UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Ricardo Ventura

Desenvolvimento de um assistente virtual híbrido com propriedades de chatbot e de automação de processos de negócios

Florianópolis
2019

Ricardo Ventura

Desenvolvimento de um assistente virtual híbrido com propriedades de chatbot e de automação de processos de negócios

Relatório submetido à Universidade Federal de Santa Catarina como requisito para a aprovação na disciplina **DAS 5511: Projeto de Fim de Curso** do curso de Graduação em Engenharia de Controle e Automação.

Orientador: Prof. Ricardo José Rabelo

Co-orientador: Prof. Saulo Popov Zambiasi

Florianópolis
2019

Ricardo Ventura

Desenvolvimento de um assistente virtual híbrido com propriedades de chatbot e de automação de processos de negócios

Esta monografia foi julgada no contexto da disciplina DAS5511: Projeto de Fim de Curso e aprovada na sua forma final pelo Curso de Engenharia de Controle e Automação.

Florianópolis, 8 de fevereiro de 2019.

Banca Examinadora:

Prof. Ricardo José Rabelo
Orientador
Universidade Federal de Santa Catarina

Prof. Saulo Popov Zambiasi
Co-orientador
Universidade do Sul de Santa Catarina

Cleber Jorge Amaral
Avaliador
Universidade Federal de Santa Catarina

Henrique Peixe Maziero
Debatedor
Universidade Federal de Santa Catarina

Nilmar Luiz Guarda Junior
Debatedor
Universidade Federal de Santa Catarina

À minha família e amigos.

AGRADECIMENTOS

À Universidade Federal de Santa Catarina, especialmente ao Departamento de Automação e Sistemas pelo apoio oferecido ao longo dos anos de graduação.

Aos meus orientadores, Prof. Dr. Ricardo José Rabelo e Prof. Dr. Saulo Popov Zambiasi por todo o apoio prestado, o qual foi fundamental para realização deste trabalho.

À minha família e amigos, que sempre me apoiaram e incentivaram em todos os momentos de dificuldade.

“É importante retirar sabedoria de lugares diferentes. Se você a retira de um lugar apenas, ela se torna inflexível e obsoleta. “

Mako Iwasuma

RESUMO

As pessoas vivem cada vez mais ocupadas, com diversas tarefas para cumprir e com prazos sempre mais curtos. Ao mesmo tempo, elas também utilizam novas ferramentas como smartphones e computadores por longos períodos, muitas vezes desviando o foco de assuntos importantes. Neste contexto, os assistentes virtuais possibilitam tornar mais produtivo o tempo passado nestes dispositivos através da realização de diferentes tipos de tarefas. Esse tipo de assistente pode ser utilizado nos mais diversos contextos, atendendo a requisições e até resolvendo problemas de forma autônoma. Este trabalho propõe o desenvolvimento de um assistente virtual integrado a serviços externos em três diferentes cenários de atuação: gerenciamento de informações e execução de operações para alunos e professores da UFSC, auxílio ao operador de uma máquina e atendimento ao cliente e venda de produtos. Uma base de conhecimento para o assistente foi arquitetada na plataforma para criação e gerenciamento de assistentes virtuais chamada Arisa Nest. Para suprir os requisitos do projeto, foi necessário a criação de scripts em linguagem de programação Lua e o acesso à serviços web estilo SOA através da plataforma. O trabalho resultou num assistente virtual capaz de executar diversas ações para estes diferentes cenários de uma maneira bastante eficaz.

Palavras-chave: Arisa Nest, Assistente virtual, Base de conhecimento, *Chatbot*, Lua, PHP, Serviços web.

ABSTRACT

Nowadays people are always busy, they have several tasks to perform in a short time frame. At the same time, people also use smartphones and computers for long periods of time, often shifting focus from important issues. In this context, virtual personal assistants make it possible that the time spent on these devices become more productive by performing different tasks. This kind of assistant can be used in various contexts, meeting requests and even solving problems autonomously. This paper proposes the development of a virtual assistant integrated to external services in three different scenarios: management of information and execution of tasks for UFSC's students and professors, assistance to the operator of an industrial machine, customer service and sale of products. A knowledge base for the chatbot was architected on the Arisa Nest platform. To meet the project requirements, it was necessary to create scripts using Lua programming language and access SOA-style web services through the platform. The work resulted in a virtual assistant capable of performing several actions for these different scenarios in a very effective way.

Key-words: Arisa Nest, Chatbot, Knowledge base, Lua, Virtual assistant, PHP, Web services.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AIML – *Artificial Intelligence Markup Language*

ARISA – *Assistant Representative: an Instance using Service Architecture*

AV – Assistente Virtual

CAGR – Sistema de Controle Acadêmico da Graduação

IA – Inteligência Artificial

PHP – *PHP: Hypertext Preprocessor*

RU – Restaurante Universitário

SOA – *Service-oriented Architecture*

SOAP – *Simple Object Access Protocol*

UFSC – Universidade Federal de Santa Catarina

WSDL – *Web Services Description Language*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Com a evolução dos meios de comunicação a troca de informações acontece de forma cada vez mais rápida, dinâmica e através dos mais variados canais. O advento da internet e também das redes sociais permitiu que as pessoas e também as instituições pudessem diversificar as formas pelas quais elas se relacionam e se comunicam. Como consequência disso, as pessoas passaram a ser mais exigentes em relação a rapidez e a qualidade do atendimento prestado a elas (EXAME, 2018).

Uma possível alternativa para alcançar essas novas expectativas é o uso da comunicação homem-máquina. Devido ao constante aprimoramento da tecnologia, a linguagem de comunicação homem-máquina está se tornando muito próxima à Linguagem Natural utilizada em conversas entre os seres humanos. A utilização de simples comandos de voz, por exemplo, é uma maior comodidade para muitos usuários (MANFIO; MORENO, 2015).

A ideia de softwares que funcionam como assistentes pessoais não é um pensamento tão novo. São apresentados como softwares baseados em Inteligência Artificial e tem a função de executar diversas atividades como pagamento de contas, envio de e-mails, etc. (Michaell et al., 1994).

A literatura referente a Softwares Assistentes Pessoais (SAP/*Personal Assistant Software)* não é muito clara quanto a sua definição, geralmente as suas definições são baseadas na finalidade dos seus requisitos funcionais. De uma forma geral é possível afirmar que os SAPs têm a função de ajudar na execução de tarefas longas e mecanicamente repetitivas, podendo ser chamados de *softbots* (Hoyle, 1997).

A Inteligência Artificial (IA), segundo Rover (2010, p. 59), “É a ciência do conhecimento que busca a melhor forma de representá-lo, como também é a ciência que estuda o raciocínio e os processos de aprendizagem em máquinas.”. Para Coppin (2015), a IA é um campo de estudo de sistemas que atuam de forma inteligente na percepção de um indivíduo qualquer.

A capacidade de uma IA expressar características humanas em Linguagem Natural em diálogos com seres humanos é um dos maiores propósitos buscado pelos desenvolvedores. Dentro desse contexto, foi proposto por Alan Turing na

década de 50 um desafio para determinar se um computador era inteligente ou não. Se não fosse possível distinguir um ser humano de uma máquina através da troca de mensagens escritas, o sistema era considerado inteligente (FOSSATTI, 2011).

Esse foi o ponto inicial dos sistemas que simulam um diálogo, que atualmente podem ser desenvolvidos para diversas áreas de aplicação. Esses sistemas são conhecidos como *chatbots*, desenvolvidos para tornar natural a interação entre ser humano e computador (SGANDERLA, 2003).

Em uma perspectiva diferente, sistemas passam a ser desenvolvidos num estilo de orientação a serviços. Entende-se que o paradigma SOA (*Service Oriented Architecture*) tem sido usado como um estilo arquitetural que integra e dá suporte à interoperação entre sistemas heterogêneos e distribuídos sob uma mesma arquitetura, de uma forma flexível, escalável e com baixo acoplamento, permitindo que diferentes aplicações possam mais agilmente ser geradas e alinhadas ao negócio das empresas (SINGH et al., 2005).

1.1 Problema

Este projeto visa desenvolver um assistente virtual (AV) que possa atuar tanto como um *chatbot* (respondendo rápida e corretamente a perguntas de usuários dentro de certos contextos) como um assistente pessoal que possa executar ações (processos de negócios) associados ao o que o usuário deseja. Para tal, se utiliza de um framework chamado Arisa Nest, que permite a derivação de AV abertos e integráveis em diversos domínios de aplicação, e por ser orientado a serviços permite que todas suas funcionalidades sejam modeladas e programadas como serviços de software (como serviços web), sejam desenvolvidos pelo próprio implementador, seja serviços externos, providos por empresas ou ecossistemas. Ao final, um AV derivado se torna internamente um sistema orientado a serviços implementado sob uma ótica SOA.

A premissa básica associada a esta pesquisa é de que as empresas e demais organizações precisam cada vez mais tomar decisões confiáveis e de forma ágil, mas seus colaboradores estão cada vez mais imersos em inúmeros processos de negócios, simultâneos, quer repetitivos quer complexos, e cada vez mais sem o devido tempo para geri-los. Portanto, AVs têm o potencial de atuar como uma

importante ferramenta de trabalho para auxiliar seus usuários a executar suas atividades melhor ou mesmo automaticamente, substituindo-os em certas ações.

1.2 Objetivo Geral

O objetivo geral do projeto é integrar áreas do conhecimento e tecnologia para desenvolver e utilizar um *chatbot* assistente virtual apropriado para captação e gerenciamento de informações de usuários e que possa executar operações por si próprio ou provindas de serviços externos, tais como serviços web.

1.3 Objetivos Específicos

Para nortear este trabalho, são definidos os seguintes objetivos específicos:

- Pesquisar e efetuar levantamento de material bibliográfico com base em pesquisas sobre assistentes virtuais e integração entre sistemas.
- Definir os cenários para criação da base de conhecimento do assistente virtual.
- Analisar a plataforma de criação de assistentes virtuais Arisa Nest.
- Criar um AV na plataforma supracitada, levantando a base de conhecimentos e adaptando-a para esta instância.
- Pesquisa, análise, desenvolvimento e utilização de serviços externos, integrando com o AV.
- Testes e avaliação dos resultados obtidos.

1.4 Organização do Documento

Este documento está organizado em 8 capítulos. O capítulo 2 refere-se à metodologia de pesquisa utilizada para a elaboração do projeto. No capítulo 3 são citadas algumas das referências literárias que serviram como pesquisa base.

Apresentou-se no capítulo 4 a proposta de desenvolvimento do assistente virtual e também foram determinadas as especificações a serem atendidas. As ferramentas e tecnologias utilizadas são descritas no capítulo 5. No capítulo 6 é explicado a forma como todos os elementos do projeto foram implementados.

A análise e discussão dos resultados obtidos é feita no capítulo 7. O capítulo 8 é o capítulo final deste documento e tem como objetivo fazer uma síntese do trabalho e também discutir sugestões de trabalhos futuros.

2 CONTEXTUALIZAÇÃO E METODOLOGIA DA PESQUISA

Antes de se descrever a metodologia-base dentro da qual este trabalho foi desenvolvido, é importante dar uma contextualização maior do porquê deste trabalho.

2.1 Contextualização

Um *chatbot* foi inicialmente desenvolvido como um projeto de estágio na UFSC para a secretaria do Programa de Pós-Graduação em Engenharia Mecânica (POSMEC) com o objetivo de responder a dúvidas frequentes de alunos e professores do programa. Neste projeto, tomou-se como ponto de partida o fato da secretaria estar sobrecarregada com solicitações repetitivas, o que indicava que seria bastante vantajosa a automação do atendimento para serviços específicos. O chatbot desenvolvido mostrou-se bastante efetivo para responder às perguntas frequentes feitas pelos alunos do seu programa de pós-graduação.

Devido ao bom resultado, sugeriu-se que, além de somente responder a padrões de perguntas, que houvesse a possibilidade de executar tarefas e acessar serviços externos poderiam trazer vantagens no comportamento e inteligência do mesmo, possibilitando melhor assistência aos usuários e caracterizando-o como um assistente virtual.

Inicialmente propôs-se desenvolver um assistente virtual na forma de assistente pessoal exclusivamente dentro do contexto das necessidades de alunos e professores da UFSC. O assistente virtual teria acesso a diversos serviços web da UFSC, podendo executar diversas funções através dos mesmos. Devido a problemas técnicos e burocráticos de não possibilidade legal de pleno acesso a certos serviços web da UFSC (via Setic) não foi possível utilizar os serviços da forma esperada. Para compensar essa limitação, optou-se por acrescentar outros dois cenários de atuação do assistente pessoal para que as diversas funcionalidades da plataforma Arisa Nest pudessem ser exploradas.

2.2 Metodologia da pesquisa

Demo (1996, p.34) considera a pesquisa como uma atividade do cotidiano, como uma atitude, um “questionamento sistemático crítico e criativo, mais a intervenção competente na realidade, ou o dialogo crítico permanente com a realidade em sentido teórico e prático”.

De acordo com Gil (2010), a pesquisa é um processo formal e sistemático que tem como objetivo fundamental descobrir respostas para problemas a partir do uso de procedimentos científicos. É um processo que envolve diferentes fases, iniciando com a formulação do problema até a apresentação e discussão dos resultados finais.

O método de pesquisa adotado neste trabalho foi o *Design Science* (JÄRVINEN, 2004). A escolha foi sugerida pelos orientadores do projeto por ser considerada adequada para a dinâmica de trabalho e resultados esperados. *Design Science* pode ser definido, segundo Järvinen (2007), como o desenvolvimento de um artefato qualquer com o objetivo de solucionar ou aprimorar um problema já devidamente analisado e entendido. Esse método de pesquisa é realizado através de um método progressivo, em que ciclicamente teorias e conhecimentos já fundamentados são utilizados para melhorar o que está sendo desenvolvido e avaliar os seus resultados.

Como característica do *Design Science,* é comum que que ocorra avanço e retorno de etapas em função dos aspectos do projeto (JÄRVINEN, 2004). Em função disso, pode-se afirmar que o desenvolvimento do projeto foi de forma incremental, muito comum nessa área de pesquisa. As etapas do processo desse método de pesquisa condizem com as atividades do trabalho realizado descritas nas próximas seções.

2.1 Procedimentos gerais para desenvolvimento do assistente virtual

Esta seção resume as direções tomadas para a realização das etapas desse trabalho.

O primeiro passo foi revisar e analisar os resultados alcançados no *chatbot* implementado na secretaria do PosMec da UFSC, podendo usar parte de sua

estrutura como base do novo projeto. Junto a isso foi realizada uma extensa pesquisa sobre o conceito e o estado da arte dos assistentes virtuais e uma pesquisa técnica sobre a linguagem de programação Lua, serviços web, protocolo SOAP e Arquitetura Orientada a Serviços (SOA).

Com a plataforma de desenvolvimento do AV já definida, um tempo foi dedicado para adaptação à interface e às funcionalidades do sistema de gerenciamento Arisa Nest e iniciou-se a construção de uma pequena base de conhecimento para o assistente pessoal.

Após a conclusão destas tarefas, foi feito um levantamento dos serviços web disponibilizados pela UFSC e analisado como eles poderiam ser utilizados para compor as funcionalidades do assistente pessoal. Após o término deste levantamento, criou-se novos serviços web para complementar os serviços com uso autorizado pela UFSC e simular os indisponíveis ou inexistentes.

Por ter sido definido também como objetivo do trabalho a adição de dois outros cenários de atuação do AV, foi necessária a utilização de outras plataformas para complementar esses cenários. Foi preciso um ambiente de computação virtual da Amazon com servidor Apache para hospedar novos serviços web e uma instância de banco de dados MySQL para armazenar os dados acessados por esses serviços.

A etapa seguinte foi o desenvolvimento dos scripts em linguagem Lua que são responsáveis por consumir os serviços web para o AV. Neste momento do trabalho diversos testes e modificações foram feitos nos scripts e nos serviços web para melhorar as interações e respostas do assistente. Em paralelo a muitas das etapas, mas principalmente ao final do processo, a escrita deste documento passou a fazer parte das atividades.

Todas essas ações foram desenvolvidas dentro da metodologia de pesquisa Design Science, a seguir descritas.

2.1.1 Entendimento do Problema

A etapa inicial trata-se do entendimento e da identificação do problema de forma geral. O problema foi apresentado como uma possibilidade de criar uma nova versão aperfeiçoada de um *chatbot* já existente dentro do contexto da UFSC. Notou-se que a possibilidade de usar a nova plataforma de gerenciamento de AV

juntamente com os serviços web disponibilizados pela UFSC seria um cenário ideal para a concepção de um novo *bot*.

Considerando o objetivo do trabalho, foi necessário estudar quais os serviços web oferecidos pela UFSC e entender como eles poderiam ser usados pelo assistente para poder anteder às necessidades dos usuários. Após um período de análise, entendeu-se que para possibilitar o desenvolvimento de um trabalho mais robusto, era necessário explorar mais áreas de atuação para o *bot.*

2.1.2 Sugestões

Após a compreensão do problema, realizou-se uma pesquisa bibliográfica com o objetivo de compreender as diferentes categorias de assistentes virtuais, seu funcionamento e as possíveis aplicações. Uma pesquisa sobre bases de conhecimento para AV também foi necessária para consolidar a proposta de solução.

Apesar da plataforma de gerenciamento de assistentes virtuais Arisa Nest ter sido escolhida previamente para suportar o projeto, a pesquisa bibliográfica contribuiu para o entendimento da atribuição de outros exemplos de *chatbots* já implementados.

Sugeriu-se então que os cenários de venda e de auxílio a um operador eram suficientes para explorar algumas das novas funcionalidades da plataforma.

2.1.3 Desenvolvimento

O desenvolvimento iniciou-se pouco tempo depois da definição do projeto, com o objetivo inicial de criar uma base de conhecimento simples na nova plataforma e solicitar o uso dos serviços web da UFSC. Após o contato inicial com a SeTIC, setor responsável pela liberação de acesso aos serviços web, foi possível ter uma perspectiva inicial do que seria possível implementar.

A partir daí o desenvolvimento comprometeu-se um pouco devido a uma longa espera burocrática para ter acesso às credenciais de acesso. Logo após isso também foi observado uma dificuldade extra para usar os serviços, o que gerou a

necessidade de desenvolver serviços web intermediários hospedados em um servidor na UFSC.

Após o desenvolvimento desses serviços, foi possível dar continuidade ao trabalho expandindo a base de conhecimento e implementando os *scripts* em Lua. Necessitou-se também desenvolver outros serviços web e hospedá-los num servidor externo para executar as ações dos outros dois cenários de implementação.

2.1.4 Avaliação

O projeto não foi disponibilizado para testes de usuários, de modo que o mesmo foi avaliado através da análise do cumprimento dos requisitos de projeto. Após a implementação de algumas funções, elas foram fortemente testadas. Os resultados desses testes forneciam material para melhorar essas funções e também evitar futuros problemas em novas funções implementadas. Esses ciclos de melhoramento já estavam previstos na metodologia *Design Science*.

2.1.5 Conclusões

O objetivo dessa última etapa do método *Design Science* é apresentar uma visão geral do que foi realizado e dos objetivos cumpridos. Além disso são descritas algumas sugestões para melhorias e trabalhos futuros.

3 REVISÃO DA LITERATURA

Este capítulo contém uma revisão bibliográfica dos principais conceitos utilizados neste trabalho.

3.1 *Chatbots* e Assistentes Virtuais

Um *chatbot* ou *chatterbot* é um agente inteligente que usa técnicas de inteligência artificial com o objetivo de interagir com usuários em determinado domínio, simulando conversas em linguagem natural. Na maior parte dos casos o *chatbot* tem a finalidade de responder perguntas dos usuários como se fosse uma pessoa. O objetivo é que os usuários fiquem com a impressão de que estão se comunicando com outra pessoa ao invés de um programa de computador (TEIXEIRA, 2005).

Segundo Neves (2005) pode-se dividir os *chatbots* em três gerações em conformidade com as técnicas empregadas.

- Técnica de Casamento de Padrões e Regras Gramaticais: funciona utilizando palavras-chave extraídas das perguntas e com respostas pré-programadas.

- Técnica de Inteligência Artificial: funciona com regras de produção e redes neurais.

- Linguagens de Marcação: é considerado o método mais completo na categoria do processamento natural. É baseado em AIML.

Os *chatbots* também podem ser organizados em categorias de acordo com a sua finalidade (LAUREANO, 1999).

- Educacional: tem o objetivo de atender alunos e professores, fornecendo diversos tipos e informação e respondendo à duvidas.

- Entretenimento: tem o propósito de entreter e divertir o usuário de diversas maneiras.

- Perguntas mais Frequentes: são responsáveis por responder perguntas dos usuário relacionadas a um tópico específico. Intenção de substituir um FAQ (*Frequent Asked Questions*).

- Comerciais: tem a função de atender clientes ou dar suporte aos consumidores.

- Propósito Geral: outros *chatbots* que não se encaixam nas categorias anteriores ou não tenham propósito específico.

Para os assistentes virtuais há diversas definições na literatura, mas nenhuma engloba todos conceitos associados de forma satisfatória. É comum então utilizar os requisitos do assistente para descrever a sua definição. As mais comuns são (ZAMBIASI; RABELO, 2012): autonomia, flexibilidade, adaptabilidade de contextos, integração com o usuário, ser baseado em rede, ser de um propósito geral, adaptabilidade de necessidades e ser integrável e interoperável.

Existem várias implementações de chatbots e AVs. Algumas das principais encontradas na literatura são descritas nas seções a seguir.

3.1.1 ELIZA

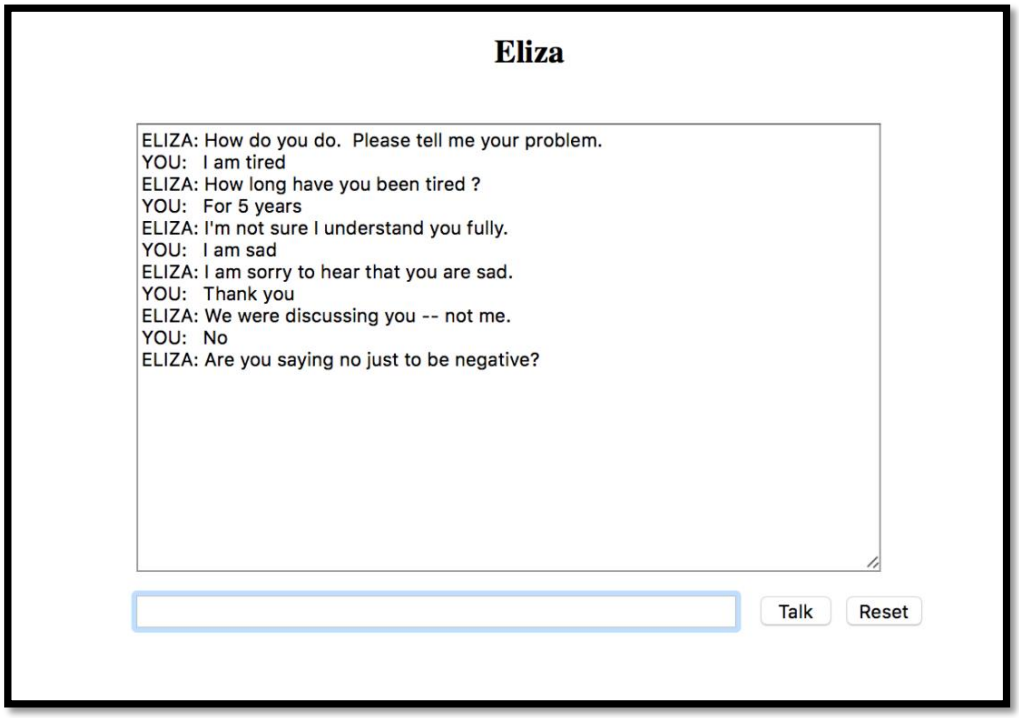
O primeiro *chatbot* criado foi o ELIZA, desenvolvido por Joseph Weizenbaum em 1966. Ele tinha como objetivo principal emular um psicoterapeuta, sendo um dos primeiros programas de inteligência artificial (WEIZENBAUM, 1966).

Quando pessoas que não conheciam inteligência artificial foram colocadas para conversar com ELIZA, a maioria deles pensava que estava falando com uma pessoa real mesmo depois de horas de conversa. Esse acontecimento foi uma surpresa até mesmo para Weizenbaum.

ELIZA era implementado de uma forma simples. A ideia principal era tentar transformar as entradas dos usuários em novas perguntas e usava frases prontas

quando não conseguia fazer isso. ELIZA não tinha a capacidade de armazenar informações de conversa e suas respostas tinham a intenção de serem amigáveis (WEIZENBAUM, 1966). A Figura 1 mostra um exemplo de diálogo com ELIZA.

Figura 1 – Diálogo em uma versão alternativa do ELIZA.



Fonte: Adaptado de Weizenbaum.

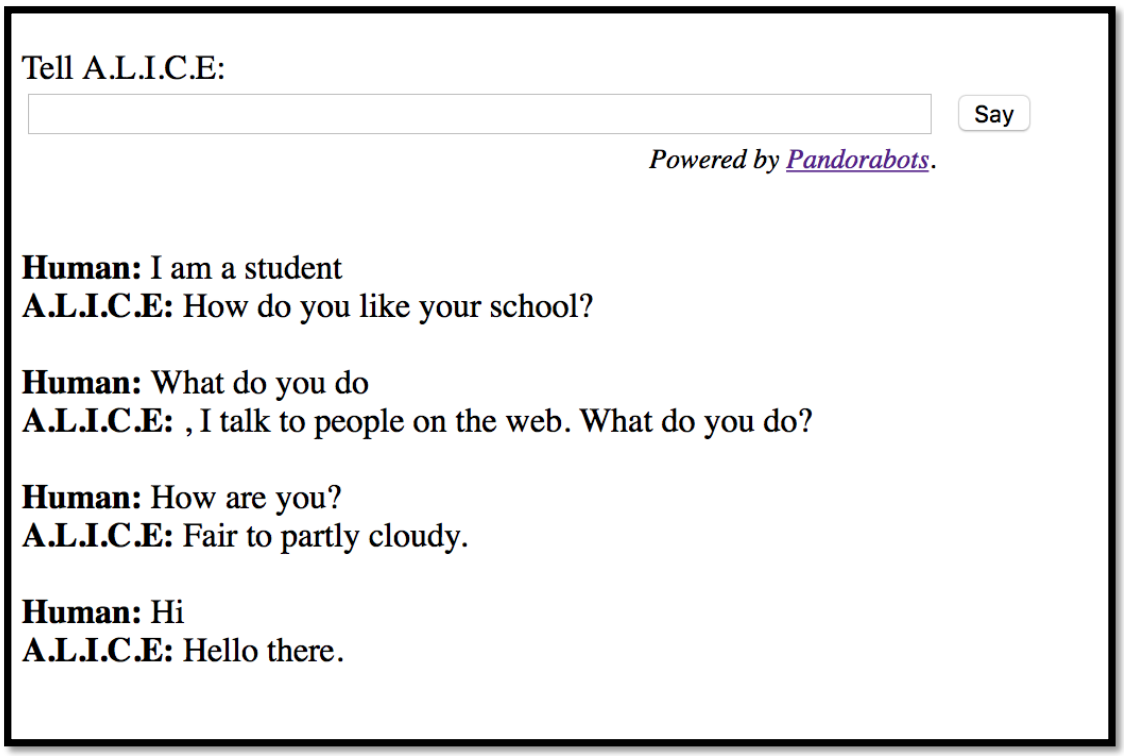
3.1.2 A.L.I.C.E.

Artificial Linguistic Internet Computer Entity (A.L.I.C.E.), um *chatbot* criado por Richard S. Wallace, foi um dos mais populares do seu tempo e pode ser considerado como uma evolução do ELIZA (LEONHARDT, 2003). A.L.I.C.E. possui uma base de conhecimento formada a partir de diversos fatos, citações e ideias do seu autor e um vocabulário bastante extenso, com mais de 5000 palavras. Possui módulos de conversação capaz de categorizar os usuários por sexo, idade, profissão e localização geográfica (KRAUS, 2007).

Esse *chatbot* opera com *Artificial Intelligence Markup Language* (AIML), linguagem desenvolvida por Wallace e que permite que os usuários também possam criar seus próprios *chatbots*. Uma das características dessa linguagem é a capacidade do *chatbot* armazenar informações fornecidas pelo usuário como nome ou idade (MIKIC, 2012). AIML é uma linguagem baseada em *tags* que segue os mesmos padrões do *eXtensible Markup Language* (XML). Ainda de acordo com Mikic (2012), um dos maiores problemas do AIML é a falta de resposta do *chatbot* quando a mensagem de entrada não está contida na base de conhecimento.

O *chatbot* foi implementado com a ideia essencial de possuir um *botmaster*, uma pessoa que administra a base de conhecimento. A função do *botmaster* é analisar os *logs*, identificar a necessidade de melhorias e criar novos conteúdos para a base de conhecimento na forma de arquivos AIML, com o objetivo de gerar respostas mais apropriadas aos usuários. A Figura 2 mostra um exemplo de diálogo com ALICE.

Figura 2 – Diálogo com o chatbot ALICE.



Fonte: Adaptado de Pandorabots.

3.1.3 Robô Ed

Um exemplo de *chatbot* para dar informações muito conhecido no Brasil é o Robô Ed. O Robô Ed pertence ao Conpet e está ligado à Petrobrás e ao Ministério de Minas e Energia. A sua função é conversar sobre temas relacionados ao meio ambiente, mas responde também a algumas perguntas fora do contexto. Tudo é feito através do teclado, com as respostas obtidas na tela (MANFIO; MORENO, 2015). A Figura 3 mostra um exemplo de diálogo com o Robô Ed.

Figura 3 – Diálogo com Robô Ed.

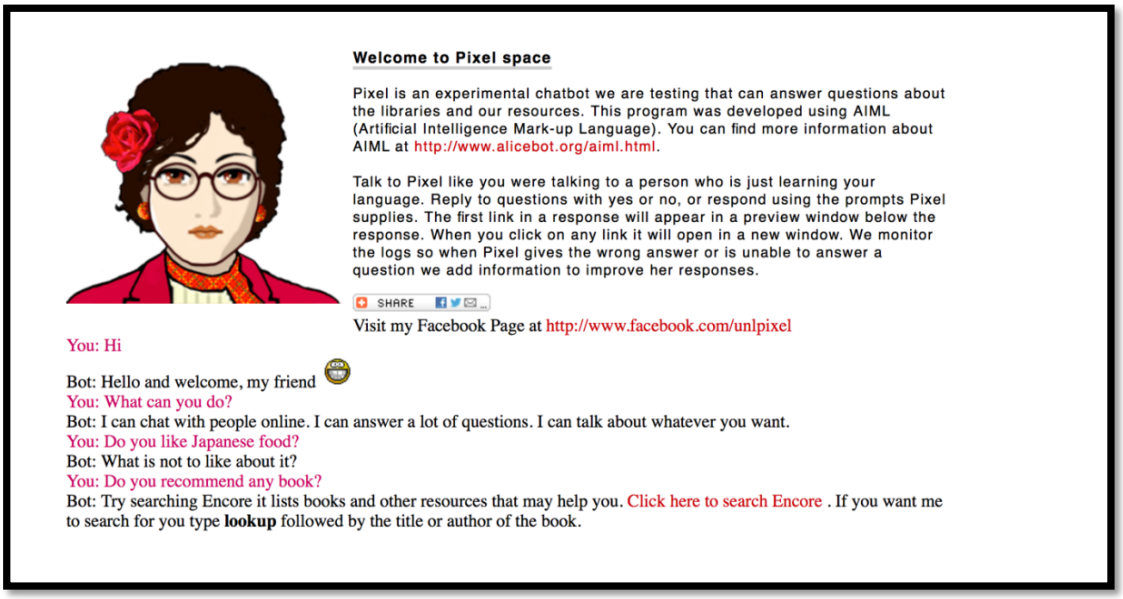
Fonte: Conpet.

3.1.4 Pixel

Pixel, um *chatbot* da Universidade de Nebraska-Lincoln com o objetivo de responder perguntas sobre a biblioteca da universidade e seus recursos, também se destaca (ALLISON, 2012). Foi desenvolvido em PHP e teve o banco de dados criado

a partir da mineração de informações proveniente de sites de biblioteca e registros de bate-papo. Isso tornou possível que o *chatbot* consiga responder as mais variadas perguntas e ficar disponível em tempo integral. A Figura 4 mostra um exemplo de diálogo com Pixel.

Figura 4 – Diálogo com Pixel.



Fonte: Adaptado Allison.

3.1.5 Mec

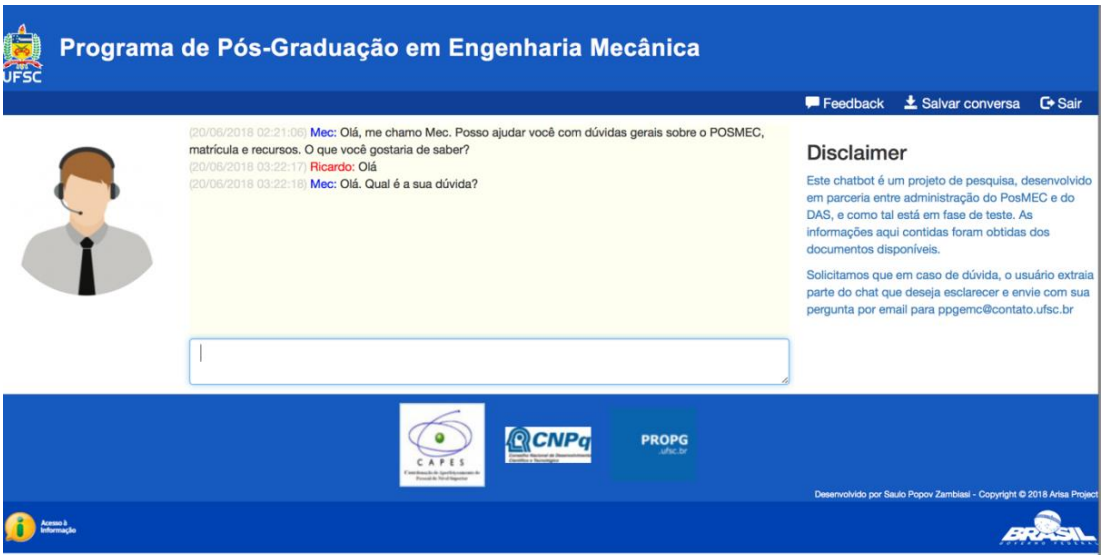
Mec é um *chatbot* de atendimento implementado para a secretaria do Programa de Pós-Graduação em Engenharia Mecânica da UFSC (PosMec). A sua base de conhecimento foi construída a partir de informações fornecidas pelos colaboradores do programa. Foram informadas as perguntas mais frequentes feitas à secretaria e também disponibilizado um histórico de e-mails com dúvidas para pesquisa.

Os dados referentes aos assuntos escolhidos foram retirados, em sua maioria, da *webpage* do PosMec e também de um documento de título “Manual do Bolsista do Programa de Pós-Graduação em Engenharia Mecânica”. Além dessas

fontes, diversos dados foram fornecidos diretamente pelos colaborados do programa.

A base de conhecimento foi implementada no módulo de *chatbot* do Software Assistente Pessoal Arisa. Na base de conhecimento estavam inseridos contextos referentes a dúvidas gerais sobre o PosMec, processo de matrícula, solicitação de recursos e inscrição para o mestrado. Para a interação do *chatbot* com o usuário foi utilizado uma página *web*, como mostra a Figura 5.

Figura 5 – Interface web do Mec.



Fonte: O autor.

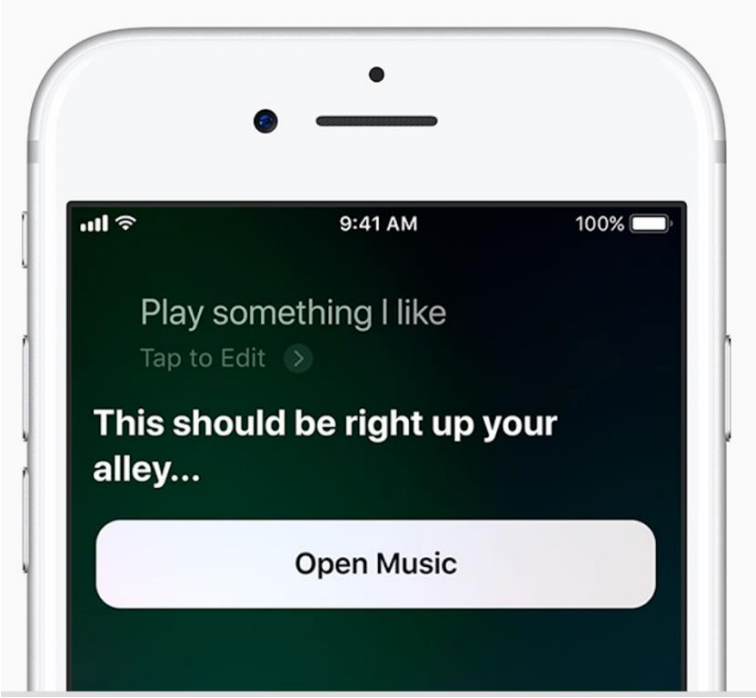
3.1.6 Siri

Siri é um assistente pessoal proprietário da Apple Inc. com diversas funcionalidades, capaz de realizar tarefas através de comandos de voz. O principal foco desse tipo de assistente é tornar a interação do usuário mais natural. Originalmente seu desenvolvimento foi em Inglês, mas atualmente é possível utiliza-la em português. O diferencial da Siri perante outros assistentes é sua capacidade de processar linguagem natural, o que significa que a Siri é capaz de utilizar da fonologia, morfologia, sentido léxico, sintático, semântico e pragmático para

interpretar aquilo que lhe foi dito além de fazer uma análise de significado tomando o texto como um todo (APPLE).

O assistente é capaz de assimilar características da voz e do dialeto falado pelo usuário, tornando assim sua capacidade de compreensão melhor através do uso contínuo. Além das funcionalidades relacionadas a linguagem a Siri também é capaz de enviar mensagens de texto, e-mails e interagir com outras aplicações instaladas no mesmo dispositivo (APPLE).

Figura 6 – Siri.



Fonte: Apple.

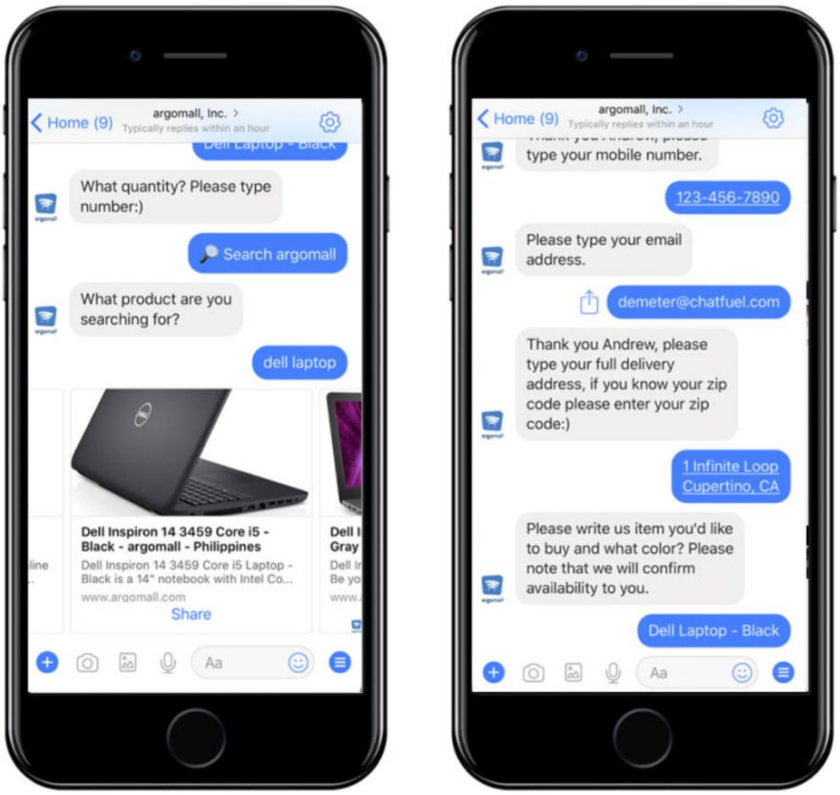
3.1.7 Argomall

Argomall é uma loja virtual de produtos eletrônicos com sede nas Filipinas. Eles possuem um *bot* que tem como função: fazer a venda de produtos, sanar dúvidas sobre produtos, fornecer informações sobre o rastreio de entrega e responder dúvidas gerais sobre a loja.

Poucos meses após o seu lançamento, o *bot* já se tornou uma ferramenta muito importante para a empresa, com alto retorno de investimento (READ, 2019). O

bot pode ser usado diretamente através do Facebook Messenger ou da página da Argomall.

Figura 7 – Argomall



Fonte: SUMO.

3.2 Artificial Intelligence Markup Language (AIML)

Artificial Intelligence Markup Language (AIML) é uma linguagem de marcação fundamentada na linguagem *eXtensible Markup Language* (XML). Ela foi criada com a finalidade de gerar diálogos semelhantes à escrita natural do ser humano através de programas de computador, simulando então a inteligência humana (LEONHARDT, 2003).

É uma linguagem de fácil entendimento porque é baseada em um conjunto de comandos e marcações simples. É utilizada para programar a base de conhecimento de um *chatbot*, tendo como função interpretar as mensagens

enviadas pelos usuários e selecionar qual a resposta a ser enviada (LEONHARDT, 2003).

Os objetos AIML são formados pelas unidades *category* (categoria) e *topic* (tópico). Essas unidades serão mais detalhadas a seguir.

3.2.1 Category

Category é considerada uma unidade básica de conhecimento que é formada por uma entrada (do usuário), uma saída (resposta dada pelo *bot*) e por um contexto (que é opcional). A entrada é denominada como *pattern* e a saída como *template*, sendo definidas respectivamente pelas *tags* <pattern> e <template> (WALLACE, 2008). A Figura 8 exemplifica o uso dessas *tags*.

Figura 8 – Estrutura da linguagem AIML

```
<aiml>
   <category>
      <pattern> Entrada do usuario </pattern>
      <template> Saida do bot </template>
   </category>
</aiml>
```

Fonte: O autor.

É especificado que o <pattern> precisa ser o primeiro elemento dentro de <category> e também que deve existir apenas um elemento de <pattern> por categoria.

3.2.2 *Topic*

Topic é um elemento que não é obrigatório e contém elementos *category* dentro dele. Um *topic* pode conter uma ou mais categorias, com a necessidade de

um atributo como um nome. Ele fica fora da *tag* <*category*> e representa um grupo de categorias reunidas. O nome do elemento pode ser definido dentro um *template* qualquer (WALLACE, 2008).

3.2.3 Outras *tags*

Outras *tags* relevantes que valem ser mencionadas são:

- <srai>: ela pode ser utilizada dentro da *tag* <template> e tem a função de evitar a repetição de informações, redirecionando a pergunta para outra de mesmo significado.
- <random>: é utilizada para escolher de forma aleatória alguma expressão pelas *tags* .
- <think>: é uma *tag* utilizada para salvar uma variável sem notificar o usuário.
- <set>: tem a função de armazenar uma variável.

A Figura 9 exemplifica o uso das *tags* <srai> e <random>.

Figura 9 – Exemplo de uso de algumas tags.

```
<category>
        <pattern>ADEUS</pattern>
        <!-- comments:Despedida -->
        <!-- link:  -->
        <template>
                <random>
                        <li>Até mais :)</li>
                        <li>Tchau, até breve.</li>
                        <li>Tchau! Obrigado pela conversa.</li>
                </random>
        </template>
</category>

<category>
        <pattern># ADEUS #</pattern>
        <template>
                <srai>ADEUS</srai>
        </template>
</category>

<category>
        <pattern># ATE LOGO #</pattern>
        <template>
                <srai>ADEUS</srai>
        </template>
</category>
```

Fonte: O autor.

3.3 Serviços web

Um serviço web é definido pela W3C (*World Wide Web Consortium*) como um sistema de software projetado para suportar a interoperabilidade entre máquinas através de uma rede. De uma forma mais simplificada, pode-se afirmar que serviços web tem o objetivo de conectar duas aplicações distintas independente das linguagens de programação utilizadas.

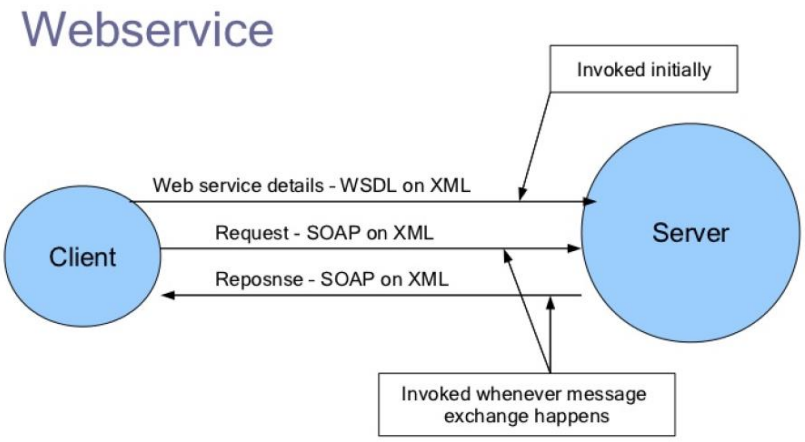
Os serviços web possibilitam que as aplicações enviem e recebam dados no formato XML, uma linguagem universal. Entretanto, a comunicação entre as aplicações deve respeitar protocolos referentes a essa transmissão de dados. Essa integração entre serviços distintos e a padronização do retorno de cada requisição e serviço são os principais motivos para uso de serviços web.

SOAP (*Simple Object Access Protocol*) é um protocolo de mensagem baseado em XML capaz de invocar aplicações remotas por meio de HTTP. Uma mensagem SOAP é um documento XML que contém um *envelope*, que define o conteúdo da mensagem e informa como processá-la.

O WSDL (*Web Services Description Language*) é um padrão de descrição de todos os métodos os serviços web. É um arquivo em XML que descreve a interface dos serviços web, de onde as aplicações que consomem os serviços web extraem informações do WSDL como: parâmetros, nomes de métodos, endereço do serviço e formato de arquivos de entrada e saída (FERREIRA; MOTA, 2014).

A Figura 10 a seguir demonstra a interação entre cliente e servidor SOAP. O cliente faz a leitura do arquivo WSDL no servidor no primeiro contanto, tomando então conhecimento dos parâmetros e métodos e parâmetros do serviço web através do mesmo. Com essas informações, o cliente torna-se apto a fazer uma requisição ao servidor e recebe um retorno esperado.

Figura 10 – Exemplo de protocolo SOAP



Fonte: IndicThreads

4 PROPOSTA

Neste capítulo é abordado a proposta de modelagem e o design do projeto.

4.1 Base de conhecimento

A modelagem da base de conhecimento é uma atividade fundamental para a execução do projeto. Como foi utilizada a plataforma de gerenciamento de assistentes virtuais Arisa Nest para armazenar a base de conhecimento, não é necessário trabalhar com os dados diretamente em AIML. A plataforma conta com uma interface para a entrada de dados de forma muito intuitiva. Os detalhes da implementação estão descritos nos capítulos seguintes. Para a compreensão da criação da base de conhecimento, é importante definir os seguintes termos utilizados por Zambiasi (2017).

- Contexto: é um grupo de diálogos de conversação baseados em um assunto específico.

- Diálogo: servem para identificar se a fala do usuário se encaixa em algum deles para fornecer uma resposta condizente.

- Padrões: contém os padrões de pergunta para as possíveis mensagens enviadas pelo usuário. Se a mensagem se enquadrar nesse padrão, o diálogo relacionado a essa pergunta será utilizado para a resposta.

- Respostas: contém as mensagens enviadas ao usuário caso o diálogo seja selecionado.

4.1.1 Contextos da base de conhecimento

Como o assistente virtual atua em três cenários diferentes, esses cenários são separados em três contextos principais, sendo um para cada área de atuação. Por tratar-se de um protótipo com o objetivo de mostrar funcionalidades da

plataforma, o conteúdo das bases de conhecimento foi limitado apenas ao necessário.

4.1.2 UFSC

O cenário relacionado à UFSC tem como objetivo mostrar a interação do assistente virtual com os serviços web da universidade. Para este projeto, foi liberado o acesso à alguns métodos de leitura de serviços específicos reais. Para deixar o cenário mais completo, outros serviços foram criados foram criados para simular os indisponíveis.

A escolha das funcionalidades para este cenário foi baseada nas atividades que se acredita serem as mais comuns realizadas pelos alunos nos ambientes da UFSC. Existem alguns aplicativos de *smartphones* voltados a alunos da UFSC que possuem boa parte das funções aqui escolhidas, o que contribui para justificar essas escolhas. A própria experiência do autor como aluno da universidade serviu como base para perceber potenciais funcionalidades.

As principais funções propostas são: informar o IAA (Índice de Aproveitamento Semestral Acumulado) do usuário, informar as aulas de um dia específico da semana, informar o cardápio do dia do restaurante universitário, informar a nota de disciplinas cursadas, alterar o e-mail cadastrado, fazer matrícula, fornecer um histórico semestral. Além disso foi também sugerido uma funcionalidade de proatividade do assistente virtual. Todo dia em um determinado horário o AV envia uma mensagem informando se o usuário tem aulas no dia seguinte.

Outra funcionalidade proposta diz respeito a função de Coordenador de Estágios e PFC do curso de Engenharia de Controle e Automação da UFSC. Uma das funções do Coordenador é analisar diversos critérios de TCEs (Termo de Compromisso de Estágio) de alunos do curso, onde o mesmo precisa procurar muitas informações em diferentes locais. Sugeriu-se então criar uma funcionalidade que possa automatizar toda ou a maior parte deste processo, fazendo as verificações necessárias e confirmando o resultado por e-mail ao coordenador e ao aluno analisado.

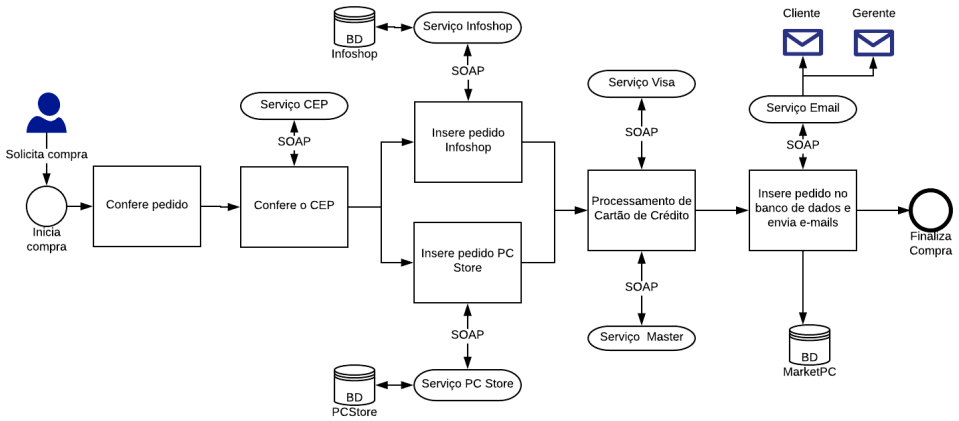
4.1.3 Atendimento ao cliente

Este cenário de atendimento ao cliente tem como objetivo principal automatizar por completo uma operação de venda. O panorama simulado é o de um *e-commerce* que vende produtos de terceiros (Infoshop e PC Store), como um *marketplace* de nome PC Market.

É considerado que o usuário pode ter acesso ao catálogo de produtos para saber o que a loja oferece. A empresa Infoshop vende teclados, mouses e microfones, enquanto a empresa PC Store vende processadores e placas de vídeo. Em resumo, o cliente deve escolher os produtos, inserir seus dados pessoais, inserir os dados de pagamento e no final receber uma confirmação da compra. Em meio a essas operações, o assistente deve fazer diversas chamadas de serviços para auxiliar na venda.

Considerando o fluxo de operações que fazem parte do processo de compra aqui simulado, pode-se dizer que o assistente deve se comportar de forma similar a uma ferramenta de ESB (*Enterprise Service Bus*), pois é quem de certa forma gerência as informações e o curso das atividades. A Figura 11 a seguir ilustra o processo de venda.

Figura 11 – Processo de venda



Fonte: O autor.

É característica de uma ferramenta ESB lidar com diferentes protocolos de comunicação, mas nesse caso o sistema está limitado apenas ao protocolo SOAP. Foi omitido do diagrama algumas interações entre o assistente e o usuário para deixar o diagrama mais claro. Essas interações são mais exploradas no capítulo seguinte.

4.1.4 Operador 4.0

O cenário do operador 4.0 tem como objetivo explorar a forma com que um assistente virtual pode auxiliar um operador de equipamentos industriais em um ambiente inteligente, nos moldes da indústria 4.0.

Duas funcionalidades diferentes são abordadas neste cenário. No primeiro caso o assistente virtual se comporta apenas como um agente passivo, que fica disponível para o operador fazer perguntas relacionadas à operação da máquina, substituindo a necessidade de um manual de usuário tradicional.

Para este caso, possíveis perguntas e respostas devem ser extraídas de algumas seções de um manual de uma máquina enfardadeira, servindo de material para criação da base de conhecimento. É proposto que a base de conhecimento responda perguntas relacionadas aos modos de operação da máquina, a paradas de emergência, a ajustes rápidos na tela, etc.

Para o segundo caso é considerada a existência um cenário específico para que o assistente virtual possa tomar algumas decisões e realizar ações de forma independente, sem a intervenção do operador. Esse cenário é uma adaptação simplificada de um caso mais complexo abordado no artigo *Softbots Supporting the Operator 4.0 at Smart Factory Environments* (RABELO; ROMERO; ZAMBIASI, 2018).

Neste projeto, supõem-se que a enfardadeira utilizada como exemplo possui um sensor que envia uma mensagem ao AV quando o filme plástico está próximo a acabar. Supõem-se também que a duração do filme mesmo após o sensor ser acionado é de 7 dias, possibilitando que a empresa tenha o hábito de comprar uma nova bobina de filme plástico apenas quando se nota o sensor acionado.

Com esse cenário montado, é proposto a implementação de uma funcionalidade para automatizar esse processo de compra de filme plástico a partir do sinal do sensor. O AV deve emitir uma ordem de compra sem interferência do usuário caso um fornecedor pré-registrado como prioritário possua o produto em estoque. Caso contrário, deve ser feito uma busca do produto mais barato entre os outros fornecedores cadastrados e emitir uma ordem de compra para este fornecedor mediante uma autorização do usuário. Em ambos os casos o operador e um supervisor devem ser informador por e-mail.

É importante ressaltar que esta é uma versão bastante simplificada do problema. A simulação acontece somente até o AV tomar a decisão ou for instruído a emitir a ordem de compra. No caso do artigo, é feito a emulação de uma rede industrial para comunicação, tornando o caso mais próximo da realidade.

4.2 Seleção e desenvolvimento de serviços web

A UFSC possui múltiplos serviços web relacionados a diferentes setores da universidade. A partir desses serviços, foi necessário selecionar os mais adequados à proposta do primeiro cenário de atuação do assistente virtual. O serviço fornece os dados do histórico escolar do aluno possui a maior parte dos dados necessários para alimentar as funções deste contexto.

Devido a problemas técnicos e burocráticos citados nos próximos capítulos, foi indispensável implementar serviços web intermediários aos serviços da UFSC, e também implementar serviços web que simulam ou complementam esses serviços.

4.3 Especificação dos requisitos

As decisões para os requisitos do projeto foram feitas com o objetivo de explorar ao máximo os serviços web disponibilizados pela UFSC e a plataforma Arisa Nest. Ao longo do desenvolvimento algumas alterações foram feitas em relação aos requisitos principais.

4.3.1 Requisitos funcionais

De acordo com Sommerville (2007, p.80), os Requisitos Funcionais (RF) “são as declarações de serviço que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas ações”. A seguir estão os requisitos funcionais especificados.

- RF 01: o AV deve responder sobre os três contextos pré-definidos.
- RF 02: o AV deve dialogar de forma similar a um ser humano.
- RF 03: o AV deve enviar mensagens de forma proativa.

4.3.2 Requisitos não funcionais

Os Requisitos não Funcionais (RNF) são as restrições das operações que formam o sistema. Os RNF estão relacionados ao como, quando, onde, de que forma e por quanto tempo as operações se realizam (WAZLAWICK, 2004). A seguir estão os requisitos não funcionais especificados.

- RNF 01: criar uma base de conhecimento e implementar na plataforma Arisa Nest.
- RNF 02: utilizar a linguagem de programação Lua para desenvolvimento de *scripts.*
- RNF 03: o AV deve consumir serviços web para responder às perguntas.
- RNF 04: utilizar a linguagem de programação PHP para criar serviços web.

- RNF 05: utilizar serviços web para realizar consultas num banco de dados MySQL.

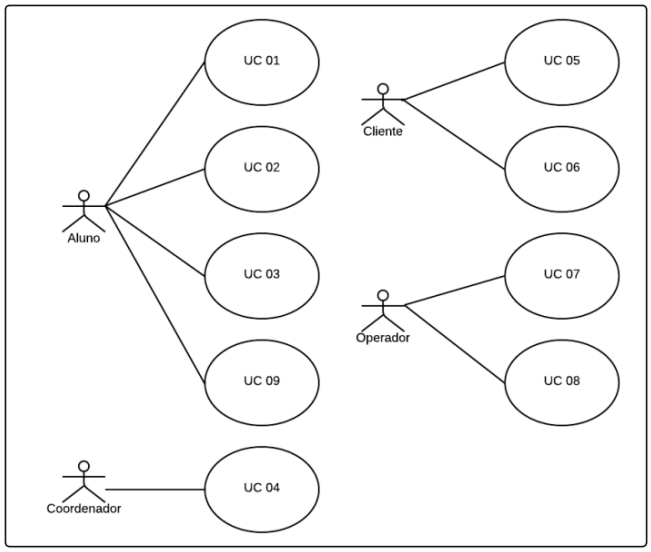
4.4 Casos de uso

Para evitar muitas repetições, alguns casos de uso dos mesmos cenários de atuação serão generalizados pois têm a mesma finalidade prática. Os casos de uso são sintetizados em:

- UC 01: Salvar número de matrícula.
- UC 02: Solicitar funcionalidade básica UFSC (IAA, nota em disciplina, histórico semestral, horário).
- UC 03: Alterar e-mail da UFSC.
- UC 04: Analisar TCE.
- UC 05: Fazer uma compra.
- UC 06: Solicitar funcionalidade básica de atendimento (informação sobre produto, formas de pagamento, contato).
- UC 07: Enviar sinal do sensor.
- UC 08: Solicitar funcionalidade básica do operador 4.0.
- UC 09: Receber mensagem proativa do AV.

A Figura 12 abaixo ilustra o diagrama de casos de uso do sistema simplificado.

Figura 12 – Diagrama de casos de uso.



Fonte: O autor.

4.4.1 UC 01

O usuário do tipo aluno salva o seu número de matrícula através de uma mensagem dentro do contexto UFSC. Relações: RF 01, RF 02, RNF 01, RNF 02.

4.4.2 UC 02

O usuário do tipo aluno faz uma pergunta ao AV que exige apenas o consumo de serviços da UFSC. Relações: RF 01, RF 02, RNF 01, RNF 02, RNF 03, RNF 04.

4.4.3 UC 03

O usuário do tipo aluno faz uma solicitação ao AV que exige o consumo dos serviços criados que acessam banco de dados. Relações: RF 01, RF 02, RNF 01, RNF 02, RNF 03, RNF 04, RNF 05.

4.4.4 UC 04

O usuário do tipo coordenador faz uma solicitação ao AV que exige o consumo dos serviços web da UFSC e também dos serviços web adicionais. Relações: RF 01, RF 02, RNF 01, RNF 02, RNF 03, RNF 04, RNF 05.

4.4.5 UC 05

O usuário do tipo cliente faz uma solicitação ao AV que exige o consumo dos serviços criados que acessam banco de dados. Relações: RF 01, RF 02, RNF 01, RNF 02, RNF 03, RNF 04, RNF 05.

4.4.6 UC 06

O usuário do tipo cliente faz uma pergunta ao AV. Relações: RF 01, RF 02, RNF 01.

4.4.7 UC 07

O usuário do tipo operador faz uma solicitação ao AV que exige o consumo dos serviços criados que acessam banco de dados. Relações: RF 01, RF 02, RNF 01, RNF 02, RNF 03, RNF 04, RNF 05.

4.4.8 UC 08

O usuário do tipo operador faz uma pergunta ao AV. Relações: RF 01, RF 02, RNF 01.

4.4.9 UC 09

O AV envia uma mensagem ou realiza uma ação para o usuário sem ele tenha feito uma requisição. Relações: RF 01, RF 02, RF 03, RNF 01, RNF 02, RNF 03, RNF 04, RNF 05.

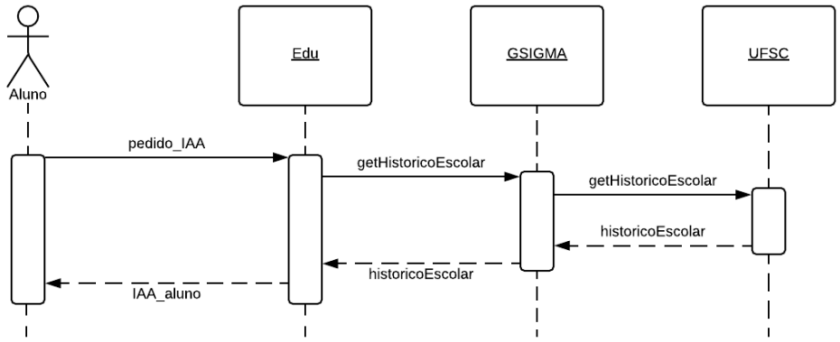
4.5 Diagramas de sequência

De acordo com Wazlawick (2004), “o diagrama de sequência representa a sequência dos eventos de sistema em um cenário de uso”. Um diagrama de sequência pode ser composto por atores e objetos. Estão dispostos no documento apenas alguns diagramas de sequência selecionados, de modo a evitar diagramas muito simples e repetição de casos iguais ou muito parecidos.

4.5.1 Sequência UC 02

O diagrama de sequência UC 02 apresentado a seguir na Figura 13 representa a sequência de eventos que ocorrem quando o usuário do tipo aluno pergunta alguma informação que deve ser retirada dos serviços web da UFSC. Para este exemplo foi utilizado o diálogo em que o aluno pergunta o seu IAA. A sequência é análoga para outros diálogos similares.

Figura 13 – Diagrama de sequência UC 02.



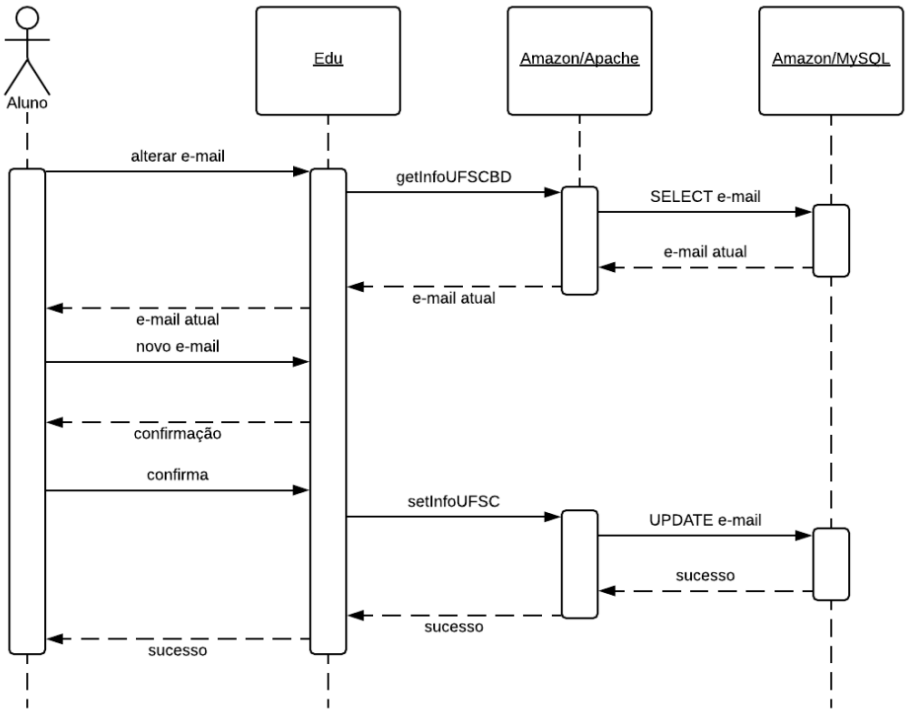
Fonte: O autor.

4.5.2 Sequência UC 03

O diagrama de sequência UC 03 apresentado a seguir na Figura 14, representa a sequência de eventos que ocorrem para fazer uma alteração de e-mail cadastrado na UFSC através de um serviço web simulado que se comunica com um

banco de dados. Nota-se que na primeira interação do usuário o AV já consome um serviço web e consulta um banco de dados para verificar o e-mail atual do aluno.
Após receber a mensagem com o novo e-mail e a autorização para a mudança, o AV consome um segundo serviço web que dessa vez atualiza um dado no banco de dados ao invés de somente ler.

Figura 14 – Diagrama de sequência UC 03.



Fonte: O autor.

5 FERRAMENTAS E TECNOLOGIAS

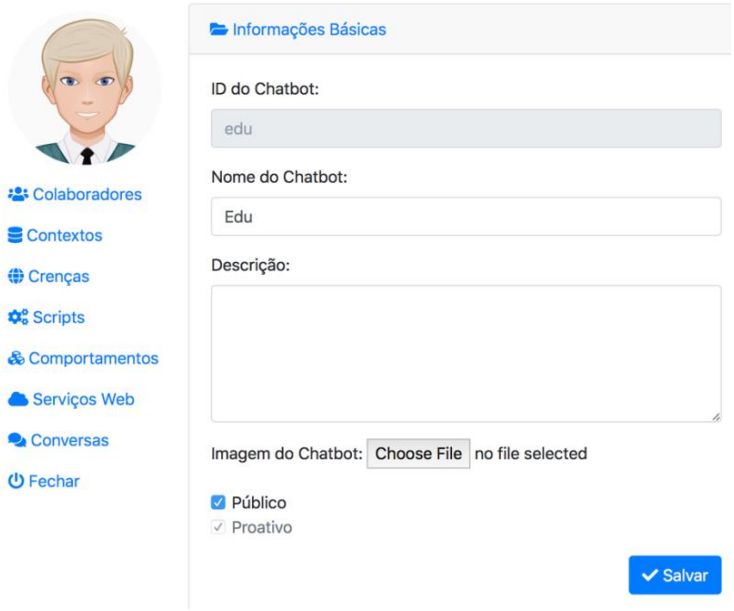
5.1 Arisa Nest

Como já mencionado, foi utilizado a plataforma de gerenciamento de assistentes virtuais Arisa Nest para alimentar a base de conhecimento. Essa seção tem como objetivo explicar como funciona este sistema.

5.1.1 Configuração do Assistente Virtual

Para adicionar um novo assistente virtual na plataforma é necessário ser um usuário do tipo criador. O ID do *chatbot* é a sua identificação única e não pode ser alterado, sendo o mesmo escolhido na criação do *chatbot*. Sendo um usuário cadastrado e com privilégios de escrita, é possível então fazer algumas alterações. O nome e a imagem do *chatbot* são campos muito importantes, pois são eles que definem, de certa forma, um pouco da personalidade do AV. Neste projeto o nome escolhido foi Edu e a sua imagem representa um menino. Pode ser observado na Figura 15 a página de configuração de um *bot* e a imagem utilizado para o Edu.

Figura 15 – Página de configuração do chatbot



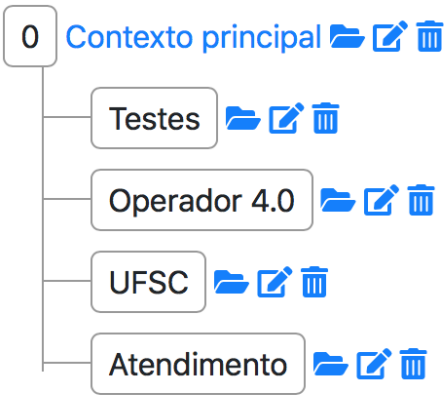
Fonte: O autor.

5.1.2 Contextos

Como o sistema de conversas do Arisa Nest é orientado a contextos, os primeiros elementos a serem criados devem ser os contextos. Apenas dentro dos contextos é que são criados os possíveis diálogos do *chatbot*. Por padrão deve existir pelo menos um contexto, o contexto genérico "**0**". Dentro de cada contexto podem ser adicionados os diálogos referentes a ele.

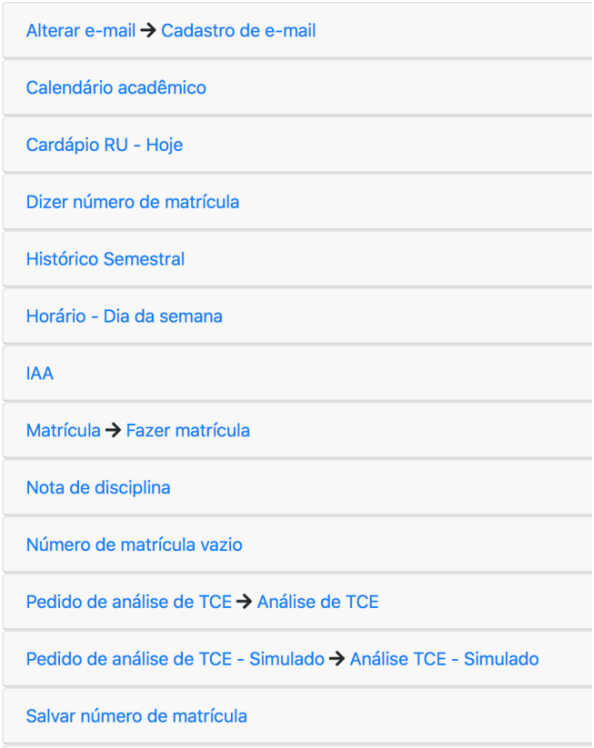
Na Figura 16 pode ser observado os diferentes contextos principais da base de conhecimento do *chatbot* Edu. A Figura 17 mostra alguns dos diferentes diálogos possíveis dentro do contexto "**UFSC**".

Figura 16 – Contextos do Edu



Fonte: O autor

Figura 17 – Diálogos do contexto UFSC



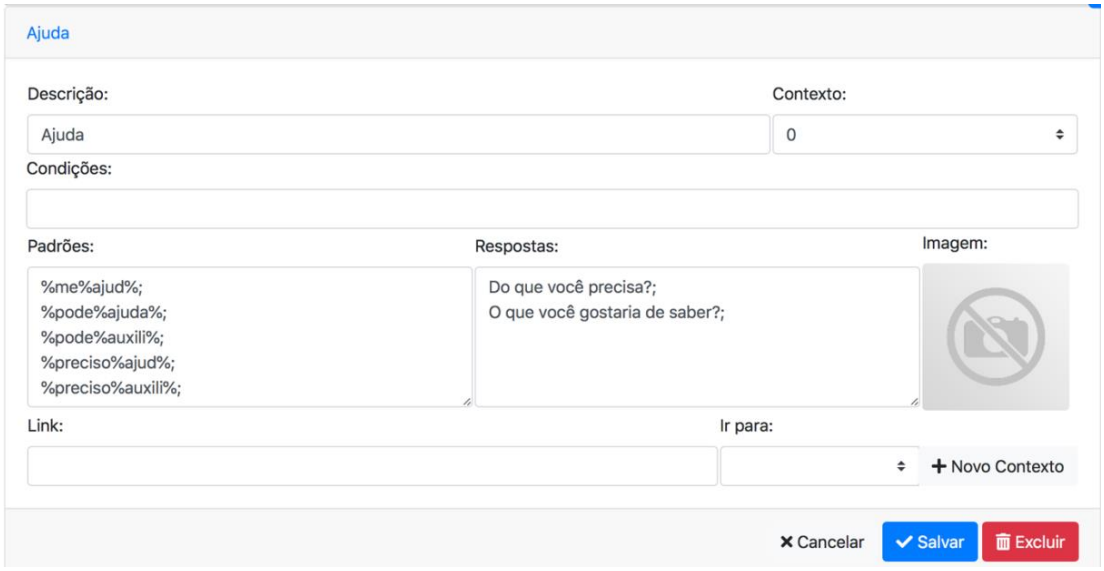
Fonte: O autor.

5.1.3 Diálogos

Os diálogos estão contidos nos contextos, podendo existir diversos diálogos dentro do mesmo contexto. Os diálogos servem para identificar se o texto enviado pelo usuário corresponde com algum padrão de pergunta para então fornecer a resposta desejada.

Os diálogos possuem um campo “Contexto” que identificam a qual contexto esse diálogo está associado. O campo “Link” serve para que o AV envie também um *link* junto com a resposta. Ao entrar num diálogo, o contexto em que o AV se encontra pode ser alterado. “Ir para” representa o contexto para o qual o AV será direcionado. O *bot* também pode responder com uma imagem, que pode ser configurada no campo “Imagem”. Os campos “Padrões”, “Respostas” e “Condições” serão detalhados nas próximas seções. A Figura 18 contém um exemplo da interface de configuração de diálogos.

Figura 18 – Interface de configuração de diálogo



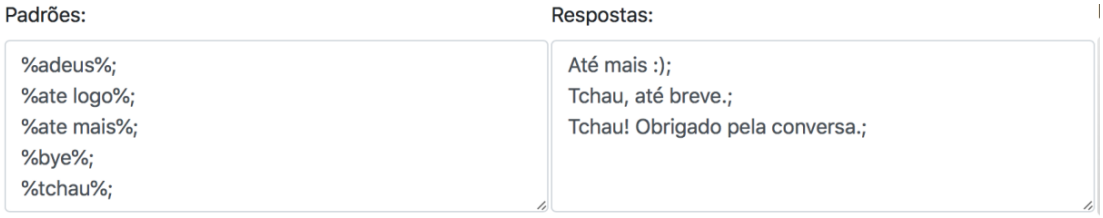
Fonte: O autor.

5.1.4 Padrões

Dentro do campo “Padrões” são inseridos os padrões de pergunta das possíveis mensagens enviadas pelo usuário. Caso a mensagem enviada pelo usuário se enquadre em algum desses padrões, o diálogo será um candidato à resposta. A resposta enviada será uma das frases no campo “Respostas”.

Os diferentes padrões de entrada possíveis são separados por “;“. O símbolo “%” representa um caractere coringa, então quaisquer caracteres antes e depois de “tchau”, representado na Figura 19 abaixo, são aceitos.

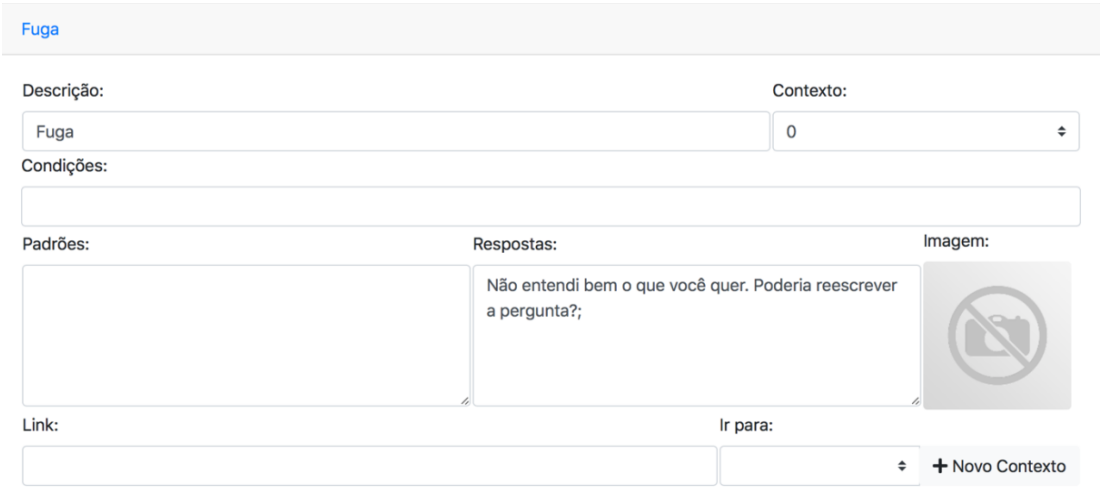
Figura 19 – Exemplo de alternativas de padrões



Fonte: O autor.

Dentro do contexto “**0**” deve existir um diálogo chamado “Fuga”, onde o campo “Padrões” é propositalmente deixado em branco. Caso a entrada do usuário não se enquadre em nenhum padrão de pergunta dentro dos contextos disponíveis, o diálogo ativado é esse de “Fuga”. A resposta então informa ao usuário que a entrada dele não foi compreendida pelo AV. A Figura 20 mostra a estrutura do diálogo “Fuga”.

Figura 20 – Implementação do diálogo “Fuga”



Fonte: O autor.

5.1.5 Respostas

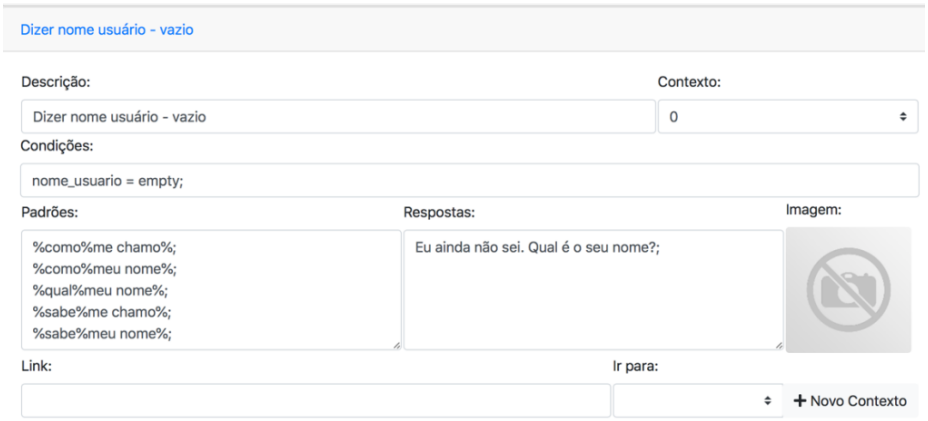
No campo “Respostas” estão contidas as possíveis respostas à mensagem do usuário que se enquadrou em algum padrão de pergunta deste diálogo. Mais de

uma resposta pode ser escrita dentro desse campo, separando as diferentes respostas com “;”. Caso exista mais de uma frase de resposta, ela será escolhida aleatoriamente. Esse recurso é bastante útil para deixar o AV mais flexível. Na Figura 19 pode ser observado que há três respostas diferentes para aquele diálogo.

5.1.6 Condições

“Condições” é um campo de uso opcional, que serve para verificar o valor de uma crença local ou global, que são apresentadas no próximo tópico. Caso o resultado da entrada deste campo seja “verdadeira”, o *bot* vai executar este diálogo, caso contrário isso não ocorre. A Figura 21 contém um exemplo de uso desse campo, onde o *bot* só acessa esse nó de diálogo caso a crença local “nome_usuario” esteja vazia.

Figura 21 – Exemplo do campo “Condições”



Fonte: O autor.

5.1.7 Crenças globais e locais

É possível armazenar variáveis na forma de crenças globais e locais. As crenças globais podem ser acessadas por todos os usuários do AV, uma vez que estas devem conter informações acessíveis a todos. As crenças locais podem ser alteradas e consultadas apenas pelo usuário que elas estão relacionadas. Essas

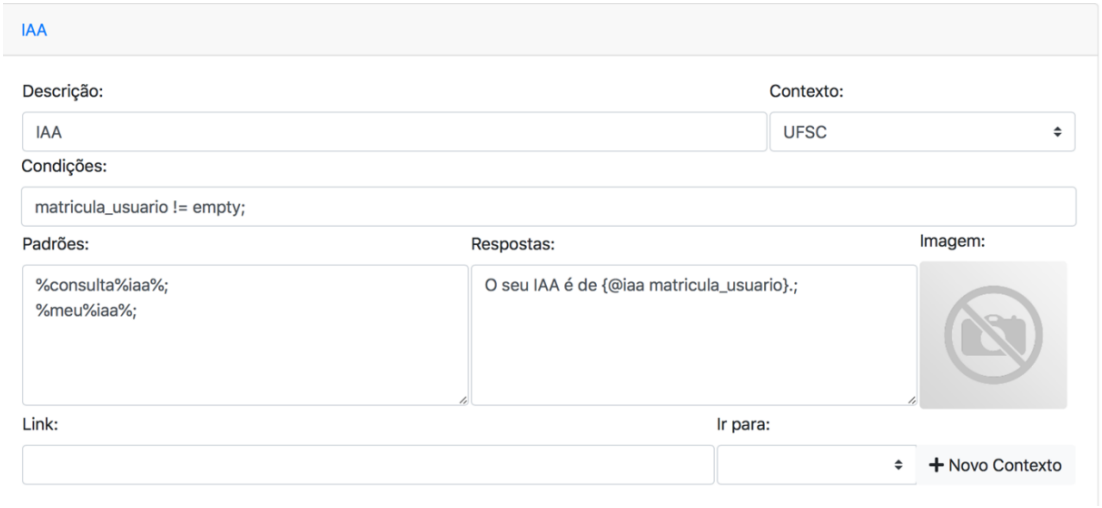
crenças são bastante úteis para armazenar parâmetros após a execução de um *script.*

5.1.8 Scripts

Dentro do campo “Respostas” de um diálogo é possível chamar um *script* cadastrado no AV. Esses *scripts* são implementados em linguagem Lua e podem fazer diversas tarefas como efetuar cálculos, executar algoritmos mais complexos, tratar dados para fornecer melhores respostas ao usuário, chamar serviços web, manipular os dados recebidos de serviços, etc.

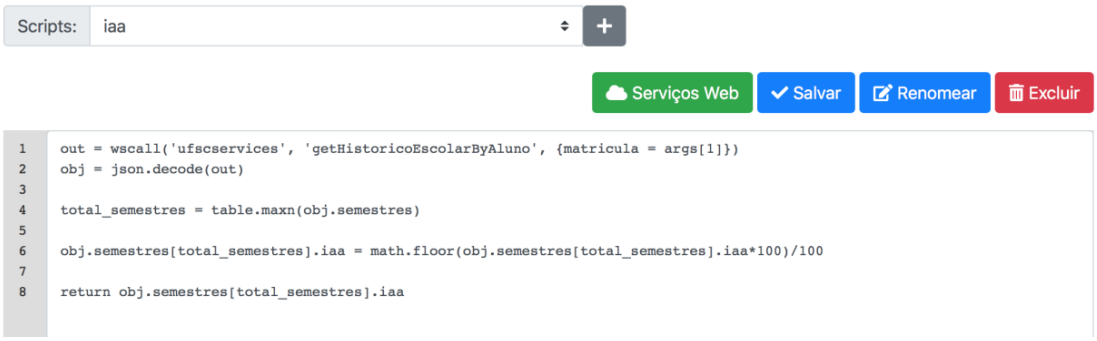
A Figuras 22 mostra um diálogo em que o padrão de resposta executa um *script* chamado “iaa”. Nota-se também que junto a chamada do *script* há outro elemento também. Esse elemento é uma variável que é passada como parâmetro para o script, possibilitando a manipulação desse dado. A Figura 23 apresenta o código Lua implementado para execução desse *script*.

Figura 22 – Diálogo com script



Fonte: O autor.

Figura 23 – Exemplo de script Lua

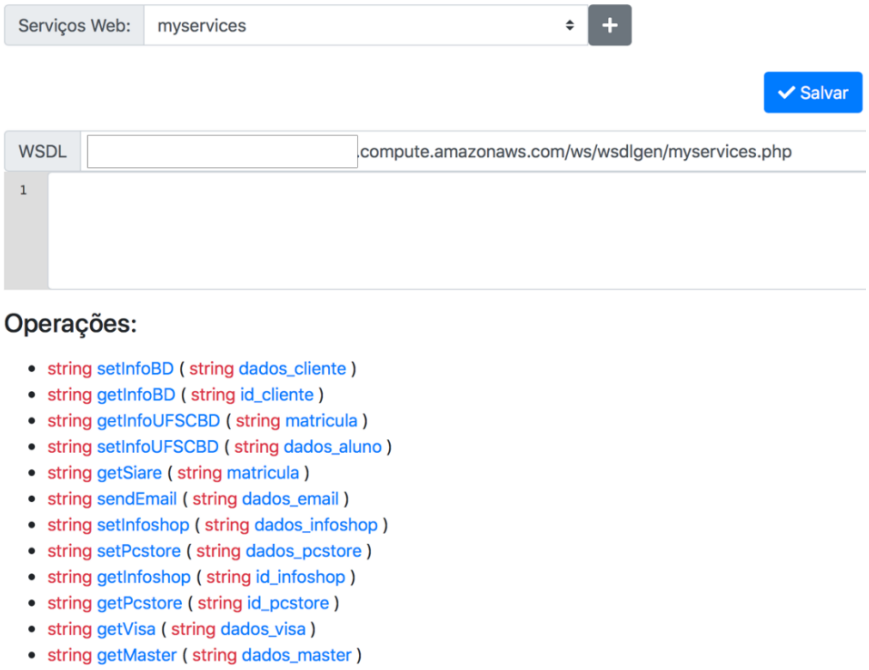


Fonte: O autor.

5.1.9 Serviços web

É possível consumir serviços web após cadastrá-los na plataforma através do endereço desses serviços. Depois disso, basta invocá-los nos na execução dos *scripts,* como no exemplo da Figura 23. Na Figura 24 pode ser observado que após a leitura do arquivo WSDL é possível reconhecer o nome das operações, das variáveis e o tipo de dado utilizado.

Figura 24 – Exemplo de serviço web na plataforma



Fonte: O autor.

5.1.10 Comportamentos

É possível implementar também scripts Lua numa aba de comportamentos. Esses scripts são sempre executados a cada poucos segundos e podem ter as mesmas funcionalidades dos scripts anteriores, podendo consumir serviços web, manipular crenças, etc. A Figura 25 mostra a aba onde são implementados esses comportamentos. O que diferencia um *script* de um comportamento na Arisa Nest, é que os scripts são executados por meio de um diálogo de resposta ao usuário, os comportamentos são executados a cada período de tempo, dando a capacidade de proatividade ao assistente virtual.

Figura 25 – Exemplo de comportamento.

Fonte: O autor.

5.1.11 Dispositivos móveis

O AV também pode ser acessado através do aplicativo de mensagens Telegram. Para isso é necessário fazer o *download* do aplicativo para iOS ou Android, fazer *login* ou criar uma conta, clicar no ícone da lupa e procurar pelo id do AV, clicar no ícone e iniciar uma conversa. Essa integração com o Telegram foi implementada pelo Prof. Dr. Saulo Popov Zambiasi, co-orientador do projeto.

Existe também o aplicativo “Arisa Chat”, em que é possível adicionar os *bots e* assistentes virtuais da plataforma para conversar. Um diferencial do “Arisa Chat” é a possibilidade de usar a voz para interagir com o AV. Essa integração também foi implementada pelo Prof. Dr. Saulo Popov Zambiasi, co-orientador do projeto.

5.2 Amazon Web Services

O Amazon Elastic Computer Cloud (Amazon EC2) é um serviço da Amazon Web Services (AWS) que disponibiliza capacidade computacional na nuvem. Para este projeto, foi criado um ambiente de computação virtual Linux, conhecido com instância. Nesta instância está instalado um servidor HTTP Apache. O servidor Apache disponibiliza páginas e recursos que podem ser acessados e executados por terceiros através da internet.

Também foi empregada uma instância de banco de dados MySQL do Amazon RDS (Amazon Relational Database Service) devido a necessidade de utilizar banco de dados em alguns dos serviços criados. O MySQL é um gerenciador de banco de dados relacional de código aberto amplamente utilizado e é compatível com diversas plataformas.

6 IMPLEMENTAÇÃO

Neste capítulo está descrita a forma que os elementos do projeto foram implementados. Nas seções deste capítulo são apresentados os detalhes da base de conhecimento no Arisa Nest, os serviços web reais e simulados mais importantes, os principais *scripts* e como eles funcionam e exemplos de diálogos explorando algumas funcionalidades dos cenários implementados.

6.1 Serviços web

A proposta de trabalho inicial foi baseada na possibilidade da utilização dos serviços web da UFSC, mas após a autorização do uso de alguns serviços selecionados, surgiu um problema no projeto. Não era de conhecimento do autor e dos orientadores do trabalho que o consumo dos serviços da UFSC, da maneira como eles foram oferecidos, era possível apenas dentro da rede da universidade.

Pelo fato do servidor do Arisa Nest estar alocado externamente à UFSC, o consumo dos serviços web direto pelo AV não foi possível. Para contornar o problema, foi criado um serviço web intermediário no servidor do GSIGMA, que é hospedado na UFSC. Dessa forma, no servidor do GSIGMA foram implementados serviços web intermediários para consumir os serviços reais da UFSC e repassar os dados obtidos para o cliente externo que os invocou.

Para cada operação dos serviços da UFSC que se desejasse utilizar, foi necessário criar uma operação nesse serviço intermediário para consumir chamar a operação real da UFSC e então enviar os dados como resposta à Arisa Nest. As operações criadas foram relacionadas as operações reais “getHistoricoEscolarByAluno” e “getGradeHorarioAluno”, do serviço web referente ao CAGR e “getCardapioSemanaByCodigoRestaurante”, do serviço web do Restaurante Universitário. A operação “getHistoricoEscolarByAluno” fornece a maior parte dos dados necessários para implementar as funcionalidades desejadas. Para criação dos serviços no servidor do GSIGMA foi utilizado uma biblioteca, chamada wsdlgen, implementada em PHP para gerar o arquivo WSDL de autoria do Prof. Dr. Saulo Popov Zambiasi, co-orientador desde projeto.

Além da necessidade de criar esses serviços intermediários, foi também preciso criar os serviços web que simulam os serviços da UFSC que não disponibilizados e também novos serviços para compor os outros cenários especificados.

Esses novos serviços não foram hospedados no servidor do GSIGMA. Essa decisão foi tomada porque as credencias recebidas de acesso ao servidor eram limitadas, não sendo possível fazer alterações. Para tal, foi utilizado o servidor web Apache no ambiente de computação virtual da Amazon citado anteriormente. Essa configuração foi escolhida devido a qualidade e gratuidade desses sistemas para aplicações leves.

Operações do serviço web intermediário aos serviços da UFSC criadas no servidor do GSIGMA:

- getHistoricoEscolarByAluno: tem como entrada a matrícula do aluno e retorna todos os dados do histórico escolar desse aluno.

- getGradeHorarioAluno: tem como entrada a matrícula do aluno e retorna a grade de horários do semestre atual desse aluno.

- getCardapioSemanaByCodigoRestaurante: tem como entrada o código de uma unidade do Restaurante Universitário e retorna o cardápio semanal dessa unidade.

Exemplos de operações mais relevantes do serviço web complementar criado no servidor Amazon:

- sendEmail: operação utilizada em múltiplos cenários. Tem como entrada os dados do destinatário e uma mensagem de e-mail. Envia o e-mail e retorna a informação se a mensagem foi enviada ou não.

- getSiare: operação que simula uma operação real de um serviço da UFSC que não foi disponibilizado. Contém dados simulados referente ao TCE

(Termo de Compromisso de Estágio) de um aluno. Sua entrada é a matrícula do aluno e o retorno são todos os dados armazenados referente a este aluno.

- getVisa: operação que simula parcialmente o que seria um serviço web real de uma operadora de cartões de crédito. Sua entrada são os dados inseridos do cliente e o retorno é o código da transação efetuada.

A Figura 26 abaixo mostra um trecho do código em PHP da operação “setInfoBD”, que serve para armazenar os dados de uma compra finalizada como no UC 05, em que um cliente faz uma compra por intermédio do AV.

Figura 26 – Exemplo de operação

```php
// ----------------------ESCREVER-DADOS-COMPRA---------------------

// Implementa a operação
function setInfoBD($dados_cliente) {

    global $servername, $username, $password, $dbname;

    $var = json_decode($dados_cliente);
    $var[4] = intval($var[4]);
    $var[5] = intval($var[5]);
    $var[8] = intval($var[8]);

    // Create connection
    $conn = mysqli_connect($servername, $username, $password, $dbname);

    $sql = "INSERT INTO marketpc (cliente, email, codigois, codigopc, totalis, totalpc, operadora, cartao, total)
    VALUES ('$var[0]', '$var[1]', '$var[2]', '$var[3]', $var[4], $var[5], '$var[6]', '$var[7]', $var[8])";

    if (mysqli_query($conn, $sql)) {
        $last_id = mysqli_insert_id($conn);
        $resultado = "New record created successfully";
    } else {
        $resultado = "Error: " . $sql . "<br>" . mysqli_error($conn);
    }

    $conn->close();

    return json_encode($last_id);
}

// Informa que a operação acima vai estar no wsdl e formato

$wsdl->operation("setInfoBD", array(
    'dados_cliente' => 'xsd:string'
), array(
    'return' => 'xsd:string'
), 'encoded', "Envia dados do cliente");
```

Fonte: O autor.

6.2 Base de conhecimento

A base de dados foi estruturada baseada em conversação orientada a contextos. Um 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 (ZAMBIASI, 2017).

Existe um contexto "**0**" que tem a função de ser a base inicial para todas as conversas. Esse é o contexto onde acontecem as conversas genéricas como cumprimentos, despedidas e outros assuntos mais gerais (ZAMBIASI, 2017). Neste caso de aplicação, por exemplo, existe um contexto "**UFSC**" que pode ser acessado a partir contexto "**0**". Para isso, deve-se criar um diálogo no contexto "**0**" com perguntas que tentem interpretar que o usuário quer falar sobre a UFSC. Como este é um projeto piloto e há três casos de uso bem diferentes no mesmo AV, foram criadas palavras-chave que levam até o contexto do cenário desejado. Considera-se sempre que o usuário inicia dentro seu contexto específico.

Enquanto no contexto "**0**", todas as perguntas do usuário serão pesquisadas apenas dentro desse contexto. Enquanto no contexto "**UFSC**", por exemplo, primeiro as perguntas são pesquisadas dentro desse contexto, mas caso nenhum diálogo seja encontrado, a busca será feita dento do contexto "**0**", pois é o contexto anterior ao contexto "**UFSC**".

6.3 Exemplos

6.3.1 Exemplos UFSC

6.3.1.1 Salvar número de matrícula

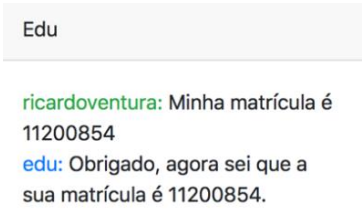
Uma funcionalidade importante para os casos de uso da UFSC é a função "Salvar número de matrícula". Como no momento não há nenhum tipo de cadastro ou procedimento de autenticação referente aos serviços web da UFSC, o usuário precisa informar seu número de matrícula para o assistente executar as buscas requisitadas. A Figura 27 abaixo apresenta como foi implementado este diálogo.

Figura 27 – Diálogo “Salvar número de matrícula”

Fonte: O autor.

Nota-se que quando o usuário envia uma frase como “Minha matrícula é 11200854”, o assistente irá salvar o número “11200854” como uma crença (ou variável) local, que pode ser usada posteriormente no campo condições, no campo respostas ou em *scripts*. A Figura 28 mostra essa ocorrência na interface de conversação.

Figura 28 – Conversa da funcionalidade “Salvar Matrícula”

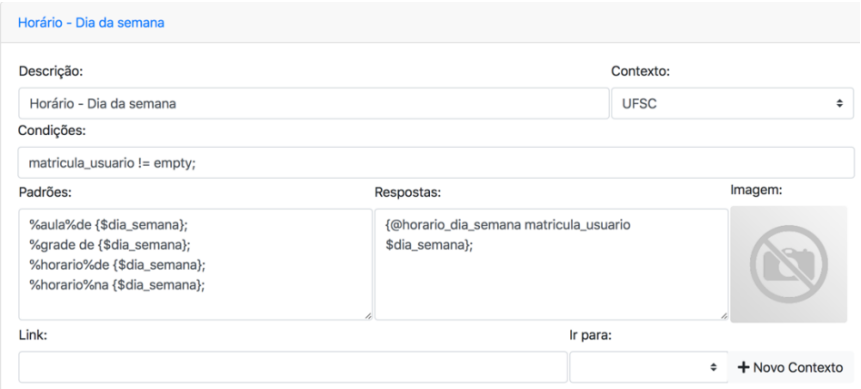


Fonte: O autor.

6.3.1.2 Horário – Dia da semana

A funcionalidade “Horário – Dia da semana” tem como objetivo informar ao usuário quando for solicitado, a grade de horários referente ao dia informado. Ao reconhecer um dos padrões de entrada para esse diálogo, o assistente executa um *script* e retorna o resultado para o usuário. A Figura 29 mostra como este diálogo está implementado.

Figura 29 – Diálogo “Horário – Dia da semana”

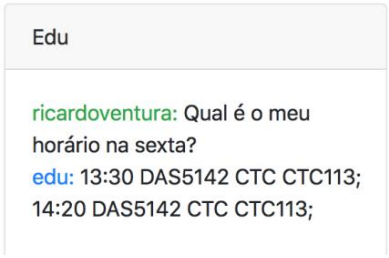


Fonte: O autor.

É possível observar neste diálogo que diversos elementos do Arisa Nest são utilizados. O campo condições é utilizado para impedir que o usuário acesse esse diálogo caso ele não tenha um número de matrícula cadastrado. Se o aluno não possuir um número de matrícula cadastrado, o AV entrará na condição de outro diálogo implementado chamado “Número de matrícula vazio”, que informará ao usuário que ele deve inserir o seu número de matrícula.

Caso o aluno possua um número de matrícula preenchido, o AV irá executar o *script* “horario_dia_semana”, e usará como entrada a crença local “matricula_usuario” e a variável “dia_semana” recebida através da entrada do aluno. O *script* tem como função verificar o dia da semana recebido, chamar o serviço web “getGradeHorarioAluno”, procurar os horários do dia especificado, tratar os dados e retornar ao aluno. A Figura 30 demonstra essa ocorrência na interface de conversação.

Figura 30 – Conversa “Horário – Dia da semana”

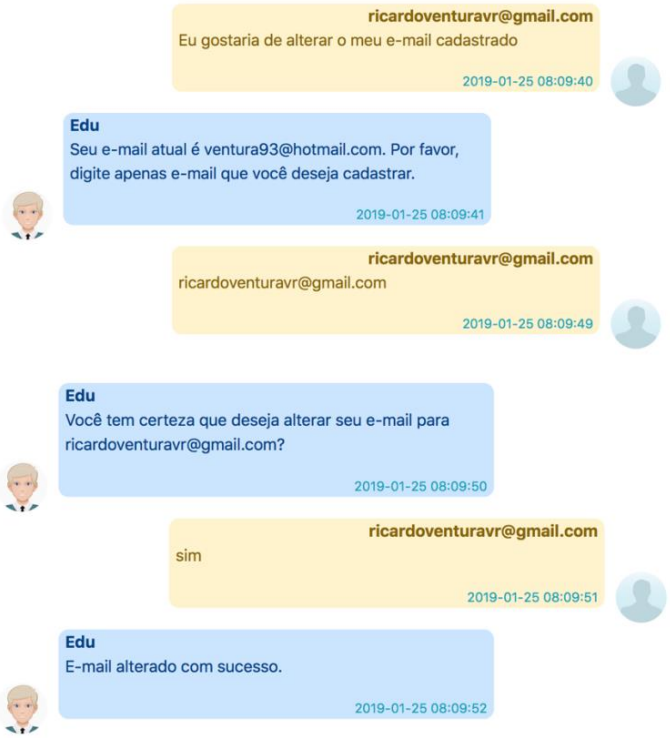


Fonte: O autor.

6.3.1.3 Alterar e-mail

Essa funcionalidade, representada no UC 03, utiliza operações do serviço web simulado criado que têm como função ler e atualizar o campo “e-mail” de um banco de dados de alunos. Assim como a funcionalidade anterior as outras funcionalidades que precisam da variável “matricula_usuario”, ela é condicionada à não nulidade dessa variável. Nota-se que, para fins de simplificação, a condição não foi considerada no diagrama de sequência da Figura 31. A Figura y demonstra essa funcionalidade no registro de *logs* do AV.

Figura 31 – Diálogo com alteração de e-mail



Fonte: O autor.

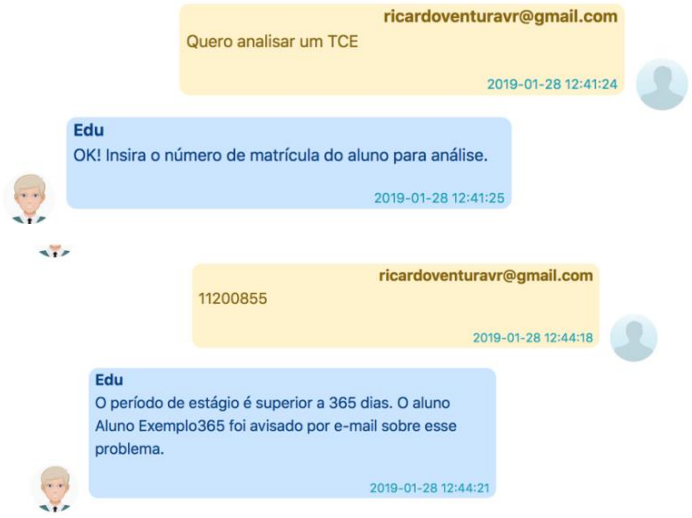
6.3.1.4 Análise de TCE

A tarefa de análise de TCE é a função mais complexa implementada neste cenário. Existe uma operação “getInfoTCEbyMatriculaGraduacao” no serviço web da UFSC “SIAREService” retorna dados do TCE do aluno pesquisado. Esse serviço não autorizado para uso, então foi necessário criar uma operação similar a essa operação já existente, criando um banco de dados de alunos com os mesmos tipos informação. Também só é permitida a consulta de dados em relação a matrícula do autor, então para testes de casos específicos, também foi necessário criar uma operação que faz a leitura de um banco de dados similar aos dados retornados pela operação real da UFSC “getHistoricoEscolarByAluno”.

O *script* responsável por essa tarefa funciona, resumidamente, da seguinte forma: o AV recebe o número de matrícula do aluno a ser analisado, consome os serviços simulados “getSiare” e “getHistorico” e analisa passo a passo diversos critérios que classificam o TCE como válido ou não. Ao ser identificado um critério de invalidade, o algoritmo, ao consumir o serviço “sendEmail”, envi-a um e-mail ao aluno e ao coordenador informando sobre o erro. O AV também responde na tela uma versão da mensagem enviada por e-mail. Caso não tenha nenhum problema, os dados são confirmados por e-mail com cópias para o aluno e o coordenador.

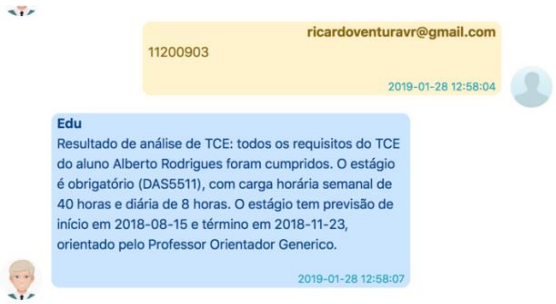
Na Figura 32 é exemplificado uma conversa em que o aluno analisado não atendeu aos critérios necessários. Na Figura 33, o outro aluno analisado atende aos requisitos. A Figura 34 exibe o e-mail enviado ao coordenador.

Figura 32 – Análise TCE insuficiente



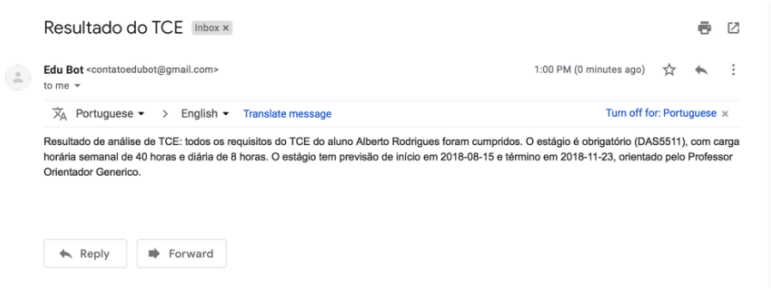
Fonte: O autor.

Figura 33 – Análise TCE suficiente



Fonte: O autor.

Figura 34 – E-mail análise TCE



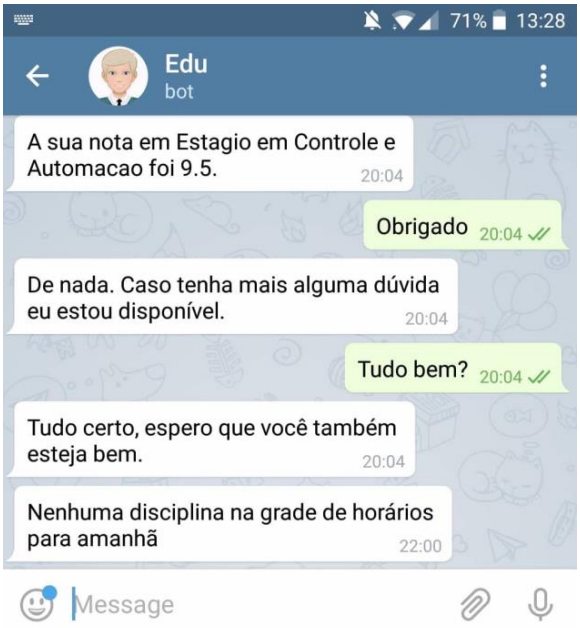
Fonte: O autor.

6.3.1.5 Comportamento: Alerta de aula

O comportamento “Alerta de aula” é uma funcionalidade que exemplifica e exibe o potencial da plataforma Arisa Nest, apresentando um caso em que mostra que o AV pode ser programado para agir proativamente. Um comportamento nada mais é do que um *script* que é executado repetidamente, sem depender de entradas do usuário. Os comportamentos encontram-se num menu separado aos diálogos e aos *scripts* padrões.

O comportamento implementado nesse caso de uso do AV é bastante similar a função já mencionada “Horário – Dia da semana”. O *script*, enquanto ativado, entra em execução a cada 5 segundos. A função do algoritmo é verificar se o horário atual da execução está entre 20:00:00 e 20:00:05. Caso o horário buscado estiver entre esse intervalo, o AV consome o serviço “getGradeHorarioAluno” e envia ao aluno, por meio do seu contato do Telegram, a grade de horários do dia seguinte. Caso o aluno não tenha aulas no dia seguinte, o AV notifica esse status também. A Figura 35 mostra que a mensagem chegou ao destinatário.

Figura 35 – Alerta de aula

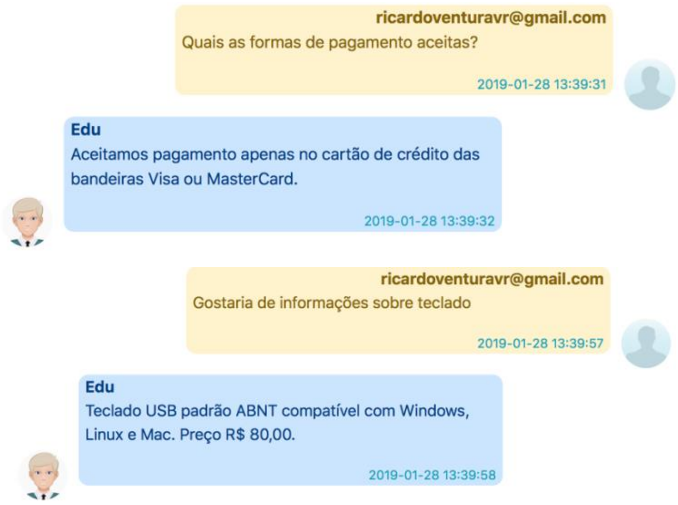


Fonte: O autor.

6.3.2 Exemplos Atendimento

Para este cenário de atuação de AV foram implementados alguns diálogos genéricos que não necessitam da chamada de serviços web apenas para direcionar o AV mais próximo a uma implantação real. Alguns desses diálogos podem ser vistos na Figura 36.

Figura 36 – Exemplo de atendimento

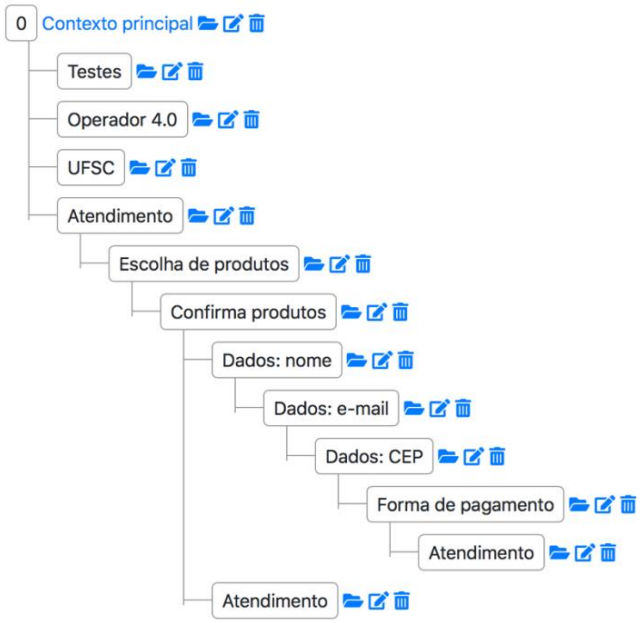


Fonte: O autor.

O objetivo principal desse cenário é apresentar uma funcionalidade completa de venda de mercadoria, onde o AV torna-se uma espécie de atendente de uma loja virtual. Para tal função ser executada sem falhas, é necessário o consumo de diversos serviços (getCEP, getInfoBD, setInfoBD, sendEmail, setInfoshop, setPCStore, getInfoshop, getPCStore, getVisa, getMaster).

A funcionalidade, como mostrada no modelo da Figura 11, necessita executar uma sequência de ações pré-estabelecidas para chegar no resultado final da venda. No Arisa Nest, isso pôde ser implementado de uma forma em que, depois de cada passo concluído na sequência, a conversa é direcionada para um sub-contexto, onde o próximo passo pode ser executado. A Figura 37 abaixo ilustra a forma que isto é arquitetado no Arisa Nest.

Figura 37 – Contextos e sub-contextos de Atendimento



Fonte: O autor.

Registro de conversa:

ricardoventuravr@gmail.com: oi

Edu: Oi. No que posso te ajudar?

ricardoventuravr@gmail.com: Quero fazer um pedido

Edu: Por favor, digite os produtos e a quantidade que você gostaria de comprar. Exemplo: "Teclado 1, microfone 6".

ricardoventuravr@gmail.com: Microfone 2, Processador 1, Placa de video 1, Teclado 5, Mouse 5

Edu: Seu pedido foi: 5 teclado(s), 5 mouse(s), 2 microfone(s), 1 processador(es), 1 placa(s) de video. Deseja prosseguir com a compra destes produtos? Valor total: R$ 1800.00.

ricardoventuravr@gmail.com: Sim

Edu: OK! Informe seu nome completo para dar prosseguimento ao seu pedido.

ricardoventuravr@gmail.com: Ricardo Ventura

Edu: Digite agora o seu endereço de e-mail.

ricardoventuravr@gmail.com: ricardoventuravr@gmail.com

Edu: Certo, agora eu preciso do CEP do endereço de entrega como no exemplo: 88180000

ricardoventuravr@gmail.com: 88037000

Edu: Obrigado! Agora eu preciso que você informe a bandeira do cartão utilizado para pagamento (Visa ou Mastercard), juntamente com o número do cartão e sua data de validade.

ricardoventuravr@gmail.com: Visa 4500123412341234 12/20

Edu: Compra realizada com sucesso! Você receberá em breve um e-mail com a sua ordem de compra. A nota fiscal será enviada assim que a sua compra for processada. Obrigado por comprar conosco!

ricardoventuravr@gmail.com: Obrigado

Edu: De nada. Caso tenha mais alguma dúvida eu estou disponível.

A Figura 38 mostra e-mail enviado ao cliente. Um e-mail com outros detalhes também é enviado ao gerente da loja.

Figura 38 – E-mail de compra ao cliente

Edu Bot <contatoedubot@gmail.com>
para eu

Olá Ricardo Ventura, obrigado por comprar na MarketPC! Segue o resumo da sua ordem de compra:

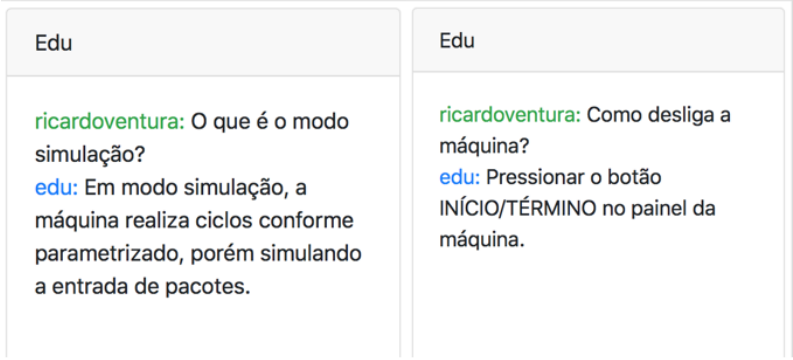
Produto	Quantidade	Valor unitário	Valor total
Teclado	5	R$ 80.00	R$ 400.00
Mouse	5	R$ 100.00	R$ 500.00
Microfone	2	R$ 50.00	R$ 100.00
Processador	1	R$ 300.00	R$ 300.00
Placa de video	1	R$ 500.00	R$ 500.00
Total da compra			R$ 1800.00

Fonte: O autor.

6.3.3 Exemplos Operador

Para este cenário de atuação de AV foram implementados alguns diálogos genéricos que não necessitam da chamada de serviços web apenas para direcionar o AV mais próximo a uma implantação real. Alguns desses diálogos podem ser vistos na Figura x.

Figura 39 – Exemplo de auxílio ao operador



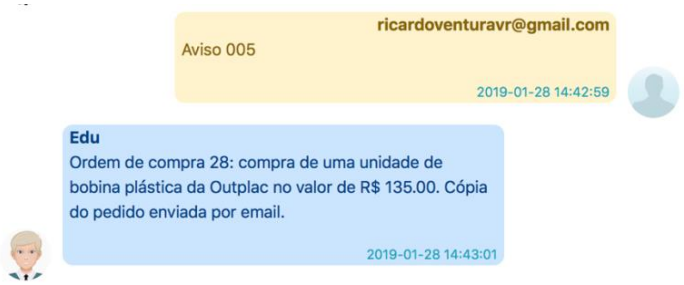
Fonte: O autor.

O principal objetivo da implementação desse cenário de aplicação é explorar um conceito em que o AV tome algumas decisões sem a necessidade da intervenção do operador. A funcionalidade foi desenhada da seguinte forma: quando o sensor detectar que o filme está próximo de acabar, o AV recebe uma mensagem informando a situação (mensagem: Aviso 005) e executa um *script*.

Ao executar esse *script*, o AV tem acesso a uma lista de fornecedores e procura emitir uma ordem de compra a um fornecedor configurado como prioritário. Caso for reconhecido que este fornecedor possui o produto em estoque, uma ordem de compra é feita e o operador e supervisores são informados por e-mail. Se o fornecedor prioritário não possuir o produto em estoque, é feito então uma busca do produto mais barato entre os outros fornecedores. Após isso o AV informa a situação ao operador e pede autorização para dar prosseguimento a essa compra ou não.

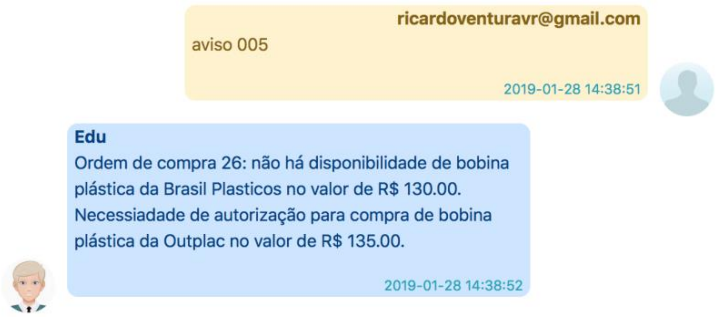
Na primeira situação, Figura 40, o fornecedor prioritário é a empresa Outplac e ela possui o produto em estoque, então, ao receber o sinal do sensor, o AV toma a decisão de fazer uma ordem de compra e apenas avisa o operador, enviar uma cópia do pedido por e-mail ao gerente. Na segunda situação, Figura 41, o fornecedor prioritário está configurado como a empresa Brasil Plásticos, mas ela não possui o produto em estoque. Assim o AV procura na base de fornecedores da empresa o fornecedor com o produto mais barato e pede autorização para emitir a ordem de compra.

Figura 40 – Diálogo: de compra efetuada



Fonte: O autor.

Figura 41 – Diálogo: autorização para ordem de compra



Fonte: O autor.

A implementação é, de certa forma, simples. Após receber o sinal, que nesse caso é simulado como uma entrada no sistema, o AV executa um *script* que faz a leitura do banco de dados de fornecedores através da chamada de um serviço. Com os dados armazenados, o algoritmo procura pelo fornecedor prioritário e verifica se ele possui o produto em estoque. A consequência desse resultado é a execução de uma ordem de compra ou não, seguindo as diretrizes pré-programadas.

7 ANÁLISE DE RESULTADOS

A proposta inicial de implementação deste projeto era a criação de um assistente virtual para alunos e professores da UFSC com completa integração com os sistemas da universidade. Por motivos técnicos e burocráticos, um projeto totalmente focado na UFSC passou a ser inviável. A partir dessa situação, foi modelado um novo sistema proposto a atuar em diferentes áreas com novos objetivos.

De um ponto de vista de implementação, é possível afirmar de que os resultados alcançaram os objetivos esperados. A proposta de criar um assistente virtual capaz de dialogar e resolver tarefas em três diferentes contextos previamente escolhidos se mostrou funcional para as situações testadas.

Idealmente, para dar suporte à análise de resultados a projetos de software dessa natureza, o assistente virtual deveria ter sido disponibilizado para ser testado por um grupo específico. Neste projeto não houve tempo hábil para avaliação com usuários e também, no momento em que se encontra o projeto, não seria viável por questões de segurança de dados. Após a fase de testes, poderia ser aplicado um questionário para melhor avaliar o desempenho do AV.

Alguns exemplos de perguntas relevantes são:

- O assistente virtual entendeu os questionamentos?
- O assistente virtual respondeu aos questionamentos corretamente?
- O assistente virtual tornou mais fácil o acesso as informações?
- O assistente virtual tornou mais rápido o acesso as informações?

Além do questionário aplicado, os padrões de entrada nos diálogos não foram muito explorados devido ao fato de ser um projeto fechado, mas caso o projeto for a ser testado por outros usuários, é bastante importante ampliar as possibilidades de padrões de entrada. O próprio *feedback* dos usuários e os

registros das conversas já fornecem um ótimo material de pesquisa para melhorar esse aspecto.

Algumas tarefas como a análise de TCEs, se corretamente implementadas, certamente garantem agilidade no processo, pois não existe, ainda, outra forma de fazer isso além de manualmente. Outros tipos de tarefas como a de consultar o cardápio do restaurante universitário podem ser feitas de diversas maneiras, então não é possível afirmar, sem uma pesquisa prévia, o impacto do AV nesse tipo de atividade.

Todos as funcionalidades do Arisa Nest testadas neste projeto se mostram bastante robustas e complementares entre si. A possibilidade de consumir serviços web, executar *scripts,* armazenar crenças globais, armazenar crenças globais e demais elementos da plataforma permite uma grande variedade de implementações em diferentes cenários de aplicação.

O aspecto de segurança não foi abordado no projeto. Da forma como os serviços estão implementados, não há critérios de autenticação sofisticados para impedir que os serviços externos criados sejam executados por terceiros. Antes de aperfeiçoar os cenários de atuação já existentes ou desenvolver novos cenários, é interessante focar em aspectos de segurança primeiro.

Ainda por se tratar de apenas uma proposta de aplicação, não foi investido tempo para tratar de todos os problemas e erros de entrada de dados. Apenas os requisitos e resultados mais relevantes foram aprimorados completamente.

8 CONSIDERAÇÕES FINAIS E PERSPECTIVAS

O presente projeto visou o desenvolvimento de um assistente virtual com propriedades de *chatbot* e automação de processos de negócio. Trata-se de uma evolução de um projeto anterior de desenvolvimento de *chatbots* para secretarias de curso de universidades.

Mesmo após uma alteração de foco no trabalho, foi possível chegar a um resultado interessante, mostrando o potencial das tecnologias utilizadas. Para uma futura retomada do projeto, é importante utilizar as ferramentas mais próximas da realidade, evitando simulações. Apenas três operações de serviços web reais foram utilizadas, os outros serviços eram apenas simulados. Deve ser buscada uma maior integração com sistemas reais de outras plataformas, outras APIs.

Não é possível apontar com muita precisão os impactos que o projeto pode trazer apenas com os resultados obtidos no trabalho, mas certamente um próximo passo do trabalho pode permitir que isso seja analisado.

Espera-se que este projeto possa servir como motivação e fonte de informações para novas aplicações neste mesmo tema e também na mesma plataforma. A possível notar uma considerável evolução do projeto de *chatbot* Mec para o projeto atual do assistente pessoal Edu. Almeja-se uma evolução ainda mais significante para os próximos projetos executados.

REFERÊNCIAS

APPLE. <https://www.apple.com/siri/> Acesso em: 10 de janeiro de 2019

ALLISON, D. **Chatbot: Artificial intelligence.** Reference, Libraries, Library users, Library facilities, Library Hi Tech, 2012, Vol.30 (1), p. 95-107.

COPPIN, Ben. **Inteligência Artificial**. Rio de Janeiro: Ltc, 2015. 636 p.

DEMO, Pedro. **Pesquisa e construção de conhecimento.** Rio de Janeiro: Tempo Brasileiro, 1996.

EXAME. **Chatbot é aposta para redução de custos de atendimento.** Disponível em: <https://exame.abril.com.br/negocios/dino/chatbot-e-aposta-para-reducao-de-custos-de-atendimento/> Acesso em: 8 de junho de 2018.

FERREIRA, Cleber; MOTA, Roberto. **Comparando aplicação web service rest e soap.** Trabalho de Conclusão de Curso, Universidade Paranaense Unipar, Paranavaí, 2014.

FOSSATI, Matheus Canali; RABELLO, Roberto dos Santos; MARCHI, Ana Carolina Bertoletti de. **AGEbot: um chatterbot em AIML voltado para responder questões sobre Epilepsia.** Passo Fundo: 2011.

GIL, A. C. **Como elaborar Projetos de Pesquisa**. São Paulo: Editora Atlas S.A., 2010.

HOYLE, M.A. and LUEG, C. **Open Sesame!: A Look at Personal Assistants.** Proceedings of the International Conference on the Practical Application of Intelligent Agents (PAAM97), London, 1997, 51 60, 1997.

JÄRVINEN, P**. Action Research is Similar to Design Science**. Quality & Quantity, n. 41, Springer. 2007, p.37-54.

JÄRVINEN, P**. On Research Methods. Tampere**, Finlândia: Opinpajan Kirja 2004.

KRAUS, Helton Machado. **Protótipo de um Chatterbot para área imobiliária integrando a tecnologia de raciocínio baseado em casos.** 2007. 131 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, 2007.

LAUREANO, Eduardo A. G. C. **ConsulBot – Um Chatterbot Consultor para Ambientes Virtuais de Estudo na Internet**. 1999. Disponível em: <http://www.di.ufpe.br/~tg/1999-1/eagcl.doc> Acesso em: 13 de junho de 2018.

LEONHARDT, Michelle; NEISSE, Ricardo, TAROUCO; ROCKENBACH, Liane Margarida. Meara: **Um Chatterbot temático para uso em ambiente educacional**. Porto Alegre, 2003.

MANFIO, Edio Roberto; MORENO, Fábio. **A EVOLUÇÃO DOS CHATTERBOTS: PLN, I.A. E DIFUSÃO CULTURAL**. Revista e-f@tec, Garça, v. 5, n. 1, p.1-8, 2015. Disponível em: <http://fatecgarca.edu.br/revista/volume5/artigos/1.pdf> Acesso em: 10 de junho 2018.

MICHAELL, T.; et all. **Experience With a Learning Personal Assistant**. Communications of the ACM, July, 1994.

MIKIC, F. et al. **Using tags in an AIML-Based chatterbot to improve its knowledge**. Computer Science, 2012, Vol. 13 (2), p. 123. Disponível em: <http://journals.bg.agh.edu.pl/COMPUTER/2012.13.02/cs.2012.13.02.123.pdf> Acesso em: 13 de junho de 2018.

NEVES; André M. M.; BARROS, Flávia A. **iAIML: Um Mecanismo para Tratamento de Intenção em Chatterbots**. 2005. XVIII Encontro Nacional de Inteligência Artificial. São Leopoldo.

RABELO, R.J.; ROMERO, D; ZAMBIASI, S.P. **Softbots Supporting the Operator 4.0 at Smart Factory Environments.** 2018. Moon I., Lee G., Park J., Kiritsis D., von Cieminski G. (eds) Advances in Production Management Systems. Smart Manufacturing for Industry 4.0. APMS 2018. IFIP Advances in Information and Communication Technology, vol 536. Springer, Cham.

READ, Ash. **5 Ecommerce Chatbots (Plus How To Build Your Own In 15 Minutes).** Disponível em: <https://sumo.com/stories/ecommerce-chatbot-marketing> Acessado em: 15 de janeiro de 2019.

ROVER, Aires José. **Informática no Direito**: inteligência artificial. Curitiba: Juruá, 2010. 270 p.

SGANDERLA, Rachele B.; FERRARI, Débora N.; GEYER, Cláudio F. R. **BonoBOT: Um Chatterbot para Interação com Usuários em um Sistema Tutor Inteligente.** 2003. XIV Simpósio Brasileiro de Informática na Educação. Rio de Janeiro.

SINGH, M. P. and HUHNS, M. N. S**ervice-Oriented Computing: Semantics, Processes**. Agents. John Wiley & Sons, New York, NY, EUA, 1a. Edição. 2005.

SOMMERVILLE, Ian. **Engenharia de Software**. 8.ed.São Paulo: Pearson, 2007.

TEIXEIRA, S.; RAMIRO, T. B.; OLIVEIRA, E. de; MENEZES, C. S. de. **Chatterbots em ambientes de aprendizagem - uma proposta para a construção de bases de conhecimento**. In: Anais do Workshop de Informática na Escola. [S.l.: s.n.], 2005. v. 1, n. 1.

WALLACE, Richard S. **AIML Overview**. 2008. Disponível em: <https://www.pandorabots.com/pandora/pics/wallaceaimltutorial.html> Acesso em: 13 de junho de 2018

WAZLAWICK, Raul S. **Análise e projeto de sistemas de informação orientados a objeto**. Rio de Janeiro: Elsevier 2004.

Web Services Architecture. 2004. Disponível em:<http://www.w3.org/TR/wsdl/> Acesso em: 14 de janeiro de 2019.

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

ZAMBIASI, Saulo P.; RABELO, R. J. **A Proposal for Reference Architecture for Personal Assistant Software Based on SOA.** Revista IEEE América Latina, v. 10, p. 1227-1234, 2012.

ZAMBIASI, Saulo P. **Chatbot Arisa: Tutorial.** Disponível em: <https://saulo.arisa.com.br/wiki/index.php/Chatbot_Arisa:_Tutorial> Acesso em: 10 de junho de 2018