Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

MOODLEBOT: um chatbot baseado no IBM Watson para o Moodle

Marcos Vinícius Timóteo Nunes

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:

Bruno Cerqueira Hott

COORIENTAÇÃO:

Rafael Frederico Alexandre

Setembro, 2021
João Monlevade–MG

Marcos Vinícius Timóteo Nunes

MOODLEBOT: um chatbot baseado no IBM Watson para o Moodle

Orientador: Bruno Cerqueira Hott

Coorientador: Rafael Frederico Alexandre

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Setembro de 2021

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

N972m Nunes, Marcos Vinicius Timoteo .
MOODLEBOT [manuscrito]: um chatbot baseado no IBM Watson para o Moodle. / Marcos Vinicius Timoteo Nunes. - 2021.
67 f.: il.: color., tab..

Orientador: Prof. Me. Bruno Hott.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Sistemas de Informação .

1. Aprendizado do computador. 2. Ensino à distância. 3. IBM (Computadores) . 4. Sistemas operacionais distribuídos (Computadores) . 5. Software de aplicação - Desenvolvimento. I. Hott, Bruno. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.775

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Marcos Vinícius Timóteo Nunes

MoodleBot: um chatbot baseado no IBM Watson para o Moodle

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação

Aprovada em 01 de novembro de 2021

Membros da banca

Me. Bruno Cerqueira Hott - Orientador (Universidade Federal de Ouro Preto)
Dr. Rafael Frederico Alexandre - Coorientador (Universidade Federal de Ouro Preto)
Dr. Bruno Pereira dos Santos (Universidade Federal de Ouro Preto)
Me. Tales Mota Machado (Universidade Federal de Ouro Preto)

Bruno Cerqueira Hott, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 11/12/2021

Documento assinado eletronicamente por **Bruno Cerqueira Hott**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 11/12/2021, às 21:11, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0215709** e o código CRC **32A9C29A**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Este trabalho é todo dedicado aos meus pais, amigos, e a todos que me ajudaram diretamente e indiretamente durante o curso. Dedico também à Universidade Federal de Ouro Preto e todo seu corpo docente, discente e administrativo.

Agradecimentos

Este é um agradecimento aos meus irmãos, Hudson Oberdan Nunes e Reginaldo Anderson Andre Timoteo, ao meu Pai Vitor Celso Nunes e principalmente à minha mãe, Elizabeth Timoteo Nunes. Desta forma, deixo aqui os meus agradecimentos por terem permanecido este tempo todo ao meu lado, não somente durante este percurso, mas por estarem em todos os momentos da minha vida.

Sou grato ao meu orientador, Bruno Cerqueira Hott, pelo acompanhamento durante a execução do projeto, e pela proatividade em relação as reuniões e encontros. Expresso também minha gratidão aos meus amigos, companheiros de curso, à Universidade Federal de Ouro Preto, e todo o Departamento de Computação e Sistemas, pois todos eles foram importantes durante esta caminhada.

"Science is more than a body of knowledge; it is a way of thinking."

— Carl Sagan (1934 – 1996),
in: The Demon-Haunted World: Science as a Candle in the Dark.

Resumo

O ensino à distância (EAD) é uma realidade nas escolas e universidades impulsionada pelo isolamento social vivido pela sociedade atualmente. Neste modelo de ensino, é importante manter um ambiente de aprendizagem virtual (AVA) adequado. Alunos e professores precisam conhecer as utilidades e limitações do ambiente de aprendizagem. Neste trabalho apresentamos uma solução que visa a aproximação do usuário aos serviços do sistema Moodle. Integrando ferramentas de conversação às funcionalidades do Moodle e disponibilizando novas interfaces de comunicação com o usuário. Para o desenvolvimento deste sistema foram utilizadas tecnologias de nuvem presentes na plataforma IBM Cloud, possibilitando a utilização de processamento de linguagem natural. Uma conexão Moodle e WhatsApp foi possibilitada pelos serviços de integração e orquestração de API's, conectando o usuário às funcionalidades do Moodle através de um microsserviço de acesso ao banco de dados do sistema. O funcionamento do Moodlebot foi avaliado através de diálogos iniciados por um usuário através de um celular e um notebook, simulando requisições ao Moodle e obtendo informações do banco de dados da aplicação. Ao final deste trabalho concluímos que, através de um chatbot integrado à ferramentas de gerenciamento acadêmico, propiciamos um ambiente coeso entre usuário e aplicação, gerando novas formas de utilização da aplicação.

Palavras-chaves: moodlebot, IBM Watson, chatbot.

Abstract

E-learning is a reality in schools, and was stimulated by the nowadays social isolation. In this learning model, is important to keep an adequate E-learning platform. Students and teachers need to know the usefulness and limitations of this tool. In this work we present a solution that aims to bring the user closer to the utilities of the Moodle system. By integrating conversation tools and providing new user communication interfaces. For the development of this system, we used technologies present in the IBM Cloud platform, enabling the use of natural language processing. A Moodle and WhatsApp connection was made possible by the API's integration and orchestration services, connecting the user to the Moodle functionalities through a microservice to access the system's database. In order to evaluate Moodlebot we simulate a variety of dialogs and requisitions between a student and the Moodle platform Finally we conclude that, by integrating a chatbot and a E-learning platform, we provided a cohesive environment between the users and the platform, generating new ways of using the application.

Key-words: moodlebot, dialog, conversation.

Lista de ilustrações

Lista de algoritmos

Lista de tabelas

Sumário

1 Introdução

Automatizar sistemas, rotinas e atendimentos vem se tornando necessário para a grande maioria das organizações. Visto que as pessoas estão cada vez mais voltadas a procedimentos práticos e de usabilidade simples. O processo de automatização requer conhecimento prévio do fluxo do processo e isso auxilia na obtenção das melhores respostas às diversas situações. Um exemplo de automatização é a utilização de chatbots para atendimentos aos usuários.

Nesta monografia, trabalhamos com a implantação de um chatbot, nomeado como Moodlebot que utiliza técnicas de processamento de linguagem natural para a compreensão de mensagens de texto. O Moodlebot será implantado para auxiliar as atividades do ambiente Moodle. A proposta é utilizar uma estrutura de comunicação automatizada a fim de aprimorar o atendimento em relação ao ambiente de aprendizagem virtual (AVA) utilizado pela comunidade acadêmica.

Chatbots são programas de computador que realizam tarefas específicas por meio de conversação textual ou por voz. Esta ferramenta é muito utilizada em atendimentos baseados em perguntas e respostas (HAN.N JONG PARK.H; LEE, 2018). Este modelo de ferramenta pode substituir o atendimento humano, trazendo também uma escalabilidade maior em relação ao número de pessoas atendidas simultaneamente. Com a utilização de técnicas de Big Data, o chatbot processa e obtêm a melhor resposta ao usuário.

Segundo (Eduardo GIGCH; PIPINO, 2017), Big Data e o aumento do número de aplicações de geração de dados com internet das coisas, mídias sociais e soluções cognitivas, estão definindo as estratégias de negócios contemporâneos. Toda esta evolução trouxe uma proximidade maior entre os dispositivos eletrônicos e os usuários. Esta é a ideia básica da computação ubíqua: que computadores e demais dispositivos estejam cada vez mais inclusos no cotidiano das pessoas, para que passem a não apenas coexistir com os indivíduos, mas sim conviver com eles de forma direta (WEISER, 1991).

Neste trabalho vamos abordar o *Moodle*[1] como a ferramenta de alvo de automatização. O Moodle é um projeto de software aberto, sob General Public License (GNU), podendo ser baixado, utilizado e modificado gratuitamente (BECHARA; HAGUENAUER, 2010). No Moodle pode-se criar cursos, alunos, professores, disciplinas, gerenciamento das atividades, notas, além de relacionar disciplinas com alunos e professores.

A utilização dos chatbots tem como intuito aproximar o usuário de uma resposta automatizada e em tempo real. Pensando nisto, este trabalho propõe o desenvolvimento de um chatbot para integrar o Moodle ao aplicativo de mensagens WhatsApp, utilizando

[1] https://moodle.org/

o IBM Watson como ferramenta de processamento das mensagens.

O aplicativo *WhatsApp*[2] é uma ferramenta popularizada no Brasil, chegando a mais de 120 milhões de usuários no Brasil (TORRES[1] et al., 2008). Em razão disto, se tornou o serviço de conversação escolhido para este trabalho. Aliado à ferramenta Moodle, o Moodlebot se tornará um serviço disponível 24 horas por dia e 7 dias por semana.

Para processamento das mensagens trafegadas durante a comunicação, utilizamos a ferramenta *IBM Watson*[3]. O IBM Watson é um serviço que utiliza de processamento de linguagem natural para identificação de contextos em mensagens de texto estruturadas e não estruturadas, gerenciando também as mensagens de resposta enviadas aos usuários. O IBM Watson é um serviço que pode ser provisionado no ambiente de nuvem da IBM.

Na Figura 1 podemos visualizar a estrutura do projeto de forma simplificada. As mensagens são enviadas pelo usuário através do WhatsApp, sendo posteriormente processada pelo IBM Watson, identificando a intenção do usuário. Ao identificar a solicitação, o sistema faz acesso ao Moodle em busca dos dados que serão retornados ao usuário através de uma mensagem de resposta elaborada pelo Watson e enviada ao WhatsApp do usuário.

Figura 1 – Arquitetura da comunicação - Desenvolvida pelo autor

[2] https://whatsapp.com/
[3] https://ibm.com/watson

2 Referencial Teórico

Nesta seção, serão apresentados o referencial teórico e as ferramentas que serviram como base para o desenvolvimento deste trabalho. Estas ferramentas foram adotadas, tendo como principal objetivo, a implementação do chatbot e sua estrutura de comunicação.

2.1 Inteligencia Artificial - IA

Durante muito tempo o homem buscou uma forma de replicar suas formas de trabalho a fim de aumentar sua produtividade. O pensamento se baseou em identificar e mapear as características do pensamento humano, abrangendo raciocínio lógico e capacidades de resoluções de problemas. Neste conceito surgiram técnicas de inteligência artificial (IA) a fim de estudar a capacidade de algoritmos na resolução destas atividades.

Projetos incluindo inteligência artificial surgiram inicialmente com intuitos militares durante a segunda guerra mundial (TEIXEIRA, 2014). Com o avanço das tecnologias de bombardeio desenvolvidas pelo exército nazista, exércitos aliados se encontraram na necessidade de se defender de forma mais eficaz. Desta forma com a utilização da inteligência artificial, sistemas de rastreio foram desenvolvidos para corrigir os desvios causados por manobras do alvo e dos abatedores, possibilitando um disparo preciso contra os bombardeiros.

Inteligência artificial é um ramo de pesquisa caracterizado pela busca de soluções tecnológicas voltadas para execuções de tarefas de forma inteligente. Para o desenvolvimento destas tecnologias são utilizadas técnicas como redes neurais artificiais, processamento de linguagem natural (PLN), aprendizagem de máquina, entre outros. Redes neurais é o ramo que busca o desenvolvimento de algoritmos utilizando a aproximação de modelos matemáticos às redes neurais biológicas e desenvolver no algoritmo uma capacidade similar do ser humano em entender e reconhecer padrões (KOVÁCS, 2002).

PLN conceitua de forma computacional a análise da forma humana de se comunicar, seja ela textual ou sonora, trabalhando com a compreensão de sentenças através da análise de sintaxe, semântica e contexto. Pode-se dizer que PNL aproxima a lógica computacional às formas de comunicação da linguagem humana (GONZALEZ; LIMA, 2003).

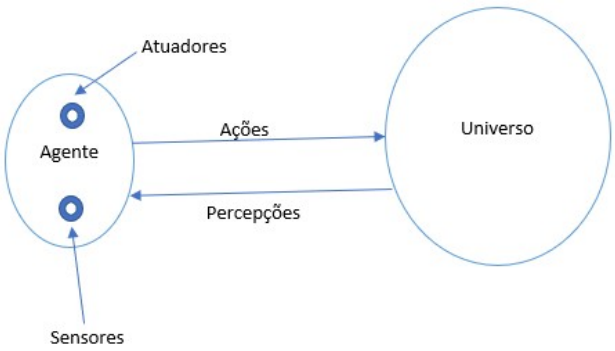
2.1.1 Aprendizagem de máquina - AM

Aprendizagem de máquina é um subconjunto da inteligência artificial voltado à implementação de algoritmos baseados em modelos analíticos capazes de supervisionar, predizer e aprender com as decisões do próprio algoritmo (Russel, Stuart NORVING,

2004). Em outras palavras, sistemas baseados em aprendizagem de máquina são capazes de modificar o seu comportamento com base em suas experiências na resolução de um problema, demonstrando assim o aprendizado realizado pelo algoritmo. Através do que se considera algoritmos inteligentes, a aprendizagem de máquina busca a compreensão do universo em que atua, aprimorando sua lógica de execução com o aprendizado obtido durante suas ações. A modificação de um comportamento para algoritmos baseados em aprendizagem de máquina está associada às tarefas de reconhecimento de padrões nos dados analisados.

A definição dos agentes inteligentes é importante para os modelos de aprendizagem de máquina, pois é através destes agentes que um algoritmo realiza a interação entre suas ações e o universo em que foi aplicado. Um algoritmo inteligente é formado por componentes denominados de agentes inteligentes, relacionado à capacidade do mesmo de lidar com as próprias ações e aprender com elas, assim possibilitando que identifique possíveis consequências para cada passo realizado. Os agentes são considerados propriamente inteligentes quando adquirem a capacidade de observar e identificar possibilidades de evoluções em suas decisões. É de extrema importância a compreensão do agente em relação ao ambiente em que se está operando, esse fator é um facilitador para que o agente possa coletar maiores informações sobre as entradas e saídas que se esperam do algoritmo (Russel, Stuart NORVING, 2004). Como podemos verificar na Figura 2, o agente inteligente é composto por sensores e atuadores. Os sensores são capazes de identificar situações no universo observável para o algoritmo, desta forma o sensor obtêm percepções em relação ao ambiente, ajudando na tomada de decisão do agente. Os atuadores possuem a finalidade de analisar o que os sensores identificaram e dessa forma realizar uma ação no universo em que foram aplicados.

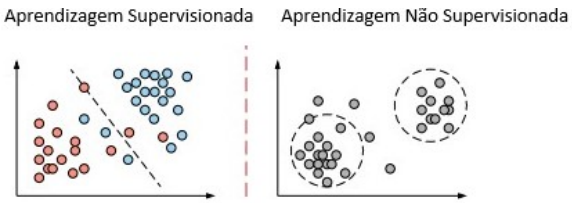
Figura 2 – Agentes Inteligentes



No contexto da aprendizagem de máquina há dois modelos principais; o modelo supervisionado e o não supervisionado. Podemos verificar a diferença entre estes dois métodos de aprendizagem na Figura 3. Na aprendizagem supervisionada o algoritmo recebeu os rótulos em relação às amostras de entrada, possibilitando a classificação dos dados em dois conjuntos diferentes. Após realizar o treinamento sobe os dados de treino

conhecidos, o algoritmo prediz os novos dados de entrada através da função aprendida para mapeamento. A saída para este algoritmo é o conjunto de dados classificados através de uma função de mapeamento. Em relação à aprendizagem não supervisionada, podemos visualizar na Figura 3 que somente os dados de entrada são informados, não contendo os rótulos. Portanto, o algoritmo não possui informações iniciais sobre os dados de entrada, seu trabalho é identificar uma função para descrever os dados não rotulados. Estes padrões podem ser definidos através de características similares entre os dados, como por exemplo distância a um dado central. A finalidade deste algoritmo é o retorno dos rótulos associados aos dados de entrada.

Figura 3 – Aprendizagem supervisionada e não supervisionada



2.1.2 Processamento de linguagem natural - PLN

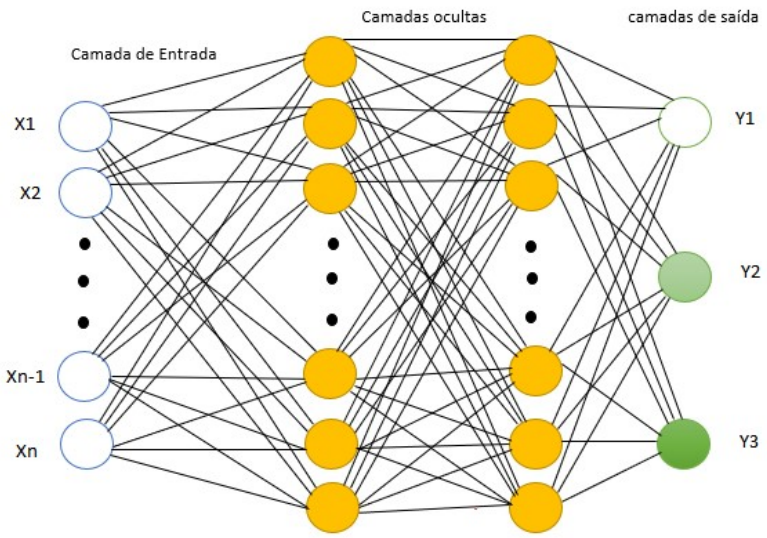
Processamento de linguagem natural (PLN) é o estudo das técnicas cognitivas de reconhecimento de linguagens não estruturadas, além de incluir a identificação de padrões de comunicação por voz. Os primeiros estudos em relação a NLP ganharam força entre nos anos de 1950. Durante este período Alan Turing desenvolveu um questionamento baseado na capacidade da máquina de se comportar ou responder equivalente ao raciocínio humano e sua percepção (DENG; LIU, 2018). Este teste foi nomeado como *teste de turing* e buscava a avaliação da eficiência da máquina durante um diálogo entre três participantes, sendo dois deles humanos, um deles com o papel de interrogador e uma máquina projetada para retornar respostas a partir de um questionamento. O resultado do teste depende da avaliação do interrogador, caso ele não seja capaz de distinguir através das respostas quem é a máquina, o teste é considerado como positivo, sendo assim a máquina passou no teste.

Em 1954, uma parceria entre a University of Georgetown e a IBM, deu início a um novo experimento que trazia com ele evolução em relação aos primeiros estudos de NLP. O experimento consistiu-se na tradução de dezenas de frases em Russo para o idioma Inglês e visava a disseminação do tema perante a comunidade cientifica e instituições privadas.(HUTCHINS, 2004). Para a realização da tradução das sentenças, foi utilizado o IBM 701, considerado o primeiro computador eletrônico fabricado em larga escala pela IBM. Inicialmente os sistemas não possuíam ferramentas sofisticadas e nem mesmo um número vasto de regras gramaticais, portanto o processamento da máquina não conseguia se aprofundar nas estruturas das formações das sentenças e vocabulários de um idioma.(SCHWITTER, 2007).

Perante à grande necessidade de maior agilidade e processamento dos algoritmos artificiais, se torna um trabalho árduo encontrar ou desenvolver ferramentas capazes de abstrair a forma exata ou, até mesmo aproximada, em que o usuário quer se comunicar. Desta forma, técnicas de aprendizagem profunda foram adicionadas ao desenvolvimento de aplicações baseadas em NLP. Aprendizagem profunda ou *deep learning* é normalmente definida como a utilização de redes neurais profundas para processamento de dados, utilizando-se de uma arquitetura multicamadas.

A arquitetura de multicamadas em aprendizagem profunda geralmente é composta por camadas de entrada, camadas ocultas e camadas de saída, estas informações podem ser verificadas na Figura 4. Nas camadas de entrada estão concentrados os neurônios de entrada, na figura foram representados como variáveis "Xi" estes neurônios são responsáveis pelo recebimento dos dados de entrada do algoritmo. As camadas ocultas são responsáveis. O termo camada oculta, ou camada profunda, indica, que está camada não é visível para o solicitante ou observador, sendo assim uma camada interna do algoritmo. Um algoritmo de aprendizagem profunda pode conter uma ou mais camadas ocultas e, à medida que estas camadas crescem, mais precisos podem ser os dados de saída do algoritmo. Por fim tem-se as camadas de saída contendo os neurônios de saída, representados como "Yi". As camadas de saída podem conter um ou mais neurônios de saída, atribuindo a cada um deles um valor de previsão desejado como resposta aos dados processados na camada profunda.

Figura 4 – Deep Learning - Multicamadas

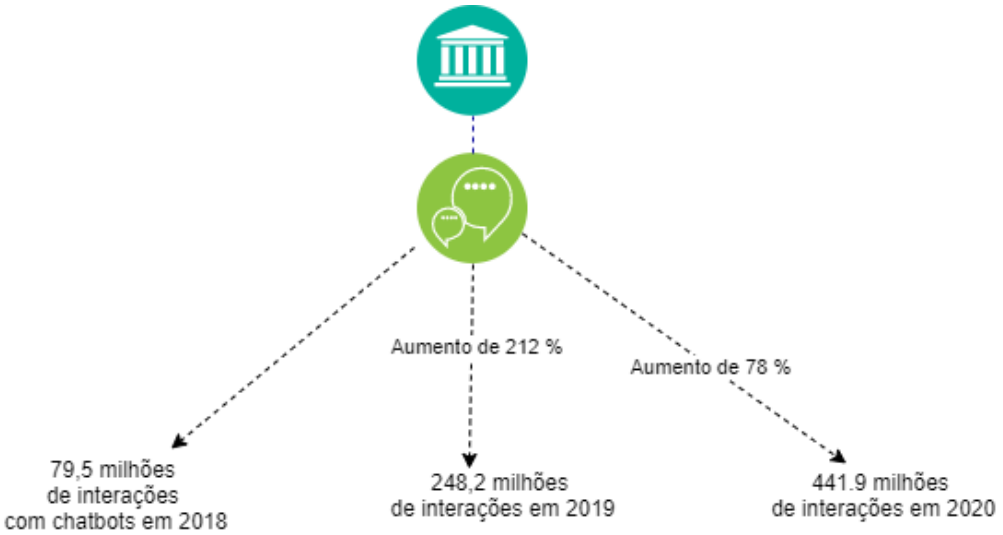


2.2 Chatbot:

Chatbots são sistemas que simulam o atendimento humano a fim de melhorar a experiência do usuário. Estes sistemas trabalham com ferramentas de reconhecimento e

interpretação de linguagem natural, podendo processar mensagens textuais ou voz. Para cada frase aplicada durante um diálogo, haverá uma resposta associada no chatbot. A aplicação de chatbots em atendimentos automatizados possibilita ganho em termos de escalabilidade. A programação dos Chatbots para problemas reais é algo amplamente utilizado, seja nas instituições financeiras, comércio eletrônico, saúde, educação, entre outros. Nos tempos atuais, várias são as instituições financeiras que possuem chatbots como serviço de atendimento ao cliente, se tornando um dos setores mais atuantes na utilização da ferramenta. O banco Bradesco[1], principal banco privado do Brasil, utiliza desde 2014 um sistema baseado em inteligência artificial nomeado como *BIA*[2]. BIA é um serviço de atendimento virtual acionado por voz ou texto disponibilizado de forma gratuita para os clientes Bradesco, compatível para dispositivos móveis e tem o objetivo atender os clientes do banco perante a dúvidas em relação a produtos e serviços. Da mesma forma, o Banco do Brasil[3], utiliza um chatbot desenvolvido através do Watson Assistant da IBM, para atendimento aos seus clientes, incluindo opção de diálogo através do WhatsApp. O chatbot do Banco do Brasil auxilia para que o cliente possa realizar, até então, 15 operações pelo WhatsApp. Entre as operações estão transferência entre contas, recarga de celular, consultas de saldo, extratos de conta corrente, liberação de poupança e investimentos, entre outros. Segundo a própria instituição, desde a integração desta ferramenta foram realizadas mais de 220 mil transações.

Figura 5 – Relação do crescimento de interações através de chatbot em instituições financeiras a partir de 2018 segundo a Federação Brasileira de Bancos (FEBRABAN).



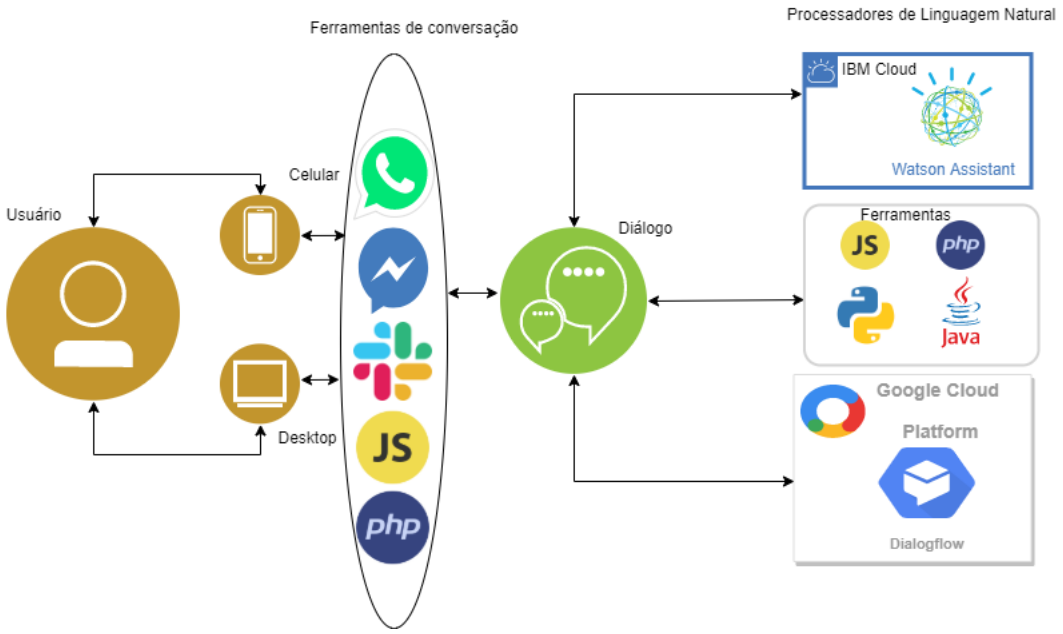
Podemos visualizar na Figura 5 um grande aumento em relação ao uso de chatbots para a comunicação de clientes com instituições financeiras. Segundo a *Federação Brasileira de Bancos (FEBRABAN)*[4], houve um crescimento no número de iterações com chatbots

[1] https://banco.bradesco/
[2] https://banco.bradesco/canaisdigitais/conheca-bia.shtm
[3] https://www.bb.com.br/
[4] https://portal.febraban.org.br/pagina/3106/48/pt-br/pesquisa

durante o período de 2018 à 2020, ainda segundo o órgão fiscalizador o aumento de 2019 ocorreu por fatores relacionados a políticas de inclusão digital e acesso a informação. Já os aumentos relacionados a 2020 estão diretamente relacionados ao crescimento do mercado digital e atividades remoto devido a pandemia do Covid 19, portanto trazendo uma necessidade de adaptação das instituições e seus clientes. De acordo com a Figura 5, 2019 em números gerou 168.7 milhões de interações a mais que em relação a 2018, já em 2020, o aumento em relação ao ano anterior foi em 193.7 milhões.

Esta tecnologia também pode beneficiar as instituições acadêmicas, automatizando, e permitindo um atendimento 24 horas para o processo de atendimento individual em relação a cada aluno. Chatbots são ferramentas muito úteis na otimização do tempo já que estes algoritmos não sofrem com o congestionamento humano. Sua utilidade também atinge o fator de economia de recursos já que o sistema pode reduzir consideravelmente os custos de funcionários voltados para esta tarefa.

Figura 6 – Exemplo de arquitetura de um ChatBot



Podemos visualizar na figura 6 uma arquitetura para desenvolvimento de chatbots. Nesta arquitetura o usuário possui formas distintas de iniciar um diálogo, através do celular ou computador. Este aparelho deve possuir instalado alguma ferramenta de conversação, como WhatsApp, Facebook Messenger, Slack, Telegram, entre outros. Estas ferramentas serão responsáveis pelo *front end* da aplicação. Após iniciar um diálogo com o aparelho, o sistema definido para o *back end* processa as informações de entrada e inicializa a atividade de definição da resposta ao usuário, as ferramentas de processamento da linguagem variam entre sistemas prontos e desenvolvidos pela equipe de projeto. Dentre as possibilidades existem ferramentas como o Watson Assistant e DialogFlow que funcionam como um intermediário entre o usuário e o chatbot, nestas ferramentas se encontram encapsuladas

as premissas de análise da linguagem e contexto das mensagens.

2.3 Ambientes Virtuais de Aprendizagem (AVA)

Entende-se ambiente virtual de aprendizagem (AVA) como um sistema desenvolvido para auxiliar na distribuição e organização de conteúdos variados em um ambiente de aprendizagem. Ambientes de aprendizagem se caracterizam como um espaço que possibilita a transformação do conhecimento construído pelo ser humano (DUARTE et al., 2003). Estas ferramentas são de extrema importância no cenário acadêmico, pois funcionam como um sistema auxiliar para processo de aprendizagem. Neste sistema é possível que se organize atividades, defina cronogramas, diálogo através do fórum, como biblioteca, quadro de anúncios, entre outros.

Ferramentas AVA são comuns no auxílio a universidades e instituições que mantém seu cronograma de estudos à distância, pois suas funcionalidades facilitam o controle do calendário acadêmico destas instituições. Segundo a Associação Brasileira de Educação a Distância (ABED[5]), uso de ambientes virtuais de aprendizagem tem sido um dos principais fatores que levaram ao rápido crescimento da educação à distância (EAD).

Um momento importante para estas ferramentas e sua valorização, foi o início da pandemia do Corona Vírus ou *SARS-COV-2*[6]. Instituições que utilizavam o sistema sem o aproveitamento de todas as suas funcionalidades, passaram a utilizar o mesmo de forma integral, aliado pela necessidade ao novo cenário acadêmico e as limitações de distanciamento social (Sandra V. M. Pereira GLAUCIA O.A.B.MEIRELES; REI, 2020).

2.3.1 Moodle:

O Moodle é um software de código aberto fornecido gratuitamente sob a GNU *Public License*. O sistema Moodle é protegido por direito autoral, mas oferece permissões para alterações em seu fonte. A geração do ambiente do sistema pode ocorrer através da virtualização com o *Docker*[7], ou clonagem do repositório com o *Github*[8] através do seu repositório oficial. Os recursos do Moodle são distintos para usuários administradores e usuários denominados participantes, como alunos e professores. Para os administradores, o sistema permite as funcionalidades de criação de cursos, categorias, disciplinas, tarefas, atualização de disciplinas e criação de novos usuários participantes. Para alunos, o sistema permite a visualização de disciplinas, atividades, notas e cronogramas. Para professores, o

[5] http://www.abed/
[6] https://g1.globo.com/bemestar/coronavirus/noticia/2020/03/11/oms-declara-pandemia-de-coronavirus.ghtml
[7] https://docker.com/
[8] https://github.com/

sistema dispõe de funcionalidades como, criação de tarefas, aplicações de provas, geração de lista de chamadas, cronogramas de entregas, geração de notas, entre outros.

A escolha do Moodle como ferramenta de gerenciamento das atividades acadêmicas, ocorreu pelo fato da universidade utilizar este sistema em suas atividades. Este fator torna possível uma maior interação com os administradores do ambiente acadêmico da UFOP e uma melhor visualização da situação atual em relação ao seu uso pela comunidade de discentes e docentes.

2.4 IBM Watson Assistant:

Em 1989 a IBM lançava sua máquina automatizada para competições de xadrez Deep Blue, um sistema artificial que respondia comandos de movimentações das peças. Em 1996, o algoritmo venceu o campeão mundial da época, Match Garry Kasparov. Durante a partida o Deep Blue neutralizava as jogadas do adversário, utilizando os seus conhecimentos do tabuleiro e movimentações possíveis. Com um controle de tempo e de jogadas programadas bem acima de um ser humano, o IBM Deep Blue se tornava um dos primeiros algoritmos propriamente testados em um duelo homem contra máquina, se tornando um percursor para a área de desenvolvimento de algoritmos inteligentes(WATSON, 1999).

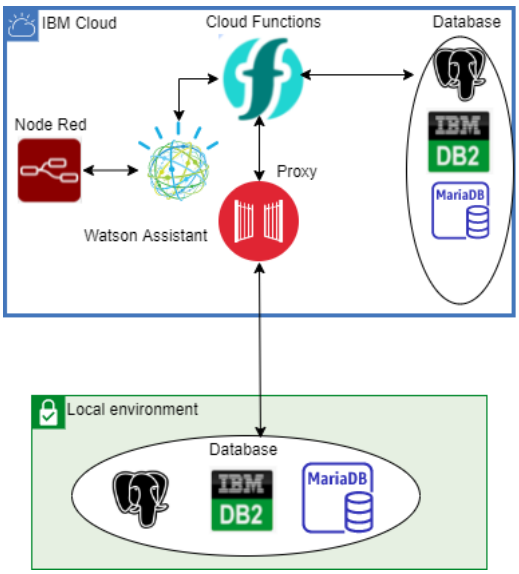
Posteriormente, pensando em substituir o Deep Blue, surgia o IBM Watson. O IBM Watson desenvolve-se na necessidade de aprimorar o algoritmo a fim de desenvolver compreensão da linguagem natural humana. A primeira aparição do IBM Watson de forma concreta, aconteceu em 2011 no *Jeopardy*[9] , um programa televisão baseado em perguntas e respostas. A participação do IBM Watson buscava testar os conhecimentos do algoritmo e sua capacidade de filtrar as milhares de possibilidades de respostas identificadas em arquivos nos seus conjuntos de banco de dados. Utilizando de mais de 90 servidores, para um total de 2.880 núcleos de processadores, o IBM Watson iniciava sua historia[10] de concretização como um sistema baseado em algoritmos artificiais que possuem a capacidade de compreender a linguagem humana.

Hoje, disponível em seu ambiente de nuvem através do IBM Cloud o IBM Watson possui vasta aplicabilidade, que vai desde de serviços de aprendizagem profunda para reconhecimento de imagens a aprendizagem de máquina para compreensão de linguagens e mensagens de voz, tudo isso incluso em seu serviço cognitivo. A comunicação entre cliente e aplicação é estruturada através de um editor de fluxo de conversação integrado ao serviço, neste editor se definem as premissas do diálogo do IBM Watson com as aplicações clientes, como entidades, contextos, nós de diálogos e intenções.

[9] https://www.jeopardy.com/
[10] https://www.ibm.com/ibm/history/ibm100/us/en/icons/watson/

O serviço IBM Watson pode ser instanciado de forma gratuita para uma conta *lite* possuindo funcionalidades limitadas. A conta *lite* do IBM Cloud possibilita acesso a serviços como o próprio IBM Watson, Node Red, serviços de *proxy* para autenticação de acessos, ferramentas *devops* para definição de *pipelines*, entre outros serviços. A limitação para estes serviços é definida pela forma de provisionamento e criação do mesmo. Para contas pré-pagas ou pagas, aumentam o número de serviços que podem ser provisionados, são adicionados serviços como gerenciador de dependências para banco de dados, criação Clusters para ambientes de computação distribuída, além de ferramentas como Kubernets, Red Hat Open Chift, plataformas Blockchain, ferramentas de testes automatizados como Jenkins, entre outras novas opções para o catálogo. De acordo com o nível de escalabilidade que o detentor da conta necessita, os valores variam à medida que os serviços sofrem *upgrade*, por exemplo, caso haja a necessidade de se aumentar o número de mensagens enviadas pelo ChatBot com o serviço IBM Watson Assistant, é preciso que seja realizado uma atualização da conta para realizar o *upgrade* do serviço. No caso do trabalho de conclusão de curso, os serviços instanciados são gratuitos.

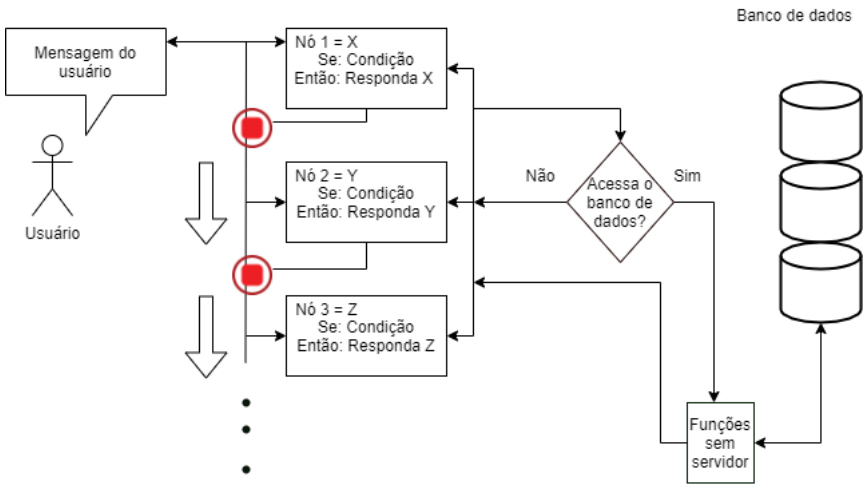
Figura 7 – Arquitetura de um chatbot através IBM Watson Assistant



Podemos observar na Figura 7, uma estrutura de comunicação através do IBM Watson Assistant, onde são necessárias outras ferramentas internas à nuvem, como Cloud Functions, Secure Gateway, e Node Red. Em comunicações com serviços externos à nuvem, casos de WhatsApp, Messenger, Slack, Telegram, entre outros, precisamos incluir o Node Red para gerenciar as informações que chegam dos aplicativos de conversação. O Node Red é um orquestrador responsável por coordenar, através de um fluxo de comunicação, as ferramentas de conversação e o IBM Watson, podendo se requisitar informações de um banco de dados em nuvem ou em um ambiente local. Para que isso seja possível, precisamos utilizar a ferramenta Cloud Functions, executando funções sem servidor para a realização das buscas ao banco de dados. Caso o IBM Watson se comunique com o

ambiente de nuvem, a função sem servidor se redirecionara a um banco provisionado no IBM Cloud. Para comunicações com o ambiente local, é preciso um *proxy* ou túnel de acesso seguro entre o ambiente cloud e ambiente local. Este acesso seguro é possibilitado pelo Secure Gateway, criando uma conexão autenticada entre os ambientes e permitindo que a execução das funções sem servidor obtenha dados externos e os redirecione para o IBM Watson.

Figura 8 – Comunicação Watson Assistant



Através da estrutura baseada em nós de diálogo, o IBM Watson Assistant analisa as condições que acarretarão em respostas para cada nó, estas respostas são processadas a partir da associação de um nó que contém uma intenção abstraída na mensagem do usuário. Desta forma, cada entrada de mensagem de um usuário, acarretará em fluxos diferentes dentro desta estrutura de diálogo. Podemos verificar na Figura 8 um processo de comunicação entre um usuário e o Watson Assistant. Assim que a mensagem é processada pelo serviço, o sistema passa por um processo de identificação do nó que contenha as conjunturas adequadas para esta requisição, neste processo é executada uma análise textual da requisição em relação aos textos definidos como resposta para a intenção. Caso a intenção seja identificada pelo primeiro nó e não haja necessidade de obter informações do banco de dados, a resposta é prontamente retornada de acordo com as mensagens definidas para este diálogo. Caso a intenção seja identificada, mas haja necessidade de retornar dados do banco, funções sem servidor são acionadas a fim de buscar no banco de dados as informações requeridas. Por fim os dados são retornados dentro do diálogo.

2.4.1 Node RED:

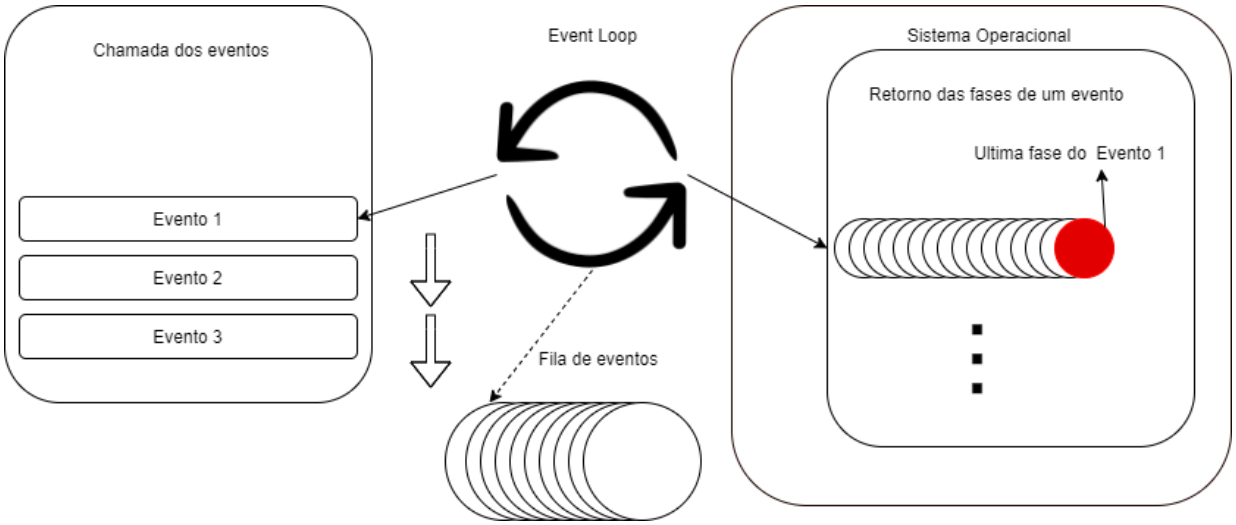
Node-Red[11] é uma ferramenta visual, baseada em nós programáveis e que tem como finalidade a orquestração de dispositivos de hardware e software. Esta ferramenta fornece

[11] https://nodered.org/

uma implementação de fluxos conectados em tempo de execução, através dos serviços construídos à partir da linguagem Node JS para o *back-end* da aplicação. A ferramenta foi originalmente desenvolvida pela IBM para facilitar no processo de recrutamento de pessoas ligadas a área de internet das coisas (IOT). Nos tempos atuais, o serviço da plataforma Node-Red ficam a cargo da *JS Foundation*. O serviço pode ser instanciado na IBM Cloud.

O Node Red trabalha com uma arquitetura que consiste no escalonamento de eventos, esta característica é possível graças ao ciclo de eventos, ou (*event loop*). A utilização de ciclos de eventos do Node Red, se deve ao fato da aplicação ter seu desenvolvimento baseado em Node JS. A linguagem Node JS é voltada para o *back end* de aplicações trabalhando com eventos assíncronos e organizados com *multithread*. Em aplicações *multithread*, o sistema é capaz de processar várias *threads* ao mesmo tempo, desde que cada uma esteja executando um passo diferente, isso faz com que a aplicação continue executando, mesmo que não se obtenha prontamente uma resposta a um evento. Esta lógica se aplica aos nós do Node Red, onde cada nó pode ser executado ao mesmo tempo, desde que executem tarefas diferentes. O fluxo registra um retorno de chamada para cada nó, e este retorno é chamado quando sua ação a ação for concluída.

Figura 9 – Fases do Node Red - Event Loop



Podemos visualizar na Figura 9 a estrutura de um evento no Node Red. Os eventos são enfileirados em uma fila que posteriormente são acionados de acordo com sua ordenação. Cada evento pode conter uma ou mais fases para execução. Portanto, para que o próximo evento seja acionado, todas as fases do processo anterior devem ter sido executadas. À medida que cada fase é executada e seus retornos são identificados, o sistema operacional aciona a próxima fase do evento e, quando todas as fases de um evento são executadas, o próximo evento é acionado e alocado no sistema operacional como uma *thread*. Antes de ser escalonado, cada evento especifica o seu tempo de espera, o seu número de fases, um identificador para o próximo evento e a chamada para este próximo evento. Como as fases

de cada evento podem conter uma ou mais operações, as fases podem ser escalonadas em filas secundárias para que cada operação seja executada individualmente, os seus retornos são armazenados para identificar respectivamente o término das operações da fase, da fase em si e por fim o término do evento. Assim que um evento termina completamente, outro evento é chamado, adicionado na fila e preparado para realizar sua chamada. O escalonamento dos processos de cada evento é realizado pelo sistema operacional onde a aplicação está sendo executada.

2.5 Twillio API:

O *Twillio*[12] é uma plataforma para serviços de integração de API's que possibilita a comunicação de sistemas através de mensagens SMS, MMS, voz, video, entre outros. Entre os aplicativos compatíveis para integração estão WhatApp e Facebook Messenger. Através do Twillio API é possível analisar as mensagens trafegadas, visualizar os logs de comunicação com os resultados de envios confirmados e não confirmados, análises de custo e monitoramento através do *dashboard* do serviço.

Os serviços do Twillio possuem conexões facilitadas com o WhatsApp, portanto a integração entre as duas ferramentas é feita de maneira simples. O Twillio suporta desenvolvimento através das principais linguagens de Back End, Node JS, Python, PHP, .NET, Java e Ruby.

A API do Twillio possui serviços concorrentes similares como, *Zenvia API*[13] e o *Bitrix24*[14]. Os produtos das plataformas Zenvia, Bitrix24 e Twillio, são compostos de integradores para serviços SMS, Voz, e Web chat. As plataformas possuem um serviço do tipo OpenAPI individual que dispõem de recursos para acesso de aplicações externas. As API's Rest disponibilizadas por estes serviços podem enviar mensagens para SMS, WhatsApp, Facebook Messenger, Telegram e Instagram. É importante salientar que as plataformas, apesar de semelhantes, possuem características distintas. As plataformas Zenvia e Bitrix24, além dos serviços de integração, possuem uma central de relacionamentos com o cliente (CRM), além disso a integração destas plataformas com o WhatsApp, necessita de uma versão empresarial do aplicativo, trazendo um custo adicional.

Os eventos de envio de mensagens nestas plataformas são acionados através de WebHooks, que são funções de acionamento de recursos e alteração de estado de aplicações Web. Assim como o Twillio, o Zenvia e o Bitrix24 utilizam de um ambiente SandBox para a criação do canal de comunicação. O SandBox é um ambiente pré configurado, com canais de comunicação compostos por segurança e proteção para mensageiro e receptor. Para a configuração do SandBox é preciso criar uma instância do serviço na plataforma.

[12] https://www.twilio.com/docs/api
[13] https://desenvolvedores.zenvia.com/sms/documentacao/
[14] https://www.bitrix24.com/features/docs.php

Requisitos	**Bitrix24**	**Twillio**	**Zenvia**
Teste grátis	X	X	X
Versão Gratuita	X	X	-
Nuvem SaaS, baseado na web	X	X	X
Desktop - Mac	X	-	X
Desktop - Windows	X	-	X
Desktop - Linux	-	-	X
Celular - Android	X	-	X
Celular - iPhone	X	-	X
Celular - iPad	X	-	X
Mensagem bidirecional	-	X	X
Gestão de Contatos	X	X	X
MMS	-	X	X
Mensagens de texto em massa	X	X	X
Personalização de mensagens	X	X	X
Palavras-chave para celular	-	X	X
Relatórios / análises	X	X	X
Mensagem Agendada	-	X	X
Shortcodes	-	X	X

Tabela 1 – Comparação das ferramentas de integração

Podemos verificar na Tabela 1 a comparação entre estes serviços. O Bitrix24 e o Twillio possuem a possibilidade de realização de testes gratuitos e de forma rápida, bastando apenas o cadastro no sistema e a configuração das integrações, o Zenvia indica em sua documentação que esta possibilidade também existe para a plataforma, mas é necessário um envio de e-mail para o usuário receber uma autorização de liberação de acesso. Em relação a uso das plataformas através de uma versão gratuita, somente o Bitrix24 dispõem desta possibilidade. As três plataformas estão em nuvem e possuem softwares como serviço baseados na web. Em termos de adaptabilidade, o Bitrix24 se destaca, pois é compatível a acessos através do site, aplicativo *desktop* para mac, windows, e linux, além de acessos através de dispositivos móveis com seus apps nativos para Android e IOS. O Twillio não possui estas opções até o momento, o acesso da plataforma através de aplicativos desktop é disponibilizado somente com a utilização de aplicativos terceiros integrados ao Twillio. Os três serviços possuem a capacidade de envio de mensagens de texto em massa, personalização das mensagens enviadas e relatórios de análise.

2.6 WhatsApp:

WhatsApp é uma ferramenta de comunicação que abrange mais de 1,5 bilhões de pessoas conectadas no mundo, possibilitando o compartilhamento de mensagens, arquivos de texto, mídia, entre outros (TORRES[1] et al., 2008). Para sua sua utilização é necessário um smartphone com conexão à Internet. No Brasil cerca de 120 milhões de usuários ativos

utilizam do WhatsApp tornando a ferramenta adequada para o nosso trabalho. (Figueiredo, Carlos, M.,S. NAKAMURA, 2003).

WhatsApp possui sistemas de segurança[15] em relação as mensagens trafegadas em seu sistema. Em abril de 2016, a empresa anunciou em suas redes de comunicação que estaria adotando uma nova política de privacidade, a criptografia de ponta a ponta entre as mensagens enviadas pelos usuários[16]. A criptografia do aplicativo funciona como uma proteção em relação as mensagens trafegadas durante um diálogo entre dois usuários, onde apenas os usuários participantes tem acesso às mensagens.

[15] https://faq.whatsapp.com/general/security-and-privacy/end-to-end-encryption/?lang=pt_br

[16] http://g1.globo.com/tecnologia/noticia/2016/08/whatsapp-atualiza-termos-de-servico-pela-primeira-vez-em-quatro-anos.html

3 Trabalhos Correlatos

Em 1966, Joseph Weizenbaum no MIT criou o Eliza, primeiro chatbot que se aproximou da conversação humana. Dada uma frase de entrada, o Eliza utilizava um conjunto de regras pré-programadas, para identificar palavras-chave e os padrões correspondentes, definindo assim uma resposta apropriada. (WEIZENBAUM et al., 1966). Posteriormente, agindo como um dicionário, Eliza conseguia indexar palavras-chave, guardando um histórico e um possível ranqueamento de suas recorrências. Apesar de ser um estudo embrionário da época, o desenvolvimento do algoritmo Eliza possibilitou um grande avanço para a atividade de processamento de linguagens naturais.

Como pontos positivos do deste projeto podemos citar a contribuição nas técnicas de definições de respostas pré programadas. Este trabalho foi um dos percursores na obtenção da lógica indexar palavras chaves em uma frase recebida pelo chatbot, e posteriormente associar as palavras à padrões existentes. Este raciocínio também é utilizado no IBM Watson através das intenções que representam as palavras chave, e o contexto que associa uma ou mais possibilidades de respostas para esta intenção. De negativo podemos citar a falta de demonstração da arquitetura da aplicação final.

O projeto Tical, abreviação de Tecnologia Interativa Conversacional sobre Assuntos Linguísticos, é baseado em um sistema de reconhecimento e tratamento de linguagens naturais. O objetivo do Tical foi possibilitar um estudo sobre o ALiB (Atlas Linguístico do Brasil) analisando as variações do significado de uma palavra levando em consideração a região do usuário participante do diálogo. O Tical trabalha com o armazenamento de um conjunto de palavras chave e seus sinônimos, variando de acordo com as regiões do Brasil(MORENO et al., 2015).

Para entender a funcionalidade do algoritmo, analisemos a palavra “mandioca”. No Brasil, “mandioca” pode possuir sinônimos associados como, “macaxeira” e “aipim”. Desta forma, durante um diálogo entre Tical e um usuário, o algoritmo busca recolher informações e associar os estados em que as pessoas responderam “macaxeira” ou “aipim”. Este estudo possui um viés geolinguístico, sendo utilizado para reconhecimento de padrões de comunicações locais de cada região do Brasil.

Como ponto positivo deste trabalho podemos citar a utilização do WhatsApp como interface de comunicação com o usuário. Tanto o projeto Tical como o Moodlebot, podem ser acessados pelo aplicativo. Porém os propósitos dos dois sistemas são diferentes, o Tical busca a geração de dados geolinguísticos através de um diálogo composto por perguntas e respostas. Este sistema não possui a finalidade de auxiliar o usuário em relação a um produto ou serviço como o Moodlebot. Em contrapartida, de negativo para o projeto

Tical podemos dizer que o sistema não possui uma grande variedade de possibilidades de entradas pelo usuário. À medida que o diálogo é estático entre uma pergunta enviada pelo Tical, e uma resposta informada pelo usuário, o sistema não identifica uma grande necessidade de processamento e identificação de um contexto para a mensagem respondida pelo usuário. Diminuindo o dinamismo do diálogo entre chatbot e usuário.

O trabalho de (MOTTA, 2019) é uma adaptação de um chatbot desenvolvido por*Farhan Karmali*, um autor indiano que propôs a criação de um chatbot para auxiliar os usuários do Moodle em resoluções de questionamentos e duvidas acadêmicas, podendo ser utilizado de forma genérica para outras instituições.

Este trabalho tem como pontos positivos a demonstração de técnicas para a utilização do DialogFlow na construção de chatbots, trazendo informações de como criar intenções, entidades e contextos na plataforma DialogFlow. Apesar de não trabalhar com a mesma ferramenta de PLN do Moodlebot, o trabalho contribuiu também para o entendimento da importância de se utilizar acionamentos baseados em eventos através de Webhooks, seja para se obter dados de um banco ou de aplicações externas ao ambiente do chatbot. Como ponto negativo citamos a superficialidade das informações, já que o trabalho não possui um artigo associado, portanto não é possível verificar metodologias e demais ferramentas utilizadas no desenvolvimento do chatbot.

O trabalho de (RANAVARE; KAMATH, 2020) propôs a utilização de um chatbot para o auxílio das atividades administrativas em ambientes educacionais. Abordando a utilização da ferramenta Dialogflow para a criação de diálogos e desenvolvimento de respostas para os usuários. Sua conversação é definida através de intenções, entidades e definições de contextos de diálogos.

Na Tabela 2, podemos verificar um comparativo entre o trabalho do autor e o Moodlebot. Os dois chatbots utilizam plataformas de nuvem distintas, enquanto o projeto do autor utiliza do Google Cloud, o Moodlebot é desenvolvido utilizando os serviços o IBM Cloud. Ferramentas PLN também são diferentes entre os dois trabalhos, os autores do artigo, utilizam da ferramenta DialogFlow já o MoodleBot é baseado no Watson Assistant. As duas plataformas utilizam de fluxos baseados em nós de diálogo, contendo em sua estrutura intenções, entidades e contextos. Os dois sistemas são integrados ao site das instituições utilizadas no estudo, em contrapartida o MoodleBot adicionalmente é integrado também ao WhatsApp. Outro fator em comum dos dois estudos é o uso de Webhooks para acionamento de acesso ao back end.

Podemos destacar pontos positivos em relação ao trabalho do autor como por exemplo a utilização de tecnologias distintas às utilizadas no Moodlebot, trazendo novas opções de ferramentas para desenvolvimento de chatbots. O sistema demonstra uma adaptação do chatbot para diálogos por texto e voz. O trabalho ressalta a utilização de um banco de dados para a construção de um diálogo dinâmico entre chatbot e usuário,

utilizando Webhooks para acionamento das requisições ao banco, sendo assim similar ao Moodlebot. Cabe ressaltar que o trabalho do autor se integra somente ao site da instituição, diferentemente do Moodlebot, que além de se integrar ao Moodle pode ser acessado também pelo Whatsapp.

Aplicação	**MoodleBot**	**Ranavare**
Plataforma de nuvem	Ibm Cloud	Google Cloud
Ferramenta PLN	Watson Assistant	Dialogflow
Conversação	Texto	Texto/Voz
Integração	Moodle/Whatsapp	Site da instituição
Quantidade de intenções	7	6
Acionamento de eventos	Cloud Functions/WebHook	WebHook
Banco de dados	MariaDB	Firebase

Tabela 2 – Comparação do MoodleBot com o chatbot do autor Ranavare

O trabalho de (OLIVEIRA et al., 2019) consiste no desenvolvimento de uma aplicação baseada em um chatbot utilizando ferramentas IBM Watson e o sistema Moodle. Nesta aplicação, o chatbot se comporta como um atendente ou facilitador para auxiliar os usuários do Moodle. A interação entre aluno e sistema, acontece através de uma mensagem inserida pelo aluno, iniciando um diálogo com o chatbot, que por sua vez responde à mensagem utilizando técnicas de processamento de linguagem natural inclusos no IBM Watson.

A aplicação resultante é uma integração entre a API do Facebook para envio de mensagens e Moodle. Utilizando se da API Facebook Messenger e um token de acesso da API, integrou-se a ferramenta de comunicação com o Moodle através de um link gerado para se externar o serviço de envio e recebimento de mensagens dentro do ambiente Facebook Messenger.

Podemos citar pontos negativos em relação à este trabalho, como por exemplo a forma em que o sistema processa as perguntas recebidas pelo chatbot. As respostas do chatbot não são customizadas para acessar um banco de dados, pois, o chatbot somente responde com retornos associados às intenções existentes no IBM Watson, diferentemente do Moodlebot. Desta forma, o chatbot se comporta de maneira pré definida, respondendo perguntas já conhecidas pelo mesmo. Como pontos positivos, podemos citar por exemplo a utilização de ferramentas em comum ao Moodlebot. Tanto o Moodlebot, como o projeto do autor utilizam do IBM Watson para a construção do chatbot, assim como a implementação de envio de eventos com o Webthook, e possuem o Moodle como ferramenta de interface contendo o link do chatbot. Podemos visualizar na Tabela 3 um quadro comparativo entre o Moodlebot e o trabalho do autor.

O trabalho de (ROCIO; WESLEY, 2020) implementou um chatbot para um Módulo de Ambientação Online (MAO), que é um modulo pertencente a um curso e

Aplicação	**MoodleBot**	**Assistente FAQ**
Plataforma de nuvem	IBM Cloud	IBM Cloud
Ferramenta PLN	Watson Assistant	Watson Assistant
Conversação	Texto	Texto
Integração	Moodle/Whatsapp	Moodle/ Messenger
Quantidade de intenções	7	2
Acionamento de eventos	Cloud Functions/WebHook	WebHook
Banco de dados	MariaDB	ND

Tabela 3 – Comparação do MoodleBot com o chatbot Assistente de FAQ

voltado inteiramente para a ambientação do aluno à ferramenta de aprendizagem virtual. Este modulo tem a função de preparar os alunos para a realização do curso de forma digital. O projeto propôs a utilização de um chatbot para auxílio aos usuários do Moodle na obtenção de informações acadêmicas, se comportando como um intermediário responsável por responder perguntas frequentes dos alunos da instituição. No projeto foram utilizadas as ferramentas IBM Watson Assistante, juntamente com o Moodle. A integração entre os dois sistemas acontece através de uma aplicação cliente definida para acesso ao IBM Watson.

Podemos citar como pontos positivos deste trabalho a utilização do Moodle como ambiente de aprendizagem virtual, se comportando de maneira similar ao Moodlebot. Outro ponto de destaque é o desenvolvimento de uma aplicação intermediária ao Moodle e o IBM Watson, onde o autor cita a criação de um sistema cliente para acesso ao IBM Watson. Comparando com o Moodlebot este sistema intermediário se comporta como o getaway configurado para possibilitar o acesso do IBM Watson às funcionalidades e dados Moodle. Em contrapartida, podemos destacar como ponto negativo a falta de comunicação com um banco de dados. Ao contrário do Moodlebot, que utiliza de respostas padrões definidas no IBM Watson Assistant, e também de respostas customizadas com informações obtidas no banco de dados Moodle. O trabalho do autor implementa todo o banco de dados da aplicação em respostas padrões definidas no IBM Watson Assistant, sem a comunicação com um banco de dados externo. Podemos visualizar na Tabela 4 um quadro comparativo entre o chatbot UAbot e o Moodlebot:

Podemos concluir que no decorrer da história dos chatbots, vários foram os contextos utilizados para a aplicação destes sistemas. Um destes contextos é a educação, sendo uma área de grande valia para atuação de chatbots. As primeiras formas de aplicação destes sistemas no âmbito educacional baseou-se em chatbots simples e que possuem uma arquitetura não muito complexa. Estes chatbots possuíam uma finalidade voltada a didática, não se atendo a questões atuais, como grande quantidade de usuários, performance, escalabilidade, entre outros.

Nos modelos de chatbots mais recentes visualizamos uma gama maior em relação a

Aplicação	**MoodleBot**	**UAbot**
Plataforma de nuvem	IBM Cloud	IBM Cloud
Ferramenta PLN	Watson Assistant	Watson Assistant
Conversação	Texto	Texto
Integração	Moodle/Whatsapp	Moodle
Quantidade de intenções	7	33
Quantidade de dialógos	7	35
Acionamento de eventos	Cloud Functions/WebHook	ND
Banco de dados	MariaDB	ND

Tabela 4 – Comparação do MoodleBot com o UAbot

ferramentas utilizadas no desenvolvimento do sistema, e também uma arquitetura mais robusta. É possível identificar nos trabalhos correlatos que tecnologias de plataformas distintas e concorrentes, foram utilizadas por autores diferentes, pois apesar de serem de plataformas diferentes, os serviços possuem a mesma finalidade. Como exemplo, podemos citar as ferramentas DialogFlow e IBM Watson, pertencentes a Gloogle e IBM respectivamente. Utilizar uma ferramenta em detrimento à outra, dependerá da arquitetura utilizada pelo chatbot. Quanto maior o número de ferramentas da mesma plataforma de nuvem, mais simples será a integração entre estes serviços, pois são nativos da mesma plataforma.

Outro fator interessante em relação aos trabalhos relacionados foi encontrar aplicações que utilizam prioritariamente o Moodle como ferramenta de gerenciamento acadêmico. Porém, estes projetos foram desenvolvidos para se comportar como um chatbot menos dinâmico, ou seja, com somente respostas padrões definidas no IBM Watson, e não sendo possível obter respostas do chatbot contendo informações do banco de dados Moodle.

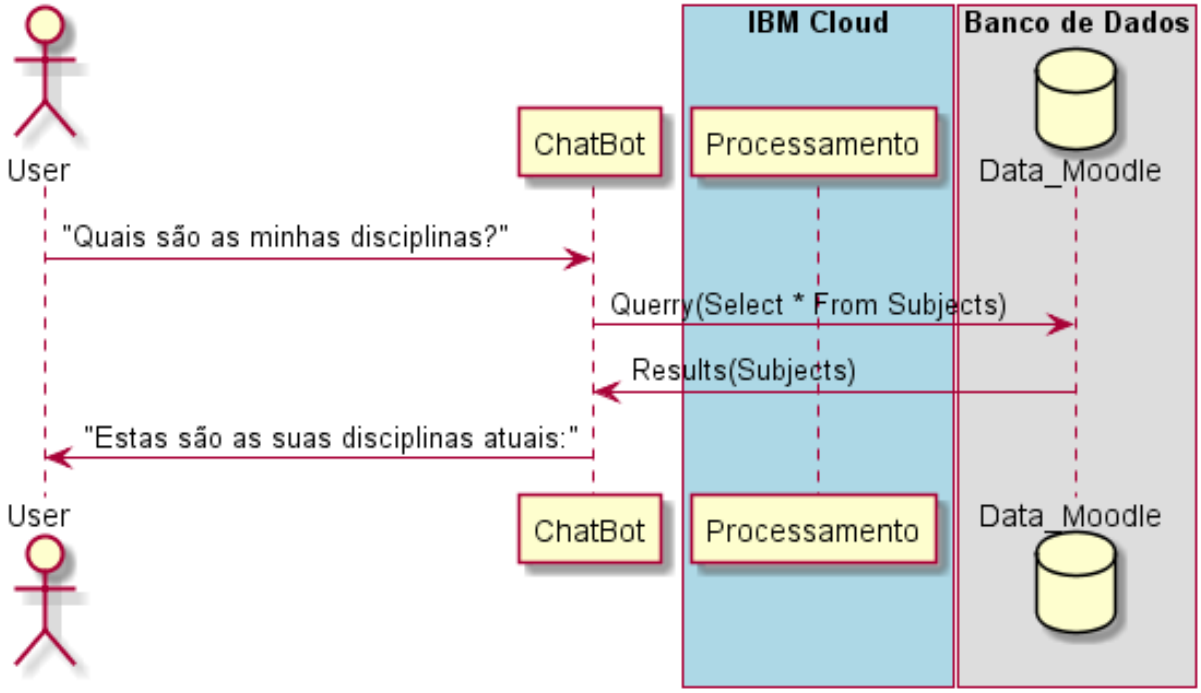
4 Desenvolvimento

Nesta seção, será apresentado o desenvolvimento do trabalho, juntamente com as etapas de configuração do ambiente e ferramentas utilizadas pelo MoodleBot. As ferramentas para execução do ambiente Moodle e desenvolvimento de sua API, foram definidas analisando praticidade de desenvolvimento, documentação e compatibilidade com o ambiente local e nuvem.

4.1 Arquitetura da comunicação do Moodlebot com um usuário:

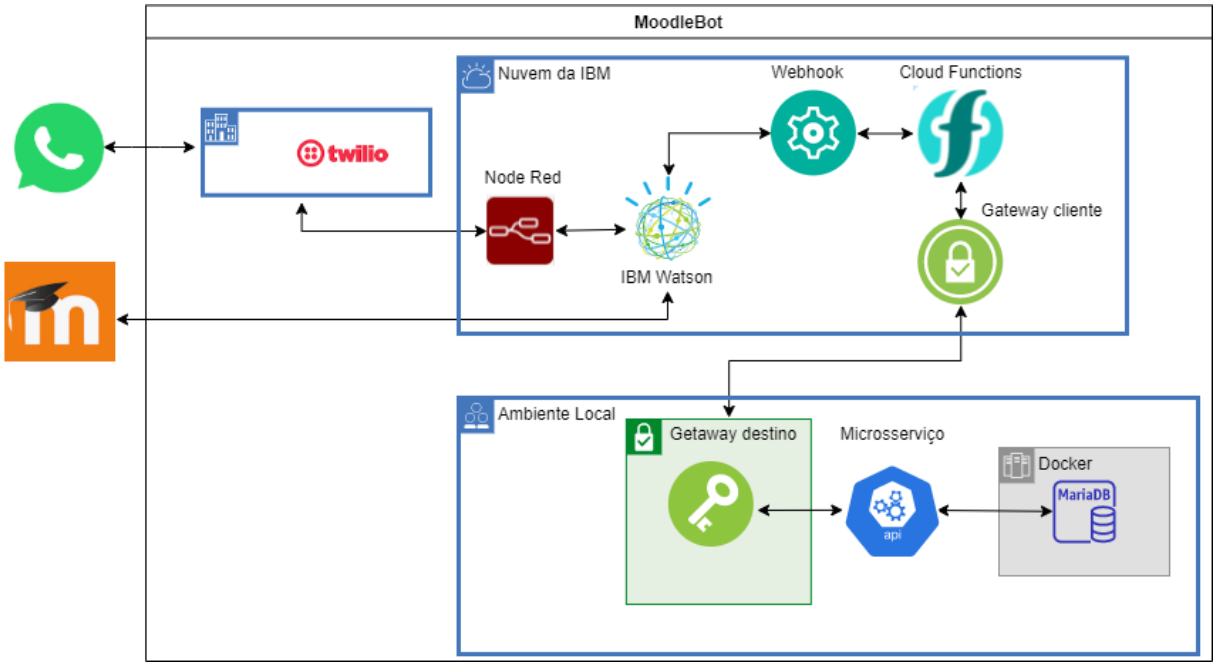
A aplicação desenvolvida permite que o usuário solicite informações através de um serviço de interface, seja ele o aplicativo WhatsApp, ou sistema Moodle versão *web*. Nesta aplicação, o IBM Watson Assistant processa a solicitação do usuário e requisita ao Moodle as informações pertinentes à requisição, como disciplinas, notas, tarefas, entre outras informações. A arquitetura geral da aplicação pode ser explicada pela Figura 10. O usuário requisita informações ao chatbot, essa solicitação é repassada pela plataforma de comunicação, que é então processada pelo IBM Watson Assistant. Para requisições que não necessitem de acesso ao banco de dados, o próprio IBM Watson Assistant possui mensagens pré-programadas de retorno.

Figura 10 – Arquitetura da comunicação



Detalhes da arquitetura do Moodlebot podem ser visualizados na Figura 11, onde temos uma comunicação entre o usuário e o Moodlebot, onde a solicitação do usuário necessita de respostas contendo dados persistidos em um banco. Nesta solicitação, o IBM Watson Assistant processa a mensagem do usuário, acessa o banco de dados e redireciona uma resposta ao usuário.

Figura 11 – Arquitetura do Moodlebot - Integração das ferramentas



Podemos visualizar com mais detalhes a arquitetura do Moodlebot em relação à composição de suas ferramentas. Para requisições obtidas através do Whatsapp, necessitamos do Twillio, uma ferramenta de integração licenciada servindo como um intermediário ao Whatsapp durante uma comunicação. A partir do momento em que Whatsapp e Twillio estão integrados, passamos a necessitar de uma configuração de acesso do Twillio e o IBM Cloud. Desta necessidade surge o Node Red, um serviço responsável por realizar o gerenciamento da comunicação entre Twillio e IBM Watson Assistant, direcionando o fluxo dos dados entre os dois sistemas. Possibilitando que o Twillio acesse o IBM Watson Assistant e receba o retorno das respostas a solicitação do usuário. Conectado ao Node Red, o Twillio passa a obter acesso às funcionalidades do serviço IBM Watson Assistant. Para solicitações através da interface Moodle, a mensagem é redirecionada diretamente para a serviço IBM Watson Assistant, pois sua configuração permitiu acesso direto ao chatbot desenvolvido. O IBM Watson Assistant é a ferramenta provedora do processamento e análise das mensagens do usuário, sua principal funcionalidade é identificar padrões no texto recebido do serviço Twillio e retornar respostas condizentes à mensagem processada.

O IBM Watson Assistant possui dois caminhos para definir respostas às solicitações: respostas customizadas ou padrões. Esta arquitetura aborda o primeiro caso, onde há a

necessidade do IBM Watson Assistant obter dados de um banco. Para que o IBM Watson Assistant identifique um texto que necessita de uma informação do banco, são utilizados webhooks. Como vimos, um webhook é um método de acionamento baseado em eventos, e possui como objetivo o acionar funções externas a um ambiente. No caso desta arquitetura, o webhook é acionado por um evento definido dentro de um diálogo no IBM Watson Assistant. O evento enviado pelo IBM Watson Assistant contém parâmetros utilizados na busca ao banco de dados e o endereço do serviço que o webhook irá acionar, neste caso, o Cloud Functions. O serviço Cloud Functions é composto por funções sem servidor que executam na plataforma de nuvem, mas podem acessar serviços externos a mesma. Para o caso da nossa aplicação, estas funções sem servidor acessarão um microsserviço que executa no ambiente local, possui pontos de entrada expostos para a realização de buscas no banco do Moodle, o MariaDB. O acesso ao microsserviço é possibilitado pela configuração do serviço Secure Gateway, composto por um *gateway* cliente e um destino. Este microsserviço expõe pontos de entrada para ser consumido pelo Cloud Functions, disponibilizando ao ambiente de nuvem o acesso a dados como, disciplinas, notas, tarefas, entre outros. A inclusão do microsserviço na arquitetura exime que as funções de busca ao banco de dados do Moodle sejam inteiramente implementadas no Cloud Functions, diminuindo a necessidade de processamento pela ferramenta.

4.2 Front-End do Moodlebot

O *Front-End* da aplicação é definido pelo WhatsApp e a versão *Web* do Moodle, possibilitando duas formas de interação com o sistema. Ao enviar uma solicitação, sua mensagem é processada pelo sistema, iniciando-se um diálogo entre o usuário e o Moodlebot.

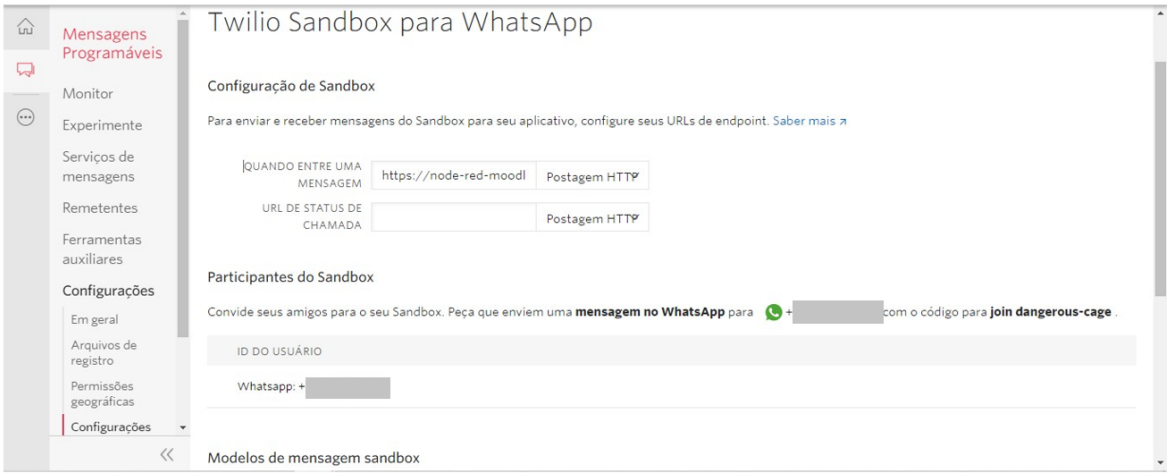
Podemos visualizar na Figura 12 a demonstração da comunicação de um usuário através do WhatsApp, solicitando informações ao Moodlebot. Para a comunicação ocorrer é necessário que o usuário em questão tenha um perfil ativo do aplicativo WhatsApp associado ao seu número de telefone. O usuário começa o diálogo com uma intenção de saudação: “Olá tudo bem”, esta saudação é respondida de forma padrão pelo Moodlebot para este tipo de interação: “Ola, tudo bem? Eu sou o MoodleBot, fui programado para atender suas solicitações! Me diga em que posso te ajudar você? :)”. O diálogo prossegue e o usuário solicita ajuda em relação ao seu login na plataforma Moodle: “Poderia me ajudar com o meu login?”, o Moodlebot informa que: “O seu login é formado pelo seu CPF e sua senha!”. Durante esta comunicação, não foram solicitadas informações do banco de dados do Moodle, portanto o IBM Watson não aciona nenhuma requisição ao banco. Posteriormente o usuário requisita informa que já tentou realizar o acesso ao Moodle, mas que não conseguiu acessar o ambiente: “Já tentei realizar o login, mas não consegui”, o Moodlebot responde para que o usuário realize uma verificação em seus dados de login: “Caso esteja tendo erros em seu `login`, verifique se inseriu o seu CPF corretamente e a

sua credencial!". O diálogo prossegue com uma solicitação do usuário para acesso às suas disciplinas: "E como posso verificar as minhas disciplinas? Preciso saber quais matérias estou fazendo!", desta forma o Moodlebot informa que para que os dados em relação as suas disciplinas sejam requisitados, é preciso autenticar os dados deste usuário: "Insira seu nome de usuário no Moodle para que possamos processar a busca! O seu nome de usuário é contido pelo seu primeiro.ultimo nome!". Este processo é importante para que o Moodlebot possa buscar as informações no banco de dados através de uma chave. O usuário responde a mensagem informando sua chave: "Meu usuário é marcos.nunes", esta chave do usuário é referente ao seu nome de usuário. O Moodlebot responde sua solicitação com os dados requisitados: "Obrigado marcos.nunes, estamos verificando no sistema as os seus dados! As disciplinas para o usuário marcos.nunes são: 'CSI463 - SEGURANCA E AUDITORIA DE SISTEMAS CSI499 - TRABALHO DE CONCLUSÃO DE CUSROCSI567 - PADROES DE PROJETOS CSI503 - DESENVOLVIMENTO AGIL DE SOFTWARE '", terminado diálogo.

Figura 12 – Comunicação através do WhatsApp - Interface WhatsApp

Para que esta comunicação aconteça, é necessário configurar o Twillio, que será a ferramenta integradora do Whatsapp e o IBM Watson Assistant. A configuração do Twillio é composta pela criação de um projeto na plataforma, seguido do cadastro do número de telefone para testes. Tendo o projeto criado, precisamos autenticar a conta Twillio inserindo um número de telefone válido. Após a autenticação, configuramos o aplicativo criado para se comunicar com o Whatsapp através do serviço *Programmable SMS*, este processo pode ser visto na Figura 13.

Figura 13 – Configuração do Sandbox - Twillio



O serviço *Programmable SMS* disponibiliza um ambiente de testes chamado *Sandbox* e nele podemos ativar mensagens automáticas para o número de telefone cadastrado, utilizando mensagens *templates* ou configurando respostas originadas de outros ambientes, para isso temos de inserir a URL do serviço onde as mensagens serão obtidas. Para este trabalho, foi inserida o endereço `https://node-red-moodlebot-marcos.mybluemix.net/moodle`, contendo o domínio do serviço Node Red destinado à conta IBM Cloud utilizada no desenvolvimento do Moodlebot, juntamente com o método `POST` para o envio da mensagem obtida pelo Whatsapp. Este endereço é responsável por inicializar o fluxo do Node Red. Para ativação do ambiente de testes, precisamos enviar o seu código de ativação para o número gerado para o *Sandbox*.

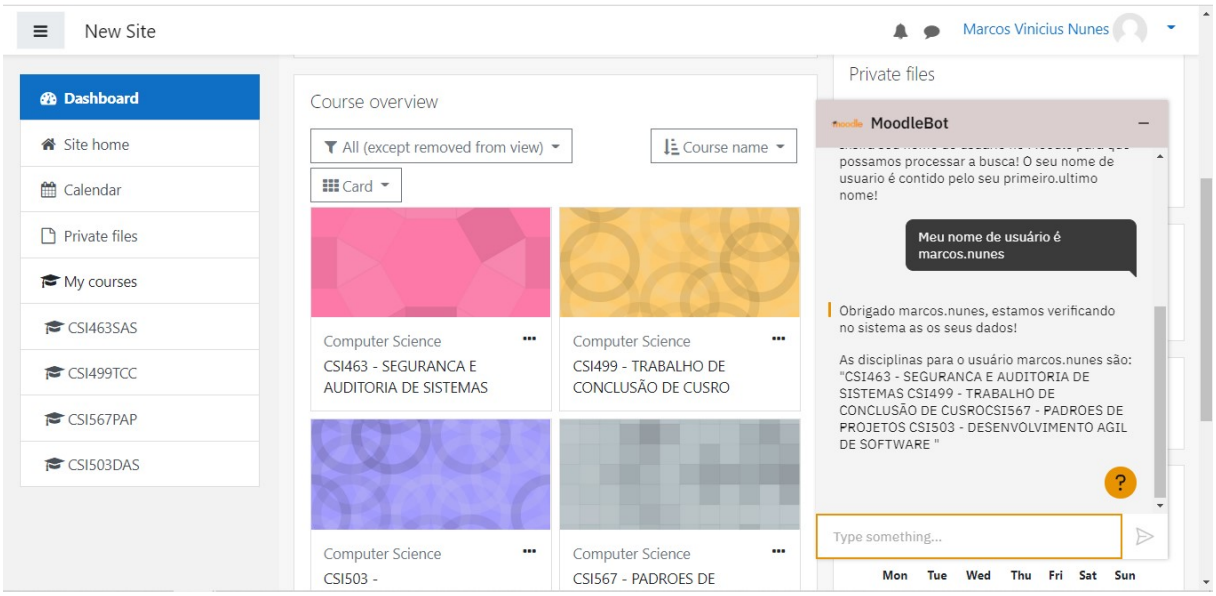
Podemos visualizar na Figura 14, a ativação do *Sandbox* diretamente pelo Whatsapp, e para isso precisamos entrar no aplicativo Whatsapp, e inserir o código de ativação do *Sandbox* disponibilizado pelo Twillio. A partir da ativação do ambiente *Sandbox*, podemos interagir com o chatbot normalmente. Este código pode ser compartilhado com outros usuários do Whatsapp, possibilitando que acessem o Moodlebot. Para que novos usuários acessem o sistema, precisam enviar uma mensagem de ativação, ao número de telefone

Figura 14 – Ativação do SMS Programável - Twillio

gerado para o Sandbox.

Também foi implementado o diálogo através da versão *web* do Moodle, onde o processo decorre de forma similar ao WhatsApp. O usuário acessa uma interface contendo as funcionalidades do sistema Moodle, e interage com o MoodlebBot. O acesso a essa ferramenta se dá por meio de uma caixa de diálogo contida na barra inferior da página do Moodle.

Figura 15 – Comunicação através do Sistema Web Moodle - Interface Moodle



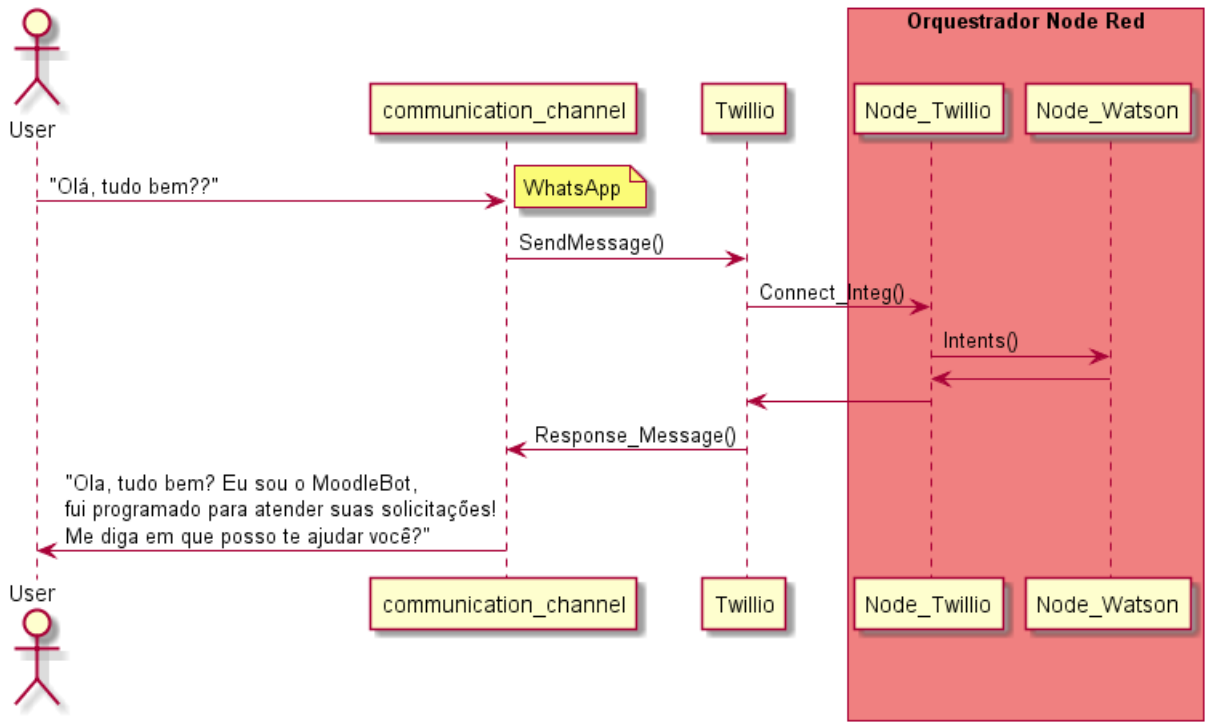
Na Figura 15, podemos ver a interface do sistema Moodle exibida após a realização do *login* e redirecionamento para o sua página inicial. Nesta página o usuário pode verificar

seus cursos, suas atividades pendentes, entre outras funcionalidades. Do lado direito da figura, podemos verificar também o final da interação entre o Moodlebot e o usuário, com o Moodlebot informando as disciplinas do usuário requisitante. Este diálogo é semelhante ao diálogo através do WhatsApp.

4.3 Back-end do Moodlebot:

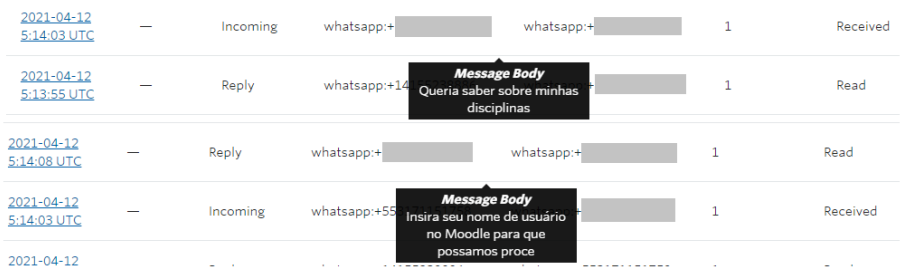
O *back-End* do MoodleBot é formado por três ferramentas principais, o IBM Watson, o Twillio, e o Node Red. Em relação ao IBM Watson, o mesmo tem por finalidade o processamento das mensagens do usuário durante o diálogo, que é realizado por meio de inteligência artificial. O Twillio tem seu papel nas requisições originadas do WhatsApp, funcionando como um integrador entre o WhatsApp e o Moodlebot. Esta integração ocorre com a configuração do número de telefone associado ao WhatsApp no Twillio, neste momento o Twillio passa a ser o representante da interface de conversação o IBM Watson. O Twillio também armazena as mensagens trafegadas, gerando relatórios e informações de monitoramento das comunicações. O Node Red, por sua vez, organiza a comunicação entre as mensagens interceptadas pelo Twillio na comunicação entre o WhatsApp e o MoodleBot. Através de uma programação visual baseada em nós, o Node Red define o início da comunicação em um nó requisição relacionado à mensagem que se encontra no Twillio e, a partir deste momento, direciona aos nós de captura dos dados, até que a informação chegue no nó representante do IBM Watson.

Figura 16 – Estrutura do Back End através do WhatsApp - WhatsApp



Toda a comunicação discutida até aqui pode ser visualizada pela Figura 16. Durante o envio de uma mensagem do usuário pelo WhatsApp, esta mensagem segue para o Twillio até chegar ao Node Red, onde são configurados nós em relação ao Twillio e o IBM Watson. Após o fluxo chegar em seu último nó, a resposta é atribuída à requisição e retornada ao usuário. Neste diálogo simulamos a intenção do usuário em iniciar um diálogo com o `ChatBot`. A configuração da intenção envolve definir as possíveis mensagens do usuário de acordo com cada tema. A definição destas mensagens é realizada pelo desenvolvedor que terá de transformar em texto o intuito do usuário, captando assim as possíveis formas de saudação ao ChatBot como, “Ola, tudo bem?”, “Ola, boa noite!”, “Oi, tudo bem?”, “Opa!”, “E ai”, “Bom dia!”, entre outras possibilidades.

Figura 17 – Back-End do serviço Twillio - Twillio



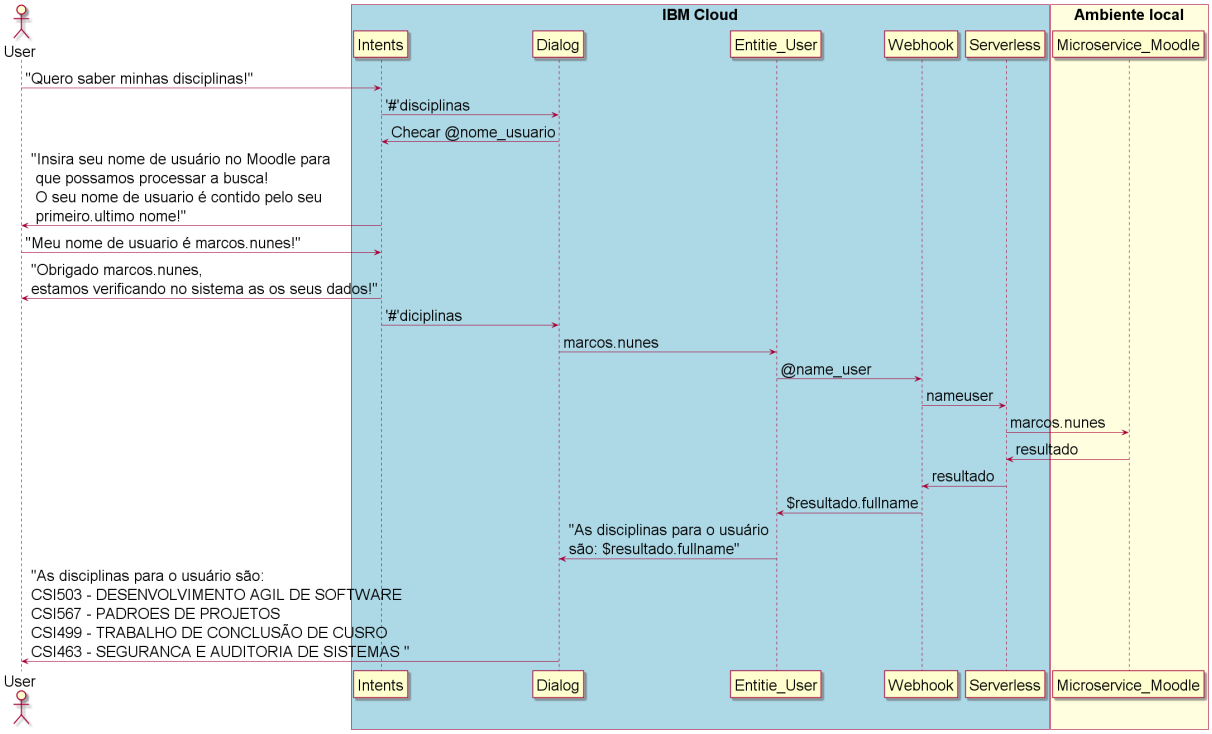
Como podemos visualizar na Figura 17, uma mensagem enviada pelo WhatsApp fica armazenada, podendo ser analisado posteriormente seu *log* de comunicação no monitor do Twillio. Em geral, os *logs* possuem custo computacional, o tempo de processamento da mensagem e identificação de recebimento perante o Twillio. A ferramenta também armazena o período em que as mensagens foram enviadas, períodos em que houveram mais envios ou recebimentos, número de mensagens com problemas no envio, *insights* de períodos com maior número de erros de comunicação, entre outros dados relevantes.

O *back-end* da aplicação também é realizado pelo IBM Watson Assistant. Neste serviço o diálogo é processado através de fluxos de conversação, estes fluxos são definidos de acordo com as intenções cadastradas no sistema. Cada intenção está relacionada a uma necessidade do usuário e armazena um ou mais modelos de mensagens esperadas durante um diálogo. Para este trabalho, foram criadas as intenções de visualizar disciplinas, saudações, realizar login, obter notas, entre outras.

Toda intenção deve iniciar com um simbolo cerquilha (#) seguido de um nome identificador da intenção. É importante definir um conjunto de ao menos 20 intenções, para que o `Chatbot` possa realizar um treinamento eficaz. Após a criação da intenção, a mesma deve ser associada a um nó de diálogo. Um diálogo relaciona as intenções com as possíveis respostas do `Chatbot`. As respostas são definidas de duas maneiras: de forma padrão, ou de forma customizada através de entidades. Nas respostas padrões somente é necessário a inserção das possibilidades de respostas à uma intenção, retornando uma resposta pré-programada. Para a geração destas respostas é necessário associar a intenção ao nó de diálogo e posteriormente inserir os valores de resposta.

Já para respostas customizadas, é necessário a utilização do que chamamos de entidades. Entidade é uma palavra ou frase esperada em uma conversa e está diretamente relacionada a um substantivo esperado na conversação que representa uma informação pertinente para o sistema. Utilizar uma entidade é importante para situações onde o chatbot necessita de armazenar uma informação da resposta do usuário, por exemplo o nome de um aluno, uma disciplina, entre outras possibilidades.

Figura 18 – Fluxo do Watson para respostas customizadas - IBM Watson Assistant



No exemplo da Figura 18, temos uma conversação onde o usuário requisita suas disciplinas através da mensagem “Quero saber minhas disciplinas”. Ao processar esta frase na intenção disciplinas, o chatbot identifica o início de um diálogo. Para responder, o chatbot analisa o texto inserido e identifica a intenção que se encontra na mensagem, neste caso a intenção de disciplinas. Pela definição do nó de dialogo referente à intenção

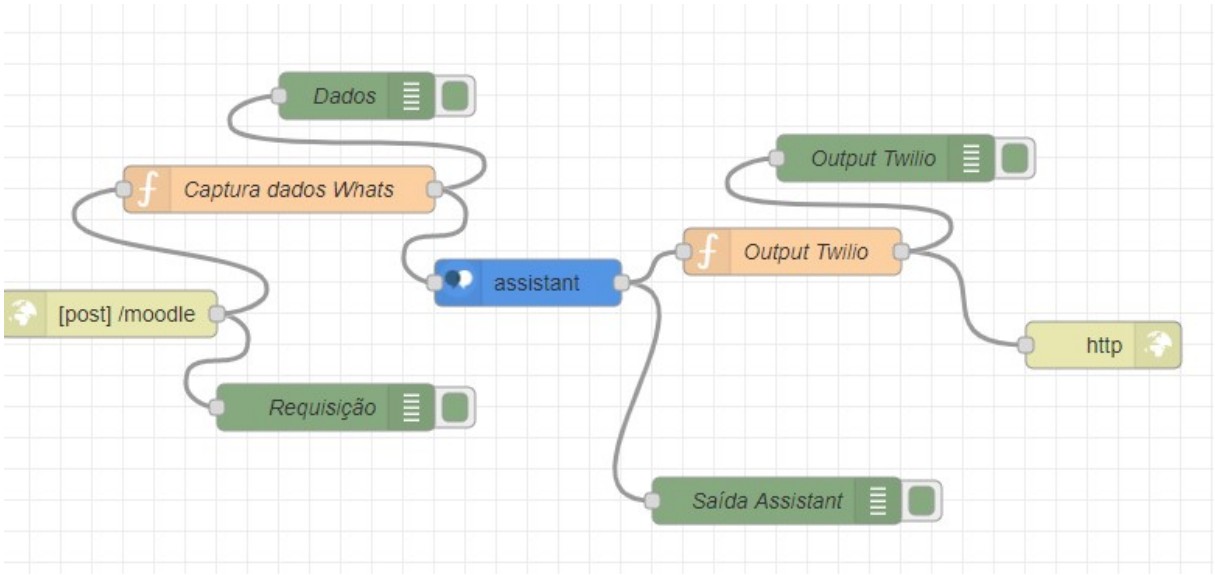
disciplinas, o chatbot detecta a necessita de buscar as disciplinas deste usuário no banco de dado. Para que a requisição ao banco aconteça, é preciso que o usuário insira sua chave de autenticação de usuário, a busca ao banco acontece através desta chave. Portanto, o chatbot requisita ao aluno para que insira o seu nome de usuário, este nome representará a chave de autenticação.

Para a filtragem dos dados de um usuário durante uma busca ao banco de dados, é importante que uma chave consistente e segura seja definida, diminuindo os riscos de redundância de dados. Para o caso deste trabalho, definimos uma chave formada pelo seu nome de usuário para acesso ao sistema. O Aluno insere o nome de usuário através da frase, "Meu nome de usuário é marcos.nunes!" ou uma frase qualquer. A partir deste momento o Chatbot responde amigavelmente que os dados estão sendo processados e segue para a atividade de recuperação das disciplinas deste aluno. Como esta intenção possui a entidade "@name-user", o gatilho para esta intenção é acionado no Webhook. O Webhook requisita ao *Serverless* os dados referentes a esta entidade. Para o trabalho, foi utilizado o serviço Cloud Functions para o desenvolvimento das funções *Serverless*, este serviço é disponibilizado na plataforma IBM Cloud. Desta forma, quando as funções *Serverless* forem acionadas, as mesmas redirecionam uma requisição ao microsserviço do Moodle, que por sua vez acessa o banco de dados enviando uma solicitação. O microsserviço do Moodle é um sistema desenvolvido para realizar buscas no banco de dados do Moodle e disponibilizar seus pontos de entrada para que o ambiente de nuvem acesse o banco de dados local. Após a requisição chegar ao microsserviço, os parâmetros enviados pelo Webhook ao *Serverless* são processados e alocados na *query* de consulta ao banco de dados. Desta maneira, pode se obter informações do banco por meio de parâmetros, que podem ser utilizados como filtros na *query* executada pelo microsserviço. Por fim, ao obter os dados do banco, o microsserviço retorna os dados para a nuvem e as informações trafegam até que cheguem no IBM Watson e sejam enviadas ao usuário solicitante. Este microsserviço será apresentado na seção 4.8. Por fim, o Webhook aciona novamente a entidade e retorna a listagem de suas disciplinas para o IBM Watson.

4.4 Configuração do Fluxo de dados - Node Red

Para orquestrar o fluxo de comunicação do Twillio e o IBM Watson Assistant é utilizado o Node Red. Cada nó definido no Node Red possui uma configuração destino a um ambiente ou API, especificando uma funcionalidade para o sistema, seja recebimento, envio, ou processamento de informações. Os dados originados do WhatsApp e armazenados no Twillio, são configurados para o nó `"post"/moodle`. Este nó representa um ponto de entrada ao fluxo do Node Red. A mensgem prossegue ao nó `HTTP`, ou o nó de retorno dos dados ao Twillio, que posteriormente redireciona a resposta ao WhatsApp, determinando o fim desta interação.

Figura 19 – Fluxo de nós e orquestração das API's - Node Red



Algoritmo 1 – Função de captura dos dados do WhatsApp - Node Red

```
msg.user = msg.payload.From
msg.payload = msg.payload.Body
return msg;
```

Na Figura 19 podemos visualizar a disposição da comunicação à partir de uma requisição do WhatsApp. O nó `"post"/moodle` recebe os dados provenientes do Twillio e valida o recebimento através do nó requisição, responsável pela depuração da mensagem. Logo após, o nó definido como captura dos dados do WhatsApp é acionado.

Podemos visualizar no Algoritmo 1 a composição do nó captura dos dados whats, contendo uma variável denominada de `msg.user`, responsável por receber os dados de cabeçalho e identificar o solicitante da requisição que se encontram em `msg.payload.From`. A segunda etapa da função é associar a variável `msg.payload`, os dados do corpo da requisição que pertencem a `msg.payload.body`. Por fim, o objeto que recebe os dados da requisição é retornado em `msg`. A função de captura dos dados possui um nó de depuração chamado de dados, que é responsável por validar o recebimento das informações da requisição pelo objeto `msg`.

Voltando à Figura 19, podemos visualizar o nó `Assistant`, que tem por finalidade receber os dados da função de captura e, posteriormente, redirecionar estes dados ao serviço do IBM Watson Assistant que, com o seu processo de análise textual da mensagem, irá identificar a resposta através de suas ferramentas de NLP. Este nó necessita do recebimento das credenciais do serviço IBM Watson implementado para o Moodlebot. As informações para configuração do nó `Assistant` são: chave da API criada, ID do ambiente de trabalho

Algoritmo 2 – Função de retorno ao Twillio - Node Red

```
var iterator = msg.payload.output.text.length;
var resMessage = ''
for ( var i = 0; i < iterator; i ++ ){
    if(msg.payload.output.text[i]){
        resMessage = resMessage + msg.payload.output.text[i]
    }
}
msg.payload =
    "<Response>" +
        "<Message><Body>" +
       resMessage +
        "</Body></Message>" +
    "</Response>"
return msg;
```

onde o serviço é instanciado e ponto de entrada do serviço. O `Assistant` possui o nó, Saída Assistant para seu processo de validar os dados recebidos da função de captura, validar as informações da API do IBM Watson, enviar dados da mensagem para a API processar, além de retornar os dados da resposta à requisição. Para a resposta retornar ao usuário é necessário a configuração de um nó `Output Twillio`. Este nó formata os dados obtidos no processamento realizado pelo IBM Watson Assistant e os retorna como uma mensagem de texto, alocando os dados em um corpo de resposta na variável `msg.payload.output.text`.

Podemos visualizar no Algoritmo 2 o tratamento das respostas recebidas do nó `Assistant`. Nesta função, temos uma variável chamada `iterator` que recebe o tamanho do array de respostas do `Assistant` e estão contidos em `msg.payload.output.text.length`, como os dados de resposta do `Assistant` estão contidos em um array que armazena as frases de resposta identificadas pelo `Assistant`, podemos ter mais de uma frase contida na resposta e devemos tratar no retorno das respostas ao Twillio. Posteriormente temos a criação da variável `resMessage`, que será encarregada de receber todas as frases de respostas concatenadas. À medida que esta verificação acontece, a variável `resMessage` é incrementada com as frases existentes. Ao fim da interação, criamos uma variável chamada `msg.payload`, que recebe o corpo da resposta ao Twillio, juntamente com `resMessage`, contendo todas as frases obtidas na interação sobre o array. Por fim, temos o retorno da função através de `return msg`. Por fim é acionado um nó `http` que retorna os dados contidos nesta variável para o aplicativo Twillio e os redireciona ao WhatsApp do usuário.

Algoritmo 3 – Configuração do Dockerfile

```
FROM nginx:alpine
RUN rm /etc/nginx/conf.d/*
EXPOSE 80
EXPOSE 3306
COPY proxy.conf /etc/nginx/conf.d/
```

Algoritmo 4 – Configuração da virtualização do MariaDB - Docker

```
mariadb:
    image: 'docker.io/bitnami/mariadb:10.5-debian-10'
    ports:
      - '3306:3306'
    environment:
      - ALLOW_EMPTY_PASSWORD=yes
      - MARIADB_USER=bn_moodle
      - MARIADB_DATABASE=bitnami_moodle
      - MARIADB_PORT=3306
      - MARIADB_PASSWORD=
    volumes:
      - mariadb_data:/bitnami/mariadb
```

4.5 Virtualização do Moodle e MariaDB - Docker:

Para a geração dos serviços do Moodle e MariaDB foi necessária a utilização da ferramenta Docker. Para a exportação e acesso das portas dos serviços Moodle e MariaDB, foi necessária a utilização do NGINX[1]. O serviço NGINX é um servidor web baseado em HTTP, utilizado para balanceamento de carga, *proxy* reverso, entre outras funcionalidades.

Podemos visualizar no Algoritmo 3 a configuração do serviço NGINX. Através do comando `FROM nginx:alpine`, especificamos a etiqueta referente a imagem oficial do serviço no Dockerhub, repositório oficial de imagens para o Docker. Logo após, o comando `RUN rm/etc/nginx/conf.d/*` executa as configurações no contêiner gerado. O comando `EXPOSE` exportas os serviços Moodle e MariaDB para as portas 80 e 3306, respectivamente. Por fim, temos o comando `COPY proxy.conf/etc/nginx/conf.d/` que copia o arquivo `proxy.conf` que se encontra no ambiente local, para o destino `/etc/nginx/conf.d/` do servidor.

Podemos visualizar no Algoritmo 4 as configurações para a virtualização do banco de dados MariaDB. Primeiramente temos a definição do comando para buscar a imagem do serviço no Dockerhub através da etiqueta `docker.io/bitnami/mariadb:10.5debian-10`, posteriormente temos a definição da porta 3306 para a execução do serviço. Logo após,

[1] http://nginx.org

Algoritmo 5 – Configuração da virtualização do Moodle - Docker

```
moodle:
    image: 'docker.io/bitnami/moodle:3-debian-10'
    ports:
      - '80:8080'
      - '443:8443'
    environment:
      - MOODLE_DATABASE_HOST=mariadb
      - MOODLE_DATABASE_PORT_NUMBER=3306
      - MOODLE_DATABASE_USER=bn_moodle
      - MOODLE_DATABASE_NAME=bitnami_moodle
      - ALLOW_EMPTY_PASSWORD=yes
      - MOODLE_USERNAME=root
      - MOODLE_PASSWORD=123456
      - MOODLE_EMAIL=marcos.nunes@aluno.ufop.edu.br
    volumes:
      - moodle_data:/bitnami/moodle
      - moodledata_data:/bitnami/moodledata
    depends_on:
      - mariadb
```

definimos as variáveis de ambiente do serviço, como senha, nomes do banco de dados e do usuário padrão para acesso ao mesmo. Por último, temos a definição do endereço onde os dados serão armazenados no servidor através do volume `mariadb_data:/bitnami/mariadb`.

As configurações do serviço Moodle acontecem de forma semelhante ao MariaDB. Vizualizamos no algoritmo 5 as definições da imagem oficial para o Moodle às portas para a execução do serviço, neste caso a porta 8080. Para as variáveis de ambiente temos o endereço usado pelo servidor de banco de dados, por padrão definido por mariadb, a porta 3306 usada para o servidor de banco de dados ao qual o Moodle vai se conectar ao nome do usuário padrão, `bn_moodle`. Configuramos também as credenciais de acesso, usuário, e-mail e senha para um usuário padrão ao o sistema Moodle. Para o Moodle, a persistência dos dados do volume no servidor são definidos por dois endereços, `moodle_data:/bitnami/moodle` onde os códigos fonte serão armazenados e o endereço `moodledata_data:/bitnami/moodledata`, usadda para armazenar dados da interação entre a aplicação Moodle e o banco de dados MariaDB. Por fim temos o comando de dependência na execução dos servidores, `depends_on`. Para este caso o servidor Moodle será virtualizado somente caso o houver um status positivo em relação à virtualização do servidor do banco de dados MariaDB.

Algoritmo 6 – Consulta para retornar disciplinas de um aluno - MariaDB

```
SELECT *
    FROM mdl_user u
    INNER JOIN mdl_user_enrolments ue ON ue.userid = u.id
    INNER JOIN mdl_enrol e ON e.id = ue.enrolid
    INNER JOIN mdl_course c ON e.courseid = c.id
    WHERE u.username = 'marcos.nunes';
```

4.6 Estrutura das consultas - MariaDB:

Para a utilização de uma interface para o MariaDB, foi utilizado o MySQL Workbench [2], uma ferramenta de gerenciamento de banco de dados e compatível com consultas SQL. Utilizamos esta ferramenta para a administração do bancos de dados gerado para o Moodle em ambiente local. Neste ambiente, criamos a conexão de nome `teste` para acesso ao banco de dados virtualizado do MariaDB. Como credenciais de acesso, temos o usuário `root` para usuário padrão da aplicação, contendo a porta 3306 como porta disponível no servidor `localhost`, ou de endereço 127.0.0.1.

O banco de dados `bitname-moodle` contém as informações em relação à aplicação Moodle, como registros de usuários, departamentos, cursos, disciplinas, notas, tarefas, notificações, calendários, entre outros. Estas informações são importantes para o funcionamento e gerenciamento de um sistema acadêmico. Alguns exemplos de consultas possíveis são a seleção de disciplinas de um aluno e suas notas finais.

As tabelas do MariaDB e seus registros, seguem em sua composição nomenclaturas padronizadas pelo `moodle.org`. No algoritmo 6 é possível analisar as tabelas envolvidas em uma consulta pelas disciplinas do aluno `marcos.nunes`. Começamos pela tabela `mdl_user`, que é responsável por armazenar todos os usuários criados para a aplicação Moodle, desta tabela iremos obter o `id` do usuário, podendo então usá-lo para associações com outras tabelas. A tabela `mdl_enrol` também é utilizada na consulta, nesta tabela encontramos os registros de todas os modelos de matrículas realizadas na plataforma moodle. Desta forma podemos encontrar se o usuário possui uma matrícula vigente na tabela `mdl_user_enrolments`, onde persistem os registros de todas as matrículas de um aluno ou tutor. Por fim temos a criação de um relacionamento destas tabelas citadas anteriormente, com a tabela `mdl_course`, que tem por finalidade armazenar os dados das disciplinas ou cursos cadastrados no banco.

Um outro exemplo pode ser visualizado no algoritmo 7, onde acontece a consulta realizada para retornar as notas finais do aluno "marcos.nunes". Assim como para retorno das disciplinas a consulta utiliza a tabela `mdl_user`, responsável por conter os registros de todos os usuários do banco de dados moodle. Com o `id` do usuário `marcos.nunes`,

[2] https://www.mysql.com/products/workbench/

Algoritmo 7 – Consulta para retornar as notas finais de um aluno - MariaDB

```
SELECT
    u.id AS userid,
    u.username AS studentid,
    gi.grademax AS itemgrademax,
    g.finalgrade AS finalgrade
    FROM mdl_user u
    JOIN mdl_grade_grades g ON g.userid = u.id
    JOIN mdl_grade_items gi ON g.itemid =  gi.id
    JOIN mdl_course c ON c.id = gi.courseid
    WHERE u.username = 'marcos.nunes';
```

filtramos na tabela `mdl_grade_items`, todas as notas já lançadas para este usuário, esta tabela também nos da acesso aos títulos das atividades avaliadas itens avaliados. Outra tabela utilizada nesta consulta é a `mdl_grade_grades`, responsável por persistir todas as notas notas finais dos alunos. Por fim, temos a tabela `mdl_course` que irá nos proporcionar as disciplinas cadastradas no banco de dados. Possibilitando que retornemos os nomes das disciplinas e suas respectivas notas finais.

4.7 Interface do chatbot no Moodle

O Moodle é um sistema que utiliza em grande parte de sua implementação as linguagens PHP e JavaScript. Para acessar o código fonte do Moodle é necessário realizar acesso remoto ao contêiner referente à aplicação. Ao acessar o código fonte, qualquer alteração pode ser compilada, resultando em uma mudança de estado da aplicação Moodle.

Para a disponibilização do chatbot através da versão *web* do Moodle, incluímos o *link* do serviço IBM Watson na configuração das páginas do Moodle, definimos que antes do fechamento do corpo das páginas o *script* será executado. Desta forma, em todas as interações entre o usuário e páginas do sistema, o símbolo do chatbot aparecerá.

Podemos visualizar no algoritmo 8, o *script* inserido no sistema Moodle. Este trecho de código contém o comando de acionamento da ferramenta do chatbot nas páginas do Moodle, e possui como credenciais o ID de integração do sistema, a região onde o serviço está localizado, e o ID da instância do serviço no IBM Cloud. Por último temos a definição do momento do evento de acionamento. Este evento especifica que ao carregar o html da página, o *script* do chatbot será chamado antes da geração do documento HTML.

Figura 20 – Dashboard do Curso - Moodle

Algoritmo 8 – Script para a disponibilização do chatbot na interface do sistema Moodle

```
<script>
window.watsonAssistantChatOptions = {
    integrationID: "0687f9a7-9b29-4ee1-a29c-4e30f8474523",
    region: "us-south",
    serviceInstanceID: "c3b04df6-a4fb-4aae-965f-7102
 e58b58aa",
    onLoad: function(instance) { instance.render(); }
  };
setTimeout(function(){
  const t=document.createElement('script');
  t.src="https://web-chat.global.assistant.watson.appdomain
        .cloud/loadWatsonAssistantChat.js";
  document.head.appendChild(t);
});
</script>
```

4.8 Microsserviço de acesso ao banco de dados Moodle

O microsserviço de acesso ao Moodle funciona como um intermediário entre o ambiente de nuvem e o ambiente local, tornando a arquitetura híbrida. O acesso às ferramentas deve ser autenticado, seguro e deve especificar os serviços expostos por cada ambiente.

Este microsserviço trouxe uma camada de processamento das solicitações ao banco de dados Moodle, diminuindo a necessidade do desenvolvimento das funções de buscas ao banco no próprio ambiente de nuvem, o que acarretaria em maior custo em relação às funções sem servidor. A definição da API do Moodle como a ferramenta que será exposta

Algoritmo 9 – Definições da aplicação e validações- Microsserviço

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');

const rotaUsers = require('./routes/users');
const rotaDisciplinas = require('./routes/disciplinas');

app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.use('/disciplinas', rotaDisciplinas);
app.use(function (req, res, next) {
    const error = new Error('Não encontrado');
    res.status(404).send(error)
    next();
});
app.use((error, req, res, next) =>{
    const error=new Error('Falha de conexão com o servidor');
    res.status(500).send(error)
    next()
});
module.exports = app;
```

ao ambiente da nuvem e não o Banco de Dados MariaDB, se deve ao fato de que o acesso direto da nuvem ao banco de dados necessitaria de uma quantidade maior de codificação no ambiente de nuvem, trazendo maior consumo de dados das funções sem servidor e maior processamento neste ambiente. Desta forma as funções em nuvem realizarão requisições ao ambiente local onde se encontra o microsserviço e terão uma comunicação direta através de um túnel seguro, distribuindo processamento entre ambiente local e ambiente de nuvem. Todas as buscas e validações dos dados obtidos do banco, serão feitas pelo microsserviço, restando as funções em nuvem apenas o consumo do microsserviço através do seu ponto de entrada. A API desenvolvida baseou se na linguagem node JS, utilizando como gerenciador de dependências o NPM, os frameworks express.js e Next para requisições HTTP.

Detalhes da implementação podem ser vistos no Algoritmo 9, onde temos a configuração do domínio das rotas e as validações de exceção. Este código é referente ao arquivo `app.js`, e é composto pela importação do pacote `express` que posteriormente é atribuída a variável `app`, representando a nossa aplicação. Ainda no processo de definição dos pacotes necessários para o microsserviço, temos a importação do pacote `body-parser`, este pacote é responsável por definir que as respostas do microsserviço serão retornadas no tipo JavaScript Object Notation (JSON), este formato possui estrutura em forma de objeto contendo chave e valor, e é utilizado para troca de informações entre sistemas. Após a definição dos pacotes, temos a importação dos arquivos de rotas, que serão apresentados no

Algoritmo 10 – Criação do serviço local e execução da aplicação - Microsserviço

```
const http = require('http');
const app = require('./app');
const port =  3000;
const server = http.createServer(app);
server.listen(port, ()=>{
    console.log("Serviço rodando na porta:", port )
});
```

decorrer do texto. A medida que importamos `body-parser`, temos de inicializá-lo através da função `use`, e definir o tipo de dados `json`.

A primeira rota implementada é a de disciplinas, e se encontra no arquivo `disciplinas.js`. A funcionalidade desta rota é obter as disciplinas de um determinado aluno. Para que a aplicação possa executar as chamadas da rota de disciplinas, utilizamos a função `use` que se encontra dentro do pacote `express`, e disponivel em nossa aplicação. Ao inicializar as rotas, podemos disponibilizar as chamadas `REST` da `API` no domínio `/disciplinas`, desta forma todos os métodos implementados para esta rota poderão receber requisições neste domínio, esta lógica de arquitetura de código foi utilizada também para a busca de notas finais. Após a implementação das buscas, temos as validações de `erro 404` e `erro 500` para rotas invalidas e falha de conexão com o servidor, respectivamente. As duas funções possuem a mesma estrutura, recebendo como parâmetros as variáveis `req`, `res`, `next`, representando, requisição, resposta e próxima execução respectivamente. A verificação do erro acontece através do status contido na resposta à requisição, e retornado a partir da função `send`, também contida na resposta. Após o retorno do erro, a função `next` aciona o próximo bloco de execução. Por fim, temos a exportação da variável `app`, contendo as especificações definidos para a aplicação.

Para que a aplicação esteja disponível, precisamos configurar o servidor de execução da aplicação. No Algoritmo 10, podemos verificar as configurações do arquivo `server.js`, que é encarregado de criar o servidor para a aplicação. Este programa contém uma variável `http`, que importa o pacote http para a realizações das requisições. Para a importação da aplicação, criamos uma variável `app` que importa o arquivo `app.js`, definido anteriormente. Logo após, definimos uma variável `port` para receber a porta de execução do microsserviço e para usarmos na criação do servidor. A próxima etapa é a definição da variável `server`, recebe a instanciação do método `createServer` contido em `http`, criando o servidor e o tornando compatível para requisições `http`. Por fim, temos a execução do servidor na porta 3000 e um *log* de status do servidor. A partir deste momento, o microsserviço se encontrará disponível quando a aplicação for executada, e pode ser acessado através do endereço `http://localhost:3000/`.

Algoritmo 11 – Configuração de acesso ao Banco de Dados - Microsserviço

```
const mysql = require ('mysql');

var pool = mysql.createPool({
    "user" : "bn_moodle",
    "password": "",
    "database": "bitnami_moodle",
    "host": "localhost",
    "port" : 3306
});
```

4.8.1 Conexão com o banco de dados MariaDB - Microsserviço

A conexão com o banco de dados MariaDB, acontece através da importação do pacote MySQL. Podemos observar no Algoritmo 11 as configurações do arquivo `mysql.js`. Neste trecho de código, temos a definição das credenciais utilizadas na configuração do banco de dados MariaDB. Após importação da dependência do `MySQL` para a varável `Mysql` é necessário apenas a definição das variáveis de ambiente para a conexão com o banco de dados local. Os parâmetros são, usuário, senha, nome do banco de dados, `host` ou endereço do banco e porta, onde o mesmo está sendo executado.

No Algoritmo 12, temos a definição do arquivo `disciplinas.js`, este arquivo contém a lógica de desenvolvimento das buscas das disciplinas no banco de dados MariaDB. Podemos verificar a estrutura de chamada de uma rota `GET`, requisitando as disciplinas de um aluno. A variável `express` importa as propriedades do framework. Posteriormente temos a variável `router`, que importa a função `Router` contida em `express`, esta função é utilizada para a manipulação de rotas em aplicativos NodeJS. Por fim temos a variável `mysql`, que importa as configurações de conexão definidas no arquivo `mysql.js`.

Para o endereço `/disciplinas/:name_user` através do método `GET`, temos a implementação de uma função que recebe como parâmetros uma requisição, uma resposta e um comando para o próximo bloco de execução, como mostrado anteriormente. Além destes dados a função utiliza também um parâmetro `/disciplinas/:name_user`, referente a chave inserida na URL, através desta chave que a busca no banco acontecerá. Esta chave está sendo direcionada para o microsserviço na variável `req` e contida em `req.params.name_user`.

Dentro desta função `GET`, há uma conexão MySql responsável pela realização da busca das disciplinas para a chave informada. Esta consulta SQL foi apresentada na Figura 6. Caso ocorra erro de conexão com o bando de dados, a função de retorno responderá com um erro 500. Em caso de sucesso, a função retornará os dados obtidos e atribuindo status 200 para a requisição.

Algoritmo 12 – Função de Busca de Disciplina - Microsserviço

```
const express = require ('express');
const router = express.Router();
const mysql = require('../mysql').pool;

router.get('/:name_user', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){
      return res.status(500).send({ error: error});
    }
      conn.query(
        `SELECT c.fullname
        FROM mdl_user u
        INNER JOIN mdl_user_enrolments ue ON ue.userid = u.id
        INNER JOIN mdl_enrol e ON e.id = ue.enrolid
        INNER JOIN mdl_course c ON e.courseid = c.id
        WHERE c.fullname != 'New Site' and u.username = ?;`,
        [req.params.name_user],
        (error, resultado, fields) => {
        if(error){
          return res.status(500).send({ error: error});
        } res.status(200).send({Response: resultado})
        }
      )
  });
});
```

Os testes em relação ao microsserviço, foram realizados através da ferramenta Postman. Estes testes foram importantes para a validação dos retornos das buscas ao banco de dados do Moodle.

4.9 Acesso da nuvem ao ambiente local - Secure Gateway

Existem algumas formas de conectar o ambiente local à nuvem, como VPN, *Direct Link* e *Secure Gateway*. Normalmente, aconselha-se o uso de VPN quando o recurso a ser acessado está no ambiente de nuvem, e a aplicação requisitante no ambiente local. Este cenário se difere do cenário deste trabalho, já que deslumbramos disponibilizar um acesso originado da nuvem e consumindo recursos do ambiente local. A utilização do *Direct Link* segue as mesmas premissas da VPN, acrescentando a possibilidade de vários usuários ou áreas de trabalho de um mesmo grupo acessando simultaneamente recursos da nuvem. Por estes fatores, neste trabalho de TCC, foi utilizado o *Secure Gateway*, pelo fato de ser um serviço destinado a uma conexão cliente e destino, onde os mesmos podem se alternar em requisitantes ou disponibilizadores de um recurso. Este serviço é gratuito pode

ser provisionado no próprio ambiente de nuvem da IBM. O *Secure Gateway* é utilizado através da configuração de um destino e um cliente, onde o destino é o ambiente onde estão alocados os serviços que serão compartilhados, e o cliente se configura como o ambiente que irá consumir recursos presentes no destino. Neste trabalho, o cliente foi definido como IBM Cloud e o destino foi definido como o servidor Moodle.

O serviço cliente pode ser obtido de três maneiras: através da virtualização do sistema com o Docker, utilizando o IBM DataPower, ou pela instalação do software IBM Installer. Neste este trabalho, optamos pela utilização do IBM Installer. O IBM Installer é um *software* para a identificação da nuvem na máquina hospedeira. A configuração deste software necessita do *token* e ID do *gateway* definido na criação do serviço *gateway*. Após a instalação do software é gerado um arquivo executável referente ao serviço, este arquivo é responsável pela inicialização do serviço.

Para a configuração do destino é necessário inserir informações referentes à máquina local, como o endereço IP do ambiente virtualizado pelo WSL2. As demais credenciais do serviço estão descritas nas opções de configuração. No campo segurança, seleciona-se a opção protocolo HTTP, que será o modelo de comunicação entre os dois ambientes. O campo TLS está relacionado à segurança da camada de transporte. Nesta etapa é importante definir o tempo para espiração do `Token` do `Gateway` criado.

O controle de acesso do cliente aos recursos do destino é realizado através de uma lista de IP's permitidos para conexão. Nesta lista, ficam definidos o IP público da máquina onde o recurso está alocado e a porta em que é exposto ao acesso do cliente. À medida que o IP público desta maquina se altera, é necessário que a lista seja atualizada com o novo endereço para que o cliente não perca seu acesso. Além da permissão de acesso, podemos também negar acesso a um IP específico. Esta lista se encontra acessível no serviço Secure Gateway Client instalado no ambiente local e disponível através do endereço `http://localhost:9003/acl`.

4.10 Funções sem servidor - IBM Cloud Functions

Funções sem servidor ou *serverless*, são trechos de código que especificam uma funcionalidade do servidor de nuvem geralmente são utilizadas para comunicações entre ambientes. Para conectar a nuvem ao ambiente local, utilizamos o Cloud Functions, serviço que pode ser provisionado no ambiente IBM Cloud. Podem ser utilizadas vários tipos de linguagens na ferramenta Cloud Functions, as mais conhecidas são Node JS, Python, Ruby, PHP, Java, Swift e Go. Para o projeto, utilizamos o Node js versão 12.

Após a criação das funções sem servidor, as mesmas estarão disponíveis para acesso no *dashboard* inicial do serviço, listando o tipo de ação, a memória alocada e o tempo limite da requisição. O acesso da nuvem para o ambiente local segue as premissas de

Algoritmo 13 – Função Serverless para Busca de Disciplinas - Cloud Functions

```
const rp = require('request-promise');
function main(params) {
return rp({
    method: 'GET',
    uri: 'http://cap-sg-prd-2.securegateway.appdomain.cloud
         :19282/disciplinas/${params.nameuser}',
    json: true
    })
    .then(body => {
      var resposta = {}
      var string = ''
      for(var i=0; i< body.Response.length; i++){
            string = body.Response[i].fullname + string
                     || undefined;
            resposta = {
                fullname : string
            }
        }
        return resposta;
    })
    .catch(err => {
        return err;
    });
}
```

comunicação HTTP, sendo necessária a definição das credenciais de acesso e URL. Podemos visualizar no algoritmo 13, o código referente à função sem servidor construída para acesso ao microsserviço executado no ambiente local.

A função de busca das disciplinas utiliza credenciais de acesso à API local e da importação de um *Middleware* para a requisição, o `Request-promise`. Este `Middleware` é uma requisição HTTP que utiliza declarações de funções assíncronas e respostas através de promessas. As promessas garantem que um trecho de código assíncrono, que não obteve uma resposta à requisição, não interfira no funcionamento da aplicação até que a mesma receba uma resposta. O desenvolvimento de funções assíncronas seguem os princípios *Async/Await*. Posteriormente, temos a URI que utiliza o endereço do ponto de entrada definido pelo `secure getaway` destino, juntamente com a rota de disciplinas e o nome do usuário. Como resposta da requisição, temos um objeto `Json` estruturado com as disciplinas concatenadas para serem disponibilizadas ao usuário.

5 Resultados

Nesta são demonstrados os resultados do trabalho, iniciaremos com o processo de prepararão do ambiente para que as ferramentas se comuniquem, possibilitando o funcionamento do Moodlebot. E então simularemos conversações através das diferentes formas de interação com o usuário.

5.1 Preparação do ambiente

O Moodlebot foi desenvolvido através de um notebook, utilizando um sistema operacional Windows 10 Home Single Language e uma máquina virtual linux Ubuntu 20.04.2 versão LTS, virtualizado pelo subsistema Windows para Linux. A máquina virtual linux foi utilizada para o desenvolvimento do microsserviço do Moodle e para a criação do ambiente virtualizado para o Moodle e MariaDB, utilizando o Docker na versão 19.03.13. Para gerenciamento e visualização de logs dos serviços virtualizados Moodle e MariaDB, foi utilizado o Docker Desktop versão 3.6.0. Em relação às ferramentas de nuvem, foi utilizada uma conta pré paga na plataforma IBM Cloud. Abaixo seguem as especificações da conta IBM Cloud e as configurações dos serviços provisionados durante o desenvolvimento do Moodlebot.

- Serviço Watson Assistant versão gratuita com 500 MB de transmissão/mês;
- IBM SDK para Node.JS com 524 GB disponíveis para execução do Node Red;
- Secure Gateway-z2 com 500 MB disponíveis ao mês, versão gratuita;
- Serviço Continuous Delivery versão gratuita, para até 500 tarefas por Delivery pipeline executadas ao mês;
- Serviço Cloud Functions com configurações de 500 ms de execução por ação, 128 MB de memória da ação, e até 5.000 execuções por mês; Grátis;

Para o funcionamento do Moodlebot, temos um conjunto de fases de execução realizadas manualmente antes da disponibilização do serviço, exigindo inicialização das ferramentas do IBM Cloud e serviços utilizados no ambiente local. Logo abaixo, segue a lista de etapas para a disponibilização do Moodlebot:

1. Virtualização dos serviços Moodle e MariaDB através do Docker Desktop;
2. Inicialização do sistema windows subsystem for linux (WSL2);

3. Acesso e execução do microsserviço dentro do ambiente Ubuntu;
4. Realização de login na plataforma IBM Cloud:
   4.1. Inicialização do serviço Node Red;
   4.2. Atualização do serviço Secure Gateway destino com o IP pertencente ao WSL2;
5. Execução do Secure Gateway Client no ambiente local, atualizando também sua lista de controle de acesso com o IP do WSL2;
6. Acesso ao Moodle *web* e ao Whatsapp, aplicativo ou web;
7. Ativação do Sandbox do Twillio, inicializando o ambiente de comunicação pelo WhatsApp;

5.2 Populando informações no MariaDB

Para a realização dos testes de funcionamento do sistema Moodlebot, foram criados 5 usuários, sendo 4 deles alunos e 1 administrador do sistema. Criamos também 2 cursos, 5 disciplinas, e 8 tarefas. O cadastro de informações no banco de dados MariaDB ocorreu através da utilização do Moodle *web*. Criamos alunos, disciplinas e cursos. Relacionamos estes alunos aos cursos e disciplinas. Em seguida, criamos atividades para as disciplinas existentes, definimos cronogramas de entrega, notas mínimas e máximas, modelos de arquivos aceitos, entre outras opções de configuração.

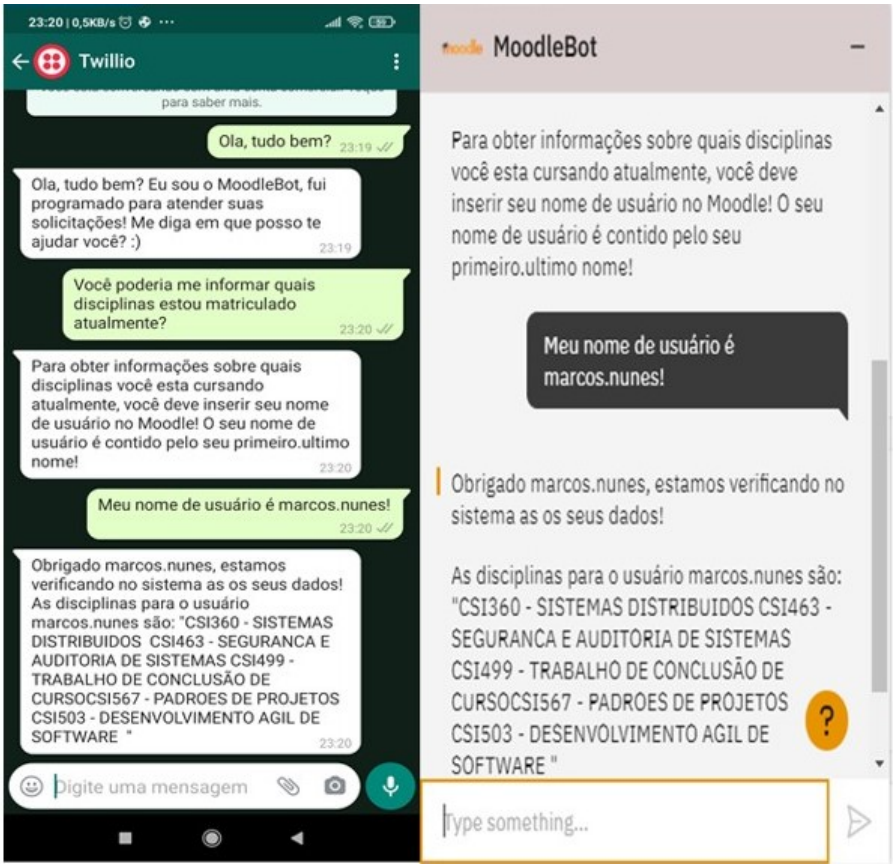
5.3 Diálogo - Interação Aluno e Moodlebot

No exemplo da figura 21, podemos visualizar uma solicitação realizada utilizando um aparelho celular com o WhatsApp instalado, e a mesma solicitação feita utilizando a plataforma Moodle *web*. Nesta solicitação, visualizamos a requisição de um aluno às suas disciplinas. Para melhor visualização, apresentaremos os demais experimentos através do aplicativo WhatsApp versão *web*.

Para exemplificar o funcionamento do Moodlebot, simulamos requisições entre um aluno e o chatbot. Ainda na Figura 21, podemos acompanhar o início da comunicação através de uma mensagem de saudação enviada pelo aluno através do WhatsApp: “Olá tudo bem?”. Esta intenção é programada para aceitar também mensagens como: “Oi, tudo bem?”,“Olá, bom dia!”, “Opa, boa noite!”, entre outras possibilidades de saudação. A partir do momento em que a mensagem é enviada pelo Whatsapp, o Moodlebot identifica a intenção de saudação e responde à mensagem do usuário.

O diálogo prossegue com o aluno requisitando suas disciplinas e o Moodlebot responde solicitando a inserção sua chave de identificação para a busca das disciplinas.

Figura 21 – Acessos por plataformas diferentes



Prosseguindo com a solicitação, o aluno insere sua credencial de autenticação para que o Moodlebot realize a busca: “Meu nome de usuário é marcos.nunes!”. O Moodlebot valida a solicitação e a responde apresentando as do usuário disciplinas.

Em novas requisições, valores de entidades que foram informados durante diálogos anteriores, são armazenados para próximas requisições. Este é o caso das buscas por notas finais, como podemos verificar na Figura 22. Nesta requisição o Moodlebot não solicita as informações de autenticação de usuário, pois obteve este parâmetro na busca anterior. Processando esta nova solicitação, o Moodlebot analisa o texto inserido e encontra a intenção de obtenção das notas finais, entendendo a necessidade de ir ao banco de dados para o retorno destes dados. Posteriormente, o Moodlebot responde demonstrando as notas encontradas para o usuário.

Durante uma requisição, o Moodlebot verifica perante as intenções, a necessidade de recebimento de um ou mais parâmetros para realizar a busca ao banco de dados. Caso um parâmetro necessário, ainda não tenha sido informado em buscas anteriores, este parâmetro é requisitado pelo chatbot. Podemos visualizar tal situação na Figura 23, onde temos a demonstração de uma busca realizada pelo aluno em relação às suas tarefas.

O aluno dá sequência na conversação e solicita visualização de suas tarefas: “E sobre as minhas tarefas, oque pode me informar?”, o Moodlebot responde informando a

Figura 22 – Solicitação de disciplinas e notas

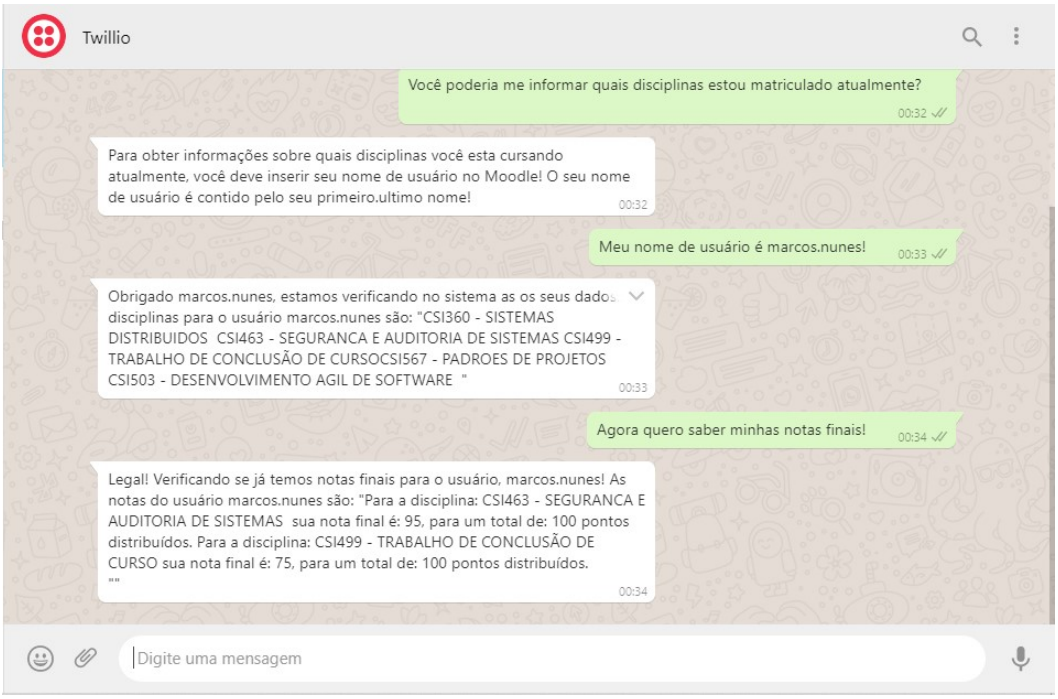
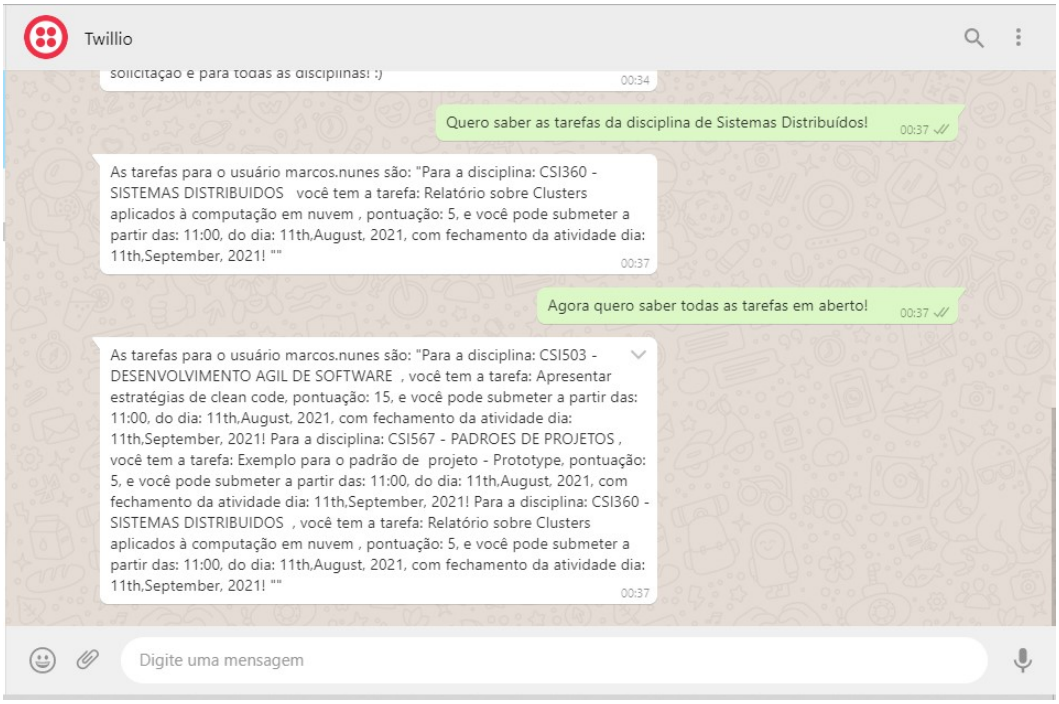


Figura 23 – Solicitação de tarefas



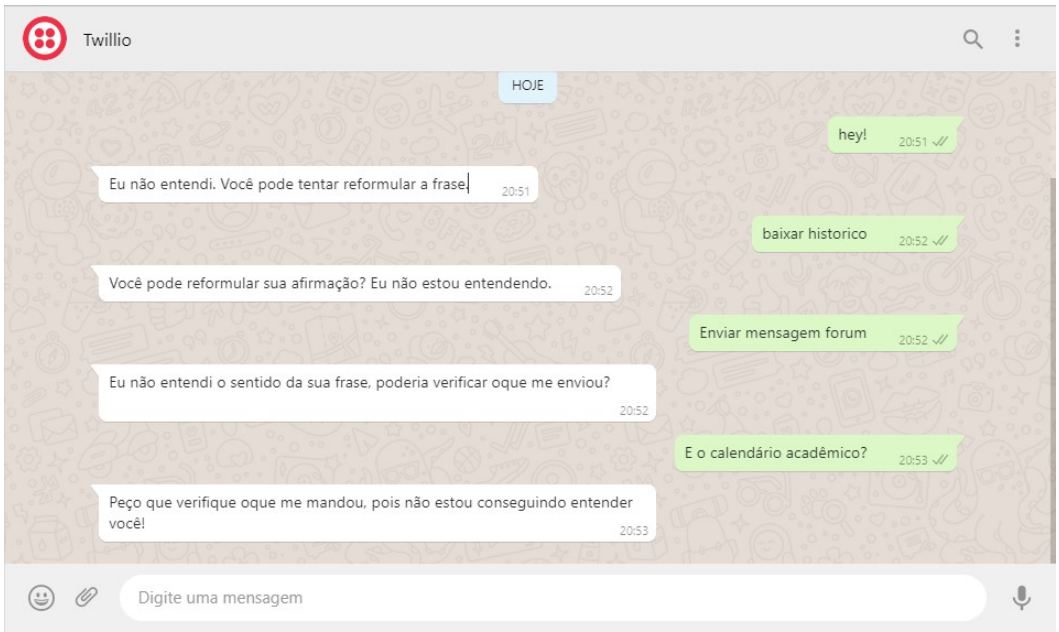
necessidade de saber se a busca é sobre todas as tarefas, ou tarefas por uma disciplina especifica. Caso a exigência do aluno seja para o retorno de todas as disciplinas, isso deve ser informado através da chave “todas”, esperada pelo Moodlebot durante o diálogo. Já na busca de tarefas por disciplina, se espera a inserção do código da disciplina ou do nome da disciplina. O aluno opta por informações das tarefas por disciplina, utilizando o nome da disciplina: “Quero saber as tarefas da disciplina Sistemas Distribuídos!”. Em

resposta a solicitação, o Moodlebot lista as tarefas em aberto referentes ao do usuário "marcos.nunes".

A resposta para a requisição é formada por dados importantes para um aluno, como título da tarefa, nome da disciplina em que a tarefa está associada, pontuação, horário, data de abertura e data de fechamento da tarefa. O fato de o aluno obter todas estas informações, possibilita que ele tenha uma visão maior em relação às suas responsabilidades para com a tarefa. Portanto, poderá se organizar para cumprir prazos e não perder nenhuma tarefa.

Por fim, podemos visualizar na figura 23, um diálogo para retorno de todas as tarefas do aluno. Nesta fase da conversação, o aluno requisita todas as suas tarefas: "Agora quero saber de todas tarefas!". Como esta solicitação necessita apenas da chave de autenticação do usuário, informada e armazenada nas requisições anteriores, o Moodlebot informa as tarefas para o usuário"marcos.nunes" imediatamente.

Figura 24 – Mensagens não compreendidas



Existem situações em que o Moodlebot não será capaz de entender uma mensagem enviada pelo usuário, estas situações estão atreladas a diálogos onde o chatbot não consegue identificar uma intenção para aquela mensagem. Para estas solicitações, o chatbot possui o nó de diálogo "entre outros casos". Este nó é responsável por enviar uma resposta padrão ao usuário em casos onde a mensagem processada não pode ser compreendida. As mensagens de retorno são acompanhadas de uma negativa do chatbot em relação à mensagem enviada pelo usuário. O IBM Watson Assistant armazena as mensagens não compreendidas, e possibilita que associemos estas mensagens a uma intenção existente, ou até mesmo criando uma nova intenção referente a esta mensagem.Ao associar estas mensagens a uma intenção, podemos treinar o algoritmo para que em futuras conversações sejam respondidas corretamente.

6 Conclusão

Este trabalho apresentou Moodlebot, uma arquitetura de chatbot para o Moodle. A arquitetura foi testada para casos de uso simulados. As ferramentas utilizadas no projeto, contribuíram para a aproximação de serviços de nuvem ao Moodle, permitindo requisições ao sistema utilizando o WhatsApp ou a versão web do sistema Moodle. Com a utilização da ferramenta Node Red, realizamos a orquestração das API's utilizadas, conectando o sistema Moodle ao WhatsApp. Utilizando funções sem servidor presentes no IBM Cloud, conseguimos acessar o microsserviço executado no ambiente local e buscando informações no banco de dados. A infraestrutura básica para que esta aplicação seja executada em produção se encontra desenvolvida e testada para casos de uso simulados. Na seção de trabalhos futuros, abordaremos possíveis melhorias idealizadas a fim de otimizar o atual sistema, tornando a plataforma mais robusta e com mais funcionalidades disponíveis ao usuário.

7 Trabalhos Futuros

As metas para este trabalho foram concluídas, mas determinados pontos podem ser aprimorados. Atualmente o sistema realiza requisições somente para comunicações textuais, não sendo compatível para mensagens de voz. Outros exemplos de melhorias da aplicação estão nas inclusões de novas possibilidades de busca, como por exemplo, envio de mensagens automáticas à fóruns de disciplinas, busca de notas por disciplinas, envio de arquivos, criar tarefas, entre outros.

Outra atualização seria a diferenciação dos tipos de usuários cadastrados, onde alunos acessam funcionalidades distintas de professores. As implementações de alunos foram demonstradas, diferentemente para os usuários professores. Em versões futuras, propomos desenvolver de funcionalidades relacionadas às atividades dos professores perante o Moodle.

Em relação à segurança, o sistema possui atualmente solicitações autenticadas pelo nome do usuário no sistema, desta forma, com um maior número de usuários aptos a se comunicarem com a aplicação, as chaves de autenticação necessitarão de uma combinação que inviabilizasse a duplicação das mesmas. Para esta situação seria necessário adaptação da política de geração de nomes de usuários perante o sistema, ou pela substituição da chave de autenticação por registros únicos para cada usuário, como CPF ou ID. Estas implementações significam menor risco de duplicidade de dados e maior segurança para os usuários do sistema.

Referências

BECHARA, J. J. B.; HAGUENAUER, C. J. Por uma aprendizagem adaptativa baseada na plataforma moodle/functional specifications on an adaptive learning environment based on the moodle platform. *Revista EducaOnline*, v. 4, n. 1, p. 1–10, 2010. Citado na página 14.

DENG, L.; LIU, Y. *Deep learning in natural language processing*. [S.l.]: Springer, 2018. Citado na página 18.

DUARTE, D. et al. Ambientes de aprendizaje: una aproximación conceptual. *Estudios pedagógicos (Valdivia)*, Universidad Austral de Chile. Facultad de Filosofía y Humanidades, n. 29, p. 97–113, 2003. Citado na página 22.

Eduardo GIGCH, J. P.; PIPINO, L. L. Ucb-bot: um exemplo de aplicação de computação cognitiva. *Future Computing Systems*, v. 1, n. 1, p. 71–97, 2017. Citado na página 14.

Figueiredo, Carlos, M.,S. NAKAMURA, E. ComputaÇÃo mÓvel: Novas oportunidades e novos desafios. TC Amazônia„ n. 2, p. 1–13, 2003. Citado na página 29.

GONZALEZ, M.; LIMA, V. L. S. Recuperação de informação e processamento da linguagem natural. In: *XXIII Congresso da Sociedade Brasileira de Computaçao*. [S.l.: s.n.], 2003. v. 3, p. 347–395. Citado na página 16.

HAN.N JONG PARK.H, J. C. I. B. J.; LEE, Y. K. Implementation of a neologism chatbot system using watson assistant. *Department of Computer Science and Engineering, Dongguk University-Seoul*, p. 1–4, 2018. Citado na página 14.

HUTCHINS, W. J. The georgetown-ibm experiment demonstrated in january 1954. In: SPRINGER. *Conference of the Association for Machine Translation in the Americas*. [S.l.], 2004. p. 102–114. Citado na página 18.

KOVÁCS, Z. L. *Redes neurais artificiais*. [S.l.]: Editora Livraria da Fisica, 2002. Citado na página 16.

MORENO, F. et al. Tical: Chatbot sobre o atlas linguístico do brasil no whatsapp. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2015. v. 26, n. 1, p. 279. Citado na página 30.

MOTTA, L. C. P. Chatbot para o moodle. *REVISTA ACADÊMICA ALCIDES MAYA*, v. 1, n. 2, p. 17–27, 2019. Citado na página 31.

OLIVEIRA, J. da S. et al. Ibm watson application as faq assistant about moodle. In: IEEE. *2019 IEEE Frontiers in Education Conference (FIE)*. [S.l.], 2019. p. 1–8. Citado na página 32.

RANAVARE, S. S.; KAMATH, R. Artificial intelligence based chatbot for placement activity at college using dialogflow. *Our Heritage*, v. 68, n. 30, p. 4806–4814, 2020. Citado na página 31.

ROCIO, V.; WESLEY, A. Construção de um chatbot para apoio ao estudante. *Revista de Ciências da Computação*, Universidade Aberta, p. 103–114, 2020. Citado na página 32.

Russel, Stuart NORVING, P. Inteligência artificial. TC Rio de Janeiro „ n. 2, 2004. Citado na página 17.

Sandra V. M. Pereira GLAUCIA O.A.B.MEIRELES, J. M. R. R.; REI, M. A. dos. Modalidade ava nos dias atuais de pandemia da covid-19 e sua contribuição para a aprendizagem. UniEVANGÉLICA, p. 1–6, 2020. Citado na página 22.

SCHWITTER, R. Controlled natural languages for knowledge representation. Centre for Language Technology Macquarie University, 2007. Citado na página 18.

TEIXEIRA, J. de F. *Inteligência artificial*. [S.l.]: Pia Sociedade de São Paulo-Editora Paulus, 2014. Citado na página 16.

TORRES[1], A. C. M. et al. Interação e comunicação da comunidade surda em grupos do aplicativo whatsapp. *Dados*, p. 30, 2008. Citado 2 vezes nas páginas 15 e 28.

WATSON, T. J. Ibm's deep blue chess grandmaster chips. Research Center, USA, 1999. Citado na página 23.

WEISER, M. The computer for the 21st century. Scientific American, n. 265, p. 94–104, 1991. Citado na página 14.

WEIZENBAUM, J. et al. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, New York, NY, USA, v. 9, n. 1, p. 36–45, 1966. Citado na página 30.