UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Elcio Luciano Furtili Junior

UnespVoice: Desenvolvimento de uma Plataforma para Armazenamento e Gerenciamento de Dados em Pesquisas com Sinais de Fala Imaginada

São José do Rio Preto

2024

Elcio Luciano Furtili Junior

UnespVoice: Desenvolvimento de uma Plataforma para Armazenamento e Gerenciamento de Dados em Pesquisas com Sinais de Fala Imaginada

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Financiadora: CAPES - Prc. 88887.680185/2022-00

Orientador: Prof. Dr. Rodrigo Capobianco Guido

São José do Rio Preto, SP
2024

F992u

Furtili Junior, Elcio Luciano

UnespVoice: Desenvolvimento de uma Plataforma para Armazenamento e Gerenciamento de Dados em Pesquisas com Sinais de Fala Imaginada / Elcio Luciano Furtili Junior. -- , 2024

50 p. : tabs., fotos

Dissertação (mestrado) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências Farmacêuticas, Araraquara,

Orientador: Rodrigo Capobianco Guido

1. Sistemas de Computação. 2. Sistemas de Informação. 3. Inteligência Computacional. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Universidade Estadual Paulista (UNESP), Faculdade de Ciências Farmacêuticas, Araraquara. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Elcio Luciano Furtili Junior

UnespVoice: Desenvolvimento de uma Plataforma para Armazenamento e Gerenciamento de Dados em Pesquisas com Sinais de Fala Imaginada

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Financiadora: CAPES - Prc. 88887.680185/2022-00

Orientador: Prof. Dr. Rodrigo Capobianco Guido

Comissão Examinadora

Prof. Dr. Rodrigo Capobianco Guido
UNESP - Campus de São José do Rio Preto
Orientador

Profa. Dra. Veronica Oliveira de Carvalho
UNESP - Campus de Rio Claro

Profa. Dra. Luciene Cavalcanti Rodrigues
Faculdade de Tecnologia de São José do Rio Preto

São José do Rio Preto, SP
06 de Setembro de 2024

AGRADECIMENTOS

Acredito que essa parte é a mais complicada entre todo processo, agradecer não é só necessário, mas também primordial, pois sem essas pessoas que fizeram parte do processo não seria possível alcançar o êxito, independente do resultado final, pois pra mim mais do que essa última etapa o caminhar de toda essa jornada foi simplesmente incrível.

Agradeço a todos, eu só tenho agradecimentos a fazer, pai, mãe, esposa, irmãos, família, amigos, professores, orientador, Ibilce, tenho até receio de citar nomes e injustiçar alguém, mas sem dúvida nenhuma agradeço por ter conseguido chegar até aqui, 2 anos atrás eu era outra pessoa, apenas um estudante com sonho de ser um cientista e hoje sei que sou, e dedico completamente a isso, a ser um cientista, um pesquisador, e principalmente a ser um pai, pois esse será meu maior projeto.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - 88887.680185/2022-00, à qual agradeço.

RESUMO

Este estudo discute a criação e implementação da *unespVoice*, uma plataforma dedicada à organização e compartilhamento de informações provenientes de pesquisas sobre fala imaginada. A fala imaginada, também conhecida como imaginação verbal, é o processo mental no qual alguém “ouve” ou “fala” com sua própria voz na mente, sem emitir sons audíveis. Esse trabalho descreve o desenvolvimento de um sistema que não só armazena os dados de forma eficaz, mas também facilita sua reutilização em diferentes pesquisas, incentivando a colaboração e troca de recursos na comunidade científica. A pesquisa ressalta a importância de protocolos rigorosos para manipulação e armazenamento dos dados, visando superar as deficiências nos sistemas existentes que dificultam a reutilização. A *unespVoice* é apresentado como uma solução para reduzir custos e ampliar o acesso a grandes volumes de dados, fundamentais para o progresso das pesquisas em processamento de fala.

Palavras-chave: Fala Imaginada, unespVoice, Armazenamento de Dados, Reutilização de Dados, Processamento de Fala

ABSTRACT

This study discusses the creation and implementation of unespVoice, a platform dedicated to organizing and sharing information from research on imagined speech. Imagined speech, also known as verbal imagination, is the mental process in which someone "hears"or "speaks"with their own voice in their mind without producing audible sounds. This work describes the development of a system that not only stores data effectively but also facilitates its reuse in different research projects, encouraging collaboration and resource sharing within the scientific community. The research highlights the importance of rigorous protocols for handling and storing data, aiming to overcome deficiencies in existing systems that hinder reuse. unespVoice is presented as a solution to reduce costs and expand access to large volumes of data, which are essential for the advancement of speech processing research.

Keywords: Imagined Speech, unespVoice, Data Storage, Data Reuse, Speech Processing

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Este texto acompanhará o processo de desenvolvimento, avaliação e montagem de processos de armazenamento de dados gerados por pesquisas de fala imaginada. O objetivo é promover a reutilização de dados e a redução de custos em todo o processo de distribuição desses arquivos para múltiplas pesquisas, incluindo aquelas que necessitam de grandes volumes de dados.

A proposta deste projeto é desenvolver a comunidade do processamento da fala, criando um sistema de arquivamento e distribuição de dados gerados no processo de desenvolvimento de pesquisas que envolvem técnicas de fala imaginada. Para contextualizar o projeto, é essencial explicar o conceito de fala imaginada, ou imaginação verbal, que é um processo mental em que uma pessoa ouve ou fala com sua própria voz na mente, sem emitir sons reais.

No campo da computação, é necessário apresentar uma prova de conceito que associe a fala propriamente dita com a fala imaginada, criando um padrão que demonstre que o ato de falar e pensar na mesma palavra gera um padrão nas ondas cerebrais. Para isso, coletamos gravações das vozes com os tempos das palavras ditas e arquivos de eletroencefalografia.

No decorrer das explicações, será demonstrado o que é a fala imaginada, como estão sendo montados os protocolos de manipulação e armazenamento, e por que esse processo está sendo criado dessa forma. A pesquisa tem como base as falhas no armazenamento, que acabam gerando a impossibilidade de reutilização dos dados de outros pesquisadores. Do ponto de vista apresentado, essa situação se torna um inconveniente, e a reutilização desses dados não só pode ser benéfica como também é incentivada.

Outro ponto de grande relevância a ser avaliado neste projeto é o fato de que ele visa a criação de uma comunidade de fala imaginada, oferecendo uma ferramenta de armazenamento e facilidade de distribuição. Além disso, será demonstrado o motivo e o processo de desenvolvimento dessa plataforma. O intuito é utilizá-la como um meio para um fim, focando na disseminação e facilitação de acesso a esses dados para os pesquisadores, sem perder o foco no produto final: a possibilidade de disseminação e acesso facilitado aos dados.

Capítulo 2

Fundamentação Teórica

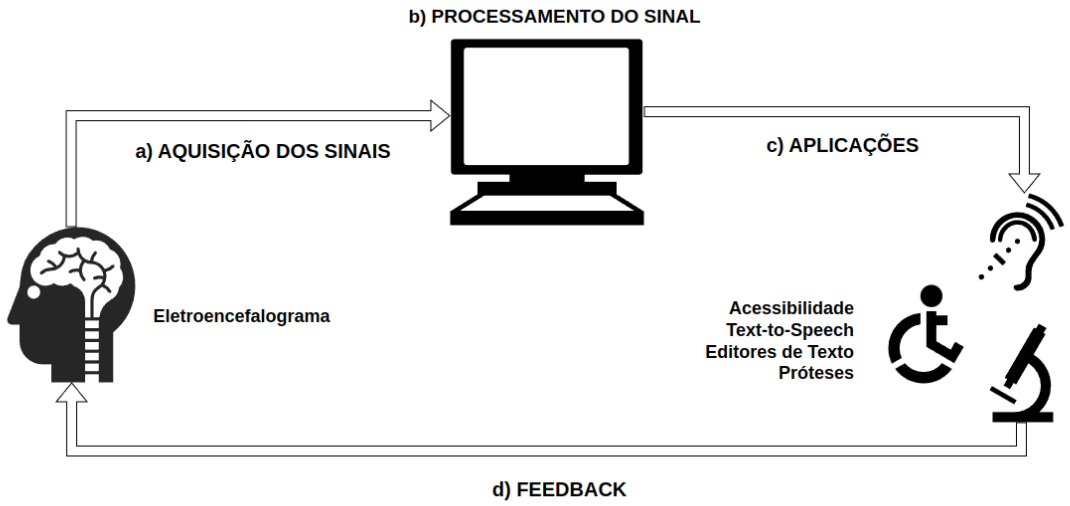
Neste capítulo serão abordados conceitos teóricos fundamentais para o entendimento da plataforma e suas tecnologias e como será desenvolvido a comunidade proposta no projeto. Em seguida, serão abordadas as principais técnicas utilizadas para desenvolver o sistema e as referências que foram utilizadas para as pesquisas. Adicionalmente, constam também discussões envolvendo trabalhos relacionados documentados na literatura.

2.1 Interface Cérebro-Computador

O ICC (Interface Cérebro-Computador) se trata de uma forma de transitar os dados de aparelhagem que capta informações do cérebro e permite um processamento digital por meio de uma transcrição e processamento de dados lógicos das aparelhagens (em especial eletroencefalograma), um exemplo que pode ser visualizado na Figura 2.1. Dessa forma, os dados captados do cérebro tornam-se arquivos digitais, permitindo o armazenamento em arquivos [1].

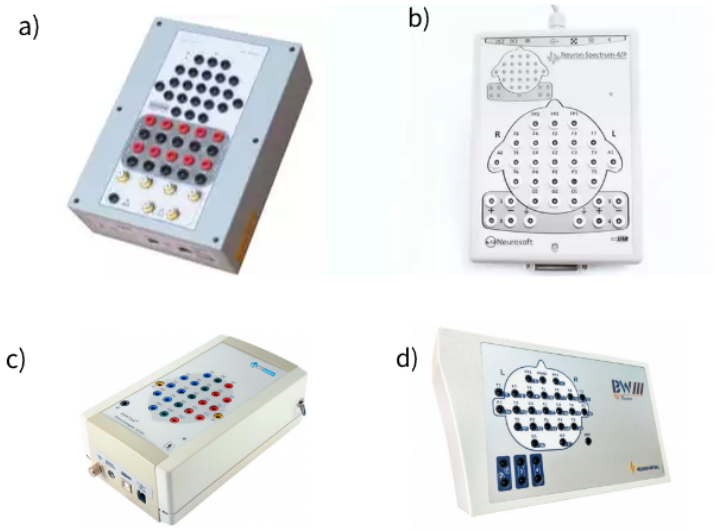
Processar esses dados e poder armazená-los é um dos pontos centrais, pois, ao armazená-los, é possível criar padrões e protocolos para que esses dados, uma vez gerados, possam ser transferidos, processados, tratados e até mesmo modelados de acordo com as necessidades dos trabalhos para os quais serão usados.

Existem vários modelos de eletroencefalograma disponíveis no mercado, todos consistindo em executar atividades muito similares, tendo como atividade fundamental analisar processos eletrofisiológicos cerebrais, alguns deles pode ser visualizado na Figura 2.2. Há diversos modelos e marcas para atender a várias necessidades, e como não é questão aprofundar nesse tema e destrinchar todos os modelos e todas as suas aplicações, diferenciamos os tipos pelos arquivos que são gerados. Nesse ponto, é preciso controlar, no momento do armazenamento, os protocolos utilizados para gerar esses arquivos, dado que se torna inviável mapear os milhares de tipos de aparelhos que existem. Isso é feito por meio da forma como os dados são coletados pela interface cérebro-computador.



Fonte: Autoria Própria

Figura 2.1: Modelo de Interface Cérebro-Computador: (a) Captação por meio de sensores do EEG, (b) Processamento dos dados de forma digital, (c) Envio para possíveis aplicações, (d) Retorno do que foi processado



Fonte: Autoria Própria

Figura 2.2: Modelos de EEG: (a) Emsa TIEEG, (b) Neurosoft, (c) Vertex SC823, (d) Brain Wave III Basics

2.2 Tipos de arquivos

Devido à quantidade desconhecida de aparelhos de eletroencefalograma e ao fato de que a ideia geral é criar um sistema de armazenamento de projetos de fala imaginada, será necessário validar alguns pontos primordiais quando o assunto é arquivos que serão armazenados, como exemplo na Figura 2.2 cada aparelho possui especificações e modelos de arquivos diferentes.

Os arquivos provenientes dos aparelhos de eletroencefalografia possuem diversas marcas e modelos, o que faz com que os tipos de arquivos sejam variados e possuam especificações distintas, entre eles os mais comuns estão na Tabela 2.1.

Durante o desenvolvimento da parte documental desta dissertação, dados provenientes de artigos científicos foram tabulados para embasar a pesquisa. No entanto, ao acessar os distribuidores dos aparelhos de eletroencefalograma e validar as respectivas documentações técnicas, foi possível obter resultados mais precisos para os dados previamente tabulados, aumentando a confiabilidade e acurácia das informações utilizadas e a Tabela 2.1 foi atualizada com base nesses dados.

Por isso, foi necessário, a princípio, mapear os tipos de arquivos mais comuns para tomar decisões mais precisas na programação. Com isso, foi gerada uma listagem de especificações, conforme pode ser visualizado na tabela de exemplos mapeados [2].

A fala imaginada demanda os seguintes arquivos: um arquivo de áudio gravado e um arquivo de eletroencefalografia, sendo também necessária a associação desses dois arquivos, como a temporização das palavras ditas para comparar com a eletroencefalografia [3].

Para contornar esse problema e facilitar o armazenamento dos dados de forma que eles sejam dados brutos, mas específicos e funcionais para múltiplos ambientes de testes, será utilizado, neste projeto, campos de metadados. Ao armazenar, é criado um ambiente generalizado e organizado, gerando metadados para esse aglomerado de arquivos, nos quais se armazenam o cabeçalho com as informações de nome de arquivo, tamanho, extensão, e os arquivos coletados. Para simplificar, pode-se visualizar como esses arquivos serão englobados, lembrando que a demonstração é um modelo generalizado para servir em todos os ambientes [4].

Um complemento importante é que todos os arquivos são brutos, não recebem qualquer tratamento em seus dados. A única função que a plataforma tem como base é a organização, mapeamento e armazenamento, possibilitando toda e qualquer interação com esses arquivos. Dessa forma, ela se encaixa tanto em futuras aplicações que demandam um arquivo limpo, que será tratado pelo receptor, quanto em aplicações que exigem arquivos brutos, pois a plataforma já os disponibiliza dessa forma, o modelo de arquivo juntamente com seus metadados foi criado justamente para essa necessidade, conforme está na Figura 2.3.

O modelo apresentado na Figura 2.5 representa a totalização e um formato de

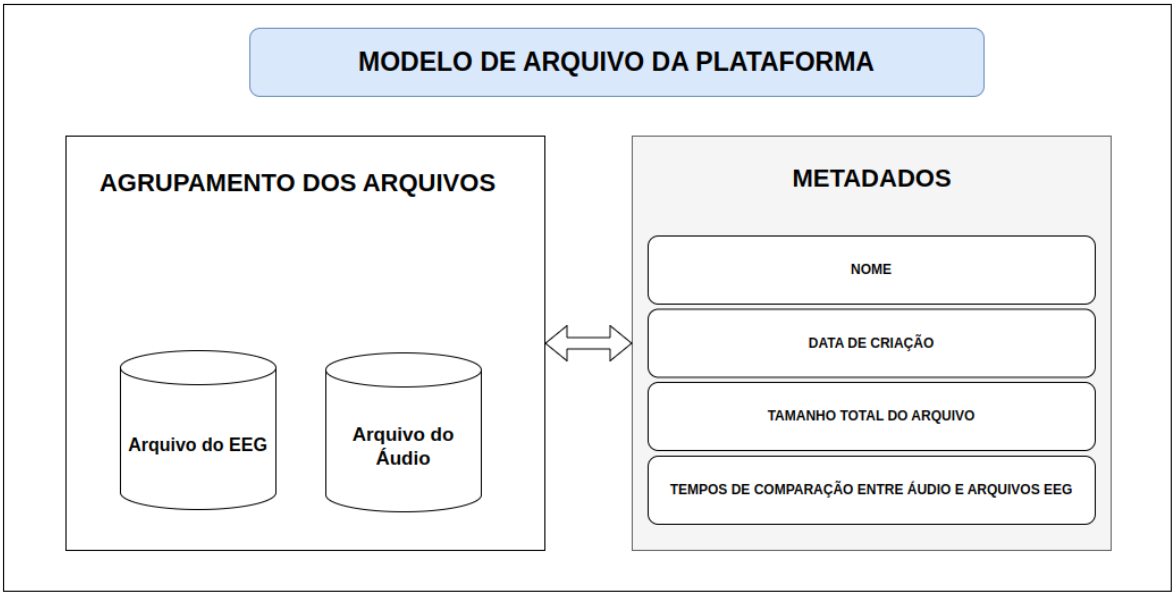
Formato	Descrição	Metadados	Marcação /Evento	Uso Comum
.EDF	Formato padrão para dados de EEG e sinais bioelétricos, suportando múltiplos canais de sinal e marcações	Sim	Sim	EEG e outros sinais bioelétricos
.EDF+	Extensão do EDF, inclui anotações adicionais e informações de eventos	Sim	Sim	Dados de EEG com anotações adicionais
.BDF	Semelhante ao EDF, associado a sistemas de aquisição de dados da BioSemi	Sim	Sim	Sistemas de aquisição de dados BioSemi
.CDF	Usado para armazenar dados EEG e outros dados científicos, incluindo metadados e eventos	Sim	Sim	Diversos tipos de dados científicos, incluindo EEG
.MAT	Arquivos MATLAB usados para armazenar dados científicos, incluindo EEG, no ambiente MATLAB	Sim	Sim	Processamento e análise de dados EEG no MATLAB
.TXT, .CSV	Formatos de texto padrão, valores separados por vírgulas; podem não conter todos os metadados e informações detalhadas do EEG	Limitado	Limitado	Exportação simples de dados EEG

Tabela 2.1: Comparação de Formatos de Arquivo para Dados EEG

arquivo desenvolvido para atender às necessidades do sistema, atualmente denominado *".unpv"*. Dentro desses arquivos compactados, encontram-se os dados brutos de fala imaginada, bem como os metadados, armazenados em formato *".xml"*, que tabulam todas as informações de cada captação, incluindo seus respectivos tempos de execução da fala, conforme ilustrado na Figura 2.4.

2.3 Integração de fala e sinais biológicos gerados por EEG

Um dos problemas encontrados durante o desenvolvimento é justamente como integrar esses dados uns aos outros. Nos projetos que utilizam a fala imaginada encontrados na internet, os arquivos são frequentemente apenas armazenados em pas-



Fonte: Autoria Própria

Figura 2.3: Modelo de Arquivo e Modelagem dos Metadados do Arquivo da Plataforma

tas similares ou, em muitos casos, de forma ainda menos organizada.

Buscando alguns artigos e pesquisas, entre eles [5] [6], foi validado que possuem um medidor de compatibilidade por temporização. Esse medidor utiliza como base para comparação o momento da fala com o momento da captação da fala imaginada, permitindo comparar diretamente como um tem impacto no outro. Trabalhando com modelos de algoritmos baseados em tempos de reação e validando vários aspectos neurais, mesmo não sendo o foco aqui, é importante ter essa noção para que, no momento de armazenar esses dados, seja possível criar uma forma de organizá-los com o máximo de informações possíveis, sem criar uma rotina massante e problemática para os usuários.

Com isso em mente, foi desenvolvido um método de amostragem de tempo que seria uma tabela que guarda dados de temporização tanto dos arquivos de áudio como dos arquivos de fala imaginada, como se fosse uma trilha temporal desses arquivos. Isso facilita possíveis usos futuros desses dados. Para melhor visualização, valide como é o processo que liga um arquivo ao outro e gera uma trilha temporal para ambos na Figura 2.5.

2.4 Tecnologias de armazenamento

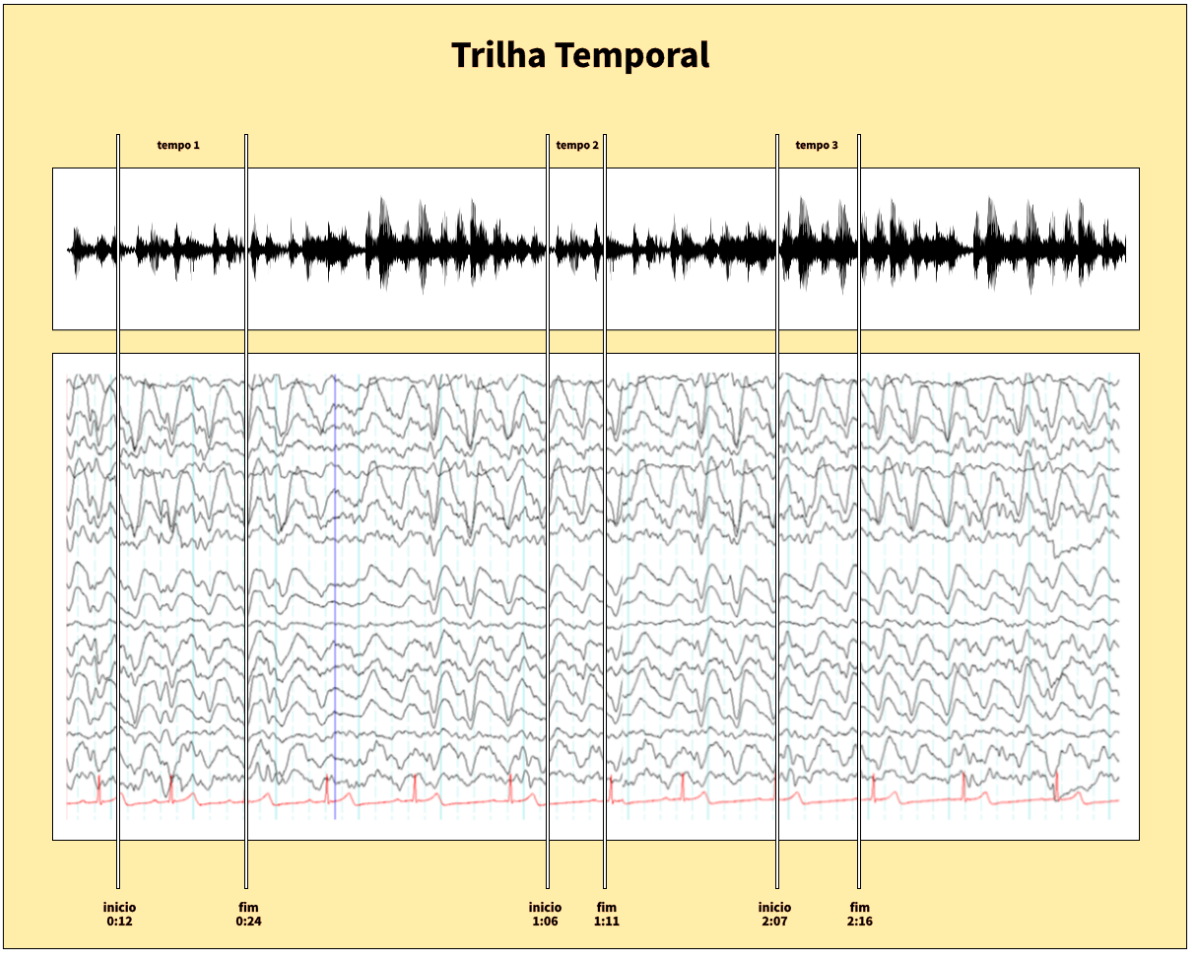
Atualmente, com as tecnologias e técnicas disponíveis, contamos com diversas formas de armazenamento de arquivos, desde as mais simples, como uma organização por algum tipo de ordem específica, até as organizações mais intrincadas, como compressão, armazenamento em camadas e tipos de sistemas de arquivos avançados. O ponto primordial do modelo necessário para o desenvolvimento da plataforma

```xml
<?xml version="1.0" encoding="UTF-8"?>
<file>
    <name>example_data.xml</name>
    <creation_date>2024-09-16</creation_date>
    <total_size>1024KB</total_size>

    <times>
        <time>
            <start_time>08:00:00</start_time>
            <end_time>12:00:00</end_time>
        </time>
        <time>
            <start_time>09:00:00</start_time>
            <end_time>11:00:00</end_time>
        </time>
        <time>
            <start_time>10:00:00</start_time>
            <end_time>13:00:00</end_time>
        </time>
    </times>
</file>
```

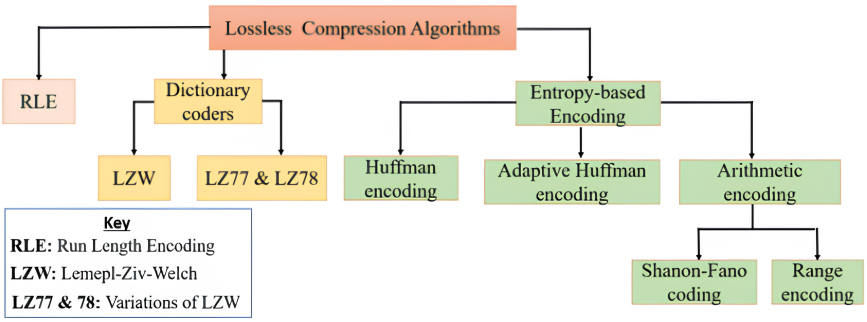
Fonte: Autoria Própria

Figura 2.4: Modelo do Arquivo XML de Metadados



Fonte: Autoria Própria

Figura 2.5: Modelo de Divisão Temporal dos Arquivos e Funcionamento da Trilha Temporal



Fonte: Performance Evaluation of Data Compression Algorithms for IoT-Based Smart Water Network Management Applications [8]

Figura 2.6: Tipos de Algoritmos de Compressão *Lossless*

é o modelo mais simplista e menos nocivo para os arquivos, deixando-os o mais bruto possível para que não impactem em nenhuma futura aplicação para esses dados. Por isso, foi adotado o modelo *"Lossless"*, que é o modelo de compressão sem perda, capaz de diminuir o espaço de armazenamento sem perder dados no processo [7], no artigo citado se trata de técnicas voltadas para dados comprimidos em redes *Wireless*, mas como se trata de uma técnica de modificação e transmissão de dados o processo para armazenamento é muito similar.

As compressões sem perdas são empregadas em situações onde é crucial que os dados originais e os descomprimidos sejam idênticos, ou onde qualquer alteração nos dados iniciais impactaria significativamente os resultados. Por exemplo, os dados gerados por um sistema de monitoramento de abastecimento de água devem ser completamente restaurados após a compressão, sem perda de informações. Qualquer perda de dados ou números nas leituras dos sensores alteraria as informações, afetando gravemente a análise e a tomada de decisões. Portanto, este estudo foca na compressão de dados sem perdas. Existem diversas técnicas de compressão de dados sem perdas, como mostrado na Figura 2.6, que vão desde a codificação por comprimento de execução (RLE) até a codificação baseada em entropia. Neste estudo, avaliamos o desempenho dos algoritmos RLE, LZW, *Huffman* e *Shannon-Fano*, selecionados de cada subcategoria conforme apresentado na Figura 3. Embora esses algoritmos gerem resultados diferentes, todos se baseiam no princípio fundamental de eliminar a redundância dos dados originais.

Em relação aos dados já existentes e disponíveis por meio de vários artigos e dissertações de pessoas que já pesquisaram esse assunto e precisaram gerar esses dados para validar suas pesquisas, essas informações não seguem qualquer padrão, nem mesmo na armazenagem ou padrão de organização. Esses dados, ao término das pesquisas, são disponibilizados apenas por convenção, às vezes em um servidor na nuvem ou um endereço dedicado que se perde com o tempo. Comparando diversos desses armazenamentos, nota-se que não existe um padrão, o que gera uma escassez desses dados quando o assunto é reutilização. São arquivos, dados e informações perfeitos para serem reutilizados inúmeras vezes e criar grandes massas de dados que poderão ser usadas para múltiplos fins, mas como o armazenamento

se torna precário, esses dados se perdem completamente [8].

A escassez desses dados tem como pilares dois grandes fatores: o péssimo modelo de armazenamento adotado pela grande maioria dos pesquisadores de fala imaginada, que, diga-se de passagem, não são todos da área da computação, e o fato de que a aparelhagem para executar tais pesquisas é muito cara, o que não é acessível para todos. Como resultado, os dados computacionais se tornam apenas um meio para um fim e não um ponto primordial, como exemplos esses dois artigos já citados [5] [6].

O intuito é criar grandes massas de dados justamente para suprir a necessidade das pesquisas e de futuros pesquisadores que queiram trabalhar com essas informações, seja para aplicações simples ou para grandes redes neurais e inteligências artificiais. O objetivo é possibilitar que essas aplicações alcancem resultados primorosos e precisos com acesso a uma base sólida de informações.

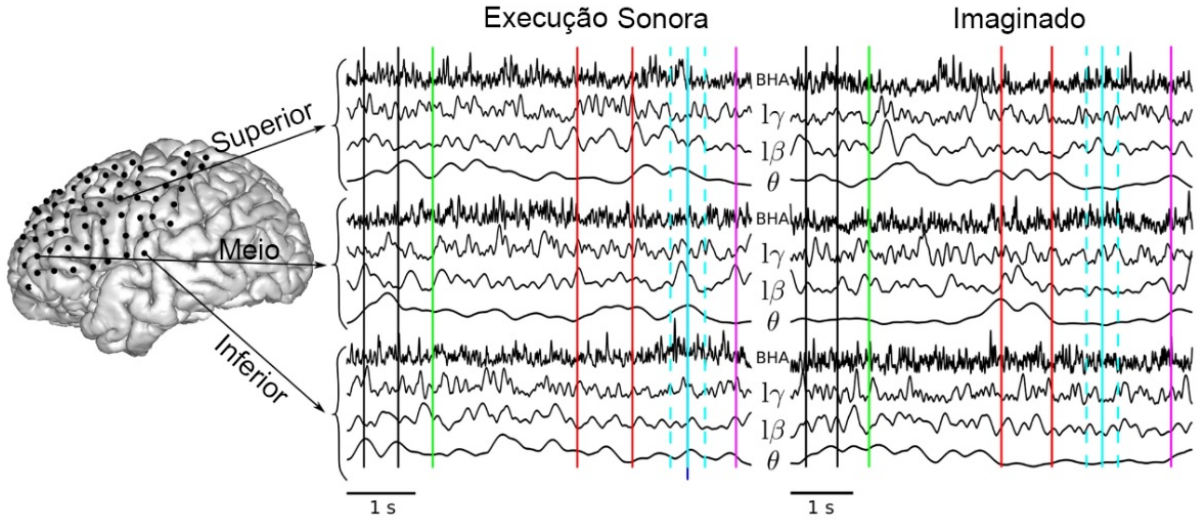
2.5 Desenvolvimento de massa de dados

O intuito geral de toda a aplicação é desenvolver uma base sólida de dados que possibilitará o uso das técnicas mais recentes de análise e interpretação desses dados. Atualmente, contamos com formas de manipulação e busca de padrões em dados robustos e poderosos, que permitem visualizar informações ressonantes que não seriam possíveis sem essas ferramentas. Inteligência artificial generativa, redes neurais intrincadas, aprendizado de máquina, entre outros modelos de trabalho com dados, todos necessitam de uma coisa em comum: uma grande quantidade de dados. Quanto maior a massa de dados, mais precisas se tornam as aplicações [9].

A fala imaginada, como já mencionado anteriormente, é mais uma forma de lidar com os dados do que uma técnica computacional de fato. Nesse processo, é necessária a fala propriamente dita e os impulsos neurais no momento em que essa fala foi executada, assim tendo o resultado físico e cerebral do ato de falar. Com base nisso, cria-se um padrão e, depois de etapas de treino, trabalha-se apenas com os impulsos neurais, eliminando a necessidade da fala de fato. Esse modelo de aplicação tem várias formas de utilização, em sistemas de acessibilidade, universos digitais e outras aplicações em diversos ambientes [10].

Os modelos mais simples de interpretação desses dados envolvem fatiar os tempos utilizando medidas neurais e espaços de confirmação, até mesmo de falhas de leitura. Conforme mostra a Figura 2.7, é possível visualizar esses espaçamentos. Logicamente, isso pode ser alterado de acordo com a necessidade ou teoria do projeto em questão. Esta demonstração serve apenas para gerar um modelo de visualização de como essa análise é feita, proporcionando uma ideia geral de como uma grande massa de dados pode ser benéfica nas pesquisas mais abrangentes e minuciosas desses dados [5].

A aplicação como um todo leva em sua base um modelo de micro-serviços. Até mesmo quando o assunto é a base de dados, as informações foram separadas em



Fonte: Autoria Própria

Figura 2.7: Demonstração dos Espaçamentos na Utilização da Fala Imaginada

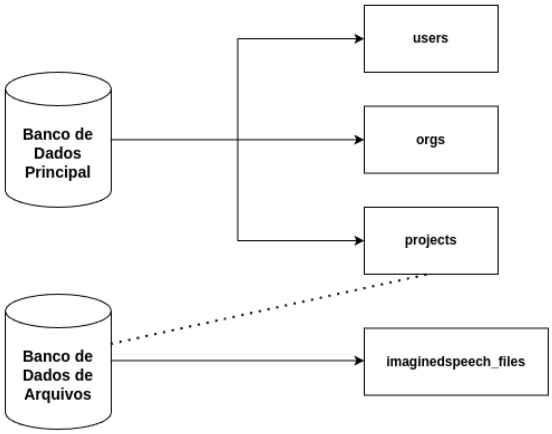
essenciais, necessárias e não primordiais para que as instituições que decidam usar e implantar a plataforma em suas formas de organizar seus projetos possam compartilhar esses dados com outras instituições apenas sincronizando os dados entre as bases [11]. O modelo da base de dados foi criado justamente com o intuito de facilitar e suprir a questão da disseminação dos dados. Com isso em mente, a base foi criada da forma como pode ser visualizada na Figura 2.8.

Dessa forma, como foi desenvolvida, é possível personalizar, incrementar, remover e moldar de acordo com a necessidade de cada instituição, e a base primordial de dados não será afetada em uma possível necessidade de sincronizar dados entre múltiplas organizações, tornando a base sempre sólida e confiável.

2.6 Arquitetura flexível para acesso e reutilização de dados em pesquisas com fala imaginada

Tanto a base de dados quanto a aplicação foram desenvolvidas de forma que aplicações externas possam acessar os dados sem a necessidade de utilizar a aplicação de fato para alimentar aplicações terceiras. Dessa forma, é possível consultar, capturar e reutilizar esses dados sem a necessidade de várias requisições. Justamente por esse motivo, a aplicação foi desenvolvida por meio da técnica de microserviços [11].

Desde a concepção e durante todo o desenvolvimento, a aplicação foi voltada para futuros usos em aplicações de inteligência artificial e aprendizado de máquina. Assim, ela se tornou bem maleável, possibilitando que, futuramente, todos que a utilizem possam criar aplicações baseadas nos dados coletados, bem como terceiros que possam se alimentar desses dados para criar novas aplicações e técnicas de



Fonte: Autoria Própria

Figura 2.8: Modelo dos Bancos de Dados, Banco de Informações Gerais e Banco de Arquivos

processamento [9].

A base da proposta é justamente compartilhar esses dados coletados e facilitar futuras pesquisas de forma simples para ambos os lados: para as organizações que coletam esses dados e para os pesquisadores que desejam trabalhar com a fala imaginada, mas que não possuem recursos para tal. Dessa forma, a aplicação resolve dois problemas com uma única solução, tornando a flexibilidade uma necessidade desde o momento de sua concepção.

Capítulo 3

A Abordagem Proposta

Apresenta-se, neste Capítulo, a estratégia geral proposta neste trabalho, com os devidos detalhes que permitem reproduzí-la.

3.1 Desenvolvimento da Plataforma

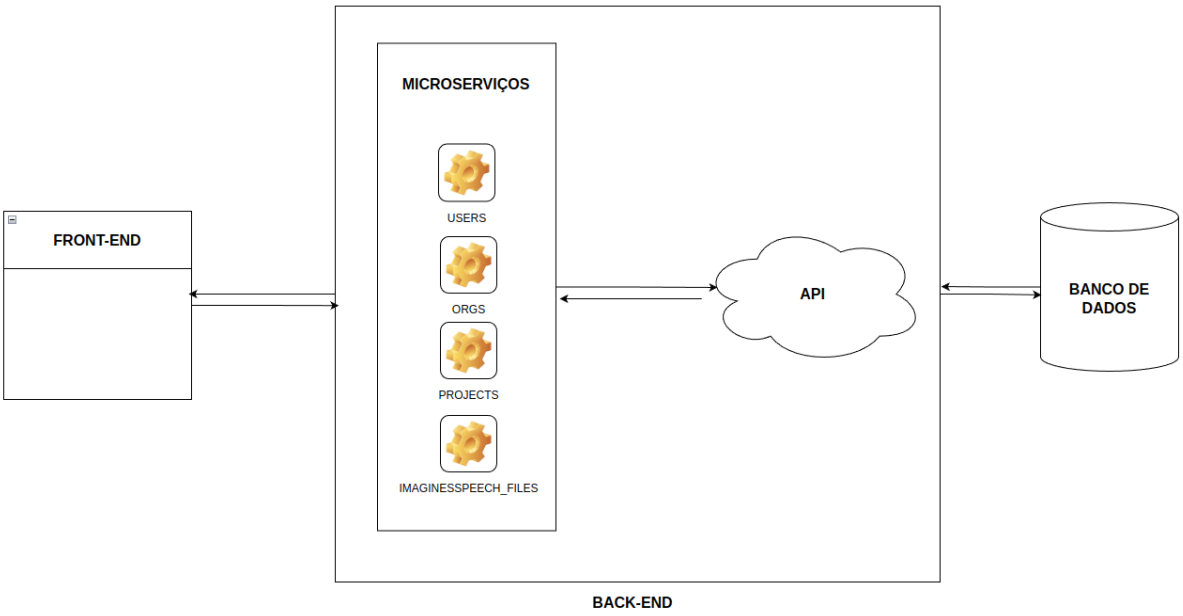
O desenvolvimento da aplicação tem como base o modelo de microsserviços, com cada uma de suas rotinas executadas separadamente. Ao serem requisitadas, essas rotinas enviam informações para uma fila de requisições, consultando registros na base de dados.

A aplicação possui dinamismo em suas partes, principalmente nas requisições e consultas que são feitas por meio de *APIs*, tanto para o *front-end*, que traz as rotinas de consulta, cadastros, alterações e todas as funções básicas de uma aplicação, quanto para aplicações terceiras por meio dos *Endpoints*. Isso significa que, embora tenha sido criado um *front-end* com as funções básicas do sistema, cada organização pode gerar um *front-end* de acordo com suas necessidades e padrões, pode ser visualizado na Figura 3.1.

Toda a aplicação será criada dentro de *Containers* utilizando o sistema de *Dockers*, conforme ilustrado na Figura 3.2, para gerenciar os projetos e criar um ambiente único de execução. Isso permitirá que, independentemente de quem execute o download, o ambiente no qual o sistema será executado será o mesmo, diminuindo as possíveis falhas que uma distribuição de software pode gerar.

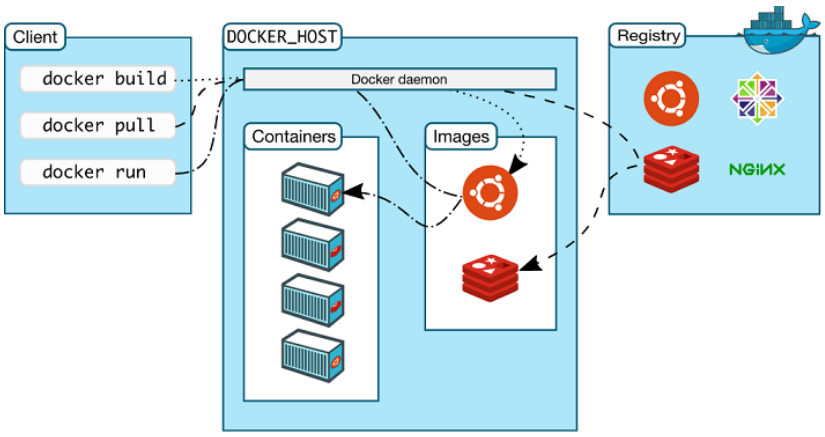
3.2 *Front-end*

O *front-end* já desenvolvido é uma aplicação base feita em *Vue.js*, utilizando recursos visuais do pacote *Bootstrap* e *Typescript*. A aplicação foi projetada para ser simples e prática, atendendo todas as demandas necessárias de cadastros e consultas no banco de dados. A arquitetura é bastante simplista, focando em um modelo multilíngue, permitindo a implementação de novas línguas, se necessário,



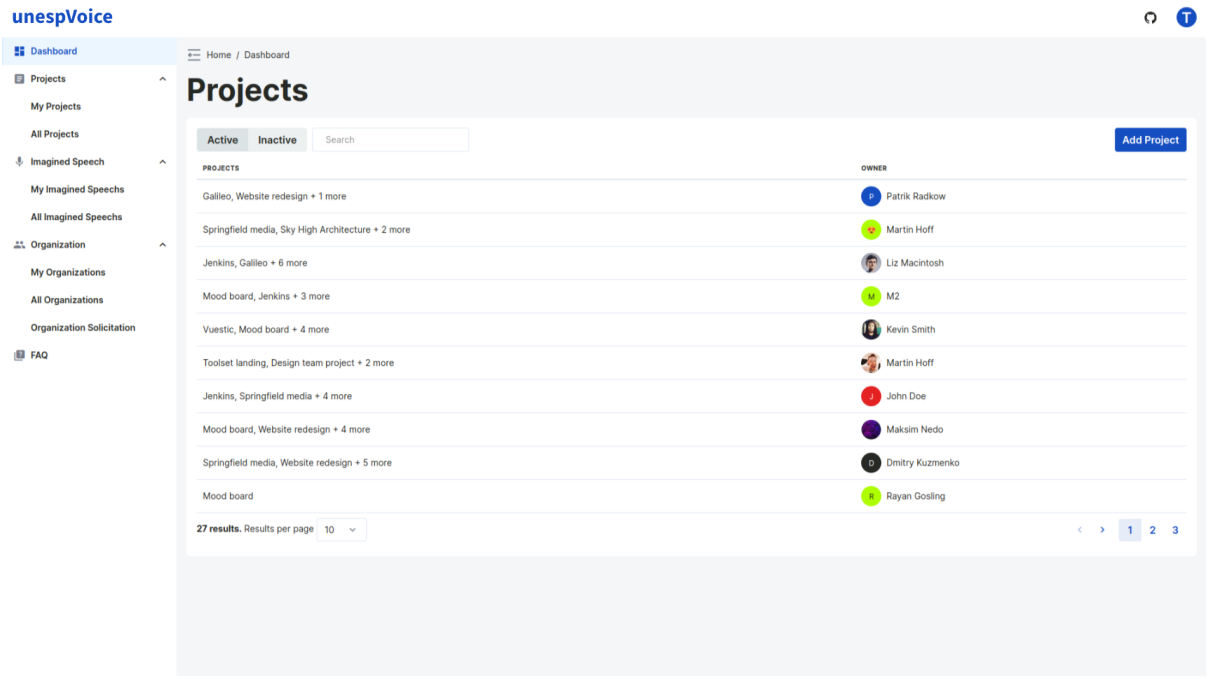
Fonte: Autoria Própria

Figura 3.1: Modelagem Básica da Comunicação do Sistema



Fonte: docker.docs [12]

Figura 3.2: Funcionamento do Docker



Fonte: Autoria Própria

Figura 3.3: Interface da Tela Inicial

conforme é possível visualizar na Figura 3.3.

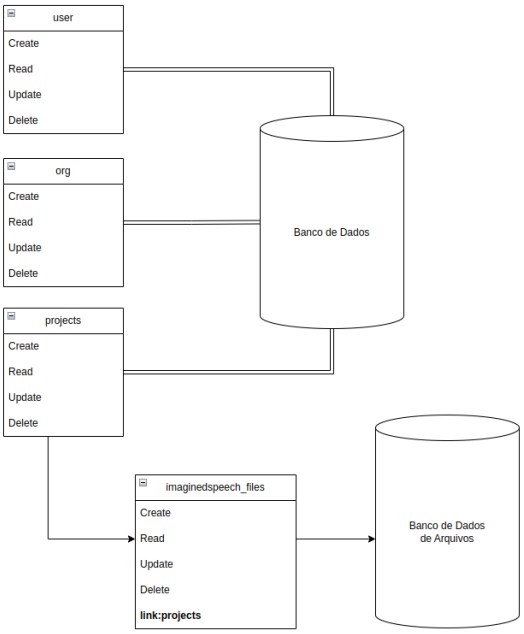
Ela não possui qualquer método de fixação com o *back-end*, comunicando-se apenas através de chamadas de *APIs*. Isso elimina a necessidade de ser o único modelo de *front-end*, mas fornece toda a base para possíveis desenvolvimentos, adições e modificações, caso sejam necessárias. Dessa forma, a aplicação oferece dinamismo e incentiva qualquer organização, pesquisador ou grupo a utilizá-la conforme suas necessidades.

3.3 *Back-end*

O *back-end* foi todo desenvolvido utilizando *Python* com o *framework Django* para desenvolvimento web. Ele tem como base a flexibilidade, funcionando como intermediário entre as requisições e suas regras de organização. Foi moldado de tal forma que não possua uma regra específica, sendo genérico para atender todos os modelos de requisições, sejam modelos únicos ou em lote, tratando tanto cadastros simples vindos do *front-end* quanto uma grande aglomeração de dados para uma aplicação terceira.

O desenvolvimento foi pensado levando em consideração um banco de dados dinâmico e abrangente, possibilitando tanto poucos cadastros por vez quanto grandes lotes de requisições e até mesmo sincronismos com outras bases de outras organizações. Para esse propósito e para as *APIs*, foi utilizado o *framework Django REST*.

A modelagem dos *Endpoints* pode ser visualizada na Figura 3.4. Ela é bem



Fonte: Autoria Própria

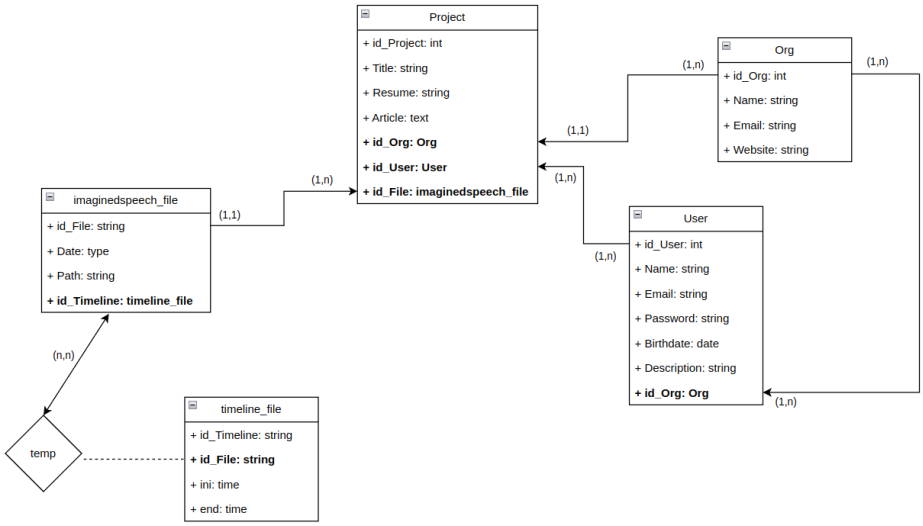
Figura 3.4: *Endpoints* em Comunicação com o Banco de Dados, vinculo do Banco de Arquivos com o *Endpoint* "imaginedspeech_file" e seu vinculo com o *Endpoint* de Projetos

simples, porém moldada de uma forma que essa base sirva como um modelo para futuros desenvolvimentos e adições de novas tabelas, campos e dados de acordo com a necessidade de cada projeto.

3.4 Banco de Dados

Para o banco de dados, a princípio, foi cogitada uma base de dados não relacional, especificamente o *MongoDB*. Embora essa escolha tivesse suas vantagens no processo de desenvolvimento, não seria o melhor cenário, considerando que nem todas as possíveis organizações que poderiam utilizar a aplicação possuam um departamento de tecnologia. Como a ideia é ser inclusivo, optou-se pelo *PostgreSQL*, que solucionará todos os problemas, permitirá as implementações necessárias e manterá cada base de dados modular, conforme necessário para o projeto. A modelagem do sistema base como um banco de dados relacional pode ser visualizada na Figura 3.6.

Para gerenciar essas requisições, está sendo utilizado o *Apache Kafka*, que é uma plataforma de *streaming* distribuída utilizada para construir pipelines de dados em tempo real e aplicativos de *streaming*. Ele permite publicar, subscrever, armazenar e processar fluxos de registros de dados de maneira escalável e resiliente. O *Kafka* criará uma fila de comunicação com o banco de dados a partir das solicitações feitas pelas *APIs*, salvando as posições para que existam uma regra de demanda e redundâncias para casos de perdas de conexões, evitando solicitações duplicadas. Embora



Fonte: Autoria Própria

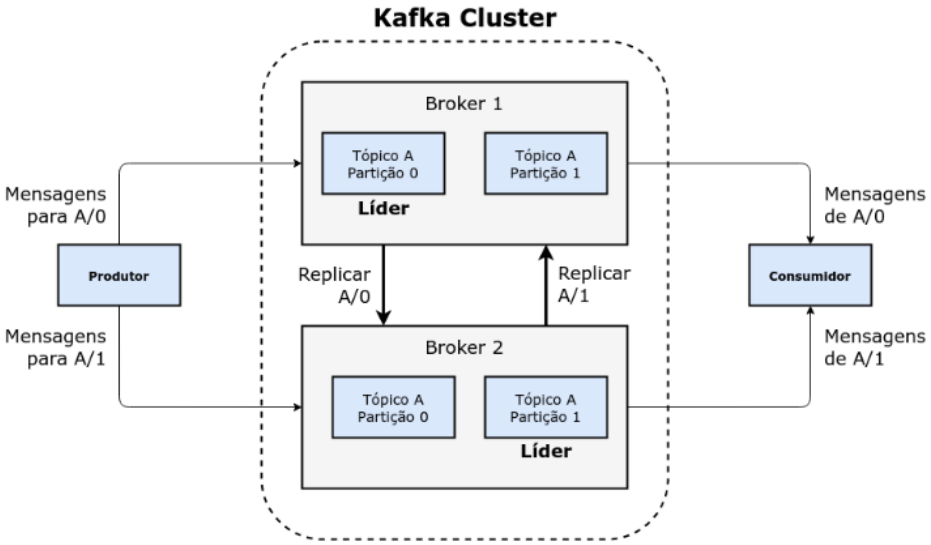
Figura 3.5: Modelagem do Banco de Dados e das Tabelas do Sistema

uma solicitação simples não seja um problema, uma solicitação de uma grande massa de dados poderia gerar uma dupla solicitação, o que seria muito custoso para a base de dados. A fila funcionará conforme visualizado na Figura 3.6.

3.5 Adaptabilidade

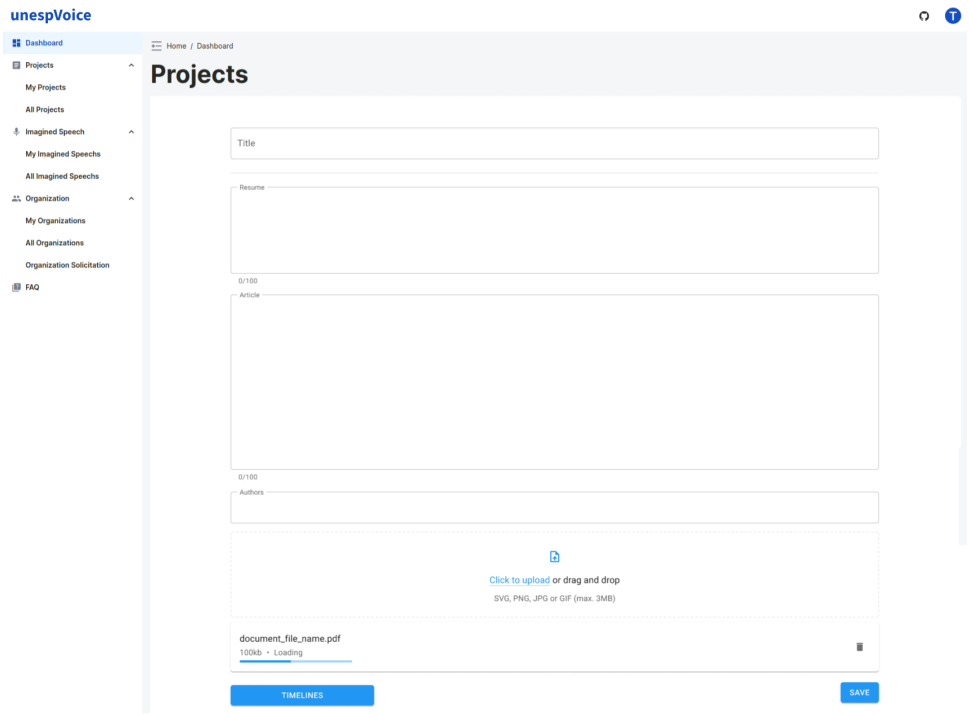
Foi levado em consideração no desenvolvimento o quesito adaptabilidade, buscando alcançar o maior público possível, independentemente do nível de *expertise* em tecnologia. Como mencionado e ilustrado em algumas referências citadas, frequentemente os desenvolvedores de pesquisa não contam com suporte especializado, o que demanda uma linguagem de fácil compreensão e usabilidade acessível. Portanto, o cadastro de um novo artigo ou projeto no sistema foi simplificado ao máximo para facilitar (Figura 3.7 e o cadastro das Linhas do Tempo na Figura 3.8). Isso não apenas simplifica o processo de entrada de dados, mas também incentiva o uso da plataforma por um público diversificado.

O maior fator de adaptabilidade da plataforma é a separação dos seus módulos. O *front-end* possui seu próprio ecossistema, assim como o *back-end* e o banco de dados. É totalmente possível isolar qualquer um desses módulos para implementação em um ambiente geral, a forma como é separado os módulos do sistema por ser visualizado na Figura 3.1. Dessa forma, podem ser criados sistemas de funções autônomas que executam tarefas automaticamente de acordo com a demanda, como a criação de projetos diretamente na base de dados através de aplicações de armazenamento. Implementar soluções que não alterem a rotina atual dos usuários é viável devido ao formato em que o sistema foi concebido. A adaptabilidade do sistema foi projetada com foco na inclusão, permitindo a criação de uma base sólida para compartilhar e incentivar estudos nessa área.



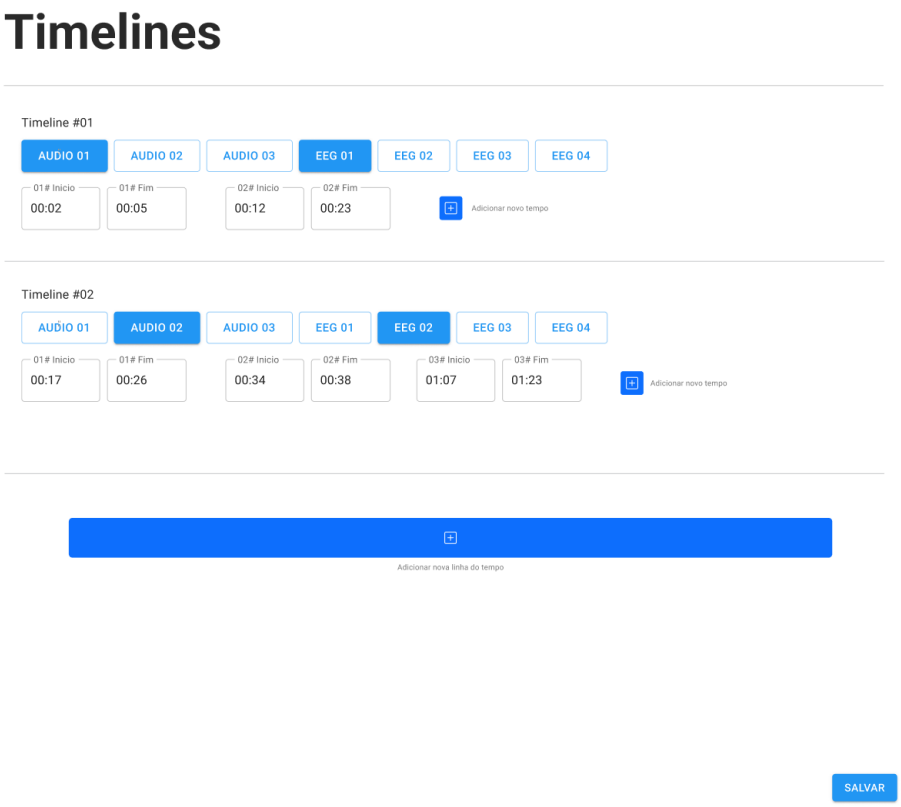
Fonte: *Gitlab* [13]

Figura 3.6: Fluxo de Dados *Streaming* do *Apache Kafka*



Fonte: Autoria Própria

Figura 3.7: Tela de Adição de Novo Projeto



Fonte: Autoria Própria

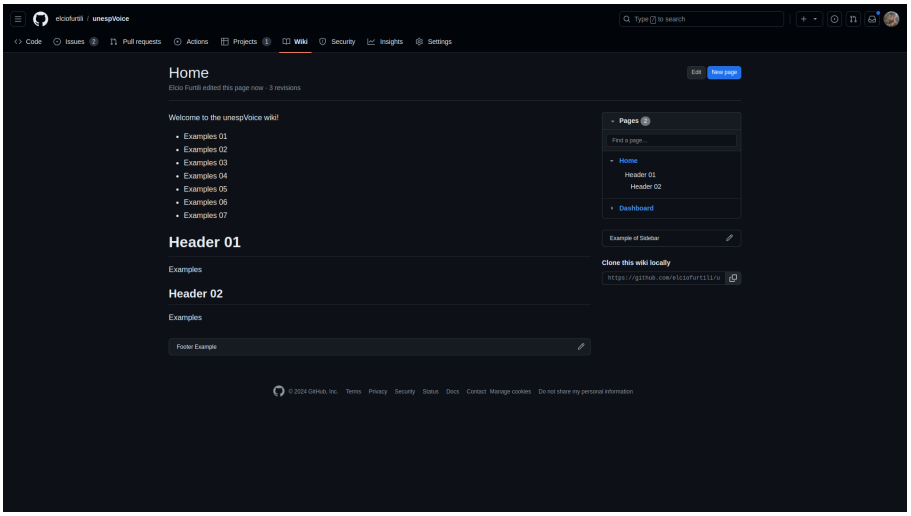
Figura 3.8: Tela de Espaçamento de Tempos ao Clicar em *"Timeline"*

3.6 Desenvolvimento da Comunidade

A plataforma é somente uma ferramenta que foi pensada em um ambiente muito maior, o grande diferencial de todo o processo é criar uma comunidade, facilitar a disseminação de grandes bases de dados para pesquisas que envolvam a fala é o ponto crucial de toda pesquisa, então um fato extremamente importante é justamente desenvolver a comunidade.

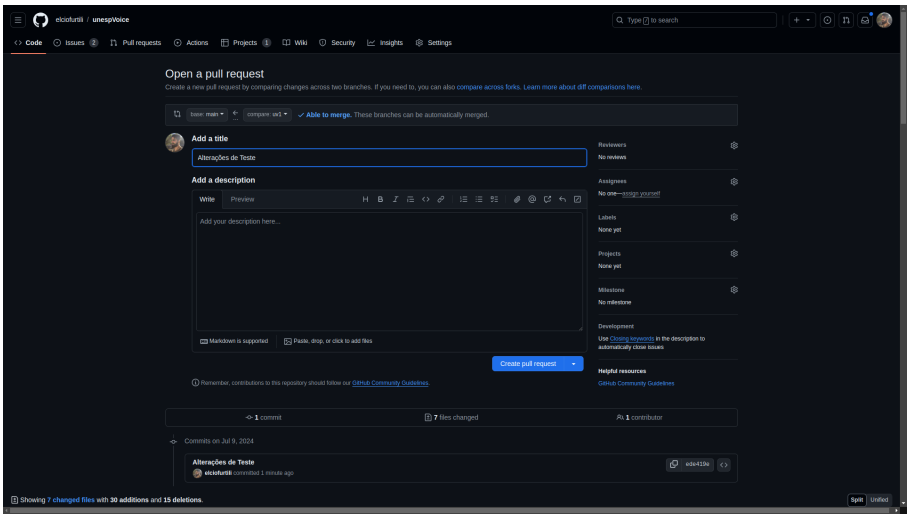
Para desenvolver a comunidade toda a plataforma adota o modelo de código aberto, sendo possível analisar tudo que está sendo programado, corrigido, removido ou incrementado no código fonte da aplicação, dessa forma a comunidade de desenvolvimento terá total acesso a aplicação e poderá replicar, alterar e adaptar de acordo com a necessidades, tudo isso está sendo feito por meio da comunidade *unespVoice* na plataforma *Github*, utilizando recursos de comunidade que a plataforma provê como por exemplo o *Github Wiki*, como se pode visualizar o exemplo na Figura 3.9, que permitirá que seja compartilhada todas as documentações técnicas da plataformas e seus diagramas.

A utilização de *Pull Requests* permitirá a comunicação e o comentário em cada nova alteração de código realizada no sistema (Figura 3.10). Além disso, o controle de *issues* enviado pela comunidade contribuirá para a identificação de possíveis correções ou adições necessárias (Figura 3.11).



Fonte: Autoria Própria

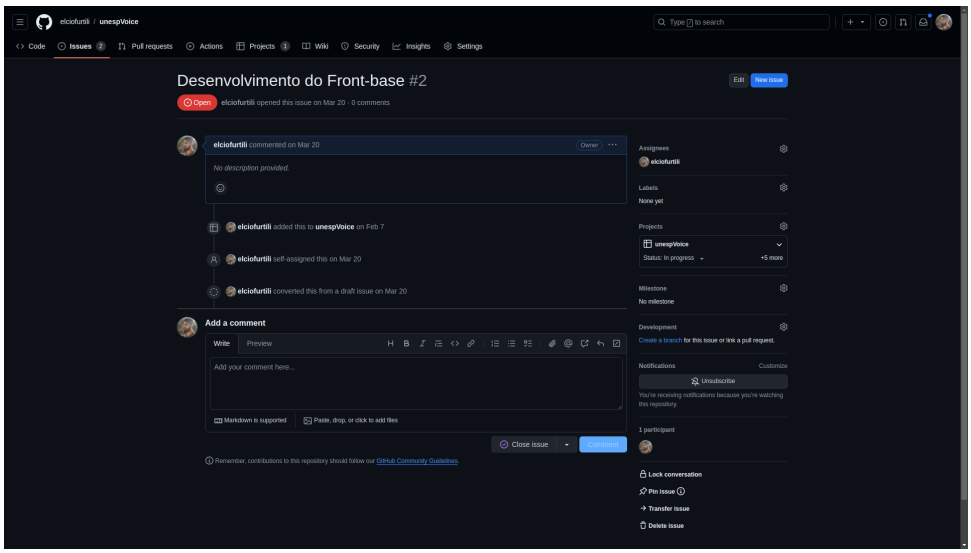
Figura 3.9: Tela de Exemplo *Github Wiki*



Fonte: Autoria Própria

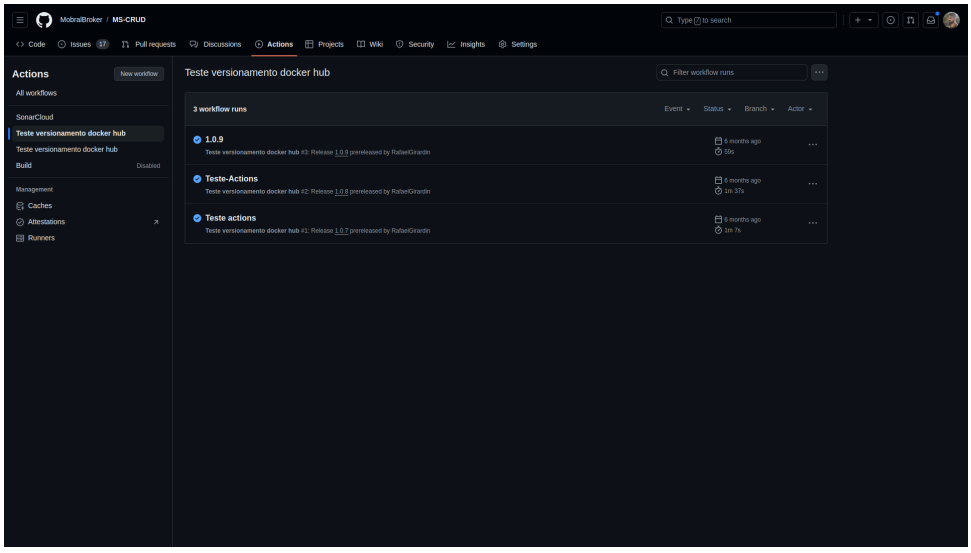
Figura 3.10: Tela de Exemplo *Github Pull Request*

A plataforma *Github* juntamente com a rotina de *Actions* (Figura 3.12) dará total autonomia para que seja adicionado às mais diversas validações, testes de códigos e criação dos *containers* que serão responsáveis por manter o ambiente seguro, estável e funcional para que todos possam e efetuar o download, configurar, testar e começar utilizar seus ambientes para gerenciar seus projetos de fala.



Fonte: Autoria Própria

Figura 3.11: Tela de Exemplo *Github Issues*



Fonte: Autoria Própria

Figura 3.12: Tela de Exemplo *Github Actions*

Capítulo 4

Testes e Resultados

Neste Capítulo, são apresentados os testes realizados e os respectivos resultados obtidos, confirmando a aplicabilidade da estratégia proposta.

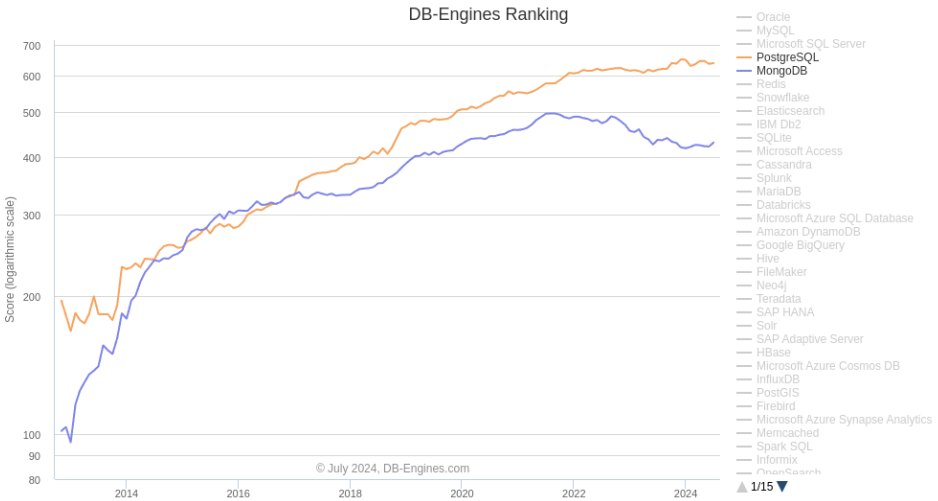
4.1 Tecnologia da Plataforma

A linguagem do *front-end* foi selecionada com base no principal aspecto que o Vue.js nos proporciona: uma programação intuitiva e simples. Além disso, ele oferece uma grande reatividade, ou seja, observa automaticamente as alterações nos dados e atualiza a interface do usuário em resposta a essas mudanças, tornando o desenvolvimento de interfaces dinâmicas e interativas mais intuitivo e eficiente. O *Vue.js* também suporta *Typescript*, facilitando o processo de programação das rotinas e melhorando a comunicação com *APIs*. Além disso, permite uma abordagem simplificada em relação ao *CORS* (*Cross-Origin Resource Sharing*), um mecanismo de segurança que define quais origens externas podem acessar os recursos do servidor.

Python foi escolhido devido à sua flexibilidade que se adequava ao projeto inicial. Em conjunto com *Django* e *Django REST*, *Python* oferecia todos os recursos necessários para criar a aplicação, incluindo *Endpoints*, comunicação com o banco de dados, e integração eficiente com *Kafka*. Em termos gerais, *Python* se mostrou a melhor solução para o desenvolvimento completo da aplicação.

O banco de dados, como mencionado anteriormente, foi escolhido principalmente pela facilidade que um banco relacionado oferece em comparação com tecnologias *NoSQL* ou similares. O *PostgreSQL* também se destacou pelo desempenho de ciclos (Figura 4.1), embora não tenha sido o único critério decisivo na seleção.

O *PostgreSQL* também integra-se bem com o *Kafka*, que gerencia a fila de requisições das *APIs*, garantindo um alto nível de atomicidade nos dados e evitando inconsistências durante o armazenamento.



Fonte: *DB-Engines* [14]

Figura 4.1: Gráfico de Comparação de Desempenho dos Bancos de Dados

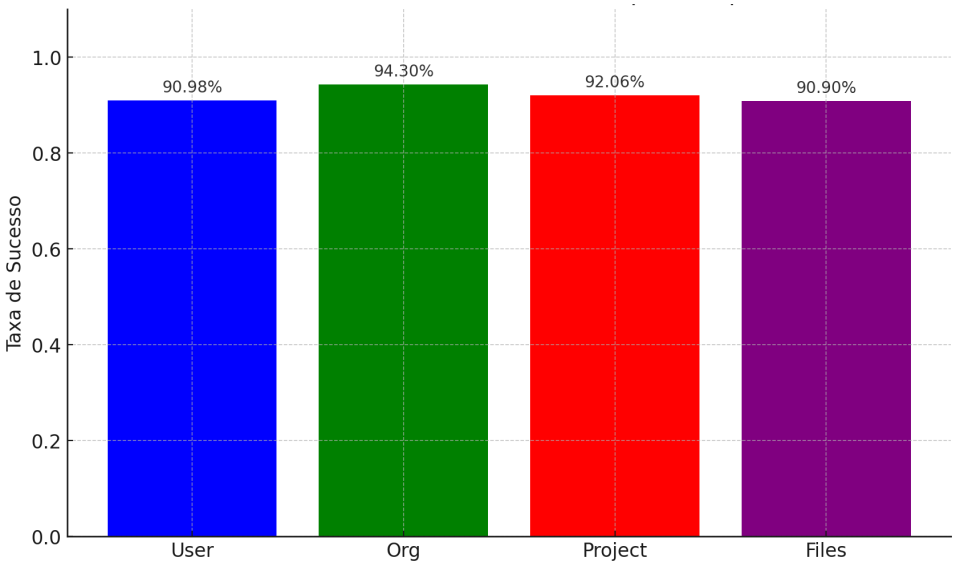
4.2 Performance da Tecnologia

Os testes de tecnologia relacionados ao *Python* concentram-se principalmente nos testes unitários das funções. A escolha pelo *Python* foi motivada pela familiaridade, pela facilidade de escrita de código e pelas ferramentas disponíveis. Os resultados obtidos com a linguagem são baseados em métricas de testes unitários. Como o objetivo era lançar uma versão inicial com todos os recursos funcionando para permitir o uso completo da plataforma, foram realizados testes em cada um dos *Endpoints* para validar a integridade do armazenamento e a precisão na leitura de informações. O gráfico de erros apresenta uma margem satisfatória de sucesso nos testes unitários, conforme mostrado na Figura 4.2.

No que diz respeito ao banco de dados, os testes foram focados no número de ciclos. Foram testados dois modelos: um com filas separadas para cada banco e outro com filas unificadas. Esses testes visavam validar a relação entre memória, fluxo de dados, ciclos e tempo de resposta. Conforme observado na Figura 4.3, o modelo de fila única se mostrou mais eficiente e menos custoso. Os testes foram conduzidos com uma taxa significativa de 10 mil registros por segundo, adequada ao modelo proposto. Assim, devido à sua eficiência superior, o modelo de filas unificadas foi selecionado. É importante ressaltar que as filas mencionadas referem-se às filas gerenciadas no *Kafka* para o tráfego das solicitações.

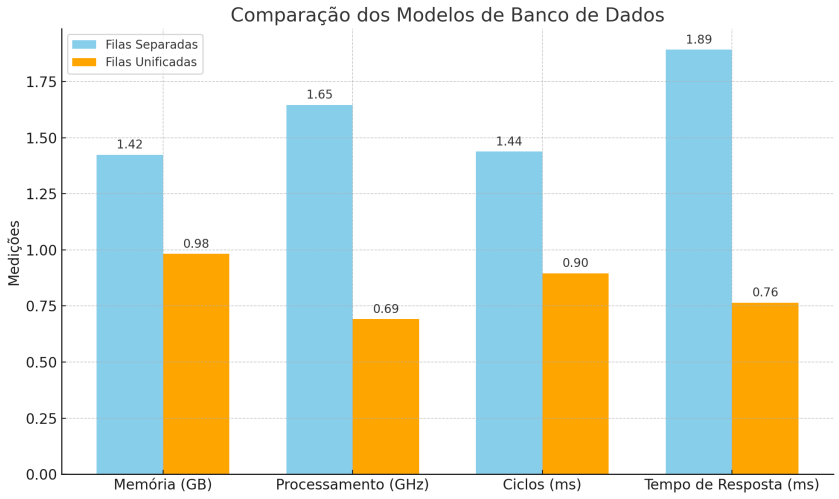
4.3 ACID dos Dados

Todos os testes confirmam que o modelo escolhido é prático e eficiente, retornando resultados satisfatórios que permitem a criação de uma base de dados escalável e resiliente para atender às necessidades do projeto. Para validar todos os aspectos do banco de dados, foram realizados testes ACID, que verificam se uma transação cumpre os quatro princípios fundamentais: Atomicidade, Consistência, Iso-



Fonte: Autoria Própria

Figura 4.2: Teste Unitário: Taxa de Sucesso por *Endpoint*



Fonte: Autoria Própria

Figura 4.3: Teste de 10 Mil Requisições: Consumo de Recursos Computacionais por Modelo de Banco

lamento e Durabilidade.

A Atomicidade assegura que todas as operações de uma transação sejam completadas ou nenhuma delas seja; a Consistência garante que o banco de dados transite de um estado válido para outro; o Isolamento previne que transações concorrentes interfiram umas nas outras; e a Durabilidade assegura que, uma vez confirmada, a transação persista, mesmo em caso de falha do sistema.

Para validar a atomicidade do banco, foram realizados testes para avaliar a taxa de transações completadas em relação às que falharam. Esse método nos permite entender o impacto dos problemas no processo e verificar a consistência mantida. Os resultados obtidos foram dentro dos parâmetros aceitáveis, garantindo a integridade do processo completo, como ilustrado na Figura 4.4, o resultados trouxeram valores de 91,2% das transações concluídas e 8,8% de transações tiveram algum tipo de falha no processo.

Já os testes de consistência dos dados utilizaram o cálculo das taxas de violações de restrições, alcançando valores significativos e demonstrando alta confiabilidade na consistência dos dados, conforme visualizado na Figura 4.5, o resultado sendo uma taxa máxima de 10% de violações no tipo 2 das execuções.

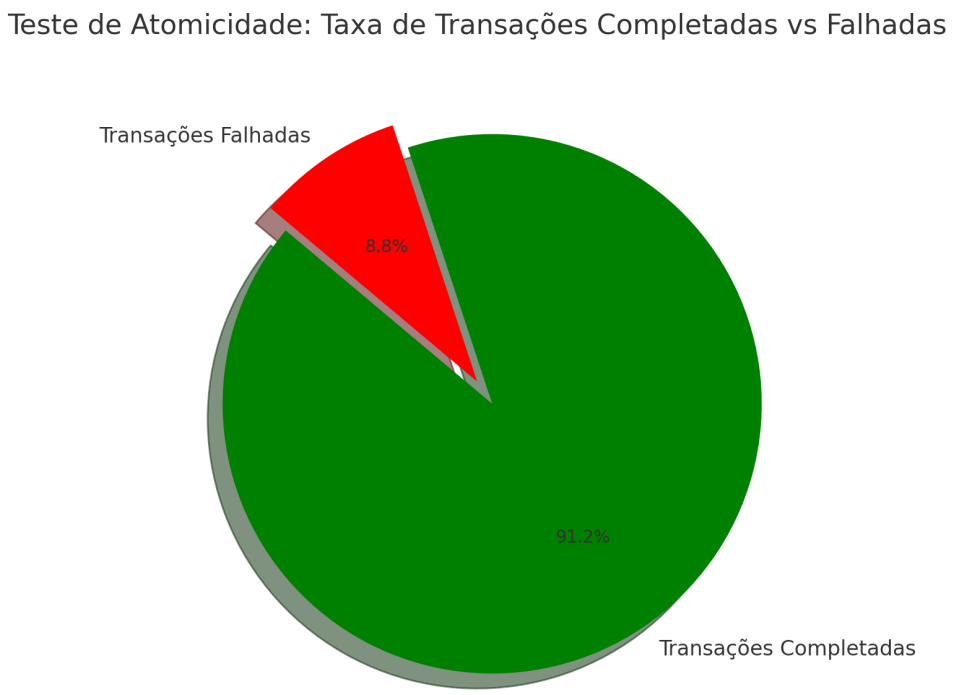
Na analise de conflitos de transação foi utilizado um modelo gráfico de teste de isolamento, verificando a possibilidade de conflitos mesmo com redundâncias do *Kafka*. Os resultados indicaram que o isolamento de dados não representa um ponto problemático, como evidenciado no gráfico na Figura 4.6. demonstrando um total de 3 conflitos de transações.

Os testes de durabilidade foram realizados para validar o tempo de recuperação de transações após falhas, demonstrando que o banco é resiliente e capaz de se auto-recuperar. Os números obtidos garantem que o modelo escolhido não apresenta riscos relacionados à durabilidade, como mostrado na Figura 4.7, o tempo de maior atraso na recuperação dos serviços foi de 12 segundos no cenário 3.

4.4 API

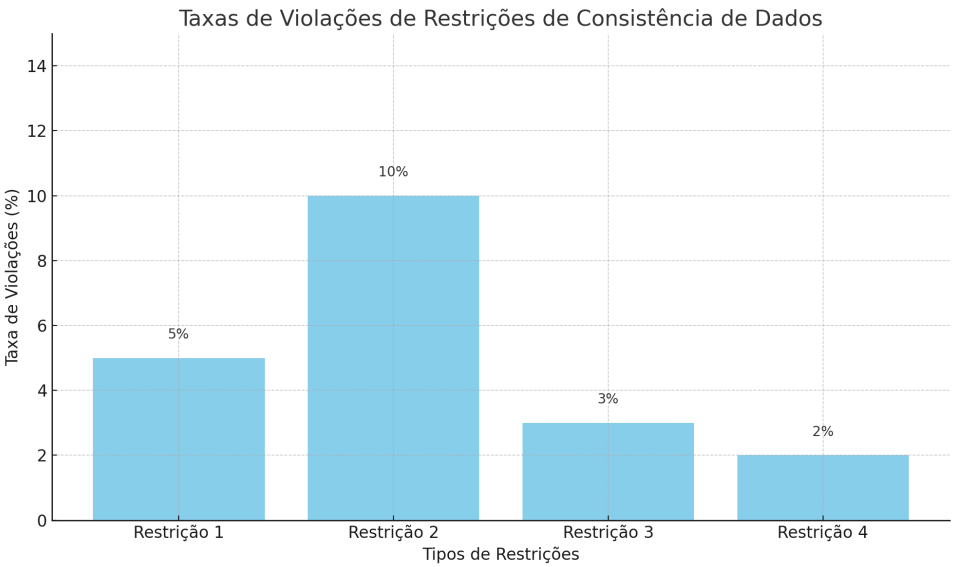
Após a realização dos testes anteriores nos outros tópicos, ficou claro que todos os modelos separados atendem às expectativas estabelecidas pelo projeto. Resta agora realizar testes automatizados de uso para validar se possíveis picos de uso podem ser um problema para a plataforma. Considerando a ideia de permitir o acesso de terceiros ao sistema, testes de estresse são necessários para garantir que o acesso aos dados armazenados não seja um obstáculo.

Todos os testes foram conduzidos com 1 mil requisições por segundo. O primeiro teste realizado foi o de tempo de resposta, que avalia o tempo que a *API* leva para gerar uma resposta. Os valores obtidos foram aceitáveis, com um retorno de menos de 300 milissegundos, mostrando que lidar com cargas elevadas não seria



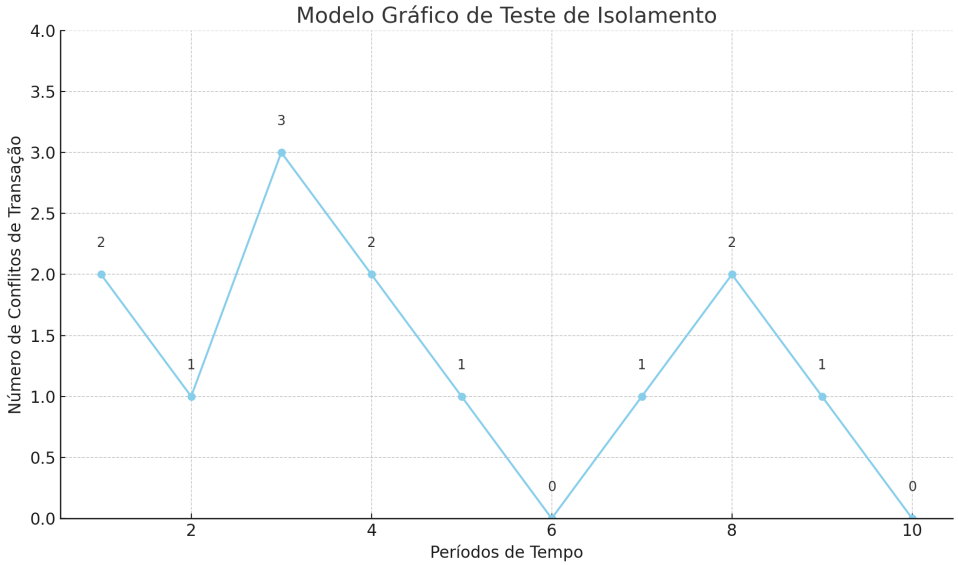
Fonte: Autoria Própria

Figura 4.4: Teste de Atomicidade: Transições Falhadas vs Transições Concluídas



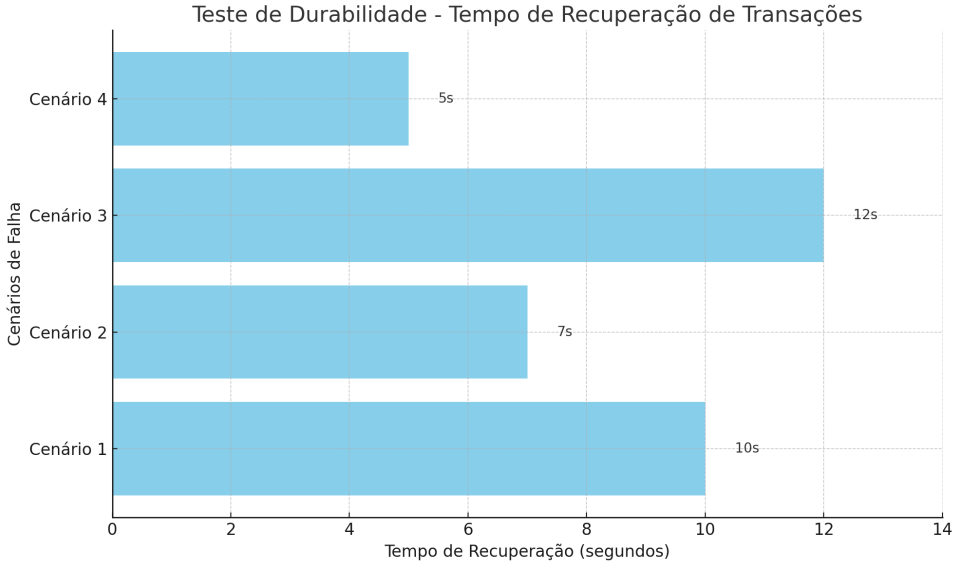
Fonte: Autoria Própria

Figura 4.5: Teste de Consistência: Taxa de Violações por Ciclo



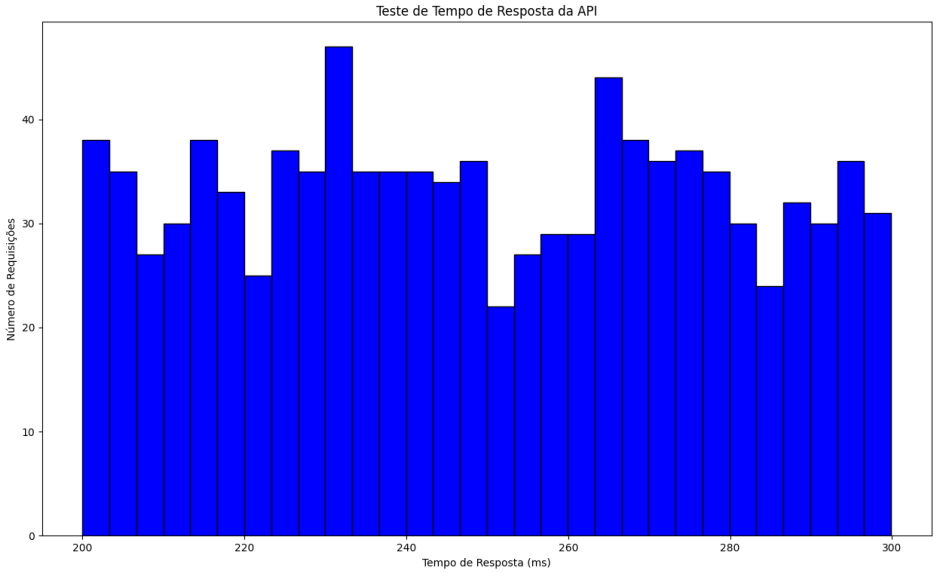
Fonte: Autoria Própria

Figura 4.6: Teste de Isolamento: Conflitos de Transações



Fonte: Autoria Própria

Figura 4.7: Teste de Durabilidade: Cenários de Teste e Tempo de Resolução



Fonte: Autoria Própria

Figura 4.8: Teste de Tempo de Resposta: Requisições em Relação ao Tempo

um problema para o retorno das respostas da *API*, como demonstrado na Figura 4.8, tendo uma taxa de 48 por 300 milissegundos em seu ciclo mais lento.

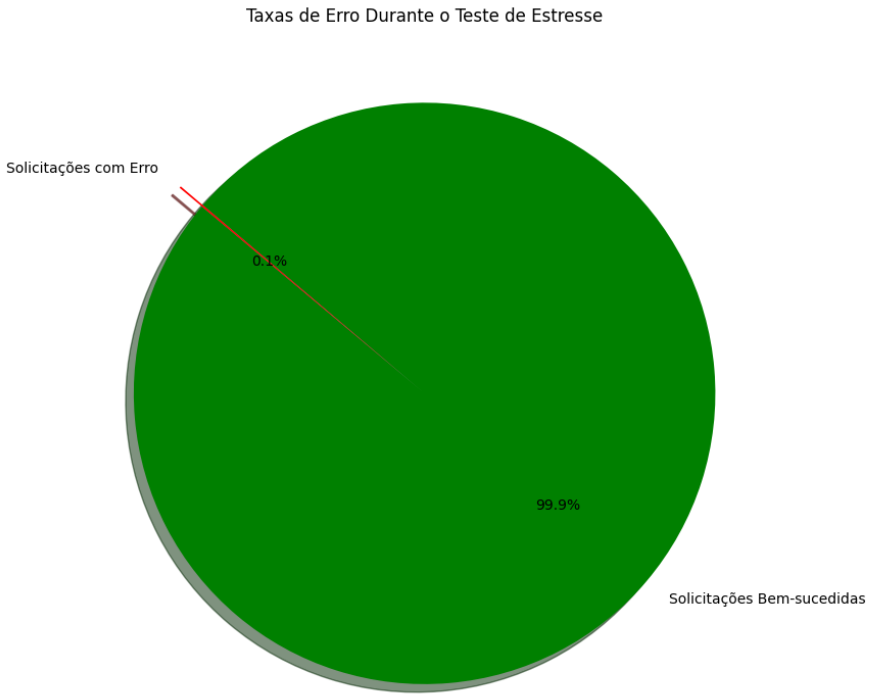
Também foi realizado um teste baseado nas taxas de erros, demonstrando quantas das 1 mil solicitações resultaram em erro para validar como o sistema se comporta nesses casos e quanto tempo leva para se recuperar. Comparando o tempo de resposta com a taxa de erros, a Figura 4.9 mostra que houve um percentual de erros de apenas 0,1% do total durante o teste de estresse.

O teste de vazão revelou um valor médio aceitável, semelhante ao teste de tempo de resposta, com cada 100 requisições levando, em média, 30 milissegundos. O número de solicitações por segundo também foi satisfatório e dentro do esperado, reforçando a confiabilidade das *APIs*, conforme evidenciado na Figura 4.10, seu pior resultado nos ciclos chegou a 38,27 milissegundos, sendo 8,27 milissegundos acima da média.

O último ponto validado nas *APIs* foi o consumo de recursos computacionais da aplicação, utilizando a mesma quantidade de testes anteriores para verificar o impacto no sistema. Foram realizados testes nas especificações conforme a Tabela 4.1.

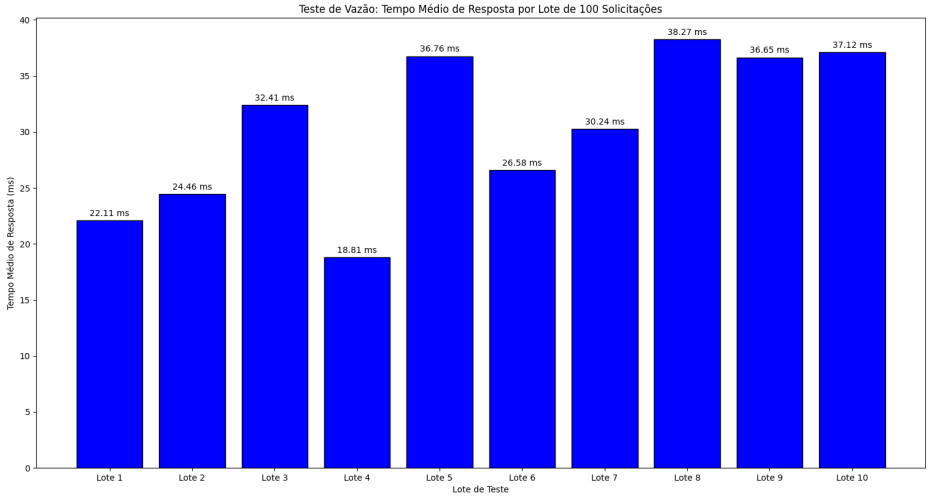
Os valores obtidos foram satisfatórios, utilizando em média 150 *megabytes* de memória e apenas 2% do total do poder de processamento, como pode ser visto no gráfico na Figura 4.11.

Com base em todos os testes realizados, os valores obtidos foram considerados aceitáveis para continuar o desenvolvimento. Foi validado que os modelos e tecnologias selecionados são viáveis para o desenvolvimento completo da aplicação e para a utilização da plataforma.



Fonte: Autoria Própria

Figura 4.9: Teste de Estresse: Validação de Erros em Momentos de Muitas Transações

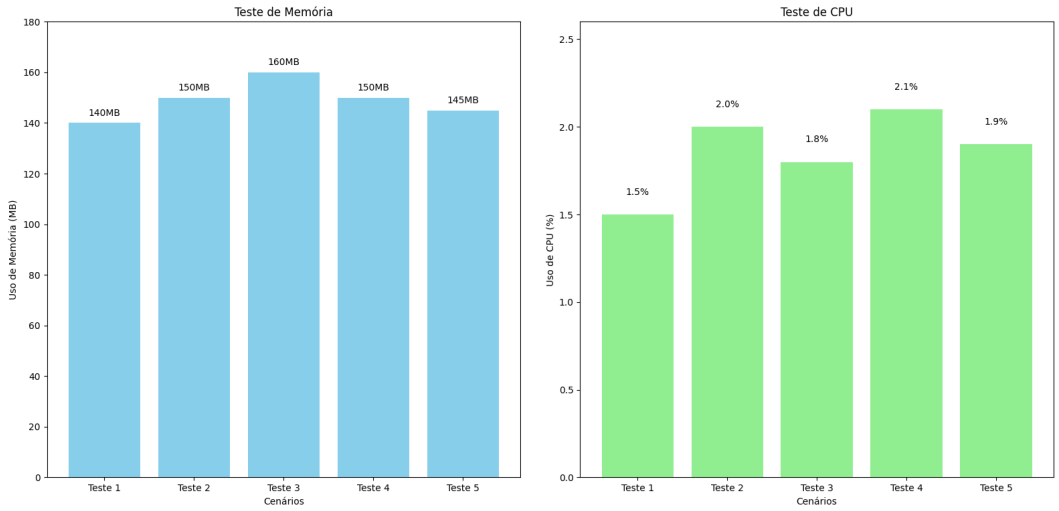


Fonte: Autoria Própria

Figura 4.10: Teste de Vazão: Tempo Médio de Resposta por Lote de 100 Solicitações

Modelo: AMD Ryzen 7 PRO 5875U
Arquitetura: 64x
Frequência: 2.0 GHz - *Turbo Speed*: 4.5 GHz
Núcleos: 8
Memória: 32 Gigabytes
Frequência: 4267 MHz

Tabela 4.1: Especificações do Computador de Desenvolvimento



Fonte: Autoria Própria

Figura 4.11: Teste de Memória e CPU: Recursos Utilizados na Execução da Plataforma

4.5 Retorno da Comunidade

Não foi viável mensurar esses dados por vários motivos. Durante o desenvolvimento da aplicação, não foi considerada uma metodologia prática para medir o retorno da comunidade. O modelo de marketing simplista usado para divulgação resultou apenas em alguns contatos e *feedbacks* superficiais sobre as necessidades potenciais da aplicação, sem números concretos ou testes finais com usuários, devido ao tempo restrito de desenvolvimento.

O projeto foi concebido com base na necessidade de criar uma base sólida para o desenvolvimento de aplicações de fala imaginada. A comunidade que desenvolve essas aplicações é pequena e não muito aberta a colaborações extensas. No entanto, vários membros receberam *e-mails* sobre o funcionamento da plataforma e expressaram questões importantes que foram essenciais para o desenvolvimento da pesquisa. Recebemos um total de apenas 12 retornos anônimos, o que consideramos um resultado satisfatório para uma proposta inicial, embora não tenha atingido as expectativas. Esses retornos foram úteis para dar continuidade ao projeto.

A montagem do questionário foi simples, utilizando a ferramenta de enquetes do *Google*. Foram elaboradas 10 perguntas básicas para entender melhor as preferências e tecnologias mais aceitas pela comunidade. Os resultados estão representados no Apêndice A.

Capítulo 5

Conclusões

O desenvolvimento da plataforma representa o primeiro passo para conscientizar a comunidade sobre a diversidade de áreas que trabalham com a fala imaginada, evidenciando seu potencial significativo para diversas pesquisas em vários campos. Esta área é ampla e requer muita mão de obra e dados.

No que diz respeito à plataforma como um todo, o produto final foi satisfatório, mas ainda requer muitas horas de desenvolvimento contínuo para refinamentos. Apesar disso, o produto mínimo viável já demonstra um potencial computacional relevante, com baixo número de erros, funcionalidades que consomem poucos recursos e um alto retorno em termos de requisições às bases de dados.

Considero este projeto como uma primeira etapa. Inicialmente, surgiu como uma solução para minha pesquisa, na qual não tinha acesso a arquivos para continuar com a pesquisa ou a um eletroencefalograma devido ao alto custo. O desenvolvimento da comunidade tornou-se fundamental para o futuro da plataforma e das pesquisas referente a fala imaginada. Quanto mais pessoas, pesquisadores, organizações e instituições colaborarem e compartilharem informações, mais próxima estará essa pesquisa de alcançar o sucesso, que é criar uma comunidade ativa nesse campo.

O GitHub se mostrou uma ferramenta capaz de atender integralmente aos serviços e necessidades exigidos por um sistema de código aberto e de acesso gratuito. Aliado aos padrões das leis gerais de proteção de dados, o GitHub se mostrou apto a suprir as demandas necessárias para manter um projeto acessível a todos. Para atender a essa demanda, foram realizadas diversas reformulações que adequam o sistema às restrições de proteção de dados de diferentes países. Com isso em mente, o desenvolvimento seguiu um padrão de mínima invasão e incorporou os protocolos de segurança de dados necessários para atender a uma ampla gama de usuários potenciais da ferramenta. Além disso, a natureza modular do sistema permite ajustes específicos para atender às necessidades particulares de projetos distintos. Utilizando o modelo de arquivo ".unpv"criado para o sistema, é possível compartilhar dados mesmo quando o sistema possui características particulares.

A disponibilidade do repositório será viabilizada por meio do link `https://github.com/elciofurtili/unespVoice`. No entanto, o repositório não está

acessível na data deste documento (17 de setembro de 2024) devido às exigências associadas aos selos de software *Open Source* e *Open Code*. Essas exigências podem levar alguns meses para serem plenamente atendidas, e ocasionalmente, os órgãos responsáveis (*OSI*, *GNU* e *FSF*) pela validação dos códigos solicitam alterações adicionais para assegurar o cumprimento dos requisitos estabelecidos para esse tipo de projeto.

Referências

[1] Jonathan R WOLPAW, Niels BIRBAUMER, Dennis J MCFARLAND, Gert PFURTSCHELLER, and Theresa M VAUGHAN. Brain–computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767–791, 2002.

[2] Parul PANDEY and Dario POMPILI. Handling limited resources in mobile computing via closed-loop approximate computations. *IEEE Pervasive Computing*, 18(1):39–48, 2019.

[3] Aneta KIELAR, Lisa MILMAN, Borna BONAKDARTPOUR, and Cynthia K. THOMPSON. Neural correlates of covert and overt production of tense and agreement morphology: Evidence from fmri. *Journal of Neurolinguistics*, 24(2):183–201, 2011. Clinical Speech and Language Studies in Honour of Susan Edwards.

[4] Doug MULHOLLAND, Paulo ALENCAR, and Donald COWAN. ienvironment: Perspectives on metadata-oriented testing of research software. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4687–4690, 2019.

[5] Sadaf IQBAL, P.P. MUHAMMED SHANIR, Yusuf Uzzaman KHAN, and Omar FAROOQ. Eeg analysis of imagined speech. *Int. J. Rough Sets Data Anal.*, 3(2):32–44, apr 2016.

[6] Jerrin Thomas PANACHAKEL, A.G. RAMAKRISHNAN, and T.V. ANANTHAPADMANABHA. Decoding imagined speech using wavelet features and deep neural networks. In *2019 IEEE 16th India Council International Conference (INDICON)*. IEEE, December 2019.

[7] Kenneth C. BARR and Krste ASANOVIC. Energy-aware lossless data compression. *ACM Trans. Comput. Syst.*, 24(3):250–291, aug 2006.

[8] Kazeem B. ADEDEJI. Performance evaluation of data compression algorithms for iot-based smart water network management applications. *Journal of Applied Science Process Engineering*, 7:554–563, 10 2020.

[9] Yehuda VARDI and Simone VERDICCHIO. Data: The fuel of artificial intelligence. *Communications of the ACM*, 65(12):52–61, 2022.

[10] B. BOUSEMAIJ, M. POEL, S. VAN PUTTEN, D. HERMES, J. BOURLAND, and D. TAYLOR. Advanced speech imagination detection techniques for brain-computer interfaces. *IEEE Transactions on Biomedical Engineering*, 66(10):2499–2508, 2019.

[11] Lewis RICHARDSON. *Databases in Microservices: A Practical Approach*. O'Reilly Media, 2016.

[12] DOCKER, Inc. Docker overview, 2024. Acessado em: 17 de setembro de 2024. Disponível em: https://docs.docker.com/guides/docker-overview/.

[13] GITLAB. Kafka quick start, https://ajuda.gitlab.io/guia-rapido/aplicativos/kafka/, 2024. Acessado em: 17 de setembro de 2024. Disponível em: https://ajuda.gitlab.io/guia-rapido/aplicativos/kafka/.

[14] RED GATE SOFTWARE. Db-engines ranking systems encyclopedia, 2024. Acessado em: 17 de setembro de 2024. Disponível em: https://db-engines.com/.

APÊNDICE A

Respostas do Questionário de Avaliação de Necessidade da Plataforma

Neste apêndice, constam, a partir da próxima página, as informações básicas sobre as respostas adquiridas por meio de envio direcionado do questionário para faculdades, pesquisadores e *e-mails* de referência em artigos do gênero.

unespVoice - Platform for Imagined Speech Project Management

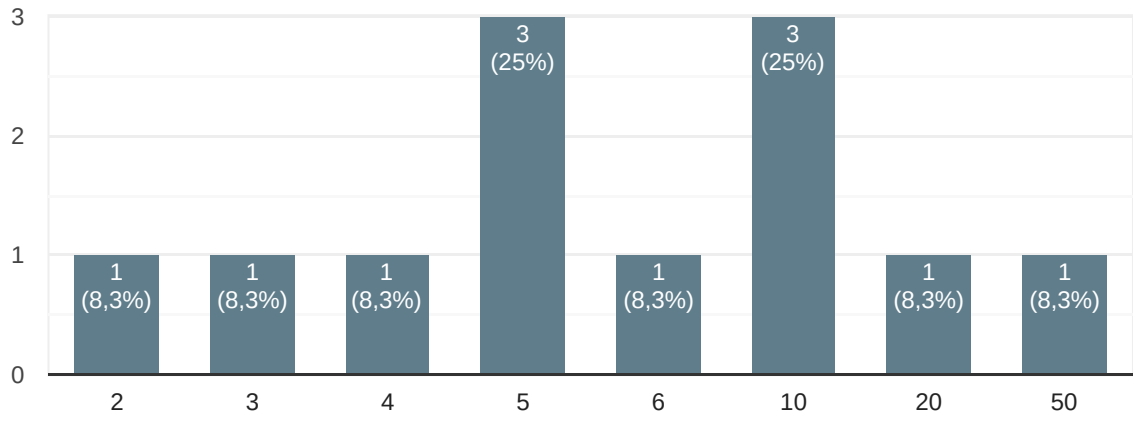
12 respostas

Publicar análise

How many imagined speech files do you generate per project?

Copiar

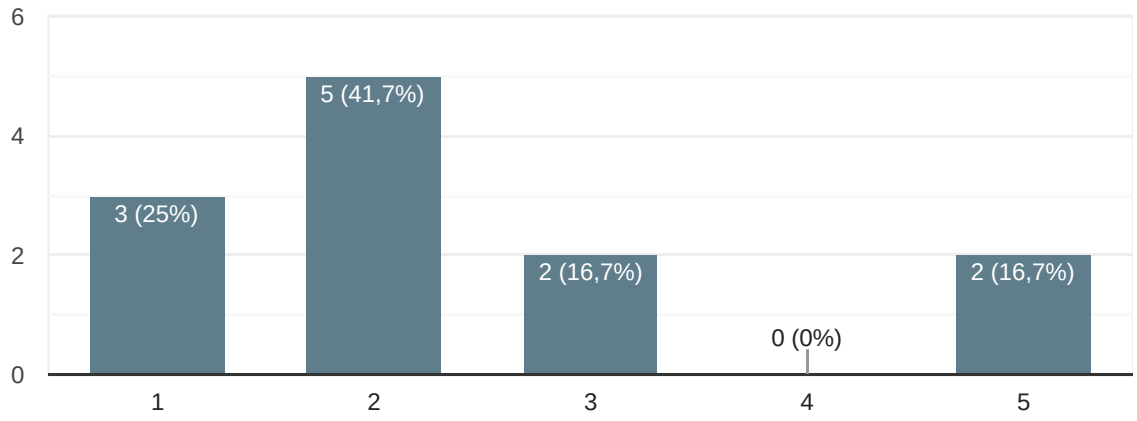
12 respostas



On a scale of 1 to 5, what is the level of difficulty in organizing your files?

Copiar

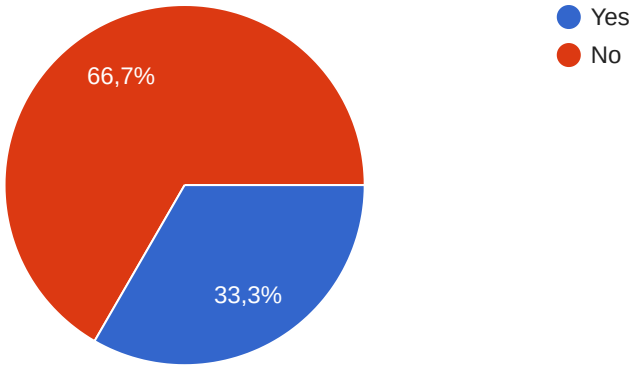
12 respostas



Do you use many EEG devices from different brands?

Copiar

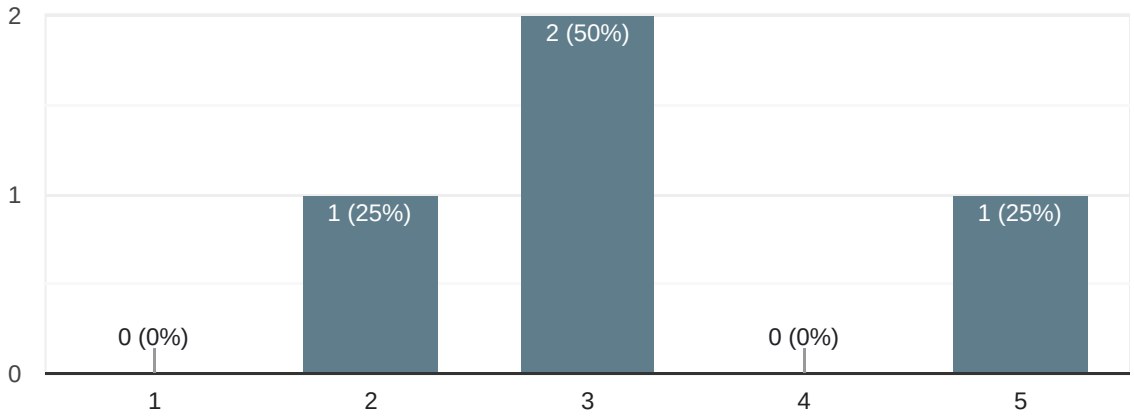
12 respostas



How many different brands do you use?

Copiar

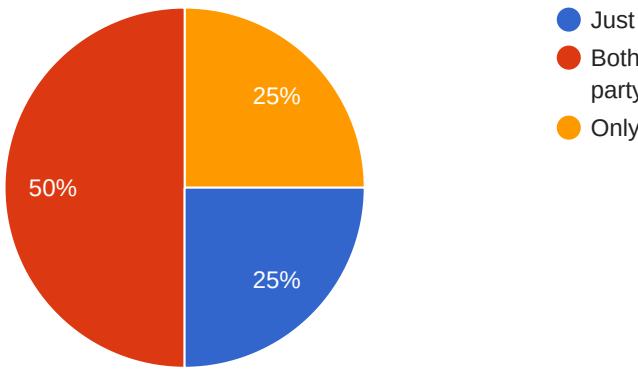
4 respostas



Do you use only your own files or also files from third parties?

Copiar

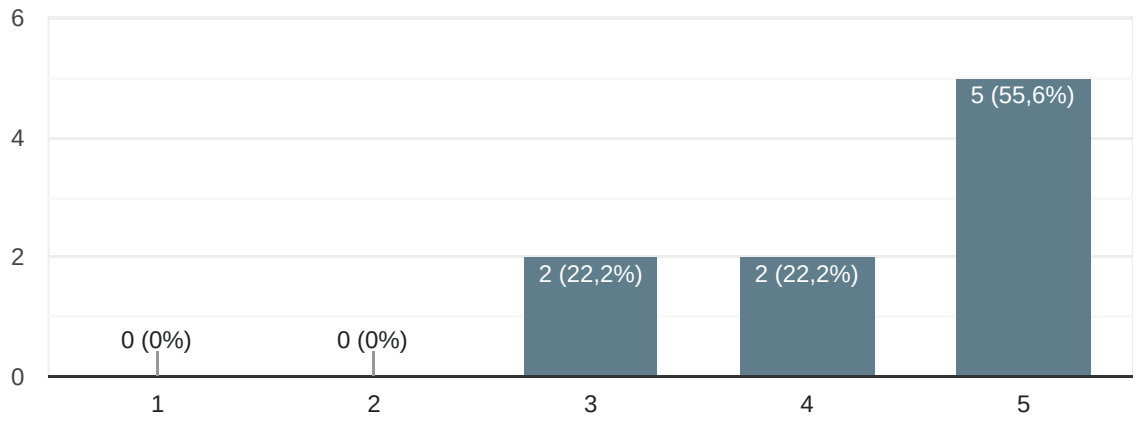
12 respostas



On a scale of 1 to 5, how difficult is it to find imagined speech files for your projects?

Copiar

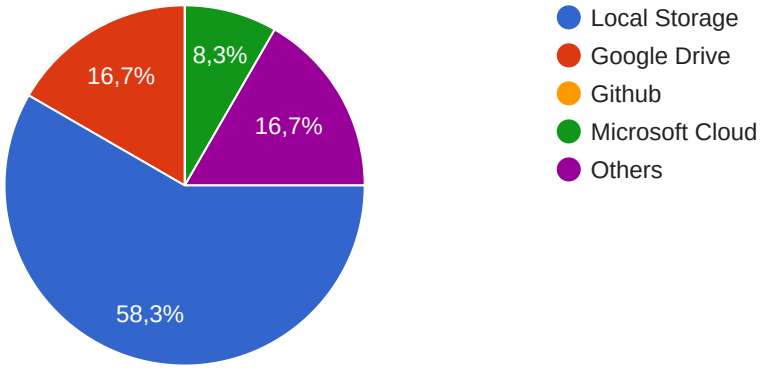
9 respostas



Which platform do you use to store your imagined speech files?

Copiar

12 respostas



Which other platforms do you use?

2 respostas

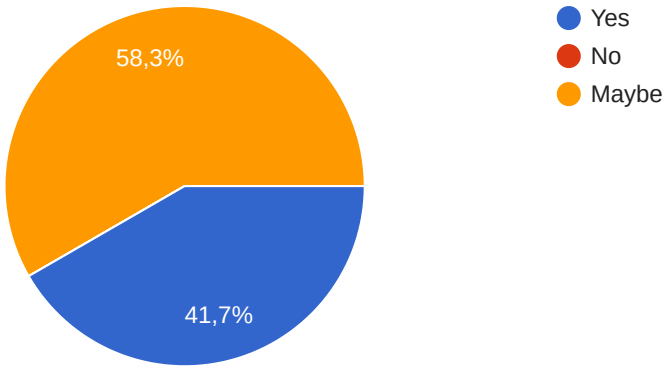
External Hard Disk

University Cloud Storage

Would you use a unified platform where you can generate files and use third-party files for your research?

Copiar

12 respostas



What would make you use the platform in your research?

5 respostas

An easy-to-use platform with the capability to handle multiple projects with many files.

I need a platform that doesn't disrupt the data collection process, as it's only for occasional use and doesn't require something complex in this regard.

I need a platform that allows me to search for models of imagined speech usage.

The challenge of integrating a platform into the project is the difficulty and the lack of availability of researchers to input the information.

A platform that makes it easier to search for new imagined speech files to train my software.