Faculdades Integradas de Taquara - Faccat
Av. Oscar Martins Rangel, 4.500
Taquara, RS, CEP 95600-000

Curso de Sistemas de Informação

BOOKWISE: CHATBOT INTELIGENTE PARA INTEGRAR AS BIBLIOTECAS DA FACCAT[1]

Otávio Muck Schein[2]
Débora Cristina Engelmann[3]

RESUMO

Este projeto visa aperfeiçoar a eficiência na pesquisa de livros nas três bibliotecas da FACCAT, integrando-as em uma única plataforma. Com o objetivo de facilitar o acesso à informações para alunos, professores e funcionários, desenvolveu-se um chatbot utilizando o Rasa. Além disso, empregou-se uma API Java e um banco de dados MongoDB, seguindo a metodologia ágil do Scrum Solo. O resultado é um sistema integrado que otimiza o processo de busca de livros nas bibliotecas, demonstrando a eficácia de um chatbot em conectar recursos e simplificar o acesso a informações.

Palavras-chave: chatbot; rasa; processamento de linguagem natural; spacy.

ABSTRACT

This project aims to improve efficiency in book searching in the three FACCAT libraries, integrating them into a single platform. We developed a chatbot using Rada to facilitate students', teachers', and staff's access to information. Furthermore, we used a Java API and a MongoDb database, following the afile Scrum Solo methodology. The result is na integrated system that optimizes the book search process in libraries, demostrating the effectiviness of a chatbot in connecting resources and simplifying access to information.

---

[1] Trabalho de Conclusão de Curso. Data da submissão e aprovação: 02 dez. 2023.

[2] Acadêmico do curso de Sistemas de Informação, Faculdades Integradas de Taquara - Faccat/RS. E-mail: otavioschein@sou.faccat.br

[3] Professora das Faculdades Integradas de Taquara, Doutora em Ciência da Computação. E-mail: deboraengelmann@faccat.br

Keywords**:** *chatbot; rasa; natural language processing; spacy.*

1 INTRODUÇÃO

No cenário acadêmico, a eficiência e praticidade no acesso à informação são fundamentais para o sucesso dos estudantes, professores e funcionários. Nas Faculdades Integradas de Taquara (FACCAT[4]), por exemplo, essa necessidade se destaca no uso conjunto das três bibliotecas distintas que são disponibilizadas aos alunos: Minha Biblioteca, Biblioteca Pearson e a Biblioteca Física no campus. Cada uma dessas bibliotecas possui sua própria plataforma e catálogo, criando um desafio para aqueles que buscam um livro específico. Esta diversidade de plataformas dificulta a experiência de busca e recuperação de informações, prejudicando alunos, professores e funcionários.

Tendo em vista que um aluno, para buscar um livro específico pode ter que consultar todas as três plataformas distintas antes de encontrar o livro desejado, percebe-se que o processo de busca é demorado e pouco prático. Além disso, os funcionários encarregados das conferências bibliográficas enfrentam um desafio ainda maior, tendo que realizar pesquisas nas três bibliotecas para um único livro.

Diante desse problema, a implementação de um chatbot torna-se uma solução capaz de otimizar o acesso à informação nas bibliotecas da FACCAT. O uso de tecnologias como o Rasa *Open Source* aliado a uma API Java e um banco de dados MongoDB, permite a criação de um chatbot capaz de integrar as três bibliotecas em um único meio de comunicação. Um agente conversacional permite que o usuário pesquise e acesse catálogos das três bibliotecas de forma unificada. Ao simplificar o processo de busca, o usuário pode encontrar os materiais desejados com facilidade, economizando tempo e esforço.

Portanto, neste contexto, este trabalho explora a criação, implementação e avaliação de um chatbot que integra as três bibliotecas da FACCAT, buscando resolver o problema descrito.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Agentes Conversacionais

[4] https://www2.faccat.br/portal/

Agentes conversacionais são programas capazes de interpretar o que o usuário está digitando com o auxílio de inteligência artificial (IA). A partir da interpretação, o programa é capaz de entregar respostas ao usuário, criando um fluxo de conversa tendo como propósito simular uma conversa humana com um usuário real. Um exemplo para um agente conversacional é um chatbot (ABU SHAWAR; ATWELL, 2007).

Um chatbot é definido por Adamopoulou e Moussiades como um programa de computador que responde como entidades espertas quando conversam por texto ou voz e entendem uma ou mais línguas humanas, com o auxílio de Processamento de Linguagem Natural.

Agentes conversacionais utilizam Processamento de Linguagem Natural (PLN). Para Lopez e Kalita, PLN é uma maneira pela qual os computadores são capazes de analisar, interpretar e compreender a linguagem humana de maneira inteligente e eficaz. A implementação de PLN permite que os programadores organizem e estruturem informações, facilitando a execução de variadas tarefas, como tradução, elaboração de resumos, análise de sentimentos e reconhecimento de fala. Assim, agentes conversacionais utilizam PLN para interpretar e entender as intenções humanas.

Existem bibliotecas dedicadas para facilitar o emprego de PLN em agentes conversacionais, uma delas é o spaCy. O spaCy é uma biblioteca de código aberto para PLN que processa texto através de uma série de etapas, incluindo tokenização, análise morfológica, análise sintática e identificação de entidades. Embora ele não seja uma biblioteca exclusiva para chatbots, ele executa o processamento de texto recebido, auxiliando na compreensão linguística (SPACY, 2023). Também, citado por Vasiliev, o spaCy é fácil de utilizar pois ele disponibiliza objetos contêineres que representam elementos de texto de linguagem natural, além de ser possível customizar o modelo a ser utilizado.

O processo executado pelo spaCy inicia com o carregamento de um modelo linguístico específico para a língua em questão, seja ela português, inglês, espanhol, entre outras. Esse modelo encapsula o conhecimento linguístico e as regras necessárias para a análise textual. Após o carregamento do modelo, é gerado um *container* “Doc”, onde são armazenadas anotações linguísticas, que são utilizadas para processar o texto e avançar para as etapas seguintes do *pipeline* de processamento (SPACY, 2023).

Uma vez que o modelo linguístico esteja carregado, para Vasiliev (2020) a primeira etapa é a tokenização, onde o texto é dividido em *tokens*, que podem ser palavras, partes de palavras, espaços em branco, etc. O spaCy garante uma tokenização precisa, pois lida com

nuances linguísticas, como palavras compostas e contrações, razão pela qual existe um modelo específico para cada idioma (SPACY, 2023).

Ainda, é feita a análise morfológica, que identifica as características morfológicas de cada *token*, como lema, verbos, substantivos e informações de conjugação. E também, é realizada a análise sintática, criando uma árvore de dependência que indica a relação entre as palavras no texto, dessa forma garantindo a estrutura gramatical (SPACY, 2023). E por fim, é realizado o reconhecimento de entidades no texto, em que uma entidade é um objeto real que pode ser identificado com um nome próprio, podendo ser uma pessoa, uma organização, um local, etc. Entidades são úteis para gerar contexto com o que o usuário está a dizer (VASILIEV, 2020).

2.2 Rasa

Há diversas formas de desenvolver um chatbot. Uma delas é utilizar frameworks especializados nisso, como o Rasa. O Rasa é um *framework* utilizado para criar assistentes virtuais, sua versão *open-source* é uma plataforma de IA conversacional que pode ser integrada com sistemas de mensagem. A inteligência artificial disponibilizada é capaz de entender e manter conversas com os usuários (RASA, 2023).

Esse *framework* segue uma arquitetura simples, uma mensagem é recebida, ela é interpretada, é escolhida uma ação, essa ação é executada e então retorna uma resposta (BOCKLISCH et al., 2017). O Rasa possui dois principais componentes, o Rasa NLU (*Natural Language Understanding*) responsável por receber os dados de entrada da interação e o Rasa Core que executa o processamento dos dados recebidos (SHARMA et al., 2020).

> Ambos Rasa NLU e Core trabalham com dados de treinamento legíveis por humanos. Rasa NLU requer uma lista de enunciados anotados com intenções e entidades. Estes podem ser estruturados tanto no formato json como em *markdown* (BOCKLISCH, 2017, p. 4, tradução nossa).

E para auxiliar no desenvolvimento de um chatbot, utiliza-se um diagrama de fluxo conversacional. Um fluxo conversacional é uma sequência de interações entre um usuário e um chatbot, onde o chatbot faz perguntas e fornece respostas com base nas intenções do usuário e nas informações armazenadas em sua base de conhecimento. O objetivo é guiar o usuário para uma solução ou resposta específica (NEO, 2020).

Assim, um fluxo de diálogo é um conjunto de perguntas e respostas envolvidos em um contexto (NEO et al., 2023) e que também é uma representação visual de como uma conversa entre um usuário e um chatbot pode ocorrer. Ele mostra a sequência de perguntas e respostas que podem ocorrer durante a interação (NEO, 2020).

O Rasa é responsável por processar as mensagens de texto que um chatbot recebe dos usuários. Ele lida com a tarefa de compreender a intenção do usuário, identificar entidades e extrair informações relevantes em um texto. No Rasa NLU, as intenções representam os objetivos das mensagens do usuário que são classificadas automaticamente nas intenções correspondentes de acordo com o treinamento realizado (RASA, 2023). O Rasa NLU utiliza aprendizado de máquina para treinar os modelos que podem classificar as intenções e identificar as entidades nas mensagens dos usuários. Isso envolve a alimentação do sistema com exemplos de mensagens que contém intenções e entidades rotuladas. O sistema usa dados de treinamento para aprender a fazer previsões e ainda há flexibilidade na configuração dos modelos de PLN. É possível personalizar parâmetros como algoritmos de aprendizado, recursos de extração de entidades e estratégias de processamento de texto para atender às necessidades em um arquivo de configuração. Nesse contexto, pode-se utilizar a biblioteca spaCy (RASA, 2023).

Para fins de treinamento do modelo, utilizam-se as histórias e as regras em conjunto com os dados de NLU, que indicam casos em que as intenções são identificadas, quais respostas são adequadas a cada intenção e ações a serem executadas. As histórias simulam casos reais de interação, que demonstram como uma sequência de eventos pode ocorrer. Nelas não são inseridos dados de uma mensagem real, mas utilizam-se as intenções declaradas no NLU, fazendo com que o modelo treinado possa fazer predições das próximas ações a serem tomadas. Além disso, as regras ditam as ações que devem ocorrer para determinados cenários. Elas se assemelham às histórias, porém são úteis para cenários fixos onde uma ação deve ocorrer após um evento específico (RASA, 2023). Assim como é possível também o desenvolvedor corrigir tomadas de decisão feitas pelo sistema, possibilitando uma margem maior de aprendizado. Ainda, o Rasa disponibiliza uma visualização de gráficos de acordo com as tomadas de decisão dos nós (BOCKLISCH et al., 2017).

Para completar o processo do Rasa, há ainda as *custom actions* que fazem um papel fundamental para o chatbot. Elas são responsáveis por fazer conexões externas com APIs (*Application Program Interface*), acessar bancos de dados, etc. O desenvolvimento delas é feito em Python e proporciona completude ao chatbot (RASA, 2023).

2.3 MongoDB

O MongoDB é um banco de dados NoSQL, que é mais vantajoso em relação a outros bancos de dados SQL ao trabalhar com uma alta carga de dados, já que um banco relacional possui limitações ao lidar com dados não estruturados e complexos em grande escala. Nesses bancos os dados não são armazenados da mesma forma que em um banco relacional. Bancos NoSQL são livres de *schemas.* Além disso, o Mongo utiliza documentos para armazenamento sendo que, cada um possui uma identificação por chave (GYŐRÖD et al., 2022).

O Mongo é um banco orientado a documentos, ele disponibiliza um sistema inteligente de desempenho otimizado, o que se faz útil para ter uma boa performance com altas cargas de dados. Algumas das vantagens de utilizar um banco de dados orientado a documentos são: o suporte a polimorfismo, a redução na necessidade de *joins* e o suporte a diversas linguagens de programação (MONGODB, 2023a).

Esse banco de dados tem como características a alta performance, o que entrega agilidade, alta disponibilidade, já que ele possui um conjunto de réplicas que possibilita a redundância de dados, além da escalabilidade horizontal, que garante o crescimento do banco em um sistema (CHAUHAN, 2019). Outro ponto importante que garante a eficiência no Mongo é a presença de índices. A sua utilização torna as buscas mais eficientes nos documentos de uma coleção, por não precisar escanear uma coleção inteira para retornar um resultado. Havendo um índice, o banco reduz o número de documentos a serem escaneados (MONGODB, 2023a). Isso faz com que as buscas em documentos indexados sejam mais eficientes.

2.4 Java

O Java é uma linguagem de programação orientada a objetos que possui práticas como "*Write Once Run Anywhere (WORA)*" (escreva uma vez, rode em qualquer lugar). Logo, é utilizado em diversas plataformas, desde dispositivos móveis até grandes sistemas empresariais (ORACLE, 2023).

Integrado a uma linguagem de programação estabelecida e desempenhando um papel auxiliar no processo de elaboração de Interfaces de Programação de Aplicações (APIs), o Spring Boot direciona a atenção do desenvolvedor para a definição de regras de negócio e a produção de código. Esta abordagem delega a responsabilidade pela infraestrutura ao framework Spring, cujo propósito é proporcionar uma metodologia eficiente para a criação de aplicações (SPRING, 2023a).

A utilização do framework Spring simplifica o desenvolvimento de APIs em Java. Esta ferramenta, quando aplicada em uma arquitetura monolítica, como descrito por Villamizar et

al. (2017), centraliza todas as funcionalidades, que englobam desde a lógica de aplicação até as operações de acesso ao banco de dados e chamadas externas, em um único sistema integrado. Isto proporciona uma estrutura unificada e coesa, facilitando a gestão e a manutenção do código, ao mesmo tempo em que aproveita as capacidades robustas e eficientes do Spring para o desenvolvimento de APIs.

2.5 Scrum

Os Métodos Ágeis tornaram-se populares por volta de 2001 com o estabelecimento do manifesto ágil, que traz como princípios principais os indivíduos, software funcionando, colaboração com o cliente e responder a mudanças. Esses princípios são levados em consideração para criar um fluxo de processos mais ágil em um âmbito empresarial em que a competitividade está presente em alto nível, sendo crucial para a adaptação e respostas rápidas à volatilidade do mercado (PRIKLADNICKI; WILLI; MILANI, 2014). Então, metodologias ágeis possuem o maior foco em pessoas e em processos para que o resultado final seja melhor.

Tendo isso em vista, pode-se citar o Scrum, que é um framework ágil utilizado por empresas para a previsibilidade e controle de riscos de um ou mais projetos. Ele é constituído por eventos Scrum que são a *sprint* (um período que dura, normalmente, entre duas e quatro semanas em que se executam as tarefas propostas), reuniões diárias, reunião de planejamento, reunião de revisão e reunião de retrospectiva. Além dos eventos, há o time Scrum, que nele estão o dono do produto, o time de desenvolvimento e o Scrum Master. E por fim, os artefatos Scrum, que são o *Backlog* do produto (lista de atividades propostas para entregar um produto), *Backlog* da *Sprint* (lista de atividades propostas para se fazer na *sprint*) e o incremento (soma de todas as tarefas do *Backlog*) (SCHWABER, 2017).

Tendo em vista que o Scrum é uma metodologia em time, para adequar-se em um time de apenas um desenvolvedor utiliza-se o Scrum solo, que de acordo com Pagotto et al. (2012) “se caracteriza como um processo iterativo e incremental que une as boas práticas delineadas pelo Personal Software Process e pelo Scrum”.

No Scrum solo são seguidos alguns artefatos e atores presentes no Scrum, como *Product Backlog*, *Sprints*, *Sprint Backlog*, *Product Owner*, além do próprio desenvolvedor e do cliente. O que diferencia as duas metodologias é principalmente a condução dos artefatos. O tempo de duração de uma *sprint*, por exemplo, no método solo é em torno de uma semana e requer no final um protótipo de uma nova funcionalidade desenvolvida. Além disso, algumas cerimônias do scrum não são utilizadas, como a *planning*, a *daily* e a retrospectiva. Mas ainda há a constante

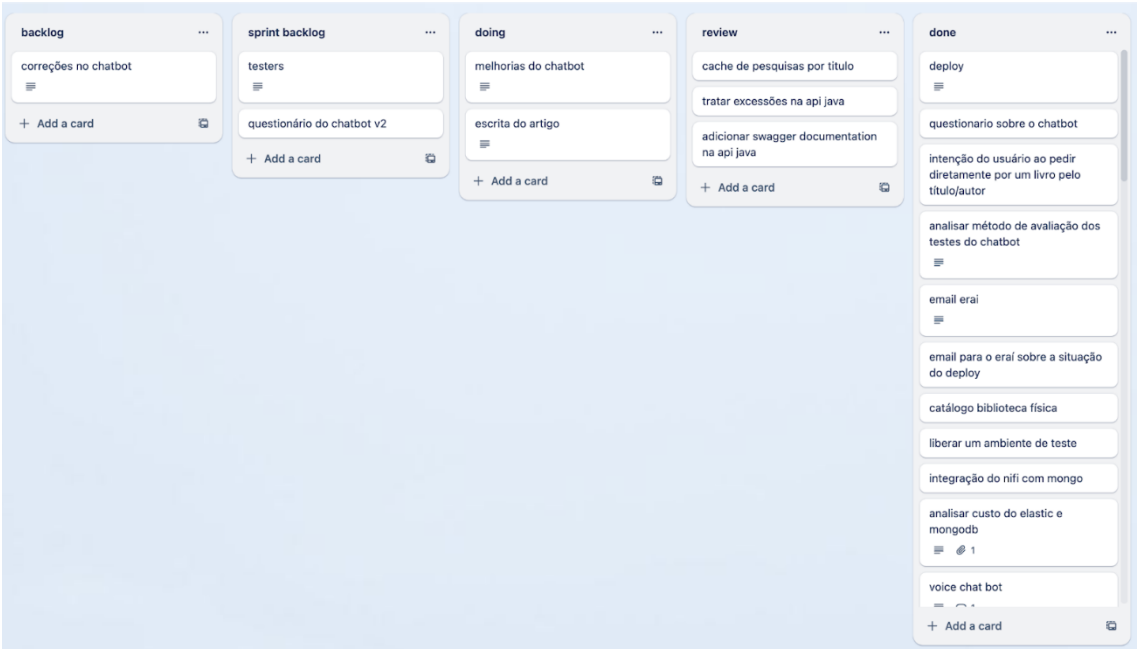
comunicação entre o desenvolvedor e o cliente para atender o escopo definido (PAGOTTO et al., 2012).

3 DESENVOLVIMENTO DO SISTEMA

3.1 Metodologia

Utilizou-se o Scrum Solo como metodologia, garantindo um fluxo de trabalho organizado para apenas um desenvolvedor que responde para uma equipe pequena, composta por um cliente e um orientador.

Figura 1 – Quadro Kaban



Fonte: A pesquisa (2023)

Em conjunto com o Scrum Solo, utilizou-se um quadro Kanban para organizar e monitorar as atividades propostas para cada *sprint*, havendo ainda, reuniões semanais entre o desenvolvedor e o orientador buscando sempre um resultado ao final de cada ciclo. Como pode ser visto na figura 1, o quadro Kanban foi dividido em colunas de *backlog*, *sprint backlog*, *doing*, *review* e *done*. A primeira coluna refere-se a atividades planejadas que ainda devem ser realizadas para o projeto. A segunda coluna comporta as atividades que devem ser realizadas durante a *sprint* atual, enquanto a terceira é a atividade que está sendo executada no momento. Ainda, a coluna *review* comporta as atividades a serem revisadas, para correções ou alterações.

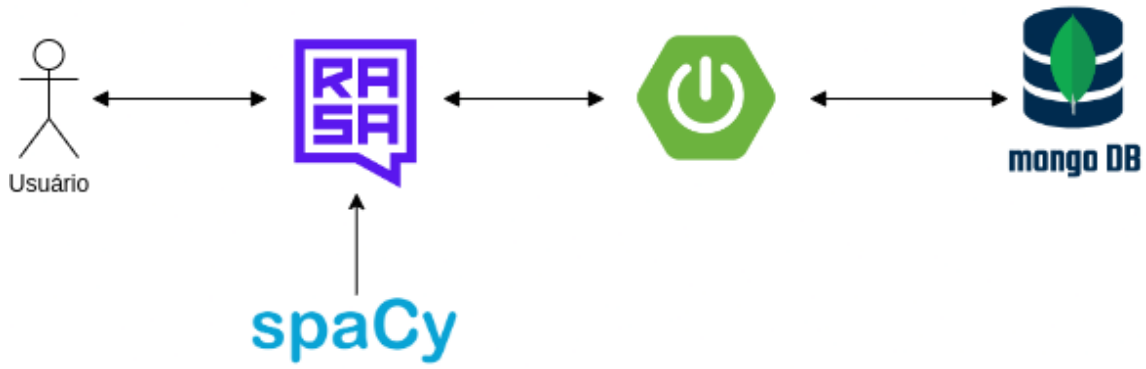
Se for constatado que ainda deve ser feito algo em alguma atividade, ela volta para a coluna de *sprint backlog* ou de *doing*. E por fim, a coluna *done* que indica as atividades já finalizadas da *sprint* e do projeto.

3.2 Análise

3.2.1 Arquitetura

A arquitetura utilizada no desenvolvimento do projeto é composta por três componentes principais: o chatbot como ponto de entrada e comunicação com o usuário, um banco de dados com os catálogos separados de cada biblioteca e para o *backend* uma API para fazer a conexão do chatbot com o banco, como é possível observar pela figura 2.

Figura 2 – Diagrama da arquitetura do sistema



Fonte: A pesquisa (2023)

Essa arquitetura é simples e performática visto que o Rasa (seção 2.3) fornece customização do modelo, sendo versátil e flexível em processar linguagem natural. Ele oferece uma estrutura completa para desenvolver chatbots. O Rasa NLU, mais especificamente, permite que o sistema compreenda as intenções nas mensagens dos usuários, sendo essencial para a interação. Aliado a isso, a integração com a biblioteca spaCy melhora o processamento de texto, sendo um elemento que traz precisão em identificar entidades por sua eficiência em análise linguística.

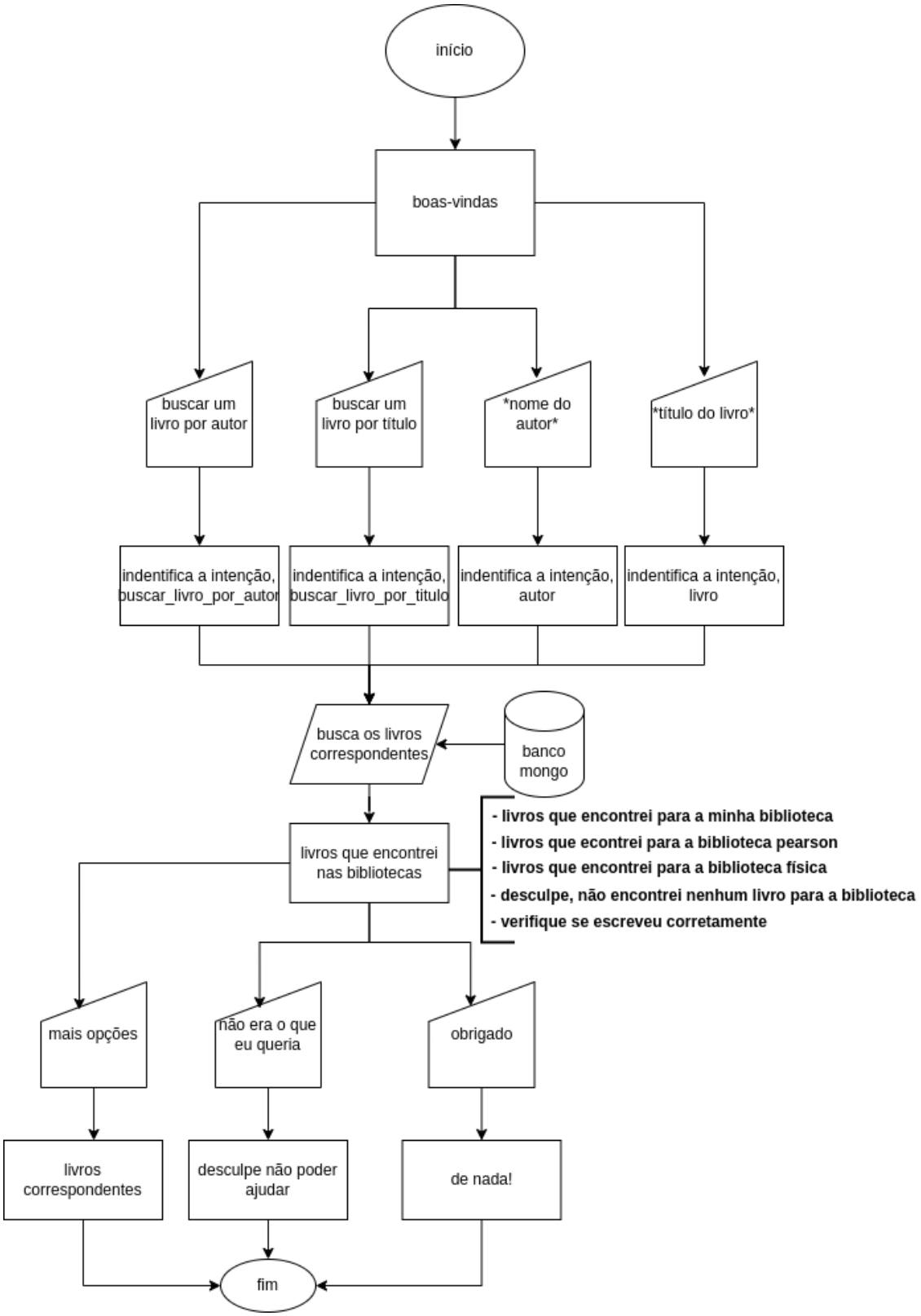
Ainda, a API Java garante o acesso aos dados do banco Mongo permitindo ao chatbot funcionalidades específicas, como buscar um livro, que não é uma funcionalidade disponibilizada diretamente pelo Rasa. E para isso, o MongoDB garante escalabilidade e flexibilidade, por permitir o armazenamento eficiente de dados não estruturados e buscas por

texto completo com o auxílio dos índices. Dessa forma, essa arquitetura mostrou-se eficiente por proporcionar um chatbot responsivo e capaz de fornecer respostas de acordo com as necessidades do usuário.

3.2.1 Agente Conversacional

O diagrama da figura 3 ilustra o processo de design de conversa de um chatbot construído com o Rasa, com o objetivo de fornecer ao usuário uma listagem de livros baseada na sua pesquisa, seja ela pelo título ou pelo autor da obra. Esse fluxo de diálogo (MEDIUM, 2020) é planejado para permitir que o usuário inicie a interação e especifique suas preferências de busca de forma direta ou indireta, através de um diálogo. Por exemplo, o usuário pode expressar “pesquisar um livro pelo título” ou pedir ao chatbot “encontre um livro da autora Gilza Helena” ou ainda pedir diretamente pelo livro “O Pequeno Príncipe”. Em seguida, o chatbot busca os livros correspondentes com a pesquisa do usuário, sendo que ele pode retornar algum livro, não retornar livro algum ou retornar algum livro incorretamente. Esses cenários são mapeados para garantir que a sequência do diálogo ocorra mesmo em casos indesejados.

Figura 3 – Fluxo conversacional do chatbot



Fonte: A pesquisa (2023)

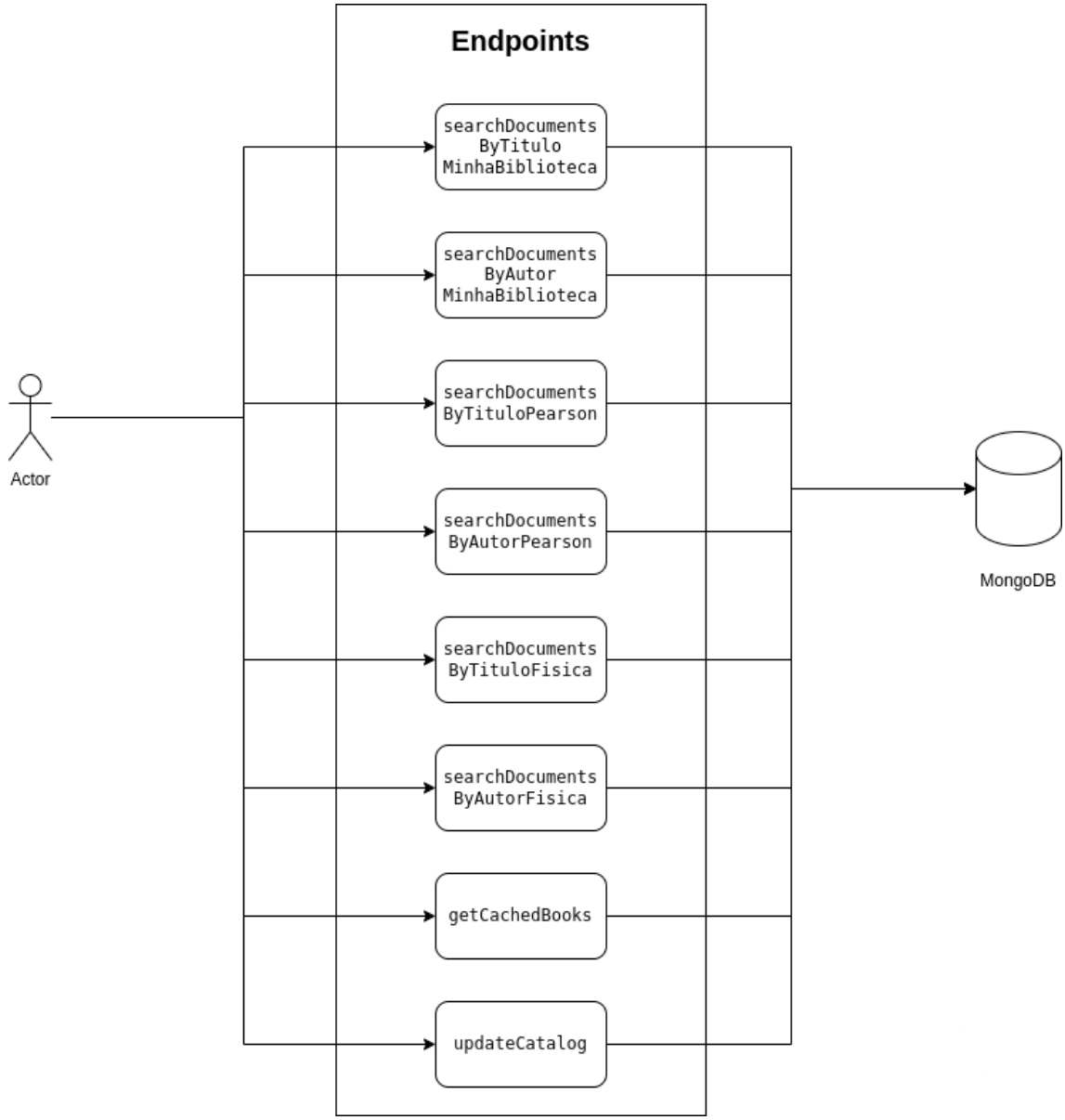
Para atender a essas solicitações, o chatbot utiliza intenções definidas no Rasa NLU, que são essenciais para interpretar corretamente a necessidade do usuário de acordo com as mensagens enviadas. A cada intenção identificada, o chatbot responde de forma a manter o diálogo fluindo ou para fornecer as informações buscadas sobre o livro. Para devolver os resultados dos livros durante a interação, o Rasa emprega “custom actions”, que permitem integrar uma API Java externa, assim como utilizar o spaCy para identificação de títulos e autores.

Ainda, as *stories*, *rules* e o componente NLU trabalham em conjunto no treinamento da inteligência artificial fornecida pelo Rasa, capacitando o chatbot a identificar as intenções dos usuários e a seguir o fluxo de diálogo mapeado, assim como acessar a API externa conforme as ações definidas.

3.2.2 Api

A API Java Spring Boot foi projetada para facilitar a interação entre o chatbot e o banco de dados MongoDB, que contém dados sobre os livros disponíveis para consulta. A arquitetura da API segue um modelo monolítico, uma escolha de acordo com a sua facilidade de implementação.

Figura 4 – Diagrama do backend



Fonte: A pesquisa (2023)

Conforme ilustrado na figura 4, a API estrutura seus *endpoints* seguindo uma lógica dividida por contexto de busca por biblioteca e por parâmetro de pesquisa, como título e autor. Este design organiza as funções de pesquisa dentro de categorias específicas, facilitando o mapeamento de requisições para ações correspondentes no banco de dados.

Além dos *endpoints* padrão de busca, dois outros se destacam por suas funcionalidades especializadas. O primeiro, “getCachedBooks”, serve para trazer mais opções de livros para a busca já realizada anteriormente. O segundo, “updateCatalog”, é utilizado para manter o

catálogo de livros atualizado, permitindo que funcionários da FACCAT mantenham os catálogos das bibliotecas atualizados.

3.2.3 Banco de dados

O MongoDB desempenha um papel crucial no sistema do chatbot. Sua escolha foi motivada por características como flexibilidade de esquema, escalabilidade e performance ágil em operações de leitura e escrita, essenciais para a interação com um chatbot que precisa acessar rapidamente uma vasta quantidade de informações. A estrutura não relacional do MongoDB é especialmente adequada para o armazenamento de catálogos de livros, pois permite uma grande variedade de consultas. Livros com diferentes atributos, como edições variadas, podem ser facilmente acomodados e pesquisados sem as restrições de um esquema rígido, como seria o caso de um sistema relacional.

Além disso, o MongoDB oferece recursos de indexação que são utilizados para acelerar as buscas por título e autor, as duas funcionalidades do chatbot. A criação de índices adequados é vital para garantir respostas rápidas e trazer uma abrangência para as buscas, já que executam pesquisas *full text search* (busca por texto completo), organizando o retorno da busca por uma pontuação *score*. A integração entre o MongoDB e a API Java Spring Boot garante eficiência nas buscas pelos livros, sendo essencial para uma boa interação do chatbot com o usuário.

4 DESENVOLVIMENTO E TESTES

4.1 Desenvolvimento

Para o desenvolvimento do chatbot com o Rasa, uma etapa essencial foi a definição de intenções, estruturadas no arquivo “nlu.yml”. Neste arquivo cada intenção é declarada com um título para sua identificação e com algumas frases de exemplo. Esses exemplos alimentam o algoritmo de treinamento da IA para que ela possa reconhecer padrões de frases como “procure um livro” ou “quero um livro”. O design do fluxo de diálogo foi planejado para incluir intenções que facilitam uma interação fluida, tais como “saudacao”, “agradecimento”, “despedida”, entre outras. Além das intenções específicas para a busca de livros por título e por autor.

E para que o Rasa possa entender as intenções de um usuário em uma interação, foram criadas as *stories* e as *rules*. Enquanto as *stories* mapeiam sequências de interações esperadas, indicando a intenção seguida da ação correspondente, as *rules* seguem uma linha parecida,

porém descrevendo cenários estáticos que devem ocorrer. Como exemplo, quando a intenção "saudacao" é identificada, a ação "utter_saudacao" sempre deve ser executada após, sendo uma resposta ao usuário.

O diálogo é aprimorado através das *custom actions* do Rasa, que permitem a execução de chamadas para uma API externa. Essas ações comunicam-se com o ambiente do Rasa conforme o necessário, como exemplo, preencher *slots* durante o diálogo com o usuário para então continuar sua execução. (Rasa, 2023) Os *slots* são formas que o Rasa utiliza para armazenar valores que foram colhidos em uma conversa. Esse mapeamento pode ocorrer por identificação do próprio Rasa ou pode ser manipulado por uma *custom action*.

Neste contexto, foram criados três *slots* "titulo_livro", "nome_autor" e "session_id". O *slot* "session_id" armazena um identificador de sessão do chat atual, o *slot* "titulo_livro" armazena o título de um livro informado e o "nome_autor" armazena o nome de um autor. Importante ressaltar que nesse caso, essas variáveis não são preenchidas diretamente pelas intenções especificadas no Rasa, mas sim pelas *actions* que processam a mensagem enviada pelo usuário e extraem as informações necessárias. A ação que executa esse processo de preenchimento dos *slots* é denomida "action_extrai_nome_de_livro_ou_nome_de_autor". Essa ação é essencialmente significativa, pois ela analisa a mensagem do usuário para extrair o nome de um livro ou de um autor, identificando-os como entidades do tipo BOOK ou PER, respectivamente. Com essas entidades identificadas e os *slots* correspondentes preenchidos, o chatbot está então capacitado para realizar buscas nos catálogos de livros, interagindo com a API externa para entregar resultados ao usuário. A ação de chamada para a API ocorre quando um dos *slots* "titulo_livro" ou "nome_autor" é preenchido, sendo esse o ponto chave para que ocorra essa execução.

Embora o Rasa identifique entidades e intenções, ele enfrenta uma complexidade maior em identificar nomes de pessoas e nomes de livros em frases. Mesmo com o treinamento específico nesse cenário, essa complexidade não diminuiu e o chatbot enfrentava dificuldades em identificar e separar essas duas entidades, principalmente nomes de pessoas. Para melhorar a identificação de nomes de autores, a biblioteca de PLN spaCy foi integrada ao sistema. Com seu modelo pré-treinado, *pt_core_news_md*, (SPACY, 2023, tradução nossa) "spaCy pode reconhecer vários tipos de entidade em um documento", como nomes de pessoas (PER). Assim, ele foi designado para lidar com a identificação de nomes de autores dentro das frases recebidas no chatbot.

Dessa forma, a complexidade de identificação de nome de autor com o modelo NLU do Rasa foi abstraída para o spaCy, que é utilizado nas *custom actions* para identificar a entidade.

Entretanto, a identificação de títulos de livro se mostrou um desafio mesmo para o spaCy, devido à diversidade e variabilidade dos títulos. Foi preciso a implementação de um treinamento adicional sobre o mesmo modelo utilizado (*pt_core_news_md*) para reconhecer a entidade BOOK (criada pelo autor) e para isso, um *dataset*[5] contendo cerca de 11.000 títulos de livros foi utilizado. O processo de treinamento aplicou o algoritmo de *backpropagation* do próprio spaCy, com o objetivo de minimizar os erros e melhorar a identificação de entidades em diversos contextos. O modelo foi exposto a esses dados ao longo de 42 épocas, com uma abordagem de aprendizagem genuína ao invés de uma memorização de padrões. Realizado esse treinamento, utilizou-se esse modelo refinado no spaCy juntamente ao Rasa. Assim, a tarefa de identificação de títulos de livro ficou a cargo do spaCy da mesma forma que a identificação de nomes de autores, enquanto o Rasa NLU faz a identificação das intenções do usuário.

Ainda com as tarefas bem divididas entre Rasa e spaCy, para que o Rasa NLU pudesse identificar as intenções de forma mais exata, foi preciso obter um volume de dados de títulos de livros e de nomes de autores para treinar o modelo disponibilizado pelo Rasa, com cerca de 1000 exemplos para cada. Dessa forma, o Rasa consegue identificar a intenção do usuário e então executar a ação que extrai a entidade PER ou BOOK, com o auxílio do spaCy e por fim, a consulta é encaminhada para a API.

Na API desenvolvida, foi integrada a funcionalidade de pesquisa de texto completo (*full text search*) do MongoDB, que é uma funcionalidade chave, especialmente otimizada para os campos “titulo” e “autor” nos catálogos das três bibliotecas. Cada biblioteca é representada por uma coleção separada dentro do banco de dados, onde cada livro é tratado como um documento individual. Esta segmentação permite que as buscas sejam realizadas de maneira específica e abrangente dentro de cada coleção.

A habilidade do MongoDB em executar pesquisas de texto completo se destaca por sua precisão e rapidez, mesmo com uma grande quantidade de documentos. Ao realizar uma busca, o sistema examina os campos indexados por meio de tokenização, onde o texto é dividido em palavras-chave, que são usados para localizar os documentos correspondentes. Esta indexação aumenta a performance das buscas, pois os índices podem ser percorridos rapidamente para encontrar os termos desejados (MONGODB, 2023b) .

Para a interação com o banco de dados, a API utiliza o Spring Data MongoDB que através dele, moldam-se as consultas de texto completo que o MongoDB executa. As consultas são construídas para tirar vantagem dos índices de texto, resultando em operações de busca que

[5] https://www.kaggle.com/datasets/victorstein/livros-skoob

são, não apenas rápidas, mas também, precisas e abrangentes. Além disso, o MongoDB assegura a relevância dos resultados das buscas por meio da pontuação de texto, calculando-a com base na frequência com que os termos da pesquisa aparecem nos documentos. Esta abordagem de classificação dos resultados de busca garante que os usuários tenham acesso rápido e informações abrangentes com o que procuram, potencializando a experiência do usuário final.

Desse modo, a API Java desempenha um papel fundamental nesse processo que é garantir o tráfego dos dados entre o chatbot e o banco de dados de forma precisa e veloz. Uma arquitetura monolítica foi projetada com esse intuito, para que o próprio desenvolvimento fosse rápido assim como a execução do sistema. Aliado ao bom desempenho que o MongoDB apresenta realizando as buscas indexadas, o sistema consegue desempenhar bem e entregar os resultados de forma rápida ao usuário sem que ele precise ficar um longo tempo esperando por uma resposta.

Os *endpoints* foram divididos em categorias de acordo com a biblioteca em que é feita a busca e o termo de pesquisa, um título ou um autor. Dessa forma cada endpoint possui um único contexto o que possibilita o chatbot lidar com cada contexto específico de forma separada. Ainda, seguindo o padrão do Spring MVC (*Model*, *View*, *Controller*) (SPRING, 2023b) a api ficou dividida em *controller*, *services*, *models* e *repositories*.

Contando que uma busca *full text search* no Mongo retorna diversos dados, até mesmo alguns indesejados, foi desenvolvido um esquema de validação para os livros retornados. Para requisições realizadas por autor, é feito um filtro dos livros que possuem o autor condizente com o termo pesquisado, não havendo brechas para que livros de autores que não se identificam com o termo de pesquisa sejam retornados também. Em contrapartida, esse filtro torna a interação do usuário mais engessada com uma escrita mais precisa do termo que deve ser pesquisado. Essa abrangência é mitigada quando se trata de buscas por título. Levando em consideração que em uma busca por um título pode haver algumas unidades semelhantes ao termo pesquisado, há interesse de retorno ao usuário também desses resultados, sendo possível que os livros semelhantes ao termo pesquisado também estejam contidos na resposta. Como essa busca mais abrangente pode retornar uma quantidade maior de livros, foi adotada a estratégia de retornar apenas 5 livros em uma primeira pesquisa e armazenar o restante em tabelas de *cache*. Quando há *cache* de livros armazenados, o chatbot apresenta para o usuário a opção de buscar mais opções de livros.

E por fim, há o fluxo para a atualização dos catálogos de cada biblioteca. Sendo necessário a indicação de qual biblioteca está sendo atualizada e a inserção do arquivo com o

catálogo de livros. O serviço realiza a deleção do catálogo atual, para então executar a inserção do novo e então a criação dos índices para os campos “titulo” e “autor”.

4.2 Testes

Uma sessão de testes foi realizada com a participação de 27 usuários, incluindo 24 alunos, 1 professor e 2 funcionários, todos da FACCAT sendo que os alunos estavam divididos entre os cursos de Ciências Contábeis, Design, Direito, Enfermagem, Engenharia de Produção, Fisioterapia, Gestão da Qualidade, Pedagogia, Psicologia e Publicidade e Propaganda. Esta diversidade de participantes foi crucial para avaliar a usabilidade e a intuitividade do chatbot. Assim também como é possível observar pelas perguntas 1 e 2 (APÊNDICE A) que a maioria dos usuários de teste são pessoas entre 18 e 24 anos.

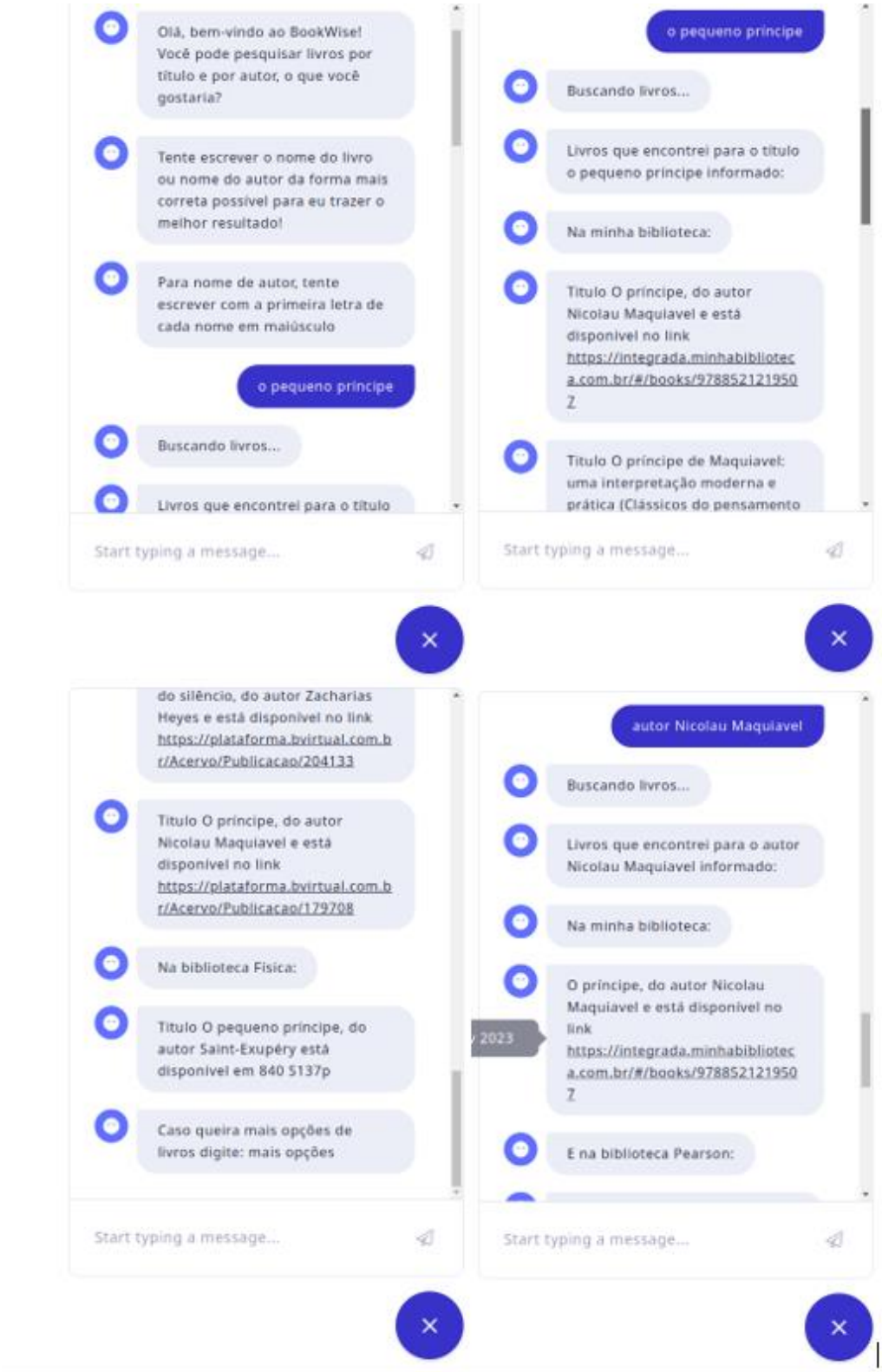
O teste foi estruturado em duas etapas principais. Na primeira etapa, o chatbot foi disponibilizado para os participantes sem quaisquer instruções prévias. Os usuários foram incentivados a interagir com o sistema por um período máximo de 10 minutos. Este procedimento tinha como objetivo avaliar a intuitividade e facilidade de uso do chatbot, observando como os usuários utilizavam a aplicação.

Após a interação direta com o chatbot, os participantes foram convidados a responder um questionário (APÊNDICE A). Este questionário foi elaborado para coletar feedbacks específicos sobre a experiência do usuário e entender a eficácia do chatbot em fornecer as informações solicitadas.

5 RESULTADOS ALCANÇADOS

O chatbot desenvolvido realiza a função de busca de livros nas bibliotecas disponibilizadas pela FACCAT, as virtuais Minha Biblioteca e a Biblioteca Pearson, assim como a biblioteca física no campus da faculdade. As buscas de livros podem ser realizadas por título ou por autor e além das buscas, o chatbot é capaz de manter uma conversa com o usuário neste contexto, como exemplo na figura 5.

Figura 5 – Interação com o chatbot



Fonte: A pesquisa (2023)

Levando em consideração que quase a metade dos usuários não utilizaram um chatbot anteriormente (pergunta 4, APÊNDICE A), os resultados com a interação demonstraram-se

positivos, como é possível observar com as respostas das perguntas 5, 6 e 7 (APÊNDICE A). Embora hajam avaliações negativas, os resultados se mostraram satisfatórios, demonstrando que o chatbot realmente é útil tanto para os alunos quanto para professores e funcionários, como é possível observar pelas perguntas 8 e 9 (APÊNDICE A) onde mais de 74% dos usuários relataram que o chatbot BookWise facilitou as buscas de livros no ambiente da FACCAT assim como mais de 81% deles indicaria para alguém utilizá-lo.

Em relação ao que foi descrito sobre possíveis melhorias e o que os usuários não gostaram no chatbot, em sua grande maioria está relacionado com a interface de teste utilizada. Essa interface foi utilizada apenas para teste, sendo que o objetivo final do projeto é a sua implementação no ambiente virtual da FACCAT, sendo um item que depende então da faculdade.

Dessa forma, levando em consideração os testes realizados com a coleta de avaliações dos usuários, é possível concluir que o objetivo do projeto foi alcançado assim como o problema foi resolvido.

6 CONSIDERAÇÕES FINAIS

Neste projeto foi criado o chatbot BookWise para unificar o acesso às três bibliotecas da FACCAT: as bibliotecas virtuais Minha Biblioteca e Biblioteca Pearson, e a Biblioteca Física no campus. Os testes realizados com uma variedade de usuários incluindo alunos, professores e funcionários, indicam que o chatbot atendeu com sucesso ao seu objetivo principal. O BookWise facilita a pesquisa de livros por título ou por autor através de um sistema de conversação por texto, adaptado ao ambiente digital da faculdade.

Embora o projeto tenha alcançado suas metas iniciais, identificou-se uma área para melhoria: a acessibilidade para usuário com deficiência. Isso abre caminho para pesquisas futuras focadas no desenvolvimento de funcionalidades de conversação por voz. Além disso, há potencial para expansão, incluindo a implementação de uma busca avançada de livros com filtros personalizados e a capacidade de pesquisar livros em uma biblioteca específica, incluindo informações sobre a disponibilidade de exemplares.

Em resumo, este projeto conseguiu integrar com sucesso as três bibliotecas da FACCAT, oferecendo uma ferramenta valiosa e eficiente para a comunidade acadêmica na busca por livros.

REFERÊNCIAS

ABU SHAWAR, Bayan; ATWELL, Eric. Chatbots: Are they Really Useful? **Journal for Language Technology and Computational Linguistics**, v. 22, n. 1, p. 29–49, 2007.

ADAMOPOULOU, E.; MOUSSIADES, L. An Overview of Chatbot Technology. *In*: MAGLOGIANNIS, I.; ILIADIS, L.; PIMENIDIS, E. (Eds.). **Artificial Intelligence Applications and Innovations**. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, 2020. v. 584, p. 373–383.

BARNEY JR, Michael. **Fluxogramas para Interfaces Conversacionais.** 2020. Dísponível em: https://www.google.com/url?q=https://medium.com/botsbrasil/fluxogramas-para-interfaces-conversacionais-16e8e16a7fca&sa=D&source=docs&ust=1700762508859450&usg=AOvVaw3gKJtnu69Ohl24ugiAyJUF. Acesso em: 02 nov. 2023.

BOCKLISCH, Tom; FAULKNER, Joey; PAWLOWSKI, Nick; *et al*. Rasa: **Open Source Language Understanding and Dialogue Management.** 2017. Disponível em: https://arxiv.org/abs/1712.05181. Acesso em: 02 nov. 2023.

CHAUHAN, Anjali. A Review on Various Aspects of MongoDb Databases, **INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT)**. v. 8, Issue 05, 2019.

GYŐRÖDI, Cornelia A.; DUMŞE-BURESCU, Diana V.; ZMARANDA, Doina R.; *et al*. **A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management**. **Big Data and Cognitive Computing**, v. 6, n. 2, p. 49, 2022.

LOPEZ, Marc Moreno; KALITA, Jugal. **Deep Learning applied to NLP.** 2017. Disponível em: https://arxiv.org/abs/1703.03091. Acesso em: 08 out. 2023.

NEO, Giseldo S.; BARROS, Evandro C.; NEO, Alana V. B. S.; *et al*. **Integrando BPMN e AIML para construção de fluxos de diálogo para Chatbots.** 2023. Disponível em: https://sol.sbc.org.br/index.php/semish/article/view/25087. Acesso em: 14 nov. 2023**.**

NEO, Giseldo S. **Construção de chatbots AIML com a ajuda de uma ferramenta de modelagem visual baseada na linguagem BPMN.** 2020. Disponível em: https://www.repositorio.ufal.br/handle/riufal/7266 . Acesso em: 10 nov. 2023.

MONGODB. **MongoDB**. Disponível em: https://www.mongodb.com/docs/manual/introduction/. Acesso em: 19 out. 2023a.

MONGODB. **MongoDB**. Disponível em: https://www.mongodb.com/basics/full-text-search. Acesso em: 19 out. 2023b.

ORACLE. **Java**. Disponível em: https://developer.oracle.com/languages/java.html#:~:text=,Master%20Java%20and%20Database. Acesso em: 10 nov. 2023.

PAGOTTO, T. *et al*. **Scrum solo:** Software process for individual development**.** 2016 11th Iberian Conference on Information Systems and Technologies (CISTI). Disponível em: http://ieeexplore.ieee.org/document/7521555/. Acesso em: 14 nov. 2023.

RASA. **Rasa**. Disponível em: https://rasa.com/docs/rasa//. Acesso em: 13 set. 2023.

RAKESH KUMAR SHARMA; NATIONAL INFORMATIC CENTER. **An Analytical Study and Review of open source Chatbot framework, Rasa.** International Journal of Engineering Research and Technology, v. V9, n. 06, p. IJERTV9IS060723, 2020.

SCHWABER, Ken; SUTHERLAND, Jeff. Guia do Scrum. Disponível em: https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Portuguese-Brazilian.pdf. Acesso em 09 out. 2023.

SPACY. **Spacy**. Disponível em: https://spacy.io/usage/spacy-101. Acesso em: 13 out. 2023.

SPRING. **Spring**. Disponível em: https://spring.io/guides/gs/spring-boot. Acesso em: 09 out. 2023a.

SPRING. **Spring**. Disponível em: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html. Acesso em: 09 nov. 2023b.

VASILIEV, Yuli. **Natural Language Processing with Python and spaCy:** A Practical Introduction. Editora No Starch Press, 2020, páginas 1-29.

VILLAMIZAR, M. et al. **Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures**. Service Oriented Computing and Applications, v. 11, n. 2, p. 233–247, jun. 2017.

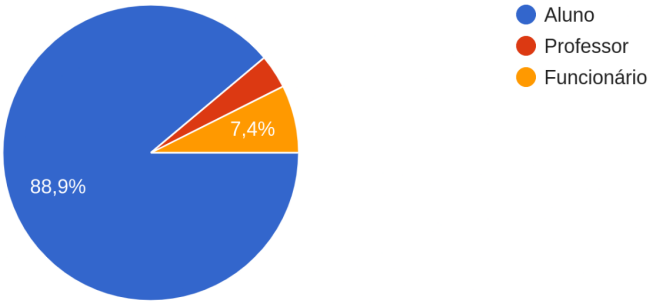
Prikladnicki, Rafael; Willi, Renato; Milani, Fabiano. **Métodos Ágeis para Desenvolvimento de Software.** Editora Bookman, 2014, páginas 3-15.

APÊNDICE A - QUESTIONÁRIO DE AVALIAÇÃO COM OS USUÁRIOS

Questionário de avaliação com o usuário

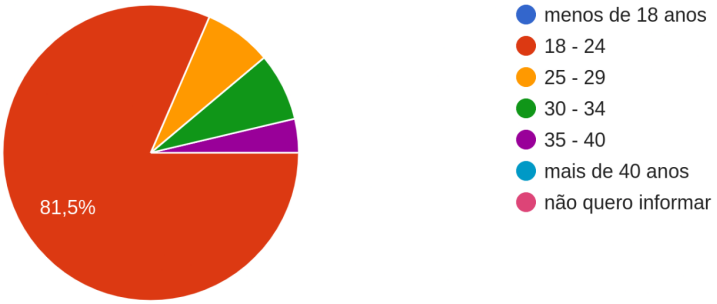
Pergunta 1:

Você é:
27 respostas



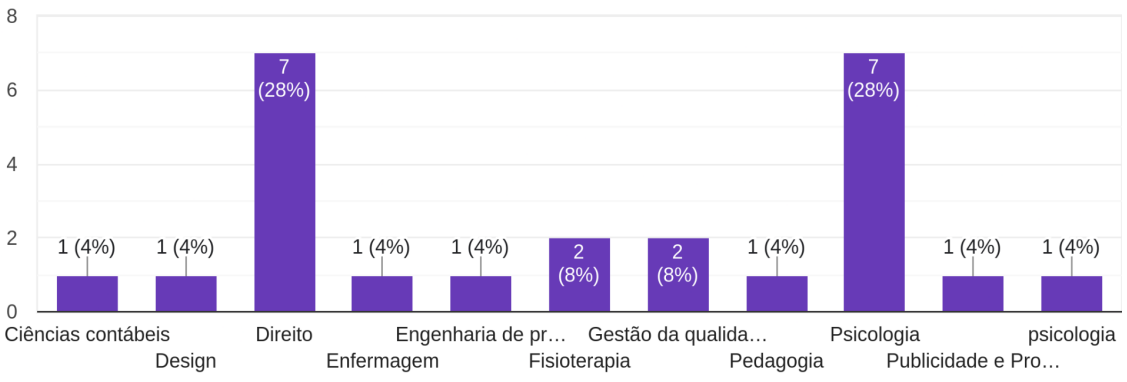
Pergunta 2:

Qual a sua faixa etária?
27 respostas



Pergunta 3:

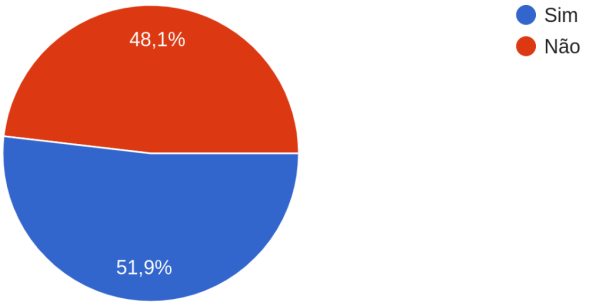
Se você é um aluno, qual o seu curso?
25 respostas



Pergunta 4:

Você já utilizou algum chatbot antes?

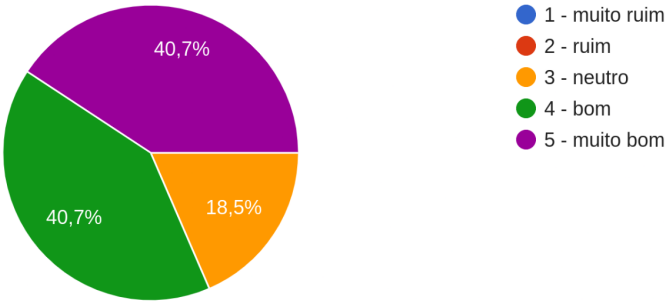
27 respostas



Pergunta 5:

Como você classifica a sua interação com o chatbot BookWise de modo geral?

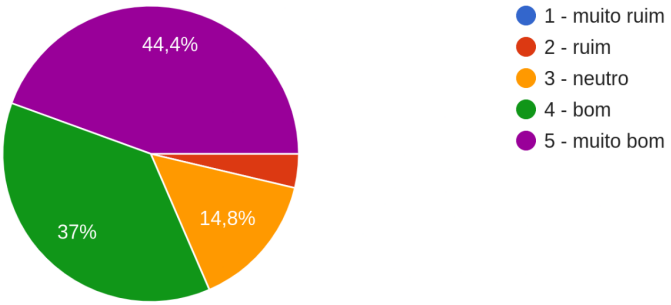
27 respostas



Pergunta 6:

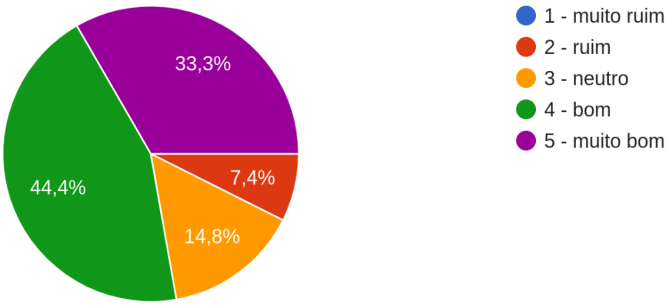
O que você achou das respostas que recebeu do chatbot BookWise?

27 respostas



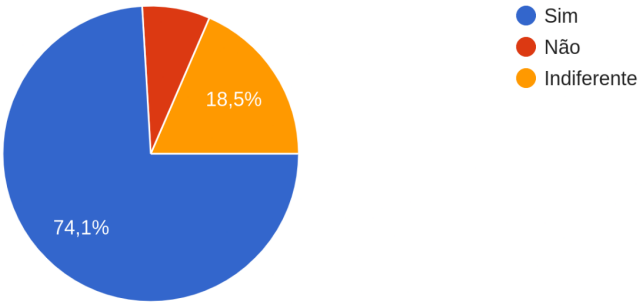
Pergunta 7:

Sobre os resultados obtidos das pesquisas de livros nas bibliotecas, o que você achou?
27 respostas



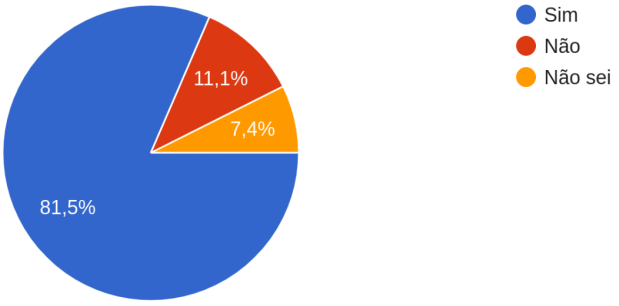
Pergunta 8:

O chatbot BookWise facilitou suas buscas de livros no ambiente da faccat?
27 respostas



Pergunta 9:

Você indicaria o chatbot BookWise para algum aluno da faccat utilizar?
27 respostas



Pergunta 10:

Se sua resposta para a pergunta anterior foi não, por que não?

- Achei a interface um pouco estranha. Podia ser em tela inteira

- Muito complicado a interface e a utilização precisa ser mais simplificado
- O chatbot não trouxe se havia ou não o livro na biblioteca, ele trouxe uma série de análises e não respondeu a pesquisa inicial
- Por que não achou meu livro
- Provavelmente vou esquecer

Pergunta 11:

Campo para sugestões de melhorias

- Em caso de livros ou autores não encontrados (como foi o meu caso quando usei) poderiam ser indicados livros ou autores de gêneros semelhantes. Também um layout mais interativo (acredito que isso será feito mais pra frente)
- Um “balãozinho” maior, para localizar mais facilmente.
- O chat atende ao proposto, quando pesquisei pelo nome de um autor ele fez a busca de vários livros, até mesmo os diversos volumes do mesmo livro, ele atende em todo o estabelecido, mas visando uma evolução do programa é visível que ele tem um grande potencial, então seria interessante aprimora-lo no sentido de realizar uma busca mais ampla até mesmo em outros sistemas, ou ainda, quando clicamos no link entramos no portal da biblioteca mas não conseguimos acessa-la, então seria interessante entrarmos diretamente ao acesso do livro, como um meio prático, ou no acesso a biblioteca para que possamos acessar o livro.
- Muito bom, completo e rápido! Gostei muito, parabéns pelo trabalho!
- O chat é muito relevante, parabéns pela iniciativa. Sugiro deixar a interface do chat mais inclusiva, com ícones/fontes em tamanho maior e cores fortes.
- Não precisa mostrar nas bibliotecas que não foram encontradas, somente o que foi encontrado. E podia ter um filtro, para uma busca avançada.
- O "balão" de acesso para conversar com o chatbot deve ser maior para que seja mais fácil visualizar ao entrar no link.
- design mais atrativo
- layout
- Pra mim tá ótimo assim.
- Deixar mais atrativo a interface e a utilização mais simples
- As respostas serem mais objetivas
- Melhora o visual, o tamanho da letra e a velocidade.

Pergunta 12:

Se você encontrou algum problema, descreva ele

- Não encontrei nenhum, no mais ficou excelente, já tive que desenvolver um chatbot no meu TCC do ensino médio e sei que é bem chatinho, então parabéns!
- Leve demora na busca dos livros porém a resposta é satisfatória.
- Não encontrei nenhum problema.
- O chatbot não respondeu com a pesquisa inicial
- Muito lento e feio