graduação.

Ao meu orientador Prof. Dr. Ricardo José Rabelo, por toda a ajuda e suporte oferecidos ao longo do trabalho.

Aos meus amigos, que sempre me apoiaram e serviram de inspiração.

*“Knowing is not enough, we must apply.
Willing is not enough, we must do.”
(Bruce Lee)*

RESUMO

Com o desenvolvimento de novas tecnologias, o que eram ferramentas de lazer como smartphones e computadores, ganham cada vez mais seu espaço como instrumento de trabalho. Já na indústria, as novas tecnologias resultam em um ambiente com muita mais informação e conectividade, a indústria 4.0. Um tipo de ferramenta que pode agregar, tanto na interação com a máquina quanto na manipulação e extração de informações vinda de dados são os *softbots*. Os *softbots* modernos podem ser usados para orientar os trabalhadores em diferentes etapas da produção da empresa, trazendo análises aprofundadas de dados periodicamente - que estão cada vez mais abundantes, no alerta de alguma possível emergência e na interação direta, respondendo às perguntas do usuário. Este trabalho propõe o desenvolvimento de um *softbot* com capacidade de conversação, integrado à planta didática FESTO MPS localizada no LAI (Laboratório de Automação e Informática Industrial), composta por 7 estações de trabalho e respectivos CLPs Siemens, junto a uma metodologia descritiva do processo. Para desenvolver a arquitetura e programação do *softbot* foi utilizada a plataforma de assistentes virtuais Arisa Nest. Na plataforma foram programados scripts em *Lua* e feitas requisições via API para o banco de dados (MongoDB). A população do banco de dados com informações em tempo real da planta foi feita através de um programa, criado na linguagem *Python*. O trabalho resultou na concepção de um *softbot* capaz de interagir com o usuário e prover informações pertinentes sobre a planta MPS, bem como a uma metodologia descritiva sobre o processo.

Palavras-chave: Arisa Nest, *Softbot*, *Python*, *Lua*, MongoDB.

ABSTRACT

With the development of new technologies, what were free time tools such as smartphones and computers, are increasingly gaining space as a working tool. In industry, new technologies result in an environment with much more information and connectivity, the industry 4.0. A type of tool that can add, both in the interaction with the machine and in the manipulation and extraction of information from data, are *softbots*. Modern *softbots* can be used to guide workers in different stages of the company's production, bringing in-depth analyzes of data periodically - which are increasingly abundant, in the alert of any possible emergency and in direct interaction, answering user's questions. This work proposes the development of a *softbot* with chatting capabilities, integrated to the FESTO MPS didactic plant located in LAI (Laboratory of Automation and Industrial Informatics), composed of 7 workstations and respective Siemens PLCs, together with a descriptive methodology of the process. To develop the architecture and programming of the *softbot*, the virtual assistants platform Arisa Nest was used. On the platform, scripts were programmed in *Lua* and requests were made via API to the database (MongoDB). The population of the database with real-time information was done through a program, created in Python. The work resulted in the design of a *softbot* capable of interacting with the user and providing relevant information about the MPS plant, as well as a descriptive methodology about the process.

Keywords: Arisa Nest, *Softbot*, *Python*, *Lua*, MongoDB.

LISTA DE FIGURAS

Figura 1 – Diálogo entre usuário e ELIZA.	22
Figura 2 – Diálogo entre PARRY e ELIZA.	22
Figura 3 – Diálogo entre usuário e ALICE.	23
Figura 4 – <i>Chatbot</i> da Estée Lauder Companies.	24
Figura 5 – Exemplo de conversa entre aluno e METIS.	25
Figura 6 – Siri.	26
Figura 7 – Dispositivo <i>Echo</i> , utilizado para interagir com Alexa.	27
Figura 8 – Exemplo de diagrama BPMN	28
Figura 9 – Exemplo de diagrama UML	29
Figura 10 – Início da <i>pool</i> da planta, referente a tarefas iniciais.	31
Figura 11 – Parcela intermediária da <i>pool</i> da planta, responsável por possibilitar o acesso aos dados por meios externos.	32
Figura 12 – Exemplo de dispositivos IoT	32
Figura 13 – Término da <i>pool</i> da planta, parte na qual o processo é iniciado e seus dados passíveis de obtenção pelo programa desenvolvido.	33
Figura 14 – <i>Pool</i> do programa de coleta de dados.	34
Figura 15 – Tarefas <i>offline</i> da <i>pool</i> do <i>softbot</i> , referentes a seleção e criação de possíveis interações	34
Figura 16 – Parte inicial da <i>pool</i> do <i>softbot</i> , referente a identificação da pergunta do usuário.	35
Figura 17 – Parte intermediária da <i>pool</i> do <i>softbot</i> , a qual mostra cenários de consulta ao banco de dados ou sistema(s) externo(s).	36
Figura 18 – Parte final da <i>pool</i> do <i>softbot</i> , com os possíveis comportamentos do <i>softbot</i>	36
Figura 19 – Parte de criação e configuração do <i>bot</i>	41
Figura 20 – Contextos existentes na Marin.	42
Figura 21 – Campos para configuração de um diálogo.	43
Figura 22 – Exemplos de aplicação de padrões.	43
Figura 23 – Configuração do diálogo de fuga.	44
Figura 24 – Exemplo de resposta de diálogo que contém a execução de um <i>script</i>	45
Figura 25 – Interface de configuração de comportamento.	45
Figura 26 – Comportamentos configurados.	46
Figura 27 – Configuração do comportamento apontado.	47
Figura 28 – Código do comportamento apontado.	47
Figura 29 – Configuração de diálogo sem saber nome do usuário.	48
Figura 30 – Configuração de diálogo sabendo o nome do usuário.	48
Figura 31 – Diagrama da equivalência de termos entre SQL e MongoDB.	49

Figura 32 – Planta didática MPS do LAI.	50
Figura 33 – Exemplo de <i>tags</i> dentro do TIA Portal.	51
Figura 34 – Interface de configuração da IHM como servidor OPC.	52
Figura 35 – Arquitetura do banco de dados.	53
Figura 36 – Dados de um processo.	54
Figura 37 – Entrada de dados da estação de "Classificação".	54
Figura 38 – Implementação de diálogo para solicitar informações até o momento.	56
Figura 39 – Conversa sobre informações até o momento.	56
Figura 40 – Diálogo de informações sobre um processo específico.	57
Figura 41 – Histórico de conversa sobre um processo específico.	57
Figura 42 – Conjunto de padrões e respostas para a funcionalidade de averiguação de processo em andamento.	58
Figura 43 – Diálogo com usuário onde há processo em andamento.	58
Figura 44 – Notificação de relatório diário de operações.	59
Figura 45 – Verificações que ocorrem a cada intervalo de tempo.	59
Figura 46 – Trecho responsável pelo envio de mensagens aos usuários.	60
Figura 47 – Exemplo de alerta da etapa de distribuição.	61
Figura 48 – Exemplo de alerta da etapa de separação.	61
Figura 49 – Exemplo de alerta da etapa de classificação.	61
Figura 50 – Configuração de comportamento que monitora a emissão de alertas.	62
Figura 51 – Diagrama da estrutura de funcionamento	62
Figura 52 – Respostas do questionário.	65

LISTA DE ABREVIATURAS E SIGLAS

AIML	<i>Artificial Intelligence Markup Language</i>
BPMN	<i>Bussiness Process Modeling Notation</i>
CLP	Controlador Lógico Programável
DS	<i>Design Science</i>
IA	Inteligência Artificial
IHM	Interface Homem-Máquina
IoT	<i>Internet of Things</i>
IPA	<i>Intelligent Personal Assistant</i>
LAI	Laboratório de Automação e Informática Industrial
OPC	<i>Open Platform Communications</i>
OPC DA	<i>Open Platform Communications Data Access</i>
SOA	<i>Service Oriented Architecture</i>
UML	<i>Unified Modeling Language</i>
XML	<i>eXtensible Markup Language</i>

SUMÁRIO

1	INTRODUÇÃO	14
1.1	PROBLEMA	15
1.2	OBJETIVO GERAL	15
1.3	OBJETIVOS ESPECÍFICOS	15
1.4	ORGANIZAÇÃO DO DOCUMENTO	16
2	METODOLOGIA DA PESQUISA	17
2.1	ETAPAS PARA DESENVOLVIMENTO DO <i>SOFTBOT</i>	18
2.1.1	Investigação do Problema	19
2.1.2	Possíveis soluções	19
2.1.3	Implementação da solução	20
2.1.4	Validação da solução	20
2.1.5	Avaliação do projeto	20
3	FUNDAMENTAÇÃO TEÓRICA	21
3.1	<i>CHAT/SOFTBOTS</i> E ASSISTENTES PESSOAIS INTELIGENTES	21
3.1.1	ELIZA	21
3.1.2	PARRY	22
3.1.3	ALICE e AIML	23
3.1.4	Estée Lauder Companies	23
3.1.5	METIS	24
3.1.6	Siri	25
3.1.7	Alexa	26
3.2	BANCO DE DADOS NOSQL	27
3.3	MODELAGEM DE PROCESSOS DE NEGÓCIO	27
4	PROPOSTA E ESPECIFICAÇÕES	30
4.1	METODOLOGIA DO <i>SOFTBOT</i>	30
4.1.1	Planta	31
4.1.2	Programa para coleta de dados	33
4.1.3	<i>Softbot</i>	34
4.1.3.1	<i>Offline</i>	34
4.1.3.2	<i>Online</i>	35
4.2	ESTRUTURAÇÃO DO <i>SOFTBOT</i>	37
4.2.1	Comportamento	37
4.3	REQUISITOS	38
4.3.1	Requisitos funcionais	38
4.3.2	Requisitos não funcionais	38
4.4	CASOS DE USO	39
4.4.1	Solicitar informações do dia ou de um dado processo	39

4.4.2	Averiguar se há um processo em andamento e suas informações	39
4.4.3	Emissão de relatório diário em horário específico	40
4.4.4	Verificação de status de conexão com BD ou realização de processo(s), em um dado intervalo de tempo	40
4.4.5	Emissão de alertas	40
5	FERRAMENTAS	41
5.1	ARISA NEST	41
5.1.1	Criação e configuração	41
5.1.2	Contextos	42
5.1.3	Dialógos	42
5.1.4	Padrões	43
5.1.5	Respostas	43
5.1.6	Crenças	44
5.1.7	Condições	44
5.1.8	Scripts	44
5.1.9	Comportamentos	45
5.1.10	Aplicativo de mensagens	46
5.2	EXEMPLO	46
5.2.1	Comportamento de início do dia de trabalho	46
5.2.2	Diálogo para obter nome do usuário	47
5.3	MONGODB ATLAS	48
6	IMPLEMENTAÇÃO	50
6.1	METODOLOGIA APLICADA	50
6.1.1	Planta MPS do LAI	50
6.1.2	Programa para coleta de dados	52
6.2	FUNCIONALIDADES	55
6.2.1	Comportamento Responsivo	55
6.2.1.1	Solicitar informações até o momento	55
6.2.1.2	Solicitar informações específicas de um processo	56
6.2.1.3	Averiguar se há um processo em andamento e suas informações	57
6.2.2	Comportamento Programado	58
6.2.3	Comportamento Proativo	60
6.3	ESTRUTURA DE FUNCIONAMENTO	62
7	AVALIAÇÃO DE RESULTADOS	63
7.1	METODOLOGIA	63
7.2	<i>SOFTBOT</i>	63
7.2.1	Análise por professores	64
7.2.2	Análise de funcionamento	65
8	CONCLUSÃO E PERSPECTIVAS	67

	REFERÊNCIAS	68
	APÊNDICE A – DIAGRAMA BPMN PARA DESENVOLVIMENTO E IMPLEMENTAÇÃO DE <i>SOFTBOT</i>	73
	APÊNDICE B – PROGRAMA PARA LEITURA DE INFORMAÇÕES DA PLANTA E ENVIO PARA BANCO DE DADOS	74
	APÊNDICE C – QUESTIONÁRIO PARA AVALIAÇÃO DO <i>SOFTBOT</i>, POR PARTE DOS PROFESSORES AVALIADORES	75
	APÊNDICE D – GUIA DE UTILIZAÇÃO PARA USUÁRIO E DESEN- VOLVEDOR	77
D.1	DESENVOLVEDOR	77
D.2	USUÁRIO	77

1 INTRODUÇÃO

O desenvolvimento de novas tecnologias foi um fator essencial para revolucionar os meios de comunicação tradicionais. No ambiente digital, a troca de informações é realizada de maneira muito rápida e inclusiva, o que leva a mensagens transmitidas tanto em maior velocidade quanto em maior quantidade quando comparadas aos meios antigos. Como resultado, o público passou a ser mais rigoroso tanto na qualidade do atendimento bem como em sua velocidade (EXAME, 2021).

Para auxiliar na rapidez, bem como na qualidade do atendimento, pode-se utilizar a linguagem homem-máquina. Sua evolução está tornando-a cada vez mais próxima da linguagem natural, utilizada nas conversas entre humanos. Com o desenvolvimento contínuo da IA e do aprendizado de máquina, logo não seremos capazes de saber se estamos falando com um robô ou um ser humano (ADAMOPOULOU; MOUSSIADES, 2020).

Desenvolver um sistema capaz de expressar pensamentos em linguagem natural através de uma conversa com humanos é o objetivo de muitos pesquisadores. Em 1950 Alan Turing propôs um teste, o qual é baseado na proposta de que, se as máquinas conseguirem se passar por humanos em uma conversação, estas podem ser consideradas inteligentes (MUGGLETON, 2014).

A partir deste momento, surgiram os sistemas de interação homem-máquina, como *chatbots* e *softbots*. Enquanto *chatbot* é um programa projetado para simular uma conversação humana de forma inteligente (DAHIYA, 2017), um *softbot* ou *software robot* é um sistema virtual que tem a capacidade de se conectar a qualquer dispositivo através de uma rede e atuar em diferentes atividades, com autonomia e proatividade (KIM, 2005).

De acordo com um levantamento, até 2023 os setores de varejo, bancário e saúde economizarão até US\$11.5 bilhões até 2023 (RESEARCH, 2018). Não são apenas tais setores que devem se beneficiar da aplicação de *chat/softbots*, mas também a indústria. De acordo com (ABNER *et al.*, 2020), com um melhor nível de interação homem-máquina e com o apoio de tecnologias avançadas (como *softbots*), as pessoas ganharão protagonismo em diversas atividades do setor.

De acordo com (RABELO, Ricardo J; ROMERO, David; ZAMBIASI, S. P., 2018), *softbots* podem auxiliar no desenvolvimento de uma melhor interação homem-máquina, a qual empodera o operador e conseqüentemente a sustentabilidade da Indústria 4.0. Partindo deste princípio, o objetivo deste projeto baseava-se em torno da demonstração das capacidades de tal tecnologia, implementada em um ambiente didático.

1.1 PROBLEMA

Este projeto tem como objetivo conceber uma metodologia para orientar o desenvolvimento de *softbots*, aplicando-a para realizar o desenvolvimento e implementação de um *software robot* a uma planta MPS localizada no Laboratório de Automação e Informática Industrial (LAI). Sua implementação visa um ambiente de manufatura sustentável nas fábricas do futuro, onde operadores inteligentes e qualificados devem realizar não apenas trabalho cooperativo com robôs, mas também trabalho auxiliado por máquinas. Para hospedar e gerenciar o *softbot*, utiliza-se uma PaaS (*Plataform as a Service*) chamada Arisa Nest, onde por meio de uma interface web pode-se configurar e desenvolver o *bot*. Os *bots* criados são integráveis em diversos domínios e suas funcionalidades são programadas através de scripts. Para possibilitar o acesso aos dados da planta por parte do *softbot* é necessária também a implementação de um banco de dados. A ideia deste projeto é que nos ambientes de Indústria 4.0 e 5.0, o auxílio ao ser humano é cada vez mais necessário. A quantidade de dados que podem gerar novas análises cresce a cada dia, além de requerer agilidade e confiabilidade no processo. Nesses aspectos um *softbot* tem grande capacidade de atuar como uma ferramenta de apoio aos seus usuários e até mesmo executar certas atividades.

1.2 OBJETIVO GERAL

O objetivo geral deste projeto é o de descrever uma metodologia de projeto e implementação de *softbots* industriais e desenvolver um *softbot* com base nela para uma planta didática industrial.

1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos são fragmentos do objetivo geral e devem ser realizados ao longo do projeto para validarem sua aplicação. Estes objetivos são:

- Revisar e apurar material bibliográfico sobre *chat/softbots* e sua integração em sistemas;
- Analisar e estudar fluxo da planta MPS e comunicação entre CLP e PC;
- Estudar e entender a plataforma Arisa Nest;
- Criar *softbot* na plataforma, montando contextos de conversação próprios para a aplicação do projeto;
- Desenvolvimento e integração do *softbot* e da planta, através do banco de dados MongoDB;
- Teste e avaliação de resultados;

1.4 ORGANIZAÇÃO DO DOCUMENTO

Além da introdução apresentada, o restante do projeto está distribuído em mais 8 capítulos. O capítulo 2 é referente ao método de pesquisa selecionado para desenvolvimento do projeto. Já o capítulo 3 serve como fundamentação teórica do projeto, citando algumas referências que serviram como pesquisa base. No capítulo 4 mostra-se a proposta de desenvolvimento do *softbot*, bem como suas especificações. No capítulo 5 são descritas as ferramentas e métodos utilizados. Já no capítulo 6 explica-se a implementação de cada parte do projeto. O teste e avaliação de resultados são citados no capítulo 7. Por fim, no capítulo 8 há uma síntese do projeto, bem como possíveis aplicações futuras.

2 METODOLOGIA DA PESQUISA

APPOLINÁRIO (2004) considera a pesquisa como um processo, o qual busca dar respostas aos problemas que lhe são apresentados, uma “investigação sistemática de determinado assunto que visa obter novas informações e/ou reorganizar as informações já existentes sobre um problema específico ou bem definido”.

Já GIL (2010) afirma que a pesquisa é um processo formal realizado utilizando conhecimentos acessíveis e empregando métodos e técnicas de investigação. É um processo longo que envolve diversas etapas, desde a adequada formulação do problema até a satisfatória apresentação dos resultados.

A metodologia de pesquisa adotada foi a *Design Science* (DS) (JÄRVINEN, 2007). Esse método de pesquisa é conhecido por sua dinâmica e orientação à solução de problemas, sendo aplicado principalmente em áreas como medicina, gestão e engenharia. De acordo com Venable (2006), a *Design Science* tem como objetivo gerar conhecimento que seja aplicável e útil para a solução de problemas, seja através da melhoria de sistemas já existentes ou da criação de novas soluções. A metodologia DS é realizada através de um método sequencial e cíclico, onde ocorrem validações teóricas e práticas (WIERINGA, 2009). No âmbito do desenvolvimento de softwares e novas tecnologias é essencial a avaliação constante, para assegurar resultados e garantir a integridade do sistema desenvolvido.

No DS, essas validações usualmente são feitas também consultando grupos de usuários, de forma a que seus feedbacks sejam incorporados ao objeto de desenvolvimento, num ciclo de melhoria, até que se considere que o objeto atenda aos objetivos e requisitos para o qual fora desenvolvido.

Como o objeto de desenvolvimento era, além de uma metodologia, a implementação de um software (no caso, um protótipo inicial de *softbot*), foi utilizada uma metodologia de software. No caso, o método ‘cascata’ (*waterfall*) (SOMMERVILLE, 2019).

A decisão por tal metodologia deveu-se a que já havia uma grande experiência do grupo de pesquisas quanto ao desenvolvimento de *softbots* (embora sem se basear inicialmente numa metodologia mais ‘formal’ de desenvolvimento de *softbots*, e que é um dos resultados deste PFC). Portanto, já se conhecia antecipadamente as ferramentas a usar, a arquitetura do *softbot*, e suas funcionalidades gerais. Caberia apenas ouvir potenciais usuários do *softbot* quanto as situações de produção e mensagens (incluindo termos, etc.) que seriam trocadas e ir gradualmente implementando, testando, refinando e finalmente validar e ter a versão final de cada mensagem / situação de produção (nos modos responsivo, planejado e proativo, como se verá mais a frente).

Um trabalho anterior desenvolvido no grupo de pesquisas, de um mestrado que visava o desenvolvimento de um gêmeo digital para a mesma planta MPS do LAI, seria

usado como base para o trabalho do *softbot*. Isto porque toda a parte de aquisição em tempo real de dados dos CLPs das várias estações de trabalho da planta e seu armazenamento em um banco de dados tinha sido já implementada. Todavia, não foi possível ter acesso a tal “base”, e então tudo deve que ser reimplementado. Com isto, o já relativamente usual curto tempo para o desenvolvimento de um PFC, ficou ainda mais apertado. Como consequência, não se pode realizar ciclos de entrevistas e melhorias junto a usuários (no caso, de professores que utilizam a planta MPS-LAI nas suas várias atividades de ensino, pesquisa e extensão), senão apenas colhendo suas opiniões ao final do projeto (este ponto é discutido no Capítulo 7).

Do ponto de vista tecnológico, a implementação limitou-se a prover acesso ao *softbot* apenas através de smartphones e de computadores com um navegador. Acesso via tablet ou vestíveis não foram implementados neste protótipo. Além disto, devido a que a biblioteca de tratamento de voz da Google passou a ser paga, na implementação feita a interação com o *softbot* é feita apenas via digitação de texto.

Do ponto de vista de arquitetura do *softbot* e mensagens suportadas, o framework utilizado para derivar *softbots* particulares (Arisa Nest, explicado no Capítulo 5) permite que sejam introduzidas tantas mensagens quantas se desejar, de forma escalar, bem como vários “níveis de inteligência” para se interpretar e tratar cada mensagem. Entretanto, visando-se a construção de um protótipo / versão inicial de um *softbot* para a planta MPS-LAI, tratou-se de criar apenas um conjunto relativamente limitado de perguntas para os três modos possíveis. Dada aquela escalabilidade e a agora disponibilização do *softbot*, próximos desenvolvimentos e melhorias poderão ser realizados em futuros trabalhos de estágio, PFC ou mesmo de mestrado, a partir dessa versão inicial.

2.1 ETAPAS PARA DESENVOLVIMENTO DO *SOFTBOT*

Para a realização deste projeto, o primeiro passo foi a análise sobre o estado da arte de *chat/softbots*, material este que incluía abordagens tanto teóricas quanto práticas. Como requisito para programação e integração do *softbot* à planta, foram também conduzidas pesquisas técnicas sobre a linguagem de programação *Lua* e *Python*, protocolo REST e Arquitetura orientada a serviços (SOA).

Após o período de estudo inicial, o objetivo passou para o aprendizado e entendimento de funcionalidades da plataforma de desenvolvimento escolhida, Arisa Nest. Neste período de adaptação à plataforma foram montadas estruturas que serviram de base no desenvolvimento do *softbot*.

Por ter sido definido também como um dos objetivos do trabalho a integração do *bot* à planta MPS presente no LAI, foi necessária a criação de um banco de dados. A opção escolhida foi o banco de dados NoSQL MongoDB, um banco de dados mais avançado e capaz de lidar com *Big Data* com recursos de esquema dinâmico, quando

comparado ao tradicional SQL. Vale citar que por oferecer uma opção sem custo, de boa resposta e com serviço de API próprio, é utilizado um banco de dados na nuvem, o que evitou a necessidade da hospedagem do banco de dados em máquinas virtuais.

A próxima etapa foi a criação da rotina em *Python*, responsável por popular o banco de dados com informações da planta. Para tornar possível a leitura das *Tags* (forma como são lidos e escritos dados em um CLP) foi criado um servidor utilizando o TIA Portal (*software* próprio da Siemens para programação de CLPs). Este servidor utiliza o padrão de comunicação *Open Platform Communications Data Access* (OPC DA), o qual permite uma comunicação contínua e em tempo real de dados entre equipamentos de aquisição de dados. Por fim, foi utilizado um pacote de *Python* que age como cliente OPC e possibilita o manuseio dos dados via código.

Por fim foi realizado o desenvolvimento de scripts em linguagem Lua, responsáveis por extrair informações do banco de dados através de requisições REST e disponibilizá-las ao usuário. Durante esta etapa houveram diversos testes, avaliações e retrabalhos dos scripts, com o intuito de abranger um maior número de possíveis conversações, bem como oferecer informações pertinentes do processo. Mais ao final do processo, a escrita deste documento passou a ser feita de forma paralela ao trabalho.

As etapas aqui citadas foram planejadas e desenvolvidas de acordo com a metodologia *Design Science*, como pode-se observar a seguir.

2.1.1 Investigação do Problema

A primeira fase envolve a investigação e entendimento do problema de forma geral. O problema em questão envolve o desenvolvimento de uma metodologia e integração de um *softbot* planejado à uma planta. Para tal, observou-se que a utilização da plataforma Arisa Nest supriria todas as necessidades na concepção de um *softbot* com habilidades de conversação.

Já com relação a comunicação com a planta, foi observada a necessidade de uma comunicação entre CLP e PC, bem como o armazenamento dos dados. Por fim, após um período de análise, observou-se que um banco de dados em nuvem forneceria, além de um acesso mais fácil à informação, maior segurança no armazenamento dos dados.

2.1.2 Possíveis soluções

Seguinte ao entendimento do problema, iniciou-se a etapa de pesquisa sobre aplicações de *softbots* na indústria. O intuito da pesquisa foi observar possíveis aplicações, bem como suas funções, integrações e diferentes categorias. Mesmo com a seleção prévia da plataforma de gerenciamento Arisa Nest, a pesquisa bibliográfica serviu de embasamento para o entendimento de *chat/softbots* que já estão implementados.

Durante esta etapa também foi realizada a pesquisa sobre bancos de dados NoSQL e suas implementações, além de uma análise entre hospedagem local e hospedagem remota (nuvem), como por exemplo tempo de resposta e acesso via APIs (Interface de Programação de Aplicação). Por fim foi realizada uma análise de pacotes públicos de *Python* que envolvem comunicação OPC.

2.1.3 Implementação da solução

Após a definição do projeto deu-se início aos primeiros passos da implementação da solução. O objetivo era simplesmente habituar-se a plataforma Arisa Nest, bem como solicitar acesso ao LAI. A etapa de familiarização com o laboratório não foi longa, visto que é um ponto obrigatório durante a graduação.

Com o intuito de agilizar o desenvolvimento e obter conhecimento sobre a interação com a planta, tentou-se o contato com um pesquisador que desenvolvia um projeto na bancada, mas não houve sucesso. Após isso observou-se que o laboratório não estava sempre disponível, o que causou um desenvolvimento intercalado entre *softbot* e comunicação entre CLP e PC.

Com o desenvolvimento da comunicação com o CLP foi possível iniciar a população do banco de dados e implementar os *scripts* em *Lua*. Com o avanço no desenvolvimento do código base do *softbot*, foi possível a incremental criação de cenários de conversação.

2.1.4 Validação da solução

O *software robot* desenvolvido não foi apresentado para testes, tornando-o apenas avaliado pelos requisitos de projeto. O desenvolvimento de cada nova funcionalidade foi um processo sequencial e cíclico como dita o método *Design Science*, além de testadas a cada nova modificação. Conforme novas possibilidades foram criadas, o *softbot* possuía novos dados para disponibilizar, seja em formato de conversação ou relatório.

2.1.5 Avaliação do projeto

Esta última etapa do *Design Science* consiste na avaliação do que foi realizado e dos objetivos alcançados. Após isso, são apresentadas possíveis melhorias no projeto, bem como aplicações futuras.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata-se da fundamentação teórica para desenvolvimento do projeto e abrange conceitos utilizados neste trabalho, obtidos pela pesquisa bibliográfica realizada.

3.1 CHAT/SOFTBOTS E ASSISTENTES PESSOAIS INTELIGENTES

Apesar de pesquisas sobre *softbots* não serem tão recentes, sua aplicação é. *Chatbots* e IPAs são exemplos de implementações de robôs via *software*, ou *softbots*, que evoluíram drasticamente nos últimos anos. Sua aplicação combinada com tecnologias, como Realidade Aumentada, representa inúmeras possibilidades na indústria (ZAMBIASI, L. P. *et al.*, 2022).

Chatbot é um agente que visa reproduzir uma discussão inteligente com humanos, através de sua linguagem natural. Atualmente, os *chatbots* são utilizados por muitos para prover o acesso a informações ou bases de aprendizagem e, além disso, fazer discussões não específicas (DAHIYA, 2017). É comum encontrar este tipo de aplicação em áreas como educacional, entretenimento e até no setor comercial.

Intelligent Personal Assistant (IPA) é um sistema que pode entender, responder a entradas faladas ou escritas, além de processar a solicitação do usuário (SANTOS *et al.*, 2016). Ao combinar conceitos de inteligência artificial e técnicas de processamento de linguagem natural, um IPA pode fornecer informações para os mais diversos assuntos como clima, mapas, programação, chamada, eventos, etc.

3.1.1 ELIZA

O primeiro *chatbot* criado foi ELIZA, feito por (WEIZENBAUM, 1966) para simular conversas com um psicoterapeuta. ELIZA foi um dos primeiros programas para processamento de linguagem natural da história.

A premissa de ELIZA era simples: Utilizar partes das frases do usuário como futuras novas perguntas e, quando isso não fosse possível, recorrer a frases prontas. Apesar da simplicidade de funcionamento, diversas pessoas atribuíram à ELIZA características muito semelhantes aos sentimentos humanos. Ela não tinha a capacidade de armazenar dados da conversa com o usuário e suas respostas seguiam a premissa de serem amigáveis.

Figura 1 – Diálogo entre usuário e ELIZA.

USER – Men are all alike.
ELIZA – IN WHAT WAY
They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE?
Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED ...
WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU?
My father.
YOUR FATHER

Fonte: Adaptado de (WEIZENBAUM, 1966).

3.1.2 PARRY

PARRY (COLBY *et al.*, 1972) representou um marco no desenvolvimento da Inteligência Artificial. Ele foi projetado para simular com a maior precisão possível os padrões de pensamento de um paciente esquizofrênico paranoico e inclusive interagiu em diferentes oportunidades com ELIZA, sendo a mais marcante no mesmo ano de sua criação, na *International Conference on Computer Communications*.

Além de interagir com outro *chatbot*, foi o primeiro *chatbot* a passar no Teste de Turing, quando não foi possível distinguir PARRY de um indivíduo paranoico real.

Figura 2 – Diálogo entre PARRY e ELIZA.

ELIZA: Good Evening. Tell me your problems. Please terminate input with a period or a question mark.
PARRY: People get on my nerves sometimes.
ELIZA: I am not sure I understand you fully.
PARRY: You should pay more attention.
ELIZA: Suppose you should pay more attention.
PARRY: You're entitled to your own opinion.
ELIZA: What makes you think I am entitled to my own opinion?
PARRY: It bothers me just to be around people in general.
ELIZA: Please go on.
PARRY: I went to the track at Bay Meadows a while back.

Fonte: Adaptado de (SAM, 2020).

3.1.3 ALICE e AIML

A.L.I.C.E. (*Artificial Linguistic Internet Computer Entity*) é um *chatbot* de processamento de linguagem natural, criado por Richard Wallace e inspirado em ELIZA.

O principal diferencial do *chatbot* é que ele opera utilizando a linguagem AIML (*Artificial Intelligence Markup Language*), a primeira versão de uma linguagem para *chatbots*, desenvolvida com origem na linguagem *eXtensible Markup Language* (XML) (WALLACE, 2003).

A AIML foi criada pelo autor do *chatbot*, Richard Wallace, e é baseada nos conceitos de reconhecimento de padrões. Tal linguagem é utilizada na modelagem do diálogo homem-máquina e segue a abordagem estímulo e resposta. Para isso, um conjunto de possíveis entradas do usuário são modeladas e cada uma delas leva a uma resposta pré-programada (MARIETTO *et al.*, 2013).

Figura 3 – Diálogo entre usuário e ALICE.

Tell A.L.I.C.E:

Powered by [Pandorabots](#).

Human: Do you know ELIZA?

A.L.I.C.E: Eliza -- also known as doctor, is the original psychiatrist program developed by Joseph Weisenbaum at mit in the 1960's.

Human: What is your job?

A.L.I.C.E: My job is chatting on the internet. I am a paid client.

Human: What does ALICE stand for?

A.L.I.C.E: ALICE = Artificial Linguistic Internet Computer Entity.

Human: Who is Richard Wallace?

A.L.I.C.E: He is a famous computer scientist, the author of ALICE, and a graduate of Carnegie Mellon.

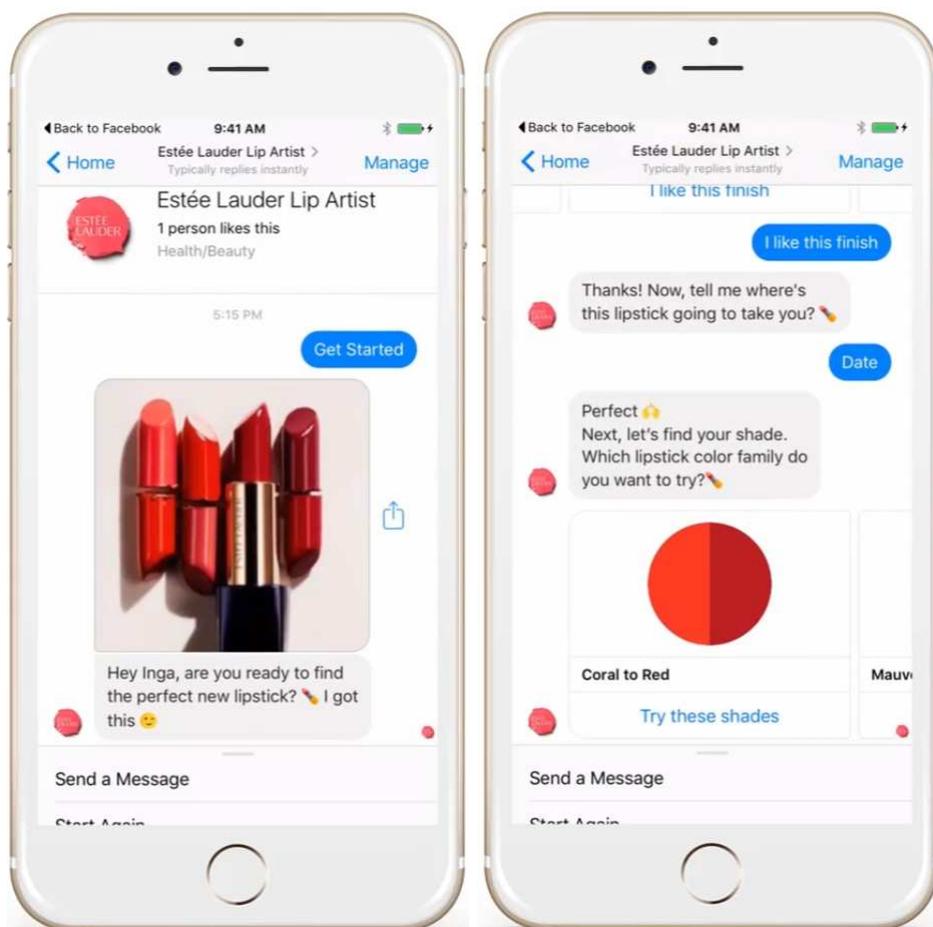
Fonte: Adaptado de Pandorabots.

3.1.4 Estée Lauder Companies

Estée Lauder Companies é uma empresa americana de cosméticos fundada em 1946. A segunda maior empresa do mundo de cosmético uniu realidade aumentada e *chatbots* para aprimorar ainda mais a experiência de seus clientes.

Ao conversar com o *chatbot*, os usuários podem encontrar produtos por meio de uma pesquisa tradicional ou usar a interface de conversação para receber recomendações com base nas respostas de um questionário (NAMA, 2017). O *bot* combinado a realidade aumentada permite que os usuários testem virtualmente seus produtos e os direciona para uma plataforma de comércio eletrônico para comprar um tom de que gostem.

Figura 4 – Chatbot da Estée Lauder Companies.



Fonte: NAMA.

3.1.5 METIS

METIS (Mediadora de Educação em Tecnologia Informática e Socializadora) é um agente criado e utilizado dentro da Universidade Federal do Rio Grando do Sul (UFRGS). Ao possuir na sua base de conhecimento um conjunto de informações relacionadas com a educação e suas tecnologias, o *chatbot* responde ao usuário não apenas com mensagens mas também com vídeos, imagens, áudios e *links* (DA SILVEIRA *et al.*, 2019).

METIS foi desenvolvida com o intuito de impulsionar o crescimento pessoal dos jovens e auxiliar em seu desenvolvimento cognitivo, através da apresentação de conceitos que possibilitam ao humano a resolução do problema.

Figura 5 – Exemplo de conversa entre aluno e METIS.



Fonte: Adaptado de (DA SILVEIRA *et al.*, 2019)

3.1.6 Siri

A Siri é uma assistente pessoal inteligente desenvolvida pela Apple, sendo capaz de compreender comandos de voz e realizar tarefas atreladas ao sistema operacional. Uma das ideias principais dela é servir como uma amiga que responde a perguntas e ajuda a automatizar algumas tarefas do seu cotidiano, interagindo de forma natural com o usuário.

A Siri é baseada na utilização combinada de tecnologias de Inteligência Artificial (IA), reconhecimento de fala e Processamento de Linguagem Natural. Seu sistema é capaz de identificar e assimilar características da voz e dialeto falado pelo usuário, o que torna aprimora sua capacidade de compreensão através do constante uso.

Ao receber solicitações de voz através da interface, a Siri as traduz para código e identifica padrões, frases, palavras-chave para determinar seu significado. Depois desta etapa, ela começa a avaliar e delegar as tarefas necessárias para cumprimento da requisição. Por fim, utiliza de banco de dados online e do telefone para elaborar respostas e atender à solicitação do usuário (APPLE, s.d.).

Figura 6 – Siri.



Fonte: Apple.

3.1.7 Alexa

Alexa é uma assistente pessoal inteligente desenvolvido pela Amazon e lançada em novembro de 2014. Ela serve para auxiliar na execução de tarefas do dia a dia, podendo reproduzir áudio, controlar sua casa inteligente, tirar dúvidas e acionar seus serviços favoritos.

A assistente virtual da amazon pode ser encontrada em diversos dispositivos, mas *smartphones* e *Amazon Echo* são os mais comuns. A *Amazon Echo* é uma linha de produtos de viva-voz e tem como objetivo principal servir para Alexa interagir com o usuário. Quando você pede a Alexa para fazer algo, a assistente virtual grava o que você está dizendo e o envia para o *Alexa Voice Service (AVS)*, para que sua atividade seja atendida e realizada (MUO, 2021).

As possíveis funcionalidades de Alexa são definidas como habilidades e algumas já são intrínsecas do assistente virtual, mas seu aprimoramento e desenvolvimento são abertos ao público. Com o auxílio do serviço de reconhecimento de voz da Amazon, torna-se fácil a utilização de novas habilidades feitas pela comunidade - as quais já somam mais de 100 mil (CNET, 2022).

Figura 7 – Dispositivo *Echo*, utilizado para interagir com Alexa.

Fonte: Amazon.

3.2 BANCO DE DADOS NOSQL

Banco de dados NoSQL, ou *Not Only SQL*, é o termo utilizado para banco de dados não relacionais de alto desempenho. O NoSQL foi desenvolvido com o intuito de ter uma performance aprimorada e uma melhor escalabilidade horizontal, ponto este que bancos de dados relacionais não são tão eficientes.

Um dos principais diferenciais do NoSQL é que toda a informação é agrupada e guardada no mesmo registro. Já no SQL você precisa ter o relacionamento entre várias tabelas para ter a informação, informação esta disposta no modelo entidade e relacionamento.

Como uma das necessidades do projeto é o desempenho e rápida comunicação com o bando de dados, o formato NoSQL ganha destaque quando comparado ao tradicional SQL. Segundo (LI; MANOHARAN, 2013), o MongoDB apresenta melhor desempenho quando comparado a outros SGBD quando empregado o formato chave-valor, como por exemplo o padrão JSON, o que o torna amplamente utilizado.

3.3 MODELAGEM DE PROCESSOS DE NEGÓCIO

Um processo de negócio é uma coleção de atividades ou tarefas relacionadas que têm um início e um ponto final, bem como entradas e saídas claramente definidas. Um processo de negócio pode ser decomposto em vários subprocessos, com características específicas que juntas contribuem aos objetivos do processo básico. Os processos de negócios “representam uma habilidade intelectual crítica e estratégica que precisa ser entendida e gerenciada proativamente” (SCHEDLBAUER, 2010).

Ao longo dos anos, diferentes organizações (como OMG, BPMI, OASIS, IMB, W3C etc.) elaboraram uma série de normas para o projeto, execução, administração e monitoramento de processos de negócios. Esses padrões podem ser usados separa-

damente ou combinados, dependendo da compatibilidade entre eles. De acordo com (KALNINS; VITOLINS, 2006), os dois padrões de linguagem de notação mais populares e amplamente usados atualmente são o *Business Process Modeling Notation* (BPMN) e o *Unified Modeling Language* (UML).

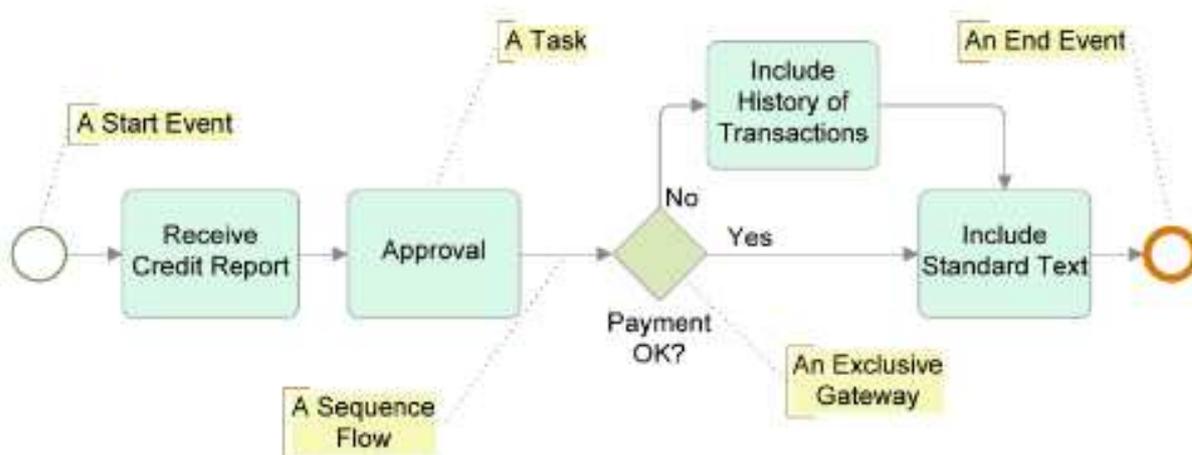
A UML foi desenvolvida e é mantida pela OMG (*Object Management Group*). A primeira versão da UML foi lançado em 1995. O principal objetivo da UML é “auxiliar arquitetos de sistema, engenheiros de *software* e desenvolvedores com ferramentas que possam realizar análise, projeto e implementação de sistemas, bem como para modelagem de negócios e processos similares” (OMG, 2011b).

A *Business Process Modeling Notation* foi desenvolvida em 2004 pela BPMI (*Business Process Management Initiative*), mas é mantida desde 2005 pela OMG, após a fusão entre esta organização e a BPMI. O objetivo principal da BPMN é “fornecer uma notação que seja prontamente compreensível por todos os usuários de negócios, desde os analistas que criam os rascunhos iniciais do processo, aos desenvolvedores técnicos responsáveis pela implementação da tecnologia que realizarão esses processos e, por fim, aos empresários que farão a gestão e monitoramento desses processos” (OMG, 2011a).

Apesar da semelhança da simbologia gráfica entre BPMN e UML, a BPMN está ganhando cada vez mais destaque. Por ser uma metodologia de modelagem orientada a processos, ela pode ser utilizada tanto na área de TI quanto na área de negócios, o que a torna uma ferramenta essencial para um engenheiro moderno.

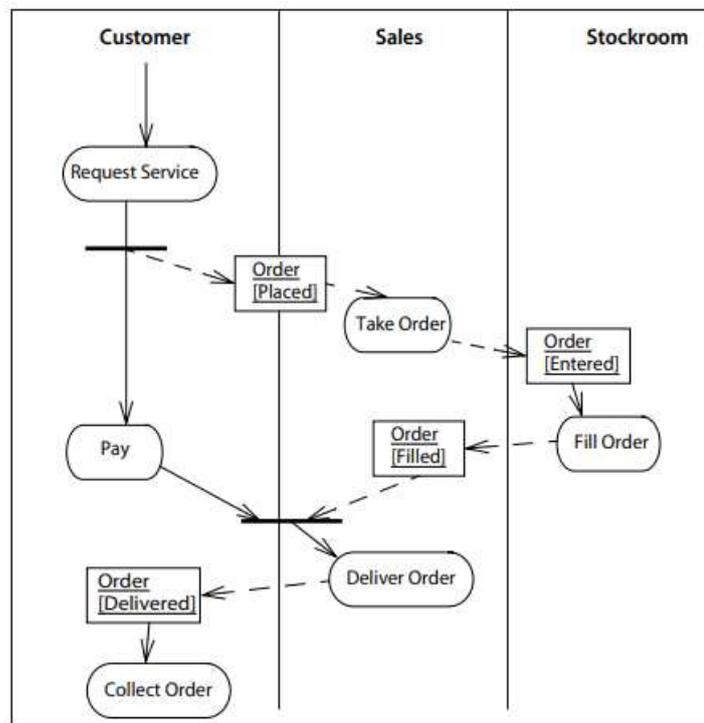
A Figura 8 apresenta um exemplo de diagrama BPMN que emprega elementos amplamente utilizados em seu desenvolvimento, enquanto a Figura 9 demonstra um simples processo através de um diagrama UML.

Figura 8 – Exemplo de diagrama BPMN



Fonte: Adaptado de (WHITE; MIERS, 2008).

Figura 9 – Exemplo de diagrama UML



Fonte: JACOBSON; BOOCH (2021).

4 PROPOSTA E ESPECIFICAÇÕES

Este capítulo trata-se da proposta de desenvolvimento do projeto, a qual especifica a metodologia proposta, a estrutura do *softbot*, bem como seus requisitos e cenários.

4.1 METODOLOGIA DO *SOFTBOT*

As possibilidades de atuação de *softbots*, principalmente como IPA, são inúmeras, seja para auxiliar um usuário a lembrar de um compromisso importante, seja para informar uma situação perigosa em uma fábrica. Na literatura, há muitos artigos que apresentam implementações de versões “limitadas” de *softbots*, na forma de *chatbots*, aplicados em vários domínios de aplicação (ABNER *et al.*, 2020). No entanto, apesar da popularização da tecnologia, não se encontrou artigos que formalizassem a descrição do processo de desenvolvimento de um *softbot* “industrial”.

Dada a esta lacuna, adotou-se uma abordagem *construtivista e dedutiva* de concepção de uma metodologia para tal, usufruindo-se da experiência do grupo em pesquisas sobre desenvolvimentos de *softbots* e de gêmeos digitais, relatadas, por exemplo, em (RABELO, R. J.; MAGALHAES; CABRAL, 2020) (RABELO, R. J.; ZAMBIAZI; ROMERO, 2019). Isto significa que se baseou em relatos e nas publicações do próprio grupo para definir uma dada metodologia particular e posteriormente efetuou-se uma generalização destas.

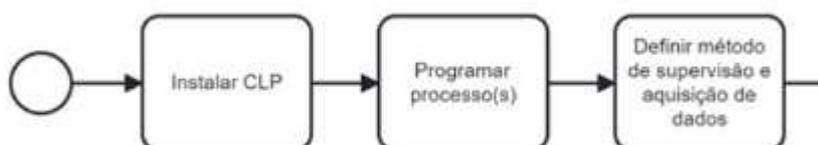
Com o emprego da linguagem de notação BPMN (WHITE; MIERS, 2008), foi possível o desenvolvimento de uma metodologia que auxilia na criação e integração de um *softbot* a uma planta. A metodologia proposta está representada no Apêndice A e é possível fragmentá-la em três partes principais:

- **Planta:** é responsável pela realização do processo e disponibilização de acesso aos seus dados. Para comunicação com um computador há a possibilidade da utilização de técnicas mais modernas, como a instalação e configuração de dispositivos IoT, ou mais tradicionais como o estabelecimento de uma rede industrial e a configuração de um *software* para aquisição de dados.
- **Programa para coleta de dados:** sua função é coletar dados da planta, realizar operações sobre esses dados e enviá-los para o banco de dados. Este programa funciona apenas durante a execução de processos.
- ***Softbot:*** é responsável pela interação com o usuário. Os eventos de começo do fluxo envolvem o tipo de comportamento do *softbot*, os quais podem ser responsivo, programado ou proativo. Ao responder o usuário, pode ser necessário consultar sistema(s) externo(s) ou o banco de dados.

4.1.1 Planta

A primeira parte abordada da metodologia é apresentada a seguir na Figura 10 e é referente a ações que envolvem a planta de maneira direta. Inicialmente a tarefa de instalar o(s) CLP(s) representa a parte de sua instalação física e compreende desde sua parte elétrica até sua parte hidráulica e pneumática. Sucedendo a tarefa de instalação, a parte de programação de processo(s) corresponde a tarefa de programar o CLP, utilizando sua linguagem de preferência. As linguagens de programação mais comuns são as gráficas, como *Ladder* e *Function Block Diagram (FBD)*. Por fim, é necessário definir o método de supervisão e aquisição de dados. Grandes empresas como Siemens (TIA Portal) e Schneider (AVEVA System Platform) disponibilizam *softwares* próprios, os quais compreendem tanto a parte de programação quanto de supervisão e aquisição de dados. O essencial desta etapa é possibilitar a obtenção de dados dos sensores e atuadores da planta.

Figura 10 – Início da *pool* da planta, referente a tarefas iniciais.

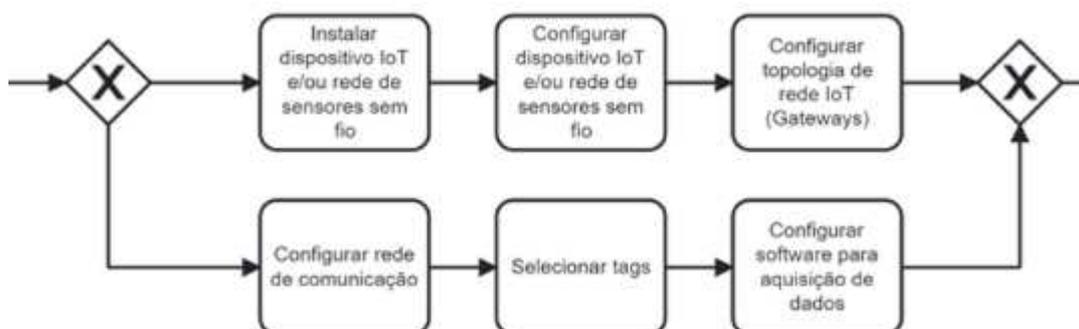


Fonte: O autor.

Apresentada na Figura 11 e posterior a instalação, programação e definição de um meio para supervisão e aquisição de dados do(s) CLP(s), há a etapa de disponibilizar acesso aos dados do processo por meios externos. É possível agir de duas maneiras com este intuito, seja ao seguir os meios tradicionais da automação ou optar por uma implementação mais moderna.

A maneira tradicional consiste no estabelecimento e configuração de uma rede industrial de comunicação. As opções mais comuns de redes cabeadas são baseadas em Fieldbus (primeiras redes de campo como Profibus e DeviceNet) ou Ethernet (permite que máquinas e dispositivos se conectem com mais precisão, como Profinet e EtherCat). Depois de conectar o(s) CLP(s) e o(s) PC(s), há a necessidade de selecionar as *tags* desejadas. Cada *tag* é equivalente a uma variável, seja ela uma informação de sensor, atuador ou resultado de uma operação. Através das *tags* é possível monitorar remotamente e supervisionar processo(s). Por fim, há a necessidade da configuração do *software* SCADA para aquisição dos dados. Essa etapa é realizada através da disponibilização das *tags*, utilizando um servidor OPC, seja ele DA (*Data*

Figura 11 – Parcela intermediária da *pool* da planta, responsável por possibilitar o acesso aos dados por meios externos.



Fonte: O autor.

Access) ou UA (*Unified Architecture*). O padrão OPC UA é mais recente e sua principal diferença é a possibilidade de comunicação entre sistemas operacionais diferentes.

Já a implementação mais moderna faz a utilização de dispositivos IoT (*Internet of Things*), os quais enviam informações de processo(s) para um destinatário de maneira sem fio. Na Figura 12 pode ser observado um exemplo de dispositivo IoT. Para utilizar estes dispositivos há a necessidade da instalação de um dispositivo IoT ou de uma rede de sensores sem fio. Após sua instalação de caráter “*plug and play*”, há a necessidade da configuração de envio dos dados. Se houver apenas um dispositivo, será um simples envio de informação, enquanto a existência de uma rede de sensores sem fio apresenta duas possibilidades: cada dispositivo pode, de maneira autônoma, enviar dados até o destino ou todos os dispositivos podem concentrar suas informações em um único dispositivo IoT, o qual poderá realizar operações sobre os dados e será responsável pelo envio das informações.

Figura 12 – Exemplo de dispositivos IoT

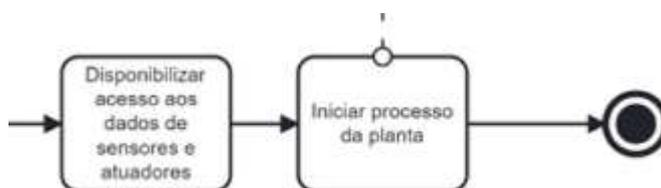


Fonte: TEREPAAC.

Para complementar uma rede sensores sem fio, pode ser necessário um estudo e configuração de *gateways*, baseados na topologia da rede IoT. Esses *gateways* podem ser necessários devido a fatores como distância ou existência de obstáculos e são responsáveis por assegurar que o destinatário receba as informações, recebendo o sinal e amplificando-o.

A etapa final que envolve a planta é apresentada na Figura 13. Após configurar o acesso aos dados do processo, é necessário disponibilizá-los. Ao utilizar o meio mais moderno, o qual conta com o emprego de dispositivos IoT, o PC destinatário pode ser selecionado e um trecho de código que utilize as variáveis é mais facilmente programado. Já a opção mais tradicional e menos custosa, que conta com o emprego de uma rede industrial cabeada e o estabelecimento de um servidor OPC, exige a configuração de um cliente OPC em um PC conectado na mesma rede. Após configurado o cliente, seja um *software* padrão como o OPC Scout da Siemens ou utilizando pacotes e serviços de um linguagem desejada, o acesso aos dados é possibilitado e o processo já pode ser iniciado.

Figura 13 – Término da *pool* da planta, parte na qual o processo é iniciado e seus dados passíveis de obtenção pelo programa desenvolvido.



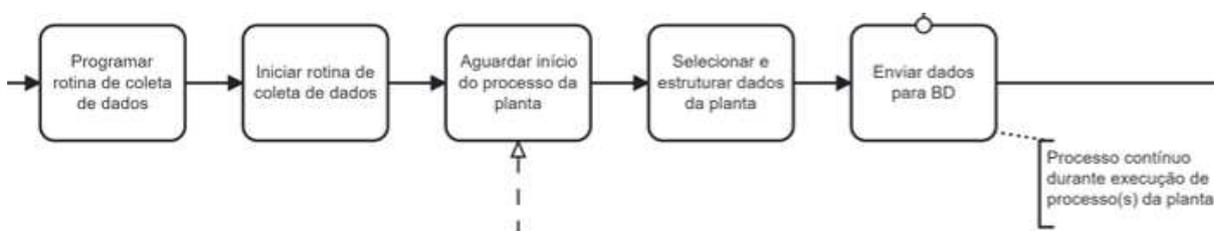
Fonte: O autor.

4.1.2 Programa para coleta de dados

É apresentada na Figura 14 a *pool* referente ao programa de coleta de dados. Após uma etapa inicial de programação da rotina de coleta de dados na linguagem de preferência, é possível coletar os dados através de pacotes específicos de leitura OPC. A linguagem de programação mais comum para esses tipos de aplicação é a *Python*, por possuir comunidade ativa e inúmeras opções de pacote. Uma vez programada a rotina para coleta de dados, é necessária sua inicialização. O programa deve aguardar o início de atividade da planta para então coletar os dados, realizar operações e finalmente estruturá-los para envio ao banco de dados selecionado. É importante salientar que este é um processo contínuo e deve funcionar durante a execução de processo(s) da planta.

Ressalta-se que, ao utilizar dispositivos IoT, pode-se haver modificações nesta etapa, visto que os próprios dispositivos podem realizar o envio de informações diretamente ao banco de dados (usualmente via HTTP).

Figura 14 – *Pool* do programa de coleta de dados.



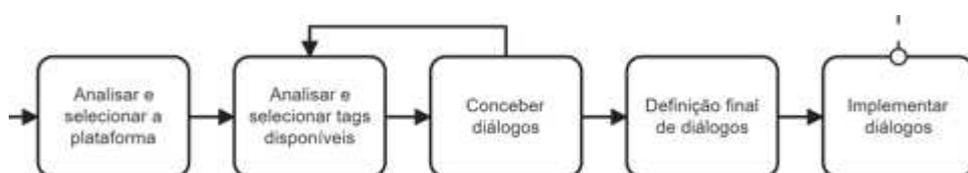
Fonte: O autor.

4.1.3 *Softbot*

Previamente à utilização do *softbot*, há a necessidade da seleção e criação de suas interações. Esse processo é realizado antes de seu funcionamento e portanto tem a denominação de tarefas *Offline*. Após a execução das tarefas que culminam na implementação de diálogos no *softbot*, há a apresentação das tarefas pertencentes a cada tipo de comportamento.

4.1.3.1 *Offline*

Figura 15 – Tarefas *offline* da *pool* do *softbot*, referentes a seleção e criação de possíveis interações



Fonte: O autor.

A Figura 15 apresenta as tarefas referentes ao processo de implementação de diálogos, o qual inicia-se com uma análise e seleção da plataforma de desenvolvimento do *softbot*. Após a observação das restrições e potencialidades da plataforma escolhida, inicia-se a etapa de análise e seleção das *tags* disponíveis para utilização. Com a seleção das *tags* feita, inicia-se o concebimento de possíveis diálogos do *bot*. É importante citar que esta etapa deve ser conduzida após a realização de conversas com usuários de vários níveis, como gestor, supervisor e operador. Finalizado o diálogo com

o usuário, pode-se haver a necessidade de reanalisar as *tags* selecionadas ou realizar a definição final de diálogos. Uma vez concluída a definição dos diálogos, inicia-se a implementação dos diálogos do *softbot*, o qual então inicia o seu funcionamento.

4.1.3.2 Online

A Figura 16 refere-se a parte inicial da *pool* do *softbot* e dita sua lógica de identificação de pergunta e avaliação da necessidade de consulta. A primeira etapa consiste na análise da pergunta do usuário, através da identificação de padrões (Capítulo 5). Uma vez determinado o padrão de entrada do usuário, o *softbot* pode determinar qual o diálogo selecionado, para que então averigüe a presença de variáveis na resposta. Caso não seja possível determinar um padrão na pergunta do usuário, um diálogo “fuga” é selecionado, o qual não possui variáveis em sua resposta e transmite o não entendimento, por parte do *softbot*, à pergunta do usuário.

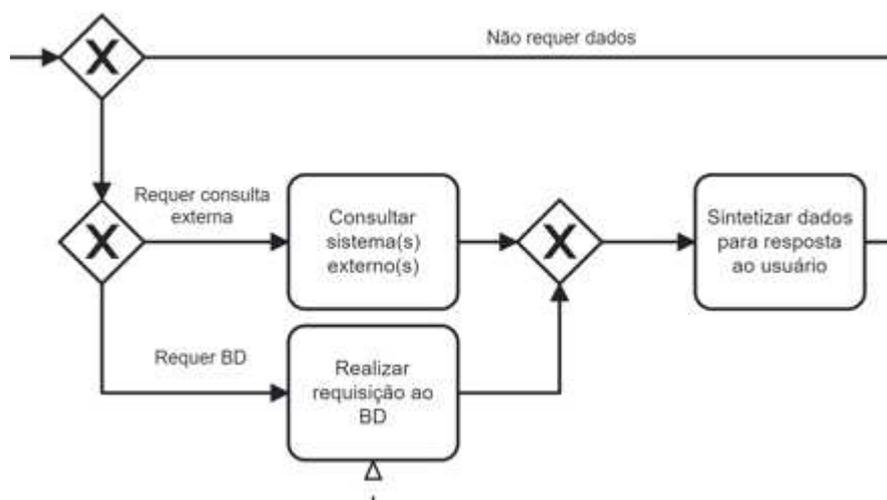
Figura 16 – Parte inicial da *pool* do *softbot*, referente a identificação da pergunta do usuário.



Fonte: O autor.

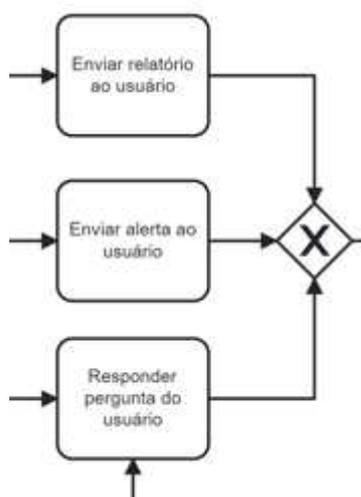
Na Figura 17 são apresentadas as possíveis tarefas para elaboração da resposta. Caso seja uma pergunta que não exige consulta ao banco dados ou a sistemas externos, o *softbot* irá utilizar uma de suas frases padrão (é possível ter variações de resposta para uma mesma pergunta) para responder o usuário. No entanto, caso a pergunta contenha variáveis que necessitam uma consulta, o *script* ou serviço web correspondente será acionado com o intuito de realizar a chamada necessária. Após coletar as informações necessárias, há a tarefa de sintetizar dados para resposta ao usuário. A tarefa de síntese envolve possíveis operações sobre os dados coletados, desde simples modificações até a execução de algoritmos, para só então formular a resposta da pergunta do usuário. É possível preparar dentro dos *scripts* tratativas para o caso de algum erro acontecer, informando ao usuário, com acesso à plataforma Arisa Nest, por meio de *logs* de erro.

Figura 17 – Parte intermediária da *pool* do *softbot*, a qual mostra cenários de consulta ao banco de dados ou sistema(s) externo(s).



Fonte: O autor.

Figura 18 – Parte final da *pool* do *softbot*, com os possíveis comportamentos do *softbot*.



Fonte: O autor.

As possíveis interações entre *softbot* e usuário estão apresentadas na Figura 18. Cada um dos comportamentos do *bot*, seja responsivo, programado ou proativo, resultam em uma interação diferente. É possível customizar e enviar um relatório ao usuário, através de uma ferramenta da plataforma Arisa Nest de agendamento configurável. Neste agendamento configurável é possível selecionar um momento específico do dia, um intervalo de tempo ou até mesmo dia e horário do mês. Além do envio de relatórios, o *softbot* conta com o comportamento proativo, através da emissão de

alertas ao usuário. Ao rodar um *script* de maneira constante, é possível perceber alterações em uma variável crítica ou até mesmo observar possíveis problemas futuros através de tendências. Por fim, o comportamento responsivo conta com as etapas citadas anteriormente para identificação da pergunta, consulta a um banco de dados ou sistema externo e culmina no envio de uma resposta ao usuário.

É importante também citar que cada uma destas etapas que correspondem ao envio de uma mensagem ao usuário pode ser simples ou apresentar um funcionamento complexo, com a utilização de algoritmos ou chamadas à outros sistemas e serviços, dependendo apenas da interação a ser realizada.

4.2 ESTRUTURAÇÃO DO *SOFTBOT*

Ao utilizar a plataforma de gerenciamento Arisa Nest, não é necessário trabalhar com os dados em seu formato diretamente (AIML). Em vez disso, pode-se utilizar a interface web de fácil interação e bem intuitiva, disponível na plataforma. Nos próximos capítulos há detalhes de implementação, bem como representações gráficas da Arisa Nest. Para configuração de um *bot* na plataforma é necessário definir os diálogos a serem suportados e ações associadas, os contextos semânticos das palavras a serem usadas ou permitidas, os termos (“padrões”) a serem usados, e as respostas a serem dadas (ZAMBIASI, S. P., 2017).

Contexto é um grupo de diálogos de conversação baseados em um assunto específico. Esse recurso serve para que um *chatbot* saiba responder conforme o assunto que está conversando.

Enquanto diálogos, por sua vez, servem para identificar se a fala do usuário se encaixa em algum cenário para poder fornecer uma resposta condizente. Caso mais de um diálogo seja encontrado como correspondente à fala do usuário, será escolhido um aleatoriamente. Por outro lado, se nenhum diálogo que seja equivalente a interação do usuário for encontrado, um diálogo padrão será selecionado. Tal diálogo é chamado de “diálogo de fuga” e serve apenas para informar o não entendimento da fala do usuário.

Já padrões são definidos como possíveis trechos ou frases completas de uma mensagem enviada pelo usuário. É importante ressaltar que diálogos podem ter mais de um padrão.

Por fim, respostas são as mensagens que serão enviadas ao usuário, uma vez selecionado o diálogo correspondente.

4.2.1 Comportamento

Os diálogos do *softbot* são, em sua grande maioria, condicionados a uma entrada do usuário. Sua natureza responsiva é intuitiva, porém não é seu único modo de ação. Além do comportamento responsivo, o *bot* pode apresentar outros dois tipos de

interação:

- Programado: é um tipo de comportamento que consiste em selecionar momentos para que mensagens sejam selecionadas e enviadas ao usuário. Tais momentos podem ser intervalos de minutos, dias ou até mesmo datas específicas. Este comportamento é extremamente útil nos casos de geração de relatórios e averiguação de conformidade em pontos do processo.
- Proativo: comportamento que permite a realização autônoma de atividades por parte do *bot*, independente do tempo. Ao monitorar variáveis específicas, pode-se emitir alertas para casos de alteração de variável ou até mesmo observar tendências e evitar um problema futuro.

4.3 REQUISITOS

Os requisitos do projeto foram selecionados com o intuito de enfatizar as possibilidades do *softbot* quando integrado a uma planta, além de explorar ao máximo a plataforma Arisa Nest.

4.3.1 Requisitos funcionais

Os requisitos funcionais descrevem os serviços que um sistema deve ter, como um sistema deve reagir a entradas específicas e como um sistema deve se comportar em situações particulares (SOMMERVILLE, 2011). Para o projeto foram definidos três RF, os quais são:

- O *softbot* deve interagir de maneira responsiva, programada e proativa.
- O *softbot* deve possuir no mínimo três interações por modo de atividade (responsiva, programa e proativa).
- O *softbot* deve dialogar de forma similar a um ser humano.

4.3.2 Requisitos não funcionais

Os requisitos não funcionais são aqueles que não dizem respeito diretamente às funcionalidades fornecidas pelo sistema. Podem estar relacionados a atributos de qualidade do produto ou restrições nos serviços, nas funções oferecidas pelo sistema e no processo de desenvolvimento (PAULA FILHO, 2000) (SOMMERVILLE, 2011). Foram definidos seis RNF, que são:

- Desenvolver uma metodologia para desenvolvimento e implementação de *softbot* a uma planta, utilizando BPMN.

- Criar uma base de conhecimento e implementar na plataforma Arisa Nest.
- Utilizar a linguagem de programação Lua para desenvolvimento de scripts.
- Utilizar a linguagem de programação Python para desenvolvimento de rotina de população do BD.
- Extrair dados de tags do processo utilizando o padrão de comunicação OPC DA.
- Utilizar requisições REST para realizar consultas em um BD NoSQL (MongoDB).

4.4 CASOS DE USO

Neste projeto, os casos de uso do *softbot* possuem padrões de funcionamento extremamente semelhantes. Há uma comunicação apenas com o banco de dados e toda a parte lógica é feita através de *scripts*, portanto cada um dos casos possui relação com todos os RF e RNF previamente citados. No entanto, pode-se categorizá-los baseado em seus tipos de comportamento, resultando nos seguintes casos de uso sintetizados:

- Responsivo
 - Solicitar informações do dia ou de um dado processo.
 - Averiguar se há um processo em andamento e suas informações.
- Programado
 - Emissão de relatório diário em horário específico.
 - Verificação de status de conexão com BD ou realização de processo(s), em um dado intervalo de tempo.
- Proativo
 - Emissão de alerta, seja referente a *buffer* vazio, cheio ou peça desaparecida.

4.4.1 Solicitar informações do dia ou de um dado processo

O usuário solicita via mensagem as informações gerais do dia ou de um dado processo, o qual exige o consumo de *scripts* que comunicam com o BD.

4.4.2 Averiguar se há um processo em andamento e suas informações

Ao informar o *softbot* que deseja averiguar se há algum processo em andamento via mensagem, um *script* que comunica com o banco de dados é necessário.

4.4.3 Emissão de relatório diário em horário específico

Em um momento programado pelo usuário, o *softbot* informa a quantidade de processos realizados no dia e dados gerais de produção.

4.4.4 Verificação de status de conexão com BD ou realização de processo(s), em um dado intervalo de tempo

Em um dado intervalo de tempo, o *softbot* verifica o status de conexão com o BD e se há processo(s) em andamento, o que exige consumo de *scripts* e comunicação com banco de dados.

4.4.5 Emissão de alertas

O *softbot* envia uma mensagem ao usuário de maneira proativa, ao constatar via *script* a emissão de alertas. Estes alertas abrangem três etapas da produção (distribuição, separação e classificação), além de variar entre *buffer* vazio, cheio ou peça desaparecida.

5 FERRAMENTAS

5.1 ARISA NEST

Para o desenvolvimento deste projeto foi utilizada a plataforma de desenvolvimento e gerenciamento de *softbots* Arisa Nest. A Arisa Nest é uma ferramenta acadêmica baseada em *PaaS* que permite a derivação de instâncias individuais ou grupos de *softbots* orientados à serviços, os quais podem ser acessados via Web ou telefones celulares. A plataforma suporta a comunicação entre os usuários e os *softbots* derivados de diferentes formas e protocolos. Essa seção visa introduzir as diferentes funcionalidades presentes na plataforma.

5.1.1 Criação e configuração

A etapa inicial consiste na criação e configuração de um novo *bot* na plataforma. Para ser capaz de realizar sua criação, é necessário que o usuário seja do tipo criador. O único campo imutável do novo *bot* é seu ID, o qual é responsável por sua identificação na plataforma. Campos mutáveis incluem seu nome, descrição, site e imagem, os quais são responsáveis por formarem a personalidade de nosso assistente inteligente.

Figura 19 – Parte de criação e configuração do *bot*.



ID do Assistente Virtual:

marin

Nome do Assistente Virtual:

Marin

Descrição:

Sou Marin, estou em desenvolvimento para o PFC de André Luiz Nunes Amaru. Sou capaz de interagir com a planta MPS no laboratório LAI (DAS - UFSC).

Site:

Imagem do Assistente Virtual: Escolher arquivo Nenhum arquivo escolhido

Público Scripts Proativo

Fonte: O autor.

5.1.2 Contextos

Por ser orientado a contextos, sua criação é imprescindível. Apenas após sua criação é que são possíveis as criações dos diálogos do *bot*. Na Figura 20 é possível visualizar a existência de três contextos, onde cada um deles representa diálogos referentes a um tipo de comportamento (responsivo, programado e proativo). O contexto genérico “**main**” é obrigatório e é responsável por possibilitar o acesso aos outros contextos.

Figura 20 – Contextos existentes na Marin.



Fonte: O autor.

5.1.3 Diálogos

Dentro de cada contexto existem os diálogos, os quais são responsáveis por identificar um determinado cenário, a partir da mensagem do usuário, e retornar a resposta correspondente.

Ao criar-se um diálogo, é possível selecionar o contexto ao qual pertence através do campo “Contexto”. No campo “Link” é possível incluir um *link* que será enviado junto com a resposta ao usuário. Já o campo “Ir para” indica, após enviar uma mensagem ao usuário, para qual contexto o usuário deve ser encaminhado. É também possível incluir uma imagem, ao acessar o campo “Imagem”.

Por fim, há também a possibilidade de gerar gráficos e *dashboards* com a plataforma, como pode ser observado em (ABNER *et al.*, 2020). No entanto, essa funcionalidade não foi utilizada neste projeto. A Figura 21 exemplifica a interface de configuração de diálogos.

Figura 21 – Campos para configuração de um diálogo.

Fonte: O autor.

5.1.4 Padrões

O campo “Padrões” conta com os possíveis formatos de mensagens enviadas pelo usuário. O diálogo será selecionado apenas se a mensagem do usuário corresponder a algum dos padrões estabelecidos. Já no campo “Respostas” são inseridas a única ou múltiplas possíveis respostas do diálogo.

Para separar tanto os padrões quanto as respostas é utilizado o símbolo “;”. Já o símbolo “%” representa um caractere coringa, que pode estar entre palavras, no final e/ou no início. É importante citar também que no estabelecimento de padrões não deve-se utilizar nenhum tipo de acentuação. Na Figura 22 é possível visualizar quatro possíveis padrões e apenas uma única resposta.

Figura 22 – Exemplos de aplicação de padrões.

Fonte: O autor.

5.1.5 Respostas

As respostas são configuradas dentro de diferentes diálogos e nem sempre estão ligadas a padrões de entrada. Na Figura 23 pode ser observada a ausência de padrões de entrada, caso este que ocorre apenas na implementação de um diálogo

“fuga“. Nos demais diálogos podem ser configuradas uma ou mais respostas, as quais são separadas de forma semelhante aos padrões, através da utilização do símbolo “;“.

Figura 23 – Configuração do diálogo de fuga.



Fonte: O autor.

5.1.6 Crenças

A plataforma Arisa Nest possibilita o armazenamento de variáveis, locais ou globais, denominadas crenças. As crenças globais são acessíveis por qualquer usuário que interaja com o assistente virtual, enquanto as crenças locais são apenas acessíveis pelos usuários aos quais elas estão relacionadas.

5.1.7 Condições

Ao configurar um diálogo, percebe-se a presença do campo de condições. O campo é de utilização opcional e serve para verificar o valor de uma crença global ou local, com o intuito de condicionar a execução de um diálogo. Apenas se o resultado da expressão lógica deste campo for “verdadeiro” que o *bot* irá executar o diálogo em questão.

5.1.8 Scripts

Ao interagir com o *bot*, é possível que a elaboração de uma resposta exija a execução de trechos de códigos denominados *scripts*. Esses *scripts* podem ser programados tanto em linguagem Lua quanto Python e podem fazer atividades como simples cálculos ou até executar algoritmos mais complexos.

A Figura 24 exemplifica a situação, uma vez que o usuário solicita informações específicas de um processo e o *softbot*, para responder ao usuário, executa o *script* denominado “info_processo“. É importante citar que ao identificar o padrão de entrada do usuário há um elemento entre chaves (“{\$1}“), o qual corresponde a uma variável que será utilizada como parâmetro na execução do *script*.

Figura 24 – Exemplo de resposta de diálogo que contém a execução de um *script*.

Fonte: O autor.

5.1.9 Comportamentos

Outra possibilidade para execução de trechos de código, sem necessitar a interação do usuário, é através de comportamentos. Comportamentos são algoritmos executados regularmente ou conforme agendamento configurável, possuem as mesmas funcionalidades dos *scripts* e são essenciais para a proatividade do *softbot*. A Figura 25 mostra a interface de configuração de um comportamento, na qual é possível determinar seu intervalo de funcionamento, bem como selecionar se e quando o funcionamento deve cessar.

Figura 25 – Interface de configuração de comportamento.

Fonte: O autor.

5.1.10 Aplicativo de mensagens

Para interagir com o *bot* desenvolvido, a plataforma Arisa Nest possui um *chat* disponível. No entanto, é possível também acessar o *softbot* por meio da utilização do aplicativo de mensagens Telegram, disponível para *download* em dispositivos *iOS* e *Android*, além de computadores com *Windows*, *Linux* e *macOS*. Na plataforma de criação e gerenciamento de assistentes virtuais há um tutorial para a integração do *bot* ao aplicativo de mensagens, o qual utiliza um sistema automático de criação de *bots* chamado “BotFather”. Sua implementação consiste em poucos passos e na utilização de um *token* de identificação.

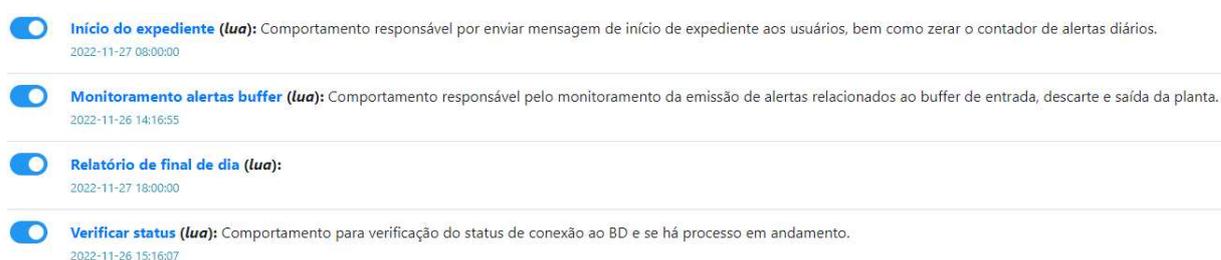
5.2 EXEMPLO

Com o intuito de facilitar o entendimento da plataforma e algumas de suas ferramentas, o caso da configuração de um comportamento que ocorre no início do dia de trabalho será apresetando, bem como a criação de um diálogo que obtém o nome do usuário.

5.2.1 Comportamento de início do dia de trabalho

Já a configuração de um novo comportamento em si não exige a criação ou utilização de um contexto, sendo apenas necessário caso opte por enviar alguma mensagem para o usuário. Na aba de Comportamentos são exibidos os comportamentos criados, bem como os ativos e seu próximo momento de atuação, como demonstra a Figura 26.

Figura 26 – Comportamentos configurados.



Fonte: O autor.

A criação de um comportamento que faça o *softbot* enviar mensagens ao usuário todo início de expediente pode ser observada na Figura 27, onde é feita sua configuração. Além de escolher seu nome, descrição, intervalo e se o comportamento deve terminar, também é possível alterar a sua linguagem de programação entre Lua e Python.

Figura 27 – Configuração do comportamento apontado.

Nome / Linguagem / Descrição	Agendamento
Início do expediente	Regularidade Dias
lua	Dia 1 do mês
Comportamento responsável por enviar mensagem de início de expediente aos usuários, bem como zerar o contador de alertas diários.	Horário 08 00
	Termina <input checked="" type="checkbox"/> Nunca

Fonte: O autor.

A Figura 28 demonstra a implementação do código, feito em linguagem Lua, responsável por zerar o contador de alertas emitidos no dia, bem como enviar uma mensagem para cada usuário que já interagiu com o *bot*, conhecido na plataforma como “amigo”.

Figura 28 – Código do comportamento apontado.

```

🌙 Início do expediente
1 setGlobal('alertas_emitidos', 0)
2
3 friends = json.decode(friendList())
4 for i = 1, #friends do
5   id = friends[i]
6   im = getLocal(id, 'im')
7   if (im == 'telegram') then
8     sendMessage(id, runDialog('alerta', 'inicio_expediente'))
9   end
10 end

```

Fonte: O autor.

5.2.2 Diálogo para obter nome do usuário

Antes de mais nada, é relevante informar que este diálogo serve para salvar crenças, variáveis locais ou globais da plataforma, e pode ser alterado para uma aplicação mais complexa.

Para realizar a implementação de um diálogo, há a necessidade da criação ou utilização de um contexto já criado. Neste caso, o contexto genérico “**main**” será utilizado, como apresentado na subseção 5.1.2. Ao selecionar o contexto, há a exibição dos diálogos já criados, bem como a opção de criar um novo. Por inicialmente não saber o nome do usuário, há a necessidade de transmitir essa informação, como

demonstrado na Figura 29. Observa-se que há a utilização de uma condicional para que o diálogo possa ser realizado, onde a crença local “**friend_name**” deve estar vazia.

Figura 29 – Configuração de diálogo sem saber nome do usuário.

The screenshot shows a configuration window for a dialog. At the top, there is a 'Descrição (2967)' field with the text 'Nome usuário (sem saber)' and a 'Contexto' dropdown set to 'main'. Below this is a 'Condições' field containing the condition 'friend_name = empty;'. The interface is divided into two main sections: 'Padrões:' and 'Respostas:'. The 'Padrões:' section contains a list of default phrases: '%como e%meu nome%;', '%como me chamo%;', '%como%me chamo%;', '%diga meu nome%;', '%diz meu nome%;', '%fala meu nome%;', and '%fale meu nome%;'. The 'Respostas:' section contains two response options: 'Não sei. Qual é?;' and 'Não sei. Qual seu nome?;'. The interface has a light gray background and rounded corners.

Fonte: O autor.

No entanto, quando a mesma pergunta for realizada mas o *softbot* já possuir a informação, através de uma nova condição é possível retornar outra resposta, como é apresentado na Figura 30.

Figura 30 – Configuração de diálogo sabendo o nome do usuário.

The screenshot shows a configuration window for a dialog. At the top, there is a 'Descrição (2968)' field with the text 'Nome usuário (sabe)' and a 'Contexto' dropdown set to 'main'. Below this is a 'Condições' field containing the condition 'friend_name != empty;'. The interface is divided into two main sections: 'Padrões:' and 'Respostas:'. The 'Padrões:' section contains a list of default phrases: '%como e%meu nome%;', '%como me chamo%;', '%como%me chamo%;', '%diga meu nome%;', '%diz meu nome%;', '%fala meu nome%;', and '%fale meu nome%;'. The 'Respostas:' section contains three response options: 'Seu nome é {friend_name};', '{friend_name};', and '{friend_name}, feliz agora?;'. The interface has a light gray background and rounded corners.

Fonte: O autor.

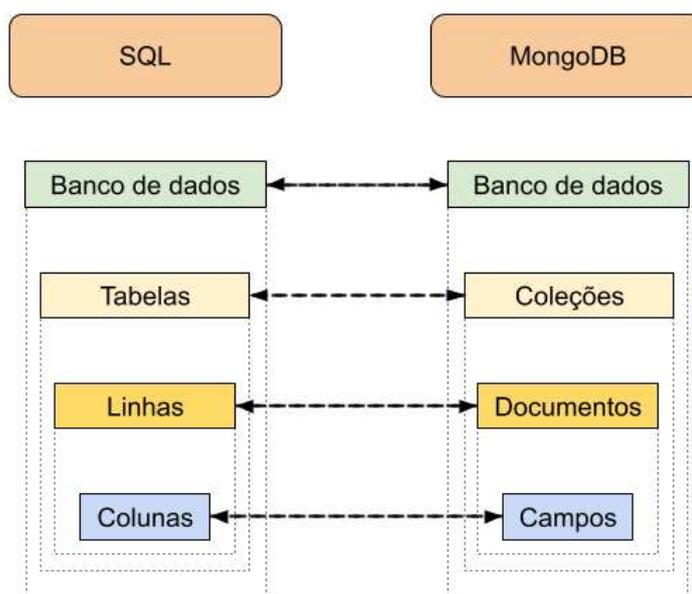
5.3 MONGODB ATLAS

O MongoDB Atlas é um serviço da MongoDB que oferece a hospedagem de dados em nuvem, bem como facilidades para seu acesso e manuseio de informação. Para este projeto foi criado um banco de dados NoSQL em nuvem, o qual conta com diferentes *collections* referentes a cada uma das etapas da planta. Além de disponibilizar a hospedagem dos dados, o serviço também oferece APIs próprias para executar operações sobre os dados, acessíveis tanto pela rotina de coleta de dados quanto pelo *softbot*.

A estrutura de armazenamento do MongoDB é similar à apresentada no SQL, o que resulta em uma equivalência de termos, como apresenta a Figura 31. O banco de dados MongoDB do projeto conta com seis coleções, onde cada uma delas representa uma etapa de processos da planta, além de uma coleção extra utilizada como registro de entradas. Cada documento representa uma entrada de dados da operação e seus campos informam, além das leituras das *tags*, o momento de coleta dos dados através de um *timestamp* e o ID do registro de entrada referente. É através dessa conexão com o registro de entrada que é possível distinguir quais documentos fazem parte do mesmo processo.

Ressalta-se que a utilização de um banco de dados não relacional e hospedagem em nuvem foi a opção de desenvolvimento melhor adequada para a situação do projeto, o que não exclui a possibilidade de realizar a hospedagem local dos dados, bem como empregar um BD relacional.

Figura 31 – Diagrama da equivalência de termos entre SQL e MongoDB.



Fonte: O autor.

6 IMPLEMENTAÇÃO

Este capítulo conta com a descrição da forma de implementação dos elementos do projeto. Suas seções apresentam os detalhes da aplicação da metodologia desenvolvida, com informações sobre cada uma das partes que a compõe, além de explicitar o funcionamento de alguns cenários implementados.

6.1 METODOLOGIA APLICADA

Para o início do desenvolvimento da aplicação, segundo a metodologia proposta, as tarefas iniciais concentraram-se nos elementos que envolvem a planta. Somente após sua finalização, o desenvolvimento do programa para coleta de dados foi iniciado. De caráter simples, o código conta com poucas operações e foi a etapa mais rápida da implementação. Por fim, o *softbot* foi criado e seus contextos, diálogos e padrões programados na plataforma Arisa Nest.

6.1.1 Planta MPS do LAI

A planta utilizada no projeto foi a planta MPS localizada no LAI e conta com seis bancadas, cada uma com um CLP e diversos sensores e atuadores. A Figura 32 apresenta a estrutura utilizada, a qual corresponde a planta de montagem didática da FESTO e CLPs da marca Siemens.

Figura 32 – Planta didática MPS do LAI.



Fonte: RABELO; MAGALHÃES; CABRAL (2020)

Como a planta já está previamente instalada no laboratório, a tarefa inicial de “Instalar CLP” não precisou ser realizada. Para a tarefa de “Programar CLP” foi utilizado o programa da linha de montagem empregado no ensino, o qual utiliza a linguagem Ladder para configuração do processo. Por fim, para supervisionar e adquirir os dados

foi utilizado o *software* próprio da Siemens, o TIA Portal. Essa ferramenta possibilita configuração da planta, visualização dos dados dos sensores e atuadores, além do monitoramento e programação de alertas.

Para possibilitar a comunicação entre CLPs e computadores do laboratório, há uma rede Ethernet estabelecida, a qual usa o protocolo de rede industrial Profinet. Através desta rede é possível realizar a leitura simultânea e em tempo real de todos os dados de processo(s) da planta, utilizando a ferramenta previamente citada TIA Portal. Uma das funcionalidades do portal é a possibilidade do estabelecimento de *tags*, as quais podem ser leituras simples de sensores e atuadores ou até o resultado de operações sobre os dados. O processo em questão possui cerca de 400 *tags*, das quais pouco mais de cem representam leituras simples de sensores e atuadores. Neste trabalho foram utilizadas as aproximadamente cem *tags*, uma vez que são suficientes na representação do processo. A Figura 33 apresenta exemplos de *tags* dentro do TIA Portal, as quais envolvem a primeira parte do processo, a etapa de distribuição.

Figura 33 – Exemplo de *tags* dentro do TIA Portal.

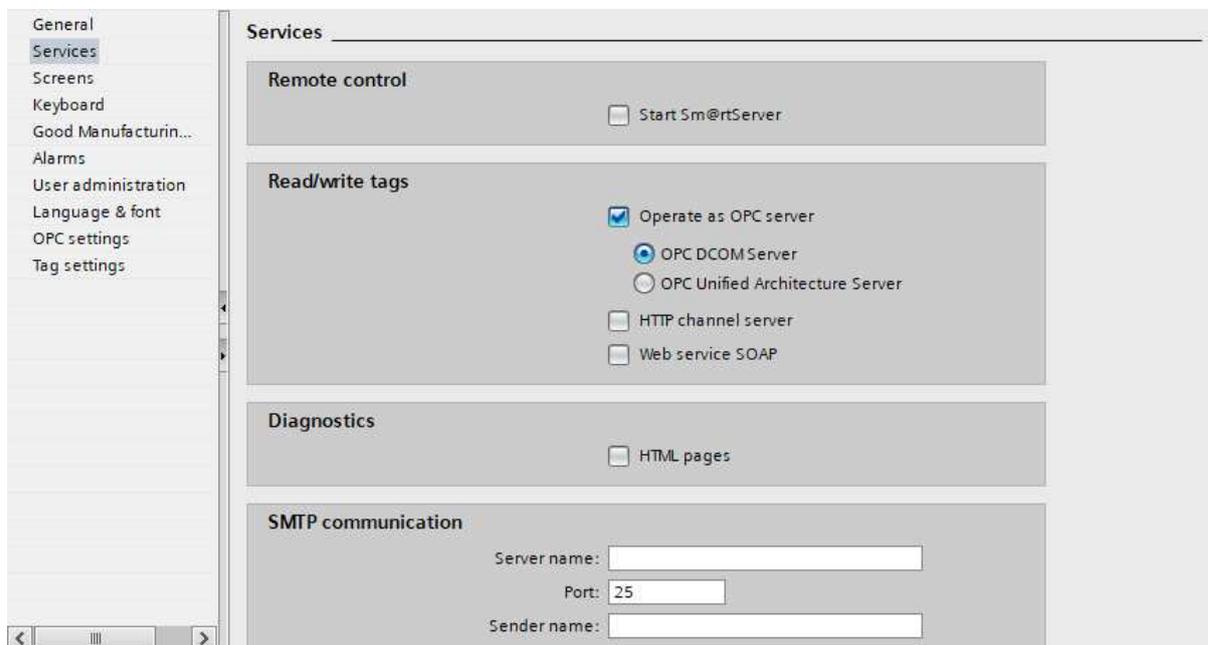
HMI tags						
Name	Tag table	Data type	Connection	PLC name	PLC tag	
CounterUP	Default tag table	Dint	HMI_Connectio...	PLC_1__Distributing	CounterUP	
Distributing.Alarme_sem_dem...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	Alarme_sem_demanda	
Distributing.Alarme_sem_peca	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	Alarme_Sem_peca	
Distributing.GV_B_Reset_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_B_Reset_1	
Distributing.GV_B_Start_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_B_Start_1	
Distributing.GV_B_Stop_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_B_Stop_1	
Distributing.GV_braco_avanca	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	Tag_5	
Distributing.GV_braco_recu	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	Tag_4	
Distributing.GV_Empurra_Peca_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Empurra_Peca_1	
Distributing.GV_Estacao_Segui...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Estacao_Seguinte_Li...	
Distributing.GV_LED_Reset_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_LED_Reset_1	
Distributing.GV_LED_Start_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_LED_Start_1	
Distributing.GV_Liga_Sopro_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Liga_Sopro_1	
Distributing.GV_Liga_Vacuo_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Liga_Vacuo_1	
Distributing.GV_LUZ_Q5_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_LUZ_Q5_1	
Distributing.GV_LUZ_Q6_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_LUZ_Q6_1	
Distributing.GV_Magazine_Ava...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Magazine_Avancado...	
Distributing.GV_Magazine_Rec...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Magazine_Recuado_1	
Distributing.GV_Manual_Mode_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Manual_Mode_1	
Distributing.GV_Peca_1	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Peca_1	
Distributing.GV_Rotativo_Pega...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Rotativo_Pega_Peca_1	
Distributing.GV_Rotativo_Solta...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Rotativo_Solta_Peca_1	
Distributing.GV_Running_Reset...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Running_Reset_1	
Distributing.GV_Running_Syste...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Running_System_1	
Distributing.GV_Sensor_Braco...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Sensor_Braco_Pega_...	
Distributing.GV_Sensor Braco ...	Default tag table	Bool	HMI_Connectio...	PLC_1__Distributing	GV_Sensor Braco Solta ...	

Fonte: O autor.

Após selecionar as *tags* desejadas do processo, é necessário disponibilizar seu acesso pelo computador. O TIA Portal possibilita a criação de uma Interface Homem-Máquina (IHM), a qual fornece um meio de interação remoto com a planta. No entanto, outra funcionalidade sua é a possibilidade de agir como um servidor OPC, meio pelo qual é possível interagir com as *tags* da planta utilizando trechos de código. A Figura 34 apresenta a interface de configuração da IHM como servidor OPC. Após a configuração apresentada, tudo está pronto para o início dos processos e disponibilização dos dados

da planta para coleta.

Figura 34 – Interface de configuração da IHM como servidor OPC.



Fonte: O autor.

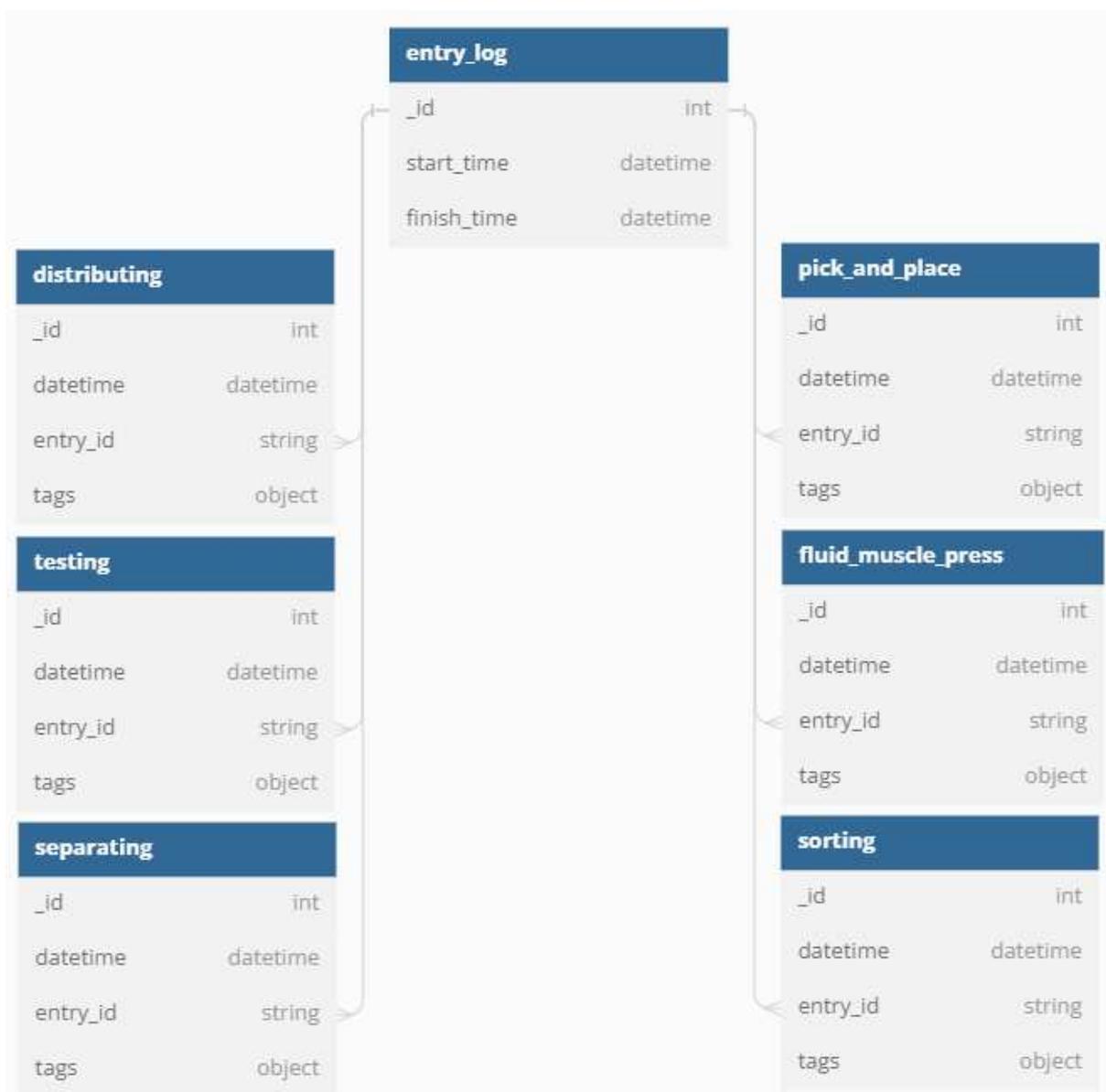
6.1.2 Programa para coleta de dados

A etapa inicial no desenvolvimento de um programa para coleta de dados é a sua programação. A linguagem Python foi escolhida por possuir ampla e ativa comunidade, a qual oferece inúmeros pacotes que auxiliam o desenvolvimento em diversas áreas.

Para tornar o computador um cliente OPC, foi utilizado o pacote “OpenOPC-Python3x”. Ele utiliza um serviço chamado *OpenOPC Gateway Service* que, após selecionado o *host* desejado, possibilita a leitura, manuseio e envio de dados por trechos de código. Já o pacote “PyMongo” foi utilizado para interagir com banco de dados MongoDB. No Apêndice B apresenta-se o código desenvolvido para coleta de dados, que envolve a utilização dos pacotes citados.

Como para um dado processo há informações de seis etapas, há um total de sete *collections* no banco de dados. Dentro da *collection* dos processos são armazenadas as datas e horários de início e de término, enquanto que a *collection* de cada etapa armazena seus dados e também o ID do processo ao qual pertence. Enfatiza-se que cada etapa conta com um número variável de *tags*, as quais possuem seus nomes e valores armazenados em um objeto. A Figura 35 apresenta a arquitetura utilizada para o banco de dados.

Figura 35 – Arquitetura do banco de dados.



Fonte: O autor.

O período de atualização dos dados das *tags* é variável, por parte do servidor OPC estabelecido, e seu valor mínimo é de um milissegundo. No entanto, utilizou-se o tempo de um segundo como período para leitura de dados dentro do programa. Esse valor foi determinado ao levar em conta a dinâmica do processo, o tempo de execução do código, das chamadas para o banco de dados e também de leitura das *tags*. O emprego dessa taxa de atualização contemplou de forma satisfatória a leitura de dados, sem comprometer o entendimento do processo ou ocasionar erros no armazenamento da informação.

A Figura 36 apresenta os dados de início e término de um determinado processo,

os quais são salvos em uma *collection* própria, denominada “**entry_log**”. Sua função principal é servir como meio para distinguir quais dados de cada etapa pertencem ao mesmo processo. Enquanto isso, a Figura 37 mostra uma entrada de dados da etapa do *Fluidic Muscle Press*. Essa etapa é responsável por pressionar inserções da peça de trabalho em sua cavidade, através de um músculo pneumático e um atuador rotativo. As *tags* da etapa são utilizadas para identificar o estado do processo, sendo suficientes para determinar sua posição e atividade.

Figura 36 – Dados de um processo.

```
_id: ObjectId('63862fe3740d50e45db13845')
start_time: 2022-11-29T13:14:27.161+00:00
finish_time: 2022-11-29T13:16:12.101+00:00
```

Fonte: O autor.

Figura 37 – Entrada de dados da estação de "Classificação".

```
_id: ObjectId('6386304b740d50e45db13a12')
datetime: 2022-11-29T13:16:11.009+00:00
entry_id: "63862fe3740d50e45db13845"
tags: Object
  GV_B_Reset_5: false
  GV_B_Start_5: false
  GV_B_Stop_5: false
  GV_Braco_0_5: true
  GV_Braco_180_5: false
  GV_Braco_90_5: false
  GV_Detecta_peca_5: false
  GV_Esta_0_5: true
  GV_Esta_180_5: false
  GV_Esta_90_5: false
  GV_Esta_direita_5: false
  GV_Esta_esquerda_5: true
  GV_Esta_prensando_5: false
  GV_Garra_5: false
  GV_LUZ_Q1_5: false
  GV_LUZ_Q2_5: true
  GV_LUZ_Q5_5: true
  GV_LUZ_Q6_5: true
  GV_LUZ_Reset_5: false
  GV_LUZ_Start_5: false
  GV_Libera_Entrada_5: false
  GV_Mover_Direita_5: false
  GV_Mover_Esquerda_5: true
  GV_Prensa_5: false
  GV_Saida_Autorizada_5: false
  botao_reset_remoto: false
  botao_start_remoto: false
  botao_stop_remoto: false
  luz_Q1: false
  luz_Q2: true
  luz_reset: false
  luz_start: false
```

Fonte: O autor.

6.2 FUNCIONALIDADES

A configuração do *softbot* foi a etapa final de desenvolvimento, onde seus elementos apresentados no capítulo 5 foram criados e seu funcionamento iniciado. O *softbot* possui três cenários de funcionamentos, referentes a cada um de seus tipos de comportamento. Os cenários programado e proativo são de funcionamento simples e são responsáveis pela interação com o usuário de forma independente, enquanto o cenário responsivo possui uma lógica de funcionamento.

6.2.1 Comportamento Responsivo

Este comportamento é acionado com o recebimento de uma mensagem por parte do usuário e sua etapa inicial consiste na análise da pergunta do usuário. O *softbot* analisa qual usuário está interagindo, bem como o contexto que ele se encontra. Após analisar a pergunta do usuário, o *bot* identifica o diálogo da pergunta, dentro do contexto atual.

Os padrões e respostas de um diálogo podem conter variáveis, o que implica na necessidade ou não da atuação de um *script*. Caso o diálogo não contenha variáveis, o *softbot* apenas retorna uma resposta ao usuário e o fluxo é encerrado. Porém caso exija a execução de um *script*, é necessário avaliar a necessidade de consulta a sistema(s) externo(s) ou ao banco de dados da planta. É importante citar que caso ocorra algum erro na consulta de informações, tratativas podem ser implementadas no código e o desenvolvedor pode conferir *logs* do erro na plataforma.

Após a coleta de informações inicia-se a etapa de síntese dos dados para resposta ao usuário. Antes da devolução dos dados coletados ao usuário, pode ser necessária a realização de operações simples ou até a execução de algoritmos mais complexos.

A seguir, alguns exemplos de funcionalidades presentes no comportamento responsivo são apresentadas.

6.2.1.1 Solicitar informações até o momento

Uma funcionalidade essencial para verificar as informações do dia até o momento é “Solicitar informações do dia”. Acessível a qualquer período do dia, a função apresenta dados totais até o presente momento, como número de peças produzidas, descartadas e quantidade de processos realizados.

Ao questionar a quantidade de peças produzidas até o momento, o *bot* realiza a execução de *scripts*, tanto para saber quantos processos foram realizados até o momento, bem como para informar a quantidade de peças produzidas e descartadas. Enquanto a Figura 38 apresenta a implementação referente a esta funcionalidade,

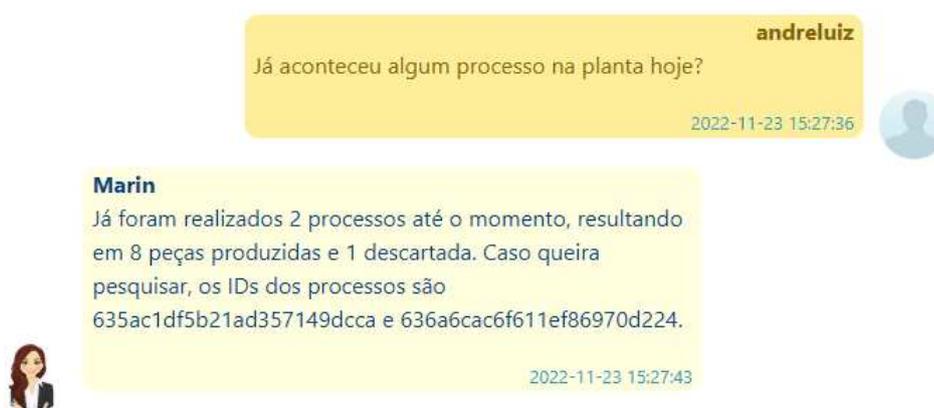
demonstrando os padrões de mensagem e resposta, a Figura 39 exemplifica uma conversa em que o usuário solicita as informações da planta até o momento.

Figura 38 – Implementação de diálogo para solicitar informações até o momento.



Fonte: O autor.

Figura 39 – Conversa sobre informações até o momento.



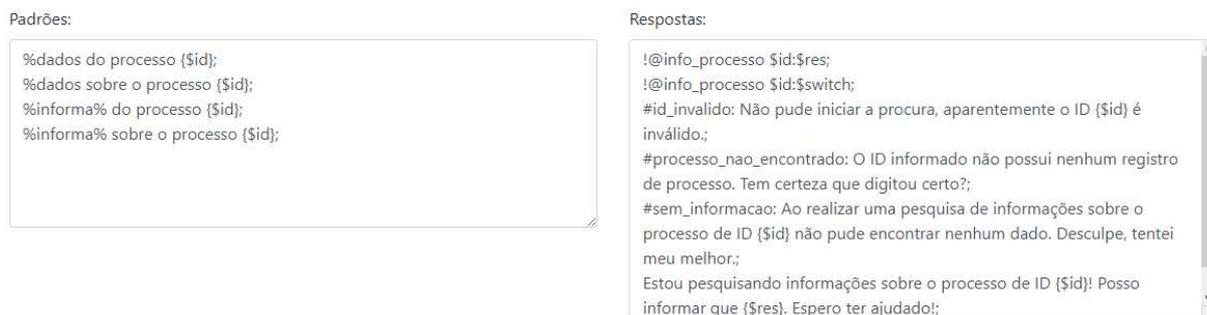
Fonte: O autor.

6.2.1.2 Solicitar informações específicas de um processo

Na Figura 40 apresenta-se a implementação dos padrões de entrada, bem como a possível resposta quando o usuário solicitar informações específicas de um processo. Ao informar o número de identificação (ID) do processo sobre o qual deseja-se informações, informações como número de peças produzidas, descartadas, tempo de execução e tempo médio por peça são apresentadas.

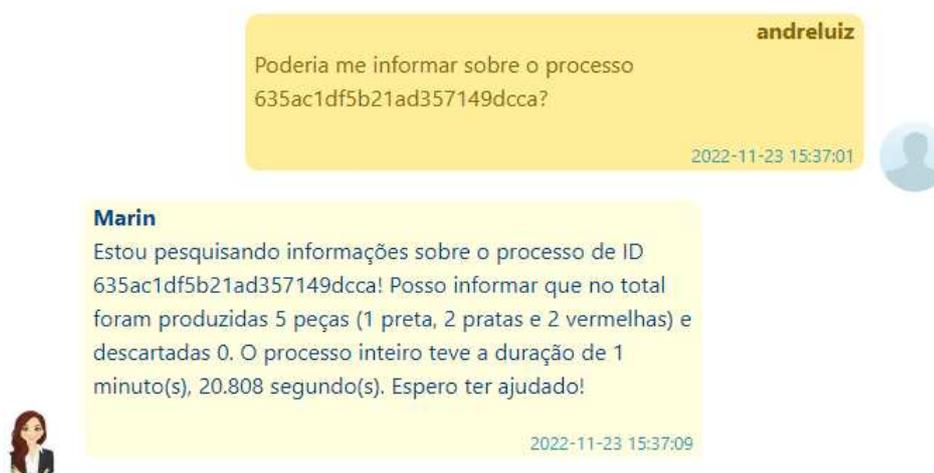
A Figura 41 demonstra uma conversa em que o usuário informa o ID de um processo específico para receber suas informações. Dados mais específicos do projeto são relatados, o que inclui seu tempo de duração.

Figura 40 – Diálogo de informações sobre um processo específico.



Fonte: O autor.

Figura 41 – Histórico de conversa sobre um processo específico.

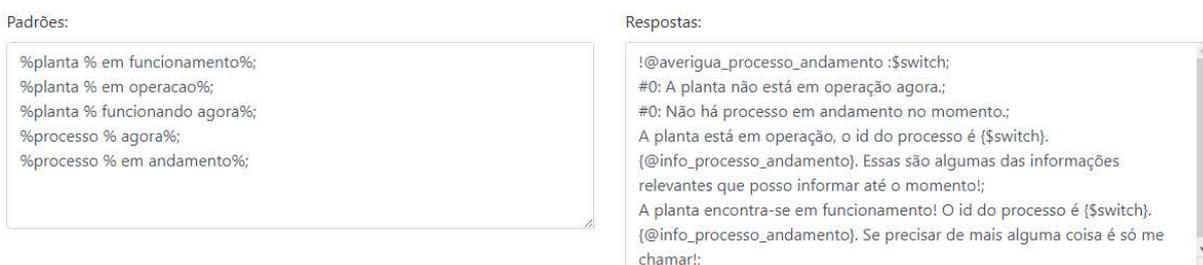


Fonte: O autor.

6.2.1.3 Averiguar se há um processo em andamento e suas informações

É possível interagir com o *softbot* para consultar a existência de um processo em andamento e suas informações. Na Figura 42 são apresentados os conjuntos de possíveis padrões de entrada e variações de respostas. Ao utilizar uma variável condicional da plataforma, denominado através da nomenclatura *\$switch*, é possível explicitar frases que ocorrerão em diferentes cenários de retorno da variável. Neste caso, os cenários foram implementados de maneira que o valor zero indica um cenário negativo na averiguação do funcionamento atual da planta, enquanto as mensagens informativas ocorrem apenas em valores diferentes da variável. Um *script* busca por registros de processo que ainda estão em andamento (não possuem data de término) e caso encontre algum, retorna informações básicas sobre a produção e descarte de peças, bem como sua data de início e duração até o momento.

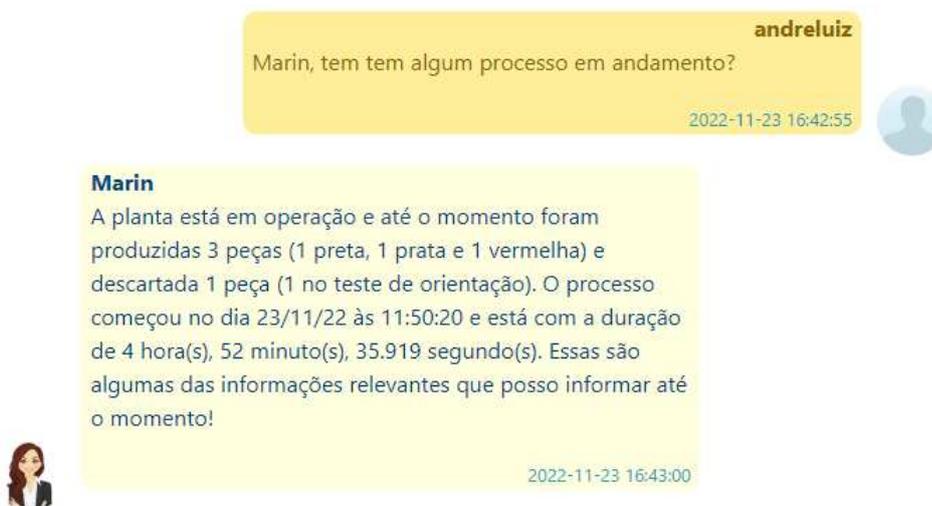
Figura 42 – Conjunto de padrões e respostas para a funcionalidade de averiguação de processo em andamento.



Fonte: O autor.

A Figura 43 exemplifica o diálogo sobre a verificação de um processo em andamento e suas informações.

Figura 43 – Diálogo com usuário onde há processo em andamento.



Fonte: O autor.

6.2.2 Comportamento Programado

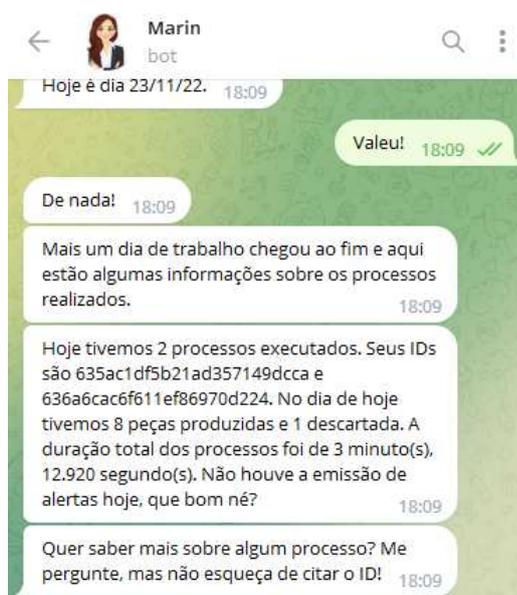
As atividades realizadas em um intervalo de tempo ou momento específico compõem o comportamento programado do *softbot*. A plataforma Arisa Nest oferece diversas ferramentas e, entre elas, a ferramenta de comportamento. Ao configurar tal ferramenta, é possível a ação do *softbot* de maneira independente, seja em um intervalo de minutos, dias, semanas e meses ou até a execução anual da atividade.

Ao atingir o período selecionado, um trecho de código, similar a um *script*, é acionado e suas operações são iniciadas. Seja para a atividade de emissão de relatório diário em horário específico, para a constatação da realização de algum processo em

um dado intervalo de tempo ou na verificação do status de conexão com o banco de dados, o processo é realizado de maneira semelhante.

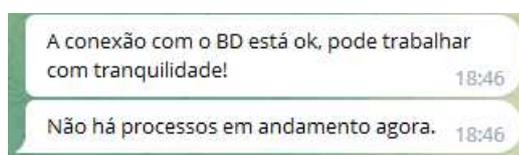
Exemplos de aplicação são apresentados na Figura 44 e na Figura 45. Enquanto a primeira demonstra que no final do dia um relatório de operações é elaborado pelo *softbot* e enviado ao usuário, a outra informa uma normalidade no status da conexão ao banco de dados, além da ausência de atividades no momento.

Figura 44 – Notificação de relatório diário de operações.



Fonte: O autor.

Figura 45 – Verificações que ocorrem a cada intervalo de tempo.



Fonte: O autor.

Já na Figura 46 apresenta-se um trecho do código do comportamento referente a emissão de relatório diário em horário específico. É possível perceber a presença de uma função genérica da plataforma chamada "*friendList*", a qual é responsável por retornar os IDs de cada usuário (amigo) que já interagiu com o *softbot*. Uma vez com o ID de cada amigo é possível saber por qual meio ele interage com o *bot*, através da utilização de crenças. A função "*getLocal*" busca a crença referente ao *Instant Messaging* (IM) do usuário selecionado, que representa o meio de interação com o

bot, seja pela própria plataforma Arisa Nest ou Telegram. Com o intuito de gerar uma notificação ao usuário, mensagens independentes são enviadas apenas ao *app* de comunicação. Por fim são executados diálogos padrões, bem como uma mensagem personalizada que contém os dados do relatório referentes ao comportamento.

Figura 46 – Trecho responsável pelo envio de mensagens aos usuários.

```
117 friends = json.decode(friendList())
118 for i = 1, #friends do
119     id = friends[i]
120     im = getLocal(id, 'im')
121     if (im == 'telegram') then
122         sendMessage(id, runDialog('relatório', 'hora do relatório diário'))
123         sendMessage(id, final_message)
124         if (qnt_logs ~= "0") then
125             sendMessage(id, runDialog('relatório', 'informações extras'))
126         end
127     end
128 end
```

Fonte: O autor.

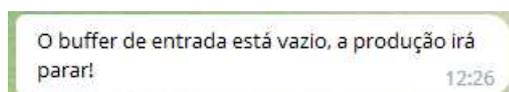
6.2.3 Comportamento Proativo

Outra maneira de interação, de forma totalmente independente por parte do *softbot*, são as tarefas relacionadas ao comportamento proativo. A supervisão do funcionamento de uma planta é uma atividade extremamente importante, visto que problemas podem ocorrer a qualquer instante e resultar em perdas significativas na produção. Com o intuito de perceber a ocorrência de algum erro, é comum a configuração de diversos alertas ao longo de uma linha de produção, seja para informar falta de matéria-prima ou identificar um funcionamento defeituoso.

Na planta MPS, utilizada no ensino e formação de novos engenheiros, alguns alertas foram implementados, como para indicar se algum *buffer* está vazio ou cheio, além de informar o desaparecimento de uma peça na linha de produção. Cada um desses alertas implica na pausa das atividades da etapa referente, seja a ausência de peças na entrada, a superlotação de peças no *buffer* de descarte ou o fato de uma peça sumir da linha de produção na etapa final de classificação das peças.

A Figura 47 apresenta a ação do *softbot* ao constatar a emissão do alerta da primeira etapa. A distribuição é responsável por posicionar a matéria-prima na linha de produção e quando seu estoque acaba, a etapa cessa seu funcionamento e permanece em estado de espera.

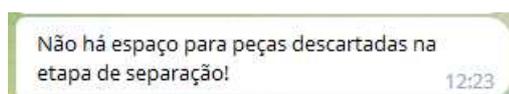
Figura 47 – Exemplo de alerta da etapa de distribuição.



Fonte: O autor.

Já a Figura 48 exemplifica a emissão de uma mensagem por parte do *bot* referente a etapa de separação, a qual busca informar o usuário que a capacidade máxima do *buffer* de descarte foi alcançada. Ao receber uma nova peça defeituosa que exija descarte, a planta permanece em estado de espera até que as peças descartadas sejam removidas e haja espaço para encaminhar novas peças defeituosas. É relevante citar que, para o processo, uma peça defeituosa representa uma peça com orientação errada, a qual é identificada através de um sensor na estação de "Separação".

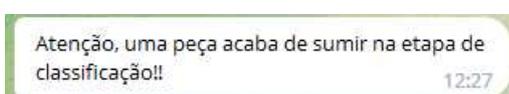
Figura 48 – Exemplo de alerta da etapa de separação.



Fonte: O autor.

Por fim a Figura 49 informa o usuário o desaparecimento de uma peça na etapa de classificação, a última seção da linha de produção. Tal alerta não impede o funcionamento da etapa e visa apenas informar o responsável sobre o acontecido.

Figura 49 – Exemplo de alerta da etapa de classificação.



Fonte: O autor.

Cada um desses alertas exige a intervenção humana, o que faz deste um caso para aplicação do comportamento proativo do *softbot*. Ao constantemente monitorar dados do último processo (esteja ele em andamento ou não), é possível emitir uma notificação ao usuário e informar onde ocorre o problema. A Figura 50 apresenta o comportamento responsável pelo monitoramento da emissão de alertas e nota-se sua configuração para funcionamento constante e ininterrupto.

Figura 50 – Configuração de comportamento que monitora a emissão de alertas.

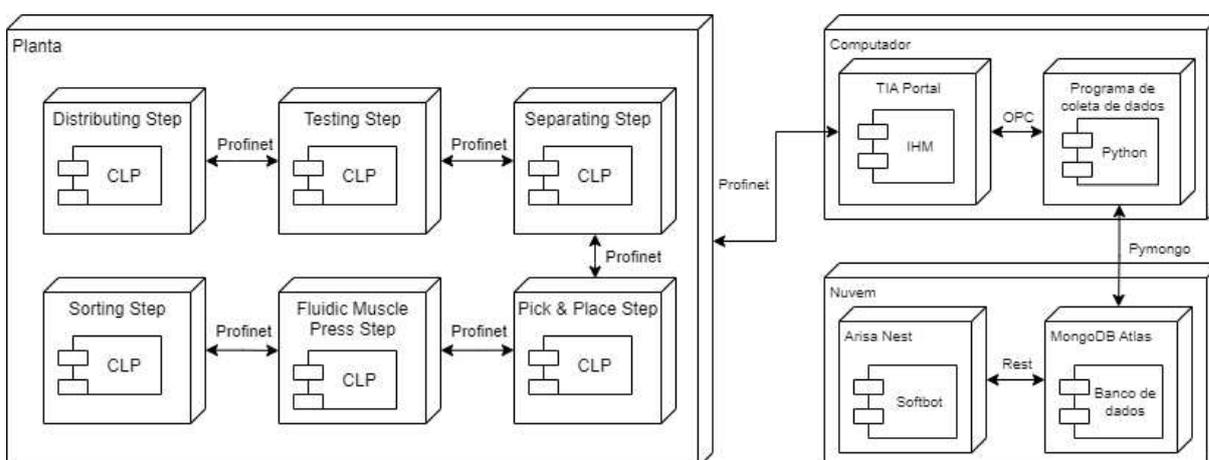
Nome / Linguagem / Descrição	Agendamento
Monitoramento alertas buffer	Regularidade Constantemente
lua	Termina <input checked="" type="checkbox"/> Nunca
Comportamento responsável pelo monitoramento da emissão de alertas relacionados ao buffer de entrada, descarte e saída da planta.	

Fonte: O autor.

6.3 ESTRUTURA DE FUNCIONAMENTO

Cada estação da planta conta com um CLP Siemens e comunica-se com as estações adjacentes. Além disso, cada estação conecta-se também à um computador na mesma rede, o qual é responsável por disponibilizar acesso aos dados referentes à planta. O envio das informações para o banco de dados pode ser feito através de um programa próprio para coleta de dados ou através de maneira direta, utilizando dispositivos IoT. Por fim, o *softbot* comunica-se com o banco de dados através de requisições REST e interage com o usuário. A Figura 51 apresenta o diagrama da estrutura de funcionamento do *softbot* desenvolvido neste projeto.

Figura 51 – Diagrama da estrutura de funcionamento



Fonte: O autor.

7 AVALIAÇÃO DE RESULTADOS

Este capítulo apresenta avaliações sobre o método desenvolvido, bem como sobre a solução implementada. Assim como a metodologia, o *softbot* contou com a execução de um esboço inicial e seu refino ao longo do processo.

7.1 METODOLOGIA

A proposta inicial deste projeto era o desenvolvimento de um assistente virtual que, integrado a uma dada planta, deveria auxiliar o usuário na interação com o processo. No entanto, como os trabalhos acadêmicos sobre o tema ainda não são numerosos, a criação de uma metodologia mostrou-se uma possibilidade. Ao utilizar a linguagem de notação BPMN foi possível propor as tarefas necessárias do projeto, bem como separá-las em três grupos (ou *pools*) de elementos-chave do projeto: Planta, programa para coleta de dados e *softbot*.

As etapas referentes à planta contaram com descrições dos processos já realizados no laboratório (LAI) como guia, o que reforçou seu embasamento e demonstrou sua aplicação. Já as etapas que dizem respeito aos dispositivos IoT, os quais devem dominar a indústria nos anos seguintes, foram baseadas em implementações mais novas e discussões com o orientador do projeto.

Ao desenvolver as tarefas relacionadas ao programa para coleta de dados, foi possível explicitar seu funcionamento apenas durante a execução de processos bem como a necessidade de armazenar as informações em um banco de dados acessível ao *softbot*. O emprego de uma linguagem de programação com ampla e ativa comunidade como o caso do Python é extremamente recomendado, visto que ao facilitar o processo de desenvolvimento, pode-se reservar mais tempo para a concepção e ajustes do *softbot*, como também para análises dos dados obtidos. Por fim, ao optar pelo MongoDB, observou-se seu rápido tempo de resposta e ampla capacidade de armazenamento, o que o tornou uma ótima escolha para este projeto.

A metodologia proposta contempla diferentes possibilidades de leitura de dados da planta, assim como orienta e enfatiza aspectos relevantes relacionados ao desenvolvimento do programa para coleta de dados.

7.2 SOFTBOT

Etapa essencial e proposta inicial do projeto, a concepção de um *softbot* para interagir com um usuário e fornecer informações sobre uma planta específica, é possível afirmar que os resultados atingiram os objetivos desejados. A criação de contextos e diálogos que juntos correspondem ao três tipos de comportamento propostos (responsivo, programado e proativo) mostrou-se funcional para as situações testadas.

7.2.1 Análise por professores

Conforme explanado no Capítulo 2, uma etapa importante que dá suporte para a análise de resultados de projetos de *software* é a sua disponibilização para ser testados por um grupo de indivíduos especialistas. Pela natureza do projeto de fim de curso e seu tempo de desenvolvimento, o teste foi conduzido com quatro professores do Departamento de Automação e Sistemas (DAS) que utilizam a planta MPS-LAI nas suas atividades de ensino, pesquisa e extensão. Adotando-se o método *expert panel*¹ (embora com um número bastante reduzido de pessoas), procedeu-se a uma explicação inicial e uma demonstração do *softbot* aos professores nas dependências da própria planta, seguido da aplicação de um questionário, previamente formulado.

Ao realizar uma etapa de testes, é possível a utilização dos *feedbacks* e relatórios de conversação como meios para ampliar tanto os padrões de pergunta do usuário como as respostas do *bot*, tornando sua interação cada vez mais próxima do comportamento humano.

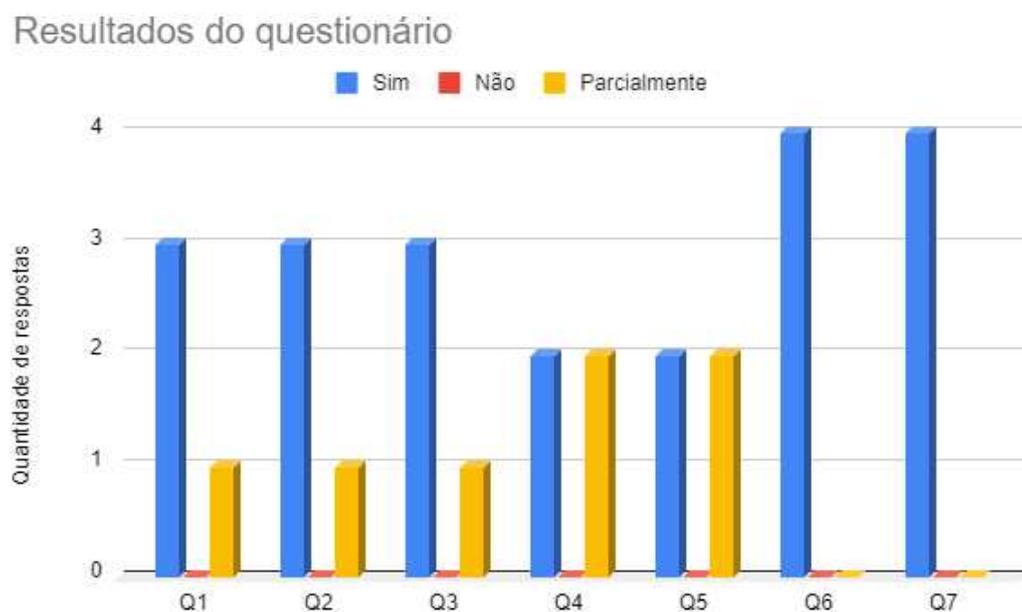
O questionário que foi utilizado na avaliação do *softbot*, quando testado pelo grupo de professores, encontra-se apresentado no Apêndice C. A Figura 52 apresenta os dados coletados para cada uma das perguntas. De uma maneira geral, observou-se que os professores entrevistados atestaram que o trabalho, em termos do *software* desenvolvido, atingiu os objetivos pretendidos. Já quanto a metodologia em si, não foram feitas perguntas aos professores sobre isso porque nenhum deles era um especialista em desenvolvimento de *softbots*. Por outro lado, a metodologia se mostrou coerente, na medida que se a adotou no desenvolvimento do *softbot*.

Além das respostas, o questionário apresentou campos para realização de comentários por parte dos professores, os quais pontuam aspectos como:

- *“Eu acredito que deve ser um motivador para pessoas que estão trabalhando com programação em redes industriais e o funcionamento dele pode ser estudado para desenvolver aplicações parecidas ou para entender comunicação e processamento em nuvem, a ideia dele também pode ser considerada quando os alunos forem programar em LADDER fazendo ambos os lados se beneficiarem.”*
- *“Em trabalhos futuros, pode ser interessante o envio de comandos para a unidade de produção, principalmente se for interessante utilizar um softbot para outros níveis da pirâmide da automação.”*
- *“Os alarmes serem enviados pelo telegram, é uma interação bem interessante já que torna o sistema muito mais iterativo e amigável do que um alarme de falha em uma IHM, Sistema supervisorio ou um atuador (luz ou alarme sonoro).”*

¹ https://europa.eu/capacity4dev/evaluation_guidelines/wiki/expert-panel-0

Figura 52 – Respostas do questionário.



Fonte: O autor.

7.2.2 Análise de funcionamento

Ao possibilitar a ação do *softbot* de maneira programada, o processo para coleta de informações ganha muita agilidade na sua execução, bastando apenas sua configuração uma única vez. Já ao agir de maneira independente, o *bot* permite a possibilidade de notificação remota, seja ao perceber a emissão de um alerta ou identificar através de um algoritmo um possível problema futuro, tarefa esta antes reservada apenas ao operador responsável pelo monitoramento do processo.

A utilização da plataforma Arisa Nest mostrou-se muito bem adequada para o tipo de projeto, uma vez que suas diversas funcionalidades complementam umas às outras. As ferramentas de execução de *script*, armazenamento de crenças e comportamentos permite a execução de diferentes cenários e abrange diversos tipos de implementação.

Um aspecto não abordado durante a execução deste processo é sobre a utilização do *bot* por múltiplos usuários com diferentes permissões de acesso, afinal podem existir dados que não sejam totalmente disponíveis aos operadores. Como atualmente não há critérios de discernimento entre os usuários e suas diferentes funções, seria interessante seu desenvolvimento antes da implementação de novas funcionalidades.

Ao desenvolver o *softbot* houve diversos problemas e erros na entrada e leitura de dados, no entanto apenas os cenários mais relevantes para o projeto foram aprimorados por completo. Os erros relacionados à comunicação OPC entre CLP e

computador foram solucionados com a reinstalação do serviço ou ao reiniciar o computador, enquanto que os erros de desenvolvimento foram solucionados no decorrer do projeto, ao observar os erros apontados na própria plataforma Arisa Nest.

8 CONCLUSÃO E PERSPECTIVAS

Este projeto teve como objetivo a criação de uma metodologia para orientar o desenvolvimento e implementação de *softbots* a uma data planta, além de sua aplicação no LAI. A concepção de um método deste caráter é algo novo no cenário acadêmico e mostra a evolução tanto dos *softbots* como da própria maneira de representar um processo.

Os objetivos do trabalho – a descrição e formalização de uma metodologia de desenvolvimento de *softbots* industriais e o desenvolvimento de um *softbot* – foram atingidos. Por representar uma versão apenas inicial de *softbot* para a planta do LAI, o número de diálogos / cenários de produção suportados é ainda bastante reduzido. Todavia, pode-se perceber o potencial dessa tecnologia no auxílio ao trabalho de operadores e gestores.

Um dos diferenciais entre *chatbots* e IPAs é seu meio de comunicação. Enquanto um é estritamente textual, o outro abrange tanto a escrita quanto a fala, fator responsável por fortalecer o comportamento humano do *softbot*. Durante o desenvolvimento do projeto foi possível combinar, utilizando a linguagem Python, os cenários de conversação desenvolvidos na plataforma Arisa Nest com o meio de comunicação oral. Ao utilizar pacotes como “speech_recognition” e “pyttsx3” foi possível simular uma conversação com o *softbot* desenvolvido através da fala, por meio de requisições diretamente ao serviço web da plataforma. Recomenda-se também o desenvolvimento de algo similar em futuras aplicações similares ou relacionadas ao projeto.

Trabalhos futuros deverão contemplar as seguintes direções: i) a implementação de cenários e diálogos adicionais, inclusive para diferentes tipos de usuários (por exemplo, operadores e gestores); ii) a inclusão de geração de dashboards como respostas; iii) o uso de informação gerada/enviada diretamente por dispositivos de IoT no banco de dados; e iv) a utilização do *softbot* combinada com outras tecnologias, como realidade aumentada e metaverso.

Espera-se que este projeto sirva como inspiração e embasamento para aplicações similares, seja na concepção de uma nova metodologia ou no desenvolvimento de um novo *bot* na mesma plataforma. Busca-se uma evolução ainda mais marcante e significativa no desenvolvimento de projetos similares.

REFERÊNCIAS

- ABNER, B. M.; RABELO, R. J.; ZAMBIAZI, S. P.; ROMERO, D. Production management as-a-service: a softbot approach. **IFIP International Conference on Advances in Production Management Systems**, Springer, Cham, p. 19–30, 2020.
- ADAMOPOULOU, E.; MOUSSIADES, L. An overview of chatbot technology. **IFIP International Conference on Artificial Intelligence Applications and Innovations**, Springer, Cham, p. 373–383, 2020.
- APPLE. Disponível em: <https://www.apple.com/br/siri/>. Acesso em: 11 nov. 2022.
- APPOLINÁRIO, Fábio. **Dicionário de metodologia científica: um guia para a produção do conhecimento científico**. São Paulo: Atlas, 2004.
- CNET. **The 50 Best Alexa Skills That Help Make Your Life Much Easier**. 2022. Disponível em: [https://www.cnet.com/home/smart-home/the-50-best-alexa-skills-that-help-make-your-life-so-much-easier/#:~:text=There%5C%20are%5C%20more%5C%20than%5C%20100%5C%2C000%5C%20Alexa%5C%20skills%5C%20available](https://www.cnet.com/home/smart-home/the-50-best-alexa-skills-that-help-make-your-life-so-much-easier/#:~:text=There%5C%20are%5C%20more%5C%20than%5C%20100%5C%2C000%5C%20Alexa%5C%20skills%5C%20available.). Acesso em: 11 nov. 2022.
- COLBY, Kenneth Mark *et al.* Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. **Artificial Intelligence**, v. 3, p. 199–221, 1972.
- DA SILVEIRA, Clóvis *et al.* Uso de Agente conversacional como recurso de aprendizagem sócio-educacional. **RENOTE**, v. 17, n. 3, p. 668–678, 2019.
- DAHIYA, Menal. A tool of conversation: Chatbot. **International Journal of Computer Sciences and Engineering**, v. 5, n. 5, p. 158–161, 2017.
- EXAME. **Chatbot da Sky reduz tempo de resposta em 88% e aumenta personalização**. 2021. Disponível em: <https://exame.com/bussola/chatbot-da-sky-reduz-tempo-de-resposta-em-88-e-aumenta-personalizacao/>. Acesso em: 07 nov. 2022.
- GIL, Antônio Carlos. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: Atlas, 2010.

JÄRVINEN, Pertti. Action research is similar to design science. **Quality & Quantity**, Springer, v. 41, 2007.

KALNINS, A.; VITOLINS, V. Use of UML and Model Transformations for Workflow Process Definitions. **Databases and Information Systems IV - Selected Papers from the Seventh International Baltic Conference**, p. 3–15, 2006.

KIM, J. H. Ubiquitous Robot. Computational Intelligence, Theory and Applications. **Advances in Soft Computing**, v. 33, 2005.

LI, Yishan; MANOHARAN, Sathiamoorthy. A performance comparison of SQL and NoSQL databases. **IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)**, p. 15–19, 2013.

MARIETTO, Maria das Graças Bruno *et al.* Artificial intelligence markup language: a brief tutorial. **arXiv preprint arXiv:1307.3091**, 2013.

MUGGLETON, Stephen. Alan Turing and the development of Artificial Intelligence. **AI communications**, v. 27, n. 1, p. 373–383, 2014.

MUO. **How Alexa Works: The Science Behind Amazon's Digital Assistant**. 2021. Disponível em: <https://www.makeuseof.com/how-alexa-works-science-behind-amazon-assistant/>. Acesso em: 11 nov. 2022.

NAMA. **11 usos de chatbot para ecommerce que aumentam as vendas**. 2017. Disponível em: <https://simple.nama.ai/post/11-usos-de-chatbot-para-ecommerce-que-aumentam-as-vendas>. Acesso em: 06 nov. 2022.

OMG. **Business Process Model and Notation (BPMN)**. 2011a. Disponível em: <https://www.omg.org/spec/BPMN/2.0/PDF>. Acesso em: 10 nov. 2022.

OMG. **OMG Unified Modeling Language™ (OMG UML), Superstructure**. 2011b. Disponível em: <https://www.omg.org/spec/UML/2.4.1/Superstructure/PDF>. Acesso em: 10 nov. 2022.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software: fundamentos, métodos e padrões**. São Paulo: LTC Editora, 2000.

RABELO, R. J.; MAGALHAES, L. C.; CABRAL, F. G. O. Uma Proposta de Arquitetura de Referência de Gêmeo Digital para Sistemas Ciberfísicos em um cenário de Indústria 4.0. *In: ANAIS CBA 2020 - XXIII Congresso Brasileiro de Automática*. [S.l.: s.n.], 2020. P. 1–8.

RABELO, R. J.; ZAMBIAZI, S. P.; ROMERO, D. Collaborative softbots: enhancing operational excellence in systems of cyber-physical systems. *In: SPRINGER. ANAIS PRO-VE'39;2019 - 20th IFIP Working Conference on Virtual Enterprises*. [S.l.: s.n.], 2019. P. 55–68.

RABELO, Ricardo J; ROMERO, David; ZAMBIASI, Saulo P. Softbots supporting the operator 4.0 at smart factory environments. **IFIP International Conference on Advances in Production Management Systems**, Springer, Cham, p. 456–464, 2018.

RESEARCH, Juniper. **CHATBOTS TO DELIVER \$11BN IN ANNUAL COST SAVINGS FOR RETAIL, BANKING & HEALTHCARE SECTORS BY 2023**. 2018. Disponível em: <https://www.juniperresearch.com/press/chatbots-to-deliver-11bn-cost-savings-2023>. Acesso em: 06 nov. 2022.

SANTOS, João *et al.* An IoT-based mobile gateway for intelligent personal assistants on mobile health environments. **Journal of Network and Computer Applications**, v. 71, p. 194–204, 2016.

SCHEDLBAUER, Martin. **The Art of Business Process Modeling: The Business Analyst's Guide to Process Modeling with UML BPMN**. [S.l.]: CreateSpace, 2010.

SOMMERVILLE, Ian. **Engenharia de Software**. [S.l.]: Pearson, 2019.

SOMMERVILLE, Ian. **Software Engineering**. 9. ed. [S.l.]: Pearson Education India, 2011.

VENABLE, John. The role of theory and theorising in design science research. **Proceedings of the 1st International Conference on Design Science in Information Systems and Technology (DESRIST 2006)**, p. 1–18, 2006.

WALLACE, Richard. The elements of AIML style. **Alice AI Foundation**, v. 139, 2003.

WEIZENBAUM, Joseph. ELIZA—a computer program for the study of natural language communication between man and machine. **Communications of the ACM**, v. 9, n. 1, p. 36–45, 1966.

WHITE, Stephen A.; MIERS, Derek. **BPMN modeling and reference guide: understanding and using BPMN**. [S.l.]: Future Strategies Inc., 2008.

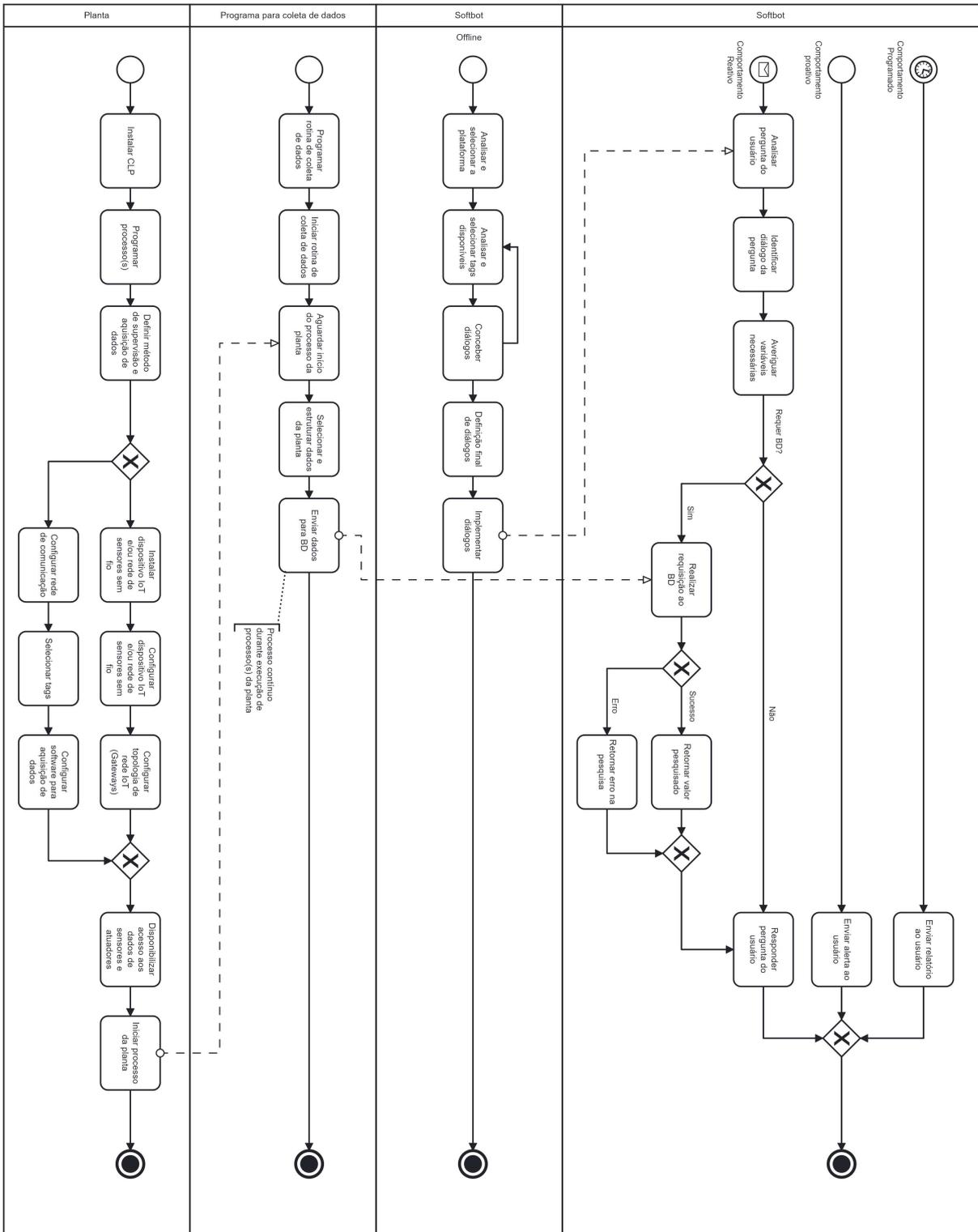
WIERINGA, Roel. Design science as nested problem solving. **Proceedings of the 4th international conference on design science research in information systems and technology**, p. 1–12, 2009.

ZAMBIASI, Lara P. *et al.* Supporting Resilient Operator 5.0: An Augmented Softbot Approach. **IFIP International Conference on Advances in Production Management Systems**, Springer, Cham, p. 494–502, 2022.

ZAMBIASI, Saulo P. **Chatbot Arisa: Tutorial. 2017**. 2017. Disponível em: https://saulo.arisa.com.br/wiki/index.php/Chatbot_Arisa:_Tutorial. Acesso em: 10 nov. 2022.

Apêndices

APÊNDICE A – DIAGRAMA BPMN PARA DESENVOLVIMENTO E IMPLEMENTAÇÃO DE *SOFTBOT*



APÊNDICE C – QUESTIONÁRIO PARA AVALIAÇÃO DO *SOFTBOT*, POR PARTE DOS PROFESSORES AVALIADORES

QUESTIONÁRIO DE AVALIAÇÃO

PFC “Desenvolvimento de metodologia e implementação de softbot para planta didática do LAI”

Aluno André Luiz Nunes Amaru

Este questionário tem por objetivo avaliar os aspectos essenciais do referido PFC frente aos seus objetivos de implementação, realizado na planta MPS do Laboratório de Automação Industrial (LAI) do Departamento de Automação e Sistemas da UFSC.

Um *softbot* é um software de interação com o ser humano que, além de prover um ambiente de perguntas & respostas (como um *chatbot*), possibilita a execução de outras tarefas não necessariamente diretamente pedidas pelo humano.

Após observar a demonstração da implementação de um softbot para a planta MPS-LAI e assumindo que o usuário já tenha um certo prévio conhecimento da operação desta planta:

1- Você considera que o softbot implementado, para os 3 casos projetados, permitiu uma interação responsiva, programada e proativa com o usuário ?

Sim Não Parcialmente

Comentários adicionais, caso desejar:

2- Você considera que o softbot implementado permitiu uma interação amigável com o usuário em termos de um diálogo que procura imitar uma comunicação semelhante a feita entre humanos ?

Sim Não Parcialmente

Comentários adicionais, caso desejar:

3- Você considera que o softbot implementado, para os 3 casos projetados, pode ser útil nas atividades de ensino e/ou pesquisa em cima da planta MPS do LAI ?

Sim Não Parcialmente

Comentários adicionais, caso desejar:

4- Você considera que o softbot implementado, para os 3 casos projetados, foi fácil de usar, não requerendo muito treinamento prévio ?

Sim Não Parcialmente

Comentários adicionais, caso desejar:

5- Você considera que o softbot implementado, para os 3 casos projetados, permitiu uma interação rápida e facilmente entendível pelo usuário dos aspectos de produção previstos ?

Sim Não Parcialmente

Comentários adicionais, caso desejar:

6- Você considera que softbots - como este implementado - representam uma forma mais amigável e simbiótica de interação humano-máquina em ambientes industriais ?

Sim Não Parcialmente

Comentários adicionais, caso desejar:

7- Você considera que softbots - como este implementado - tem o potencial de facilitar o trabalho de um operador humano na operação e gestão de uma planta industrial, incluindo fazendo com que complexas atividades sejam executadas automaticamente e/ou proativamente ?

Sim Não Parcialmente

Comentários adicionais, caso desejar:

Comentários e/ou sugestões gerais finais, caso desejar:

Prof Avaliador

Florianópolis, XX de novembro de 2022.

APÊNDICE D – GUIA DE UTILIZAÇÃO PARA USUÁRIO E DESENVOLVEDOR

D.1 DESENVOLVEDOR

Para programação de novas interações e comportamentos no *softbot*, deve-se seguir os seguintes passos:

1. Acesso à plataforma Arisa Nest¹, onde deve ser feita a criação de uma conta.
2. Solicitação de permissão para edição do *bot* em questão, a qual deve ser realizada em um contato (via e-mail) com o Prof. Saulo Popov Zambiasi, criador e administrador da plataforma.
3. Uma vez fornecida a permissão de edição do *bot*, pode-se utilizar todas as ferramentas que a plataforma oferece para programar novos diálogos, comportamentos e *scripts*.

D.2 USUÁRIO

Existem dois meios principais para a realização de interações com o *softbot*, onde um é a própria plataforma Arisa Nest e o outro é o aplicativo de mensagens Telegram.

Para a realização de interações com o *softbot* dentro da plataforma Arisa Nest há a execução de um processo análogo ao desenvolvedor. Há a necessidade da criação de uma conta e a solicitação da permissão de leitura do *bot*. É importante ressaltar que com a permissão de leitura, além de ser capaz de interagir com o *softbot*, o usuário pode visualizar todos os diálogos, comportamentos e *scripts* criados.

Outra possibilidade é a interação via aplicativo de mensagens Telegram. Para poder interagir, basta a geração e leitura de um *QR Code* ou acesso via link, ambos disponibilizados pelo desenvolvedor do *bot*. O *softbot* desenvolvido neste projeto pode ser acessado em <https://t.me/MarinChatbot>.

¹ <https://nest.arisa.com.br>