**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA**

**ESCOLA TÉCNICA ESTADUAL DA ZONA LESTE**

**Mtec Desenvolvimento de Sistemas AMS**

**Emily Cristina dos Santos Primo**

**João Pedro Santana Mota**

**Rodrigo da Silva Lima**

# M.E.R.LIN: Sistema de Assistência à Acessibilidade para Pessoas com Deficiência Motora no Uso de Computadores.

**São Paulo**

**2025**

**Emily Cristina dos Santos Primo**

**João Pedro Santana Mota**

**Rodrigo da Silva Lima**

# M.E.R.LIN: Sistema de Assistência à Acessibilidade para Pessoas com Deficiência Motora no Uso de Computadores.

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Desenvolvimento de Sistemas da ETEC da Zona Leste, orientado pelo Prof. Esp. Jeferson Roberto de Lima, como requisito parcial para obtenção do título de técnico em Desenvolvimento de Sistemas.

**São Paulo**

**2025**

# Dedicatória

Dedicamos este trabalho a todas as turmas que vieram antes de nós, seus esforços individuais, orientações, exemplos e conversas despretensiosas foram a chave para alcançarmos os resultados deste projeto. A todos os que nos motivaram, inspiraram e apoiaram de todas as formas, voluntariamente ou não, saibam que somos muito gratos a vocês. Deixamos nessas palavras nossa mais sincera gratidão.

# Agradecimentos

Para a realização deste Trabalho de Conclusão de Curso (TCC), foi necessária uma boa quantidade de esforço, cooperação e paixão de cada membro do grupo, além é claro, do apoio de muitos. Apoio esse que jamais deixaríamos de reconhecer.

Queridos amigos, Felipe Vieira e Giovanna Torres, saibam que vocês fizeram uma enorme diferença para este projeto. Agradecemos a vocês pelas conversas, conselhos, por sua paciência e ajuda ao longo do ano.

Admirável mentora, Camila França, agradecemos imensamente por todo o seu encorajamento e pelo apoio à nossas ideias fora da caixa. Graças as suas palavras de incentivo e atenção aos nossos detalhes, o projeto conseguiu ser verdadeiramente autentico.

Amados pais, agradecemos a vocês pelas mais diversas orações que fizeram em prol do nosso trabalho, por se esforçarem continuamente para que nós pudéssemos nos dedicar a aquilo que gostamos, por seu genuíno carinho conosco e por toda motivação que nos concederam.

Memoráveis professores, esses que muitas vezes atravessaram a barreira aluno-professor e demonstraram orgulho, admiração, empolgação e solidariedade com aquilo que fizemos, saibam que jamais esqueceremos de suas palavras e reações, Andreza Rocha, Carlos Alberto, Marlon Marques e Salomão Santana, a vocês nossos mais sinceros agradecimentos.

“Só porque tem que ter os pés fincados no chão
não quer dizer que não possa alcançar o céu. “

**Bárbara Millicent Roberts**

## Resumo

O corrente estudo contém a documentação escrita de um software que auxilia PcDs no uso de computadores, além de um site expositivo para o mesmo. O objetivo é desenvolver um software assistivo que ofereça interfaces controláveis pelos movimentos faciais, voltado para pessoas com deficiências motoras, especificamente aquelas que não afetam os músculos mímicos, de modo a possibilitar que os indivíduos portadores dessas deficiências sejam devidamente inseridos no mercado de trabalho.

A pesquisa realizada utiliza o estudo de requisitos, a estruturação de diagramas e o desenvolvimento de interfaces gráficas, aderindo ao método quantitativo, pois este permite verificar dados e resultados obtidos por meio de testes e observações das pessoas com deficiência motora no mercado de trabalho. Por meio dessa abordagem, esperava-se analisar informações que possibilitariam a medição do nível de acessibilidade e integração social de pessoas com deficiência motora ao mercado de trabalho nacional. Como resultado, espera-se que o software promova a inserção e a autonomia profissional do público-alvo, possibilitando uma melhora na qualidade de vida.

**Palavras-Chave:** PcDs, Deficiências Motoras, Músculos Mímicos, Software Assistivo, Mercado de Trabalho.

## Abstract

The project contains written documentation for software that assists people with disabilities in using computers and desktops, as well as a website for its presentation. The goal is to develop software that offers interfaces controllable by facial movements, aimed at people with motor disabilities—specifically those that do not affect the mimic muscles (such as ataxia, myopathies, amputations, and Ehlers-Danlos Syndrome). This software will enable them to successfully enter the job market.

The research uses requirements analysis, diagram structuring, and graphical interface development, adopting a qualitative method, as it allows for an understanding of the experiences, perceptions, and difficulties faced by people with motor disabilities in the job market. This approach enabled the analysis of reports, observations, and opinions, guiding the development of the software according to their real needs and accessibility expectations.

As a result, the software is expected to promote the professional integration and autonomy of the target audience, enabling an improvement in their quality of life.

**Keywords:** PwDs, Motor Disabilities, Mimic Muscles, Assistive Software, Job Market.

# LISTA DE ILUSTRAÇÕES

# LISTA DE ABREVIAÇÕES E SIGLAS

*Cascading Style Sheets* (CSS)

Guias de Interface de Usuário (GUIs).

*HyperText Markup Language* (HTML).

*Open Source Computer Vision Library* (OpenCV).

*Package Installer for Python* (PIP).

*Tool Command Language* (TCL).

*Tool Kit* (TK).

*Tool Kit Interface* (TKinter).

*Unified Modeling Language* (UML).

*User Experience* (UX).

*User Interface* (UI).

# SUMÁRIO

# 1 INTRODUÇÃO

O M.E.R.LIN consiste em um software capaz de devolver a independência do uso de aparelhos Desktop para portadores de deficiência física, baseado no conceito de emprego apoiado, com o objetivo de desenvolver um software assistivo que interage diretamente com a câmera e o sistema operacional do computador, viabilizando a execução de funções por meio de uma interface visual, com uma ampla gama de ações a serem realizadas através do movimento ocular.

Sendo capaz de utilizar o movimento dos olhos para realizar funções essenciais no uso de um computador, de forma dinâmica e que facilite a autossuficiência do usuário, tornando-o apto a exercer seus direitos na sociedade, como o direito ao trabalho e à privacidade.

Realizando uma avaliação dos impactos que são esperados do M.E.R.LIN no cotidiano das pessoas com deficiência motora, será possível analisar as demandas do público alvo e elaborar o software mediante tais particularidades.

Sendo estas decorrentes da falta do uso da tecnologia assistiva em ambientes de trabalho, cujos princípios não atendam a condições dignas para aqueles que apresentam características motoras singulares realizarem seu trabalho de maneira adequada, impedindo a facilitação, inclusão e reabilitação de pessoas com deficiência física da forma devida.

As empresas, em sua grande maioria, não estão preocupadas com o aumento de empregabilidade desse grupo da sociedade, a maior preocupação é cumprir a porcentagem regida por lei, no que se refere à contratação de PcDs para isenção de taxas de impostos.

As pessoas com deficiência motora têm oportunidades de emprego muito limitadas decorrente do excesso de capacidades físicas necessárias para executar determinados trabalhos.

Assim como os locais de trabalho não são devidamente adaptados para os portadores de alguma deficiência motora, seja por escolha ou resultado das impressões sociais.

Em suma, o sistema de assistência à acessibilidade demonstra potencial para tornar a participação de pessoas com disfunções motoras mais ativa no âmbito do trabalho,

e fornece ferramentas para uma inclusão nesses espaços, no que diz respeito à eficiência em tarefas, e à vivência social concedida pela oportunidade de emprego.

Como é referido por Moreira et al. (2015 apud RODRIGUES; PEREIRA, 2021, p.9), “É comum a sociedade e, em muitos casos, a própria família, reforçar o estigma de que a pessoa com deficiência é incapaz de realizar atividades diárias ou de trabalho de forma independente e autônoma”. Analisando esse paradigma da sociedade, o projeto busca desenvolver uma aplicação que enfrente tais preceitos, entregando uma solução inclusiva e acessível para os portadores de deficiências motoras. A aplicação se fundamenta no conceito de emprego apoiado, visando se tornar um modelo de sistema especializado no auxílio para pessoas com alguma disfunção motora. “Pessoas com deficiência, muitas vezes consideradas não aptas para o trabalho, poderiam exercer atividades de trabalho se lhes fosse proporcionado o apoio necessário” de Sousa (2000, apud RODRIGUES; PEREIRA, 2021, p.15). Considerando isso, o projeto visa ser apto a ter o reconhecimento como um assistente no mercado de trabalho, promovendo interações sociais entre essa parcela segmentada da sociedade.

Como foi apresentado por Lima (2013, apud RODRIGUES; PEREIRA, 2021), às pessoas portadoras de alguma deficiência vivenciam as experiências sociais com mais densidade que os outros, decorrente da sensação de pertencimento social e reconhecimento do seu trabalho.

O projeto se apodera das definições metodológicas descritas por Lakatos e Marconi (2017), para embasar os procedimentos utilizados em seus estudos de resultados, aplicações e impactos. Sendo estes caracterizados como estudos aplicados, buscando entregar um protótipo sólido para a resolução do problema de acessibilidade digital para pessoas com necessidades motoras específicas. A ênfase do tratamento das pesquisas é majoritariamente quantitativa, atentando-se aos dados relacionadas ao contato do usuário.

Os seguintes capítulos desta monografia irão abordar os princípios por trás do embasamento teórico e técnico desse assistente, ademais as etapas de seu desenvolvimento prático. Dentre os conceitos técnicos fundamentais podem ser apontados como alguns deles o Python (Banin, 2018), o OpenCV (Castro, 2021), códigos do tipo Shell (Negus, 2014) e a UML 2 (Guedes, 2018).

# 2 REFERENCIAL TEÓRICO

Para a clara compreensão dos conceitos por trás da promoção do M.E.R.LIN, o seguinte capítulo, é responsável por abordar a base teórica e conceitual utilizada em seu desenvolvimento.

## 2.1 Deficiência Motora no Brasil

A deficiência motora representa uma condição que afeta significativamente a qualidade de vida e a participação social de milhões de brasileiros. Segundo dados do Instituto Brasileiro de Geografia e Estatística (IBGE, 2023), aproximadamente 8,9% da população nacional convive com algum tipo de deficiência, o que corresponde a cerca de 18,6 milhões de pessoas. Este cenário revela a necessidade de políticas públicas e ações afirmativas voltadas para a promoção da inclusão e da acessibilidade. As limitações físicas inerentes à deficiência motora frequentemente comprometem a realização de atividades cotidianas e profissionais, evidenciando a importância do desenvolvimento e da aplicação de tecnologias assistivas que promovam a autonomia, a autossuficiência e a dignidade desses indivíduos no contexto social e econômico.

### 2.1.1 Desafios no Cenário Trabalhista

No âmbito profissional, as pessoas com deficiência (PCDs), especialmente aquelas com deficiência motora, enfrentam barreiras estruturais, culturais e atitudinais que dificultam sua plena inserção e permanência no mercado de trabalho. Dados divulgados pelo G1 (2022) indicam que aproximadamente 70% dessa população encontra-se em situação de desemprego, e aqueles que conseguem inserção laboral percebem, em média, um salário R$1.000 inferior ao dos demais trabalhadores. Além disso, embora a participação dos PCDs represente 28,3% do mercado de trabalho, essa estatística não reflete necessariamente uma inclusão efetiva, uma vez que a falta de acessibilidade e de adequações no ambiente laboral compromete o desempenho de suas funções. Conforme ressaltado por Rodrigues et al. (2021), a mera disponibilização de vagas não assegura a efetiva inclusão dessas pessoas, sendo imprescindível a criação de ambientes acessíveis e inclusivos que respeitem as especificidades de cada indivíduo, assegurando, assim, a igualdade de oportunidades e a promoção da diversidade no mundo do trabalho.

## 2.2 Tecnologias

Para que fosse possível o desenvolvimento do M.E.R.LIN, uma série de bibliotecas, ferramentas e linguagens foram utilizadas.

### 2.2.1 Python

Para a realização da solução proposta neste estudo, foi extensivamente utilizado do Python como matriz de toda sua criação.

Criado pelo programador Guido van Rossum, durante os anos 90, para fins pessoais, a linguagem de programação chamada Python apresenta exclusividades em sua criação, conforme aponta Banin (2018).

Dentre essas exclusividades, Matthes (2016) cita a natureza escalonável, sucinta e legível do Python. Tornando-a flexível para os desenvolvedores.

Parte dessa flexibilidade do Python, segundo Sweigart (2015), é resultante da quantidade massiva de recursos externos adicionais que estão disponíveis para a construção de programas nesta linguagem.

Figura 1 - Exemplo de Código em Python

```python
def aoAcertar():
    Valor1 = 2
    Valor2 = 4
    Conta = Valor1 * Valor2
    Pergunta = int(input("Qual o valor de 2 x 4? "))

    if Pergunta == Conta:
        print("Você acertou!")

    else:
        print("Puts, resposta errada")

aoAcertar()
```

Fonte: Autoria Própria, 2025.

Na imagem, temos um exemplo de uma função simples em Python, onde com base em um valor pré-estabelecido, uma condição é aplicada e uma pergunta é feita para o usuário. Abaixo, é possível encontrar mais detalhes sobre o código.

Linha 1: Nessa linha é criado um pacote de comandos (comumente chamado de função) intitulado de "AoAcertar". Ele reúne toda a lógica que compõem o objetivo desse código.

Linha 2 e 3: Nessas duas linhas são criados registros na memória (variáveis), para guardar os valores numéricos utilizados no código.

Linha 4: Esses registros são passados para esta linha, onde é feito uma operação matemática com os valores anteriormente definidos.

Linha 5: Um novo tipo de registro é criado nessa linha, ele é responsável por receber a resposta do usuário para a pergunta: "qual o valor de 2 X 4?", que será feita ao executar o código.

Linha 7:  Aqui é criado um requisito para a sequência do código (esses requisitos são chamados de condição), determinando que caso a resposta do usuário seja igual ao resultado correto da operação, uma mensagem positiva será exibida para ele.

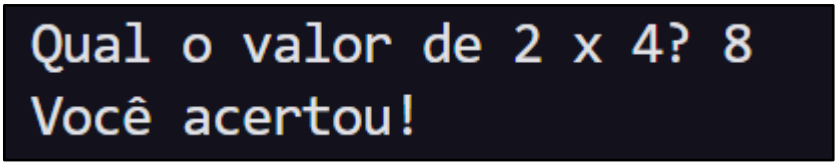
Linha 8: Esse comando chamado "print" é responsável por exibir mensagens para o usuário, no caso dessa linha, a mensagem é positiva.

Linha 10:  Nessa linha é criada outra condição, determinando que caso a resposta do usuário seja diferente da fornecida na primeira condição, uma mensagem negativa será exibida para ele.

Linha 11: O “print” é utilizado mais uma vez para exibir essa mensagem negativa.

Linha 13: Na última linha do programa está o comando utilizado para acionar a função criada nas linhas de código anteriores.

Figura 2 - Resultado do Código em Python



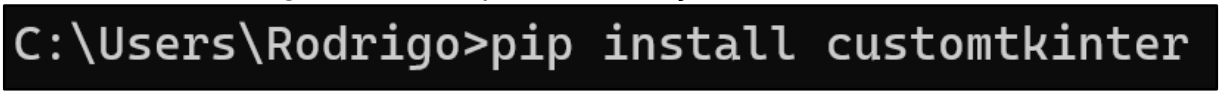

Fonte: Autoria Própria, 2025.

Na imagem, está sendo apresentado o resultado após a execução do programa. A pergunta da condição foi realizada, ao final da linha a resposta foi introduzida, o programa fez a comparação definida e apresentou a mensagem de sucesso.

Para ter o Python na sua máquina, é necessário o uso do PIP. Como demonstrado na documentação oficial de empacotamento Python (2025), o PIP, responsável pela instalação de dependências e recursos do Python, é a ferramenta mais popular para a implantação dos pacotes da linguagem.

Abaixo, o PIP é utilizado para a instalação da biblioteca de desenvolvimento visual CustomTkinter.

Figura 3 - Exemplo de Instalação Por Meio do PIP




Fonte: Autoria Própria, 2025.

### 2.2.2 Bibliotecas do Python

Ao longo deste trabalho, algumas bibliotecas foram usadas para implementar, automatizar e facilitar o uso do Python.

O Subprocess de acordo com os estudos de Figueiredo (2023), representa uma biblioteca desenvolvida em Python, possui recursos utilizaveis para tornar um código nessa linguagem, capaz de administrar, por meio do *Shell*, operações originalmente externas a si.

Figura 4 - Exemplo de Código Utilizando Subprocess

```python
import subprocess
import random

Programas = [
    r"notepad.exe",  # Bloco de notas
    r"calc.exe",     # Calculadora
    r"mspaint.exe"   # Paint
]

AppSorteado = random.choice(Programas)
print(f"Abrindo: {AppSorteado}")
subprocess.Popen(AppSorteado)
```

Fonte: Autoria Própria, 2025.

Linha 1:  Import da biblioteca do subprocess.

Linha 2: Import da biblioteca Random, que serve para gerar ocorrências aleatórias.

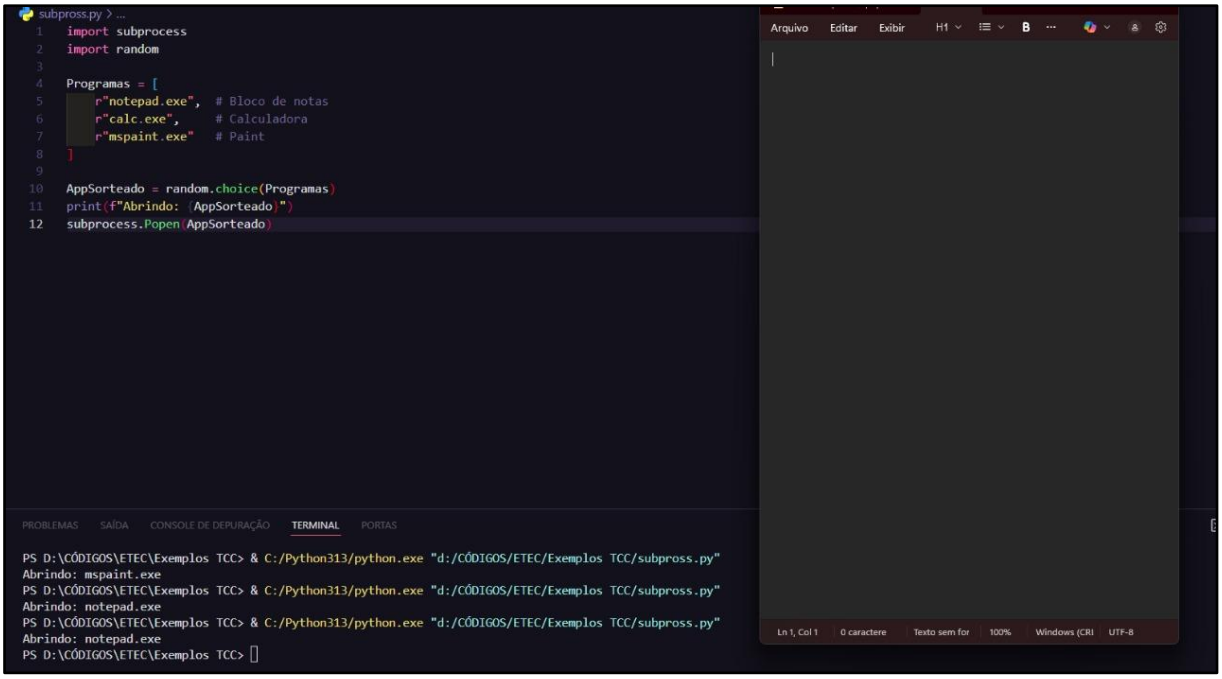
Linha 4 - 8: Uma lista do python que guarda programas do computador (ta especificado quem é quem).

Linha 10: uma função do random é utilizada para escolher aleatoriamente um programa dentro da lista, essa escolha é passada para uma variável chamada "AppSorteado".

Linha 11: Comando print para exibir uma mensagem dizendo qual programa foi escolhido.

Linha 12: Comando do subprocess para iniciar programas do computador.

Figura 5 - Resultado do Código Utilizando Subprocess




Fonte: Autoria Própria, 2025.

Sweigart (2015), ressalta que a biblioteca PyAutoGui é capaz de executar movimentos com o mouse ou o teclado a uma velocidade incrível, e de forma autônoma. Além disso, essa ferramenta possui algumas precauções de segurança ativas por padrão.

A seguir, será demonstrado um exemplo de programa utilizando a biblioteca.

Figura 6 - Exemplo de Código Utilizando PyAutoGui

```python
import pyautogui
from time import sleep

pyautogui.click(1025, 1055, duration=0.1)
pyautogui.click(378,184, duration=2)

with open(r'D:\CÓDIGOS\Automato\msgs.txt', 'r', encoding='utf-8') as arquivo:
    for linha in arquivo:
        mensagem = linha.strip()
        if mensagem:
            pyautogui.click(892, 1006, duration=1)
            pyautogui.write(mensagem)
            pyautogui.press('enter')
            sleep(1)
```

Fonte: Autoria Própria, 2025.

Linha 1 e 2: São importadas as bibliotecas pyautogui e time do python, respectivamente.

Linha 4 e 5: É puxado uma propriedade chamada “click” da biblioteca sendo utilizada, essa propriedade fará com que um clique seja feito nas coordenadas definidas entre parênteses.

Linha 7: O comando padrão “with open” faz com que o programa possa acessar algum arquivo específico, no caso está sendo aberto um arquivo de texto chamado “msgs”.

Linha 8: Essa linha inicia uma tarefa em loop, será percorrido cada linha do arquivo de texto e então armazenado dentro de “linha”.

Linha 9: Nessa linha, a funcionalidade “strip” remove espaços em branco e quebras de linha no início e no fim de cada linha do arquivo texto. A variável agora contém o texto limpo da linha percorrida.

Linha 10: É criada uma condição que verifica se a variável mensagem está vazia. Caso contenha algum texto, o grupo indicado a seguir será executado.

Linha 12: Este novo comando apresentado utiliza a biblioteca para que um texto seja digitado automaticamente, no caso o texto são as informações contidas na variável “mensagem”.

Linha 13: Nessa linha é utilizado um comando da biblioteca que utiliza teclas do teclado de forma autônoma, no caso é iniciado a tecla “enter”.

Linha 14: Pausa a execução de um programa por um determinado número de segundos.

#### 2.2.2.1 Bibliotecas Gráficas

As bibliotecas aqui descritas foram implementadas com a finalidade de potencializar a interatividade do sistema originalmente proposto.

O TKinter é uma biblioteca para criação de Interfaces Gráficas do Usuário (GUIs) que possui a função de criar elementos visuais, comumente usados em aplicações da linguagem Python onde são requeridos usos constantes de interação com o usuário do software, como apontado por Esperança (2011).

Assim, é possível obter uma conexão da interface gráfica entre o usuário e a aplicação, que por sua vez pode criar janelas com elementos para comandar ações,

parâmetros, desenhar gráficos e exibi-los, conforme abordado por Camargo (2020) em seus estudos.

Com base à documentação oficial do Python (2025), o pacote Tkinter faz parte de um kit de ferramentas Tcl (Tool command Language) e TK (Toolkit), que possui compatibilidade entre múltiplos sistemas operacionais.

Figura 7 - Exemplo de Código Utilizando Tkinter

```python
import tkinter as tk

Tela = tk.Tk()
Tela.title("Exemplo de Interface em Tkinter")
Tela.geometry("500x400")

Cliques = 0

def aoClicar():
    global Cliques
    Cliques += 1
    Texto.config(text=f"Você Clicou no Botão {Cliques} Vezes")

Texto = tk.Label(
    master=Tela,
    text="Você Clicou no Botão 0 Vezes",
    fg="pink"
)

Botao = tk.Button(
    master=Tela,
    text="Clique Aqui",
    bg="purple",
    fg="white",
    activebackground="magenta",
    command=aoClicar
)

Texto.place(relx=0.5,
            rely=0.5,
            anchor=tk.CENTER)

Botao.place(relx=0.5,
            rely=0.6,
            anchor=tk.CENTER)

Tela.mainloop()
```

Fonte: Autoria Própria, 2025.

No exemplo, temos uma tela minimalista de um botão com clicker utilizando a padronização de interface oferecido pelo Tkinter.

Linha 1: Importação da biblioteca do tkinter, com a chamada do objeto padrão tk.

Linha 3: Criação de um objeto chamado “Tela” que utiliza da importação “tk” com uma propriedade padrão específica “TK”, que inicia o objeto como uma interface gráfica TKinter.

Linha 4: Utilização do objeto criado mais a propriedade “tittle” que dará um nome a janela “Tela” de interface do usuário.

Linha 5: Atribui ao Objeto “Tela” o parâmetro “geometry”, que redimensiona o tamanho da tela para o tamanho desejado de 500 pixels por 400 pixels.

Linha 7: Criação do objeto “Cliques” que recebe o valor inicial 0.

Linha 9: Declaração de uma função com o nome de “aoClicar” que agrupa as três linhas seguintes.

Linha 10: Definição do objeto “Cliques” como global, assim todo o código da página criada irá reconhecê-la.

Linha 11: O objeto “Cliques” está sendo chamado para uma nova declaração de valor em que o seu valor inicial de zero receberá mais 1.

Linha 14: Novo objeto “Texto” que recebe a propriedade padrão juntamente com outra específica “Label” para criação de textos.

Linha 15: O elemento “Master” faz a relação de hierarquia do objeto mestre que é a “Tela”, assim o botão obrigatoriamente será exibido na janela “Tela”.

Linha 16: A propriedade “text”, adiciona um texto embutido por meio de aspas dentro de parênteses.

Linha 17: Recebimento de uma cor dominante com o nome “Pink”.

Linha 20: Definição de um Objeto “Botao” que recebe um método “tk” e um elemento “Button” que fará o objeto obter a formatação padrão do tk de um botão.

Linha 24: Define uma cor para a fonte do texto exibido dentro do botão, “white” (Branco).

Linha 25: Define que quando o objeto “Botao” for pressionado, ele ficará com a cor “magenta”.

Linha 26: Define que quando o botão for pressionado, a função “aoClicar” será ativada.

Linha 29: Atribui uma posição específica a variável “Texto”, utilizando a propriedade “place”, “relx” define o eixo x nas especificações da tela.

Linha 30: “rely” define o eixo y de acordo com as especificações da tela.

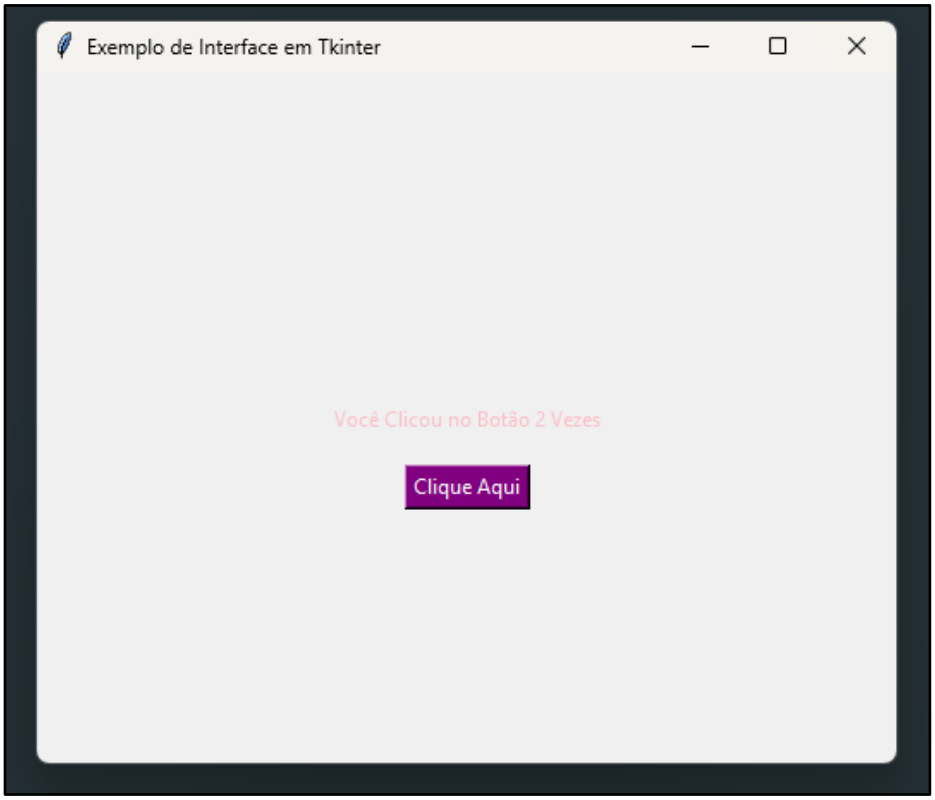
Linha 31: Define que o texto ficará centralizado seguindo o padrão dado pelos eixos atribuídos.

Linha 37: Permite que a tela fique em constante exibição.

Linha 35: define que o texto ficará centralizado seguindo o padrão dado pelos eixos atribuídos.

Linha 37: Permite que a tela fique em constante exibição.

Figura 8 - Resultado do Código com Tkinter




Fonte: Autoria Própria, 2025.

Para aprimorar as ferramentas disponibilizadas pelo TKinter, será utilizada uma biblioteca que complementa suas capacidades visuais, o CustomTKinter.

Segundo Frota, Ruver e Figueiredo (2024), o CustomTkinter é uma biblioteca que permite a criação de *Interfaces Gráficas do Usuário* (GUIs) sendo utilizado por possuir uma estética mais atualizada, moderna e personalizável sem complexidade excessiva.

Figura 9 - Exemplo de Código Utilizando CustomTKinter

```python
from customtkinter import *

Tela = CTk()

Tela.title ("Exemplo de Interface em CustomTk")

Tela.geometry ("500x400")

Cliques = 0

def aoClicar():
   global Cliques
   Cliques += 1

   Texto.configure(text = f"Você Clicou no Botão {Cliques} Vezes ")

Texto = CTkLabel(master = Tela,
                 text = "Você Clicou no Botão 0 Vezes",
                 text_color = "pink")

Botao = CTkButton(master = Tela,
                  text = "Clique Aqui",
                    corner_radius = 30,
                      fg_color = "purple",
                        text_color = "white",
                          hover_color = "magenta",
                          command = aoClicar)

Texto.place(relx = 0.5,
            rely = 0.5,
            anchor = CENTER)

Botao.place(relx = 0.5,
            rely = 0.6,
            anchor = CENTER)

Tela.mainloop()
```

Fonte: Autoria Própria, 2025.

Código anterior atualizado para comportar o CustomTkinter em sua composição

Linha 1: importação de todos os elementos da biblioteca do CustomTKinter.

Linha 3: Criação de um objeto chamado “Tela” que utiliza da importação “customtkinter”, o elemento “CTk”, que define o objeto declarado como interface gráfica.

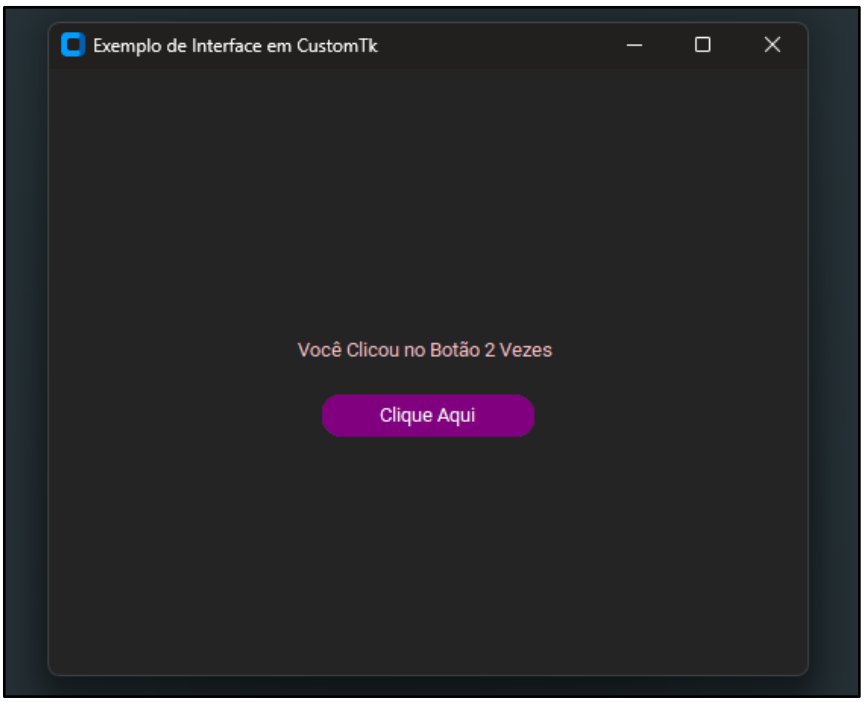
Linha 15: O objeto “Texto” recebe a propriedade padrão juntamente com outra específica “configure” para configurar o texto que exibirá após a ação de “Clique”.

Linha 17: Cria um rótulo “CTkLabel” dentro da janela Tela, que será usado para exibir o texto.

Linha 21: Definição de um Objeto “Botao” que recebe a propriedade “CTkButton”, assim o elemento se comporta como botão.

Linha 23: Define um valor para arredondamento do botão para fins estéticos.

Figura 10 - Resultado do Código com CustomTKinter




Fonte: Autoria Própria, 2025.

### 2.2.3 Reconhecimento Ocular

Este projeto fundamentalmente aplica os conseguintes conceitos para a coesa realização de sua proposta inicial.

Tal como observa Barelli (2018), a visão computacional vem como grande ajuda para procurar, detectar e tratar informações, analisando-as de forma similar à visão humana.

Assim como afirmam Backes e Sá (2016), a visão computacional pode ser utilizada em aplicações corriqueiras e não particularmente voltadas para o cenário corporativo, a criação de filtros para fotos e o tratamento de imagens é um exemplo disso.

Por meio do OpenCV, o projeto será capaz de aplicar os conceitos de visão computacional que acabam de ser explicados.

Como afirma Castro (2021), o OpenCV é uma biblioteca voltada ao desenvolvimento de aplicações de visão computacional, composta por ferramentas que atendem às mais diversas finalidades nessa área.

Marengoni (2010) ressalta que a biblioteca criada originalmente pela Intel, foi pensada com o objetivo de tornar mais acessível a interação em tempo real entre homem e máquina.

Assim como defende Barelli (2018), o conjunto de instrumentos disponíveis na biblioteca mostra-se altamente eficaz para a manipulação de conteúdos visuais.

Figura 11 - Exemplo de Instalação do OpenCV Por Meio do PIP


Fonte: Autoria Própria, 2025.

Figura 12 - Exemplo de Código Utilizando OpenCV.

```python
import cv2
import cv2.data

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarscascade_frontalface_default.xml')

imagem = cv2.imread('grupoctc.jpg')
cinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)

rosto = face_cascade.detectMultiScale(cinza, scaleFactor = 1.1, minNeighbors = 5)

for (x, y, w, h) in rosto:
    cv2.rectangle(imagem, (x , y), (x + w, y + h), (0, 255, 0), 2)

    print(f"Total de Pessoas Detectadas: {len(rosto)}")

    cv2.imshow("Pessoas Detectadas", imagem)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Fonte: Autoria Própria, 2025.

Acima, temos um exemplo simples da aplicação da biblioteca OpenCV para detecção de rostos, demarcação de retângulos e a exibição de uma mensagem referente a quantidade de rostos presentes.

Linha 1 e 2: Esses são os comandos necessários para a utilização das ferramentas do OpenCV no código.

Linha 4: Essa linha carrega a ferramenta pré-pronta de identificação de rostos do OpenCV.

Linha 6: A variável “imagem” é responsável por guardar qual imagem irá ser usada e o que será feito com ela, no caso, ela será analisada.

Linha 7: Para facilitar a interpretação da imagem pelo programa, ela recebe um filtro de cores cinzas próprios do OpenCV.

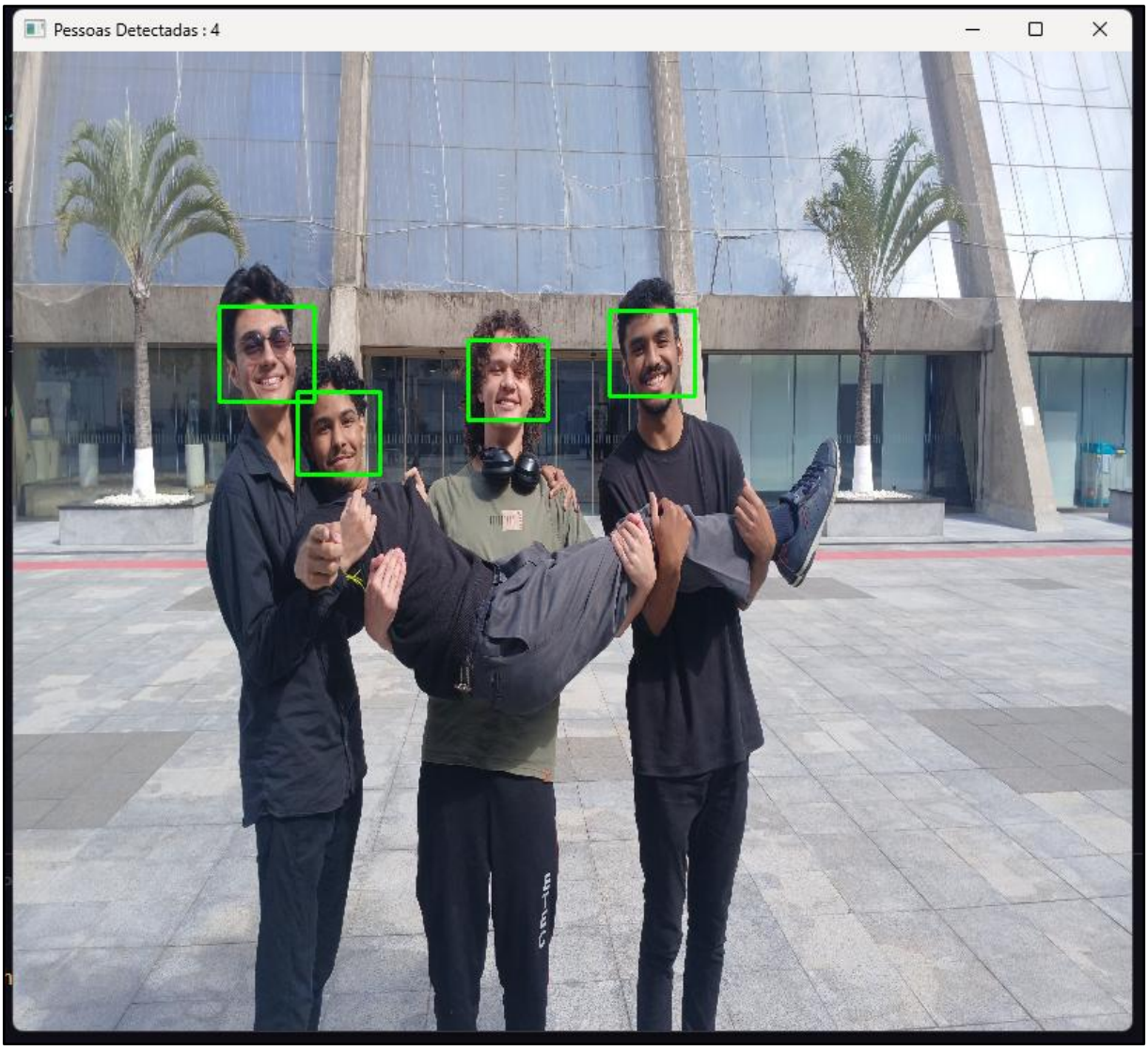
Linha 9: São definidas algumas especificações para a detecção de rostos na imagem, sendo elas a escala da análise e a sensibilidade dos resultados.

Linha 11 e 12: Responsáveis por criar marcações nos rostos que forem detectados, as variáveis aqui recebem algumas especificações de cor e formato a ser exibido.

Linha 14: Utilizando o “print”, é exibido uma mensagem de texto acompanhada da quantidade de rostos identificados.

Linha 16 a 18: São passadas propriedades de nome, intervalo de aparição e fechamento para a tela onde a análise feita é apresentada.

Figura 13 - Resultado do Código com OpenCV




Fonte: Autoria Própria, 2025.

Em seguida, serão demonstrados os conceitos que fundamentam a aplicação do MediaPipe no projeto. Sendo essa mais uma biblioteca voltada para a visão computacional.

Segundo Silva (2023), o MediaPipe é uma ferramenta poderosa para detecção e reconhecimento em tempo real, sendo especializada em oferecer alta precisão nesses ofícios.

Schirmer (2023), enfatiza que o alcance de localização da ferramenta é relativamente pequeno, porém uma vez que se define o alvo, a detecção perdura de maneira eficiente por cerca de vinte metros.

De acordo com a Google (2025), a biblioteca disponibiliza meios simplificados para o aperfeiçoamento livre dos modelos de identificação da ferramenta.

Figura 14 - Exemplo de Código Utilizando MediaPipe

```python
import cv2
import mediapipe as mp

leitor_maos = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
drawing_styles = mp_drawing.DrawingSpec

imagem = cv2.imread('ExemploCTC.jpeg')
imagem_rgb = cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)

with leitor_maos.Hands(static_image_mode = True, max_num_hands=2, min_detection_confidence=0.5) as hands:
    resultados = hands.process(imagem_rgb)

    if resultados.multi_hand_landmarks:
        for hand_landmarks in resultados.multi_hand_landmarks:

            mp_drawing.draw_landmarks(
                imagem,
                hand_landmarks,
                leitor_maos.HAND_CONNECTIONS,
                drawing_styles(color=(0, 0, 255), thickness=3, circle_radius=5),
                drawing_styles(color=(0, 0, 255), thickness=2))

imagem_redimensionada = cv2.resize(imagem, (800, 600))

cv2.imshow('Reconhecendo os Dedos', imagem_redimensionada)
cv2.waitKey(0)
cv2.destroyAllWindows
```

Fonte: Autoria Própria, 2025.

Acima, temos um exemplo da utilização da biblioteca, nele podemos ver um programa capaz de ler a imagem e reconhecer as mãos presentes nela. A seguir, é possível encontrar mais informações sobre este exemplo.

Linha 1 e 2: Importação da biblioteca OpenCv e da biblioteca do MediaPipe respectivamente, no caso do Mediapipe, ele é referenciado pelo nome “mp”.

Linha 4: Nessa linha é definido qual é a função principal do programa, no caso estão sendo utilizadas ferramentas de reconhecimento de mãos.

Linha 5: Nessa linha é definido as ferramentas que permitem a marcação daquilo que o programa identifica nas imagens.

Linha 6: Essa linha permite que as marcações recebam estilizações.

Linha 8: É declarada uma variável chamada “imagem” que recebe a propriedade de identificação da biblioteca OpenCv, e entre parênteses está sendo indicado o caminho até a imagem.

Linha 9: Como procedimento para a imagem poder ser reconhecida, a cor dela é capturada e modificada para outro padrão de coloração pelo OpenCV.

Linha 11: O objeto “leitor_maos” contém as ferramentas necessárias para a detecção de mãos. Dentro dos parênteses são passadas algumas especificações para essa leitura, como: modo de reconhecimento fixo, número máximo de mãos identificadas e credibilidade necessária para ser exibido.

Linha 12: O resultado do processamento da imagem é passado para uma variável chamada “resultados”.

Linha 14: Nessa linha é criada uma condição, caso alguma mão seja identificada, uma ação irá ocorrer.

Linha 15: Contém a ação da condição, fará uma análise dos pontos que compõem a mão na visão do computador.

Linha 17 a 22: Nas seguintes linhas, é passado uma estilização para as marcações que são feitas quando uma mão é identificada pelo programa.

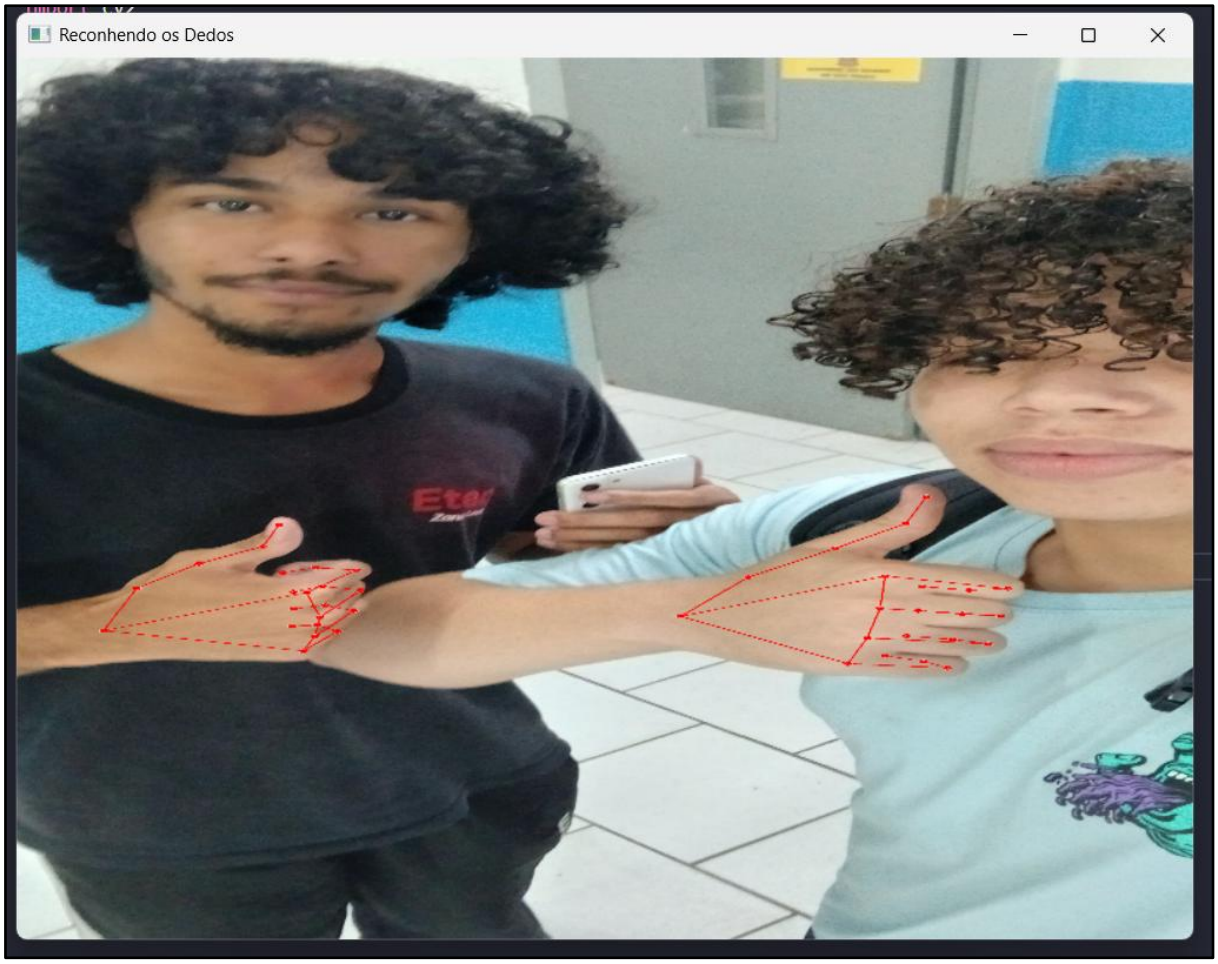
Linha 24: Está sendo declarado uma nova variável que redimensiona a imagem original, utilizando uma propriedade do OpenCV.

Linha 26: Esta linha cria a janela em que será exibida a imagem já analisada pelo programa.

Linha 27: Define um tempo de duração do programa. Ao declarar o valor como “0”, o programa ficará aberto até que uma tecla seja pressionada.

Linha 28: Comando padrão do OpenCV para encerrar o programa.

Figura 15 - Resultado do Código com MediaPipe




Fonte: Autoria Própria, 2025.

A imagem utilizada no exemplo foi demarcada com os contornos vermelhos, demonstrando que as mãos foram identificadas.

### 2.2.4 Banco de dados

De acordo com os estudos de Elmasri e Navathe (2011), a tecnologia de banco de dados é inegável para a nossa era, uma vez que praticamente todas as informações atuais se encontram em alguma forma de registro digital.

Conforme proposto por Brito (2010), o modelo de gerenciamento predominantemente seguido hoje não perdeu sua credibilidade desde sua criação.

Para a aplicação do conceito de Banco de Dados neste trabalho, foi utilizada a biblioteca SQLite, por conta de sua prática integração com a linguagem Python.

Em um artigo da plataforma Alura elaborado por Louzada (2022), é afirmado que o SQLite é um banco de dados simples, prático e intuitivo, no qual não há obrigatoriedade de utilização da arquitetura de dados convencional.

Banin (2018) enfatiza que devido à sua instalação integrada ao Python, o SQLite se revela um banco de dados ideal para aplicações nesta linguagem. Graças a sua tamanha versatilidade, ele pode ser utilizado em diversas plataformas.

Outra vantagem do SQLite é a ausência de configuração inicial, o que permite ao desenvolvedor adota-lo sem maiores complicações, como observado por Comachio (2011).

Figura 16 - Exemplo de Código Utilizando SQLite

```python
import sqlite3

conn = sqlite3.connect('exemplo.db')

cursor = conn.cursor()

cursor.execute('''
    CREATE TABLE IF NOT EXISTS professores (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nome TEXT NOT NULL,
        idade INTEGER NOT NULL
    )
    ''')

cursor.execute('''
    INSERT INTO professores (nome, idade)
    VALUES ('Jeferson', 25), ('Edna', 30), ('Carlos', 22)
    ''')

conn.commit()

cursor.execute('SELECT * FROM professores')

professores = cursor.fetchall()
for professor in professores:
    print(f"ID: {professor[0]}, Nome: {professor[1]}, Idade: {professor[2]}")

conn.close()
```

Fonte: Autoria Própria, 2025.

Na imagem, temos um exemplo simples de aplicação do banco de dados SQLite para a criação de uma base de dados. Abaixo, seguem informações mais detalhadas sobre o que cada linha do código realiza.

Linha 1: Importa o módulo sqlite3 para trabalhar com banco de dados SQLite.

Linha 3: Conecta (ou cria) o banco de dados exemplo.db.

Linha 5: Cria um cursor para executar comandos SQL.

Linha 7 - 13: Executa um comando SQL para criar a tabela “professores”, se ela ainda não existir.

Linha 15 - 17: Insere três registros (Jeferson, Edna, Carlos) na tabela “professores”.

Linha 19: Confirma (salva) as alterações no banco de dados.

Linha 21: Executa um comando SQL para selecionar todos os registros da tabela “professores”.

Linha 23: Recupera todos os registros retornados pela consulta.

Linha 24 - 25: Itera sobre os registros e imprime o ID, nome e idade de cada professor.

Linha 27: Fecha a conexão com o banco de dados.

### 2.2.5 Shell Script

A fim de otimizar o projeto por meio de camadas auxiliares e uma execução de comandos discreta, foi empregado este conceito de tecnologia.

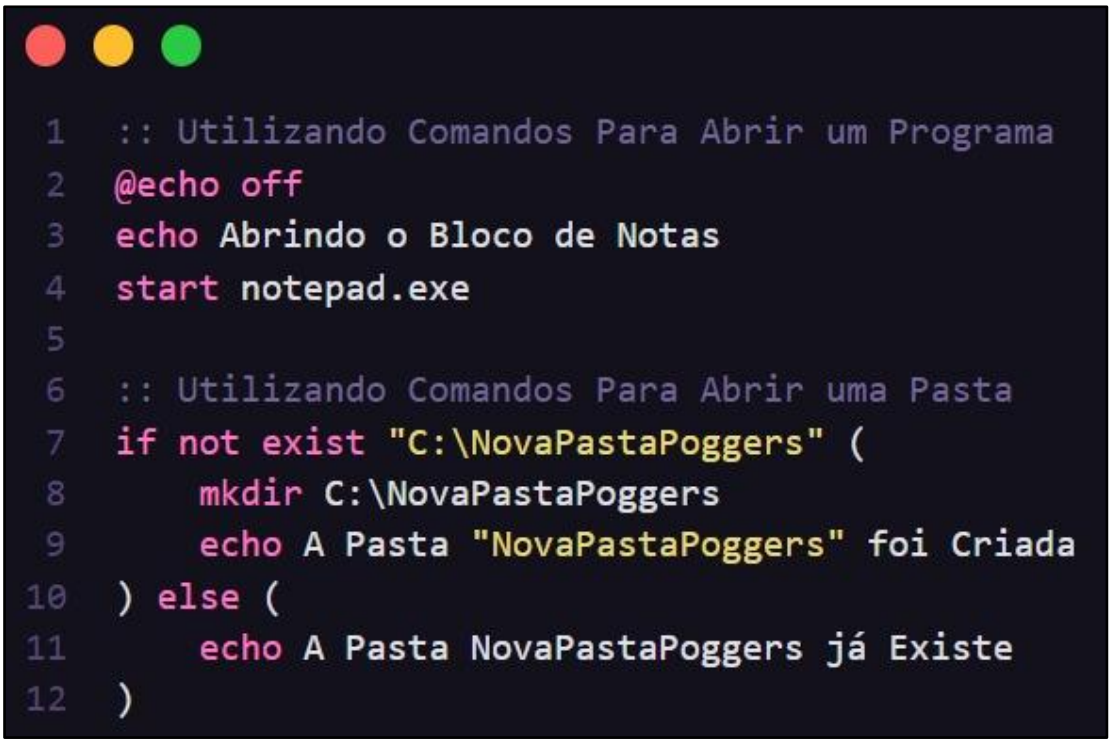
Para Negus (2014), anteriormente ao uso de atalhos e símbolos, os sistemas operacionais do tipo UNIX, permitiam a comunicação entre o usuário e a camada mais primitiva do sistema, por meio de um Shell.

Em sua obra, Neves (2010) demonstra que um Shell, quando bem utilizado, pode ser considerado uma linguagem de programação, por admitir quantidades múltiplas de scripts, mas sendo originalmente um simples conversor de comandos para a linguagem do Sistema Operacional.

Enquanto os scripts, em consonância com as ideias de Jargas (2008), são códigos broncos e executáveis que permitem o desempenho de tarefas, podendo elas serem simples ou complexas.

A seguir, um terminal baseado nos princípios Shell é utilizado para a execução de um script.

Figura 17 - Exemplo de Script no Interpretador




Fonte: Autoria Própria, 2025.

Linha 1: Simboliza um comentário no código (anotação que não interfere no funcionamento do código).

Linha 2: Comando para evitar que o script inteiro seja interpretado como uma mensagem.

Linha 3: O comando echo é usado para exibir uma mensagem.

Linha 4: Comando utilizado para iniciar um programa.

Linha 7: Condição para criar uma pasta nova, a condição diz que só sera criado se não existir outra pasta com o mesmo nome.

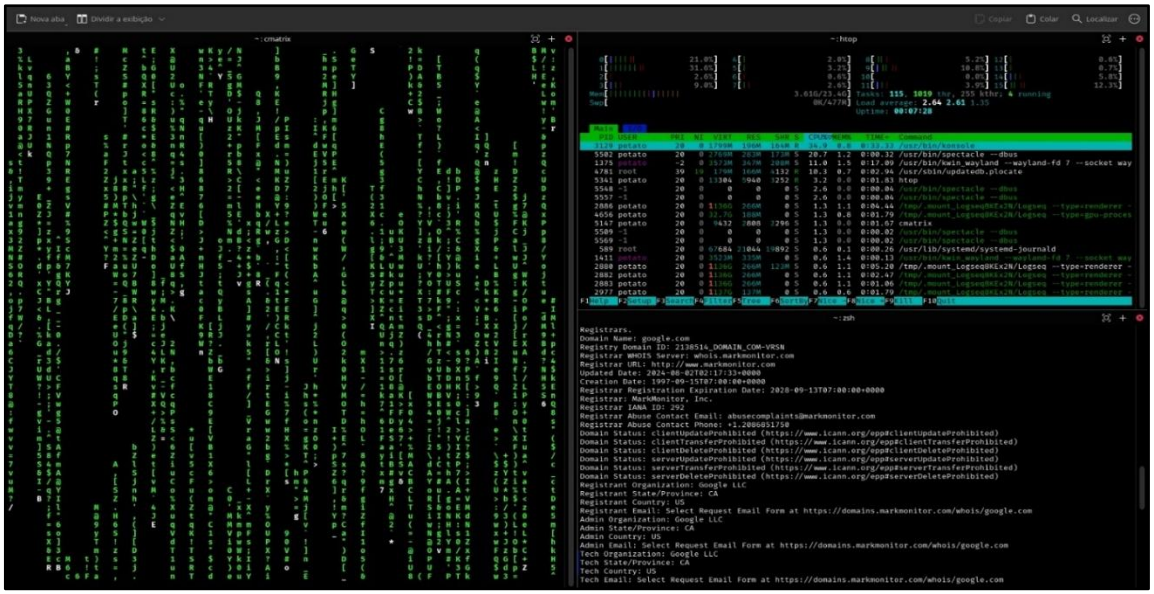
Linha 8: Comando utilizado para criar uma pasta, ao lado dele é informado a localização e o nome da nova pasta.

Linha 10-12: Segunda parte da condição anterior, ela diz que se já houver uma pasta com o nome que informamos uma mensagem será exibida informando.

A fim de complementar a explicação do termo UNIX, eis uma observação fundamentada a respeito.

Tal como observa Negus (2014), o Sistema Operacional Linux, é um motor para o desenvolvimento de diversas aplicações, entre inúmeras empresas. Sendo originalmente baseado em sistemas UNIX, o padrão código aberto do Linux, faz com que ele seja amplamente utilizado na raiz de muitas tecnologias.

Figura 18 - Exemplo de Interface no Linux



Fonte: Autoria Própria, 2025.

Na imagem em questão, mais especificamente no quadrante à direita superior, é possível ver um comando Linux em operação, chamado Htop, ele exibe todos os processos em execução, e especifica a quantidade de recursos destinada a cada um.

Na direita inferior, está sendo utilizado a ferramenta “whois google”, que permite consultar informações públicas sobre sites na internet.

No quadrante à esquerda, é exibido um efeito gráfico que faz o terminal de scripts simular uma chuva de caracteres. Esse efeito é gerado graças ao programa cmatrix.

### 2.2.6 Unified Model Language (UML)

Como parte dos requisitos técnicos para a conclusão deste curso, é necessário seguir os fundamentos que serão explicados abaixo ao se elaborar um sistema.

À luz do pensamento de Booch (2012), após o que foi chamado de Guerra de Métodos, com duração estimada de seis anos, um grupo de engenheiros de software se uniu para alinhar suas ideias em apenas um único método de modelagem.

A análise de Guedes (2018), indica que graças a essa linguagem de modelagem, é possível estruturar, elencar particularidades e identificar necessidades físicas para uso, antes mesmo de algum sistema sequer ter seu desenvolvimento iniciado.

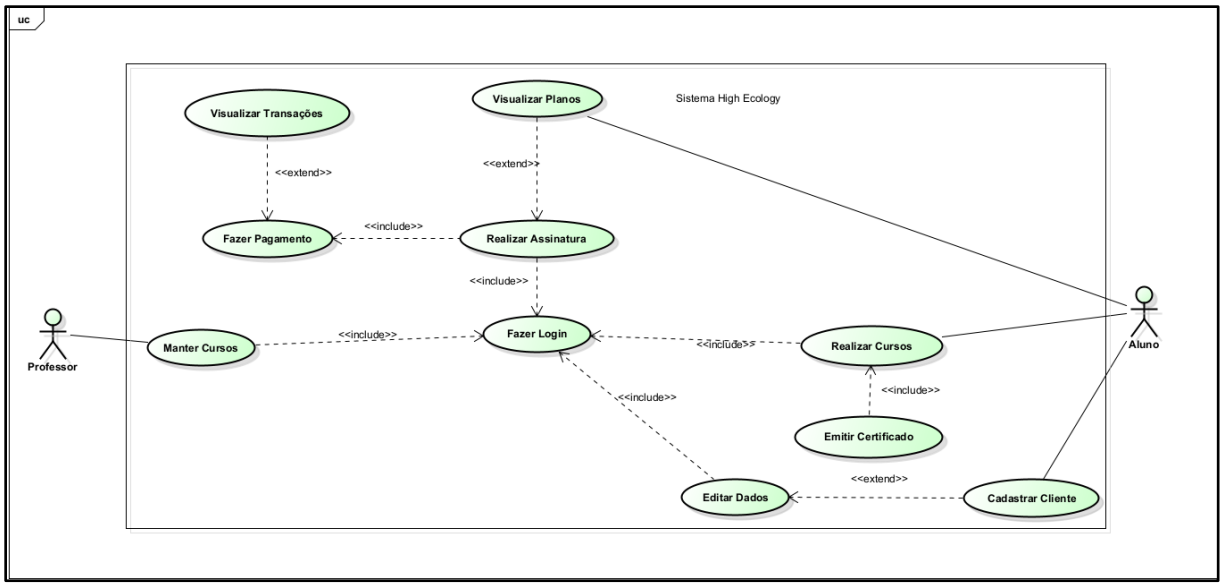
Na concepção de Fowler (2005), as normas definidas e descritas pela UML sobre a criação de diagramas, entregam aos desenvolvedores a oportunidade de comunicar de forma mais compreensível a visão que se tem de um projeto.

No total existem treze diagramas no padrão da UML, para o desenvolvimento do projeto em questão, foram utilizados os seguintes diagramas: Diagrama de Caso de Uso, Diagrama de Atividade, Diagrama de Sequência, Diagrama de Classe.

#### 2.2.6.1 Diagrama de Caso de Uso

Pressman (2021) argumenta que um diagrama de caso de uso seria a comunicação entre todas as ações que podem ser tomadas no sistema. De forma isolada, um caso de uso pode ser comparado a um contrato, nele é definida a intenção que se tem com um sistema.

Figura 19 - Exemplo de Diagrama de Caso de Uso




Fonte: Autoria Própria, 2025.

Na imagem acima, temos um exemplo de um diagrama deste tipo. O diagrama está pautado em um sistema de uma plataforma de cursos, onde os principais usuários são definidos como “Professor”, o responsável pela administração dos conteúdos disponíveis no sistema e, assumindo o papel de cliente neste sistema, há o “Aluno”.

### 2.2.6.2 Documentação do Caso de Uso

Elencar os interessados e envolvidos com o sistema, relacionar contratos presentes no diagrama, apontar os requisitos para a execução e informar resultados da atividade em questão, essas são as funções da documentação de um caso de uso, conforme apontado por Guedes (2018).

Figura 20 - Exemplo de Documentação de um Caso de Uso

| **Nome do Caso de Uso** | **UC01 – Realizar Ligação** |
|---|---|
| Ator Principal | Usuário |
| Atores Secundários | |
| Resumo | Descreve os passos necessários para que um usuário de celular possa fazer uma ligação para um determinado número |
| Pré-condições | |
| Pós-Condições | |
| **Cenário Alternativo I – Número do Contato Conhecido** | |
| **Ações do Ator** | **Ações do Sistema** |
| 1. Solicitar a lista de contatos | |
| | 2. Apresentar todos os contatos registrados em ordem alfabética |
| 3. Selecionar contato | |
| | 4. Consultar e apresentar o número do contato selecionado |
| **Cenário Alternativo II – Número não Registrado** | |
| **Ações do Ator** | **Ações do Sistema** |
| 1. Informar o número a discar | |
| **Cenário Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
| 1. Confirmar o número da ligação | |
| | 2. Estabelecer ligação |
| Restrições/Validações | |

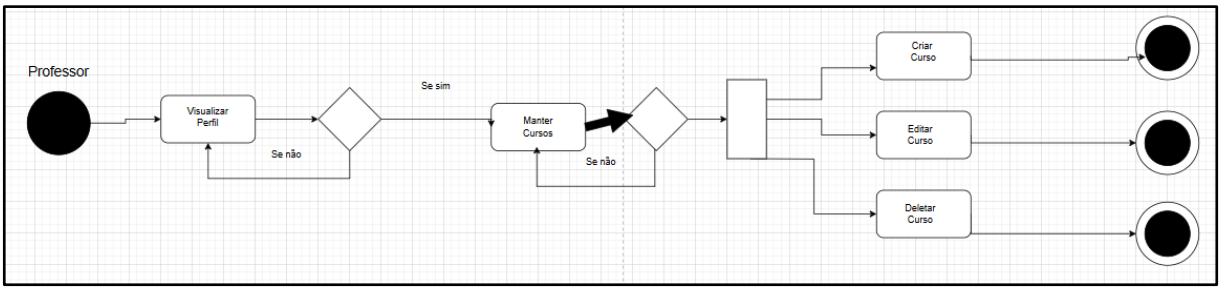
Fonte: Guedes, 2018.

No exemplo acima, é possível ver como uma funcionalidade é descrita seguindo o modelo de documentação. Ela segue um modelo de tabela, assim dividindo o tópico principal de sua descrição. Nota-se que o padrão a ser seguido para essa documentação é composto de funcionalidade principal, explicação de objetivo, condições da ação, resultados da ação, especificação das responsabilidades do sistema e do respectivo ator para a conclusão daquela funcionalidade.

### 2.2.6.3 Diagrama de Atividade

De acordo com Booch (2012), este diagrama detém a imcumbência de demonstrar o fluxo das operações de uma aplicação.

Figura 21 - Exemplo de Diagrama de Atividade



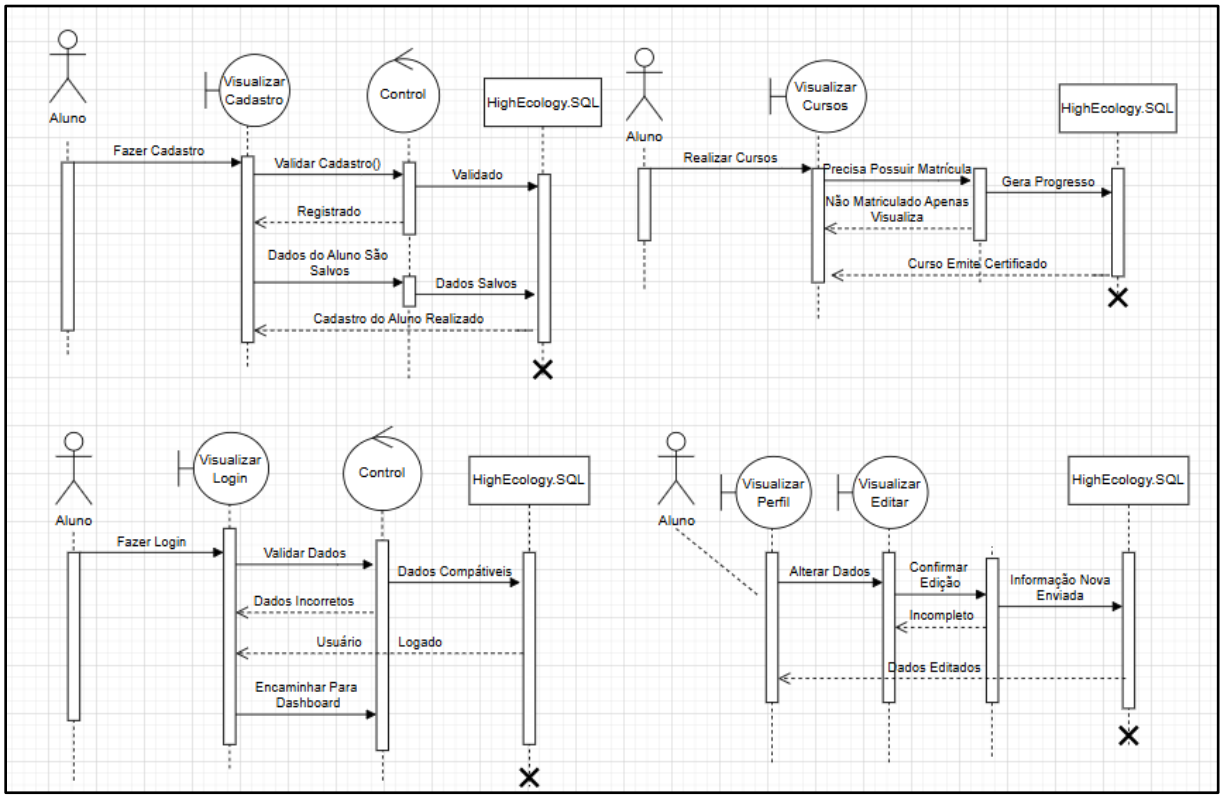

Fonte: Autoria Própria, 2025.

No diagrama apresentado, é descrito um fluxo de ação de uma das funcionalidades disponiveis para o ator “Professor” nesse sistema. Os retangulos arredondados se referem a ações, o círculo totalmente preenchido indica o ponto de inicio do fluxo e os circulos com bordas, por sua vez indicam o fim de um fluxo.

### 2.2.6.4 Diagrama de Sequência

Fowler (2005) ressalta que o objetivo deste diagrama é demonstrar um fluxo sequencial para a interpretação e conclusão por parte do sistema de uma etapa por vez.

Figura 22 - Exemplo de Diagrama de Sequência



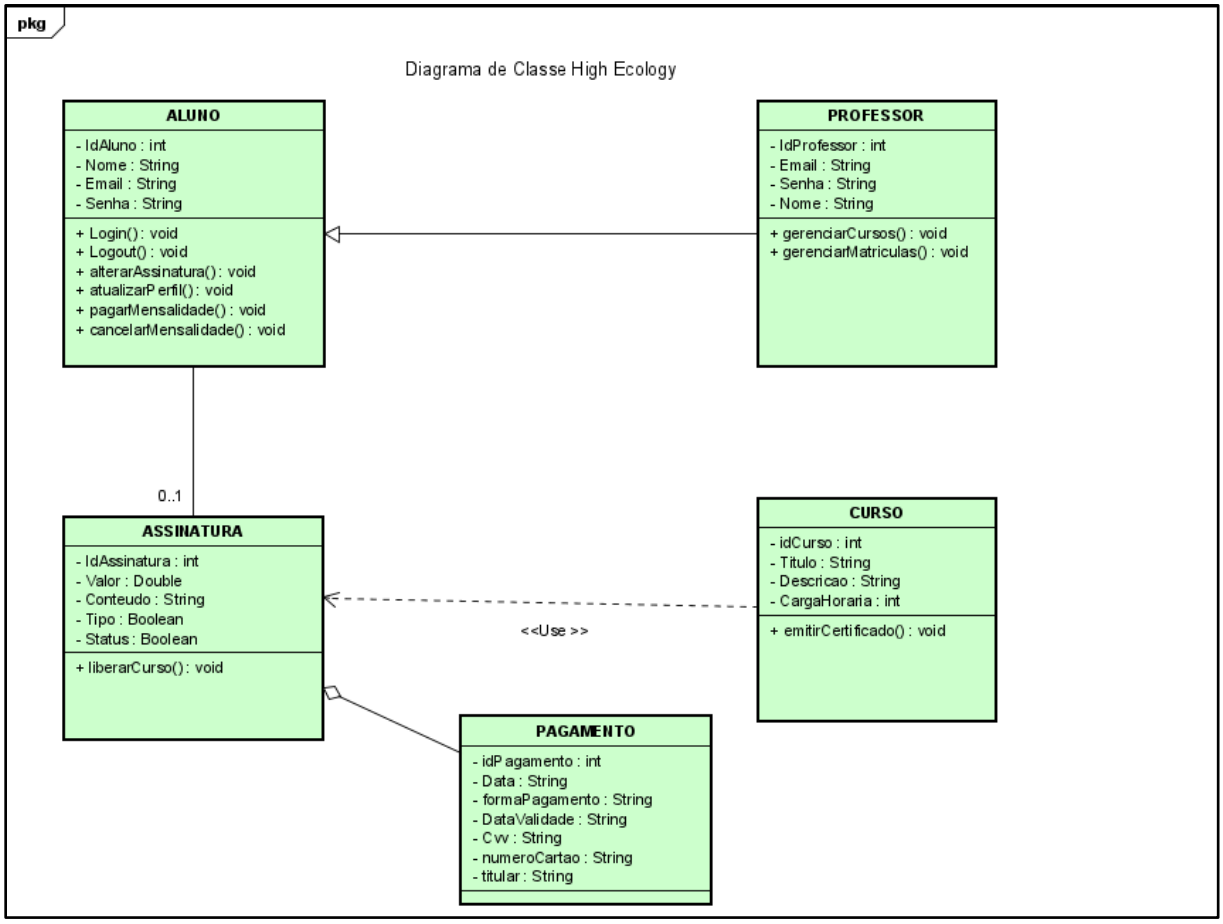

Fonte: Autoria Própria, 2025.

Este diagrama demonstra quatro cenários de interação, todas se referindo a como o sistema interpreta as ações do ator “Aluno”. Nele é possível ver o fluxo completo das ações “Fazer Cadastro”, “Fazer Login”, “Alterar Dados” e “Realizar Cursos”. Os círculos ligados a uma barra indicam interfaces visuais, o círculo “Control” indica validações do próprio sistema High Ecology, o dito "HighEcology.SQL” no retângulo ao final de cada fluxo indica que os dados estão sendo validados e então gravados no Banco de Dados do sistema.

### 2.2.6.5 Diagrama de Classe

Consoante a Fowler (2005), o papel desse diagrama é retratar os componentes de um sistema, bem como os seus respectivos relacionamentos entre si. Ademais, em sua análise, Guedes (2018) indica que é essencial para a construção dos demais diagramas do modelo UML que este esteja presente durante seu desenvolvimento.

Figura 23 - Exemplo de Diagrama de Classe




Fonte: Autoria Própria, 2025.

No diagrama de classe utilizado como exemplo, é possível ver as entidades (retrato de pessoas ou objetos que recebem atributos e interagem com o sistema) que fazem parte dessa aplicação representadas por meio das tabelas retangulares. As setas que interligam essas tabelas são chamadas de relacionamentos. A seta pontilhada indica que a tabela “Curso” precisa de uma “Assinatura” para existir, enquanto a seta entre a tabela “Professor” e a tabela “Aluno” indica que elas partilham de alguns atributos em comum.

### 2.2.7 Prototipagem

Para a compreensão de como foram elaborados os conceitos visuais deste projeto, serão referenciadas descrições técnicas dos mesmos.

Como destaca Grilo (2019), projetar telas, organizar a disposição dos elementos e pensar nos comportamentos destes, são práticas determinantes no desenvolvimento de Interfaces de Usuário (UI).

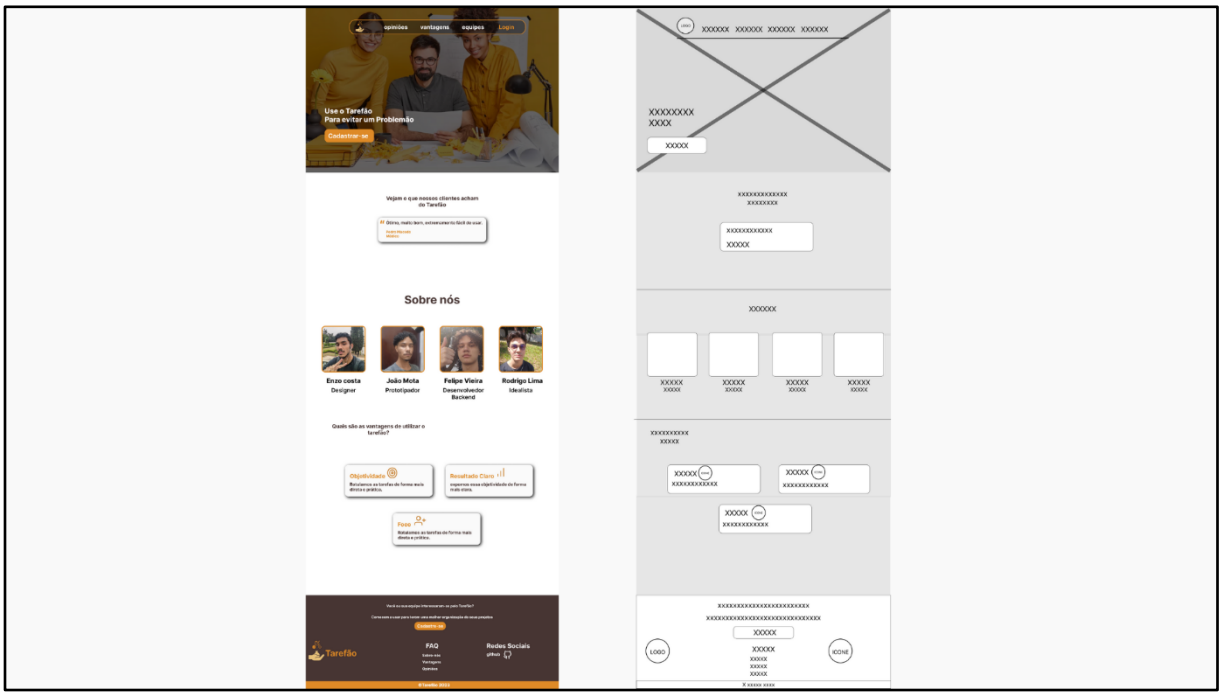
Parte de tudo que o ser humano vivencia em um ambiente digital é chamado de experiência, o papel da UX é estudar como aprimorar a percepção do usuário sobre elas, como observado por Teixeira (2014).

Nos estudos de Grilo (2019), é apresentada a ideia de que o desenvolvimento de Experiências de Usuário é determinante para a criação de sistemas completos e bem feitos.

Os conceitos de UI e UX demonstrados foram estudados durante a construção de Wireframes para este projeto.

A plataforma Miro (2025) aponta em uma de suas publicações que os Wireframes são protótipos simples para  a visualização prévia de uma aplicação, sendo estes comumente apresentados em uma forma de baixa e outra de alta fidelidade. Sendo o de baixa formado por um conteúdo básico, sem cores ou imagens e o wireframe de alta fidelidade formado por uma complexidade maior, conforme descrito na plataforma GoDaddy (2024).

Figura 24 - Exemplo de Wireframes de Alta e Baixa Fidelidade



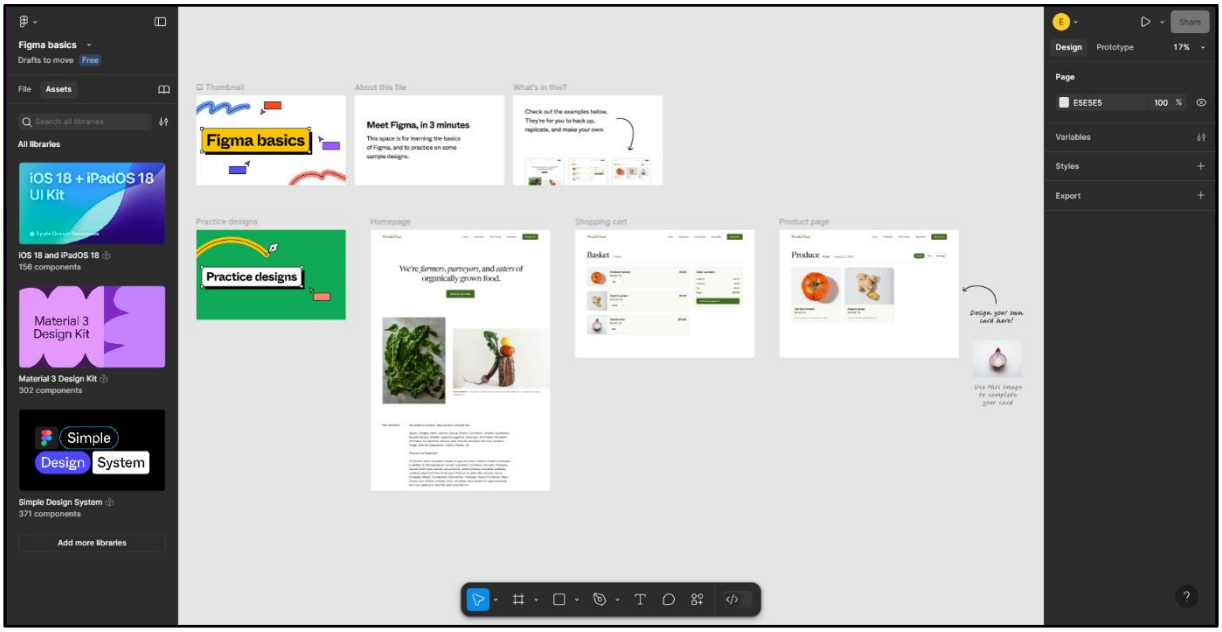

Fonte: Autoria Própria, 2025.

A reunião dos conceitos explicados anteriormente ocorre por meio da ferramenta Figma.

Tal como observa Oliveira (2022), o Figma obteve uma grande comunidade pela sua capacidade de criar um bom espaço público para o compartilhamento de designs entre os usuários.

A análise publicada na plataforma Alura (2022), indica que a ferramenta aproveita bem de sua comunidade, uma vez que ela é a responsável pela produção de sistemas, anexos e guias de estilo para a ferramenta.

Figura 25 - Exemplo de Interface da Ferramenta Figma




Fonte: Autoria Própria, 2025

### 2.2.8 Desenvolvimento Web

Na criação de um site expositivo para este projeto, conceitualmente foram aplicadas as ideias de HTML, CSS e JavaScript.

A Linguagem de Marcação de Hipertexto, pode ser definida como a responsável pelo apontamento do conteúdo e dos elementos que compõem uma página web, como apontado por Ferreira (2013). A fim de esclarecer suas nomenclaturas, Silva (2015) considera como hipertexto todo o conteúdo textual registrado em um arquivo que busca relacionar operações web.

Em seguida, é apresentado um exemplo de como essa tecnologia é utilizada e quais resultados ela traz consigo.

Figura 26 - Exemplo de Código em HTML

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
     <link rel="shortcut icon" href="LOGO M.E.R.LIN.png">
  <title>Exemplo de Página Web</title>
</head>
<body>
  <div class="container">
      <h1>EQUIPE <strong>M.E.R.LIN</strong></h1>
    <div class="cards">
      <div class="card" data-nome="Emily Cristina">
      <img src="Foto Emi.png" width="200px" alt="Foto da Emily">
        <h2>Emily Cristina</h2>
      </div>
      <div class="card" data-nome="Rodrigo Lima">
          <img src="Foto Eu.png" width="200px" alt="Foto do Rodrigo">
        <h2>Rodrigo Lima</h2>
      </div>
      <div class="card" data-nome="João Mota">
          <img src="Foto Mota.png" width="200px" alt="Foto do João">
        <h2>João Mota
      </div>
    </div>
  </div>
</body>
</html>
```

Fonte: Autoria Própria, 2025.

Para compreender as linhas que compõem o exemplo, é preciso saber que o HTML segue um padrão de escrita por tags, sendo estas todo o argumento, escrito entre os sinais de < e >, conforme ressalta Silva (2015).

As tags presentes no exemplo são esclarecidas a seguir:

“<DOCTYPE html>” : Uma instrução dedicada ao navegador, indica qual versão do HTML será utilizada para processar a página.

“<html lang>” : O “html” presente na tag, é essencial para o desenvolvimento de códigos, pois armazena todos os elementos necessários para compor uma página. Ao

lado dela temos o “lang”, utilizado para informar ao navegador em qual idioma a página é escrita.

“<head>”: Informa discretamente ao navegador, e apenas para ele, informações de comportamento da página web.

“<meta charset>”: As tags meta permitem descrever o comportamento dos dados de uma página. Em específico, o ”charset= UTF-8”, indica que o código é compatível com praticamente todas as letras e símbolos de qualquer idioma do mundo.

“<link>”: Existem diversos usos para tags deste tipo, o objetivo delas é definir relações entre recursos externos e o código em questão. No exemplo, essa tag é utilizada para adicionar um ícone personalizado à página, geralmente podendo ser visto no canto superior esquerdo dela.

“<title>”: Define um título para a página web que será exibido em uma aba, janela ou guia.

“<body>”: Em uma visão geral, todo o conteúdo visível do site é criado dentro dessa tag, o que torna ela primordial para a construção deste tipo de sistema.

“<div class>”: Esta tag é muito utilizada para organizar o código HTML, sendo capaz de isolar estilos e elementos presentes nele.

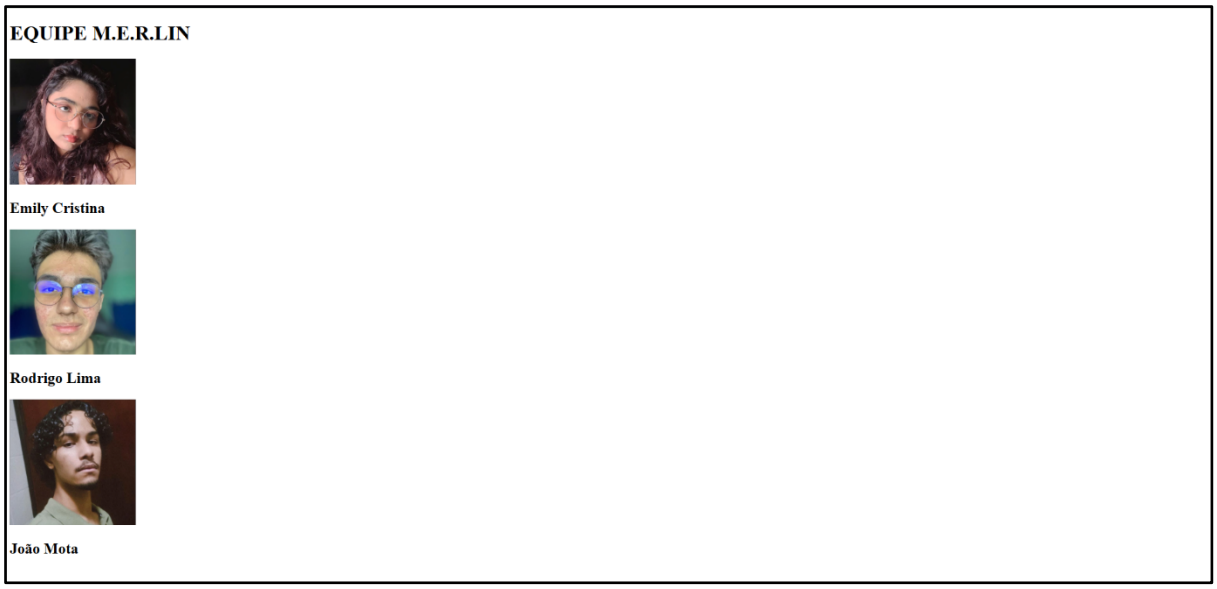
“<h1>”: Possuindo uma hierarquia de relevância na página, numerada de “<h1>” até “<h6>”, sendo essa segunda a de menor impacto, esta tag é utilizada em textos presentes na página.

“<img src>”: “img” é a parte da tag responsável por representar que uma imagem será chamada, e “src” é utilizado para especificar de onde essa imagem virá.

É importante evidenciar que as tags HTML não atuam sozinhas, muitas vezes elas aparecem acompanhadas de atributos (informações e características complementares aos seus propósitos originais), como demonstrado por Ferreira (2013).

Ao aplicar as tags anteriormente mencionadas, o resultado da página se assemelha à seguinte imagem.

Figura 27 - Resultado do Código HTML




Fonte: Autoria Própria, 2025.

As Folhas de Estilo em Cascata, em tradução livre, se relacionam com o HTML para solucionar discordâncias de padronização entre navegadores, bem como atribuir mais personalidade às páginas web criadas, como foi discutido por Mazza (2014).

Consoante aos Freeman (2009), existe um modelo de desenvolvimento do CSS, sendo necessário sempre mencionar: o seletor, o componente que será estilizado; a propriedade que será estilizada; e o novo valor dessa propriedade.

Esses conceitos podem ser observados no próximo exemplo, nota-se que foram passados valores de propriedades que tornassem a página demonstrativa o mais condizente possível com a temática do projeto.

Figura 28 - Exemplo de Código em CSS

```css
@import url("https://fonts.googleapis.com/css2?family=Playfair+Display:ital,wght@0,400..900;1,400..900&display=swap");

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: Arial, sans-serif;
}

body {
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  background: linear-gradient(135deg, #654e82, #c58ade);
}

.container {
  text-align: center;
}

h1 {
  font-size: 48px;
  margin-bottom: 30px;
  color: #34214bff;
  letter-spacing: 2px;
  font-family: Playfair;
}

strong {
  font-size: 48px;
  margin-bottom: 30px;
  color: #c58ade;
  letter-spacing: 2px;
  font-family: Playfair;
}

.cards {
  display: flex;
  gap: 20px;
  justify-content: center;
}

.card {
  background: #f9b14f;
  border-radius: 12px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.5);
  padding: 20px;
  width: 400px;
  text-align: center;
  transition: transform 0.3s, box-shadow 0.3s;
  cursor: pointer;
}

.card img {
  width: 100%;
  border-radius: 8px;
  margin-bottom: 15px;
}

.card h2 {
  font-size: 20px;
  color: #c58ade;
  font-family: Playfair;
  text-shadow: 0 1px 1px rgba(0, 0, 0, 1);
}

.card:hover {
  transform: translateY(-8px);
  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.25);
}
```

Fonte: Autoria Própria, 2025.

Aqui é possível observar 6 seletores, sendo eles:

“ * “: Se referindo a uma estilização generalizada de todo o documento.

“body”: Que representa algumas modificações que todo o conteúdo presente no corpo do HTML receberá.

“container”: Se refere a uma tag “ ” que está responsável por agrupar o conteúdo apresentado nesse exemplo.

“h1”: Que diz respeito aos títulos presentes no HTML.

“strong”: Esse seletor indica que uma parte específica dos títulos terá sua própria formatação.

“cards”: Esse seletor se refere a uma das tags “ ” anteriormente definidas, representando que uma seção da página terá especificidades em seu estilo. Os demais seletores são características menores deste seletor.

Dentro de cada seletor é possível encontrar diferentes propriedades de estilo, sendo mais comuns aquelas que alteram o posicionamento, o tamanho, a cor e o formato dos elementos da página original.

Uma vez que o código CSS foi desenvolvido, é preciso apontar para o navegador qual arquivo ele deve considerar na hora de estilizar a página, para isso utiliza-se a seguinte linha (Freeman, 2009):

Figura 29 - Exemplo de Link Entre HTML e CSS


Fonte: Autoria Própria, 2025.

Ao anexar o CSS apresentado neste capítulo ao documento HTML anteriormente criado, alcança-se este resultado.

Figura 30 - Resultado da Página HTML Estilizada


Fonte: Autoria Própria, 2025.

A linguagem de programação mais competente para a otimização de serviços web, originou–se em 1996, como uma criação da antiga empresa de serviços de computadores, Netscape, como informado na obra de Flanagan (2013).

Quando introduzido nos anos 90, o JavaScript não demonstrava ser necessário para a maioria dos sites, entretanto, com a evolução da internet, ele passou a ser primordial para evitar sobrecargas nos navegadores web, segundo Zakas (2010).

Figura 31 - Exemplo de Código em JavaScript

```
1  const cards = document.querySelectorAll('.card');
2
3  cards.forEach(card => {
4    card.addEventListener('click', () => {
5      const nome = card.getAttribute('data-nome');
6      alert(`Você Clicou na Foto de : ${nome}`);
7    });
8  });
```

Fonte: Autoria Própria, 2025.

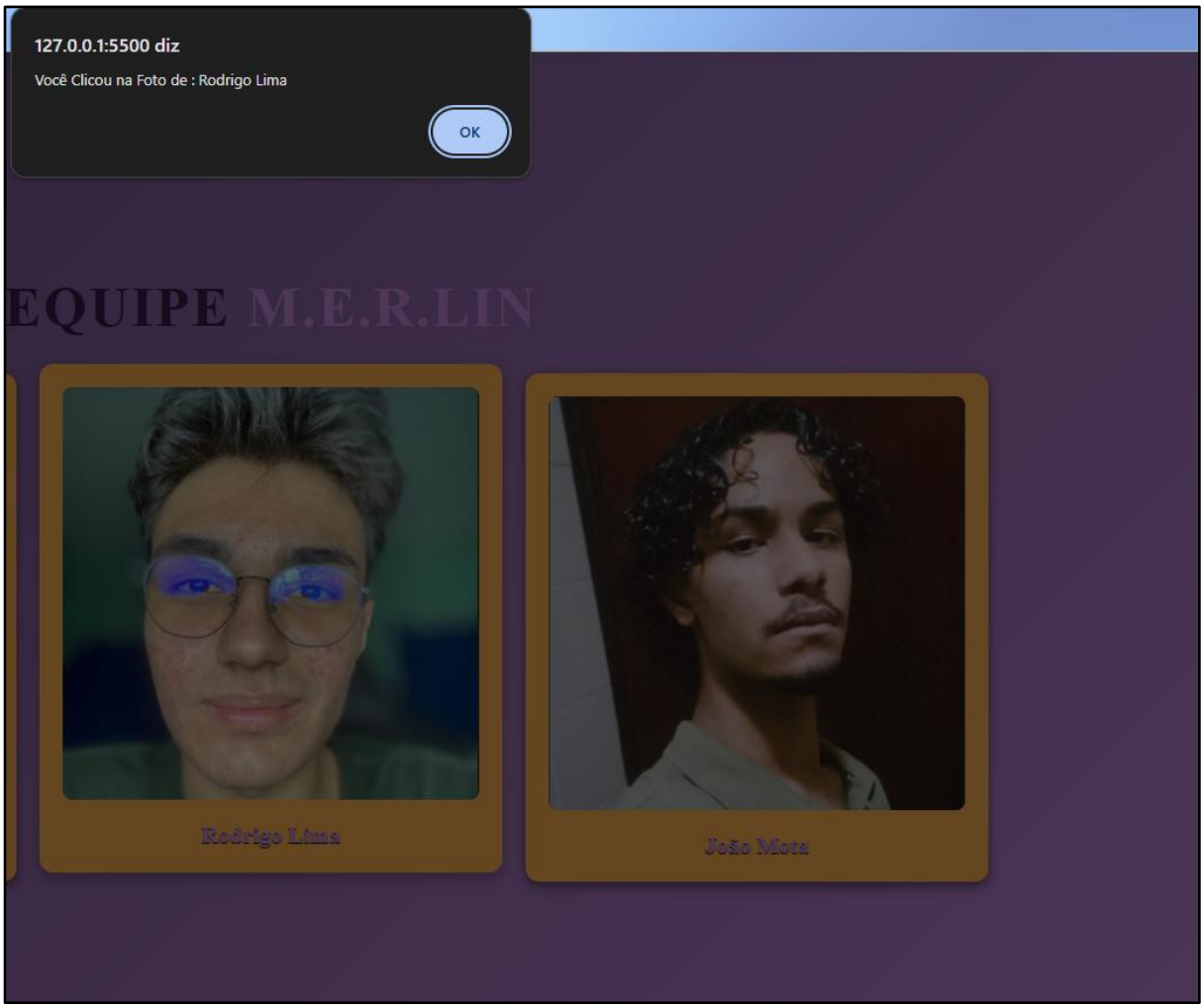
Na figura há uma demonstração de codificação em JavaScript. O objetivo desse código é exibir uma mensagem quando o usuário clicar em uma das 3 fotos presentes na tela.

Em suma, a lógica presente nesse código é formada por variáveis, funções próprias do JavaScript, mensagens e ligações com o HTML antes apresentado.

Quando se aloca o comando “<script>” no código HTML, o navegador analisa e executa a codificação JavaScript, como afirma Zakas (2010).

Ao adicionar esta linha no código HTML original, a página passa a poder executar a seguinte novidade:

Figura 32 - Resultado do Uso de JavaScript na Página




Fonte: Autoria Própria, 2025.

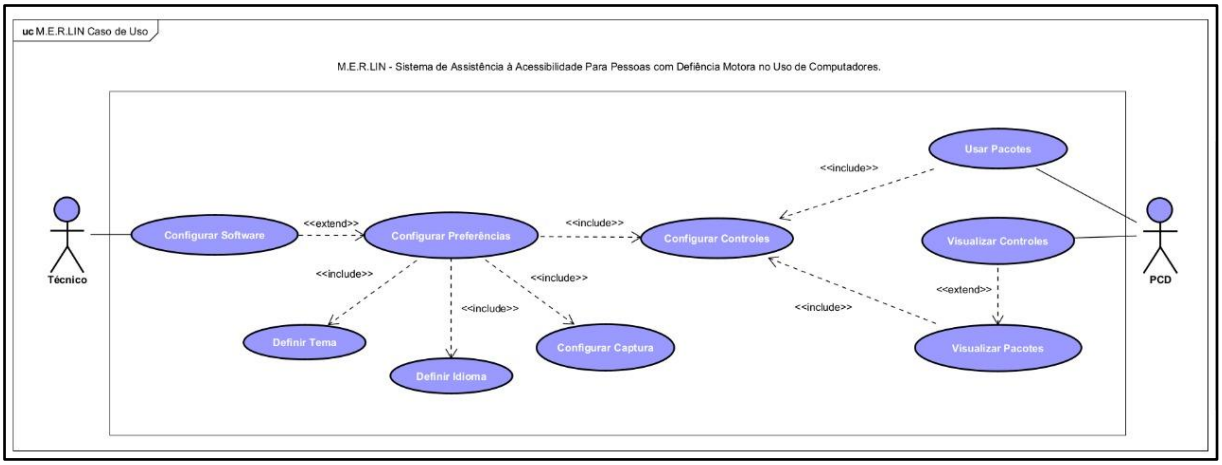
# 3 DESENVOLVIMENTO

Neste capítulo, será documentado e apresentado o desenvolvimento do software M.E.R.LIN. Serão apresentados os diagramas com base a UML, protótipos das telas do sistema e conceitos da marca criada para o projeto.

## 3.1 Diagrama de Caso de Uso

Com esse diagrama é possível ver uma síntese visual das principais funcionalidades do sistema, bem como os atores que o utilizarão, sendo o “PCD” o ator principal e o “Técnico” o responsável por configurar o sistema.

Figura 33 – Diagrama de Caso de Uso do M.E.R.LIN




Fonte: Autoria Própria, 2025.

## 3.2 Documentação do Caso de Uso

Sucedendo o Caso de Uso, temos a sua Documentação, na qual são apresentados conceitos não visuais do Caso de Uso, como: Requisitos Funcionais (RF), Requisitos Não Funcionais (RNF) e Regras de Negócio (RN).

O Requisito Funcional é o responsável por definir a ação que o sistema deverá executar para cumprir com o seu propósito de desenvolvimento. Os Requisitos Não Funcionais servem para definir como deverá ser operada a aplicação, especificando seu desempenho, usabilidade, segurança, confiabilidade e entre outros. Já as Regras de Negócios fazem parte da marca, são o processo jurídico e político privado da aplicação, na qual fazem parte as: diretrizes, condições, finanças e serviços.

Requisitos Funcionais do Técnico:

- RF01 - O Técnico poderá Definir Tema
- RF02 – O Técnico poderá Definir Idioma

- RF03 - O Técnico poderá Configurar Captura
- RF04 - O Técnico poderá Configurar Preferências
- RF05 - O Técnico poderá Configurar Controles
- RF06 - O Técnico poderá Configurar Software

Requisitos Funcionais do PCD

- RF09 – O PCD poderá Visualizar Controles;
- RF10 - O PCD poderá Selecionar Pacotes;
- RF11 - O PCD poderá Visualizar Pacotes;
- RF12 - O PCD poderá Usar Pacotes;

Requisitos Não Funcionais:

- RNF01 - Variação de Temas;
- RNF02 - Disponibilidade de Idiomas;
- RNF03 - Suporte à Diferentes Câmeras;
- RNF04 - Tutoriais de Uso;
- RNF05 - Telas Eficientes;
- RNF06 - Bom Desempenho;
- RNF07 - Software Otimizado.

Regras de Negócio:

- RG1 - Atendimento empresarial a partir de microempresas.
- RG2 - A câmera sempre estará ativa.
- RG3 - Termos de Uso (Definições, descrição, direitos e responsabilidades, direitos autorais, privacidade, processo de uso, limitação de responsabilidade, alteração dos termos, contato).

Tabelas descritivas de cada Caso de Uso isolado:

Figura 34 - Documentação do Caso de Uso: Configurar Software

| **Nome do Caso de Uso** | Configurar Software |
|---|---|
| **Ator Principal** | Técnico |
| **Resumo** | Este caso de uso descreve a ação de definir o modo de configuração do sistema, podendo ser padrão ou personalizado |
| **Pós-Condições** | 1. O técnico definirá como o sistema vai se comportar após a escolha do modo |
| **Cenário Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
| | 1. Exibir os modos de configuração |
| 2. Selecionar um modo de configuração | 3. Definir todas as configurações de maneira automática em caso da escolha ser a “padrão” |

Fonte: Autoria Própria, 2025.

Figura 35 - Documentação do Caso de Uso: Configurar Preferências

| **Nome do Caso de Uso** | Configurar Preferências |
|---|---|
| **Ator Principal** | Técnico |
| **Resumo** | Este caso de uso define a definição das opções que envolvem a configuração personalizada do software |
| **Pré-Condições** | 1. O técnico deve ter requisitado o modo de configuração personalizado |
| **Cenário Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
| | 1. Apresentar todas as configurações que podem ser customizadas pelo usuário |

Fonte: Autoria Própria, 2025.

Figura 36 - Documentação do Caso de Uso: Definir Tema

| | |
|---|---|
| **Nome do Caso de Uso** | Definir Tema |
| **Ator Principal** | Técnico |
| **Resumo** | Este caso de uso se refere à ação de especificar o padrão de cores aplicado ao software |
| **Pré-Condições** | 1. O técnico deve ter requisitado o modo de configuração personalizado |
| **Cenário Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
| | 1. Exibir todas as opções de tema disponiveis |
| | 2. Salvar a definição escolhida |

Fonte: Autoria Própria, 2025.

Figura 37 - Documentação do Caso de Uso: Definir Idioma

| | |
|---|---|
| **Nome do Caso de Uso** | Definir Idioma |
| **Ator Principal** | Técnico |
| **Resumo** | Este caso de uso se refere à ação de especificar a linguagem em que o software é apresentado |
| **Pré-Condições** | 1. O técnico deve ter requisitado o modo de configuração personalizado |
| **Cenário Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
| | 1. Exibir todos os idiomas em que o software estiver disponível |
| | 2. Salvar a definição escolhida |

Fonte: Autoria Própria, 2025.

Figura 38 - Documentação do Caso de Uso: Configurar Controles

| **Nome do Caso de Uso** | Configurar Controles |
|---|---|
| **Ator Principal** | Técnico |
| **Resumo** | Este caso de uso se refere à ação de utilizar algum dos comandos definidos pelos usuários |
| **Pós-Condições** | 1. O PCD poderá utilizar as ferramentas de acessibilidade do software |
| **Cenário Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
| | 1. Exibir os comandos disponiveis no sistema |
| 2. Definir um uso para cada comando | |

Fonte: Autoria Própria, 2025.

Figura 39 - Documentação do Caso de Uso: Usar Pacotes

| **Nome do Caso de Uso** | Usar Pacotes |
|---|---|
| **Ator Principal** | PCD |
| **Resumo** | Este caso de uso se refere a ação do usuário de executar um comando que abre vários programas de uma só vez |
| **Pré-Condições** | 1. As ações requisitadas pelo PCD devem ter sido registradas anteriormente |
| **Cenário Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
| 1. Requisitar a ativação de um comando | 2. Identificar a requisição |
| | 3. Atender à ação |

Fonte: Autoria Própria, 2025.

Figura 40 - Documentação do Caso de Uso: Visualizar Controles

| **Nome do Caso de Uso** | Visualizar Controles |
|---|---|
| **Ator Principal** | PCD |
| **Resumo** | Este caso de uso se refere a ação de visualizar como executar algum comando em especifico |
| **Pré-Condições** | 1. As ações requisitadas pelo PCD devem ter sido registradas anteriormente |
| **Cenário Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
| | 1. Apresentar todos os comandos anteriormente definidos |
| 2. Selecionar um comando especifico | 3. Exibir o vídeo de demonstração adequado |

Fonte: Autoria Própria, 2025.

Figura 41 - Documentação do Caso de Uso: Visualizar Pacote

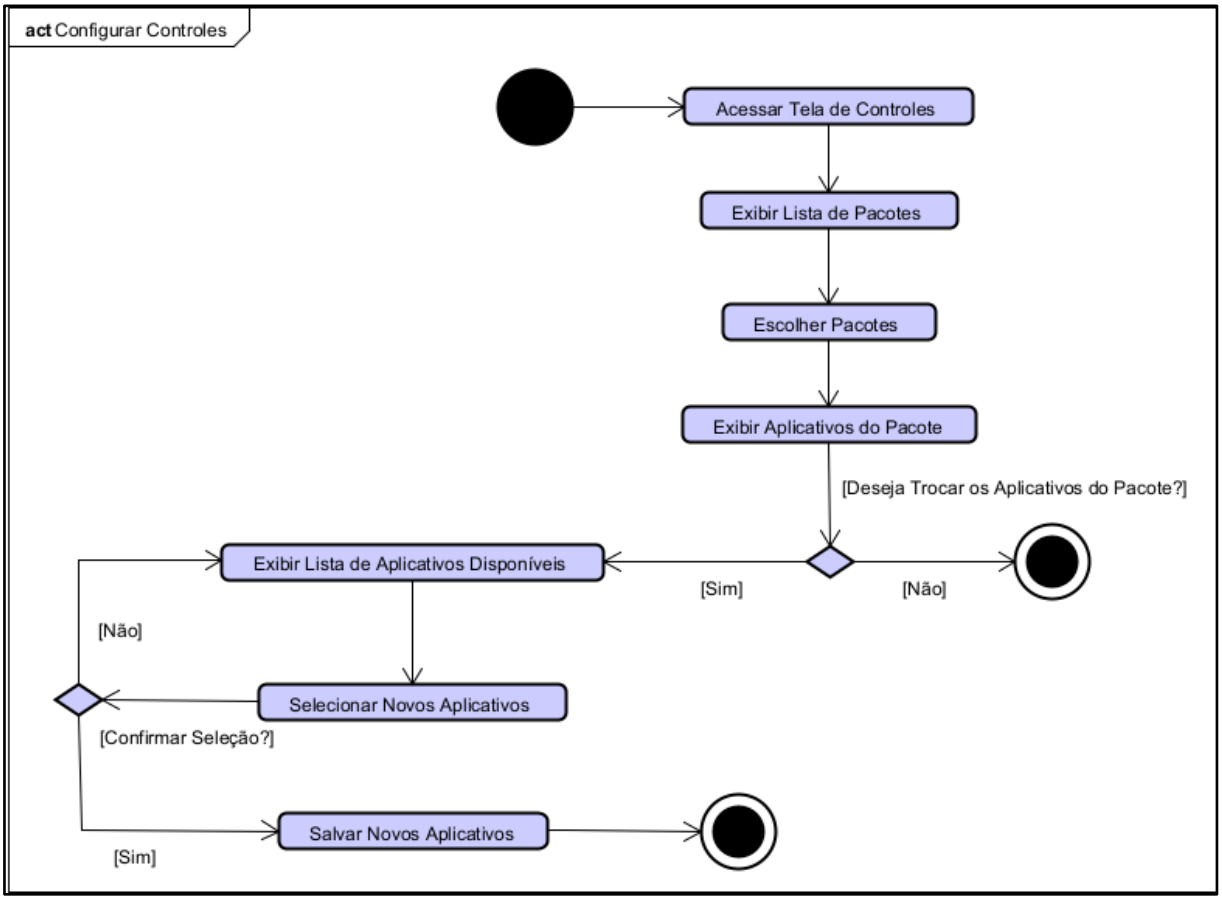
| **Nome do Caso de Uso** | Visualizar Pacote |
|---|---|
| **Ator Principal** | PCD |
| **Resumo** | Este caso de uso se refere a ação de escolher entre um dos pacotes de comandos definidos, para então poder visualizar como executá-lo |
| **Pré-Condições** | 1. As ações requisitadas pelo PCD devem ter sido registradas anteriormente |
| **Cenário Principal** | |
| **Ações do Ator** | **Ações do Sistema** |
| | 1. Apresentar todos os pacotes anteriormente criados |
| 2. Selecionar um pacote especifico | 3. Exibir o vídeo de demonstração adequado |

Fonte: Autoria Própria, 2025.

## 3.3 Diagrama de Atividade

As consecutivas imagens demonstram os Diagramas de Atividade do principal sistema do M.E.R.LIN, o objetivo destas é mostrar o fluxo de uso das ações do sistema.

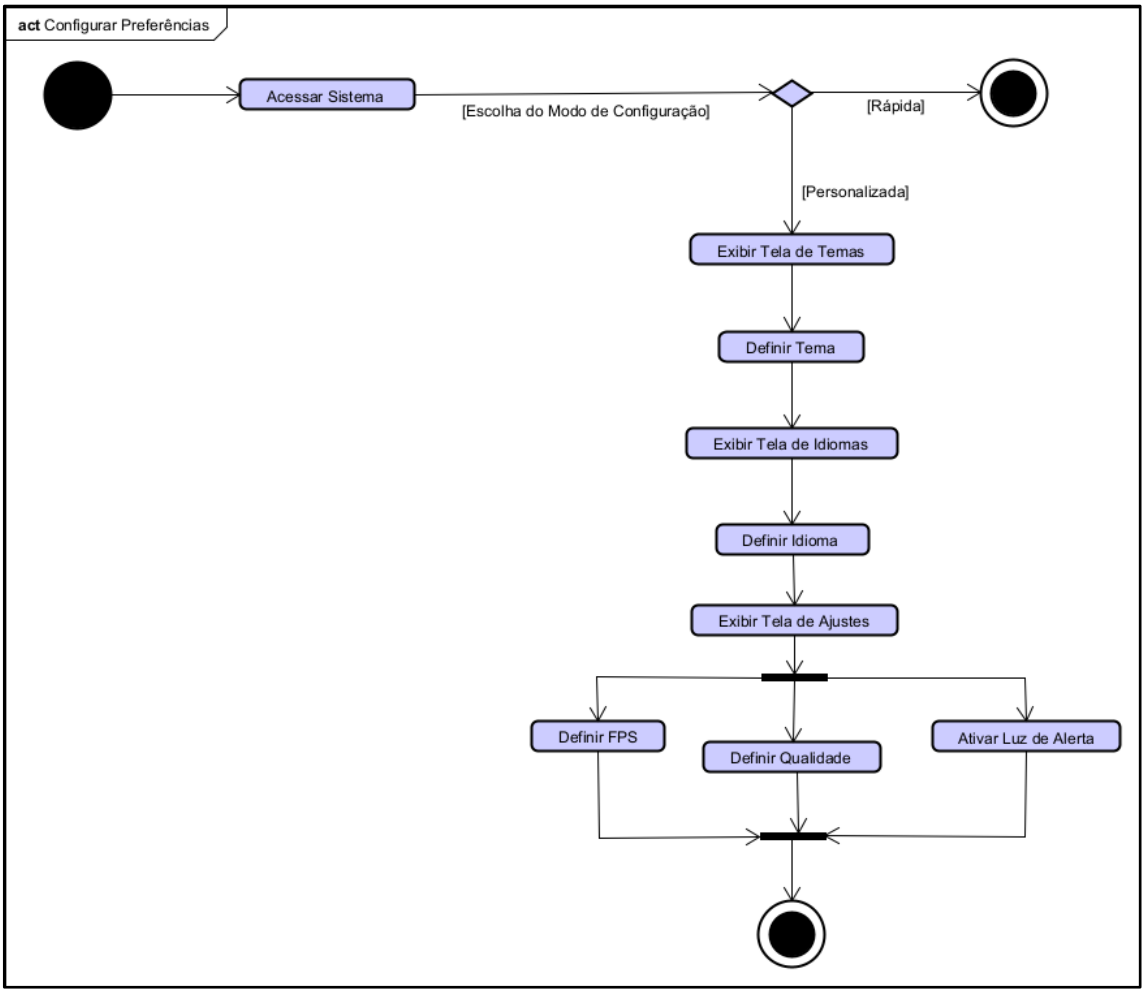
Figura 42 - Diagrama de Atividade: Configurar Controles




Fonte: Autoria Própria, 2025.

Este diagrama busca representar o fluxo de ações da funcionalidade de Configurar Controles, uma responsabilidade atribuída exclusivamente ao ator “Técnico”.

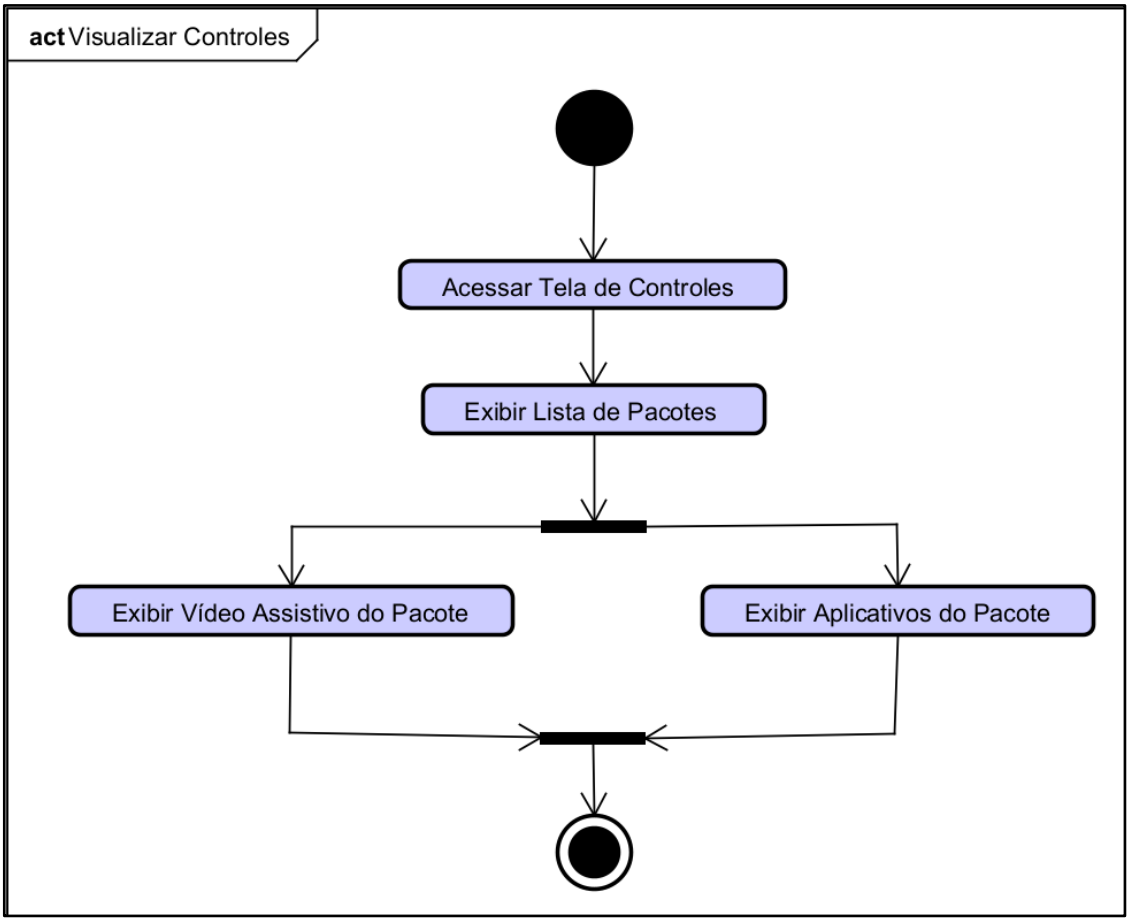
Figura 43 - Diagrama de Atividade: Configurar Preferências




Fonte: Autoria Própria, 2025.

Este diagrama busca representar o fluxo de ações da funcionalidade de Configurar Preferências, outra tarefa exclusiva apenas para o ator “Técnico”.

Figura 44 - Diagrama de Atividade: Visualizar Controles



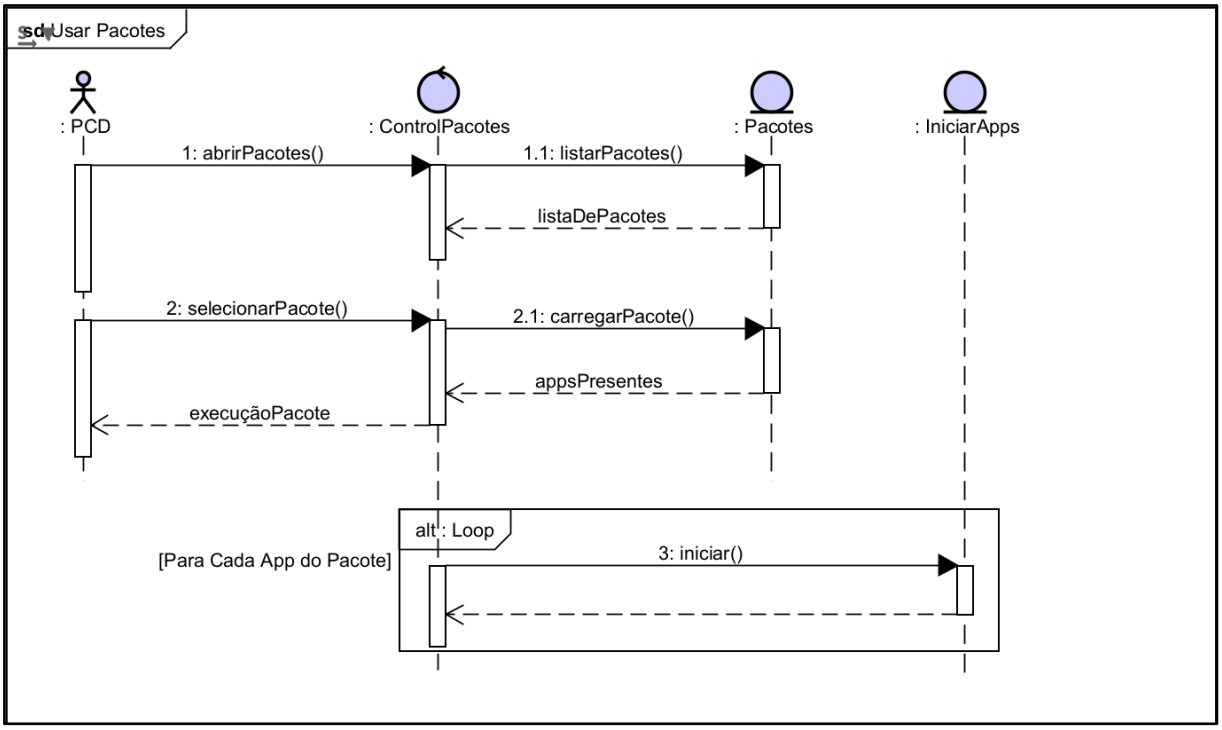

Fonte: Autoria Própria, 2025.

Este diagrama diz respeito ao fluxo de ação da funcionalidade de Visualizar Controles, uma ação pertencente ao ator “PCD”.

### 3.4 Diagrama de Sequência

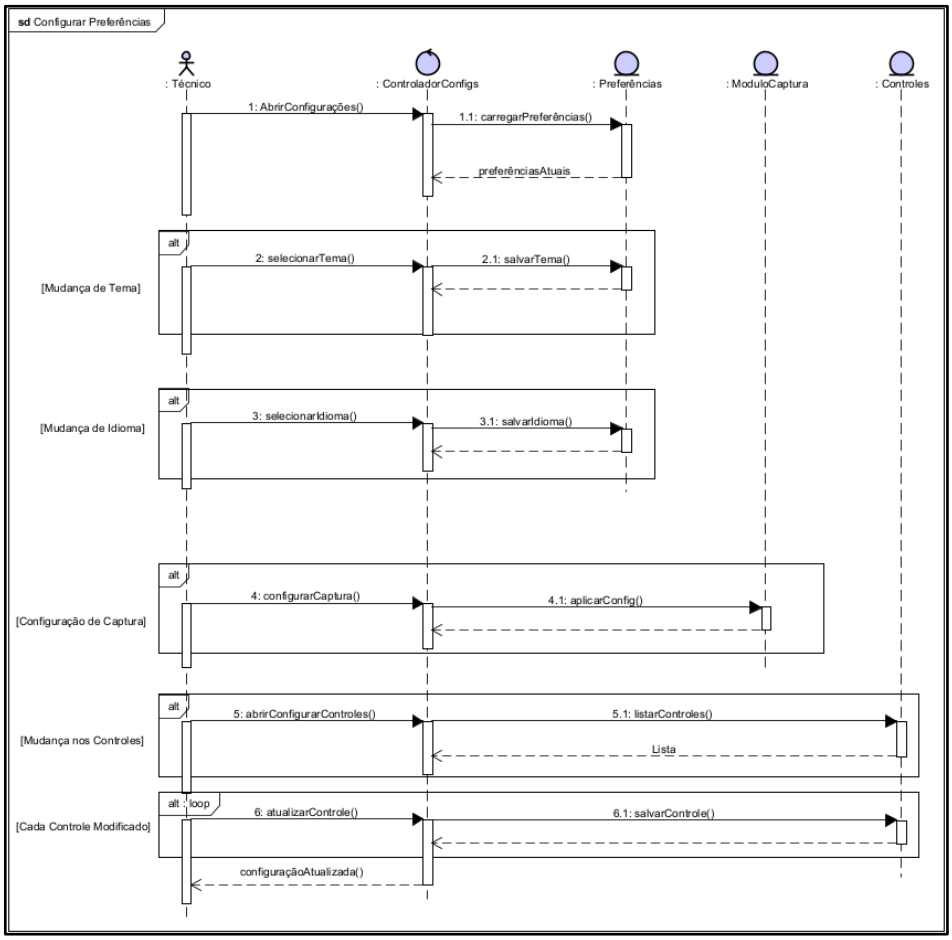
O penúltimo diagrama presente no desenvolvimento do M.E.R.LIN evidencia a visão interna que o sistema têm de uma ação. Como dito em seu nome, representa a ordem da obtenção, envio e intepretação das informações necessárias para o sistema completar uma operação.

Figura 45 - Diagrama de Sequência: Usar Pacotes



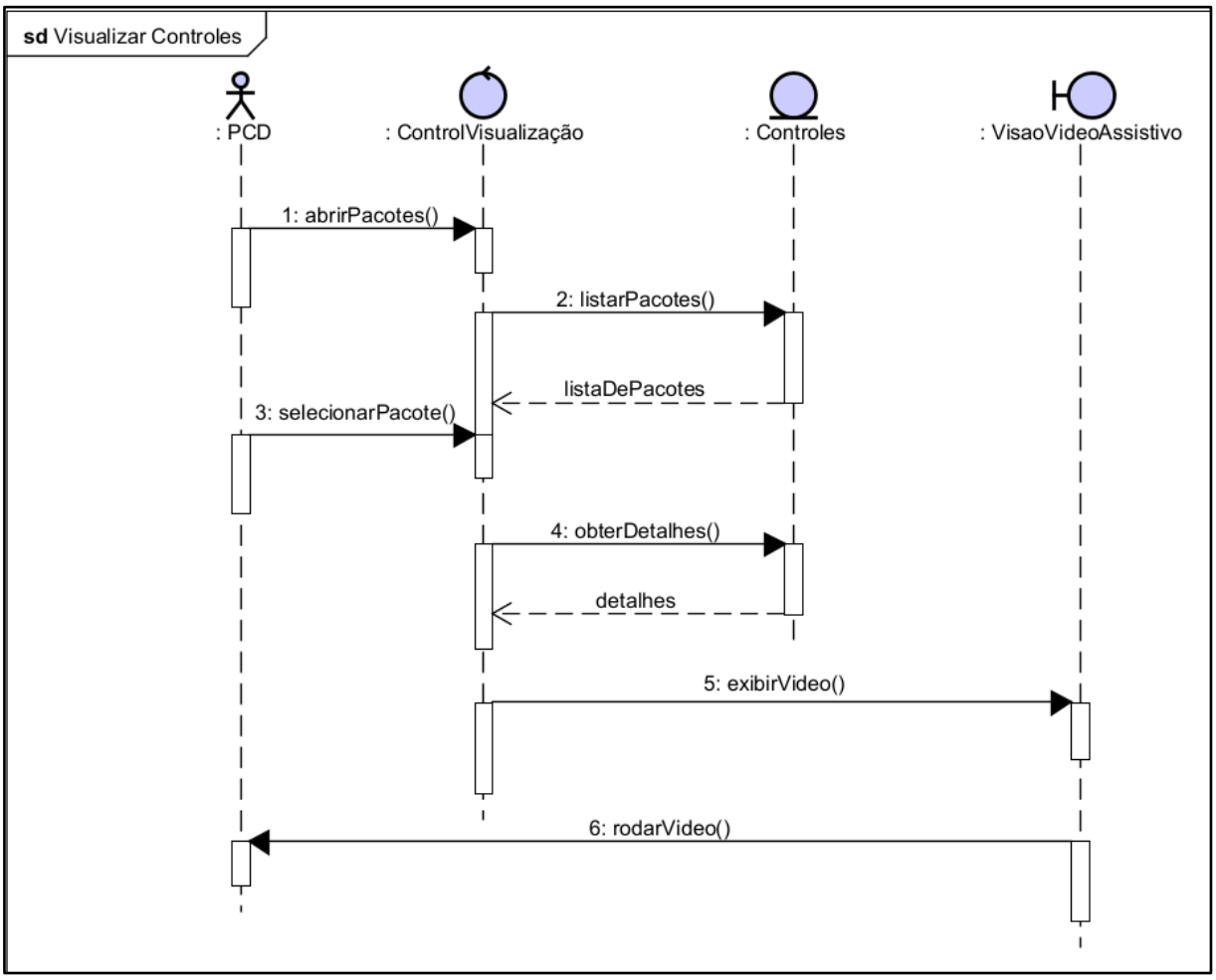

Fonte: Autoria Própria, 2025.

Figura 46 - Diagrama de Sequência: Configurar Preferências




Fonte: Autoria Própria, 2025.

Figura 47 - Diagrama de Sequência: Visualizar Controles

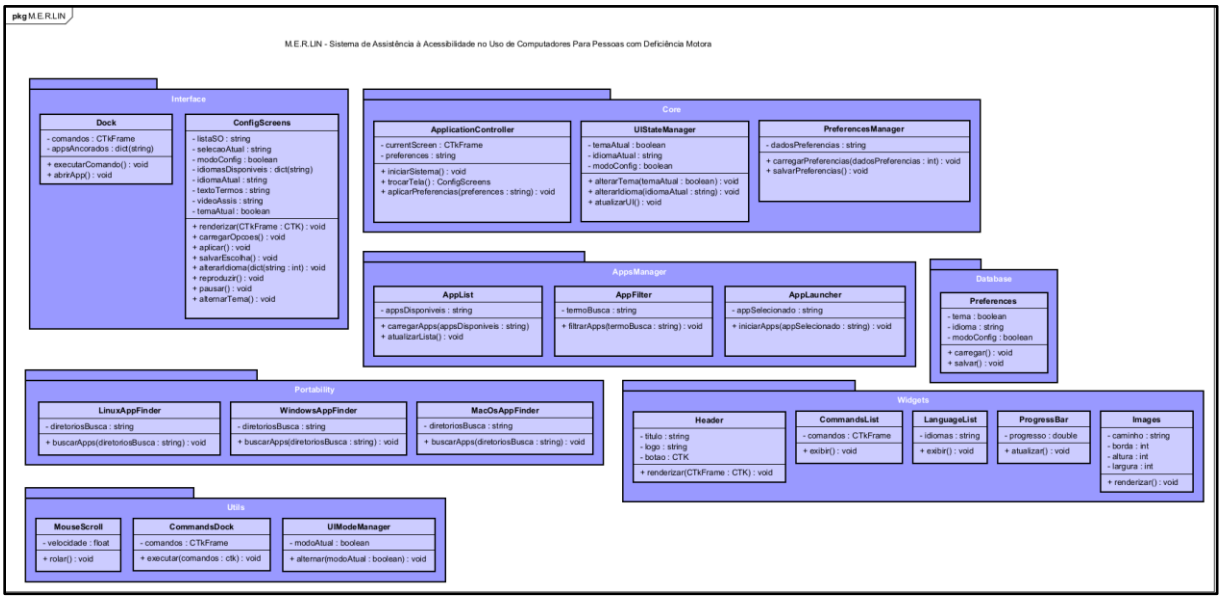



Fonte: Autoria Própria, 2025.

## 3.5 Diagrama de Classe

A seguir, podemos observar uma representação esquemática de como o código do projeto foi estruturado e organizado. Por meio deste diagrama, apresentamos de forma simplificada e visual o conteudo de cada classe do programa.

Figura 48 - Diagrama de Classe do M.E.R.LIN




Fonte: Autoria Própria, 2025.

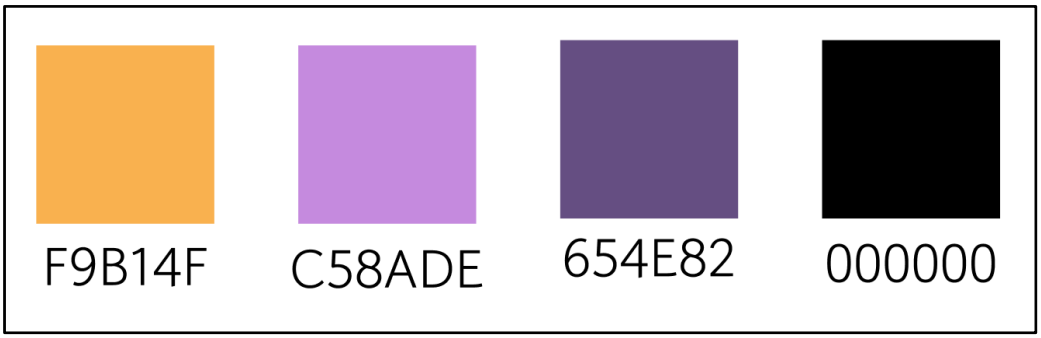
## 3.6 Marca

O presente capitulo irá apresentar as motivações por trás da marca do projeto M.E.R.LIN, abrangendo a escolha do nome, elaboração do logotipo e definição de cores e tipografia.

O nome “M.E.R.LIN” é uma referência as iniciais dos três integrantes, sendo M de Mota, E de Emily e R de Rodrigo. O sufixo “LIN” é um complemento utilizado para formar o nome do mago Merlin, personagem ficticio da cultura popular. O nome do projeto é uma alusão a ideia de que utilizar os olhos como controles é uma espécie de magia.

Se tratando de cores, o software possui quatro cores principais, dois tons de roxo, um tom de amarelo e um tom de preto.

Figura 49 - Cores do Software




Fonte: Autoria Própria, 2025.

O amarelo remete ao otimismo e a clareza, além de encorajar a comunicação. Essa cor é utilizada como um acabamento ou para adição de detalhes no software.

O roxo remete a imaginação, sabedoria, sendo animado ao mesmo tempo que acalma a mente. Essa é a principal cor do software, estando presente na logo, nas cores de fundo, detalhes e contrastes.

O preto remete a autoridade e poder, também evocando um mistério adicional ao conceito da marca. É amplamente utilizado nos textos do sistema e no acabamento da logo.

A logo do projeto é um olho com brilho amarelo complementando o contorno preto, com a íris do olho sendo dois tons de roxo que imitam chamas, formando discretamente o “M” presente no nome M.E.R.LIN.

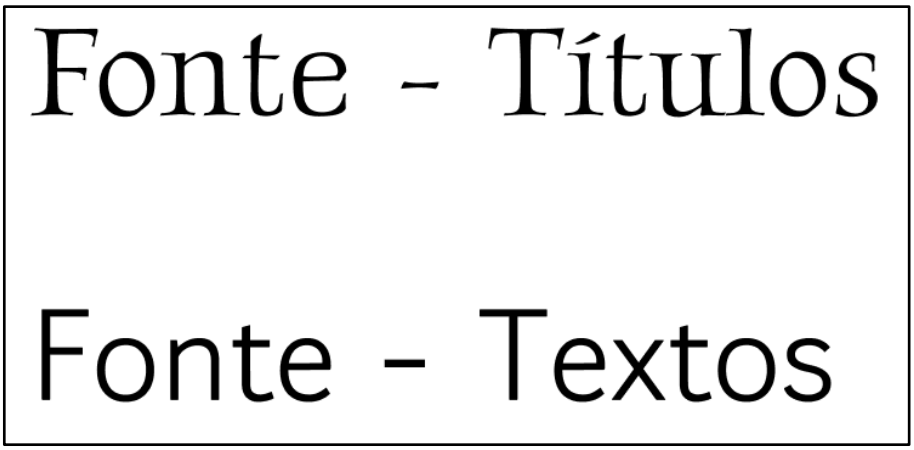
Figura 50 - Logo do M.E.R.LIN

Fonte: Autoria Própria, 2025.

A logo da marca possuí esse formato por remeter a principal característica do projeto, o controle ocular. Já as chamas são a representação do conceito de magia.

Para a tipografia do trabalho foram escolhidas duas fontes, sendo uma para os títulos e a outra para os textos. Nos títulos foi empregada a fonte “Gideon Roman”, enquanto que para os textos, a estilização ficou por conta da “Gowun Dodum”.

Figura 51 - Tipografia do M.E.R.LIN



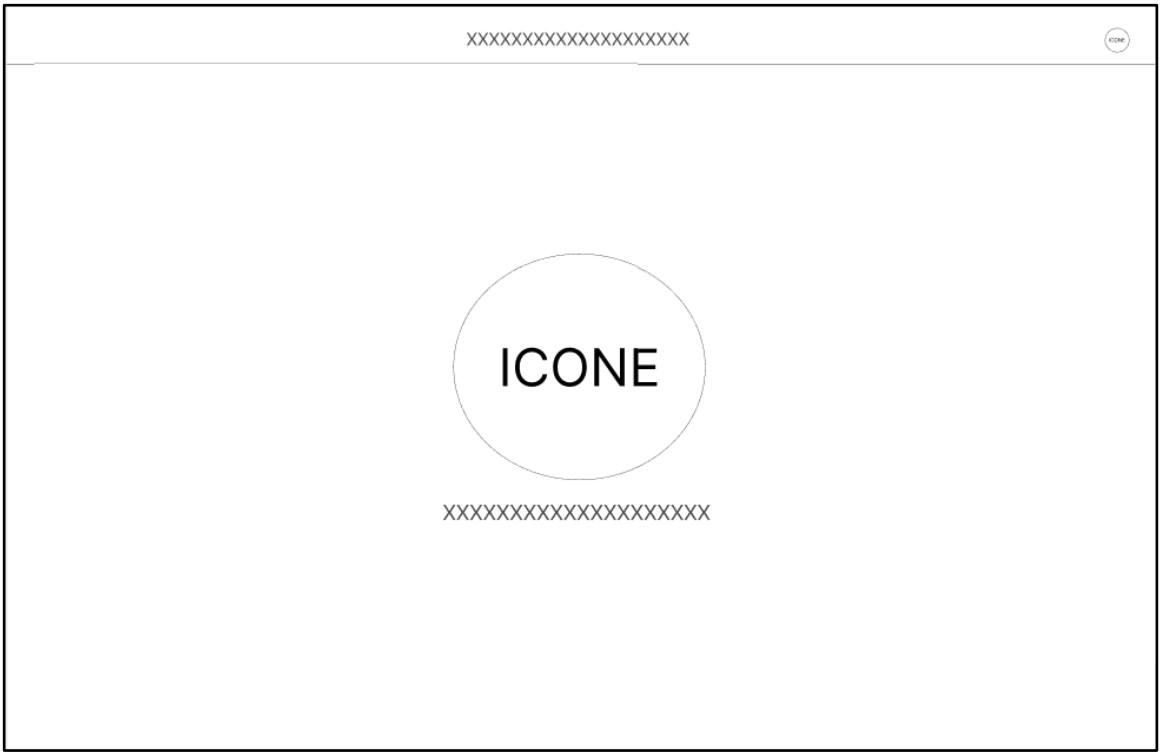

Fonte: Autoria Própria, 2025.

## 3.7 Prototipação das Interfaces do Sistema

Neste capítulo serão apresentadas as interfaces que compõem o sistema principal e o site expositivo do projeto, representadas por meio de Wireframes de baixa e alta fidelidade.

A primeira página do software é a Tela de Inicialização, a primeira a ser vista assim que o aplicativo é instalado na máquina. Em Wireframes de Baixa Fidelidade, é comum retratar textos com uma série de “X” e imagens como circulos vazios.

Figura 52 – Wireframe de Baixa Fidelidade do Software: Tela Inicial




Fonte: Autoria Própria, 2025.

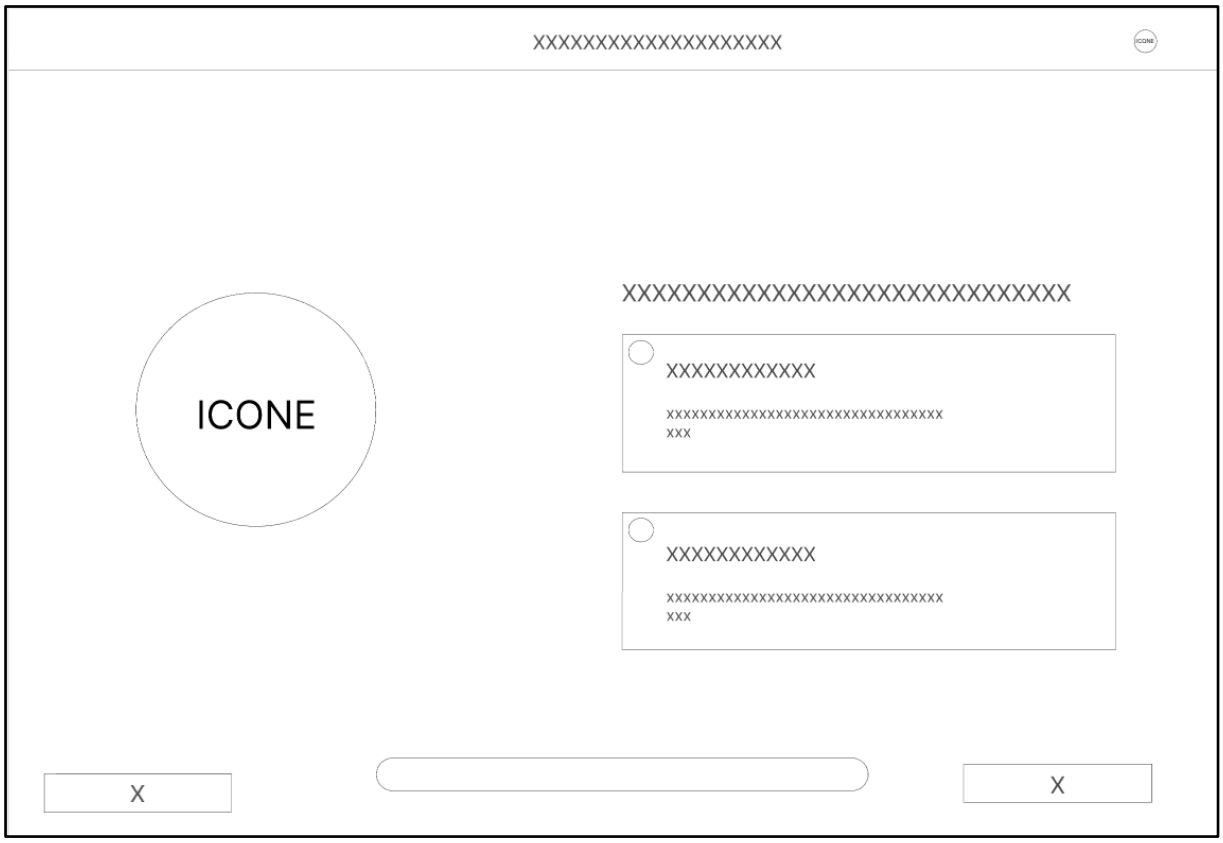
A seguir será apresentado o Wireframe de Alta Fidelidade da página de inicialização. Este padrão de apresentação será seguido pelas demais imagens do capitulo.

Figura 53 - Wireframe de Alta Fidelidade do Software: Tela Inicial


Fonte: Autoria Própria, 2025.

Após a Tela Inicial temos a tela de Seleção do Modo de Configuração, onde o usuário pode optar por fazer uma configuração rápida ou personalizada do software.

Figura 54 – Wireframe de Baixa Fidelidade do Software: Tela Modo de Configuração



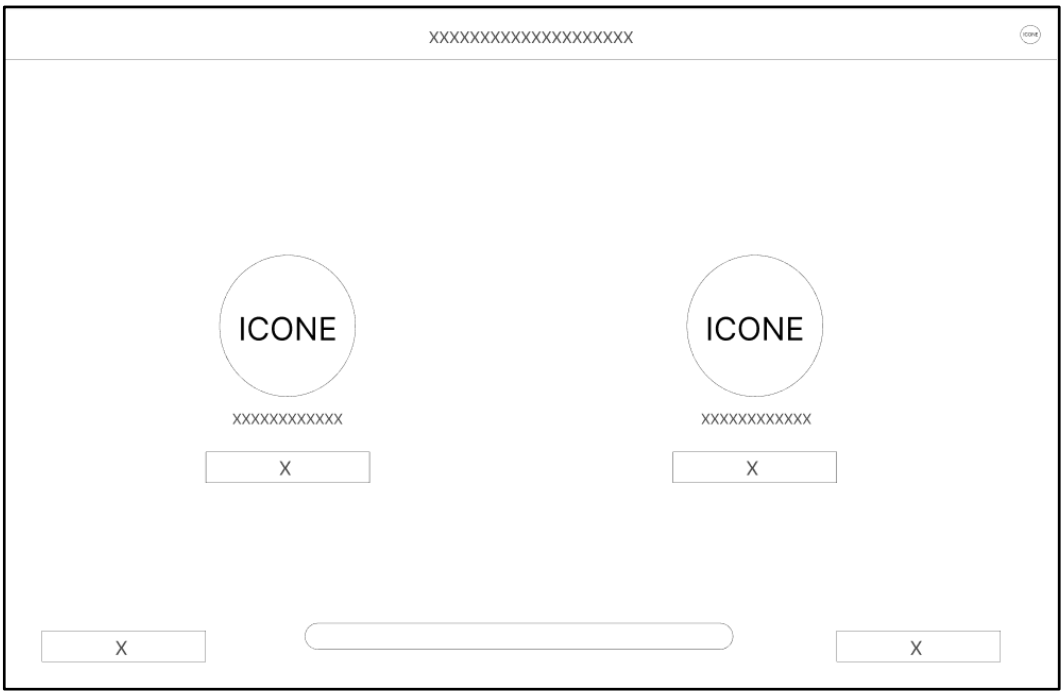

Fonte: Autoria Própria, 2025.

Figura 55 – Wireframe de Alta Fidelidade do Software: Tela Modo de Configuração


Fonte: Autoria Própria, 2025.

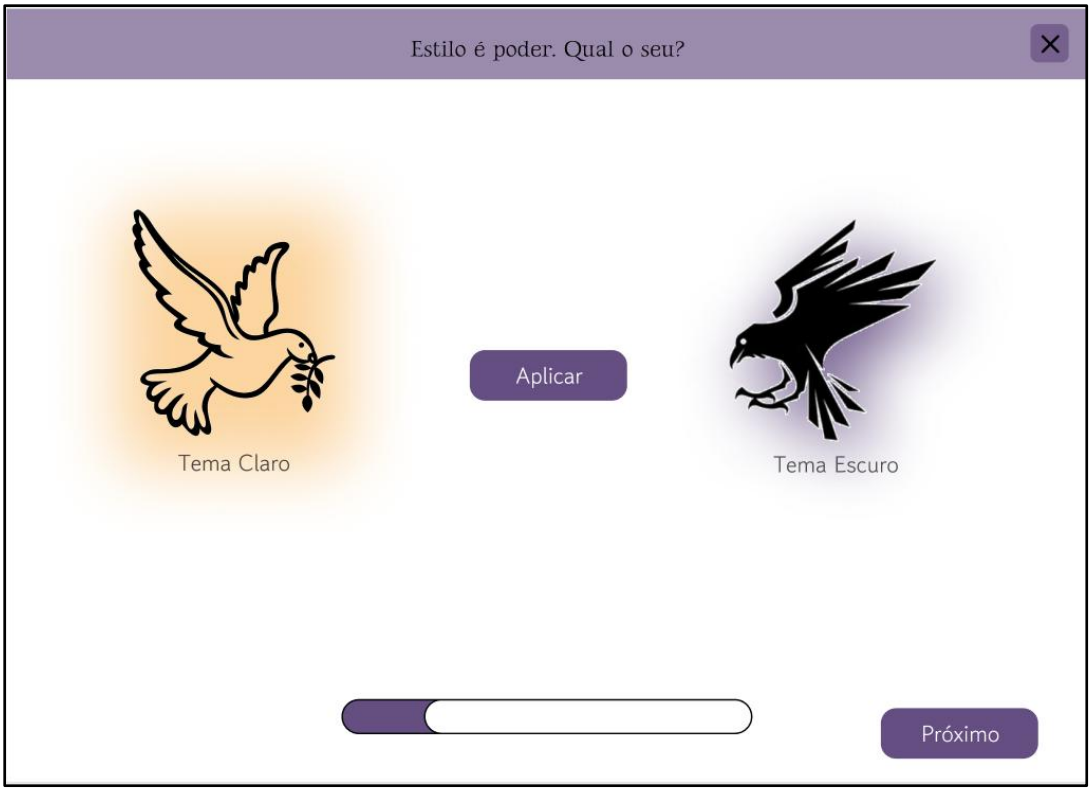
Caso a escolha do usuário tenha sido a configuração rápida, a configuração do software é encerrada por aqui. Se a escolha tiver sido a personalizada, haverá um redirecionamento para a página de Escolha do Tema, onde será possível decidir entre o padrão de cores escuras ou claras.

Figura 56 – Wireframe de Baixa Fidelidade do Software: Tela de Temas




Fonte: Autoria Própria, 2025.

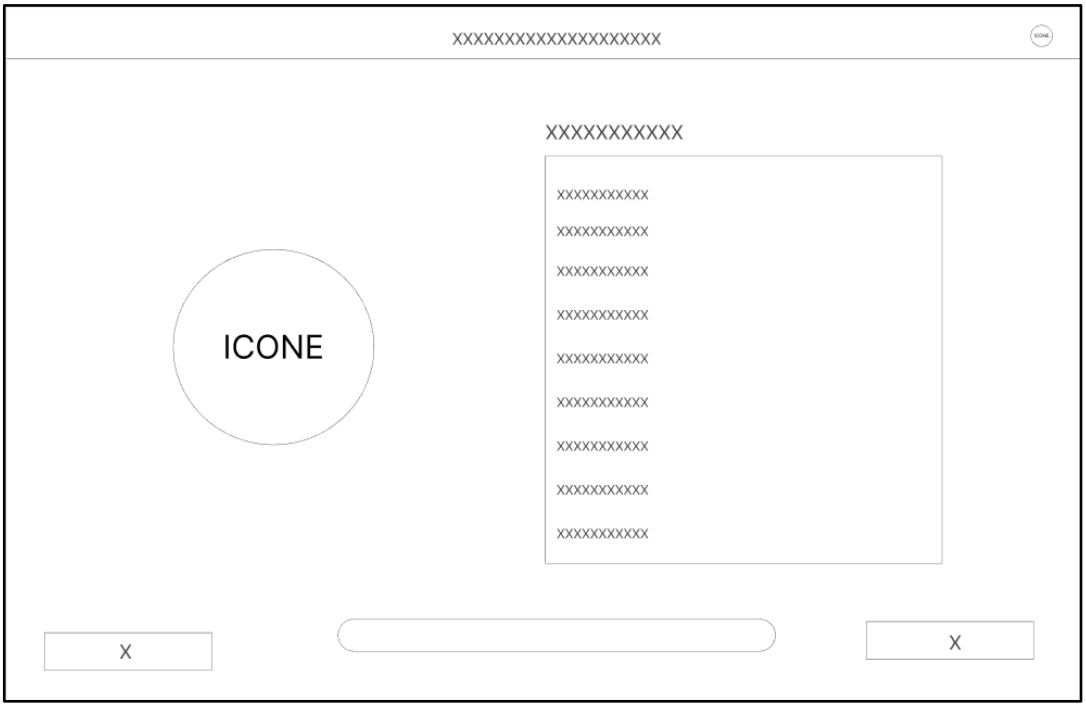
Figura 57 – Wireframe de Alta Fidelidade do Software: Tela de Temas




Fonte: Autoria Própria, 2025.

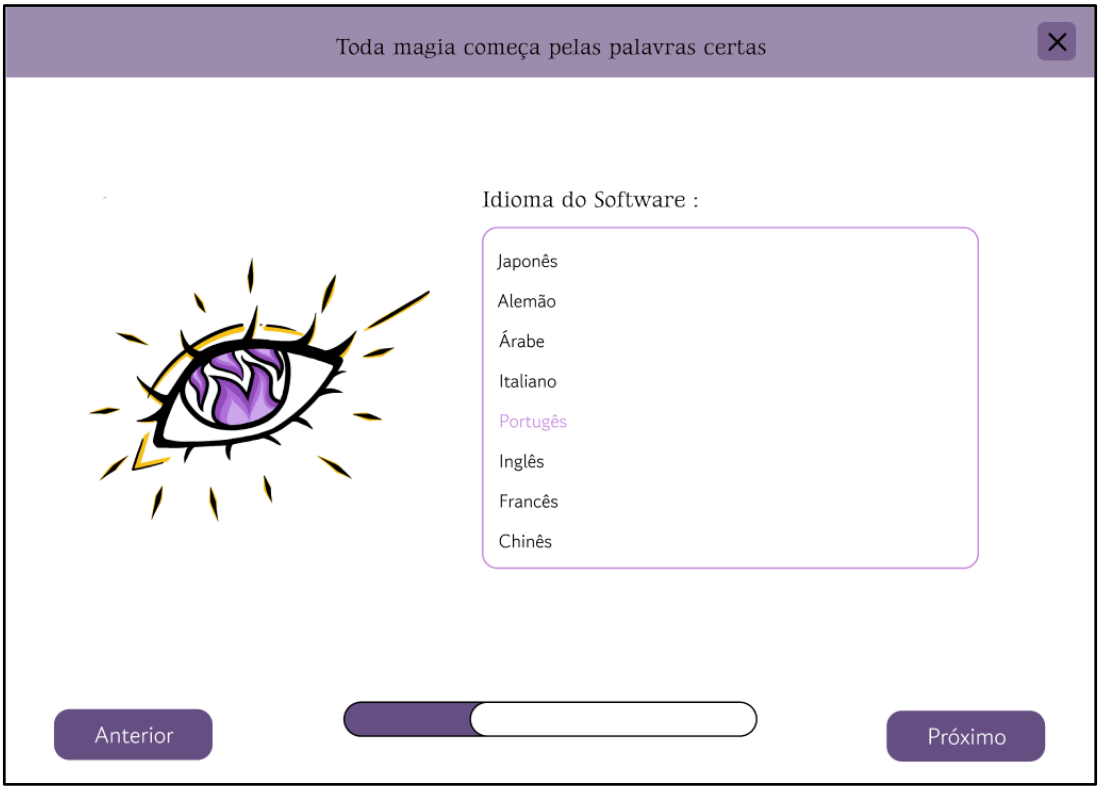
Após essa tela, o usuário poderá escolher o idioma da aplicação utilizando a Tela de Idiomas.

Figura 58 – Wireframe de Baixa Fidelidade do Software: Tela de Idiomas




Fonte: Autoria Própria, 2025.

Figura 59 – Wireframe de Alta Fidelidade do Software: Tela de Idiomas




Fonte: Autoria Própria, 2025.

Na próxima tela é possível fazer alterações nos ajustes da câmera, incluindo qualidade, FPS (frames por segundo) e se a luz de alerta da câmera ficará ativa durante o uso do M.E.R.LIN.

Figura 60 – Wireframe de Baixa Fidelidade do Software: Tela de Ajustes


Fonte: Autoria Própria, 2025.

Figura 61 – Wireframe de Alta Fidelidade do Software: Tela de Ajustes


Fonte: Autoria Própria, 2025.

Para finalizar a configuração, temos a Tela de Coletâneas, onde o usuário pode definir um único comando para multiplas ações de uma vez só. Na próxima imagem será demonstrado como é a visualização dessas coletâneas.

Figura 62 – Wireframe de Alta Fidelidade do Software: Tela de Coletâneas


Fonte: Autoria Própria, 2025.

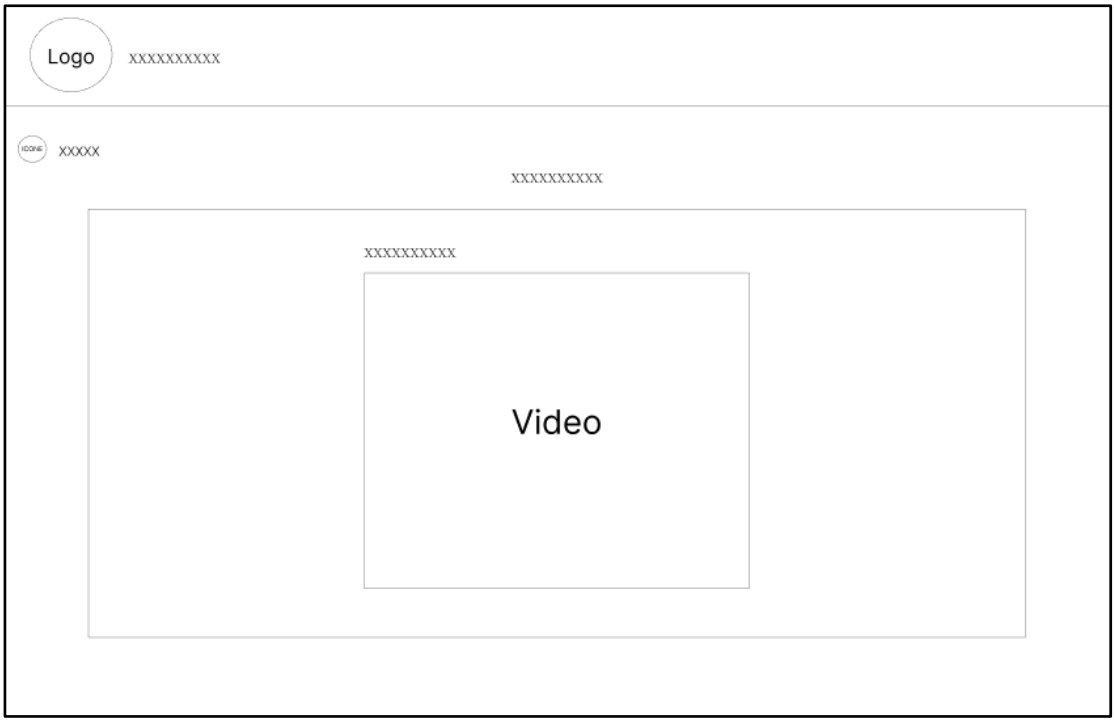
Cada coletânea possui seus comandos internos, permitindo que com apenas um gesto multiplos programas possam ser iniciados de uma vez.

Figura 63 – Wireframe de Alta Fidelidade do Software: Tela Comandos da Coletânea


Fonte: Autoria Própria, 2025.

Para aumentar a adequação dos usuários ao M.E.R.LIN, é disponibilizado no software um vídeo demonstrativo de como comandar o sistema com os movimentos faciais, bem como o resultado esperado de determinado gesto.

Figura 64 – Wireframe de Baixa Fidelidade do Software: Tela de Vídeo Assistência




Fonte: Autoria Própria, 2025.

Figura 65 – Wireframe de Alta Fidelidade do Software: Tela de Vídeo Assistência


Fonte: Autoria Própria, 2025.

Após todo o processo de configuração do software, o sistema passa a se resumir ao componente a seguir.

Figura 66 – Wireframe de Alta Fidelidade do Software: Dock do Sistema

Fonte: Autoria Própria, 2025.

Esse componente do software é primordial para a navegação do usuário PCD pelo sistema de seu computador. Por meio dele é possível executar o Teclado Integrado do M.E.R.LIN (utilizado para a digitação via comandos oculares), a Lista de Apps (utilizada para abrir qualquer programa existente no sistema do usuário), a coletânea configurada e um modo de navegação baseado em comandos oculares, próprio para navegadores web.

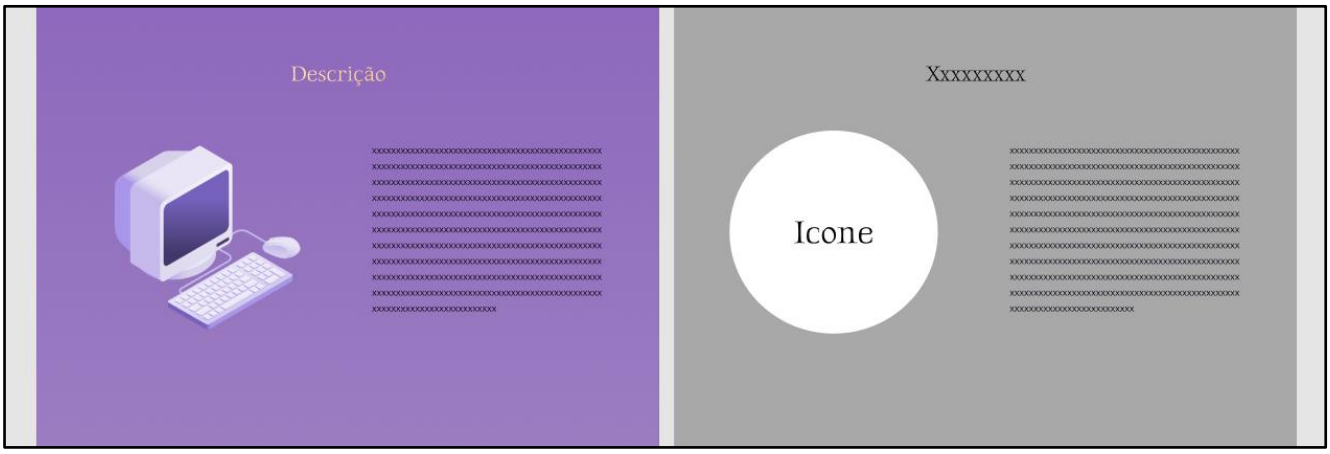
Para fins de divulgação, obtenção e esclarecimento sobre o projeto, foi desenvolvido para ele um site expositivo (também chamado de WebVitrine). A fim de documenta-lo, a seguir serão apresentadas por meio de Wireframes, suas respectivas seções.

Figura 67 – Wireframes de Alta e Baixa Fidelidade do Site: Seção Home


Fonte: Autoria Própria, 2025.

O primeiro contato ao acessar o site é com a Seção Home, ela agrupa os links que levam o usuário diretamente para a seção desejada.

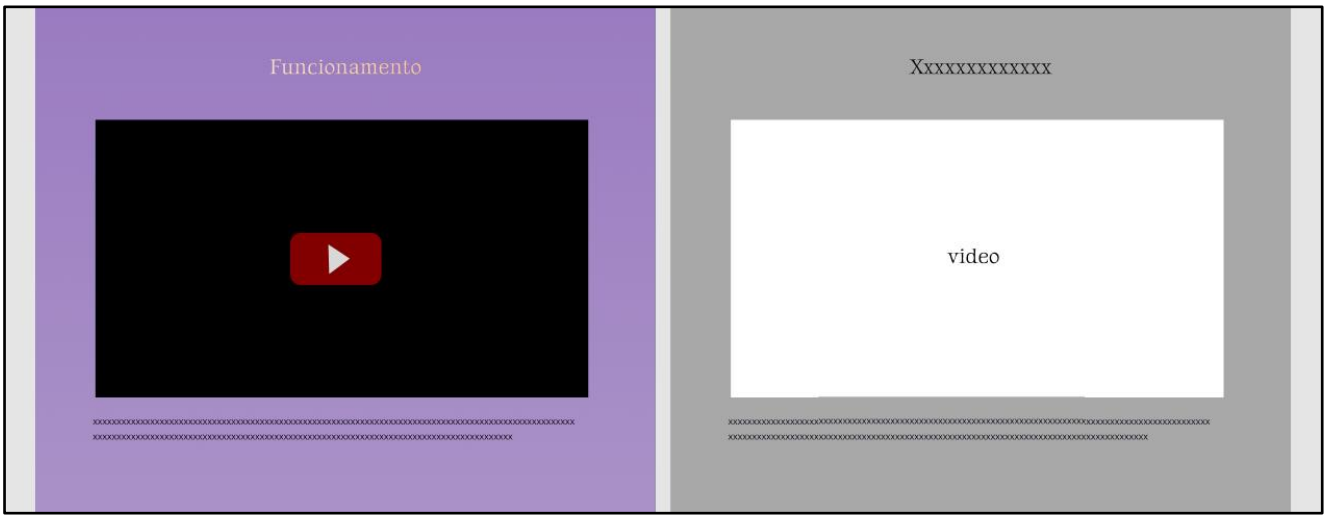
Figura 68 – Wireframes de Alta e Baixa Fidelidade do Site: Seção Sobre




Fonte: Autoria Própria, 2025.

Na Seção Sobre é descrito o propósito do software, contando de forma lúdica as possibilidades que o projeto fornece.

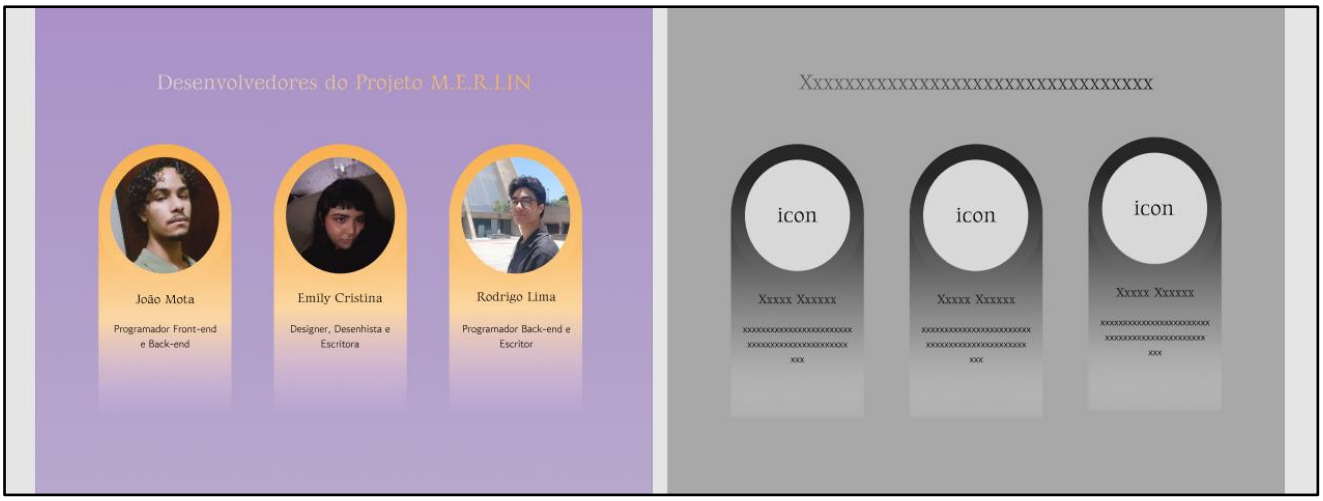
Figura 69 – Wireframes de Alta e Baixa Fidelidade do Site: Seção Demonstração




Fonte: Autoria Própria, 2025.

Na Seção Demonstração é possível acessar um vídeo que demonstra todo o projeto de forma prática, assim todos que navegarem pelo site podem ter uma clara noção do alcance do M.E.R.LIN.

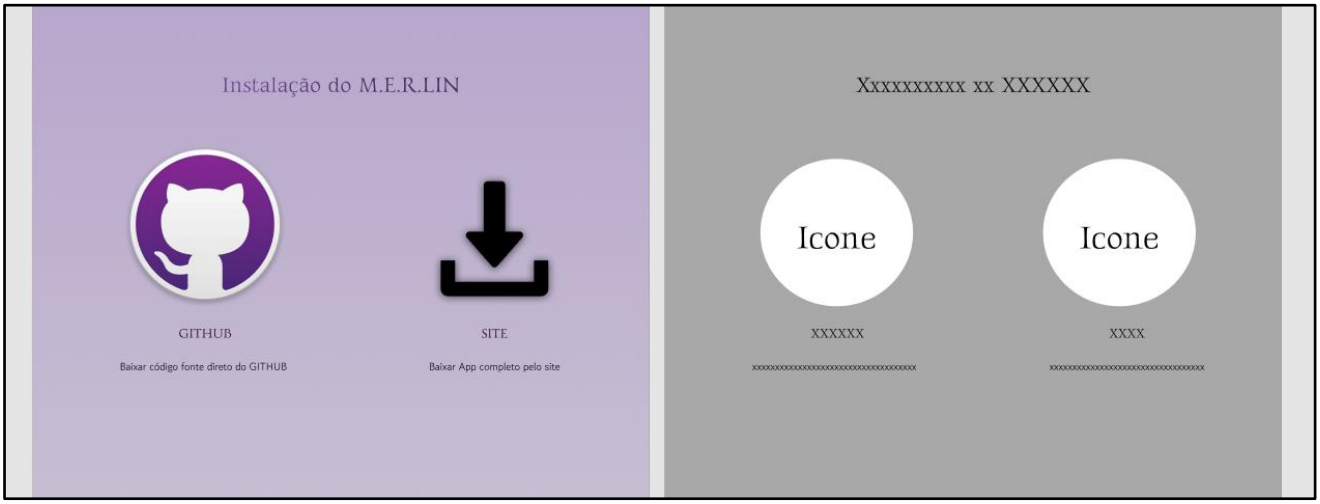
Figura 70 – Wireframes de Alta e Baixa Fidelidade do Site: Seção Desenvolvedores




Fonte: Autoria Própria, 2025.

A Seção Desenvolvedores serve para expor um pouco dos integrantes do grupo. Clicando em cada card de desenvolvedor é possível acessar suas redes sociais, a cada clique, uma nova rede social é aberta. Dentre as opções que o usuário pode sortear estão o GitHub(portfólio para programadores), Instagram e LinkedIn.

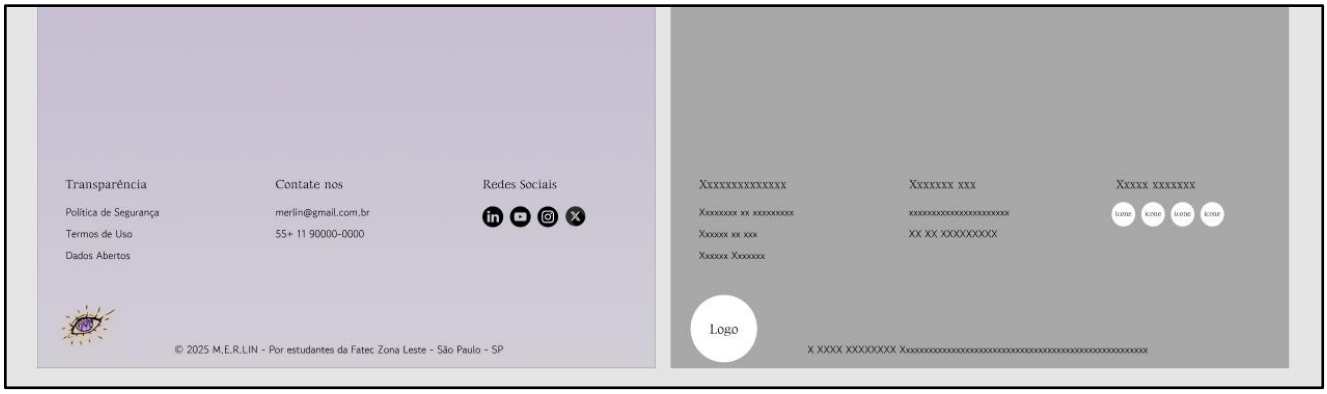
Figura 71 – Wireframes de Alta e Baixa Fidelidade do Site: Seção Download




Fonte: Autoria Própria, 2025.

As seções finais do site são Download e Footer, respectivamente responsáveis por permitirem a obtenção da versão atual do projeto por meio de um download ou acesso ao código e por agrupar links das redes sociais de todos os membros, informações de contato ficticias e links para as apresentações do M.E.R.LIN realizadas ao longo do ano.

Figura 72 – Wireframes de Alta e Baixa Fidelidade do Software: Seção Footer




Fonte: Autoria Própria, 2025.

# 4 CONSIDERAÇÕES FINAIS

Baseado em pesquisas quantitativas obteve-se a análise das necessidades e fragilidades do mercado, com isso esperava-se a entrega de um software assistivo que pudesse por meio de reconhecimento facial mitigar as comprovações mencionadas.

Foi constatado durante a finalização do projeto M.E.R.LIN que este possui potencial para fomentar interações sociais e a adaptação ao mercado de trabalho, uma vez que, sua proposta de software assistivo inicialmente sugerida conseguiu ser atendida por completo.

Assim, permitindo o foco no aprimoramento da usabilidade e no alcance do projeto, por meio da Integração com navegadores web e sugestões durante o uso do teclado. Ademais, pretende-se expandir o M.E.R.LIN para suportar dispositivos móveis, incluindo assim novos potenciais usuários.

## REFERÊNCIAS

RODRIGUES, P. S.; PEREIRA, E. L. **A Percepção das Pessoas com Deficiência Sobre o Trabalho e a Lei de Cotas: Uma Revisão da Literatura**, v.31, p,. 15 nov. 2021.

MARCONI, M. A.; LAKATOS, E. M. **Fundamentos de Metodologia Científica.** 8. ed. São Paulo: Atlas, 2017.

AGÊNCIA IBGE NOTÍCIAS. PNAD contínua. **Pessoas com deficiência têm menor acesso à educação, ao trabalho e à renda.** 2024. Disponível em: https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/37317-pessoas-com-deficiencia-tem-menor-acesso-a-educacao-ao-trabalho-e-a-renda. Acesso em: 23 de março de 2025, 14:29.

G1. **7 em cada 10 pessoas com deficiência estão fora do mercado de trabalho; salário médio dessa população é R$ 1 mil menor, diz IBGE**. 2024. Disponível em: https://g1.globo.com/economia/noticia/2022/09/21/7-em-cada-10-pessoas-com-deficiencia-estao-fora-do-mercado-de-trabalho-salario-medio-dessa-populacao-e-r-1-mil-menor-diz-ibge.ghtml. Acesso em: 23 mar. 2025, 14:29.

BANIN, Sérgio Luiz. **Python 3: Conceitos e Aplicações – Uma Abordagem Prática.** 1. ed. São Paulo: Saraiva Educação, 2018.

MATTHES, Eric. **Curso Intensivo de Python: Uma Introdução Prática e Baseada em Projetos à Programação.** São Paulo: Novatec Editora, 2016.

SWEIGART, Al. **Automatize Tarefas Maçantes com Python.** São Paulo: Novatec, 2015.

PYTHON SOFTWARE FOUNDATION. **Python Packaging User Guide.** Disponível em: https://packaging.python.org/pt-br/latest/overview/?. Acesso em: 17 ago. 2025.

FIGUEIREDO, Alexsandro Leite. **Desenvolvimento de uma API baseada no padrão Redfish para monitoramento remoto de Raspberry Pi.** Salvador: Universidade Federal da Bahia – Instituto de Computação, Bacharelado em Ciência da Computação, 2023.

ESPERANÇA, Claudio. **Python: Interfaces Gráficas com Tk.** 2011. Apresentação de aula — Universidade Federal do Rio de Janeiro. Disponível em: https://ic.ufrj.br/~fabiom/mab225/. Acesso em: 2 set. 2025.

CAMARGO, Arthur Adabo de; EGGERS, Pedro Alvares. **Desenvolvimento de um aplicativo de propriedades de compostos com aroma.** São Paulo: Universidade de São Paulo – Escola Politécnica, 2020.

PYTHON SOFTWARE FOUNDATION. **Documentação do Tkinter.** Disponível em: https://docs.python.org/pt-br/3/library/tkinter.html?. Acesso em: 17 ago. 2025.

FROTA, H. S.; RUVER, F. S.; FIGUEIREDO, J. M. **Implementação de software desktop em Python.** Cuiabá, MT: Universidade Federal de Mato Grosso – UFMT, 2024.

BARELLI, Felipe. **Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV.** São Paulo: Casa do Código, 2018.

BACKES, André Ricardo; SÁ JUNIOR, Jarbas Joaci de Mesquita. **Introdução à visão computacional usando MATLAB**. 1. ed. Rio de Janeiro: Alta Books, 2016.

CASTRO, Mariana Oleone Marinho de**. Estudo de visão computacional e criação de um Haar Cascade utilizando a biblioteca OpenCV em Python para detecção de objetos.** Lorena: Universidade de São Paulo – Escola de Engenharia de Lorena, 2021.

MARENGONI, Maurício; STRINGHINI, Denise. **Tutorial: introdução à visão computacional usando OpenCV.** Revista de Informática Teórica e Aplicada, 2010.

SILVA, Karolyne Pereira da. **Análise de aplicação de visão computacional e redes neurais, em conjunto com o uso de técnicas de aumento de dados, na tradução automática de Libras.** Porto Alegre: Universidade Federal do Rio Grande do Sul, Escola de Engenharia – Engenharia de Controle e Automação, 2023.

SCHIRMER, Ricardo Dias. **Projeto e implementação de um robô de precisão com visão computacional baseada em algoritmos de inteligência artificial.** Santa Maria: Universidade Federal de Santa Maria – Centro de Tecnologia, Curso de Engenharia de Controle e Automação, 2023.

GOOGLE. MediaPipe Solutions Guide. Disponível em: https://ai.google.dev/edge/mediapipe/solutions/guide?. Acesso em: 17 ago. 2025.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados.** 6. ed. São Paulo: Pearson Education, 2011.

BRITO, Ricardo W. **Bancos de dados NoSQL x SGBDs relacionais: análise comparativa.** Fortaleza: Faculdade Farias Brito; Universidade de Fortaleza, 2010.

ALURA. **SQLite: da instalação até sua primeira tabela.** Disponível em:

https://www.alura.com.br/artigos/sqlite-da-instalacao-ate-primeira-tabela . Acesso em: 17 ago. 2025.

COMACHIO, Vanderson. **Funcionamento de banco de dados em Android: um estudo experimental utilizando SQLite.** Curitiba: Universidade Tecnológica Federal do Paraná – UTFPR, Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, 2011.

NEGUS, Christopher; com a colaboração de Christine Bresnahan. **Linux: a Bíblia.** 8. ed. Rio de Janeiro: Alta Books, 2014.

NEVES, Julio Cezar. **Programação Shell Linux.** 8. ed. Rio de Janeiro: Brasport, 2010.

JARGAS, Aurélio Marinho. **Shell Script Profissional.** 1. ed. São Paulo: Novatec, 2008.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: Guia do Usuário.** 2. ed. Rio de Janeiro: Elsevier, 2012.

GUEDES, Gilleanes T. A. **UML 2: Uma Abordagem Prática.** 3. ed. São Paulo: Novatec, 2018.

FOWLER, Martin. **UML Essencial: Um Breve Guia para a Linguagem-Padrão de Modelagem de Objetos.** 3. ed. Porto Alegre: Bookman, 2005.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software: Uma Abordagem Profissional.** 8. ed. Porto Alegre: AMGH, 2016.

GRILO, André. **Experiência do usuário em interfaces digitais.** Natal: SEDIS-UFRN, 2019.

TEIXEIRA, Fabricio. **Introdução e boas práticas em UX Design.** São Paulo: Casa do Código, 2014.

MIRO. **Os 10 melhores templates gratuitos de wireframe para acelerar o design.** Disponível em: https://miro.com/pt/modelos/wireframe/ . Acesso em: 17 ago. 2025.

GODADDY. **O que é wireframe? Entenda tudo sobre esse recurso de UX!** Disponível em: https://www.godaddy.com/resources/br/artigos/o-que-e-wireframe Acesso em: 17 ago. 2025.

OLIVEIRA, George Moreno de. **Desenvolvimento e avaliação do plugin para o Figma para documentação de acessibilidade para interfaces – DAI.** Quixadá: Universidade Federal do Ceará – Campus de Quixadá, Curso de Graduação em Design Digital, 2022.

ALURA. **Entenda o Figma: uma solução inovadora para projetos de design.** 2022. Disponível em: https://www.alura.com.br/artigos/figma-uma-solucao-inovadora-projetos-design. Acesso em: 7 ago. 2025.

FERREIRA, Silvio. **Guia prático de HTML 5**. 1. ed. São Paulo: Universo dos Livros Editora LTDA, 2013. E-book.

SILVA, Maurício Samy. **Fundamentos de HTML5 e CSS3**. 1. ed. São Paulo: Novatec, 2015.

MAZZA, Lucas. **HTML5 e CSS3: Domine a Web do Futuro**. 1. ed. São Paulo: Casa do Código, 2014. E-book.

FREEMAN, Eric; FREEMAN, Elisabeth. **Use a Cabeça! HTML com CSS e XHTML**. 2. ed. Rio de Janeiro: Alta Books, 2009.

FLANAGAN, David. **JavaScript: o Guia Definitivo**. 6. ed. Porto Alegre: Bookman, 2013.

ZAKAS, Nicholas C. **JavaScript de Alto Desempenho**. 1. ed. São Paulo: Novatec, 2010.