



UNIVERSIDADE ESTADUAL DO TOCANTINS

CURSO DE SISTEMAS DE INFORMAÇÃO

WESLEY TAVARES DE AGUIAR E QUADROS

**SOLUÇÃO WEB RESPONSIVA PARA SERVIÇOS DE
TRATAMENTOS MÉDICOS E PRÉ-DIAGNÓSTICO VIA
CHATBOT**

Palmas

2019



UNIVERSIDADE ESTADUAL DO TOCANTINS

CURSO DE SISTEMAS DE INFORMAÇÃO

WESLEY TAVARES DE AGUIAR E QUADROS

**SOLUÇÃO WEB RESPONSIVA PARA SERVIÇOS DE
TRATAMENTOS MÉDICOS E PRÉ-DIAGNÓSTICO VIA
CHATBOT**

Projeto apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS como parte dos requisitos para a obter a aprovação e colação de grau como Bacharel em Sistemas de Informação, sob a orientação do professor Me. Alex Coelho.

Palmas

2019

Dedico aos meus pais que me incentivaram e torcem pelo meu sucesso me dando forças pra nunca desistir, e a todos que acreditaram em mim e me ajudaram na conclusão desse trabalho.

Agradecimentos

Àquele que está me permitindo tudo isso, e por ter me concedido força para superar todas as dificuldades até aqui enfrentadas durante esse período, a Ti meu DEUS, obrigado! Reconheço cada vez mais em todos os meus momentos, que Tu és o meu maior Mestre!

Agradeço aos meus pais, por absolutamente tudo! Cada um de seus atos foi uma oportunidade que eu tive para crescer e me tornar o que sou, e que nos momentos de minha ausência, sempre fizeram entender que o futuro, é feito a partir da constante dedicação no presente.

Aos meus irmãos, pelo apoio e amor incondicional e que felizmente posso dizer ser recíproco. A minha querida namorada, que também sempre esteve comigo, me apoiando e suportando a minha ausência nos dias mais difíceis durante esse processo.

A Universidade Estadual do Tocantins, por estar tornando possível minha formação, ao meu orientador Prof. Alex Coelho e todos os outros envolvidos, pelo tempo e paciência a mim dedicados.

Aos meus colegas e amigos que aqui fiz, divido com vocês essa conquista, porque ela também pertence a vocês! Levarei vocês em meu coração. Sempre estarão em minhas orações. A todos Muito Obrigado!!!

*“Não fui eu que ordenei a você?
Seja forte e corajoso!
Não se apavore nem desanime,
pois o Senhor, o seu Deus, estará
com você por onde você andar.”
(Bíblia Sagrada, Josué 1:9)*

Resumo

Neste trabalho será apresentado um estudo sobre as diversas ferramentas e desenvolvimento de uma solução que contribui para a informatização e automatização dos serviços médicos, a fim de resolver as problemáticas que aqui serão citadas. Aqui foi arquitetado todo o escopo do projeto, bem como fundamentos para se desenvolver uma aplicação web resposiva, ou seja, uma aplicação que se adapta a qualquer dispositivo, que visa a realização de gerenciamento e agendamento de consultas médicas. Além disso, foi feita uma abordagem da criação de um assistente médico virtual(Chatbot) que visa realizar pré-diagnósticos médicos, contribuindo para tomada de decisão do paciente. O Chatbot foi criado utilizando ferramentas disponíveis no mercado, que serão fundamentais para a criação de uma API própria que possa ser integrada a qualquer sistema especialista.

Palavras-chaves: Chatbot, Sistema Web, Responsividade, Inteligencia Artificial, Sistemas Especialistas, Redes neurais Artificiais, Processamento de Linguagem Natural, Aprendizado de Maquina, Reconhecimento de imagem

Abstract

This project will present a study on the various tools and development of a solution that can contribute to the computerization and automation of medical services, in order to solve the problems that will be mentioned here. Here the entire scope of the project will be architected. The proposal is to develop a responsive web application, that is, an application that adapts to any device, which is aimed at the management and scheduling of medical consultations. In addition, an approach will be taken to create a virtual medical assistant (Chatbot), which aims to perform a medical pre-diagnosis, contributing to patient decision-making. Chatbot will be created using tools available in the market, which will be fundamental for the creation of an own API that can be integrated to any expert system.

Key-words: *Chatbot, Web System, Responsiveness, Artificial Intelligence, Expert Systems, Artificial Neural Networks, Natural Language Processing, Machine Learning, Image recognition.*

Lista de ilustrações

Figura 1 – Elementos básicos do Angular	17
Figura 2 – Modelo operacional do MongoDB	19
Figura 3 – Detalhamento do funcionamento do MVC	20
Figura 4 – Ilustração de um Layout Responsivo	22
Figura 5 – Exemplo de página com Bootstrap	23
Figura 6 – Diagrama de blocos do Aprendizagem Supervisionada	26
Figura 7 – Representação de camadas RNA	27
Figura 8 – Exemplo básico do funcionamento do NLP	29
Figura 9 – Funcionamento de um Sistema Especialista.	30
Figura 10 – Funcionamento do Chatbot	33
Figura 11 – Exemplo de classificação de imagens no VR	38
Figura 12 – Figura demonstrativa do funcionamento do Node-RED	39
Figura 13 – Metodologia do projeto	42
Figura 14 – Arquitetura da solução proposta	45
Figura 15 – Diagrama de Caso de uso	47
Figura 16 – Diagrama de Classe (Parte do paciente)	48
Figura 17 – Diagrama de Classe (Parte do Médico)	49
Figura 18 – Comparação entre Watson Assistant e Dialogflow	50
Figura 19 – Fluxograma do Chatbot	52
Figura 20 – Protótipo inicial do sistema desenhado	53
Figura 21 – Protótipo de tela Implementado	54
Figura 22 – Protótipo de Chatbot Implementado(Watson Assitant e Dialogflow).	55
Figura 23 – Fluxo de diálogo - Watson Assistant.	56
Figura 24 – Assistente médico em execução.	59
Figura 25 – Imagens de queimaduras	60
Figura 26 – Fluxo da integração dos serviços no Node-RED	61
Figura 27 – Resposta do chatbot x Resposta da API	62
Figura 28 – Tela de Login da aplicação	66
Figura 29 – Painel do médico	68
Figura 30 – Tela agendamento de consulta por parte do Paciente	68
Figura 31 – Tela agendamento de consulta responsiva (Mobile)	69
Figura 32 – Diagrama de classe	86
Figura 33 – Diagrama de Componentes	87
Figura 34 – Diagrama de sequencia (Cadastrar Paciente)	89
Figura 35 – Diagrama de sequencia (Agendar Consulta)	90
Figura 36 – Diagrama de Atividade (Cadastrar Paciente)	91

Figura 37 – Diagrama de Atividade (Agendar Consulta) 92

Lista de tabelas

Tabela 1 – Requisitos Funcionais.	77
Tabela 2 – Requisitos Não Funcionais.	78

Lista de abreviaturas e siglas

API - *Application Programming Interface* (Interface de Programação de Aplicativos).

HTML - *HyperText Markup Language* (Linguagem de Marcação de Hipertexto).

IA - Inteligência Artificial

IHC - Interação Humano Computador

JSF - JavaServer Faces

MVC - *Model-View-Controller* (Modelo-visão-controlador).

NLP - *Natural Language Processing* (Processamento de Liguagem Natural).

RNA - Redes Neurais Artificiais

SE - Sistemas Especialistas

UI - *User Interface* (Interface de Usuário).

UX - *User Experience* (Experiencia do Usuário).

VR - *Visual Recognition*

XML - *Extensible Markup Language* (Linguagem Extensível de Marcação).

Sumário

1	INTRODUÇÃO	13
2	REFERENCIAL TEÓRICO	16
2.1	Linguagem Java e Framework JSF	16
2.2	Angular	17
2.2.1	TypeScript	18
2.3	Node.js	18
2.4	MongoDB	19
2.5	Padrão de arquitetura MVC	20
2.6	Interface Homem Computador(IHC)	21
2.6.1	Responsividade	21
2.6.2	Framework Bootstrap	23
2.6.3	Framework Materialize - Material Design	24
2.7	Inteligencia Artificial	24
2.7.1	<i>Machine Learning</i> (Aprendizado de Maquina)	25
2.7.2	Redes Neurais Artificiais - RNA	27
2.7.3	Natural Language Processing (NLP)	28
2.7.4	Sistemas Especialistas	29
2.7.5	Chatbots	31
2.8	APIs e Plataformas para Criação de Chatbots	34
2.8.1	API IBM Watson:	34
2.8.1.1	Vantagens e Desvantagens do Watson Assistant	34
2.8.2	Dialogflow (API.AI)	35
2.8.2.1	Vantagens e Desvantagens do Dialogflow	35
2.8.3	API WIT.AI	36
2.8.3.1	Vantagens e Desvantagens do Wit.ai	36
2.8.4	Chatfuel	37
2.8.5	Botpress	37
2.8.6	Watson Visual Recognition	37
2.8.6.1	Node-RED	39
2.9	Solução correlata	40
2.9.1	ApiMedic	40
3	METODOLOGIA	41
4	RESULTADOS	45

4.1	Arquitetura geral do Sistema	45
4.2	Diagramas	46
4.2.1	Diagrama de Caso de Uso	47
4.2.2	Diagrama de Classe	48
4.3	Watson Assistant vs Dialogflow	50
4.4	Fluxograma Chatbot	51
4.5	Protótipos	53
4.5.1	Protótipo da aplicação	53
4.5.2	Protótipo de Chatbot	54
4.6	Desenvolvimento	55
4.6.1	Chatbot - Assistente Médico Virtual	55
4.6.1.1	Diagnóstico por imagem - Visual Recognition	60
4.6.2	Estrutura do Back-end	63
4.6.3	Aplicação web	65
5	CONCLUSÃO	70
	REFERÊNCIAS	72
	Appendices	75
	A LEVANTAMENTO DE REQUISITOS	76
	B DETALHAMENTO DO CASO DE USO	79
	C DIAGRAMA DE CLASSE	85
	D DIAGRAMA DE COMPONENTES	87
	E DIAGRAMAS DE SEQUENCIA	88
	F DIAGRAMAS DE ATIVIDADE	91

1 Introdução

A situação da saúde brasileira passa por vários momentos difíceis, no qual temos bons médicos, bons profissionais, embora exista um grande impasse se tratando de saúde Pública e Privada. Porém, ao se falar de saúde no Brasil, temos que abordar inúmeros fatores que são os elementos motivacionais para o estudo desse projeto.

A saúde pública do Brasil, sendo fundamentada no Sistema Único de Saúde(SUS) tem se estruturado afim de cumprir com o dever que assim ficou definido pela Constituição de 1988, na qual ficou estabelecido que a saúde é direito de todos e dever do estado. É visível, que, por mais que o SUS tenha sido baseado em um princípio aparentemente impecável, o Sistema não tem atendido como se esperava. Os atendimentos deixam muito a desejar, sendo visto em vários hospitais, as enormes filas de espera, falta de médicos e entre outro problemas.

De acordo com [Mendonça \(2015\)](#), isso ocorre devido ao baixo valor investido pelo Estado na saúde pública, sendo basicamente 100 bilhões de dólares anuais para custear toda a operação, e tendo que atender mais de 150 milhões de pessoas, fazendo com que não haja o devido investimento necessário para demandar toda a população, com uma saúde de qualidade.

Por outro lado, temos a saúde Privada, na qual se encontram grande centros médicos, clínicas especializadas, melhores atendimentos e melhores profissionais. Atendendo mais de 48 milhões de pessoas no Brasil ([LOFTI, 2016](#)), a saúde privada tem crescido muito como uma alternativa mais viável e confiável para poder obter o serviço esperado. Porém, mesmo com esse número que vem crescendo exponencialmente, temos milhões de pessoas que ainda não tem condições de pagar um procedimento particular, ou um plano de saúde por exemplo, fazendo com que fiquem a mercê da saúde pública e enfrentem a longas filas de espera. Além disso, os custos com essa cobertura e procedimentos privados ainda estão relativamente caros.

A sociedade hoje busca resolver seus problemas de forma mais rápida. As filas do SUS, as filas em hospitais e clinicas para poder realizar uma simples triagem são motivos para que contribuam para a ineficiência e falta de qualidade de atendimento na área de saúde atual. Dessa forma, uma solução que visa a informatização desses serviços, trazendo-os mais pra perto do paciente, mais transparente, pode ser um pontapé para melhoria dos serviços médicos. Neste contexto, uma aplicação *responsiva*¹ atenderia um número significativo de pacientes em busca de agendamentos de consultas, realização de pagamentos facilitados, entre outro serviços que fazem dessa solução uma possível forma

¹ Um site flexível, cujo seu layout se adapta automaticamente ao dispositivo de acesso. ([PENZE, 2016](#))

de resolução de problemas como, pegar filas, deslocamento entre outros fatores, melhores condições de pagamento, com a utilização de Cartão Virtual agregado de vantagens a qualquer operação.

Assim, o presente trabalho propõe a utilização de linguagens e *frameworks* para a web que estão em crescimento no mercado. Com isso, é realizado o estudo de um fator que é o primordial dessa solução, que é a implantação de um *ChatBot*² para realização de um pré-diagnóstico capaz de detectar possíveis doenças e direcionar o paciente ao médico específico. O estudo do *ChatBot* foi realizado em um modelo de conversação “*text-to-text*” que faz um diagnóstico do paciente utilizando a linguagem natural, em que se fundamenta na ideia de que o usuário possa descrever o que está sentindo e sua condição, sendo então necessário aprofundar no estudo de técnicas da Inteligência Artificial como de a NLP, que na seção 2 foi detalhado.

Em vista do cenário atual, que foi descrito acima, o brasileiro vive em uma situação desconfortável em relação a serviços médicos, ficando claramente perceptível que a solução aqui proposta, tende a tornar a rotina do usuário mais satisfatória em relação a disponibilidade de serviços, pois nos dias atuais comprehende-se que o usuário final busca por agilidade e rapidez no processo.

Portanto, nesse trabalho, temos como objetivo, realizar, através de estudo de caso, um projeto que possa viabilizar a automação dos serviços médicos para contribuir na melhoria deste serviços no Brasil, assim como, estudar ferramentas e métodos para montar a arquitetura de uma aplicação web, utilizando-se das normas da IHC para criar um protótipo do sistema, que garanta uma melhor experiência do usuário. Também é definida a metodologia de criação e implantação do Chatbot, assim como as ferramentas de IA que são utilizadas no desenvolvimento. Então, será realizado o levantamento dos requisitos baseado na resolução da problemática aqui levantada, e por meio destes, estruturar todo o projeto.

Deste modo, com essa solução obteve-se a compreensão da metodologia para a implantação de um sistema Web responsivo integrado a um Chatbot, com o intuito de que no final seja apresentados todos os elementos que compõem essa solução, como a arquitetura do projeto, diagramas, e um protótipo do sistema, para um futuro desenvolvimento.

Este trabalho está estruturado da seguinte forma: primeiramente, na seção 2 será mostrado o referencial teórico utilizados para fundamentar o tema e a ideia proposta de acordo com todas as ferramentas estudadas a fim de utilização posterior; na seção 3 será apontado os métodos utilizados para compor o estudo, assim como as ferramentas utilizadas durante todo o processo; na seção 4 serão apresentados os resultados referente ao projeto. Por fim, é apresentada a conclusão e foi especificado as referências que foram

² Programa que simula um ser humano na conversação com as pessoas. (GHISE, 2016)

utilizadas no referido estudo.

2 Referencial Teórico

Nesta seção será discorrido sobre todo o referencial estudado, mostrando o entendimento sobre o assunto e explicando a funcionalidade de cada uma e a aplicação prática.

2.1 Linguagem Java e Framework JSF

A linguagem Java foi criada em meados da década de 90 pela empresa, Sun Microsystem, com intuito de resolver alguns problemas que se tinha na época , como a organização, a inexistência de bibliotecas, o uso de ponteiros, o gerenciamento de memória, que até então é implementado com grande frequência em outras linguagens como a linguagem C. ([LUCKOW; MELO, 2010](#))

Desde a época da sua criação, até os dias atuais, o Java já passou por inúmeras mudanças. Inicialmente não foi bem aceita no mercado, porém a partir das primeiras atualizações ela já começou a ganhar espaço, fazendo com que em 2009, a Oracle(uma das empresas que mais utilizava serviços por meio da plataforma Java) comprasse a Sun que veio a fortalecer a marca.

Com o passar dos anos, a evolução da computação e a abrangência da Internet mundialmente, surgiu uma necessidade das empresas em expandir seus negócios para a Web. Dessa forma, a Sun criou o Java EE(Java Enterprise Edition), que é uma plataforma de desenvolvimento que possui um conjunto de especificações que são responsáveis por facilitar o desenvolvimento das aplicações que as empresas utilizam (Enterprise). Com o Java EE pode ser estabelecido um conjunto de tarefas que organizam e facilitam o desenvolvimento, tais como, as persistências dos dados, tratamento das requisições do protocolo de HTTP, validações e entre outros.([FARIA, 2015](#))

Uma das especificações usadas no processo de desenvolvimento Web é o JSF, Java Server Faces. Em seu artigo, [Faria \(2015\)](#) explica que o JSF é um framework Java que foi desenvolvido para a construção de interfaces de usuário, baseada nos componentes para aplicações voltadas para Web, fazendo com que seja distinguido as regras de negócio para com os serviços, se baseando no modelo MVC.

Por ter uma grande quantidade de componentes, e o design flexível, o JSF vem se acomodando nas novas tecnologias. Dessa forma a Google tem investido no Angular para desenvolvimento Web, no qual sempre tem apresentado tecnologias e recursos novos.

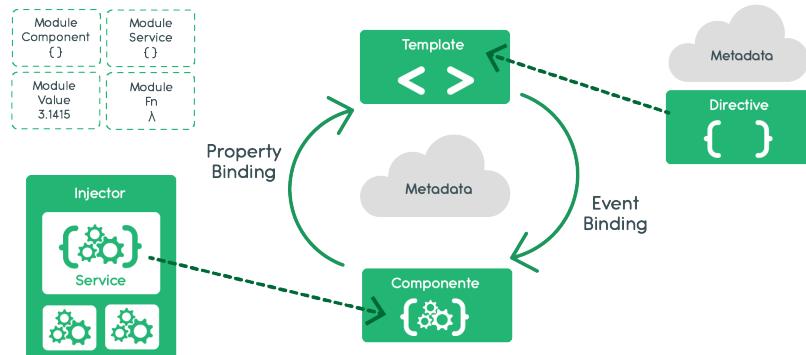
2.2 Angular

Segundo [Branas \(2014\)](#), o AngularJS foi criado por Miško Hevery e Adam Abrons no ano de 2009. O AngularJS é um *framework* do lado cliente de código aberto, no qual a estrutura tende a entregar uma alta produtividade e experiência no desenvolvimento web. No AngularJS utiliza-se a sintaxe HTML, estendendo seu vocabulário, que permite utilizar componentes, tornando mais fácil o trabalho dos desenvolvedores. O nome do framework criado por Adam Abrons, que teve como inspiração os colchetes angulares dos elementos HTML.

O resultado final disso tudo é a criação de aplicações mais expressivas, de fácil manutenção e que permita a reutilização de componentes, fazendo com que não haja códigos desnecessários e priorizando as coisas mais importantes para a equipe no processo de desenvolvimento. ([BRANAS, 2014](#))

Além da utilização de componentes, o AngularJS tem outros elementos básicos que facilitam o processo de desenvolvimento, como os módulos, diretivas, *templates*, roteamento serviços, injeção de dependências e ferramentas de infraestrutura que automatizam tarefas como demonstrado na Figura 1.

Figura 1 – Elementos básicos do Angular



Fonte: [Afonso \(2018\)](#)

Através da Figura 1 pode se observar que os elementos funcionam de forma simultânea comunicando entre si. Através desses elementos mostrados é possível criar as aplicações *Single-Page* de qualidade, além disso, o *framework* conta com uma comunidade grande, que facilita nos estudos e aperfeiçoamento das aplicações. ([AFONSO, 2018](#))

O AngularJS lançou a sua versão 2 em 2014, mudando completamente o *framework*, começando pelo nome que passou a ser apenas *Angular*. Essa mudança partiu da equipe do Angular que visava funcionalidades novas, no qual passou a ter compatibilidade com as novas versões do *TypeScript*. Como de fato a mudança de versão fez com que na realidade o AngularJS e o Angular se tornassem totalmente diferente, dessa forma as aplicações

feitas em AngularJS não são compatíveis com a nova versão, o que fez com que diversos profissionais tivessem que aprender a utilizar um novo *framework*. ([BOOTH, 2017](#))

2.2.1 Typescript

O *TypeScript* é uma linguagem de programação que foi criada pela *Microsoft*. O seu código é compilado para um código JavaScript mais limpo e simples que permite ser executado em qualquer navegador, no Node.js ou em qualquer mecanismo JavaScript que tenha suporte ao ECMAScript 3 (ou mais recente), isso é, ele faz exatamente a mesma coisa que o JavaScript e muito mais. ([MICROSOFT, 2014](#))

No processo de compilação, ao invés de criar um código de baixo nível, é criado um código em outra linguagem, ou na mesma linguagem, só que com o objetivo de deixar compatível com o ambiente que vai ser executado. ([AFONSO, 2018](#))

Segundo Fenton ([2017](#)), a *Microsoft* lançou o *TypeScript* em 2004, tendo como uma linguagem *open source*, que estava sob licença da Apache 2.0. A linguagem surgiu para resolver os problemas que o Javascript não podia resolver, como o uso de tipagem da linguagem, sendo seu principal recurso, além de ferramentas checagem estática, a possibilidade de criação de interfaces, entre outras coisas que ajudam no processo de desenvolvimento, como por exemplo, o recurso que permite descobrir erros durante o desenvolvimento. Dessa forma, o typescript contribui na melhoria e otimizando do tempo de desenvolvimento.

2.3 Node.js

Criado no ano de 2009 por Ryan Dahl, o Node.js é um ambiente de execução em Javascript que foi feita para rodar no lado Servidor. Desde o seu início, o Node.js tem se tornado popular e hoje já oferece um papel significativo no setor de desenvolvimento web. ([COPES, 2018](#))

O Node.js roda a partir de um mecanismo JavaScript V8, o núcleo do Google Chrome, fora do navegador. Com ele é capaz de rodar o Javascript em qualquer *browser* rapidamente, melhorando o desempenho da aplicação.

Uma aplicação feita com Node.js segue o princípio de uma *single-threaded* executado um único processo, sendo assim, independente da quantidade de solicitação, ele utiliza apenas uma *thread*. Diferente de outros servidores que utilizam um serviço *multi-thread*, o Node.js utiliza primitivas de E/S (entrada e saída) de forma assíncrona sem bloqueio. ([COPES, 2018](#))

O desenvolvimento com Node.js se baseia na arquitetura de desenvolvimento MVC. Tendo como o princípio que irá iniciar o desenvolvimento de uma aplicação com Node

utilizando o *Express*¹, então será criado uma estrutura que se seria dividida basicamente em *modules, middlewares, models, controllers, views, routes* e *config*.

2.4 MongoDB

De acordo com a [MongoDBInc \(2018\)](#), o mongoDB é um banco de dados baseado em documentos com a escalabilidade e flexibilidade para a realização de consultas e indexação. Já o [Pinto et al. \(2013\)](#) diz que, o mesmo que foi criado pela *MongoDB Inc*, armazena os dados em documentos que são passados em BSON (*Binary Object Notation*) que é uma codificação binária dos documentos JSON.

O mongoDB é um software de código aberto (*open-source*) que foi escrito em C++. O mesmo é classificado como uma banco de dados NoSql, em outras palavras, é uma banco não relacional que fornece as aplicações uma modelagem mais simplificada e natural. [Cuer \(2015\)](#) diz no seu artigo que o objetivo final dos criadores do Mongo é fechar a lacuna entre o rápido e o altamente escalável.

Segundo [Politowski e Maran \(2014\)](#), o mongoDB permite que os documentos sejam armazenados em coleções (*collections*), de forma em que serão realizados operações de busca (*queries*) que são objetos JSON que são enviados como BSON para o banco, e por fim efetuar a indexação (*indexing*).

Figura 2 – Modelo operacional do MongoDB



Fonte: ([MONGODBINC, 2018](#))

A Figura 2 foi ilustrada pela [MongoDBInc \(2018\)](#) para mostrar a forma inteligente que o banco de dados foi criado para operar. Nele pode observar que o Mongo foi criado para atender as necessidades e demandas de aplicações mais modernas, em que através da orientação a documentos, melhora a maneira de trabalhar os dados, tornando mais rápido, fácil, flexível e versátil. Além disso o mongoDB se baseia em um sistema distribuído inteligente, possuindo princípios como de disponibilidade e escalabilidade, sendo possível também ser executado em qualquer lugar. A seguir será abordada a arquitetura de desenvolvimento MVC, que é praticamente o padrão de desenvolvimentos de todas as tecnologias que até aqui foram citadas.

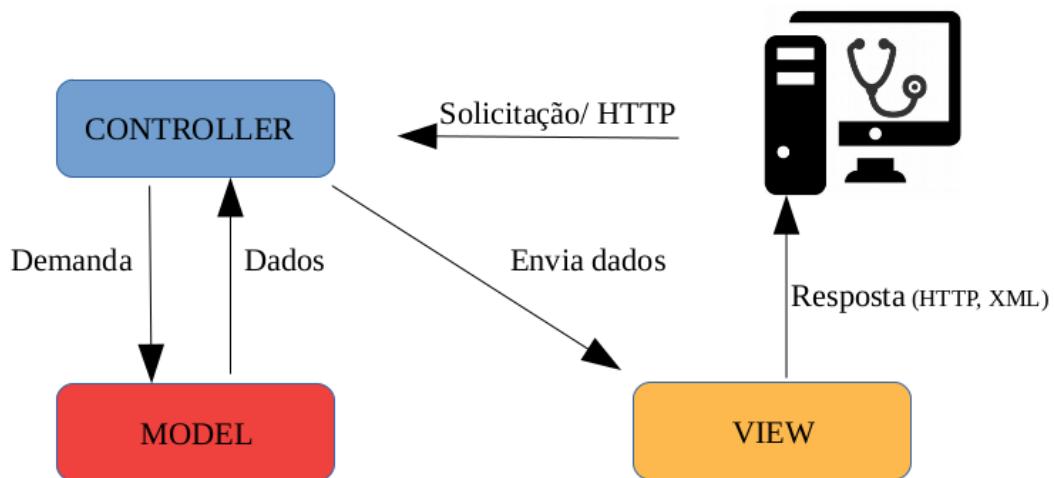
¹ Express é um framework da Web Node.js. ([NODEBR, 2016](#))

2.5 Padrão de arquitetura MVC

O modelo padrão de arquitetura de software MVC (*Model - View - Control*), é de forma geral um padrão que como o nome já diz, organiza a aplicação separando em três partes, sendo, *Model*/modelo(camada de manipulação dos dados), *View*/visão (camada de interação com usuário) e *Controller*/controle(camada de controle). Ele é utilizado na GUIs, que são as Interfaces gráficas de usuário, que consistem em arquitetar as aplicações web, ou também as aplicações mobile. (MACORATTI, 2008)

Pensando na reutilização do código e na organização e separação dos conceitos neles existentes é que o modelo MVC foi criado. A utilização desse modelo acontece da seguinte forma:

Figura 3 – Detalhamento do funcionamento do MVC



Fonte: Adaptado de (RAMOS, 2015)

Segundo Ramos (2015), conforme se verifica na Figura 3, se tem uma representação de um “diálogo entre as camadas”. O diálogo começa com a solicitação do usuário através da *View*, que é responsável por tratar as saídas para o usuário final, exibindo os dados solicitados, que consequentemente informa ao *Controller* a intenção do usuário, fazendo com que o *Controller* envie ao *Model* a demanda realizada. O *Model* vai então fazer a leitura, escrita ou a validação dos dados, dependendo da demanda, e então retornar os dados para o *Controller* que vai enviar os dados para a *View*, e então ser mostrado ao usuário.

Um ponto importante para o prosseguimento do trabalho consiste na análise de recursos de Interface Homem Computador(IHC), uma vez que a interação é parte fundamental do processo de desenvolvimento e utilização de Chatbots, sendo objeto de

análise a seguir apresentado.

2.6 Interface Homem Computador(IHC)

A IHC segundo Preece, Rogers e Sharp (2015), está associada ao design de sistemas computacionais que ajudam os usuários de maneira em que eles possam realizar suas atividades de forma produtiva e com segurança. Dumas e Redish (1999) diz que a IHC retrata a forma de projetar produtos que facilitem a interação entre pessoas e computadores. A IHC consiste em atender a questões como: “Quais os requisitos que estabelece se uma interface é boa?” ou “O que caracteriza uma interface boa de uma ruim ?”.

A IHC é um termo que tem sido utilizado desde os anos 80, no qual se apresentava como uma nova área de estudo que se tinha como preocupação todos os assuntos relacionados à interação entre usuários e computadores. Desde a década de 80 tentam achar uma definição final para a IHC, porém essa definição sempre vem retratando o momento vivido. A definição a seguir retrata bem o momento vivido na época (BAECKER, 2014):

“Conjunto de processos, diálogos e ações através das quais os usuários se baseiam para interagir com computadores”.

No momento atual, tem sido utilizado a seguinte definição realizado pela SIGCHI (HEWETT et al., 1992):

“IHC é a disciplina que se ocupa com o design, avaliação e implementação de sistemas computacionais interativos para uso humano e com o estudo dos fenômenos ao seu redor”.

A IHC se preocupa muito na Experiência do usuário, fazendo com que tudo quanto for realizado, seja realizado para a satisfação do usuário final. Estudos de métodos e ferramentas que serão explanados a seguir, traz exatamente um dos principais pontos da UX, que é a interface amigável ao usuário, no qual pode ser realizado através de inúmeras formas como a dos *Layouts Responsivos*.

2.6.1 Responsividade

Segundo o dicionário de língua portuguesa Aurélio, a responsividade é definida como uma “Capacidade de responder rapidamente e do modo mais adequado à situação em questão”, que de fato a definição também se aplica ao mundo da Tecnologia.

Uma breve definição sobre a Responsividade abordada por Rogatnev et al. (2015):

“Web design responsivo ou RWD é uma abordagem de criação de web design na qual um website é desenvolvido com a expectativa de fornecer um design fácil e fácil de usar, incluindo fácil visualização da página da web com um mínimo de redimensionamento e giros extras em uma ampla gama de dispositivos.”

Atualmente a navegação na internet tem aumentado bastante, da mesma forma o uso de dispositivos variados, principalmente os dispositivos móveis. Dispositivos móveis frequentemente são limitados pelo tamanho de suas telas e exigem uma abordagem diferente para o layout do conteúdo. Dessa forma surge a necessidade de otimizar a Web para esses dispositivos.

Figura 4 – Ilustração de um Layout Responsivo



Fonte: (ROGATNEV et al., 2015)

Hoje em dia existem celulares com infinidades de modelos de telas, temos tablets, notebooks, desktops, TVs , entre outros modelos. O fato é que a evolução é constante, e essa variedade vai continuar aumentando, pois isso é de fundamental importância a utilização da responsividade nesse projeto, tendo em vista a atualidade e também já visando o futuro.

A utilização da responsividade busca trazer uma maior praticidade, fazendo com que o usuário final tenha o serviço disponível, independente de seu dispositivo, fazendo com que o mesmo se adapte o layout da página, que as imagens sejam redimensionadas automaticamente para que caibam na tela, e também melhorar a visualização da página quando necessário, simplificando os elementos da tela e ocultando elementos desnecessários

nos dispositivos menores, e também adaptar para as possíveis mudanças na forma de visualização, seja ela na vertical ou horizontal.

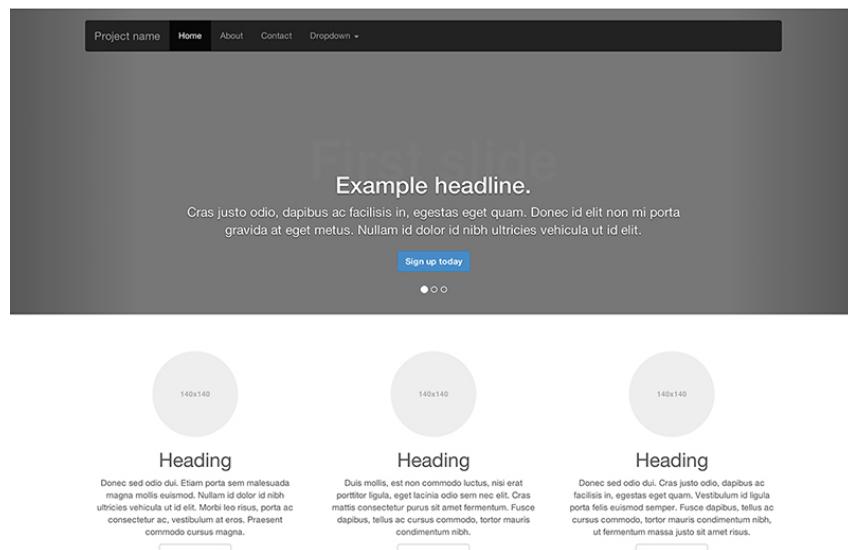
Para a criação de *layout* responsivos para a Web, atualmente existem diversos *frameworks* gratuitos e de código aberto, como será mostrado a seguir, que podem ser agregados facilmente as páginas Web, deixando-a mais atraente e adaptável a qualquer tela, cujo é um dos objetivos de estudo deste projeto.

2.6.2 Framework Bootstrap

Para a criação de sites com *layouts* responsivo, o Bootstrap é hoje um dos *frameworks* CSS mais utilizados para desenvolvimento de sites Web responsivos. O objetivo desse *framework* é facilitar o desenvolvimento web, criando sites responsivos, ou seja, com tecnologia mobile, fazendo com que se adapte a qualquer tipo de dispositivos, tudo isso com a vantagem de não ter necessidade de escrever sequer uma linha de um arquivo CSS.

O Bootstrap foi desenvolvido por *Jacob Thorton* e *Mark Otto*, que são engenheiros do *Twitter*, que inclusive disponibilizou seu *template* para desenvolvedores, na tentativa de resolver incompatibilidades dentro da sua própria equipe de trabalho ([BARBIERE, 2017](#)). *Thorton* e *Mark* aspiravam otimizar o desenvolvimento de sua plataforma fazendo uso de uma estrutura única, que reduziria as inconsistências entre as diferentes formas de desenvolver de cada desenvolvedor.

Figura 5 – Exemplo de página com Bootstrap



Fonte: <http://v4-alpha.getbootstrap.com/examples/screenshots/carousel.jpg>

Atendendo um dos principais pontos da UX, o Bootstrap contém uma interface bastante amigável, como mostra no exemplo acima (Figura 5) no qual oferece vários *plugins*,

componentes e temas compatíveis com o *framework* e pode ser utilizado em qualquer linguagem de programação, totalmente grátis e *Open Source*. Assim como ele, também tem surgido com forte força no mercado, o *Framework Materialize* da Google, da qual será referenciado a seguir.

2.6.3 Framework Materialize - Material Design

Criado e projetado pela Google, Material Design é uma linguagem de design que combina os princípios clássicos de projetos bem-sucedidos junto com inovação e tecnologia. O objetivo da Google é desenvolver um sistema de design que permite unificar a experiência do usuário em todos seus produtos em qualquer plataforma. ([MATERIALIZE, 2018](#)).

O Materialize foi projetado com o intuito de melhorar e tornar o desenvolvimento mais ágil se tratando de Material Design. Por conter componente especificamente já criados, o Material Design pode ser utilizado para inúmeras finalidades, uma delas é o fato de poder modelar a interface do usuário (UI), e consequentemente dando uma melhor experiência ao usuário(UX), no qual permite apenas fazer as devidas alterações e adaptações para a forma desejada.

Se tratando de melhor Experiencia do Usuário, o Materialize, assim como o Bootstrap, nos fornecem ferramentas e recursos que tem como objetivo configurar as características das páginas, fazendo com que sejam responsivas a qualquer dispositivo que está a disposição do usuário.

Este *framework* possui infinitas vantagens se tratando de agilidade e facilidade. Uma das vantagens iniciais é o fato de ser uma ferramenta de Código aberto, em que qualquer desenvolvedor pode utilizá-la. Outra vantagem é o fato de ser um *framework* muito completo, na qual pode ser usado sem a necessidade do uso de outro *frameworks*. Com ele podemos utilizar a variedade de recursos de JQuery disponíveis(*Modais, Slider, Transitions, Lightbox, Waves e Captions*).

2.7 Inteligencia Artificial

A respeito da definição de Inteligência Artificial (IA), Elaine Rich ([RICH et al., 1994](#)) fala que, embora a maioria das tentativas para definir com precisão termos complexos e de utilização ampla seja exercício de futilidade, é necessário delinear pelo menos uma fronteira aproximada em torno do conceito para que se tenha ideia sobre a discussão. Dessa forma, fica definido que a Inteligência Artificial é o nome dado para a capacidade em que as máquinas podem ter de pensar como os seres humanos.

Desde muito tempo a IA tem sido estudada. Na segunda guerra mundial, o matemático *Alan Turing* desenvolveu a sua primeira solução de IA, que seria a máquina

hipotética, que mais tarde ficou conhecida como a “Máquina de Turing”. Após a guerra, Turing lançou o artigo *”Computing Machinery and Intelligence”*. O professor universitário, McCarthy (1989), a define a IA como ”a ciência e engenharia de produzir máquinas inteligentes”.

A Inteligência Artificial é um ramo da computação que tem se empenhado em procurar meios que possam aumentar a capacidade de que o ser humano tem para resolução de problemas e de pensar. Seu objetivo central é realizar tarefas que os seres humanos de certa forma não realizam, de forma inteligente e eficaz. (TEIXEIRA, 2014)

Para ensinar a máquina a pensar e aprender com nossas interações, é necessário utilizar técnicas computacionais como, *Machine Learning* e Algoritmos Genéticos, e também, Redes Neurais Artificiais, Processamento de Linguagem Natural, no qual são objetos de estudo deste projeto e será aprofundado mais adiante nessa seção, assim como vários outros meios que irão auxiliar no desenvolvimento de um Sistema mais inteligente.

2.7.1 *Machine Learning* (Aprendizado de Maquina)

O Aprendizado de máquina tem sido uma das técnicas mais utilizadas dentro da IA. Isso tudo se dá pelo fato de que o Aprendizado de Máquina é a ciência tem feito com que máquinas aprendam e interajam como os seres humanos, tudo isso usando processos de treinamentos a fim de melhorar seu aprendizado ao longo do tempo de maneira, aumentando a sua base de conhecimentos para que seja mais confiável possível.

Em uma definição feita, Domingos (2012) da Universidade de Washington, diz que os Algoritmos de aprendizado de máquina podem descobrir como realizar tarefas importantes generalizando a partir de exemplos. Uma outra definição muito boa feita por Pyle e Jose (2015), diz que o aprendizado de máquina é baseado em algoritmos que podem aprender com dados sem depender de programação baseada em regras.

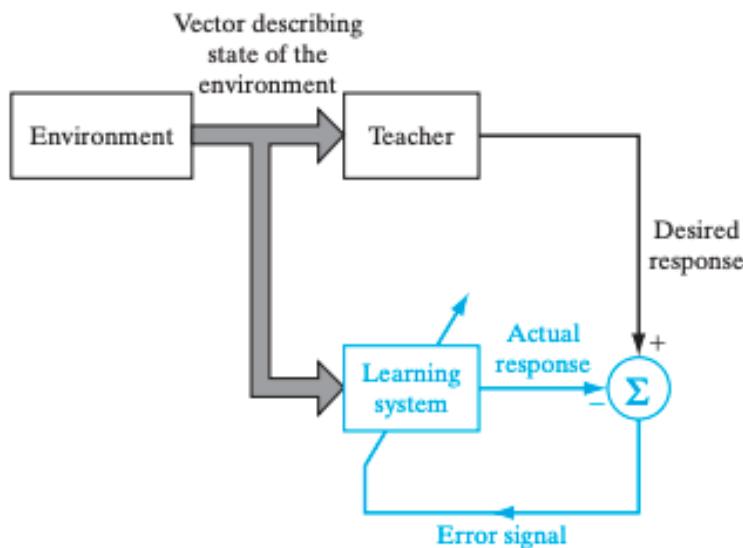
Ao compreender o objetivo do *Machine Learning*, percebe-se que o mesmo será de suma importância para a realização deste projeto de automação inteligente, desde o processo de aprendizado e treinamento, até a tomada de decisões, pois para resolver a problemática aqui colocada, deve-se oferecer o máximo de confiança nas informações fornecidas.

O aprendizado de máquina pode ser dividido em vários tipos, sendo os três principais o, aprendizado Supervisionado, o aprendizado por Reforço e o aprendizado Não Supervisionado. No aprendizado Supervisionado existe um ator que é chamado de professor(supervisor) que contem o conhecimento sobre o ambiente, fornecido uma série de dados na qual o algoritmo irá usar para aprender e gerar uma saída semelhante. O aprendizado por Reforço consiste tentativas e erros, na qual um ”agente” irá trabalhar na tomada de decisão com o objetivo de maximizar as recompensas.Já o Não Supervisionado não existe

um *feedback*, ou seja, não há supervisão, fazendo com que o algoritmo busque aprender de forma independente e automática.

O processo Supervisionado e por Reforço é utilizado por problemas de Regressão, e que se baseia em prever respostas em uma variável continua, ou seja, prever um valor real, e também existe os problemas de Classificação, no qual busca prever os dados em uma saída discreta, "0 ou 1". Já o processo Não Supervisionado é utilizado em problemas de associação ou agrupamento. Na Figura 6 mostra um exemplo de um diagrama de blocos de aprendizado Supervisionado:

Figura 6 – Diagrama de blocos do Aprendizagem Supervisionada



Fonte: Haykin et al. (2009)

Na Figura 6 mostra um diagrama de bloco que ilustra a aprendizagem supervisionada. Em termos conceituais, podemos pensar no professor como tendo conhecimento do ambiente, sendo esse conhecimento representado por um conjunto de exemplos de entrada-saída. O ambiente é, no entanto, desconhecido para a rede neural. Desse modo, com o objetivo de construir o conhecimento, o professor é capaz de fornecer à rede neural uma resposta desejada para esse vetor de treinamento. A resposta desejada representa a ação “ótima” a ser executada pela rede neural. Os parâmetros da rede são ajustados sob a influência combinada do vetor de treinamento e do sinal de erro. O sinal de erro é definido como a diferença entre a resposta desejada e a resposta real da rede. Este ajuste é realizado iterativamente em um passo-a-passo moda com o objetivo de eventualmente fazer a rede neural emular o professor. Dessa forma, o conhecimento do professor é transferido para a rede neural. (HAYKIN et al., 2009)

No exemplo acima vemos que o processo de aprendizado supervisionado constitui um sistema de feedback em malha fechada. Para melhorar o processo de aprendizado, pode

ser utilizado um outro ramo da IA, que é a RNA, no qual será abordado nessa próxima seção.

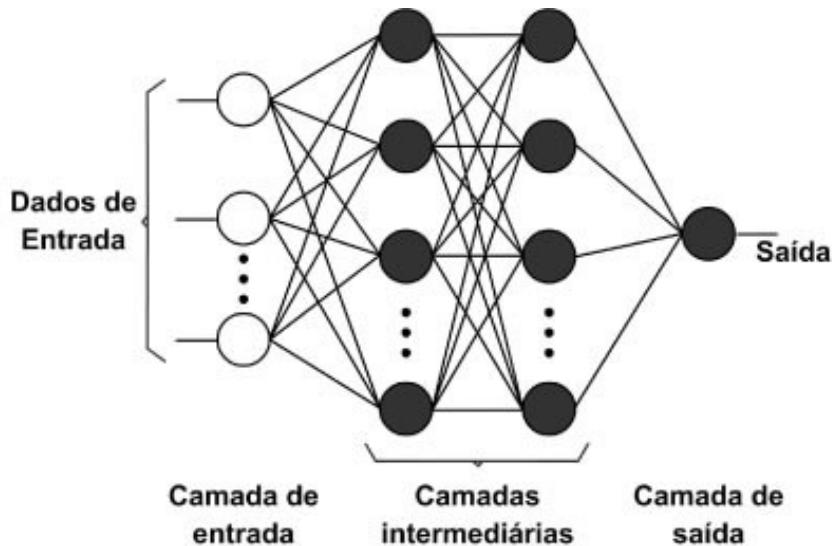
2.7.2 Redes Neurais Artificiais - RNA

As Redes Neurais Artificiais(RNA), são métodos que foram feitos para sanar alguns problemas da IA, fazendo com que ele fosse um sistema totalmente constituído por mecanismos que imitam o cérebro humano e sua capacidade cognitiva, agindo diretamente no seu comportamento. As RNA são técnicas utilizadas na computação que tem como características a presença de um paradigma que foi inspirado no sistema neural de seres ou corpos inteligentes e que adquirem conhecimento por meio de uma experiência. Em seu trabalho [Haykin et al. \(2009\)](#) fala que:

"Uma RNA é um sistema massivamente paralelo e distribuído, composto por unidades de processamento simples que possuem uma capacidade natural de armazenar e utilizar conhecimento."

As RNA são organizadas em camadas, com unidades que podem estar conectadas às unidades da camada posterior. Na Figura 7 temos um exemplo de uma RNA com três entradas, duas camadas intermediárias com quatro neurônios e uma camada de saída com um neurônio, produzindo uma única informação de saída.

Figura 7 – Representação de camadas RNA



Fonte: [Fiorin et al. \(2011\)](#)

Conforme a Figura 7, [Fiorin et al. \(2011\)](#) afirma que a camada de entrada é muitas vezes ignorada por diversos autores por que apenas recebem e distribuem dados, não realizando nenhuma operação matemática. As redes com uma camada intermediária podem representar qualquer função contínua enquanto as redes que possuem duas ou mais

camadas intermediárias podem, teoricamente, implementar qualquer função, linearmente separável ou não, sendo que a precisão dos resultados gerados pela RNA está relacionada com o número de nodos nas camadas intermediárias.

Em uma RNA mais robusta, pode haver uma enorme quantidade de unidades de processamento, cerca de centenas ou milhares, em contrapartida, um cérebro de um mamífero, por exemplo, pode conter bilhões de neurônios.

A seguir iremos abordar o estudo da técnica de Processamento de Linguagem Natural (NLP), que alinhado às RNAs, deve fortalecer o objetivo deste projeto.

2.7.3 Natural Language Processing (NLP)

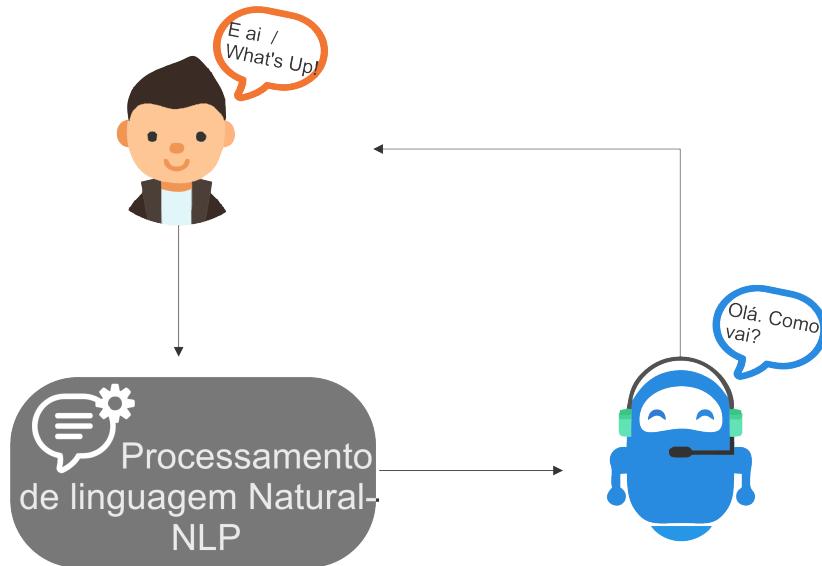
O NLP, que no português é Processamento de Linguagem Natural, se refere ao estudo que é feito a respeito das interações feitas entre os computadores e as linguagens naturais do ser humano. Com isso, é necessário ser implementado métodos da IA na comunicação com sistemas inteligentes usando uma linguagem natural como o português, ou qualquer outro idioma.

Embora o NLP pode ser também chamado de vários nomes como, análise de texto, linguagem computacional ou mineração de dados, o princípio inicial é mesmo. O processamento da Linguagem Natural é uma técnica que se torna necessária no momento em que se tem a necessidade de que um sistema inteligente, como um Chatbot no caso, funcione de acordo com as instruções pré-definidas no desenvolvimento.

Além das operações comuns de processamento de texto que trata o texto como uma mera sequência de símbolos, o NLP considera a estrutura hierárquica da linguagem: várias palavras formam uma frase, várias frases fazem uma sentença e, em última análise, sentenças transmitem ideias (REHLING, 2011).

Dentro da NLP, temos dois componentes que fazem parte do desenvolvimento do processamento da linguagem. Um deles é o “*Natural Language Understanding*” ou Entendimento da Linguagem Natural(NLU) , que trata a respeito dos diferentes aspectos e variações da linguagem e do mapeamento da entrada fornecida pelo remetente. O outro componente é o “*Natural Language Generation*” ou Geração de Linguagem Natural (NLG), que trata da produção dos textos e sentenças de forma natural.

Figura 8 – Exemplo básico do funcionamento do NLP



Fonte: do Autor, 2018.

De acordo com Lalek (2018), a fala humana geralmente tende a não ser tão precisa, as vezes é ambígua e a sua estrutura linguística pode mudar de acordo com as muitas variáveis complexas, incluindo gírias, dialetos regionais e contexto social. Deste modo, a Figura 8 retrata exatamente um dialogo ”informal” em que foi empregado o uso de gírias, contudo, o NLP foi capaz de processar a mensagem, permitindo que a máquina pudesse responder a requisição(mensagem) recebida.

2.7.4 Sistemas Especialistas

Ao se falar em Inteligência Artificial, na maioria dos casos já vem em mente o ato de se criar sistemas especialistas. Em um contexto geral, um especialista, que também pode ser chamado de perito, é a pessoa que se dedica ao estudo de uma determinada área de conhecimento.

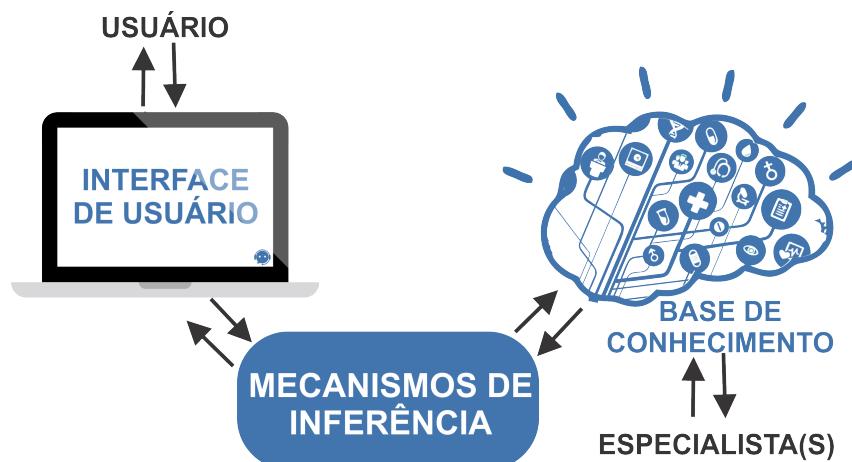
A especialidade no contexto de Sistemas não é diferente. Segundo FEIGENBAUM e Barr (1981) em seu livro “The Handbook of Intelligence”, fala que os Sistemas Especialistas são sistemas que solucionam problemas que são resolvidos apenas por pessoas especialistas (que acumularam conhecimento exigido) na resolução destes problemas. De modo geral, os SE são sistemas que carregam conhecimento provido de um especialista humano para sanar algum problema ou demanda que necessita da presença de um especialista.

O intuito do estudo sobre SE é buscar meios de resolver problemas mais relevantes, de maneira que possa ser solucionado de uma forma mais aproximada à utilizada pelos peritos humanos. Assim como é o caso da Medicina, no qual tem sido uma das áreas mais favorecidas pelos SE, isso justamente por ser dotada de problemas que são tipicamente usuais a esses tipo de sistemas, na qual é o objetivo dessa solução. Na medicina tem se

intensificado o estudo de SE com a capacidade de aprender sozinhos as associações lógicas, para ser utilizado no apoio à decisão Médica, no qual são desenvolvidos através das RNA.

De acordo com [WIDMAN \(1998\)](#), os Sistemas Especialistas podem ser utilizados no “Apoio à decisão” no qual o sistema deve ajudar o tomador de decisões a se lembrar de diversos tópicos ou opções, que se considera que ele saiba, mas que possa ter esquecido ou ignorado, sendo comum e muito utilizado na área da medicina. Também pode ser utilizado na “Tomada de decisão” no qual pode tomar a decisão no lugar de uma pessoa, pois isso implicaria algo que está acima de seu nível de treinamento e experiência.

Figura 9 – Funcionamento de um Sistema Especialista.



Fonte: do Autor, 2018.

Ainda segundo [WIDMAN \(1998\)](#), a Figura 9 retrata como funcionam os SE, sendo dividido da seguinte forma:

- Base de conhecimento: é onde fica todo o conhecimento especializado que será utilizado nas decisões, o qual pode ser estruturado e codificado de várias maneiras possíveis;
- Base de dados padrão: é onde contém o tratamento do vocabulário que será usado, como por exemplo na medicina, termos, frases, elementos de diagnóstico e tratamento, etc.
- Mecanismo de inferência: é um algoritmo capaz de criar conclusões a partir dos dados que o usuário fornece ao sistema, e do conhecimento armazenado em suas bases. Ele integra várias funções, como de receber dados, tanto do usuário quanto das bases de dados e de conhecimento, determinar metas de decisão e elaborar conclusões baseadas em alguma forma de raciocínio automático.
- Interface do usuário: Na interface o usuário irá realizar o diálogo com o sistema, em que pode ser feito o uso da linguagem natural para entender as frases ditas, como é o caso desse projeto.

A seguir será abordado o estudo sobre os Chatbots, que consiste em uma tecnologia de suporte a criação de Sistemas especialistas, que tem ganhado espaço no mercado atual, cujo é o principal objetivo de estudo para a construção dessa solução.

2.7.5 Chatbots

Os *Chatbots* (“conversa” Chat e “robô” Robot), são serviços de conversação que geralmente provém de Regras ou de Inteligência Artificial (IA), que de modo geral, provém de uma inteligência ou capacidade para a interação de forma mais natural possível. São considerados um tipo de SE que auxiliam na tomada de decisão e funcionam basicamente como um sistema que roda em torno de uma aplicação de mensagem ou *Web sites*, no qual é capaz de interpretar e enviar respostas de forma automáticas. ([GOMES, 2017](#))

Para compreender sobre os Chatbot, é importante conhecer a sua história e evolução. Embora se houve falar com frequência hoje em dia, o Chatbot não é um assunto da atualidade, ele já vem sendo discutido e implementado desde a década de 60. ([MILIOZZI, 2017](#))

O primeiro Chatbot foi desenvolvido por Joseph Weizenbaum entre 1964 e 1966, se chamava ELIZA, o primeiro simulador de diálogos,também chamados de ”robôs de conversação”, já desenvolvido com o uso de NLP cujo trouxe características muitos semelhante aos do humano. Segundo Weizenbaum, o Eliza é um programa que possibilita a conversação em linguagem natural com um computador. Sua implementação atual está no sistema de compartilhamento de tempo do MAC no MIT. Está escrito em MAD-Slip[4] para o IBM 7094. ([WEIZENBAUM, 1966](#))

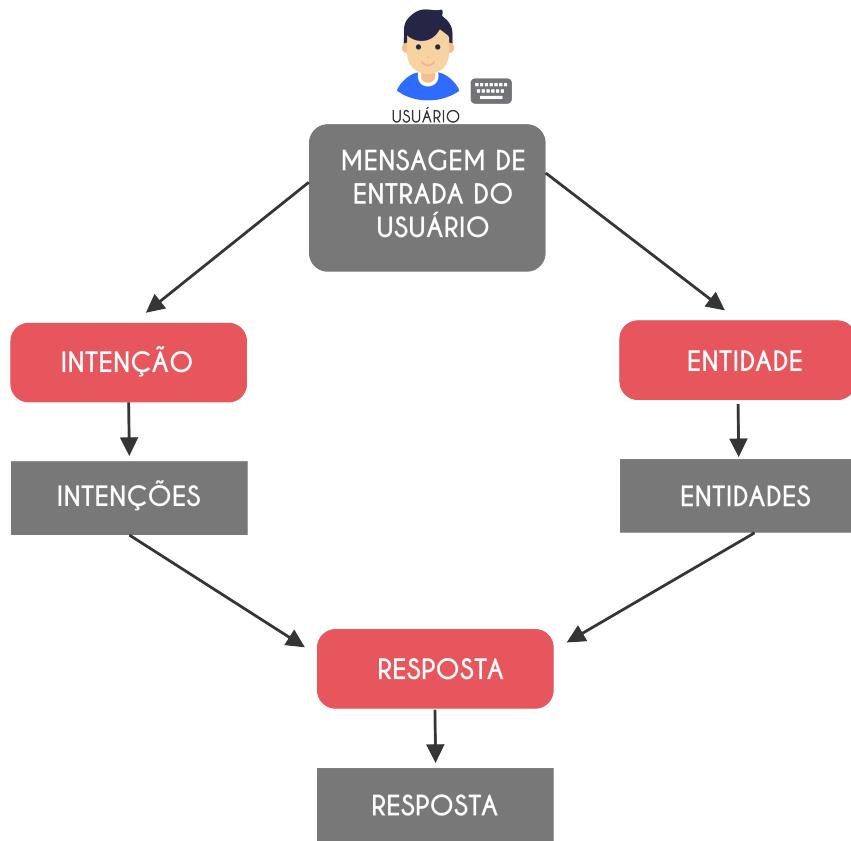
O tempo se passou, e pouco foi se falando sobre os Chatbot, porém na atualidade eles começaram a se destacar, principalmente com a evolução da Inteligência Artificial, e a capacidade cognitiva das máquinas atuais, que fazem com que esses tipos de softwares sejam os mais objetivos possíveis na resolução de problemas, fornecendo serviço prioritário a qualquer momento. Segundo o desenvolvedor e palestrante, [Carvalho \(2018\)](#), ele costuma chamar os Chatbots de um amigo que está ali para prestar um atendimento ou serviço com prioridade, na hora e sem espera.

Hoje existem Chatbots com capacidades enormes de conversação e resolução de problema. Como exemplo na área da saúde, temos o recém-chegado no Brasil, o GYANT, que é um chatbot responsável por analisar e identificar suspeitos do Zika vírus nos usuários. O funcionamento é padrão, baseado no protocolo padrão de identificação do Zika vírus, é feito uma série de perguntas e por fim é apresentado a avaliação feita com suas devidas recomendações.

Para entender o funcionamento do Chatbot, foi desenhado um diagrama (Figura 10), que traz os pontos fundamentais para serem entendido antes de desenvolver um

chatbot nos dias atuais:

Figura 10 – Funcionamento do Chatbot



Fonte: do Autor, 2018.

Com o recente aumento na popularidade do chatbot, é necessário pensar na interação entre o usuário e o bot. Para isso, deve se analisar as mensagens de entrada do usuário e também a resposta do chatbot. Dessa forma, foi criado a figura acima (Figura 10) que representa um modelo de funcionamento genérico de um chatbot. Nele pode se observar que a interação começa com a "mensagem de entrada do usuário" sobre o que o usuário está falando. Em seguida, a mensagem de entrada do usuário será processada através de dois módulos de classificação de intenção e reconhecimento de entidade que está em vermelho. O módulo de classificação de intenção verifica a mensagem de entrada do usuário e identifica a mensagem do usuário final.

Nos nós abaixo, representados na cor cinza, mostrar o funcionamento do Chatbot na qual, com base no número de intenções e no contexto da mensagem de entrada, identifica os intentos. O módulo de reconhecimento de entidade reconhece a mensagem do usuário estruturada e extrai a palavra-chave principal dos bits de informação, por exemplo, o chatbot desse sistema proposto pode extrair os sintomas(local) e a datas. Ambos os módulos de reconhecimento de entidade e reconhecimento de entidade são muito importantes para descobrir as intenções e entidades ao longo da interação entre o usuário e um bot. Próximo módulo importante é gerador de resposta do chatbot, ele pode utilizar algumas APIs e algoritmos externos para gerar a resposta, representado em vermelho. Por fim o gerador

de resposta usa sua intenção e entidades, bem como o contexto da conversação, extraído a última mensagem do usuário.

Para o poder desenvolver esses Chatbot, como dito anteriormente, existem diversas APIs e plataformas que auxiliam nessa tarefa. A seguir será abordada algumas dessas ferramentas que futuramente serão selecionadas para a implantação do modelo aqui criado.

2.8 APIs e Plataformas para Criação de Chatbots

Nessa seção será realizado o estudo de algumas ferramentas que possam auxiliar nesse processo de desenvolvimento do Chatbot aqui proposto.

2.8.1 API IBM Watson:

O Watson é uma um sistema para programação cognitiva, nele existe a possibilidade de se criar soluções inteligentes para qualquer funcionalidade. As APIs do Watson estão disponíveis para o qualquer desenvolvedor que queira criar uma solução pessoal ou para sua empresa, seja ela para interpretar texto, imagens análises de sentimentos e emoções e muitos outros. ([WATSON, 2018](#))

A IBM tem uma plataforma Cloud chamada de IBM Cloud, o antigo Bluemix, no qual é permitido ter acesso a todas as APIs do Watson. Um exemplo é o *Watson Assistant*, que anteriormente era chamado de *Watson Conversation*, no qual, segundo a IBM é permitido criar Chatbots com a utilização da NLP em conversas semelhantes às humanas - em vários idiomas. ([WATSON, 2018](#))

O *Watson Assistant* utiliza o aprendizado de máquina para responder à entrada de linguagem natural em plataformas como dispositivos móveis, sites, robôs e aplicativos de mensagens.

No IBM Cloud, existem APIs que estão liberadas gratuitamente com limitações, e por plano, com pagamentos mensais ou anuais. O *Watson Assistant*, que é o objeto de estudo deste trabalho, tem limite de 10.000 chamadas de API gratuitas.

2.8.1.1 Vantagens e Desvantagens do Watson Assistant

O *Watson Assistant*, antigo *Conversation*, da IBM, conta com uma série de vantagens que agrada para o desenvolvimento de Bots inteligentes. Primeiramente, cabe listar que o desenvolvimento com o *Watson Assistant* é bem intuitiva, no qual ao descrever as intenções e entidades, fica fácil criar o fluxo de diálogo, fluxo este que facilita bastante a visualização da conversa.

O *Watson* conta com o reconhecimento de fala automatizado, usando do Processamento de linguagem natural. Ele também se enquadra como uma aplicação Escalável, ou

seja, ele funciona bem mesmo quando seu contexto é alterado, com o objetivo de atender à necessidade de um usuário, de forma mais confiável, usando-se do aprendizado de máquina e lógica *fuzzy*, é Multilíngue, e possui tradutor para vários idiomas.

O Watson também possui desvantagens, ao começar pelas formas de integrações e até então ainda estão limitadas em comparação aos concorrentes, porém tem passado por evoluções e tem melhorado consideravelmente. Porém, a maior desvantagem do Watson é o custo, que consta com seu plano Lite com limitações, porém fica atrás neste quesito de outros concorrentes que são gratuitos. Uma outra desvantagem, se é que pode ser considerada uma, é que o Watson se enquadra como uma tecnologia disruptiva ².

2.8.2 Dialogflow (API.AI)

Assim como o *Watson Assistant*, o *Dialogflow* é uma plataforma de criação de bots, especialidade nos Chatbots baseados em intenções(*Intents*). Ele desempenha um papel de desenvolvedor de interação humano-computador baseado em conversas em linguagem natural(NLP), no qual permite que os desenvolvedores forneçam aos usuários, maneiras inovadoras de interagir com suas interfaces por meio de trocas de voz e texto alimentadas por inteligência artificial.(DIALOGFLOW, 2018)

Em setembro de 2016, a Google comprou a Api.ai, que então mudou a nomenclatura para “Dialogflow”. É uma plataforma web que pode ser acessada por qualquer computador e em qualquer lugar, assim como o Watson Assistant por meio do Bluemix.

Ao acessar sua plataforma Web, o API.ai reformulado nos apresenta agentes pré-criados que ajuda aos novos desenvolvedores começarem a criar seus aplicativos mais rapidamente. Além disso, ele possui um editor de código em linha para que os codificadores possam executar várias tarefas diretamente do console.

Através do Dialogflow, o desenvolvedor pode ter acesso a tecnologias integradas de aprendizado de máquina e processamento de linguagem natural. Com isso, o desenvolvimento se torna mais ágil, pois permite que eles se concentrem em outras partes integrantes da criação de aplicativos, em vez de delinear regras gramaticais detalhadas.

O Dialogflow, concede permissão aos desenvolvedores para criar experiências de conversação, permitindo utilizar dois tipos diferentes de tarefas, compreender e gerar enunciados com linguagem natural (NLP) e gerir a conversa. (JANARTHANAM, 2018)

2.8.2.1 Vantagens e Desvantagens do Dialogflow

Uma das vantagens do Dialogflow é a codificação rápida, no qual não precisa levar muito tempo para poder criar um chatbot. Ele consta com tecnologia de Aprendizado

² Tecnologias que causam uma interrupção nos padrões ou tecnologias já existentes no mercado. (BOWER; CHRISTENSEN, 1995)

de Máquina Google. O mesmo também já trás mais de 30 modelos pré-construídos, isso facilita para os desenvolvedores começarem os seus projetos.

O Dialogflow também é baseado em NLP, e ainda é multilingue, o que também ajuda no desenvolvimento de aplicações para o mercado. Porém, ele têm várias desvantagens como, gerenciamento de falhas deficiente, falta de otimização de diálogo, nenhuma maneira de influenciar o conhecimento especializado e menor precisão, que nesse caso podem dificultar a experiência do usuário.

2.8.3 API WIT.AI

O Wit.ai é uma API que também foi desenvolvida com a finalidade de melhorar o desenvolvimento de soluções inteligentes, contando com uma plataforma aberta e extensível. Nela pode ser desenvolvida aplicações mobile, Chatbot, Robôs, soluções para automação residencial entre outros.

Segundo o Wit.ai, a API aprende a linguagem humana de todas as interações e aproveita a comunidade, em que o que é aprendido, é compartilhado entre os desenvolvedores. ([WIT.AI, 2018](#)) Com o Wit.ai pode se criar *bots* que interagem com os humanos através de uma plataforma de mensagem. Nela, assim como o *Watson Assistant*, também pode se criar soluções para os quais você pode conversar ou enviar mensagens de texto. A IA envolvida permite que os *bots* continuem aprendendo com os usuário a medida que eles tenha um diálogo.

2.8.3.1 Vantagens e Desvantagens do Wit.ai

Assim como as APIs que anteriosmente foram citadas, o Wit.ai também tem uma lista de vantagens e desvantagens. Segundo ([KANG, 2017](#)), uma das principais vantagens do Wit.ai é que o mesmo seja fácil para desenvolver. Além disso, o mesmo apresenta formas que permitem gerenciar parâmetros de contexto por meio da interface do usuário. No Wit.ai também é possível adicionar facilmente entidades, destacando-as nos enunciados por meio da interface do usuário. Assim como no Watson, o Wit.ai também possibilita visualizar e editar a árvore de fluxo de conversação diretamente no interface.

As desvantagens do Wit.ai começa pelos exemplos documentados pela empresa, no qual só invocam a lógica no modo interativo localmente ou só funcionam com os aplicativos do *Facebook Messenger*. Além disso, o Wit.ai não consegue trabalhar com o recurso de *slot*, ou seja, para resolver isso, seria necessário buscar informações ausentes que não foram faladas pelo usuário. Além disso, a integração como outras aplicações, *Webhooks*³, é bem limitada e complexa.

³ Um método de ampliar ou modificar o comportamento de uma página da Web, ou aplicação da Web.

2.8.4 Chatfuel

O Chatfuel é uma plataforma que surgiu no ano de 2015 já com o objetivo específico que era tornar a construção de *bots* mais fácil e intuitivo. De acordo com o Site oficial da empresa, o Chatfuel começou sendo implantado no *Telegram* e foi rapidamente crescendo para várias aplicações, e consequentemente, para milhões de usuários.(CHATFUEL, 2018)

Atualmente eles vêm se dedicando principalmente na criação de *chatbots* para o *Facebook Messenger*, mais também para outras inúmeras aplicações como *Slack*, *web sites*, que é o foco desse projeto. O Chatfuel além de fácil, é gratuito e de código aberto (*Open Source*), o que faz com que o desenvolvimento fique ainda mais agradável, pois existe uma força muito grande por parte da comunidade.

A escolha do Chatfuel para o acervo de ferramentas presentes nesse projeto, se deu pelo motivo principal que é sua facilidade de construção e agregação com as API existentes. Porém, existem outras plataformas de desenvolvimento, como o Botpress, que é a próxima plataforma estudada.

2.8.5 Botpress

Assim como o Chatfuel que foi anteriormente citado, o Botpress também é uma plataforma de desenvolvimento de chatbots. De acordo com a empresa fundadora (BOTPRESS, 2018) é uma ferramenta de nível profissional, que tem como objetivo, desenvolver, implantar, gerenciar e dimensionar os bots. Uma plataforma simples e intuitiva, que além de tudo, é também de código aberto, e muito bem documentado.

No Botpress é colocado como meta, manter um padrão de qualidade da ferramenta, pois eles acreditam que isso influencia diretamente na qualidade do chatbot. A principal diferença entre o Botpress e as outras plataformas de desenvolvimento de *chatbots* está na sua arquitetura e sua filosofia, no qual é o único *framework* que tem a arquitetura modular, ou seja, qualquer desenvolvedor pode adicionar alguma funcionalidade ao *framework*, fazendo com que eles possam oferecer rapidamente uma ampla gama de recurso e funcionalidade na prateleira.

2.8.6 Watson Visual Recognition

Com o objetivo de aprimorar, dando mais funcionalidades ao chatbot, foi realizado um estudo de possibilidade de realizar um assistente capaz de reconhecer imagens e então realizar um diagnóstico. A ferramenta estudada é o *Watson Visual Recognition*.

O *Visual Recognition* é mais um dos serviços do motor de inteligência Artificial da IBM, o Watson. O autor Soni (2018) diz que o *Visual Recognition* é o ramo da computação visual que é especialista na identificação de conteúdo de uma imagem. Ele nos auxilia ao

encontrar significado para o conteúdo visual.

Com o VR, fica possível desenvolver aplicações mais inteligente que façam análise de um conteúdo visual, seja ele imagem ou vídeo. Vários exemplos práticos em relação ao uso da análise visual, além das identificações de objetos e de pessoas, podem serem citados como os exemplos das lojas virtuais que utilizam assistentes para ajudar na procura de um produto, no qual o cliente envia uma imagem de algo que ele procura e recebe uma resposta, também tem os exemplos de verificação de conteúdos inapropriados em mídias sociais. (SONI, 2018)

O serviço da IBM *Watson, Visual Recognition*, utiliza algoritmos de aprendizado profundos para identificar cenas, objetos e faces nas imagens que são enviadas para a API. Com ele pode-se criar e treinar uma API de classificação de imagem própria e personalizado que visam atender uma necessidade específica e especializada em algum assunto. (IBM, 2019)

A IBM também disponibiliza o VR com plano *Lite*(gratuito) que permite 1000 requisições de API por mês. A utilização do VR é bem simples, o processo de treinamento é um pouco demorado por se tratar de imagens, porém bastante eficaz.

Figura 11 – Exemplo de classificação de imagens no VR

Results



Fonte: [Digital \(2016\)](#).

Na Figura 11 é possível ver a forma que o VR opera. Ao realizar o envio de uma imagem para a API, a IA realizar a classificação, pontuando o grau de proximidade em porcentagem. Essa saída é chamada de *score*, no qual é enviado no formato de um JSON. Dessa forma é possível utilizar um *backend* para poder personalizar e destrinchar a essa saída que vem da API. Uma forma de trabalhar e conectar esses dados é utilizando o *Node-RED*, disponibilizado pela própria IBM.

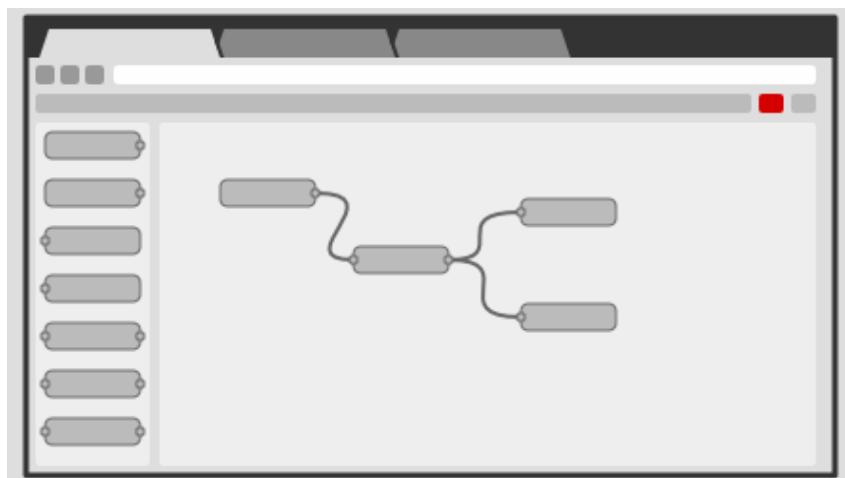
2.8.6.1 Node-RED

O Node-RED foi desenvolvido pela IBM *Emerging Technology*. Segundo o [Node-RED \(2019\)](#), a ferramenta de desenvolvimento, *Node-RED*, foi criada para poder realizar programações em fluxo para aplicações com Internet das coisas(IoT), porém, a ferramenta foi posteriormente estendida para a utilização em hardwares, APIs e *web services*, de maneira inovadora e interessante.

Dentre as várias coisas que podem ser feitas no Node-RED, uma delas é ler arquivos CSV, ouvir requisições *http*, *tcp*, *websocket*, *twitter*, *mqtt* e vários outros. O mesmo é uma ferramenta de código aberto, no qual conta com uma grande comunidade que colabora para o desenvolvimento de soluções. ([BASÍLIO, 2018](#))

Com ele também é possível fazer edição de fluxos e códigos através do navegador, na qual faz com que tudo seja mais rápido e prático, permitindo que fluxos sejam implementados em um único clique. Para edição de código utiliza-se um editor de *rich text*, no qual permite que funções *JavaScript* possam serem criadas. ([NODE-RED, 2019](#))

Figura 12 – Figura demonstrativa do funcionamento do Node-RED



Fonte: [Node-RED \(2019\)](#).

A imagem acima ilustra a forma em que, visualmente o Node-RED funciona. No mesmo contém vários tipos de nós(*nodes*)que podem ser usado basicamente sendo apenas arrastados para o diagrama de fluxo da aplicação (*flow*). Ao selecionar os nós, deve ser feita a configuração dos mesmo, conectando-os a outros nós. Como a ferramenta é *open source*, além dos *nodes* já existentes na ferramenta, pode-se adicionar outros tipos de nós que a comunidade disponibiliza para utilização. Após a criação dos fluxos e das funções, basta apenas fazer um *deploy* para poder salvar e executar o serviço. ([BASÍLIO, 2018](#))

2.9 Solução correlata

Nessa seção será apresentada uma solução correlata para o processo de desenvolvimento do projeto.

2.9.1 ApiMedic

O *ApiMedic* é um verificador de sintomas que são apresentadas pelo mundo. Ele nos oferece a verificação de sintomas médicos se baseado nos sintomas em que são descritos pelo próprio paciente. Dessa forma, ele faz uma triagem, e nos dá um pré-diagnóstico com informações médicas e mostra que médico deve ser recorrido para receber tratamento. O verificador de sintomas pode ser integrado através da API flexível (*Application Programming Interface*). Esta é uma interface de programação modular, que oferece as funcionalidades do verificador de sintomas para um programa principal. ([APIMEDIC, 2018](#))

3 Metodologia

Nesta seção trará de forma detalhada e sequencial, a Metodologia utilizada neste trabalho, desde a estruturação das atividades a serem desenvolvidas, até a realização das mesmas com o intuito de atingir o objetivo final deste projeto.

Inicialmente é importante saber que a Metodologia pode ser compreendida por várias modalidades de pesquisa. Segundo [Severino \(2017\)](#), várias são as modalidades de pesquisa que se podem praticar o que implica coerência epistemológica, metodológica e tecnológica, para seu adequado desenvolvimento.

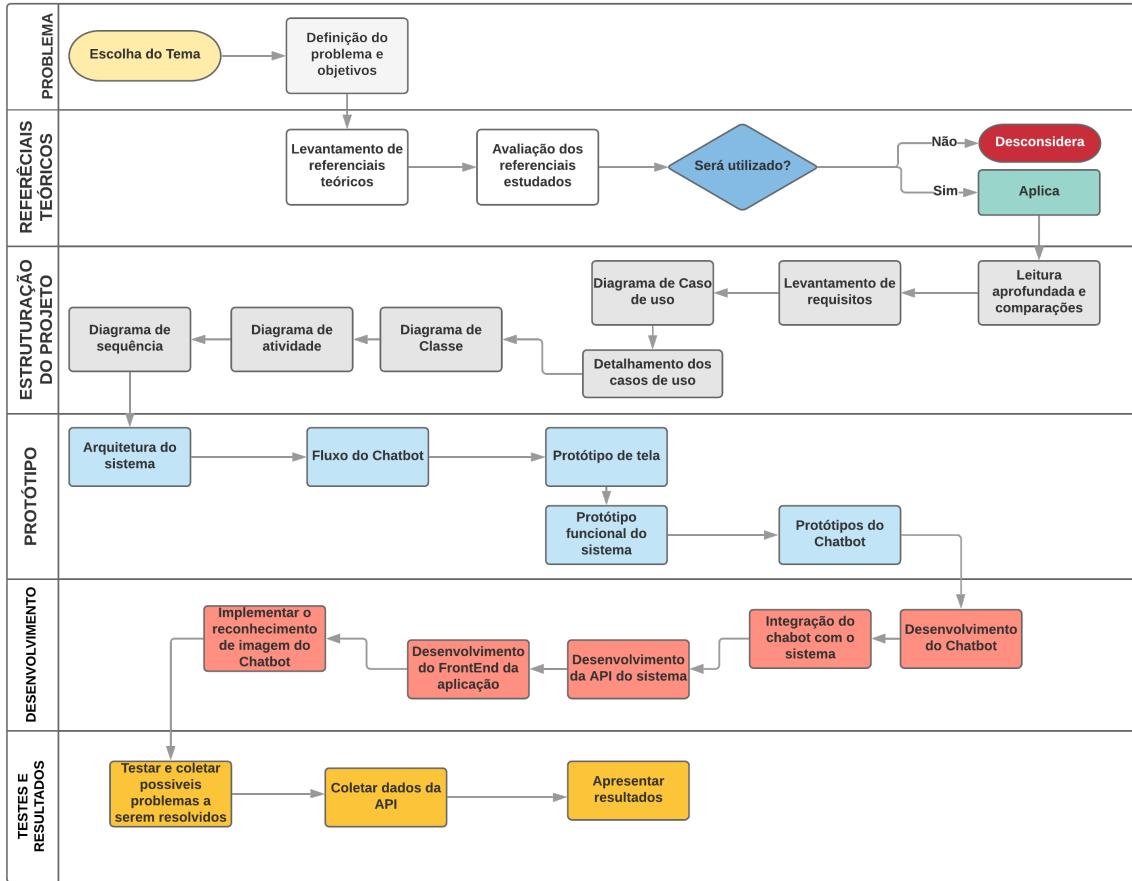
Desta forma, é fundamental que ao realizar um projeto seja feita uma pesquisa bibliográfica. A autora [Fonseca \(2002\)](#) descreve essa metodologia de pesquisa da seguinte forma:

”A pesquisa bibliográfica é feita a partir do levantamento de referências teóricas já analisadas, e publicadas por meios escritos e eletrônicos, como livros, artigos científicos, páginas de web sites. Qualquer trabalho científico inicia-se com uma pesquisa bibliográfica, que permite ao pesquisador conhecer o que já se estudou sobre o assunto. Existem porém pesquisas científicas que se baseiam unicamente na pesquisa bibliográfica, procurando referências teóricas publicadas com o objetivo de recolher informações ou conhecimentos prévios sobre o problema a respeito do qual se procura a resposta.”

Assim, parece ser mais apropriada a metodologia quando os dados e sua interpretação derivam de fontes documentais de eventos passados. Dessa forma foi traçado um plano de metodológico, no qual envolve uma pesquisa bibliográfica com efeito exploratório, pois será necessário uma familiarização com o assunto ainda pouco conhecido, pouco explorado pelo autor. ([SANTOS, 2014](#))

Para entender melhor a metodologia utilizada neste projeto, foi criado um diagrama de fluxo, assim como mostra na Figura 13.

Figura 13 – Metodologia do projeto



Fonte: do Autor, 2018.

No presente trabalho foi realizado um levantamento das características da realidade do público alvo, no qual foi descoberto as visões e preferências das mesmas, facilitando para uma tomada de decisão, visando minimizar os problemas que até aqui nos trouxeram. Dessa forma, foi descrito os procedimentos que foram realizados para sanar esse problema, tendo como motivação, descobrir de qual forma esses procedimentos irão influenciar na vida dessas pessoas.

Para dar início ao desenvolvimento dessa solução, foi escolhida a linguagem que seria utilizada para a implementação, feito pesquisas sobre a utilização e arquitetura para desenvolvimento do projeto, e então, pesquisar sobre Responsividade, para então tornar essa solução adaptável a qualquer tela, sendo projetado para uma melhor experiência do usuário.

Após o estudo da forma de implementação do sistema, foram realizados mais estudos sobre a Inteligência Artificial e suas técnicas de desenvolvimento, para uma futura implementação em um *Chatbot*. Ao realizar esse estudo, foi necessário pesquisar sobre técnicas para que no *Chatbot* tenha maior interação com o humano, como o Processamento de Linguagem Natural(NLP).

Após a fase anterior, foi necessário participar de palestra e discussões a respeito do desenvolvimento do *Chatbot* e a revolução que eles tem gerado desde o princípio, mais principalmente nos dias atuais. Então, foi adquirido uma carga maior de informações e foi esclarecido a aplicação das técnicas de IA sobre o *Chatbot* e o desenvolvimento do mesmo. Desta forma, foi necessário buscar mais a respeito das APIs e plataformas existentes que contribuem para o desenvolvimento de *Chatbot*, pois serão fundamentais para o desenvolvimento desta solução, e por fim criar um comparativo entre eles, para que seja tomada a decisão de qual a melhor ferramenta a se utilizar.

Ao concluir a fase de estudo dos referenciais teóricos , foi dado sequência a criação desse projeto, realizando um levantamento dos principais requisitos funcionais e não funcionais para este sistema, no qual detalha todas as funcionalidades e como serão desenvolvidas.

O próximo passo foi criar a estrutura do projeto, no qual foram desenhados os diagramas como o de caso de uso e diagramas de classe. Nestes foram ilustrados e detalhados como são realizadas a interação de cada ator e como as classes estão organizadas, além de ser possível também a descrição dos diagramas de componentes, atividade e sequência (Apêndices), com objetivo de mostrar o funcionamento de cada componente e seu fluxo.

Após estruturar o projeto, foi desenhada a arquitetura do projeto, de forma a representar como os atores e as tecnologias e aqui estudadas irão trabalhar. Depois de construir a arquitetura do projeto foi necessário criar ainda os protótipos do sistema, sendo desenhada a tela principal do site do sistema, para então implementar.

Após efetuar o passo anterior, foi necessário criar uma interação através de fluxos, para exemplificar o funcionamento do *chatbot*. Além disso, foi necessário implementar a tela principal da aplicação que seria o *site* para integrar o *chatbot* que aqui será desenvolvido. Nesse momento já ficou claro a escolha do Angular e Node.js para desenvolver a aplicação web por, além de serem linguagens bastante usadas na atualidade, tem uma grande comunidade de desenvolvedores, que facilita na resolução de problemas no processo de desenvolvimento. Como banco de dados, foi utilizando o MongoDB, que é um banco de dados não relacional(NoSql), na qual é utilizado por grande empresas do mercado, e o principal motivo da escolha desse banco é a Escalabilidade, performance e alta disponibilidade, escalando de implementações simples para grandes e complexas. Em relação ao assistente virtual (*chatbot*), ficou decidido a utilização do *Watson Assistant*, juntamente com o *Visual Recognition* para realizar diagnósticos por imagem, usando também o NodeRED para integrar esses dois serviços.

Depois de estruturar o projeto foi iniciada a fase de desenvolvimento da aplicação, começando pelo *chatbot*, sendo o objeto central e o elemento diferencial desse projeto. Para tal realização, foi necessário tomar conhecimento sobre a realização de diagnósticos. Para isso, foi necessário seguir protocolos padrões de diagnósticos médicos, e para tal ficou

evidenciado a necessidade de focar um protocolo previamente existente. Além de estudar os protocolos, buscou-se ajuda de profissionais da área da saúde para orientação nesse processo.

Após o desenvolvimento do *chatbot*, foi realizado a integração com o sistema. Após concluir, foi dada continuidade com o desenvolvimento da aplicação web, principalmente nas atividades ligadas a aspectos de gestão. Inicialmente foi estruturado a API, que seria o *Back-End* da aplicação. O mesmo foi desenvolvido a partir das regras definidas no caso de uso da aplicação. Ao finalizar, foi realizado os testes com a API através de softwares de testes, como o *Postman* no caso.

O próximo passo foi estruturar o *front-end*, criando os componentes necessários e organizando de forma que possa ser usual ao paciente. Ainda, do lado "cliente" da aplicação, foram criadas as chamadas Http através dos *endpoints* criados anteriormente.

Para melhorar a viabilidade do chatbot, foi necessário implementar o processamento de imagem, com o objetivo de realizar diagnósticos por imagens. Após a implementação e o treinamento da IA para processamento de imagem, foi realizando a integração com o chatbot, e disponibilizando-o também no Facebook, para que mais usuários pudessem alcançá-lo.

Por fim, a aplicação foi submetida a testes e correções, que visam a melhoria do sistema e do chatbot, afim de obter melhores resultados, na qual serão apresentados a seguir.

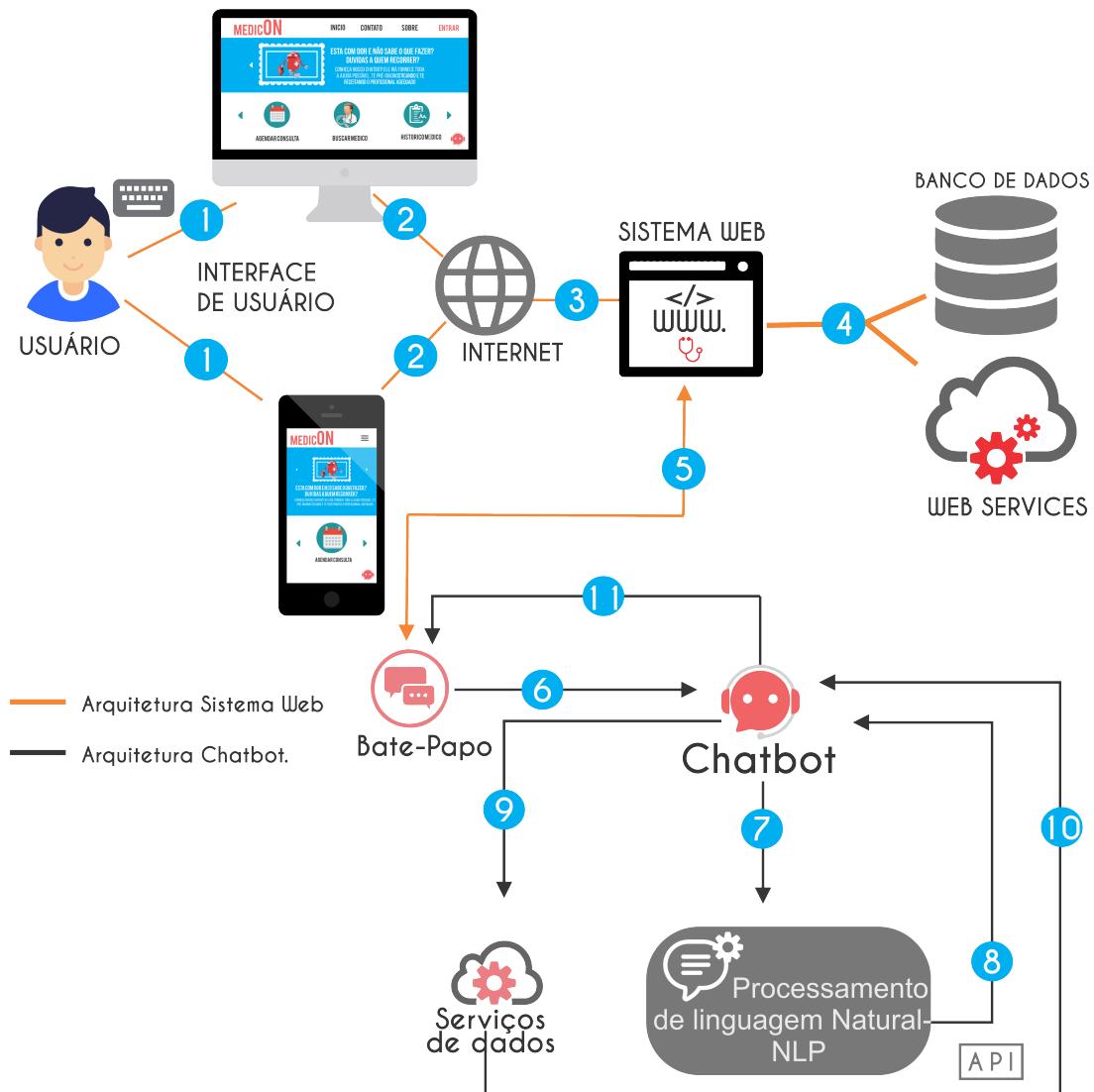
4 Resultados

Nesta seção serão apresentados os resultados obtidos no decorrer do estudo e estruturação desse projeto.

4.1 Arquitetura geral do Sistema

Depois de analisar e realizar o estudo das ferramentas que aqui foram mostradas, e que futuramente será feito a seleção eletiva das que melhor se adequaram para essa solução, próximo passo foi elaborar a arquitetura da solução aqui proposta. Dessa forma, a proposta inicial desse projeto está representada conforme na figura abaixo:

Figura 14 – Arquitetura da solução proposta



Fonte: do Autor, 2018.

Com base no modelo que foi desenhado (Figura 14), temos como sequenciamento os seguintes pontos:

1. Inicialmente, na arquitetura do sistema web, um Usuário deverá realizar uma entrada no sistema através da Interface de Usuário, em que terá acesso às funcionalidades do sistema;
2. A internet é responsável pelo transporte da solicitação do usuário ao sistema;
3. O Sistema Web receberá as solicitações feitas na entrada;
4. O banco de dados fica responsável por tratar os dados e informações que serão pertinentes neste modelo. Durante alguma operação no sistema pode ser consumido algum Web service;
5. Nesse ponto, supõe-se que o usuário tenha solicitado o auxílio do pré-diagnóstico do Chatbot;
6. Na arquitetura do Chatbot, o usuário abre bate-papo e digita uma frase ao bot;
7. O chatbot envia a frase a um motor NLP de aprendizagem de máquina;
8. O mecanismo de NLP extrai a intenção e as entidades dos usuários da frase fornecida e envia de volta ao Chatbot, ressaltando a presença ou não de uma API;
9. Nos serviços de dados, a intenção é usada para chamar o serviço adequado, usando informações da entidade para localizar dados apropriados;
10. Logo após, os dados são retornados para o Chatbot;
11. Ao finalizar, o Chatbot agrupa os dados na resposta adequada para exibição pela interface de bate-papo.

Para a efetivação deste modelo arquitetado, foi necessário estruturar o projeto, de forma que teve que ser criado Diagramas, como de caso de uso e de classe, assim como veremos na seção 4.2.

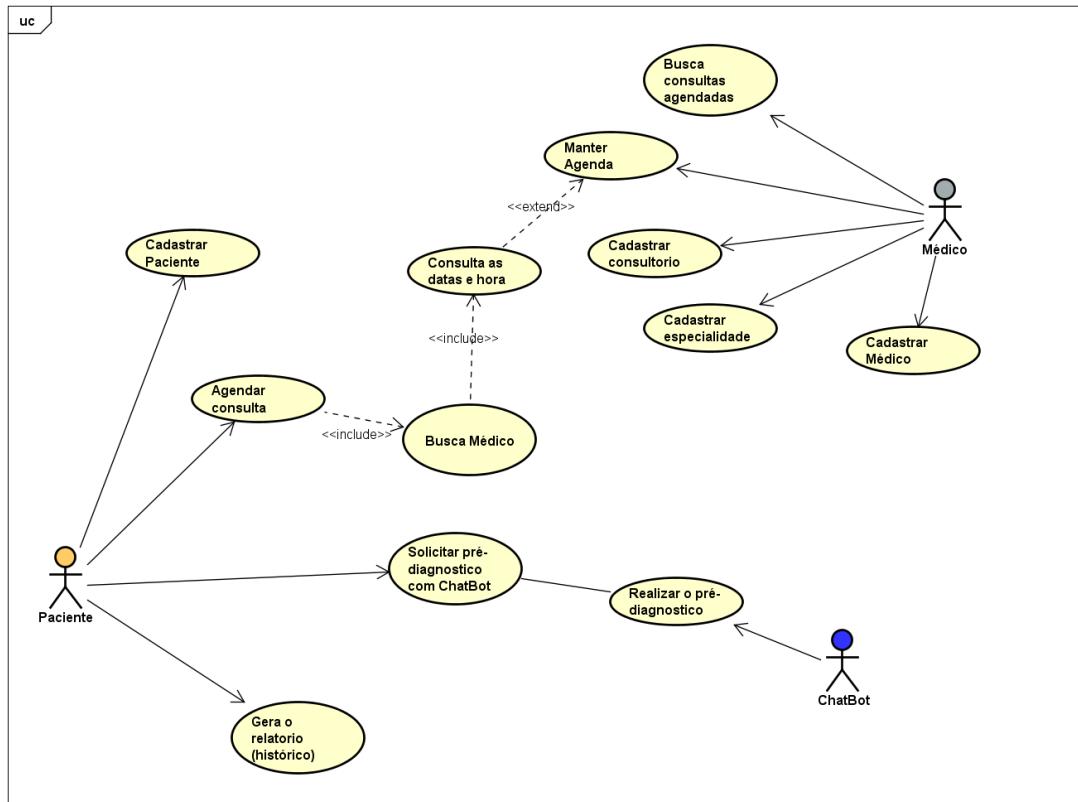
4.2 Diagramas

Nessa seção, serão apresentados os diagramas de caso de uso e de classes, em que ambos serão a base da engenharia e estruturação desse sistema.

4.2.1 Diagrama de Caso de Uso

Ao terminar a realização da análise e levantamento de requisitos e os casos de uso, foi desenhado o diagrama de caso de uso, no qual está representado abaixo na Figura 15.

Figura 15 – Diagrama de Caso de uso



Fonte: do Autor, 2019.

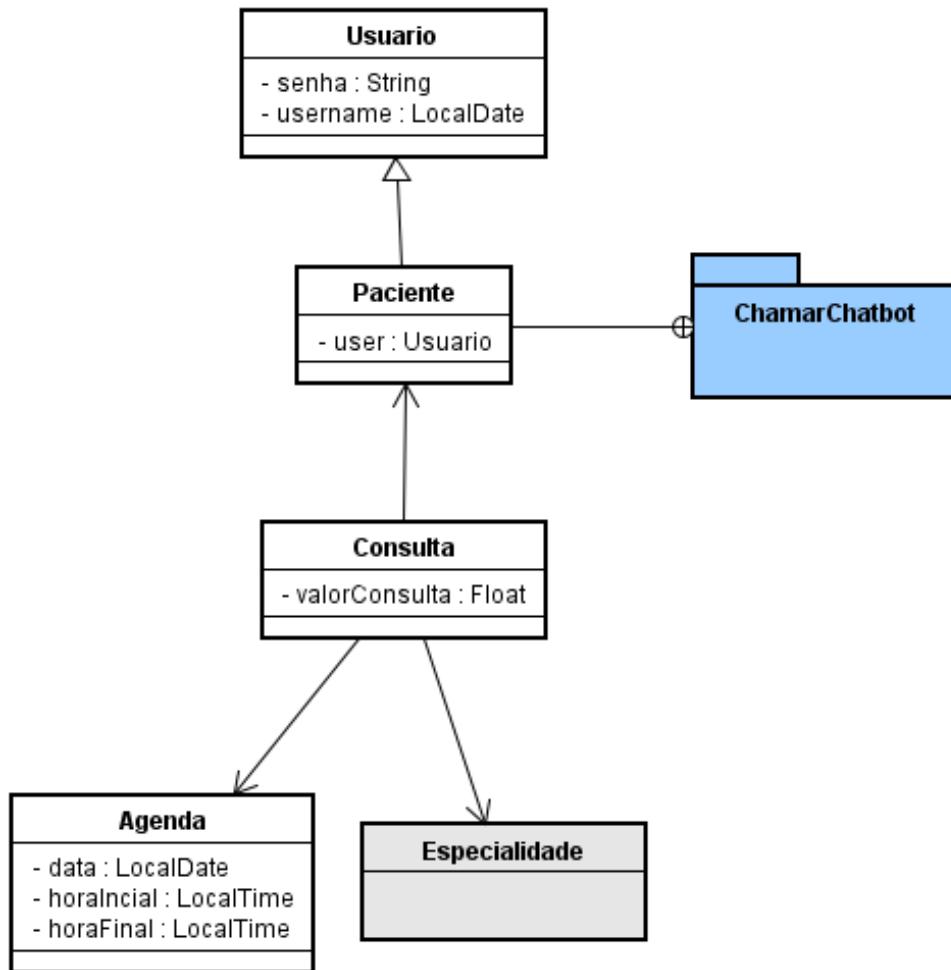
Nesse diagrama da Figura 15, foram definidas as atividades que serão executadas pelo usuário no qual é mostrado cada autor, que são os pacientes, os médicos,e o chatbot, que entra como um componente que está integrado ao sistema a fim de ser requisitado pelo usuário.

Cada perfil de usuário terá sua permissão dentro do sistema, por exemplo, o paciente terá acesso a área de agendamento de consultas, na qual ele terá todos os médicos disponíveis, com suas determinadas agendas para a marcação da consulta. Por outro lado, o médico terá apenas que verificar na sua agenda, as datas e horários que ele tem paciente agendado. Foi realizado o detalhamento desses casos de uso, que está listado no APÊNDICE B.

4.2.2 Diagrama de Classe

Após ter criado o diagrama de caso de uso, foi necessário desenhar também o diagrama de classes do sistema para melhor documentar e entender o funcionamento do sistema. Como resultado, será aqui mostrado o diagrama por partes, dividido pelas principais partes, o paciente e o médico, para uma melhor visualização.

Figura 16 – Diagrama de Classe (Parte do paciente)



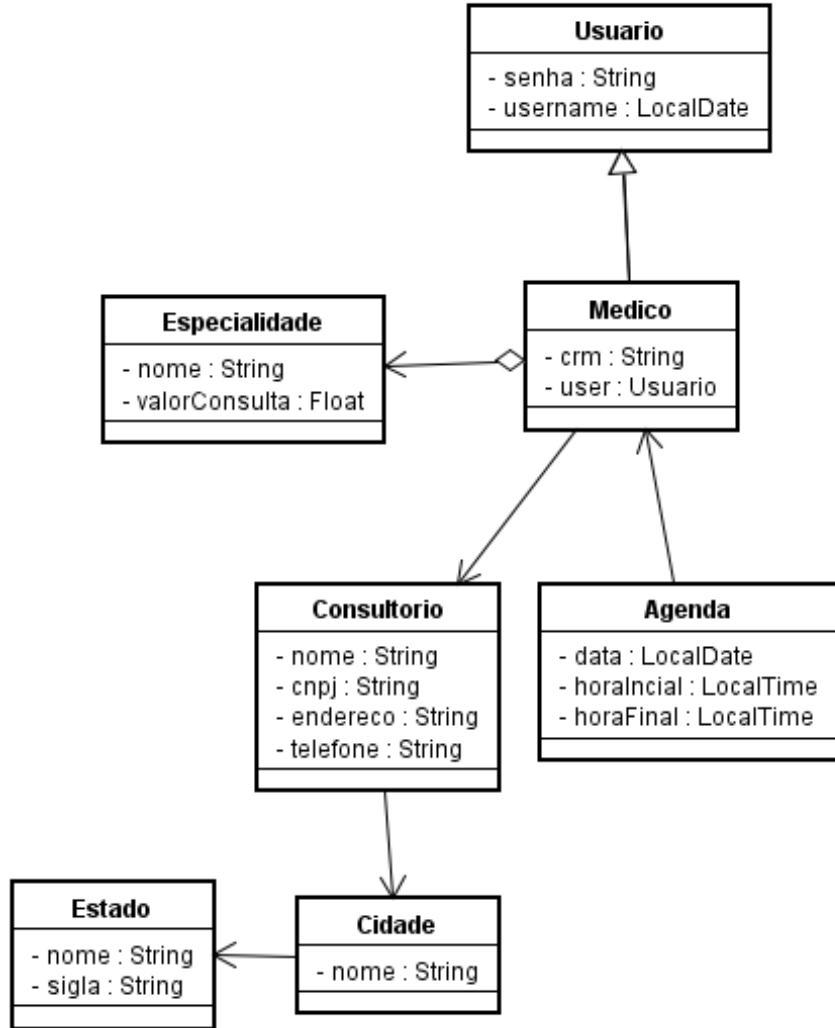
Fonte: do Autor, 2019.

Na Figura 16, representa as classes que estarão compondo o paciente. Nele pode ser observado a presença do paciente. Caso já tenha realizado seu cadastro no sistema, irá então seguir o seu principal objetivo que é agendar uma consulta com um médico. Para isso ele fará um filtro por especialidade, e então irá selecionar o médico. Porém, antes de realizar o agendamento da consulta, o Paciente poderá primeiramente interagir com o Chatbot, requisitando-o através da interface, e então realizar seu pré-diagnóstico e seguir para um suposto agendamento.

Ao selecionar um médico para agendar uma consulta, o paciente deverá consultar a

lista de datas e horários disponíveis para aquele médico, para então realizar uma consulta. Ao realizar o agendamento, será apresentado o valor a ser pago, valor este que foi definido pelo médico, de acordo com sua especialidade, como foi especificado na Figura 17.

Figura 17 – Diagrama de Classe (Parte do Médico)



Fonte: do Autor, 2019.

Como mostrar a Figura 17, o médico ao está devidamente logando ao sistema, ele terá a disponibilidade de se cadastrar com uma ou mais especialidade. Cada especialidade que for a ele associada, receberá um valor pelo mesmo para ser cobrado nas consultas. Deste modo, quando o Paciente selecionar o médico pela sua especialidade, a ele já será cobrado o valor da consulta.

O Médico também deverá cadastrar seu consultório, passando seus dados, a sua agenda, colocando seus horários e datas disponíveis e tendo a necessidade de sempre manter a agenda atualizada. Por fim, o diagrama de classes completo está no APÊNDICE C, no qual foi desenhando todas as classes que compõe o sistema. Dessa forma, seguiremos com os resultados, mostrando o que mais foi realizado no presente projeto.

4.3 Watson Assistant vs Dialogflow

Na fase de estudo de referenciais teóricos, surgiu uma dúvida sobre qual ferramenta utilizar neste projeto para o desenvolvimento do chatbot. Na Figura 18 foi abordado os principais pontos do processo de criação de um chatbot que se deve considerar, a fim de realizar uma comparação entre essas duas poderosas ferramentas.

Figura 18 – Comparação entre Watson Assistant e Dialogflow

IBM Watson Assistant	Dialogflow
- SIM, PORÉM COM UM PEQUENO CONJUNTO DE ENTIDADES DO SISTEMA	- SIM, E COM UM BOM CONJUNTO DE ENTIDADES PRONTAS PARA USAR
- SIM, ATRAVÉS DE UMA DEFINIÇÃO VISUAL DE FLUXO DE TRABALHO	- SIM, INCORPORADO EM INTENÇÕES
- SIM	- SIM
- SIM, LINGUAGEM DE EXPRESSÃO SIMPLES, MAS PODEROSA, PARA DEFINIR DIÁLOGOS E RESPOSTAS CONDICIONAIS.	- NÃO
- SIM, QUANDO OCORREM CHAMADAS DE API OU EM ALTERAÇÕES FEITAS NA HORA DO	- SIM, ONLINE COM A CAIXA DE DIÁLOGO DO BATE-PAPO AO VIVO
- SIM	- SIM
- SIM	- SIM
- SIM	- SIM
- SIM, UM CONJUNTO QUE VEM AUMENTANDO COM ATUALIZAÇÕES	- SIM, UM GRANDE CONJUNTO
- SIM	- SIM
- PLANOS DIFERENTES ESTÃO DISPONÍVEIS, DEPENDENDO DO NÚMERO DE ESPAÇO DE TRABALHO E DO NÚMERO DE CHAMADAS DE API / MES.	- GRATIS, PORÉM COM PLANOS PAGOS TAMBÉM

Fonte: do Autor, 2018.

Ao colocar lado a lado essas duas ferramentas, ficou claro que ambas tem recursos semelhantes, às vezes um mais que o outro, porém com as mesmas finalidades. No Dialogflow ficou evidente a enorme possibilidade de agregar conjuntos de entidades já pré-definidos na ferramenta, o que facilita bastante no desenvolvimento. Por outro lado, o Watson Assistant também trás entidades já pré-definidas, porém com um conjunto menor, o que acaba exigindo mais do desenvolvedor na hora de definir as entidades.

Um outro ponto importante dessa comparação fica por conta do diálogo. No Dialogflow o diálogo já é realizado juntamente com as intenções, o que ao meu ponto de vista acaba sendo desvantajoso em relação ao concorrente. O Watson Assistant trata o diálogo separadamente , criando fluxos de conversas, em que é dividido por nós principais e nós filhos, como em uma árvore binária. Dessa forma , ficou mais visível a criação do fluxo de diálogo , possibilitando a utilização de operações lógicas e expressões regulares para melhor fluxo de conversa.

Além desses pontos observados, foi analisado a forma de aprendizado. Ambos apresentam a utilização de Aprendizado de máquina e o uso de NLP, porém o Watson Assistant se destaca pela clareza no processo de aprendizagem e melhoria, no qual fica visível a cada alteração realizada que o bot sempre será treinado, possibilitando também que seja feito o processo de aprendizagem manualmente. O Dialogflow acaba não transparecendo esse processo , porém possibilita que o desenvolvedor acesse a área de treinamento para poder realizar as melhorias necessárias.

A Google disponibiliza os recursos do Dialogflow totalmente grátis, porém também permite que o desenvolvedor seja um assinante premium . Por outro lado, a IBM libera através do plano Lite(gratuito), recurso limitados de 10 mil requisições por mês, porém, suficientes para a conclusão do projeto e a obtenção dos resultados.

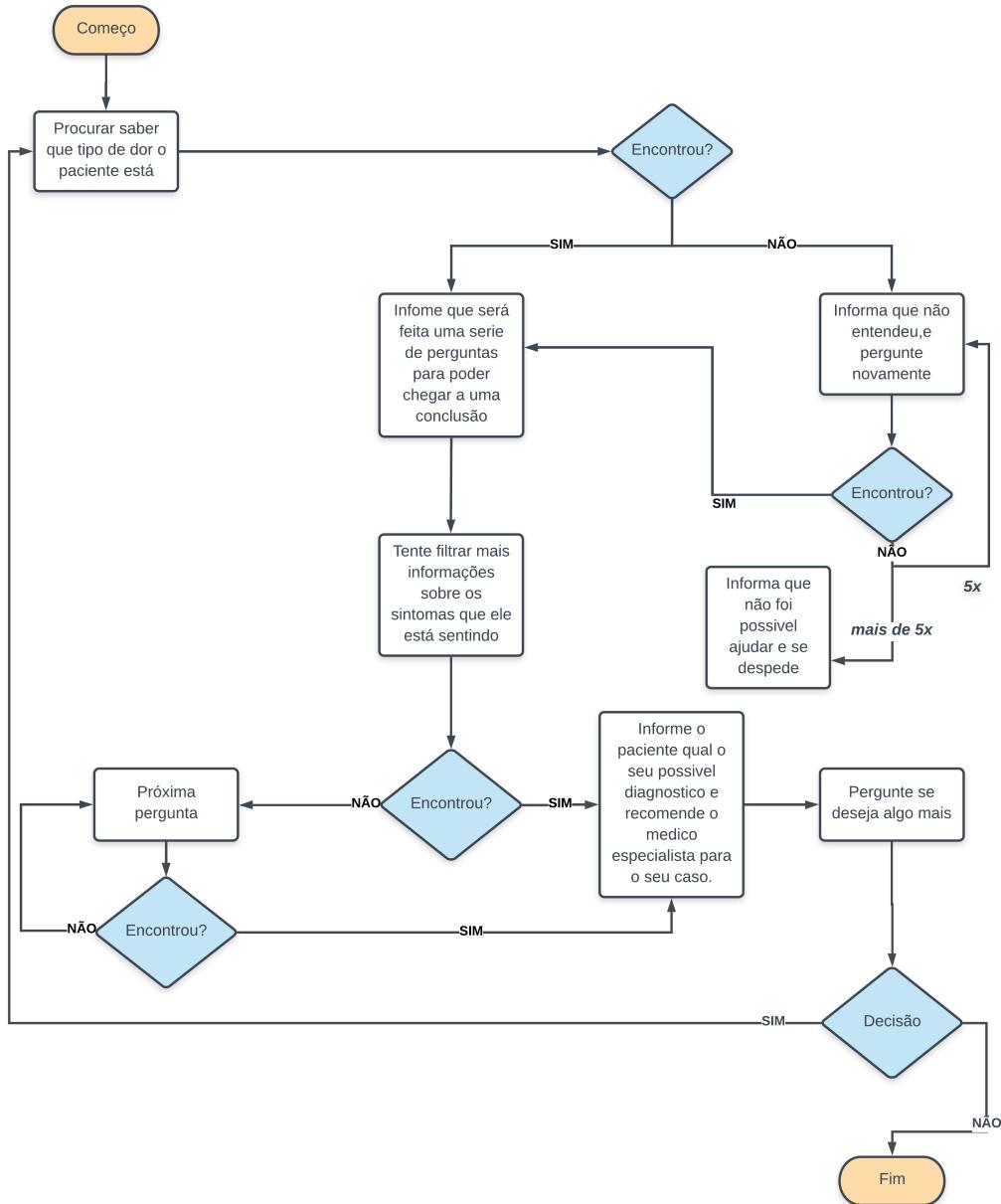
Embora o Dialogflow, o antigo API.ai, já esteja no mercado a mais tempo, e esteja mais maduro, ficou claro a preferencia pelo Watson Assistant, que nesse projeto será o ideal, possibilitando um desenvolvimento mais agradável, e um produto mais confiável, que é exatamente o que se espera desse projeto.

Dessa forma, o Watson Assistant foi escolhido como a principal ferramenta para o desenvolvimento do Chatbot neste projeto. Embora, que no processo de estudo da ferramenta tenha sido encontrada algumas dificuldades como a de integração com serviços de Web, porém a IBM tem conseguindo resolver esse problemas com suas atualizações, e já vem possibilitando uma melhora na integração com os Web sites. Na próxima seção será mostrado o resultado do fluxo genérico do Chatbot que será desenvolvido.

4.4 Fluxograma Chatbot

Neste primeiro momento, antes de se aprofundar no desenvolvimento do chatbot em sim, fica mais viável desenhar o fluxo de funcionamento do Bot. Logo abaixo, na Figura 19, podemos compreender o processo.

Figura 19 – Fluxograma do Chatbot



Fonte: do Autor, 2018.

O chatbot aqui arquitetado irá funcionar de forma interrogativa, simulando uma consulta com um médico, a fim de obter um pré-diagnóstico adequado. Na Figura 19 vemos que o Bot inicia a conversa e já pergunta qual o tipo de dor que o paciente está sentindo, sendo que, a partir deste ponto o bot irá começar a fazer uma série de perguntas já pré-definidas para aquele determinado sintoma listado pelo paciente, e caso o Chatbot encontre algo correspondente, ele envia ao paciente o seu possível diagnóstico e recomenda seu médico ideal, finalizando então o seu atendimento ou não.

4.5 Protótipos

Para melhorar o entendimento sobre o resultado final dessa aplicação, foi desenhado e implementado alguns protótipos do sistema para ter como norte na fase de desenvolvimento.

4.5.1 Protótipo da aplicação

Como parte de documentação e boas práticas de desenvolvimento, foi realizado o desenho da ideia inicial de como deveria ser apresentado o sistema.

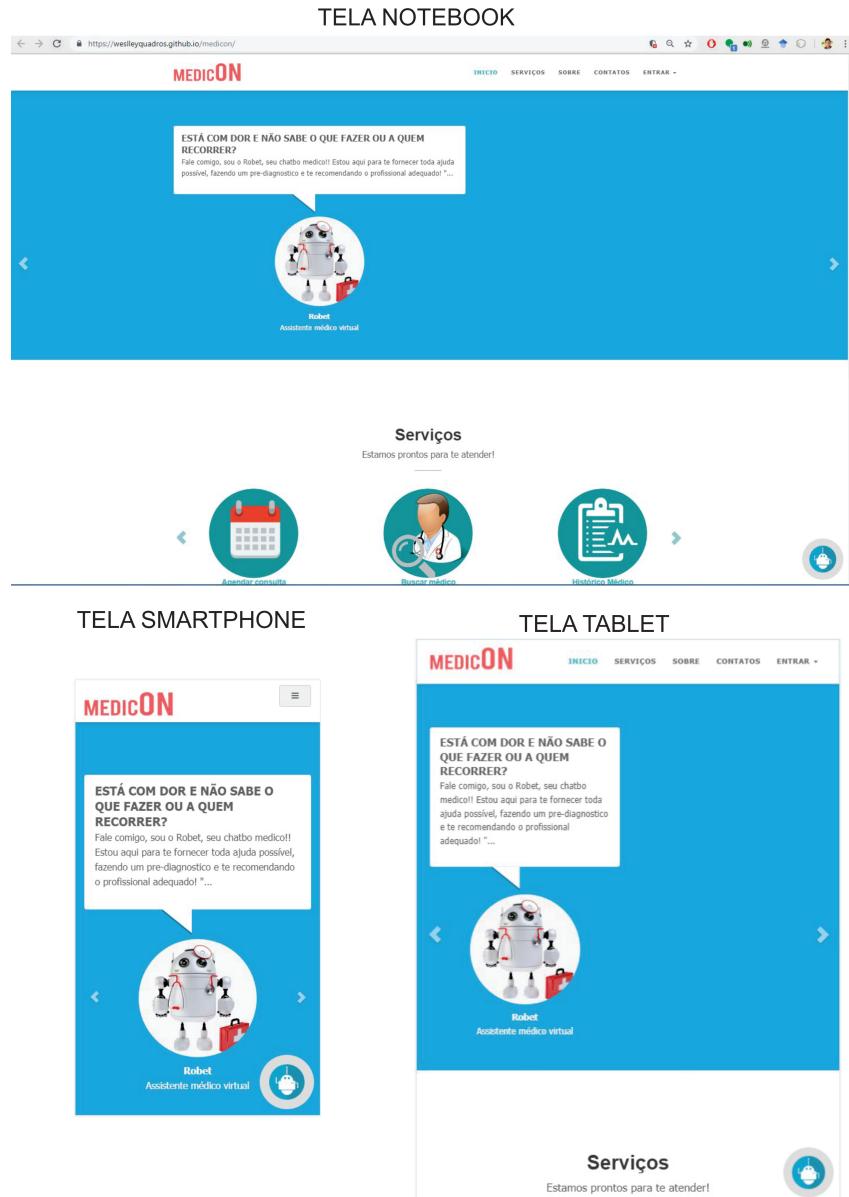
Figura 20 – Protótipo inicial do sistema desenhado



Fonte: do Autor, 2018.

Na Figura 20, mostra a tela inicial do sistema desenhada, na qual o mesmo foi replicado em diversas telas padronizadas, com dimensões diferentes, para mostrar como ficará a aplicação nas mais diversos dispositivos. Na imagem percebe-se que o layout da tela principal foi dividido por seções, no qual estão informações e também os botões para direcionar a outras páginas do sistema, como por exemplo, a página de agendamento de consultas. Todas as outras páginas foram implementadas em cima do mesmo template da tela inicial, a fim de padronizar o sistema.

Figura 21 – Protótipo de tela Implementado



Fonte: do Autor, 2018.

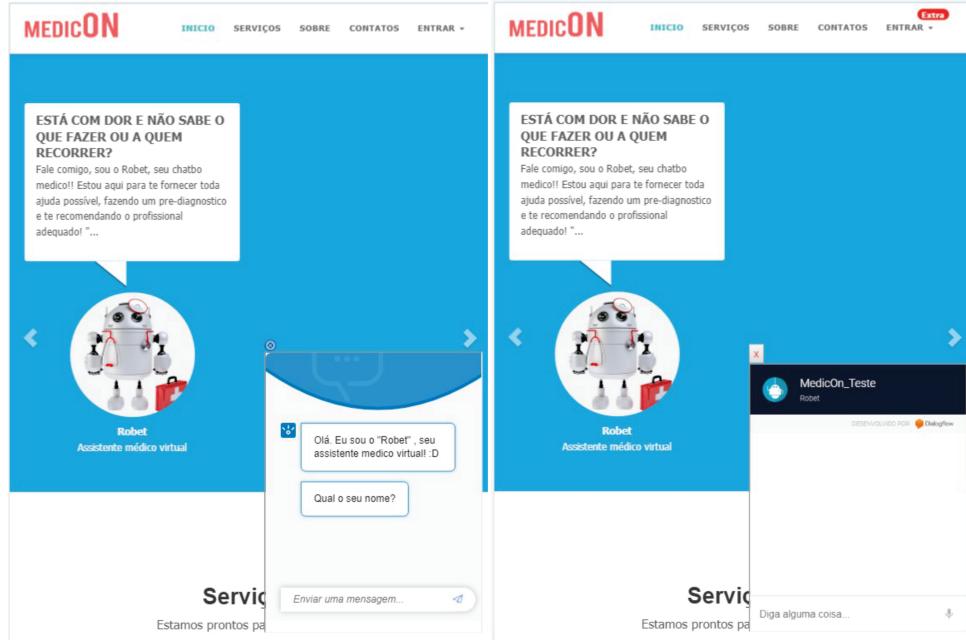
Após o desenho, foi realizado a implementação deste protótipo, na qual é a parte comercial do produto, o que tende a ser o cartão de visita ao paciente que deseja utilizar os serviços aqui proposto. O mesmo foi desenvolvido utilizando o framework Bootstrap, com algumas adequações em HTML, CSS3 e JavaScript. Na Figura 21 acima também pode se perceber a presença de um botão flutuante para o Chatbot, no qual o mesmo estará chamando a interface do Chatbot, que será mostrado na próxima seção.

4.5.2 Protótipo de Chatbot

Após a implementação inicial da aplicação, foram realizados alguns testes de implementação do chatbot utilizando ambas as ferramentas que foram comparadas na

seção 4.3. A partir da implementação deste protótipo do chatbot, surgiu a decisão de optar em trabalhar com o Watson Assistant.

Figura 22 – Protótipo de Chatbot Implementado(Watson Assitant e Dialogflow).



Fonte: do Autor, 2018.

Na Figura 22 mostra os dois protótipos rodando no na página web. Do lado esquerdo mostra o chatbot implementado com o Watson Assistant e do lado direito, implementado com o Dialogflow. As implementações iniciais foram simples, servindo apenas para teste, porém, como dito anteriormente, o Watson teve um desempenho melhor, mais confiável, e por fim, com a última atualização da IBM, também ficou com uma interface para Web mais agradável e fácil de fazer integração.

4.6 Desenvolvimento

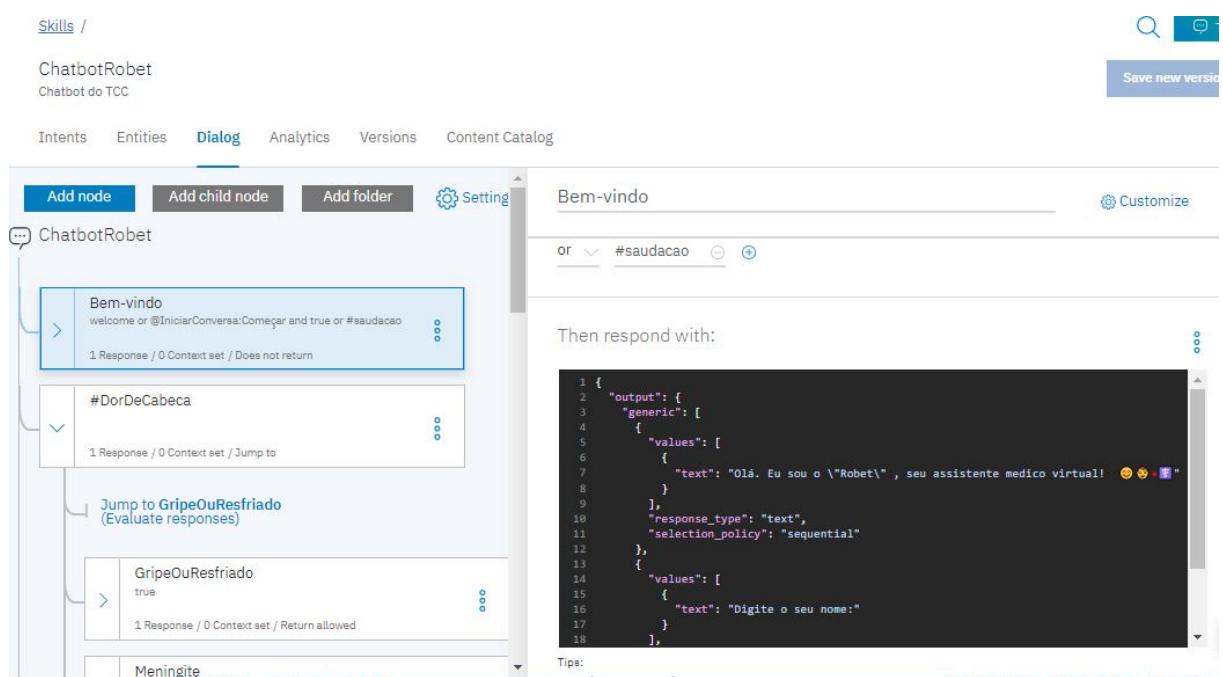
Nessa seção será apresentado todo resultado obtido da fase de desenvolvimento do presente projeto.

4.6.1 Chatbot - Assistente Médico Virtual

O chatbot criado, que é um assistente médico virtual, fora nomeado de "Robet" para melhor interação entre homem e máquina. Este foi mais visivelmente trabalhado no projeto devido a sua alta complexidade. Para a realização do mesmo foi necessário montar uma base de dados para então usá-los no processo de desenvolvimento da base de conhecimento do chatbot. A base de conhecimento foi feito utilizando a própria ML do Watson Assistant, sendo usado um processo de aprendizado supervisionado, mapeando as intenções e entidades

manualmente, e treinando a IA sempre que necessário de forma manual. Se tratando de diagnósticos, essa base de dados foi criada a partir de pesquisas e estudo através de profissionais da área da saúde e de protocolos.

Figura 23 – Fluxo de diálogo - Watson Assistant.



Fonte: do Autor, 2019.

Na imagem 23 é demonstrado como foi criado o fluxo de diálogo do chatbot. A IBM possui uma plataforma de desenvolvimento visual que facilita e agiliza o processo de criação do assistente. Porém, o Watson Assistant é construído e codificado no formato de JSON, portanto o mesmo pode ser estruturado dessa forma como mostra no trecho de código abaixo.

```

1 {
2     "intent": "DorDeCabeça",
3     "examples": [
4         {
5             "text": "sinto incomodo na cabeça"
6         }, // ...
7     ],
8     "description": "Diagnóstico por dor de cabeça"
9 }
10 ],
11 "entities": [
12     {
13         {
14             "entity": "derrame",
15             "values": [
16                 {
17                     "text": "Olá. Eu sou o \"Robot\", seu assistente médico virtual! 😊💡"
18                 }
19             ]
20         }
21     }
22 ]

```

```

17     "type": "synonyms",
18     "value": "não",
19     "synonyms": [
20       "nao",
21       "não"
22     ]
23   },
24   {
25     "type": "synonyms",
26     "value": "sim",
27     "synonyms": [
28       "sim",
29       "isso",
30       "exato"
31     ]
32   }
33 ],
34   "fuzzy_match": true
35 }
```

O código acima mostra basicamente toda a estrutura do chatbot utilizando Json. Nessa estrutura foi inicialmente codificado as *Intents*(intenções) e seus respectivos valores, em que está contido as prováveis intenções do paciente ao dialogar com o assistente. Depois foi criado as *Entity*(entidades) que são abstraídas de uma intenção. Com a entidade, foi passado valores e seus sinônimos para ajudar no processo de decisão do chatbot. Ao finalizar esse processo, o chatbot passa por uma processo de treinamento, no qual a ML do Watson é preparada para realizar o trabalho de reconhecimento de padrões. Nota-se que na linha 35 do código foi passado o parâmetro *"fuzzy_match"* sendo igual a *"true"*. Isso significa que a IA será treinada com capacidade de entender palavras próximas, que não foram passadas como sinônimos nas entidades. Isso evita que o Chatbot não entenda algumas palavras, ou até mesmo erros de digitação do usuário.

Ao finalizar o desenvolvimento do chatbot, obteve-se uma API que pode ser utilizado em qualquer aplicação e aplicativos de mensagens no qual busca um auxilio ou suporte a tomada de decisão pra pacientes. Para testar essa API, pode ser usado a própria plataforma da IBM, porém foi criado um servidor *back-end* em Node.js que além de possibilitar fazer testes através de métodos *"post"*, ele também possibilita integrar com outros serviços e qualquer aplicação.

```

1 const AssistantV1 = require('watson-developer-cloud/assistant/v1');
2 require('dotenv').config();
3 const axios = require('axios');
4 const assistant = new AssistantV1({
5   username: process.env.ASSISTANT_USERNAME,
6   password: process.env.ASSISTANT_PASSWORD,
```

```

7     url: "https://gateway.watsonplatform.net/assistant/api",
8     version: '2019-03-29'
9 );
10 module.exports = {
11   createWatson(req, res){
12     const { text, context = {} } = req.body;
13     const params = {
14       input: { text },
15       workspace_id: process.env.WORKSPACE_ID,
16       context,
17     };
18     assistant.message(params, (err, response) => {
19       if (err) {
20         return res.status(500).json(err);
21       }
22       else{
23         return res.json(response);
24       }
25     }
26   });
27 },
28 }
```

No código acima mostra como a chamada do chatbot foi realizada. No servidor foi criado um arquivo chamado ”.env”, no qual foi passado as credenciais da API, assim como esta sendo montado a constante da linha 4 do código. Através de tais credenciais, foi possível ter acesso à API. Após isso foi criado um contante que recebe como parâmetro, o texto do usuário, o ”id”do *Workspace* do chatbot e o contexto do dialogo(linha 13). Por fim, a partir da linha 18, a API manda como mensagem, os parâmetros anteriormente criados, e os tipos de mensagens(”err”e ”response”). Para poder acessar as informações que foram passadas, foi criado um arquivo de rotas(*routes*), no qual é criado um método *post* de chamada http.

```

1
2 const express = require('express');
3 const routes = express.Router();
4 const AssistantController = require('../controllers/AssistantController');
5
6 routes.post('/conversation', AssistantController.createWatson);
7
8 module.exports = routes;
```

Para facilitar a integração com sistemas web, sites, neste trabalho foi feito uma integração mais simples. Essa integração foi feita no código html, que faz uso do próprio modelo de caixa de dialogo do Assistant. O mesmo foi manipulado e customizado para melhor atender inicialmente as demandas de conversação.

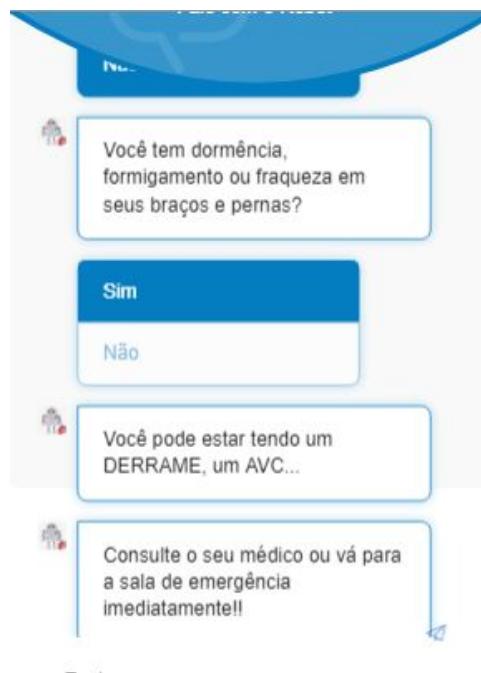
```

1 <div class='Chat__body'>
2
3 </div>
4 <script src="https://assistant-chat-us-south.watsonplatform.net/web/chat/
5     ibm.chat.0.0.4.js"></script>
6 <script>
7     const config = {
8         bot_image: 'img/botTeste.jpg',
9         bot_name: 'Fale com o Robet',
10        debug: true,
11        element: document.querySelector('.Chat__body'),
12        integration_id: '54f29a5c-cb18-469e-9dfe-c05891dc289c',
13        post_url: 'https://assistant-chat-us-south.watsonplatform.net/public/
14            message/'
15    };
16    const chat = new WatsonChat(config);
17 </script>

```

Esse *script* foi chamado em uma outra *div* que contém um botão flutuante com *iframe*. Nota-se que na linha 13, a url de chamada vem direto da API do Watson, porém pode ser passado a url da API criada. O resultado desse desenvolvimento mostra um chatbot que realiza uma diagnóstico através de supostas doenças que podem ser causadas por uma dor de cabeça, assim como é mostrado na Figura 24.

Figura 24 – Assistente médico em execução.



Fonte: do Autor, 2019.

Com o objetivo de melhorar e tornar mais inteligente o processo de diagnóstico,

foi desenvolvido um reconhecimento de imagens para o chatbot em questão, em que será detalhado a seguir.

4.6.1.1 Diagnóstico por imagem - Visual Recognition

Apesar de já se ter um produto viável que realiza-se um diagnóstico prévio para tomada de decisão por meio textual, foi pensando e codificado uma forma de fazer com que o mesmo chatbot, anteriormente criado, pudesse fazer diagnóstico por imagem, utilizando uma outra API criada a partir de um outro serviço do Watson, o Visual Recognition.

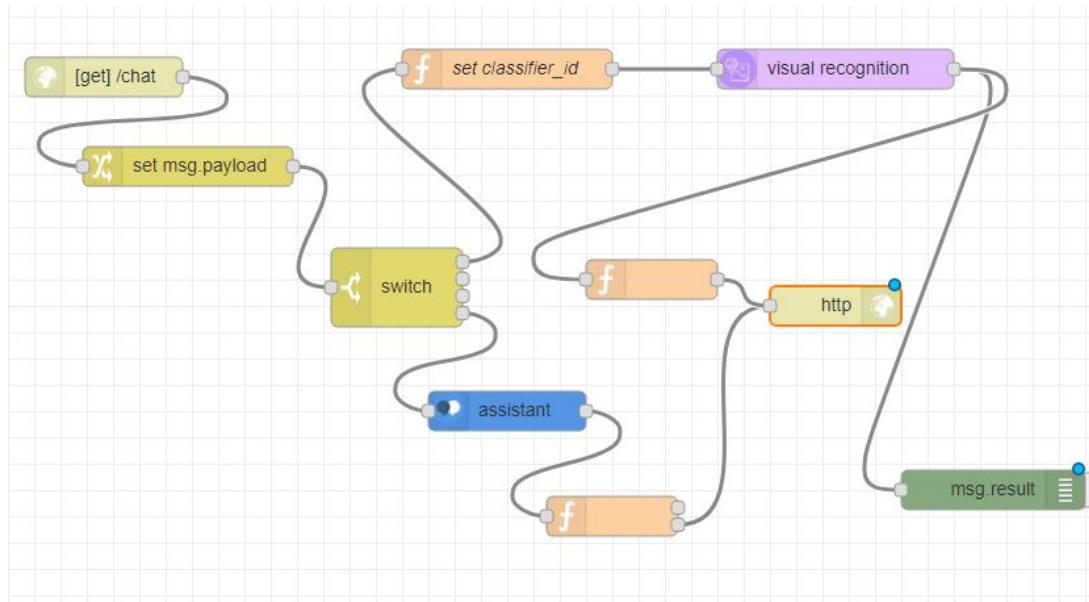
Figura 25 – Imagens de queimaduras



Fonte: do Autor, 2019.

Na Figura 25 mostra as imagens de queimaduras que foram selecionadas e subdivididas por grau. Por meio da ferramenta visual do Watson, esse processo de criação do banco de imagens é bastante ágil. Ao terminar o envio das imagens, foi necessário realizar o treinamento do Watson. Concluído isso, foram realizados os testes passando imagens e verificando sua respectiva saída. Assim, o próximo passo foi integrar essa API criada ao chatbot, e transformar os dados do VR em respostas no chatbot. Para isso foi utilizando o Node-RED, que realizou o papel de conectar os serviços desenvolvidos.

Figura 26 – Fluxo da integração dos serviços no Node-RED



Fonte: do Autor, 2019.

Na Figura 26 é apresentada a área de trabalho criada no Node-RED, em que foi criado fluxo que ligam os dois serviços desenvolvidos no projeto. No inicio do fluxo esta sendo recebido a requisição do usuário, que é tratado e analisando posteriormente através de um *switch* que verifica o tipo de mensagem recebida, caso seja uma mensagem em texto, realizando o fluxo normal como foi mostrado na Figura 24. Porém, se a mensagem recebida for uma imagem, passa para a função "*set classifier-id*" que é passado o Id da API criada.

```
1
2 var cid = "VRxqueimaduras_82384505"
3 msg.params = [];
4 msg.params["classifier_ids"] = cid;
5
6 return msg;
```

Após a execução da função mostrada no código acima, os parâmetros foram enviados para o Visual Recognition. Na linha 2 do código a variável "cid" recebeu um valor que é correspondente ao identificador da API criada na Figura 25. O VR envia um corpo de informações em JSON, que para ter uma resposta mais elaborada, foi criado um algoritmo que trata essas respostas da API do VR e cria as condições para que o chatbot possa enviar a resposta ao paciente.

```
1 var aux;  
2 if (msg.result.images.length > 0)  
3 {  
4     if (msg.result.images[0].classifiers)  
5     {  
6         aux = "Eu vejo ";  
7         var tam = aux.length;
```

```

8     for(x = 0; x < msg.result.images[0].classifiers.length; x++)
9     {
10        if(aux.length > tam)
11        {
12            aux += "E ";
13        }
14        if(msg.result.images[0].classifiers[x].name == "VR-queimaduras"
15           && msg.result.images[0].classifiers[x].classes[x].class ==
16           "1    grau")
17        {
18            aux += "uma queimadura de 1    grau. \n\n Não é grave, mas
19            sugiro que vá ao médico! Caso contrário, mantenha a pela
20            sempre hidratada! \n\n - Essa análise conta com uma
               margem de acerto de " + msg.result.images[0].
               classifiers[x].classes[x].score;
} else if(msg.result.images[0].classifiers[x].name == "VR-
21         queimaduras" && msg.result.images[0].classifiers[x].classes[
22         x].class == "2    grau")
23        {
24            aux += "uma queimadura de 2    grau \n\n Devido a gravidade,
25            sugiro que vá ao médico para tomar as devidas providê
26            ncias! \n\n - Essa análise conta com uma margem de
27            acerto de " + msg.result.images[0].classifiers[x].
28            classes[x].score;
}

```

Depois de criar a função mostrada no código acima, as respostas foram criadas e enviadas para o chatbot, pronto para o *deploy* da API para a realização dos testes.

Figura 27 – Resposta do chatbot x Resposta da API



Fonte: do Autor, 2019.

A Figura 27 apresenta a forma em que os dados são enviados para o servidor e

como ele é entregue ao cliente. Nota-se que ao final da mensagem foi colocado o *score*, que é em números, a porcentagem de acerto, na qual foi impresso da forma que a API envia, sendo sempre ”*x*/100”, onde ”*x*” é igual a porcentagem.

4.6.2 Estrutura do Back-end

Para a realização da API do sistema Web, foi utilizado o Node.js, no qual junto com o mesmo foi utilizado o MongoDB como banco de dados. Embora seja um banco não relacional, com o Mongo pode criar tabelas referenciadas através dos Models da aplicação. Essa referencia foi feita através de ”ids” no qual permaneceu a mesma regra desenhada na arquitetura.

Como citado anteriormente, na fase de estudo dos referenciais teóricos, o Node.js se baseia na arquitetura MVC, dessa forma alguns diretórios foram criados para estruturar o projeto. Foi criado os *Models* de cada entidade, passando seus atributos, para então poder criar as rotas(*Routes*) de cada componente. Nas rotas é onde foi criados os métodos de Http em que é realizado a inserção, deleção, atualização e listagem(Get, Put, Post, Delete, entre outros).

```

1
2 const UserSchema = new Schema({
3     name: { type: String, required: true },
4     cpf: { type: String, required: true, unique: true, lowercase: true },
5     rg: { type: String, required: true, unique: true, lowercase: true },
6     email: { type: String, required: true, unique: true, lowercase: true },
7     usuario: { type: String, required: true, unique: true, lowercase: true
8         },
9     password: { type: String, required: true, select: false },
10    is_medic: { type: Boolean, required: true },
11    created: { type: Date, default: Date.now },
12 });
13 UserSchema.pre('save', async function (next) {
14     let user = this;
15     if (!user.isModified('password')) return next();
16
17     user.password = await bcrypt.hash(user.password, 10);
18     return next();
19 });
20
21 module.exports = mongoose.model('User', UserSchema);

```

O código acima a criação do *model* Usuário. Nesse modelo, foi criado um tipo de usuário que por padrão é um paciente. Porém como mostra na linha 9, caso o ”*is_medic*” seja setado como ”*true*”, esse usuário já passa ser do tipo médico, que contendo então atributos

de médico, como o CRM.

```

1 //FUN ES AUXILIARES
2 const createUserToken = (userId) => {
3     return jwt.sign({ id: userId }, config.jwt_pass, { expiresIn: config.
4         jwt_expires_in });
5 }
6
7 router.get('/', Authorization, async (req, res) => {
8     try {
9         const users = await Users.find({});
10        return res.send(users);
11    }
12    catch (err) {
13        return res.status(500).send({ error: 'Erro na consulta de usuários!
14            ' });
15    }
16 });
17
18 router.post('/create', async (req, res) => {
19     const {name, cpf, rg, email, usuario, password, is_medic} = req.body;
20
21     if (!name || !cpf || !rg || !email || !usuario || !password || !
22         is_medic) return res.status(400).send({ error: 'Dados insuficientes!
23             ' });
24
25     try {
26         if (await Users.findOne({ usuario })) {
27             return res.status(400).send({ error: 'Usuario já registrado!' });
28         }
29         const user = await Users.create(req.body);
30         user.password = undefined;
31
32         return res.status(201).send({mensagem: {title: "Sucesso", mensagem:
33             "Usuário Cadastrado com sucesso!"}, user, token:
34             createUserToken(user.id)});
35     }
36     catch (err) {
37         console.log(err);
38
39         return res.status(500).send({ error: { title: 'Algo deu errado',
40             msgn: 'Erro ao buscar usuário!', erro: err } });
41     }
42 });

```

Nó código anterior apresenta-se como foi realizado a rotas de usuário. Devido ao

tamanho do arquivo, nesse trecho de código demonstra-se apenas os *endpoints* contendo as funções de listagem de usuários, e também a *create* que é um método *post* para cadastro de usuários. A estrutura geral da API se baseia nesses trechos de código. Ao terminar a estrutura da API, já foi possível prosseguir e construir o *Front-end* da aplicação.

4.6.3 Aplicação web

A aplicação Web que foi proposta nessa solução, foi desenvolvida utilizando o Angular. Como já mencionado, o Angular utiliza o *TypeScript*, no qual através do mesmo foi criado funções para consumir a API criada e realizar as operações do sistema. O primeiro passo do desenvolvimento do *Front-end* da aplicação foi criar os componentes. Em geral, são partes do sistema que são estruturadas em *Html*, *CSS* e *TypeScript*, porém criadas de forma ”componentizadas”, ou seja, separadamente.

No angular é necessário se criar um servidor que fica do lado cliente. Esses *services* tem o papel de realizar a requisição Http no *endpoint* da API. Esses endpoints foram chamados em um arquivo chamado ”api.ts”. como mostra o trecho de código a seguir.

```

1 export const API = environment.API;
2
3 export class EndPoints {
4
5   //AUTENTICACAO
6   public static login = () => {
7     return `${API}/users/auth`;
8   };
9
10  //USUARIO
11  public static createUser = () => {
12    return `${API}/users/create`;
13  };
14  //MEDICO
15  public static createMedico = () => {
16    return `${API}/medico/create`;
17  };
18
19
20  public static getEspecialidade = () => {
21    return `${API}/especialidade`;
22  };

```

Inicialmente, foi criado a tela de Login no qual faz a requisição na API gerando um *token*, em que para ser autenticado no sistema, o token tem que estar válido. Para isso foi criado um arquivo *services* onde foi criado a função que realiza a autenticação do usuário no sistema.

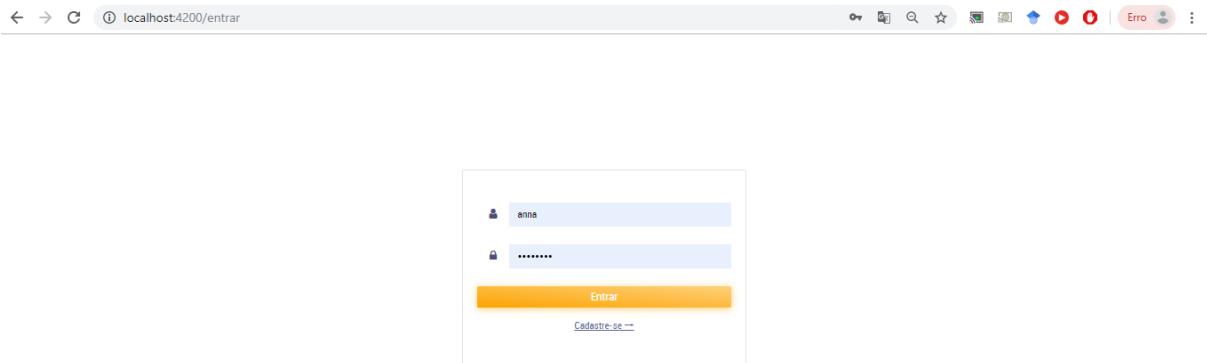
```

1 login(usuario): Observable<any> {
2
3     return this.http.post(EndPoints.login(), usuario).pipe(
4         tap(
5             data => {
6                 let token = data['token'];
7                 localStorage.setItem('token', token);
8                 return { ...data };
9             },
10            error => {
11                return error;
12            }
13        )
14    );
15}
16

```

Através do trecho de código desenvolvido e mostrado acima, foi possível realizar a chamada da função que gera a autenticação do usuário, tendo como resultado a Figura 28.

Figura 28 – Tela de Login da aplicação



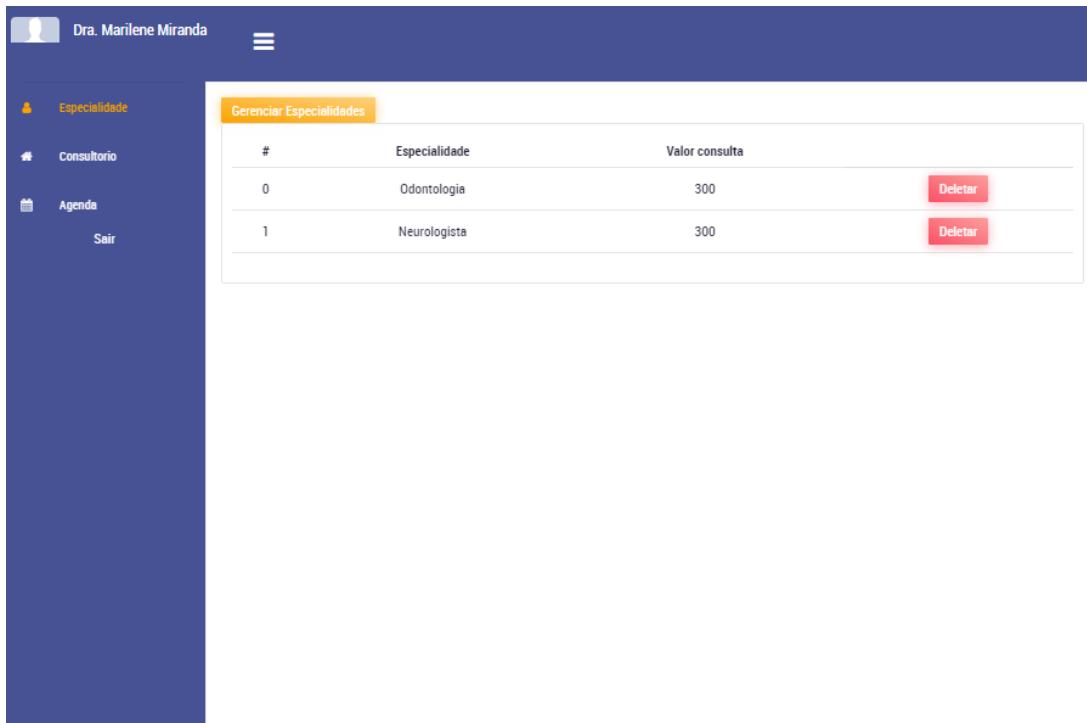
Fonte: print screen, do Autor, 2019.

Além do login, foi criado a tela de cadastro onde tem um formulário, e um *checkbox* para passar o tipo de usuário, se é o usuário padrão (paciente) ou médico. Após criar a autenticação do usuário no sistema, foi desenvolvido dois *dashboards* diferentes, painéis de controle que foram diferenciados pelo tipo de usuário utilizando exatamente o *endpoint* mostrado na seção anterior. Embora seja dois painéis, a API que é requisitada é a mesma, diferenciando apenas os *endpoints* que são chamados.

```
2   public listEspecialidades: any[] = [];
3   public localsForm: FormGroup;
4
5   constructor(public especialidadeService: EspecialidadeService, private fb
6       : FormBuilder) { }
7
8   ngOnInit() {
9     this.initForms();
10    this.getEspecialidades();
11  }
12
13  private initForms() {
14    this.localsForm = this.fb.group({
15      nome: new FormControl(''),
16      valor: new FormControl('')
17    });
18
19  private getEspecialidades(){
20    this.especialidadeService.getEspecialidade().subscribe(
21      res =>{
22        this.listEspecialidades = res;
23      },
24      err =>{
25        console.log(err);
26      });
27  }
28
29
30  private deleteLocals(id){
31    this.especialidadeService.deleteLocals(id).subscribe(
32      res=> {
33        this.getEspecialidades();
34        console.log(res);
35      },
36      err => {
37        console.log(err);
38      }
39    );
40  }
```

No perfil de Médico foi criado os componentes responsáveis por gerenciar as Especialidades e valor de consulta, foi criado também o de gerenciamento de agenda, e o de consultório. O médico ao se cadastrar no sistema já deve informar a sua especialidade para que, no momento em que o paciente estiver no sistema para agendar uma consulta, ele possa filtrar por especialidade e selecionar o médico desejado.

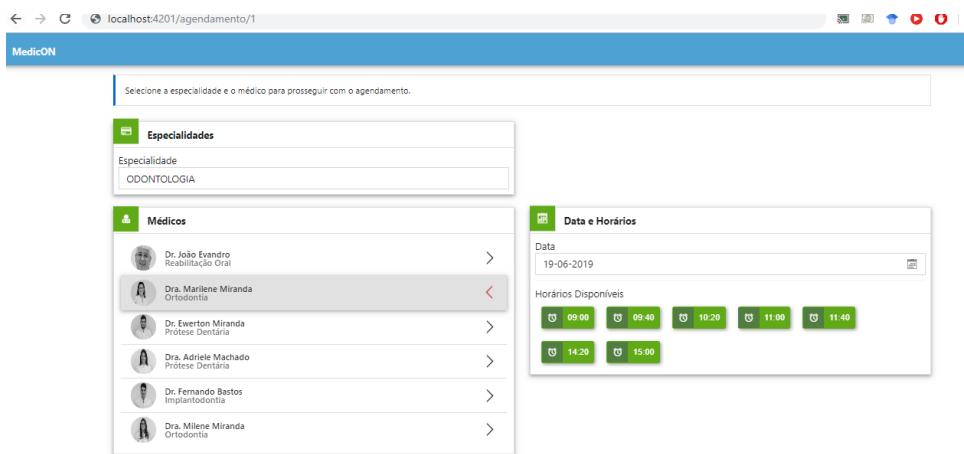
Figura 29 – Painel do médico



Fonte: print screen, do Autor, 2019.

Na Figura 29 e no trecho de código acima contém o resultado da estrutura do Painel do médico, e como ele se apresenta. Além dessas funcionalidades, no painel de médico também conta com os componentes de consultório e agenda.

Figura 30 – Tela agendamento de consulta por parte do Paciente

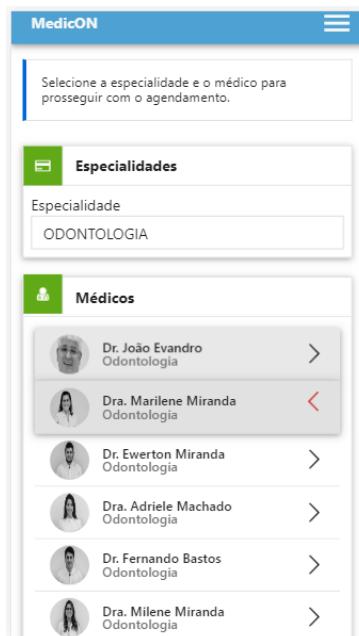


Fonte: print screen, do Autor, 2019.

No perfil de paciente, como mostra a Figura 30, foi criado funções que consomem os dados cadastrados pelos médicos. O componente principal é o de agendamento, no qual foi feito a requisição da API para trazer os médicos filtrados pela especialidade, sendo

assim, realizando a listagem do mesmo.

Figura 31 – Tela agendamento de consulta responsiva (Mobile)



Fonte: print screen, do Autor, 2019.

Na Figura 31 já fica claro o resultado da responsividade da aplicação. O mesmo foi adaptado a um aparelho celular, mostrando que é possível ter a aplicação em qualquer dispositivo, melhorando então a experiência do usuário com a aplicação.

5 Conclusão

O presente trabalho apresentou uma solução que mostra formas de resolver ou abrir um caminho para resolução da problemática que a saúde tem enfrentado. Deste modo, conclui-se que, a informatização e automatização nesses serviços médicos, tende a resultar em uma melhoria nesse processo, no qual para o usuário final(paciente), fica mais simples, transparente e de fácil acesso a todos, em vista da tecnologia disponível atualmente.

A proposta inicial de realizar um assistente médico para realizar um pré-diagnóstico foi bastante aceita desde o desenvolvimento até as fases de teste. Porém, ficou claro que para desenvolvimento de um assistente que consiga atender a situações diversas, é necessário muito mais estudo da área médica. Dessa forma, também ficou claro que o processo de treinamento da IA de forma manual acaba não sendo uma forma tão viável, fazendo com que seja levantado um novo estudo que visa, no futuro, tornar essa solução não supervisionada, se tornando inteligente ao ponto de conseguir aprender sozinha. Devido a esse fator foi que o *Chatbot* aqui desenvolvido, realiza diagnóstico apenas pelo protocolo de possíveis doenças que causa dores de cabeça.

Porém, tendo em vista o fluxo de usuários em hospitais e clínicas, e as altas demandas para realização de triagem, pode-se concluir que o assistente médico aqui desenvolvido pode ser uma solução que venha melhorar os atendimentos e auxiliar pacientes que necessitam de informação. Deste modo, essa solução tecnológica mostrou ter total espaço no mercado, isso devido aos benefícios que por eles são agregados. Ter um assistente que trabalha fazendo recomendações, estando totalmente disponível ao usuário foi o motivo que levou a levar esse projeto a aceitação do público alvo, mesmo que em alguns casos não chegando a um resultado esperado.

O projeto proposto conclui aqui com resultados considerados aceitáveis. Os erros e acertos nesse projeto iram contribuir para um desenvolvimento mais avançado de uma solução que visa a automação dos serviços médicos. Portanto, fica aqui definidos como metas futuras que podem enriquecer o projeto:

- Implementar os demais protocolos de diagnósticos por texto e por imagem, tendo em vista tornar o chatbot um assistente médico completo;
- Melhorar a ML, de forma que o Assistente seja capaz de aprender sozinho;
- Permitir acesso ao chatbot no sistema, possibilitando que o usuário logado consiga realizar agendamento através do próprio assistente;

- Resolver alguns *bugs* e inconsistências do sistema Web, além de melhorar ou implementar um *Widget* mais robusto que tenha suporte para envio de imagens.

Referências

- AFONSO, A. O que é angular? Disponível em <https://blog.algaworks.com/o-que-e-angular/>, 2018.
- APIMEDIC, A. *A Symptom Checker that stands out from the crowd*. 2018. Disponível em: <https://apimedic.com/>. Acesso em: 04 Jun. 2018.
- BAECKER, R. M. *Readings in Human-Computer Interaction: toward the year 2000*. [S.l.]: Elsevier, 2014.
- BARBIERE, L. *O Que é Bootstrap e Para Que Serve?* 2017. Disponível em: <https://www.ciawebsites.com.br/dicas-e-tutoriais/o-que-e-bootstrap/>. Acesso em: 02 Mar. 2018.
- BASÍLIO, S. *O que é Node-RED? Conhecendo e instalando*. 2018. Disponível em: <http://blogmasterwalkershop.com.br/outros/o-que-e-node-red-conhecendo-e-instalando/>. Acesso em: 29 Mai. 2019.
- BOOTH, J. D. Angular 2 sucintamente. Disponível em <https://www.syncfusion.com/ebooks/angular2succinctly/introduction>, 2017.
- BOTPRESS. *The Ultimate Bot Framework*. 2018. Disponível em: <https://botpress.io/>. Acesso em: 24 mai. 2018.
- BOWER, J. L.; CHRISTENSEN, C. M. Disruptive technologies: catching the wave. Harvard Business Review Video, 1995.
- BRANAS, R. Angularjs essentials. Disponível em <https://pepa.holla.cz/wp-content/uploads/2015/10/AngularJS-Essentials.pdf>, 2014.
- CARVALHO, L. Chatbots e a revolução digital. In: . FACTO, Palmas, Tocantins, Brasil: [s.n.], 2018.
- CHATFUEL. *Sobre o Chatfuel*. 2018. Disponível em: <https://chatfuel.com/newlp/about-us.html>. Acesso em: 28 mai. 2018.
- COPES, F. *The definitive Node.js handbook*. 2018. Disponível em: <https://www.freecodecamp.org/news/the-definitive-node-js-handbook-6912378afc6e/>. Acesso em: 27 Mai. 2019.
- CUER, E. d. S. Comparação de desempenho de bancos de dados sql e nosql. 2015.
- DIALOGFLOW, G. *Build natural and rich conversational experiences*. 2018. Disponível em: <https://dialogflow.com/>. Acesso em: 28 Mai. 2018.
- DIGITAL, O. *Site mostra como funciona o reconhecimento de imagens do IBM Watson*. 2016. Disponível em: <https://olhardigital.com.br/noticia/site-mostra-como-funciona-o-reconhecimento-de-imagens-do-ibm-watson/56320>. Acesso em: 29 Mai. 2019.

- DOMINGOS, P. A few useful things to know about machine learning. *Communications of the ACM*, ACM, v. 55, n. 10, p. 78–87, 2012.
- DUMAS, J. S.; REDISH, J. *A practical guide to usability testing*. [S.l.]: Intellect books, 1999.
- FARIA, T. Java ee 7 com jsf, primefaces e cdi. *SI sn*, 2015.
- FEIGENBAUM, E. A.; BARR, A. The handbook of intelligence. *Vol I*, 1981.
- FENTON, S. *Pro TypeScript: Application-Scale JavaScript Development*. [S.l.]. [S.l.: s.n.], 2017.
- FIORIN, D. V. et al. Aplicações de redes neurais e previsões de disponibilidade de recursos energéticos solares. *Revista Brasileira de Ensino de Física*, v. 33, n. 1, p. 1309, 2011.
- FONSECA, J. Metodologia da pesquisa científica.[apostila] fortaleza: Uec. 2002.
- GHISE, V. *O que é um Chatbot?* 2016. Disponível em: <<http://www.viniciusghise.com.br/blog/o-que-e-chatbot/>>. Acesso em: 28 mai. 2018.
- GOMES, C. *Chatbot: entenda tudo sobre o assunto*. 2017. Disponível em: <<http://blog.simply.com.br/chatbot/>>. Acesso em: 25 Mai. 2018.
- HAYKIN, S. S. et al. *Neural networks and learning machines*. [S.l.]: Pearson Upper Saddle River, 2009. v. 3.
- HEWETT, T. T. et al. *ACM SIGCHI curricula for human-computer interaction*. [S.l.]: ACM, 1992.
- IBM. *Visual Recognition*. 2019. Disponível em: <<https://www.ibm.com/watson/services/visual-recognition/index.html#overview>>. Acesso em: 29 Mai. 2019.
- JANARTHANAM, S. *Building Your Own Chatbot Using DialogFlow*. 2018. Disponível em: <<https://tutorials.botsfloor.com/building-your-own-chatbot-using-dialogflow-1b6ca92b3d3f>>. Acesso em: 15 Nov. 2018.
- KANG, A. *Understanding the Differences Between Alexa, API.ai, WIT.ai, and LUIS/Cortana*. 2017. Disponível em: <<https://medium.com/@abraham.kang/understanding-the-differences-between-alexa-api-ai-wit-ai-and-luis-cortana-2404ece0977c>>. Acesso em: 28 Mai. 2018.
- LALEK, N. *O que é Processamento de Linguagem Natural (PLN)*. 2018. Disponível em: <<https://tudosobrerobos.com.br/deep-learning/processamento-de-linguagem-natural-pln/>>. Acesso em: 25 Nov. 2018.
- LOFTI, R. *Saúde Pública versus Saúde Privada*. 2016. Disponível em: <<https://www.planodesaude.net/saude-publica-versus-saude-privada/>>. Acesso em: 15 Fev. 2018.
- LUCKOW, D. H.; MELO, A. A. de. *Programação Java para a WEB*. [S.l.]: Novatec Editora, 2010.
- MACORATTI, J. C. Padroes de projeto: O modelo mvc-model view controller. *Disponivel em http://www. macoratti. net/vbn mvc. htm*, 2008.

- MATERIALIZE, G. *Aprenda sobre Material Design e nossa equipe de projeto*. 2018. Disponível em: <https://materializecss.com/about.html>. Acesso em: 02 Mar. 2018.
- MCCARTHY, J. Artificial intelligence, logic and formalizing common sense. In: *Philosophical logic and artificial intelligence*. [S.l.]: Springer, 1989. p. 161–190.
- MENDONÇA, A. P. *Saude publica e saúde privada*. 2015. Disponível em: <https:////economia.estadao.com.br/noticias/geral,sauda-publica-e-sauda-privada,1618631>. Acesso em: 15 Fev. 2018.
- MICROSOFT, C. *TypeScript Language Specification*. 2014. Disponível em: <http://typescriptlang.org>. Acesso em: 27 Mai. 2019.
- MILIOZZI, J. *Chatbots: conheça a história dessa fascinante tecnologia*. 2017. Disponível em: <https://ibramerc.liveuniversity.com/2017/12/15/chatbots-e-a-historia-dessa-fascinante-tecnologia/>. Acesso em: 25 Mai. 2018.
- MONGODBINC. *What is Mongodb*. 2018. Disponível em: <https://www.mongodb.com/what-is-mongodb>. Acesso em: 25 Mai. 2019.
- NODE-RED. *Node-RED*. 2019. Disponível em: <https://nodered.org>. Acesso em: 29 Mai. 2019.
- NODEBR. *Primeiros passos com Express em Node.js*. 2016. Disponível em: <http://nodebr.com/primeiros-passos-com-express-em-node-js/>. Acesso em: 28 mai. 2019.
- PENZE, A. D. *5 vantagens de ter um site responsivo*. 2016. Disponível em: <http://www.penze.com.br/blog/5-vantagens-de-ter-um-site-responsivo/>. Acesso em: 28 mai. 2018.
- PINTO, A. P. et al. Testes de performance utilizando o db4o e mongodb. *e-RAC*, v. 3, n. 1, 2013.
- POLITOWSKI, C.; MARAN, V. Comparaç ao de performance entre postgresql e mongodb. *X Escola Regional de Banco de Dados. SBC*, p. 1–10, 2014.
- PREECE, J. et al. *Interaction design: beyond human-computer interaction*. [S.l.]: John Wiley & Sons, 2015.
- PYLE, D.; JOSE, C. S. An executive's guide to machine learning. *Mckinsey Quarterly*, (3), p. 44–53, 2015.
- RAMOS, A. *O que é MCV?* 2015. Disponível em: <https://tableless.com.br/mvc-afinal-e-o-que/>. Acesso em: 20 fev. 2018.
- REHLING, J. *How Natural Language Processing Helps Uncover Social Media Sentiment*. 2011. Disponível em: <https://mashable.com/2011/11/08/natural-language-processing-social-media/#nmrg5.VWMEqn>. Acesso em: 15 Abr. 2018.
- RICH, E. et al. *Inteligencia artificial*. [S.l.: s.n.], 1994.
- ROGATNEV, N. et al. Responsive web design. Mikkelin ammattikorkeakoulu, 2015.
- SANTOS, C. J. G. d. Tipos de pesquisa. *SANTOS, Antonio Raimundo dos*, 2014.

- SEVERINO, A. J. *Metodologia do trabalho científico*. [S.l.]: Cortez editora, 2017.
- SONI, Y. *Visual Recognition in Android Using IBM Watson*. 2018. Disponível em: [⟨https://heartbeat.fritz.ai/visual-recognition-in-android-using-ibm-watson-9b1fea83e8d⟩](https://heartbeat.fritz.ai/visual-recognition-in-android-using-ibm-watson-9b1fea83e8d). Acesso em: 29 Mai. 2019.
- TEIXEIRA, J. de F. *Inteligência artificial*. [S.l.]: Pia Sociedade de São Paulo-Editora Paulus, 2014.
- WATSON, I. *IBM Watson Assistant*. 2018. Disponível em: [⟨https://www.ibm.com/cloud/watson-assistant/⟩](https://www.ibm.com/cloud/watson-assistant/). Acesso em: 15 Mai. 2018.
- WEIZENBAUM, J. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, ACM, v. 9, n. 1, p. 36–45, 1966.
- WIDMAN, L. E. Sistemas especialistas em medicina, traduzido e adaptado por renato m.e. sabbatini. *Revista Informática Médica*, v. 1, n. 5, 1998.
- WIT.AI, F. *Linguagem Natural para Desenvolvedores*. 2018. Disponível em: [⟨https://wit.ai/⟩](https://wit.ai/). Acesso em: 28 Mai. 2018.

Apêndice A – Levantamento de Requisitos

Tabela 1 – Requisitos Funcionais.

ID	Requisito Funcional	Descrição	Caso de Uso
RF001	Cadastrar no sistema	A aplicação deverá ter tela Login e cadastro do paciente ou médico, cujo os dados estarão atrelados a um banco de dados	Cadastrar Paciente e Cadastrar Médico
RF002	Agendar consulta	A aplicação irá informatizar o atendimento, permitindo o agendamento e cancelamento de procedimentos médicos	Agendar consulta.
RF003	Buscar médicos	O paciente irá fazer uma busca por um determinado médico, de uma determinada especialidade.	Busca médica.
RF004	Buscar consultas agendadas	A aplicação além de permitir o medico possa verificar as consultas que foram agendadas pelos seus pacientes	Buscar consultas agendadas.
RF005	Realiza pré-diagnóstico com ChatBot	A aplicação permite que o paciente faça um pré-diagnóstico com o ChatBot, na qual será iniciado um dialogo entre os dois autores(Paciente e Chatbot)	Solicitar pré-diagnóstico com ChatBot/ Realizar pré-diagnóstico.
RF006	Gerar o relatório(histórico)	A aplicação permitirá emitir relatórios médicos(histórico), uma espécie de prontuário	Gerar o relatório(histórico).
RF007	Consulta as datas e hora	A aplicação permitirá ter acesso a agenda médica com datas e horários disponíveis para agendamento	Consulta as datas e hora.
RF008	Atualizar agenda médica	A aplicação permitirá o médico atualize sua agenda, com datas e horários disponíveis para agendamento	Manter Agenda.
RF009	Gerenciar especialidades	A aplicação permitirá o médico cadastre e gerencie suas especialidades, passando os devidos valores de consulta.	Cadastrar especialidade.
RF010	Gerenciar consultórios	A aplicação permitirá o médico cadastre seu consultório, passando o nome e o endereço.	Cadastrar consultório.

Tabela 2 – Requisitos Não Funcionais.

ID	Descrição
RNF001	A aplicação será Web Responsiva.
RNF002	A aplicação será desenvolvida em Node.js.
RNF003	A aplicação deverá utilizar banco de dados não relacional(NoSql).
RNF004	A aplicação deverá funcionar para o máximo de browser possíveis(Mozilla Firefox, Google Chrome, Safari).
RNF005	A aplicação deverá ter uma Interface amigável.
RNF006	O Chatbot será desenvolvido com o Watson Assistant (Lite).

Apêndice B – Detalhamento do Caso de uso

CASO DE USO: CADASTRAR PACIENTE.

1. DESCRIÇÃO

O caso de uso “Cadastrar Paciente” contará com uma tela Login e cadastro do paciente, cujo os dados estarão atrelados a um banco de dados.

2. ATORES

O Usuário, paciente (poderão atuar sobre este caso de uso)

3. PRÉ-CONDIÇÕES

Os pacientes deverão acessar o sistema para então realizar seu cadastro.

4. FLUXO DE EVENTOS

Este caso de uso se inicia quando o usuário realizar a operação que corresponde ao cadastro e login no sistema.

a) FLUXO PRINCIPAL

- O paciente irá acessar a aplicação.
- O paciente informará seu login e senha na página de autenticação do sistema.

5. PÓS-CONDIÇÕES

É permitido o acesso ao sistema do usuário cadastrado.

CASO DE USO: AGENDAR CONSULTA.

1. DESCRIÇÃO

No caso de uso “Agendar consulta” aplicação irá informatizar o atendimento, permitindo o agendamento e cancelamento de procedimentos médicos.

2. ATORES

Os pacientes (após o cadastro,poderão atuar sobre este caso de uso).

3. PRÉ-CONDIÇÕES

Os pacientes devem estar cadastrados no sistema.

4. FLUXO DE EVENTOS

Este caso de uso se inicia quando o paciente realizar a operação que corresponde a Agendar consulta.

a) FLUXO PRINCIPAL

- O paciente informará seu login e senha na página de autenticação do sistema.
- O paciente clicará no botão para agendar a consulta.
- O paciente irá agendar a consulta.

5. PÓS-CONDIÇÕES

É permitido listar consultas agendas e até cancelar.

CASO DE USO: BUSCA MÉDICO.

1. DESCRIÇÃO

No caso de uso “Busca médico”, o paciente irá realizar uma busca por médicos, informando a qual a especialidade procura e então selecionar o médico desejado.

2. ATORES

Os pacientes (ao entrar na tela de agendamento).

3. PRÉ-CONDIÇÕES

Os pacientes devem estar logado na aplicação e ter clicado para realizar uma agendamento.

4. FLUXO DE EVENTOS

Este caso de uso se inicia quando o paciente realizar a operação que corresponde a Agendar consulta.

a) FLUXO PRINCIPAL

- O paciente informará seu login e senha na página de autenticação do sistema.
- O paciente clicará no botão para agendar a consulta.
- O paciente deverá informar a especialidade que ele procura.
- O paciente deverá selecionar um dos médicos listados.

5. PÓS-CONDIÇÕES

Poderá visualizar as datas e horários disponíveis do médico.

CASO DE USO: BUSCAR CONSULTAS AGENDADAS.

1. **DESCRIÇÃO** O caso de uso “Buscar consultas agendadas” deve permitir o paciente e o médico buscar os dados da consulta médica agendada.

2. ATORES

Os pacientes (após o cadastro, poderão atuar sobre este caso de uso) e os Médicos.

3. PRÉ-CONDIÇÕES

O paciente deve ser cadastrado e está logado no sistema. O paciente deve ter efetuado algum agendamento de consultas.

4. FLUXO DE EVENTOS

Este caso de uso inicia quando o paciente realizar a operação correspondente a Buscar consultas agendadas.

a) FLUXO PRINCIPAL

- O paciente irá logar no sistema.
- O paciente clicará no botão Agendar consulta.
- O paciente então poderá clicar no botão que lista as consultas agendadas.
- O médico irá poder visualizar as consultas marcadas na sua agenda.

5. PÓS-CONDIÇÕES

É permitido realizar cancelamentos ou imprimir os dados da consulta.

CASO DE USO: REALIZA PRÉ-DIAGNÓSTICO COM CHATBOT.

1. **DESCRIÇÃO** O caso de uso “Realiza pré-diagnóstico com ChatBot” permite que o paciente faça um pré-diagnóstico com o ChatBot.

2. ATORES

Os pacientes (por intermédio do Chatbot, também poderão atuar sobre este caso de uso).

3. PRÉ-CONDIÇÕES

O paciente deve estar acessando o sistema.

4. FLUXO DE EVENTOS

Este caso de uso inicia quando o paciente realizar a operação correspondente a Realiza pré-diagnóstico com ChatBot.

a) FLUXO PRINCIPAL

- O paciente informará o login e senha na página de autenticação do sistema.
- O paciente clicará no ícone do Chat, em que abrirá o bate papo.

5. PÓS-CONDIÇÕES

É permitido realizar agendamento de consultas após a avaliação. Também deverá ser permitido avaliar a avaliação do bot.

CASO DE USO: GERAR O RELATÓRIO(HISTÓRICO).

1. **DESCRIÇÃO** O caso de uso “Gerar o relatório(histórico).” permitirá emitir relatórios médicos(histórico), uma espécie de prontuário.

2. ATORES

Os pacientes (poderão atuar sobre este caso de uso).

3. PRÉ-CONDIÇÕES

O paciente deve estar logado no sistema e ter realizado algumas consultas.

4. FLUXO DE EVENTOS

Este caso de uso inicia quando o paciente realizar a operação correspondente a Gerar o relatório(histórico).

a) FLUXO PRINCIPAL

- O paciente irá logar no sistema.
- O paciente irá clicar no botão para direcionar à página de relatório.

5. PÓS-CONDIÇÕES

É permitido realizar o download do relatório ou imprimi-lo.

CASO DE USO: CONSULTA AS DATAS E HORA.

1. DESCRIÇÃO O caso de uso “Consulta as datas e hora” permitirá ter acesso a agenda médica com datas e horários disponíveis para agendamento.

2. ATORES

Os pacientes (poderão atuar sobre este caso de uso).

3. PRÉ-CONDIÇÕES

O paciente deve estar logado no sistema e abrir a pagina de agendamento.

4. FLUXO DE EVENTOS

Este caso de uso inicia quando o paciente realizar a operação correspondente a Consulta as datas e hora.

a) FLUXO PRINCIPAL

- O paciente informará o login e senha na página de autenticação do sistema.
- O paciente abrirá a tela de agendamento.
- O paciente, enfim terá acesso à agenda médica

5. PÓS-CONDIÇÕES

É permitido realizar o agendamento da consulta.

CASO DE USO: CADASTRAR ESPECIALIDADE.

1. DESCRIÇÃO O caso de uso “Cadastrar especialidade” permitirá que o médico gerencie, cadastrando suas especialidades e os valores de consulta.

2. ATORES

Os médicos (poderão atuar sobre este caso de uso).

3. PRÉ-CONDIÇÕES

O médico deve estar logado no sistema e abrir a pagina de especialidades.

4. FLUXO DE EVENTOS

Este caso de uso inicia quando o médico realizar a operação correspondente a cadastrar especialidades.

a) FLUXO PRINCIPAL

- O médico informará o login e senha na página de autenticação do sistema.
- O médico abrirá a tela de cadastro de especialidade.

- O médico preencherá o formulário de cadastro.

5. PÓS-CONDIÇÕES

Tais especialidades serão utilizadas como filtro para agendamento.

CASO DE USO: CADASTRAR CONSULTÓRIO.

1. **DESCRIÇÃO** O caso de uso “Cadastrar consultório” permitirá que o médico cadastre seu consultório, com nome e endereço.

2. ATORES

Os médicos (poderão atuar sobre este caso de uso).

3. PRÉ-CONDIÇÕES

O médico deve estar logado no sistema e abrir a pagina de cadastro de consultório.

4. FLUXO DE EVENTOS

Este caso de uso inicia quando o médico realizar a operação correspondente a Cadastro de consultório.

a) FLUXO PRINCIPAL

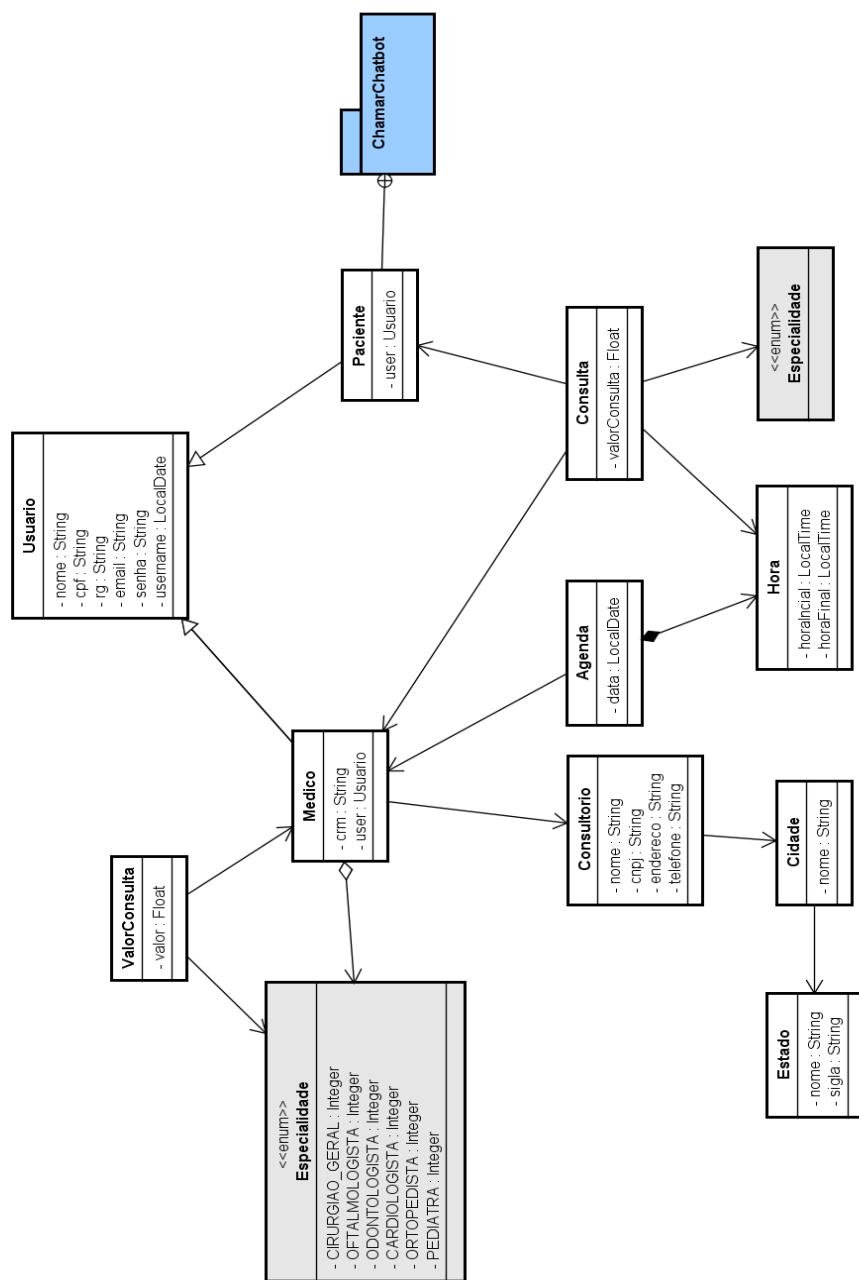
- O médico informará o login e senha na página de autenticação do sistema.
- O médico abrirá a tela de cadastro de consultório.
- O médico, enfim terá seu consultório cadastrado.

5. PÓS-CONDIÇÕES

É permitido realizar o agendamento da consulta.

Apêndice C – Diagrama de Classe

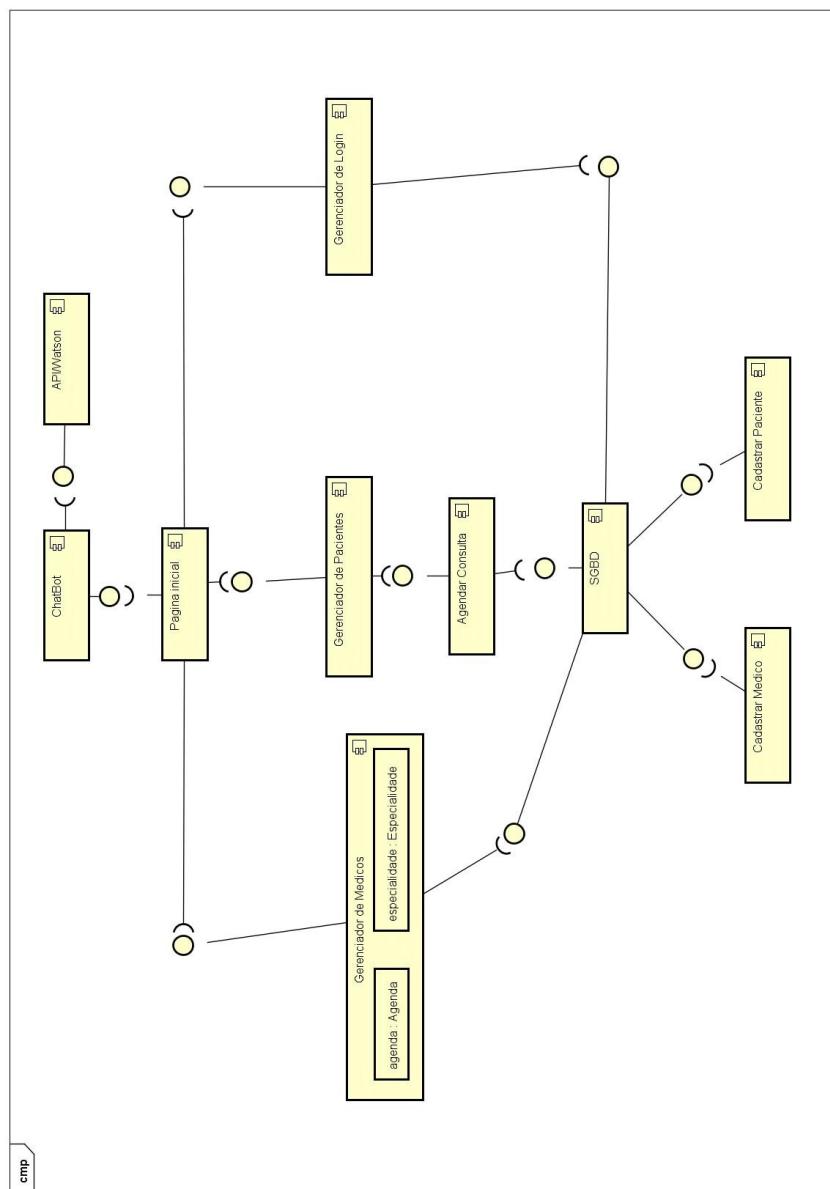
Figura 32 – Diagrama de classe



Fonte: do Autor, 2019

Apêndice D – Diagrama de Componentes

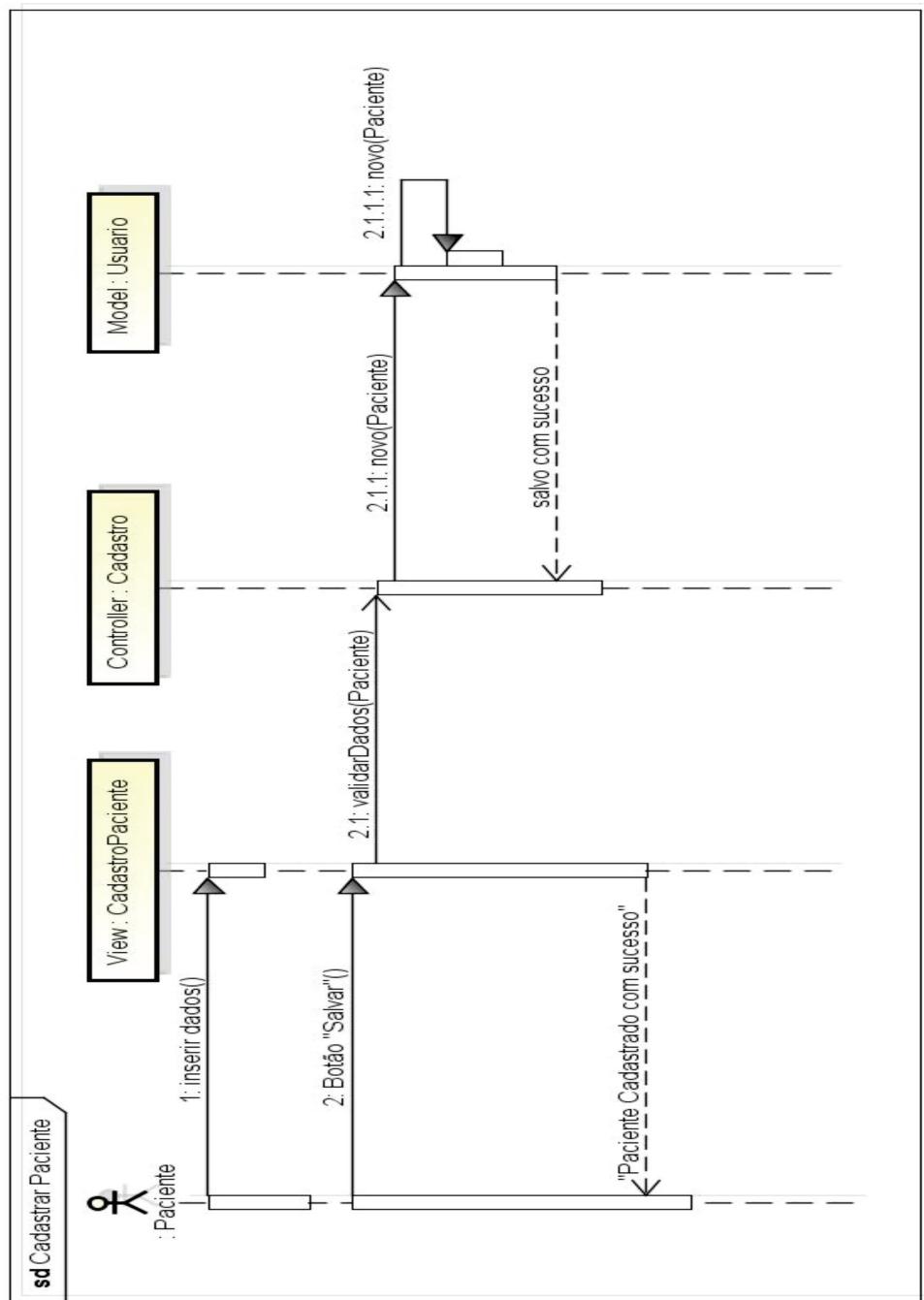
Figura 33 – Diagrama de Componentes



Fonte: do Autor, 2019

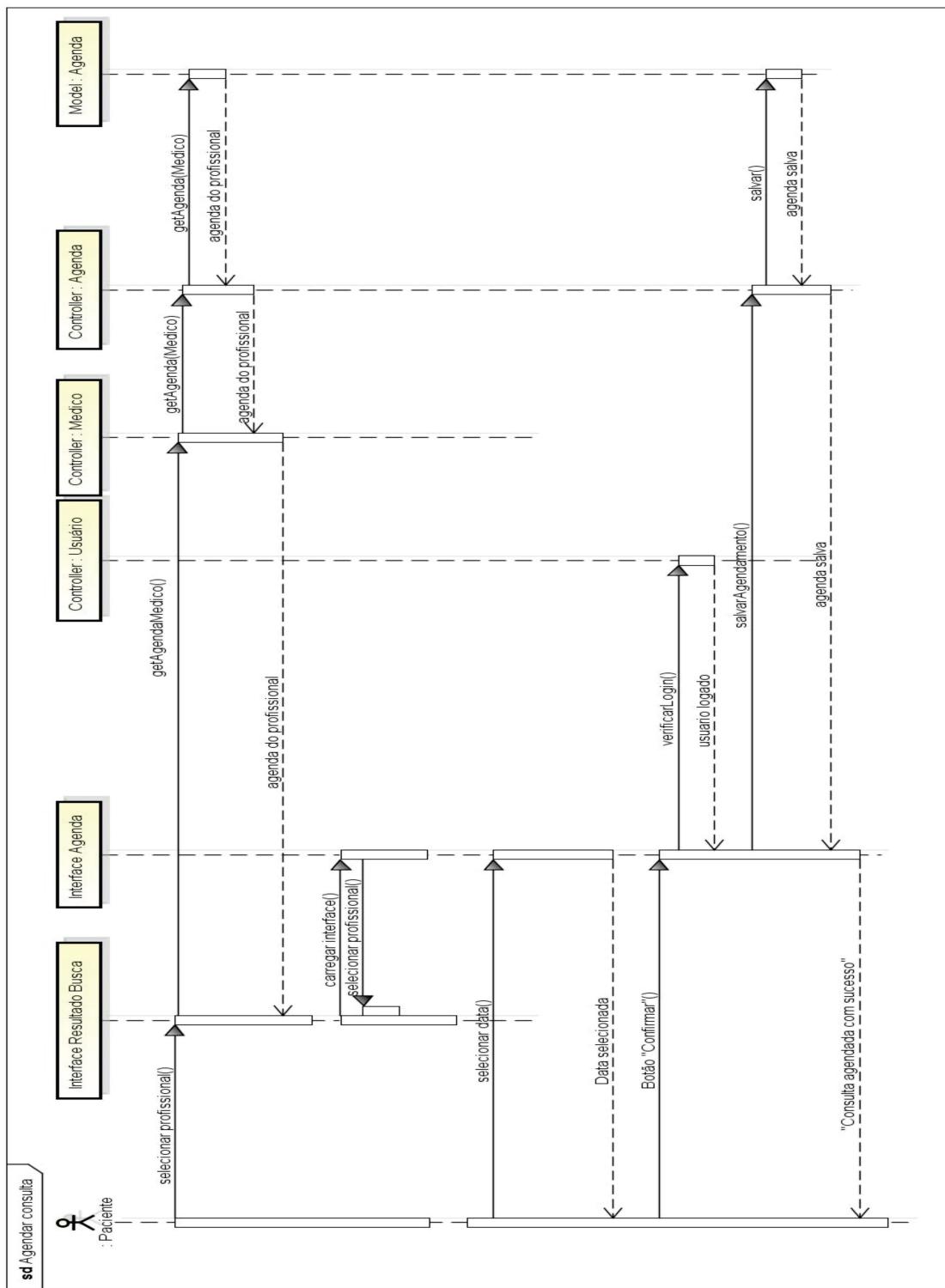
Apêndice E – Diagramas de Sequencia

Figura 34 – Diagrama de sequencia (Cadastrar Paciente)



Fonte: do Autor, 2018

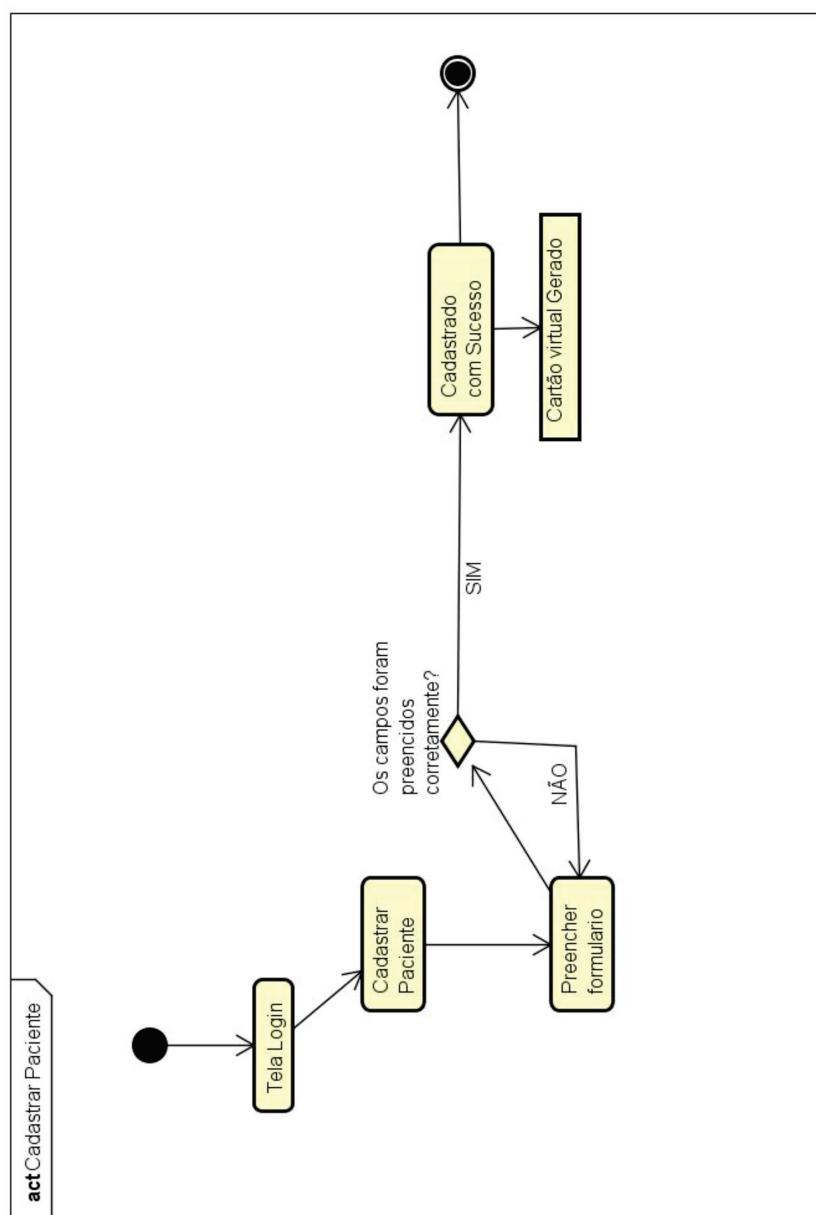
Figura 35 – Diagrama de sequencia (Agendar Consulta)



Fonte: do Autor, 2018

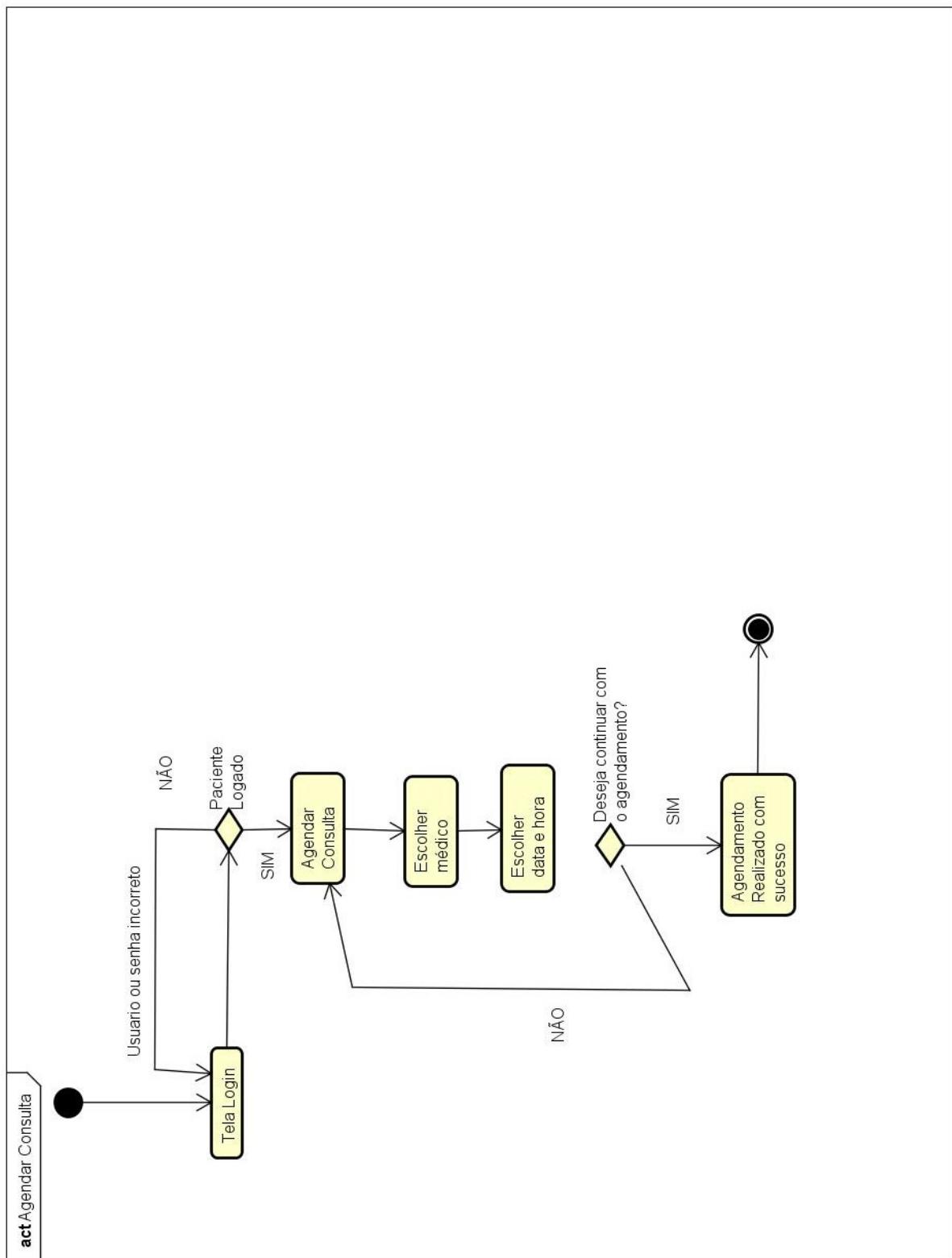
Apêndice F – Diagramas de Atividade

Figura 36 – Diagrama de Atividade (Cadastrar Paciente)



Fonte: do Autor, 2018

Figura 37 – Diagrama de Atividade (Agendar Consulta)



Fonte: do Autor, 2018