



GOVERNO DO ESTADO DO RIO DE JANEIRO  
SECRETARIA DE ESTADO DE CIÊNCIA E TECNOLOGIA E INOVAÇÃO  
FUNDAÇÃO DE APOIO À ESCOLA TÉCNICA  
FACULDADE DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO  
FAETERJ/PETRÓPOLIS

## *Desenvolvimento da assistente virtual SUNDAY*

**Lucas Rodrigues Barboza**

Petrópolis - RJ

Novembro, 2023

**Lucas Rodrigues Barboza**

***Desenvolvimento da assistente virtual SUNDAY***

Trabalho de Conclusão de Curso apresentado à Coordenadoria do Curso de Tecnólogo em Tecnologia da Informação e da Comunicação da Faculdade de Educação Tecnológica do Estado do Rio de Janeiro Faeterj/Petrópolis, como requisito parcial para obtenção do título de Tecnólogo em Tecnologia da Informação e da Comunicação.

Orientador:  
Douglas Ericson Marcelino

Petrópolis - RJ

Novembro, 2023



**GOVERNO DO ESTADO DO RIO DE JANEIRO  
SECRETARIA DE ESTADO DE CIÊNCIA E TECNOLOGIA  
FUNDAÇÃO DE APOIO À ESCOLA TÉCNICA  
CENTRO DE EDUCAÇÃO PROFISSIONAL EM TECNOLOGIA DA INFORMAÇÃO  
FACULDADE DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO**

**FAETERJ/PETRÓPOLIS**

Desenvolvimento da assistente virtual Sunday

LUCAS RODRIGUES BARBOZA

Trabalho apresentado no curso de Formação em  
Tecnologia da Informação e Comunicação da  
FAETERJ-Petrópolis como requisito parcial para  
obtenção do grau de tecnólogo.

Aprovado em 29 /11/2023

Banca Examinadora

Prof. D.Sc. Douglas Ericson Marcelino de Oliveira - FAETERJ-Petrópolis

Prof. D.Sc. Alberto Torres Angonese - FAETERJ-Petrópolis

Prof. Dsc. Victor de Almeida Thomaz - UNIFESO

## *Declaração de Autor*

Declaro, para fins de pesquisa acadêmica, didática e tecnico-científica, que o presente Trabalho de Conclusão de Curso pode ser parcial ou totalmente utilizado desde que se faça referência à fonte e aos autores.

O código fonte pode ser acessado no links

[https://github.com/LucasBarboza-maker/Sunday\\_TCC](https://github.com/LucasBarboza-maker/Sunday_TCC)

---

Lucas Rodrigues Barboza  
Petrópolis, em 20 de Outubro de 2023

## ***Dedicatória***

Gostaria de expressar meus sinceros agradecimentos a todas as pessoas e instituições que contribuíram para a conclusão deste Trabalho de Conclusão de Curso. Primeiramente, desejo agradecer a minha mãe, Marcia Cristina Alvez Rodrigues Barboza, pelo apoio em momentos de dificuldade, agradecer meu orientador Douglas Ericson Marcelino pela orientação, paciência e apoio contínuo ao longo deste processo. Também gostaria de agradecer aos meus colegas de classe que forneceram apoio mútuo e compartilharam ideias durante nossas discussões acadêmicas. Além disso, quero expressar. Agradeço à FAETERJ pela oportunidade de realizar este estudo e pelos recursos disponibilizados. Por fim, gostaria de agradecer aos participantes do meu estudo, cujas contribuições foram essenciais para a coleta de dados. Este TCC não teria sido possível sem o apoio e a colaboração de todos vocês. Obrigado por tornar esta jornada acadêmica significativa e gratificante.

## ***Agradecimentos***

Primeiramente a Deus que me permitiu chegar onde estou.

Gostaria de expressar meus sinceros agradecimentos a todas as pessoas e instituições que contribuíram para a conclusão deste Trabalho de Conclusão de Curso. Primeiramente, de- sejo agradecer a minha mãe, Marcia Cristina Alvez Rodrigues Barboza, pelo apoio em momentos de dificuldade, agradecer meu orientador Douglas Ericson Marcelino pela orientação, paciência e apoio contínuo ao longo deste processo. Também gostaria de agradecer aos meus colegas de classe que forneceram apoio mútuo e compartilharam ideias durante nossas discussões acadêmicas.

Agradeço à FAETERJ pela oportunidade de realizar este estudo e pelos recursos disponibilizados. Por fim, gostaria de agradecer aos participantes do meu estudo, cujas contribuições foram essenciais para a coleta de dados. Este TCC não teria sido possível sem o apoio e a colaboração de todos vocês. Obrigado por tornar esta jornada acadêmica significativa e gratificante.

# *Epígrafe*

"Simplicity is the ultimate sophistication"

---

(Leonardo da Vinci (1452-1519))

## ***Resumo***

Nos dias atuais, flexibilidade de horário e organização são pilares do sucesso de cada pessoa. Por isso, uma boa gestão do tempo é importante para maior produtividade diária. Uma gestão mal feita pode gerar muitos problemas de organização, causando muito estresse.

O objetivo deste trabalho é mostrar como pode ser criado um assistente virtual com JAVASCRIPT que utiliza processamento de linguagem natural para ter uma interação de fala com o usuário, armazenar eventos na agenda e entretê-lo com um jogo. Este assistente será nomeado de Sunday, e sua implementação será feita em um sistema de login de usuário, em um sistema com uma página web desenvolvida em React e um servidor HTTP desenvolvido em Node. A combinação destas duas bases para o desenvolvimento se dá pelo motivo de que estas tecnologias são muito recentes e interagem muito bem entre si, tornando-as menos verbosas. Além disso, utilizará processamento de linguagem natural, voltado para reconhecer e/ou gerar texto em linguagem humana, para uma interação mais natural entre homem e máquina.

**Palavras-chave:** Backend. Frontend. Assistente. Virtual. Sunday. React. Node. Javascript.

# *Abstract*

In today's world, schedule flexibility and organization are pillars of success for each individual. Therefore, effective time management is crucial for increased daily productivity. Poor management can lead to numerous organizational problems, causing significant stress.

The goal of this project is to demonstrate how a virtual assistant can be created using JAVASCRIPT, employing natural language processing for speech interaction with the user, storing events in the agenda, and engaging them with a game. This assistant will be named Sunday, and its implementation will be carried out in a user login system, within a system featuring a React-developed web page and an HTTP server developed in Node. The combination of these two platforms for development is driven by their recent nature and excellent synergy, resulting in less verbosity. Additionally, it will utilize natural language processing, designed to recognize and/or generate text in human language, for a more natural interaction between humans and machines.

**Palavras-chave:** Backend. Frontend. Virtual Assistant. Sunday. React. Node. Javascript.

# ***Listas de Figuras***

2.1	Hello World em Node (Fonte: Autor).	p. 22
2.2	Hello world retornado por um servidor <i>Hypertext Transference Protocol</i> (HTTP) usando express (Fonte: Autor).	p. 23
2.3	Criando um componente básico com REACT (Fonte: Autor).	p. 24
2.4	Ilustrando um componente básico com REACT (Fonte: Autor).	p. 25
3.1	Google Assistant <i>ecosystem</i> (fonte: store.google.com)	p. 29
3.2	Siri Fluxo (Fonte: machinelearning.apple.com)	p. 31
4.1	Diagrama de casos de uso	p. 34
4.2	Sunday Fluxo (Fonte: Autor)	p. 35
4.3	Diagrama de Classes (Fonte: Autor)	p. 38
4.4	Sunday Diagrama de Classes (Fonte: Autor)	p. 39
5.1	Tela de <i>Login</i> (fonte: Autor)	p. 44
5.2	Tela de cadastro (fonte: Autor)	p. 45
5.3	Tela da Agenda com a Sunday(fonte: Autor)	p. 46
5.4	Tela de Eventos (fonte: Autor)	p. 47
5.5	Tela do Jogo da Velha (fonte: Autor)	p. 48
5.6	Como aparecem os aviso (fonte: Autor)	p. 48
5.7	Exemplo Modal (fonte: Autor)	p. 49

## ***Listas de Tabelas***

3.1	Comparação de Tecnologias Utilizadas. . . . .	p. 32
3.2	Comparação de funcionalidades. . . . .	p. 32
4.1	Lista de Rotas HTTP . . . . .	p. 37
4.2	Tabela de requisitos funcionais . . . . .	p. 40
4.3	Tabela de requisitos não-funcionais . . . . .	p. 40
4.4	Tabela de regras de negócio . . . . .	p. 41
4.5	Lista de Rotas HTTP . . . . .	p. 42
4.6	Lista de Rotas HTTP . . . . .	p. 43

# ***Lista de abreviaturas***

**PLN** Processamento de Linguagem Natural

**HTTP** *Hypertext Transference Protocol*

**HTML** *hypertext markup language*

**XML** *Extensible Markup Language*

**CSS** *Cascading Style Sheets*

**SQL** *Structured Query Language*

**IOT** *Internet of Things*

**REST** *Representational State Transfer*

**MIT** *Massachusetts Institute of Technology*

**UML** *Unified Modeling Language*

**API** *Application Programming Interface*

**JSON** *JavaScript Object Notation*

**FK** *Foreign Key*

**TCC** Trabalho de Conclusão de Curso

**IBM** *International Business Machines*

**SaaS** *Software as a Service*

**DTO** Data Transfer Object

**NoSQL** Not Only SQL

**REGEX** *Regular Expressions*

# *Sumário*

<b>1</b>	<b>Introdução</b>	p. 16
1.1	Contexto . . . . .	p. 16
1.2	Justificativa . . . . .	p. 16
1.3	Objetivos . . . . .	p. 16
1.4	Contribuições do Trabalho . . . . .	p. 16
1.5	Problema e Necessidade . . . . .	p. 17
1.5.1	Problema de Gestão de Tempo e Tarefas . . . . .	p. 17
1.5.2	Necessidade de Entretenimento e Relaxamento . . . . .	p. 17
1.5.3	Comunicação Eficiente . . . . .	p. 17
1.5.4	Eventos e Calendário . . . . .	p. 17
1.5.5	Comunicação Eficiente e Resolução de Problemas . . . . .	p. 17
1.6	História das Assistentes Virtuais . . . . .	p. 18
1.7	Metodologia . . . . .	p. 18
1.8	Estrutura do Trabalho . . . . .	p. 19
<b>2</b>	<b>Fundamentação teórica</b>	p. 20
2.1	Servidor . . . . .	p. 20
2.1.1	Protocolo de Transferência de Hipertexto . . . . .	p. 20
2.1.2	Aplicação web . . . . .	p. 20
2.1.3	Arquitetura REST . . . . .	p. 21
2.2	Javascript . . . . .	p. 21
2.3	NodeJS . . . . .	p. 21

2.4	Framework . . . . .	p. 22
2.4.1	Express . . . . .	p. 22
2.4.2	Processamento de Linguagem Natural . . . . .	p. 23
2.5	Frontend . . . . .	p. 23
2.5.1	React . . . . .	p. 24
2.5.2	Captação da Voz . . . . .	p. 25
2.5.3	NLP.js . . . . .	p. 25
2.5.4	Regex . . . . .	p. 25
2.5.5	Cascading Style Sheets . . . . .	p. 25
2.5.6	Expressão Utilizada . . . . .	p. 26
2.6	Armazenamento de Dados . . . . .	p. 26
2.6.1	Banco de dados . . . . .	p. 26
2.6.2	Modelo Relacional . . . . .	p. 26
2.6.3	Modelo Não Apenas Relacional . . . . .	p. 26
2.6.4	Banco de Dados Orientados a Documentos . . . . .	p. 27
2.7	Docker . . . . .	p. 27
<b>3</b>	<b>Revisão bibliográfica</b>	<b>p. 29</b>
3.1	Google Assistant . . . . .	p. 29
3.1.1	Arquitetura do Google Assistant . . . . .	p. 29
3.1.2	Como Funciona o Google Assistant . . . . .	p. 30
3.2	Siri . . . . .	p. 30
3.2.1	Arquitetura da Siri . . . . .	p. 31
3.2.2	Como Funciona a Siri . . . . .	p. 31
3.3	Comparação entre as Assistentes . . . . .	p. 32
3.3.1	Treinamento Manual . . . . .	p. 32

<b>4 Implementação</b>	p. 33
4.1 Arquitetura do Sistema . . . . .	p. 33
4.1.1 Diagrama de Casos de Uso . . . . .	p. 33
4.1.2 Detalhando diagrama de casos de uso . . . . .	p. 34
4.1.3 Diagrama de Fluxo de Dados . . . . .	p. 34
4.1.4 Detalhando o Fluxo . . . . .	p. 35
4.1.5 Processamento de Dados . . . . .	p. 35
4.1.6 Processamento de Mensagens . . . . .	p. 36
4.1.7 Processamento do Jogo . . . . .	p. 36
4.1.8 Processamento de Datas . . . . .	p. 36
4.1.9 Diagrama de Classes . . . . .	p. 37
4.1.10 Detalhando Diagrama de Classes . . . . .	p. 38
4.1.11 Mapeamento Objeto-Relacional . . . . .	p. 39
4.2 Levantamento de Requisitos . . . . .	p. 39
4.2.1 Requisitos Funcionais . . . . .	p. 39
4.2.2 Requisitos Não Funcionais . . . . .	p. 40
4.2.3 Regras de Negócio . . . . .	p. 40
4.3 Dicionário de dados . . . . .	p. 41
4.3.1 Tabela User . . . . .	p. 42
4.3.2 Tabela Schedule . . . . .	p. 42
<b>5 Telas e resultados</b>	p. 44
5.1 Tela de Login . . . . .	p. 44
5.2 Tela de Registro . . . . .	p. 45
5.3 Tela de Conversa . . . . .	p. 46
5.4 Agenda com a Sunday . . . . .	p. 46
5.5 Jogando com a Sunday . . . . .	p. 47

5.6	Intuitividade da Sunday . . . . .	p. 48
5.6.1	Modais do Sistema . . . . .	p. 49
<b>6</b>	<b>Conclusão e trabalhos futuros</b>	<b>p. 50</b>
	<b>Referências</b>	<b>p. 52</b>

# ***1     Introdução***

## **1.1   Contexto**

Assistentes virtuais são softwares que interpretam a interação humana projetados para ajudar os usuários em tarefas, como responder a perguntas, executar comandos, fornecer informações ou realizar funções automatizadas, agendar Eventos. Eles atuam como tradutores entre os humanos e os computadores, mudando o meio de como interagirmos com as máquinas, tornando as interações mais intuitivas e eficazes.(KENTON, 2023)

## **1.2   Justificativa**

A realização deste trabalho está na importância de criar uma Assistente Virtual prática e funcional para interagir de forma intuitiva com os usuários, agendar eventos e entreter o usuário com um jogo da velha também com a possibilidade de interagir com o usuário por voz.

## **1.3   Objetivos**

O objetivo geral Trabalho de Conclusão de Curso (TCC) é desenvolver uma assistente virtual multifuncional que combine um sistema de chat, os objetivos adicionais são:

- Criar uma agenda personalizada que se adapte às preferências individuais dos usuários.
- Desenvolver um jogo da velha com a Assistente Virtual.

## **1.4   Contribuições do Trabalho**

Este TCC busca contribuir para o campo de Assistente Virtual, fornecendo uma implementação prática de uma Assistente Virtual multifuncional. Além disso, espera-se que este trabalho

demonstre como a Assistente Virtual pode ser aplicada em diversas áreas, desde assistência pessoal até entretenimento.

## 1.5 Problema e Necessidade

### 1.5.1 Problema de Gestão de Tempo e Tarefas

A vida moderna é frequentemente caracterizada pela agitação e pelas múltiplas responsabilidades. Muitas pessoas enfrentam desafios na gestão eficaz do tempo e na organização de tarefas. A falta de um sistema eficiente pode levar ao esquecimento de compromissos importantes e à sobrecarga de informações.

### 1.5.2 Necessidade de Entretenimento e Relaxamento

É bem mais divertido interagir por voz com a **Sunday** e jogar um jogo no qual o usuário não precisa usar o teclado nem o mouse é um diferencial a se considerar.

### 1.5.3 Comunicação Eficiente

A comunicação eficiente é essencial em diversas esferas da vida, desde o ambiente de trabalho até a interação com sistemas automatizados.

### 1.5.4 Eventos e Calendário

O sistema de agenda da **Sunday** possibilita aos usuários uma maneira de organizar eventos e gerenciar os mesmos. Ele visa tornar a gestão de tempo mais eficaz e ajudar o usuário a se organizar.

### 1.5.5 Comunicação Eficiente e Resolução de Problemas

O sistema de chat da Assistente Virtual é projetado para fornecer comunicação eficiente com os usuários. No decorrer deste trabalho, exploraremos em detalhes como a implementação da **Sunday** como pode contribuir para a simplificação do processo, sendo desenvolvida em apenas uma linguagem.

## 1.6 História das Assistentes Virtuais

A primeira aplicação de Processamento de Linguagem Natural (PLN) ou *chatbot*, chamada ELIZA, foi inventada pelo *Massachusetts Institute of Technology* (MIT) na década de 1960. ELIZA foi projetada para simular conversas aplicando correspondência de configuração e procedimentos de substituição em suas respostas escritas, criando a impressão de que a máquina entendia o que estava sendo dito.

Em 1986, foi lançada uma melhoria para o Shoebox, chamada Tangora, uma máquina de datilografia por reconhecimento de fala. Com um vocabulário de 20.000 palavras, era capaz de antecipar o resultado mais provável com base em suas informações. Foi apelidada de "Máquina de Datilografia Mais Rápida". Para isso, a *International Business Machines* (IBM) utilizou um modelo oculto de Markov, que incorpora estatísticas na previsão de quais fonemas seguiriam um fonema dado.

Esses desenvolvimentos marcaram os primeiros passos na evolução das tecnologias de reconhecimento de fala e processamento de linguagem natural que eventualmente levariam ao desenvolvimento das assistentes virtuais modernas.

Para competir por clientes na década de 1990, empresas como IBM, Philips e Lemont e Hauspie começaram a integrar o reconhecimento de voz digital em computadores pessoais. O primeiro smartphone introduzido em 1994, o IBM Simon, lançou as bases para os assistentes virtuais inteligentes de hoje.

De 2017 a 2021, todos os assistentes virtuais mencionados acima foram desenvolvidos, e existem assistentes virtuais mais inteligentes utilizados para atividades pessoais e profissionais. Empresas em diferentes áreas utilizam assistentes virtuais para melhorar a qualidade de suas decisões em diferentes níveis, desde operações até a alta administração. (SOFASTAEI, 2021)

## 1.7 Metodologia

A metodologia utilizada neste trabalho inclui pesquisa bibliográfica, desenvolvimento de código, testes de usabilidade e análise de resultados.

## 1.8 Estrutura do Trabalho

A presente monografia segue uma organização composta por seis capítulos distintos. O primeiro capítulo corresponde à Introdução, no qual são apresentados os objetivos e o contexto do trabalho. Em seguida, o segundo capítulo aborda a Fundamentação Teórica, fornecendo as bases conceituais necessárias para a compreensão do tema. O terceiro capítulo consiste na Revisão Bibliográfica, onde são explorados os estudos e pesquisas relevantes já realizados sobre o assunto em questão. O quarto capítulo descreve a Implementação do projeto, detalhando as etapas práticas e metodologias empregadas. Os resultados obtidos e os experimentos conduzidos são discutidos no quinto capítulo, denominado Experimentos e Resultados. Finalmente, o sexto capítulo encerra a monografia com a Conclusão, onde são apresentadas as considerações finais, destacando as contribuições do estudo e possíveis direções para futuras pesquisas. Essa estrutura visa proporcionar uma abordagem abrangente e coerente do tema abordado ao longo do trabalho.

## 2 Fundamentação teórica

Neste capítulo, serão exploradas as tecnologias fundamentais que desempenham um papel central no desenvolvimento da **Sunday**. Essas tecnologias fornecem as bases necessárias para a criação de uma assistente com uma agenda personalizada e um jogo da velha interativo, além da integração da PLN para comunicação. Entender a funcionalidade e o contexto de cada tecnologia é crucial para a compreensão do funcionamento do projeto.

### 2.1 Servidor

Nesta seção, serão abordadas as tecnologias responsáveis pela criação do servidor HTTP, que é encarregado pelo processamento das informações enviadas pelo usuário.

#### 2.1.1 Protocolo de Transferência de Hipertexto

O HTTP é o meio pelo qual programas se comunicam na *World Wide Web*. Embora tenha diversas aplicações, sua característica mais reconhecível é a facilitação de diálogos entre navegadores da *web* e servidores.(GOURLEY, 2002).

#### 2.1.2 Aplicação web

Uma aplicação *web* como "uma aplicação de software que roda na internet, em vez de funcionar com base em sistemas operacionais". Ele destaca que as aplicações *web* são sistemas completos programados com base em requisitos de engenharia de software, mas sua principal característica é operar na internet. Sacramento também ressalta que a computação em nuvem impulsionou o desenvolvimento de plataformas *web* robustas, transformando o paradigma de produtos em serviços. Ele menciona exemplos como *e-mail*, redes sociais e editores de texto online como *web apps*, alguns dos quais seguem o modelo *Software as a Service* (SaaS), exigindo assinaturas para acesso. Além disso, o autor enfatiza que o desenvolvimento de tecno-

logias *web* é uma escolha viável e atrativa para muitos profissionais de desenvolvimento de software. Sunday irá ser categorizada como uma aplicação *web*, ela irá atuar na internet e não necessitará de instalação apenas um navegador *web*. (SACRAMENTO, 2022)

### 2.1.3 Arquitetura REST

*Representational State Transfer* (REST) é um conjunto de restrições que informam o design de um sistema *hypermedia*, feito para que caso seguida corretamente seja altamente escalável, esse ponto não foi provado, mas é amplamente aceito atualmente.(WILDE, 2011)

## 2.2 Javascript

"JavaScript® é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas *web*, mas usada também em vários outros ambientes sem browser, tais como node.js, Apache CouchDB e Adobe Acrobat. O JavaScript é uma linguagem baseada em protótipos, multi-paradigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos (como por exemplo a programação funcional)."

"O padrão JavaScript é ECMAScript. Desde 2012, todos os navegadores modernos possuem suporte total ao ECMAScript 5.1. Navegadores mais antigos suportam pelo menos ECMAScript 3. Em 17 de Junho de 2015, a ECMA International publicou a sexta versão do ECMAScript, que é oficialmente chamado de ECMAScript 2015, e foi inicialmente conhecido como ECMAScript 6 ou ES6. Desde então, as especificações do ECMAScript são lançadas anualmente."(MOZILLA.ORG, 2023b)

## 2.3 NodeJS

O Node é uma plataforma de tempo de execução Javascript assíncrona orientada a eventos, o Node foi projetado para construir aplicações de rede escaláveis. a Figura 2.1 exemplifica como é feito um *hello world* em node (NODEJS.ORG, 2023)

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
    res.send('hello world')
})

app.listen(3000)
```

Figura 2.1: Hello World em Node (Fonte: Autor).

## 2.4 Framework

O *framework* é uma base que serve de auxílio na criação de sistemas complexos com uma finalidade específica, facilitando o desenvolvimento, que, dependendo do que será desenvolvido, pode ser muito trabalhoso.

Por ser um pacote de códigos padronizado, ele já vem com um objetivo pelo qual foi desenvolvido, como, por exemplo, o Express, que é um *framework* desenvolvido exclusivamente para a criação de servidores HTTP. Uma das grandes vantagens na utilização de um *framework* é o alto ganho na produtividade no momento em que o software está em desenvolvimento, economizando assim uma grande parte do processo. O acoplamento é uma grande desvantagem, pois causa uma grande dependência do *framework*, como por exemplo, fragilidades no código que podem resultar em erros de segurança (KRIGER, 2022).

### 2.4.1 Express

Express é um framework para Node que fornece recursos mínimos para a construção de servidores *web*. O código da Figura 2.2 ilustra como é simples iniciar um servidor HTTP com Express.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Figura 2.2: Hello world retornado por um servidor HTTP usando express (Fonte: Autor).

#### 2.4.2 Processamento de Linguagem Natural

O PLN é a área da Ciência da Computação que se dedica ao desenvolvimento de programas de computador capazes de analisar, reconhecer e/ou gerar texto em linguagens humanas, ou seja, linguagens naturais, é conhecida como PLN. É uma tarefa complexa devido à rica ambiguidade presente na linguagem natural, o que o torna diferente do processamento de linguagens de programação de computador, que são formalmente definidas e evitam ambiguidades.

O PLN tem vários objetivos comuns, incluindo:

- **(a)** Recuperação de informação a partir de textos.
- **(b)** Tradução automática.
- **(c)** Interpretação de textos.
- **(d)** Realização de inferências a partir de textos.

É importante destacar que muitos esforços na área de PLN atualmente abordam múltiplos desses objetivos simultaneamente.(LIDDY, 2001)

## 2.5 Frontend

Esta é a parte responsável pela criação da interface visual da aplicação. As páginas da aplicação são desenvolvidas em *hypertext markup language* (HTML), *Cascading Style Sheets* (CSS) e JavaScript (linguagem de programação que já foi mencionada anteriormente).

## 2.5.1 React

O React é uma biblioteca de Javascript que facilita a criação de aplicações frontend, com os seus métodos de como criar uma página *web*, aplicativos ou aplicações *desktop*, ele é focado em componentização, que seria, separar o códigos em várias partes, as que mais se repetem, e economizar linhas de código, por exemplo, um item em uma lista, geralmente eles tem o mesmo jeito, só muda o nome ou a imagem, assim possibilitando a componentização (REACT.DEV, 2023).

a Figura 2.3 ilustra um exemplo de como é a estrutura de código do React

```
1 function MyButton() {
2   return (
3     <button>
4       I'm a button
5     </button>
6   );
7 }
8
9 export default function MyApp() {
10   return (
11     <div>
12       <h1>Welcome to my app</h1>
13       <MyButton />
14     </div>
15   );
16 }
```

Figura 2.3: Criando um componente básico com REACT (Fonte: Autor).

O React consegue simplificar a implementação de algo mais complexo, exigindo menos linhas de código. Seu método fácil de componentização economiza tempo e agiliza a criação de páginas muito complexas, tornando o React uma das bibliotecas mais populares no mundo do JavaScript. A Figura 2.4 é apenas um exemplo de como a aplicação aparecerá em um navegador web.

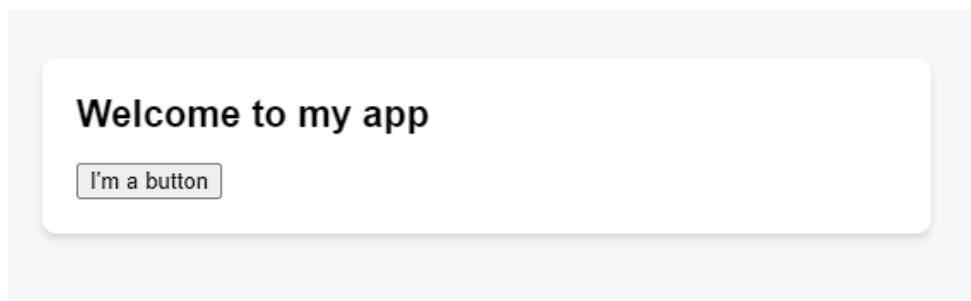


Figura 2.4: Ilustrando um componente básico com REACT (Fonte: Autor).

## 2.5.2 Captação da Voz

A voz do usuário é capturada utilizando a dependência *react-speech-kit*. Esta biblioteca identifica as falas do usuário e as transforma em texto, utilizando o microfone capturado pelo navegador.(COLETIVA, 2020b)

## 2.5.3 NLP.js

Esta dependência é responsável pela gerenciamento da linguagem natural da **Sunday**, ao utilizá-la é possível fazer o treinamento das perguntas e respostas da Assistente.(COLETIVA, 2020a)

## 2.5.4 Regex

*Regular Expressions* (REGEX) são padrões utilizados para selecionar combinações de caracteres em uma string. O *frontend* utiliza REGEX para separar a data de eventos para agenda quando um usuário requisitar um agendamento para a **Sunday**. (MOZILLA, 2023)

## 2.5.5 Cascading Style Sheets

O CSS é uma linguagem de marcação utilizada para definir a apresentação visual de documentos HTML e *Extensible Markup Language* (XML), bem como outros tipos de documentos web. O CSS descreve como os elementos HTML devem ser exibidos na tela, no papel, na fala ou em outras mídias, controlando aspectos como layout, cores, fontes, espaçamento, tamanho e posicionamento dos elementos (MOZILLA.ORG, 2023a).

Trecho de código 2.1: Exemplo de expressão regular

---

```
/(\d+)\s+de\s+(\w+)\s+(\w+)\s+(\d{4})\s+(\d{2})\s+horas/g
```

---

### 2.5.6 Expressão Utilizada

A expressão regular no trecho de código 2.1 é utilizada pelo sistema para identificar, na fala do usuário, os dias do evento que ele deseja agendar no calendário. Primeiramente, são buscadas palavras que correspondam a datas. Se não houver um ano especificado, é utilizado o ano atual. Caso não haja uma hora especificada, é configurado o horário para meia-noite. Se tanto a hora quanto o ano forem especificados, ambos são utilizados no agendamento do evento no calendário.

## 2.6 Armazenamento de Dados

Nesta seção, serão exploradas as formas de armazenamento, uma vez que o banco utilizado é o MongoDB, um banco NoSQL. Será explicado o banco de dados orientado a documentos.

### 2.6.1 Banco de dados

"Um banco de dados é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador."(ORACLE.COM, 2023)

### 2.6.2 Modelo Relacional

"O modelo relacional em bancos de dados é fundamentado no princípio de que dados são guardados em tabelas. Toda sua definição é teórica e baseada na teoria dos conjuntos, ramo da matemática que estuda conjuntos, que são uma coleção de elementos. O modelo relacional foi idealizado por Edgar Frank Codd, que o descreveu no artigo “Relational Model of Data for Large Shared Data Banks” (“Modelo de dados relacional para grandes bancos de dados compartilhados”) quando era pesquisador da IBM em San José. Com o passar do tempo, o modelo relacional se tornou o sucessor do modelo hierárquico e do modelo em rede, amplamente utilizados anteriormente. O esquema de uma relação é inváriável ao longo do tempo, sendo modificado apenas por comandos específicos.”(OLIVEIRA, 2014)

### 2.6.3 Modelo Não Apenas Relacional

"Para evitar o custo da escalabilidade em ambientes relacionais, iniciou-se, ao longo do tempo, o desenvolvimento de bancos distribuídos capazes de gerenciar dados semiestruturados

provenientes de diversas origens e que possibilitavam escalabilidade mais barata e menos complexa, não necessitando de servidores muito robustos e nem um grande número de profissionais para o gerenciar.(...) Os bancos de dados não relacionais são classificados em Bancos de esquema Chave/Valor (Key/Value Store), Bancos de dados orientados à documentos, Bancos de dados de Colunas e Bancos de dados de Grafos."(OLIVEIRA, 2014)

#### 2.6.4 Banco de Dados Orientados a Documentos

"Bancos de dados orientados a documentos são baseados no armazenamento de pares de chave-valor, tendo um esquema altamente flexível. Esta característica torna os bancos de dados orientados à documentos ótimas opções para dados semi-estruturados, como os utilizados em ferramentas web colaborativas."(OLIVEIRA, 2014)

### 2.7 Docker

Docker é uma plataforma de contêinerização que permite a criação e o gerenciamento de contêineres, oferecendo uma abstração eficiente e leve do sistema operacional. Um contêiner pode ser visto como uma unidade encapsulada e autossuficiente que contém todos os recursos necessários para executar um aplicativo, incluindo bibliotecas, dependências, configurações e o próprio código do aplicativo. A tecnologia Docker utiliza recursos do kernel do Linux, como namespaces e cgroups, para fornecer isolamento eficiente entre contêineres e garantir a portabilidade e consistência das aplicações em diferentes ambientes.

Principais Componentes e Conceitos do Docker:

- **Imagen:** Uma imagem Docker é uma representação estática de um aplicativo, contendo tudo o que é necessário para sua execução. Isso inclui o sistema de arquivos do aplicativo, bibliotecas, dependências e metadados. As imagens são usadas como base para criar contêineres em execução.
- **Contêiner:** Um contêiner é uma instância em execução de uma imagem. Ele encapsula o ambiente de execução do aplicativo, garantindo isolamento dos recursos do sistema operacional. Contêineres são efêmeros e podem ser facilmente criados, iniciados, parados, movidos e excluídos.
- **Dockerfile:** Um Dockerfile é um arquivo de configuração que descreve como uma imagem Docker deve ser construída. Ele inclui instruções para copiar arquivos, instalar de-

pendências e configurar o ambiente necessário para o aplicativo.(DOCKER, 2023)

O Docker é utilizado na criação do banco responsável pela **Sunday**. O nome do banco é **Sunday**, e são criadas duas tabelas: uma para *user* e outra para *schedule*. A tabela *user* é responsável pelo armazenamento dos usuários persistidos no sistema, enquanto a tabela *schedule* é responsável pela agenda de cada usuário.

### 3 Revisão bibliográfica

Este capítulo aborda exemplos de Assistentes Virtuais que inspiraram a **Sunday**, de onde foram inspiradas funções, como a agenda e o reconhecimento de fala humana.

#### 3.1 Google Assistant

O Google Assistant implementa PLN. Ele gera respostas que se assemelham à linguagem humana, com base nas informações enviadas para o Google Assistant.

##### 3.1.1 Arquitetura do Google Assistant

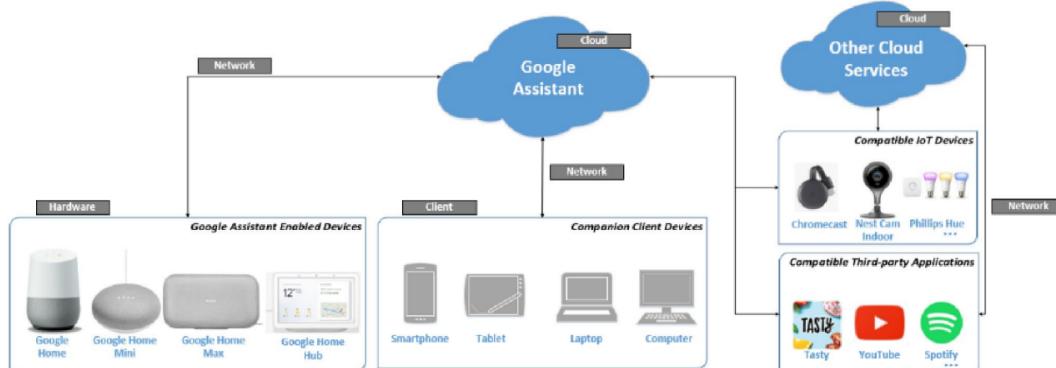


Figura 3.1: Google Assistant ecosystem (fonte: store.google.com)

A Figura 3.1 ilustra a arquitetura detalhada do Sistema de Reconhecimento de Voz do Google Assistant e o fluxo que a informação percorre, desde o *hardware* até a nuvem. Primeiramente, a voz do usuário é captada por um *hardware* ou por um cliente e, em seguida, é diretamente redirecionada para a nuvem. Quando a informação chega à nuvem, é processada pelo *backend* do Google, que é o próprio Google Assistant. Depois, uma resposta é enviada de volta ao usuário.

### **3.1.2 Como Funciona o Google Assistant**

Recentemente, houve uma grande demanda por parte dos consumidores por assistentes de voz inteligentes, como o Amazon Echo e o Google Home, que apresentam um assistente digital ativado por voz que permite aos usuários controlar centros de automação residencial e outros dispositivos IoT usando comandos de voz. O assistente digital ativado por voz do Google, semelhante ao Alexa da Amazon, à Siri da Apple e ao Cortana da Microsoft, é chamado de Google Assistant. Como um serviço de assistente digital virtual na nuvem, o Google Assistant interage com vários dispositivos compatíveis, aplicativos nativos do Google e alguns aplicativos de terceiros, convertendo as solicitações de voz em protocolos de comunicação nativos. Os dispositivos Google Nest (tanto a linha Nest quanto o Google Home) são compatíveis por padrão com o Google Assistant. O Google Assistant oferece comandos de voz e interações conversacionais que permitem aos usuários realizar buscas de voz na Internet, controlar dispositivos com automação de voz, ouvir música, fazer chamadas de vídeo e mensagens de vídeo com amigos e familiares, entre outras funções, após o uso das palavras de ativação "OK Google" ou "Hey, Google". (AKINBI, 2020)

## **3.2 Siri**

A Siri é uma assistente virtual da Apple. Sua arquitetura utiliza alguns elementos específicos da Apple e é desenvolvida na linguagem própria da empresa, a linguagem Swift.

### 3.2.1 Arquitetura da Siri

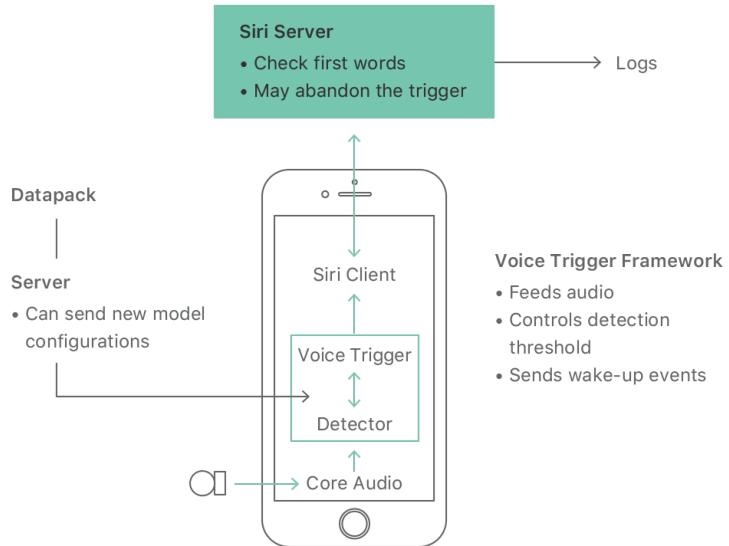


Figura 3.2: Siri Fluxo (Fonte: machinelearning.apple.com)

A Figura 3.2 ilustra o caminho que a informação percorre. No iPhone, a voz do usuário é captada. Após essa captação, a voz é redirecionada para o servidor da Siri, onde as primeiras palavras enviadas pelo usuário são verificadas. Em seguida, ocorre outra verificação para determinar se a ação da Siri deve ser abortada ou continuada. Se a ação prosseguir, a resposta é redirecionada de volta para o iPhone do usuário.

### 3.2.2 Como Funciona a Siri

A funcionalidade "*Hey Siri*" permite que os usuários acessem a Siri sem a necessidade de tocar em um botão. Um pequeno reconhecedor de fala permanece ativo o tempo todo, ouvindo atentamente apenas as palavras-chave "*Hey Siri*". Quando essa expressão é detectada, a Siri entra em ação, interpretando o restante da fala como um comando ou consulta. O detector "*Hey Siri*" utiliza uma Rede Neural Profunda (DNN) para transformar o padrão acústico da voz em tempo real em uma distribuição de probabilidade relacionada aos sons da fala. Em seguida, ele emprega um processo de integração temporal para calcular um índice de confiança de que a frase pronunciada corresponde a "*Hey Siri*". Se o índice de confiança atingir um nível suficientemente alto, a Siri é ativada, proporcionando uma experiência de acesso rápido e sem a necessidade de tocar em um botão(APPLE, 2023)

### 3.3 Comparação entre as Assistentes

Esta comparação ocorre entre as funcionalidades, sendo um dos motivos que a **Sunday** tem recursos mais limitados em comparação com o Google Assistant ou a Siri. A diferença é mais evidente na implementação dos componentes internos da aplicação e no treinamento manual da **Sunday**.

Na Tabela 3.1, ficam evidentes as diferenças entre as tecnologias utilizadas. Um dos diferenciais da **Sunday** é ser mono-língua, utilizando apenas Javascript para o fluxo completo de dados, não dependendo de Python ou qualquer outra linguagem para seu funcionamento.

Tabela 3.1: Comparação de Tecnologias Utilizadas.

Trabalho	Linguagem	Bibliotecas
Google Assistant	Prioritariamente Python	TensorFlow
Siri	Prioritariamente Swift	Intent
<b>Sunday</b>	Javascript	NodeNLP

A Tabela 3.2 demonstra algumas funcionalidades compartilhadas entre as assistentes. Como a **Sunday** é mais limitada, ela se concentra em treinamento manual para armazenar informações em memória e aprender a interpretar algumas falas.

Tabela 3.2: Comparação de funcionalidades.

Assistente	Armazenamento	Resposta em tempo real	Treinamento Manual com JSON
Google Assistant	Sim	Sim	Não
Siri	Sim	Sim	Não
<b>Sunday</b>	Sim	Sim	Sim

#### 3.3.1 Treinamento Manual

O treinamento manual é delegado para a biblioteca **NLP.js**. Ela recebe um arquivo de texto enviado pelo próprio usuário, com o formato em *JavaScript Object Notation* (JSON), contendo as informações para o treinamento da **Sunday**. Após receber o arquivo de texto em sua própria estrutura, a **Sunday** identifica o que é uma pergunta e uma resposta para esta mesma pergunta. Em seguida, é iniciada a separação das perguntas contidas no arquivo. O próximo passo para a separação é a geração de tags com algumas informações únicas de cada pergunta, e são gerados arquivos para serem consultados pela **Sunday**.

## 4 *Implementação*

Neste capítulo, será descrito todo o processo de criação da **Sunday**, com detalhes e foco na arquitetura. Será abordado como a **Sunday** mantém uma instância de um objeto para cada usuário, além de como a voz é coletada e reconhecida, entre outros pontos.

### 4.1 Arquitetura do Sistema

A modelagem de sistemas visa apresentar várias perspectivas do sistema a ser representado, examinando-o e criando representações sob várias abordagens. O objetivo é alcançar uma representação completa, onde cada diagrama se complementa mutuamente.(GUEDES, 2020)

#### 4.1.1 Diagrama de Casos de Uso

O diagrama de casos de uso, o mais amplo e informal entre os diagramas da *Unified Modeling Language* (UML), é comumente empregado nas fases iniciais do levantamento e análise de requisitos do sistema. Contudo, ele mantém sua relevância ao longo de todo o processo de modelagem e pode servir de alicerce para a criação de outros diagramas. Este diagrama utiliza uma linguagem simples e de fácil compreensão, proporcionando uma visão geral do comportamento planejado do sistema aos usuários. Seu principal propósito é a identificação dos atores, que podem ser usuários, outros sistemas ou até mesmo componentes de hardware específicos, os quais interagirão de alguma forma com o software.(GUEDES, 2020)

Na Figura 4.1 é apresentado o Diagrama de Casos de Uso que exibe as funcionalidades que do sistema.

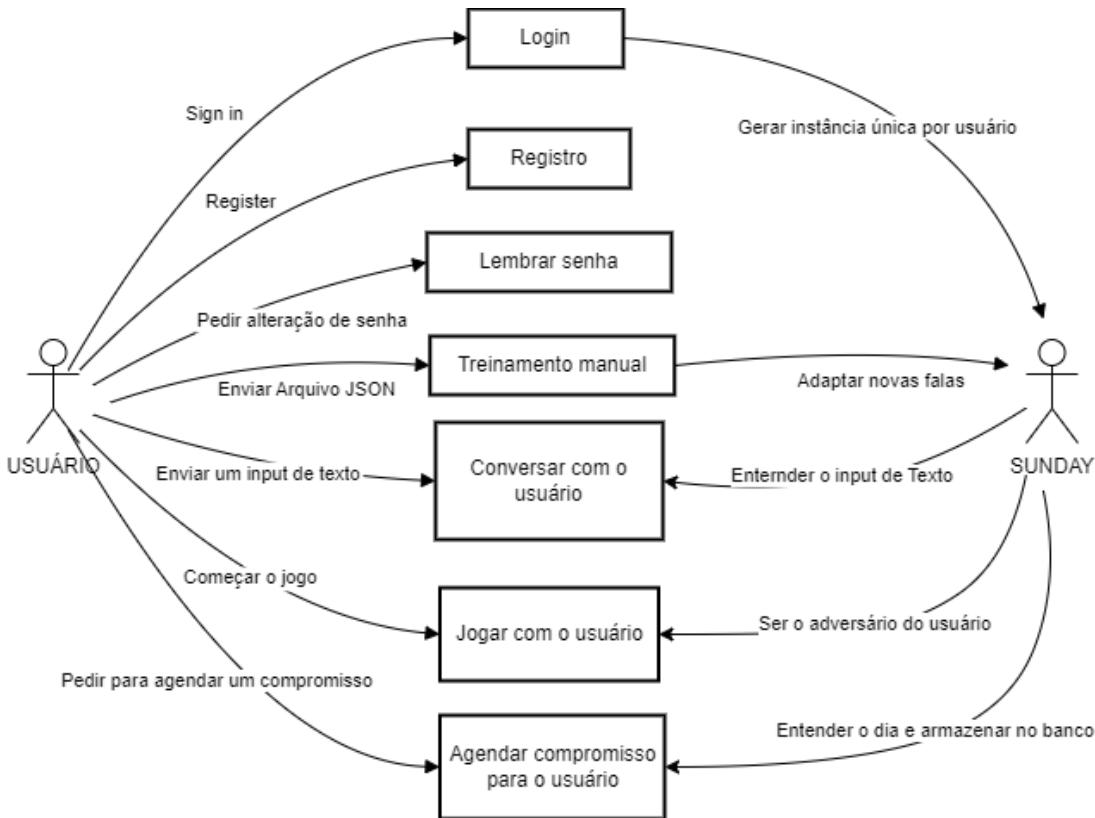


Figura 4.1: Diagrama de casos de uso

#### 4.1.2 Detalhando diagrama de casos de uso

Os Casos de Uso da Figura 4.1 ilustram as possibilidades do usuário e as capacidades da **Sunday**. O usuário pode realizar a conexão com o sistema, registrar-se, resgatar a senha e treinar a **Sunday**. As próximas três possibilidades são compartilhadas entre o Usuário e a **Sunday**, que incluem as Conversas, o Jogo da Velha e a Agenda, exigindo uma resposta da **Sunday**.

#### 4.1.3 Diagrama de Fluxo de Dados

Um Diagrama de Fluxo de Dados mapeia o fluxo de informações para qualquer processo ou sistema. Ele utiliza símbolos definidos, como retângulos, círculos e setas, além de rótulos de texto curtos, para mostrar as entradas, saídas, pontos de armazenamento de dados e as rotas entre cada destino. Os diagramas de fluxo de dados podem variar desde visões gerais de processos simples.(LUCIDCHART, 2023)

Na Figura 4.2 é apresentado o Diagrama de Casos de Uso que exibe as funcionalidades que do sistema.

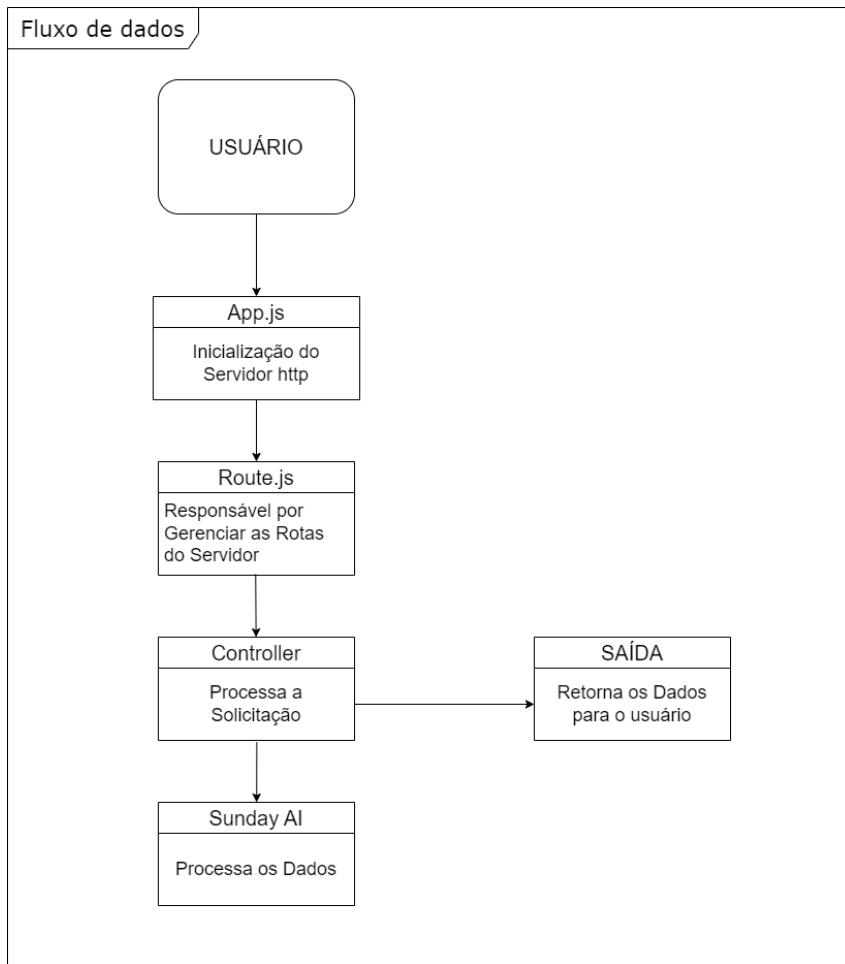


Figura 4.2: Sunday Fluxo (Fonte: Autor)

#### 4.1.4 Detalhando o Fluxo

O fluxo ilustrado na Figura 4.2 do sistema é bastante linear e segue uma lógica simples, o que facilita a compreensão do sistema. O usuário inicia o processo gerando um pedido para o sistema, passando por uma *Application Programming Interface* (API) REST, onde realiza uma requisição para iniciar o fluxo do sistema. O primeiro passo ocorre no arquivo `app.js`, onde o sistema escuta as requisições HTTP que chegam no endereço do servidor. Ele categoriza essas requisições como *GET*, *POST*, *DELETE* ou *UPDATE* (poderiam haver outras, como *OPTIONS*, *HEAD*, mas o foco do sistema são apenas as mais utilizadas). Ao identificar o tipo de requisição, ele é redirecionado para uma das rotas existentes na Tabela 4.1.

#### 4.1.5 Processamento de Dados

O processamento da **Sunday** está localizado em arquivos dentro da pasta `controller`, os quais recebem as informações provenientes das rotas. A Assistente está dividida em três partes:

- Processamento de Mensagens.
- Processamento do Jogo.
- Processamento de Datas.

#### 4.1.6 Processamento de Mensagens

Quando o usuário acessa o sistema, o treinamento da **Sunday** é carregado imediatamente. Após ser treinada e receber uma mensagem pelo chat, o fluxo das informações chega ao controlador. Então, a instância da **Sunday**, única para cada usuário, é chamada. Nessa instância, a função *process* é acionada. Esta função utiliza a mensagem do usuário e processa uma resposta para ser enviada de volta ao usuário.

#### 4.1.7 Processamento do Jogo

Quando o usuário se comunica por voz, dizendo "Começar o jogo", a instância inicia um tabuleiro vazio. O controlador, então, fica encarregado de receber as requisições responsáveis pelo jogo. O usuário pode, por exemplo, dizer "Casa um", e então a casa um do tabuleiro é preenchida. Posteriormente, a Assistente seleciona a casa a ser preenchida. O processo continua dessa forma até que um vencedor seja encontrado.

#### 4.1.8 Processamento de Datas

O usuário tem a possibilidade de agendar eventos no sistema por meio de requisições feitas pela própria voz. Quando é dito, por exemplo, "Evento de aniversário de Fulano no dia 28 de dezembro", a parte do *frontend* da aplicação utiliza um REGEX para separar o dia, mês, ano (caso exista), hora, minuto e segundo. Após essa separação, as informações são enviadas para o servidor para serem persistidas. O controlador, por sua vez, apenas insere no banco de dados MongoDB.

Tabela 4.1: Lista de Rotas HTTP

TIPO	ROTA	Explicação
POST	basic-training	Esta é a rota de treinamento básico para testes
GET	loadSubject/:subject	Esta é a rota que testa as respostas do treino
POST	loadSubject/:id	treinamento manual por arquivo JSON
POST	speech/:id	Esta é a rota de conversa da assistente por usuário
POST	signup	Esta é a rota de cadastro de usuário
POST	login	Esta é a rota de login do usuário
GET	logout	Esta é a rota de logout do usuário
POST	tictactoe/:id	Esta é a rota de que o jogo retorna as informações
POST	tictactoePlayer/:id	Esta é a rota que o usuário joga
GET	init/:id	Esta é a rota que inicia o jogo
POST	event/:id	Esta é a rota que insere um evento na agenda
GET	event/:id	Esta é a rota que retorna um evento da agenda
GET	event/:id/:initDate/:endDate	Esta é a rota que retorna por data
DELETE	init/:id	Esta é a rota que deleta um evento da agenda

Ao serem requisitadas algumas das rotas mostradas acima pelo *frontend* da aplicação, a informação percorre pelo caminho escolhido. No entanto, para ser possível realizar consultas, é necessário estar registrado no sistema e conectado. A arquitetura da **Sunday** previne o compartilhamento de informações entre os usuários, tornando-as únicas para cada usuário. Uma instância da assistente é criada toda vez que o usuário acessa o sistema. Após o registro no sistema, utilizar a **Sunday** é apenas uma questão de utilizá-lo.

#### 4.1.9 Diagrama de Classes

O Diagrama de Classes, uma ferramenta essencial na linguagem de modelagem UML, desempenha um papel central na representação da organização das classes em um sistema. Ele detalha as propriedades e funções de cada classe, estabelecendo conexões e interações entre elas. Esse diagrama serve como alicerce para a criação de outros diagramas complementares.(GUEDES, 2020).

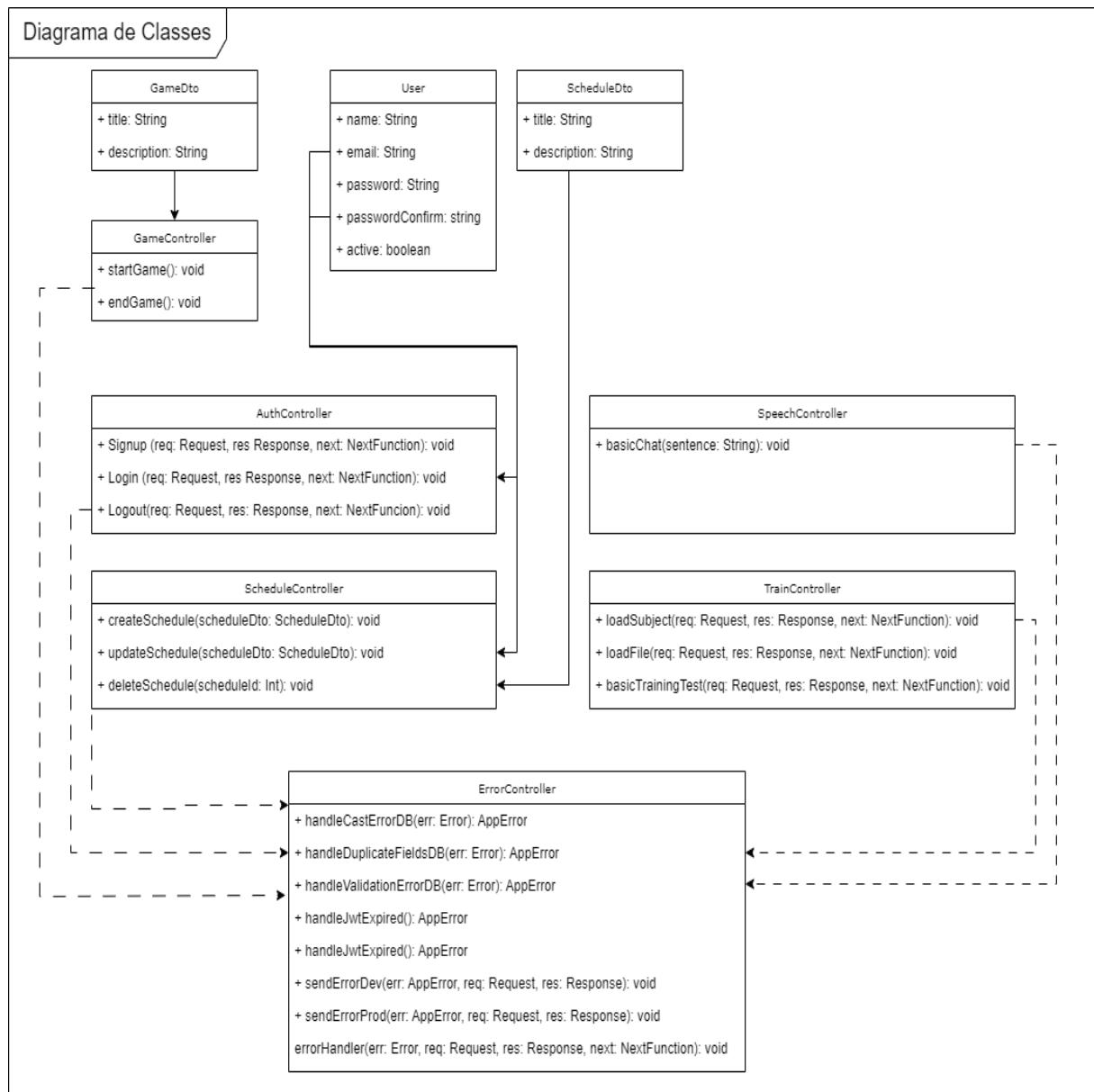


Figura 4.3: Diagrama de Classes (Fonte: Autor)

#### 4.1.10 Detalhando Diagrama de Classes

O Diagrama de Classes apresentado na Figura 4.3 ilustra como os arquivos se conectam com o arquivo de tratamento de erros da **Sunday** e mostra as funções e o tipo de retorno de cada uma. Também é apresentado o Data Transfer Object (DTO) utilizado por cada arquivo. Todos os arquivos responsáveis pelo redirecionamento das informações estão ligados ao "Error-Controller", que tem a tarefa simples de identificar erros e tratá-los para o usuário.

### 4.1.11 Mapeamento Objeto-Relacional

Considerando que tenha sido feita uma modelagem conceitual do sistema utilizando a UML, aderindo ao paradigma orientado a objetos, e o banco de dados tenha sido projetado de acordo com o modelo de dados relacional, foi necessário realizar a conversão do diagrama de classes para o diagrama de entidade-relacionamento, como mostrado na Figura 4.4.

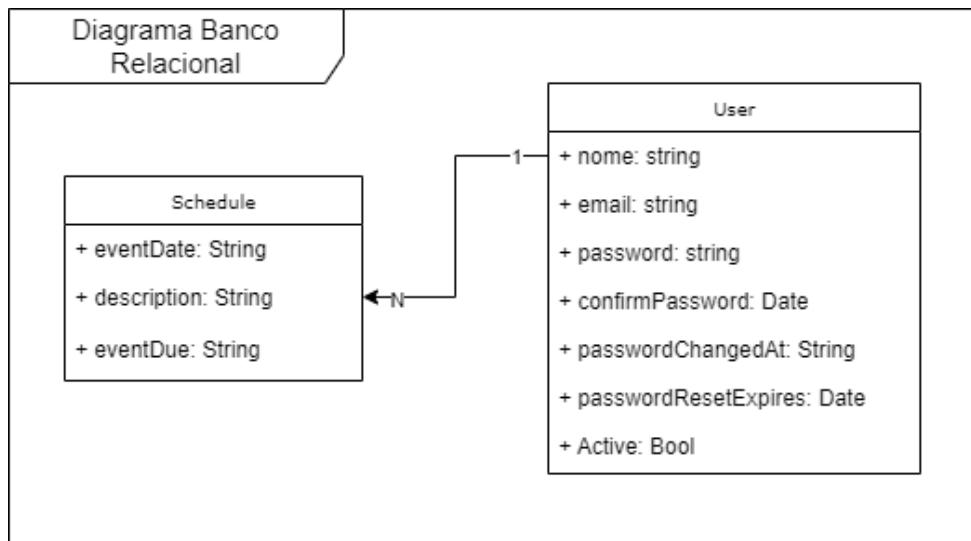


Figura 4.4: Sunday Diagrama de Classes (Fonte: Autor)

Não é necessário mais do que isso, uma vez que o treinamento da **Sunday** é realizado em memória e armazenado por meio de arquivos gerados pela biblioteca do Node.

## 4.2 Levantamento de Requisitos

A fase inicial de um processo de desenvolvimento de *software* é a coleta de requisitos.(GUEDES, 2020)

### 4.2.1 Requisitos Funcionais

Os Requisitos Funcionais correspondem ao que o cliente quer que o sistema realize, ou seja, as funcionalidades do *software*.(GUEDES, 2020)

Na Tabela 4.2 tem a lista de todos os requisitos impostos pelo sistema.

Tabela 4.2: Tabela de requisitos funcionais

Requisitos Funcionais	Caso de Uso
RF01 - Eu como usuário quero logar no sistema	UC01
RF02 - Eu como usuário quero redefinir minha senha	UC02
RF03 - Eu como usuário quero me cadastrar	UC03
RF04 - Eu como usuário quero conversar por texto com a Sunday	UC04
RF05 - Eu como usuário quero conversar utilizando a voz	UC05
RF06 - Eu como usuário quero agendamento de compromissos por voz	UC06
RF07 - Eu como usuário quero o jogo da velha por voz	UC07
RF08 - Eu como usuário quero jogo da velha manualmente	UC08
RF09 - Eu como usuário quero o treinamento manual da Assistente	UC09

### 4.2.2 Requisitos Não Funcionais

Requisitos não Funcionais, em um contexto de desenvolvimento de *software* ou projeto de sistemas, são elementos igualmente essenciais, mas distintos dos Requisitos Funcionais. Eles definem as características e propriedades do sistema que não estão relacionadas diretamente às funcionalidades específicas que o sistema deve desempenhar, mas que têm um impacto significativo em sua qualidade, desempenho e usabilidade. Os Requisitos não Funcionais ajudam a determinar como um sistema deve ser, em vez do que ele deve fazer.(GUEDES, 2020)

Na Tabela 4.3, temos uma lista de requisitos não funcionais impostos pelo sistema.

Tabela 4.3: Tabela de requisitos não-funcionais

Requisitos Não Funcionais
RNF01 - O sistema deve ser intuitivo
RNF02 - O sistema deve ter o frontend desenvolvido em Javascript (React)
RNF03 - O sistema deve ter o backend desenvolvido em Javascript (Node)
RNF04 - O sistema deve se comunicar com o banco NoSql Mongodb

### 4.2.3 Regras de Negócio

As regras de negócio que definem como o sistema funciona incluem especificações detalhadas da aplicação e suas funcionalidades que foram desenvolvidas. Essas diretrizes abrangem diversos aspectos cruciais, como as regras que o cadastro deve seguir e o que é obrigatório na

senha, por exemplo, caracteres especiais, letras e números.

A Tabela 4.4 lista as regras de negócio seguidas pela **Sunday**.

Tabela 4.4: Tabela de regras de negócio

Regras
RN01 - A senha deve conter caracteres especiais
RN02 - O cadastro deve conter um email
RN03 - O treinamento manual deve ser feito por um arquivo JSON
RN04 - A Assistente de responder no mínimo por texto
RN05 - A Assistente deve ter uma instância por usuário

## 4.3 Dicionário de dados

Para descrever as entidades no banco de dados é necessário entender como uma informação é armazenada em um **Banco não Apenas Relacional** ou Not Only SQL (NoSQL) mais especificamente em MongoDB.

Ao armazenar em MongoDB não é necessária a criação de tabelas, elas são criadas de acordo com código e a necessidade do desenvolvedor na hora da inserção o armazenamento é feito no formato JSON e a **Sunday** utiliza uma foreign key com o id do usuário.(ORACLE, 2023)

Os tipos de dados suportados pelo JSON são:

- SEQUÊNCIA que é JSON é composta de caracteres Unicode, com escape de barra invertida. Ex: { "name": "Jones" }
- NÚMERO que segue o formato de ponto flutuante de precisão dupla do JavaScript. Ex: {Num: 1}
- OBJETO é um conjunto de pares de nomes ou valores inseridos entre chaves. As chaves devem ser sequências e devem ser exclusivas, separadas por vírgula. Ex: { "Objeto": { "name": "Jakson", "idade": "22" } }
- (d) MATRIZ é tipo de dado que é uma coleção ordenada de valores. Ex: [1,2,3]
- (d) BOOLEANO são designados como true ou false. Ex: { "Ativo": false }
- (d) NULO é um valor vazio. Ex: { "Vazio": null }

### 4.3.1 Tabela User

Esta é a tabela que tem como objetivo armazenar os usuários, para que, durante o login, as informações do mesmo sejam buscadas. A Tabela 4.5 mostra os campos do usuário.

Tabela 4.5: Lista de Rotas HTTP

CHAVE	ATRIBUTOS	TIPO	PROPRIEDADES
PK	_id	Sequência	NOT NULL
	nome	Sequência	NOT NULL
	nome	Sequência	UNIQUE
	password	Sequência	NOT NULL
	confirmPassword	Sequência	NOT NULL
	passwordChangedAt	Sequência	NULL
	passwordResetExpires	Sequência	NULL
	Active	Booleano	TRUE

- **\_id:** Essa é a "Primary Key" da tabela, onde tem sua chave única
- **nome:** Esse é o nome do usuário para ser armazenado no banco de dados
- **email:** Esse é o email do usuário para ser armazenado no banco de dados e não pode se repetir
- **password:** Essa é a senha do usuário para ser armazenado no banco de dados, deve ser encriptografada.
- **confirmPassword:** Essa é a confirmação da senha do usuário, também deve ser criptografada
- **passwordChangedAt:** Essa é a data da última modificação da senha
- **passwordResetExpires:** Esse é o tempo limite para alterar a senha caso tenha requisitado alteração
- **Active:** Esse é o campo que checa se o usuário está ativo

### 4.3.2 Tabela Schedule

Esta tabela tem como principal finalidade armazenar informações relacionadas a eventos agendados. Cada registro na tabela

representa um evento agendado e é identificado exclusivamente pelo campo `_id`, que atua como a chave primária da tabela. Também é gerada uma *Foreign Key* (FK) de usuário para relacionar o evento com o respectivo usuário. A Tabela 4.6 lista os campos.

Tabela 4.6: Lista de Rotas HTTP

CHAVE	ATRIBUTOS	TIPO	PROPRIADEADES
PK	<code>_id</code>	String	NOT NULL
FK	<code>user_id</code>	String	NOT NULL
	<code>eventDate</code>	Date	NOT NULL
	<code>description</code>	String	NULL

- **`_id`:** Essa é a "*Primary Key*" da tabela, onde tem sua chave única
- **`user_id`:** Esse é o campo equivalente a uma FK do usuário
- **`eventDate`:** Esse é o campo que guarda a data do evento agendado
- **`description`:** Essa é a descrição do evento agendado

## 5 Telas e resultados

Neste capítulo, exploramos as interfaces e o *design* do sistema, buscando proporcionar uma maior compreensão de como é a aparência visual da aplicação.

### 5.1 Tela de Login

A primeira interface que será vista pelo usuário é a de *login*, que possui dois campos principais: usuário e senha. Além disso, há um link que redireciona o usuário para a recuperação de senha e outro link que o direciona para o formulário de registro.

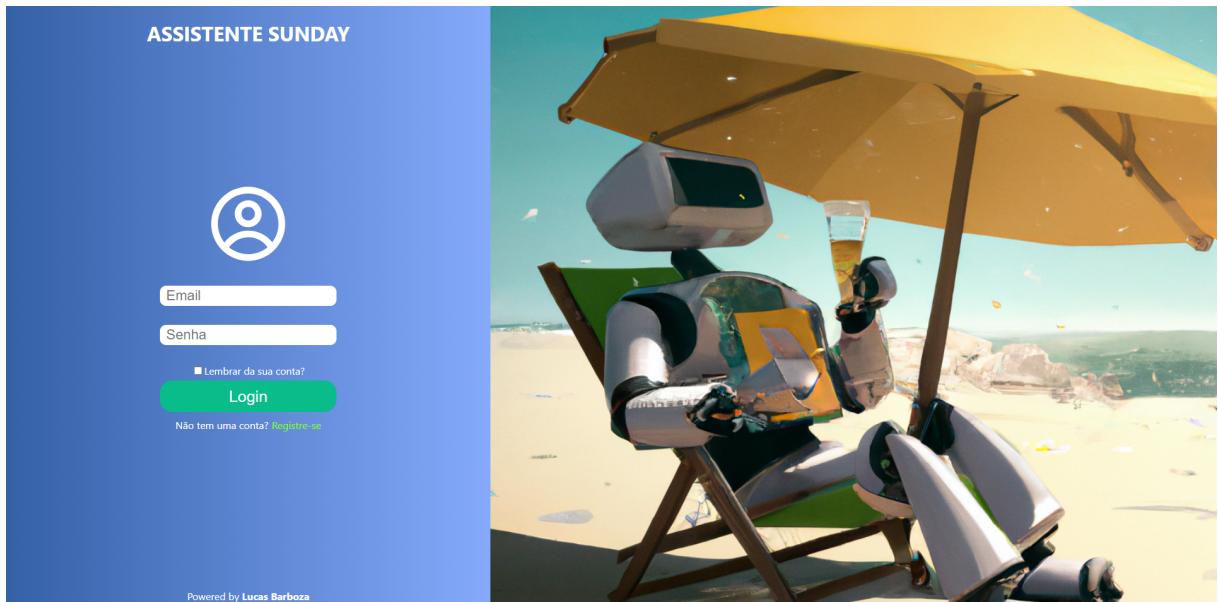


Figura 5.1: Tela de *Login* (fonte: Autor)

A Figura 5.1 ilustra a interface de *login*, que segue o Requisito Funcional RF01 e o RF02, no qual o usuário pode acessar o sistema caso esteja cadastrado e que o usuário pode redefinir sua senha. O usuário precisa digitar suas credenciais para acessar o sistema. Quando ele as insere e confirma clicando em "Login", o *frontend* envia uma requisição para o servidor por HTTP. Se as informações estiverem corretas, o sistema retorna o ID para o *frontend* do usuário

acessar. Este ID é responsável por localizar a instância única da **Sunday**. A primeira interface após o login do usuário é a interface de conversa com a **Sunday**.

## 5.2 Tela de Registro

Nesta interface, haverá um formulário para o registro do usuário. Serão necessárias apenas as informações básicas para o teste e funcionamento da aplicação, como email, nome e senha.

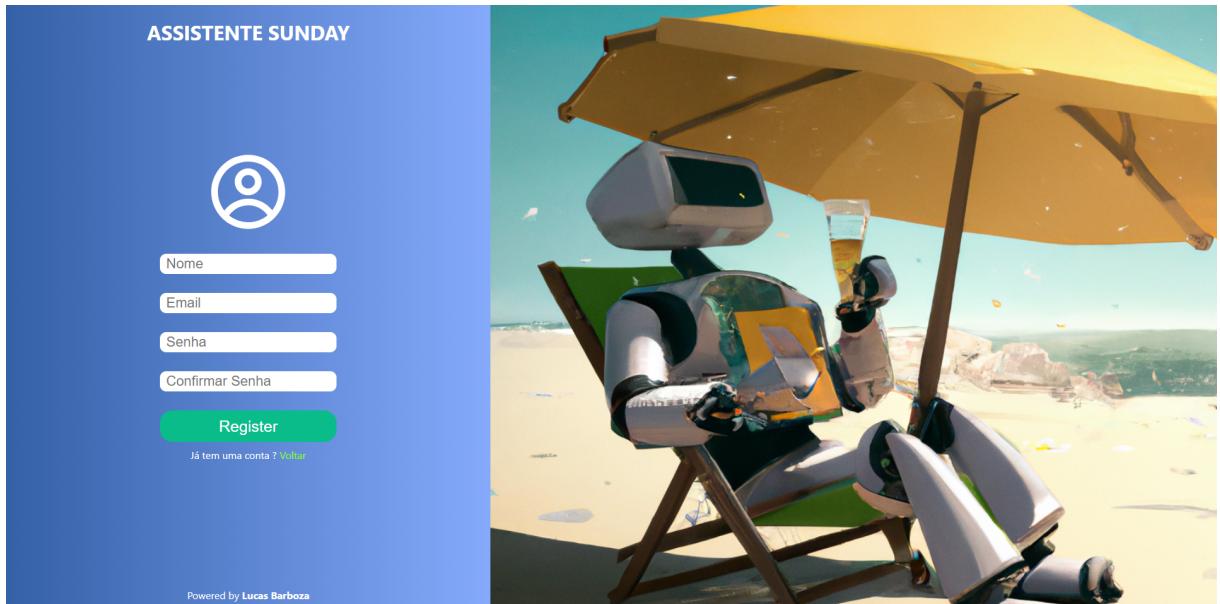


Figura 5.2: Tela de cadastro (fonte: Autor)

A Figura 5.2 ilustra a interface de cadastro, responsável pelo atendimento ao Requisito RF03 que permite que o usuário se cadastre no sistema, também requer as Regras de Negócio RN01 e RN02 que ditam que a senha deve conter caracteres especiais e o usuário deve ter um email válido. Para se registrar, o usuário deve preencher este formulário. Caso algum campo esteja fora das regras de negócio RN01 e RN02, que definem quantidade e limitações que os campos devem seguir para prosseguir com o cadastro, será necessário ajustá-los.

Após preencher todos os campos e confirmar, a requisição de cadastro será enviada para o servidor. As credenciais do usuário e as informações enviadas passarão por outra confirmação no servidor. Se confirmadas, o servidor enviará o ID criado pelo banco de volta para o usuário, possibilitando o acesso.

## 5.3 Tela de Conversa

Esta é a interface que representa a primeira interação com a **Sunday**. Ela pode responder tanto por texto quanto por voz.

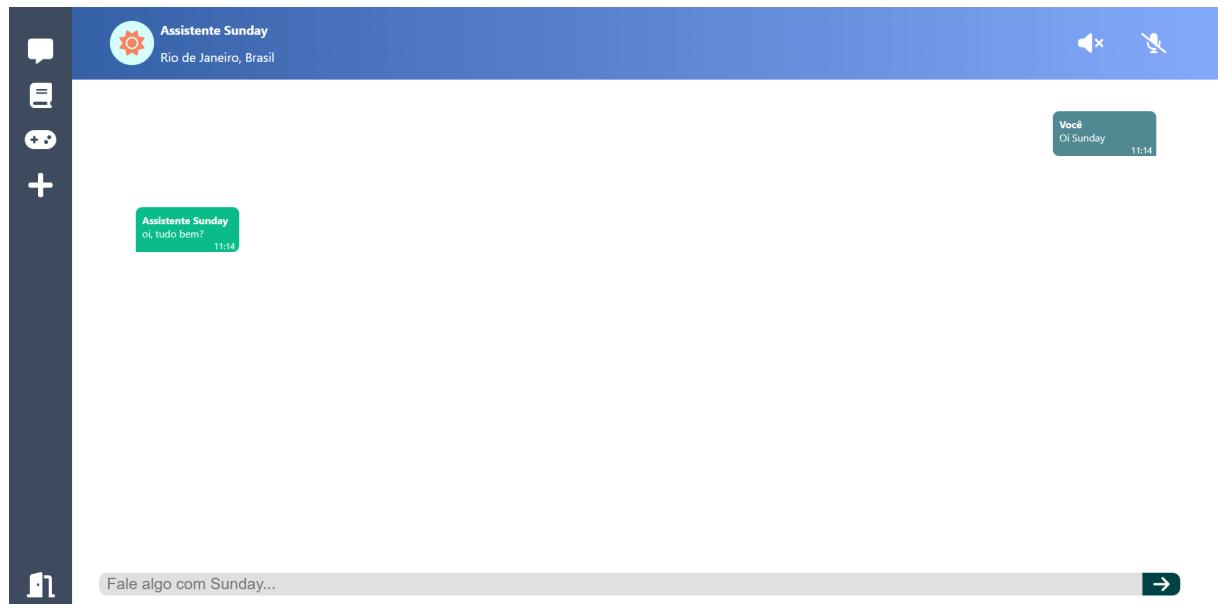


Figura 5.3: Tela da Agenda com a Sunday(fonte: Autor)

Esta interface de conversa com a **Sunday** é responsável pelos requisitos RF04 e RF05, que estabelecem que o usuário pode conversar com a **Sunday** por texto e por voz e a Regra de Negócio RN04 que exige no mínimo a interação por texto.

Quando o usuário inicia uma conversa com a **Sunday**, uma requisição é enviada para a instância única do próprio usuário, que identifica se há uma tag para categorizar o que foi enviado para o servidor. Se existir, uma resposta é direcionada de volta para o usuário; caso contrário, a **Sunday** retorna "Não Entendi".

## 5.4 Agenda com a Sunday

Esta é a interface usada para armazenar e mostrar eventos que a **Sunday** guardou em seu banco de dados.

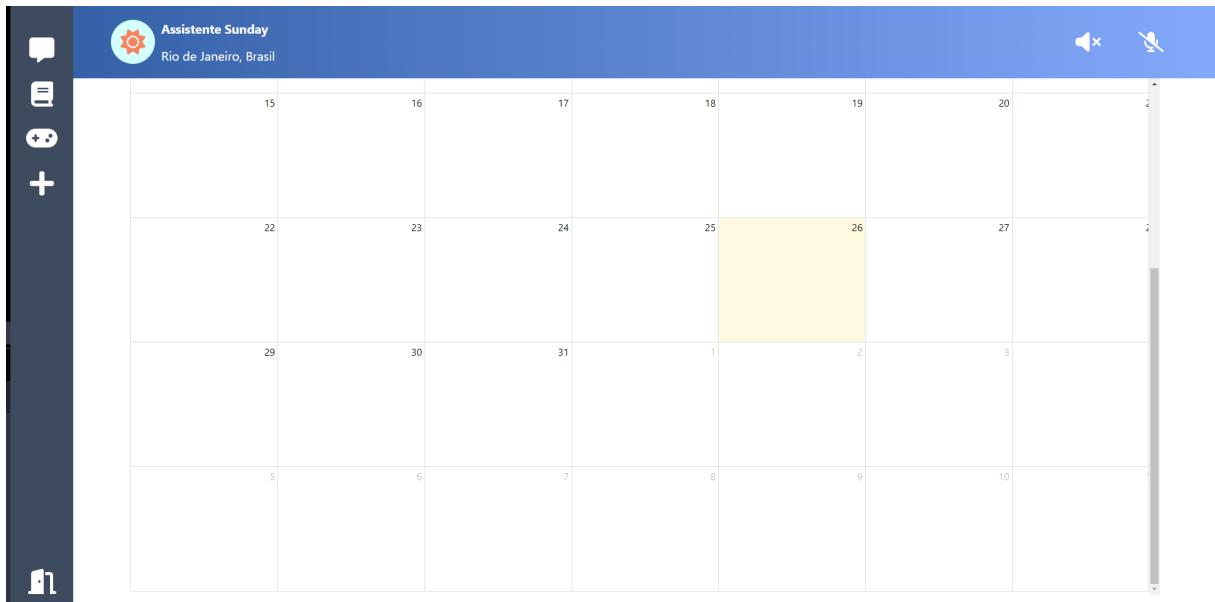


Figura 5.4: Tela de Eventos (fonte: Autor)

A Figura 5.4 ilustra a interface de agenda, que atende ao requisito RF06, o qual define que o usuário deve poder agendar eventos utilizando a própria voz e se informar sobre os eventos inseridos também utilizando a voz.

O usuário pode interagir com a **Sunday** na agenda usando a voz. Quando o usuário diz, por exemplo: "Evento para o dia 18 de Setembro até o dia 20 de Setembro", a fala é direcionada para o servidor, que separa os dias mencionados. O primeiro dia torna-se o inicial e o segundo dia é considerado o final do evento. O servidor envia esses dias para o banco de dados e armazena a fala como a descrição completa do evento.

## 5.5 Jogando com a Sunday

Esta é a interface de entretenimento com a **Sunday**. Nela, é possível realizar uma pequena interação divertida jogando o jogo da velha.

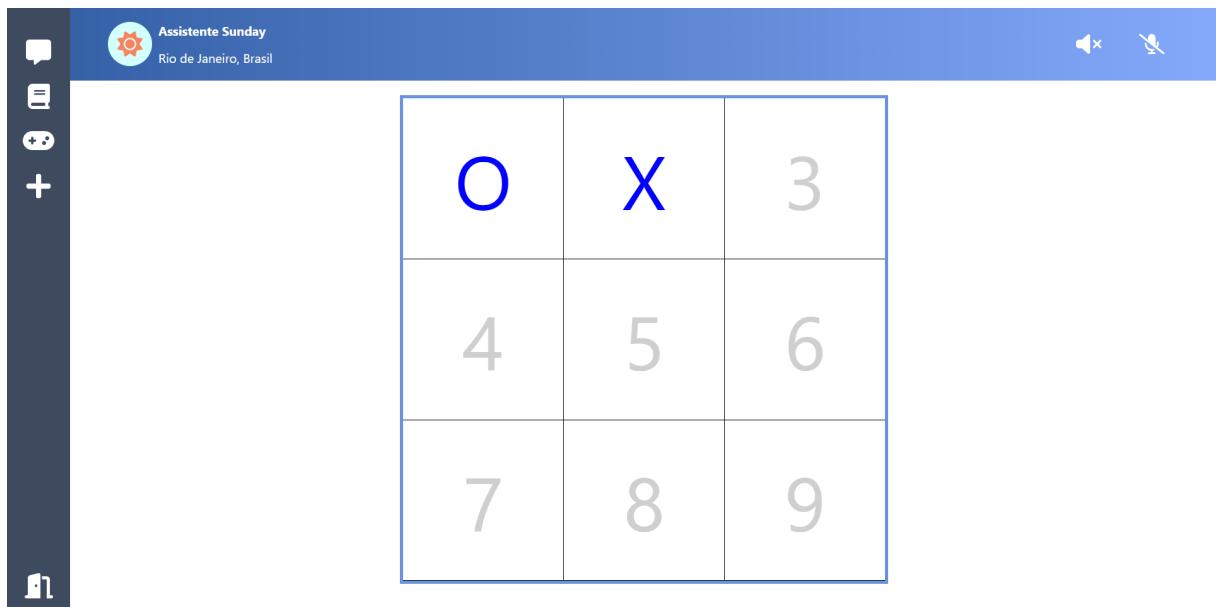


Figura 5.5: Tela do Jogo da Velha (fonte: Autor)

A Figura 5.5 ilustra a interface do jogo da velha, atendendo aos Requisitos Funcionais RF07 e RF09, nos quais o usuário pode optar por jogar com a **Sunday**, interagindo por voz ou clicando na casa desejada.

Após o usuário escolher uma casa para jogar com a **Sunday**, uma requisição é enviada para o backend, informando a situação das casas escolhidas. O servidor coleta a lista e seleciona apenas as casas vazias. Após coletar essas casas, é feita uma seleção randômica da próxima casa que será preenchida pela **Sunday**. Ao escolher a casa, a requisição retorna para o *frontend* com a informação sobre qual casa foi escolhida, e isso é mostrado ao usuário.

## 5.6 Intuitividade da Sunday

Os avisos que o usuário recebe aparecem no canto superior direito da interface, como ilustrado na Figura 5.6.

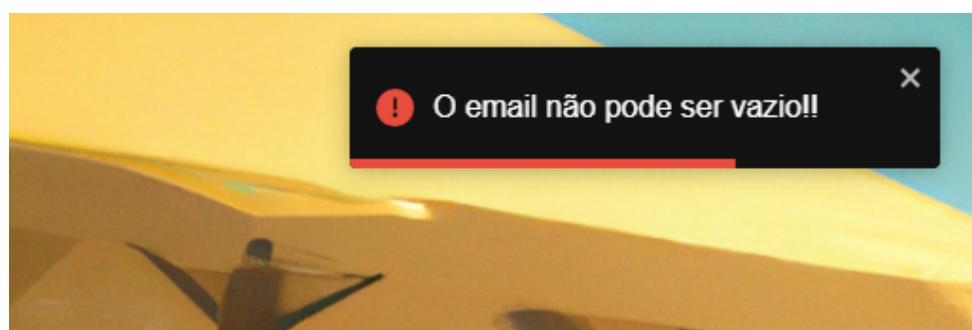


Figura 5.6: Como aparecem os aviso (fonte: Autor)

Estes avisos atendem ao requisito não funcional RNF01, que exige que o sistema seja intuitivo. Assim, quando o usuário precisar de informações sobre algo que está faltando em um formulário, por exemplo, esse aviso aparece na interface e o informa.

### 5.6.1 Modais do Sistema

Os modais são pequenos avisos nas interfaces de cada parte da aplicação, explicando como é o funcionamento e orientando o usuário sobre como proceder para a interação.

Cada interface do sistema possui um modal responsável pela explicação do funcionamento. A Figura 5.7 ilustra um exemplo desses modais, atendendo ao requisito não funcional RNF01.

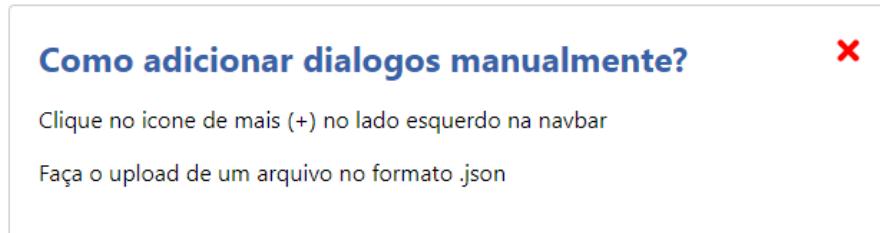


Figura 5.7: Exemplo Modal (fonte: Autor)

## 6 *Conclusão e trabalhos futuros*

Neste trabalho de conclusão de curso, exploramos de maneira abrangente o desenvolvimento de uma assistente virtual multifuncional denominada **Sunday**, representando uma solução moderna e versátil para a interação entre humanos e máquinas. Ao longo deste TCC, destacamos o uso de tecnologias como JavaScript, Node.js e React. A **Sunday** consegue compreender a voz do usuário, um diferencial decorrente de sua arquitetura focada em uma única linguagem.

**Sunday** abrange um chat, uma agenda de eventos e um jogo da velha, consolidando-se como uma ferramenta que abarca diversos campos em uma única aplicação. Essas funcionalidades, aliadas à capacidade de comunicação por voz, proporcionam versatilidade a esta assistente virtual.

Exploramos também os desafios e etapas envolvidos no desenvolvimento da **Sunday**, desde o design da aplicação até a integração de reconhecimento de voz e processamento. Discutimos a importância da experiência do usuário e da usabilidade como fatores cruciais para o sucesso de uma assistente virtual.

Adicionalmente, ressaltamos a necessidade contínua de pesquisa e desenvolvimento nesse campo, comparando a **Sunday** com outras assistentes como a Siri e o Google Assistant. Apesar das limitações de recursos, conseguimos desenvolver uma assistente útil e usável, que pode ser facilmente aprimorada sem a necessidade de recorrer a outras linguagens de programação. **Sunday** pode servir de exemplo para futuros trabalhos e contribuir para a reflexão sobre a simplificação no desenvolvimento de aplicações e softwares, mostrando que o caminho mais simples não é necessariamente errado. A possível evolução futura da **Sunday** é viável, impulsionada por essa abordagem simplificada.

Concluindo, este TCC oferece uma visão geral do processo de desenvolvimento de uma assistente virtual, destacando desafios e conquistas, e ressaltando a viabilidade de criar uma assistente virtual completa utilizando apenas JavaScript, do *Frontend* ao *Backend*. O futuro das assistentes virtuais promete, com a possibilidade de interações completas apenas por voz, inte-

gradas a ambientes de casas inteligentes, entre outras aplicações, indicando um futuro promissor para essa tecnologia.

## *Referências*

AKINBI, T. B. A. Forensic investigation of google assistant. **Revista da Escola de Administração Pública do Amapá**, v. 10, n. 4, p. 272–282, 2020.

APPLE. **SiriKit**. 2023. Disponível em: <<https://developer.apple.com/documentation/sirikit>>. Acesso em: 10 Set. 2023.

COLETIVA, A. **NLP.js**. 2020. Disponível em: <<https://github.com/axa-group/nlp.js/blob/master/docs/v3/README.md>>. Acesso em: 05 Oct. 2023.

COLETIVA, A. **react-speech-kit**. 2020. Disponível em: <<https://www.npmjs.com/~bigmike>>. Acesso em: 05 Oct. 2023.

DOCKER. **O que é o Docker?** 2023. Disponível em: <<https://www.docker.com/what-docker>>. Acesso em: 14 Set. 2023.

GOURLEY, B. T. D. **HTTP the definitive guide**. [S.l.]: O'Reilly Media, Inc., 2002.

GUEDES, G. T. A. **UML 2 uma abordagem prática**. [S.l.]: Novatec Editora, 2020.

KENTON, W. **What Is a Virtual Assistant, and What Does One Do?** 2023. Disponível em: <<https://www.investopedia.com/terms/v/virtual-assistant.asp>>. Acesso em: 13 Dec. 2023.

KRIGER, D. **O QUE SÃO FRAMEWORKS E POR QUE SÃO IMPORTANTES PARA OS DEVS**. 2022. Disponível em: <<https://kenzie.com.br/blog/framework/>>. Acesso em: 10 Dec. 2023.

LIDDY, E. D. **Encyclopedia of Library and Information Science**. [S.l.]: Marcel Decker, Inc., 2001.

LUCIDCHART. **What is a Data Flow Diagram?** 2023. Disponível em: <<https://www.lucidchart.com/pages/data-flow-diagram>>. Acesso em: 16 Set. 2023.

MOZILLA. **Expressões Regulares**. 2023. Disponível em: <[https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Regular\\_expressions](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Regular_expressions)>. Acesso em: 10 Dec. 2023.

MOZILLA.ORG. **CSS**. 2023. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/css>>. Acesso em: 22 Set. 2023.

MOZILLA.ORG. **JavaScript**. 2023. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>>. Acesso em: 22 Set. 2023.

NODEJS.ORG. **NodeJs**. 2023. Disponível em: <<https://nodejs.org/en/about>>. Acesso em: 22 Set. 2023.

- OLIVEIRA, S. S. de. Bancos de dados não-relacionais: Um novo paradigma para armazenamento de dados em sistemas de ensino colaborativo. ano de publicação. **Revista da Escola de Administração Pública do Amapá**, v. 10, n. 4, p. 184–194, 2014.
- ORACLE. **Orarcle**. 2023. Disponível em: <<https://www.oracle.com/br/database/what-is-json/>>. Acesso em: 12 Set. 2023.
- ORACLE.COM. **O que é um Banco de Dados?** 2023. Disponível em: <<https://www.oracle.com/br/database/what-is-database/>>. Acesso em: 10 Set. 2023.
- REACT.DEV. **React**. 2023. Disponível em: <<https://react.dev/>>. Acesso em: 22 Set. 2023.
- SACRAMENTO, G. **Aplicação web: o que é, diferença para website, como funciona e mais!** 2022. Disponível em: <<https://rockcontent.com/br/talent-blog/aplicacao-web/>>. Acesso em: 10 Set. 2023.
- SOFASTAEI, A. **Virtual Assistant**. [S.l.]: IntechOpen0, 2021.
- WILDE, C. P. E. **REST: from research to practice**. [S.l.]: Springer Science Business Media, 2011.