Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 27, Num. 2 (2020) 116-126

RESEARCH ARTICLE

Arisa Nest – A Cloud-Based Platform for Development of Virtual Assistant

Arisa Nest – Uma Plataforma Baseada na Nuvem para Desenvolvimento de Assistentes Virtuais

Saulo Popov Zambiasi[1], Ricardo J. Rabelo[2]

Resumo: The use of virtual assistants has become popularized by their adhesion by large companies. This is because they have brought benefits to a certain extent in people's daily activities or in their tasks in organizations, with reminders, automation of repetitive tasks, proposing solutions for various situations, etc.. To meet this demand, several tools have been developed to create this type of agent. Following the same trends, this paper presents the Arisa Nest platform, under the Platform as a Service model, as a tool to create virtual assistants with resources to manage information about users, create algorithms to give more effective answers, consume web services to the interoperability with other systems and with proactivity behaviors. Finally, some cases are presented using the platform in real scenarios and tests.
Keywords: Virtual Assistants — Cloud Computing — Chatbots — Platform

Resumo: O uso dos assistentes virtuais tem se popularizado pela sua adesão por grandes companhias. Isso porque eles têm trazido benefícios, até certo ponto, nas atividades diárias das pessoas ou em suas tarefas nas organizações, com lembretes, automatização de tarefas repetitivas, propondo soluções para várias situações, etc.. Para comportar essa demanda, diversas ferramentas têm sido desenvolvidas para a criação desse tipo de agente. Seguindo o mesmo viés, esse artigo apresenta a plataforma Arisa Nest, sob o modelo de Platform as a Service, como uma ferramenta de criação de assistentes virtuais com recursos de gerir informações sobre os usuários, criar algoritmos para dar respostas mais eficazes, consumir serviços web para a interoperabilidade com outros sistemas e com comportamentos de proatividade. Ao final, são apresentados alguns casos utilizando a plataforma em cenários reais e de testes.
Palavras-Chave: Assistentes Virtuais — Computação na Nuvem — Chatbots — Plataforma

[1] *Centro Tecnológico, Universidade do Sul de Santa Catarina, Palhoça, Santa Catarina, Brasil*
[2] *Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, Brasil*
***Corresponding author**: saulopz@gmail.com
DOI: http://dx.doi.org/10.22456/2175-2745.97411 • **Received:** 15/10/2019 • **Accepted:** 05/03/2020

1. Introdução

Certos princípios estão sendo considerados cada vez mais de grande importância no que tange as empresas e organizações. Entre eles, pode-se citar a necessidade de acesso as informações em tempo real, a interoperabilidade entre os diversos sistemas envolvidos e seus processos relacionados, a disponibilidade de softwares orientados a serviços, a descentralização das informações e sistemas, virtualização, modularidade, entre outros [1]. Se por um lado esses princípios têm trazido benefícios, por outro têm exigido maior preparo das pessoas envolvidas, tendo que interagir com os mais diversos tipos de programas e máquinas, cada um com suas próprias interfaces e padrões de operação. Um caminho para facilitar essa interação entre os colaboradores e os sistemas de forma mais natural e intuitiva é pela utilização de robôs de software (*softbots*, bot ou assistentes virtuais) que permite aos usuários interagir com os sistemas via conversa em linguagem natural (*chatbots*), além de, em algumas plataformas de assistentes pessoais, automatizar e auxiliar as pessoas na execução de certas tarefas, em vários níveis de inteligência e autonomia. Um tipo de *softbot*, na forma de um Software Assistente Pessoal, foi concebido em Zambiasi e Rabelo [2] para interagir com máquinas, computadores, bancos de dados e outros sistemas de informação para auxiliar as pessoas na execução de diversas tarefas. O bot interagia por conversa em linguagem natural via *instant messaging* no dispositivo móvel do usuário, podia mandar relatórios por e-mail e executava tarefas de forma autônoma, com ou sem a necessidade da confirmação do usuário.

A tecnologia de assistentes virtuais e sua utilização não se trata de nenhuma novidade, mas apenas recentemente, com o aumento do uso e necessidade das Tecnologias de Informação e Comunicação (TIC's) e com os avanços das tecnologias envolvidas, as empresas e organizações passaram a desenvolver sistemas computacionais assistentes para auxiliar nos processos de negócio [3]. Contudo, apesar desses esforços, ainda existem diversas limitações quanto à complexidade na implementação de chatbots e assistentes virtuais mais sofisticados e inteligentes na integração com outros sistemas, flexibilidade e escalabilidade em termos de modificação e adição de novas ações, etc.

Nesse sentido, o presente artigo mostra recursos e funcionalidades da plataforma Arisa Nest que objetivam trazer contribuições para o contexto apresentado. Essa plataforma, apesar do nome atual, se trata da evolução de um projeto iniciado em 2005, com um protótipo parcial em 2009 e a apresentação da sua primeira versão totalmente funcional em 2011 [4]. O projeto se encontra em constante desenvolvimento e evolução. Ela não se trata apenas de uma outra plataforma nessa linha, mas de ser um ambiente de pesquisa para alunos e pesquisadores com melhorias e adaptações conforme as necessidades dos usuários. O processo de evolução da plataforma em si já se trata de um objeto de pesquisa. A plataforma é uma ferramenta baseada na nuvem, modelo *Platform as a Service* (PaaS), para criar assistentes virtuais com recursos que permite implementar *scripts* em linguagem de programação Lua para poder responder ao usuário com mensagens mais eficazes, permitem o consumo de serviços web externos à plataforma, permite a execução de comportamentos para que o assistente possa ser pró-ativo, permite a interação via voz e o autogerenciamento de sua base de dados como recurso de autoaprendizado.

O artigo apresenta algumas ferramentas atuais de assistentes virtuais para poder fazer um paralelo com a Arisa Nest. Em tempo é apresentado um conjunto de casos em que a plataforma foi utilizada e avaliada.

2. Assistentes Virtuais

Os Assistentes Virtuais se originaram das tecnologia dos *chatbots*, agentes de conversação para interação com os usuários em linguagem natural utilizando técnicas de inteligência artificial, e dos Agentes inteligentes. Hoje, muitos desses artefatos de software não apenas interagem com o usuário por meio de conversa em linguagem natural, como também podem executar tarefas para eles. O primeiro *chatbot* foi criado por Joseph Weizenbaum em 1966 com o nome de ELIZA e simulava a conversa com uma psicoterapeuta [5]. Aqueles princípios do ELIZA foram utilizados por diversos pesquisadores em suas pesquisas no campo dos agentes de conversação nos anos que se seguiram. Em 2001 Richard Wallace lançou a primeira versão de uma linguagem para chatbots baseada no XML (*eXtensible Markup Language*), chamada de AIML (*Artificial Intelligence Markup Language*). O intuito dessa linguagem é que ela fosse uma padronização na escrita e estrutura da base de conversação dos chatbots, podendo ela ser projetada independente do programa de execução (motor). Ela permite ao chatbot reconhecer diferentes assuntos e armazenar em memória informações coletadas durante as conversas, podendo seguir diferentes fluxos de conversação. O programa para execução da base de conversação pode ser implementado em qualquer linguagem, conforme a necessidade dos criadores de um chatbot ou podem utilizar bibliotecas ou motores já prontos, incluindo motores gratuítos. A linguagem AIML encontra-se atualmente na sua segunda versão [6].

Além de linguagens específicas e técnicas de programação de chatbots, existem também plataformas, desde as mais simples até as mais complexas, que tornam o processo de criação e utilização de chatbots mais simples e acessível. Elas servem como ambiente de execução, dando suporte ao ciclo de vida do bot e ferramentas de administração, contabilização, etc.. Existem, hoje, algumas plataformas bastante em ênfase, tal como a **Alexa** da Amazon. Sua estrutura é bastante complexa e todo o seu domínio envolve um conjunto de dispositivos e ambiente de desenvolvimento. O usuário pode interagir por meio de diversos dispositivos. A interação com a Alexa pode se dar via software multiplataforma, além de dispositivos próprios para interação via voz, chamados de Alexa Echo. O Alexa *Skills* fornece um conjunto de ações e interações com a Alexa. Essas ações são as *skills*, que executam tarefas como tocar uma música, responder questões, fornecer informações do tempo, controlar as luzes da casa, etc.. Mas também é possível ao usuário criar seus próprios *skills* personalizados utilizando o Alexa *Skills Kit*. Ele permite a chamada de *Amazon Web Services* (AWS), integração com outros serviços desenvolvidos por outros, em diferentes linguagens de programação e em outros servidores. O Alexa *Skills Kit* fornece também um conjunto de APIs para integração com diversos dispositivos [7]. É possível programar serviços para serem utilizados nos Skills da Alexa através da interface da *Amazon Web Services* (AWS). Quando um *skill* é criado, deve-se fornecer um nome e, ao conversar com a Alexa, o usuário chama diretamente aquele *skill*. Os *skills* são responsivos, executando algo requisitado pelo usuário. Há uma plataforma para pesquisa de *skills* já publicados que o usuário pode agregar ao seu assistente. Por meio dos *skills*, é possível construir uma sequência de diálogos entre o usuário e a Alexa [8]. No AWS Lambda, o usuário paga apenas pelo tempo de computação consumido, sendo permitido uma quantidade de requisições/execuções gratuitas por mês. É possível criar, nas *skills*, respostas aos usuários por meio de uma estrutura de fatos em Node.js e em diversas linguagens de programação, como javascript, python, etc. [9].

Outro exemplo é o **Watson *Conversation***. Ele se caracteriza como "uma API para desenvolvimento de bots, com uma interface simples", que permite que pessoas sem conhecimentos de programação possam alimentar sua base de conhecimento. É preciso ter uma conta na IBM *Cloud* e, até certo ponto, pode ser utilizado de forma gratuita. A plataforma possui um ambiente de desenvolvimento da base de

conversação baseada em intenções, entidades e diálogos. As intenções são ações atreladas às perguntas feitas pelo usuário. É necessário dar exemplos de frases de entrada que, posteriormente, o sistema generaliza para tentar identificar padrões nas intenções do usuário. As entidades são complementos. Por exemplo, se o usuário quer fazer um pedido de pizza, isso é uma intenção, as entidades são informações como sabor, tipo de massa, CEP da entrega, etc. Para as entidades, é possível trabalhar com sinônimos, como no caso de mussarela e queijo, ou adicionar expressões regulares. Os diálogos são utilizados para construir a interação com os usuários. Cada nó do diálogo pode responder algo, e um nó pode possuir nós filhos, ou um complemento de resposta. Os nós possuem uma interação *if-then-else* e podem armazenar variáveis [10]. As vantagens da utilização dele é a facilidade de criação, sem necessidade de conhecimentos de programação, além da generalização dos exemplos de frases de entrada do usuário nas intenções.

Outra plataforma é o **Dialogflow** da Google. Essa permite a criação de bots com interação tanto por voz quanto por textos, usando um conjunto de tecnologias de Inteligência Artificial. Sua interação pode ser via site web, dispositivos móveis, Google Assistente, Amazon Alexa, Facebook e outros. A estrutura da sua base de conversação é baseada em fluxos de diálogos, permitindo que os bots possam trabalhar em assuntos diferentes, gerenciar os padrões de entradas de mensagens (*intents*), e gerar respostas que podem levar por um caminho específico ou ramificado de perguntas e respostas, tal como uma estrutura em forma de fluxograma. O Dialogflow usa algoritmos da Google de inteligência artificial e aprendizado de máquina para generalizar os *intents*, aprendendo a encontrar padrões para responder de forma mais eficiente. Para alimentar os *intents*, basta adicionar um conjunto de exemplos de mensagens do usuário. O site do Dialogflow possui boa documentação em texto, imagem e vídeo sobre como trabalhar com a plataforma e boas práticas no processo da criação da base de conhecimento de um agente de conversação, *Conversation Design*. No Dialogflow há uma forma de proatividade por meio de eventos, sendo possível ao bot invocar *intents* baseados em algo que acontece, ao invés de uma mensagem do usuário. Ele suporta eventos do Google Assistente e outras plataformas com base em ações realizadas pelo usuário [11].

A **Microsoft Azure**, também fornece uma plataforma para a criação de chatbot, permitindo aos usuários criarem assistentes virtuais personalizados com o uso de uma SDK e um conjunto de ferramentas, incluindo ferramentas opensource. Por meio de APIs, o usuário pode, então, desenvolver seus chatbots criados na Azure para diversas plataformas e diversos tipos de *instant messaging*. Na Azure é possível criar e usar um chatbot de forma gratuíta, com recursos limitados, ou com mais recursos na versão paga. Ao criar seu primeiro chatbot, o usuário pode seguir um modelo de aprendizagem, ou tutorial, com um conjunto de exercícios práticos passo-a-passo com diversos níveis de complexidade, facilitando a criação do bot. Para isso, é necessário uma assinatura ativa do Azure e uma instalação local do Visual Studio Code [12].

O **Pandorabots**[1] é um serviço online para construção e disponibilização de chatbots na web. O usuário adiciona elementos AIML na base de conversação e pode testar o bot na própria plataforma, analisando os *logs* e identificando o que o bot não consegue responder para, então, adicionar novos elementos AIML.

Devido à atual relevância da utilização de assistentes virtuais em grandes empresas e do seu efetivo auxílio numa primeira camada de atendimento ao consumidor nos sites web de comércio eletrônico, o mercado desse tipo de serviço tem atraído diversos investidores, startups e empresas de desenvolvimento [13]. Dessa forma, diversos outros softwares, bibliotecas, plataformas, linguagens poderiam ser citadas aqui, mas tornaria a pesquisa muito ampla, fugindo do objetivo desse artigo.

3. Plataforma Arisa Nest

A Plataforma Arisa Nest (Figura 1) é uma ferramenta acadêmica, evolução da implementação apresentada em [2], desenvolvida como um ambiente baseado em PaaS que perrmite o gerenciamento de diversos assistentes virtuais e uma complexa estrutura de orquestração e coreografia algoritmos e serviços web. Neste artigo, os usuários criadores e mantenedores dos bots são chamados de *Bot Designers* ou apenas *Designers* e usuários são as pessoas que interagem/conversam com o assistente virtual.

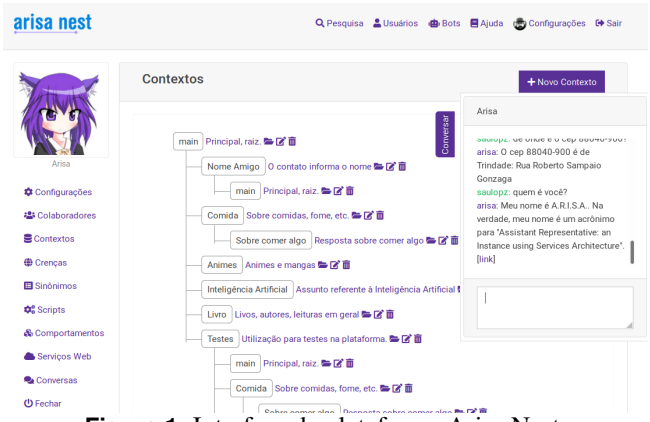

Figura 1. Interface da plataforma Arisa Nest.

Conforme Figura 1, o ambiente web para gerenciamento da plataforma Arisa Nest é composto pelo menu principal, na parte superior. Os itens são: Pesquisa, que mostra os bots públicos, disponíveis para serem pesquisados e utilizados por qualquer pessoa, com informações como Nome, id, descrição, telegram e site web; Usuários, item disponível apenas para usuários administradores, para gerenciar os demais usuários da plataforma e bots das quais colaboram; Bots, mostra a listagem das instâncias de assistentes virtuais criados na plataforma. O usuário vê apenas os bots das quais ele é colaborador, os administradores têm acesso à todos os bots. Ao

[1] https://home.pandorabots.com

clicar no ícone do bot, abre o menu lateral (ou menu do bot); Ajuda, provê um conjunto de informações sobre a plataforma, como utilizar e informações das funções de gerenciamento da base de conhecimento e consumo de serviços web que podem ser utilizadas nos *scripts* e comportamentos; Configurações, se referem aos dados dos usuários da plataforma, ou bot designers. Sair, efetua o logout. No menu lateral há os itens: Configurações referentes aos dados do bot, informações para conectar o bot ao bot criado no telegram, backup e exclusão; Colaboradores, se refere aos designers responsáveis por gerenciar a base de conhecimento do bot e seu privilégio (total, escrita, leitura); Contextos, conforme mostado no meio da figura, é o conjunto de assuntos da base de conhecimento do bot. Ao abrir um contexto específico, pode-se gerenciar os diálogos daquele contexto; Crenças, referente às crenças globais do bot; Sinônimos, para fazer substituições, por exemplo, tu e *vc* por *voce*, simplificando a quantidade de padrões necessários no diálogo; *Scripts*, para a criação e gerenciamento dos *scripts*. Há um sistema de debug para executar, testar e debugar os *scripts*; Comportamentos, para a criação, agendamento e debug dos algoritmos proativos do bot; Serviços Web, para registrar serviços web no padrão SOAP, gerando automaticamente as operações com seus parâmetros para facilitar seu uso nos *scripts* e comportamentos; Conversas, para verificar os *logs* das conversas e visualizar as informações as crenças dos usuários que conversam com o bot; Fechar, fecha o bot atualmente em edição. À direita há um botão lateral que serve para testar a conversação com o bot ativo no menu lateral.

A plataforma possui uma estrutura de criação de diálogos orientados a contextos para alimentar a base de conhecimento (ou base de conversação), um editor de *scripts* em linguagem de programação Lua que pode ser usado para complementar as respostas dos diálogos com resultados da execução de algoritmos, executar operações complexas e consumir serviços externos. Tanto serviços web no padrão SOAP (*Simple Object Access Protocol*) como no padrão REST (*REpresentational State Transfer*) podem ser consumidos pelos *scripts*. Na Arisa Nest há também um gerenciamento de crenças, que permite armazenar informações globais e informações relativas à cada usuário do bot; comportamentos em Lua que podem ser agendados e executados conforme mudança do estado do bot fornecendo proatividade. Ela permite a criação de vários bots e cada bot pode ter diversos designers que podem colaborar na criação da sua base de conhecimento.

A base de conhecimento pode ser vista como uma árvore de contextos (Figura 1). Um **contexto** é um assunto geral ou específico que o bot tem conhecimento, definido pelos designers, e o bot pode-se mudar de assunto conforme o fluxo da conversa entre usuário e bot. Por exemplo, se o usuário perguntar ao bot "o que você prefere?", a pergunta é respondida de forma diferente para os contextos **comida** e **livro**. É necessário haver um contexto inicial/pai chamado *main* (ou 0). Os diálogos são criados dentro dos contextos.

Os **diálogos** servem para identificar uma mensagem do usuário e produzir uma resposta. O campo Condições efetua testes lógicos baseados nas crenças do bot e nas crenças dos usuários para permitir ao bot usar ou não esse diálogo para responder uma mensagem. O campo Padrões é usado para identificar se o diálogo bate com a mensagem do usuário e, em caso caso positivo, é escolhida uma das resposta aleatória do diálogo escolhido. É possível inserir vários padrões e respostas separados por ponto e vírgula. Um link e uma imagem também podem ser utilizados como parte complementar da resposta. O campo **Ir Para** serve para direcionar o motor do chatbot para outro contexto, ou seja, caso esse diálogo for escolhido, o contexto muda. Por exemplo o bot designer cria um contexto chamado Inteligência Artificial, para tratar de assuntos referentes à essa disciplina. No contexto main, o designer cria um diálogo que identifica palavras relacionadas com o assunto de IA, nos padrões. Quando o bot identifica na mensagem do usuário algo sobre o assunto, o diálogo no main é chamado e possui um **Ir Para** "Inteligência Artificial". A partir desse momento, o bot irá responder coisas específicas de IA.

Caso o usuário converse sobre algo que o bot não entendeu nesse contexto, o bot volta para o contexto *main*. Há um tipo de diálogo especial, chamado Fuga, em que deixa-se o campo padrões vazio e, caso o bot não encontre nenhum diálogo que fecha naquele contexto, e existir um diálogo tipo fuga, ele escolhe uma resposta aleatória do mesmo. Só pode haver diálogo de fuga no contexto main. Na área de visualização das conversas, com informações das crenças locais e registros das conversas, uma mensagem com cor diferente nos *logs* das conversas indica que o bot usou uma resposta de fuga. Isso facilita para identificar as mensagens que o bot não está conseguindo responder. Do lado direito do plataforma, no botão que permite abrir uma janela de chat para testar o bot na própria plataforma (Figura 1) o usuário faz os testes necessários e ajustes para que o bot possa responder mais efetivamente as mensagens respondidas com "fuga". A plataforma suporta multi linguagem e está disponível atualmente nos idiomas português e inglês.

Nos padrões, o símbolo **%** representa um caractere curinga, que pode estar entre palavras, no final e/ou no início, conforme pode ser visto na Figura 2.

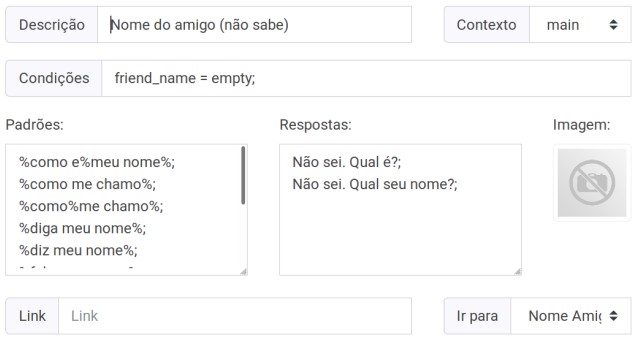

Figura 2. Formulário de diálogos.

Alguns tipos especiais podem ser utilizados nos padrões e respostas dos diálogos. Esses elementos servem para per-

mitir gerir crenças globais (do bot) e crenças locals (de cada usuário do bot), além de chamar *scripts* em linguagem de programação Lua criados no chatbot. Por exemplo:

Padrão: %meu nome e **{friend_name}**;
Resposta: oi **{friend_name}**. Me chamo **{bot_name}**;
Mensagem usuário: olá, meu nome é **Saulo**
Resposta do bot: oi **Saulo**. Me chamo **Arisa**

Nesse caso, o bot pega uma informação da mensagem passada pelo usuário que está conversando com o bot, chamada {friend_name}, e armazena como uma crença local. {bot_name} é usando na resposta e é proveniente de uma crença global. Quando a informação não precisa ser armazenada, mas apenas usada para algo e depois descartada, coloca-se um sinal de $ no início do nome da crença, como por exemplo:

Padrão: %quanto e **{$1}** mais **{$2};**
Resposta: O resultado é **{@soma $1 $2}**;
Mensagem usuário: Quanto é **2** mais **5**?
Resposta do bot: O resultado é **7**

As informações $1 e $2 são utilizadas apenas para o processamento nesse diálogo. Nele também que foi usado o sinal @ que indica a chamada do *script* soma, passando como parâmetro duas informações, 2 e 5. Os ***scripts*** são algoritmos desenvolvidos em linguagem de programação Lua na plataforma e podem ser usados de diversas formas, como executar cálculos e operações complexas, além de consumir serviços web externos à plataforma. Por exemplo, o *script* do diálogo anterior poderia ser apenas:

```
return tonumber(args[1]) + tonumber(args[2])
```

Ou o retorno de uma chamada a um serviço web:

```
return wscall('math', 'sum',  a = args[1], b = args[2] )
```

Em ambos os casos pode-se observar os argumentos de entrada **args**. No segundo caso, é usada a função **wscall**, que consome um serviço web cadastrado na plataforma chamado **math** usando a operação **sum**. No editor de *scripts* e comportamentos há um botão que, quando clicado, o usuário seleciona o serviço e a operação. O texto da chamada do serviço é adicionado automaticamente no editor, no cursor de edição, facilitando a criação do código.

Os **Comportamentos** são executados regularmente ou conforme agendamento configurável. Por exemplo, o comportamento da Figura 3 é agendado para todo dia 1º de Janeiro, às 0h01min, desejando feliz ano novo para todos os seus contatos do telegram.

Na Figura 3 a função `friendList` retorna os IDs de todos os contatos do usuário em uma *string* no formato JSON. Para cada contato, por meio de um laço, é utilizada a função

```
1 friends = json.decode(friendList())
2 for i = 1, #friends do
3     id = friends[i]
4     im = getLocal(id, 'im')
5     if im == 'telegram' then
6         message = runDialog('main', 'feliz_ano_novo')
7         sendMessage(id, message)
8     end
9 end
```

Figura 3. Comportamento feliz_ano_novo.

`getLocal` que retorna uma crença de um contato, conforme seu **ID**. No caso, foi retornado o *instant messaging* (IM) usado pelo usuário. Esse *script* considera os usuários que utilizam o telegram, então, caso a informação recebida seja 'telegram', utiliza uma resposta aleatória de um diálogo chamado **feliz_ano_novo** do contexto **main**, com a função `getDialog`, e envia como mensagem para o usuário com `sendMessage`. Além dessas funções já citadas, a plataforma Arisa Nest possui um conjunto de funções para diferentes tipos de atividades. Algumas das funções básicas que podem ser utilizadas tanto nos *scripts* quanto nos comportamentos são apresentadas na tabela 1.

Tabela 1. Funções básicas

Função	Descrição
globalList	retorna a lista dos nomes das crenças globais.
localList	lista dos nomes das crenças locais de um contato.
getGlobal	retorna o valor de uma crença global.
setGlobal	altera uma crença global.
setLocal	altera uma crença local.
deleteGlobal	exclui uma crença global.
deleteLocal	exclui uma crença local.
last	retorna dia e horário da última mensagem enviada por um usuário ao bot.
sendToBot	envia uma mensagem para outro bot.
callScript	permite usar um *script* existemte em outro *script* ou coportamento.
callRest	chama um serviço REST.

A Figura 4 apresenta uma chamada da função `callRest`, passando como parâmetro o serviço, a operação, o tipo, configurações do cabeçalho da mensagem, e as informações da mensagem no formato o JSON. O serviço é chamado, retornando, nesse caso, uma string json que é decodificada com a função `json.decode` na forma de um objeto JSON.

Além disso, a plataforma também possui um conjunto de funções para o gerenciamento da base de conhecimento do bot via *scripts* e comportamentos (tabela 2). O objetivo dessas funções é que o próprio bot possa gerenciar sua base de conhecimento por meio de *scripts* e comportamentos, permitindo um certo nível de aprendizado e uma evolução da sua interação com os usuários.

```
out = callRest(
  'https://dev.arisa.com.br/myservice',
  '/v1/Product/', 'POST', {
    'Content-Type: application/json-patch+json',
    'Accept: application/json',
    'ApiKey: 0123456789ABCDEFGHIJKLMNOPKRSTUW'
  },
  json.encode({
      productId = args[1],
      productName = args[2],
      description = args[3]
    })
  )
obj = json.decode(out)
return obj.response
```

Figura 4. Exemplo de *script* com chamada REST.

Mesmo que não seja implementada uma inteligencia artificial de extração de conhecimento ou de *deep learning* no chatbot, essas funções mais complexas podem ser feitas externamente e consumidas por serviços web. O resultado da execução desses serviços pode servir como fonte de aprendizado para o bot, criando novos contextos, diálogos, readaptando os existentes, etc.

3.1 Considerações sobre a Implementação

O motor de execução do chatbot da Arisa Nest foi implementado na linguagem PHP e sua base de conhecimentos é armazenada em banco de dados Postgresql. O sistema de execução baseado em contextos inicia no contexto raiz main. A mensagem entrada do usuário é comparada com os padrões cadastrados nos diálogos daquele contexto. Caso algum padrão de um ou mais diálogos fechar com a entrada do usuário, é escolhido um diálogo aleatório, dentre os encontrados, e uma resposta aleatória desse. Antes do bot enviar uma resposta, a mensagem do usuário é quebrada em partes e as crenças são extraídas da mensagem e usadas para as crenças, para compor a resposta ou para serem utilizadas como parâmetros de algum *script*. Se o diálogo escolhido possui um "Ir Para" para outro contexto, o usuário é direcionado para ele. Quando o usuário envia uma nova mensagem, caso ela não encontre um diálogo no contexto atual, o motor navega para o contexto pai, e assim segue até que chegue ao contexto raiz. Se mesmo no contexto raiz, o motor não encontrar padrões que fecham com a mensagem, ele responde com uma resposta de "fuga". Os *scripts* são utilizados pelos diálogos, podem ler e escrever crenças, mandar mensagens para outros usuários ou bots, executar operações externas por meio de serviços web SOAP ou REST. Os comportamentos, por sua vez, são a forma dos bots agirem de forma proativa. Para que essa autonomia seja possível, é necessário que um programa executor de comportamentos. Esse executor de comportamentos foi feito na forma de programa de computador implementado em linguagem de programação Go[2]. O programa executa em *background* no sistema operacional, verificando quais comportamentos estão configurados para execução e os executa, conforme configuração de agendamento. A interface do usuário foi implementada utilizando PHP, HTML, CSS, JavaScript, Ajax e Bootstrap 4. A plataforma encontra-se instalada e rodando em um servidor na nuvem com Sistema Operacional Ubuntu 18.04, servidor HTTP Apache 2 e banco de dados Postgresql, sob o domínio https://arisa.com.br/nest.

Tabela 2. Funções da base de conhecimento

Função	Descrição
contextList	lista os ids dos contextos do bot.
getContext	retorna as informações de um contexto específico.
addContext	adiciona um novo contexto.
setContext	altera as informações de um contexto.
deleteContext	exclui um contexto e todos os seus diálogos.
dialogList	lista os ids dos diálogos de um contexto.
getDialog	retorna as informações de um diálogo.
addDialogo	cria um diálogo.
setDialogLink	altera o link de um diálogo.
setDialogGoto	configura o Ir Para de um diálogo.
deleteDialog	exclui um diálogo.
getConditions	retorna as condições de um diálogo.
addCondition	adiciona uma condição à um diálogo.
setConditions	adiciona várias condições à um diálogo, excluindo as existentes.
deleteConditions	exclui as condições do diálogo.
getPatterns	retorna os padrões de um diálogo.
setPatterns	adiciona vários padrões, excluindo os existentes.
addPattern	adiciona um padrão.
deletePatterns	exclui todos os padrões.
getResponses	retorna as respostas de um diálogo.
setResponses	adiciona várias respostas à um diálogo, excluindo as existentes.
addResponse	adiciona uma resposta.
deleteResponses	exclui todas as respostas do diálogo.

3.2 Arisa Nest como Plataforma para Agentes

A plataforma Arisa Nest permite a criação de múltiplos bots que podem executar algoritmos, consumir serviços web e conversar entre si para colaborar na resolução de problemas. Tal como os agentes [14] eles podem perceber os mais diversos possíveis ambientes, conforme funcionalidades e interoperabilidades fornecidas por meio do acesso à serviços web e mensagens de usuários ou outros bots (habilidade social). Ar-

[2]Linguagem de programação criada pela Google, baseada na linguagem Limbo do sistema operacional Inferno, focada em programação concorrente e produtividade. Em 2009 foi lançada como código livre e atualmente com licença BSD. https://golang.org/

mazenam crenças para compor seu estado interno, executam seu processamento interno por meio de *scripts* e algoritmos de comportamentos que podem alterar seu estado e agir de forma autônoma e proativa. Por meio desses recursos, é possível construir na plataforma agentes que vão desde os puramente reativos, até os cognitivos e/ou baseados em objetivos e utilidade.

Considera-se aqui o seguinte cenário fictício simplificado para uma prova de conceito. O funcionário Saulo é responsável pelo controle de estoques de produtos em uma empresa. Um agente foi criado para auxiliá-lo, verificando regularmente a situação em um sistema legado e, caso o estoque de um produto esteja com a quantidade menor do que uma certa quantidade pré configurada, o agente inicia um processo de compra, com a autorização do funcionário. Caso o processo de compra, seja autorizado, o agente entra em contato com um agente de vendas da empresa fornecedora e faz o pedido.

Algumas considerações sobre as atividades e comunicação entre os agentes na plataforma Arisa Nest: (i) a comunicação entre os agentes pode ser padronizada para facilitar o entendimento entre eles com palavras chaves e parâmetros; (ii) são utilizados o ID do agente e o servidor onde a plataforma está instalada para o reconhecimento do agente, permitindo a comunicação de agentes alocados em servidores diferentes; (iii) Um agente entra em contato com o outro por meio de mensagens e ela é analisada pelo motor de chatbot, que encontra o diálogo correspondente e inicia um *script*; (iv) Um processo automatizado ou conversação entre os agentes ou entre um agente e seu usuário pode ser iniciado por um comportamento ao verificar mudanças nas crenças ou por mudanças em uma informação consumida via um serviço web.

O algoritmo da Figura 5 é um comportamento agendado para executar a cada hora. Ele verifica o estoque e, caso necessário, cria uma ordem de compra, pedindo confirmação do usuário. Em caso afirmativo, efetua a compra.

```
1  produto.id = json.decode(wscall('estoque', 'baixo', null))
2  if produto.id ~= '' then
3      fornecedor = json.decode(wscall('fornecedor', 'get',
4          { produto = produto.id }))
5      if fornecedor.id ~= '' then
6          ordemjson = wscall('ordem', 'novaCompra', {
7              fornecedor = fornecedor.id, produto = produto.id,
8              quantidade = produto.quantidadeCompra })
9          setLocal(getGlobal('funcionario_id'), 'ordem', ordemjson)
10         ordem = json.decode(ordemjson)
11         mensagem = runDialog('estoque', 'envia_ordem_funcionario',
12             'ordem ' .. ordem.id .. ' produto ' .. produto.nome ..
13             ' quantidade ' .. produto.quantidadeCompra ..
14             ' fornecedor ' .. fornecedor.nome ..
15             ' preco ' .. fornecedor.preco)
16         sendMessage(getGlobal('funcionario_id'), mensagem)
17     end
18 end
```

Figura 5. Comportamento verifica_estoque.

Seguindo sua execução, na linha 1 ele consome um serviço web de acesso a um sistema legado, que retorna informações de um produto necessárias para o algoritmo, ou seja, se o produto está com o estoque baixo. Na linha 3 consome um serviço que encontra um fornecedor cadastrado e que trabalha com aquele produto e, caso exista mais de um, faz uma escolha conforme critérios daquele serviço. Na linha 6 é criada uma ordem de compra em um sistema legado, e as informações retornam no formato JSON. Na linha 9, as informações dessa ordem são armazenadas como crenças do usuário, pois a mesma ainda necessita de confirmação e o usuário pode perguntar mais informações sobre a mesma antes de confirmar. Na linha 11 é criada uma mensagem para o usuário com base no diálogo de nome `envia_ordem_funcionario` do contexto ordem, passando os parâmetros necessários (Figura 6). Na linha 16, a mensagem é enviada para o funcionário responsável (Figura 7).

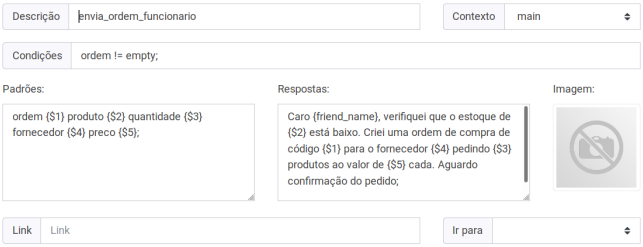

Figura 6. Comportamento envia_ordem.

No momento em que o usuário envia a mensagem para seu assistente virtual, confirmando ou cancelando a ordem de compra, o bot recebe essa mensagem, seleciona o diálogo específico e aciona um *script* que envia uma mensagem ao agente fornecedor em conformidade com a opção do usuário. Caso o fornecedor não possa cumprir com a venda da ordem de compra, a ordem é cancelada e o usuário é informado, caso contrário a compra é efetivada (Figura 7).

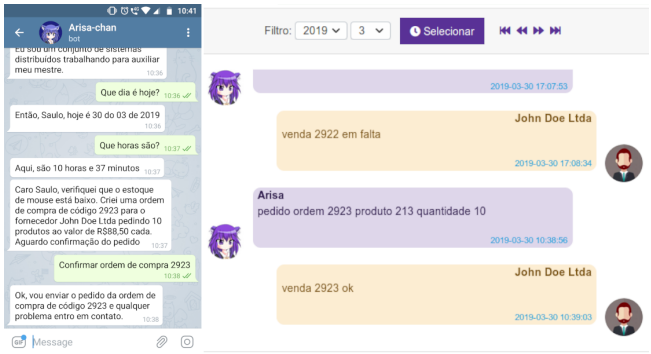

Figura 7. Esquerda: interação com o usuário via telegram; Direita: registro da interação entre os bots na plataforma.

Mensagem Arisa para fornecedor:

pedido ordem 2923 produto 213 quantidade 10

Três possíveis respostas do fornecedor:

venda 2923 ok
venda 2923 em falta
venda 2923 produto inexistente

Ao interagirem, os bots alteram seus estados internos por meio das crenças, *scripts* e comportamentos. Eles podem interagir tanto com seus usuários como com outros bots para alcançar seus objetivos. Mesmo que a plataforma possa ter vários bots executando ao mesmo tempo, cada bot precisa ser cadastrado e configurado individualmente. No momento a plataforma não permite a criação e exclusão dinâmica de bots como para um sistema multiagentes dinâmico, mas tal recurso já encontra-se em desenvolvimento.

3.3 Considerações da Plataforma

Comparando a plataforma Arisa Nest com as ferramentas citadas na seção 2, a ELIZA, ALICE e Pandorabots são apenas agentes de conversação e não executam tarefas e pouco fazem no caso de adquirir informações dos usuários. No caso da ALICE e Pandorabots, há o uso de uma linguagem AIML, já bastante madura, mas trabalhar com essa linguagem não é tão intuitivo e requer um certo aprendizado e conhecimento, enquanto a alimentação dos diálogos da Arisa Nest é mais simples do que a criação de estruturas AIML e requer apenas entender a estrutura dos padrões e crenças. Já no caso do Watson Conversation, ele tem uma interface de fácil aprendizado e de uso relativamente intuitivo, não requer entender padrões dos intents, além de utilizar algoritmos de inteligência artificial do Watson para identificar padrões de entrada parecidos com os intents criados pelo usuário para poder generalizar. Entretanto, ele não executa algoritmos criados pelo usuário, nem tem proatividade e seu foco é na conversação. O Dialogflow é uma ferramenta bem aprimorada para trabalhar com agentes de conversação e é orientado a fluxos de conversas estilo fluxograma, algo que poderia se comparar em parte com os contextos da Arisa Nest. Ele possui também um recurso de proatividade através de eventos, mas ele serve para iniciar um contato com o usuário por algum fluxo de diálogo. A forma dele trabalhar com execução de algum algoritmo é por meio do recurso de *fulfillment*, sendo necessário a configuração de um *webhook* para executar um *script* em javascript sob Node.js. Entretanto, mesmo se mantendo nos padrões da grande estrutura da Google, há uma certa complexidade para se trabalhar com os *scripts* e requer um conhecimento mais aprimorado. Para trabalhar com *scripts* na Arisa Nest basta conhecer as funções de acesso à plataforma, um pouco da linguagem de programação Lua, largamente utilizada em jogos digitais. Além disso, eles podem ser escritos, testados e executados na própria plataforma. A Alexa é uma plataforma bastante completa e tem a vantagem de utilizar toda a estrutura da Amazon, incluindo o AWS para a criação de *scripts*, hardwares e dispositivos. Ela já vem com uma grande gama de skills e o usuário pode adicionar outros, via ferramenta de compartilhamento ou produzidas pelo próprio usuário. Também há proatividade por meio de eventos. Contudo, seu nível de complexidade está no mesmo nível, ou maior, que o Dialogflow. A plataforma de bots da Microsoft Azure também fornece uma boa estrutura de desenvolvimento e tem um conjunto de tutoriais passo a passo, em várias línguas, que ajudam bastante o usuário na criação dos seus chatbots, além de que a ferramenta tem toda uma carga de algoritmos e estrutura de inteligência artificial para ajudar na inteligência do chatbot. O bot da Microsoft também trabalha com proatividade por meio de gatilhos de eventos para iniciar uma conversa com os usuários. Há um conjunto de funcionalidades gratuítas, mas os demais recursos são pagos.

Como contribuição, a Arisa Nest provê uma plataforma para a criação de bots autônomos, proativos, permitindo sua programação via linguagem de programação Lua de forma embarcada na própria plataforma, consumo de serviços web e comunicação com outros bots de outras instâncias da plataforma. Contudo, a principal diferença entre a Arisa Nest e as demais plataformas aqui citadas é que ela é um ambiente em constante pesquisa e evolução. Os estudantes e pesquisadores com projetos implementados nela estão em contato direto com o desenvolvedor, trocando ideias, sugerindo mudanças, melhorias, adaptações à outros recursos, plataformas e softwares. Muitos dos recursos hoje existentes foram necessidades descobertas durante os projetos já implementados e atuais. Plataformas de grandes empresas, como Google, Microsoft, etc. fornecem um certo leque de recursos, mas há um entrave quanto à novas adaptações.

A plataforma Arisa Nest provê recursos de proatividade aos bots, uma estrutura de programação para implementar os comportamentos e uma forma de comunicação via serviços web, tanto para consumir serviços externos como para se comunicar com bots de outras instâncias da plataforma instaladas em outros servidores, dando uma base para que se possa trabalhar com agentes. Contudo, ainda há muito o que se fazer para deixar a plataforma mais adequada para comportar de forma mais padronizada e facilitada a tecnologia de agentes. Dessa forma, como trabalhos futuros nessa linha pretende-se implementar funções que comportem KQML, FIPA-ACL, arquitetura BDI e outros.

A tabela 3 apresenta um resumo comparativo dos recursos das ferramentas pesquisadas. As linhas representam as ferramentas e as colunas as caracteristícas selecionadas numeradas de 1 à 14 e logo abaixo a legenda das características. O *"x"* marca as características que a ferramenta possui.

Tabela 3. Comparação das ferramentas pesquisadas.

Ferramentas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ELIZA	x	.	.	.	.	.	.	.	.	.	.	x	.	.
ALICE	x	.	x	.	x	.	.	.	.	.	.	x	.	.
Pandorabots	x	.	x	.	x	.	.	.	.	.	.	x	.	.
Watson C.	x	.	x	.	.	x	x	.	.	.	.	x	.	.
DialogFlow	x	x	x	x	.	x	x	x	x	x	x	.	.	.
Alexa	x	x	x	x	.	x	.	x	x	x	x	.	.	.
Azure Bot	x	x	x	x	.	x	.	x	x	x	x	.	.	.
Arisa Nest	x	x	x	x	.	x	.	x	x	x	x	x	x	x

Legenda:

1. Possui recursos de conversação estilo chatbot.

2. Os bots podem ter autonomia e serem proativos.
3. Adquirem dados dos usuários.
4. Recursos de autoaprendizagem.
5. Utiliza AIML para gerenciar a base de conversação.
6. Usa interface de formulários para gerenciar a base de conversação.
7. Aprende com padrões de entrada para generalizar novas entradas.
8. Utiliza scripts reativos.
9. Utiliza proatividade via algum tipo de programação.
10. Pode consumir serviços web para compor/melhorar seu comportamento.
11. Pode se comunicar com outros dispositivos ou softwares.
12. Menor nível de complexidade para o desenvolvimento de instâncias.
13. Implementa alguns recursos e pretende evoluir no caminho de agentes inteligentes e sistemas multiagentes.
14. Acessíveis mudanças e melhorias na plataforma conforme necessidade.

4. Casos e Testes

A Arisa Nest foi testada e avaliada em quatro principais casos. O primeiro foi um projeto de estágio no curso de Engenharia de Controle e Automação da UFSC, em 2018, pelo aluno Ricardo Ventura e aplicado na secretaria acadêmica do curso de Pós-Graduação em Engenharia Mecânica (PosMec) [15]. Para este projeto foi utilizada uma versão anterior da plataforma, apenas com os recursos de conversação via árvore de contextos. O aluno passou por um breve treinamento para a utilização do sistema, criação de contextos, diálogos, estrutura dos padrões e criou uma base de conversação sem maiores problemas. A maior dificuldade foi na aquisição, filtragem e estruturação das informações para inserir no sistema. O bot foi avaliado pelas respostas preenchidas em um formulário na interface web do bot e, conforme os resultados, para 70% dos usuários, o bot respondeu corretamente aos questionamentos, 60% responderam que tornou mais fácil o acesso à informação da PosMec e para 60% o sistema contribuiu para padronizar as informações. Atualmente o bot ainda encontra-se em funcionamento e foi migrado para a versão atual da plataforma Arisa Nest.

O segundo caso foi o bot Riunita, desenvolvido em 2017 e 2018 como projeto de pesquisa do Artigo 170[3] pelo acadêmico Laércio de Sant'Anna Filho para o Repositório Institucional (RIUNI[4]) da Universidade do Sul de Santa Catarina [16]. Para a interação com os usuários da plataforma RIUNI, foi criado um módulo de chat no canto inferior do site. O projeto focou na resolução de um conjunto inicial dúvidas dos usuários. Houve um treinamento de 30 minutos para a utilização da plataforma e formas de criação dos contextos e diálogos. Contudo, foi na prática que o aluno compreendeu a estrutura de uma base de conversação baseada em padrões de entrada e contextos. O projeto passou por uma primeira fase de validação pelos próprios funcionários da biblioteca e alguns conhecidos. Apesar de bem recebido, foram feitas sugestões de melhorias para uma possível próxima versão. Os resultados do projeto foram apresentados na Jornada Unisul de Iniciação Científica (JUNIC) de 2018 [17].

O terceiro caso foi o chatbot CMBOT, desenvolvido pelos acadêmicos Itamar Ghidini e Winicius Mattos [18] como Trabalho de Conclusão de Curso (TCC) no curso de Sistemas de Informação na Universidade do Sul de Santa Catarina (UNISUL) e aplicado no ambiente de desenvolvimento da empresa em que um dos alunos envolvidos no TCC era funcionário. Sua finalidade era recepcionar o cliente (usuário dos sistemas da empresa) e auxiliá-lo a completar o atendimento até seu fechamento. No site de atendimento foi criado um módulo web de chat no canto inferior direito do site para que os clientes pudessem interagir com o bot. O projeto foi desenvolvido na versão atual da plataforma, mas, além da base de conversação, os alunos usaram apenas o recurso de crenças para tratar algumas poucas informações dos clientes. Foi dado um pequeno treinamento aos alunos, mas eles tiveram acesso a um tutorial na web, facilitando o aprendizado. O maior problema encontrado por eles foi a falta de documentação interna da empresa e dos atendimentos feitos pelos funcionários, indicando a necessidade de se documentar o processo e as vantagens de haver um sistema que gere registros automatizados, como no caso da utilização de um chatbot. O sistema foi avaliado por um conjunto de usuários (6 desenvolvedores e 3 atendentes de suporte) e à eles foi aplicado um questionário. Conforme os resultados, 78% responderam que o chatbot deixa a forma de atendimento mais eficiente e assertiva, 66% concordaram que a ferramenta facilita, pois permite a abertura de chamado fora do horário do trabalho, 66% disseram que o bot deve ser proativo para interagir com o cliente no caso dele ficar muito tempo na seção sem atividade, 100% concordaram que a tecnologia é importante para o atendimento, resolvendo pelo menos as dúvidas mais simples do cliente. Por fim, 89% disseram que a base de conhecimento está totalmente relacionada a eficiência, o que demonstra a importância de uma boa construção dessa base. Foi também identificado que para os atendimentos mais diretos e simples foi preferido o uso do chatbot pela agilidade e disponibilidade da ferramenta, sem a necessidade da espera em fila do atendimento.

O quarto caso foi o Projeto de Final de Curso (PFC) do aluno Ricardo Ventura [19], do curso de Engenharia de Controle e Automação da UFSC, que foi uma evolução do seu estágio acadêmico, já citado. A avaliação produzida naquele projeto trouxe questionamentos para melhorias no processo de atendimento aos usuários de chatbots. O aluno usou todos recursos disponíveis na atual versão da plataforma Arisa Nest, indo mais no viés da assistência virtual do que especificamente de um chatbot. A intenção foi criar um bot para executar ações

[3] Artigo 170: programa de bolsas de pesquisas da Secretaria da Educação do Estado de Santa Catarina para incentivar potenciais talentos entre alunos de cursos de graduação com um grau de carência econômica e financeira.

[4] RIUNI: repositório de produções intelectuais da comunidade acadêmica da Unisul

para os usuários, automatizar processos e, inclusive, com integração à serviços web padrão SOAP, desenvolvidos pelo próprio aluno, providos pela UFSC e por outras empresas ou ecossistemas. Como o projeto fazia o consumo de serviços web da UFSC que envolvia questões de segurança dos dados, não foi possível nessa primeira instância fazer uma validação com a utilização da ferramenta por usuários reais da instituição. Dessa forma projeto foi validado em um conjunto de destes nas funções implementadados a partir de exemplos de utilização da feramenta. Os resultados foram utilizados para os ciclos de melhoramentos conforme metodologia Design Science.

Após analisar algumas situações, foram modelados os processos, definindo os diálogos, as crenças, *scripts* e implementação dos comportamentos. Entre algumas atividades do bot estão: informar o índice de aproveitamento semestral acumulado do aluno, as aulas de um dia na semana, cardápio do dia no restaurante universitário, a nota das disciplinas cursadas, alterar e-mail cadastrado, fazer matrícula, fornecer histórico semestral e, como proatividade, informar diariamente as aulas do usuário do dia seguinte. Outra atividade do bot, mais para um coordenador de estágios e PFC, foi automatizar o processo analisar alguns critérios do termo de compromisso de estágio em que é necessário pesquisar informações em diferentes locais, findando o resultado das análises com o envio de um e-mail ao coordenador e ao aluno analisado.

Também foi criado um outro cenário para automatizar parte do atendimento aos clientes no processo de venda em uma empresa fictícia. Este cenário está mais para testes na plataforma para verificar a aplicabilidade de cenários do Operador 4.0 na mesma. Este se daria pela utilização de assistente virtual para auxiliar um operador de equipamentos industriais em um ambiente inteligente, nos moldes da indústria 4.0, em que o assistente pode tomar algumas decisões e realizar algumas tarefas de forma independente. Esse cenário foi uma adaptação simplificada de um caso mais complexo abordado no artigo "*Softbots Supporting the Operator 4.0 at Smart Factory Environment*"[1]. Segundo Ventura [19], com a utilização dos recursos da plataforma, a implementação foi simples. O bot recebia uma informação que iniciava o processo por meio da execução de um *script*. A busca de informações se dava através de chamadas na camada de serviços e, conforme a orquestração dos serviços e atividades implementadas no *script* Lua, o resultado cumpriu com as diretrizes pré-programadas e concluindo a tarefa. Para ele, "todas as funcionalidades da plataforma Arisa Nest se mostraram bastante robustas e complementares entre si. A possibilidade de consumir serviços web, executar *scripts*, armazenar crenças"e automatização com os comportamentos possibilita "uma grande variedade de implementações em diferentes cenários".

Atualmente a plataforma está sendo utilizada para três outros projetos em andamento: um Projeto de bolsa PUIC do aluno João Ricardo dos Santos Kleine Buckstegge na Universidade do Sul de Santa Catarina com o objetivo de resolver dúvidas dos alunos de Trabalhos de Conclusão de Curso quanto à metodologia científica e procedimentos na instituição; um projeto de mestrado no Programa de Pós-Graduação em Engenharia de Automação e Sistemas (PGEAS) na Universidade Federal de Santa Catarina pelo aluno Brunno Abner Machado, trabalhando com diversos recursos da plataforma para gerência de chão de fábrica pelos operadores de máquina via assistente virtual e complexa orquestração e consumo de serviços web no contexto da Indústria 4.0 e Operador 4.0; e em um teste de utilização de assistentes virtuais para o PJe (Processo Judicial Eletrônico) do Conselho Nacional de Justiça (CNJ).

5. Conclusões e Trabalhos Futuros

Este artigo apresentou uma plataforma de assistentes virtuais chamada Arisa Nest, no modelo arquitetural *Platform as a Service* (PaaS) de disponibilização baseada na nuvem. A plataforma provê aos *Bot Designers* recursos de gerenciamento de uma base de conversação, gerenciamento de informações na forma de crenças globais (do bot) e locais (de cada usuário que se comunica com o bot), *scripts* para execução de algoritmos, acesso a serviços externos padrão SOAP e REST, funções que permitem o auto-aprendizado, comportamentos proativos que permite uma certa autonomia no ciclo de vida do bot e está em constante evolução, sendo modificada e melhorada em conformidade com as necessidades encontradas nos projetos sendo desenvolvidos em cima da ferramenta. Foram pesquisdas e apresentadas algumas ferramentas atuais no mercado que possuem correlação com as atuais características da plataforma Arisa Nest como forma de fazer uma análise comparativa.

Foi feita uma apresentação da plataforma, alguns recursos, uma descrição breve de como utilizá-los e, como proposta complementar, um caso foi apresentado como forma de utilização da plataforma também como um ambiente de criação de agentes, suportando todo seu ciclo de vida, comportamento e acesso a outros recursos, dispositivos e informações por meio do consumo de serviços web. Em seguida, como forma de validação e avaliação, foram apresentados alguns casos que utilizaram a plataforma como base para projetos de pesquisa e trabalhos de fim de curso de graduação. Cada caso foi avaliado em termos da utilização da plataforma. No contexto do que foi apresentado, testado e avaliado, a Plataforma se mostrou como uma ferramenta com certa maturidade e robustez, tanto para chatbots como para assistentes virtuais e agentes, resolvendo os casos testados de forma satisfatória.

Como contribuições, a Arisa Nest é uma ferramenta que se encontra em constante evolução, se adaptando às necessidades dos alunos e pesquisadores que a utilizam. Ela serve como ambiente de pesquisa, tanto nas teorias e tecnologias utilizadas para seu desenvolvimento com a agregação de algoritmos e novos recursos, como para os projetos que a utilizam como plataforma para assistentes virtuais. Quanto à utilização da plataforma, outros projetos, já citados, encontram-se em processo de criação e avaliação. Quanto ao seu desenvolvimento propriamente dito, ainda estão sendo feitas várias

implementações para que a plataforma fique mais fácil de usar e intuitiva, criação de mais funções de acesso aos recursos da plataforma pelos *scripts* e comportamentos, funções para o bot se auto-clonar, criar outros bots e migrar para outros servidores e, ainda em análise, a adição de um recurso/editor de comportamentos baseado em agentes BDI *(Beliefs, Desire, Intentions*) e outros recursos de inteligência artificial e *deep learning* já implementados na plataforma e que possam ser usados pelos bot designers.

Como trabalhos futuros, pretende-se implementar funções para facilitar a comunicação e ciclo de vida dos bots, como agentes, por meio de padrões e tecnologias de agentes inteligentes e sistemas multiagentes. Pretende-se também aplicar e avaliar a plataforma para mais algumas instâncias de casos no contexto da indústria 4.0 e como um artefato complementar na educação.

Contribuição dos Autores

Saulo Popov Zambiasi contribuiu no desenvolvimento e manutenção da plataforma, escrita do artigo, orientação e coorientação de alguns projetos utilizando a plataforma.

Ricardo J. Rabelo foi orientador de Doutorado e Pós-doutorado de Saulo Popov Zambiasi, atuou na orientação de alguns projetos utilizando a plataforma, na correção e no auxílio da construção do artigo.

Referências

[1] RABELO, R. J.; ROMERO, D.; ZAMBIASI, S. P. Softbots supporting the operator 4.0 at smart factory environments. In: MOON, I. *et al.* (Ed.). *Advances in Production Management Systems. Smart Manufacturing for Industry 4.0*. [S.l.]: Springer, 2018. p. 466–464.

[2] ZAMBIASI, S. P.; RABELO, R. J. A proposal for reference architecture for personal assistant software based on soa. *IEEE Latin America Transactions*, 2012.

[3] MARKOFF, J. *A Software Secretary That Takes Charge*. New York Time, 2019. Disponível em: <http://www.nytimes.com/2008/12/14/business/14stream.html?_r=1&scp=7&sq=personal%20assistant&st=cse>.

[4] ZAMBIASI, S. P.; RABELO, R. J. Uma arquitetura de referência para softwares assistentes pessoais baseada em agentes e soa. *WESAAC'2011: V Workshop – Escola de Sistemas de Agentes, seus Ambientes e Aplicações*, Curitiba (PR), Brasil, 2011.

[5] WEIZENBAUM, J. Eliza - a computer program for the study of natural language communication between man and machine. In: CHIEN, A. A. (Ed.). *Communications of the ACM*. [S.l.]: CACM, 1966. v. 9, n. 1, p. 36–45.

[6] ABUSHAWAR, B.; ATWELL, E. Alice chatbot: Trials and outputs. *Comp. y Sist.*, v. 19, n. 4, p. 625–632, 1993.

[7] ALEXA. *Developer documentation: browse the technical documentation for Alexa, Amazon Appstore, and devices*. 2019. Disponível em: <https://developer.amazon.com/documentation>.

[8] TINGIRIS, S. *Amazon Alexa development 101*. 2018. Disponível em: <https://youtu.be/QkbXjknPoXc>.

[9] AWS. *AWS Lambda*. 2019. Disponível em: <https://aws.amazon.com/pt/lambda>.

[10] MAZON, S. *Desenvolvendo Chatbots com Watson Conversation*. IBM Developer, 2018. Disponível em: <https://www.ibm.com/developerworks/br/library/desenvolvendo-chatbots-com-watson-conversation/index.html>.

[11] DIALOGFLOW. *Dialogflow: build natural and rich conversational experiences*. 2019. Disponível em: <https://dialogflow.com>.

[12] AZUREBOT. *Serviço de bot do Azure: um serviço gerenciado criado especificamente para o desenvolvimento de bot*. 2019. Disponível em: <https://azure.microsoft.com/pt-br/services/bot-service/>.

[13] PANDORABOTS. *Build intelligent conversational agents on the leading platform*. PandoraBots, 2019. Disponível em: <https://home.pandorabots.com/>.

[14] RUSSELL, S.; NORVIG, P. *Inteligência Artificial*. 3. ed. Rio de Janeiro: Editora Campus / Elsevier, 2013.

[15] VENTURA, R. *Desenvolvimento de um chatbot para apoio a secretarias de cursos de universidades: um caso no programa de Pós-Graduação em Engenharia Mecânica*. Universidade Federal de Santa Catarina, Departamento de Automação e Sistemas: Relatório de Estágio, 2018.

[16] FILHO, L. S. *Estratégia para apoiar o povoamento do repositório institucional de uma universidade de santa catarina utilizando um assistente virtual*. Universidade do Sul de Santa Catarina: Relatório de Estágio, 2018.

[17] UNISULHOJE. *Assistente digital que auxilia repositório institucional é apresentada na Junic*. Unisul Hoje, 2018. Disponível em: <http://hoje.unisul.br/assistente-digital-que-auxilia.-repositorio-institucional-e-apresentada-na-junic/>.

[18] GHIDINI, I.; MATTOS, W. W. *Desenvolvimento e aplicação de um chatbot para auxiliar o atendimento ao cliente*. Universidade do Sul de Santa Catarina, Sistemas de Informação: Monografia, 2019. Disponível em: <http://www.riuni.unisul.br/handle/12345/5986>.

[19] VENTURA, R. *Desenvolvimento de um assistente virtual híbrido com propriedades de chatbot e de automação de processos de negócios*. Universidade Federal de Santa Catarina, Departamento de Automação e Sistemas: Monografia, 2019.