INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS CENTRO
CURSO SUPERIOR DE TECNOLOGIA EM DESENVOLVIMENTO DE SISTEMAS

GUSTAVO DE ALMEIDA COSTA

USO DE INTERFACE CONVERSACIONAL PARA APRIMORAR A INTERAÇÃO ALUNO-BIBLIOTECA.

Manaus, Amazonas - Brasil
2023

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS CENTRO
CURSO SUPERIOR DE TECNOLOGIA EM DESENVOLVIMENTO DE SISTEMAS

GUSTAVO DE ALMEIDA COSTA

USO DE INTERFACE CONVERSACIONAL PARA APRIMORAR A INTERAÇÃO ALUNO-BIBLIOTECA.

Trabalho de Conclusão de Curso apresentado à banca examinadora do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Amazonas – IFAM Campus Manaus – Centro, como requisito para o cumprimento da disciplina TCC II – Projeto de Software.

Orientador: Prof. Dr. Renildo Viana Azevedo

Manaus, Amazonas - Brasil
2023

Biblioteca do IFAM – Campus Manaus Centro

C837u Costa, Gustavo de Almeida.
Uso de interface conversacional para aprimorar a interação aluno-biblioteca / Gustavo de Almeida Costa. – Manaus, 2023.
114 p. : il. color.

Monografia (Tecnologia em Análise e Desenvolvimento de Sistemas). – Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Centro, 2023.
Orientador: Prof. Dr. Renildo Viana Azevedo.

1. Desenvolvimento de sistemas. 2. Chatbot. 3. Biblioteca. I. Azevedo, Renildo Viana. (Orient.) II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 005.3

Elaborada por Márcia Auzier CRB 11/597

GUSTAVO DE ALMEIDA COSTA

USO DE INTERFACE CONVERSACIONAL PARA APRIMORAR A INTERAÇÃO ALUNO-BIBLIOTECA.

Trabalho de Conclusão de Curso apresentado à banca examinadora Curso Superior de Tecnologia de Desenvolvimento de Análise e Desenvolvimento de Software do Instituto Federal de Educação, Ciências e Tecnologia do Amazonas – IFAM Campus Manaus - Centro, como requisito para o cumprimento da disciplina TCC II – Projeto de Software
Orientador: Prof. Dr. Renildo Viana Azevedo

Aprovado em _____ de _____ de 2023

BANCA EXAMINADORA

Prof. Dr. Renildo Viana Azevedo (Orientador)
Instituto Federal do Amazonas

Prof. MSc. Antonio Ferreira dos Santos Junior
Instituto Federal do Amazonas

Prof. MSc. Vinicius Oliveira Barra
Instituto Federal do Amazonas

RESUMO

Este trabalho aborda o desenvolvimento de uma interface conversacional, um chatbot inteligente destinado a facilitar a interação entre os alunos e o ecossistema de uma biblioteca. O projeto surgiu da necessidade de proporcionar um meio eficiente e intuitivo para os alunos acessarem informações e serviços da biblioteca, como consulta de disponibilidade de livros, visualização de reservas existentes e realização de novas reservas. Para alcançar uma interação natural e eficaz, foi empregada a tecnologia de Processamento de Linguagem Natural (PLN), um ramo da inteligência artificial que busca compreender e interpretar a linguagem humana. Através do uso de PLN, o chatbot é capaz de entender as requisições dos usuários e fornecer respostas precisas, criando uma experiência de atendimento amigável e eficiente. Este trabalho visa demonstrar como soluções tecnológicas, como chatbot inteligentes, podem ser desenvolvidas e implementadas para melhorar a eficiência dos serviços de biblioteca, contribuindo para uma experiência acadêmica mais enriquecedora para os alunos.

Palavras Chaves: Processamento de Linguagem Natural; Chatbot; Biblioteca;

ABSTRACT

This work addresses the development of a conversational interface, an intelligent chatbot designed to facilitate interaction between students and the ecosystem of a library. The project arose from the need to provide an efficient and intuitive means for students to access information and services of the library, such as checking the availability of books, viewing existing reservations, and making new reservations. To achieve natural and effective interaction, Natural Language Processing (NLP) technology was employed, a branch of artificial intelligence that seeks to understand and interpret human language. Using NLP, the chatbot is capable of understanding user requests and providing precise responses, creating a friendly and efficient customer service experience. This work aims to demonstrate how technological solutions, such as intelligent chatbots, can be developed and implemented to improve the efficiency of library services, contributing to a more enriching academic experience for students.

Keywords: *Natural Language Processing*; *Chatbot*; *Library*.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

PLN	Processamento de Linguagem Natural
IA	Inteligência Artificial
IUC	Interfaces de usuário conversacionais
IAG	Inteligência Artificial Geral
ML	*Machine Learning*
POO	Programação orientada a objetos
SRP	*Single Responsibility Principle*
LSP	*Liskov Substitution Principle*
ISP	*Interface Segregation Principle*
JSON	*JavaScript Object Notation*

INTRODUÇÃO

A evolução tecnológica tem proporcionado avanços significativos na maneira como interagimos com sistemas digitais. Um dos avanços mais notáveis é a emergência de interfaces conversacionais, que estão redefinindo a interação homem-máquina, proporcionando uma comunicação mais natural e intuitiva. Interfaces conversacionais são sistemas que permitem a interação com o usuário em linguagem natural, seja por texto ou voz, simulando uma conversa humana. Essa inovação é alimentada principalmente pelo rápido desenvolvimento no campo da Inteligência Artificial (IA), especialmente no Processamento de Linguagem Natural (PLN).

As interfaces de usuário conversacionais (IUC), como *chatbots* e assistentes de voz, estão se tornando cada vez mais comuns em áreas do dia a dia, e espera-se que se tornem ainda mais onipresentes no futuro. Essas interfaces estão sendo projetadas para interações cada vez mais complexas, e parecem ter o potencial de beneficiar pessoas com deficiências para interagir através da web e com tecnologias embutidas no ambiente. (LISTER et al., 2020, p. 11).

Um dos benefícios das IUC é a emulação de uma conversa natural, dessa forma são empregadas ferramentas para melhorar essa naturalidade, muitas IUCs empregam ferramentas de PLN com o objetivo de interpretar os objetivos de um usuário que está usando o sistema conversacional, possibilitando assim um melhor auxílio dentro do fluxo de um sistema. O Processamento de Linguagem Natural (PLN) consiste em uma subárea da inteligência artificial e refere-se, de forma ampla, ao estudo e desenvolvimento de sistemas computacionais que podem interpretar a fala e o texto conforme naturalmente os seres humanos falam e o digitam, além de desenvolver melhor a forma de interpretação da linguagem humana em diferentes dispositivos (RODRÍGUEZ; BEZERRA, 2020, p. 1).

Nessa perspectiva, percebeu-se a oportunidade de IUC para ganho na interatividade do aluno com uma biblioteca, onde este *chatbot* poderia ter acesso ao contexto da biblioteca, sabendo estoque dos livros, permitir que alunos busquem

livros somente por uma descrição não assertiva, detalhes de alunos que estão atrasados na devolução de livros, oferecendo uma resposta completa ao aluno sobre o status de sua reserva, a fim de diminuir atrasos e melhorar o fluxo de reservas da biblioteca.

Nos próximos capítulos será apresentado a descrição dos conceitos que formam a base deste projeto. O Processamento de Linguagem Natural (PLN) e outros fundamentos de inteligência artificial que vão dar a dinâmica para nosso chatbot, para responder de forma mais humana e assim atender os conceitos de interfaces conversacionais.

Neste trabalho será apresentado os conceitos básicos sobre *chatbots*, um pouco da história dos primeiros *bots* e alguns dos componentes mais usados. Também será falado sobre os conceitos mais técnicos como base de dados que este gravará nossos dados e sobre a comunicação em tempo real que permitirá ao nosso chatbot ter uma conversação mais fluida e em tempo real. Em seguida, será focado nas tecnologias específicas que dão vida ao projeto. Este detalhando as tecnologias escolhidas, analisando como cada uma contribui para a criação de uma interface conversacional eficaz e intuitiva. Vamos discutir não só a funcionalidade destas tecnologias, mas também como elas se integram para formar um sistema coeso.

Por fim, a proposta do projeto, esta parte é dedicada a descrever a implementação prática do chatbot, serão discutidos aspectos da camada de interface para o usuário (*front-end*), o serviço central de gerenciamento do bot (*back-end*), detalhes de implementação assim como a modelagem dele.

O objetivo é desenvolver um chatbot com as funcionalidades mais comuns para alunos de uma biblioteca, incluindo consultar um livro, reservar um livro e verificar reservas realizadas. Este projeto visa proporcionar uma melhor experiência aos alunos que utilizam os serviços de uma biblioteca. Espera-se melhorar a eficiência na gestão dos materiais, reduzir o tempo gasto na procura de livros e facilitar a gestão das reservas por parte dos alunos.

Justificativa

A partir do contexto exposto na seção anterior, torna-se evidente que a aplicação de IUC em conjunto com as tecnologias de PLN podem atuar como facilitadores significativos para a utilização das bibliotecas. Essa integração tecnológica visa mitigar a incidência de atrasos na devolução de livros, proporcionando uma gestão mais eficaz dos estoques de materiais. Além disso, oferece uma interface amigável e intuitiva que auxilia os alunos na busca por livros, na consulta de reservas realizadas, permitindo o acesso a informações cruciais como a data de solicitação, o status da reserva, a data de devolução, bem como fornecer um histórico detalhado aos usuários sobre os livros que já foram reservados e a disponibilidade para novas reservas.

A implementação de um sistema conversacional inteligente não apenas simplifica o acesso a recursos bibliotecários, mas também promove uma interação mais eficaz e satisfatória entre os alunos e a biblioteca. Ao automatizar e otimizar esses processos, espera-se não apenas melhorar a eficiência operacional da biblioteca, mas também enriquecer a experiência educacional dos alunos, proporcionando-lhes acesso fácil e rápido a recursos de aprendizagem essenciais. Além disso, o projeto proposto pode servir como um modelo inovador para outras instituições educacionais que buscam modernizar suas operações bibliotecárias e melhorar a interatividade com os alunos através da adoção de tecnologias de IA e interfaces conversacionais.

Objetivos

A seguir são definidos o objetivo geral e os objetivos específicos estabelecidos para o presente trabalho.

Objetivo Geral

Desenvolver um protótipo de chatbot para interação do aluno com as funcionalidades de uma biblioteca comum.

Objetivos Específicos

- Implementar módulo de consulta de Livro
- Implementar módulo de reserva de livro
- Implementar módulo de consultar minhas reservas de livros
- Implementar módulo de autenticação ao chatbot

Metodologia

Conforme os objetivos estabelecidos, a execução do trabalho se deu a partir do cumprimento da seguinte metodologia:

- Levantamento bibliográfico que inclui arquivos e pesquisas referentes ao tema, assim como trabalhos relacionados.

- Análise dos trabalhos correlatos como forma de entender como se deu o desenvolvimento deles, a fim de buscar técnicas que poderão ser utilizadas neste trabalho.

- Realização de experimentação com ferramentas de PLN, a fim de escolher qual ferramentas mais se aplica aos propósitos da proposta deste trabalho.

- Implementação da aplicação proposta no trabalho, com base nas técnicas e ferramentas escolhidas.

- Desenvolvimento da aplicação utilizando a metodologia de entrega incremental, onde um pedaço pequeno da aplicação é entregue durante um ciclo de desenvolvimento, este ciclo podendo ser uma semana ou mais.

CAPÍTULO 1: AS INTERFACES DE USUÁRIOS CONVERSACIONAIS (IUCs)

A interação entre computador e pessoas falam em diferentes linguagens, pessoas utilizam palavras e frases, enquanto computadores se comunicam mais em um e zeros (MITREVSKI, 2018).

De acordo com Mitrevski (2018) essa lacuna entre a comunicação na tradução entre essas duas partes é preenchida por um mediador chamado *graphical user interfaces* (GUIs)

Um dos desafios enfrentados por esta tecnologia é o reconhecimento de voz e o entendimento da linguagem natural ou processamento de linguagem natural, onde uma palavra como “não” pode mudar totalmente o sentido de uma frase (MITREVSKI, 2018).

Sobre a evolução das GUI para as interfaces de usuários conversacionais (IUC) Janarthanam (2017), com os recentes desenvolvimentos em IA e as crescentes restrições, como o tamanho cada vez menor dos dispositivos, as interfaces de usuário conversacionais estão novamente se tornando uma realidade.

A base tecnológica das IUCs modernas inclui o processamento de linguagem natural (PLN), aprendizado de máquina, reconhecimento de fala e síntese de voz. Estas tecnologias permitem às IUCs não apenas compreender e processar a linguagem humana, mas também responder de maneira coerente e contextualizada. O uso de algoritmos avançados e bases de dados contribui para a capacidade dessas interfaces de aprender e se adaptar com o tempo, melhorando continuamente a qualidade das interações.

De acordo com Becker Jacintho e Silva Penha (2016), "Estas aplicações já existem há algum tempo, com os avanços de tecnologias, como a inteligência

artificial, tecnologias da linguagem, de dispositivos móveis, permitiram o desenvolvimento dessas interfaces e a realização de tarefas mais complexas por meio delas."

Interfaces de Usuários Conversacionais (IUCs) são tecnologias na interação humano-computador que permitem a comunicação entre usuários e sistemas por meio da linguagem natural. Essas interfaces evoluíram consideravelmente graças ao progresso em campos como a inteligência artificial e o aprendizado de máquina.

Um dos conceitos mais importantes de interfaces conversacionais é uma intenção, que é basicamente o que o usuário quer do sistema para fazer. (MITREVSKI, 2018).

De acordo com Janarthanam (2017) O sistema deve, ser capaz de entender as solicitações e respostas dos usuários em linguagem humana natural. Esses recursos dos sistemas podem reduzir o esforço humano para aprender e compreender as complexas interfaces atuais.

Outro conceito utilizado nas interfaces de usuário são os contextos, o usuário pergunta várias coisas durante a conversa. Espera-se que o sistema armazene as informações sobre os dados anteriores que o usuário forneceu (MITREVSKI, 2018).

“A tecnologia avança a passos largos para tornar as ferramentas amigáveis, intuitivas e integradas. Nesse cenário, surge o conceito de interfaces conversacionais, que propõe uma interação mais natural entre os seres humanos e recursos tecnológicos” (ZENDESK, 2023).

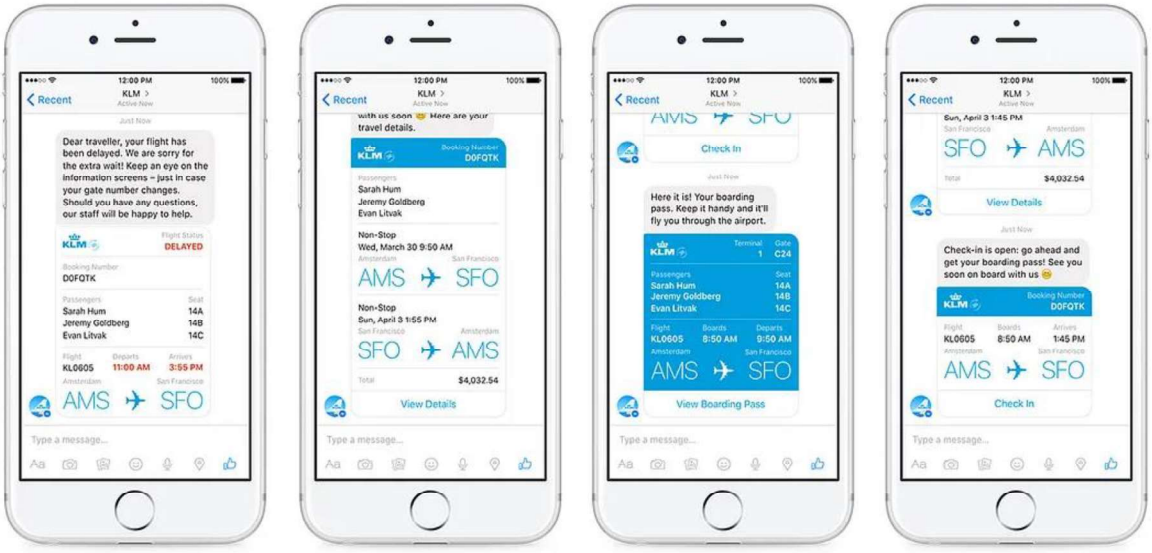
“Sejam lojas virtuais, celulares, televisores etc. O objetivo principal é fazer com que nossa relação com a tecnologia seja a mais orgânica possível, melhorando a experiência que esse contato proporciona” (ZENDESK, 2023).

As interfaces de conversação com o usuário são conhecidas por vários nomes: interfaces de linguagem natural, sistemas de diálogo falado, chatbots,

agentes virtuais inteligentes, assistentes virtuais e assim por diante (JANARTHANAM, 2017)

As aplicações das IUCs são variadas e abrangentes, estendendo-se desde assistentes virtuais pessoais, como Siri e Google *Assistant*, até sistemas de atendimento ao cliente e automação residencial. Em ambientes educacionais, as IUCs oferecem novas formas de aprendizado e interação. Essa versatilidade demonstra a capacidade das IUCs de se integrarem a diferentes aspectos da vida cotidiana, facilitando tarefas e proporcionando acesso fácil a informações.

Figura 1 - Exemplo de uma tela de uma IUC para reserva de uma passagem



Fonte: Brasil.uxdesign.cc (2017)

A Figura 1 demonstra um exemplo de tela de IUC, nela apresenta um conjunto de cinco telas de interface de usuário de um aplicativo móvel para reserva de passagens aéreas e demonstra diferentes etapas do processo de reserva. As duas primeiras telas à esquerda mostram a seleção de voos com detalhes como datas, horários e preços. A tela central exibe a escolha de assentos no voo, enquanto as duas telas à direita mostram uma tela de confirmação de voo e a opção de check-in eletrônico.

Segundo Becker Jacintho e Silva Penha (2016), "É possível pensar nos benefícios que o uso dessas interfaces pode causar na vida das pessoas não

apenas em ganho de produtividade, mas também em melhora na qualidade de vida, como nos casos de pessoas com deficiência, em que podem tornar sua rotina e realização de tarefas mais fácil."

1.1 Inteligência Artificial

A Inteligência Artificial (IA) é um campo da ciência da computação que tem como objetivo fazer com que as máquinas façam coisas inteligentes, como aprender e resolver problemas, como acontece na inteligência natural dos humanos e animais (XIAO, 2022).

A IA representa um campo da ciência da computação dedicado à criação de sistemas capazes de realizar tarefas que, então, exigiam intervenção humana.

Como parte da decisão sobre o que é a inteligência, a categorização da inteligência também é útil. Os seres humanos não usam apenas um tipo de inteligência, mas dependem de várias inteligências para realizar tarefas. (MUELLER; MASSARON, 2021).

IA é um assunto bastante extenso e envolve diversas áreas e tipos, e diversas utilidades no dia de hoje, tem um potencial de grandes transformações nas indústrias e nas relações humano máquina atualmente. A definição sobre IA é bastante complexa e tem diversas definições para este tema.

A história da IA pode ser mapeado desde 1940, durante a segunda guerra mundial, quando o matemático britânico Alan Turing desenvolveu uma máquina de quebra de código que decifrou as mensagens encriptadas alemãs (XIAO, 2022)

Os primeiros computadores eram exatamente isso: dispositivos de computação. Eles imitavam a capacidade humana de manipular símbolos para realizar tarefas matemáticas básicas, como a adição. (MUELLER; MASSARON, 2021).

O termo "inteligência artificial" foi cunhado pela primeira vez pelo cientista da computação americano John McCarthy em 1956, durante um workshop no *Dartmouth College* em *Hanover*, EUA, Syam e Nguyen(2019).

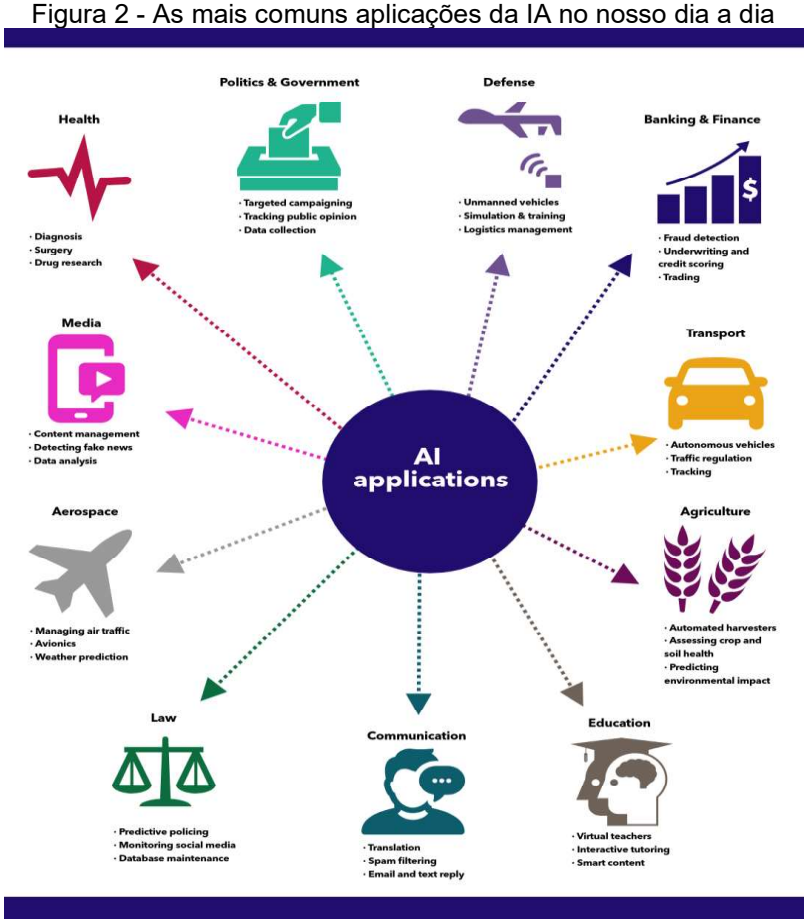
Ainda sobre uma das definições de IA, Dignum(2019) diz que IA é como um artefato computacional construído por meio de intervenção humana que pensa ou age como os humanos, ou como esperamos que os humanos pensem ou ajam.

No campo da ciência da computação, a IA é a disciplina que se preocupa com a criação de sistemas artificiais, que exibem características que associamos à inteligência. Com esses sistemas, também podemos entender melhor como funciona a inteligência humana, Dignum (2019).

Alguns dos sistemas usados para classificar a IA por tipo são genéricos e não distintos. Alguns grupos consideram a IA como forte (inteligência generalizada que pode se adaptar a uma variedade de situações) ou fraca (inteligência específica projetada para executar bem uma determinada tarefa). (MUELLER; MASSARON, 2021).

Historicamente, a IA evoluiu de simples máquinas de lógica para sistemas avançados capazes de aprendizado profundo e processamento de linguagem natural. Atualmente, a IA encontra aplicações em uma variedade de campos, incluindo medicina, onde auxilia no diagnóstico e tratamento de doenças; no setor financeiro, através de algoritmos de negociação automatizados; e no campo da automação, onde possibilita a criação de veículos autônomos e sistemas de manufatura inteligentes. Essas aplicações demonstram a versatilidade da IA e seu potencial para resolver problemas complexos em diversos domínios.

Figura 2 - As mais comuns aplicações da IA no nosso dia a dia



Fonte: Cgtn (2019)

A Figura 2 mostra um infográfico sobre a utilização das IA no nosso cotidiano, os exemplos vão desde o cenário de transporte com veículos autônomos até o campo de medicina com pesquisa de drogas, diagnósticos entre outros.

O uso de IA é amplo nos aspectos da nossa vida, sobre isso Xiao (2022) Assistentes pessoais, como a Alexa da Amazon, a Siri do iPhone, a Cortana da Microsoft e o Google Assistant, dependem da IA para entender o que você disse e seguir as instruções para executar as tarefas de acordo.

A IA também tem sido usada nos setores de saúde, manufatura, carros sem motorista, finanças, agricultura e muito mais (XIAO,2022)

Sobre os objetivos da IA Dignum(2019) discute que a computação reúne duas perspectivas principais sobre a IA. Em primeiro lugar, a perspectiva da engenharia,

que postula que o objetivo da IA é resolver problemas do mundo real por meio da criação de sistemas que apresentem comportamento inteligente. Em segundo lugar, a perspectiva científica, que tem o objetivo de entender que tipo de mecanismos computacionais são necessários para modelar o comportamento inteligente.

A campo de estudo da IA é abrangente e é dividido em diversas subcampos de estudos que podemos citar, que são: Aprendizado de Máquina (*Machine Learning*), Redes Neurais, Aprendizado Profundo (*Deep Learning*), Processamento de Linguagem Natural, Computação Cognitiva, Visão Computacional.

O Machine Learning (ML) é uma subárea da IA consiste em um conjunto de algoritmos matemáticos que podem analisar dados automaticamente. O ML clássico pode ser dividido em aprendizado supervisionado e aprendizado não supervisionado (XIAO, 2022)

De acordo com Xiao (2022) exemplos de aprendizado supervisionado incluem reconhecimento de fala e reconhecimento de imagem. Exemplos de aprendizado não supervisionado incluem segmentação de clientes, detecção de defeitos e detecção de fraudes.

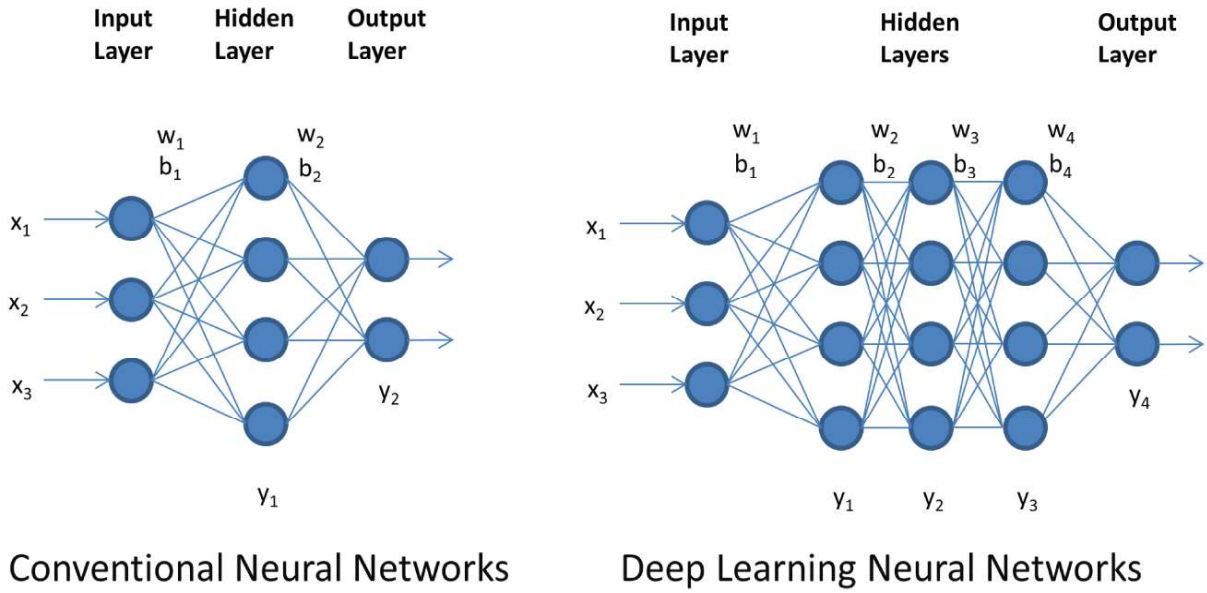
O *Machine Learning* é, basicamente, a prática de usar algoritmos para analisar dados, aprender com eles e prever ou decidir algo no mundo. Em vez de escrever sequências de código de software manualmente com um conjunto específico de instruções para realizar uma determinada tarefa, a máquina é “treinada” com grandes volumes de dados e algoritmos que conferem a ela a capacidade de aprender a executar a tarefa, Copeland (2021).

Redes neurais foram desenvolvidas baseados no cérebro humano imitando as redes neurais biológicas humanas, geralmente possui três camadas, uma camada de entrada, camada oculta e uma camada de saída (XIAO, 2022).

As redes neurais usam uma grande quantidade de dados e após o treinamento é possível usar para prever resultados de dados não previstos, o que causou uma grande atenção durante a década de 70 (XIAO, 2022)

*Deep Learning (*DL) é um tipo especial de rede neural que tem mais de uma camada oculta, o que foi possível a partir do avanço da capacidade computacional. (XIAO, 2022)

Figura 3 - Exemplificação das redes neurais tradicionais e de DL



Fonte: XIAO (2022)

A Figura 3 representa a evolução das redes neurais convencionais para as redes neurais da DL, é claramente notável o como a quantidade das camadas ocultas cresceram e tornaram estes algoritmos mais complexos.

Exemplificando *Deep Learning*, Copeland(2021) diz, imagine que uma imagem seja dividida em pedaços menores e eles sejam colocados na primeira camada da rede neural. Os neurônios individuais da primeira camada transferem os dados para a segunda camada. A segunda camada de neurônios faz sua parte e assim por diante, até que a camada e os resultados sejam produzidos.

Sobre a PLN Xiao (2022) destaca que se trata do desenvolvimento de aplicativos e serviços capazes de processar e entender a linguagem humana, recuperar informações significativas a partir dela ou até mesmo gerar mensagens/conversas de texto.

O processamento de linguagem natural (PLN) é a área da IA que dá aos computadores a habilidade de entender e manipular a linguagem humana, seja em texto ou fala.

No próximo subcapítulo, será explorado mais profundamente o PLN. Essa discussão permitirá entender como o PLN é fundamental na criação de interfaces de comunicação eficientes e intuitivas entre humanos e máquinas.

Dado os desafios do aumento da quantidade de dados, complexidade e disponibilidade e cada vez mais poder computacional, um novo paradigma para liderar com dados complexos precisava ser criado, esse algoritmo é conhecido como computação cognitiva (HIGH; BAKSHI, 2019).

A computação cognitiva é uma evolução da tecnologia que tenta dar sentido a um mundo complexo que está se enchendo de dados que tem todas as formas e formatos (HURWITZ; KAUFMAN; BOWLES, 2015)

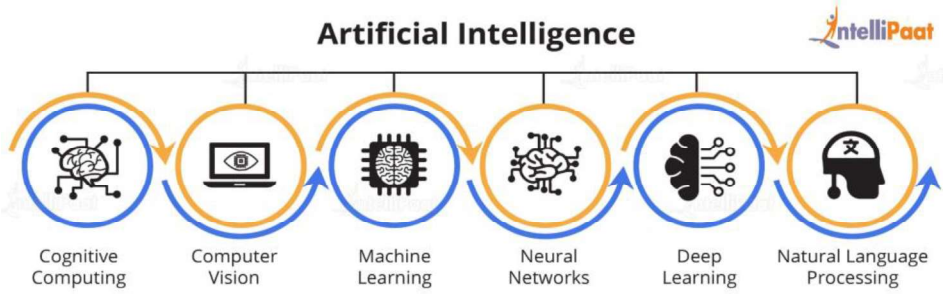
Conforme Hurwitz, Kaufman e Bowles (2015) a computação cognitiva permite examinar um conjunto vasto de diversos tipos de dados, interpretar e prover *insigths* e ações recomendadas.

Por exemplo, estão sendo desenvolvidos sistemas que podem permitir que um gestor de cidade preveja quando o tráfego será interrompido por eventos climáticos e redirecione esse tráfego para evitar problemas, ou provisões relacionadas a vendas para clientes (HURWITZ; KAUFMAN; BOWLES, 2015)

E por fim, a visão computacional é um campo da IA que permite aos computadores interpretarem e entender o mundo visual. Sobre os desafios dessa tecnologia High e Bakshi (2019) tradicionalmente, as máquinas têm tido muita dificuldade para entender dados visuais, que são dados apresentados na forma de imagens. Isso se deve ao grande número de maneiras pelas quais uma única entidade pode ser representada.

Sobre a aplicabilidade da visão computacional Davies (2023) nos diz que, as aplicações de reconhecimento de imagens, como a detecção de faces usada em biometria, são uma de suas aplicações comuns. Outras aplicações incluem fotografia computacional, gráficos automatizados e edição de fotos, além de realidade aumentada, entre outras.

Figura 4 - Alguns dos subcampos de estudos da inteligência artificial.



Fonte: Intellipat (2023)

A Figura 4 mostra alguns dos subcampos de estudos da IA, temos desde a computação cognitiva até o processamento de linguagem natural que será utilizado neste trabalho. Importante destacar que estes subcampos têm diversas outras áreas de estudos atreladas a mesma.

Apesar de seus benefícios, a IA também apresenta desafios significativos, especialmente no que diz respeito à ética e ao impacto social. Questões como viés algorítmico, privacidade de dados e desemprego tecnológico são preocupações crescentes. O viés em sistemas de IA pode perpetuar e amplificar preconceitos existentes na sociedade, enquanto a coleta e análise de grandes volumes de dados pessoais levantam questões sobre privacidade e segurança. Além disso, a automação impulsionada pela IA ameaça deslocar trabalhadores em várias indústrias, criando desafios para a economia e para a estrutura do mercado de trabalho

1.2 Processamento de Linguagem Natural (PLN)

O Processamento de Linguagem Natural (PLN) é uma popular subárea da inteligência artificial e linguística. É sobre desenvolver sistemas capazes de

processar e entender a linguagem humana, recuperar peças significativas de informação sobre ela. (XIAO, 2022)

O PLN é um ramo da Inteligência Artificial e da Linguística, dedicado a fazer com que os computadores entendam as declarações ou palavras escritas em línguas humanas. Ele surgiu para facilitar o trabalho do usuário e satisfazer o desejo de se comunicar com o computador em linguagem natural (KHURANA et al., 2023, p.2).

O desenvolvimento da PLN durante a história pode ser dividido em três estágios: de 1950 até 1990 é um processamento de linguagem baseado em regras definidas (XIAO, 2022).

De 1990 até 2010 o processo agora usa uma quantidade enorme de dados e algoritmos de ML, ao período de que de 2010 para cá usa sistemas de *deep learning*. (XIAO, 2022)

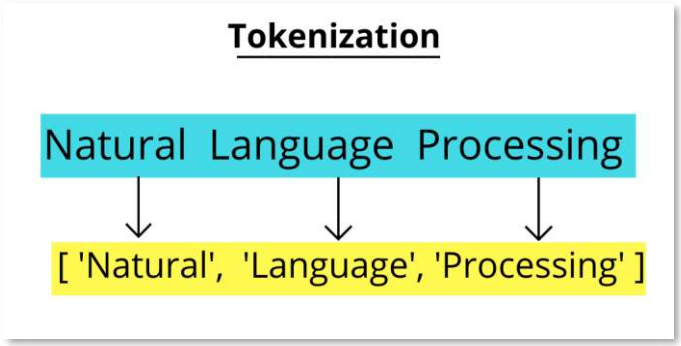
No cerne do PLN está a tarefa de traduzir a linguagem humana em um formato que os computadores possam compreender e processar. Isso envolve uma variedade de subtarefas, incluindo a análise sintática (compreender a estrutura gramatical das sentenças), a análise semântica (interpretar o significado das palavras e frases) e a análise de sentimentos (determinar as atitudes e emoções expressas no texto).

Um sistema de processamento de linguagem natural é muitas vezes chamado de pipeline porque geralmente envolve vários estágios de processamento em que a linguagem natural flui em uma extremidade e a saída processada flui na outra. (HOWARD; LANE; HAPKE, 2019)

Um dos principais desafios do PLN é a ambiguidade à linguagem humana sobre isso High e Bakshi (2019) A linguagem é algo complexo. É cheia de nuances, sutilezas, insinuações e metáforas. Não damos valor a essa complexidade, nem sequer pensamos nesses detalhes quando nos comunicamos uns com os outros.

O Processamento de Linguagem Natural (PLN) é uma tecnologia complexa que busca compreender e interpretar a linguagem humana de forma automatizada. Essa tecnologia opera em várias camadas de análise para entender uma frase. Para Xiao (2022) temos as seguintes tarefas comuns para os algoritmos de PLN, Análise Léxica, Análise Sintática, Análise Semântica, Análise de divulgação, análise pragmática.

Figura 5 - Um exemplo do funcionamento do processo de tokenização



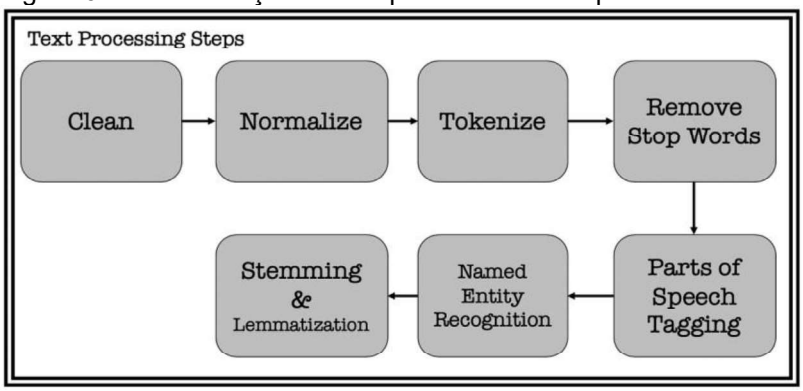
Fonte: Analytics Vidhya (2021)

A Figura 5 mostra um processo de tokenização para a frase "*Natural Language Processing",* cada palavra é separada nesta frase de forma a separar elas, e em cada palavra é criada um token.

Encontrado na camada de análise léxica, sobre a tokenização Howard, Lane e Hapke (2019) explicam que é a primeira etapa de um pipeline de PLN e, portanto, pode ter um grande impacto no restante do pipeline. Um tokenizador divide dados não estruturados, textos em linguagem natural, em pedaços de informações que podem ser contados como elementos discretos

Seguindo a tokenização, o PLN pode realizar uma série de análises mais profundas, como morfológica (estudo das formas das palavras), sintática (análise das estruturas das frases), e semântica (compreensão dos significados).

Figura 6 - Demonstração das etapas utilizadas no processo de PLN



Fonte: Medium (2020)

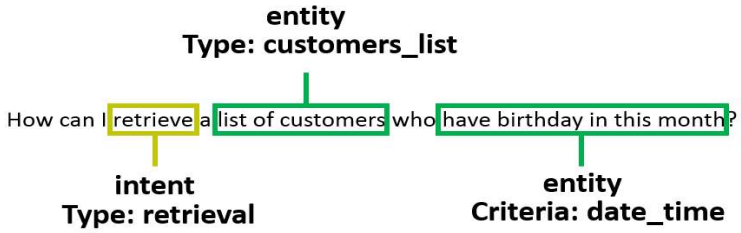
A Figura 6 mostra um processo de processamento de um texto, nela temos como primeira etapa o processo de limpeza da frase, eliminando palavras repetidas que não contribuem para entender o significado total da frase, depois temos um processo de normalização, tokenização para separar cada palavra em um token, depois remoção das palavras "*Stop Words*", tageamento de falas, reconhecimento das entidades nomeado e *stemming* e lematização

A análise semântica é uma característica essencial da abordagem do Processamento de Linguagem Natural (PLN). Indica no formato apropriado o contexto de uma frase ou parágrafo. A semântica trata do estudo da significância da linguagem. O vocabulário usado transmite a importância do assunto por causa da inter-relação entre as classes linguísticas (HUSSEN MAULUD et al., 2021).

Aplicações práticas do PLN são amplamente vistas e variadas, abrangendo desde assistentes virtuais e *chatbots* até sistemas de tradução automática e ferramentas de análise de sentimentos. Essas aplicações estão transformando a maneira como interagimos com a tecnologia, tornando possível uma comunicação mais natural e intuitiva com os computadores.

Outra funcionalidade das PLNs, sobre a etapa de extração de intenções Mitreviski (2018) a primeira etapa do processamento de linguagem natural é descobrir qual das intenções que você forneceu corresponde à frase falada pelo usuário.

Figura 7 - Exemplo de extração de uma intenção e entidades



Fonte: Medium (2023)

A Figura 7 exibe um exemplo de extração de intenção e entidade, nela temos a frase “Como eu posso obter uma lista de clientes que fazem aniversário neste mês?” tratando de um usuário que deseja obter a lista de clientes com aniversário no mês atual. O processo de extração busca primeiramente a intenção da frase, que neste caso é “*retrieval*” ou obter. Nesta frase temos duas entidades, a primeira “lista de clientes” do tipo de entidade “lista de clientes” e a outra entidade “Aniversário neste mês” do tipo data.

A pesquisa em PLN também está avançando rapidamente, impulsionada por avanços em algoritmos de IA, disponibilidade de dados e poder computacional.

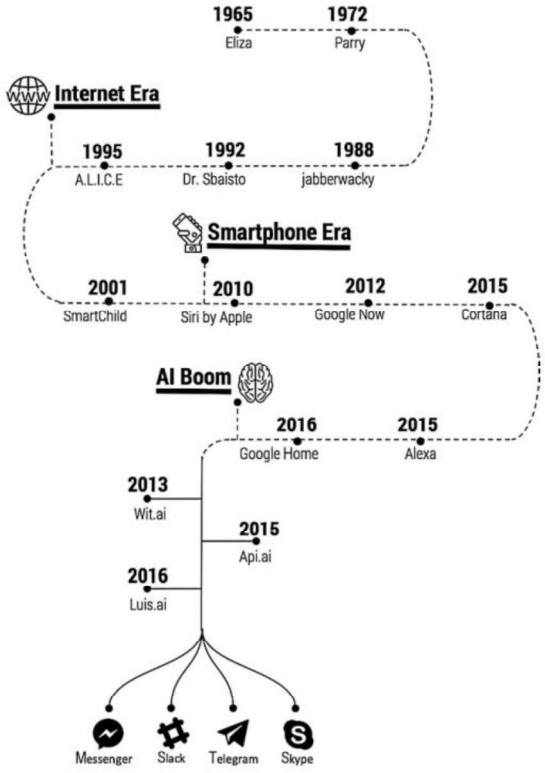
Há diversas ferramentas no mercado que oferecem as funcionalidades de PLN de forma fácil, dentre estas estão grandes empresas de mercado como Google, Microsoft, Facebook, IBM e outros.

1.3 Chatbot

Chatbots, uma combinação de "*chat*" e "*robots*", são programas de computador projetados para simular a conversação humana. Surgindo como uma inovação na interação homem-máquina. Originalmente desenvolvidos na década de 1960, com exemplos como ELIZA e PARRY, eles evoluíram com o advento da inteligência artificial (IA) e do processamento de linguagem natural (PLN). Hoje, *chatbots* são capazes de conduzir conversas complexas, oferecendo desde suporte ao cliente até assistência personalizada em vários setores, incluindo saúde, finanças e educação.

Sobre a história dos *chatbots* Khan e Das (2018) O primeiro *chatbot* de sempre foi introduzido antes mesmo de o primeiro computador pessoal ter sido desenvolvido. Chamava-se Eliza e foi desenvolvida no Laboratório de Inteligência Artificial do MIT por Joseph Weizenbaum em 1966. Eliza fazia-se passar por uma psicoterapeuta. Eliza examinava as palavras-chave de entradas do utilizador e acionava as regras de transformação da saída.

Figura 8 - Uma linha do tempo do desenvolvimento de *chatbots*



Fonte: *Introduction to Chatbots* (2017)

A Figura *8* mostra os a evolução dos *chatbots* ao longo da história, desde o chatbot Eliza até aos canais comunicadores atuais como *Telegram*, *Messenger* entre outros. Também é destacado os importantes marcos da história como a criação da internet, a era dos smartphones e a explosão mais recente das IAs.

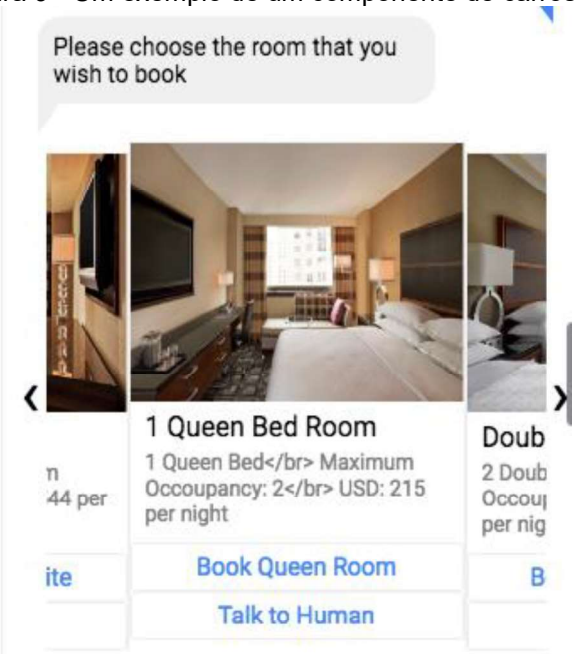
Os *chatbots* funcionam com base em um conjunto de regras programadas e podem utilizar tecnologias de aprendizado de máquina e PLN. Esses sistemas são treinados para entender e responder a consultas em linguagem natural, tornando a interação com os usuários mais intuitiva e eficiente. Empresas utilizam *chatbots* para

automatizar respostas a perguntas frequentes, coletar dados de clientes, realizar transações e até fornece recomendações personalizadas. Além disso, *chatbots* estão sendo integrados em dispositivos domésticos inteligentes, assistentes virtuais e aplicativos móveis, tornando-se uma interface cada vez mais comum para a interação digital.

A maior vantagem de usar uma interface baseada em bate-papo em comparação com aplicativos móveis/web/aplicativos móveis é oferecer ao consumidor a capacidade de transmitir suas intenções em linguagem natural, como se estivesse falando com seus amigos, Khan e Das (2018)

Ainda sobre interfaces de *chatbots*, há uma gama de componentes que podem ser usados para facilitar a interação do usuário com a aplicação, estes componentes vão desde botões agrupados, conhecidos como *quick replies* que ajudam o usuário a escolher entre opções, até mesmo carrosséis que são bem úteis na apresentação de produtos.

Figura 9 - Um exemplo de um componente de carrossel.



Fonte: hybrid.chat (2023)

A Figura 9 mostra um exemplo de componente de carrossel, nela é usada para mostrar um catálogo de quartos disponíveis para alugar, e ações especificas para cada quarto, é uma forma bem intuitiva de mostrar um catálogo.

Sobre os tipos de *chatbots* podemos listar como sendo: Baseados em menus e botões, baseados em regras, com tecnologia de IA, Baseados em voz e IA Generativa.

Sobre os *chatbots* baseados em menus e botões Church (2023), *chatbots* baseados em menus ou botões são o tipo mais básico de chatbot, em que os usuários podem interagir com eles clicando na opção de botão ou de um menu com script que melhor represente suas necessidades.

Chatbots baseados em regras o empregam a lógica condicional se/então para desenvolver fluxos de automação de conversas, Church (2023). Dessa forma esses bots são programados de forma a seguir regras estritamente, tem perguntas e respostas predefinidas e tem uma certa dificuldade para lidar com solicitações mais complexas, se limitando muitas vezes a sistemas FAQ.

Os *chatbots* com IA podem entender as perguntas do usuário, independentemente de como elas são formuladas. Com os recursos de IA como processamento de linguagem natural (PLN), o bot de IA pode detectar rapidamente todas as informações contextuais relevantes compartilhadas pelo usuário, permitindo que a conversa progrida de forma mais tranquila e coloquial, Church (2023). Será o tipo de bot utilizado neste projeto devido sua grande capacidade de adaptabilidade e reconhecimento de intenções de usuários.

Um chatbot de voz é outra ferramenta de conversação que permite que os usuários interajam com o bot falando com ele, em vez de digitar, Church (2023). Estes *bots* alinhados às tecnologias com IA oferecem ainda mais poder e uma usabilidade aprimorada para os usuários, e podem ser inclusive usados no desenvolvimento de *chatbots* específicos para acessibilidade.

Sobre *chatbots* generativos, a próxima geração de *chatbots* com recursos de IA generativa pode oferecer uma funcionalidade ainda mais aprimorada por meio da fluência na compreensão da linguagem comum, da capacidade de se adaptar ao estilo de conversa do usuário e do uso da empatia ao responder às perguntas dos usuários. Enquanto os *chatbots* com IA conversacional podem digerir as perguntas ou os comentários dos usuários e gerar uma resposta semelhante à humana, os *chatbots* com IA generativa podem dar um passo adiante, gerando um novo conteúdo como resultado, Church(2023).

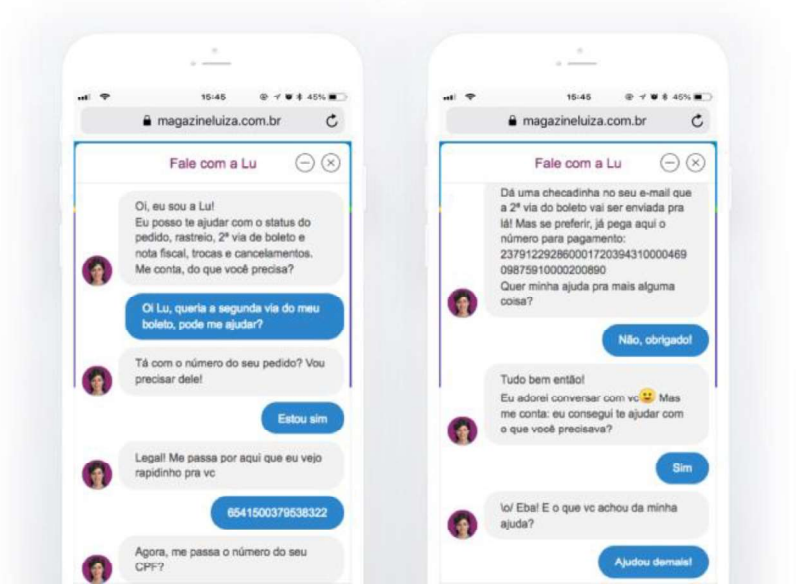
1.3.1 Exemplos de chatbots

Neste subcapitulo será apresentado o levantamento de alguns dos *chatbots* disponíveis no mercado, que utilizam diversos conceitos apresentados neste trabalho, como uma interface conversacional, utilização de IA e componentes de chatbot.

1.3.1.1 Magazine Luiza (Lu)

Sobre o que é o bot Lu, Calado(2018) criado pela Magazine Luiza, é o assistente pessoal para ajudar no atendimento e processo de pós-venda.

Figura 10 - Tela conversacional do bot Lu



Fonte: Redação Nama (2018)

Na Figura 10 é um fluxo de conversa com o bot “Lu”, na imagem é mostrado um caso de um pedido pela segunda via de um Boleto, o fluxo como pode se perceber é bem humano e natural.

O sistema da Magalu utiliza PLN para entender a linguagem natural, compreender gírias e até mesmo erros de português. Isso facilita ainda mais o entendimento entre máquina e consumidor, Redação Nama (2018).

Entre as funcionalidades estão rastreamento de entrega, emissão de segunda via do boleto, nota fiscal, levantamento de informações, status dos pedidos e entrega, Redação Nama (2018)

Sobre os dados de utilização do chatbot e o grande potencial que essa ferramenta pode fornecer temos segundo a Agência New Voice (2020), o assistente virtual do Magalu, atingiu no primeiro semestre de 2020 algo em torno de 1,4 milhões de atendimentos por mês, com média de 8,5 milhões de interações e cerca de 6,1 mensagens por usuário. Esses dados demonstram uma grande capacidade de uso por partes dos clientes destas tecnologias.

1.3.1.2 Chatbots para bibliotecas

Neste capítulo será elencado dois chatbots que utilizam IA que foram desenvolvidas por alunos para os contextos de suas experiencias com o uso de suas bibliotecas nos seus espaços acadêmicos.

Um dos chatbots desenvolvidos para uma dissertação de mestrado, é um chatbot para bibliotecas universitárias para tirar dúvidas relacionadas à lei de direito autoral, realizado por Santos (2023).

Segundo Santos (2023) O ambiente da biblioteca universitárias é um espaço de aprendizagem para a comunidade acadêmica e contribui para o suporte ao ensino, a pesquisa e extensão.

Sobre o interesse da temática Santos (2023) diz que, surgiu de uma experiencia da autora com interação direta com a comunidade acadêmica, com questionamentos sobre o direito autoral. O principal objetivo deste trabalho foi desenvolver um chatbot que pudesse responder perguntas sobre a Lei de Direito Autoral. Este chatbot utiliza IA para dinamizar as respostas.

Outro chatbot desenvolvido para o ambiente acadêmico é o Bibliotecária Informativa Automatizada (BIA) desenvolvida na Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e desenvolvido por Godinho (2019).

Para a oportunidade de uso da IA nas bibliotecas, Godinho (2019) implantada em setores da biblioteca tem grande potencial para melhorar serviços e produtos ofertados, dando à biblioteca espaço para desenvolvimento na pesquisa.

O *chatbot* é disponibilizado como uma bot no Facebook. para a realização de uma consulta utilizando a assistente virtual da PUC-Rio, é necessário ter Internet e possuir uma conta no Facebook, onde as informações públicas do perfil do indivíduo são utilizadas para personalizar o acolhimento do usuário. Godinho (2019)

Figura 11 - Tela de conversação com o Bot BIA

Fonte: Godinho (2019)

Na Figura 11 é mostrado a tela de conversação com o Bot desenvolvido por Godinho (2019).

A criação do chatbot BIA foi motivada pela necessidade de melhorar o atendimento da Divisão de Biblioteca e Documentação e a relação com o público, oferecendo um serviço acessível através do Facebook, uma plataforma popular entre os alunos.

Sobre a motivação Godinho (2019), a biblioteca da universidade percebeu os alunos da não tinha o costume de consultá-la e usufruir de seus recursos. Com um baixo número de usuários frequentes, grande parte dos alunos concluíram seus cursos sem nunca terem visitado as bibliotecas.

Estas duas soluções de chatbots para bibliotecas demonstram um poder de ganho para as bibliotecas universitárias, seja para tirar dúvidas relacionadas a procedimentos internos, ou ao consumo de serviços destes ambientes.

O uso de interfaces conversacionais permite que esses sistemas tragam os alunos para mais perto das bibliotecas, permitindo um ambiente acessível e com uma dinamicidade facilitada pelo uso de IAs.

1.4 Banco de dados

Os sistemas de gerenciamento de banco de dados apresentam distintas funções: alguns focam em dados temporários de rápido acesso, outros em armazenamento de longo prazo, alguns suportam consultas analíticas complexas, enquanto outros limitam-se a buscas por chave. Existem ainda os otimizados para dados de séries temporais e aqueles eficientes em armazenar grandes volumes. (PETROV, 2019)

Há coisas em um ambiente de negócios sobre as quais precisamos armazenar dados, e essas coisas estão relacionadas umas às outras de várias maneiras. De fato, para ser considerado um banco de dados, o local onde os dados são armazenados deve conter não apenas os dados, mas também informações sobre as relações entre esses dados. (HARRINGTON, 2016).

Bancos de dados são estruturas utilizadas para armazenar, organizar e gerenciar grandes volumes de dados. Eles são utilizados em muitas áreas, desde negócios e saúde à educação e governo, fornecem acesso rápido e seguro às informações.

Sobre os bancos de dados relacionais Conforme Foster e Godbole (2014) é, de longe, o modelo mais amplamente usado para o design de bancos de dados. O seu sucesso deve ao fato de estar firmemente fundamentado em princípios matemáticos. Eles são ideais para situações em que a integridade e a precisão dos dados são cruciais, como em sistemas financeiros e de gestão de recursos humanos. Exemplos populares incluem MySQL, PostgreSQL e Oracle.

Figura 12 - Um exemplo de tabelas e suas relações em um banco de dados relacional.

Fonte: Autor (2023)

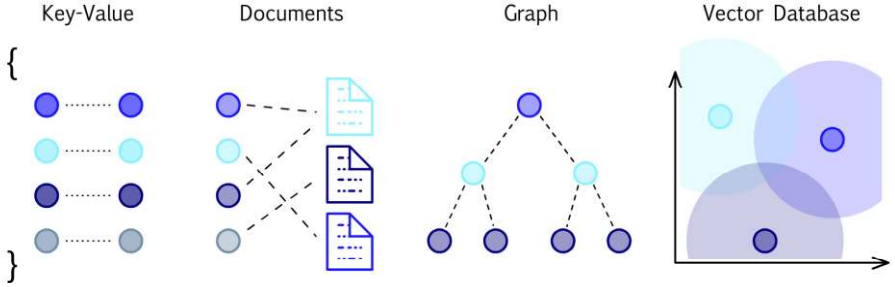
A Figura 12 mostra um exemplo de como é relacionado os dados em um banco relacional, na imagem é visto as relações entre as tabelas como Livro e Categoria, Aluno e AlunoAutenticação.

Segundo Schwaber-Cohen (2023), estamos no meio da revolução da IA. Ela está revolucionando todos os setores em que atua, prometendo grandes inovações, mas também introduz novos desafios. O processamento eficiente de dados tornou-se mais crucial do que nunca para aplicativos que envolvem grandes modelos de linguagem, IA generativa e pesquisa semântica.

Uma aplicação famosa que utiliza tecnologia de banco de dados vetorial é o Elasticsearch. O Elasticsearch é uma plataforma de busca e análise de dados amplamente utilizada, conhecida por sua velocidade, precisão e escalabilidade. Ele é capaz de indexar e pesquisar dados vetoriais, tornando-o uma ferramenta poderosa para aplicações que requerem análises geoespaciais rápidas e precisas.

A Figura 13 mostra os tipos de armazenamento de banco de dados, da esquerda para a direita temos os bancos de dados chave valor, na próxima temos os bancos não relacionais ou *no-SQL,* depois temos os bancos em estrutura de grafos, e por último os bancos de dados vetoriais.

Figura 13 - Exemplos de tipos de armazenamento para banco de dados.



Fonte: Cohen (2023)

1.5 Comunicação em Tempo Real

A Comunicação em Tempo Real (RTC, do inglês "Real-Time Communication") refere-se a qualquer forma de comunicação eletrônica que acontece

instantaneamente, permitindo a troca de informações em tempo real sem atrasos perceptíveis. Esta tecnologia é fundamental na era digital, pois oferece interações imediatas e eficientes, essenciais em vários contextos, desde comunicações pessoais até aplicações empresariais e de emergência.

Conforme Kopetz e Steiner (2022), muitos sistemas em tempo real são sistemas de computador distribuídos. Portanto, eles exigem um subsistema de comunicação em tempo real que garanta a troca de mensagens confiável e oportuna para permitir que o sistema de computador distribuído opere como um todo coordenado.

A importância da RTC pode ser vista em diversos aspectos. Primeiramente, ela facilita a colaboração e a tomada de decisões em ambientes corporativos, pois permite que as equipes se comuniquem e compartilhem informações instantaneamente. Além disso, em setores como saúde, financeiro e de emergências, a RTC é crucial para fornecer respostas rápidas, que podem ser vitais.

As aplicações da RTC são amplas e variadas, exemplos incluem chamadas de vídeo, mensagens instantâneas, jogos online, telemedicina, entre outros. Todas estas aplicações dependem da capacidade de transmitir, receber e processar dados em tempo real para proporcionar experiências fluidas e produtivas.

Dentro desse universo de RTC, temos o SignalR como uma ferramenta no desenvolvimento de aplicações web. SignalR é uma biblioteca para .NET que facilita a adição de funcionalidades de comunicação em tempo real a aplicações web. Ele permite que o servidor envie conteúdo para os clientes instantaneamente, o que é ideal para aplicações que exigem interações em tempo real, como chats online, jogos e atualizações ao vivo de dados ou de mídias sociais.

Um sistema de mensagens em tempo real pode ser desenvolvido usando o Azure SignalR Service, que é baseado na biblioteca SignalR. A SignalR é uma biblioteca de código aberto (sob a .NET Foundation) que permite que os servidores enviem conteúdo para todos os clientes conectados instantaneamente, conforme e quando os dados estiverem disponíveis.

"Um sistema de mensagens em tempo real pode ser baseado na biblioteca SignalR. A SignalR é uma biblioteca de código aberto (sob a .NET Foundation) que permite que os servidores enviem conteúdo para todos os clientes conectados instantaneamente, conforme e quando os dados estiverem disponíveis" (VEMULA, 2019).

O SignalR tem a capacidade de proporcionar uma comunicação bidirecional entre servidores e clientes, o SignalR é compatível com uma variedade de técnicas para realizar essa comunicação em tempo real, como *WebSockets*, *Server-Sent Events* e *Long Polling*, escolhendo automaticamente a melhor opção disponível conforme a capacidade do cliente e do servidor.

1.6 SOLID

Um princípio utilizado no projeto foi o SOLID, isso permite a construção de softwares mais fáceis de se entender, dar manutenção e modulares. Hoje a maioria das linguagens de programação são de programação orientadas a objeto (POO), essas linguagens permitem pensar nos programas como classes e objetos, Joshi (2016)

É importante saber que essas funcionalidades podem ser utilizadas da melhor maneira possível para resultar em um código robusto, flexível, manutenível e extensível, Joshi (2016)

Enquanto o POO provê um paradigma de programação, ele não provê um princípio que instrui como as responsabilidades das classes deverão ser, escrever POO não garante que esteja seguindo um princípio de design, Arora (2017)

SOLID são princípios que não estão atrelados a nenhuma linguagem de programação. Sobre SOLID Arora (2017) fala que, são princípios que fazem nosso código ser melhor e mais claro.

SOLID é acrônimo para cinco princípios de *design* de *software*. Michael Feathers introduziu o acrônimo SOLID para nos ajudar a lembrar desses princípios facilmente, as iniciais de cada princípio formam a palavra SOLID, Joshi (2016)

Separando cada palavra do acrônimo, temos: **S**, *Single Responsability Principle* (Princípio da responsabilidade única), **O** *Open-Closed Principle* (Princípio aberto-fechado), **L —** Liskov *Substitution Principle (*Princípio da substituição de Liskov), **I —** *Interface Segregation Principle,* também citado como ISP (Princípio da Segregação da Interface), **D —** *Dependency Inversion Principle* (Princípio da inversão da dependência).

Cada um destes princípios é uma valiosa prática de como cada desenvolvedor de softwares deveria aprender. Quando usados juntos, estes padrões nos dão uma estrutura completamente diferente, Hall (2017)

Single Responsability Principle (SRP) o princípio da responsabilidade única, Hall (2017) nos diz que, instrui o desenvolvedor que deve haver uma única razão para mudar, se uma classe possui mais de uma razão para mudar, ela tem mais de uma responsabilidade.

Uma classe deve haver apenas uma responsabilidade, se você seguir SRP suas classes se tornarão mais compactas e organizadas. É uma forma de dividir o problema inteiro em partes pequenas, Joshi (2016).

Open-Closed Principle (OCP) define que uma classe só deve ser aberta para extensão e não para modificação, isso significa que você pode criar uma classe e usá-la, mas não a modificar, Joshi (2016).

Para Robert C. Martin para um código ser aberto para extensão isso significa que o desenvolvedor deve ser capaz de responder a mudanças de requisitos e suporte a novas funcionalidades, Hall (2017). Ainda sobre essa explicação Hall (2017) nos diz que, isso deverá ser alcançado fechando para modificação, os desenvolvedores deverão suportar novas funcionalidades sem alterar o código já existente, módulo, binário ou etc.

O OCP é um princípio que guia o design de classes e interfaces para que os desenvolvedores possam criar códigos que mudam o tempo todo, Hall (2017).

Sobre Princípio da substituição de Liskov (Liskov *Substitution Principle - LSP),* Hall nos diz que*,* é uma coleção de regras para criar hierarquias de heranças em que qualquer cliente pode chamar uma classe ou subclasse sem comprometer o comportamento esperado.

Isso significa que em uma parte do programa, você pode usar uma classe derivada invés de sua classe base sem alterar o funcionamento correto do programa, Sarcar (2022)

O Princípio da Segregação da Interface define que é melhor ter algumas interfaces menores que interfaces maiores, Weisfeld (2019)

Neste sentido o *Interface Segregation Principle (ISP),* nos auxilia a implementar interfaces mais limpas. Nós queremos prover clientes que tenham interfaces que apenas contenham métodos que irão necessitar, Vieira (2023)

Há muitas razões pelas quais as interfaces devem ser segregadas - para ajudar na decoração, para esconder funcionalidades dos clientes, como auto documentação para outros programadores, Hall (2017)

Sobre o princípio da inversão da dependência, Hall (2017) conceitua que, introduz abstrações sob a forma de interfaces das quais depende o código do cliente e das quais dependem os implementadores.

Um dos objetivos do Princípio de Inversão de Dependência é escolher objetos em tempo de execução, e não em tempo de compilação. (Você pode mudar o comportamento do seu programa em tempo de execução), Weisfeld (2019)

Neste capítulo foi feito uma breve explicação dos princípios SOLID, estes princípios serão utilizados no trabalho e facilitam a manutenção, legibilidade, desacoplamento

e as boas práticas de desenvolvimento. Permite também que nosso código seja modular e utilize uma linguagem de mercado para o trabalho.

CAPÍTULO 2: EXPERIMENTOS DE FERRAMENTAS DE PROCESSAMENTO DE LINGUAGEM NATURAL (PLN)

Para a realização dos experimentos, foi definido um conjunto de frases relacionadas à temática do projeto. Cada frase deve se relacionar a uma intenção de usuário que a ferramenta deve identificar corretamente. Os experimentos foram conduzidos por meio da entrada de texto diretamente na ferramenta testada, seguida pela verificação da resposta da ferramenta, documentação de seus resultados, concluindo pela análise e comparação com os resultados das demais ferramentas analisadas.

As frases foram definidas pelo próprio autor com base num conjunto de frases que se espera ser comuns no ambiente de uma biblioteca, foram separadas 15 frases das frases totais registradas na aplicação para fins de atender estes experimentos. Importante notar que, devido a aplicação utilizar IA, o aprendizado do *chatbot* com novas formas é feito de forma automática, o chatbot passa a entender ainda mais frases nesse sentido, um dos benefícios da utilização da IA na aplicação. Estas frases foram categorizadas nas três intenções utilizadas na aplicação, sendo consultar livro, consultar minhas reservas e reservar um livro.

Para os experimentos, foram selecionadas as ferramentas de Processamento de Linguagem Natural (PLN): *CLU - Conversational Language Understanding*, da *Microsoft Azure*; *Dialogflow* da *Google*; e Wit.ai do Facebook. Este capítulo descreve os experimentos realizados com estas ferramentas.

O objetivo dos testes é verificar a acurácia das ferramentas na extração de intenções a partir de um texto. A extração de entidades pode ser um fator secundário na avaliação, sendo importante para avaliar a capacidade da ferramenta em extrair pontos-chave de uma frase. Além disso, será discutido sobre os planos, desvantagens e benefícios de cada ferramenta.

É importante, então, que as ferramentas aqui testadas possam atingir uma taxa de acurácia maior que 70% para determinar que elas possam atender aos

nossos usuários e que essas ferramentas atendam a demanda de extração de intenções de usuário a partir de uma frase.

As intenções utilizadas no sistema estão descritas no quadro 1 abaixo, onde é descrito o nome da intenção e qual sua utilidade.

Quadro 1 - Intenções usadas no chatbot e descrição

Nome da Intenção	Descrição
Consultar livro	Consultar um livro específico.
Consultar minhas reservas	Visualizar as reservas de livros feitas pelo aluno.
Reservar um livro	Reservar um livro específico.

Fonte: Autor (2023)

No quadro 1 é exibido as intenções utilizadas no projeto, descrevemos aqui três intenções para o projeto que representam as funcionalidades para o nosso *chatbot.* Temos a intenção consultar livro, que é responsável para consulta baseado por uma descrição inserida pelo usuário. A intenção de consultar minhas reservas, onde o aluno tem acesso as reservas já realizadas no *chatbot.* Reservar um livro representa a funcionalidade de alugar um livro.

Nos próximos quadros abaixo é mostrado as 10 frases utilizadas para cada intenção, desse modo esperamos que a partir de uma entrada de uma frase dessas, as ferramentas testadas, possam retornar à intenção que está de acordo com o quadro.

As frases foram escolhidas de modo que sejam frases comuns que um aluno possa utilizar na aplicação, teve-se como objetivo deixá-las aleatórias e misturar com as entidades, neste caso a entidade livro, como no exemplo “consultar um livro sobre história natural”, onde “história natural” é a entidade livro.

Quadro 2 - Frases utilizadas relacionadas a intenção “Consultar livro”.

Id Teste	Frase	Intenção
1	Estou interessado em consultar um livro.	Consultar livro
2	Quero saber mais sobre um livro.	Consultar livro
3	Gostaria de obter informações sobre um livro	Consultar livro
4	Estou procurando um livro específico.	Consultar livro
5	Você pode me ajudar a encontrar um livro?	Consultar livro
6	Estou em busca de um livro.	Consultar livro
7	Preciso de informações sobre o livro "O Leitor do Trem das 6h27".	Consultar livro
8	Gostaria de consultar o livro "Meditações" de Marco Aurélio.	Consultar livro
9	Pode me ajudar a encontrar um livro sobre geometria?	Consultar livro
10	Quero saber mais sobre o livro "A História Secreta".	Consultar livro
11	Quero consultar um livro	Consultar livro
12	Estou interessado em um livro sobre a Revolução Francesa	Consultar livro
13	Consultar um livro que detalha civilizações antigas	Consultar livro
14	Consultar um livro que aborda teoremas matemáticos	Consultar livro
15	Consultar um livro sobre a história da Renascença	Consultar livro

Fonte: Autor (2023)

O quadro 2 demonstra as frases geradas aleatoriamente pelo autor para a intenção de “Consultar livro”. Nela temos frases que se esperam serem comuns para

a intenção e são diversificadas em sua forma, podendo conter inclusive a entidade livro.

Quadro 3 - Frases utilizadas relacionadas a intenção “Consultar minhas reservas”

Id Teste	Frase	Intenção
1	Mostre-me os livros que peguei	Consultar minhas reservas
2	Quero conferir quais livros estou comigo no momento	Consultar minhas reservas
3	Posso ver a lista de livros que tenho atualmente?	Consultar minhas reservas
4	Eu gostaria de verificar os livros que estão comigo	Consultar minhas reservas
5	Mostre-me os livros que tenho alugados no momento	Consultar minhas reservas
6	Quais são os livros que reservei?	Consultar minhas reservas
7	Posso obter uma lista das minhas reservas atuais?	Consultar minhas reservas
8	Mostrar minhas reservas de livros.	Consultar minhas reservas
9	Posso verificar a reserva que fiz para o livro "Ecos do Futuro"?	Consultar minhas reservas
10	Pode me mostrar a reserva que fiz para o livro "Reflexos da Verdade"?	Consultar minhas reservas
11	Eu gostaria de verificar minhas reservas.	Consultar minhas reservas
12	Posso ver as reservas que fiz?	Consultar minhas reservas
13	Quero conferir a reserva do livro Os Inovadores em meu nome	Consultar minhas reservas

14	Consultar minha reserva do livro 'A Estrutura das Revoluções Científicas'	Consultar minhas reservas
15	Consultar minha reserva do livro 'A Arte da Guerra'	Consultar minhas reservas

Fonte: Autor (2023)

O quadro 3 mostra as frases para a intenção consultar minhas reservas, tais frases foram geradas aleatoriamente pelo autor, assim como na intenção anterior. Expressam formas comuns de frases utilizadas para consulta de reservas e tem diversas formas, tentam abranger o máximo de cenários possíveis.

O quadro 4 abaixo mostra as frases geradas aleatoriamente para as intenções de reservar um livro, estas frases variam no verbo, como reservar, alugar, solicitar de forma que a aplicação tenha várias formas para testar na plataforma de PLN.

Quadro 4 - Frases utilizadas relacionadas a intenção “Reservar um livro”

Id Teste	Frase	Intenção
1	Eu gostaria de solicitar a reserva de um livro	Reservar um livro
2	Eu gostaria de solicitar um livro	Reservar um livro
3	Eu gostaria de alugar um título específico	Reservar um livro
4	Estou interessado em alugar um livro	Reservar um livro
5	Gostaria de solicitar um exemplar de um livro	Reservar um livro
6	Desejo fazer a reserva de um livro.	Reservar um livro
7	Quero solicitar a reserva de "O Poder do Hábito".	Reservar um livro
8	Gostaria de alugar o livro "A Menina que Roubava Livros".	Reservar um livro
9	Estou interessado em solicitar "1984"	Reservar um livro

	de George Orwell.	
10	Queria solicitar um exemplar do livro "Pensando Rápido e Devagar".	Reservar um livro
11	Alugar livro sobre java	Reservar um livro
12	Reservar clean code	Reservar um livro
13	Reservar História da Arte Renascentista	Reservar um livro
14	Reservar Javascript iniciante	Reservar um livro
15	Eu estava esperando guardar um livro	Reservar um livro

Fonte: Autor (2023)

2.1 *CLU - Conversational Language Understanding*

Sobre o que é a CLU permite ao usuário fazer Microsoft (2023), permite aos usuários criarem modelos de linguagem natural, para predizer a intenção geral de uma expressão e extrair informações importantes sobre dele.

Tem como funcionalidades interpretar objetivos de usuários, extrair informações de frases de conversa, classificação de intenção, modelos de extração de entidades e suporte para 96 linguagens, Microsoft Azure (2024)

Segundo informações da Microsoft (2023) um dos exemplos de uso do CLU é de um bot de varejo personalizado para compras online sou encomendas de comida, fazendo com que o bot execute a ação desejada com base na intenção e nas informações extraídas.

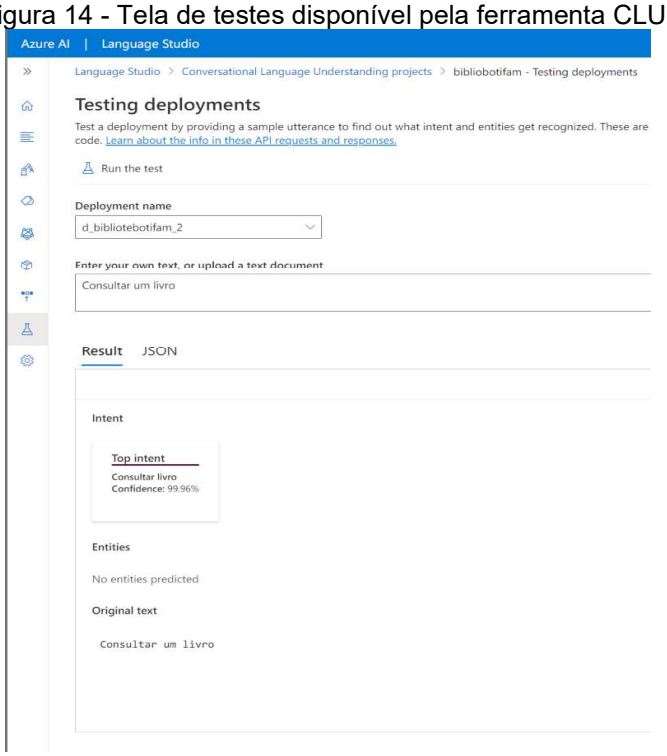
O CLU possui uma camada gratuita que permite aos usuários explorarem as funcionalidades da ferramenta sem qualquer custo inicial. Com uma quota de 5.000 registros de texto por mês, que equivale a 5.000 unidades de 1.000 caracteres cada, os usuários podem avaliar a eficácia e a precisão da ferramenta na análise e processamento de texto. Isso é útil para estudantes, pesquisadores ou pequenas

empresas que estão apenas começando a explorar as possibilidades da Inteligência Artificial e do Processamento de Linguagem Natural.

Além da camada gratuita, existem outras camadas tarifárias que permitem aos usuários acessarem uma maior capacidade de processamento e funcionalidades adicionais, conforme as suas necessidades crescem.

Para realização dos testes na ferramenta podemos utilizar o ambiente em nuvem que é oferecido pela plataforma, ou usando uma biblioteca de comunicação do .NET chamada *Azure.AI.Language.Conversations*. A figura 11 exibe a tela que será feito os testes na plataforma da ferramenta CLU, nela podemos inserir nosso texto de teste e recebemos um resultado no campo "*result*", da intenção e se houver, uma entidade contida dentro da frase inserida pelo aluno.

Figura 14 - Tela de testes disponível pela ferramenta CLU.



Fonte: Autor (2023)

A Figura 14 temos a tela da ferramenta CLU que será utilizada para rodas os testes, nesta tela falamos qual *deployment* será usada, e podemos colocar um texto de input em que será extraído as intenções e entidades.

O quadro 5 apresenta os registros dos experimentos realizados com a ferramenta *CLU - Conversational Language Understanding*, contendo o ID da frase, as intenções em testes com seu resultado, se está correta ou não.

Quadro 5 - Resultados da ferramenta CLU.

Id da frase	Consultar livro	Consultar minhas reservas	Reservar um livro
1	Correto	Correto	Correto
2	Correto	Correto	Correto
3	Correto	Correto	Correto
4	Correto	Correto	Correto
5	Correto	Correto	Correto
6	Correto	Correto	Correto
7	Errado, retornou “Consultar minha reserva”	Correto	Errado, retornou “Consultar minha reserva”
8	Errado, retornou “Consultar minha reserva”	Correto	Errado, retornou “Consultar minha reserva”
9	Correto	Correto	Errado, retornou “Consultar minha reserva”
10	Errado, retornou “Consultar minha reserva”	Correto	Errado, retornou “Consultar minha reserva”
11	Correto	Correto	Correto
12	Incorreto. Retornou intenção “Reservar um livro”	Correto	Correto
13	Correto	Correto	Correto
14	Correto	Correto	Correto

15	Correto	Correto	Correto
Total	80%	100%	73%

Fonte: Autor (2023)

De acordo com o quadro 5, a ferramenta CLU apresentou taxa acima dos 70% de acurácia em identificar as intenções do usuário, obtendo um total de 82% dos 45 testes.

Na série de testes voltada para a consulta de livros, foi registrada uma taxa de acerto de 80%, com alguns erros associados à identificação das intenções dos usuários. No que tange à intenção de "consultar minhas reservas", todos os testes foram bem-sucedidos. Já para a intenção de "reservar um livro", a acuracidade observada foi de 73%, indicando erros na identificação da intenção.

É perceptível que ocorreram erros associados à identificação da intenção do usuário; no entanto, na extração de entidades, como no caso de títulos de livros, não foi registrado nenhum erro. Esses resultados sugerem que a ferramenta pode se beneficiar de um conjunto de dados de treinamento mais robusto, para assim antecipar uma gama mais ampla de expressões utilizadas pelos usuários e identificar suas intenções de forma precisa.

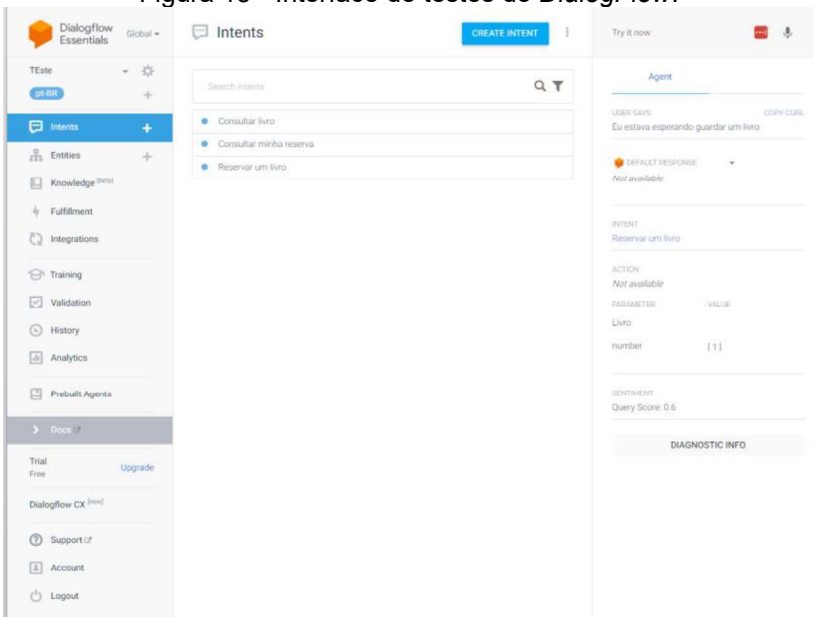
Um exemplo de erro pode ser observado na frase "Estou interessado em um livro sobre a Revolução Francesa", onde a intenção foi erroneamente identificada como "reservar um livro". Pressupõe-se que, ao fornecer mais exemplos de testes para a ferramenta, é possível alcançar taxas de acuracidade ainda maiores.

2.2 *Dialog Flow*

A ferramenta *Dialog Flow* possui edições até o presente momento, sendo *DialogFlow* CX e *DialogFlow* ES. Neste exemplo usamos a edição ES por oferecer uma camada gratuita de avaliação, além de ser a versão padrão e que atende a maioria dos casos para os projetos.

Para testarmos o *DialogFlow*, foi utilizado a interface oferecida pela ferramenta que permite uma forma intuitiva de testar as intenções de usuário. A Figura 15 exibe um exemplo de teste com uma frase e a intenção retornada na interface.

Figura 15 - Interface de testes do *DialogFlow*.



Fonte: Autor (2023)

Id da frase	Consultar livro	Consultar minhas reservas	Reservar um livro
1	Correto	Correto	Correto
2	Correto	Correto	Correto
3	Correto	Correto	Correto
4	Correto	Correto	Correto
5	Correto	Correto	Correto
6	Correto	Correto	Correto
7	Correto	Correto	Correto, porém não achou a entidade do livro
8	Correto, porém	Correto	Correto, porém

Quadro 6 - Resultados da ferramenta *DialogFlow*.

	não acho a entidade livro		não achou a entidade do livro
9	Correto, porém não achou a entidade livro	Correto, porém não achou a entidade do livro	Correto, porém não achou a entidade do livro
10	Correto	Correto, porém não achou a entidade do livro	Correto, porém não achou a entidade do livro
11	Correto	Correto	Correto
12	Correto	Correto	Correto
13	Correto	Correto	Correto
14	Correto	Correto	Correto
15	Correto	Correto	Correto
Total	100%	100%	100%

Fonte: Autor (2023)

De acordo com o quadro 6, a ferramenta *DialogFlow* obteve um resultado de 100% para a identificação das intenções do usuário para todas as intenções testadas, sendo assim, a ferramenta atingiu a meta de estar acima de 70**%**.

A ferramenta, porém, não foi eficaz na identificação das entidades em cada frase, como no exemplo “Gostaria de alugar o livro A Menina que Roubava Livros”, nesta frase, a ferramenta não conseguiu extrair a entidade livro “A menina que roubava livros".

A identificação de entidades, vai ser importante no processo de consulta de livro e reserva de um livro, onde o usuário pode numa frase colocar um título do livro e pesquisá-la diretamente.

A ferramenta *Dialog* Flow tem duas versões disponíveis, a primeira é chamada de ES e a outra CX. A versão ES é recomendada para projetos mais

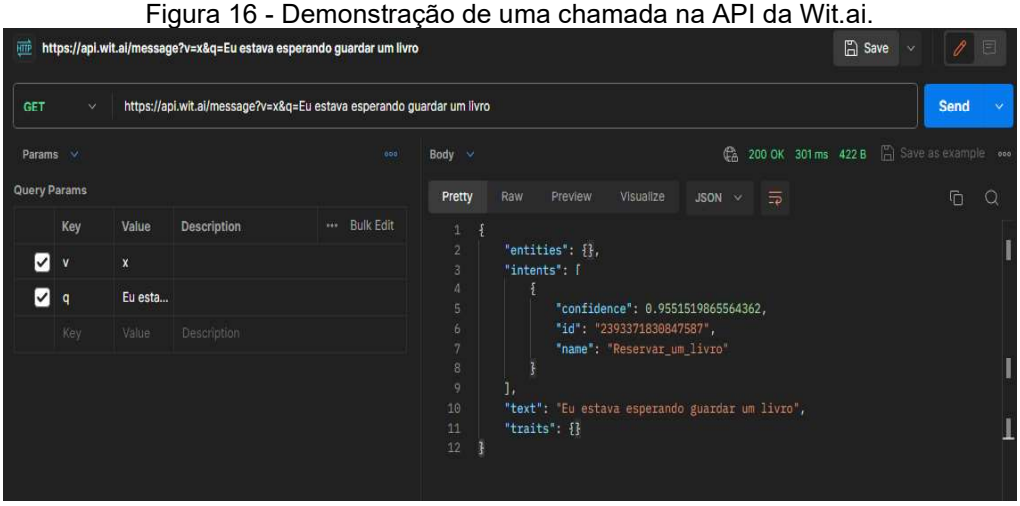
modestos e atende a maior parte das finalidades, possui uma versão de teste, com cota limitada para uso que foi a versão que utilizamos. A versão CX é uma versão mais avançada que possui funcionalidades apropriadas para projetos mais complexos.

A versão ES tem um custo de US$ 0,002 por solicitação na versão mais básica, dessa forma é importante salientar que se trata de uma ferramenta que apesar de possuir uma versão de teste da ES, ela possui uma cota limitada que pode expirar e não servir para projetos maiores.

2.3 Wit.ai

Para testarmos a ferramenta Wit.ai podemos utilizar tanto o portal, quanto utilizar uma API disponibilizada pela ferramenta junto a um código de identificador único para utilização.

Figura 16 - Demonstração de uma chamada na API da Wit.ai.



Fonte: Autor (2023)

Na Figura 16, temos a interface do programa *Postman*, este programa é usado para fazer testes em *APIs restfull,* no caso estamos chamando a API da Wit.AI e usando a rota *message*, onde passamos o texto “Eu estava esperando guardar um livro”, é visto no lado direito o retorno da API, onde temos a intenção identificada e qual score de aproximação.

Quadro 7 - Resultados da ferramenta Wit.ai

Id da frase	Consultar livro	Consultar minhas reservas	Reservar um livro
1	Correto	Correto	Correto
2	Correto	Correto	Correto
3	Correto	Correto	Correto
4	Correto	Correto	Correto
5	Correto	Correto	Correto
6	Correto	Correto	Correto
7	Correto	Correto	Correto
8	Correto	Correto	Correto
9	Correto	Correto	Correto
10	Correto	Correto	Correto
11	Correto	Correto	Correto
12	Correto	Correto	Correto
13	Correto	Correto	Correto
14	Correto	Correto	Correto
15	Correto	Correto	Correto
Total	100%	100%	100%

Fonte: Autor (2023)

De acordo com os resultados mostrados no quadro 7, a ferramenta apresentou bons resultados atingindo 100% de acerto nos testes de identificação de intenções de usuário. Além disso, a detecção de entidades pela ferramenta foi satisfatória, reconhecendo a entidade “livros” nas frases testadas.

Além disso, um ponto positivo da ferramenta é que ela aparenta ser totalmente gratuita, inclusive para uso comercial. E pode ser integrada ao *back-end* do chatbot através do uso de APIs.

2.4 Conclusão sobre os experimentos das PLN's

Todas as ferramentas que testamos apresentaram bons resultados na identificação de intenções de cada frase, atingindo uma taxa de acerto maior que 70%.

Notou-se que a ferramenta Wit.AI apresentou o melhor conjunto de resultados, além de ter atingido 100% na eficácia das intenções nas frases, obteve também um bom resultado na identificação das entidades contidas em uma frase, e possui uma maneira fácil de se integrar no serviço do nosso bot.

2.5 Tecnologias utilizadas

A seguir são descritas as tecnologias utilizadas para implementar a aplicação protótipo deste trabalho.

2.5.1 React.js

Conforme explicado por Sousa e Gonçalves (2020), A popularidade dos *frameworks Javascript* fez surgir o aparecimento de várias com o objetivo de ajudar nas tarefas de desenvolvimento.

Segundo Souza e Gonçalves (2020) "O React (também denominado React.js ou ReactJS) é uma biblioteca JavaScript de código aberto com enfoque na criação de interfaces de utilizadores (frontend) em páginas web"

Outro ponto forte do React.js é que permite o código ser modular e permite a criação de aplicações web complexas. O React é uma biblioteca para criação de interfaces modulares, ele permite o desenvolvimento de aplicações web grande e

complexas que podem mudar seus dados sem precisar atualizar a página (AGGARWAL, 2018).

Utilizamos o React.js em no projeto como o principal framework para construir as interfaces de usuário. O principal motivo para essa escolha foi a facilidade na modularização do projeto *front-end*, permitindo criar componentes visuais reutilizáveis na interface do usuário. Essa abordagem de componentes promove uma estrutura de código mais organizada e facilita a manutenção, uma vez que cada componente pode ser desenvolvido e testado de forma independente. Ainda sobre componentes, Freire (2021) diz, permite o reaproveitamento do código e facilita na manutenção e evolução do código.

Conforme informações do site oficial React (2024), a biblioteca permite a criação de interfaces de usuários através de peças individuais chamadas componentes, permite a criação de componentes como LikeButton e Video e então pode combinar em uma página inteira, telas e aplicativos.

“É mantido pelo Facebook, Instagram, outras empresas, bem como uma comunidade de desenvolvedores individuais” (SOUZA; GONÇALVES, 2018).

A adoção do React.js em nosso projeto não apenas impulsionou a eficiência do desenvolvimento, mas também resultou em uma aplicação mais robusta, escalável e de fácil manutenção. É uma tecnologia bastante utilizada pelo mercado e com bastante documentação. A capacidade de criar componentes reutilizáveis e a facilidade de integração com outras ferramentas e bibliotecas foram cruciais para atingir os objetivos do projeto.

2.5.2 .NET

Para nossos serviços *back-end*, foi optado pela linguagem de programação C#, utilizando o consolidado framework .NET.

O .NET permite uma estrutura gratuita, software livro e multiplataforma, permite criar aplicativos em nuvem, escaláveis, resilientes que podem executar nos principais serviços de nuvem (MICROSOFT, 2024)

Esta escolha foi estratégica, considerando a robustez e a versatilidade que o .NET oferece, além de sua excelente integração com uma variedade de serviços e tecnologias. Em particular, para a aplicação de *chat*, o SignalR do .NET se mostrou uma ferramenta indispensável, permitindo a implementação de comunicação em tempo real de forma eficiente e escalável.

C# é uma linguagem de programação de propósito geral, segura em termos de tipos e orientada a objetos. O objetivo da linguagem é a produtividade do programador. Para isso, C# equilibra simplicidade, expressividade e desempenho (ALBAHARI, 2022).

A plataforma .NET e a linguagem de programação C# foram oficialmente introduzidas em 2002 e rapidamente se tornaram um pilar no desenvolvimento moderno (TROELSEN; JAPIKSE, 2022)

Além disso, a adoção do .NET abre as portas para um ecossistema rico em bibliotecas e ferramentas externas, como o Elasticsearch, que utilizamos para funcionalidades avançadas de busca e análise de dados. O .NET também possui uma documentação extensiva disponível que acelera nosso ciclo de desenvolvimento e permite entregar soluções mais robustas e eficazes. A linguagem C# em si, com sua sintaxe clara e recursos modernos, complementa esse cenário, proporcionando um ambiente de desenvolvimento ágil e prazeroso.

O .NET oferece uma biblioteca de recursos de aprendizagem, possui vídeos, tutoriais e exemplos de código. Além disso possui um código fonte aberto, desenvolvido e mantido no *Github.* (MICROSOFT, 2024)

Optar pelo .NET e C# para o desenvolvimento do serviço *back-end*, portanto, não só atendeu às necessidades imediatas para a aplicação de chat, mas também posicionou estrategicamente para o futuro, garantindo ter uma plataforma confiável,

escalável e moderna, pronta para integrar as últimas inovações tecnológicas e atender às demandas crescentes do projeto.

2.5.3 Elasticsearch

O Elasticsearch é um motor de pesquisa e análise em tempo real. Permite-lhe explorar os dados a uma velocidade e a uma escala muito altos. É utilizado para pesquisa de texto integral, pesquisa estruturada, analítica e as três em combinação (GORMLEY; TONG, 2015)

No cenário de bibliotecas, onde a eficiência na recuperação de livros é primordial, o Elasticsearch se destaca como uma solução poderosa e escalável.

Sobre a construção do Elasticsearch Gormley e Tong (2015) um motor de busca *open source* e criado em cima do *Apache Lucene* uma biblioteca de motores de pesquisa de texto completo. O Lucene é indiscutivelmente a biblioteca de motor de pesquisa mais avançada, de elevado desempenho e com todas as funcionalidades atualmente existente - tanto de código aberto como proprietária.

“Ele armazena seus dados centralmente para proporcionar busca rápida, relevância com ajuste fino e analítica poderosa que pode ser ampliada com facilidade” (ELASTIC, 2024).

Exemplos de uso para o Elasticsearch são o Wikipedia utiliza para provê pesquisa de texto completo, com *snippets* de pesquisa e sugestões conforme você digita e o que você quis dizer (GORMLEY; TONG, 2015). Esta funcionalidade é de achar um livro baseado no que você quis dizer, é o que utilizaremos no nosso *chatbot.*

Outro exemplo de uso do Elasticsearch, O *Stack Overflow* combina a pesquisa de texto integral com consultas de geolocalização e o utiliza para encontrar perguntas e respostas relacionadas (GORMLEY; TONG, 2015).

Ainda sobre as funcionalidades do Elasticsearch o site oficial Elasticsearch (2024) diz “Permite realizar e combinar muitos tipos de buscas, estruturadas, não estruturadas, geográficas, métricas”.

A arquitetura da ferramenta do Elasticsearch permite que ele seja extremamente rápido e possa se integrar a uma vasta gama de linguagens de programação. Segundo Shah, Willick e Mago (2018).

> Sobre o uso do Elasticsearch Quando os utilizadores pretendem armazenar dados no Elasticsearch, eles fazem num índice. Um índice no Elasticsearch é um local para armazenar e organizar documentos relacionados. Eles não precisam ser todos do mesmo tipo de dados (ATHICK; BANON, 2022).

Portanto, a adoção do Elasticsearch para a busca de livros em um banco de dados vetorial é uma decisão estratégica que se alinha com a de busca eficiente dos livros que estão na base de dados, garantindo resultados de busca rápidos e relevantes, além de oferecer uma plataforma capaz de crescer e se adaptar às necessidades futuras de qualquer biblioteca digital.

2.5.4 SignalR

Sobre a funcionalidade de tempo real Microsoft (2024) “É a capacidade de ter o conteúdo de *push* de código do lado do servidor para os clientes conectados, pois acontece em tempo real.”

Permite a adição de funcionalidades em tempo real para painéis de controle, mapas, jogos chats entre outros (MICROSOFT, 2024).

O SignalR mantém uma conexão persistente entre o servidor e os clientes permite o envio de notificações para todos os clientes ou para clientes específicos, utilizam chamadas de procedimento remoto ou *remote procedure calls* (RPCs) (VEMULA, 2017)

Com estes pontos, a adoção do SignalR em um projeto de chatbot para biblioteca de livros oferece uma série de benefícios significativos. Primeiramente, o SignalR é projetado para facilitar a comunicação em tempo real entre o servidor e o cliente. Isso é essencial para *chatbots*, pois permite interações instantâneas e dinâmicas com os usuários, simulando uma conversa natural e fluída.

Um sistema de mensagens instantâneas em tempo real baseado na Web pode ser desenvolvido utilizando o SignalR, uma biblioteca de código aberto (VEMULA, 2017)

O SignalR é uma biblioteca da Microsoft oferecem desenvolvimento *web* em tempo real para aplicações .NET. A SignalR é utilizada principalmente por aplicações que requerem notificações push do servidor para o cliente; por exemplo, aplicações como chat, mercado de ações, jogos e dashboards (VEMULA, 2017).

Com as explicações acima, o SignalR simplifica o processo de adição de funcionalidades de web em tempo real à aplicação, o que é crucial para uma experiência de chat fluida e responsiva. Ele abstrai as complexidades da comunicação bidirecional e assíncrona, permitindo concentrar na lógica de aplicação sem se preocupar com os detalhes de implementação de baixo nível. Além disso, possui integração nativa ao framework .NET.

Conforme a Microsoft (2023), o SignalR apresenta vantagens significativas, destacando-se pela sua escalabilidade para gerenciar muitas conexões simultâneas e pelo suporte multilíngue, além de proporcionar uma redução de custos operacionais ao eliminar a necessidade de gerenciamento próprio do SignalR.

Sobre o funcionamento do SignalR, Vemula (2017), ele usa *Hubs* para enviar notificações do servidor para o cliente e suporta vários canais, como *WebSocket*, eventos enviados pelo servidor e *long polling*. O SignalR suporta várias linguagens, desde C#/C++ até JavaScript.

Além disso, o SignalR suporta automaticamente uma variedade de técnicas de comunicação em tempo real, escolhendo a melhor opção disponível com base no

cliente e no servidor. Isso significa que ele pode utilizar *WebSockets*, que é uma tecnologia avançada para comunicação em tempo real, mas também pode recorrer a métodos mais antigos, como *long polling*, quando *WebSockets* não está disponível. Essa flexibilidade garante que o chatbot possa operar de maneira eficiente em diferentes ambientes e condições de rede.

Por fim, o SignalR é parte do ecossistema .NET, o que significa que ele se integra perfeitamente com outras tecnologias da Microsoft.

2.5.5 Banco de dados PostgreSQL

Segundo o site oficial PostgreSQL (2024). O PostgreSQL é um poderoso banco de dados objeto relacional com mais de 35 anos de desenvolvimento ativo, que tem uma reputação de ser confiável, ter robustez de funcionalidades e desempenho.

Foi escolhido para ser o banco de dados relacional da aplicação. O PostgreSQL é conhecido por pela robustez, além de utiliza o padrão SQL, o que garante uma base para o armazenamento e a recuperação de dados. Sua natureza de código aberto oferece flexibilidade e uma comunidade ativa, que constantemente contribui com atualizações e melhorias.

Um ponto positivo para o uso do PostgreSQL é sua natureza *open source,* é totalmente desenvolvido neste ambiente *open source*, o que significa que não há uma entidade central responsável pelo projeto e o resultado é que o PostgreSQL não é um produto comercial (FERRARI; PIROZZI, 2020).

Na aplicação será usado como base dados que guardará as permanentemente as informações dos livros, o sistema de gerenciamento de reservas, os alunos e o sistema de autenticação dos alunos.

Sobre a performance Pilicita Garrido, Borja López e Gutiérrez Constante (2021) afirmam que o PostgreSQL é altamente valorizado por sua "estabilidade, potência, robustez e facilidade de administração e implementação", e destaca-se

pelo uso de um sistema cliente-servidor que utiliza threads para "um processamento correto das consultas à base de dados".

O PostgreSQL é uma escolha econômica para projetos que requerem um sistema de gerenciamento de banco de dados. Sendo uma solução de código aberto, o PostgreSQL é completamente gratuito para uso, o que significa que não há custos de licenciamento associados ao seu uso, independentemente do tamanho do projeto ou da organização. Isso pode representar uma economia significativa, especialmente em comparação com soluções de banco de dados proprietárias que podem exigir investimentos em licenças. A natureza de código aberto do PostgreSQL não apenas elimina custos de licença, mas também oferece a liberdade de modificar e personalizar o banco de dados conforme necessário.

Sobre a capacidade operacional do PostgreSQL Ferrari e Pirozzi (2020) discutem que como um banco de dados relacional, o PostgreSQL oferece muitos recursos, e é muito difícil ter problemas com falta de espaço uma instância do PostgreSQL. Uma única instância pode conter por exemplo mais de 4 bilhões de bancos de dados individuais.

CAPÍTULO 3: UM CHATBOT PARA APRIMORAR A RELAÇÃO ALUNO BIBLIOTECA

Neste capítulo, será exibido a proposta de anteprojeto para o desenvolvimento de um chatbot para a biblioteca do Instituto Federal do Amazonas. O objetivo deste projeto é melhorar a relação entre os alunos e a biblioteca, tornando mais eficiente o processo de reserva de livros, o gerenciamento dessas reservas e a consulta de livros.

Iniciaremos expondo os requisitos funcionais e não funcionais da aplicação. Em seguida, será abordado a modelagem do sistema, incluindo o diagrama de casos de uso e uma descrição detalhada de cada caso de uso, além de apresentar o diagrama de classes relacionado.

Além disso, será fornecido uma visão geral do funcionamento da aplicação, apresentando um esboço da arquitetura geral do sistema e destacando pontos-chave no código de implementação da aplicação.

Com essa estrutura, busca-se apresentar de forma clara e organizada todos os aspectos essenciais de nosso projeto, desde os requisitos até a implementação prática, visando aprimorar a experiência dos alunos ao utilizar os serviços da biblioteca.

3.1 Especificação dos requisitos

Nesta seção serão apresentados os diagramas de casos de uso e de atividades que foram elaborados para orientar o desenvolvimento da aplicação proposta.

3.1.1 Requisitos Funcionais

No Quadro 9 são apresentados os requisitos funcionais definidos para o presente trabalho. Tais requisitos foram definidos pelo próprio autor, baseado em

experiências própria com bibliotecas, seja em institutos, bibliotecas públicas entre outras.

Estes requisitos serão descritos posteriormente no capítulo de descrição dos casos de uso mais abaixo.

Quadro 8 - Requisitos funcionais.

Código	Requisito	Descrição	Prioridade
RF01	Consultar livro	O sistema deve permitir ao usuário pesquisar um livro por uma descrição, autor ou título.	Essencial
RF02	Reservar um livro	O sistema deve permitir a reserva de livros.	Essencial
RF03	Consultar minhas reservas	O sistema deve retornar um histórico de livros reservados, tanto atuais quanto concluídas	Essencial
RF04	Autenticar	Deve permitir o aluno autenticar no sistema via matrícula ou e-mail e senha registrada	Essencial
RF05	Inserir livros	Deve permitir ao bibliotecário a inserção de livros na base de dados	Essencial

Fonte: Autor (2023)

3.1.2 Requisitos não funcionais

O Quadro 10 apresenta os requisitos não funcionais definidos para orientar o desenvolvimento do projeto proposto. Tais requisitos não funcionais também foram definidos pelo autor.

Quadro 9 - Requisitos não funcionais

Código	Requisito	Descrição	Prioridade
RNF01	Ambiente web	O sistema tem de ser disponibilizado em um ambiente web, a partir do uso de um navegador	Essencial

RNF02	Uso de IA para consultar livros	Permitir que o aluno entre com uma descrição para consulta de livro, e mesmo que não assertiva, retornar livros com a descrição aproximada.	Importante

Fonte: Autor (2023)

O requisito não funcional RNF01 o sistema deve ser hospedado e acessível em um ambiente web, garantindo que os usuários possam acessá-lo através de navegadores de internet comuns, como Google Chrome e Mozilla Firefox. Isso envolve a adaptação do sistema para ser responsivo e compatível com diferentes tamanhos de tela e dispositivos, garantindo uma experiência de usuário consistente e acessível independentemente do dispositivo utilizado.

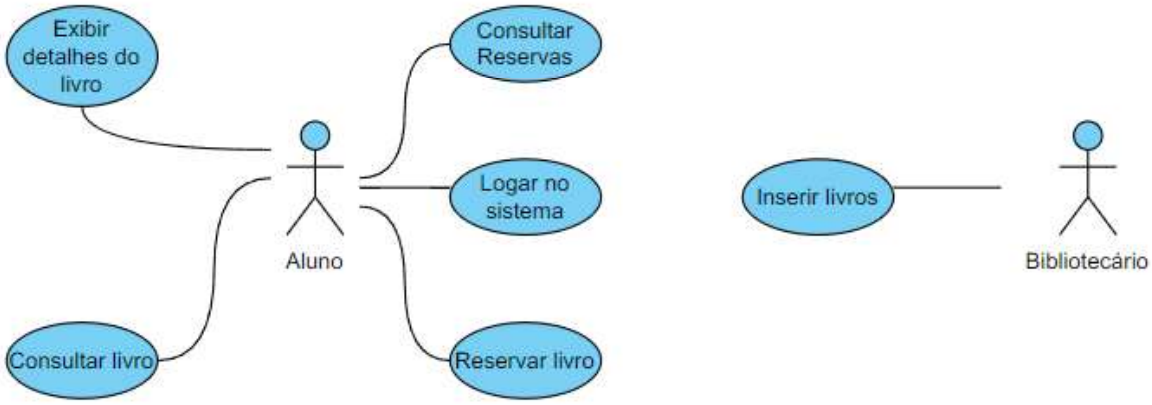
No requisito RNF02 o sistema deve incorporar um mecanismo de inteligência artificial para facilitar a busca de livros pelos alunos e na interação dos alunos com o chatbot. Esse mecanismo deverá permitir que os alunos insiram descrições, mesmo que vagas ou imprecisas, sobre os livros que desejam encontrar. O sistema, então, utilizará algoritmos de processamento de linguagem natural e técnicas de aprendizado de máquina para interpretar essas descrições e sugerir livros que correspondam ou se aproximem do que foi descrito. O objetivo é fornecer resultados relevantes mesmo na ausência de detalhes específicos ou palavras-chave exatas. A solução deve ser otimizada para lidar com uma ampla gama de consultas, garantindo precisão e relevância nas sugestões de livros, além de oferecer uma interface de usuário intuitiva para facilitar a interação dos alunos com o sistema de busca.

3.2 Modelagem do sistema

Nesta seção serão apresentados os diagramas de casos de uso e de atividades que foram elaborados para orientar o desenvolvimento da aplicação proposta.

3.2.1 Diagrama de casos de uso

Figura 17 - Diagrama de caso de uso do sistema. *s*



Fonte: Autor (2023)

Na Figura 17 temos o diagrama de caso de uso para o *chatbot*. Nela temos dois atores, o aluno e o bibliotecário. O aluno é responsável pela interação com a interface do nosso chatbot, ele pode realizar os casos de uso como “Consultar Reservas”, logar no sistema, reservar livro, consultar livro e exibir os detalhes de um livro.

Para o ator bibliotecário, temos o caso de uso “inserir livros”, esse caso de uso será responsável pela inserção de livros na base de dados do chatbot e deixá-los disponíveis para o uso nos outros casos de uso. Será disponibilizado uma API onde o bibliotecário deverá chamar uma rota do nosso serviço do chatbot e preencherá os dados do livro, ao fazer a requisição, o livro será inserido na base de dados.

3.2.2 Descrição do caso de uso

3.2.2.1 Caso de uso consultar livro

Objetivo: O usuário deve ser capaz de consultar um livro

Requisitos: RF01

Pré-condições: Usuário deve estar logado no sistema.

Pós-condições: O usuário deve receber uma lista de livros baseado na descrição dada pelo usuário na consulta.

Fluxo principal:

1. Usuário loga no sistema
2. Usuário digita “consultar livro” ou alguma mensagem com esta intenção.
3. O Usuário informa qual livro deseja consultar, por uma descrição, título ou ator, não precisando ser a descrição exata, conforme a Figura 18.
4. É exibido uma lista de livros baseada na descrição informada pelo aluno, como mostrado na Figura 18.
5. Também é exibido as ações que o usuário pode tomar para as opções de livros mostradas, sendo “Reservar” ou “Visualizar”, conforme a Figura 19.

Fluxo Alternativo:

1. O serviço identifica que não foram encontrados livros baseados na descrição informado pelo aluno, como mostrado na Figura 20.
   a. Pergunta ao usuário se deseja pesquisar novamente
   i. Se sim, aluno deve pesquisar novamente pela descrição do livro
   ii. Se não, retorna usuário a uma mensagem inicial

Extensões

1. Reservar um livro
   a. Após a etapa 3, o aluno tem opção de reservar um livro ao selecionar uma das opções disponibilizadas pelo sistema.
2. Exibir detalhes do livro
   a. Após a etapa 3, o aluno tem opção de ver detalhes do livro ao selecionar uma das opções disponibilizadas pelo sistema.

Abaixo a Figura 18, demonstra um fluxo de consultar um livro, onde o usuário seleciona a opção de “Consultar um livro” e o sistema pergunta qual livro, o usuário entra com uma descrição sobre o livro que deseja pesquisar.

Na Figura 19 temos a exibição das opções disponíveis de livros na base de dados baseados na descrição que o usuário digitou no *chatbot*, nas opções também temos ações atreladas, como: “Visualizar” que mostra os detalhes do livro e “Reservar” que aluga um livro.

Na Figura 20 temos uma tela do fluxo de exceção para caso um livro com a descrição não seja achado, é exibido a mensagem “Nenhum livro foi encontrado, quer tentar novamente? ”.

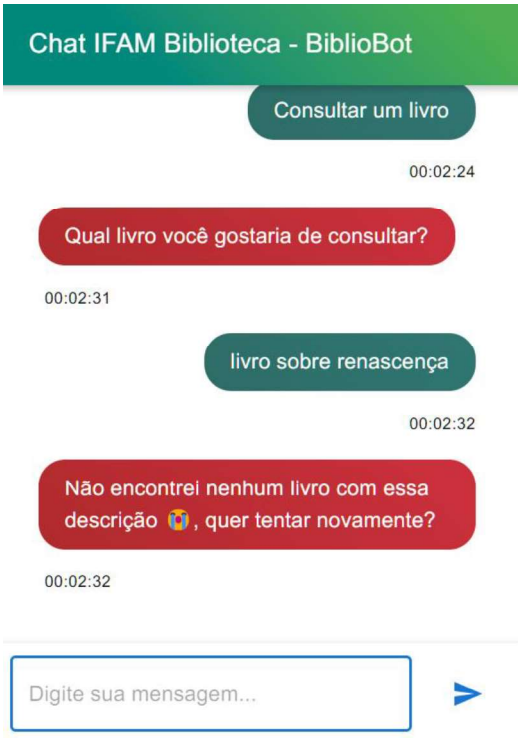
Figura 18 - Tela de demonstrando fluxo de consultar um livro.

Fonte: Autor (2023)

Figura 19 - Exibição das opções achadas pelo sistema, baseado na descrição dada pelo usuário no fluxo de consultar livro.

Fonte: Autor (2023)

Figura 20 - Mensagem alertando que o livro não foi encontrado e que o aluno pode tentar novamente.



Fonte: Autor (2023)

3.2.2.2 Caso de uso Logar no Sistema

Objetivo: O usuário deve ser capaz de logar no sistema

Requisitos: RF04

Pré-condições: Usuário deve ter registro no sistema.

Pós-condições: O usuário deve receber uma mensagem de boas-vindas e receber as opções disponíveis.

Fluxo principal:

1. O Aluno informa e-mail ou matrícula e senha, na tela de login, conforme a Figura 21 .

Figura 21 - Tela de login.

Fonte: Autor (2023)

2. O sistema deve realizar a validação das informações de login

3. O usuário recebe um feedback de que foi logado e é exibido mensagem de boas-vindas e serviços disponíveis, conforme a Figura 22.

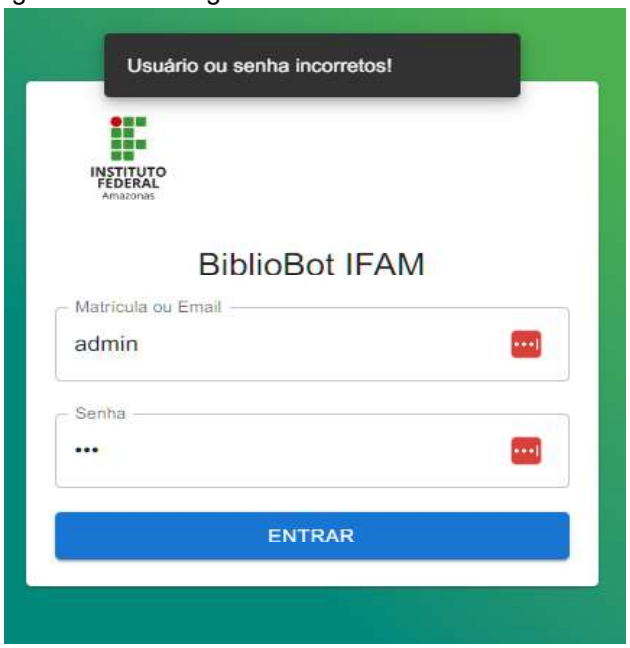
Figura 22 - Tela de boas-vindas, após login do usuário.

Fonte: Autor (2023)

Fluxo Alternativo:

1. Usuário informa e-mail ou matrícula e senha inválidas

Figura 23 - Mensagem de usuário ou senha incorretos.



Fonte: Autor (2023)

a. Sistema retorna uma mensagem dizendo que login foi inválido, conforme Figura 23.

3.2.2.3 Caso de uso reservar livro

Objetivo: O Aluno deve conseguir reservar um livro disponível.

Requisitos: RF02

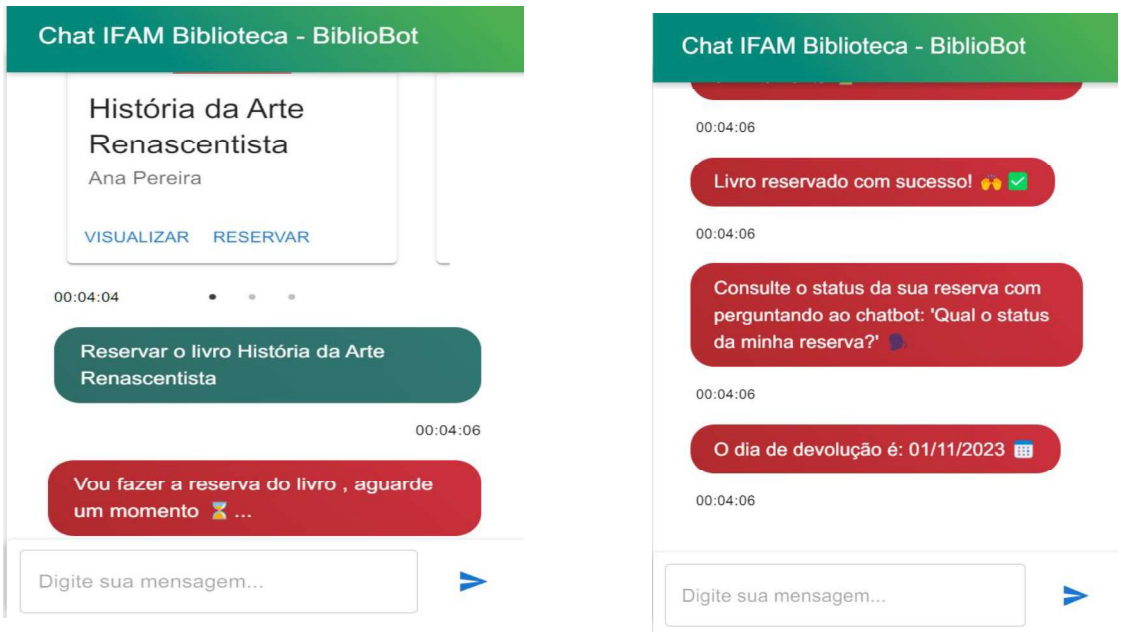
Pré-condições: Aluno deve estar logado no sistema.

Pós-condições: O usuário deve receber uma mensagem com os detalhes da reserva com a data de expiração da reserva, ou data de entrega.

Fluxo principal:

1. O aluno digita frases semelhantes a “Reservar um livro”.
2. Incluir: **Consultar um livro**
3. O aluno escolhe uma opção.
4. O sistema informa que está reservando o livro.
5. O sistema informa que a reserva foi feita, e que o aluno deve consultar suas reservas para verificar o status da reserva, além disso informa detalhes da reserva, como data de entrega ou data de expiração da reserva, conforme Figura 24.

Figura 24 - Tela mostrando a finalização da reserva do livro pelo aluno.



Fonte: Autor (2023)

Inclusões:

1. Consultar um livro

a. O sistema pergunta ao usuário qual descrição do livro que o mesmo deseja reservar

b. O sistema retorna uma lista de livros que o aluno deseja reservar

Fluxo Alternativo:

1. Aluno pesquisa por uma descrição de livro e não retorna um livro encontrado.

a. Na etapa 3 do fluxo principal, após o aluno buscar um livro baseado na descrição informada.

b. O sistema retorna que não encontrou um livro

c. O sistema pergunta se o usuário deseja pesquisar novamente.

i. Caso sim, o sistema solicita novamente a descrição do livro.

ii. Caso não, o sistema informa a mensagem inicial do sistema.

d. O sistema deve retornar as opções de livro, caso encontre os livros aproximados.

Exceções:

1. Livro já reservado pelo usuário

a. Na etapa 5 do fluxo principal, o aluno seleciona uma opção, porém ela já foi reservada pelo usuário.

b. O sistema exibe a mensagem de que o livro já foi reservado.

c. O sistema auxilia informando para o usuário olhar suas reservas de livros com a mensagem “Consultar minhas reservas”.

d. O sistema encerra o fluxo.

2. Livro indisponível para reserva.

a. Na etapa 5 do fluxo principal, o aluno seleciona uma opção, porém o livro selecionado se encontra indisponível para reserva, devido não haver estoque.

b. O sistema exibe mensagem de que o livro não está disponível para reserva.

c. O sistema auxilia o aluno a consultar outros livros.

d. O sistema encerra o fluxo.

3.2.2.4 Caso de uso consultar minhas reservas de livros

Objetivo: O Aluno deve conseguir ver detalhes de suas reservas de livros.

Requisitos: RF03

Pré-condições: Aluno deve estar logado no sistema.

Pós-condições: O aluno deve receber uma mensagem do sistema, com os livros reservados pelo usuário ativo, inativo ou em processamento.

Fluxo principal:

1. O Aluno digita frases semelhantes a “Consultar minhas reservas”
2. O sistema retorna para o aluno as reservas realizadas pelo aluno, com data da reserva, a data de expiração da reserva, o livro reservado e o status da reserva, sejam elas ativas, concluídas, ou em processamento.
3. Encerra o fluxo.

3.2.2.4 Inserir livros

Objetivo: O bibliotecário deve conseguir cadastro livros na aplicação através de uma API.

Requisitos: Ter credenciais de bibliotecário

Pós-condições: O bibliotecário recebe um retorno da API dizendo que o livro foi cadastrado.

Fluxo principal:

1. O bibliotecário preenche um objeto do tipo *JavaScript Object Notation* (JSON, em português Notação de Objetos Javascript) este tipo de notação é comumente usado para interagir com as APIs *REST* e pode ser passado pelo body da requisição através do método *POST*.
2. O preenchimento do objeto JSON no *body* da requisição deve ser no seguinte padrão conforme a reapresentação na Figura 25.

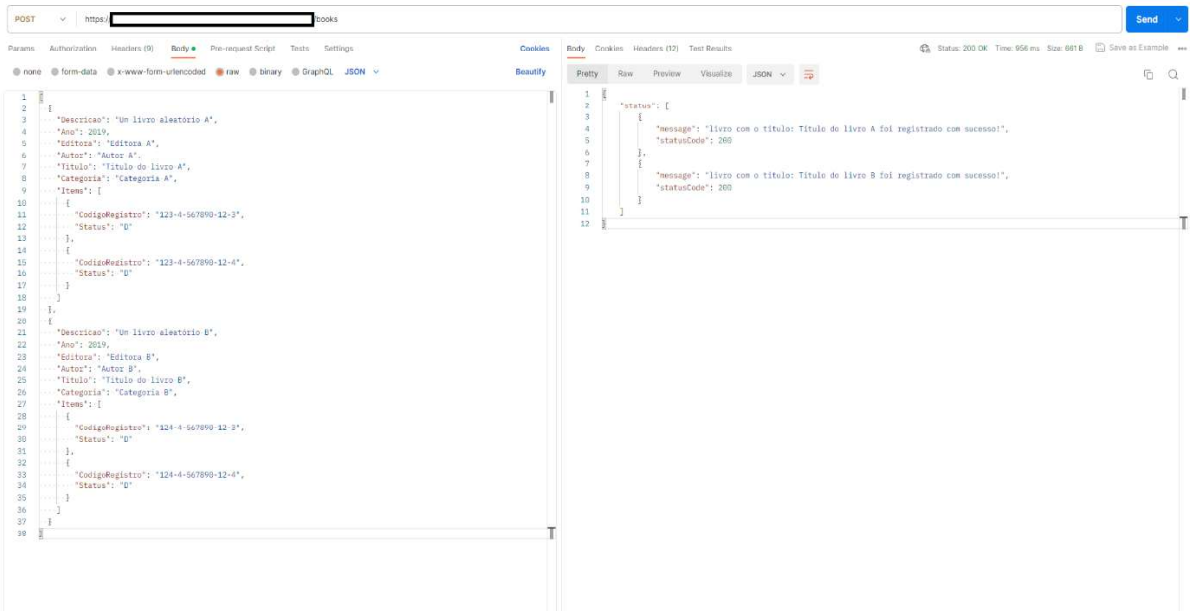
Figura 25 - Exemplo de body para a requisição de preencher o livro

```
[
  {
    "Descricao": "Um livro aleatório A",
    "Ano": 2019,
    "Editora": "Editora A",
    "Autor": "Autor A",
    "Titulo": "Titulo do livro A",
    "Categoria": "Categoria A",
    "Items": [
      {
        "CodigoRegistro": "123-4-567890-12-3",
        "Status": "D"
      },
      {
        "CodigoRegistro": "123-4-567890-12-4",
        "Status": "D"
      }
    ]
  }
]
```

Fonte: Autor (2023)

3. Após fazer a requisição para a rota *POST* de *Books* um retorno deverá ser obtido informando se houve sucesso ao inserir o livro na base, conforme o lado direito da Figura 26.

Figura 26 - Requisição realizada para rota de "Books" para preenchimento dos livros



Fonte: Autor (2023)

Exceções:

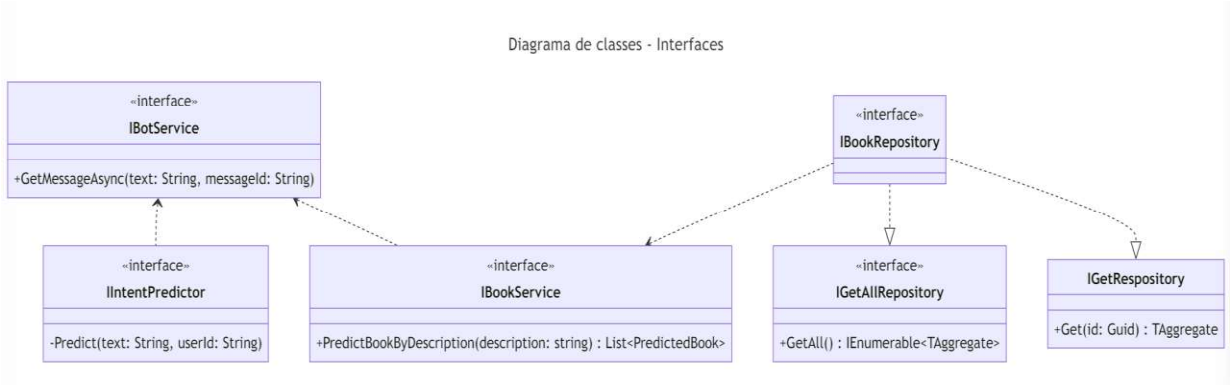
1. Livro falhou ao ser registrado na base

   a. Será retornado na resposta da requisição qual livro não foi preenchido e o *statusCode* com a numeração 400.

3.2.3 Diagrama de classes da aplicação

Os diagramas de classes foram separados em três tipos a fim de facilitar o entendimento da modelagem do sistema. Estes tipos são um diagrama mapeando as interfaces, um diagrama das entidades, modelos e um diagrama das classes concretas com os *frameworks* que foram utilizados.

Figura 27 - Diagrama de classes, comunicação das interfaces no sistema



Fonte: Autor (2023)

Na Figura 27, temos um diagrama de classes focado em interfaces, que são representações abstratas de serviços ou contratos que classes concretas devem implementar. Este diagrama ilustra a estrutura de um sistema com foco na definição de comportamentos através de interfaces, sem detalhar as implementações concretas.

A interface IBotService define um contrato com um método GetMessageAsync, que espera um texto e um identificador de mensagem como parâmetros. Esta interface será responsável por definir o método em que o usuário chamará da aplicação e esperara uma mensagem de retorno do *bot*.

A interface *IIntentPredictor* declara um método *Predict*, que recebe um texto e um identificador de usuário, retornando uma previsão. Essa interface pode ser usada para serviços de predição de intenções com base em entradas de texto como no caso do Wit.ai.

A interface *IBookService* expõe um método *PredictBookByDescription*, que aceita uma descrição e retorna uma lista de livros previstos. Esta interface seria a base para um serviço que oferece sugestões de livros com base em suas descrições.

A interface *IBookRepository* é uma especialização da interface IGetAllRepository, indicando uma relação de herança (generalização/especialização). Ela não declara nenhum método adicional, sugerindo que herda todos os métodos da *IGetAllRepository*.

A interface *IGetAllRepository* define um método *GetAll*, que retorna um enumerável de um tipo genérico *TAggregate*. Este é um contrato para repositórios que podem recuperar todos os objetos de um tipo genérico.

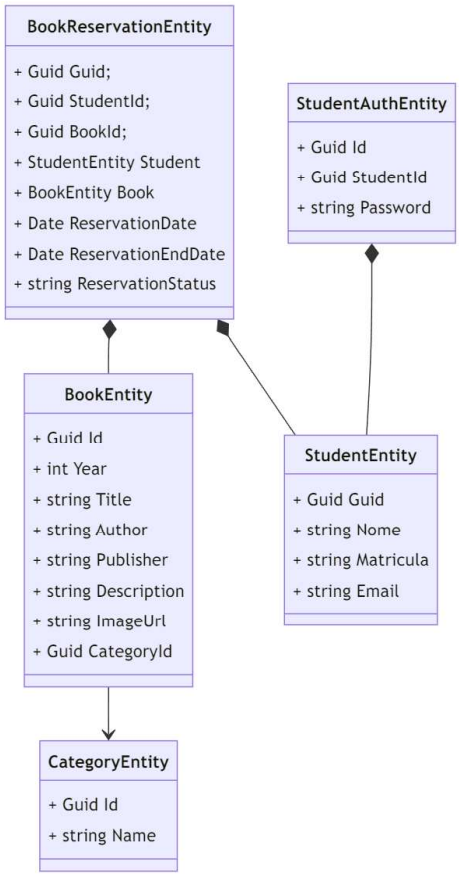
A interface *IGetRepository* especifica um método *Get*, que recebe um identificador único (Guid) e retorna um objeto do tipo TAggregate. Esta interface é um contrato para repositórios que fornecem uma maneira de recuperar objetos individuais por um identificador.

As setas tracejadas com pontas vazias representam uma relação de dependência, indicando que uma interface pode depender de outra para algumas operações, mas não especifica uma relação de propriedade ou de vida útil. A seta tracejada com ponta cheia entre IBookRepository e *IGetAllRepository* indica uma relação de herança, onde *IBookRepository* é uma extensão de *IGetAllRepository*, herda seus métodos e pode adicionar novos ou sobrescrever os existentes.

Na aplicação do *chatbot*, adotamos uma abordagem modular, inspirada nos princípios do SOLID, uma das principais metodologias de design de software. Esta decisão foi tomada para garantir que o sistema seja facilmente expansível, manutenível e robusto.

Como dito, foi utilizado os princípios do SOLID na aplicação, nota-se pela utilização das interfaces que apenas possuem métodos necessários para seus clientes, obedecendo assim o princípio ISP do SOLID. Assim como utilizamos outros princípios do SOLID, uma das mais aparentes são também o princípio da segregação de responsabilidade, onde cada classe tem apenas uma responsabilidade, e o princípio de aberto-fechado onde as classes são fechadas para modificações, mas a adoção de funcionalidades é permitida.

Figura 28 - Diagrama de classes dos modelos, entidades da aplicação



Fonte: Autor (2023)

Na Figura 28 o diagrama de classes fornecido descreve as entidades relacionadas a um sistema de reserva de livros em uma biblioteca. Cada classe é uma representação de uma entidade no sistema e contém propriedades específicas.

BookReservationEntity: Representa a reserva de um livro por um estudante. Contém identificadores únicos para a reserva (Guid), o estudante (StudentId) e o livro (BookId). Além disso, incluem instâncias das entidades StudentEntity e

BookEntity correspondentes, a data de início (ReservationDate) e término (ReservationEndDate) da reserva, e o status da reserva (*ReservationStatus*).

StudentAuthEntity: Representa as informações de autenticação de um estudante. Contém um identificador único (Id), o identificador do estudante (StudentId) e a senha (Password). Essa entidade é usada para controlar o acesso ao sistema.

BookEntity: Detalha a entidade que representa um livro, incluindo um identificador único (Id), ano de publicação (Year), título (Title), autor (Author), editora (Publisher), descrição (Description), URL da imagem (ImageUrl) e identificador da categoria (CategoryId).

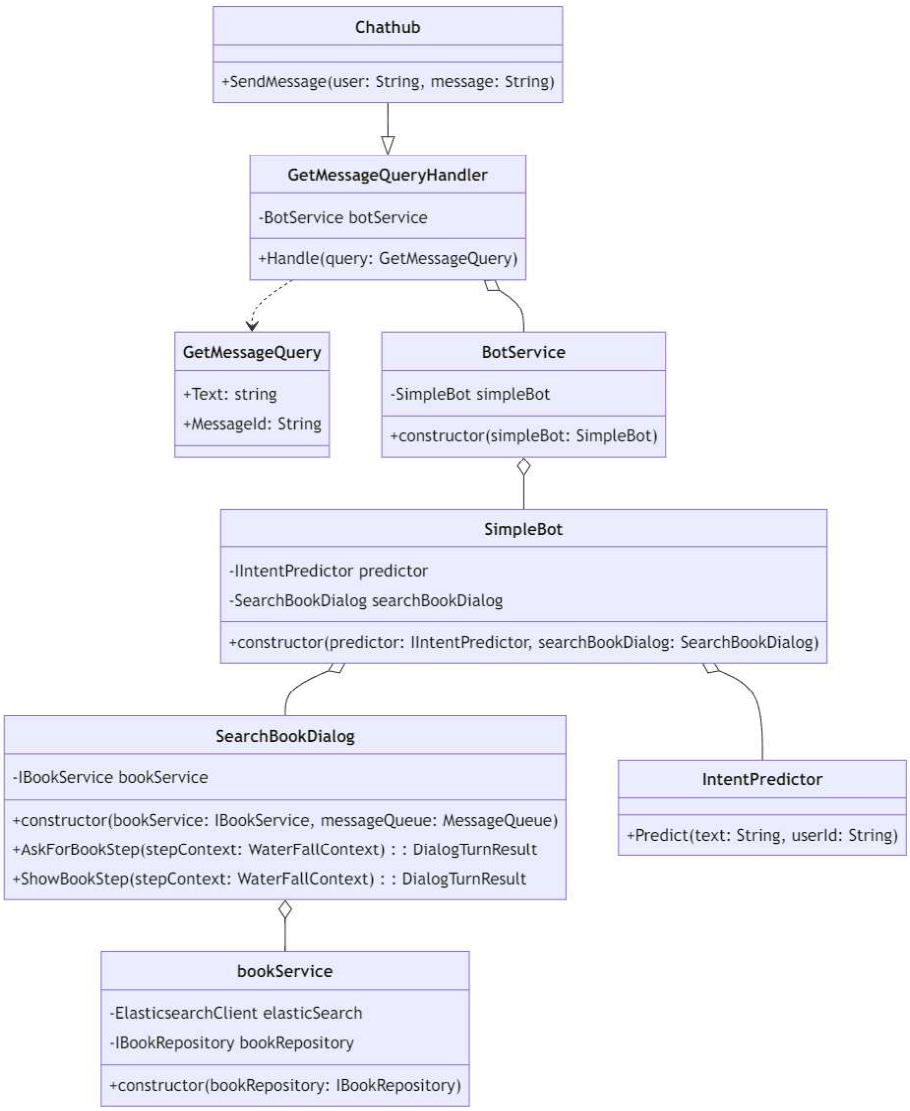
StudentEntity: Armazena informações sobre um estudante, com um identificador único (Guid), nome (Nome), número de matrícula (Matrícula) e e-mail (Email).

CategoryEntity: Descreve uma categoria de livro com um identificador único (Id) e um nome (Name).

É importante notar as relações entre os modelos, onde no BookReservationEntity tem uma relação de composição com BookEntity e StudentEntity, onde a classe BookReservationEntity não pode existir sem essas duas classes, o mesmo acontece com StudentAuthEntity que não pode existir sem a entidade StudentEntity

Esse diagrama é útil para entender como as reservas são gerenciadas no sistema e como as entidades estão inter-relacionadas. Por exemplo, para realizar uma reserva, o sistema precisa de informações sobre o livro e o estudante. As informações de autenticação são gerenciadas separadamente para garantir a segurança do processo de login do estudante.

Figura 29 - Diagrama de classes, módulos, classes concretas da aplicação



Fonte: Autor (2023)

Na Figura 29 o diagrama de classe apresenta a estrutura das classes concretas do nosso serviço, aqui foram separadas do diagrama de classes – interfaces para melhorar a visualização e podermos vermos suas relações.

Chathub é o ponto de entrada da aplicação, é uma extensão da classe Hub proveniente da biblioteca SignalR, serve como ponto de comunicação entre o aluno e o serviço de processamento do chatbot. O *ChatHub* não apenas define os métodos que a interface do usuário utiliza para enviar mensagens, mas também estabelece as regras para o roteamento das mensagens.

Uma vez que uma mensagem é enviada através do chat, ela é recebida pelo *GetMessageQueryHandler*. Este manipulador tem a tarefa crucial de processar a mensagem, o que envolve analisar seu conteúdo, interagir com outros serviços quando necessário e, finalmente, formatar e enviar uma resposta adequada ao cliente.

Para garantir uma arquitetura limpa e desacoplada, o *GetMessageQueryHandler* interage com o bot através da interface *IBotService*. Seguindo o Princípio da Segregação da Interface, parte dos princípios SOLID, esta abordagem garante que o manipulador não dependa de uma implementação específica, mas apenas de um contrato definido. A classe *BotService*, que implementa esta interface, é responsável por determinar o fluxo de interação com base na mensagem do usuário.

O *BotService*, por sua vez, delega a tarefa de decidir o fluxo de conversação ao *SimpleBot*. Esta classe é o cérebro da operação, decidindo se o usuário quer buscar um livro, fazer uma reserva ou consultar reservas existentes. Para tomar essa decisão, o *SimpleBot* utiliza o serviço de Processamento de Linguagem Natural (PLN), especificamente a interface *ICluIntentEntityPredict*, que analisa a mensagem do usuário e identifica sua intenção.

Um dos fluxos possíveis é a busca de livros, que é gerenciada pela classe *SearchBookDialog*. Esta classe guia o usuário através das etapas de busca, desde perguntar sobre o livro desejado até mostrar os resultados relevantes. Embora tenhamos usado a busca de livros como exemplo, é importante notar que temos várias outras classes de diálogo para atender a diferentes funcionalidades, como consulta de reservas e processo de reserva de livros.

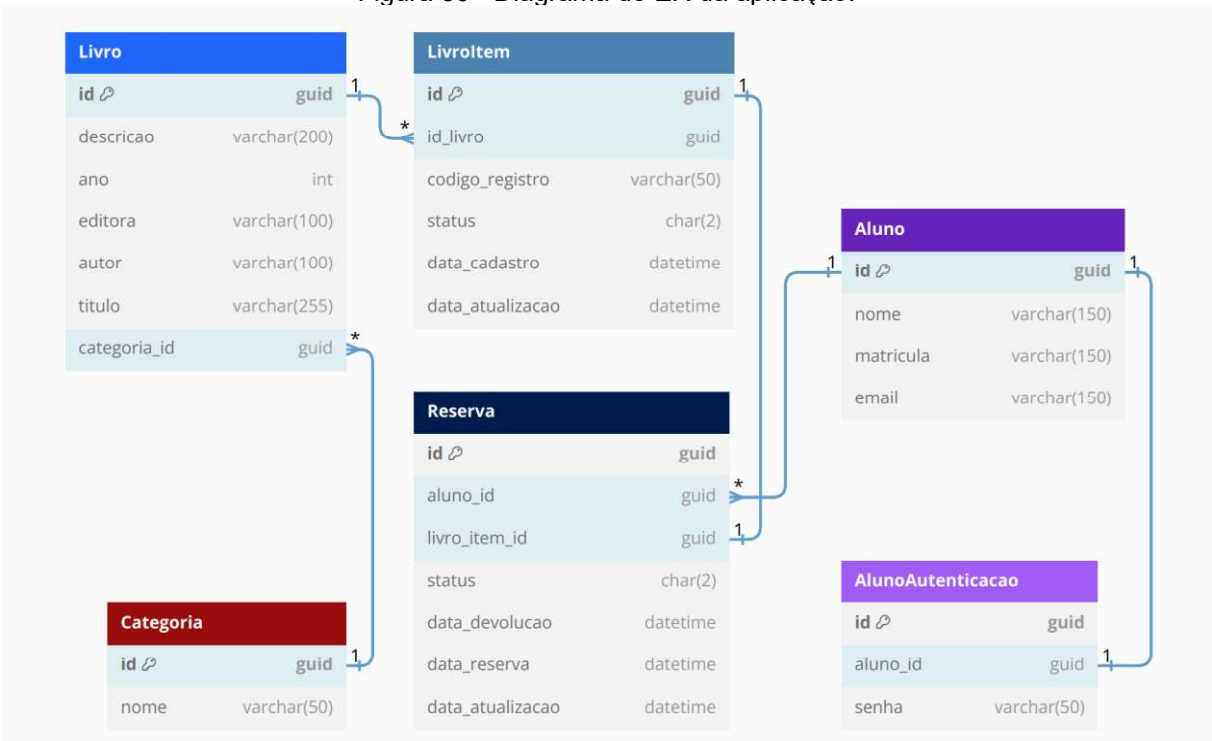
Para buscar livros, o *SearchBookDialog* depende do serviço *IBookService*. Este serviço procura na base de dados no ElasticSearch para encontrar livros que correspondam à descrição fornecida pelo usuário. A implementação concreta, *BookService*, realiza essa busca e recupera os resultados. Para garantir uma separação clara entre a lógica de negócios e o acesso aos dados, utilizamos a interface *IBookRepository*, que segue o padrão de design de repositório. Este padrão

encapsula a lógica de acesso aos dados, permitindo que a aplicação interaja com a base de dados de maneira abstrata e uniforme.

3.2.4 Diagrama de entidade relacionamento (ER)

A seguir, será apresentado o diagrama ER da aplicação. Este diagrama tem como objetivo principal ilustrar o relacionamento das entidades utilizadas em nosso banco de dados, proporcionando uma visão abrangente de como o sistema armazena e interconecta as informações. Por meio desse diagrama, será possível compreender de forma mais clara e detalhada como os dados são registrados e gerenciados na aplicação, contribuindo para uma melhor compreensão do funcionamento do sistema.

Figura 30 - Diagrama de ER da aplicação.



Fonte: Autor (2023)

Na Figura 30 é exibido o diagrama entidade relacionamento foi identificado as entidades fundamentais para esta aplicação. Inicia com a entidade 'Livro', que

representa o registro de cada livro. Cada livro está vinculado a uma única categoria, estabelecendo uma relação de 'um para muitos', pois uma categoria pode englobar diversos livros.

A entidade 'LivroItem' é essencial para a gestão do acervo de livros. Ela permite monitorar o status de cada exemplar individual, viabilizando o cálculo de disponibilidade ou reserva dos livros. 'LivroItem' mantém um relacionamento 'um para muitos' com 'Livro', indicando que um livro pode ter múltiplos itens no estoque.

Para o gerenciamento dos usuários, criamos a entidade 'Aluno', que armazena dados como nome, matrícula e e-mail, identificando os alunos que utilizam a aplicação. Associada a ela, temos a entidade 'AlunoAutenticação', dedicada à autenticação do usuário na aplicação, contendo um campo para a senha, que deve ser armazenado de forma criptografada para segurança dos dados.

Por fim, a entidade 'Reserva' é designada para registrar cada reserva efetuada. Ela contém o identificador do aluno responsável pela reserva e o 'LivroItem' reservado. Esta entidade também registra a data de devolução prevista do livro e mantém o status da reserva.

Quanto ao relacionamento da 'Reserva' com as demais entidades, observamos que um 'Aluno' pode realizar múltiplas reservas, enquanto cada 'LivroItem' é exclusivo de uma reserva. Ao concretizar uma reserva, é necessário atualizar o status nas entidades 'Reserva' e 'LivroItem' correspondentes.

3.3 Desenvolvimento

Neste tópico, aborda-se o desenvolvimento da solução proposta neste trabalho, de maneira a esclarecer todas as tecnologias, técnicas e métodos utilizados para implementar a aplicação do protótipo. O detalhamento dar-se-á apresentando, primeiramente, as tecnologias utilizadas na implementação do código. Segue-se, então, mostrando a arquitetura construída, seguido pela apresentação individual de seus módulos principais e como estes se integram para o

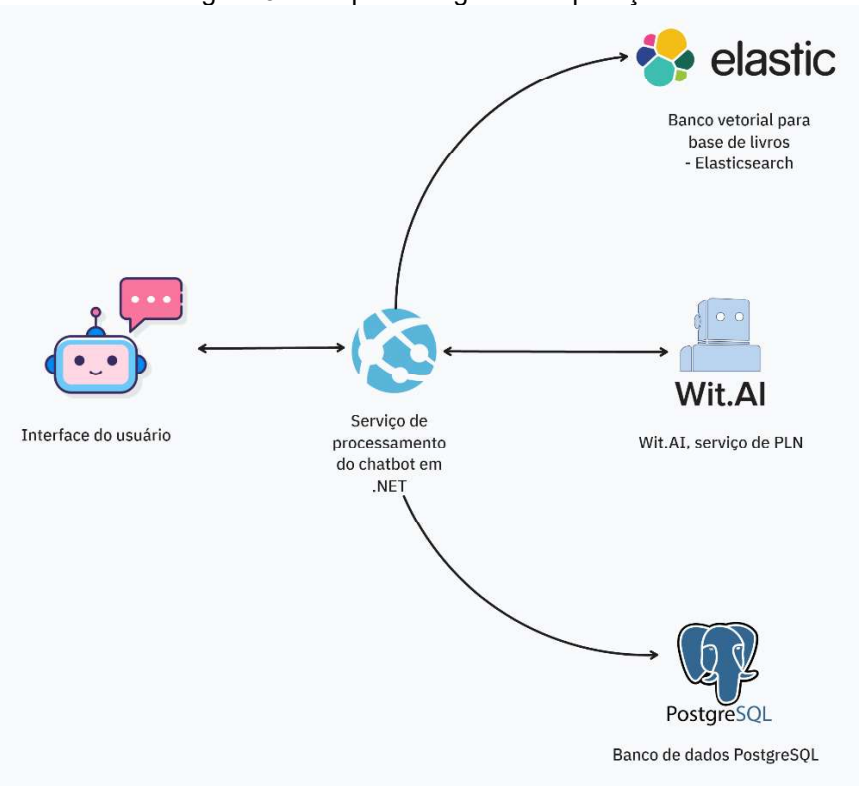
funcionamento da proposta de software. Por fim, é mostrado como o código fonte da aplicação funciona.

3.3.1 Arquitetura da aplicação

Este tópico aborda a visão arquitetural da aplicação, mostrando como os pontos dela interagem entre si.

3.3.2 Funcionamento Geral

Figura 31 - Arquitetura geral da aplicação.



Fonte: Autor (2023)

A Figura 31 apresenta a arquitetura geral da aplicação, proporcionando uma visão macro de como o usuário interage com a aplicação e como esta realiza, de modo simplificado, suas atividades.

Inicialmente, temos a interface do usuário, também referida como chatbot ou bot, que será acessada pelo aluno por meio do navegador. Esta interface constitui a camada de interação do aluno com o sistema. A cada envio de uma mensagem através do chatbot, a mensagem é encaminhada, via comunicação em tempo real facilitada pelo framework SignalR, para o Serviço de Processamento do Chatbot desenvolvido em .NET, conforme ilustrado no centro do diagrama arquitetural.

O Serviço de Processamento do Chatbot em .NET é responsável por executar as ações necessárias para continuar o fluxo do sistema, além de gerenciar a comunicação com outros serviços, como o CLU, o banco de dados PostgreSQL e o Elasticsearch. Baseado nas respostas oriundas do CLU, o serviço de processamento pode determinar o fluxo com o qual o sistema deve seguir, também é responsável pela formatação dos dados, e retorno adequado deles, como no caso de retornar uma lista de livros disponível após a consulta de um livro, ele formata os dados e retorna para o usuário de forma que a interface que interage com o aluno, entenda qual componente renderizar

Este serviço se comunica com o CLU, nosso serviço de Processamento de Linguagem Natural (PLN), para identificar as intenções do usuário, seja uma consulta de livro, uma reserva de livro ou a consulta de reservas existentes.

O CLU é encarregado de identificar as intenções, ações e entidades descritas no texto fornecido pelo usuário, e retorna o processamento desse texto ao Serviço de Processamento do Chatbot.

Para uma consulta de livro, baseada na descrição não exata fornecida pelo usuário, é utilizado o serviço do Elasticsearch, que possui a capacidade computacional de buscar o documento mais aproximado ao descrito pelo usuário.

Se o usuário efetuar uma reserva, essa reserva é salva no banco de dados PostgreSQL, que é responsável por armazenar os dados da aplicação, como: alunos, livros, reservas de livros e autenticação do usuário.

3.3.2.1 Base de dados dos livros

Para a elaboração do nosso protótipo, selecionamos um conjunto específico de 50 livros, os quais foram utilizados para preencher a base de dados e criar um ambiente de teste adequado. Esses livros foram escolhidos aleatoriamente e têm o propósito exclusivo de simular uma biblioteca já existente.

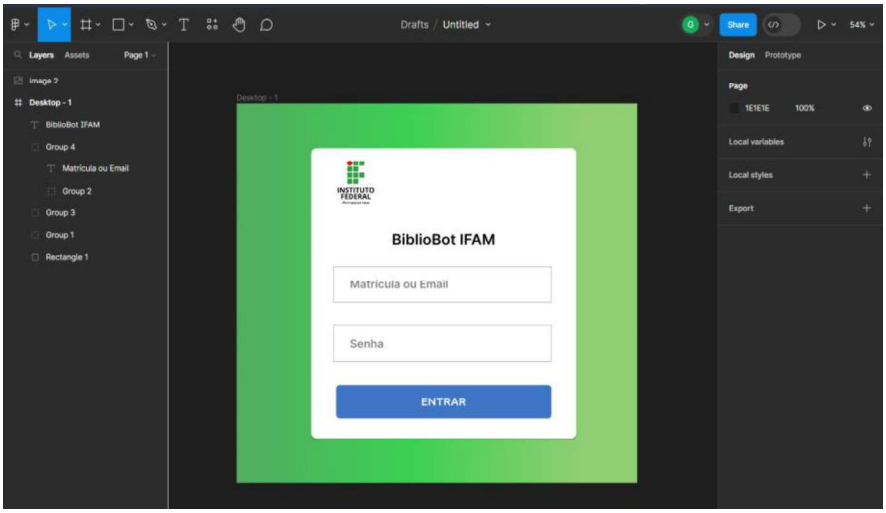
Os livros utilizados no protótipo foram escolhidos pelo autor do projeto e têm como única função ilustrar como a base de dados seria preenchida com livros, facilitando a realização de consultas ou reservas no sistema.

Em cenários de produção, que não são abordados neste trabalho, é possível desenvolver, no futuro, uma aplicação que permita aos bibliotecários inserirem os dados dos livros e, assim, popular a base de dados real. Alternativamente, poderá ser utilizada uma Interface de Programação de Aplicação (API), fornecida pelo serviço do chatbot, para facilitar a inclusão de livros na base de dados do *chatbot*.

3.3.3 Modelagem das telas

Para a modelagem das telas e servir como um guia para o desenvolvimento no código *front-end*, foi realizar a modelagem de telas. Foi utilizado a ferramenta Figma para cria o protótipo das telas.

Figura 32 - Interface do programa Figma, com a tela de prototipação de Login



Fonte: Autor (2023)

Na Figura 32 é exibido a interface do programa Figma, que foi utilizado para criar as modelagens das telas utilizadas no projeto. A ideia é ter um guia de como desenvolver as telas e ajudará no desenvolvimento das telas no front-end. É importante ressaltar que as telas não seguiram a risco todos os estilos aplicados na modelagem, mas isso se restringiu apenas a aspectos mínimos, como cores, tamanho das letras e cores de letras.

Ainda na Figura 32, é exibido a prototipação da tela de login, isso facilita o desenvolvimento das interfaces no *React.js*, pois isso permite nos dar uma noção dos estilos que serão aplicados para desenvolver uma tela o mais próxima possível da prototipação

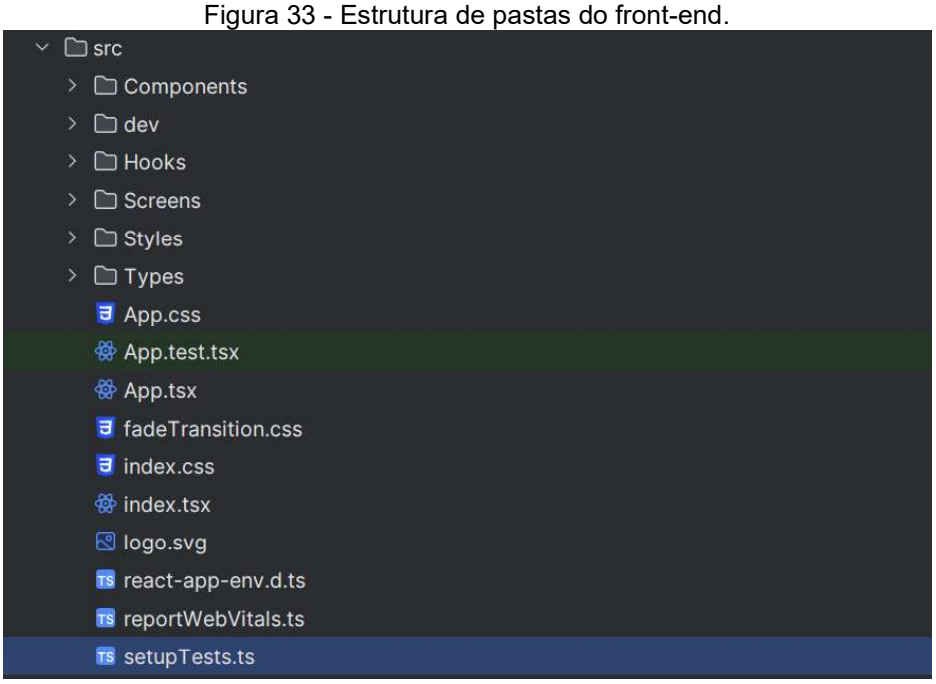
3.4 Código da aplicação

Neste tópico iremos descrever como foi feita a estruturação de pastas do projeto, tanto no *front-end* quanto no *back-end* da aplicação. Serão demonstrados alguns trechos de códigos que são importantes para a compreensão da aplicação

3.4.1 *Front-end*

O *Front-end* refere-se à interface de usuário da aplicação e é crucial para a interação efetiva entre o usuário e o ecossistema do bot. A Figura *33* ilustra a organização das pastas utilizadas no *front-end* da aplicação do chatbot.

Figura 33 - Estrutura de pastas do front-end.



Fonte: Autor (2023)

Na Figura 33 temos a estrutura de pastas do projeto front-end, nela temos as pastas como os componentes do nosso *chatbot*, temos também pasta das páginas para o nosso *chatbot* como tela de login e a tela do *bot* em si, pasta de configuração e tipos.

Nesta estruturação, a pasta central é denominada "src", abreviação de "*source*" (que traduzido significa "origem"). Ela serve como o núcleo onde todos os códigos vitais da aplicação são armazenados e categorizados em subpastas específicas.

Dentro de "src", encontramos a pasta "*Components*", cujo nome é derivado de "componentes". Essa pasta hospeda os diferentes componentes visuais da aplicação. Por exemplo, nela podemos encontrar o componente responsável pelo campo de texto onde o usuário insere suas mensagens para o bot e o componente de mensagem, designado para exibir as mensagens trocadas entre o usuário e o bot.

A Figura 34, por sua vez, destaca o código do componente denominado "*MessageListComponent*". Este componente específico gerencia e exibe a lista de

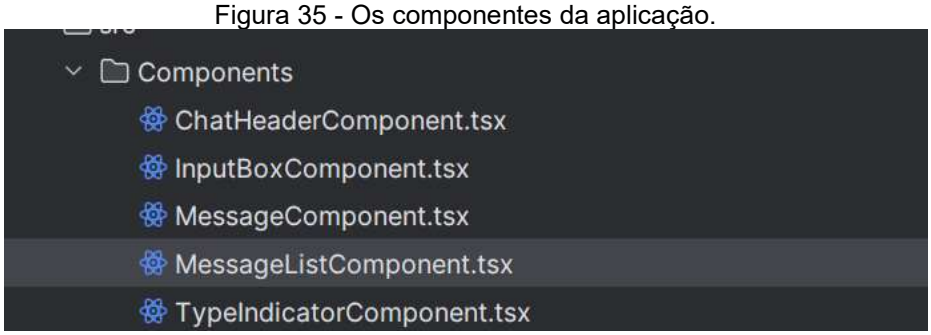
mensagens trocadas, garantindo uma apresentação clara e organizada para o usuário na interface.

Figura 34 - Exemplo do código do componente MessageListComponent.

```
const MessageListComponent: React.FC<MessageListProps> = ({ messages : MessageType[] , onOptionClick, bookReservationFunc }) => {
    const endOfMessagesRef : React.MutableRefObject<HTMLDiv...  = useRef<HTMLDivElement | null>( initialValue: null);
    useEffect( effect: () : void  => {
        if (endOfMessagesRef.current) {
            endOfMessagesRef.current.scrollIntoView({ behavior: 'smooth' });
        }
    }, deps: [messages]);

    return (
        <List>
            <TransitionGroup component={null}>
                {messages.map((message : MessageType , index : number ) => (
                    <CSSTransition key={index} timeout={300} classNames="message">
                        <ListItem style={{ justifyContent: message.sender === 'user' ? 'flex-end' : 'flex-start' }}>
                            <MessageComponent
                                text={message.text}
                                sender={message.sender}
                                timestamp={message.timestamp}
                                type={message.type}
                                options={message.options}
                                onOptionClick={onOptionClick}
                                bookReservationFunc={bookReservationFunc}
                                books={message.books}
                            />
                        </ListItem>
                    </CSSTransition>
                ))}
            </TransitionGroup>
            <div ref={endOfMessagesRef}></div>
        </List>
    );
};
```

Fonte: Autor (2023)

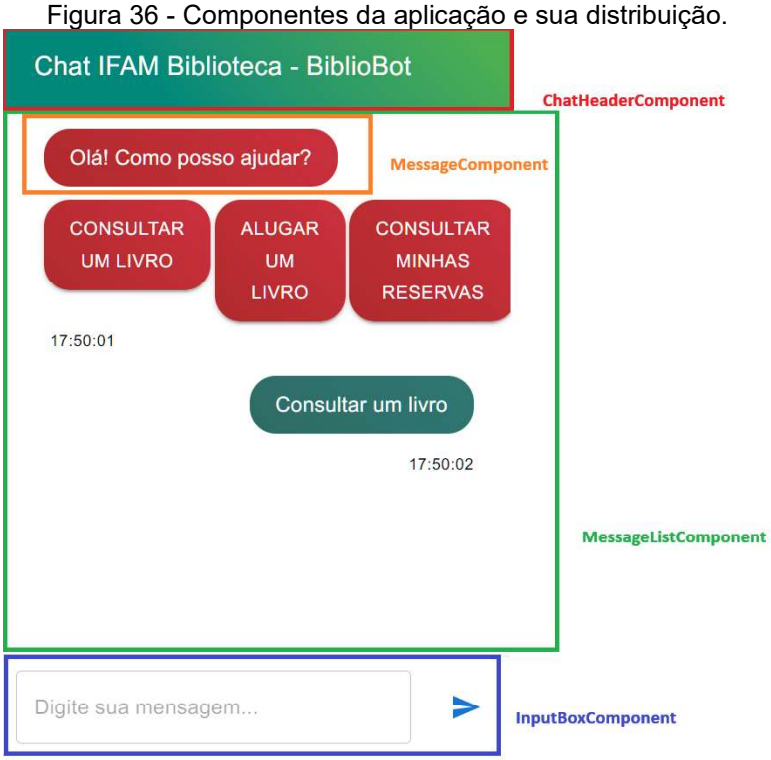
Figura 35 - Os componentes da aplicação.



Fonte: Autor (2023)

Na Figura 35 temos os arquivos que representam os componentes da nossa aplicação, cada arquivo é a implementação de um componente, nele temos o *header* do nosso chat, o campo de texto, o componente de mensagem e o componente de lista de mensagem.

Figura 36 - Componentes da aplicação e sua distribuição.



Fonte: Autor (2023)

Na Figura 36 é exibido como os componentes são organizados na tela, podemos identificar de forma clara a separação de cada, o *header,* o componente de mensagem, o componente de listagem das mensagens, cada um separado por uma cor, o que representaria na realidade como são renderizadas.

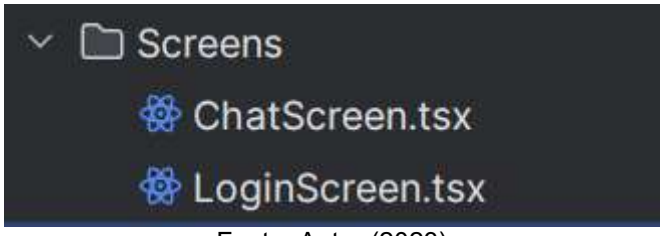
Dentro da organização de pastas do *front-end* da aplicação, encontramos uma pasta especial denominada "*Hooks*". No contexto das aplicações React, *hooks* são funções que permitem a você "ligar-se" a recursos do React sem precisar escrever uma classe. Eles proporcionam uma maneira de gerenciar o estado e os efeitos colaterais nas funções dos componentes.

Na aplicação, a pasta "Hooks" é especialmente dedicada a *hooks* customizados que atendem necessidades específicas do projeto, particularmente em relação à comunicação em tempo real e ao gerenciamento de mensagens do chatbot.

Um exemplo de *hooks* se encontra no arquivo *useSignalR.ts*, este *hook* é focado na conexão com o SignalR, uma biblioteca que facilita a comunicação em tempo real entre o nosso servidor de processamento do bot e o cliente. O *hook* define e mantém a conexão através do *HubConnectionBuilder* fornecido pelo pacote @microsoft/signalr. Uma vez inicializado, ele retorna a conexão estabelecida, permitindo que outros componentes ou *hooks* utilizem essa conexão conforme necessário.

Na arquitetura da aplicação, a pasta "*Screens*" desempenha um papel fundamental, representando as principais telas ou vistas que o usuário interage diretamente. Estas telas são responsáveis por orquestrar componentes e *hooks* para fornecer uma experiência de usuário coesa. Dentro desta pasta, encontramos dois arquivos notáveis conforme mostrado na Figura 37.

Figura 37 - Pasta *Screens* e seus arquivos.



Fonte: Autor (2023)

O arquivo *ChatScreen* contém a tela que é a interação entre o usuário e o chatbot. Utilizando vários componentes, como *MessageListComponent*, *InputBoxComponent* e *ChatHeaderComponent*, ela cria a interface principal onde as mensagens são trocadas. Além disso, a tela integra-se com os *hooks useSignalR* e *useSignalRChat* para gerenciar a comunicação em tempo real com o servidor, permitindo enviar e receber mensagens. Outras funcionalidades como a exibição de indicadores de digitação e o manuseio de reservas de livros também são gerenciadas aqui, a Figura 22 representa a tela de *ChatScreen*.

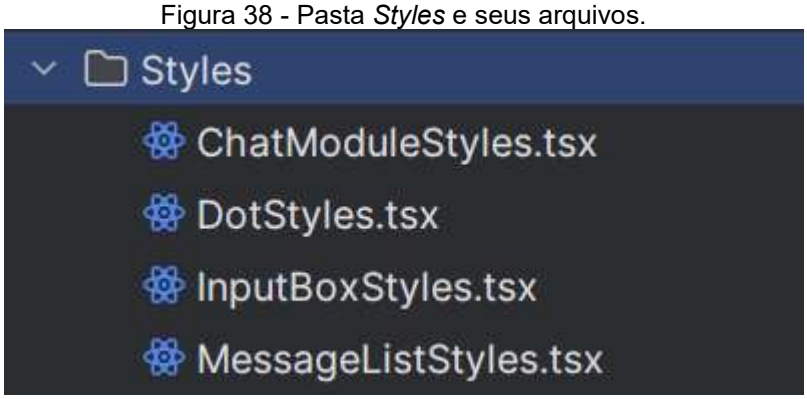
O arquivo *LoginScreen* é responsável pela autenticação do usuário na aplicação. Ela apresenta campos para inserção de matrícula ou e-mail e senha. Um login informado incorretamente emite um alerta informando que os dados de login estão incorretos, a Figura 21 representa a tela de *LoginScreen*.

A pasta "*Screens*" é uma representação clara das principais interfaces da aplicação, assegurando que o usuário tenha uma experiência interativa e intuitiva enquanto interage com o chatbot e o sistema como um todo.

No desenvolvimento de aplicações web, a separação entre a lógica da aplicação e sua apresentação visual é fundamental para manter o código limpo, modular e facilmente gerenciável. A pasta "*Styles*" exemplifica essa prática, concentrando-se exclusivamente em definir e gerenciar estilos que adornam a interface do usuário da aplicação.

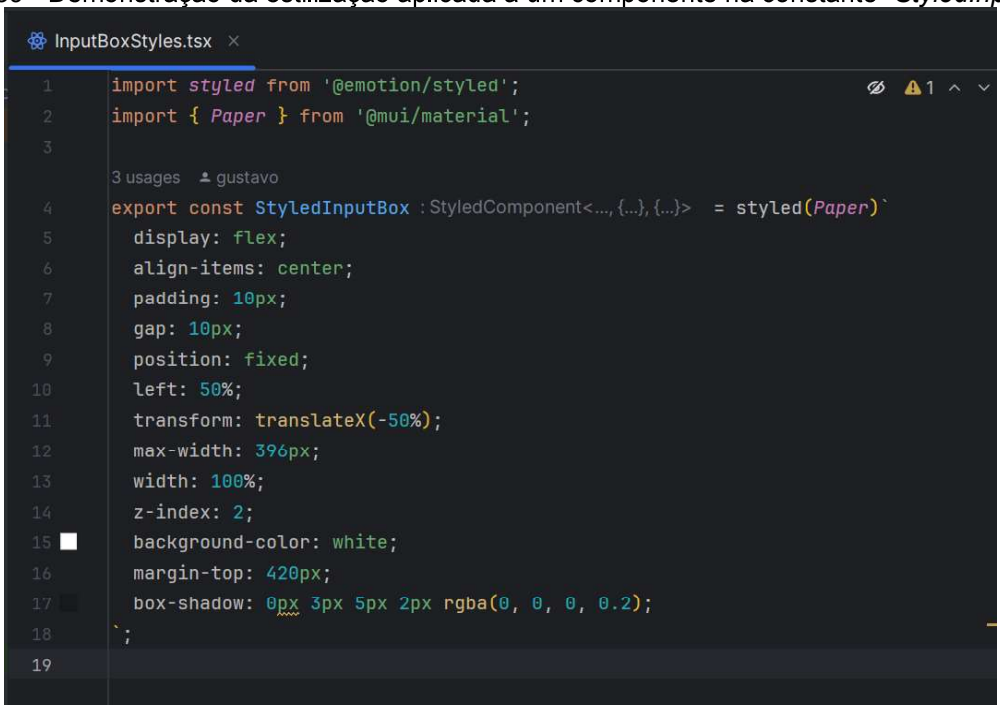
Dentro da pasta "Styles", encontramos arquivos específicos que definem estilos para diferentes partes da aplicação, a Figura 38 demonstra os arquivos que encontramos dentro desta pasta.

Figura 38 - Pasta *Styles* e seus arquivos.



Fonte: Autor (2023)

Figura 39 - Demonstração da estilização aplicada a um componente na constante “*StyledInputBox*”.



Fonte: Autor (2023)

A Figura 39 representa um destes componentes definidos dentro da pasta *Styles*. *StyledInputBox* define regras de estilos para o componente “*Paper*” do pacote @mui/material, o objetivo deste componente é definir a estilização do componente como por exemplo a largura, através da propriedade “*width*: 100%”, cor de fundo do componente pela propriedade “background-color: *white*”, entre outros.

A pasta Types contém a definição de tipos que serão usados na aplicação front-end. Isso garante maior clareza nas interações entre diferentes partes do sistema, além de garantir a integridade dos dados e robustez do código.

A Figura 40 demonstra uma destas definições de tipos, *ChatResponseBook* é responsável por representar para a aplicação um objeto que seria um livro, e contém uma propriedade chamada *Actions* que representa uma ação referente sobre este livro.

Figura 40 - Definição do tipo *ChatResponseBook*.

```
export type ChatResponseBook = {
    id: string;
    title: string;
    author: string;
    description: string;
    imageUrl: string;
    actions: Actions[];
};
```

Fonte: Autor (2023)

O ponto inicial da aplicação é o arquivo App.tsx como mostrado na Figura 41, ela se encontra fora da pasta src, este arquivo é responsável por unir os principais elementos de interface do usuário e renderizar os mesmos. O arquivo App.tsx é responsável por orquestrar a lógica principal de renderização da aplicação, alternando entre a tela de login e a tela de chat com base no estado de login do usuário.

Figura 41 - O arquivo App.tsx que representa o ponto inicial da aplicação front-end.

App.tsx

```
import React, {useState} from 'react';
import './App.css';
import LoginScreen from "./Screens/LoginScreen";
import {CSSTransition} from "react-transition-group";
import './fadeTransition.css';
import ChatScreen from "./Screens/ChatScreen";

5+ usages  gustavo
function App() : JSX.Element  {
    const [isLoggedIn : boolean , setIsLoggedIn : React.Dispatch<React.SetStateA... ] = useState( initialState: false);
    return (
        <CSSTransition
            in={isLoggedIn}
            timeout={300}
            classNames="fade"
        >
            {() => (
                <>
                    {isLoggedIn ? (
                        <ChatScreen />
                    ) : (
                        <LoginScreen onLogin={() => setIsLoggedIn( value: true)} />
                    )}
                </>
            )}
        </CSSTransition>
    );
}

4 usages  gustavo
export default App;
```

Fonte: Autor (2023)

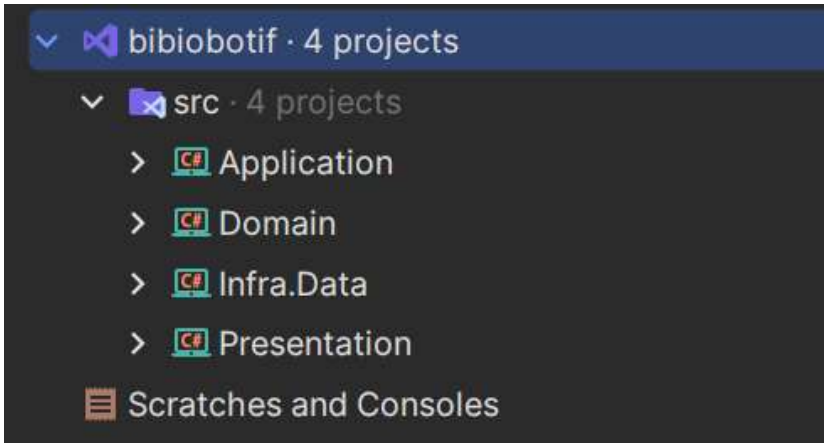
3.4.2 *Back-end*

O *Back-end* refere-se ao nosso serviço de processamento do chatbot em .Net, ilustrado na Figura 31, como dito anteriormente o *back-end* é o nosso serviço núcleo da aplicação e é responsável pela inteligência e funcionalidade do nosso bot.

Optamos pela abordagem da "*Clean Architecture*" (Arquitetura Limpa) no desenvolvimento do nosso *back-end*. Esta arquitetura não apenas permite que diferentes partes do sistema sejam desenvolvidas, testadas e escaladas independentemente umas das outras, oferecendo grande modularidade, mas também facilita a identificação e correção de problemas. Além disso, a implementação de melhorias pode ser realizada sem perturbar outros aspectos do sistema, garantindo manutenibilidade. O design desacoplado desta arquitetura também favorece a sua testabilidade. Por fim, a clareza na estruturação do código, acelera o processo de desenvolvimento, consolidando uma organização eficaz.

A Figura 42 fornece uma visão da estrutura de diretórios do *back-end,* que foi planejada para refletir a separação por camadas, tornando a localização e compreensão de diferentes partes do código intuitivas.

Figura 42 - Organização de pastas do serviço de processamento do chatbot em .NET.



Fonte: Autor (2023)

Ainda na Figura 37 foi priorizado uma organização para enfatizar a adoção de uma arquitetura limpa. Dentro da pasta principal, src, cada subpasta ou projeto

representa uma camada distinta dessa arquitetura, facilitando a manutenção, expansão e compreensão do sistema por parte dos desenvolvedores.

O sistema é projetado para iniciar pela camada mais externa, que é a camada de apresentação. Esta camada é simbolizada pelo projeto Presentation. Uma característica fundamental dessa camada é a definição do Chathub, conforme exposto no nosso diagrama de classes. Esse módulo desempenha um papel vital na mediação da comunicação e interação entre o usuário e o sistema.

O projeto Presentation não serve apenas como um ponto de entrada. Ele é essencial para configurar e definir parâmetros para a camada de apresentação. Na Figura 43 temos a representação da estrutura de diretórios do projeto Presentation, nela podemos achar nossas classes de controladores, responsável pelas nossas rotas na API, também tempos os Hubs que serão utilizados pelo SignalR.

Figura 43 - O projeto Presentation expandido e suas subpastas e arquivos.

Fonte: Autor (2023)

Figura 44 - A classe Chathub, que é o ponto de entrada da aplicação.

```
namespace Presentation.Hubs;

3 usages  Gustavo Costa +1
public class ChatHub: Hub
{
    private readonly IMediator _mediator;

    gustavo
    public ChatHub(IMediator mediator)
    {
        _mediator = mediator;
    }

    gustavo
    public async Task SendMessage(string user, string message)
    {
        var result :ChatResponseBase = await _mediator.Send( request: new GetMessageQuery
        {
            Text = message,
            MessageId = Context.ConnectionId
        }); // Task<ChatResponseBase>

        await Clients.Caller.SendAsync( method: "ReceiveMessage", result);
    }
```

Fonte: Autor (2023)

A Figura 44 temos a classe Chathub, esta classe é responsável pelo gerenciamento do fluxo no SignalR, isso significa que esta classe é o ponto de entrada da comunicação do usuário com o sistema de gerenciamento do *chatbot.* Temos o método *SendMessage,* este método é responsável por mandar a mensagem para os clientes conectados no hub do SignalR.

O projeto Infra.Data é crucial para a interação da aplicação com serviços externos, incluindo o Elasticsearch e o CLU da Azure. Ele não só gerencia essa comunicação, mas também define os serviços nas interfaces da camada Application, que será detalhada posteriormente.

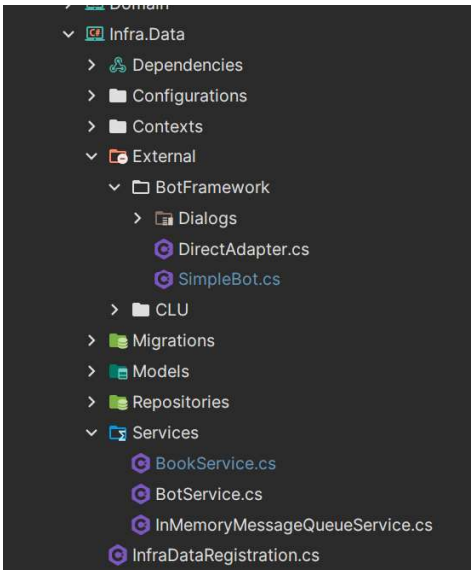
Adicionalmente, este projeto incorpora o Bot Framework da Azure, essencial para estruturar o fluxo operacional do bot. Dentre suas classes chave estão o SimpleBot e o BotService, que estabelecem e gerenciam os contextos usados pelo nosso chatbot.

Também é importante citar que é nesta camada que foi implementado os métodos de comunicação com a base de dados PostgreSQL, bem como a configuração da base de dados com ele, utilizamos para isso as classes de repositório, na pasta “Repositories”.

Neste contexto, pode-se dizer que caso haja desejo de alterar o banco de dados que se comunica, pode-se fazer neste projeto. Um exemplo seria poder se conectar a outras bases de dados de outras bibliotecas por exemplo, sem precisar alterar o núcleo do sistema

Na Figura 45 é exibido a estrutura de diretórios do projeto Infra.Data, nela podemos ver a pasta de configurações, de contextos do nosso *bot*, das classes externas, modelos e serviços.

Figura 45 - Projeto Infra.Data e seus arquivos e subpastas.



Fonte: Autor (2023)

Na camada mais interna da estrutura, encontramos o projeto *Application*. Este desempenha um papel crucial, sendo encarregado da implementação dos casos de uso, bem como da definição das interfaces de serviços e de outras classes que estabelecem as regras específicas do nosso serviço.

Um dos elementos centrais deste projeto é a definição das interfaces de serviços. Estas representam os contratos que as classes no projeto Infra.Data irão implementar, assegurando uma modularidade ao sistema. Esse desacoplamento é fundamental para a flexibilidade da aplicação. Por exemplo, seria fácil substituir o serviço do Elasticsearch por outro banco de dados, a transição poderia ser feita ajustando apenas a conexão no Infra.Data, mantendo intacta a camada Application.

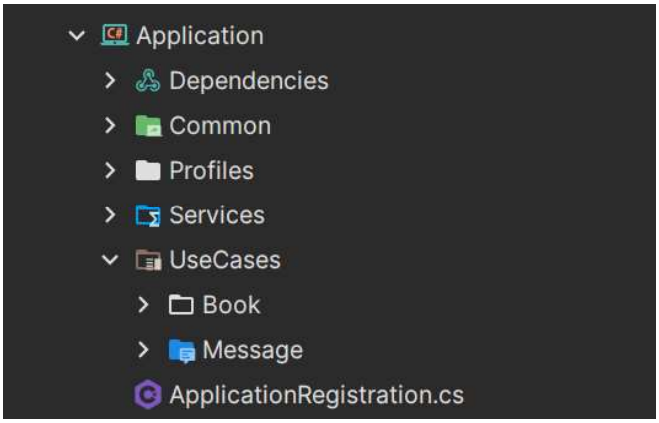
Adicionalmente, temos a definição de casos de uso, que são essencialmente as funcionalidades da aplicação. Estes são organizados em contextos e representam as regras específicas de aplicação. Como exemplo, a Figura 46 apresenta o caso de uso BookReservationCommandHandler. Esta classe é designada para uma funcionalidade específica: a reserva de livros. Ela não apenas manipula essa tarefa, mas também gerencia os dados fornecidos pelo usuário, que foram processados pela camada de apresentação. Após receber e tratar esses dados, a classe utiliza as interfaces dos repositórios para consolidar as informações de reserva na base de dados.

Figura 46 - Exemplo de um caso de uso de reserva de livro.

```
public class BookReservationCommandHandler: IRequestHandler<BookReservationCommand, ChatResponseBase>
{
    private readonly IBookRepository _bookRepository;
    private readonly IStudentRepository _studentRepository;
    private readonly IBookReservationRepository _bookReservationRepository;

    Gustavo Costa
    public BookReservationCommandHandler(
        IBookReservationRepository bookReservationRepository
        , IBookRepository bookRepository
        , IStudentRepository studentRepository)
    {
        _bookRepository = bookRepository;
        _studentRepository = studentRepository;
        _bookReservationRepository = bookReservationRepository;
    }

    Gustavo Costa
    public async Task<ChatResponseBase> Handle(BookReservationCommand request, CancellationToken cancellationToken)
    {
        var bookId :Guid  = Guid.Parse(request.BookId);
        var studentId :Guid  = Guid.Parse("54784034-fec1-4809-8b98-a72c5a079112");
        var expirationDate = DateTime.UtcNow.AddDays(30);
        var bookReservation = new BookReservationEntity()
        {
            BookId = bookId,
            StudentId = studentId,
            ReservationDate = DateTime.UtcNow,
            ReservationStatus = "C",
            ReservationEndDate = expirationDate,
        };

        await _bookReservationRepository.Insert(bookReservation, cancellationToken);

        return new ChatResponseBase()
        {
            Texts = new List<string>()
            {
                "Livro reservado com sucesso! 👀✅",
                "Consulte o status da sua reserva com perguntando ao chatbot: 'Qual o status da minha reserva?' 🔮",
                $"O dia de devolução é: {expirationDate.ToString(format: "dd/MM/yyyy")} 📅"
            },
            Timestamp = DateTime.Now.ToString(format: "HH:mm:ss")
        };
    }
```

Fonte: Autor (2023)

Figura 47 - Camada de aplicação, projeto Application seus arquivos e pasta.



Fonte: Autor (2023)

Na Figura 47 temos a estrutura de pastas da Application, onde temos as pastas de interfaces dos serviços que utilizaremos no projeto, temos os casos de uso com sua pasta, dentro dos casos de uso temos os conceitos de “Book” e “Message”.

E por fim, o último projeto é o Domain, este projeto representa a camada mais interna do projeto, a camada de domínio. Este projeto é responsável por conter as regras de negócios da aplicação.

Dentro do projeto Domain, as interfaces dos repositórios estabelecem os contratos para classes concretas existentes no projeto Infra.Data que realizam operações na base de dados PostgreSQL. Na Figura 48, temos o exemplo de um repositório para a operação de obter todos, essa interface tem como característica ser genérica, ao ponto que é possível adicionar outros tipos de repositório para agregar a ela e essas interfaces poderem implementar o método GetAll, como no caso da Figura 49, que é o repositório de Livros que estende a interface IGetAllRepository.

Figura 48 - Exemplo do repositório IGetAllRepository.

```
public interface IGetAllRepository<TAggregate> : IRepository
    where TAggregate : AggregateRoot
{
    public Task<IEnumerable<TAggregate>> GetAll(CancellationToken cancellationToken);
}
```

Fonte: Autor (2023)

Figura 49 - Exemplo do repositório IBookRepository.

```
namespace Domain.Repositories;

gustavo +1
public interface IBookRepository : IGetRepository<BookEntity>,IGetAllRepository<BookEntity>
{
}
```

Fonte: Autor (2023)

Figura 50 - Estrutura de pastas do projeto Domain.

Fonte: Autor (2023)

Na Figura 50, temos a estrutura de pastas do projeto *Domain,* este projeto representa o *core* da nossa aplicação, e é responsável por ter nossas regras de negócios.

3.4.3 Conclusões sobre o capítulo

Ao longo deste capítulo, foi possível compreender em detalhes a proposta e a estruturação do anteprojeto do sistema de chatbot para as bibliotecas. Inicialmente, foram apresentados os requisitos funcionais e não funcionais que nortearam o desenvolvimento da aplicação, deixando claro as funcionalidades essenciais que o sistema deve possuir e os padrões de qualidade desejados.

Através da modelagem, foi possível vislumbrar a lógica e os fluxos de interação do sistema, assim como a estruturação das classes, que são peças-chave para o funcionamento da aplicação. Esse entendimento estrutural foi aprofundado com a apresentação detalhada da organização das pastas e dos trechos de códigos,

tanto do *front-end* quanto do *back-end*, fornecendo uma visão completa do projeto e da distribuição das responsabilidades em diferentes camadas e módulos.

A abordagem modular, inspirada nos princípios do *SOLID* e do conceito de *Clean Architecture*, deixa nosso sistema robusto, módulo e de fácil manutenção, também evidencia um cuidado na busca por um código limpo e eficiente. A integração com tecnologias externas, como o Elasticsearch, PostgreSQL e CLU da Azure, demonstra a capacidade do sistema em interagir e operar em conjunto com ferramentas avançadas, garantindo uma operação eficaz e uma experiência de usuário agradável.

Em síntese, o anteprojeto aborda uma solução tecnológica sofisticada para atender às necessidades da interação entre aluno e biblioteca, para a reserva e consulta de livros através de um chatbot. A organização e a clareza na definição dos requisitos e a robustez na arquitetura garantem que a solução proposta possa estar aberta a novas implementações, melhorias e novas funcionalidades, estabelecendo um padrão elevado para futuras implementações e inovações no domínio da interação do chatbot de uma biblioteca.

CONSIDERAÇÕES FINAIS

Nesta monografia, desenvolvemos um chatbot para transformar a interação entre alunos e as bibliotecas. Este sistema não apenas facilita a gestão de materiais e a consulta de livros, mas também aprimora a gestão de reservas, otimizando a experiência educacional dos alunos.

Há um amplo espaço para melhorias e expansões. A introdução de funcionalidades como alertas de reserva para os usuários e um sistema de gestão para bibliotecários pode enriquecer ainda mais a plataforma, atendendo integralmente às necessidades tanto dos alunos quanto da administração da biblioteca.

No desenvolvimento deste projeto, enfrentamos o desafio de encontrar uma ferramenta de processamento de linguagem natural acessível e adequada para fins acadêmicos. Os experimentos conduzidos no segundo capítulo proporcionaram uma visão mais clara sobre qual ferramenta poderia ser empregada de maneira eficaz neste projeto. Além disso, devido às restrições de tempo e à complexidade do desenvolvimento de uma aplicação abrangente, optamos por concentrar nossos esforços na criação de funcionalidades voltadas para os alunos.

Este projeto proporcionou ganhos significativos em termos de experiência no desenvolvimento de *chatbots* e na compreensão de como melhorar a interação entre aluno e biblioteca. *Chatbots* estão ganhando cada vez mais relevância, assim como a digitalização dos negócios, que visa aprimorar a eficácia das soluções educacionais. Portanto, este projeto pode servir como base para estudos futuros sobre interfaces conversacionais e o aperfeiçoamento dos serviços institucionais.

No entanto, vale ressaltar que enfrentamos algumas dificuldades, como o tempo necessário para desenvolver uma aplicação com um nível moderado de complexidade, devido às regras de negócio envolvidas e às ferramentas de IA utilizadas, bem como a evolução constante no desenvolvimento dessas ferramentas.

O primeiro capítulo teve como objetivo reunir os conceitos fundamentais utilizados no projeto, incluindo a criação de uma interface de usuário conversacional para proporcionar uma experiência mais dinâmica e humanizada aos alunos. Em seguida, fizemos os experimentos com as ferramentas de processamento de linguagem natural, que aprimoraram a capacidade do nosso chatbot de compreender as intenções de uma entrada de texto do usuário, bem como identificar as entidades associadas a ela. No capítulo subsequente, descrevemos detalhadamente a implementação do nosso chatbot, apresentando informações sobre como o sistema foi modelado, quais cenários de uso foram considerados e como realizamos o mapeamento das entidades relacionadas ao nosso banco de dados. Também mostramos trechos do código e separação entre o desenvolvimento da interface e o desenvolvimento do serviço de gerenciamento do chatbot.

A escolha de ferramentas de processamento de linguagem natural foi crucial para alcançar uma solução de baixo custo, fácil utilização e alta eficácia. A integração da inteligência artificial permitiu que o chatbot se tornasse mais dinâmico e humanizado, proporcionando uma interface mais intuitiva e acessível para os usuários.

O desenvolvimento incremental do chatbot e sua arquitetura modular são fundamentais para a adaptabilidade do sistema a mudanças futuras, alinhando-se com práticas modernas de desenvolvimento de software, como código limpo e arquitetura limpa.

Este trabalho evidencia o potencial dos *chatbots* e interfaces de usuário conversacionais no enriquecimento da experiência acadêmica. A implementação bem-sucedida deste *chatbot*, abre caminho para futuras inovações em outras instituições educacionais, demonstrando que a tecnologia, quando bem aplicada, pode ser uma ferramenta valiosa no aprimoramento do ambiente educacional.

REFERÊNCIAS

AGGARWAL, Sanchit. **Modern web-development using reactjs.** International Journal of Recent Research Aspects, v. 5, n. 1, p. 133-137, 2018.

ALBAHARI, Joseph. **C# 10 in a Nutshell**. 1. ed. Sebastopol: O'Reilly Media, Inc., 2022. 1060 p. ISBN 9781098121921.

ARORA, Gaurav. K. **SOLID Principles** Succinctly. Syncfusion, 2016. Disponível em: https://www.freedownloads247.com/UploadedFiles/8-2017/4223/solidprinciplessuccinctly.pdf. Acesso em: 24 dezembro 2023.

ATHICK, Asjad; BANON, Shay. **Getting Started with Elastic Stack 8.0**. Packt Publishing, março de 2022. 474 p.

CALADO, Caio. **Magazine Luiza — entrevista com o time responsável pela criação da Lu**. 2018. Disponível em: https://medium.com/botsbrasil/magazine-luiza-entrevista-com-o-time-respons%C3%A1vel-pela-cria%C3%A7%C3%A3o-da-lu-8fc987fbafad. Acesso em: 04 jan. 2024.

CHURCH, Bella. **5 types of chatbot and how to choose the right one for your business**. 2023. Disponível em: https://www.ibm.com/blog/chatbot-types/. Acesso em: 23/11/2023.

COPELAND, Michael. Q**ual é a Diferença entre Inteligência Artificial, Machine Learning e Deep Learning?** 2021. Disponível em: https://blog.nvidia.com.br/2021/03/10/qual-e-a-diferenca-entre-inteligencia-artificial-machine-learning-e-deep-learning/. Acesso em: 23/11/2023.

DAVIES, Marion. **The Major Subfields of Artificial Intelligence**. 2023. Disponível em: https://www.konsyse.com/articles/the-major-subfields-of-artificial-intelligence/. Acesso em: 23/11/2023.

Dignum, Virginia. **Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way**. https://doi.org/10.1007/978-3-030-30371-6: Springer Cham, 2019. 127.

ELASTIC. **Elasticsearch**. Disponível em: https://www.elastic.co/pt/elasticsearch. Acesso em: 05/01/2024.

FERRARI, Luca; PIROZZI, Enrico. **Learn PostgreSQL**. Packt Publishing, outubro de 2020.

FOSTER, Elvis C.; GODBOLE, Shripad V. Database Systems. Apress, dezembro de 2014. 556 p.

FREIRE, Henrique. **React, Vue, Angular:** conheça suas vantagens e desvantagens e qual é melhor para seus projetos. 2021. Disponível em: https://henrique-freire.medium.com/react-vue-angular-conhe%C3%A7a-suas-vantagens-e-desvantagens-e-qual-%C3%A9-melhor-para-seus-projetos-53734bb3d37f. Acesso em: 02/11/2023.

GODINHO, Keila Ingrid dos Santos. **Inteligência artificial em bibliotecas: Bibliotecária Informativa Automatizada (BIA) da divisão de bibliotecas e documentação da PUC-RIO**. 2019. 89 f. Trabalho de conclusão de curso (Bacharelado em Biblioteconomia)—Universidade de Brasília, Brasília, 2019.

GORMLEY, Clinton; TONG, Zachary. **Elasticsearch**: The Definitive Guide. O'Reilly Media, Inc., janeiro de 2015.

HALL, Gary. **Adaptive Code**: Agile coding with design patterns and SOLID principles: Microsoft Press, 2017.

HARRINGTON, Jan L. **Relational Database Design and Implementation**, 4th Edition. Morgan Kaufmann, Abril de 2016. 712 p

HIGH, Rob; BAKSHI, Tanmay. **Cognitive Computing with IBM Watson**. Packt Publishing, Abril de 2019. 256 p.

HOWARD, Cole; LANE, Hobson; HAPKE, Hannes**. Natural Language Processing in Action**. Manning Publications, Abril de 2019. 544 p.

HURWITZ, Judith S.; KAUFMAN, Marcia; BOWLES, Adrian. **Cognitive Computing and Big Data Analytics**. Wiley, março de 2015. 288 p.

HUSSEN MAULUD, D.; ZEEBAREE, S. R. M.; JACKSI, K.; MOHAMMED SADEEQ, M. A.; HUSSEIN SHARIF, K. **State of Art for Semantic Analysis of Natural Language Processing**. Qubahan Academic Journal, v. 1, n. 2, p. 21–28, 2021. Disponível em: https://doi.org/10.48161/qaj.v1n2a44. Acesso em: 23/11/2023.

JACINTHO, Fernanda dos S. Becker; PENHA, Alice Demaria Silva. **INTERFACES CONVERSACIONAIS: ANÁLISE DE TAREFAS PARA SIRI E GOOGLE NOW**. Ergodesign; HCI, [S.l.], v. 4, n. 2, p. 72-81, dec. 2016. ISSN 2317-8876. Disponível em: https://periodicos.puc-rio.br/index.php/revistaergodesign-hci/article/view/292. Acesso em: 22 nov. 2023. doi: http://dx.doi.org/10.22570/ergodesignhci.v4i2.292.

JANARTHANAM, Srini. **Hands-On Chatbots and Conversational UI Development**. Packt Publishing, dezembro de 2017. 392 p.

JOSHI, Bipin. **Beginning SOLID Principles and Design Patterns for ASP.NET Developers**. Califórnia: Apress Berkeley.2016.

KHAN, R.; Das, **A. Introduction to *Chatbots*. In. Build Better *Chatbots***. Berkeley: Apress, 2018. Disponível em: https://doi.org/10.1007/978-1-4842-3111-1_1. Acesso em: 23/11/2023.

KHURANA, Diksha; KOLI, Aditya; KHATTER, Kiran; SINGH, Sukhdev. **Natural language processing: state of the art, current trends and challenges**. Multimedia Tools and Applications, [S.l.], v. 82, p. 3713-3744, 2023. Disponível em: https://link.springer.com/article/10.1007/s11042-022-13428-4. Acesso em: 22/11/2023.

KOPETZ, H.; STEINER, W. **Real-Time Communication. In: Springer. Real-Time Systems. Cham: Springer**, 2022. cap. 7. Disponível em: https://doi.org/10.1007/978-3-031-11992-7_7. Acesso em: 24/11/2023.

LISTER, K.; COUGHLAN, T.; INIESTO, F.; FREEAR, N.; DEVINE, P. **Accessible conversation user interfaces: considerations for design. International Web For All Conference**: New York, 2020. Disponível em: https://doi.org/10.1145/3371300.3383343. Acesso em: 25/09/2023.

MICROSOFT. **O que é o Serviço do Azure SignalR**? 2023. Disponível em: https://learn.microsoft.com/pt-br/azure/azure-signalr/signalr-overview. Acesso em: 03/11/2023.

MICROSOFT. **Conversational Language Understanding**. Azure AI Services. 19 dez. 2023. Disponível em: https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/overview. Acesso em: 05 jan. 2024.

MICROSOFT AZURE**. Conversational Language Understanding**. Disponível em: https://azure.microsoft.com/en-us/products/ai-services/conversational-language-understanding. Acesso em: 05 jan. 2024.

MITREVSKI, Martin**. Developing Conversational Interfaces for iOS**: Add Responsive Voice Control to Your Apps. Apress, janeiro de 2018.

MUELLER, John Paul; MASSARON, Luca. **Artificial Intelligence For Dummies**, 2nd Edition. For Dummies, novembro de 2021. 368 p.

REDAÇÃO NAMA. **Lu, o chatbot da Magazine Luiza que é queridinho do público**. 2018. Disponível em: https://simple.nama.ai/post/lu-o-chatbot-da-magazine-luiza-que-e-queridinho-do-publico. Acesso em: 04 jan. 2024.

REACT. 2024. Disponível em: https://react.dev/. Acesso em 05/01/2024.

PETROV, Alex. **Database Internals.** O'Reilly Media, Inc., outubro de 2019. 370 p.

PILICITA GARRIDO, Anabel; BORJA LÓPEZ, Yolanda; GUTIÉRREZ CONSTANTE, Gonzalo. **Rendimiento de MariaDB y PostgreSQL**. Revista Científica y tecnológica UPSE - CTU, La Libertad: Universidad Estatal Península de Santa Elena, v. 7, n. 2, 2021. Disponível em: https://repositorio.upse.edu.ec/handle/46000/7315. Acesso em: 03/11/2023.

POSTGRESQL. Disponível em: https://www.postgresql.org/. Acesso em: 05/01/2024.

RODRÍGUEZ, M.; DANTAS BEZERRA, **B. Processamento de Linguagem Natural para Reconhecimento de Entidades Nomeadas em Textos Jurídicos de Atos Administrativos (Portarias). Revista de Engenharia e Pesquisa Aplicada**, v. 5, n. 1, p. 67-77, 26 abr. 2020.

SANTOS, Gracelynne Oliveira**. Chatbot para bibliotecas**: Um assistente virtual para tirar dúvidas relacionadas à Lei de Direito Autoral. 2023. 122 f. Dissertação(Programa de Pós-graduação em Propriedade Intelectual e Transferência de Tecnologia para Inovação) - Universidade Federal do Maranhão, São Luís, 2023.

SARCAR, Vaskaran. **Java Design Patterns**: A Hands-On Experience with Real-World Examples: Apress, 2022.

SCHWABER-COHEN, ROI. **What is a Vector Database & How Does it Work? Use Cases + Examples**. 2023. Disponível em: https://www.pinecone.io/learn/vector-database/. Acesso em: 23/11/2023.

SHAH, N.; WILLICK, D.; MAGO, V. **A framework for social media data analytics using Elasticsearch and Kibana. Wireless Networks**, v. 28, p. 1179-1187, abr. 2022. Disponível em: https://doi.org/10.1007/s11276-018-01896-2. Acesso em: 02/11/2023.

SOUSA, Mayelson de; GONÇALVES, Alexandrino. **Humanportal – A React.js case study**. In: 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), Seville, Spain, 24-27 jun. 2020. DOI: 10.23919/CISTI49556.2020.9141070. Disponível em: https://ieeexplore.ieee.org/abstract/document/9141070. Acesso em: 05/01/2024.

SYAM, Nilay; NGUYEN, Linh. **Everything you need to know about artificial intelligence**. CGTN, 12 jun. 2019. Disponível em: https://news.cgtn.com/news/2019-06-12/Everything-you-need-to-know-about-artificial-intelligence-HswDK6

TROELSEN, Andrew; JAPIKSE, Phil. **Pro C# 10 with .NET 6**: Foundational Principles and Practices in Programming. Apress, julho de 2022. 1.680 p.

VEMULA, Rami. **Integrating Serverless Architecture: Using Azure Functions, Cosmos DB, and SignalR Service**. 1. ed. Berkeley, CA: Apress, 2019. XIX, 436 p. ISBN 978-1-4842-4489-0. Disponível em: https://doi.org/10.1007/978-1-4842-4489-0. Acesso em: 25/11/2023.

VEMULA, Rami. **Real-Time Web Application Development**: With ASP.NET Core, SignalR, Docker, and Azure. Apress, dezembro de 2017. 607 p.

VIEIRA, Davi. **Designing Hexagonal Architecture with Java**: Packt Publishing, 2023.

WEISFELD, Matt. **The Object-Oriented Thought Process**: Addison-Wesley Professional, 2019.

XIAO, Perry. **Artificial Intelligence Programming with Python**. Wiley, março de 2022. 720 p.