GUILHERME SOUZA MONTEIRO

Orientador: Jadson Castro Gertrudes

HELENA: UM *CHATBOT* PARA AUXÍLIO DOS DISCENTES DO DECOM EM TRÂMITES UNIVERSITÁRIOS.

Ouro Preto-MG

Agosto de 2021

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Ciência da Computação

HELENA: UM *CHATBOT* PARA AUXÍLIO DOS DISCENTES DO DECOM EM TRÂMITES UNIVERSITÁRIOS.

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

GUILHERME SOUZA MONTEIRO

Ouro Preto-MG
Agosto de 2021

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

M775h Monteiro, Guilherme Souza .
Helena [manuscrito]: um chatbot para auxílio dos discentes do DECOM em trâmites universitários. / Guilherme Souza Monteiro. - 2021.
63 f.

Orientador: Prof. Dr. Jadson Gertrudes.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Engenharia de Software. 2. Processamento de Linguagem Natural (Computação). 3. Internet Relay Chat. I. Gertrudes, Jadson. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.41

Bibliotecário(a) Responsável: Soraya Fernanda Ferreira e Souza - SIAPE: 1.763.787

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

FOLHA DE APROVAÇÃO

Guilherme Souza Monteiro

Helena: Um chatbot para auxílio dos discentes do DECOM em trâmites universitários

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

Aprovada em 23 de Agosto de 2021.

Membros da banca

Jadson Castro Gertrudes (Orientador) - Doutor - Universidade Federal de Ouro Preto
Pedro Henrique Lopes Silva (Examinador) - Mestre - Universidade Federal de Ouro Preto
Renata Teles Moreira (Examinadora) - Doutora - Universidade Federal de Lavras

Jadson Castro Gertrudes, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 23/08/2021.

Documento assinado eletronicamente por **Jadson Castro Gertrudes**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 24/08/2021, às 10:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 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 **0210272** e o código CRC **F836F31A**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591692 - www.ufop.br

Resumo

Podemos definir *chatbot* como um programa de computador que utiliza Inteligência Artificial para simular diálogos de forma humanizada, sendo capaz de interagir com pessoas através de mensagens de texto ou voz de forma automatizada. Apresentamos neste trabalho o processo de desenvolvimento do primeiro protótipo de Helena, um *chatbot* construído para auxiliar os discentes do curso de Ciência da Computação em relação aos trâmites universitários. Este protótipo é capaz de responder, de forma imediata, algumas das dúvidas mais recorrentes entre os estudantes do curso. O que, futuramente, pode evitar o desgaste que hoje o Colegiado do Curso de Ciências da Computação (COCIC) possui ao responder inúmeros emails com conteúdo repetitivo. Vários dos exemplos utilizados para ensinar Helena foram fornecidos pelo próprio COCIC, garantindo que ela pudesse aprender com exemplos reais. O *chatbot* Helena foi desenvolvida usando a plataforma do *IBM Watson Assistant* para o processamento de linguagem natural e utiliza serviços de nuvem, que possibilitam sua disponibilidade e escalabilidade. Todo código fonte utilizado no desenvolvimento deste trabalho está disponível para toda a comunidade.

Palavras-chave: Chatbots, Engenharia de Software, Computação nas Nuvens e Inteligência Artificial

Abstract

We can define chatbot as a computer program that uses Artificial Intelligence to simulate dialogues in a humanized way, being able to interact with people through text or voice messages in an automated way. In this work, we present the development process of Helena's first prototype, a chatbot built to assist students of the Computer Science course in relation to university procedures. This prototype is able to answer, immediately, some of the most recurring doubts among students of the course. Which, in the future, can avoid the wear and tear that today the Collegiate of the Computer Science Course (COCIC) has when answering numerous emails with repetitive content. Several of the examples used to teach Helena were provided by COCIC itself, ensuring that she could learn from real examples. Helena was developed using the IBM Watson Assistant platform for natural language processing and uses cloud services, which enable its availability and scalability. All source code used in the development of this work is available to the entire community.

Keywords: Chatbots, Software Engineering, Cloud Computing and Artificial Intelligence.

Dedico este trabalho aos meus pais, Fernando e Monica, por todo apoio que me deram.

Agradecimentos

Agradeço primeiramente a minha mãe, por ter me ensinado a ter perseverança e resiliência para encarar os desafios da vida. Ao meu pai, meus irmãos e Esthefanie, por todo apoio durante os momentos difíceis. Meus companheiros da República Jardim de Alah, por estarem sempre do meu lado e me dando forças para continuar. Aos meus queridos colegas da Koxapa, sem eles esse trabalho não seria completo. Por fim, gostaria de agradecer também ao Colegiado do Curso de Ciências da Computação (COCIC), por acreditar no potencial deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AIML Artificial Inteligence Markup Language

API Aplication Programming Interface

AWS Amazon Web Services

COCIC Colegiado do Curso de Ciência da Computação

DECOM Departamento de Ciência da Computação

HTTP Hypertext Transfer Protocol

IFTTT If-This-Than-That

JSON JavaScript Object Notation

NoSQL Not Only SQL

REST Representational state transfer

SQL Structured Query Language

Capítulo 1

Introdução

O *chatbot* é um software que utiliza Inteligência Artificial para imitar diálogos e personalidades humanas. Alguns destes programas são construídos com o objetivo de convencer as pessoas de que estão com seres humanos reais (Kane, 2016). Os *chatbots* podem ser criados para realizar conversas simples ou até imitar personalidades como *Falso Capitão Kirk*[1], por exemplo. Esta incrível expansão se deu com a ampla adoção da internet e das mídias digitais, que criaram a necessidade de programas que pudessem se comunicar com o consumidor em plataformas como: *E-commerce*, marketing, serviços ao consumidor e coleta de dados, (ZEMČÍK, 2019).

Segundo Kane (2016), os *chatbots*, com sua capacidade de reconhecimento de padrões e o poder de associar vários destes em uma única resposta, se tornam escolhas naturais para prover instruções e referências para nós, humanos. Hoje, temos vários exemplos de *chatbots* que desempenham funções de assistentes como Jill Watson (Goel e Polepeddi, 2016) e LiSA (Dibitonto et al., 2018). Mas também temos casos de *chatbots* que são tão hábeis em simular os padrões de conversa humanos, que conseguem manter sessões de conversa com usuários durante meses, criando vínculos reais com pessoas, como é o caso de XiaoIce (Zhou et al., 2020).

Em muitos casos, existem respostas padrões que podem ser transmitidas para um usuário, evitando assim o desgaste de pessoas que precisam prestar essas informações repetidamente, de forma individualizada. Um desses cenários, objeto de estudo no presente projeto, está relacionado ao Colegiado do Curso de Ciência da Computação da Universidade Federal de Ouro Preto (COCIC).

O COCIC é responsável por sanar dúvidas de muitos estudantes, recém-chegados a universidade, sobre tópicos como trancamentos de disciplinas e exames especiais. No outro sentido, alunos no final do curso possuem dúvidas relacionadas à colação de grau, por exemplo. Podemos dizer que grande parte destes questionamentos poderiam ser facilmente solucionados com uma resposta padrão ou uma consulta rápida nos portais da instituição. Por isso, a implantação deste projeto seria de grande ajuda para os integrantes do COCIC, pois reduziria, de certa forma, o tempo gasto com o atendimento de grande parte destes alunos, deixando-os com uma maior disponibilidade para atender os casos mais complexos e de maior urgência.

[1] http://callmom.pandorabots.com/pandora/talk?botid=d74317427e345ab2

1.1 Justificativa

Existem três caminhos possíveis para que alunos dos cursos de graduação possam sanar dúvidas em relação à qualquer processo dentro da universidade: Perguntar para colegas, acessar a página do Manual do Aluno[2] ou entrar em contato com o colegiado do curso (forma presencial ou via *email*). Por conta da pandemia de COVID-19, muitas das atividades presenciais da Universidade Federal de Ouro Preto foram suspensas. A pandemia do novo coronavírus (COVID-19) gerou um volume ainda maior de contatos via *email* com o COCIC, o que de certa forma acabou por sobrecarregar os funcionários que fazem parte dele. Através de *emails* de exemplo fornecidos pelo próprio COCIC, percebemos que muitas das dúvidas enviadas pelos alunos são, de fato, bastante recorrentes e vários padrões se repetem no decorrer das mensagens.

Com o objetivo de auxiliar os membros do COCIC na resposta aos alunos do Departamento de Computação da Universidade Federal de Ouro Preto (UFOP) surgiu o *chatbot* Helena, com o propósito de se tornar mais um caminho de consulta para os alunos de graduação, por meio de um serviço que funciona 24 horas e que pode sanar dúvidas de forma imediata. Eventualmente, servindo como um primeiro auxílio para aqueles alunos que possuem dúvidas mais comuns, deixando para que o COCIC resolva apenas aquelas dúvidas com teor de complexidade maior que não puderam ser atendidas pelo *chatbot* Helena.

1.2 Objetivos

O objetivo principal deste trabalho é apresentar a arquitetura, implementação e funcionamento do *chatbot* Helena, que poderá auxiliar os alunos de graduação do curso Ciências da Computação a sanar as dúvidas mais recorrentes em relação aos trâmites universitários. E que, por consequência, possa reduzir o número de horas gastas pelo Colegiado do Curso de Ciências da Computação (COCIC) respondendo *emails* com teor repetitivo.

Os objetivos específicos deste projeto são:

- Elaborar uma arquitetura válida para a disponibilização do projeto via *internet*.
- Validar o software por meio de questionários simulados;
- Coletar informações do usuário, de forma anônima, para análises posteriores.

1.3 Organização do trabalho

Este trabalho é composto da seguinte forma: O Capítulo 2 consta a fundamentação teórica necessária para leitura desta obra. No Capítulo 3, apresentamos uma breve história dos

[2]Disponível em: https://www.prograd.ufop.br/manual-dao-alunao

chatbots e como estes são utilizados no meio educacional. O Capítulo 4 apresenta os detalhes do desenvolvimento deste projeto. No Capítulo 5 realizamos as experimentações deste projeto. E por fim, apresentamos as conclusões, limitações e trabalhos futuros no Capítulo 6.

Capítulo 2

Fundamentação Teórica

Neste capítulo são descritos, de forma breve, alguns dos conceitos mais importantes utilizados durante o desenvolvimento deste projeto.

2.1 API

API é a sigla para *Application Programming Interface*, que é um software intermediário que permite que duas aplicações interajam entre si, (Lane, 2019). É uma interface de computação que define as interações entre vários tipos de software. Ela define os tipos de chamadas ou solicitações que podem ser feitas, como fazê-las, os formatos de dados que devem ser usados, as convenções a seguir, entre outros aspectos. Também pode fornecer mecanismos de extensão para que os usuários possam estender a funcionalidade existente de várias maneiras e em vários graus.

Uma API pode ser totalmente customizada, específica para um componente ou projetada com base em um padrão da indústria para garantir a interoperabilidade. Por meio da ocultação de informações, as APIs permitem uma programação modular, garantindo que os usuários utilizem a interface independentemente da implementação, (Hubspire, 2019).

2.1.1 APIs REST

REST é a sigla para *REpresentational State Transfer*. É um estilo arquitetônico para sistemas hipermídia distribuídos e foi apresentado pela primeira vez por Fielding (2000). Segundo o autor, para que uma API seja considerada ***RESTful*** é necessário que ela siga uma série de restrições que são descritas a seguir:

- **Cliente-servidor**: Deve se separar as responsabilidades da interface do usuário do armazenamento de dados. Com isso, é possível utilizar a mesma lógica de armazenamento para várias interfaces. E o mais importante, permitindo com que os dois possam evoluir de forma independente.
- ***Stateless*** Cada requisição do cliente para o servidor deve conter todas as informações necessárias para que ele entenda requisição.
- **Armazenável em cache**: Os dados contidos na resposta da requisição precisam ser rotulados como armazenável ou não em *cache*. Se a resposta for armazenável em *cache*, então o cliente tem o direito de poder usar os dados dessa resposta mais tarde.

- **Interface uniforme**: Cliente e servidor precisam estabelecer um contrato entre si para que haja uma comunicação fluente entre os lados.
- **Sistema em camada**: Arquitetura deve ser construída através de camadas gerenciadas de forma independente, onde cada uma destas camadas não pode ver além da adjacente. As mudanças feitas em uma camada não devem impactar nos demais

Dentro do padrão **REST**, a principal abstração para informação é denominada de recurso. Qualquer informação que pode ser nomeada pode ser um recurso: um documento ou imagem, uma coleção de outros recursos, um objeto não virtual (por exemplo, uma pessoa) e assim por diante (Fielding, 2000).

2.1.1.1 Métodos de Requisição HTTP

O HTTP define um conjunto de métodos de requisição para indicar a ação desejada a ser executada para um determinado recurso. Embora também possam ser substantivos, esses métodos de requisição às vezes são chamados de verbos HTTP. Cada um deles implementa uma semântica diferente, mas algumas características comuns são compartilhadas por um grupo deles: por exemplo, um método de requisição pode ser seguro, idempotente ou armazenável em cache (MDN, 2020).

- **GET**: As requisições usando GET devem apenas recuperar dados.
- **HEAD**: Muito similar a requisição GET, mas sem o corpo da resposta.
- **POST**: É usado para enviar um recurso, geralmente causando uma mudança no estado ou efeitos colaterais no servidor.
- **PUT**: Substitui todas as representações atuais no recurso de destino pelos dados contidos na requisição de origem.
- **DELETE**: Exclui o recurso especificado.
- **CONNECT**: Estabelece um túnel de conexão para o servidor.
- **OPTIONS**: É usado para descrever as opções de comunicação para o recurso de destino.
- **TRACE**: Executa um teste de *loop-back* ao longo do caminho para o recurso de destino.
- **PATCH**: É usado para aplicar modificações parciais a um recurso.

2.1.2 API Gateway

A *API Gateway* é um componente importante da arquitetura, ela abstrai as funções que são comuns dentre os serviços fornecendo um única opção de entrada. Isso faz com que este componente encapsule todas as implementações e interfaces internas do sistema, (Zhao et al., 2018).

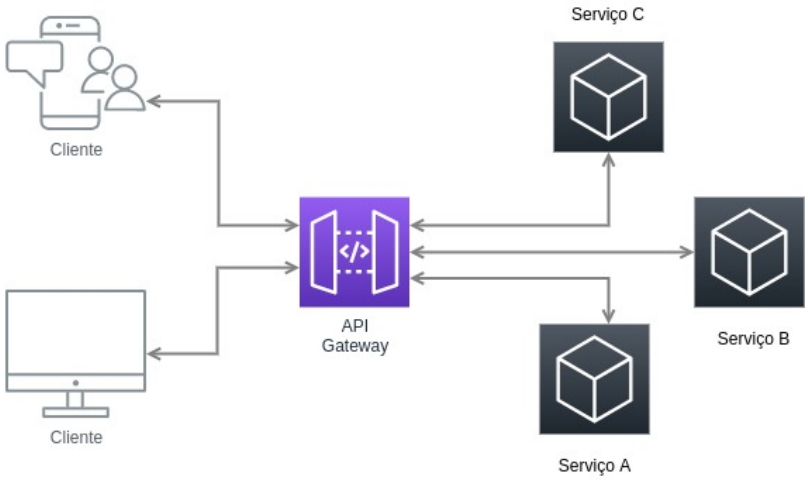


Figura 2.1: Exemplo de uma API Gateway. Fonte: Elaborado pelo autor

A Figura 2.1 exemplifica o funcionamento básico de uma *Api Gateway*.Onde temos que o cliente não precisa saber qual serviço específico está sendo chamado quando uma requisição é feita. Portanto, podemos dizer que a principal função deste componente é abstrair toda a complexidade dos serviços de *backend* para que se mostrem de forma uniforme para o cliente.

2.2 Computação em Nuvem

A computação em nuvem é um modelo para permitir acesso, sob demanda, a uma de rede de recursos de computação configuráveis por exemplo: Redes, servidores, armazenamento, aplicativos e serviços. Estes recursos podem ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento ou interação com o provedor de nuvem (Mell et al., 2011), ou podemos ter uma definição mais simplista como a de Knorr e Gruman (2008), onde a computação em nuvem seria como uma versão atualizada da computação utilitária, sendo basicamente um servidor virtual.

2.2.1 Computação Serverless

Computação *serverless* é um modelo no qual o provedor de nuvem gerencia, de forma dinâmica, os recursos computacionais que são alocados para os serviços (Rajan, 2018). Desta

forma, aumentando bastante a escalabilidade de uma aplicação e diminuindo muito a necessidade da manutenção de servidores e infraestrutura como um todo.

2.3 Banco de Dados e NoSQL

Inicialmente, a maioria dos registros eram feitos de forma manual. Porém, o advento da tecnologia levou à que mudanças drásticas ocorressem nos últimos anos, criando assim, a necessidade de formas mais robustas para armazenar e gerir dados, (Gupta et al., 2017).

O modelo mais disseminado para se armazenar dados e amplamente utilizado hoje no mercado é o modelo relacional (Khasawneh et al., 2020). Este modelo segue uma estrutura composta por tabelas, linha, colunas e utiliza uma linguagem específica para que seja possível realizar operações na base de dados. O SQL (*Structured Query Language*), serve para que o usuário consiga realizar consultas e criar, deletar e atualizar itens na base de dados.

Segundo Gupta et al. (2017), o maior desafio que temos em relação a esse crescimento espantoso do volume de dados é a sua não uniformidade. Por conta deste problema, as bases de dados não relacionais ou **NoSQL** (*Not Only SQL*) foram criadas. Uma grande vantagem deste modelo não relacional é que sua vantagem em escalabilidade, que se dá de maneira horizontal, onde se aumenta o número de servidores e não o hardware da máquina como acontece com os bancos de dados relacionais.

As bases de dados NoSQL não tem os mesmos esquemas rígidos dos relacionais e tem um foco maior em armazenar grandes quantidade de dados não estruturados e prover enorme velocidade em operações de escrita e leitura. Temos, que de forma geral, existem quatro tipos de bases NoSQL e estas são:

- **Banco de Dados em Grafo**: A fundamentação desse tipo de base de dados pode ser encontrado na teoria dos grafos, (Marcus, 2020).
- **Chave-valor**: A dados são armazenados em duas partes chamadas de chave e valor.
- **Banco de Dados de Documento**: Esta é uma extensão do armazenamento de chave-valor onde os valores são armazenados em estruturas mais complexas como o JSON (*JavaScript Object Notation*).
- **Armazenamento de Coluna**: Neste tipo os dados são armazenados na forma de seções de colunas.

Capítulo 3

Trabalhos Relacionados

3.1 Uma Breve História dos Chatbots

Nas próximas subseções a seguir iremos contar, através de exemplos, os avanços realizados no campos da Inteligência Artificial e Processamento de Linguagem Natural, que possibilitaram a grande ascensão dos *chatbots* modernos que conhecemos hoje.

3.1.1 ELIZA

Em janeiro de 1966 nasceu para o mundo acadêmico o primeiro programa de computador com a finalidade de estudar a comunicação em linguagem natural entre homem e máquina. ELIZA (Weizenbaum, 1966) usa um método para reconhecimento de palavras-chave na entrada que são correspondidas por frases pré-programadas na saída. Foi escrito usando MAD-Slip para o IBM 7094 e possui uma lógica de funcionamento bastante simples.

ELIZA lê o texto de entrada e procura por palavras-chave. Caso esta determinada palavra seja encontrada, então a sentença é transformada de acordo com a regra associada àquela chave. As palavras-chave também são elencadas em um *ranking* de importância. ELIZA é bastante sensível a este *ranking*, a ponto de abandonar uma palavra-chave de valor menor em favor de uma com maior importância. O programa reconhece pontos e vírgulas como delimitadores dentro de uma mensagem, então toda vez que umas destas pontuações for reconhecida, todo texto subsequente da mensagem é deletado. Da mesma forma, quando nenhuma palavra-chave for reconhecida dentro de um delimitador, então todos texto anterior também é removido. Deste modo, apenas uma sentença é transformada na resposta.

Podemos detalhar um pouco o processo de transformação de sentenças com o seguinte exemplo: Imagine a frase *"I'm very unhappy these days"*. ELIZA considera que *"I'm"* é uma palavra-chave, então procura encaixá-la dentro de um *template*. De acordo com ELIZA, a resposta para esta sentença seria *"How long have you been unhappy?"*, onde a regra de transformação é *"I'm X"* para *"How long have you been X?"*.

Um dos scripts mais famosos de ELIZA é aquele que a faz assumir o papel de um psiquiátra Rogeriano[1] que, de acordo com Strupp (1955), é aquele que possui

> "forte predileção por respostas reflexivas, com uma concomitante falta de respostas em todas as outras categorias".

[1] http://psych.fullerton.edu/mbirnbaum/psych101/Eliza.htm

A justificativa para isto se dá no momento em que, segundo Weizenbaum (1966):

> "Este modo de conversação foi escolhido porque a entrevista psiquiátrica é um dos poucos exemplos de comunicação diádica em linguagem natural categorizada em que um dos pares participantes é livre para assumir a postura de não saber quase nada do mundo real."

Desta forma, se você disser ao psiquiatra, por exemplo: *"Eu fiz um passeio de barco"* e ele responder com *"Me fale mais sobre barcos"*, o entrevistado em questão não acreditaria que o psiquiatra não sabe nada sobre barcos, mas sim que ele pretende levar a conversa naquela direção por um motivo específico.

No entanto, a afirmação de que ELIZA não possui nenhum conhecimento do mundo real não é totalmente verdadeira, pois ela é capaz de responder a seguinte sentença *"Todo mundo me odeia"* com *"Você consegue pensar sobre alguém em particular?"*. Isso mostra como ELIZA consegue manter uma ilusão de entendimento com um mecanismo tão simples. Porém, ELIZA não tem a capacidade de compreender contexto ou de abstrair conclusões válidas sobre o que está sendo dito. Isso se deve muito ao fato de que ELIZA não armazenava entradas de usuários, para que assim, pudesse extrair perfis e informações que contribuíssem com um diálogo ainda mais rico.

Acreditamos que ELIZA foi programada para manter o diálogo de uma maneira fluída por um certo período de tempo, mas não para chegar a conclusões. Contudo, mesmo com recursos computacionais escassos, teve sucesso em convencer alguns dos participantes durante os testes de que havia realmente um ser humano do outro lado da tela, e não um *software*, (Weizenbaum, 1966). Podemos afirmar que ELIZA foi um experimento crucial para o desenvolvimento do estudo da Inteligência Artificial e Processamento de Linguagem Natural.

3.1.2 A.L.I.C.E

Cerca de 40 anos depois é apresentado ALICE (Artificial Linguistic Computer Entity) (Wallace, 2009), considerado o primeiro chatbot baseado em AIML (Artificial Inteligence Markup Language) (Wallace, 2003), que proporcionou um avanço considerável na construção dos chatbots modernos. ALICE chegou a ganhar, por três vezes, o Prêmio Loebner [2] como *"o computador mais humano"* no anual concurso do Teste de Turing.

ALICE usa um modelo de aprendizagem supervisionada, onde uma pessoa, chamada de *botmaster*, observa os diálogos e realiza alterações nos arquivos AIML para adequar as conversas, tornando-as mais críveis e *"humanas"*.

```
<category>
    <pattern>YES</pattern>
```

[2] https://aisb.org.uk/aisb-events/

```
        <that>DO YOU LIKE MOVIES</that>
        <template>What is your favorite movie?</template>
    </category>
```

O exemplo acima, retirado de (Wallace, 2009), mostra de forma simples algumas das principais funcionalidades da linguagem AIML. Podemos dizer que a unidade básica de conhecimento é chamada de categoria, demarcada pela tag <**category**/>. Cada categoria pode ser composta por uma pergunta, uma resposta e um contexto. Neste exemplo a tag <**that**/>, nosso marcador de contexto, identifica a última declaração feita pelo *chatbot*. Portanto, temos que esta categoria será ativada se o padrão *'YES'* for encontrado na resposta do usuário.

Outra funcionalidade importante da linguagem AIML é a recursividade, feita através da tag <**srai**/>. Esta funcionalidade pode usada de varias formas durante o processamento da mensagem, como: Dividir a entrada em uma ou mais subpartes, mapear sinônimos, correções de ortografia e gramática, detecção de palavras-chave e condicionais. Porém, dentre todas estas, vale ressaltar a redução simbólica.

```
  <category>
    <pattern>DO YOU KNOW WHO * IS</pattern>
    <template><srai>WHO IS <star/></srai></template>
</category>
```

Redução simbólica é o processo de transformar sentenças complexas em formas mais simples, que possibilitem o processamento por parte de ALICE. O exemplo acima demonstra como uma categoria pode reduzir a sentença que está na forma de *"Você sabe quem é A?"* para *"Quem é B?"*. Usando deste artifício, é possível relacionar de forma mais simples a entrada com a base de conhecimento, (Wallace, 2009)

Percebemos que a linguagem AIML possui um número considerável de elementos em sua estrutura, algumas das mais básicas foram exemplificadas acima, a fim de demonstrar seus conceitos fundamentais. Acreditamos que a rica estrutura do AIML tenha proporcionado à ALICE uma personalidade crível para o tempo em que foi testada, servindo de grande inspiração para as próximas inteligências artificiais a serem desenvolvidas.

3.1.3 IBM Watson

Um ano depois da publicação do artigo de Wallace (2009), surge Watson, a inteligência artificial criada pela IBM, inaugurando assim a era dos Sistemas Cognitivos (High, 2012). Soluções cognitivas são capazes de entender diferentes tipos de dados, desde bases de dados estruturadas até textos científicos. São treinadas para abstrair conteúdo técnico e específico usando ferramentas de inteligência artificial como aprendizado de máquina e modelos preditivos (Chen et al., 2016).

Watson se tornou bastante famoso ao ganhar um jogo de *Jeopardy*[3] contra, os até então campeões, Brad Rutter e Ken Jennings. *Jeopardy* é um jogo tradicional da televisão americana que que testa o conhecimento geral de seus participantes.

Watson é classificado como um Sistema de Processamento de Linguagem Natural Profundo e consegue ter essa acurácia devido às suas tentativas de acessar o máximo de contexto possível. Esse contexto pode ser obtido tanto no conteúdo da pergunta quanto em sua base de conhecimento, conhecido como *corpus*, (Chen et al., 2016).

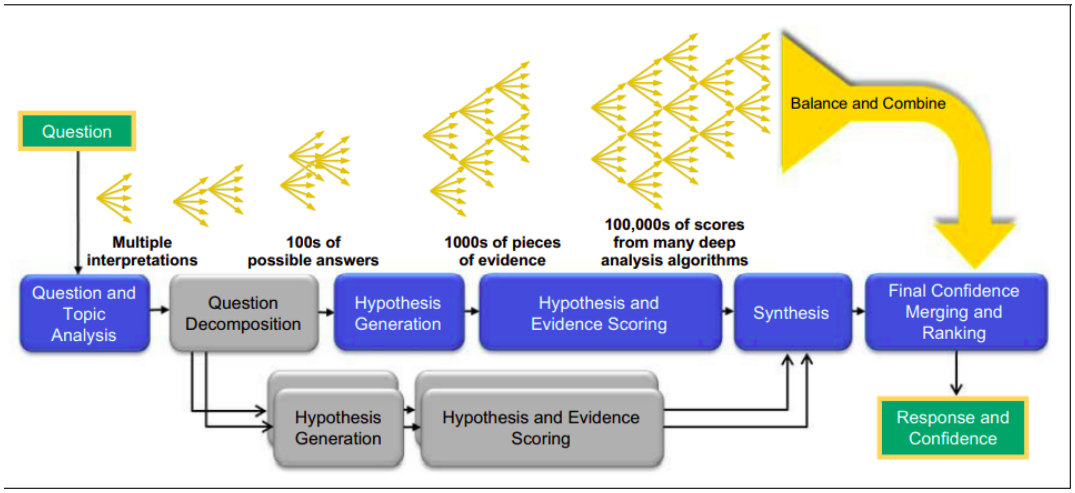


Figura 3.1: Como IBM Watson processa perguntas. Fonte:(Chen et al., 2016)

Como mostra a Figura 3.1, quando uma pergunta é feita, Watson primeiramente à analisa para que seja possível extrair suas *features*. Após extraí-las, a ferramenta Watson gera uma série de hipóteses e começa a procurar no *corpus* passagens que possam conter uma possível resposta. Depois que as possíveis respostas são elencadas, é feita uma comparação entre a linguagem da pergunta e a linguagem de cada possível resposta.

O passo descrito anteriormente é o mais complexo. São usados conjuntos de algoritmos, onde cada um deles é responsável por analisar uma espécie de *feature* diferente como: Termos parecidos, tempos verbais e informações relevantes para o contexto. Cada um destes algoritmos resulta em um score diferente que é usado em um modelo estatístico. Este modelo pode ser resumido em uma porcentagem de confiança que Watson tem em relação à resposta (High, 2012).

Watson possui em sua comunidade mais de 280 parceiros comerciais que geram mais de 3 bilhões de requisições em sua API mensalmente, (Murtaza et al., 2016). Isso demonstra que Watson é uma solução robusta para criação de assistentes, pois já é validada pelo mercado.

[3]https://www.jeopardy.com/

3.1.4 Xiaoice

Microsoft Xiaoice ou *'Little Ice'*, tem sido um dos *chatbots* mais populares do mundo. Desde que foi lançado na China em 2014, já atraiu mais de 660 milhões de usuários ativos e se encontra disponível em mais de 40 plataformas, incluindo *WeChat* e *Facebook Messenger*, (Zhou et al., 2020). O maior objetivo do software Xiaoice é estabelecer conexões duradouras com seus usuários causando um envolvimento emocional. Daí podemos dizer que Xiaoice foi um passo enorme na era dos chatbots sociais (Shum et al., 2018)

O software Xiaoice se baseia em três pilares na construção de seu sistema: Quociente de inteligência (QI), quociente emocional (QE) e Personalidade. O QI inclui seu conhecimento e seu modelo de processamento de memória, imagem e linguagem natural. Esses conceitos são fundamentais para o desenvolvimento das habilidades de conversa (*dialog skills).* Nos últimos 5 anos, foi possível que Xiaoice desenvolvesse mais de 230 habilidades, variando desde responder perguntas, até indicação de filmes. O QE trata de identificar as emoções em uma conversa para que possa adaptar suas respostas ao contexto emocional do usuário, gerando engajamento no diálogo. A personalidade é definida por um conjunto de características de comportamentos e padrões emocionais que definem o indivíduo. Xiaoice precisa de manter uma personalidade constante para que seja viável manter relações de longa duração com seres humanos.

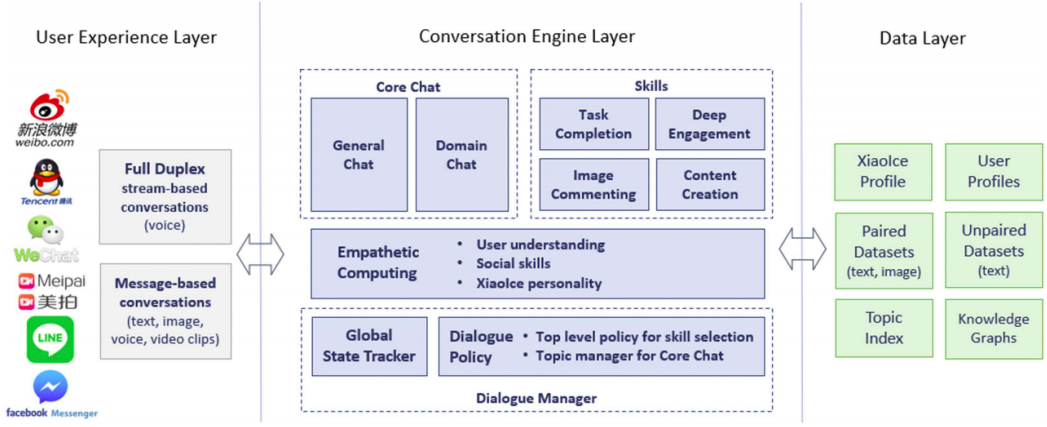


Figura 3.2: Arquitetura de Xiaoice. Fonte: (Zhou et al., 2020)

A camada de experiência do usuário, assim como demonstrado na figura 3.2, faz a conexão das plataformas de chat com o sistema de Xiaoice. A comunicação pode ser feita de duas formas diferentes: *Full-duplex* para ligar com *streaming* de voz ou por turnos, que seriam as mensagens de texto. O motor de conversa é composta pelo gerenciador de diálogo e tem a responsabilidade de interpretar as mensagens de texto, extrair emoções e detectar as *dialog skills* para que seja possível elaborar uma resposta. A Camada de dados consiste em um conjunto de bases que armazenam os dados extraídos de todas as conversas realizadas, (Zhou

et al., 2020).

Diferentemente dos modelos de *chatbot* apresentados anteriormente, Eliza (Weizenbaum, 1966) e Alice (Wallace, 2003), feitos para pequenas conversas e que procuram identificar padrões já mapeados de forma estruturada, Xiaoice tem o objetivo de criar relações de longo prazo. Desta forma, podendo servir os usuários não só de maneira organizacional, como agendar reuniões ou adicionar itens em uma lista de compras, mas sim atendê-los de em suas necessidades emocionais e de pertencimento social.

3.2 O Papel dos Chatbots na Educação

Com o crescimento vertiginoso da educação a distância durante os 20 anos de século 21 (Rocha et al., 2020), se faz necessário que adotemos ferramentas que possam contribuir com a interação e aprendizado dos estudantes que seguem com esse estilo de educação que teve sua expansão ainda mais acentuada devido à pandemia de COVID-19 (Sun et al., 2020).

As subseções a seguir vem para demonstrar exemplos de assistentes que foram implementados dentro da área da educação, com enfase em cursos que são feitos de forma totalmente online. Todos os trabalhos listados a seguir serviram de base para o desenvolvimento de Helena como um todo.

3.2.1 Jill Watson

Os *Massively Open Online Courses* (MOOCs) estão proliferando rapidamente, de acordo com Shah (2016) em 2016 mais de 56 milhões de alunos se matricularam em mais de seis mil e oitocentos (6.800) cursos online que são oferecidos por mais de setecentas (700) instituições. No entanto, para muitos MOOCs, a eficiência do aprendizado é baixa e a retenção do conteúdo pelo estudante é questionável. Uma recomendação para melhorar a eficiência do aprendizado é aumentar as interações entre professor e estudante (Hollands e Tirthali, 2014). Porém, se nós assumirmos a existência de um professor assistente para cada cinquenta estudantes, seriam necessários cerca de um milhão de professores para suprir a demanda de 56 milhões de estudantes. Isso levanta o questionamento: Como seria possível providenciar assistência para milhares de estudantes cursando MOOCs?

Jill Watson (JW) (Goel e Polepeddi, 2016), nome esse que se deve ao fato de ter sido primeiramente implementado usando IBM Watson (High, 2012). JW é um assistente virtual construído pela *Georgia Institute of Technology* para disciplina de Inteligência Artificial, com intuito de auxiliar estudantes durante o aprendizado em MOOCs. JW, de forma autônoma, respondeu às apresentações dos estudantes, postou anúncios semanais e respondeu perguntas frequentes. Então, de certa forma, JW é uma ferramenta parcialmente automatizada para prover auxílio a estudantes de cursos online.

Para formar uma base de conhecimento, os pesquisadores coletaram as perguntas feitas por alunos dentro dos fóruns de discussão das disciplinas de Inteligência Artificial durante os períodos de 2014 e 2015. Foi constatado que a grande maioria das perguntas se repetiam e, até mesmo, durante as mesmas semanas (Goel e Polepeddi, 2016).

Ao todo foram apresentadas três versões de JW, sendo que apenas a primeira delas (JW1) utilizou o IBM Watson como base para processamento de linguagem natural. De acordo com Goel e Polepeddi (2016), JW1 seria, essencialmente, pares de perguntas e respostas organizadas por categoria. Ou seja, JW1 classifica a pergunta em uma determinada categoria e busca uma resposta relacionada, caso ela tenha uma confiança acima de 97%.

Inicialmente o JW1 tinha que passar por uma checagem humana antes de ter sua resposta publicada no fórum, conceito esse conhecido como *human-in-the-loop*. Dois meses após a primeira publicação do bot a checagem humana já havia sido removida.

JW1 não respondia em tempo real, mas sim em intervalos de 15 minutos. Uma análise era feita nas discussões ativas dentro dos fóruns e JW1 atuaria se caso houvesse alguma pergunta, onde tivesse a capacidade de responder e que ainda não tivesse sido respondida por assistente humano. Foi concluído que JW1 conseguia responder uma porcentagem pequena das perguntas que são realizadas, porém as respostas que são oferecidas estão quase sempre corretas.

Durante a primeira semana da aula da disciplina de Inteligência Artificial, os alunos são convidados a se apresentar através de uma pequena introdução. A partir daí, assistentes humanos dão as boas vindas, de forma personalizada a cada estudante. Isso faz que uma breve boas vindas se torne uma tarefa extremamente cansativa e que acaba por consumir bastante tempo, pois é normal que estes cursos possuam mais de 300 alunos. Foi quando realizaram o lançamento da JW2 (Jill Watson 2.0) para que cumprisse a tarefa de realizar a recepção dos alunos.

Diferentemente da JW1, a JW2 não foi construída tendo como base o IBM Watson, mas sim um sistema próprio criado por eles em laboratório, (Goel e Polepeddi, 2016). O que JW2 faz é mapear a introdução escrita em conceitos que sejam relevantes para que possam ser usados como índices na hora de buscar uma resposta apropriada.

Um teste realizado pelos pesquisadores em 2016, no qual os *deploys* dos dois assistentes desenvolvidos até então (JW1 e JW2) foi feito em uma turma iniciante do curso de Inteligência Artificial para que se pudesse avaliar a performance dos mesmos. Foram dados pseudônimos aos assistentes, para que pudessem se passar por humanos, JW1 usando o nome de *Ian Braun* e JW2 de *Stacy Sisko*. Stacy, da mesma forma que Ian, era acionada a cada 15 minutos entre 9 horas da manhã e 11 horas da noite para realizar as checagens. Caso houvesse alguma introdução que não tivesse sido respondida por um assistente humano, então esta deverá ser respondida por Stacy. Através do estudo, foi possível perceber que Stacy (JW2) foi capaz de responder mais de 40% de todas as introduções de forma autônoma. Ian (JW1), também

obteve uma performance melhor do que a da edição passada, devido ao fato de que já havia acumulado uma base de dados mais robusta.

Devido ao sucesso da implementação de JW2, os pesquisadores então decidiram construir uma nova versão do JW1 usando os mesmo conceitos de JW2, ou seja, com tecnologia construída dentro do laboratório. Daí surge o Jill Watson 3.0, ou JW3.

Em Janeiro de 2017 foi realizado um novo experimento para comprovar a performance dos assistentes em um ambiente real. JW2, se passando por *Liz Duncan* e JW3 batizado então de *Cassidy Kimball*, foram listados nos quadros de professores assistentes, assim como no teste realizado em 2016, os estudantes não foram notificados da existência das duas inteligências artificiais. Para este teste, os horários também foram alterados para funcionarem de 6 horas da manhã até às 23:59 da noite.

De acordo com artigo, (Goel e Polepeddi, 2016), *Liz Duncan* respondeu 60% de todas as introduções, uma performance consideravelmente melhor do que o teste passado. Também foi concluído através dos testes que *Cassidy Kimball* obteve um resultado melhor do que JW1 e *Ian Braun*. Onde *Cassidy* respondeu de forma completamente autônoma 34% de todas as perguntas, onde destas, 91% estavam corretas.

Desta forma, podemos dizer que o uso de *chatbots* podem contribuir muito na área da educação, podendo ter um alcance ainda mais efetivo no ensino a distância em escala (Como em cursos online, por exemplo). Aumentando a interação entre aluno e plataforma podemos proporcionar um aprendizado mais efetivo e com mais retenção de conteúdo.

3.2.2 LiSA

LiSA (*Link Student Assistant*), é um *chatbot* criado com a intenção de ajudar os estudantes em suas vidas acadêmicas através de informações e serviços (Dibitonto et al., 2018). O Objetivo deste estudo foi compreender como a personalidade do *chatbot* pode afetar a experiência do usuário e suas interações. É essencial que um *chatbot* adote uma personalidade para que possa moldar relações críveis e de empatia com seus usuários. Neste caso, o *chatbot* deveria adotar a personalidade de um professor, membro do administrativo ou um tutor?

O autor (Dibitonto et al., 2018) classificam os *chatbots* para ambientes universitários em duas categorias principais: a primeira seria aqueles que tem o propósito de recomendar uma determinada universidade para os interesses específicos do aluno. O segundo, são aqueles *chatbots* desenvolvidos para um contexto universitário específico. Alguns destes, podem ser envolvidos em tarefas específicas, como ajudar alunos a resolverem problemas de uma determinada disciplina (Goel e Polepeddi, 2016), enquanto outros podem servir como assistentes virtuais para ajudar alunos de todos os cursos a sanarem dúvidas em relação as normas da universidade, por exemplo Dibitonto et al. (2018).

Embora LiSA tenha sido classificada na categoria de assistente virtual, foi utilizada como *Survey Bot* ou Bot de Questionário, ou seja, um *chatbot* desenhado para realizar um questioná-

rio específico e armazenar os resultados. LiSA, usa de tecnologias como *Chatfuel* (Janarthanam, 2017), para fazer o processamento da linguagem natural e a plataforma do *Facebook Messager* para realizar o contato direto com os participantes da pesquisa.

LiSA adotou uma personalidade bastante amigável e informal. Devido que público alvo da pesquisa eram em sua maioria jovens, a comunicação também era enriquecida com vários *emoticons*, para fazer com que a conversa pudesse ser mais coerente com o ambiente de chat mobile.

A pesquisa foi feita com mais de 100 estudantes de várias universidades italianas, destes a maioria situada na *Link Campus University* mostra que, quase metade, 47,4% deles possuíam idades entre 15 e 18 anos. Os que possuíam idades entre 19 e 24 anos correspondiam a 50,9% e apenas 1,8% possuíam mais de 25 anos. Apenas 51,9% dos entrevistados sabiam o que era um *chatbot*. Dentre os que sabiam, era porque já haviam utilizado em alguma plataforma de serviço ao consumidor (65,4%). Dos que não sabiam o que é um chatbot, 14,3% acreditam que o *chat*, em si, não é uma ferramenta apropriada para entrar em contato com serviço de atendimento ao consumidor, por isso, nunca tiveram contanto com um. Outros participantes apenas disseram que sempre optaram por outros meios de comunicação como e-mail e telefone, por exemplo.

Em relação a experiência com LiSA, 59,6% do participantes descreveram a experiência como interessante, 29,8% como agradável e apenas 6,4% eram indiferentes. Alguns até chegaram a flertar e ofendê-la (6%). Mas após as agressões, LiSA demonstrava um comportamento de choque em relação as mensagens, o que acabava provocando constrangimento, fazendo com que os agressores pedissem desculpas. Porém, alguns poucos participantes continuariam as agressões para testar os limites de LiSA.

Portanto, podemos concluir que é relevante realizar o design do *chatbot* para que tenha empatia e sensividade as respostas, para que assim, seja possível obter relações educadas e respeitosas que sejam proveitosas para o usuário. Podemos ressaltar também a importância de comunicar ao usuário de que temos capacidade de identificá-lo, pois desta forma vários casos de mau comportamento por parte do usuário são, de certa forma, inibidos, (Dibitonto et al., 2018).

3.2.3 Assistente FAQ Sobre o Moodle

Uma boa interação depende de uma solução eficiente que possibilite a comunicação entre o aluno e o ambiente educacional. Uma maneira de atingir esse objetivo tem sido usar *chatbots* para responder as perguntas mais frequentes dos estudantes em um contexto educacional (da Silva Oliveira et al., 2019).

O desenvolvimento desse *bot* tem por objetivo ajudar estudantes dos cursos à distância (EAD) de instituições de ensino superior do Brasil que utilizam da plataforma *Moodle*. Que, de acordo com uma pesquisa feita pelo INEP em 2017, o número de matriculados em cursos

de graduação presencial caiu em 0.4%, enquanto houve aumento de 17.6% na modalidade a distância.

Este *bot* utilizou como base de conhecimento as perguntas mais frequentes feitas por estudantes de 22 cursos de graduação a distância de uma instituição de ensino superior do sul do Brasil. Estes dados foram coletados através do departamento de tecnologia da instituição da própria instituição.

No desenvolvimento do trabalho, também foi utilizado o IBM Watson (da Silva Oliveira et al., 2019) como ferramenta para processamento de linguagem natural e criação de diálogos. E utilizaram o *Facebook Messager* como meio de comunicação com o *bot*.

Para realizar a integração do *Facebook Messager* com a plataforma Moodle foi necessário: Primeiramente, criar uma API para o *bot* no *Facebook*, de forma que fosse possível receber o **Token de Acesso**, que é responsável por dar a API do Watson a capacidade de ler, escrever e modificar a página do *bot*. Logo em seguida, foi necessário conectar o *bot* ao *Facebook Manager* usando o **Token de Acesso a Página**, gerado na criação da API. Depois foi gerado uma URL de *callback*, que permite que o *bot* seja notificado quando as interações ocorrem, no formato de uma requisição *POST*. A última parte foi realizar a integração entre o *bot* e a plataforma Moodle, que foi feito usando um *script* gerado pela API do *bot* e adicionado no código HTML da página do Moodle.

As conversas com o *bot* foram analisadas durante um período de 4 semanas, essas interações foram separadas em três grupos diferentes: Respostas corretas, fora de contexto e respostas erradas. Ao longo desse tempo foram coletados dados de 14 interações, onde 43% das questões foram fora de contexto, 43% estavam dentro do contexto e foram respondidas corretamente e 14% estavam dentro do contexto, porém foram respondidas de forma incorreta.

Foi usado o *Net Promoter Score* (NPS) (Hamilton et al., 2014) para medir o grau de fidelidade e satisfação com o produto apresentado no trabalho.

Nota	**Nível**	**Quantidade**	**Ranking**	**%**
5	Excelente	8	Promotor	67,0
4	Muito Bom	2	Promotor	16,5
3	Bom	2	Neutro	16,5
2	Razoável	-	Detrator	-
1	Razoável	-	Detrator	-
0	Terrível	-	Detrator	-

Tabela 3.1: Resultado da avaliação NPS do Chabot FAQ Moodle. Fonte: (da Silva Oliveira et al., 2019)

Doze estudantes no total responderam à pesquisa, temos que o cálculo é feito com base nas diferenças das porcentagens de promotores e detratores. Como podemos observar, não houve avaliações consideradas detratoras. Portanto, O serviço foi considerado como excelente.

3.3 Criando um Chatbot

> "A dificuldade em construir um *chatbot* é menos técnica e mais experiência do usuário. Um dos mais prevalentes bem-sucedidos os *bots* no mercado são aqueles aos quais os usuários desejam voltar regularmente e que fornecem valor consistente para suas tarefas diárias e requisitos." — Matt Hartman

De acordo com Goyal et al. (2018), um *chatbot* típico encontra o contexto em uma determinada frase e retorna uma resposta apropriada para o usuário. Mas como é possível encontrar esse contexto? Segundo Goyal et al. (2018), antes de construir um *chatbot* é preciso ter respostas para as perguntas a seguir:

- Qual problema o *bot* pretende resolver?
- Em qual plataforma o *bot* será disponibilizado?
- Qual servidor será utilizado para hospedar o *bot*?
- O *bot* irá usar algum tipo de plataforma como IBM Watson, Botsify ou Chatfuel?

O trabalho de Goyal et al. (2018) demonstra de forma bastante detalhada como é possível criar um chatbot usando processamento de linguagem natural (PLN) em Python e disponibiliza-lo usando o *Heroku*[4] como plataforma de hospedagem em conjunto o Facebook Messager[5].

3.3.1 Transformando tarefas repetitivas em serviços de Chatbot

A popularidade da tecnologia dos chatbots o grande potencial de aplicação fez com que grandes empresas de tecnologia como Google, Facebook, Microsoft e IBM a lançarem *frameworks* que ajudassem desenvolvedores a contruir esses produtos. No entanto, esses *frameworks* reduzem o processo de criação a um estilo de programação muito popular nos anos 1980, chamado de *If-This-Then-That* (IFTTT) (Telang et al., 2018).

A Abordagem IFTTT é inadequada em termos tanto de flexibilidade e manutenibilidade do código, pois acabam misturando regras de negócio com as regras para gerenciar o diálogo. Como a tendência de um *chatbot* é que sua complexidade aumente de acordo o tempo, temos que a dificuldade de implementar novas funcionalidades se torne cada vez maior.

A Figura 3.3 retrata os cinco componentes descritos no *framework* proposto por Telang et al. (2018).

[4]https://www.heroku.com/
[5]https://www.messenger.com/

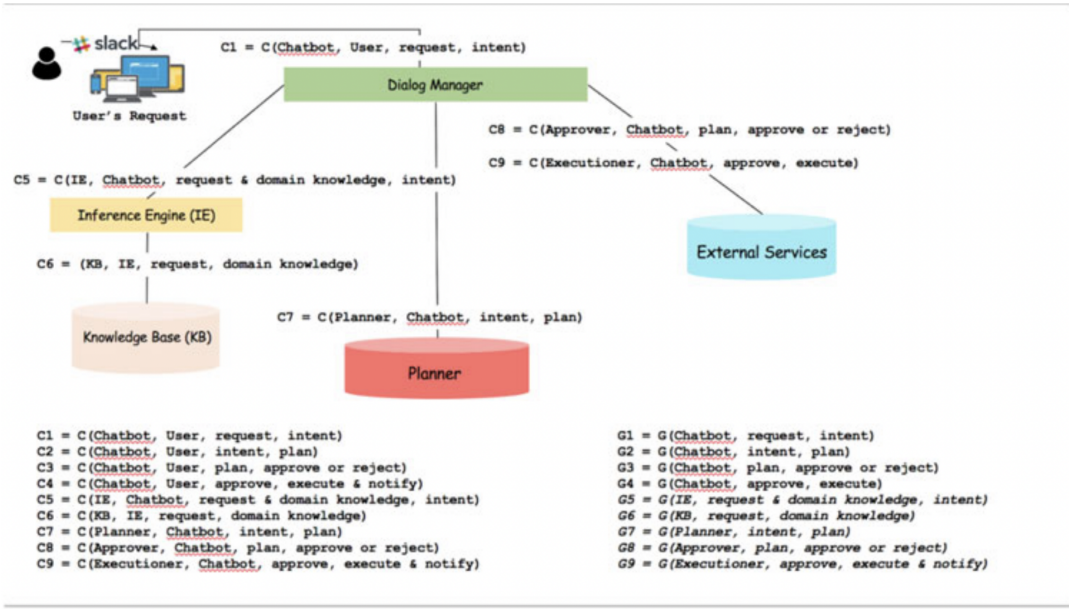


Figura 3.3: Framework conceitual para criação de chatbots complexos. Fonte: (Telang et al., 2018)

- **Gerenciador de diálogo** *(dialog manager)*: A sua função é utilizar técnicas de processamento de linguagem natural para extrair significado da mensagem do usuário. O gerenciador então comunica com outras entidades (motor de inferência e a base de conhecimento) para dar continuidade no processamento da requisição.
- **Motor de inferência** *(inference engine)*: interage diretamente com a base de conhecimento para detectar a intenção do usuário com base na inferência.
- **Base de conhecimento** *(knowledge base)*: Representa todos os dados da base, desde mensagens enviadas ao bot até o grafo de relações entre as entidades.
- **Planejador** *(planner)*: É responsável por fornecer um plano de ação ao gerenciador de diálogos. Caso o plano falhe, uma alternativa deve ser fornecida.
- **Serviços externos** *(external services)*: O gerenciador de diálogos interage com serviços externos para poder executar as regras de negócio.

Segundo Telang et al. (2018), desacoplar a base de conhecimento das ações executadas pelo *chatbot* promove uma grande flexibilidade durante o desenvolvimento, pois com isso é possível treiná-lo sem que novas regras sejam adicionadas. Desta forma, isolando os componentes da forma proposta é possível que qualquer mudança feita em qualquer um deles tenha um impacto mínimo no outro.

Capítulo 4

Desenvolvimento

Neste capítulo abordamos as etapas iniciais para a construção da primeira versão de Helena. Descreveremos os detalhes de como ocorreram as coletas de dados e como estes foram usados no treinamento. Também, apresentamos a arquitetura da aplicação e da forma como armazenamos dados de sessão e diálogo.

4.1 Coleta de Dados

A primeira coleta de dados se deu com a consulta dos principais sites de referência acerca de questões da vida acadêmica, tendo como base o aluno do curso de Ciências da Computação da Universidade Federal de Ouro Preto (UFOP) como: Colegiado do Curso de Ciências da Computação (COCIC)[1] e o Manual do Aluno escrito pela Pró-reitoria de Graduação (PROGRAD)[2]. O site do COCIC, como demostrado na Figura 4.1 oferece diversas instruções importantes acerca dos principais requerimentos apreciados pelo colegiado, sendo que estes são dúvidas frequentes dos alunos de Computação. Também é possível acompanhar as principais decisões e resoluções feitas pelo COCIC durante os últimos anos.

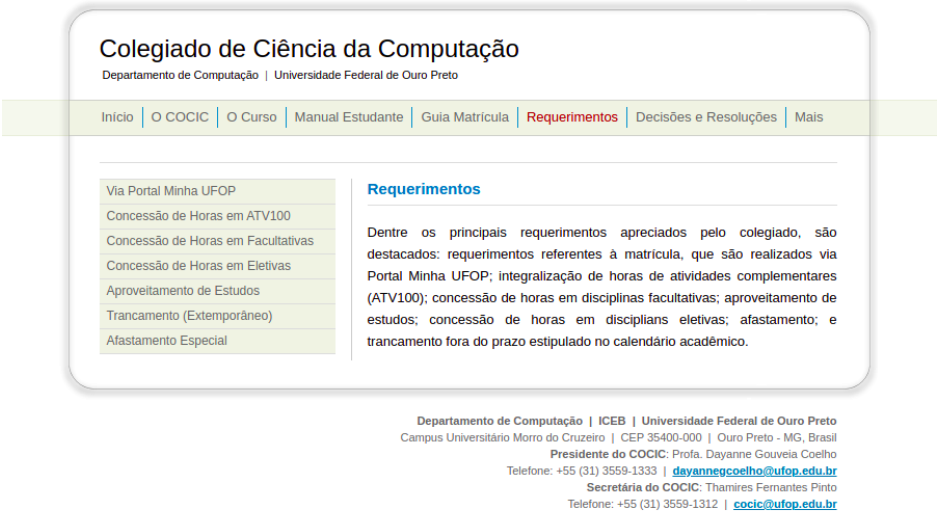


Figura 4.1: Seção de Requerimentos do site do COCIC. Fonte: (*decom.ufop.br/cocic/*)

[1] http://www.decom.ufop.br/cocic/
[2] https://www.prograd.ufop.br/manual-dao-alunao

O site da PROGRAD, como apresentado na Figura 4.2 , também oferece diversas informações acadêmicas como: Calendário acadêmico, colação de grau, horários de aulas e afins. Porém, no que tange a este trabalho, iremos focar no Manual do Aluno. Nele, as informações são elencadas no formato de tópicos que possibilitam que as respostas possam ser encontradas de forma fácil.

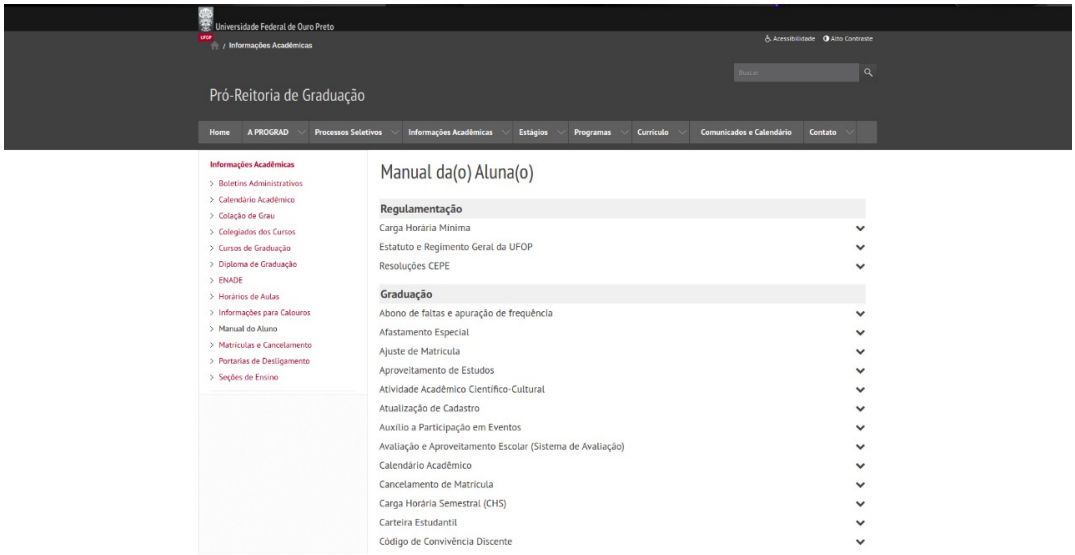


Figura 4.2: Atual site da PROGRAD. Fonte: (*prograd.ufop.br/manual-dao-alunao*)

Além do Manual como referência, tivemos acesso também a alguns exemplos das dúvidas mais recorrentes dos alunos do curso de Ciência da Computação. Estes exemplos foram coletados junto ao colegiado do curso. Os exemplos contém mensagens que abordam diversos temas no que diz respeito a vida acadêmica dos estudantes.

4.2 Treinamento

Como já descrito na seção acima, a principais fontes de dados para o treinamento vieram do Manual do Aluno, da página do COCIC e dos exemplos cedidos pelo próprio colegiado. Antes de detalhar mais o processo de treinamento do *chatbot* Helena, é importante esclarecer alguns termos que são utilizados pela plataforma do IBM Watson (IBM, 2021):

- **Intenções (*intents*)**: Intenção é o propósito ou objetivo expresso na mensagem do usuário. Ao reconhecer a intenção expressa na mensagem do usuário, Helena então pode selecionar o fluxo de conversa correto para respondê-la. Por exemplo, saber identificar se o aluno está com dúvidas sobre o cancelamento de matrícula ou documentação necessária para o contrato de estágio.

- **Entidades (*entities*)**: Uma entidade representa uma informação relacionada à finalidade do usuário na mensagem Se a intenção representa um verbo (uma ação que o usuário deseja realizar), a entidade representa um substantivo (o objeto ou contexto da ação). Por exemplo, quando se quer saber a previsão do tempo, as entidades relevantes de localização e data são necessárias antes que o aplicativo possa retornar uma previsão do tempo precisa. Identificar entidades na mensagem do usuário ajuda a formular respostas mais úteis e direcionadas. Por exemplo, você pode ter uma intenção de **comprar** algo. Quando o usuário faz uma solicitação que aciona a intenção **comprar**, a resposta do assistente deve refletir uma compreensão do que o usuário deseja **comprar**. Você pode adicionar uma entidade **produto** e usá-la para extrair informações sobre produtos que sejam de interesse dos clientes a partir da entrada do usuário. A sentença *"Quero comprar uma calça"* teria como intenção **comprar** e calça como **produto**.

Como apresentado na Figura 4.3, até o presente momento da escrita deste trabalho, o *chatbot* Helena é capaz de reconhecer apenas 16 intenções e nenhuma entidade. Dentre as 16 intenções, 12 são para reconhecimento de dúvidas dos usuários e apenas 4 são para responder perguntas básicas, que podemos relacionar a sua personalidade, como: *"Qual é sua idade?"* e *"Quem é você?"*

Intents (16) ↑	Description	Modified ↑↓	Examples ↑↓
#abater-horas-de-atv	Quando o aluno tem dúvidas sobre ATV100	an hour ago	8
#aproveitamento-de-disciplinas	Dúvidas sobre o aproveitamento de disciplina	an hour ago	5
#calendario-academico	Quando o estudade está querendo saber mais sobre o calendári...	a month ago	3
#cancelamento-de-matricula	Quando aluno deseja cancelar sua matrícula	a month ago	5
#carteira-estudantil	Quando o estudante deseja saber mais sobre sua carteirinha	a month ago	6
#certificados-monitoria-tutoria	Dúvidas sobre como aplicar as horas de monitoria e tutoria	an hour ago	4
#colação-de-grau	Dúvidas relacionadas a colação de grau	an hour ago	6
#desligamento-da-universidade	Dúvidas a respeito do desligamento	24 minutes ago	6
#estagio	Dúvidas sobre estágio	an hour ago	13
#exame-especial	Quando o aluno deseja saber mais sobre os exames especiais	a month ago	5
#horas-validas-de-atv	Quando o aluno quer saber quais horas são válidas para atv100	an hour ago	7
#idade	Saber sobre a idade de Helena	15 minutes ago	3
#olá	apenas um olá	a month ago	10
#sobre-voce	Quer saber mais sobre Helena	12 minutes ago	6

Figura 4.3: Listagem de intenções de Helena na plataforma do IBM Watson. Fonte: Elaborado pelo autor.

O exemplo da Figura 4.4, demonstra como o conteúdo é apresentado no Manual do Aluno. Estes textos explicativos também são usados pela ferramenta Helena em suas respostas. Consta como aperfeiçoamento para o próximo trabalho, uma melhor elaboração dos textos de resposta, para que não sejam uma cópia direta do material apresentado pelo Manual, mas sim que contenha a personalidade de Helena, como será apresentado no Capítulo 6.

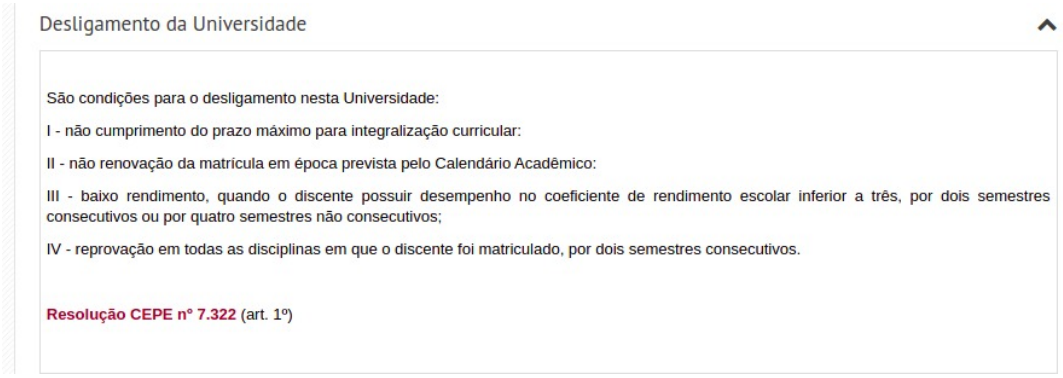


Figura 4.4: Exemplo do Manual do Aluno. Fonte: (*prograd.ufop.br/manual-dao-alunao*)

A comunicação por *email* e *chat* se dão de formas distintas. Emails tendem a ser mais formais, longos e explicativos. Já as mensagens por *chat* são mais breves, informais e com menos riqueza de detalhes, na maioria das vezes. Também, devemos levar em conta de que Helena é um *chatbot*, portanto, é mais provável de que receba apenas dúvidas escritas de forma breve (*"Como cortar minhas horas de ATV?"*), do que receber um contexto junto com a mensagem, como no exemplo de um email sobre ATV: *"...Eu já dei monitoria, estágio,iniciação e participei de eventos."*. Note que esse contexto muitas vezes é desnecessário para a formulação de uma resposta satisfatória e que muitos destes *emails*, por mais que escritos de forma diferente, podem ser resumidos facilmente dentro de uma pergunta e enquadradas dentro de um tópico, assim como demonstrado na Tabela 4.1.

Para treinamento do *chatbot* Helena, é necessário que nós transformemos as mensagens dos alunos em possíveis abordagens de nossos usuários. A partir dessas abordagens podemos extrair uma intenção. A plataforma do Watson recomenda que pelo menos 5 abordagens sejam fornecidas para que uma intenção possa ser treinada de forma correta. Uma tabela com alguns exemplos usados para o treinamento podem ser encontrados no apêndice deste trabalho.

Mensagem do Aluno	Possíveis abordagens	Intenção
Bom dia! Estou com umas ATV's para lançar, referente ao meu estágio. Está sendo possível solicitar esse lançamento? Se sim, preciso entregar os documentos onde? Estou com eles impressos. Atenciosamente .	1) Estou com umas ATV's para lançar, referente ao meu estágio. Está sendo possível solicitar esse lançamento? 2) Está sendo possível solicitar o lançamento de ATVs? 3) Onde preciso de entregar os documentos de ATV?	ATV
Estou para me formar nesse período online(2020.1), porém ainda não dei baixa nas minhas horas ATV, e queria resolver isso o mais rápido possível para que não exista essa pendência na minha formatura. Eu já dei monitoria, estágio, iniciação e participei de eventos. Gostaria de saber como está funcionando para cortar minhas horas ATV durante esse período online .	1) Gostaria de saber como está funcionando para cortar minhas horas ATV durante esse período online? 2) Como cortar minhas horas de ATV. 3) Como dar baixa nas horas de ATV.	ATV
Bom dia! Vi que as horas de ATV foram computadas, porém quando tento protocolar o requerimento de colação de grau extraordinária a seguinte mensagem aparece Estudante não apto para colação de grau. Podem verificar o motivo por favor? .	1) Protocolar o requerimento de colação de grau extraordinária. 2) Porque não estou apto para colação de grau? 3) Colação de grau extraordinária.	Colação de Grau
Em relação ao estágio não obrigatório do curso para integrar as atividades complementares, gostaria de uma orientação para a realização e preenchimento do Termo de Compromisso do Estágio Não Obrigatório e como funciona o Plano de Atividades .	1) Quais são os documentos para estágio? 2) Quais documentos preciso para contrato de estágio? 3) Documentação para contrato de estágio. 4) Como funciona o plano de atividades?	Estágio

Tabela 4.1: Exemplo de base de dados para treinamento. Fonte: COCIC

4.3 Arquitetura da Aplicação

A arquitetura do software Helena é composta de diversos sistemas que serão explicados nas subseções a seguir. Toda infraestrutura da aplicação foi feita utilizando componentes da *Amazon Web Services* (AWS), como apresentado na Figura 4.5, possuindo custo extremamente baixo por conta de seu *free tier*.[3]

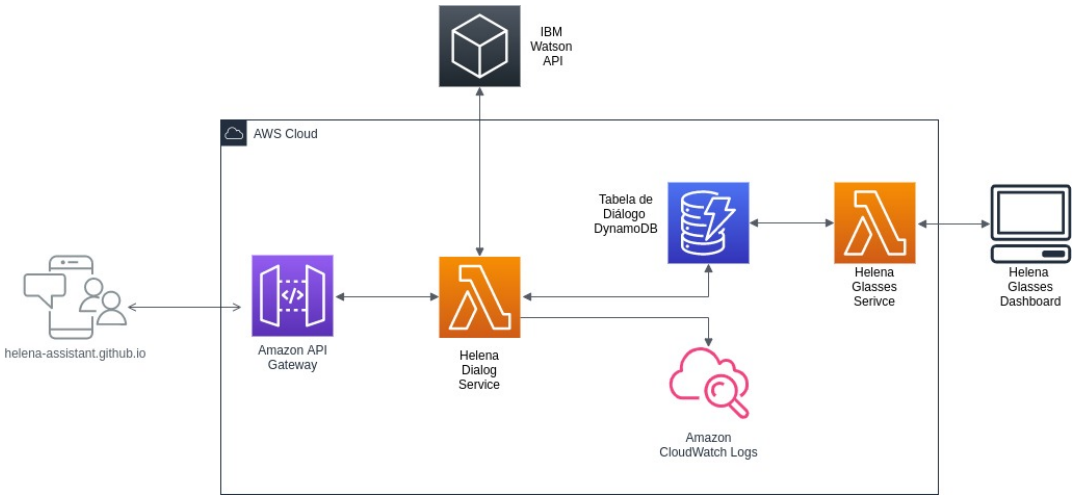


Figura 4.5: Arquitetura de Helena. Fonte: Elaborado pelo autor

4.3.1 Serviços

Os serviços que compõe o *chatbot* Helena usam do modelo *serverless*, que é incorporado através de um serviço da AWS chamado *AWS Lambda*[4]. São as funções lambda que possibilitam que a AWS escale o código de forma automática. Helena possui dois serviços, dos quais serão melhor detalhados nas próximas subseções. Deve se levar em conta que um serviço, no conceito de *serverless*, não é nada mais que um conjunto de funções que são agrupadas dentro de um mesmo contexto, onde os recursos alocados por estas funções são escalados de acordo com a quantidade de requisições que recebem.

4.3.1.1 Serviço de Diálogo

Este serviço é responsável pelo gerenciamento dos diálogos do *chatbot* Helena. Onde temos que o cliente envia uma mensagem para a API e a mesma é encaminhada para ser processada dentro da plataforma do IBM Watson. As a mensagens são armazenadas dentro de um banco de dados na nuvem para que possam ser analisadas em um momento futuro.

[3] https://aws.amazon.com/free/
[4] https://aws.amazon.com/lambda/

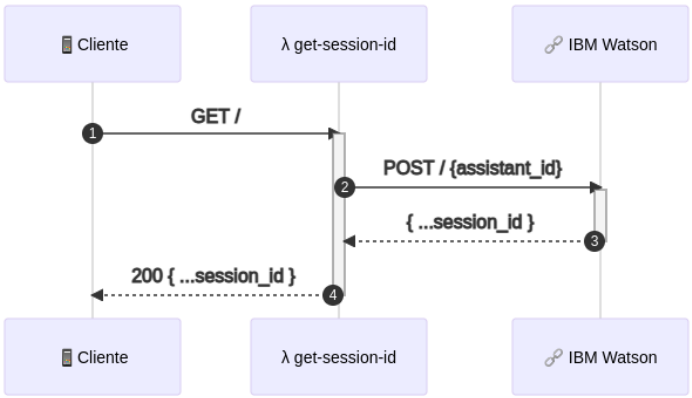


Figura 4.6: Diagrama de sequência para criação de sessão no IBM Watson. Fonte: Elaborado pelo autor.

O primeiro passo para começar a interagir com o *chatbot* Helena é obtendo um *identificador de sessão*, descrito no diagrama da Figura 4.6 como *session-id*. Essa requisição é feita no momento em que o usuário acessa a página web [5]. Como se pode notar através Figura 4.6, o cliente realiza uma requisição *GET* para o Serviço de Diálogo, onde o próprio serviço detém uma chave gerada pela plataforma da IBM chamada de *assistant-id*, que é obrigatória para que o *identificador de sessão* possa ser gerado. O *identificador de sessão* mantém o estado do diálogo e uma sessão persiste até ser excluída ou até atingir o tempo limite, devido à inatividade ou limite máximo do plano, (IBM, 2021). O plano utilizado pelo projeto Helena é o grátis, que possui uma duração máxima de 5 minutos.

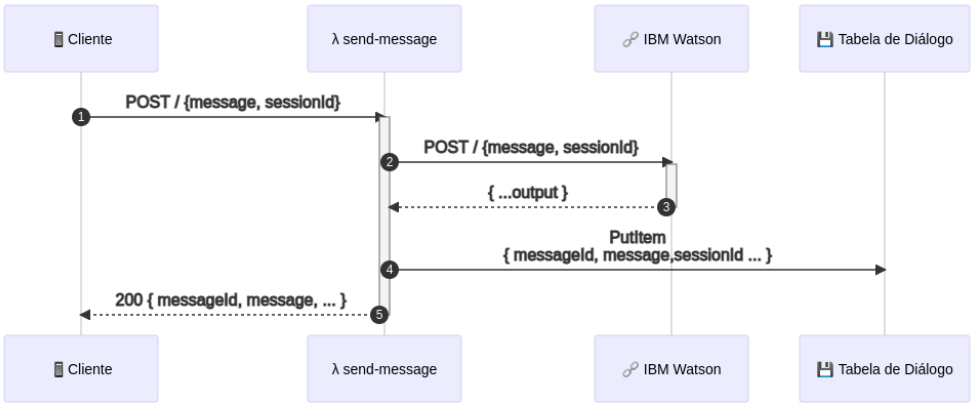


Figura 4.7: Diagrama de sequência para envio de mensagem. Fonte: Elaborado pelo autor.

As mensagens, como exemplificado na Figura 4.7, são enviadas para o Serviço de Diálogo através de uma requisição *POST* contendo no corpo da requisição o *identificador de sessão* e a mensagem a ser enviada. A mensagem é então processada pelo *IBM Watson* e o retorno deste processamento se dá através da seguinte estrutura de *JavaScript Object Notation* (JSON). A

[5] https://helena-assistant.github.io

figura 4.8 é um exemplo básico de resposta de Watson, mas mais exemplos de respostas podem ser encontradas no apêndice deste trabalho.


```
{
  "output": {
    "intents": [
      {
        "intent": "olá",
        "confidence": 1
      }
    ],
    "entities": [],
    "generic": [
      {
        "response_type": "text",
        "text": "Olá! Eu sou a Helena, como posso te ajudar?"
      }
    ]
  },
  "user_id": "c40a3821-ddfc-458c-bbd6-9fab98cbeee1"
}
```


Figura 4.8: Exemplo de resposta da API do IBM Watson Fonte: Elaborado pelo autor.

Depois de processadas por Watson, as mensagens são salvas na tabela de diálogo e a resposta é então encaminhada para o usuário, dando continuidade a conversa.

4.3.1.2 Serviço de Análise (Helena's Glasses)

O nome *Helena's Glasses* (Os óculos de Helena), é uma brincadeira para representar a utilidade deste serviço, pois os óculos fazem com que uma pessoa possa enxergar melhor e Helena não consegue enxergar sem eles. Portanto, este Serviço de Análise será capaz de realizar diversas consultas na tabela de diálogos que possibilitarão aos administradores monitorar o desempenho de Helena de forma rápida e precisa, habilitando-os a **enxergar** aquelas mensagens que não puderam ser respondidas. Daí, temos que estas mensagens servirão poderão ser agregadas na base de treino, para que ela possa se tornar ainda mais inteligente, sendo possível reconhecer intenções em contextos mais complexos.

Figura 4.9: Primeira versão do *Glasses Dashboard*. Fonte: Elaborado pelo autor.

A Figura 4.9 mostra a interface do *dashboard* de monitoramento e nela podemos elencar vários elementos importantes. Esta versão contém: Listagens de todas as mensagens separadas por intenção, relação do número de mensagens que foram respondidas, o gráfico de avaliações por parte dos usuários e o número de vezes que cada intenção foi detectada.

4.3.2 Banco de Dados

Helena usa como banco de dados o *DynamoDB*, um serviço de armazenamento em nuvem de forma escalável providenciado pela *AWS*, (Sivasubramanian, 2012). *DynamoDB* é um banco de dados *NoSQL* (*Not Only SQL*), e isso significa que é capaz de oferecer algumas vantagens sobre o tradicional modelo de banco de dados relacional *SQL*, (Han et al., 2011).

- **Suporte para alta concorrência de leitura e escrita**: É capaz de fazer múltiplas operações de leitura e escrita no banco de forma simultânea, com uma latência extremamente baixa.
- **Sem necessidade de manutenção e baixo custo operacional**: Com um modelo de dados bem mais flexível, auto escalável e com servidores de baixo custo.
- **Alta escalabilidade e disponibilidade**: Em um ambiente de computação, para tornar o sistema altamente disponível, ele precisa ser mantido no ar pelo maior tempo possível. Por outro lado, escalabilidade refere-se à sua capacidade de crescer para suportar o crescimento da demanda e manter seu desempenho.

4.3.3 Monitoramento

O monitoramento de Helena se dá pelo acompanhamento dos *logs*. Os *logs* de uma aplicação devem conter informações relevantes que podem ser disponibilizadas em diferentes níveis de detalhe determinados pela necessidade da investigação de algum evento ocorrido na aplicação. Ou seja, eles têm a função primordial de fazer o registro de todos os eventos importantes que ocorrem durante a execução do código como: Operação na base de dados, erros que acontecem ao tentar executar alguma função no código e quais dados foram enviados ao servidor para que determinada operação possa ser executada.

Helena utiliza o *CloudWatch Logs*[6] para registrar estes eventos, com ele é possível centralizar os *logs* de todos os sistemas, aplicativos e serviços da *AWS* em um único serviço altamente escalonável. Então é capaz visualizá-los facilmente, pesquisar padrões específicos ou por códigos de erro. *CloudWatch Logs* permite que você visualize todos os *logs* (independentemente de sua fonte) como um fluxo consistente de eventos organizados em ordem cronológica.

4.4 Desenvolvimento do Piloto

A Figura 4.10 mostra a árvore de diálogo que conseguimos criar durante o desenvolvimento deste trabalho. O que chamamos árvore de diálogo, na verdade, se assemelha muito à uma árvore de decisão (Quinlan, 1986). A árvore que foi criada até o momento é bastante simples e possui todos os nós na raiz. Nota-se que estes nós estão agrupados em duas pastas: **Dúvidas** e **Personalidade**. A primeira contém todas os tópicos de dúvidas que Helena consegue abordar e a segunda contém possíveis questões que irão contar um pouco sobre sua personalidade como: *"Qual sua idade?"* e *"Quem você é?"*. Não definimos ao certo a personalidade que Helena irá adotar, mas temos certeza de que será sempre alegre, enérgica e educada.

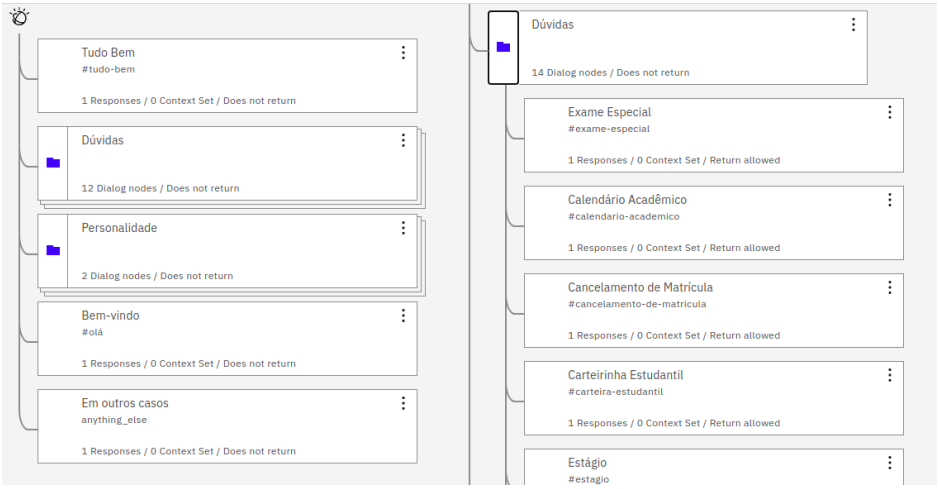


Figura 4.10: Primeira árvore de diálogo de Helena. Fonte: Elaborado pelo autor.

[6]Disponível em: https://docs.aws.amazon.com/pt_br/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html

A Figura 4.11 apresenta dois exemplos de conversas com Helena. O diálogo da esquerda mostra que quando não há uma clareza na mensagem, no que tange as intenções, Helena procura mostrar sugestões ao usuário, para que este possa escolher qual caminho a seguir.Temos com exemplo a frase *"Como aproveitar matérias para abater horas de ATV?"*, a parte de *"Como aproveitar matérias..."* indica a intenção de **Aproveitamento de Disciplinas**, já a parte *"...abater horas ATV?"* indica a intenção de **Abater Horas de ATV**. Também podemos ver pelo diálogo da direita que quando há um clareza maior na mensagem, Helena é capaz de responder corretamente e perguntar por um *feedback* no final. Como também comentado na capítulo de *Trabalhos Futuros* este *feedback* será coletado e armazenado no banco de dados para as próximas análises.

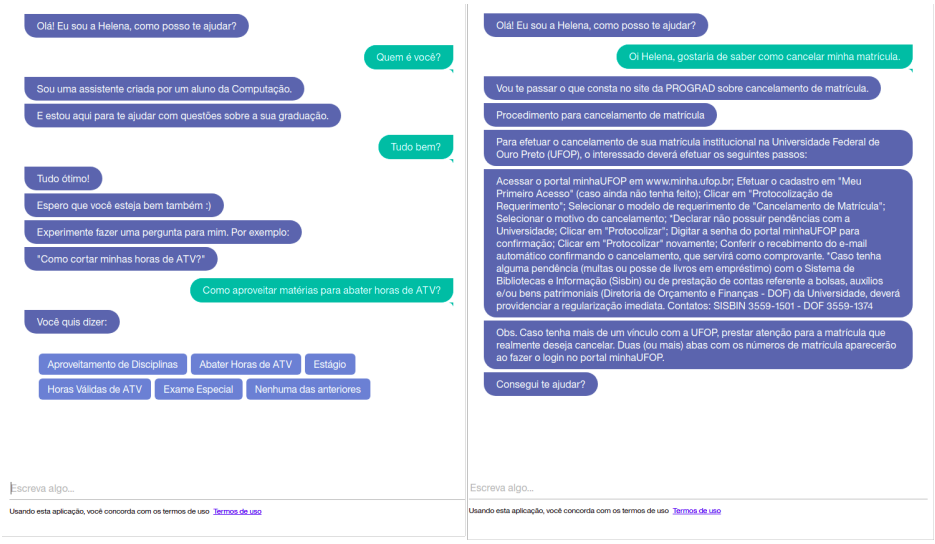


Figura 4.11: Exemplos de conversas com primeira versão de Helena, Fonte: Elaborado pelo autor.

Podemos dizer que esta este piloto contempla todos os aspectos da arquitetura de Helena, como aqueles descritos neste capítulo. Temos um banco de dados não relacional para salvar os dados de diálogo, um serviço de diálogo, que é capaz de gerenciar todos os aspectos da conversa com Helena. Além disso, um serviço integrado com *CloudWatch* para monitoramento que juntamente com o *dashboard* .

Capítulo 5

Experimentos

Este capítulo descreve como foram conduzidos os experimentos deste trabalho. Primeiramente, temos os testes que foram executados para medir a acurácia do *chatbot* Helena. Logo em seguida, detalhamos como foram às duas semanas de piloto do nosso projeto. Durante os experimentos realizados não identificamos os participantes de nenhuma forma e nenhum formulário foi apresentado, todos os dados apresentados a seguir foram coletados usando o *Glasses Dashboard*. Também vale ressaltar que não definimos um roteiro para os testes. Portanto, todas as interações com o *chatbot* Helena ocorreram de forma espontânea.

5.1 Adaptações no site

Para que pudéssemos apresentar o projeto para o primeiro público de testes foram necessárias algumas mudanças. A primeira delas foi o *layout* do site de uma forma geral e acrescentamos um texto explicativo e uma lista todas as intenções que Helena consegue reconhecer, como se pode observar na Figura 5.1.

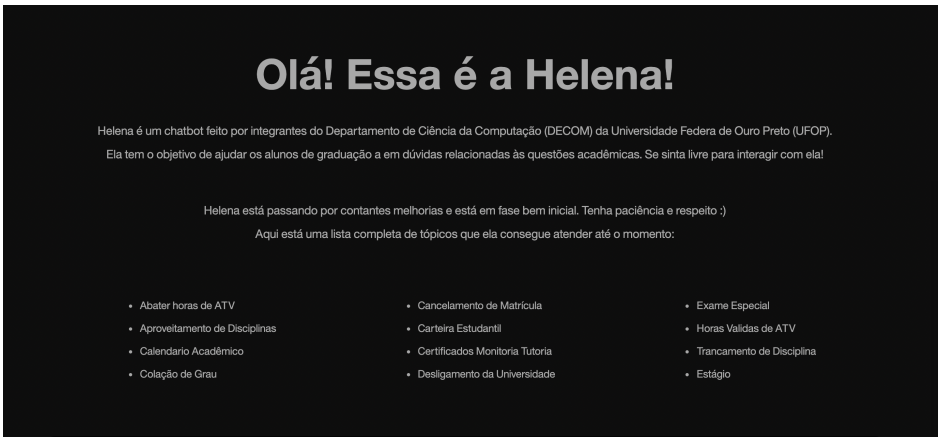


Figura 5.1: Textos explicativos adicionados no *layout* do site Fonte: Elaborado pelo autor

Outra alteração muito importante foi o acréscimo de um sistema que tornasse possível a coleta de opinião totalmente anônima. Com isso, a avaliação de 1 a 5 estrelas, como apresentado na Figura 5.2, pode ser funcionar como um termômetro no sentido de como o usuário avalia a experiência no geral.

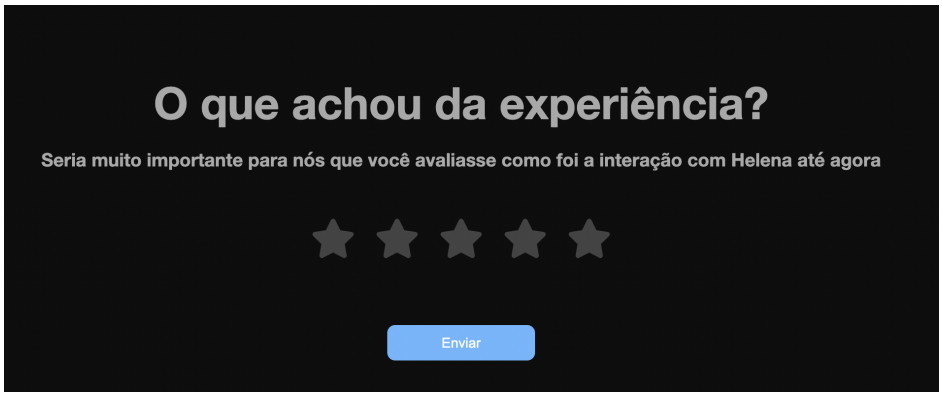


Figura 5.2: Sistema simples para avaliação de experiência. Fonte: Elaborado pelo autor

5.2 Testes de acurácia

Para realizar os testes de acurácia criamos um pequeno projeto chamado *Gym* [1] que contém vários *scripts* que foram utilizados para realização dos testes apresentados a seguir. Os *scripts* enviam as mensagens diretamente pela API do IBM Watson e de forma alguma passam pela API do *chatbot* Helena. Isso evita que os resultados obtidos pelos testes de acurácia entrem em conflito com aqueles apresentados durante as semanas de piloto. A partir da resposta vinda de Watson, verificamos se o resultado esperado é retornado. *Gym* salva automaticamente os resultados para podermos manter um histórico das medições realizadas.

Para apresentarmos os resultados deste capítulo utilizamos a Tabela 5.1 que realiza o mapeamento das intenções para respectivos códigos. A utilização de códigos permite uma melhor visualização dos resultados apresentados nos gráficos.

INTENÇÃO	CÓDIGO
Abater Horas de ATV	01
Aproveitamento de Disciplinas	02
Calendário Acadêmico	03
Cancelamento de Matrícula	04
Carteira Estudantil	05
Certificado de Monitoria e Tutoria	06
Colação de Grau	07
Desligamento da Universidade	08
Estágio	09
Exame Especial	10
Horas Válidas de ATV	11
Trancamento de Disciplina	12

Tabela 5.1: Mapeamento de intenções Fonte: Elaborado pelo autor

[1] https://github.com/helena-assistant/gym

Para realizar a primeira medição, separamos as 12 intenções mais relevantes e elaboramos 20 exemplos para cada uma delas. Alguns exemplos continham vários erros ortográficos e linguagem informal para podermos deixar o teste mais próximo possível de um cenário "real".

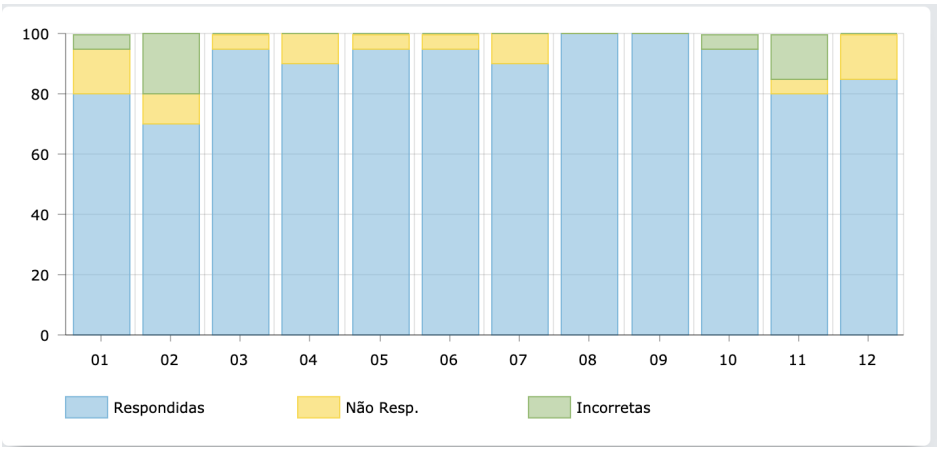


Figura 5.3: Resultados obtidos no primeiro teste de acurácia. Fonte: Elaborado pelo autor

Observando a Figura 5.3, temos que as intenções que obtiveram a maior porcentagem de mensagens respondidas foram *estágio* e *desligamento da universidade*. A que obteve a maior porcentagem de repostas incorretas foi *aproveitamento de disciplinas* e as que obtiveram o maior número de mensagens não respondidas foram *abater horas de ATV* e *trancamento de disciplina*

A segunda medição se deu de forma mais elaborada. Neste teste foram usadas as 20 mensagens para cada intenção e foram ainda acrescentadas 10 das quais esperamos que fossem respondidas de forma incorreta, totalizando 30 mensagens para cada intenção.

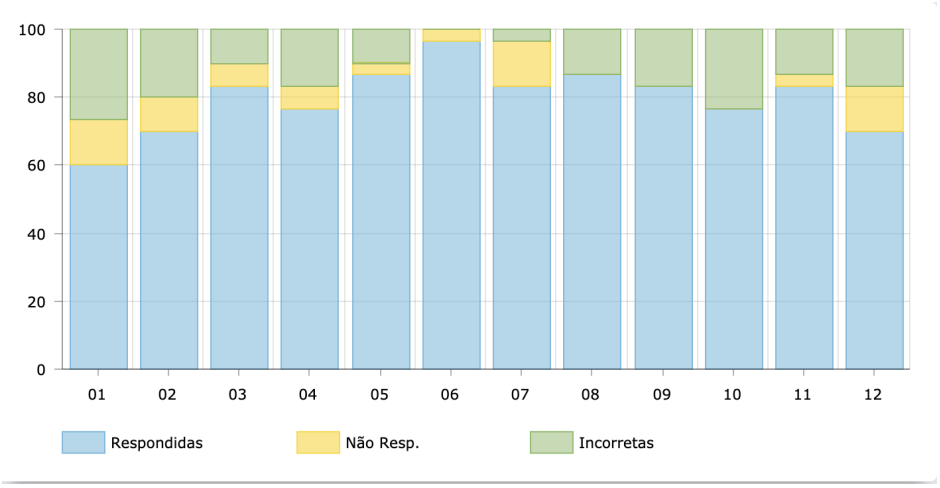
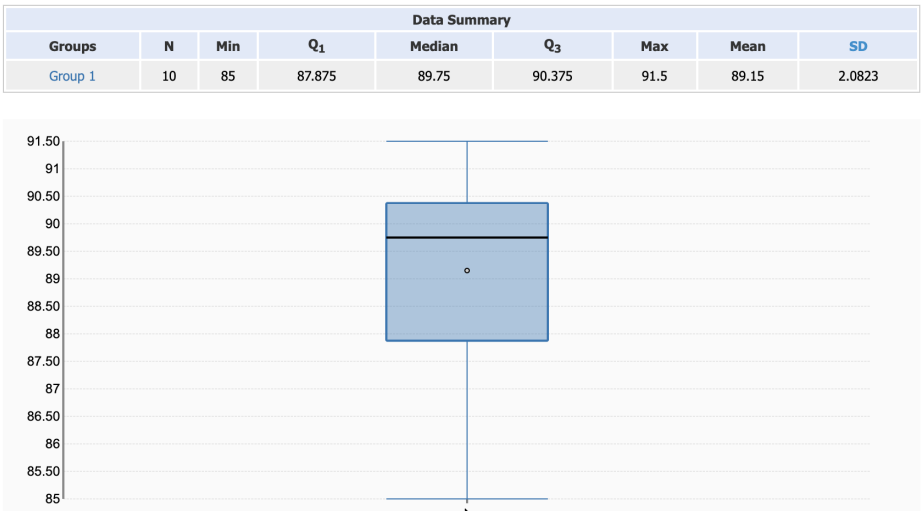


Figura 5.4: Resultados obtidos no segundo teste de acurácia. Fonte: Elaborado pelo autor

Observando os dados da Figura 5.4 percebemos que a maioria das intenções sofreu com uma

grande perda na acurácia. A intenção com a menor porcentagem de mensagens interpretadas corretamente foi *abater horas ATV* com apenas 60%.

Para analisar a robustez do *chatbot* Helena, adotamos uma estratégia de amostragem e análise de acurácia. Para isso, utilizamos a mesma base do primeiro teste de acurácia que contém cerca 240 mensagens distribuídas igualmente por 12 intenções. Destas 240 mensagens, foram selecionadas 200 de forma aleatória. O teste foi realizado 10 vezes e os resultados estão demonstrados na Figura 5.5.



Data Summary

Groups	N	Min	Q_1	Median	Q_3	Max	Mean	SD
Group 1	10	85	87.875	89.75	90.375	91.5	89.15	2.0823

Figura 5.5: Resultados obtidos no segundo teste de acurácia com randomização de mensagens. Fonte: Elaborado pelo autor

O último teste adota a mesma estratégia do anterior, porém com uma base de dados diferente. A base utilizada é mesma do segundo teste, contendo 360 mensagens. Das quais, esperamos de que 240 mensagens sejam interpretadas de forma correta e 120 de forma incorreta. Do total de mensagens, 200 eram selecionadas de forma aleatória a cada teste. Os resultados obtidos estão na Figura 5.6

Data Summary								
Groups	N	Min	Q_1	Median	Q_3	Max	Mean	SD
Group 1	10	78	79.75	81	84	85	81.6	2.4244

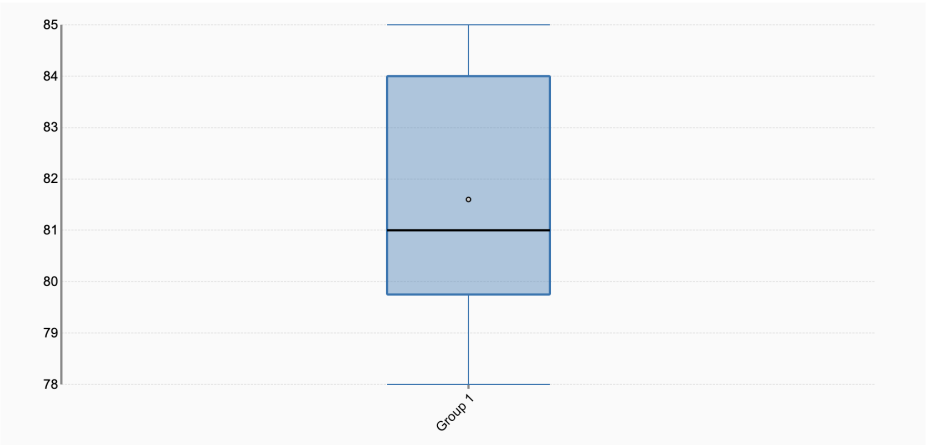


Figura 5.6: Resultados obtidos no segundo teste de acurácia com randomização de mensagens. Fonte: Elaborado pelo autor

Comparando os resultados obtidos nas Figuras 5.5 e 5.6, podemos concluir que o *chatbot* Helena apresenta um desempenho melhor quando a base de dados contém apenas casos de testes dos quais é esperado que Helena acerte o contexto. Quando esperamos que Helena erre a interpretação das mensagens, os resultados decaem de uma forma considerável. Ainda assim, consideramos importante uma acurácia média de 81% neste cenário com dados ruidosos.

A queda na performance apresentada acima pode ser resolvida por meio do treinamento do *chatbot* Helena com mensagens do tipo, *"como posso fazer para trancar minha matrícula"*, e não tão abrangentes (*"como posso trancar"*). Pois, desta forma é possível evitar que Helena interprete de forma errônea uma determinada mensagem.

5.3 Primeira semana de piloto

A primeira semana de testes com Helena envolveram apenas 5 alunos do curso de Ciências da Computação de deferentes etapas do curso. Esta primeira semana tinha como objetivo coletar alguns dados iniciais sobre o projeto e implantar as melhorias necessárias para podermos expandir número de participantes na segunda semana.

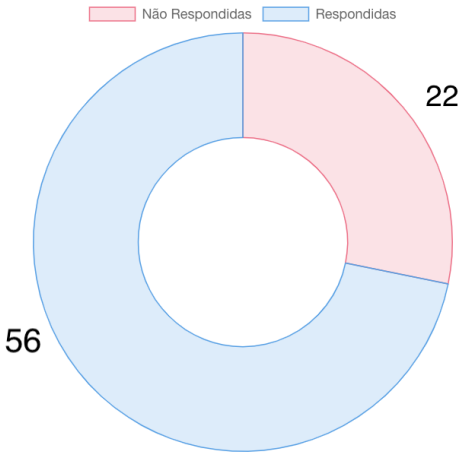


Figura 5.7: Número de mensagens respondidas/não respondidas durante a primeira semana de piloto Fonte: Elaborado pelo autor

Observando a Figura 5.7 temos que das 78 mensagens enviadas à Helena, apenas 56 foram assimiladas. Deve se notar que dessas 56 que foram compreendidas, não necessariamente foram respondidas de forma correta.

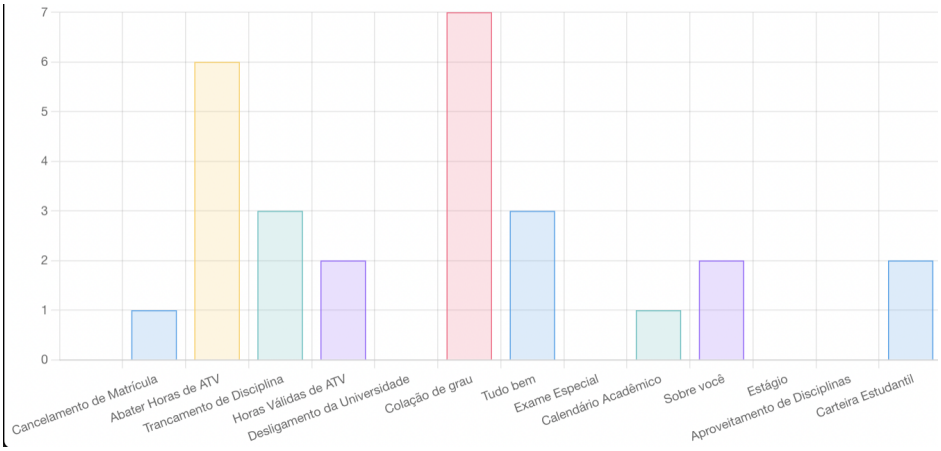


Figura 5.8: Mensagens enviadas durante a primeira semana categorizadas em intenções Fonte: Elaborado pelo autor

Na Figura 5.8, as intenções que tiveram a maior procura foram *abater horas de atv* e *colação de grau*. E algumas intenções não obtiveram nenhuma procura como *desligamento da universidade* e *exame especial*.

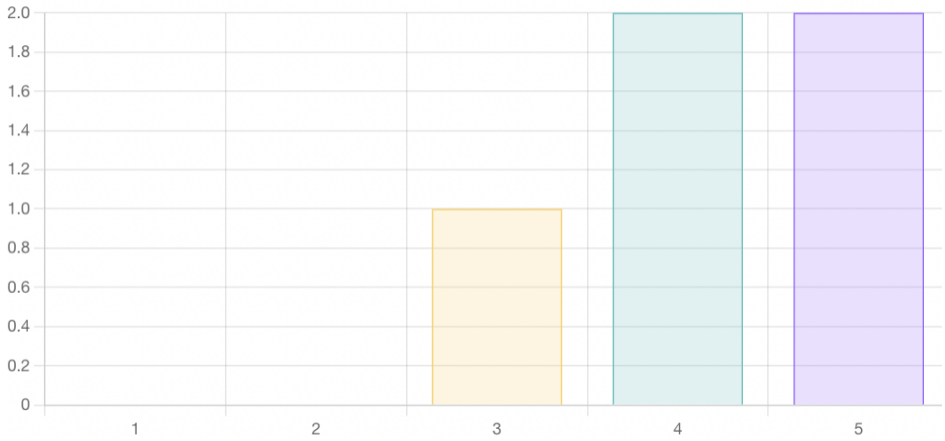


Figura 5.9: Avaliações de experiência durante a primeira semana de piloto Fonte: Elaborado pelo autor

A Figura 5.9 mostra como os participantes avaliaram a experiência de conversação com Helena em uma escala de 1 a 5 estrelas. O desempenho médio obtido foi de 4,2 estrelas. Essa avaliação é feita no próprio site[2]

5.4 Segunda semana de piloto

A segunda semana de piloto se deu logo após algumas correções relacionadas a primeira semana e também foi realizado com um número maior de pessoas, 15 no total. O *Glasses Dashboard* possibilitou o acesso às mensagens que não foram respondidas. Desta forma conseguimos realizar um novo treinamento do *chatbot* Helena para a segunda semana.

[2]Disponível em: https://helena-assistant.github.io/.

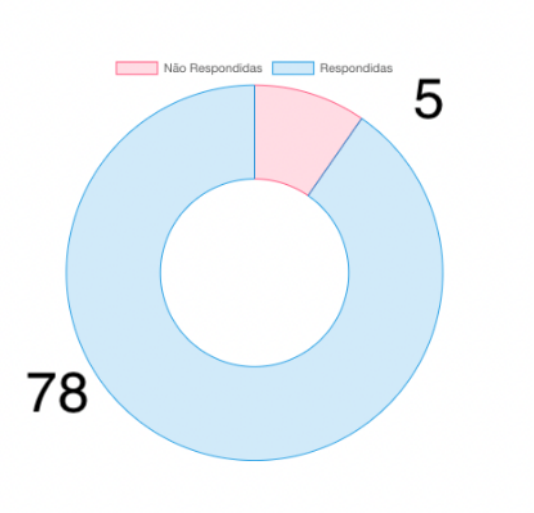


Figura 5.10: Número de mensagens respondidas/não respondidas durante a segunda semana de piloto Fonte: Elaborado pelo autor

A Figura 5.10 já evidencia uma grande melhora na quantidade de mensagens que foram respondidas. Apenas 6% das mensagens enviadas não obtiveram nenhuma resposta

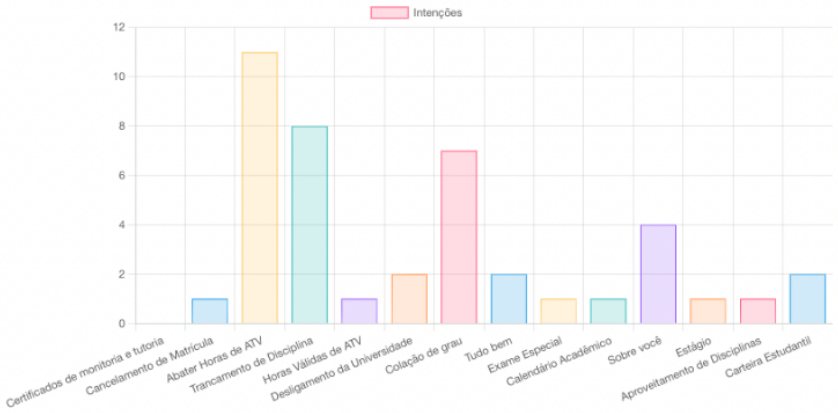


Figura 5.11: Mensagens enviadas durante a primeira semana categorizadas em intenções Fonte: Elaborado pelo autor

Conforme a Figura 5.11 a intenção que obteve a maior procura por parte dos estudantes foi *como abater horas de ATV* e questões sobre *certificados de monitoria e tutorial* não tiveram nenhuma procura.

Figura 5.12: Avaliações de experiência durante a primeira semana de piloto Fonte: Elaborado pelo autor

Pode se observar pela Figura 5.12 as avaliações também foram melhoras do que as da primeira semana, sendo 4,8 a nota média dos usuários.

Capítulo 6

Conclusão

Pelos resultados obtidos no presente trabalho, podemos dizer que o *chatbot* Helena obteve resultados satisfatórios, tanto no quesito de experiência no uso da plataforma quanto na acurácia durante os testes realizados. Até o momento da escrita deste trabalho o projeto é composto por *website*, *dashboard*, API e *scripts* para testes do chatbot, sem nenhum custo de infraestrutura. Todos esses elementos evidenciam um certo nível de maturidade do projeto, possibilitando que seja disponibilizado para o público. Portanto, acreditamos certamente que Helena consegue resolver dúvidas reais, que fazem parte do dia a dia dos discentes da UFOP

6.1 Trabalhos Futuros

Podemos dizer que o grande foco deste trabalho esteve na criação de uma arquitetura robusta e de baixo custo por onde seja possível validar, monitorar e divulgar o projeto. Com isso concluído, como demonstrado nos capítulos anteriores, acreditamos que podemos seguir as próximas etapas que envolvem diversas melhorias estruturais no diálogo.

- **Criar um sistema próprio para processamento de linguagem natural**: Substituindo o Watson, teremos mais controle sobre o que acontece no escopo da nossa arquitetura. Também teremos uma tecnologia feita na Universidade, que irá contribuir ainda mais com o desenvolvimento de Helena, que muito provavelmente irá atingir resultados ainda melhores do que os obtidos nesse trabalho.
- **Melhorias no diálogo e expansão da base de treino**: Iremos coletar mais exemplos e abordar um número maior intenções, deixando diálogos mais fluídos e também possibilitando que Helena consiga resolver uma quantidade maior de problemas.

Com os tópicos elencados acima implementados, pretendemos oferecer uma experiência ainda melhor ao conversar com Helena. Desta forma, é possível que ela se torne mais uma ponte de comunicação entre os estudantes de graduação e a Universidade, ajudando estudantes a resolverem problemas de seu cotidiano estudantil.

Apêndice A

Material Complementar

Aqui estão listados alguns *links* para materiais complementares para a apreciação deste trabalho.

- Repositório contendo todo código do projeto: Link
- Protótipo apresentado: Link
- Dashboard: Link
- Base de treino usada: Link

Referências Bibliográficas

Chen, Y.; Argentinis, J. E. e Weber, G. (2016). Ibm watson: how cognitive computing can be applied to big data challenges in life sciences research. *Clinical therapeutics*, 38(4):688–701.

da Silva Oliveira, J.; Espíndola, D. B.; Barwaldt, R.; Ribeiro, L. M. e Pias, M. (2019). Ibm watson application as faq assistant about moodle. In *2019 IEEE Frontiers in Education Conference (FIE)*, pp. 1–8. IEEE.

Dibitonto, M.; Leszczynska, K.; Tazzi, F. e Medaglia, C. M. (2018). Chatbot in a campus environment: design of lisa, a virtual assistant to help students in their university life. In *International Conference on Human-Computer Interaction*, pp. 103–116. Springer.

Fielding, R. T. (2000). Representational state transfer (rest). Accessed: 2021-03-27.

Goel, A. K. e Polepeddi, L. (2016). Jill watson: A virtual teaching assistant for online education. Technical report, Georgia Institute of Technology.

Goyal, P.; Pandey, S. e Jain, K. (2018). Developing a chatbot. In *Deep Learning for Natural Language Processing*, pp. 169–229. Springer.

Gupta, A.; Tyagi, S.; Panwar, N.; Sachdeva, S. e Saxena, U. (2017). Nosql databases: Critical analysis and comparison. In *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, pp. 293–299. IEEE.

Hamilton, D.; Lane, J. V.; Gaston, P.; Patton, J.; Macdonald, D.; Simpson, A. e Howie, C. (2014). Assessing treatment outcomes using a single question: the net promoter score. *The bone & joint journal*, 96(5):622–628.

Han, J.; Haihong, E.; Le, G. e Du, J. (2011). Survey on nosql database. In *2011 6th international conference on pervasive computing and applications*, pp. 363–366. IEEE.

High, R. (2012). The era of cognitive systems: An inside look at ibm watson and how it works. *IBM Corporation, Redbooks*, pp. 1–16.

Hollands, F. e Tirthali, D. (2014). Moocs: Expectations and reality: Full report, center for benefit-cost studies of education. teachers college, columbia university (usa).

Hubspire (2019). https://www.hubspire.com/resources/general/application-programming-interface/. Accessed: 2021-03-27.

IBM, W. (2021). Ibm watson assistant docs. Accessed: 2021-03-27.

Janarthanam, S. (2017). Development: Build chatbots and voice user interfaces with chatfuel. *Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills, Birmingham: Packt Publishing Ltd.*

Kane, D. A. (2016). The role of chatbots in teaching and learning. *E-learning and the academic library: Essays on innovative initiatives*, 131.

Khasawneh, T. N.; AL-Sahlee, M. H. e Safia, A. A. (2020). Sql, newsql, and nosql databases: A comparative survey. In *2020 11th International Conference on Information and Communication Systems (ICICS)*, pp. 013–021. IEEE.

Knorr, E. e Gruman, G. (2008). What cloud computing really means. *InfoWorld*, 7(20-20):1–17.

Lane, K. (2019). Intro to apis: History of apis. Accessed: 2021-03-27.

Marcus, D. A. (2020). *Graph theory*, volume 53. American Mathematical Soc.

MDN (2020). Http request methods. Accessed: 2021-03-27.

Mell, P.; Grance, T. et al. (2011). The nist definition of cloud computing.

Murtaza, S. S.; Lak, P.; Bener, A. e Pischdotchian, A. (2016). How to effectively train ibm watson: Classroom experience. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pp. 1663–1670. IEEE.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Rajan, R. A. P. (2018). Serverless architecture-a revolution in cloud computing. In *2018 Tenth International Conference on Advanced Computing (ICoAC)*, pp. 88–93. IEEE.

Rocha, S. S. D.; Joye, C. R. e Moreira, M. M. (2020). A educação a distância na era digital: Tipologia, variações, uso e possibilidades da educação online. *Research, Society and Development*, 9(6):e10963390–e10963390.

Shah, D. (2016). By the numbers: Moocs in 2016. Accessed: 2021-03-27.

Shum, H.-Y.; He, X.-d. e Li, D. (2018). From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26.

Sivasubramanian, S. (2012). Amazon dynamodb: a seamlessly scalable non-relational database service. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 729–730.

Strupp, H. H. (1955). An objective comparison of rogerian and psychoanalytic techniques. *Journal of Consulting Psychology*, 19(1):1.

Sun, L.; Tang, Y. e Zuo, W. (2020). Coronavirus pushes education online. *Nature Materials*, 19(6):687–687.

Telang, P. R.; Kalia, A. K.; Vukovic, M.; Pandita, R. e Singh, M. P. (2018). A conceptual framework for engineering chatbots. *IEEE Internet Computing*, 22(6):54–59.

Wallace, R. (2003). The elements of aiml style. *Alice AI Foundation*, 139.

Wallace, R. S. (2009). The anatomy of alice. In *Parsing the Turing Test*, pp. 181–210. Springer.

Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

ZEMČÍK, M. T. (2019). A brief history of chatbots. *DEStech Transactions on Computer Science and Engineering*, (aicae).

Zhao, J.; Jing, S. e Jiang, L. (2018). Management of api gateway based on micro-service architecture. In *Journal of Physics: Conference Series*, volume 1087, p. 032032. IOP Publishing.

Zhou, L.; Gao, J.; Li, D. e Shum, H.-Y. (2020). The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93.