SENAI CIMATEC

PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL E TECNOLOGIA INDUSTRIAL

Mestrado em Modelagem Computacional e Tecnologia Industrial

Dissertação de mestrado

Modelo de comunicação alternativa e aumentativa para pacientes com dificuldade de fala em leito hospitalar

Apresentada por: Rafael Levi Batista Costa
Orientador: Dr. Valter de Senna

Maio de 2019

Rafael Levi Batista Costa

Modelo de comunicação alternativa e aumentativa para pacientes com dificuldade de fala em leito hospitalar

Dissertação de mestrado apresentado ao Programa de Pós-graduação em Modelagem Computacional e Tecnologia Industrial, Curso de Mestrado em Modelagem Computacional e Tecnologia Industrial do SENAI CIMATEC, como requisito parcial para a obtenção do título de **Mestre em Modelagem Computacional e Tecnologia Industrial**.

Área de conhecimento: Interdisciplinar

Orientador: Dr. Valter de Senna
SENAI CIMATEC

Salvador
SENAI CIMATEC
2019

Ficha catalográfica elaborada pela Biblioteca do Centro Universitário SENAI CIMATEC

C837m Costa, Rafael Levi Batista

Modelo de comunicação alternativa e aumentativa para pacientes com dificuldade de fala em leito hospitalar / Rafael Levi Batista Costa. – Salvador, 2019.

58 f. : il. color.

Orientador: Prof. Dr. Valter de Senna.
Coorientador: Prof. Dr. Roberto Luiz Souza Monteiro.

Dissertação (Mestrado em Modelagem Computacional e Tecnologia Industrial) – Programa de Pós-Graduação, Centro Universitário SENAI CIMATEC, Salvador, 2019.
Inclui referências.

1. Comunicação aumentativa. 2. Comunicação alternativa. 3. Augmentative and alternative communication (AAC). I. Centro Universitário SENAI CIMATEC. II. Senna, Valter de. III. Monteiro, Roberto Luiz Souza. IV. Título.

CDD: 362.19

NDI - 02

Nota sobre o estilo do PPGMCTI

Esta dissertação de mestrado foi elaborada considerando as normas de estilo (i.e. estéticas e estruturais) propostas aprovadas pelo colegiado do Programa de Pós-graduação em Modelagem Computacional e Tecnologia Industrial e estão disponíveis em formato eletrônico (*download* na Página Web http://ead.fieb.org.br/portal_faculdades/dissertacoes-e-teses-mcti.html ou solicitação via e-mail à secretaria do programa) e em formato impresso somente para consulta.

Ressalta-se que o formato proposto considera diversos itens das normas da Associação Brasileira de Normas Técnicas (ABNT), entretanto opta-se, em alguns aspectos, seguir um estilo próprio elaborado e amadurecido pelos professores do programa de pós-graduação supracitado.

SENAI CIMATEC

Programa de Pós-graduação em Modelagem Computacional e Tecnologia Industrial

Mestrado em Modelagem Computacional e Tecnologia Industrial

A Banca Examinadora, constituída pelos professores abaixo listados, leram e recomendam a aprovação do Dissertação de mestrado, intitulada "Modelo de comunicação alternativa e aumentativa para pacientes com dificuldade de fala em leito hospitalar", apresentada no dia 01 de Maio de 2019, como requisito parcial para a obtenção do título de Mestre em **Modelagem Computacional e Tecnologia Industrial**.

Orientador:

Prof°. Dr. Valter de Senna
SENAI CIMATEC

Co-Orientador:

Prof. Dr. Roberto Monteiro
SENAI CIMATEC

Membro interno da Banca:

Profa. Dra. Ingrid Winkler
SENAI CIMATEC

Membro externo da Banca:

Prof. Dr. José Garcia Vivas Miranda
UFBa

Agradecimentos

Agradeço primeiramente aos meus familiares, principalmente a minha mãe por todo apoio e dedicação nos momentos difíceis. A minha esposa por estar sempre ao meu lado me incentivando a continuar e não deixar desistir. Aos meus amigos que me ajudaram durante o percurso e nos momentos de festividades que não estive presente. Ao meu orientador pela paciência e motivação.

Salvador, Brasil
28 de Maio de 2019

Rafael Levi Batista Costa

Resumo

Comunicação alternativa e aumentativa, do inglês Augmentative and Alternative Communication (AAC) é uma área de estudo que investiga meios de comunicação alternativos à fala para pessoas com dificuldades de oralizar suas necessidades e sentimentos. Foi proposto um sistema com solução de software e hardware para acoplar em televisões de leitos hospitalares com foco em baixo custo e acessibilidade. Para isso foi utilizado UML para modelagem e metodologia de desenvolvimento em MVC (*Model-View-Controller*). Para atender à necessidade do baixo custo, visando a sustentabilidade do sistema, em questões de hardware, escolhemos utilizar o raspberry pi, modelo 0 W para embarcar nosso projeto. Esse microcontrolador nos permitirá trabalhar em um nível de abstração sob um sistema operacional, escolhemos o Linux por ser uma opção de custo zero, ainda podendo utilizar a programação em Python. Como contribuição do trabalho ficam observações sobre o desenvolvimento de uma solução tecnológica para a área de AAC, do ponto de vista de uso de tecnologias e limitações das bibliotecas de sistemas de comunicação alternativo.
Palavras-chave: Comunicação aumentativa, Comunicação alternativa.

Abstract

Alternative and augmentative communication (AAC) is an area of study that investigates alternative means of communication to speech for people with difficulties of oralizing their needs and feelings. It was proposed a system with software and hardware solution to couple in televisions of hospital beds with focus on low cost and accessibility. For this, UML was used for modeling and development methodology in MVC (*Model-View-Controller*). In order to meet the need of low cost, aiming the sustainability of the system, in matters of hardware, we chose to use raspberry pi, model 0 W to ship our project. This microcontroller will allow us to work at an abstraction level under an operating system, we chose Linux as a zero-cost option, still able to use Python programming. As contribution of the work are observations on the development of a technology solution for the area of AAC from the point of view of use of technologies and limitations of libraries of alternative communication systems.

Keywords: Augmentative communication, alternative communication.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Siglas

PPGMCTI ..	Programa de Pós-graduação em Modelagem Computacional e Tecnologia Industrial
WWW	World Wide Web
UTI	Unidade de Tratamento Intensivo
UML	Unified Modeling Language

Capítulo Um

Introdução

1.1 Definição do problema

A afasia é um distúrbio na comunicação, que impõe restrições nos contextos social e familiar (MARTINS, 2016) . Este trabalho aborda uma situação hospitalar em que o paciente apresenta dificuldades na fala, parcial ou totalmente. Com base em estudos, podem ser possíveis causas de um distúrbio de afasia, em específico na comunicação oral, situações como traumatismo crânio-encefálico e acidente vascular encefálico (GONÇALVES, 2008). Além de tais motivos, procedimentos hospitalares podem trazer as mesmas consequências, como a traqueostomia, que, além da deglutição, pode trazer impactos na respiração e na fala (BARROS; PORTAS; QUEIJA, 2009). Essa limitação da fala pode ser causada também por uma cirurgia específica, a laringectomia, que de acordo com (JUNIOR; NASCIMENTO, 2014) se trata da remoção da cartilagem da laringe.

Esta pesquisa focará no período em que o paciente, independente da causa, está em leito hospitalar, impossibilitado de se comunicar oralmente. Além dos danos físicos e a impossibilidade de comunicação oral, tais pacientes passam por dificuldades psicológicas, conforme é abordado no trabalho de (JUNIOR; NASCIMENTO, 2014).

Ainda dentro do escopo desta pesquisa, trata-se de um ambiente com pacientes em situação de internação hospitalar com afasia de comunicação oral, em qualquer grau, os quais possuam movimento, ao menos, de uma mão. A proposta desta pesquisa é escalável para interfaces com pacientes completamente paralisados fisicamente. Entretanto, o escopo está fechado para pacientes na situação citada anteriormente. A seguir, é exposto um panorama com dados do Brasil sobre leitos hospitalares, incluindo as Unidades de Tratamento Intensivo (UTI) que é o provável local onde o afásico esteja internado.

De acordo com a Associação de Medicina Intensiva do Brasil, em seu último censo de 2016, o país possui 452.755 leitos hospitalares sem UTI e 41.741 com UTI. Os que possuem leito UTI para adultos são a maioria com 66,4% do total. No Nordeste, são 396 estabelecimentos com leitos UTI. A Bahia possui 4,5% desse total do Nordeste. Além desse segmento de estudo, essa aplicação propõe uma solução de baixo custo para comunicação alternativa ao público pré-definido. É possível expressar em números, a quantidade de leitos públicos presentes no estado da Bahia. Dos 89 leitos de UTI existentes, apenas 34 são públicos, sendo que 11 estão na cidade de Salvador e 23 se encontram no interior. Os demais são privados.

Em conformidade com os dados acima citados, evidencia-se um panorama da capacidade de abrigar pacientes em UTIs, principalmente na Bahia e em leitos públicos. Consequentemente, esta pesquisa trata-se de um protótipo de sistema pensado para esse público, pacientes que estão em leitos hospitalares com afasia de comunicação oral, os quais necessitam de uma solução de baixo custo para oralizar suas necessidades e é desse ponto de vista que a mesma está posicionada. Concluindo a análise de saldo de leitos de UTI, há no nordeste cerca de 1,44 vagas a cada 10 mil habitantes, é o segundo pior índice do país contra uma média nacional de 2,19 a cada 10 mil.

O Cadastro Nacional de Estabelecimentos de Saúde (CNES) categoriza os profissionais dessa área de medicina intensiva como enfermeiro de terapia intensiva, médico em medicina intensiva e técnico de enfermagem em terapia intensiva. No Brasil há cerca de 18.652 profissionais dentro dessas categorias. No nordeste há cerca de 2.078 médicos nessa categoria, sendo 527 na Bahia. Desse modo, pode-se concluir que há poucos profissionais capacitados para tratar esse público alvo, além do atendimento na saúde pública ser precário e com pouco uso de tecnologias assistivas a esse profissional.

A partir dos números apresentados anteriormente, é possível compreender a importância do desenvolvimento de pesquisas sobre esse cenário. A comunicação alternativa e aumentativa, do inglês AAC (*Augmentative and Alternative Communication*), aliado ao uso de tecnologias podem auxiliar os profissionais e familiares a se comunicarem com os pacientes com dificuldade de fala em ambiente hospitalar. Para isso o trabalho foca em um modelo de baixo custo, para atender a esse público.

1.2 Objetivo

O objetivo deste trabalho é desenvolver um sistema de hardware e software que atenda a pacientes com dificuldade de fala, em leitos hospitalares, focando no baixo custo, fácil reposição de peças e que seja de fácil usabilidade por parte dos pacientes. Através desse sistema o usuário deve ser capaz de formular frases e com o uso de um sintetizador de voz, expressar suas necessidades em ambiente hospitalar.

1.2.1 Objetivos específicos

- Estabelecer os requisitos do projeto
- Definir as tecnologias de hardware e software baseado nos requisitos.
- Projetar o modelo da solução em Linguagem de Modelagem Unificada (UML).

- Desenvolver um protótipo funcional.

1.3 Importância da pesquisa

De acordo com Massaud (2018), neurologista do Instituto Albert Einstein "Não há dados estatísticos específicos sobre a incidência das afasias na população brasileira, haja visto, que podem ser consequência de uma série de doenças neurológicas."(MASSAUD, 2018).

Considerando, primeiramente, os dados do SUS no ano de 2008, os quais constataram 200 mil internação por AVC, de acordo com Abramczuk & Villela (2009), e baseando-se em Martins (2016) que indica a implicação de afasia em 25% dos casos de AVC, é possivel estimar 50 mil casos de afásicos no Brasil, no ano de 2008. De acordo com o censo IBGE (2010), a proporção de nordestinos em relação à população do país é de 27,83%. Então pode-se selecionar 13.915 afásicos no nordeste. Ainda proporcionalmente, a Bahia possui cerca de 3.676 afásicos, fundamentado apenas na proporção da população e ainda baseado no mesmo critério, tem-se cerca de 702 pacientes com afasia no SUS por ano em Salvador, baseado apenas nas causas por AVC, na rede pública e em dados de 2008. É possível concluir, então, que esse número é superior aos 702 pacientes em Salvador por causas totais.

Com esse público mensurado, é possível ter uma noção da dimensão populacional ao qual esse tema está inserido, sendo o foco desse pesquisa na solução tecnológica, cabe a necessidade de institucionalizar os dados de uso da ferramenta proposta. É possível utilizar os dados gerados a partir do uso desse sistema para cruzar com informações de entrada do hospital e assim entender melhor o comportamento e necessidades dos pacientes com afasia, que tiverem contato com essa ferramenta. Esses dados trariam grande substrato para pesquisas futuras e possíveis constatações de outras pesquisas e métodos.

1.4 Limites e limitações

O escopo está limitado a pacientes com afasia e que necessitam de algum meio de comunicação como uma prancha de símbolos para se comunicar. Na pesquisa estão contemplados meios de comunicação para suprir essa necessidade, com enfoque em meios tecnológicos. Sobre o protótipo, nesse primeiro modelo é comtemplado um cenário ao qual o paciente consegue movimento de ao menos uma mão para uso via controle sem fio. Além disso, o sistema necessita de ao menos uma televisão, mesmo que compartilhada, no ambiente hospitalar ao qual se encontram os pacientes citados.

1.5 Organização da Dissertação de mestrado

Esta dissertação apresenta 6 capítulos e está estruturado da seguinte forma:

- **Capítulo 1 - Introdução**: Contextualiza o âmbito, no qual a pesquisa proposta está inserida. Apresenta, portanto, a definição do problema, objetivos e justificativas da pesquisa e como esta dissertação de mestrado está estruturada;
- **Capítulo 2 - Fundamentação Teórica**: Apresentação da concepção teórica, avaliando as referências relacionadas ao tema dos autores clássicos que fundamentam a pesquisa aos pesquisadores que estão envolvidos com o conhecimento científico relacionado;
- **Capítulo 3 - Trabalho teórico e prático**: Nesse capítulo apresenta o modelo proposto;
- **Capítulo 4 - Aspectos tecnológicos**: Nesse capítulo é apresentado as tecnologias envolvidas no projeto;
- **Capítulo 5 - Resultados e discussões**: Nesse capítulo apresenta-se os resultados do projeto;
- **Capítulo 6 - Considerações Finais**: Apresenta as conclusões, contribuições e algumas sugestões de atividades de pesquisa a serem desenvolvidas no futuro.

Capítulo Dois

Fundamentação teórica

> Quanto maior é a rapidez de transformação de uma sociedade, mais temporárias sáo as necessidades individuais. Essas flutuações tornam ainda mais acelerado o senso de turbilhão da sociedade.
>
> (Alvin Toffler)

2.1 Sistema AAC

Este capítulo do trabalho se trata do levantamento bibliográfico de revistas, eventos, instituições, livros, artigos, empresas e produtos relacionados ao tema de AAC. Com isso, pretende-se ampliar a visão do tema para incluir aspectos tecnológicos e ferramentas que utilizem os conceitos de AAC na vida de pacientes que necessitam se comunicar, em ambiente hospitalar. Os Levantamentos bibliográficos, como o feito por Cesa & Mota (2015), buscam aspectos de psicologia, pedagogia, fisioterapia e áreas afins, pouco citando ferramentas tecnológicas.

Além disso, este trabalho almeja expor além de trabalhos científicos dos últimos cinco anos, outros também relevantes há mais de cinco anos de publicação.

Como método, fizemos uma pesquisa nos bancos de dados do Google acadêmico, filtrando os artigos e livros que compõe a base científica da AAC. Os bancos de dados procurados, além do já citado foram, Scielo, Science Direct, Portal Capes de periódicos e de Teses e Dissertações. Buscamos em português por descritores, Comunicação Alternativa e comunicação alternativa e aumentativa e em inglês por Augment and alternative communication. Com essa pesquisa, obteve-se diversos dados, os quais são explicitados na Tabela 2.1. Após pesquisar sobre bases científicas, buscou-se empresas e produtos tecnológicos que utilizam como base teórica o tema investigado. Os resultados serão apresentados a seguir, no desenvolvimento do levantamento bibliográfico.

Tabela 2.1: Pesquisa por descritores
Fonte: Próprio autor

Palavras chave	Google acadêmico	Scielo	Elsevier (science direct)	Portal Capes (Periódicos)	Portal Capes (Teses e dissertações)
Comunicação alternativa	672.000	176	1.262	2.723	86.777
Augment and alternative communication	252.000	0	44.654	16	268.578
comunicação alternativa e aumentativa	3.410	7	0	9	984.137

Dos resultados pesquisados foram lidos 3 livros, que foram selecionados por cruzarem as informações sobre sistemas de símbolos de comunicação alternativa, estratégias visuais de comunicação e o uso de tecnologias para potencializar essas estratégias.

2.1.1 Sistemas de comunicação

Sobre os resultados dessa pesquisa, trazemos Spears & Turner (2010). A ideia desse livro é trazer um guia para uso de símbolos para comunicação alternativa e aprendizagem de crianças com autismo, trazendo estratégias visuais que apoiem o ensino em casa e na escola. Na primeira parte do livro os autores trazem métodos de uso, explicando o que chamam de Dispositivos de Geração de Voz (SGDs), Sistemas de Comunicação de Troca de Imagem (PECS) e os Sistemas de linguagem simbólica. Na segunda parte do livro, as estratégias de suporte para ensino, como suportes visuais, Social Stories e ambientes estruturados. Os autores Carol L. Spears e Vicki L. Turner são pesquisadores do estado de Ohio, nos Estados Unidos da América (EUA), são patologistas de fala e linguagem experientes e com especialização em serviços de comunicação alternativa / aumentativa (AAC) e estratégias visuais para pessoas com autismo e distúrbios de comunicação. Carol Spears é professora assistente clínica no Programa de fala e audição da Universidade do estado de Cleveland.

Foi encontrado também durante a pesquisa Smith (2005). O livro aborda sobre alfabetização e comunicação aumentativa e alternativa, o autor é pesquisador da Escola de Estudos Clínicos de Fala e Língua Trinity College Dublin, na Irlanda. Nos primeiros capítulos é abordado sobre o processo de leitura e escrita e sobre alfabetização. A partir do capítulo 4 têm-se exemplos de casos, fatores a levar em consideração no uso de comunicação alternativa e aumentativa. O livro ainda aborda sobre avaliações, práticas de avaliações, apresentando modelos possíveis, intervenções e apresenta uma prática chamada de SCRAWL. Neste livro o capítulo que mais se relaciona com o este trabalho é o nono, que aborda o papel da tecnologia nesse processo de alfabetização e comunicação alternativa e aumentativa.

Outro livro importante encontrado nesse processo de pesquisa é o Schlosser (2003). O autor é professor da Universidade do Nordeste, na cidade de Boston em Massachusetts, departamento de Fonoaudiologia e Audiologia. O livro aborda em seus primeiros capítulos sobre Eficácia e medição de resultados na Comunicação Aumentativa e Alternativa, Validação e eficácia de estudos na área. Schlosser apresenta modelos de intervenções utilizando a ferramenta e ensina como comparar métodos diferentes de AAC, ainda aborda sobre evidências baseadas em práticas e estratégias para iniciar uma comunicação desse tipo, ainda há um capítulo apenas para discutir os símbolos a serem utilizados. Os últimos capítulos apresentam os efeitos do uso dessa comunicação, demonstra a eficácia do uso de AAC em indivíduos com afasia grave crônica.

2.1.2 Mercado

Quanto à empresas e produtos tecnológicos que se apoiam sobre o conceito de AAC, listaremos empresas que vendam produtos anunciados com a catergoria comunicação alternativa, que divulgam seus produtos pela internet e tenham informações técnicas sobre os mesmos.

A empresa Liberator, criada em 1991, operam em Ohio (EUA), no Reino Unido com suporte na Alemanha e Austrália. Seu diretor é Ian Thompson e eles fornecem soluções em AAC, como treinamentos, eventos e produtos. Em seu site é possível ver os produtos, que de acordo com eles, são destinados para todos os orçamentos e necessidades. Há uma linha de alta tecnologia com os produtos Accent, que têm uma duração de bateria de 13 a 15 horas, o Accent 1000 (Figura 2.1), com vários tamanhos de tela de 10 a 14 polegadas (Accent Open), alguns com resistência à água, o Liberator Rugged 7 (Figura 2.2), podendo ter plataforma android 12.2, o NOVA chat 12 (Figura 2.3). Esses produtos são digitais e as figuras aparecem na tela para seleção do usuário. Eles possuem também uma linha para quem não pode selecionar as imagens manualmente, os Sistemas EyeGaze. Os produtos variam de € 3.995,00 (Accent 800) a € 7.490,00 (Accent 1400 com NuEye EyeGaze Bundle).

Figura 2.1: Accent 1000
Fonte: <https://www.liberator.co.uk/accent-1000>

Figura 2.2: Liberator Rugged 7 (7")
Fonte: <https://www.liberator.co.uk>

Figura 2.3: NOVA chat 12
Fonte: <https://saltillo.com/products>

A empresa Prentom aparece entre os resultados. Seus produtos são similares a empresa anterior, sendo que esta foi fundada em 1966, também situada em Ohio. Os produtos variam entre $6.000,00 (Accent 800) e $ 7.995,00 (Accent 1400). Essa empresa utiliza os sistemas de linguagem Unity, LAMP Words for Life®, CoreScanner™, UNIDAD®, Essence® e WordPower™. O tópico 2.1.3 esclarece alguns pontos sobre sistemas de linguagens.

Outra companhia é a Saltillo, localizada em Ohio. Seus produtos são similares aos demais, e os valores variam entre $2,995.00 (Nova CHAT 5) e $6,395.00 (FUSION 10- DPlus - IVONA). Nenhum dos produtos citados acima possuem suporte para lingua portuguesa. Foi notado que a maioria das empresas possuem sede em Ohio, onde há grandes núcleos de pesquisa.

Na pesquisa por empresas brasileiras, foi encontrado a Click Tecnologias Assistivas, fundada em 1997, em Porto Alegre - RS. Eles vendem apenas mouses, teclados e acionadores que variam entre R$ 140,00 (Teclado TClik Padrão) e R$ 7.420,00 o acionador pela boca INTEGRAMOUSE PLUS (Figura 2.4). Entretando, os produtos essa empresa são periféricos que devem ser associados a um computador.

Figura 2.4: INTEGRAMOUSE PLUS
Fonte: <https://www.integramouse.com/en/home/>

Em âmbito nacional, foi encontrado a Loja Civiam, fundada em 2008 em São Paulo - SP. Seus produtos variam entre R$2.750,00 (Tablet Acessível) e R$4.200,00 (Vocalizador S32 Tobii - Dispositivo para Comunicação Alternativa e Ampliada).

O objetivo dessa pesquisa por produtos tecnológicos relacionados com o tema é obter um panorama de valores e custo benefício desses produtos. Foi identificado que não há produtos no mercado com baixo custo quando se trata de uma solução de hardware conjunto a um software. As pesquisas em livros resultaram na identificação dos sistemas de linguagens alternativos que já existem e são utilizados como base para a construção das soluções apresentadas. Por meio delas, identificamos três principais sistemas de linguagens utilizados nos produtos apresentados anteriormente.

2.1.3 Sistemas de Linguagens

Os sistemas de linguagens, possuem uma quantidade específica de figuras, devemos levar em consideração o ano de criação do sistema, a região de origem e contexto ao qual foi criado. Para isso, foi feito um levantamento dos principais sistemas de linguagem

alternativa de acordo com a Tabela 2.2.

Tabela 2.2: Sistemas de Linguagens
Fonte: Próprio autor

Sistema	**Data criação**	**Nº de símbolos**	**Autor(es)**	**País**
PIC	1980	11.500	Roxanna Mayer Johnson	EUA
BlisSymbols	1949	900	Charles K. Bliss	Europa
PCS	1987	37000	Mayer-Johnson	EUA

Esses sistemas de linguagens possuem variações e adaptações para diversos países, sendo sempre levado em consideração a construção da linguagem do local. É importante compreender que para a construção de uma ferramenta de apoio à comunicação alternativa é interessante o uso e adoção de um sistema de linguagem, isso trás embasamento teórico e mais consistência para a aplicação, mas por outro lado também aumenta a complexidade e possivelmente o custo e tempo para desenvolvimento desta. Por esses motivos a aplicação proposta não adota nenhum sistema de linguagem, mas levaremos em consideração no modelo UML proposto.

2.1.4 Conclusões sobre AAC

Comunicação alternativa e aumentativa é uma área de estudo que investiga meios de comunicação alternativos à fala para pessoas com dificuldades de oralizar suas necessidades e sentimentos. Para Cesa & Mota (2015 apud ASSOCIATION *et al.*, 2002) é um conjunto de procedimentos e processos que vizam maximizar a comunicação, complementando ou substituindo a fala e/ou a escrita. Essa área é desenvolvida no ramo da fonoaudiologia, a própria Carla Ciceri Cesa supracitada, que é fonoaudióloga, Doutora em Distúrbios da Comunicação Humana pela UFSM, cita a Associação Americana de Fonoaudiologia, em inglês American Speech-Language-Hearing Association(ASHA) reafirmando a forte ligação entre essa área com a AAC. Assim, o uso da tecnologia nessa área é importante para potencializar a oportunidade de oralidade, existindo uma necessidade de desenvolver pesquisa com tecnologias para AAC. De acordo com Sameshima & Deliberato (2009) "As oportunidades de integração e interação social para as pessoas com comprometimento da linguagem oral vêm crescendo cada dia mais, por conta das novas tecnologias", mesmo assim temos poucas opções financeiramente acessíveis.

2.2 Padrões de projeto

A seguir, será apresentado os conceitos utilizados na criação desse projeto, como padrões de projeto, arquitetura de software e técnicas de desenvolvimento.

Padrões de projetos são soluções expressas em termos de objetos e interfaces. São tipos de padrões para um problema num determinado contexto (ERICH *et al.*, 2000, p. 19). Utilizou-se alguns padrões em neste trabalho, os quais serão explicitados cada um e evidenciar onde foram aplicados no projeto de back-end.

2.2.1 Value Object

Na classe User, atributo ID que é outra classe com um unico atributo do tipo primitivo. Vide códigos 2.1 e 2.2 abaixo.

```
1  package br.org.fieb.senai.aac;
2
3  public interface User {
4    Id id();
5    Name name();
6
7    void name(Name name);
8    Image image();
9    void image(Image image);
10   CategoryCollection categories();
11 }
```

Código 2.1: Classe User

```
1  package br.org.fieb.senai.aac;
2
3  import lombok.Value;
4  import lombok.experimental.Accessors;
5
6  @Value(staticConstructor = "of")
7  @Accessors(fluent = true, chain = false)
8  public class Id {
9
10   private Long value;
11
12   @Override
13   public String toString() {
14     return value.toString();
15   }
```

```
16 }
```

Código 2.2: Classe Id

2.3 MVC

Arquitetura de software Model, View, Controller (MVC) é utilizada, nesse projeto, dentro do contexto de aplicações web com Java. Essa arquitetura é feita pensando a aplicação em camadas, dividindo a aplicação nas camadas de interface com o ususário, persistência e camadas intermediárias. Em nosso caso utilizou-se um webservice para intermediar a interface com o usuário, que foi chamada de projeto front-end e a camada de back-end que ainda sofre sub-divisões. Como trata-se de Spring MVC, considera-se as seguintes camadas, interface com usuário, web, serviço, DOM (*Domain Object Model*) e persistência, que é a camada mais próxima do banco de dados (LADD *et al.*, 2006, p. 22).

2.4 Webservice REST

Webservice é uma interface que fornece acesso a um sistema voltado para a Web para clientes e outros serviços a serem consumidos (DEWAILLY, 2015, p. 1). Utilizou-se um webservice em Java para oferecer serviço a nossa camada de visualização, onde há a interface com o usuário, ou seja, nosso front-end. O modo como se oferece acesso ao back-end, ou seja, como as informações são acessadas e solicitadas determina que tipo de webservice é construido. O REST (*Representational State Transfer*) é uma abordagem de arquitetura de software para criar webservices escaláveis. Podemos ver no código 2.3 a classe responsável por implementar o serviço REST de gestão de categorias.

```
1  package br.org.fieb.senai.aac.rest;
2
3  import java.util.List;
4  import org.springframework.beans.factory.annotation.Autowired;
5  import org.springframework.http.HttpStatus;
6  import org.springframework.http.ResponseEntity;
7  import org.springframework.web.bind.annotation.DeleteMapping;
8  import org.springframework.web.bind.annotation.GetMapping;
9  import org.springframework.web.bind.annotation.PathVariable;
10 import org.springframework.web.bind.annotation.PostMapping;
11 import org.springframework.web.bind.annotation.PutMapping;
12 import org.springframework.web.bind.annotation.RequestBody;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RestController;
15
```

```
16 import br.org.fieb.senai.aac.Category;
17 import br.org.fieb.senai.aac.CategoryService;
18 import br.org.fieb.senai.aac.Id;
19 import br.org.fieb.senai.aac.Name;
20 import br.org.fieb.senai.aac.category.CategoryMapper;
21 import br.org.fieb.senai.aac.category.dto.CategoryDtoForCreation;
22 import br.org.fieb.senai.aac.category.dto.CategoryDtoForGetting;
23 import br.org.fieb.senai.aac.exception.CategoryNotFoundException;
24
25 @RestController
26 @RequestMapping("/category")
27 public class CategoryRestController {
28
29   @Autowired
30   private CategoryService categoryService;
31
32   @Autowired
33   private CategoryMapper categoryMapper;
34
35   @PostMapping
36   public ResponseEntity<CategoryDtoForGetting> createCategory(
       @RequestBody CategoryDtoForCreation dtoForCreation) {
37     Category category = categoryService.create(dtoForCreation);
38     CategoryDtoForGetting dtoForGetting = categoryMapper.
         mapToCategoryDtoForGetting(category);
39     return new ResponseEntity<>(dtoForGetting, HttpStatus.CREATED);
40   }
41
42   @GetMapping("/{id}")
43   public ResponseEntity<CategoryDtoForGetting> getCategory(@PathVariable
       Long id) {
44
45     Category category;
46     try {
47       category = categoryService.find(Id.of(id));
48     } catch (CategoryNotFoundException e) {
49       return new ResponseEntity<>(HttpStatus.NOT_FOUND);
50     }
51
52     CategoryDtoForGetting dto = categoryMapper.
         mapToCategoryDtoForGetting(category);
53     return new ResponseEntity<>(dto, HttpStatus.OK);
54   }
55
56   @GetMapping
57   public ResponseEntity<List<CategoryDtoForGetting>> getCategoryList() {
58
59     List<Category> categoryList = categoryService.list();
```

```
60
61      if (categoryList.isEmpty())
62        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
63
64      else {
65        List<CategoryDtoForGetting> dtoList = categoryMapper.
              mapToCategoryDtoForGetting(categoryList);
66        return new ResponseEntity<>(dtoList, HttpStatus.OK);
67      }
68    }
69
70    @PutMapping("/{id}/name")
71    public ResponseEntity<CategoryDtoForGetting> putCategoryName(
          @PathVariable Long id, @RequestBody String name) {
72
73      Category category;
74      try {
75        category = categoryService.find(Id.of(id));
76      } catch (CategoryNotFoundException e) {
77        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
78      }
79
80      category.name(Name.of(name));
81      CategoryDtoForGetting dto = categoryMapper.
            mapToCategoryDtoForGetting(category);
82      return new ResponseEntity<>(dto, HttpStatus.OK);
83    }
84
85    @DeleteMapping("/{id}")
86    public ResponseEntity<Void> deleteCategory(@PathVariable Long id) {
87
88      Category category;
89      try {
90        category = categoryService.find(Id.of(id));
91      } catch (CategoryNotFoundException e) {
92        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
93      }
94
95      category.delete();
96      return new ResponseEntity<>(HttpStatus.NO_CONTENT);
97    }
98  }
```

Código 2.3: Classe CategoryRestController

2.5 Orientação a Objetos

Orientação a Objetos é um paradgma de programação onde o software é modelado em termos semelhantes a objetos do mundo real, com suas características, comportamento e relações entre classes (DEITEL; DEITEL, 2009, p. 15). Dos recursos possíveis em Orientação a Objetos, o que mais se destaca nesse projeto é o uso de interfaces, que será apresentado a seguir.

2.5.1 Interfaces

A interface em Java, descreve um conjunto de métodos públicos e abstratos que pode ser chamado em um objeto para instruí-lo (DEITEL; DEITEL, 2009, p. 322). As interfaces são utilizadas nesse projeto com o objetivo de criar, na aplicação, uma camada com nível de abstração alta, mais próximo da regra de negócio. Observa-se o uso de interfaces, por exemplo, no código 2.4.

```
1  package br.org.fieb.senai.aac;
2
3  public interface Category {
4
5    Id id();
6
7    Name name();
8
9    void name(Name name);
10
11   void delete();
12 }
```

Código 2.4: Classe Category

Capítulo Três

Trabalho teórico e prático

3.1 Visão geral

Este trabalho trata de comunicação alternativa e aumentativa destinada a pessoas com dificuldades de fala, que utiliza o método alternativo de comunicação através de imagens e símbolos. Dentro do contexto de AAC (*augmentative and alternative communication*), existem sistemas de linguagem que associam imagens a palavras, esse recurso é utilizado para sintetizar uma voz que expressará o conteúdo determinado pela pessoa. Para esta pesquisa foi utilizado como público alvo pessoas em estado de internação hospitalar, as quais se encontram privadas da fala.

Como a aplicação proposta se trata de uma solução conjunta entre hardware e software, foi feita uma pesquisa de custo benefício de microcontroladores possíveis de serem utilizados. Foi feita a pesquisa no site norte americano da Adafruit, pois é o único site onde foram encontrados todos os microcontroladores listados na tabela 3.1. Para melhor visualização, listamos quais possuem suporte a sistema operacional, pois foi um dos fatores que influenciou no escolha, além do preço. O Raspberry pi 0 além de possuir o menor preço entre os microcontroladores pesquisados, possue o suporte aos sistemas operacionais Linux e Windows, o que nos permitiu desenvolver o sistema com mais liberdade de uso de tecnologias.

Tabela 3.1: Microcontroladores
Fonte: Próprio autor

Microcontrolador	**Valor**	**Suporte a S.O.**
Arduino	$ 17,50	Não
Raspberry pi 0	$ 5,00	Sim
Intel Edison	$ 49,95	Sim
Esp32	$ 19.95	Não

Para esse trabalho, foi construido um sistema de hardware e software, utilizando o microcontrolador raspberry pi zero w (Figura 3.1). Quanto ao sistema, desenvolvemos dois protótipos, um com um sistema em python com interface desenvolvida pelo framework Pygame, utilizando um banco de dados Sqlite3 e outro web. O microcontrolador possui uma saída HDMI que pode ser acoplada a qualquer tv em um ambiente de leito de hospital e o sistema é projetado para este ambiente, sofrendo adaptações nas categorias de imagens, baseadas em uma análise de requisitos feita por nossa equipe de pesquisadores.

O trabalho visa desenvolver um sistema de baixo custo, com expectativa de valores entre R$100,00 e R$150,00. Esse sistema compreende o microcontrolador, o controle sem fio (Joystick) para interação e o sistema embarcado, um cartão de memória para armazenar o sistema operacional e um cabo HDMI. A seguir será exposto como trabalhar com um sistema de AAC com um raspberry, utilizando as tecnologias do Python, PyGame e sqlite3.

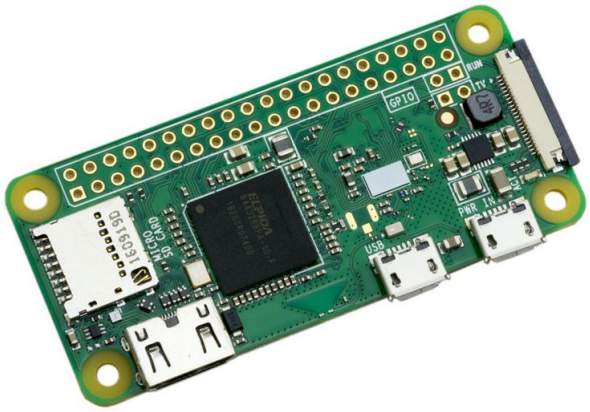

Figura 3.1: Raspberry pi 0 W
Fonte: <https://www.raspberrypi.org/products/raspberry-pi-zero/>

O raspberry pi possui diversas aplicações produzidas, o que faz com que esta tenha durante longo tempo um suporte da comunidade desenvolvedora, tanto em hardware quanto em software. Aplicações como o sistemas de automação residencial (JAIN; VAIBHAV; GOYAL, 2014), mostram que o raspberry pi tem suporte do sistema operacional Debian e utiliza como principal linguagem de programação o Python, com uma versão própria para sua plataforma, o Raspbian. Além dessa, existe uma versão da Microsoft, o Windows 10 para Raspberry pi, isso corrobora com a importância desse plataforma e o interesse em manter e investir cada vez mais nesse projeto de Hardware e Software. Outras aplicações feitas no raspberry com python a exemplo o sistema de irrigação inteligente (AGRAWAL; SINGHAL, 2015),

Mais motivos para adotar o raspberry, é o foco em um sistema de baixo custo. Zhao afirma que esta é uma boa plataforma para aplicações de baixo custo e é portátil (ZHAO; JEGATHEESAN; LOON, 2015) . Em outro projeto, SAHANI sugere a escolha do Sqlite como banco de dados para trabalhar embarcado no raspberry pi, em seu projeto de controle de acesso em um sistema de segurança residencial (SAHANI *et al.*, 2015). Assim como D'AMORE, em seu projeto de big data confirma o uso do Sqlite em projeto com raspberry pi (D'AMORE; BAGGIO; VALDANI, 2015).

Para nosso diferencial, utilizamos o Pygame, um framework do python que normalmente é utilizado em jogos, para dar o aporte da interface gráfica ao nosso sistema, assim como o projeto Probolino, que é um projeto de baixo custo para terapia em casos de autismo (CAO *et al.*, 2015). Foram pesquisados os descritores "Pygame" e "AAC", no Google Acadêmico, PubMed, nos últimos 5 anos e achou-se apenas um artigo de autoria Gavas *et al.* (2017), que desenvolveu um projeto sobre um sistema de rastreamento de olho baseado em sensores, utilizando o Pygame como ferramenta para construção do sistema, mas não utilizam o Raspberry pi como plataforma e sim um computador tradicional. Logo, foi concluído que para as bases pesquisadas, nosso sistema é o único que apresenta a proposta de sistema utilizando tais tecnologias nessa combinação, a pesar de muitos outros projetos apontarem para o uso de algumas já citadas.

Tal trabalho se trata do uso de uma interface gráfica baseada em sobreposição de imagens na tela, utilizando o framework do Pygame, com scripts em Python, que nos possibilita trabalhar com módulos específicos para imagem, como por exemplo o módulo Image, objetos de surface (superfície), nos possibilitando capturar eventos (objeto Event), para comparar a posição do cursor e clique no Joystick com a posição da imagem na tela. Assim conseguiu-se capturar eventos de movimentação de cursor e cliques em botões do joystick utilizado (3.2) o que facilita a usabilidade do paciente através dessa tecnologia sem fio. Os eventos disparados pelo Joystick acionam scripts em Python que fazem as ações no banco de dados e exibem as imagens na tela, após a seleção de um conjunto de imagens na tela, o usuário pode solicitar que o sistema pronuncie a frase montada. Foi utilizado a API do Google, o Google Speech que já foi empregado em projetos de AAC (MOHANAPRAKASH, 2015). Esse projeto é similar ao proposto, com semelhanças no uso de um banco de dados Sqlite, a api de sintetização de voz da Google, o Google Speech, mas o projeto é feito para mobiles Android, com foco em crianças com autismo e é uma solução apenas de software.

Figura 3.2: interface com usuário
Fonte:
<https://kingit.com.au/product/virtual-reality-remote-bluetooth-gamepad-controller/>

Sobre a API do Google, é necessário haver conexão com a internet para que o áudio seja baixado para o dispositivo, mas apenas na primeira vez que a palavra é sintetizada. Então foi criado um processo que automatiza o download de todos os audios das palavras pré-carregadas no dispositivo, antes da implantação no local. A internet será utilizada apenas em caso de adição de novas palavras ao sistema e apenas a primeira vez que essas palavras serão pronunciadas pelo sistema.

3.2 Metodologia

Para delinear a pesquisa foram feitos, o levantamento de requisitos, a validação do problema, a modelagem da solução utilizando UML e a protitipação do sistema. Para desenvolver a solução foram utilizados os seguintes materiais:

- 1 Raspberry pi 0 W
- 1 cabo HDMI
- 1 cartão de memória micro SD de 32GB
- 1 controle joystick bluetooth

Foi utilizado UML para modelagem do projeto, de acordo com os diagramas de classe, caso de uso, de sequência, de atividades, de implantação, de componentes, modelo lógico e conceitual, melhor detalhados no item 3.4. O primeiro protótipo foi desenvolvido com Python e o framework PyGame para interface gráfica, feito para a operar no raspberry, localmente e diretamente sob o sistema operacional. Para que o Raspberry pudesse receber o sistema projetado, foi necessário gravar no cartão de memória o sistema operacional Raspbian, disponível gratuitamente na internet. O joystick se conecta via bluetooth com o próprio Raspberry, sem necessidade de instalação de nenhum driver, funcionando diretamente como mouse, direcionando o cursor.

Após a configuração de sistema operacional, foi necessário atualizar os repositórios do sistema, baixar e instalar os pacotes atualizados do raspbian, instalar o framework PyGame e começar a desenvolver o código do sistema, etapa essa melhor descrita no item 3.3.

O segundo protótipo foi feito com arquitetura web, então para sua visualização, basta um navegador no equipamento. Foram utilizados as tecnologias de JavaEE, HTML, Java Script e CSS. As etapas de desenvolvimento desse protótipo estão desenvolvidas no capítulo 4.

3.3 Protótipo

O sistema consiste em um cadastro com funcionalidade de incluir, excluir, atualizar e ler dados de pacientes e imagens - na primeira tela, de acordo com a Figura 3.3. A depender do usuário selecionado, é carregado um conjunto de imagens pré-definidas desse usuário, ao qual podem ser modificadas a depender da necessidade do paciente.

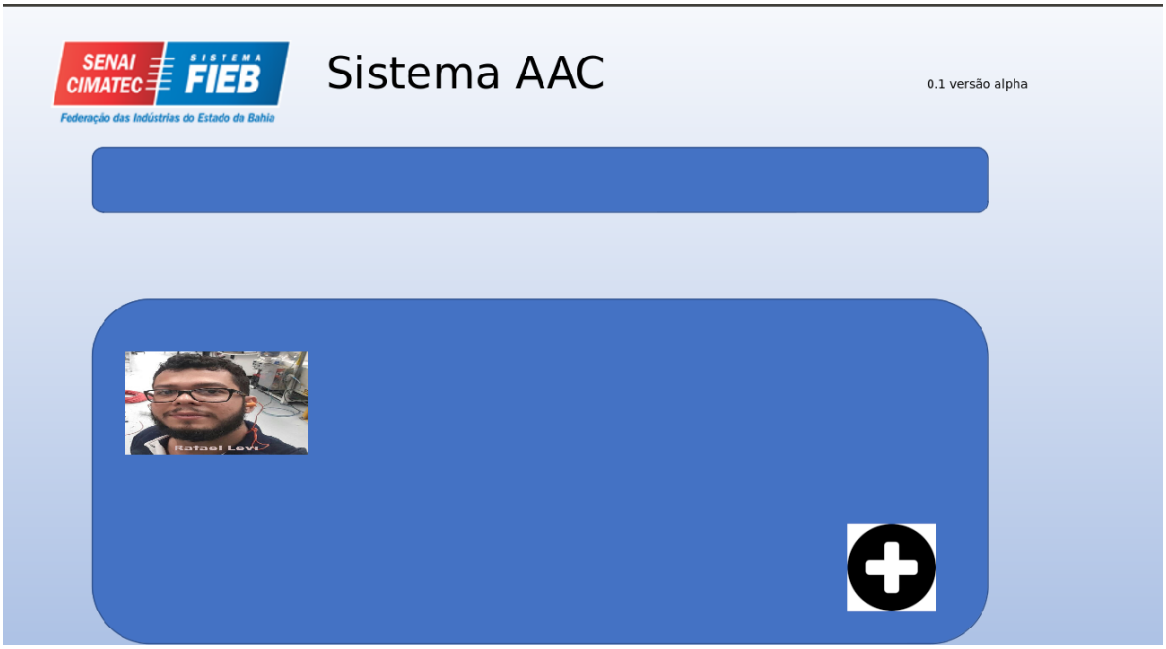

Figura 3.3: Tela de seleção de usuário
Fonte: Próprio autor

Na primeira tela, existem por padrão três opções de início de frase, sempre em primeira pessoa, "eu quero", "eu estou", "eu sou", como pode-se observar na Figura 3.4. Ao clicar em qualquer elemento que compõe a frase e é representado por uma figura, o sistema sintetiza a frase que está sendo composta a cada etapa. Então se o usuário clicar em querer, será sintetizado a frase eu quero, até que finalize a frase. Além do botão de adicionar mais elementos de frase a essa tela, é possivel também nessa etapa retornar à tela de seleção de usuário, no símbolo de home, ou ainda entrar no menu de configurações, no símbolo de engrenagem.

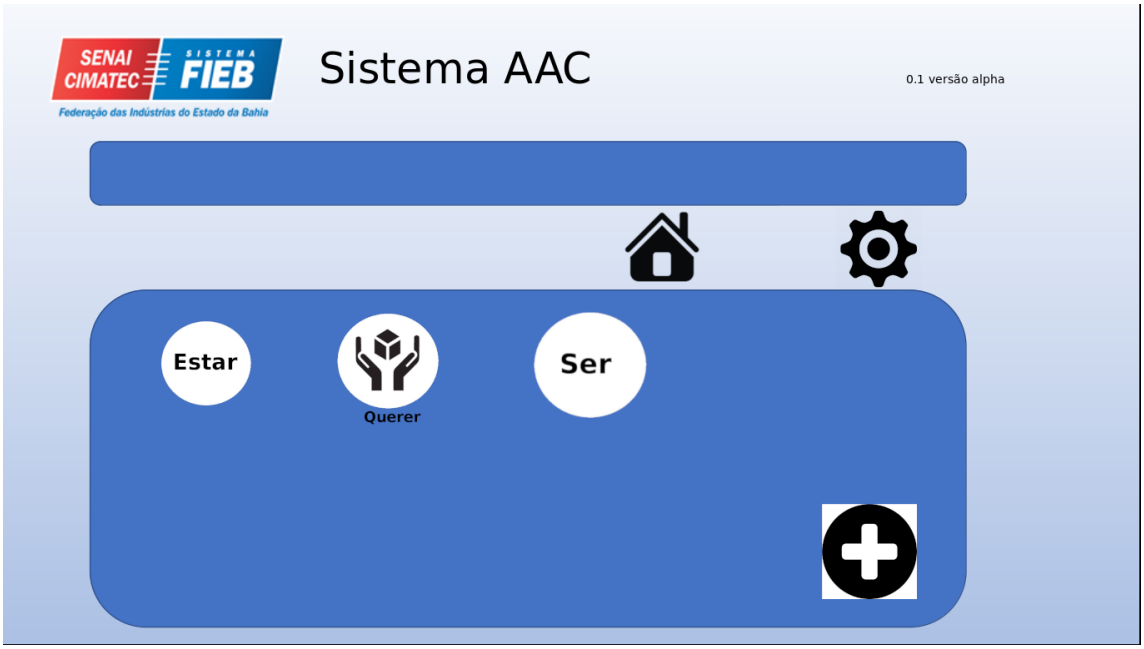

Figura 3.4: Primeira tela de composição de frase
Fonte: Próprio autor

Após cada seleção, é formando uma árvore de possibilidades de montagem de frase. Nota-se na Figura 3.5 que as opções dentro da possibilidade "eu quero", são "comer", "beber"e "falar". A partir dessa tela, pode-se identificar que a frase escrita vai se formando no retângulo superior da tela. Além de ter mais duas opões no menu, existe a opção de voltar para a tela anterior, descartando a última composição da frase, podendo também repetir a frase que foi montada até o momento para que o comunicador possa insistir ao ouvinte. A seguir, será exposto o processo de montar um exemplo de frase em uma situação, a qual o paciente deseja comer pão.

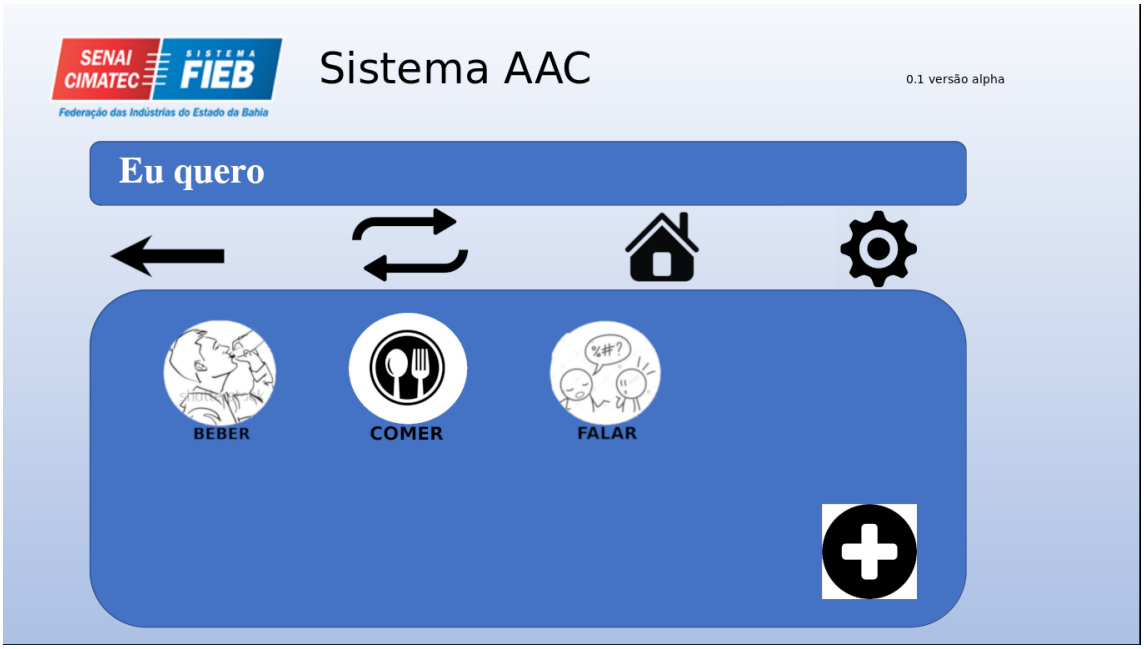

Figura 3.5: Compondo a frase
Fonte: Próprio autor

Agora, na Figura 3.6 pode-se visualizar a tela de opções de comida para o paciente, assim ele poderá finalizar a frase com a opção desejada para comer. O usuário ao finalizar a frase, pode voltar a tela de usuário, retomar a frase excluindo a última palavra, ou repetir a frase montada até que obtenha o desejado. Caso as opções de alimento não o satisfaça, ele poderá adicionar mais opções no símbolo de mais.

Figura 3.6: Frase completa
Fonte: Próprio autor

É possivel perceber que esse sistema além de contemplar especificidades de cada paciente, pode entregar um resultado personalizado, com frases através de um sintetizador de voz, em um ambiente que poderá ser compartilhado com outros pacientes em mesmo estado, baixando assim ainda mais seu custo por usuário, só é possível utilizar o sistema um paciente por vez. O código da primeira versão do sistema está disponível em uma plataforma de versionamento chamada gitlab, sob o endereço <https://gitlab.com/rafaellevissa/AAC-python>.

Já contemplamos que há limitações nesse protótipo, por conta do uso de imagens não padronizadas com os sistemas de linguagem alternativa. Para isso incorporamos no modelo em UML, a possibilidade de pré-carregamento dessas imagens vindo de uma banco de dados externo, através das bibliotecas disponibilizadas pelos desenvolvedores dos sitemas de linguagens já validados e citados no capítulo 2, na Tabela 2.

3.4 Modelagem UML

A seguir, será abordado o modelo proposto, utilizando a linguagem de modelagem unificada, do inglês UML (Unified Modeling Language). De acordo com Booch, Rumbaugh & Jacobson (2000), o UML proporciona um formato padronizado na preparação de planos em arquitetura de projetos. No decorrer desta pesquisa, serão apresentados alguns diagramas que pretendem modelar o sistema que apresentamos nesse trabalho: caso de uso, de classes, de sequência, de atividades, de implantação e de componentes. Sobre o banco de dados serão apresentados os modelos lógico e conceitual. Para cada diagrama apresentado, será discutido sua representação e como foi concebido.

3.4.1 Diagrama de classes

De acordo com Guedes (2008) esse diagrama é um dos mais importantes do UML, pois esta dará uma visão das classes com seus atributos e métodos e sua relação. Esse diagrama apresenta uma visão estática, representando apenas a sua estrutura lógica. Na Figura 3.7 observa-se uma classe representando todo o sistema, a classe SistemaAacApplication, que possui o método principal. O diagrama representa apenas o pacote principal.

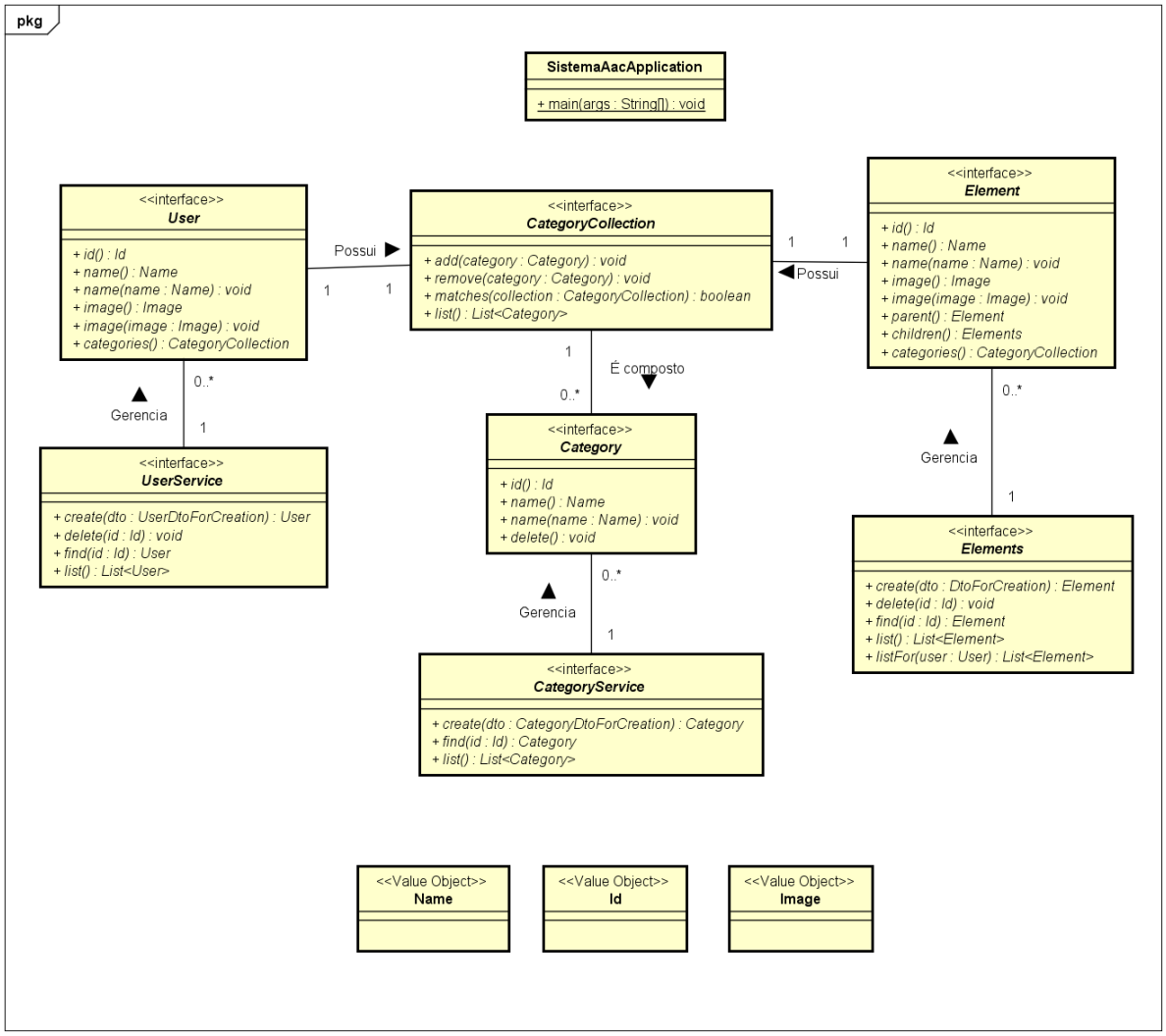

Figura 3.7: UML: diagrama de classes
Fonte: Próprio autor

Dividi-se a estrutura do sistema responsável por representar uma palavra em duas classes, a Image que é a imagem que vai representar uma palavra ou expressão dentro do sistema. A imagem é pura e simplesmente a figura representativa, enquanto a classe Word é composta do texto escrito que será sintetizado e uma imagem que é um objeto da classe Image. A composição entre classes de acordo com Guedes (2008) é uma relação de parte e todo entre duas classes que se compõe e em nosso diagrama temos a composição que podemos ver entre as classes Word e Image, sendo uma Image, parte que compõe uma Word. Para entendermos essa relação nesse ponto do diagrama, temos que analisar a classe Word, nesse caso, representa um conjunto de signo e palavra, que é a imagem e a própria palavra que é composta.

O sistema possibilita o que é chamado de CRUD de imagens e usuários. De acordo com Silva, Perri & Almeida (2011), o CRUD é o armazenamento, busca, atualização e deleção

de dados em um sistema, ou seja, um conjunto de operações básicas quando se trata de cadastros em um sistema. É através das classes Image e User que se traz para o front end, os dados do sistema. Ainda pode-se ver a classe UserType, que vai ajudar no momento do cadastro de um novo usuário, permitindo que um conjunto pré definido de imagens seja carregado para o usuário. Pode-se perceber uma agregação entre as classes UserType e User, de acordo com Guedes (2008) uma agregação é uma associação onde uma classe complementa informações contidas em outra. No caso deste trabalho, o perfil do usuário representado pela classe UserType é parte da informação que está contida na classe User.

3.4.2 Diagrama de caso de uso

Segundo Guedes (2008), esse diagrama apresenta de uma forma geral uma funcionalidade do sistema, apresentando o modo como os atores interagem com ela. Para tanto, primeiro foi feito uma tabela com os dados que vão compor o diagrama, como segue na tabela 3.2. A tabela apresentada anteriormente se refere a funcionalidade principal do sistema, que é chamado na tabela de caso de uso "Falar". Na tabela pode-se ver que temos como condições de uso da funcionalidade de falar, o cadastro do usuário e para que o caso de uso ocorra, têm-se a pós-condição de que o usuário deve selecionar ao menos uma expressão para que possa ser sintetizada. No cenário principal, pode-se ver uma sequência de ações do ator para a conclusão do caso de uso, o ator seleciona o usuário entre os cadastrados, o sistema vai exibir o conjunto de imagens relacionadas ao usuário, sendo que existem imagens associadas a cada usuário. Para isso que temos os perfis de usuários, para que ao ser criado o usuário, haver um conjunto pré configurado de imagens a ser associada a esse usuário, podendo ao longo do uso do sistema, ser adicionado ou excluído associações de imagens. Após a exibição das imagens relacionadas ao usuário, o mesmo seleciona uma, o sistema sintetiza o audio relacionado àquela expressão, exibe a frase em formato de texto e renova a tela com novas opções associadas à selecionada. O usuário repete esse processo até finalizar a montagem da frase, retornando ao menu de usuários.

A tabela gerou o diagrama apresentado na Figura 3.8, no qual podemos perceber, de forma gráfica, as ações competentes aos atores, paciente e sistema, que são responsáveis pelas ações que em nosso caso se relacionam, como por exemplo a ação de repetir uma frase, tratada como cenário alternativo e ainda relacionando com a tela da Figura 3.5, na qual podemos visualizar o botão de repetir, segundo da esquerda para a direita. Nesse botão podemos repetir a frase já montada, fazendo com que nosso ator Sistema, possa sintetizar a frase.

Tabela 3.2: Diagrama de caso de uso: Falar
Fonte: Próprio autor

Nome do caso de uso	**UC01 - falar**
Caso de uso geral	
Ator principal	Paciente
Resumo	Esse caso de uso descreve as etapas percorridas por um paciente, através do sistema, para montar uma frase a ser sintetizada.
Pré-condições	O usuário já deve estar cadastrado previamente.
Pós-condições	Ele deve selecionar ao menos uma expressão para montar a frase.
Cenário principal	
Ações do ator	Ações do sistema
1. Seleciona o usuário cadastrado	
	2. Exibe o conjunto de imagens referentes às palavras associadas ao usuário (primeira tela de palavras).
3. Seleciona a imagem	
	4. Sintetiza o som da expressão referente a imagem selecionada.
	5. Exibe o conjunto de palavras, baseado na opção selecionada.
6. Finaliza a frase.	
	7. Retorna ao menu de usuários.
Cenário alternativo - Repetir frase montada	
Ações do ator	Ações do sistema
1. Seleciona a opção de repetir frase	
	2. Sintetiza a frase montada até o momento.
Cenário alternativo - Voltar uma expressão	
Ações do ator	Ações do sistema
1. Seleciona a opção de voltar.	
	2. Exclui a última expressão selecionada à frase.
	3. Sintetiza a nova frase.
	4. Exibe o conjunto de palavras, baseado na opção anteriormente selecionada.
Cenário de exceção - Cliente não cadastrado	
1. Selecionar a opção de adicionar usuário	
	2. Solicitar dados do usuário.
3. Confirmar dados	
	4. Salvar no banco de dados
	5. Exibir opções de usuários.

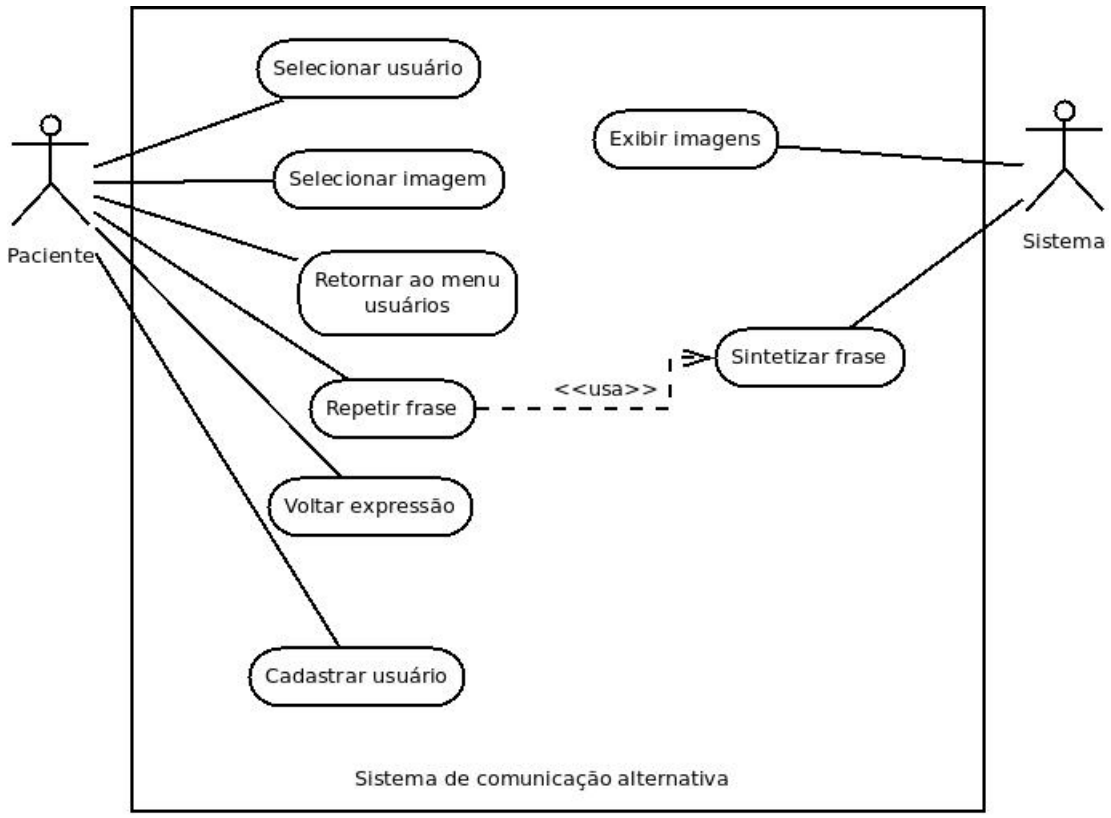

Figura 3.8: UML: diagrama de caso de uso
Fonte: Próprio autor

3.4.3 Diagrama de sequência

O diagrama de sequência é um diagrama comportamental que se preocupa com a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos em um determinado processo (GUEDES, 2008, p. 4). O diagrama que representa este trabalho é exposto na Figura 3.9, a qual pode-se verificar em sequência, o que acontece em nível de usuário ao sistema back-end, ao banco e a API de sintetização de voz, que nesse caso é o google speech. Analisando esse diagrama, pode-se ver que ao paciente selecionar o usuário, a interface do sistema, repassa para o banco, uma consulta de usuário por id, retornando assim um conjunto de imagens relacionados ao usuário selecionado. Após a seleção do usuário, é carregado um conjunto de imagens na tela, o usuário seleciona a imagem desejada, é feita o que chamamos de append na frase, que é uma adição em um vetor, na última posição, à expressão selecionada. Assim, ao compor a frase, adicionando a expressão selecionada é passada para o sintetizador de voz, o áudio é executado e o sistema carrega as novas imagens, relacionadas ao que chamamos de imagem pai. A imagem pai é uma imagem que possui um conjunto de outras imagens associadas a esta, como por exemplo na frase, eu quero comer pão, a expressão "eu quero", é pai da palavra comer, que é da palavra pão. Assim podemos montar uma árvore de possibilidades de frases a

serem montadas.

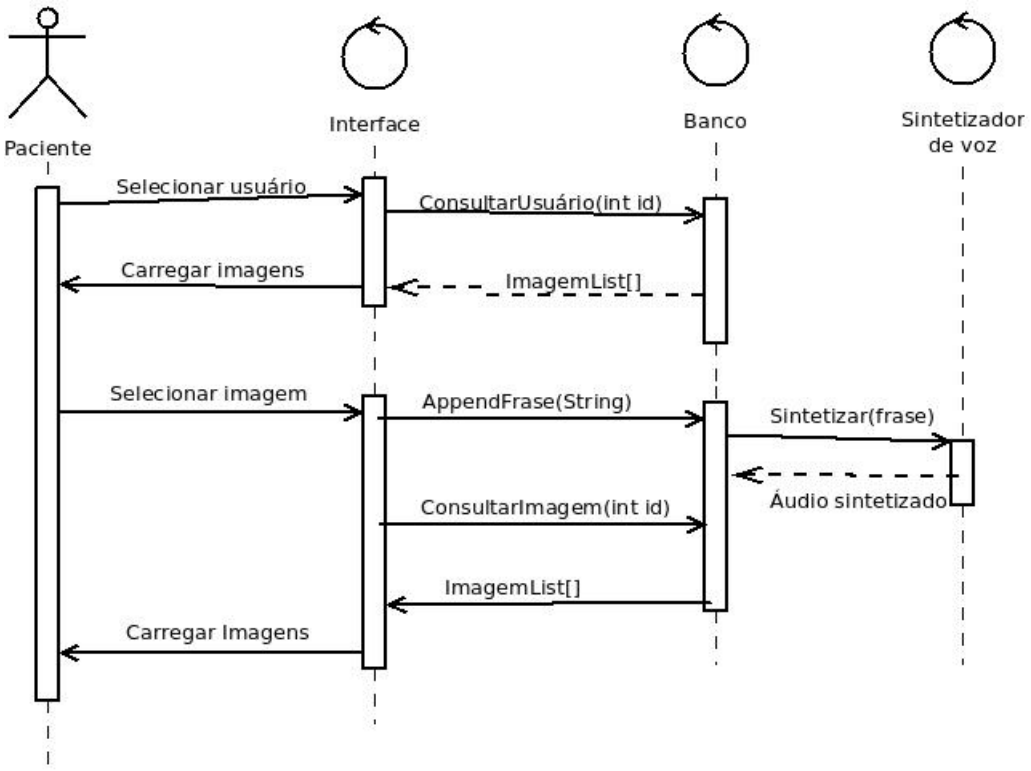

Figura 3.9: UML: diagrama de sequência
Fonte: Próprio autor

3.4.4 Diagrama de atividades

De acordo com Guedes (2008, p. 9) esse diagrama expressa a sequência e condições de objetos de uma determinada atividade. Segue o diagrama na Figura 3.10 que trás de novo uma visão condicional no cenário de um usuário montando uma frase e nesse caso pode-se ver a primeira condição de cadastro prévio do usário, comentada também no diagrama de caso de uso. Pode-se ver logo abaixo, ainda nessa figura, o outro condicional de finalização de frase que se faz repetir as operações de exibir imagens, selecionar, fazer o append na frase e sintetizar essa frase, retornando até que a frase se complete e o usuário retorne a tela de seleção de usuário.

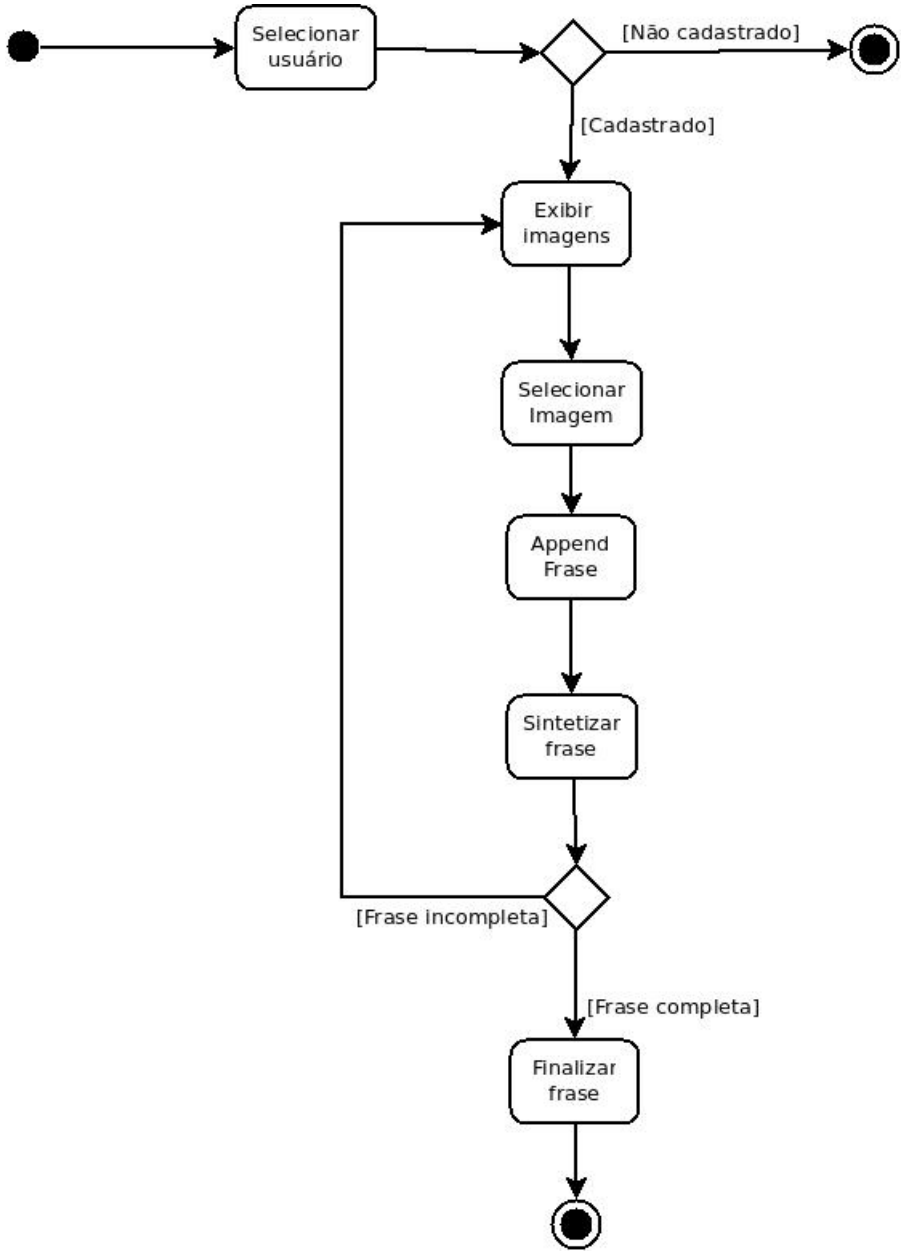

Figura 3.10: UML: diagrama de atividades
Fonte: Próprio autor

3.4.5 Diagrama de implantação

A especificação de implantação é um "conjunto de propriedades de um artefato implementado em um nó"(GUEDES, 2008). Na Figura 3.11 podemos ver a relação entre o servidor de banco de dados, que no nosso caso foi feita sobre o sqlite3, que é um SGBD (Sistema de gerenciamento de banco de dados) nativo no sistema operacional raspbian do raspberry pi. Podemos ver também um servidor de aplicação, que é o conjunto de código de fato, com a classe principal, janelaMain.py, que roda no raspberry, que é o cliente, utilizando

via HDMI, a tela da TV e via bluetooth, um joystick para interface e controle do cursor do mouse, na nossa aplicação.

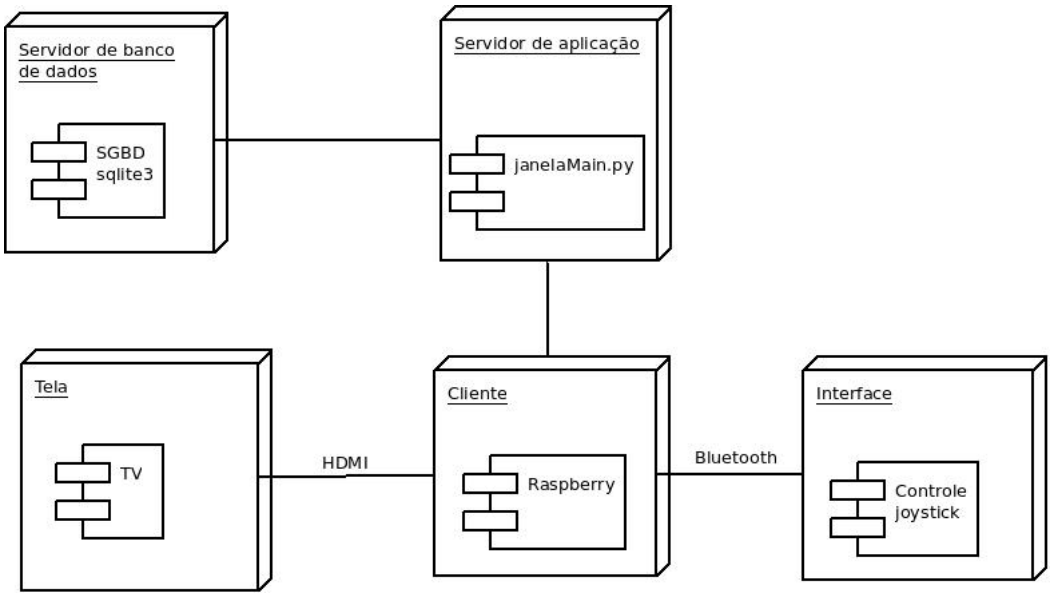

Figura 3.11: UML: diagrama de implantação
Fonte: Próprio autor

3.4.6 Diagrama de componentes

Será exposto a seguir o diagrama de componentes, que auxilia no processo de visualização de componentes e como eles se compõe. Guedes (2008, p. 10) afirma que esse diagrama identifica um subsistema ou mesmo os componentes ou classes internas. A Figura 3.12 deixa explícito quais componente serão utilizados em nosso sistema, sendo um grande pacote, o *framework* PyGame, para o desenvolvimento da interface do sistema, as querys, que são requisições ao banco de dados feitas conectando o sqlite3 com o sistema e em fim a classe fala.py, responsável por conectar a API da google de sintetização de voz, Google Speech.

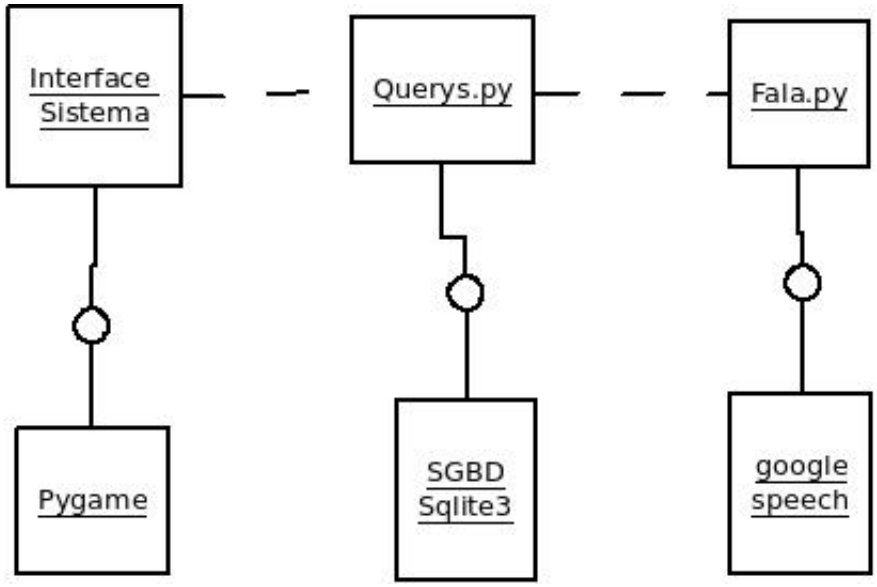

Figura 3.12: UML: diagrama de componentes
Fonte: Próprio autor

3.4.7 Modelo lógico

Na Figura 3.13, é exposto como estão montadas as tabelas em nosso banco sqlite3. Pode-se notar as duas principais tabelas, que armazenam imagem e usuários. Na tabela de imagem, temos um id, que é uma chave primária, do tipo inteiro, com a pripriedade autoincrement e unique para garantir o auto incremento e unicidade dos registros da tebela. Além disso, temos as colunas pai, que é uma chave estrangeira, ou seja, é a chave primária de outra imagem que é a imagem geradora de outro conjunto de imagens. Na coluna pai, colocaremos o id da imagem que se relaciona com esta e está por consequência em uma tela anterior. A coluna imagem possui o endereço do diretório ao qual está a imagem e a coluna nome, a palavra ou expressão que será sintetizada e geradora da frase.

Na tabela usuário possui a chave primária, id, as colunas, nome, sexo e foto que possuem dados pessoais do paciente e perfil que determinará a qual perfil pertence esse usuário. A tabela ImagemUsuario é uma entidade relacional entre imagem e usuário. Nessa tabela pode-se perceber o id do registro dessa entidade e os ids que compõe a relação entre imagem e usuário.

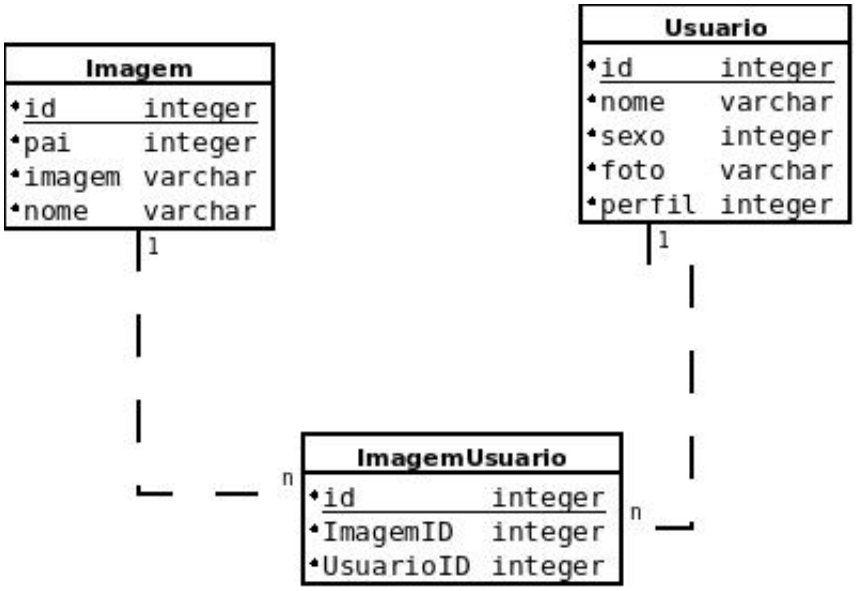

Figura 3.13: Banco: modelo lógico
Fonte: Próprio autor

3.4.8 Modelo conceitual

Por último vamos para o modelo conceitual. Na Figura 3.14 pode-se ver alguns detalhes a nível conceitual, por exemplo, um usuário que pode se especificar em enfermeiro e paciente, sendo que apenas o enfermeiro pode cadastrar um novo paciente, este por sua vez possui um conjunto de imagens que se organizam em telas, sequenciadas, onde cada imagem possui um pai, referência para montagem dessas telas.

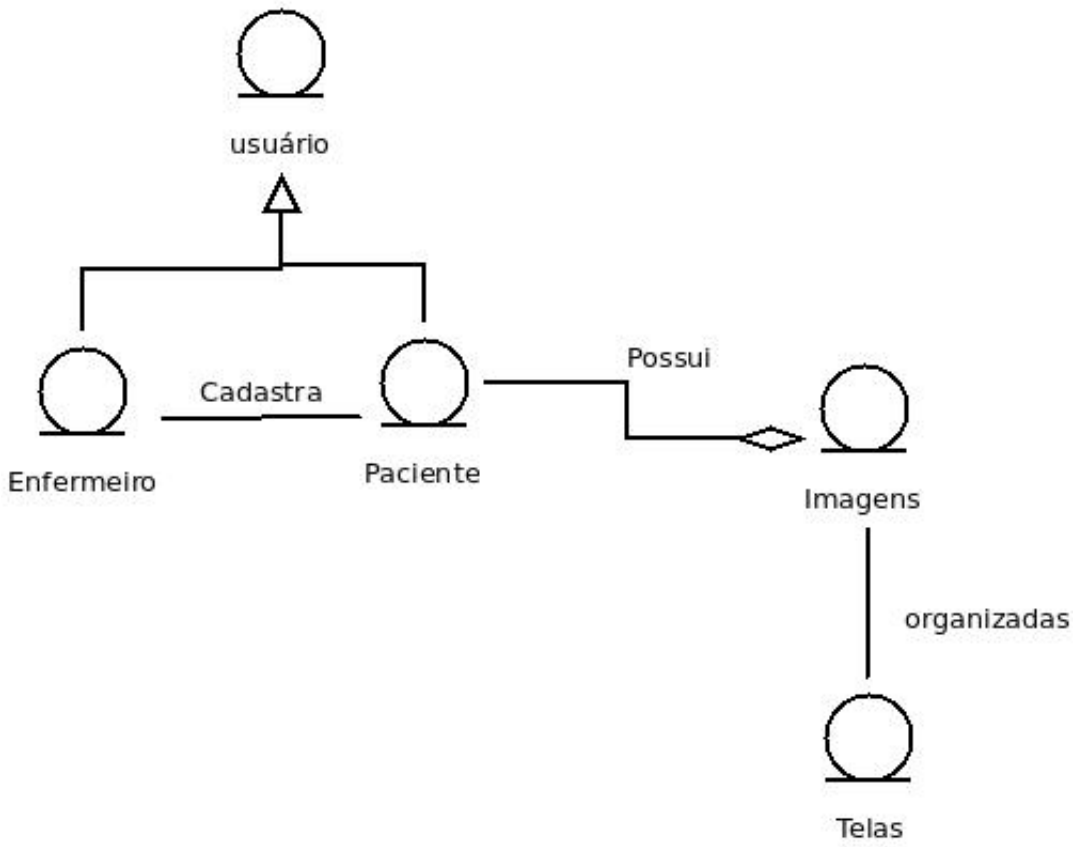

Figura 3.14: Banco: modelo conceitual
Fonte: Próprio autor

3.5 Segundo protótipo

A fim de superar algumas limitações encontradas na primeira versão do sistema, um segundo protótipo foi desenvolvido utilizando uma abordagem diferente. O primeiro protótipo possue seu código feito em python, utilizando framework Pygame, banco de dados Sqlite3 e é embarcado no raspberry pi 0 W. Esse modelo apresentou algumas limitações quanto a desempenho e usabilidade. Este, não pode ser apresentado via web e devido as necessidades de apresentação do sistema em qualquer dispositivo, inclusive do navegador, criou-se uma versão mais complexa, dividida em 2 partes, um front-end com HTML, Java Script e CSS e um back-end com Java EE. Os detalhes sobre as tecnologias e conceitos utilizados no segundo protótipo são apresentados no próximo capítulo. Os diagramas UML gerados nesse capítulo são referentes ao segundo protótipo, pois esse apresenta uma arquitetura mais complexa e melhor apresentável em tais diagramas.

Capítulo Quatro

Aspectos tecnológicos

Nesse capítulo vamos expor os aspectos técnicos da construção do sistema, referentes ao segundo protótipo que trouxe uma visão melhor do mesmo, possibilitando fazer uma análise entre os dois sistemas. O sistema foi dividido em *front* e *back-end*. De acordo com Godbolt (2016, p. 9) uma arquitetura *front-end* é um conjunto de ferramentas e processos que visam gerenciar um projeto *front-end*. O desenvolvedor *front-end* cuida do design do sistema e funcionalidades que ficam do lado do cliente, normalmente utilizado em projetos web.

O *back-end* é a aplicação responsável por prover acesso ao banco de dados, após passar por uma lógica própria do sistema, que por sua vez é feita com referência ao modelo de negócio ao qual a aplicação atende. Esse protótipo foi feito com a possibilidade de embarcar as aplicações de *front* e *back-end* ou ainda prover acesso web, embarcando essas aplicações em ambientes em nuvem. O *front-end* desse projeto está disponível no repositório git no link <https://gitlab.com/rafaellevissa/sistema-aac-front-end>.Desenvolveu-se o back-end com a linguagem de programação Java, feito com suporte da IDE Eclipse e uso dos frameworks e conceitos que apresentaremos a seguir. O projeto do back-end está disponível em <https://gitlab.com/rafaellevissa/sistema-aac-back-end>.

Foram utilizados alguns Frameworks durante o projeto desenvolvido. Framework é uma coleção de códigos, que em nosso caso do desenvolvimento Java, é composto por classes, funções e metodologias com o objetivo de facilitar o desenvolvimento de *software* (MINETTO, 2007, p. 17).

A aplicação pode ser utilizada embarcada em um raspberry pi ou em qualquer dispositido que possua conecção com a internet e disponha de um *browser*. O sistema será disponibilizado no endereço <http://35.190.138.50:8081/browser/index.html#/>, para testar, basta abrir um navegador e digitar esse link adicionando a informação que deseja, a exemplo pode-se adicionar /users ao final da url para que o navegador exiba as informações de usuários cadastrados nos sistema, como resultado obteve-se a resposta da Figura 4.1.

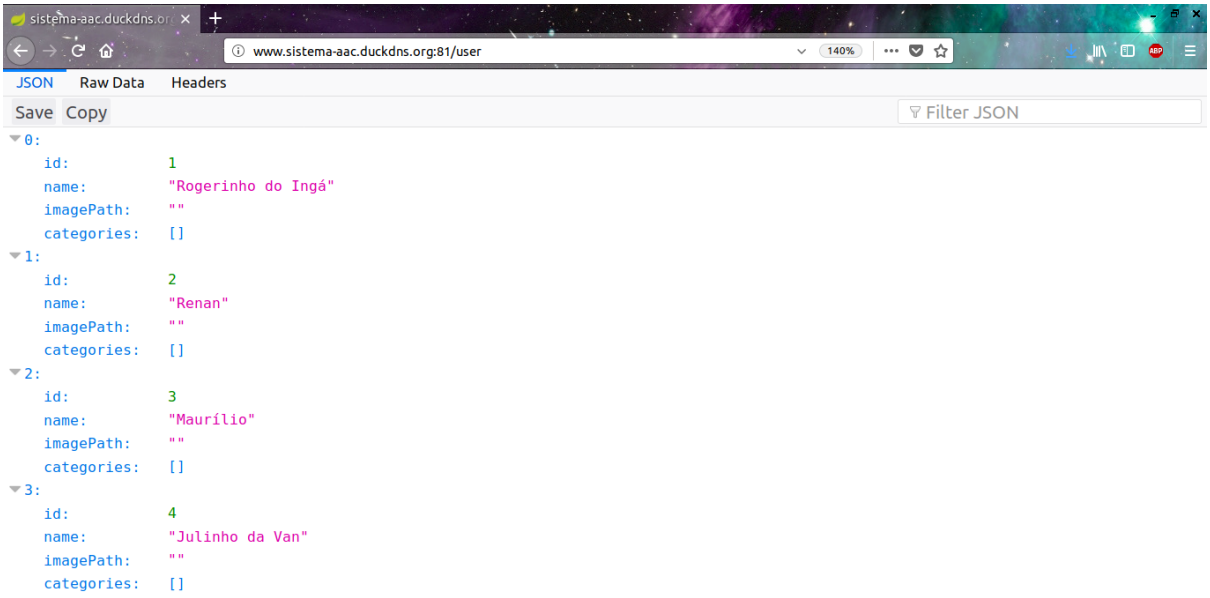

Figura 4.1: Resposta do webservice
Fonte: Próprio autor

4.1 Java

A linguagem de programação Java é orientada a objetos, de propósito geral e baseada em classes, de acordo com Gosling, Joy & Steele (2000, p. 1). É uma linguagem fortemente tipada, que ao compilar o código transforma em um *Byte Code* que é independe da máquina, ou seja, o código java pode ser compilado para uma JVM (*Java Virtual Machine*) e dessa forma este independe da máquina e do sistema operacional ao qual será executado como mostra a Figura 4.2. Essa foi a linguagem de programação utilizada no projeto do back-end exatamente por facilitar o desenvolvimento em diferentes sistemas operacionais, já que o raspberry pi possui suporte ao Linux e Windows. Além disso essa linguagem é de domínio do desenvolvedor que projetou o sistema.

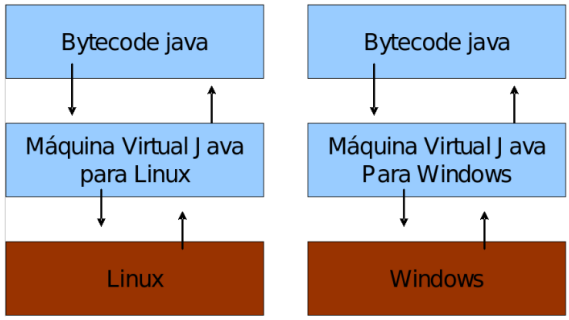

Figura 4.2: Funcionamento da máquina virtual Java.
Fonte: Próprio autor

4.1.1 JPA

O JPA é um framework de persistência, de acordo com Rombaldo, Souza & Souza (2012), é responsável por mapear os objetos da aplicação e relacionar com o acesso a dados pelo SGBD (Sistema de Gerenciamento de Banco de Dados). Foi utilizado o JPA com o Hibernate para fazer o mapeamento das classes do projeto back-end para o banco.

O Hibernate é uma implementação do JPA, sendo assim é um framework de persistência. De acordo com Hemrajani (2006, p. 15), ele é responsável pelo mapeamento de objetos relacionais, dentro de um projeto JAVA. Podemos observar o uso do JPA na figura **??**, onde temos a anotação da linha 14, determinando que essa classe CategoryEntity será mapeada para uma tabela com o nome category, na linha 23 determinamos que o atributo id será o Id (chave primária) da tabela e será gerado automaticamente pelo banco de dados.

Dessa forma com o uso do JPA, podemos dentro da linguagem escolhida, facilitar o processo de conexão do sistema com o banco de dados.

4.1.2 Spring

O Spring é um framework Java, que de acordo com Hemrajani (2006, p. 15), responsável por inversão de controle em projetos web. IoC, do inglês *Inversion of Control* é um padrão de desenvolvimento onde o controle de chamada de métodos é invertida, ou seja, não é determinada diretamente pelo programador e sim por uma infraestrutura de software. Pode-se observar o uso desse framework no código 4.1. Na linha 10 temos uma anotação determinando que essa classe CategoryDataLoader será gerenciada pelo Spring e na linha 13 temos um ponto de injeção de dependência onde o Spring atribui uma instância do atributo CategoryService.

```java
package br.org.fieb.senai.aac.category;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import br.org.fieb.senai.aac.CategoryService;
import br.org.fieb.senai.aac.category.dto.CategoryDtoForCreation;

@Component
public class CategoryDataLoader implements CommandLineRunner {

  @Autowired
  private CategoryService categoryService;

  @Override
  public void run(String... args) throws Exception {
    categoryService.create(new CategoryDtoForCreation("Paciente padr o"
        ));
    categoryService.create(new CategoryDtoForCreation("Traqueostomia"));
    categoryService.create(new CategoryDtoForCreation("C ncer"));
  }
}
```

Código 4.1: Uso do Spring

4.1.3 Spring Boot

O Spring Boot é um framework de back-end. Ele já trás um servidor web embutido, suprimindo a necessidade de preocupação com esse item de infraestrutura, tornando a aplicação inteira executável (GUTIERREZ, 2016, p. 1). No código 4.2 pode-se observar o uso do Spring Boot, pois a anotação da linha 6 determina que essa será a classe com o método main ao executar o projeto.

```java
package br.org.fieb.senai.aac;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SistemaAacApplication {

  public static void main(String[] args) {
    SpringApplication.run(SistemaAacApplication.class, args);
  }
```

```
12 }
```

Código 4.2: Uso do Spring Boot

4.1.4 Lombok

O Lombok é um framework Java que ajuda a evitar o uso de códigos que se repetem normalmente em qualquer projeto Java, a exemplo métodos construtores, Gets e Sets. Podemos observar o uso desse framework no código 4.3. Nas linhas de 15 a 20 temos anotações de uso do Lombok, por exemplo, na linha 15 temos uma anotação que gera os métodos gets e sets entre outros.

```
1  package br.org.fieb.senai.aac.category;
2
3  import javax.persistence.Entity;
4  import javax.persistence.GeneratedValue;
5  import javax.persistence.Id;
6
7  import lombok.AllArgsConstructor;
8  import lombok.Builder;
9  import lombok.Data;
10 import lombok.EqualsAndHashCode;
11 import lombok.NoArgsConstructor;
12 import lombok.ToString;
13
14 @Entity(name = "category")
15 @Data
16 @NoArgsConstructor
17 @AllArgsConstructor
18 @Builder
19 @EqualsAndHashCode(of = "id")
20 @ToString(of = "id")
21 public class CategoryEntity {
22
23   @Id
24   @GeneratedValue
25   private Long id;
26
27   private String name;
28 }
```

Código 4.3: Uso do Lombok e JPA

4.1.5 Maven

O Maven é uma ferramenta de automação de *buid* em projetos Java Bharathan (2015, p. 35). Ela automatiza o processo de construção da aplicação baseada no código fonte, através de um arquivo de configuração, o Pom.XML. Esse arquivo contém informações de como o software será construído e suas dependências. O maven se encarrega de baixar dinâmicamente as dependências do projeto Java e armazenar localmente. O arquivo POM.XML pode ser observado no código 4.4, nele observamos as tags referentes a identificação do projeto nas linhas 6, 7 e 8.

```
1  <project xmlns="http://maven.apache.org/POM/4.0.0"
2    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.
       apache.org/xsd/maven-4.0.0.xsd">
4    <modelVersion>4.0.0</modelVersion>
5
6    <groupId>br.org.fieb.senai</groupId>
7    <artifactId>sistema-aac</artifactId>
8    <version>0.0.1-SNAPSHOT</version>
9
10   <parent>
11     <groupId>org.springframework.boot</groupId>
12     <artifactId>spring-boot-starter-parent</artifactId>
13     <version>2.0.2.RELEASE</version>
14   </parent>
15
16   <properties>
17     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18     <project.reporting.outputEncoding>UTF-8</project.reporting.
         outputEncoding>
19     <java.version>1.8</java.version>
20   </properties>
21
22   <dependencies>
23
24     <dependency>
25       <groupId>org.springframework.boot</groupId>
26       <artifactId>spring-boot-starter-data-jpa</artifactId>
27     </dependency>
28
29     <dependency>
30       <groupId>org.springframework.boot</groupId>
31       <artifactId>spring-boot-starter-web</artifactId>
32     </dependency>
33
34     <dependency>
35       <groupId>org.springframework.boot</groupId>
```

```
36        <artifactId>spring-boot-devtools</artifactId>
37        <scope>runtime</scope>
38      </dependency>
39
40      <dependency>
41        <groupId>com.h2database</groupId>
42        <artifactId>h2</artifactId>
43        <scope>runtime</scope>
44      </dependency>
45
46      <dependency>
47        <groupId>org.springframework.boot</groupId>
48        <artifactId>spring-boot-starter-test</artifactId>
49        <scope>test</scope>
50      </dependency>
51
52      <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
53      <dependency>
54        <groupId>org.projectlombok</groupId>
55        <artifactId>lombok</artifactId>
56        <scope>provided</scope>
57      </dependency>
58
59      <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
60      <!-- https://stackoverflow.com/a/49707331/3831396 -->
61      <dependency>
62        <groupId>javax.xml.bind</groupId>
63        <artifactId>jaxb-api</artifactId>
64      </dependency>
65
66      <!-- https://mvnrepository.com/artifact/org.modelmapper/modelmapper
          -->
67      <dependency>
68        <groupId>org.modelmapper</groupId>
69        <artifactId>modelmapper</artifactId>
70        <version>1.1.3</version>
71      </dependency>
72
73    </dependencies>
74
75    <build>
76      <plugins>
77        <plugin>
78          <groupId>org.springframework.boot</groupId>
79          <artifactId>spring-boot-maven-plugin</artifactId>
80        </plugin>
81      </plugins>
82    </build>
```

```
83
84 </project>
```

Código 4.4: Arquivo POM

4.1.6 *Tomcat*

O Tomcat é um servidor de código aberto, baseado no apache com papel de servidor web, executar servlets Java e converter documentos JSP em servlets Java (ZAMBON, 2012, p. 259). O Tomcat é responsável por expor o projeto web, que em nosso caso está associado ao back-end. Com esse servidor web, basta um cliente web, ou seja, um navegador para consumir o webservice feito em Java, através do protocolo HTTP, que possui entre seus métodos de acesso, o GET e POST. Foi utilizado esse servidor para expor o projeto web do back-end e podemos visualizar a inicialização do servidor Tomcat na figura 4.3

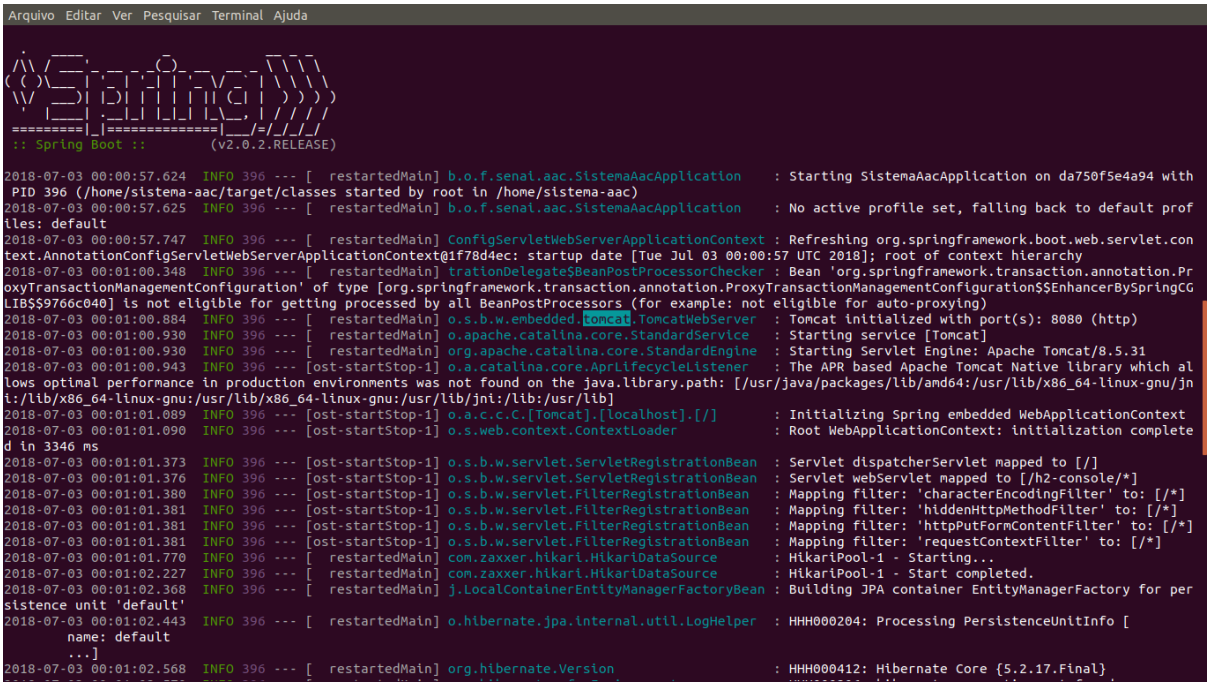

Figura 4.3: Tomcat sendo iniciado
Fonte: Próprio autor

4.2 Git

Foi utilizado o sistema de versionamento de projetos de desenvolvimento, Git. Para gerenciar o versionamento utilizou-se especificamente o Gitlab, como podemos ver na

Figura 4.4

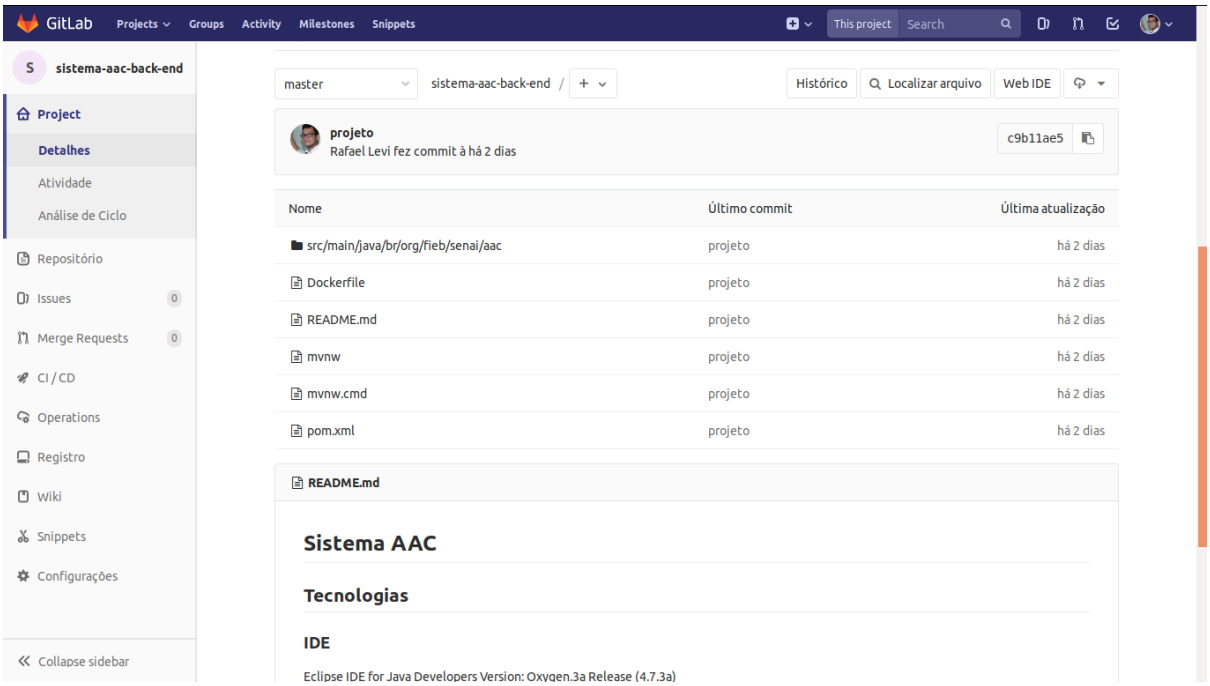

Figura 4.4: GitLab
Fonte: Próprio autor

4.3 *Cloud Computing*

Cloud Computing é um conceito que significa computação em nuvem. Neste trabalho utilizou-se um serviço específico de computação em nuvem que é a virtualização de servidores, isso significa que é utilizado uma instância de máquina com um sitema operacional Linux. O Ubuntu Server foi escolhido como servidor linux para hospedar os serviços na internet.

4.3.1 *Google Cloud Platform*

Criou-se uma máquina virtual em nuvem, utilizando a plataforma em nuvem da Google, o GCP (Google Cloud Platform). Essa plataforma possui diversos serviços, entre eles foi utilizado o *Compute Engine*. Lá é possivel criar uma instância com a imagem do Ubuntu Server 16.04LTS, como mostra a Figura 4.5.

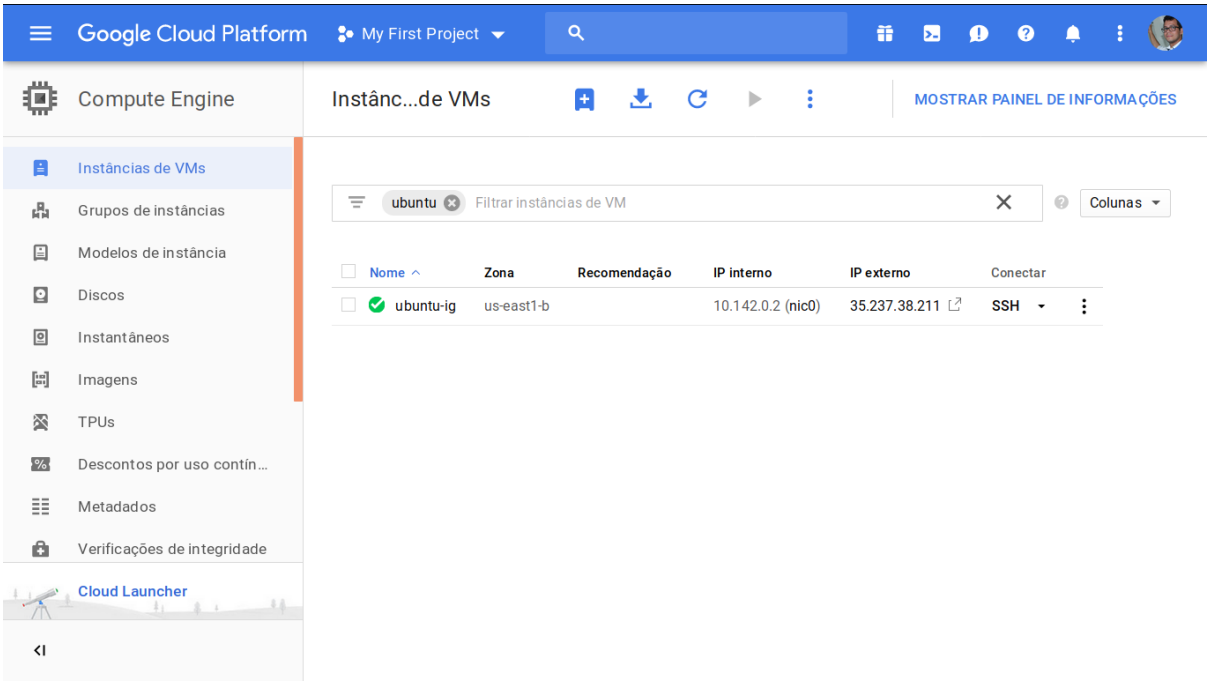

Figura 4.5: GCP: máquina virtual
Fonte: Próprio autor

Essa máquina virtual é acessivel através do protocolo ssh, com informações do IP (Internet Protocol) público, login e senha do sistema operacional. O IP público podemos observar na Figura 4.5, na coluna IP externo. Através desse número conectou-se nessa máquina por acesso remoto utilizando um cliente ssh, chamado Putty. Após acessar a máquina, configurou-se o ambiente linux para receber a aplicação. Foi realizado a configuração do ambiente para receber containers através do Docker, um gerenciador de containers. O projeto Java de back-end está em um container e o front-end também em outro. A seguir será explicado o conceito de container.

4.4 DevOps

DevOps é um conjunto de práticas avançadas para desafios comuns na entrega e operação de software (HTTERMANN, 2012, p. 3). O termo Devops, vêm de desenvolvimento (do inglês, *development*) e operação (do inglês, *operations*), portanto são práticas que tendem a facilitar a entrega de software, através de automação de infraestrutura. Entre os diversos conceitos que abarcam essa filosofia, tem-se o Container que é um recipiente de aplicações, responsáveis por hospedar e alocar recurso para aplicativos em microambientes (KANG; LE; TAO, 2016, p. 202). O container substitui a necessidade de haver uma máquina virtual para embarcar a aplicação, com as vantagens de proporcionar controle de ambiente,

tempo de inicialização rápido e alta possibilidade de reuso. Neste trabalho foi utilizado a aplicação em duas versões, uma embarcada no raspberry pi, possibilitando o uso da aplicação mesmo sem acesso a internet. A outra versão é essa aplicação, embarcada em containers, separados front e back-end, em cima de uma máquina virtual que possui o Docker (gerenciador de containers), isso na plataforma em nuvem GCP. Essa segunda, tem por objetivo possibilitar o acesso à aplicação via web.

4.4.1 Docker

O Docker é um gerenciador de containers, também definido como orquestrador de containers. (BOETTIGER, 2015). Pode-se observar o Docker na máquina virtual da GCP, listando os containers que estão iniciados na máquina, na Figura 4.6. Pode-se observar que existem dois containers iniciados, um para o projeto do front-end e outro para o back-end.

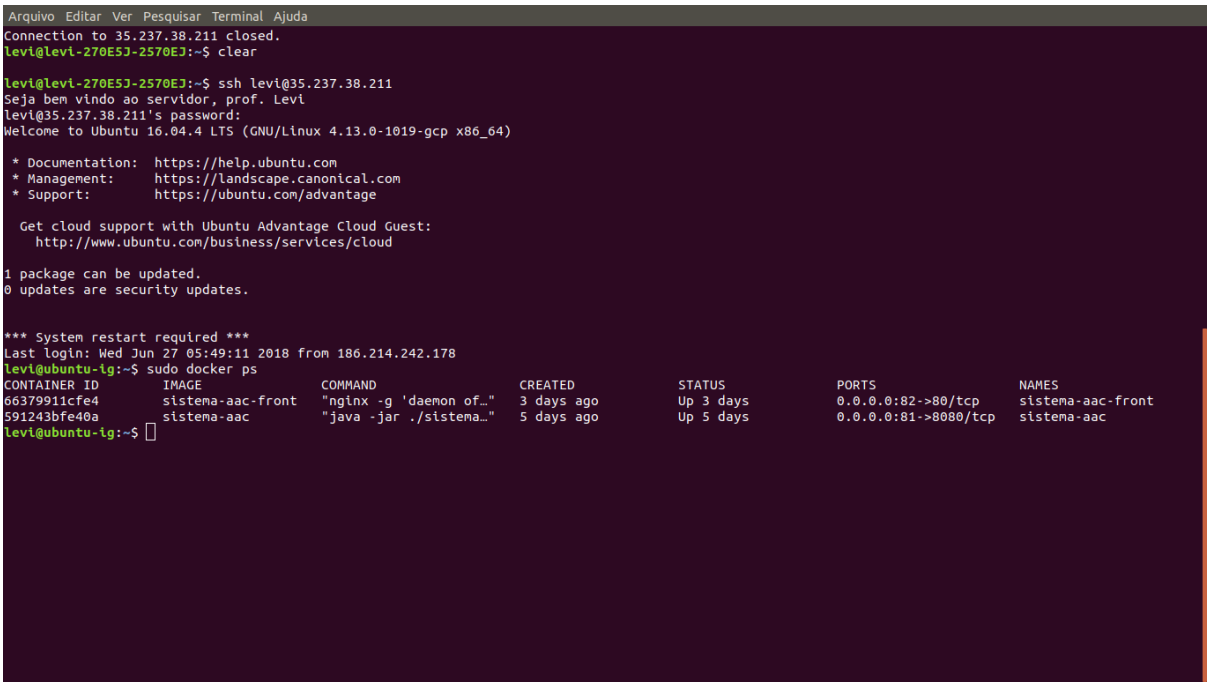

Figura 4.6: Docker: listando containers
Fonte: Próprio autor

Para o Docker construir a imagem, com o projeto de desenvolvimento embarcado nesse container, é necessário adicionar ao projeto um arquivo com o nome de Dockerfile, sem extensão. Pode-se ver o conteúdo do documento Dockerfile do projeto back-end no código 4.5. Observa-se que a imagem criada toma como base a imagem openjdk:8 que já está disponível no repositório público Docker, adiciona o projeto compilado pelo Maven, com

extensão .jar, expõe a porta 8080 para esse projeto e finalmente acessa o diretório e envia o comando de execução de um projeto java, o -jar. Após a criação dessa imagem, já pode-se criar o container baseado nessa imagem e iniciá-lo, isso fica a cargo do Jenkins, que está instalado na máquina virtual.

```
1 FROM openjdk:8
2 COPY ./target/*.jar ./usr/share/sistema-aac/
3 EXPOSE 8080
4 WORKDIR ./usr/share/sistema-aac/
5 ENTRYPOINT ["java", "-jar", "./sistema-aac-0.0.1-SNAPSHOT.jar"]
```

Código 4.5: Dockerfile do back-end

4.4.2 Jenkins

O Jenkins é uma ferramenta de integração contínua, de código aberto (SMART, 2011, p. 3). O Jenkins foi utilizado para associar o projeto armazenado no repositório do GitLab com a imagem gerada para criação do container no Docker. Podemos observar sua interface web disponível no endereço <http://sistema-aac.duckdns.org:8080/> e representada na figura 4.7. Através dessa interface configuramos para ele monitorar o repositório git e caso seja enviado alguma modificação de código ao projeto, o Jenkins executa um script shell, responsável por fazer a criação da imagem e geração do container já mapeado nas devidas portas do servidor. O Jenkins está instalado na máquina virtual da GCP e monitora os projetos de front e back-end no GitLab, gerando um container para cada projeto.

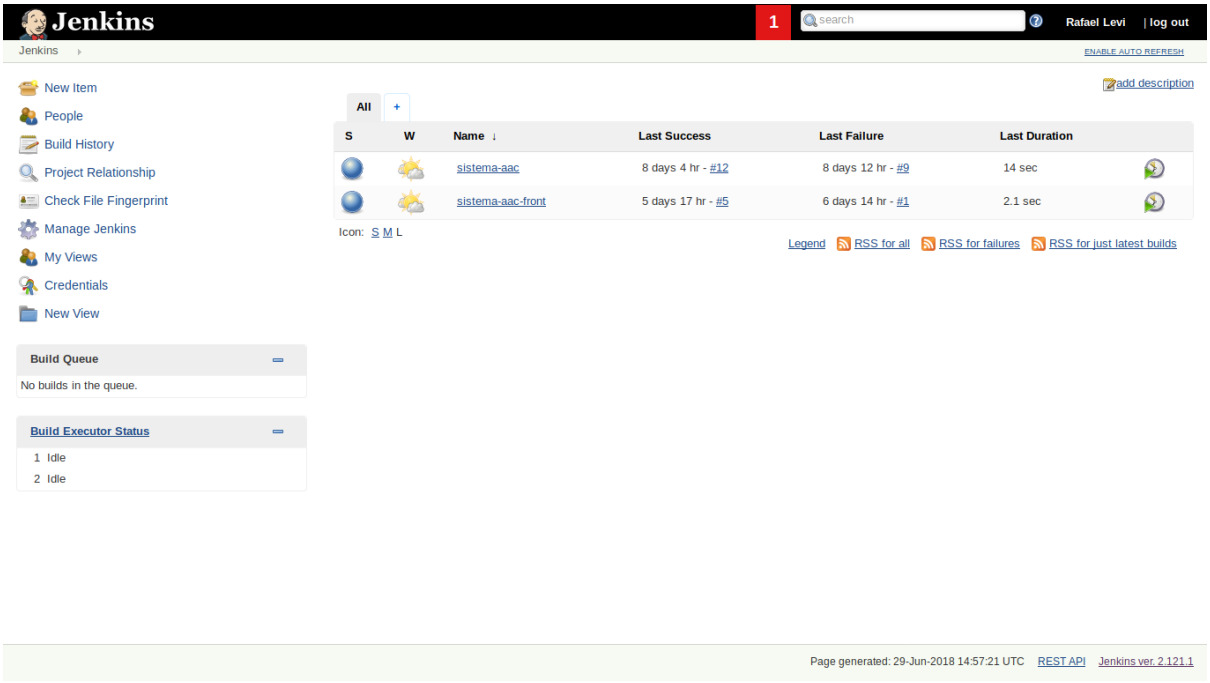

Figura 4.7: Interface web do Jenkins
Fonte: Próprio autor

4.5 *Front-end*

O front-end é um projeto feito com Javascript e php. O framework Bootstrap foi responsável por gerenciar a parte de design do layout, juntamente com a biblioteca Jquery para facilitar o código Javascript. Pode-se observar o projeto do front-end no código 4.6 e também disponível no GitLab, no link já citado anteriormente.

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4    <head>
5
6      <meta charset="utf-8">
7      <meta name="viewport" content="width=device-width, initial-scale=1,
           shrink-to-fit=no">
8      <meta name="description" content="">
9      <meta name="author" content="">
10
11     <title>AAC</title>
12
13     <!-- Bootstrap core CSS -->
14     <link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet"
           >
```

```
15
16      <!-- Custom fonts for this template -->
17      <link href="vendor/font-awesome/css/font-awesome.min.css" rel="
            stylesheet" type="text/css">
18
19      <!-- Plugin CSS -->
20      <link href="vendor/magnific-popup/magnific-popup.css" rel="
            stylesheet" type="text/css">
21
22      <!-- Custom styles for this template -->
23      <link href="css/freelancer.min.css" rel="stylesheet">
24
25      <!-- back-end scripts -->
26      <script src="back.js"></script>
27
28  </head>
```

Código 4.6: Projeto do Front-end

A interface web pode ser acessada através do link <http://www.sistema-aac.duckdns.org/AAC2>. A resposta do navegador é exibida na figura 4.8

Figura 4.8: Sistema AAC
Fonte: Próprio autor

Capítulo Cinco

Resultados e discussões

A seguir, trataremos dos resultados gerados e discussões sobre o processo de construção do modelo proposto, levando em consideração tecnologias, as dificuldades e como foram superadas. Assim ficará registrado as contribuições para gerações de modelos AAC que visem solucionar, através do uso de tecnologia, a comunicação entre pacientes com dificuldade de fala e seus famliares e cuidadores da área da saúde.

5.1 Microcontrolador

Dentro do contexto previsto no projeto, ser de baixo custo, de fácil substituição de peças e usabilidade. A primeira possibilidade analisada foi o uso apenas de software, o que evidenciou a necessidade de um computador, Smart TV, ou Smart Phone para servir de base para esse possível software. Todos esses equipamentos são de alto valor, analisando a possibilidade de haver sempre uma Smart TV em cada hospital, foi percebido que isso seria um entrave e que está fora da realidade principalmente dos hospitais públicos. Para o uso de um computador acoplado à TV, também entraria no problema da necessidade de um computador por parte do hospital ou ainda, pelo paciente. O que também foi descartado logo em seguida.

O uso do Smart Phone pessoal do paciente para base do software, pareceu inicialmente uma boa opção, mas quando foi analisado o ambiente, que seria uma ala hospitalar, possivelmente haveria mais de um paciente para ser atendido através do uso do sistema, foi notado que a quantidade de aparelhos celulares poderia gerar conflitos durante os atendimentos, já que, diversos pacientes poderiam fazer solicitações simultaneas, dificutado a identificação dos pacientes por parte dos enfermeiros. Sem mencionar a problemática de gerenciamento da fila para uso do sistema, já que haveria uma cópia para cada Smart Phone. O uso do Raspberry pi acoplado à TV contempla tanto o baixo custo, quanto o fato de que com qualquer aparelho de televisão seria possível utilizar o sistema, o que traz maior acessibilidade, além de resolver o uso por parte dos pacientes. Optou-se então por utilizar um microcontrolador para desenvolver o sistem, já que é uma tecnologia de baixo custo, podendo também ser falcimente dinfundida.A escolha do Raspberry pi em detrimento dos outros microcontroladores se deu principalmente pelo fato deste possuir um sistema operacional (o que facilitaria a usabilidade do usuário)e não apresentar dificultades de instalação quando conectado a uma televisão.

A solução modelada então partiu como um sistema embarcado, ao qual foi utilizado o

Raspberry pi como base. Já definido onde irá rodar o sistema, a próxima dificuldade foi a infraestrutura para disponibilizar o sistema.

5.2 Infraestrutura embarcada

Para base do sistema, foi proposto o uso do banco de dados nativo do Raspberry, o Sqlite3. Após a leitura nos artigos apresentados na revisão da literatura, confirmamos que para a situação é o banco que dará maior agilidade nas consultas, por conta do processamento pontual. O Sqlite3 é um banco que gera um arquivo para leitura e escrita e atua sob o sistema operacional apenas nas chamadas de acesso. As opções como MySql, PostgreSQL, dependeria mais do processamento do equipamento assim como o uso da memória RAM, que são limitados. Em contrapartida a gravação do banco em um arquivo, evidencia a necessidade de nos preocupar com a segurança do arquivo gravado.

Para esse MVP (*Minimum Viable Product* - Mínimo Produto Viável), presupomos que o microcontrolador possa ser adquirido pelo hospital, por um conjunto de pacientes, ou mesmo por um único paciente. Isso nos leva à primeira situação, onde o hospital adquire o microcontrolador, coloca o cartão de memória já com o sistema operacional e o sistema proposto, então esse raspberry ficaria atrás da tv ou acoplado a ela, quase não sofrendo acesso direto de nenhum usuário, fora os pacientes que teriam disponível apenas a tela inicial do sistema travada no modo tela cheia, não podendo assim acessar outros recursos do sistema operacional, garantindo a mínima segurança de acesso aos dados ali registrados.

Caso o microcontrolador seja comprado por um ou mais pacientes, deve haver um profissional de tecnologia para fazer a gravação da imagem no cartão de memória. Esse profissional deve executar uma atividade pontual que é preparar o raspberry pi para ser utilizado por um grupo de usuários, esse processo não exige nenhum serviço especializado, já que uma vez gravado no cartão de memória, o sistema autoexecuta e não precisa de manutenção por parte desse profissional. Cabe ao hospital apenas incluir na política de uso e regras, a possibilidade de compartilhamento de um equipamento que irá apenas utilizar a televisão do hospital como tela de visualização do sistema. Para isso seria necessário haver um termo de uso e compartilhamento de dados que autorizasse o armazenamento dos dados do paciente nesse cartão de memória, que pode e deve estar incluso nos termos de uso.

5.3 Desenvolvimento embarcado e web

A linguagem de programação nativa para o Raspberry pi é o Python. Foi desenvolvido então um protótipo nessa linguagem de programação. Era preciso um framework do Python para criação da interface visual do sistema, haviam opções para web, como o Flask, por exemplo e para interface nativa, o PyGame a exemplo. Por condução da experiência do orientador, foi feito inicialmente uma versão em PyGame.

Houve dificuldades para trabalhar com esse framework, por falta de experiência do desenvolvedor. O PyGame não possui um tratamento nativo para reponsividade, que é a capacidade do sistema de se adaptar a diversas telas. Foi feito então em todas as telas, uma proporção entre o tamanho total da tela e os objetos visuais. Para superar os desafios propostos por esse framework, foi feito um estudo com o livro (MCGUGAN, 2007). Após o desenvolvimento dessa versão, foi observado que para melhor usabilidade e compatibilidade com diversos dispositivos (via web) seria melhor ter uma versão com desenvolvimento web, para suprir a necessidade de apresentação e visualização do sistema.

O protótipo de desenvolvimento web foi feito em uma infraestrutura em nuvem, através do GCP (*Google Cloud Platform*). Foi criado uma instância em nuvem de um servidor Linux para oferecer o serviço web. A infraestrutura em nuvem funciona como um servidor na internet disponível como base para oferecer o serviço web, com o projeto de desenvolvimento. A principal diferença de um serviço de nuvem como o GCP e um servidor web é que o serviço da nuvem é responsivo, ou seja, o processamento e armazenamento pode ser aumentado diante à demanda. Existem outras opções como AWS (*Amazon Web Service*) e Microsoft Azure. Já havia uma infraestrutura disponível por parte do desenvolvedor, além de maior familiaridade, foi o que definiu a escolha por GCP. Os serviços de nuvem citados acima são bem similares e todos possuem um período de uso gratuito, no caso do GCP pode ser utilizado por um ano.

O projeto web foi feito tanto oferecendo um serviço em um servidor na internet, como de forma embarcada no raspberry, utilizando o próprio navegador do microcontrolador para visualizar o projeto web. A versatilidade do projeto web foi o que permitiu uma melhor elaboração do protótipo nessa versão. O próprio dispositivo, através do sistema operacional Raspbian (distribuição Linux) permite a instalação de um servidor web, que em nosso caso foi o apache 2, para então poder guardar a pasta do projeto web e por fim ser visualizada no navegador.

5.4 Sistemas de Linguagem Alternativa

Houve dificuldades relacionadas à importação dos sistemas de símbolos de comunicação alternativa. Muitos estão sob domínio de empresas privadas e é necessário fazer um cadastro e em alguns casos até pagar para obter acesso ao conjunto de símbolos. Os que possuem domínio público, a exemplo o BlisSymbols, está em outro idioma e seria necessário um cuidadoso trabalho de tradução para manter a coerência entre as palavras para formação da frase. Não foi encontrado nenhum sistema de linguagem em português que possua base de dados gratuita e com API de integração, como colaboração do ponto de vista tecnológico permitiria um grande avanço para desenvolvimento se houvessem bases de dados públicas via API para integração com outros sistemas. A base de dados Arasaac possui suporte ao idioma português, mas possui acesso apenas via download dos pictogramas, sem suporte à API. Uma API permite a integração de um sistema com outros, dando acesso aos dados de forma sempre atualizada, permitindo assim, que se integre um sistema, à uma biblioteca de pictogramas, que é o caso das bases de dados AAC.

Capítulo Seis

Considerações finais

6.1 Conclusões

Sistemas de comunicação aumentativos e alternativos são utilizados para auxiliar pessoas que possuem alguma dificuldade de oralizar a frase. É necessário ainda utilizar um sistema de linguagem adequado, levando em consideração o cenário ao qual o problema está inserido, como aspectos sociais, culturais, necessidades específicas do usuário do sistema, assim como o meio em que será apresentado. Para isso o ideal é que um especialista, fonoaudiólogo avalie a situação do paciente e auxilie na implantação do sistema AAC.

Na literatura é possivel encontrar produtos que visam solucionar essa problematica. Entretanto, eles hora não conseguem atender todas as necessidades, hora possuem um custo elevado para sua aquisição.

Este trabalho tratou o problema de comunicação de pacientes com AAC. Para tal, desenvolveu-se um prototipo em sistema embarcado para solucionar essa problematica. Foi visto que um sistema embarcado se apresenta como uma possível solução como ferramenta integradora entre um conjunto de procedimentos e processos para comunicação e o interlocutor em questão que necessita de uma interface de comunicação efetiva. Também foi levado em consideração o baixo custo de implementação desse sistema, já que o hardware para implementação dessa tecnologia não possui custo elevado.

O primeiro prototipo desenvolvido apresentou um bom desempenho, o qual o paciente conseguia selecionar suas necessidades e o sistema expressava sua vontade. Contudo, esse sistema está limitado a utilização do Raspberry Pi.

O segundo protótipo foi pensado para superar a limitação do primeiro. Desenvolveu-se uma interface web, a qual realizavas as mesmas funções, com a diferença que poderia ser executado em qualquer dispositivo dotado de um *browser*.

A tecnologia aliada aos processos de comunicação e saúde das pessoas, se mostra cada vez mais importante e impactante em nossa sociedade, esse projeto pretende ocupar esse espaço, oferecendo uma solução que não dependa exclusivamente de uma determinada tecnologia e que possa de forma modularizada se adequar às possíveis necessidades de mudança de tecnologias, além de responder prontamente às mudanças de hábitos e costumes de cada usuário. Para podermos avançar com o sistema, precisamos em um futuro trabalho, aplicá-la em um ambiente de produção para que possamos tanto validar a so-

lução quanto adicionar funcionalidades que só serão demandadas, após o uso efetivo do mesmo.

6.2 Trabalhos futuros

Como trabalhos futuros, pretende-se contemplar interfaces que não necessitem do movimento do paciente, para abranger casos de paralisia total do corpo. Além disso, o protótipo deve ser experimentado em ambiente hospitalar, podendo assim, averiguar o seu real funcionamento e apurar novos requisitos para futuras versões do sistema.

Quanto as questões tecnológicas, podemos utilizar serviços de gerenciamento de containers que podem facilitar nosso processo de hospedagem, a exemplo o Kubernetes da Google ou o ECS da Amazon.

Referências Bibliográficas

ABRAMCZUK, Beatriz; VILLELA, Edlaine. A luta contra o avc no brasil. *ComCiência*, SciELO Brasil, n. 109, p. 0–0, 2009.

AGRAWAL, Nikhil; SINGHAL, Smita. Smart drip irrigation system using raspberry pi and arduino. In: IEEE. *Computing, Communication & Automation (ICCCA), 2015 International Conference on*. [S.l.], 2015. p. 928–932.

ASSOCIATION, American Speech-Language-Hearing *et al.* Augmentative and alternative communication: Knowledge and skills for service delivery. ASHA, 2002.

BARROS, Ana Paula Brandão; PORTAS, Juliana Godoy; QUEIJA, Débora S. Implicações da traqueostomia na comunicação e na deglutição. *Rev Bras Cir Cabeça Pescoço*, v. 8, n. 3, p. 202–207, 2009.

BHARATHAN, Raghuram. *Apache Maven Cookbook*. [S.l.]: Packt Publishing Ltd, 2015.

BOETTIGER, Carl. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, ACM, v. 49, n. 1, p. 71–79, 2015.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. Uml: guia do usuário: o mais avançado tutorial sobre unified modeling language (uml). *Rio de Janeiro: Campus*, 2000.

CAO, Hoang-Long *et al.* Probolino: A portable low-cost social device for home-based autism therapy. In: SPRINGER. *International Conference on Social Robotics*. [S.l.], 2015. p. 93–102.

CESA, Carla Ciceri; MOTA, Helena Bolli. Comunicação aumentativa e alternativa: panorama dos periódicos brasileiros. *Revista CEFAC*, Instituto Cefac, v. 17, n. 1, 2015.

DEITEL, Harvey M; DEITEL, Paul J. *Java, como programar. Ed.* [S.l.]: Pearson, 2009.

DEWAILLY, Ludovic. *Building a RESTful Web Service with Spring*. [S.l.]: Packt Publishing Ltd, 2015.

D'AMORE, Mariano; BAGGIO, Rodolfo; VALDANI, Enrico. A practical approach to big data in tourism: a low cost raspberry pi cluster. In: *Information and Communication Technologies in Tourism 2015*. [S.l.]: Springer, 2015. p. 169–181.

ERICH, GAMMA *et al. Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos. Trad. Luiz AM Salgado*. [S.l.]: Bookman, 2000.

GAVAS, Rahul *et al.* Affordable sensor based gaze tracking for realistic psychological assessment. In: IEEE. *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*. [S.l.], 2017. p. 746–750.

GODBOLT, Micah. *Frontend Architecture for Design Systems: A Modern Blueprint for Scalable and Sustainable Websites*. [S.l.]: "O'Reilly Media, Inc.", 2016.

GONÇALVES, Maria de Jesus. O significado da comunicação no atendimento ao paciente em uti: como o fonoaudiólogo pode ajudar? *O Mundo da Saúde*, v. 32, n. 1, p. 79–84, 2008.

GOSLING, James; JOY, Bill; STEELE, Guy. *The Java language specification*. [S.l.]: Addison-Wesley Professional, 2000.

GUEDES, Gilleanes TA. *UML 2-Uma abordagem prática*. [S.l.]: Novatec Editora, 2008.

GUTIERREZ, Felipe. *Pro Spring Boot*. [S.l.]: Springer, 2016.

HEMRAJANI, Anil. *Agile Java development with spring, hibernate and eclipse*. [S.l.]: Sams publishing, 2006.

HTTERMANN, Michael. *DevOps for developers*. [S.l.]: Apress, 2012.

IBGE. *Censo 2010*. 2010. Disponível em: <https://censo2010.ibge.gov.br/>.

JAIN, Sarthak; VAIBHAV, Anant; GOYAL, Lovely. Raspberry pi based interactive home automation system through e-mail. In: IEEE. *Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on*. [S.l.], 2014. p. 277–280.

JUNIOR, Almeida; NASCIMENTO, Wilson. Técnicas e práticas psicológicas no atendimento a pacientes impossibilitados de se comunicarem pela fala. *Psicologia Hospitalar*, Centro de Estudos Psicologia da Saúde da Divisão de Psicologia ICHCFMUSP, v. 12, n. 2, p. 24–44, 2014.

KANG, Hui; LE, Michael; TAO, Shu. Container and microservice driven design for cloud infrastructure devops. In: IEEE. *Cloud Engineering (IC2E), 2016 IEEE International Conference on*. [S.l.], 2016. p. 202–211.

LADD, Seth *et al. Expert Spring MVC and Web Flow*. [S.l.]: Springer, 2006. v. 1.

MARTINS, Sabrine Amaral. Aspectos da afasia multilíngue. *Distúrbios da Comunicação*, v. 28, n. 1, 2016.

MASSAUD, Rodrigo Meirelles. *Afasias*. 2018. Disponível em: <https://www.einstein.br/guia-doencas-sintomas/afasias>.

MCGUGAN, Will. *Beginning game development with Python and Pygame: from novice to professional*. [S.l.]: Apress, 2007.

MINETTO, Elton Luís. Frameworks para desenvolvimento em php. *São Paulo: Novatec*, 2007.

MOHANAPRAKASH, TA. Assisting echolalia (repetitive speech patterns) in children with autism using android mobile app. *IJAICT*, v. 1, n. 12, p. 928–933, 2015.

ROMBALDO, Carlos Alberto; SOUZA, Solange N Alves de; SOUZA, Luiz Sergio de. Persistence framework for object-relational database. In: IEEE. *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*. [S.l.], 2012. p. 1–7.

SAHANI, Mrutyunjaya *et al.* Web-based online embedded door access control and home security system based on face recognition. In: IEEE. *Circuit, Power and Computing Technologies (ICCPCT), 2015 International Conference on*. [S.l.], 2015. p. 1–6.

SAMESHIMA, Fabiana Sayuri; DELIBERATO, Débora. Habilidades expressivas de um grupo de alunos com paralisia cerebral na atividade de jogo. *Revista da Sociedade Brasileira de Fonoaudiologia*, Sociedade Brasileira de Fonoaudiologia, p. 219–224, 2009.

SCHLOSSER, Ralf W. *The efficacy of augmentative and alternative communication*. [S.l.]: Elsevier, 2003. v. 1.

SILVA, Francisco Assis da; PERRI, Carlos Renato de Souza; ALMEIDA, Leandro Luiz de. Desenvolvimento de uma ferramenta assistente para criação de aplicações crud em java na web. In: *Colloquium Exactarum*. [S.l.: s.n.], 2011. v. 2, n. 2, p. 70–82.

SMART, John Ferguson. *Jenkins: The Definitive Guide: Continuous Integration for the Masses*. [S.l.]: "O'Reilly Media, Inc.", 2011.

SMITH, Martine. *Literacy and augmentative and alternative communication*. [S.l.]: Elsevier, 2005. v. 2.

SPEARS, Carol L; TURNER, Vicki L. *Rising to new heights of communication and learning for children with autism: The definitive guide to using alternative-augmentative communication, visual strategies, and learning supports at home and school*. [S.l.]: Jessica Kingsley Publishers, 2010.

ZAMBON, Giulio. *Beginning JSP, JSF and Tomcat: Java web development*. [S.l.]: apress, 2012.

ZHAO, Cheah Wai; JEGATHEESAN, Jayanand; LOON, Son Chee. Exploring iot application using raspberry pi. *International Journal of Computer Networks and Applications*, v. 2, n. 1, p. 27–34, 2015.