UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Daniel Gonçalves Costa

Identificação Macroscópica de Minerais Por Meio de *Chatbots* e Processamento de Línguas Naturais

Monte Carmelo - MG, Brasil

Dezembro/2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Daniel Gonçalves Costa

Identificação Macroscópica de Minerais Por Meio de *Chatbots* e Processamento de Línguas Naturais

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Professor Dr. João Batista Simão

Coorientador: Professora Dr[a]. Marília Inês Mendes Barbosa

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Monte Carmelo - MG, Brasil

Dezembro/2018

Agradecimentos

Em primeiro lugar Deus, pois olhando para trás é perceptível que minha vida tem sido guiada pelas mãos do todo poderoso.

Aos meus familiares que incentivaram a todo momento, principalmente a meus pais, Sr^{a}. Marlene Gonçalves e Sr. José Alírio Costa, pelo apoio incondicional não medindo esforços para agigantar as oportunidades na minha vida, contribuindo não apenas para estruturação de uma boa formação profissional, mas também de um bom caráter.

Ao meu orientador professor Dr. João Batista Simão, pela oportunidade, apoio, correções e incentivo no desenvolvimento deste trabalho, se mostrando além de mentor um grande amigo.

À professora Dra. Marília I. M. Barbosa, alguém admirável que além de contribuir de maneira imprescindível na elaboração deste trabalho, se tornou uma referência pessoal de profissionalismo e simpatia.

A todos os professores, que compartilharam conhecimento não apenas teórico e prático na esfera do curso, mas também instruções para a vida como um todo, algo que possibilitou o meu desenvolvimento pessoal e a busca constante por sabedoria.

A amigos como Daniel Martins, Walter Korogi, Luíz Otávio, Tânia de Assis, José Moura, Amanda Souza e muitos outros que foram pontos de apoio durante a minha graduação desempenhando um papel essencial no meu processo de formação acadêmica. Não me esquecendo também dos atendentes da biblioteca, técnicos administrativos, funcionários da coordenação, funcionários da limpeza, funcionários da lanchonete e todas as pessoas que direta ou indiretamente contribuíram para minha caminhada, a estes sou imensamente grato.

Por fim, agradeço à Universidade Federal de Uberlândia, pela oportunidade de fazer o curso.

"Be more concerned with your character than your reputation, because your character is what you really are, while your reputation is merely what others think you are."

John R. Wooden

Resumo

O trabalho tem como objetivo o desenvolvimento de uma ferramenta intuitiva, rápida e precisa que possibilite a identificação de determinado mineral por meio das suas características. Para isso o método empregado foi a criação de um *chatbot* que utilizando processamento de línguas naturais reconhece os objetivos do usuário e o guia por um fluxo conversacional para recolher informações sobre o mineral, visando informar ao usuário qual dentre os diversos minerais possui maiores probabilidades de ser o mineral buscado. Para eficácia e rapidez na realização das funções pretendidas, o *chatbot* conta com integrações a agentes externos tais como: os serviços cognitivos providos pela Microsoft e um *Web Service* que ligado a um base de dados de minerais, faz o processamento dos dados do usuário e retorna ao *chatbot*. A utilização da ferramenta poderá ser realizada por meio dos canais de comunicação *Facebook Messenger*, *Skype* e por uma aplicação Web criada para aprovisionar além dos recursos do *chatbot*, informações sobre o mesmo. Resultados preliminares mostraram que havendo correção nas informações prestadas pelo usuário ao *chatbot*, a margem de acerto na identificação mineral é próxima a 94% para uma base de dados de 100 minerais. Todavia, o método se mostrou promissor não apenas pelo resultado da identificação, mas por ter atingido o propósito de ser uma aplicação acessível, pois pode ser utilizado através de Redes Sociais, intuitivo, pois utiliza da linguagem comum para obtenção de informações e por fornecer respostas rápidas e precisas, porque integra com serviços externos para obtenção de informações.

Palavras-chave: *chatbots*, identificação mineral, processamento de línguas naturais, atendente virtual, *machine learning*.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IA	Inteligência Artificial
PLN	Processamento de Línguas Naturais
LUIS	*Language Understanding Intelligence Service*
NLU	*Natural language understanding*
IDE	Ambiente de Desenvolvimento Integrado
JSON	*Java Script Object Notation*
SGBD	Sistema Gerenciador de Banco de Dados
BD	Banco de Dados
ORM	*Object Relational Management*
HTTP	*Hypertext Transfer Protocol*
REST	*Representational State Transfer*
API	*Application Programming Interface*
DDD	*Design* Controlado por Domínio
MVC	*Model View Controller*
UML	*Unified Modeling Language*

Sumário

1 Introdução

Ao analisar a evolução tecnológica é possível distinguir na história momentos em que processos artesanais nos mais diversos segmentos foram substituídos por máquinas que realizam as atividades repetitivas com velocidade e precisão. Essa busca por facilitação na criação de produtos e serviços é constantemente aperfeiçoada juntamente com a evolução tecnológica tornando-se relevante a pergunta: As máquinas poderão pensar como seres humanos? Semelhante pergunta foi feita em 1950 pelo matemático e cientista da computação britânico Alan Mathison Turing no seu artigo *"Computing Machinery and Intelligence"* Turing (1950). Nesse artigo o autor introduz um teste que analisa se a máquina possui capacidade de agir exatamente como um ser humano, de modo que ao entrar em contato com ela não seja possível diferenciar se está falando com um ser humano ou com uma máquina.

Após essa publicação muitas discussões foram levantadas acerca da análise e questionamentos de Turing, incluindo questões filosóficas como são tratados nos artigos *"Artificial intelligence and personal identity"* Cole (1991) e *"The Turing test as a novel form of hermeneutics"* Clark (1992). Quase sete décadas depois da publicação de Turing é possível contemplar uma realidade onde máquinas "pensam" pelos seres humanos, seja no processamento e classificação de densos dados, direcionamento de propagandas, organização de processos administrativos, entre outros. resolvendo assim problemas que demandariam enormes esforços se resolvidos sem o uso de inteligência computacional.

Um problema recorrente que pode ser resolvido com o uso da inteligência das máquinas é a identificação mineral pelas suas características de maneira inequívoca e rápida, dada a quantidade significativa e complexa de minerais existentes. Ao longo dos anos tem se desenvolvido catálogos que servem como crivo de comparação para classificá-los quanto às suas características, que podem ser clivagem, cor do traço, dureza, densidade relativa, tipo de brilho, entre outras.

Apesar de muito utilizado, esse método possui a debilidade no tempo, isso porque é preciso buscar dentre os minerais presentes no catálogo qual possui atributos semelhantes ao que se precisa saber. Assim, a busca por um método de maior eficácia e acessibilidade norteou esse trabalho que tem por objetivo o desenvolvimento de uma ferramenta que seja capaz de identificar minerais por meio das suas características. Para isso foi desenvolvido um *chatbot* que utiliza processamento de línguas naturais e filtros em tabela de decisão.

Chatbots, também conhecidos como *bots*, agentes conversacionais, interfaces conversacionais, entre outros; são programas de computador que podem ser integrados às páginas *Web* ou canais de comunicação de Redes Sociais. Os *chatbots* por aprendizado e

treinamento conseguem estabelecer um diálogo com usuário simulando um ser humano, mas tendo como diferencial a integração com serviços *Web*, o que possibilita a eles retornarem diferentes tipos de conteúdos de maneira rápida e precisa. Desta forma é possível automatizar tarefas repetitivas e burocráticas utilizando diálogos pré-definidos entre usuário e "robô".

Em seu livro *Designing Bots: Creating Conversational Experiences*, Shevat (2017) começa sua abordagem apresentando os *chatbots* como sendo o "futuro dos softwares", subsequentemente ele fornece a marcante afirmação que os bots revolucionarão a indústria de software de igual maneira que as aplicações *Web* e mobile o fizeram pois a história nos ensinou que grandes oportunidades surgem nestas revoluções, exemplos como as empresas de sucesso como *Uber*, *Airbnb* e *Salesforce* foram criados como resultado de novas tecnologias, experiência do usuário e canais de distribuição.

Já Chandel et al. (2018), em seu artigo *Chatbot: Efficient and Utility-Based Platform*, mostra que *chatbots* têm se tornado uma tendência popular. Assim, muitas organizações tem os adotado como mecanismo de comunicação com clientes e parceiros. Um fator que tem contribuído para o crescimento da utilização dos *chatbots* é a abertura de plataformas com canais de comunicação como: *Messeger*, *Skype*, *Slack*, *Telegram*, entre outros; para integração.

Um exemplo de utilização de *chatbot* é a Sociedade de Previdência Privada GEBSAPrev[1] que utiliza o *chatbot* para tirar dúvidas relacionadas à previdência e planos previdenciários. A Figura 1 mostra uma conversa com o *chatbot* através do *Facebook Messenger*.

Além da popularização deste tipo de solução e do prenúncio como sendo a revolução na indústria de softwares, alguns outros fatores contribuíram para escolha dos *chatbots* como método de resolução do problema proposto ao invés de criação de um aplicativo *mobile* ou uma página *Web* comum, como por exemplo:

- **O uso de interface conversacional:** Interface conversacional consiste na possibilidade de interação com o programa somente por meio de palavras. Como a conversa é algo intrínseco ao ser humano, o seu uso torna a utilização da solução uma experiência completamente intuitiva. Para isso, além do entendimento das expressões do usuário, faz-se necessário a construção de respostas naturais visando respondê-lo como um outro ser humano responderia.

- **Acessibilidade em aplicativos de conversa:** Shevat (2017) aponta que o ecossistema de aplicativos móveis rapidamente ficou saturado, porém aplicativos de mensagens tem prevalecido dentre a grande quantidade que entra em desuso. Isso se dá

[1] Empresa de previdência privada do grupo General Eletrics.

Figura 1 – Exemplo de conversação com *chatbot* GEBSAPrev

porque os usuários passam a maior parte do tempo utilizando esses aplicativos para estarem sempre acessíveis. Assim, ao invés de criar um próprio aplicativo de conversa, torna-se mais vantajoso prover os serviços da solução por meio dos aplicativos que já estão estabelecidos no mercado e são amplamente utilizados.

- **Possibilidade de recursos:** Os *chatbots* apesar de responderem de maneira simples e objetiva, contam com uma vasta possibilidade de integrações, viabilizando o uso dos mais diferentes tipos informações pelo uso dos *Web Services*.

Por se tratar de uma solução amplamente escalável, que possibilita a integração com diferentes recursos, dispõe de uma experiência intuitiva e de fácil acesso por meio de Redes Sociais é pertinente sua utilização para desenvolvimento de uma resolução para o problema proposto. Porém, é necessário restringir quem seria o público alvo para utilização da aplicação, este é composto de pessoas pertencentes as áreas de estudo da geologia, pois é preciso que o usuário já esteja contextualizado com as propriedades dos minerais. Por exemplo, para medição da densidade ou dureza de determinado mineral faz-se necessário um conhecimento prévio da área geológica.

O objetivo geral do trabalho é modelar o problema de identificação mineral em uma estrutura conversacional de *chatbots* de modo que este seja um método viável para identificação. Os objetivos específicos são: fornecer uma utilização simples e intuitiva ao utilizar as redes sociais no processo de utilização do método, desenvolver um método otimizado porém com módulos independentes e harmônicos, retornar ao usuário informações que sejam corretas, concisas e rápidas e por fim, prover ao usuário do *chatbot* a

possibilidade de obtenção informações sobre os minerais.

No capítulo 2 são mostrados as referências e os embasamentos teóricos que contribuíram para o desenvolvimento deste trabalho tanto nas partes técnicas quanto nas partes relacionadas as propriedades minerais, no capítulo 3 é mostrado o desenvolvimento da ferramenta e como cada aplicação desenvolvida funciona de maneira harmônica. No capítulo 4 são mostrados os resultados referentes à aplicação, tanto quanto à otimização das aplicações quanto dos resultados referentes à identificação mineral. Por fim no capítulo 5 são ressaltados os pontos pertinentes ao desenvolvimento da ferramente a os trabalhos futuros.

2 Revisão Bibliográfica

Neste capítulo serão abordadas as referências que possuem importância no desenvolvimento deste trabalho. As seções 2.1, 2.2, 2.3, 2.4 e 2.5 dizem respeito a fundamentação teórica da parte técnica utilizada no desenvolvimento e uso das aplicações. Já as seções 2.6, 2.7 e 2.8 dizem respeito a fundamentação teórica referente aos minerais e sua utilização na aplicação.

2.1 *Chatbots*

Os *chatbots* são programas de computador que simulam uma conversa humana em um chat. De acordo com Shevat (2017) *chatbots* podem ser definidos como uma interface de usuário que permite a interação com serviços usando seus aplicativos de mensagens favoritos. Ou seja, o uso destes não demandam um aprendizado demorado do usuário por se tratar de uma conversa comum pelo chat de uma Rede Social já conhecida pelo usuário.

Existem várias tecnologias utilizadas na construção de *chatbots* tais como *FlowXO*, *PullString*, *Chatfuel*, entre outras. Essas tecnologias são abordadas e descritas por Shevat (2017). Todavia, estas ferramentas não atendem completamente as demandas necessárias para construção do *chatbot* proposto no presente trabalho, pois estas possuem limitações quanto às conexões com serviços externos e uso de processamento de línguas naturais. Deste modo, o método adotado foi desenvolver o código do *chatbot* criando uma estrutura personalizada de integrações que possibilita a resolução do problema proposto.

Para facilitação da construção e implantação do *chatbot* neste trabalho, foi utilizada uma coleção de recursos oferecidos pela Microsoft. Por exemplo, ao utilizar a IDE Visual Studio, foi possível criar a estrutura básica do *chatbot* em poucos cliques. Com o *Language Understanding Intelligence Service* (LUIS) como interpretador de línguas naturais foi possível integrar com facilidade o serviço cognitivo ao código. Ao utilizar a plataforma de *cloud* Azure foi possível alocar as aplicações de maneira fácil e rápida através da própria IDE. Além disso, a empresa provê as bibliotecas *Bot Builder* e *Bot Connector* que facilitam a construção e o controle de contexto do fluxo conversacional. A documentação oficial[1] contém todas as funcionalidades e recursos oferecidos pelas plataformas.

No entanto, a codificação de um *chatbot* não se limita apenas em recursos oferecidos pela Microsoft, pois podem ser criados em qualquer linguagem de programação e incorporados a qualquer serviço *Web*.

Shevat (2017) aborda os principais pontos envolvendo *chatbots* como a arquitetura

[1] "Acessado em 15 de agosto de 2018. Disponível em: https://docs.microsoft.com/en-us/azure".

das aplicações, o uso dos canais de comunicação, design e testes, entre outros. Sendo assim uma referência elementar para desenvolvimento deste trabalho.

2.2 Inteligência Artificial

Desde que Turing (1950) levantou o questionamento sobre as possibilidades das máquinas "pensarem", a inteligência artificial tem obtido várias definições. Em seu livro Charniak e McDermott (1985) define como sendo "o estudo das faculdades mentais através do uso de modelos computacionais". Já Kurzweil et al. (1990) apresenta como sendo "a arte de criar máquinas que executam funções que exigem inteligência quando realizado por pessoas". Ainda Rich e Knight (1991) diz ser "o estudo de como fazer computadores fazerem coisas que, no momento, as pessoas são melhores".

Fato é que torna-se uma tarefa inviável estabelecer uma definição para inteligência artificial que abranja toda esfera pesquisada sobre o tema, mas de maneira sintética, ela diz respeito à manipulação de processos para se obter em programas de computados ações inteligentes semelhantes às obtidas pelo pensamento humano. Em Seu livro Russell e Norvig (2016) reúne e discute as principais definições sobre Inteligência Artificial (IA), como funciona internamente o desenvolvimento da inteligência artificial e como seu uso pode resolver problemas. Shevat (2017) também aponta que a IA geralmente faz um bom trabalho em encontrar padrões e prever resultados com base em dados passados.

Apesar de a IA não ser utilizada de maneira direta no desenvolvimento deste trabalho, o seu uso foi de fundamental importância no desenvolvimento do processamento de línguas naturais do *chatbot* desenvolvido. O uso da IA na predição é a base pela qual o serviço cognitivo LUIS se baseia para treinamento e aprendizado para conseguir separar as frases do usuário em diferentes categorias, também conhecida por "intenções", identificando qual delas possui maior probabilidade de estar relacionada à expressão. Para isso, a IA entra no processo de aprendizado de máquina utilizando um grande conjunto de dados de conversas anteriores e frases de testes fornecidas no treinamento.

2.3 Arquitetura Limpa

A arquitetura da aplicação se baseia nos princípios de *Design* Controlado por Domínio (DDD) afim de buscar a construção de um modelo conhecido como arquitetura limpa. A implementação deste modelo foi proposto por Martin (2017) em seu livro *"Clean architecture: a craftsman's guide to software structure and design"*. Neste é apresentada uma série de boas práticas para criação de uma solução coesa e com fraco acoplamento, ou seja, as partes são independentes o que facilita testes e manutenções.

A Figura 2 ilustra o modelo da arquitetura limpa representando graficamente por

camadas circulares onde a dependência acontece de fora para dentro. Na camada de entidades são definidas as estruturas dos objetos utilizados nas regras de negócios. Na camada de casos de uso são definidas as regras de negócio contendo as ações que podem ser executadas por agentes externos. A camada de controle é responsável por fazer a comunicação entre os agentes externos e as regras de negócio da aplicação. Por fim a interface do usuário são os agentes externos que fazem uso dos recursos da aplicação. Cada camada do modelo são aprofundadas por Martin (2017).

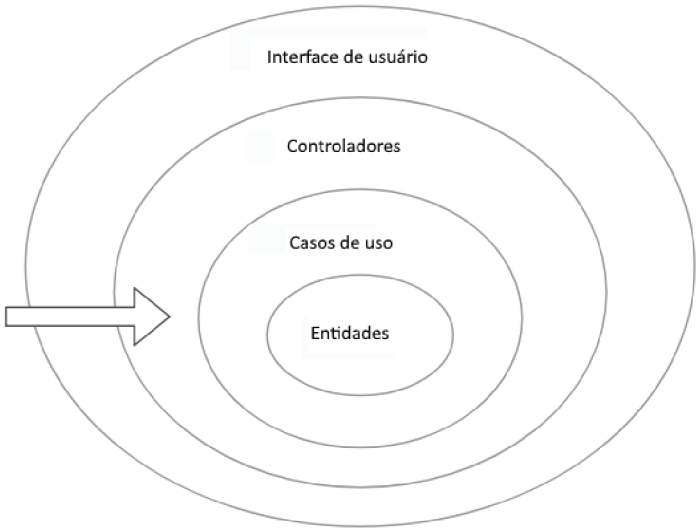

Figura 2 – Modelo da Arquitetura Limpa

Adaptado de Martin (2017)

2.4 *Entity Framework*

O *Entity Framework* é uma ferramenta desenvolvida pela Microsoft que visa facilitar o trabalho entre aplicação e banco de dados relacional. Isso se dá pelo fato de usar o conceito de mapeamento relacional de objetos (ORM) de modo que os dados e estrutura do sistema gerenciador de banco de dados (SGBD) são mapeados e manipuladas via aplicação por meio de criação de classes correspondentes às tabelas do banco de dados. Em seu livro *Programming Entity Framework: Building Data Centric Apps with the ADO. NET Entity Framework*, Lerman (2010) descreve de maneira objetiva facilidade oferecida ao se utilizar o *Entity Framework* ao apontar que os desenvolvedores gastam muito do seu precioso tempo se preocupando com seu banco de dados de *back-end*, suas tabelas e seus relacionamentos, os nomes e parâmetros de procedimentos armazenados e visualizações, bem como o esquema dos dados que eles retornam, assim o *Entity Framework* altera o jogo para que não seja preciso se preocupar com os detalhes do armazenamento de dados enquanto se escreve os aplicativos.

Lerman (2010) também explora de maneira detalhada o uso do *Entity*, assim com os aspectos técnicos envolvendo a utilização do *ADO.NET provider* que é utilizado para interpretar os comandos realizados na aplicação traduzindo-os para a linguagem de consulta estruturada (SQL), de modo que seja possível a execução no SGBD pretendido. A Figura 3 ilustra o diagrama de funcionamento do *Entity Framework* levando em consideração a troca de dados entre componentes.

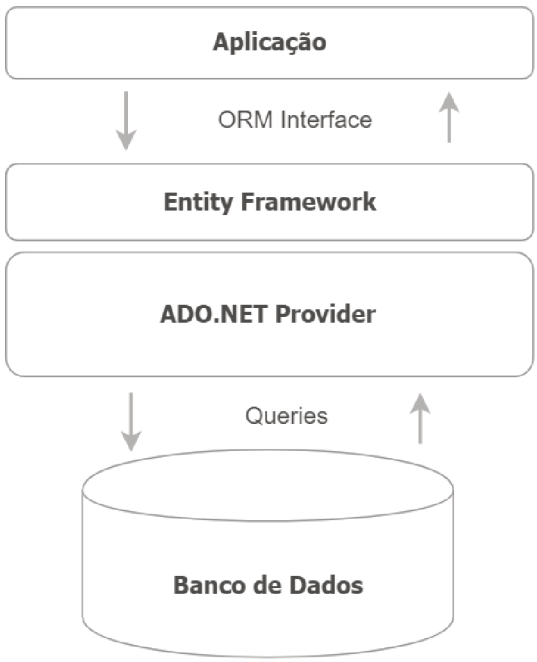

Figura 3 – Diagrama de funcionamento do *Entity Framework*
Adaptado de SANTOS (2013)

Inicialmente o *Entity* fornecia apenas a possibilidade de manipulação dos dados e estruturas dos bancos de dados (BD's) através do uso de comandos correspondentes aos comandos SQL. Todavia a ferramenta inovou incluindo a possibilidade de criação e atualização automática de estruturas por meio dos conceitos de *Code First*, *Database First* e *Model First*.

Code First, como o nome sugere, é quando o modelo de dados e a estrutura do BD são criados a partir do código da aplicação. *Database First* é quando a estrutura do banco de dados é construída manualmente e o *Entity* gera o modelo de dados e as classes correspondentes na aplicação. Por fim, *Model First* é quando o desenvolvedor gera um diagrama do modelo dos dados, por exemplo, através de um arquivo *.EDMX* no IDE Visual Studio, e o *Entity* gera a estrutura do banco de dados juntamente com o código na aplicação.

Lerman e Miller (2011) demonstram os benefícios na utilização do *Entity Fra-*

mework Code First, quando destaca a facilidade de configuração e alterações das tabelas do BD através das classes da aplicação.

Para criação e atualização automática da estrutura do BD através da aplicação, o *Entity Framework First Code* possui uma funcionalidade conhecida com *Migration* que realiza o processo de migração dos dados e configurações incluídos nas classes à base de dados.

2.5 Árvores e Tabelas de Decisão

Segundo Szwarcfiter e Markenzon (1994), árvores em computação, de maneira geral, são estruturas de dados formadas por um conjunto de elementos denominados nós ou vértices que armazenam informações. As árvores possuem um nó raiz de maior nível hierárquico que pode estar ligado a zero ou mais nós denominados filhos e estes, por sua vez, também podem possuir ligação com outros nós filhos e assim sucessivamente. O nó que não possui nenhum filho é chamado de nó folha ou terminal.

Partindo da definição de árvores, as árvores de decisão são estruturas semelhantes às árvores, porém possuem regras utilizadas na tomada de decisões. Em seu artigo *Simplifying decision trees: A survey*, Breslow e Aha (1997) esclarece a ideia ao exemplificar o uso na classificação de um caso de testes onde dada uma consulta q para classificar, uma árvore é percorrida ao longo de um caminho da sua raiz até um nó folha, cujo rótulo de classe é atribuído a q. Cada nó interno contém um teste que determina qual de suas subárvores é percorrido por q.

Ou seja, a começar pelo nó raiz, cada nó possui regras que empregadas definem qual o próximo nó filho a ser acessado de forma que no fim do fluxo se chegue a uma conclusão representada pelos nós folhas. A Figura 4 utilizada por Breslow e Aha (1997) ilustra um exemplo de árvore de decisão simplificada que categoriza residências por preço discreto de modo que cada caminho da raiz até a folha representa uma regra para inferir a participação na classe. Assim é possível inferir que se a localização for na cidade, o bairro for bom e as condições forem excelentes, então o preço será alto.

Em seu livro *Induction of decision trees* Quinlan (1987) aponta a utilização dos algoritmos de árvore de decisão em aprendizado de máquina, de modo que por meio da geração de conhecimento sejam capazes de resolver difíceis problemas de significado prático, principalmente relacionados às classificações. Desta forma, as regras dos árvores de decisão são constantemente alteradas através do aprendizado.

Partindo do conceito de árvore de decisão, as tabelas de decisão possuem regras fixas e não possuem aprendizado de máquina e treinamento das regras dos nós. Neste trabalho as características minerais são as regras fixas utilizadas para restrição de um

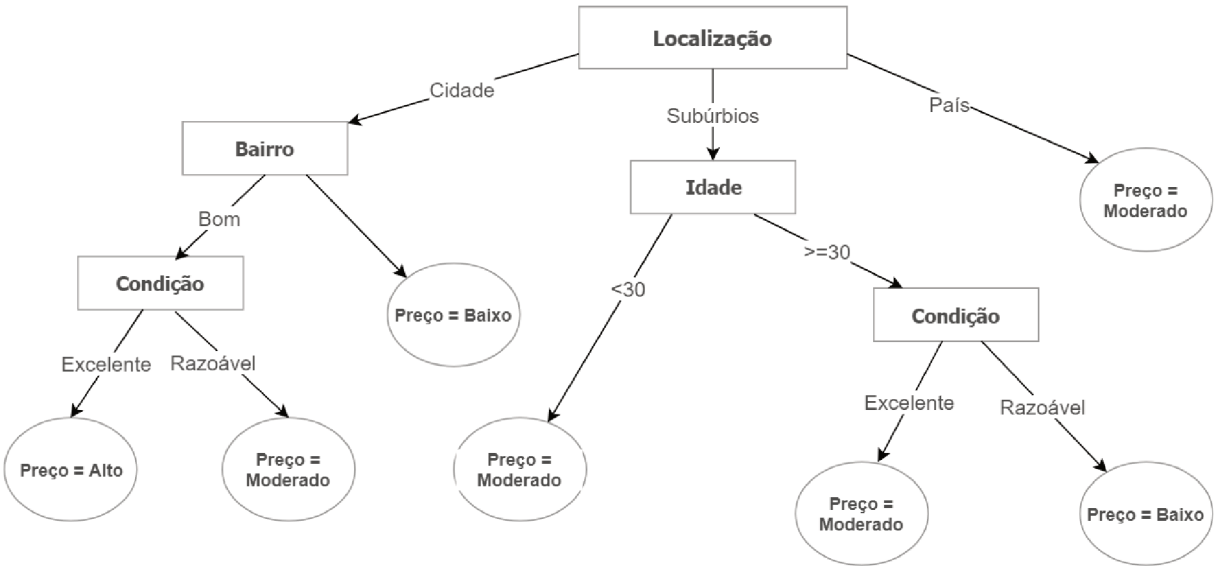

Figura 4 – Exemplo de árvore de decisão

Adaptado de Breslow e Aha (1997)

determinado mineral a um grupo de tamanho igual a um.

2.6 Minerais

Em seu livro "Manual de ciência dos minerais" Klein e Dutrow (2009) definem mineral, de maneira ampla, como sendo sólidos de ocorrência natural com um arranjo atômico altamente ordenado e uma composição química homogênea e definida. Minerais são frequentemente formados por processos inorgânicos.

O conceito "ocorrência natural" diz respeito ao processo pelo qual eles são formados que deve ser sem interferência humana, assim, assemelhados criados em laboratório são descartados pela definição. Por "sólidos" é possível excluir materiais líquidos e gasosos. Por exemplo, a água em seu estado líquido não pode ser considerada mineral, porém as geleiras formadas por processos naturais são consideradas enquanto sólidas. O mercúrio, apesar de ser excluído pela definição, alguns autores como Nickel (1995) o considera como mineral, sendo assim a única exceção. O conceito "arranjo atômico altamente ordenado", como explanado por Klein e Dutrow (2009), consiste na estrutura interna dos átomos que devem possuir um padrão geométrico regular e repetitivo. Por fim, "uma composição química homogênea e definida" determina que deve ser possível expressar a composição do mineral através de fórmula química específica, mesmo que os componentes da fórmula variem em determinado intervalo.

Quanto à composição química, Menezes (2012) categoriza os minerais em dois tipos: os que são constituídos por um dos 92 elementos químicos, como: como cobre

(Cu), ouro (Au), prata (Ag), ferro (Fe), diamante (C), entre outros, e minerais compostos por dois ou mais elementos químicos, por exemplo quartzo (SiO_2), ortoclásio ($KAlSi_3O_8$), magnetita (Fe_3O_4), galena (PbS), para citar apenas alguns. Além da composição química, os minerais possuem propriedades físicas, químicas e magnéticas específicas, as quais são exploradas na seção 2.7.

2.7 Propriedades Macroscópicas

Como abordado por Klein e Dutrow (2009) "as propriedades macroscópicas dos minerais são expressões de sua composição interna", o que torna possível tipificá-los, de modo que diferentes amostras de um mesmo tipo apresentam propriedades parecidas. Assim para identificação de determinado mineral basta analisar as propriedades físicas do mesmo, como explica Pough (1996). Quanto à identificação mineral, em seu livro: "Os Minerais: Elementos da Geodiversidade", o autor esclarece:

> [..]das mais de quatro mil espécies conhecidas, apenas alguns poucos minerais compõem a maior parte da crosta terrestre, por isso a identificação destes por suas características pode ser um importante instrumento de entendimento da geodiversidade e do meio ambiente. (LICCARDO; CHODUR, 2017, p. 59)

Alguns minerais possuem características que possibilitam sua identificação apenas pela visualização do hábito, brilho, cor, cor do traço ou outras características visuais que já restrinjam as possibilidades a apenas uma opção. Entretanto, a grande maioria depende da análise de outros tipos de propriedades para sua identificação. Klein e Dutrow (2009) exploram as propriedades físicas categorizando-as em mecânicas, relacionadas à massa, dependentes da interação com a luz e outras.

As propriedades mecânicas dizem respeito aos aspectos internos dos minerais. De acordo com (KLEIN; DUTROW, 2009, p. 54) as propriedades mecânicas "[...] refletem a intensidade das forças internas que unem os átomos individuais", ou seja, essas propriedades demonstram o comportamento do mineral ao receber aplicação de determinada força externa. Na mineralogia as propriedades mecânicas mais conhecidas são: dureza relativa, clivagem, partição, fratura e tenacidade.

Quanto as propriedades relacionadas à massa, estas também retratam aspectos internos que refletem no peso específico do mineral. Nesta categoria costumeiramente se utiliza a propriedade densidade relativa para análises.

Já as propriedades dependentes da interação com a luz representam como a luz se interage com o mineral. Este tipo de propriedade é facilmente identificada visualmente, o que facilita a restrição de opções para determinação do mineral.

Tipo	Descrição
Adamantino	Minerais em geral transparentes a translúcidos, de alto índice de refração, como diamante, zircão ou rutilo.
Resinoso	Semelhante a certas resinas, como no enxofre e na blenda.
Gorduroso ou graxo	Aspecto semelhante a óleos, como na halita, nefelina e quartzo leitoso.
Ceroso	Semelhante à cera de vela, como na calcedônia ou na opala.
Terroso	Típico em argilas, como a caulinita
Nacarado ou perláceo	Semelhante à pérola, como no talco, na gipsita e na maioria das micas.
Sedoso	Semelhante à seda, é característico em minerais fibrosos como asbestos e gipsita fibrosa.
Vítreo	Semelhante ao vidro, como quartzo, topázio, turmalina e a maioria das gemas.

Tabela 1 – Tipos de minerais não metálicos

Baseado em Liccardo e Chodur (2017).

Na subseções 2.7.1, 2.7.2, 2.7.3, 2.7.4, 2.7.5, 2.7.6, 2.7.7, 2.7.8 são explanadas as propriedades já mencionadas juntamente com outras que foram relevantes para o desenvolvimento deste trabalho, assim como a influência destas na determinação mineral. Além das propriedades aprofundadas muitas outras de caráter específico, que não foram utilizadas neste trabalho, podem ser utilizadas para desambiguação na identificação mineral. Alguns exemplos são explanados no capítulo A.

2.7.1 Brilho

O brilho diz respeito à aparência geral da superfície do mineral à medida que a luz incide sob o mesmo. Comumente o brilho é dividido em dois tipos: metálico e não metálico. De acordo com Klein e Dutrow (2009) os minerais de brilho metálico exibem superfície metálica polida, tal como a aparência do aço, cromo, cobre e ouro. Materiais com um brilho metálico refletem luz, como os metais, e são opacos a luz transmitida. Já os minerais não metálicos absorvem luz, em sua grande maioria, mas não possuem um fator comum quanto a aparência da superfície. Assim há uma subdivisão em categoria de acordo com variantes da sua apresentação. A tabela 1 mostra a classificação geral dos minerais não metálicos juntamente com a descrição da categoria.

2.7.2 Cor

(LICCARDO; CHODUR, 2017, p. 59) define a cor do mineral como sendo “o resultado da absorção ou reflexão seletiva da luz branca [...] que pode ser definida pelos comprimentos de onda do espectro luminoso não absorvidos por ele”. Ou seja, um mineral de cor vermelha é o resultado da reflexão da cor avermelhada presente na luz branca

Mineral	Principais Cores	Elemento Cromóforo
Ouro	Dourado	Ouro
Prata	Cinza	Prata
Magnetita	Castanho a preto	Ferro
Azurita	Azul	Cobre
Malaquita	Verde	Cobre
Galena	Cinza	Chumbo

Tabela 2 – Exemplos de minerais com cores idiocromáticos

Baseado em Liccardo e Chodur (2017).

Mineral	Principais Cores	Causas das Cores
Calcita	Incolor, branco, azul, amarelo, rosa	Fe, Ti, Mn
Quartzo	Incolor, violeta, amarelo, verde, castanho, rosa	Defeitos estruturais, Fe, inclusões
Berilo	Incolor, azul, verde, rosa, amarelo, vermelho	Fe, Cr, V, centros de cor
Calcedônia	Azul, laranja, verde-maçã	Fe, Ni, inclusões
Topázio	Incolor, azul, rosa, laranja, amarelo, violeta	Cr, V, Ti, Mn, defeitos estruturais, inclusões
Granada	Vermelho, rosa, verde, amarelo, laranja	Cr, Fe, Ti, Mn

Tabela 3 – Exemplos de minerais de cores alocromáticas

Baseado em Liccardo e Chodur (2017).

enquanto absorve as outras cores. É necessário ressaltar que alguns minerais absorvem e refletem quantidades diferentes de luz.

Quanto a cor, os minerais podem ser divididos em duas categorias: idiocromáticos e alocromáticos. (LICCARDO; CHODUR, 2017, p. 60) define idiocromáticos como sendo "aqueles que apresentam sempre a mesma coloração e sua cor é própria e característica". Isso se dá pela formação química específica do mineral. A tabela 2 mostra alguns exemplos de minerais idiocromáticos com suas respectivas cores e o elemento químico presente na sua composição que define a cor, também chamado de elemento cromóforo. Os minerais alocromáticos, em contrapartida, são aqueles que podem apresentar várias cores. (LICCARDO; CHODUR, 2017, p. 60) os define como sendo "aqueles cuja coloração pode variar, dependendo da presença de impurezas, elementos cromóforos ou defeitos na estrutura cristalina." A tabela 3 mostra alguns exemplos de minerais alocromáticos com suas respectivas cores e a causa das cores.

Grau de Dureza	Mineral de Escala	Objetos de Comparação	Dureza relativa
1	Talco	Risca-se com a unha	Baixa
2	Gipsita	Risca-se com a unha	Baixa
3	Calcita	Risca-se com o vidro	Média
4	Fluorita	Risca-se com o vidro	Média
5	Apatita	Risca-se com o vidro	Média
6	Feldspato	Risca o vidro	Alta
7	Quartzo	Risca o feldspato e o vidro	Alta
8	Topázio	Risca o quartzo e o vidro	Alta
9	Coríndon	Risca o topázio e o vidro	Alta
10	Diamante	Risca todos os minerais	Alta

Tabela 4 – Escala de dureza de Mohs

Baseado em Menezes (2012)

2.7.3 Cor do Traço

A cor do traço é obtida ao pulverizar uma porção do mineral de modo que se obtenha um pó da amostra. Como aborda Menezes (2012), o pó obtido não corresponde necessariamente à cor do mineral, mas representa a sua cor real.

A cor do pó deixado pelo mineral ao riscar uma placa de porcelana, de dureza elevada, corresponde à cor do traço, e deve ser observada em superfície fresca do exemplar. A propriedade cor do traço será discutida na subseção 2.7.4.

2.7.4 Dureza

De acordo com (KLEIN; DUTROW, 2009, p. 56) "dureza é a resistência que uma superfície lisa de um mineral oferece ao ser riscada". O padrão utilizado para classificação da dureza é a Escala de Dureza Relativa de Mohs proposta por Tabor (1954) em seu artigo *Mohs's hardness scale-a physical interpretation*. A Escala de Mohs consiste em dez minerais dispostos de modo crescente de dureza de maneira que cada mineral consegue riscar seus predecessores, que possuem dureza inferior, e será riscado pelo mineral subsequente de dureza superior. A tabela 4 apresenta o modelo de comparação de durezas sugerido por Tabor (1954) juntamente com objetos utilizados para comparação de dureza, que geralmente são unha, vidro e lâmina de aço de uma canivete.

Grau de perfeição	Direções	Exemplos
Muito perfeita ou excelente	uma	biotita
Perfeita	três (romboédrica)	calcita
Perfeita	três (cúbica)	galena
Perfeita	quatro (octaédrica)	fluorita
Boa	duas (1200)	anfibólio
Boa	duas (900)	piroxênio
Boa	seis (dodecaédrica)	esfalerita
Imperfeita	uma (basal)	apatita

Tabela 5 – Descrição da clivagem em minerais

Baseado em Liccardo e Chodur (2017)

2.7.5 Densidade Relativa

A densidade de um mineral é expressa por meio da densidade relativa. Klein e Dutrow (2009) definem densidade relativa como sendo um número que expressa a razão entre a massa específica (razão entre massa e volume) de uma substância e o peso de um volume igual de água a temperatura de máxima densidade, ou seja, 4 °C. Assim, um mineral de densidade 3 g/cm^3 possui três vezes a densidade da água (aproximadamente 1,0 g/cm^3) à 4 °C, para seu volume.

Como explica Liccardo e Chodur (2017), a densidade dos minerais está diretamente relacionada a sua composição química e sua estrutura, ou seja, átomos de maior peso atômico e estruturas mais compactas possuem densidades maiores.

Para determinação da densidade mineral comumente são utilizadas ferramentas como: o picnômetro, bateria de líquidos densos, balança hidrostática, entre outros. Todavia, o instrumento mais conhecido foi a balança de Jolly Linhorst (1953) patenteada em 1953 e ainda utilizada em laboratórios de mineralogia.

2.7.6 Clivagem

(KLEIN; DUTROW, 2009, p. 54) definem clivagem como sendo "a tendência dos minerais romperem-se ao longo de planos paralelos". Essa forma específica de rompimento se dá pelas ligações fracas atômicas fracas ou pela quantidade inferior de ligações em determinadas direções, como explana (LICCARDO; CHODUR, 2017, p. 92). Para identificação mineral é comumente utilizada a qualidade da clivagem, sendo esta definida pelo sentido, forma, formação de planos, etc. A tabela 5 mostra exemplos de clivagens de minerais com suas respectivas direções e graus de qualidade.

Mineral	Suscetibilidade magnética
Babingtonita	Fraca
Cromita	Fraca
Ilmenita	Fraca quando aquecida ou pulverizada
Platina	Fraca
Siderita	Fraca
Tantalita	Fraca
Magnetita	Forte
Maghemita	Forte
Pirrotita	Forte

Tabela 6 – Minerais magnéticos com suas respectivas forças magnéticas

Baseado em Liccardo e Chodur (2017)

2.7.7 Solubilidade em Ácidos

De Acordo com Liccardo e Chodur (2017), alguns minerais ao entrarem em contato com ácidos apresentam determinadas reações que podem ser detectadas visualmente. Isso se dá principalmente em minerais pertencentes à família dos carbonatos, como por exemplo calcita, aragonita, dolomita, magnesita, rodocrosita, malaquita, entre outros. O teste costumeiramente é feito utilizando-se ácido clorídrico (HCl) diluído a 10%.

2.7.8 Magnetismo

De acordo com (LICCARDO; CHODUR, 2017, p. 100) "o magnetismo ocorre quando não existe um balanço no arranjo estrutural dos íons de ferro, que pode ser encontrado em dois estados de oxidação, ferroso (Fe^{2+}) e férrico (Fe^{3+})". Os campos magnéticos gerados a partir disso podem ser fracos a ponto movimentarem apenas uma agulha de uma bússola ou fortes a ponto de levantar barras de aço, como explica Klein e Dutrow (2009). A tabela 6 mostra os exemplos mais conhecidos de minerais magnéticos juntamente com sua propensão magnética.

2.8 Manuais de Determinação

Para o desenvolvimento da ferramenta foi preciso se basear em dados estruturados em catálogos de identificação. Klein e Dutrow (2009) apresentam algumas tabelas que categorizam os minerais mais comuns quanto às características mais comumente discutidas na literatura. Todavia, para o desenvolvimento deste trabalho, os fluxos de filtragem de opções se basearam nas tabelas apresentadas por Navarro e Zanardo (2016). A opção

por trabalhar com a classificação de Navarro e Zanardo (2016) se deu em virtude do autor utilizar uma vasta quantidade de combinações diferentes de propriedades e uma quantidade superior de exemplos.

3 Desenvolvimento

O desenvolvimento deste trabalho se deu por três aplicações, o *chatbot* construído na linguagem C# utilizando a *framework* ASP.NET, um *Web Service* que intermedeia o *chatbot* ao banco de dados também construída na linguagem C# porém sob a *framework* ASP.NET Core e por fim uma aplicação *frontend* construída nessas mesmas tecnologias que tem por propósito servir de canal de acesso às funcionalidades do *chatbot*. A Figura 5 ilustra a arquitetura do conjunto de aplicações em um nível macro.

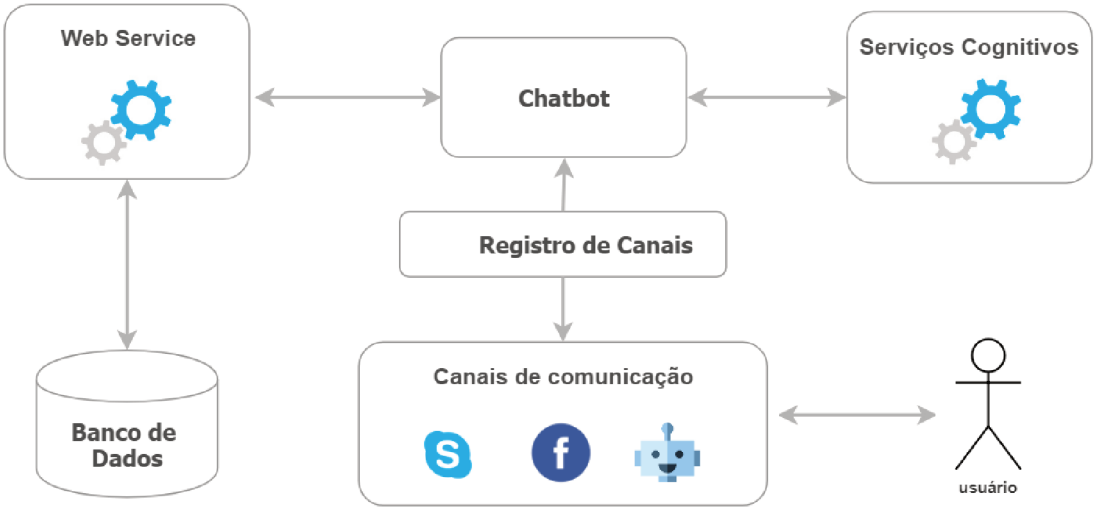

Figura 5 – Macro arquitetura do conjunto de aplicações

A interligação das aplicações e recursos se realiza da seguinte maneira: O *chatbot* consome o *Web Service* que provê métodos de consulta das informações minerais e este por sua vez se comunica com BD que armazena os dados dos minerais de maneira estruturada. Além disso, o *chatbot* consome as funcionalidades oferecidas pelos serviços cognitivos. Após publicado é possível configurar e estabelecer comunicação com os canais de comunicação. A alocação das aplicações criadas para este trabalho foi feita na plataforma de *cloud* Azure onde foi possível criar também um recurso de configuração dos canais de comunicação com o *chatbot*.

A Figura 6 ilustra o diagrama de casos de uso da Linguagem de Modelagem Unificada (UML). Este possui o propósito mostrar os casos de uso do sistema e suas interligações.

Figura 6 – Diagrama de casos de uso na aplicação

3.1 *Chatbot*

A estrutura do *chatbot* foi criada tendo por base as diretrizes de um serviço *Web* comum, mas utilizando-se como diferencial um *plugin* (*Bot Application* [1]) oferecido pela Microsoft que permite o estabelecimento de uma estrutura específica para *chatbots*. Além disso, foram oferecidas bibliotecas com métodos que facilitam o desenvolvimento. A estrutura inicial possui imprescindivelmente duas classes, uma responsável por controlar mensagens ou demais eventos recebidos pela aplicação e outra responsável por modelar os diálogos entre canal e aplicação.

Nestas duas classes toda comunicação nas chamadas de métodos e serviços acontecem de forma assíncrona. Isso otimiza as respostas gerais da aplicação, pois ela não fica aguardando uma chamada específica, mas executa as demais demandas que não dependem do retorno da chamada bloqueante.

[1] Acessado em 20 de agosto de 2018. Disponível em: https://docs.microsoft.com/en-us/azure/bot-aplication".

Na subseções 3.1.1, 3.1.2, 3.1.3, 3.1.4 e 3.1.5 serão apresentados os conceitos envolvidos no desenvolvimento da aplicação juntamente com a criação dos recursos externos necessários à construção do *chatbot*.

3.1.1 *Language Understanding*

O LUIS é um serviço cognitivo criado pela Microsoft que por meio de processamento de línguas naturais e aprendizado de máquina consegue extrair sentido de textos por meio de análises propailísticas de treinamentos realizados pelo usuário. Assim é possível identificar a intenção do usuário nas expressões inseridas e conduzi-lo à informação ou ação correta de maneira fácil e rápida como aponta Williams et al. (2015).

Para usar os recursos do LUIS é preciso criar uma conta na plataforma e criar a estrutura de intenções e expressões necessárias ao projeto. De modo que para cada intenção criada é necessário criar frases de treinamento relacionadas a esta para que por meio de cálculos probabilísticos o sistema entenda que frases semelhantes, porém variantes pertencem a intenção correta. Esta estrutura é conhecida por modelo e neste trabalho o modelo criado e treinado tem por nome "minerais" e este precisa ser treinado sempre que é adicionada alguma expressão, intenção ou entidade para recálculos de probabilidades. A Figura 7 ilustra as intenções criadas nos modelos deste trabalho na plataforma LUIS e a Figura 8 representa o treinamento da intenção de cumprimento (*greeting*) por meio de frases de exemplo para associações com outras futuras frases inseridas por usuários.

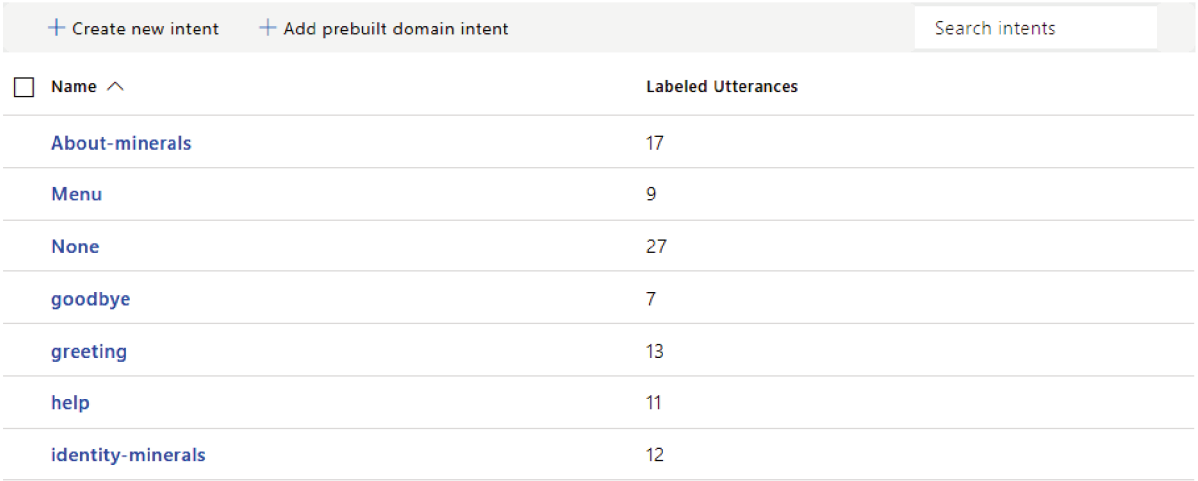

Figura 7 – Lista de intenções

A plataforma também oferece meios para extração de entidades das frases utilizadas pelos usuários, possibilitando tratar em código de forma específica palavras chaves introduzidas. Por exemplo, para o usuário saber informações específicas sobre determinado mineral é preciso que, na frase por ele digitada, o LUIS identifique o nome do mineral

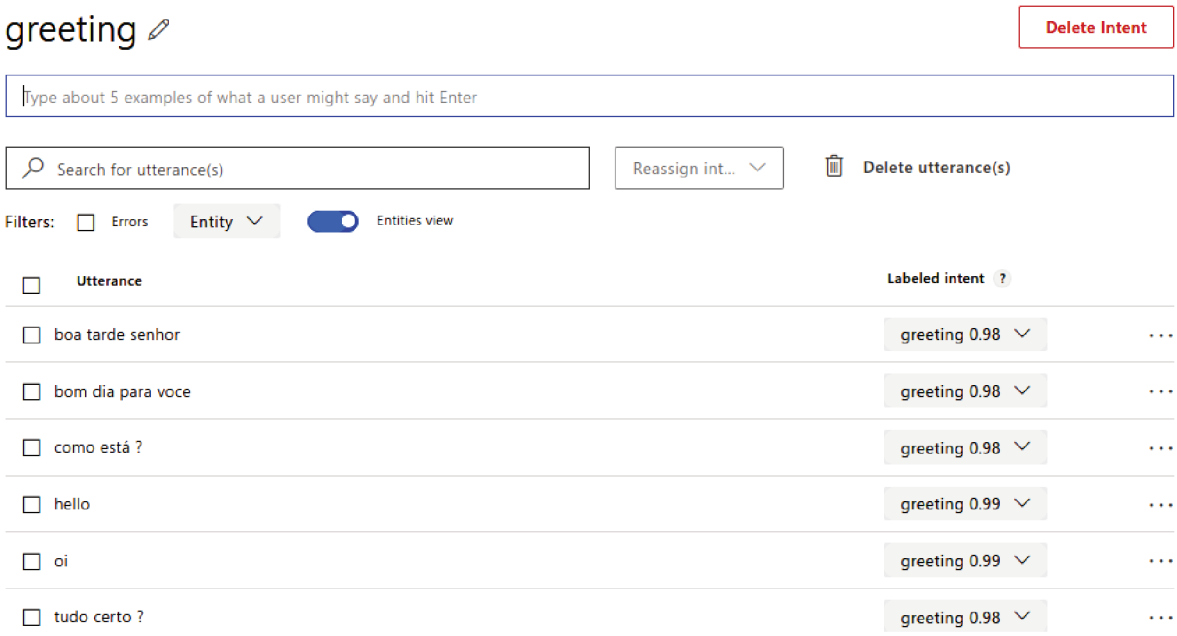

Figura 8 – Treinamento da intenção *greeting*

e salve como uma informação diferente do restante da frase, sendo possível buscar pelo nome na base de dados. Para uso do recurso ao adicionar expressões em alguma intenção de treinamento é preciso clicar nas palavras que representem entidades e sinalizá-las como tal, assim o LUIS a armazena como sendo uma informação diferenciada e ao usuário digitar alguma frase que as contenha a entidade é possível a identificação e extração desta. A Figura 9 ilustra a criação de entidades no LUIS e a Figura 10 mostra como é possível adicionar e correlacionar palavras específicas à entidades de modo que o treinamento seja facilitado.

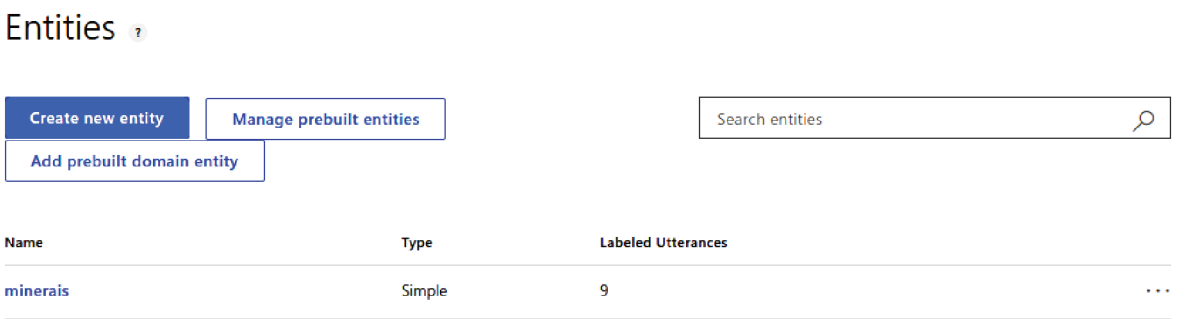

Figura 9 – Lista de entidades

O LUIS comunica com outras aplicações utilizado o padrão de comunicações JSON, a Figura 11 ilustra o retorno do LUIS quando o usuário requisita as informações de determinado mineral. O campo *topScoringIntent* mostra qual intenção tem maior probabilidade

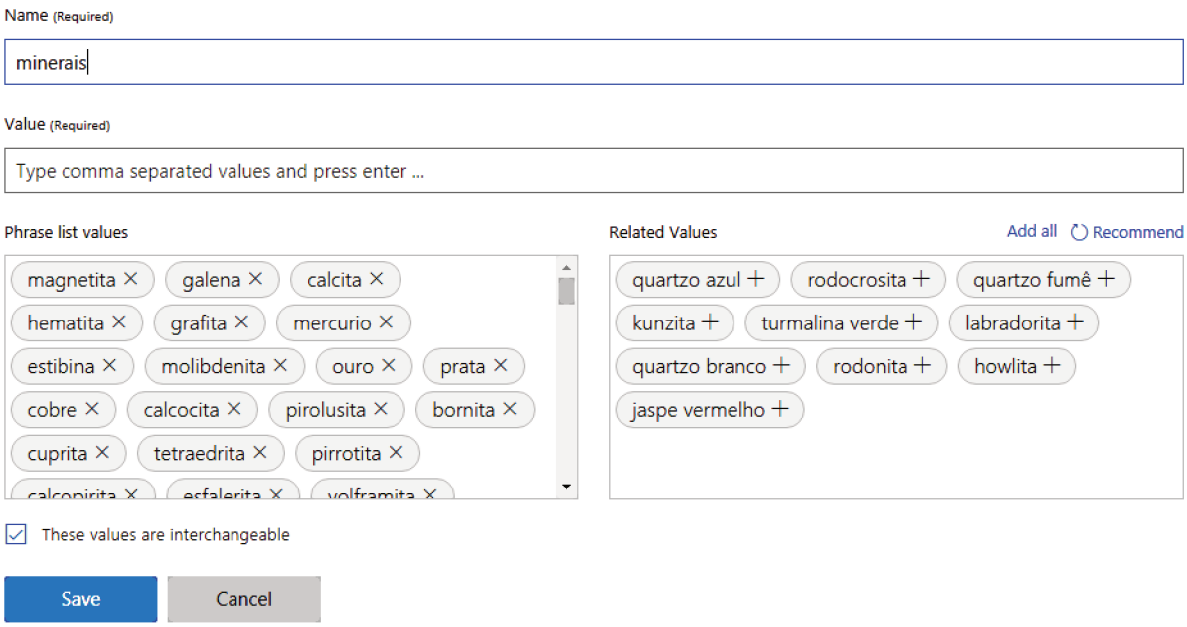

Figura 10 – Lista de palavras correlacionada à entidade

de estar relacionada à frase do usuário, pois por se tratar de um modelo probabilístico o resultado nem sempre será perfeito e único. Nesse caso o score para essa intenção foi de aproximadamente 99% e logo após, são mostradas outras intenções que possuem probabilidades inferiores de serem o objetivo do usuário. Por fim, há uma lista de entidades extraídas da frase do usuário que neste caso é apenas a entidade "galena" do tipo "mineral", mas caso o usuário queira saber informações de mais de um mineral o LUIS traria as duas entidades do tipo "mineral".

3.1.2 Controlador de Atividades

Apesar da classe possuir o nome referente ao controle de mensagens (*MessagesController*) ela vai além disso, controlando uma série de outros eventos emitidos pelo canal à aplicação tais como informações do canal, informações do usuário, tipo de mensagem recebida, formato da mensagem, mensagem propriamente dita, entre outros. A título de fácil entendimento, o conjunto de informações enviadas do usuário ao diálogo será nomeado por atividades.

O papel do controlador é mediar as atividades emitidas na interação entre canal e *chatbot* de modo que antes de direcioná-las à classe de diálogo seja possível criar filtros e definir configurações próprias na integração entre *chatbot* e canal. A Figura 12 apresenta o método *Post* presente na classe de controle de atividades. Este recebe como parâmetro uma atividade, verifica o seu tipo e identificando como do tipo "mensagem", tenta instanciar a classe de diálogo emitindo a atividade e aguarda a resposta.

```
{
  "query": "me informar sobre galena",
  "topScoringIntent": {
    "intent": "About-minerals",
    "score": 0.9887482
  },
  "intents": [
    {
      "intent": "About-minerals",
      "score": 0.9887482
    },
    {
      "intent": "None",
      "score": 0.0112518
    }
  ],
  "entities": [
    {
      "entity": "galena",
      "type": "minerais",
      "startIndex": 18,
      "endIndex": 23,
      "score": 0.956781447
    }
  ]
}
```


Figura 11 – Retorno do LUIS em formato JSON.

```
public async Task<HttpResponseMessage> Post([FromBody] Activity activity)
{
    if (activity.Type == ActivityTypes.Message)
    {
        await Conversation.SendAsync(activity, () => new Dialogo());
    }
    else
    {
        await HandleSystemMessageAsync(activity);
    }
    return Request.CreateResponse(HttpStatusCode.OK);
}
```

Figura 12 – Método para controle de atividades

Após ser instanciada, a classe de diálogo irá tratar a mensagem de modo que seja retornado ao usuário uma resposta do tipo *HttpResponseMessage* porém encapsulada em uma *Task*. Por se tratar de programação assíncrona, os métodos possuem esse padrão de

retorno e suas chamadas possuem o operador *await* indicando que a aplicação pode fazer outras coisas enquanto o resultado da chamada é processado.

Caso a atividade não seja do tipo "mensagem", ela é enviada a outro método onde é possível verificar os demais tipos e tratar de maneira específica. Por exemplo, na Figura 13 a função identifica a atividade como um evento de digitação, assim, ao invés de instanciar o diálogo é criado apenas um retorno que simula a digitação e o usuário vê no seu canal de comunicação uma mensagem ou um símbolo que represente isso.

```
if (message.Type == ActivityTypes.Typing)
{
    var uri = new Uri(message.ServiceUrl);
    var connector = new ConnectorClient(uri);

    Activity isTypingReply = message.CreateReply();
    isTypingReply.Type = ActivityTypes.Typing;

    await connector.Conversations.ReplyToActivityAsync(isTypingReply);
}
```

Figura 13 – Método de tratamento para atividades atípicas

3.1.3 Diálogos

Na classe de diálogo, acima da declaração é especificado, entre colchetes, o atributo *Serializable* indicando que essa classe é passível de serialização, ou seja, ao instanciá-la é possível passar valores que serão utilizados juntamente com suas propriedades, neste caso o valor é a atividade emitida na instanciação realizada pela classe de controle de atividades. O diálogo usa a interface *IDialogContext* para encapsular a atividade recebida juntamente com as informações de contexto para execução do processo de conversação. Essa variável é usada para controlar o contexto na execução do diálogo entre *chatbot* e usuário, servindo como condutor da conversa de modo que uma mensagem não seja sobreposta a outra e assim haja coesão entre mensagens.

Outro atributo definido acima da declaração da classe é o *LuisModel* onde é adicionado o identificador do modelo pretendido juntamente com uma chave de usuário, ambos criados através da plataforma.

Para que o diálogo lide com as intenções e entidades é preciso herdar atributos e métodos da classe *LuisDialog* pertencente à biblioteca *Bot Build Dialogs* mantida pela Microsoft, possibilitando associar métodos a intenções do LUIS de modo que estes realizem procedimentos desejados pelos usuários. A Figura 14 ilustra a associação das intenções *greeting*, *goodbye* e *help* com seus métodos correspondentes através do atributo *LuisIntent*,

de modo que quando o usuário digitar algo do tipo "Gostaria de obter ajuda" o LUIS entenderá que essa expressão possivelmente pertence a intenção de ajuda e a instância do diálogo vai conduzi-lo ao método *HelpMethod* onde será possível mostrar ao usuário algum recurso de resposta, como por exemplo uma lista serviços do que o *chatbot* pode prover.

```
[LuisIntent("greeting")]
public async Task GreetingMethod(IDialogContext context, LuisResult result)...

[LuisIntent("goodbye")]
public async Task GoodbyeMethod(IDialogContext context)...

[LuisIntent("help")]
public async Task HelpMethod(IDialogContext context)...
```

Figura 14 – *Tags* associando intenção do LUIS a métodos do diálogo

3.1.4 Recursos de Reposta

Recursos de repostas em *chatbots* estão relacionados em como a informação é respondida ao usuário, podendo ser principalmente através de textos e carrosséis.

A tratativa para apresentação de textos equivale às orquestrações comumente realizadas em *strings*, sendo assim mais simples de serem construídas, todavia, apesar de possuírem simples implementação, é preciso projetá-las de modo que o usuário tenha uma experiência semelhante a conversação entre humanos. Para isso, é necessária uma abordagem baseada em conhecimentos empíricos da conversação humana visando criar frases da maneira mais natural possível. O uso das frases naturais, sucintas e informativas proporcionam uma experiência intuitiva ao usuário.

Quanto aos carrosséis, por se tratar de um conjunto de cartões com textos, imagens e botões, fez-se necessária a criação de uma classe para montagem. A criação do recurso segue o seguinte fluxo: Na classe de diálogo, o *chatbot* requisita o *Web Service*, são retornados os dados em formato JSON, o *chatbot* converte o retorno de JSON para um objeto do tipo mineral e envia para o método responsável pela construção do recurso de resposta na classe de montagem. O método renderiza as informações dos atributos nos cartões e retorna uma lista destes ao diálogo que por sua vez apresenta ao usuário.

A Figura 15 ilustra o método de construção de uma resposta do tipo carrossel que é mostrada ao usuário quando se faz necessária a apresentação de informações sobre determinado mineral. Nesse exemplo é criado apenas um *card* no qual possui uma imagem, um título que é o nome do mineral juntamente com sua densidade, um subtítulo que possui

a descrição do mineral e por fim outro texto que contém a mais notável cor e cor do traço do mineral.

Para construção do carrossel são utilizados modelos de dados pertencentes à biblioteca *Bot Connector*. O modelo *CardAction* define a estrutura dos botões adicionados nos cartões. Possui basicamente: um título, um valor e um tipo. *CardImage* define a estrutura da imagens utilizadas no cartão. Possui basicamente o endereço da imagem e de uma imagem alternativa. *HeroCard* define a estrutura do cartão como um todo. Possui um título, um subtítulo, um texto adicional, uma lista de imagens e uma lista de botões. Por fim, *Attachment* presenta um modelo genérico de recurso de resposta no qual os *HeroCards* são encapsulados antes de retornados ao *chatbot*.

```
List<Attachment> listAttachments = new List<Attachment>();
string contentUrl = await BingImageSearch.GetContentUrl("pt-BR", m.name);

HeroCard attachment = new HeroCard
{
    Images = new List<CardImage> {new CardImage{Url = contentUrl } },
    Title = m.name + " - " + m.density + " g/cm³",
    Subtitle = m.description,
    Text = " Cor: " + m.mainColor + " Traço: " + m.mainTrace
};
listAttachments.Add(attachment.ToAttachment());
return listAttachments;
```

Figura 15 – Método de criação do carrossel de informações minerais.

A busca da imagem para inclusão no carrossel se dá a partir do uso de um serviço cognitivo que será discorrido na subseção 3.1.5.

3.1.5 *Bing Search Image*

De acordo com a documentação oficial[2], o serviço cognitivo de pesquisa *Web* do *Bing* fornece uma experiência semelhante às buscas de imagens no navegador *Bing*, retornando resultados de pesquisa que o *Bing* considera mais relevantes para a consulta de um usuário. O serviço cognitivo usa o padrão de comunicação JSON para troca de dados.

Para este trabalho, o serviço cognitivo foi utilizado na busca de imagens para composição dos carrosséis de resposta, de modo que no momento em que o *chatbot* cria o recurso ele requisita o serviço e traz a imagem que corresponde à resposta do usuário, desta forma o recurso é visualmente mais agradável. A Figura 16 ilustra um carrossel quando

[2] Acessado em 15 de agosto de 2018. Disponível em: https://docs.microsoft.com/en-us/azure/cognitive-services/bing-web-search".

o usuário requisita informações sobre o mineral "galena" no canal de comunicação *Skype*, de modo que a imagem é buscada em tempo de execução e mostrada ao usuário depois do *card* ser construído.

Figura 16 – Resposta do tipo carrossel no canal *Skype*

A criação do serviço cognitivo utilizado neste trabalho foi realizada por meio da plataforma de *cloud* Azure, quando criado é gerada uma chave de acesso para realização das requisições http do tipo *GET*. A construção da requisição é ilustrada na Figura 17. O *endpoint* é constituído de uma base juntamente com as partes parametrizáveis que é o termo a ser buscado já convertido em sua representação de escape e o tamanho desejado para imagem. Para que a requisição execute com sucesso é preciso adicionar no *Header* a key *"Ocp-Apim-Subscription-Key"* juntamente com a chave de acesso obtida anteriormente, caso a chave não seja inserida, a requisição retornará o erro do tipo 401 (acesso negado).

```
var urlB ="https://api.cognitive.microsoft.com/bing/v7.0/images/search";
string uriQuery = urlB + "?q=" + Uri.EscapeDataString(searchQuery) +
    "&size=large";

using (HttpClient client = new HttpClient())
{
    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", accessKey);
    HttpResponseMessage response = await client.GetAsync(uriQuery);
    return response;
}
```

Figura 17 – Requisição ao serviço *Bing Search Image*

O retorno da requisição traz uma lista de objetos ordenados por relevância em relação ao termo pesquisado. O retorno possui muitas informações sobre o termo pesquisado, porém o único campo relevante para uso nos carrosséis é o campo "*contentUrl*" obtido de um dos três principais resultados que é sorteado de maneira pseudo-randômica para que o usuário não veja a mesma imagem todas as vezes em consultas iguais.

3.2 Web Service

Em seu artigo *"Web Service composition-current solutions and open problems"* Srivastava e Koehler (2003) define *Web Services* como sendo unidades independentes de lógica de aplicação que fornecem funcionalidade de negócios para outros aplicativos através de uma conexão com a *internet*, ou seja, são aplicações que quando requisitadas por outras, processam e retornam um *status* da operação solicitada.

Desta forma, para este trabalho o *Web Service* foi criado tendo por base a busca pela independência entre aplicações de modo que a modularização viabilize seu consumo por algum outro programa que desejar. Sua responsabilidade é realizar processamento e consultas de dados sempre que requisitado pelo *chatbot*, fornecendo os recursos necessários às demandas do usuário.

O *Web Service* está conectado ao banco de dados SQL Server onde estão armazenados os dados referentes aos minerais de modo que a troca de informações se dá através do modelo de transmissão de dados JSON empregando-se do padrão arquitetural REST.

Por seguir as diretrizes REST, a aplicação opera através do protocolo de comunicação HTTP. Desta maneira, a utilização das operações fornecidas pelo serviço podem ser obtidas mediante realização de requisições HTTP. Por exemplo, ao usuário solicitar informações sobre determinado mineral, o *chatbot* realiza uma requisição do tipo *GET* ao serviço e este conectado à base de dados retorna o JSON das informações desejadas de modo que o *client*, no caso o *chatbot*, trata o retorno e exibe ao usuário.

3.2.1 Arquitetura

Seguindo os princípios da arquitetura limpa, o *Web Service* possui a definição da estrutura das entidades *Mineral*, *Color* e *TraceColor*. A classe *MineralDomain* possui os métodos com as regras de negócio da aplicação, esta implementa a interface *IMineralDomain* onde são definidas as assinaturas dos métodos utilizados. Por fim a classe *MineralController* faz ligação entre as requisições do usuário e os métodos da classe de domínio, de modo que quando há uma requisição HTTP o controlador analisa o tipo e o endereço da requisição e direciona ao método correto.

A Figura 18 ilustra o diagrama de classes pertencente a UML e neste é possível

observar os atributos presentes na aplicação. A entidade Mineral pode possuir uma ou muitas cores, para isto foi criada a entidade de associação *MineralColor* que armazena o identificador do mineral e o identificador da sua respectiva cor. De igual maneira acontece com a cor do traço do mineral. Foi utilizado o conceito de composição pelo fato da entidade de relacionamento existir somente havendo um mineral e sua respectiva cor ou cor do traço no Bd.

A entidade *BrightnessType* representa a descrição da cor não metálica do mineral. Um mineral pode possuir mais de um tipo de brilho não metálico enquanto cada tipo de brilho pode estar associado a muitos minerais.

O diagrama apresenta também as assinaturas dos métodos da aplicação na classe de controle. A implementação de cada método será aprofundada na subseção 3.2.2.

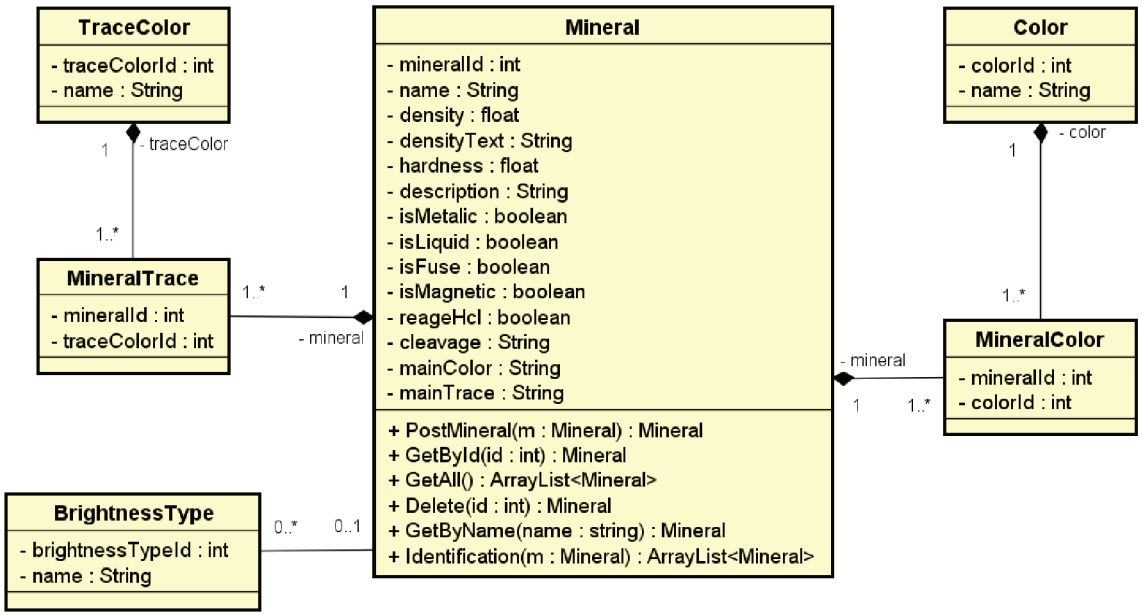

Figura 18 – Diagrama de Classes

3.2.2 Métodos

A aplicação possui na camada de controle seis métodos que quando requisitados através de requisições HTTP, acessam a camada de regra de negócios para prover as funcionalidades requisitada pelo usuário. Estes possuem dois atributos acima da declaração, *"Router"* que representa a rota que o usuário usará para requisitar o serviço e o tipo da requisição HTTP que pode ser *HttpGet*, *HttpPost*, *HttpPut*, *HttpDelete*, entre outras.

Nos tópicos 3.2.2.1, 3.2.2.2, 3.2.2.3, 3.2.2.4 e 3.2.2.5 é brevemente comentado sobre os métodos da aplicação, dando enfoque ao método responsável pela identificação mineral.

Para melhor compreenção, os métodos da camada de controle possuem mesmo nome que os requisitados por estes na camada de regra de negócio.

3.2.2.1 Métodos *GET*

As requisições do tipo *GET* são utilizadas especificamente para retorno de informações. O *Web Service* possui três métodos do tipo *GET*: *"GetAll"*, *"GetById"* e *"GetByName"*.

Em requisições para o método *"GetByName"* é recebido do usuário um nome relacionado ao mineral, a camada de controle envia o nome à classe *MineralDomain* que acessa a instância do BD, verifica a existência de algum mineral e retorna ao usuário um objeto do tipo "Mineral" encapsulado em uma *IActionResult* que representa a convenção utilizada pela *framework* para retorno de informações. Os outros dois métodos *Gets* possuem o mesmo padrão, de maneira que *"GetById"* realiza o mesmo processo de busca, entretanto utilizando um identificador ao invés do nome e *"GetAll"* retorna todos os minerais da base não filtrando por nenhum atributo.

Para este trabalho a única requisição do tipo *GET* utilizada é a *"GetByName"* de modo que quando um usuário pergunta ao *chatbot* informações sobre determinado mineral o *Web Service* retorna essas informações de acordo com o nome enviado, o *chatbot* trata o retorno e exibe ao usuário. A Figura 19 ilustra uma requisição realizada ao método *"GetByName"*, no qual é o parâmetro "magnetita" para se obter informações sobre o mineral.

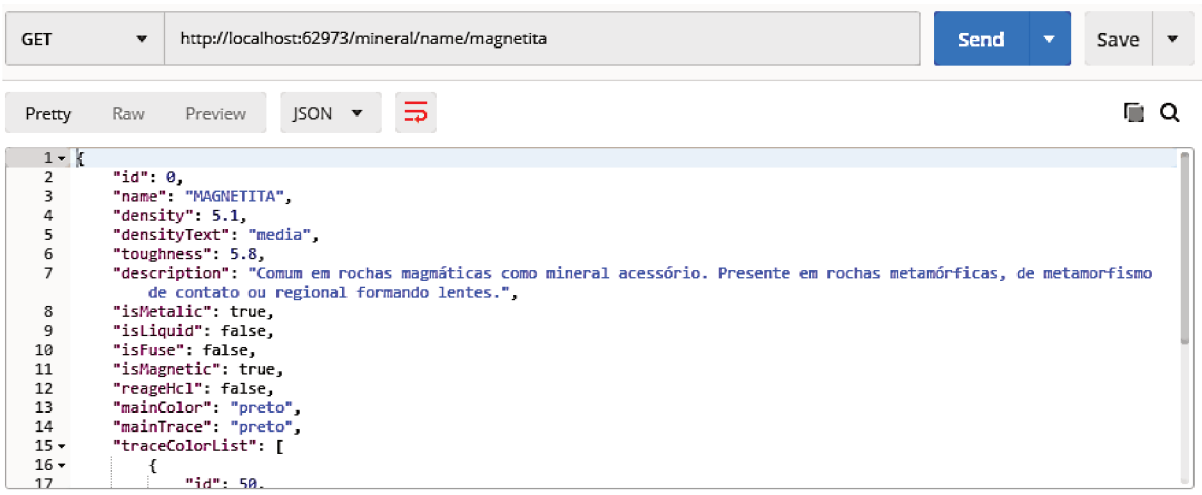

Figura 19 – Exemplo de requisição *GET* via Postman

3.2.2.2 Métodos *POST*

As requisições HTTP do tipo *POST* são utilizadas principalmente para inserção de informações em bases de dados, porém seu uso não se restringe apenas a isso. Como é possível passar informações no corpo da requisição de forma não exposta, muitas vezes é usado para outras finalidades como o login de forma segura em páginas. Na aplicação há dois métodos acessados via requisição do tipo *POST* e são eles: *PostMineral* e *Identification*. O primeiro é utilizado para inserção de minerais no BD, de modo que a camada de controle recebe um mineral, envia a camada de domínio e esta insere o mineral no BD e o retorna ao usuário o status da operação. Quanto ao método de identificação, este é abordado no tópico 3.2.2.5.

Na camada de domínio é utilizado o controle de transações provido pelo *Entity Framework* para garantia de atomicidade, de modo que o início da transação é sinalizado e caso aconteça alguma exceção, seja revertido o que foi feito ao status anterior. Esse controle é feito através dos comandos *BeginTransaction* e *Rollback*.

3.2.2.3 Método *DELETE*

As requisições HTTP do tipo *DELETE* são utilizadas para deletar informações da base de dados. O método recebe um identificador do mineral a ser deletado e passa ao método de deleção na camada de domínio que acessa à instância do BD e apaga o mineral correspondente ao identificador.

3.2.2.4 Método *Update*

As requisições HTTP do tipo *PUT* são utilizadas para atualizar informações na base de dados. Para isso o método na camada de controle recebe um objeto do tipo "Mineral" no corpo da requisição e um identificador passado através da rota que representa qual o mineral a ser atualizado. O método na camada de domínio acessa à instância do BD e modifica o mineral correspondente ao identificador com as informações recebidas no corpo da requisição.

3.2.2.5 Identificação

As requisições ao método de identificação são do tipo *POST*, isso porque fez-se necessário o envio de um corpo por parte do *chatbot*. O método recebe do usuário um mineral e um status no fluxo conversacional indicando em qual passo da identificação se encontra. Para cada passo, o método retorna uma lista de minerais que tem correspondência aos atributos do mineral recebido, de forma que o mineral recebido no corpo da requisição pode possuir todos os atributos do objeto ou apenas um.

O fluxo de identificação se divide em dois, o primeiro analisa para desambiguação as características mais gerais: estado físico, tipo de brilho, dureza, densidade, cor e cor do traço. O segundo fluxo, realizado depois do primeiro, utiliza outras propriedades mais específicas a grupos menores de minerais, estas são: reação a HCl, fusibilidade, magnetismo, clivagem e tipo de brilho (para quando não metálico). Assim, o mineral ao não ser identificado de maneira única no primeiro fluxo, passa pelo segundo a fim de restringir o resultado da identificação a quantidade mínima possível.

A Figura 20 representa o diagrama de atividades pertencente a UML que ilustra o primeiro fluxo conversacional para identificação do mineral. O fluxo se inicia com o *chatbot* perguntando sobre o estado físico do mineral, no qual há duas opções: sólido e líquido. No caso do usuário informar que o estado físico é liquido, o filtro na tabela de decisões pela propriedade resulta em apenas uma opção representada pelo mercúrio. Caso contrário, o *chatbot* pergunta a próxima questão ao usuário e assim sucessivamente até que o primeiro fluxo seja finalizado ou que seja retornado um resultado único.

No segundo fluxo, as questões feitas pelo *chatbot* dependem dos minerais filtrados pelo primeiro fluxo. A lista é iterada verificando se algum dos minerais possuem alguma das características e somente a partir disso o *chatbot* cria a pergunta ao usuário.

Apesar do *Web Service* estar preparado para a filtragem pelas características do segundo fluxo, o processamento destas é realizado em memória via aplicação no próprio *chatbot*. Esta decisão foi tomada levando em consideração que a lista filtrada não possui uma quantidade significativa de minerais que justifique o processamento destes via requisição HTTP.

3.2.3 SGBD SQL Server

Para armazenamento dos dados referentes aos minerais foi escolhido pela utilização neste trabalho do SGBD SQL Server[3] que foi criado e alocado no sistema de *cloud* Azure. O SQL Server se baseia no esquema relacional e possui robustez e uma vasta quantidade de funcionalidades.

A estrutura do banco de dados possui primariamente três tabelas: *Minerals*, *Color*, *TraceColor* e *BrightnessType* ao passo que cada mineral pode possuir zero ou um tipo de brilho não metálico e está relacionado a várias cores e traços. O relacionamento se dá a partir de duas tabelas: *MineralColor* e *MineralTrace* e essas possuem os identificadores referentes à tabela mineral e às tabelas a qual se relaciona. A Figura 21 ilustra o diagrama entidade relacionamento da aplicação.

Para conexão do BD com a aplicação, a ferramenta *Entity Framework* fornece o necessário para mapeamento das tabelas no código. Para isso, é preciso criar uma

[3] Banco de dados relacional desenvolvido pela Microsoft.

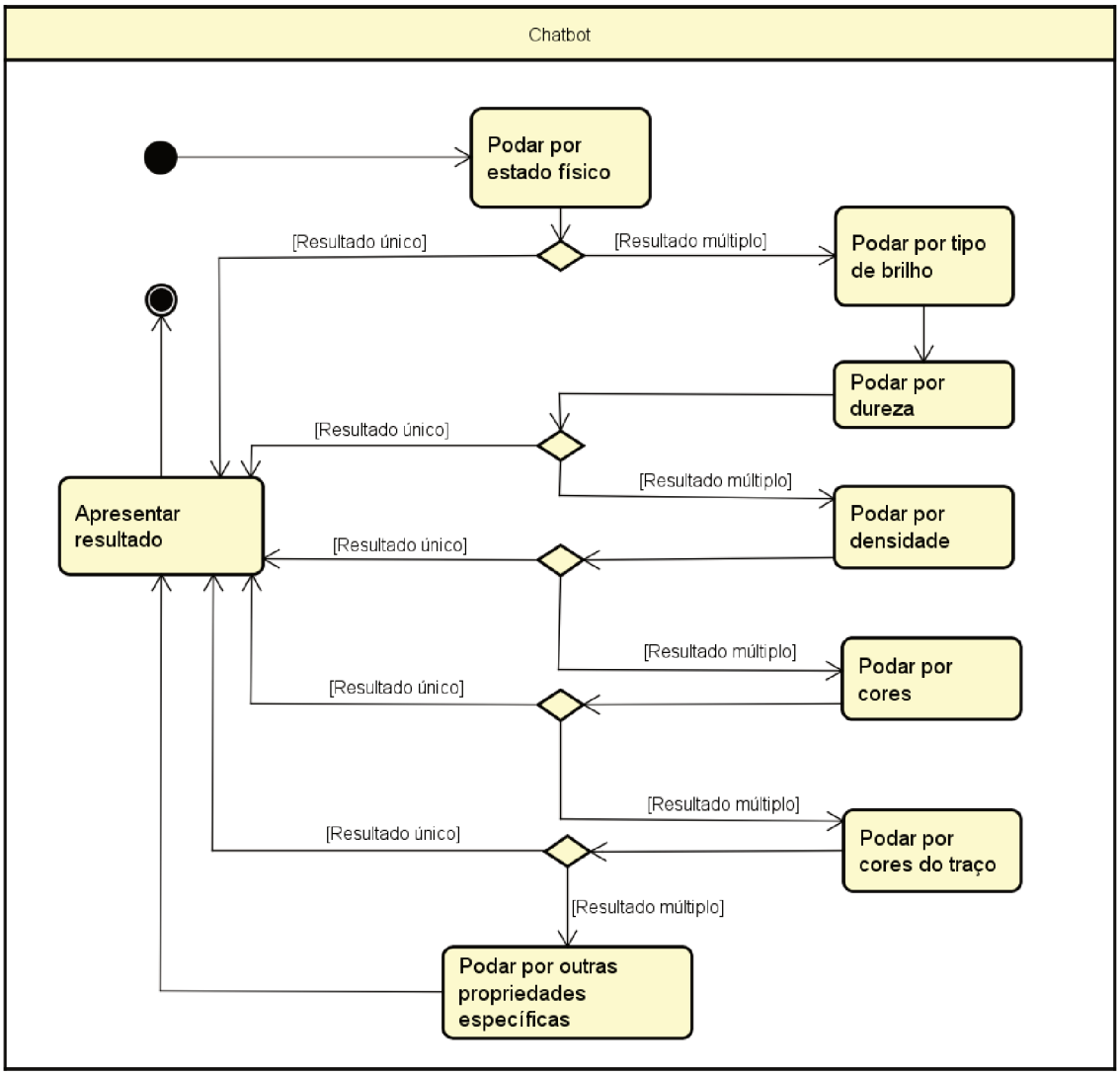

Figura 20 – Diagrama de Atividades da identificação mineral

classe que herde da classe *DbContext*, especificar na classe as tabelas a qual se quer manipular e declarar no arquivo de configurações o uso da classe como contexto de BD juntamente com a *string* de conexão do BD remoto. Assim é possível criar uma instância da classe e utilizar funcionalidade como controle de transação e inserção de comandos SQL diretamente no código C#. Além disso, a configuração da classe *DbContext* permite o uso de outra funcionalidade do *Entity* que atualiza automaticamente as tabelas do BD por meio das Entidades da aplicação. A utilização deste recurso será discutida na subseção 3.2.4.

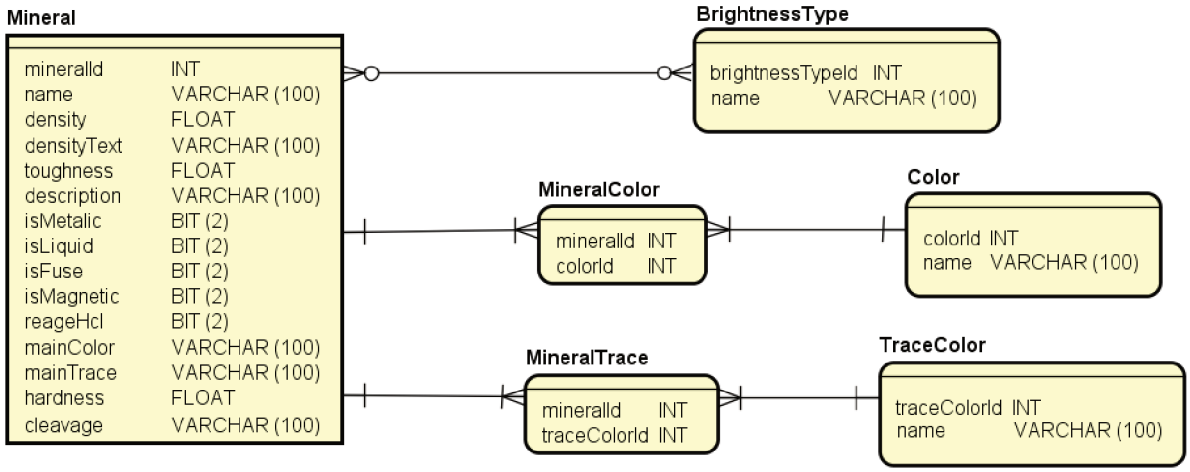

Figura 21 – Diagrama Entidade Relacionamento

3.2.4 *Migration*

A criação da estrutura do BD foi feita automaticamente a partir do uso da funcionalidade *Migration* pertencente ao *Entity Framework*. As classes criadas no *Web Service* foram convertidas em tabelas do BD mantendo as configurações estabelecidas no desenvolvimento da classe. A Figura 22 ilustra os atributos da classe *"Color"* utilizados para para criação e atualização da tabela correspondente no BD. O atributo *"Table"* representa o nome da tabela no BD, o atributo *"Column"* representa o nome da coluna relacionada à propriedade e por fim o atributo *"Key"* sinaliza a propriedade como sendo chave primária na tabela a ser criada.

```
[Table("Color")]
public class Color
{
    [Key]
    [Column("id")]
    public int id { get; set; }
    [Column("name")]
    public string name { get; set; }
}
```

Figura 22 – Classe Color pertencente ao Web Service

Para execução do *Migration* na aplicação foi preciso configurar as classes serem convertidas em tabelas, adicionando atributos de sinalização para o *Entity Framework*. Usar a biblioteca do *Entity* para criar e configurar a classe *DbContext* e adicioná-la às

configurações da aplicação juntamente com a *string* de conexão do BD. Após isto foi possível adicionar a migração através do comando *dotnet ef migrations add "NomeDaMigracao"* e com isso são gerados arquivos contendo todos os atributos juntamente com o relacionamento entre as tabelas, de modo que é possível alterar algo nesses arquivos caso necessário.

Com esse primeiro comando é gerada a estrutura, mas esta não é refletida no BD e para isso é preciso executar o comando *dotnet ef database update* que gera as tabelas no BD de acordo com as entidades da aplicação obedecendo as regras utilizadas da criação das entidades. A documentação oficial[4] apresenta também outros comandos relacionados a utilização da ferramenta.

3.3 Registro de Aplicativo

Para evitar que agentes externos desconhecidos façam uso dos recursos do *chatbot* de maneira indevida foi preciso criar configurações de autenticação. Para isso, utilizou-se uma plataforma de identidade que a Microsoft oferece, com a qual foi possível realizar registros de aplicativos. De acordo com a documentação oficial[5], a plataforma de identidade permite que os desenvolvedores criem aplicativos que assinam todas as identidades da Microsoft, obtêm *tokens* para chamar o Microsoft Graph, outras APIs da Microsoft ou APIs que os desenvolvedores criaram. É uma plataforma completa que consiste em um serviço de autenticação, bibliotecas de código aberto, registro e configuração de aplicativos, documentação completa do desenvolvedor, exemplos de código e outros conteúdos para desenvolvedores.

A Figura 23 ilustra o cadastro do *chatbot* na plataforma de identidade. Após criação do registro é gerado um par de identificador e chave os quais são inseridos no arquivo de configuração do *chatbot* antes de publicá-lo. Assim somente canais que tenham conhecimento das informações de acesso conseguirão se conectar ao *chatbot*. A configuração dos canais de comunicação serão abordada na seção 3.4.

3.4 Registro de Canais

Para utilização dos recursos providos pelo *chatbot* é necessário associá-lo aos canais de comunicação. A plataforma de *cloud* Azure fornece a possibilidade de criação de uma solução de mapeamento de canais. Nas configurações da solução foi necessário adicionar o endereço de publicação do *chatbot* juntamente com os dados de registro fornecidos do

[4] Acessado em 25 de agosto de 2018. Disponível em "https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations".

[5] Acessado em 12 de novembro de 2018. Disponível em "https://docs.microsoft.com/en-us/azure/active-directory/develop/about-microsoft-identity-platform".

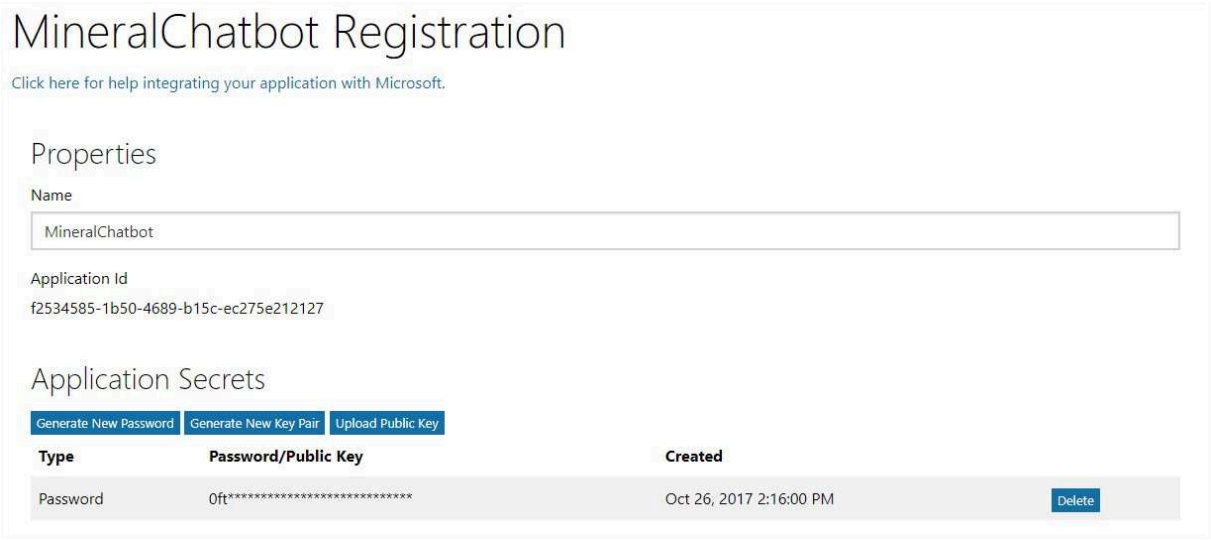

Figura 23 – Configurações de aplicativos Microsoft

aplicativo, isto é, identificador e chave de acesso. Após o registro é possível adicionar para comunicação com o *chatbot* algum dos canais disponibilizados pela solução, são estes: Aplicativo de e-mail, *GroupMe*, *Facebook Messenger*, *Kik*, *Skype*, *Slack*, *Microsoft Teams*, *Telegram*, texto/SMS, *Twilio*, Cortana e *Skype for Business*. Além disso, é oferecido um *token* de acesso que pode ser utilizado em páginas *Web*.

Para este trabalho os canais configurados foram: *Skype*, *Facebook Messeger* e uma página *Web* que será abordada na seção 3.5.

3.5 Aplicação *Frontend*

Para comunicação com o *chatbot* foi criada um página *Web* construída com a textitframework .Net Core e sob a arquitetura *Model View Controller*(MVC). Basicamente a arquitetura divide a aplicação em três camadas, sendo estas a de modelo que é responsável pela manipulação dos dados, a de visualização que é responsável pela exibição de informações ao usuário e por fim, a camada de controle que recebe requisições do usuário e controla qual modelo de dados será utilizado e qual tela será mostrada ao usuário.

A aplicação possui três telas distintas, uma página inicial, uma de informações sobre os recursos providos pelo *chatbot* e a página do *chatbot* propriamente dito. Para criação da página de comunicação foi necessário adicionar no código fonte o *token* de autorização fornecido na criação do recurso de configuração da canais. A Figura 24 ilustra a página de comunicação com o *chatbot*.

O uso de uma aplicação genérica para comunicação com o usuário pode fornecer recursos que os canais existentes não oferecem, por exemplo, ao usuário entrar na página,

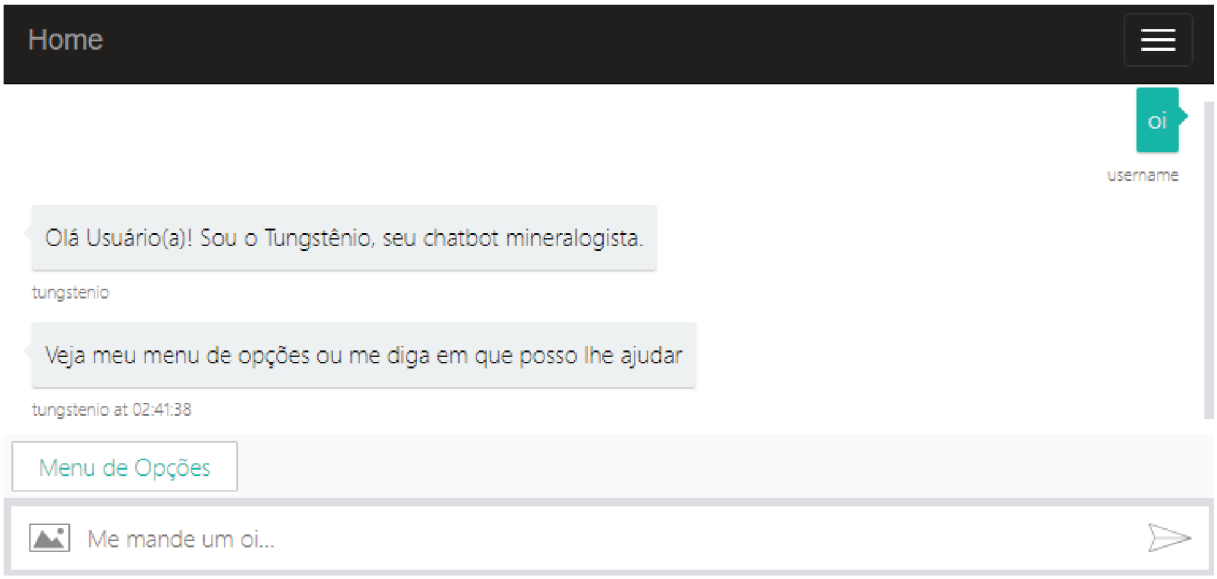

Figura 24 – Página *Web* de comunicação com chatbot

uma mensagem de cumprimento é enviada internamente ao *chatbot* e este responde fornecendo um ponto de partida na interação. Além disso, é possível estilizar da maneira desejada, fornecer informações extra ao usuário ou até mesmo evoluir os recursos disponibilizados pela aplicação. Uma possível evolução é a criação de uma página onde usuários autenticados podem fazer inserções, atualizações e deleções de minerais presentes no BD, de modo que o *Web Service* que comunica com o BD já suporta a utilização destes serviços.

4 Resultados

Os resultados contemplam dois pontos: o *chatbot* e a identificação dos minerais. O primeiro ponto visa trazer dados acerca do desempenho do *chatbot* ao realizar as requisições HTTP a fim de prover os recursos necessários. O segundo ponto diz respeito à identificação mineral mostrando quais resultados obtidos no utilizando a aplicação.

4.1 *Chatbot*

Para coleta dos dados quanto ao desempenho e uso do *chatbot*, foi utilizada a ferramenta de monitoramento *Application Insights*. Esta solução fornecida pela Microsoft provê dados das requisições de saída e entrada, erros, disponibilidade, entre outros.

Para sua utilização foi necessária a criação da ferramenta na plataforma de *cloud* Azure e a interligação ao *chatbot*. Assim, foi estabelecida comunicação simultânea com a aplicação por meio dos canais de comunicação *Facebook Messenger* e *Skype*. O tempo de conversação nos canais de forma simultânea foi de 30 minutos e os resultados obtidos são apresentados nas subseções 4.1.1, 4.1.2 e 4.1.3.

4.1.1 Requisições no Servidor

A primeira análise realizada foi quanto às requisições no servidor, também chamadas de requisições de entrada, estas representam todas as chamadas direcionadas ao *chatbot*. Este tipo de requisição, por padrão, é feita ao endereço *online* da aplicação concatenado ao complemento *"api/messages"*. Os gráficos das Figuras 25 e 26 ilustram a quantidade de requisições dos canais de comunicação feitas ao *chatbot* e o tempo de resposta em milissegundos, respectivamente.

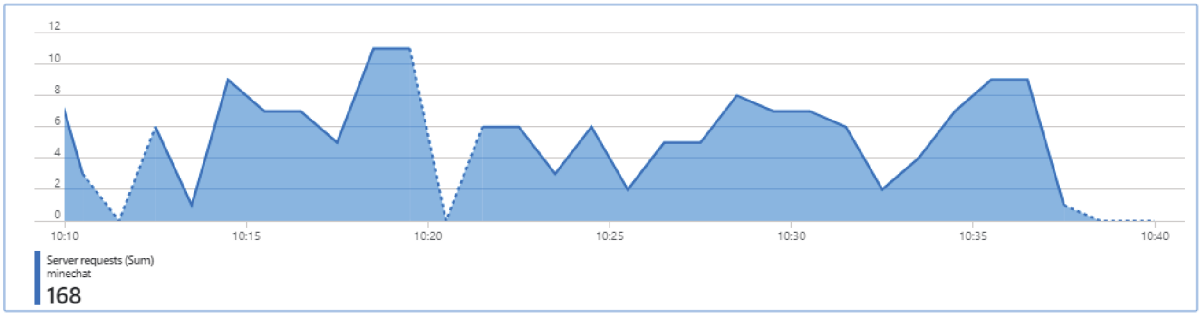

Figura 25 – Requisições ao servidor

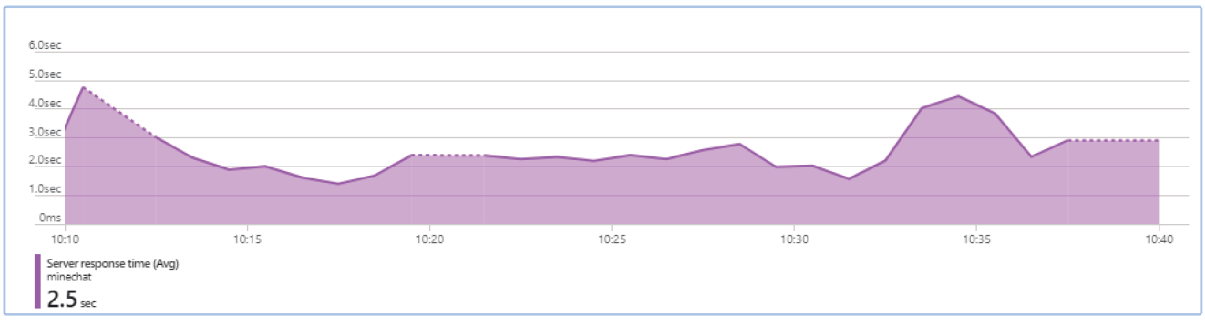

Figura 26 – Tempo de resposta do servidor

4.1.2 Requisições de Dependência

Outra análise feita foi quanto às requisições de dependência, também conhecidas por requisições de saída. Uma dependência é um componente externo chamado pelo *chat-bot*. No teste, as chamadas de dependência registrados foram ao *Web Service*, aos canais de comunicação, ao serviço cognitivo e aos componentes internos de controle de mensagens da biblioteca *Bot framework*. Os gráficos apresentados na Figura 27 e na Figura 28 ilustram as requisições feitas aos componentes externos no período de testes. Nestes são apresentados as variações das requisições com o somatório geral e o tempo médio das requisições em milissegundos, respectivamente.

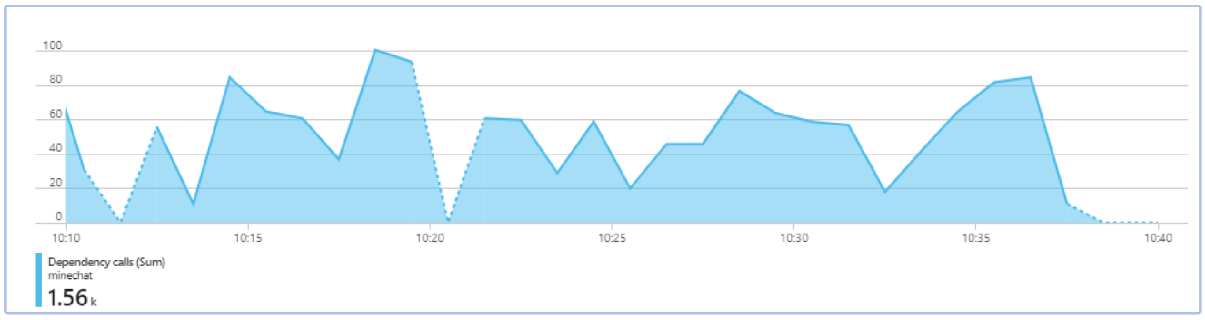

Figura 27 – Fluxo de requisições aos agentes externos

É possível perceber que o somatório de requisições atingiram aproximadamente 156.000, grande parte destas pertencentes ao controle interno de mensagens (cerca de 65.1%). Todavia, apesar da quantidade ser superior, o tempo médio de resposta das requisições desse tipo foi menor do que outras de menor quantidade. Os fatores para esta ocorrência podem ser diversos como, por exemplo, os recursos estarem alocados na mesma região do Azure e a paralelização das chamadas por parte da plataforma de *cloud*.

O gasto em processamento da aplicação nos testes acima é mostrado na Figura 29, que em seu maior pico não ultrapassou 2%.

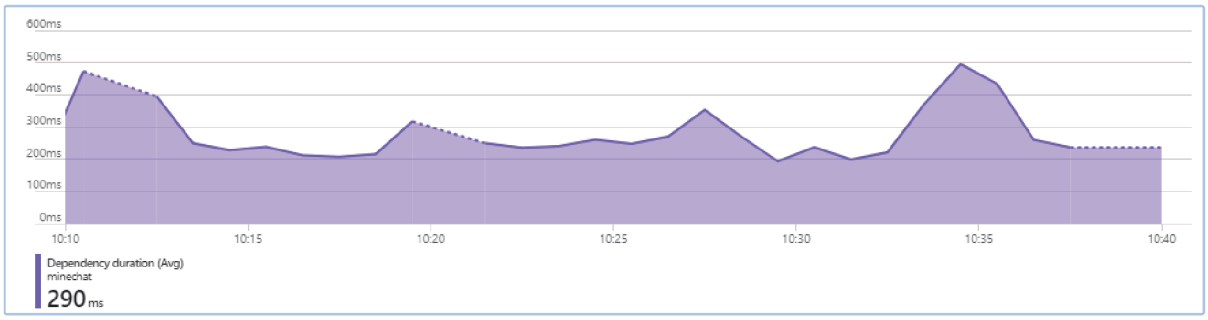

Figura 28 – Tempo de duração das requisições a agentes externos

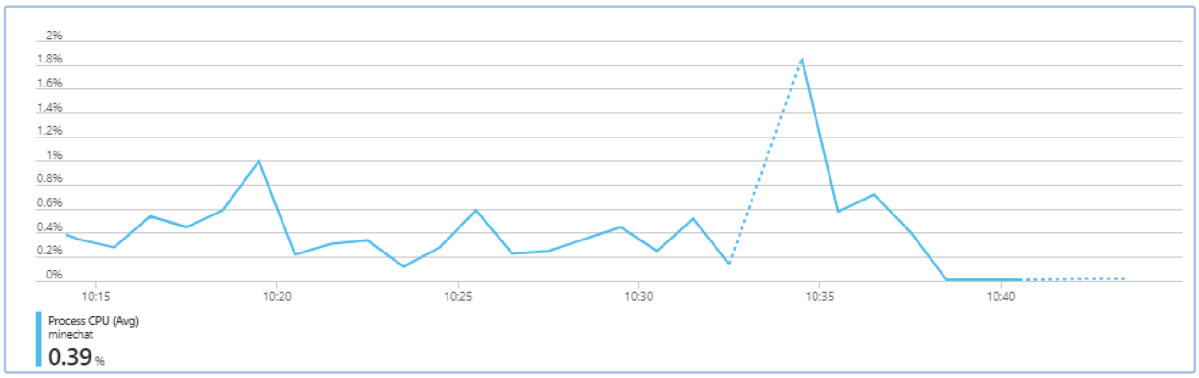

Figura 29 – Gasto de processamento da aplicação

4.1.3 Teste de Carga

Por fim, foi feito um teste de carga por meio da ferramenta *Application Insights* o qual simulou chamadas à aplicação de 250 usuários diferentes de forma simultânea em um período de cinco minutos. A Figura 30 mostra o resultado do teste que teve 70803 requisições das quais 100% tiveram sucesso com uma média de tempo de respostas de 40 milissegundos e uma quantidade de 236,01 de requisições por segundo.

4.2 Identificação

Para tabulação de resultados quanto à eficácia da identificação pelo *chatbot*, foi considerada uma amostra referente a 100 dos minerais que de maneira comum são discutidos por Klein e Dutrow (2009), Liccardo e Chodur (2017) e Navarro e Zanardo (2016). Deste modo foi possível estabelecer quais dos minerais são mais relevantes em análises.

O teste de identificação proposto foi buscar cada um dos minerais pelas suas características vendo qual destes ao final do processo retornou um resultado correto e único, correto e múltiplo ou incorreto. Para claro entendimento, o fluxo de desambiguação foi dividido em dois, como retratado no tópico 3.2.2.5, e estes serão chamados aqui de Fluxo

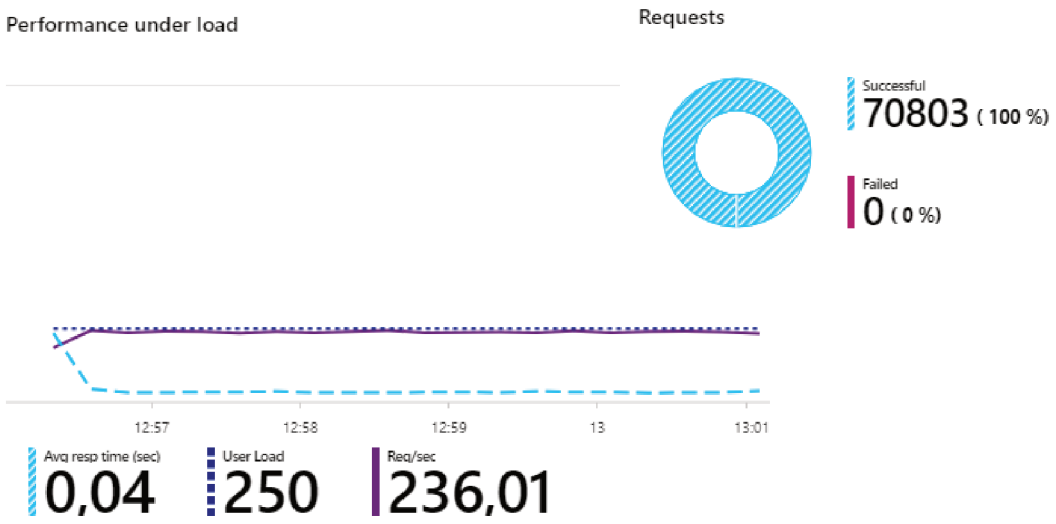

Figura 30 – Resultados do teste de carga

A e Fluxo B. A entrada dos dados foi um JSON de cada um dos minerais presentes na base obedecendo ao fluxo realizado pelo *chatbot*.

Para realização dos testes foram feitas requisições HTTP do tipo *POST* por meio da ferramenta Postman simulando dos dados que o *chatbot* envia ao *Web Service* para identificações. Seguindo o fluxo, no primeiro passo é enviado um objeto do tipo mineral com apenas a propriedade *isLiquid* preenchida. Caso o retorno obtivesse mais de um mineral, a requisição seria repetida mas enviando as próximas propriedades obedecendo o fluxo realizado pelo *chatbot*.

Ao decorrer do fluxo A, que finaliza ao usuário responder acerca da cor do traço do mineral, para alguns casos já é possível se obter uma resposta. Por exemplo, ao usuário informar que a propriedade física do mineral é líquida, já se descarta todas as possibilidades exceto o mercúrio. Ao dizer que o mineral é não metálico e de dureza 9, é possível afirmar com convicção que se trata do Coríndon, se for não metálico e de dureza 10, trata-se do diamante e assim por diante. Dos cem minerais testados no fluxo A 41 obtiveram resultados únicos e corretos, todavia foi preciso outro fluxo que garantisse a continuidade da desambiguação para o restante dos minerais.

No fluxo conversacional B, já se tem uma quantidade inferior de minerais, estes já restringidos pelo fluxo A. Desta maneira, foram feitas via Postman requisições HTTP contendo no seu corpo um objeto mineral com as propriedades mais específicas tais como: reação a HCl, fusibilidade, magnetismo, entre outros. Assim, obedecendo as regras estabelecidas no fluxo do *chatbot*, ao final do fluxo B é possível ver o resultado final da desambiguação dos minerais. Para cada mineral os testes se encerraram ao receber um resultado único ou ao chegar na última pergunta do fluxo B que é escolhida a partir da

Fluxo	Correto e único	Correto e múltiplo	Incorreto
Fluxo A	41	59	0
Fluxo B	94	6	0

Tabela 7 – Resultados de testes

análise dos mineiras filtrados mas ainda não completamente restringidos. É preciso lembrar que apesar do teste do fluxo B ser testado via *Web Sevice* no *chatbot* o processamento deste fluxo é realizado em memória via aplicação.

Ao final das requisições de teste, obedecendo o fluxo conversacional do *chatbot*, 94% dos minerais obtiveram um resultado único e correto e 6 minerais obtiveram resultados corretos porém não único. Por exemplo, os minerais Pirolusita e Molibdenita possuem brilho metálico, a mesma densidade, uma dureza semelhante e também cores semelhantes, assim, o *chatbot* ao final do fluxo devolveu um resultado contendo os dois minerais. Desta forma, é possível perceber que a desambiguação de minerais com propriedades muito parecidas depende da análise de outras características intrínsecas exclusivamente ao mineral analisado.

A tabela 7 ilustra os resultados obtidos ao final do teste e a Figura 31 mostra os resultados de maneira gráfica. Vale ressaltar que os testes ocorreram já sabendo antemão as características de cada mineral, todavia, um usuário comum ao tentar identificar um mineral não necessariamente terá os dados exatos para identificação. Portanto, é possível que os resultados variem de acordo com a percepção das características do mineral a ser identificado. Por exemplo, as propriedades de cor e cor do traço analisadas nos dois últimos passos do fluxo A podem variar de usuário para usuário sem estarem necessariamente erradas, essa percepção da propriedade poderia refletir em um resultado único já no fluxo A para alguns usuário e para outros somente ao passarem pelo fluxo conversacional B.

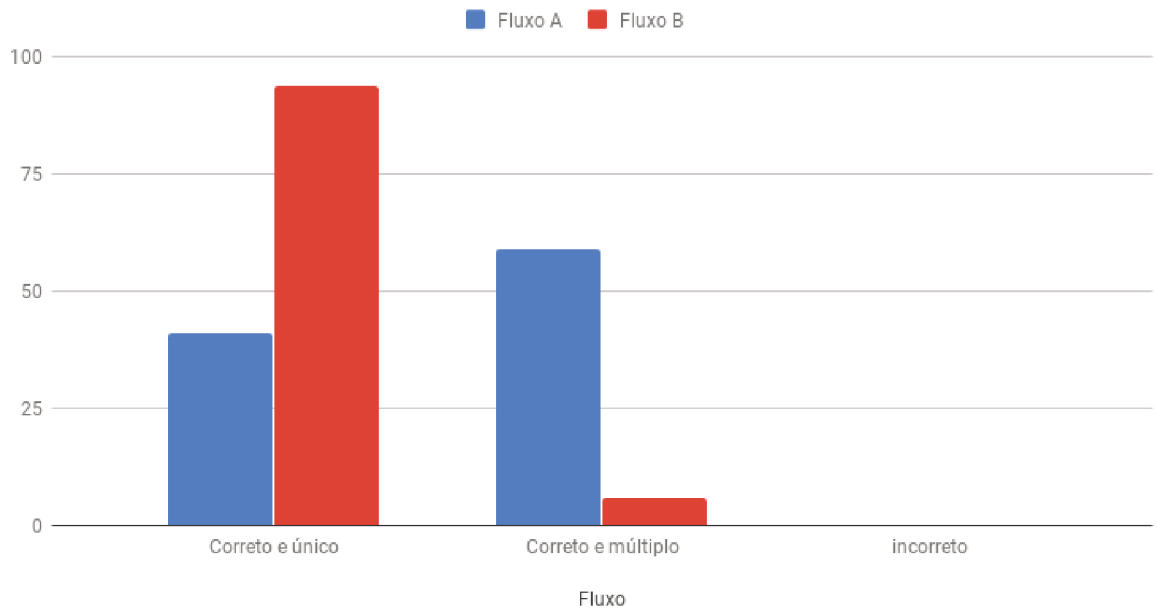

Figura 31 – Representação gráfica dos resultados

5 Conclusão

Pode se concluir que o *chatbot* se apresenta como uma solução alternativa para uma tarefa manual e muitas vezes demorada que é a identificação de alguns minerais, todavia trazendo consigo recursos e facilidades que prevê meios para se chegar ao resultado desejado de maneira rápida e concisa. Ademais, a ferramenta não se limita apenas a identificação possibilitando também a busca informações de determinado mineral por meio do seu nome sendo assim uma aplicação instrutiva sobre os minerais.

A desvantagem do uso da aplicação é a quantidade limitada de minerais a serem desambiguados. Por utilizar perguntas e regras fixas não havendo aprendizagem e recriação das regras, o *chatbot* se limita à uma quantidade pequena dos minerais. Apesar dos minerais mais comumente utilizados em análises corresponderem à uma pequena fração da quantidade total conhecida, para utilização do total de minerais conhecidos através do *chatbot* seria necessário aplicar técnicas de inteligência artificial e aprendizado de máquina no processo de desambiguação. A vantagem e desvantagem dessa aplicação é apresentada na sessão 5.1.

O uso da aplicação é destinada principalmente as pessoas ligadas as áreas de estudo dos minerais por estas já possuírem um conhecimento prévio das propriedades e suas aplicações. Também pode ser considerada uma ferramenta útil para ingressantes em cursos ligados à esta área como o curso de geologia ou ciências geológicas, permitindo assim a obtenção de informações que facilitam o entendimento das disciplinas iniciais no qual é ensinada a desambiguação dos minerais pelas suas características.

Além da identificação mineral propriamente dita, o método também atingiu o propósito de ser uma aplicação acessível, pois pode ser utilizado através de Redes Sociais como *Facebook* e *Skype* comumente utilizadas por grande parte da população, intuitivo, pois utiliza da linguagem humana comum para obtenção de informações e por ter respostas rápidas e precisas, pois integra com serviços externos para obtenção de informações.

Outro fator de relevância é o fato da ferramenta não se tornar obsoleta à medida que os canais de comunicação evoluem de modo que a evolução dos aplicativos de conversas resultam consequentemente no uso da ferramenta de identificação mineral. Não obstante, a popularização de ferramentas como esta permite que a medida que surgem novos aplicativos de conversas estes já se preocupem em viabilizar meios para se conectar a *chatbots*. Através disso é possível entender a afirmação de Shevat (2017) ao dizer que os *chatbots* se apresentam como o futuro dos *softwares*.

5.1 Trabalhos Futuros

Os testes se mostraram promissores em bases pequenas e bem estruturadas, porém uma possível evolução da aplicação seria a adição de perguntas sobre outras propriedades para desambiguação. Apesar da ferramenta atender a demanda de bases de dados que contenham minerais comumente estudados em laboratórios, a identificação de minerais menos comuns é de fundamental importância em estudos geológicos, pois os minerais mais estudados acabam se tornando fáceis de serem identificados.

Mais uma possibilidade de evolução consiste no uso de um modelo baseado em inteligência artificial para comparações de desambiguação pois o *Web Service* criado para processamento das informações o usuário utiliza lógicas engessadas de comparação das propriedades minerais para retorno dos resultados. Através do uso de um modelo de rede neural seria possível transformar as comparações algo dinâmico que variaria de acordo com o treinamento e aprendizado da rede. A vantagem dessa aplicação seria a otimização nas consultas e a possibilidades de atingir uma quantidade maior de minerais, porém esta utilização esconderia o processo pelo qual se passa até o resultado, ou seja, seria imputada as informações e os cálculos e processamentos seriam desconhecidos. Isso se caracteriza como uma desvantagem pois o caminho percorrido até o resultado é uma informação importante para as pessoas ligadas as áreas geológicas que estariam fazendo os testes.

Outra possível evolução é a integração do *chatbot* com todos os canais de comunicação existentes que possibilitam seu uso tais como: *GroupMe*, *Kik*, *Slack*, *Microsoft Teams*, *Telegram*, *Twilio*, Cortana, entre outros; isso aumenta a democratização e facilidade do uso. O *WhatsApp* que se apresenta até o momento como o mais popular aplicativo de conversas ainda não abriu completamente a possibilidade de integração com *chatbots*, apesar disso já ter sido discutido pela organização.

Referências

BRESLOW, L. A.; AHA, D. W. Simplifying decision trees: A survey. *The Knowledge Engineering Review*, Cambridge University Press, v. 12, n. 1, p. 1–40, 1997. Citado 2 vezes nas páginas 18 e 19.

CHANDEL, S. et al. Chatbot: Efficient and utility-based platform. In: SPRINGER. *Science and Information Conference*. [S.l.], 2018. p. 109–122. Citado na página 11.

CHARNIAK, E.; MCDERMOTT, D. Introduction to ai. *Reading (Mass.): Addison*, 1985. Citado na página 15.

CLARK, T. The turing test as a novel form of hermeneutics. *International Studies in Philosophy*, v. 24, n. 1, p. 17–31, 1992. Citado na página 10.

COLE, D. Artificial intelligence and personal identity. *Synthese*, Springer, v. 88, n. 3, p. 399–417, 1991. Citado na página 10.

KLEIN, C.; DUTROW, B. *Manual de ciência dos minerais*. [S.l.]: Bookman Editora, 2009. Citado 9 vezes nas páginas 19, 20, 21, 23, 24, 25, 49, 58 e 60.

KURZWEIL, R. et al. *The age of intelligent machines*. [S.l.]: MIT press Cambridge, MA, 1990. v. 579. Citado na página 15.

LERMAN, J. *Programming Entity Framework: Building Data Centric Apps with the ADO. NET Entity Framework*. [S.l.]: "O'Reilly Media, Inc.", 2010. Citado 2 vezes nas páginas 16 e 17.

LERMAN, J.; MILLER, R. *Programming Entity Framework: Code First: Creating and Configuring Data Models from Your Classes*. [S.l.]: "O'Reilly Media, Inc.", 2011. Citado na página 17.

LICCARDO, A.; CHODUR, N. *Os Minerais: Elementos da geodiversidade*. [S.l.]: Bookman Editora, 2017. Citado 9 vezes nas páginas 20, 21, 22, 24, 25, 49, 58, 59 e 60.

LINHORST, E. F. *Jolly balance*. [S.l.]: Google Patents, 1953. US Patent 2,650,494. Citado na página 24.

MARTIN, R. C. *Clean architecture: a craftsman's guide to software structure and design*. [S.l.]: Prentice Hall Press, 2017. Citado 2 vezes nas páginas 15 e 16.

MENEZES, S. de O. *Minerais comuns e de importância econômica: um manual fácil*. [S.l.]: Oficina de Textos, 2012. Citado 2 vezes nas páginas 19 e 23.

NAVARRO, G. R. B.; ZANARDO, A. Tabelas para determinação de minerais. *Material Didático do Curso de Geologia/UNESP*, 2016. Citado 3 vezes nas páginas 25, 26 e 49.

NICKEL, E. H. *Definition of a mineral*. [S.l.]: De Gruyter, 1995. Citado na página 19.

POUGH, F. H. *A field guide to rocks and minerals*. [S.l.]: Houghton Mifflin Harcourt, 1996. Citado na página 20.

QUINLAN, J. R. Simplifying decision trees. *International journal of man-machine studies*, Elsevier, v. 27, n. 3, p. 221–234, 1987. Citado na página 18.

RICH, E.; KNIGHT, K. Learning in neural network. *McGraw-Hill, New York*, 1991. Citado na página 15.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. [S.l.]: Malaysia; Pearson Education Limited,, 2016. Citado na página 15.

SANTOS, C. d. Fundamentos do entity framework 4. *MVA–Microsoft Virtual Academy, disponível em https://msdn.microsoft.com/pt-br/library/jj128157.aspx, acesso*, v. 19, 2013. Citado na página 17.

SERVER, S. Microsoft sql server. *Retrieved August 1st*, 2010. Nenhuma citação no texto.

SHEVAT, A. *Designing Bots: Creating Conversational Experiences*. [S.l.]: "O'Reilly Media, Inc.", 2017. Citado 4 vezes nas páginas 11, 14, 15 e 53.

SRIVASTAVA, B.; KOEHLER, J. Web service composition-current solutions and open problems. In: *ICAPS 2003 workshop on Planning for Web Services*. [S.l.: s.n.], 2003. v. 35, p. 28–35. Citado na página 37.

SZWARCFITER, J. L.; MARKENZON, L. *Estruturas de Dados e seus Algoritmos*. [S.l.]: Livros Técnicos e Científicos, 1994. v. 2. Citado na página 18.

TABOR, D. Mohs's hardness scale-a physical interpretation. *Proceedings of the Physical Society. Section B*, IOP Publishing, v. 67, n. 3, p. 249, 1954. Citado na página 23.

TURING, A. *Computing machinery and intelligence, Mind LIX, 433-60*. 1950. Citado 2 vezes nas páginas 10 e 15.

WILLIAMS, J. D. et al. Fast and easy language understanding for dialog systems with microsoft language understanding intelligent service (luis). p. 159–161, 2015. Citado na página 29.

Apêndices

APÊNDICE A – Outras propriedades

Algumas das propriedades minerais, apesar de não serem utilizadas diretamente no desenvolvimento deste trabalho, possuem relevância quanto à identificação mineral. A adição destas para desambiguação, seria essencial para uso da ferramenta em bases maiores. A seguir, estas propriedades serão discutidas assim como a sua possível participação na identificação.

A.1 Hábito dos Minerais

De modo geral, o hábito do mineral representa a forma comum que ele possui. Em seu livro, o autor define hábito em minerais como sendo:

> [..] Formato mais frequente, ou a combinação de formas com a qual o mineral se apresenta na natureza. Trata-se de uma propriedade bastante importante, visto que muitos minerais são facilmente reconhecidos apenas pela sua morfologia externa, como a fluorita, a granada, a mica ou o zircão. (LICCARDO; CHODUR, 2017, p. 72)

A tabela 8 apresenta os principais hábitos para cristais isolados e a tabela 9 apresenta os principais hábitos para minerais de maneira geral.

A.2 Tenacidade

De acordo com (KLEIN; DUTROW, 2009, p. 58) a tenacidade consiste na "[..] resistência de um mineral a romper-se ou deformar-se." A deformidade gerada no mineral quando este se rompe por imposição de força externa, é utilizada como uma característica na identificação do mineral. A tabela 10 mostra os termos comumente utilizados para descrever as deformidades.

A.3 Outras

Além das propriedades já mencionadas, há características bem peculiares a alguns minerais como a forma cristalina, o gosto, o odor, propriedades elétricas, propriedades térmicas, entre outras, que podem ser de fundamental importância no arranjo das perguntas feitas pelo *chatbot*.

Hábito	**Exemplos**
Cúbico	pirita, fluorita, halita, galena
Octaédrico	magnetita, diamante, fluorita
Dodecaédrico ou Rombododecaédrico	granadas, diamante
Tetraédrico	diamante
Trapezoédrico	leucita, granadas
Piramidal ou bipiramidal	zircão,
Romboédrico	calcita
Prismático (ou colunar)	berilo, turmalina, quartzo, piroxênios, anfibólios
Tabular(placas achatadas)	barita, albita, hematita, micas
Laminado (lâmina de faca)	cianita, molibdenita
Acicular (semelhante a agulhas)	rutilo, actinolita, natrolita, estibnita
Capilar ou filiforme (semelhante a fios ou cabelos)	prata nativa, amianto

Tabela 8 – Principais hábitos para cristais isolados

Baseado em Liccardo e Chodur (2017)

Hábito	**Exemplos**
Maciço ou compacto: agregado de microcristais	caulim, ágata, calcedônia
Granular: agregado de grãos aproximadamente equidimensionais	barita, calcita
Reticulado: agregado de cristais aciculares, formando um retículo ou grade	rutilo, cuprita
Fibroso: agregado compacto de cristais delgados (filiformes).	asbestos
Radial (ou divergente): agregado de cristais finos de forma radial	goethita, malaquita
Micáceo, lamelar ou foliáceo: agregado de pequenas folhas ou placas delgadas	micas, talco, grafita, hematita
Drusiforme: agregado de cristais que revestem uma superfície	ametista, calcita
Geodo: uma drusa mais ou menos esférica	quartzo, calcita, ametista
Dendrítico ou arborescente: agregado semelhante a galhos ou a folhas de plantas	ouro, prata e cobre nativos
Estalactítico: agregado em forma de cone ou cilindro	calcita, malaquita, aragonita
Globular ou esferulítico: agregado aproximadamente esferoidal	limonita, goethita, hematita
Botrioidal: semelhante a cacho de uvas	limonita, goethita, hematita
Mamelar: agregado semelhante a mama	limonita, goethita, hematita
Reniforme: agregado semelhante a rim	agregado
Concêntrico: agregado mais ou menos circular	ágata, malaquita
Bandado: agregado em faixas de cor ou textura diferente	ágata
Oolítico: agregado semelhante a ovas de peixe	calcita, limonita
Pisolítico: agregado de esferas maiores que uma ervilha	calcita, limonita

Tabela 9 – Hábito de minerais de modo geral

Baseado em Liccardo e Chodur (2017)

Termo	**Descrição**	**Exemplos**
Quebradiço	Rompe e se pulveriza facilmente.	Halita
Maleável	Pode ser partido em estampas delgadas.	Cobre
Séctil	Pode ser cortado em aparas delgadas com um canivete.	Calcocita
Dúctil	Pode ser estirado para formar fios.	Ouro
Flexível	Se encurva, mas não retorna à posição original quando a pressão cessou.	Folhas de clorita, talco
Elástico	Depois de ter sido encurvado, retoma sua posição original ao cessar a pressão	Micas

Tabela 10 – Termos utilizados para descrever a tenacidade de um mineral

Baseado em Klein e Dutrow (2009)