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

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 projeto contém a documentação escrita de um software que auxilia PCDs no uso de computadores e desktops, além de um website vitrine 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, foi possível analisar informações que possibilitaram medir o 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: PCD’s, Deficiências Motoras, Músculos Mímicos, Software, 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: PCDs, Motor Disabilities, Mimic Muscles, Software, Job Market.

LISTA DE ILUSTRAÇÕES

Figura 1 - Exemplo De Código Em Python Fonte 20

Figura 2 - Resultado Do Código Em Python 21

Figura 3 - Exemplo De Instalação Por Meio De Pip 22

Figura 4 - Exemplo De Subprocess 22

Figura 5 - Resultado Exemplo Subprocess 23

Figura 6 - Exemplo De Código Utilizando Pyautogui 24

Figura 7 - Exemplo De Uma Aplicação Usando Tkinter 26

Figura 8 - Resultado Do Código Em Tkinter Fonte. 28

Figura 9 - Exemplo Da Utilização Do Customtkinter 29

Figura 10 - Resultado Do Código De Customtkinter 30

Figura 11 - Exemplo Da Instalação Do Opencv Por Meio Do Pip 31

Figura 12 - Exemplo De Um Código Python Com Opencv. 32

Figura 13 - Resultado Do Código Em Python Com Opencv 33

Figura 14 - Exemplo De Um Código Python Com Mediapipe 34

Figura 15 - Resultado Do Código Python Com Mediapipe 36

Figura 16 - Exemplo De Um Código Python Utilizando O Sqlite 38

Figura 17 - Exemplo De Script No Interpretador 40

Figura 18 - Exemplo De Interface No Linux 41

Figura 19 - Exemplo De Diagrama De Caso De Uso 42

Figura 20 - Exemplo De Documentação Do Caso De Uso 43

Figura 21 - Exemplo De Diagrama De Atividade 44

Figura 22 - Exemplo De Diagrama De Sequência 45

Figura 23 - Exemplo De Diagrama De Classe 46

Figura 24 - Exemplo De Wireframes De Alta E Baixa Fidelidade 47

Figura 25 - Exemplo De Interface Para Uso Do Figma 48
Figura 26 - Caso De Uso M.E.R.Lin 57
Figura 27 - Cores Do Software 63
Figura 28 - Logo M.E.R.LIN 64
Figura 29 - Fontes Do M.E.R.LIN 65
Figura 30 - Página De Inicialização Software Baixa Qualidade 66
Figura 31 - Página De Inicialização Software Alta Qualidade 67
Figura 32 - Modo De Configuração De Baixa Qualidade 68
Figura 33 - Modo De Configuração De Alta Qualidade 69
Figura 34 - Tema Do Software Baixa Qualidade 70
Figura 35 - Tema Do Software Alta Qualidade 70
Figura 36 - Escolha Do Idioma Baixa Qualidade 71
Figura 37 - Escolha De Idioma Alta Qualidade 71
Figura 38 - Ajustes De Configuração Baixa Qualidade 72
Figura 39 - Ajustes De Configuração Alta Qualidade 73
Figura 40 - Configuração Das Coletâneas Alta Qualidade 74
Figura 41 - Configuração Dos Comandos Da Coletânea Alta Qualidade 75
Figura 42 - Vídeo De Assistência Baixa Qualidade 76
Figura 43 - Vídeo Assistência Alta Qualidade 76
Figura 44 - Dock Baixa Qualidade 77
Figura 45 - Home Do Site Alta E Baixa Qualidade 77
Figura 46 - Descrição Do Site Alta E Baixa Qualidade 78
Figura 47 - Funcionamento Site Baixa E Alta Qualidade 78
Figura 48 - Desenvolvedor Site Baixa E Alta Qualidade 79
Figura 49 - Instalação Site Alta E Baixa Qualidade 79

Figura 50 - Footer Do Site Alta E Baixa Qualidade 80

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

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 16

2 REFERENCIAL TEÓRICO 19

2.1 Deficiência motora no Brasil 19

2.1.1 Desafios no Cenário Trabalhista 19

2.2 Tecnologias 20

2.2.1 Python 20

2.2.2 Bibliotecas do Python 22

2.2.2.1 Bibliotecas Gráficas 25

2.2.3 Reconhecimento Ocular 31

2.2.4 Banco de dados 36

2.2.5 Shell Script 39

2.2.6 Unified Model Language (UML) 41

2.2.6.1 Diagrama de Caso de Uso 42

2.2.6.2 Documentação do Caso de Uso 43

2.2.6.3 Diagrama de Atividade 44

2.2.6.4 Diagrama de Sequência 44

2.2.6.5 Diagrama de Classe 46

2.2.7 Prototipagem 46

3 DESENVOLVIMENTO 56

3.1 Diagrama de Caso de Uso 57

3.2 Documentação do Caso de Uso 57

3.3 Diagrama de Atividade 62

3.4 Diagrama de Sequência 63

3.5 Diagrama de Classe 63

3.6 Prototipação das interfaces do sistema 65

4 CONSIDERAÇÕES FINAIS 73

REFERÊNCIAS 81

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 PCD 's 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.

Explorando uma abordagem recreativa, permite-se também o uso de ferramentas no computador utilizadas para sociabilidade. 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 qualitativa, atentando-se a experiências 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

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.

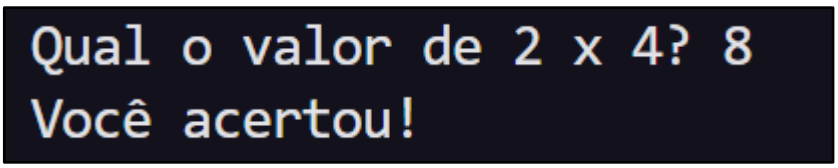
Linha 8: Esse comando chamado "print" é responsável por exibir mensagens para o usuário, no caso dessa linha, a mensagem é positiva.

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 de PIP

```
C:\Users\Rodrigo>pip install customtkinter
```

Fonte: Autoria Própria, 2025.

2.2.2 Bibliotecas do Python

Bibliotecas usadas para implementar, automatizar e facilitar o uso do Python para o funcionamento do projeto.

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 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.

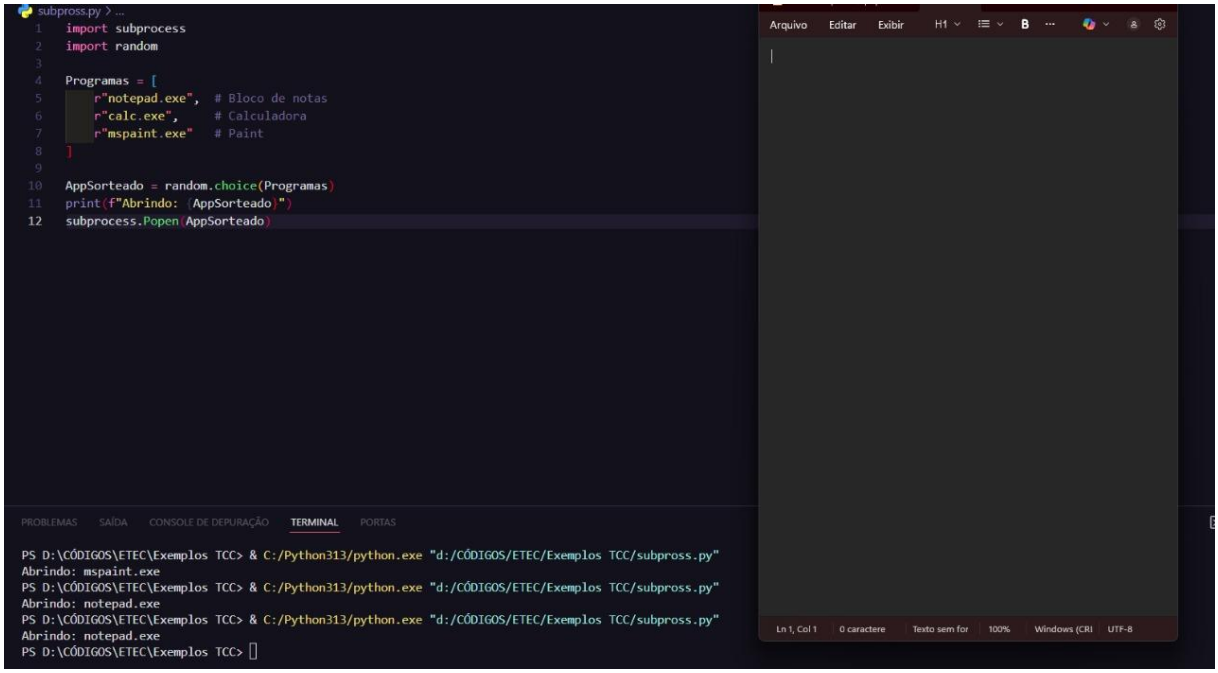
Linha 4 - 8: Uma lista do python que guarda programas do computador (ta especificado quem é quem).

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 exemplo 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

Bibliotecas usadas afim de melhorar a interface gráfica implementada ao Python.

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 uma aplicação usando 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.

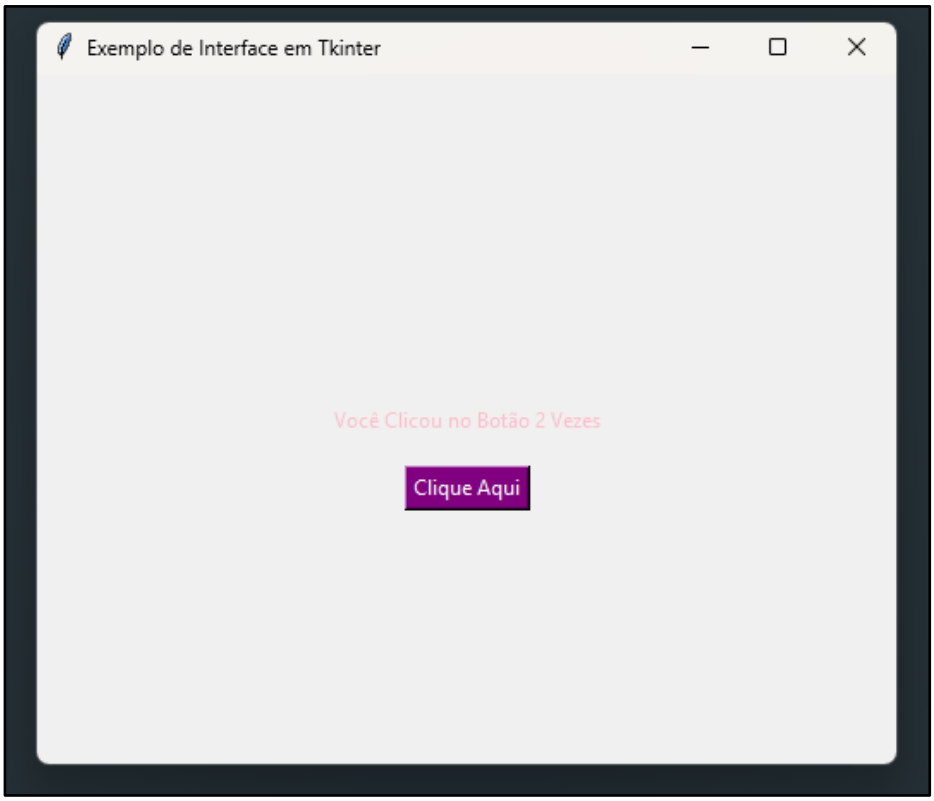
Linha 31: Define que o texto ficará centralizado seguindo o padrão dado pelos eixos atribuídos.

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 em Tkinter



Fonte: Autoria própria, 2025.

Afim de complementar e aprimorar o design deixando-o mais atual será utilizado Custom Tkinter.

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 da utilização do CustomTKinter

```
1  from customtkinter import *
2
3  Tela = CTk()
4
5  Tela.title ("Exemplo de Interface em CustomTk")
6
7  Tela.geometry ("500x400")
8
9  Cliques = 0
10
11 def aoClicar():
12    global Cliques
13    Cliques += 1
14
15    Texto.configure(text = f"Você Clicou no Botão {Cliques} Vezes ")
16
17 Texto = CTkLabel(master = Tela,
18                  text = "Você Clicou no Botão 0 Vezes",
19                  text_color = "pink")
20
21 Botao = CTkButton(master = Tela,
22                  text = "Clique Aqui",
23                    corner_radius = 30,
24                      fg_color = "purple",
25                        text_color = "white",
26                          hover_color = "magenta",
27                          command = aoClicar)
28
29 Texto.place(relx = 0.5,
30             rely = 0.5,
31             anchor = CENTER)
32
33 Botao.place(relx = 0.5,
34             rely = 0.6,
35             anchor = CENTER)
36
37 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.

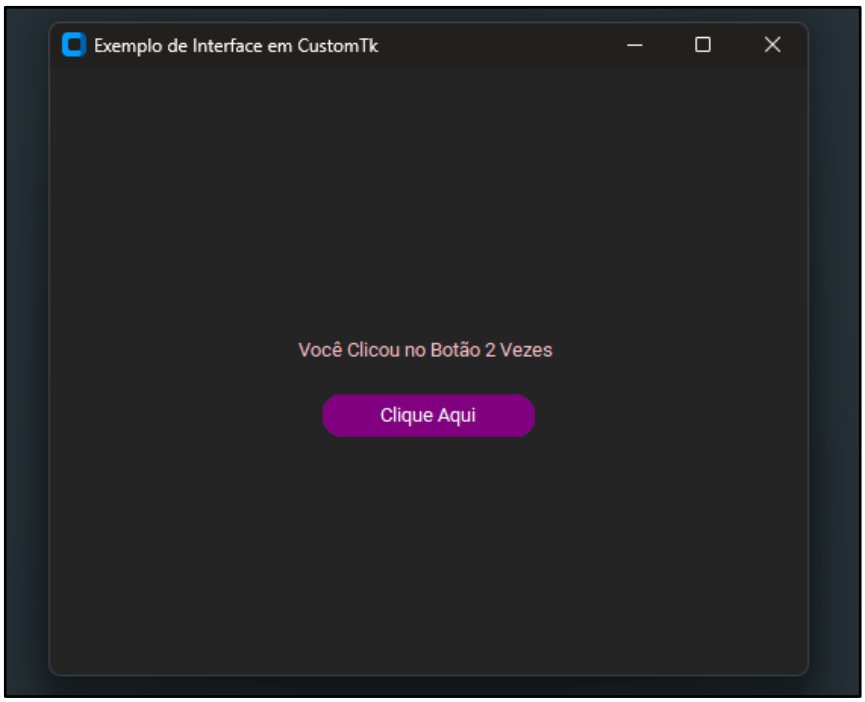
Linha 15: O objeto “Texto” recebe a propriedade padrão juntamente com outra específica “configure” para configurar o texto que exibirá após a ação de “Clique”.

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 de CustomTKinter



Fonte: Autoria Própria, 2025.

2.2.3 Reconhecimento Ocular

Tal como observa Barelli (2018), a visão computacional vem como grande ajuda para procurar, detectar e tratar informações, analisando-as como a visão humana podendo identificar características como: cor, textura e quantidade.

Como destaca Antonello (2018), a visão computacional vai além da simples leitura de imagens, ela é capaz de identificar padrões, possibilitando o desenvolvimento de sistemas mais complexos, como um detector de fungos em plantas voltado ao uso na agricultura.

Assim como afirma Backes et al (2016). A visão computacional pode ser utilizada em aplicações mais comuns e não voltadas exclusivamente ao ramo empresarial, como a criação de filtros para câmera e tratamento de imagem como redução de ruído.

O OpenCV executará funções e comandos que permitirá identificar partes específicas, assim ele será quem executa a Visão Computacional.

Como afirma Castro (2021), o OpenCV é uma biblioteca voltada ao desenvolvimento de aplicações de visão computacional, composta por módulos 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 métodos disponíveis na biblioteca mostra-se altamente eficaz para a manipulação de imagens, como, por exemplo, na leitura e interpretação de conteúdos visuais.

Figura 11 - Exemplo da Instalação do OpenCV por meio do PIP

Fonte: Autoria própria, 2025.

Figura 12 - Exemplo de um código Python com 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.

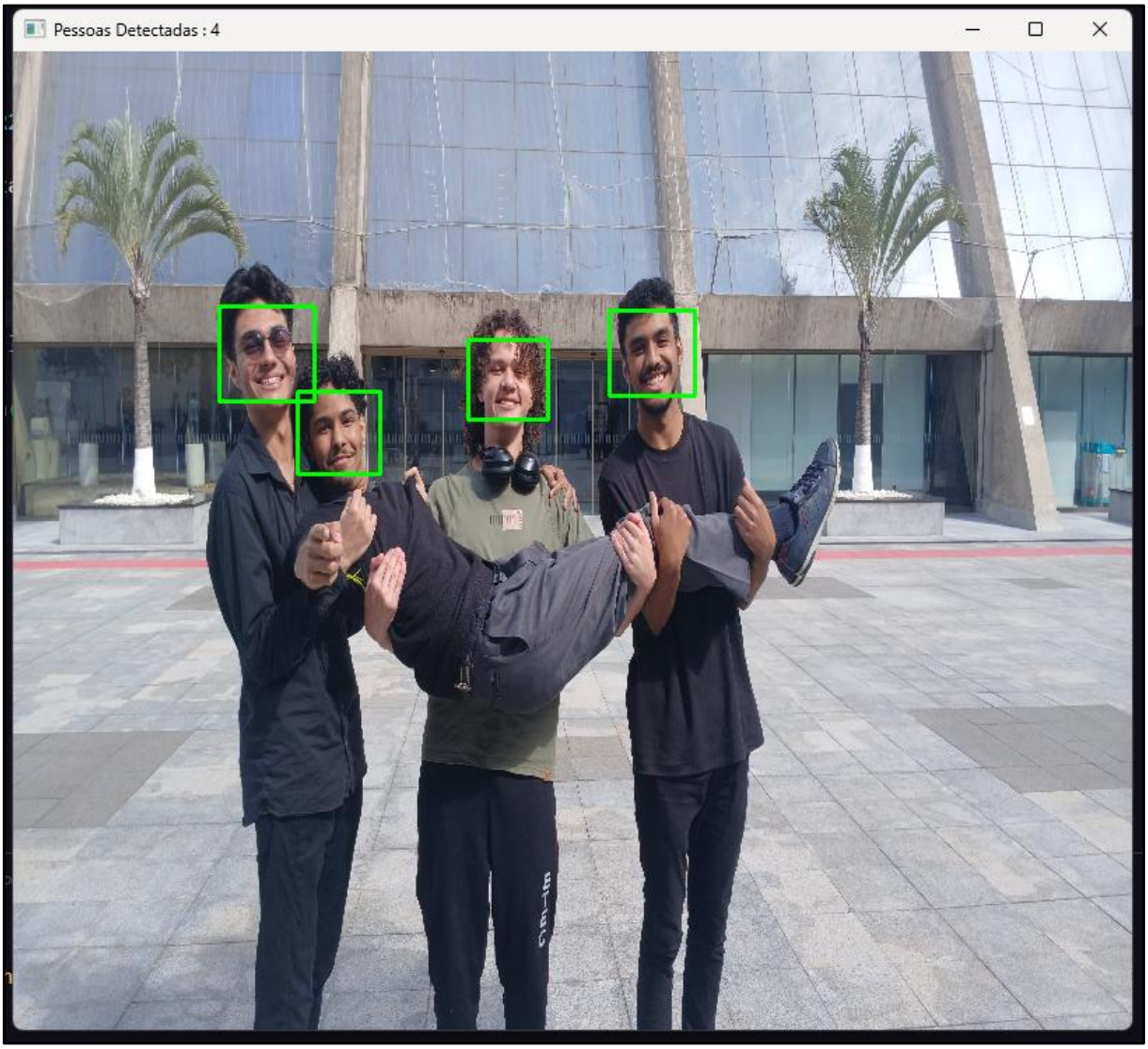
Linha 9: São definidas algumas especificações para a detecção de rostos na imagem, sendo elas a escala da análise e a sensibilidade dos resultados.

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 em Python com OpenCV



Fonte: Autoria Própria, 2025.

Mediapipe é responsável por realizar o reconhecimento em tempo real do programa.

Segundo Silva (2023), o MediaPipe é uma ferramenta poderosa para detecção e reconhecimento em tempo real, sendo integrada a uma rede neural convolucional especializada em oferecer alta precisão nas detecções.

Schirmer (2023), enfatiza que o alcance de localização da ferramenta é de aproximadamente 4 metros e, uma vez localizado o objeto ou pessoa, o sistema pode continuar funcionando eficientemente até uma distância de 20 metros.

De acordo com a Google (2025), a biblioteca disponibiliza meios simplificados para o uso de Machine Learning. Sua portabilidade e desempenho são seus fatores para desenvolvimento desktop e mobile.

Figura 14 - Exemplo de um código Python com 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.

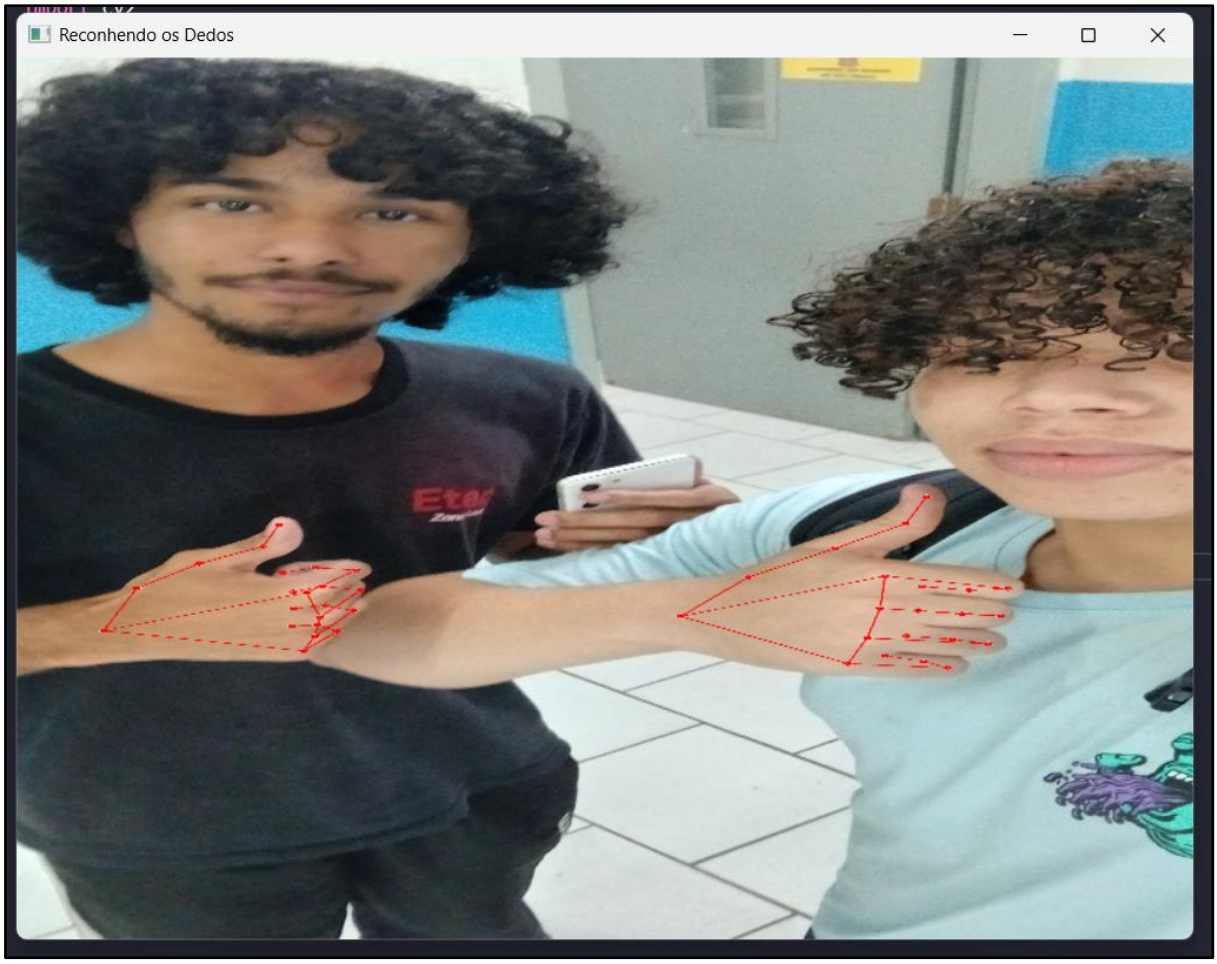
Linha 24: Está sendo declarado uma nova variável que redimensiona a imagem original, utilizando uma propriedade do OpenCV.

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 Python com MediaPipe



Fonte: Autoria Própria, 2025.

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

2.2.4 Banco de dados

Segundo Silberschatz (2003), o banco de dados pode ser comparado a um armazém de informações, cujo objetivo principal é proporcionar eficiência na localização, administração e segurança dos dados armazenados.

De acordo com os estudos de Elmasri e Navathe (2011), a tecnologia de banco de dados é essencial para a sociedade moderna, constituindo a base de quase todo o tráfego de informações na atualidade. A importância desse sistema é inegável, especialmente quando consideramos que a maior parte das informações está armazenada em algum banco de dados.

Conforme proposto por Brito (2010), desde a criação do modelo relacional, este tem se mantido predominantemente presente no cenário de gerenciamento de bancos de dados.

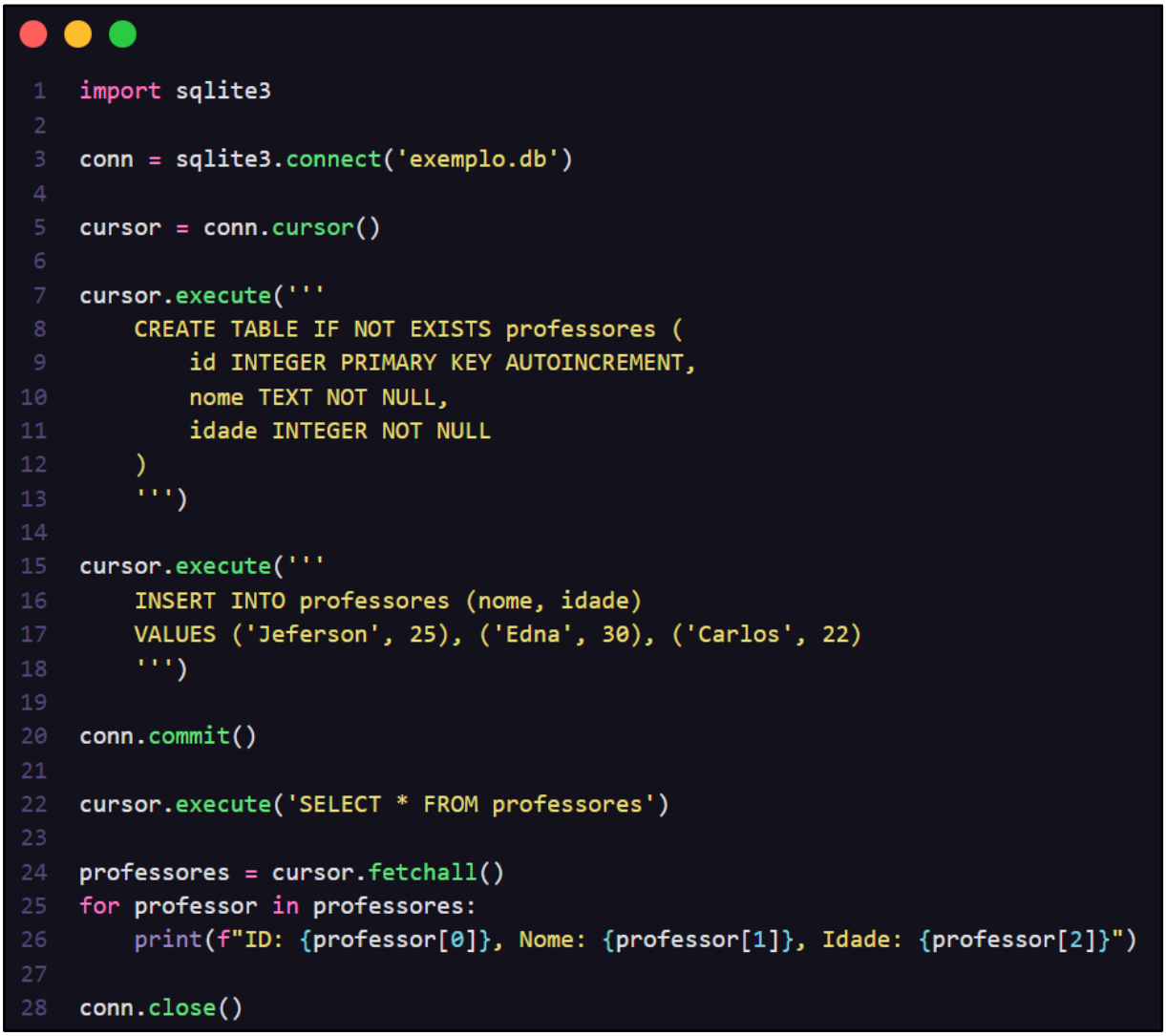
SQLite é o banco de dados que melhor se adequa com a linguagem python podendo oferecer uma experiência personalizada.

Como afirma Louzada (2022), o SQLite é um banco de dados simples, prático e intuitivo, no qual não há obrigatoriedade de utilização da arquitetura de dados cliente-servidor.

Banin (2018), enfatiza que devido ao fácil acesso e à sua instalação juntamente com o Python, o SQLite se revela um banco de dados ideal para aplicações Python. Sua versatilidade é tamanha que pode ser utilizado em diversas plataformas, como dispositivos móveis, aplicativos, sistemas embarcados, entre outros.

Outra vantagem do SQLite é a ausência de configuração inicial, o que permite ao desenvolvedor adotar a metodologia de "baixar e usar" sem maiores complicações como observado por Comachio (2011).

Figura 16 - Exemplo de um código Python utilizando o 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

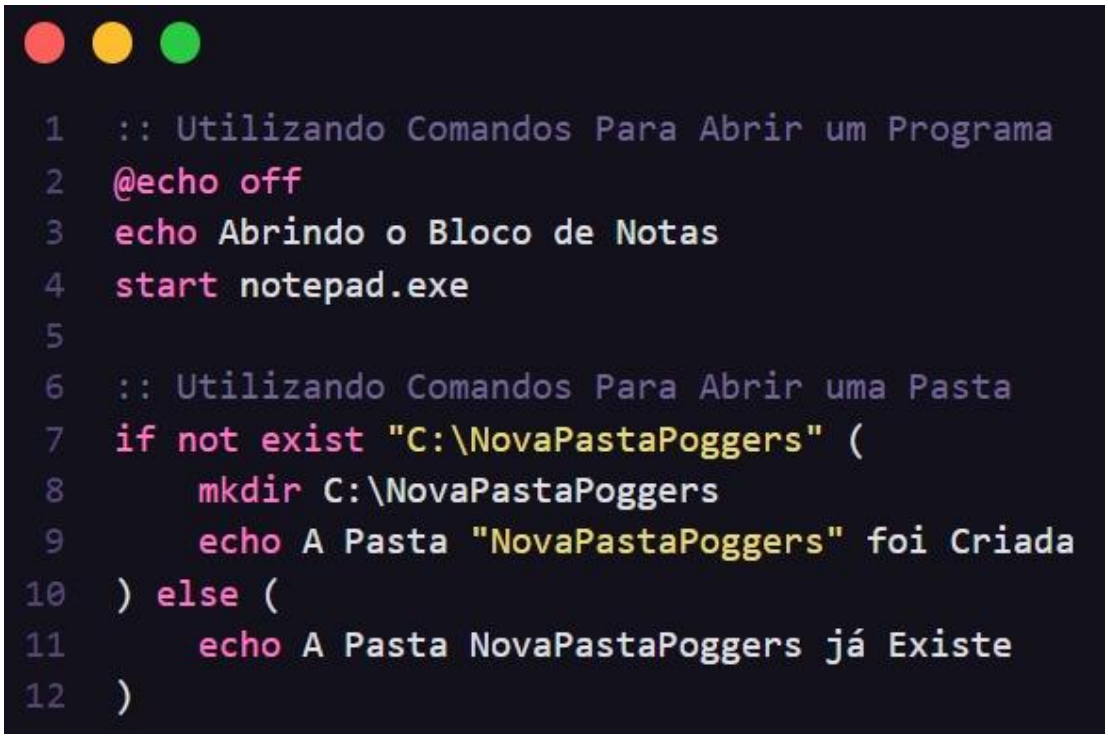
Para Negus (2014), anteriormente ao uso de atalhos e símbolos, os sistemas operacionais do tipo UNIX, permitiam a comunicação entre o usuário e a camada mais primitiva do sistema, por meio de um Shell.

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



```
1  :: Utilizando Comandos Para Abrir um Programa
2  @echo off
3  echo Abrindo o Bloco de Notas
4  start notepad.exe
5
6  :: Utilizando Comandos Para Abrir uma Pasta
7  if not exist "C:\NovaPastaPoggers" (
8      mkdir C:\NovaPastaPoggers
9      echo A Pasta "NovaPastaPoggers" foi Criada
10 ) else (
11     echo A Pasta NovaPastaPoggers já Existe
12 )
```

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

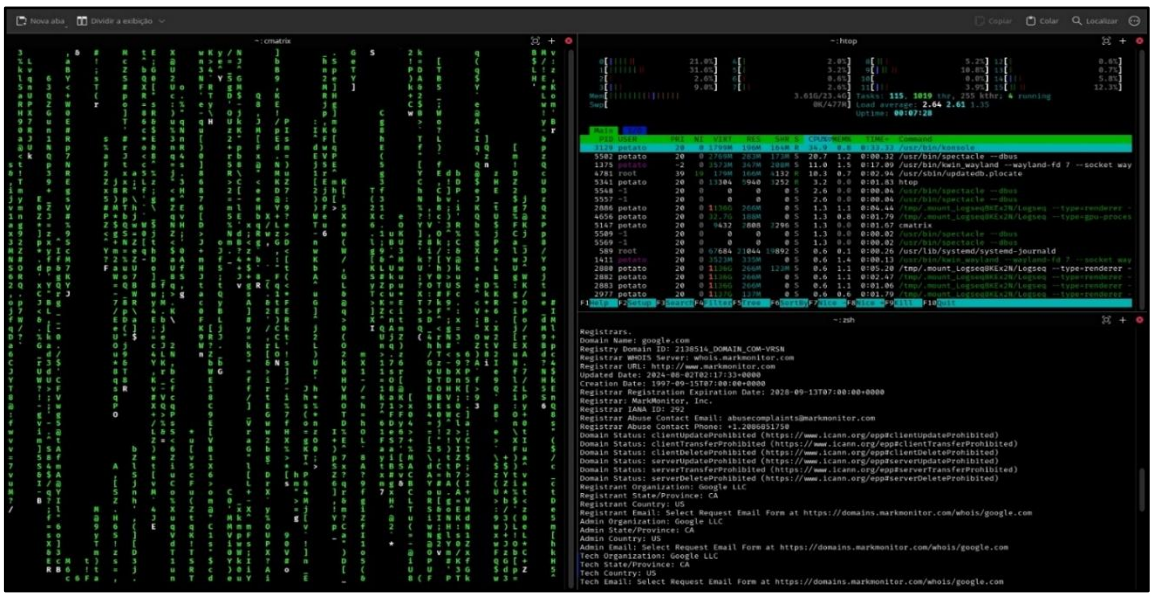
Linha 8: Comando utilizado para criar uma pasta, ao lado dele é informado a localização e o nome da nova pasta

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

Afim de complementar a explicação do UNIX um exemplo é o próprio Linux que utiliza.

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)

À luz do pensamento de Booch (2012), após o que foi chamado de Guerra de Métodos, que durou de 1989 até 1994, 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.

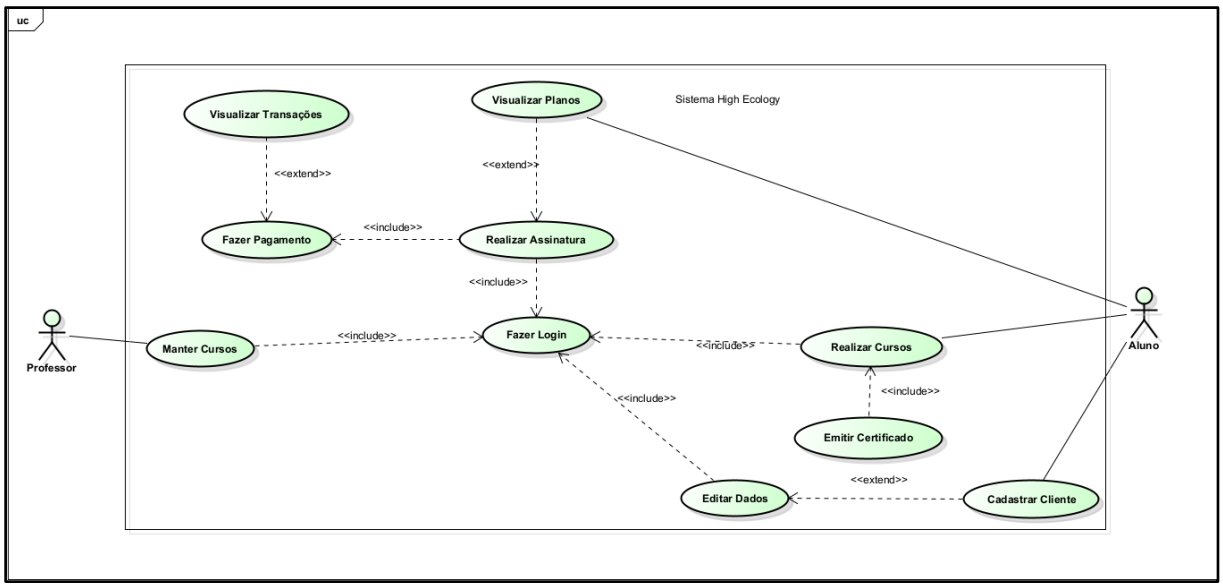
Na concepção de Fowler (2005), as normas definidas e descritas pela UML sobre a criação de diagramas, entregam aos desenvolvedores, a oportunidade de comunicar de forma mais compreensível a visão que se tem de um projeto.

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 do Caso de Uso Realizar Ligação

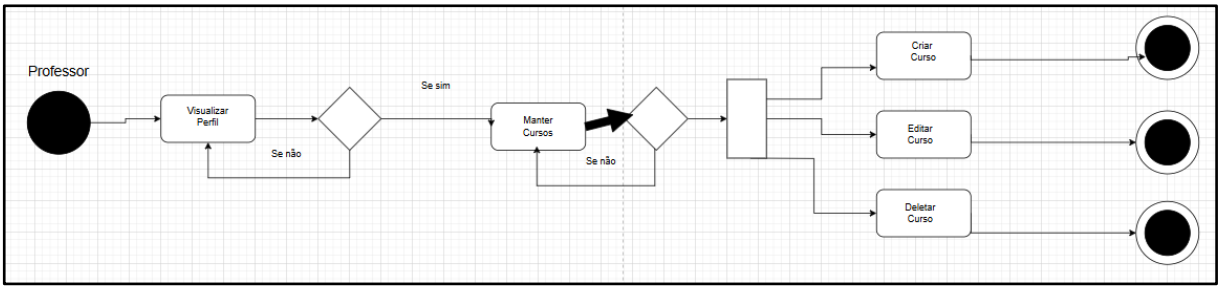
Nome do Caso de Uso	**UC01 – Realizar Ligação**
Ator Principal	Usuário
Atores Secundários	
Resumo	Descreve os passos necessários para que um usuário de celular possa fazer uma ligação para um determinado número
Pré-condições	
Pós-Condições	
Cenário Alternativo I – Número do Contato Conhecido	
Ações do 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.

2.2.6.3 Diagrama de Atividade

De acordo com Booch (2012), o diagrama de atividade é responsável por mostrar o fluxo de atividade de um sistema, fazer a estruturação dele em um conjunto de atividades que pode ser sequencial ou ramificado de uma atividade para outra.

Figura 21 - Exemplo de Diagrama de Atividade



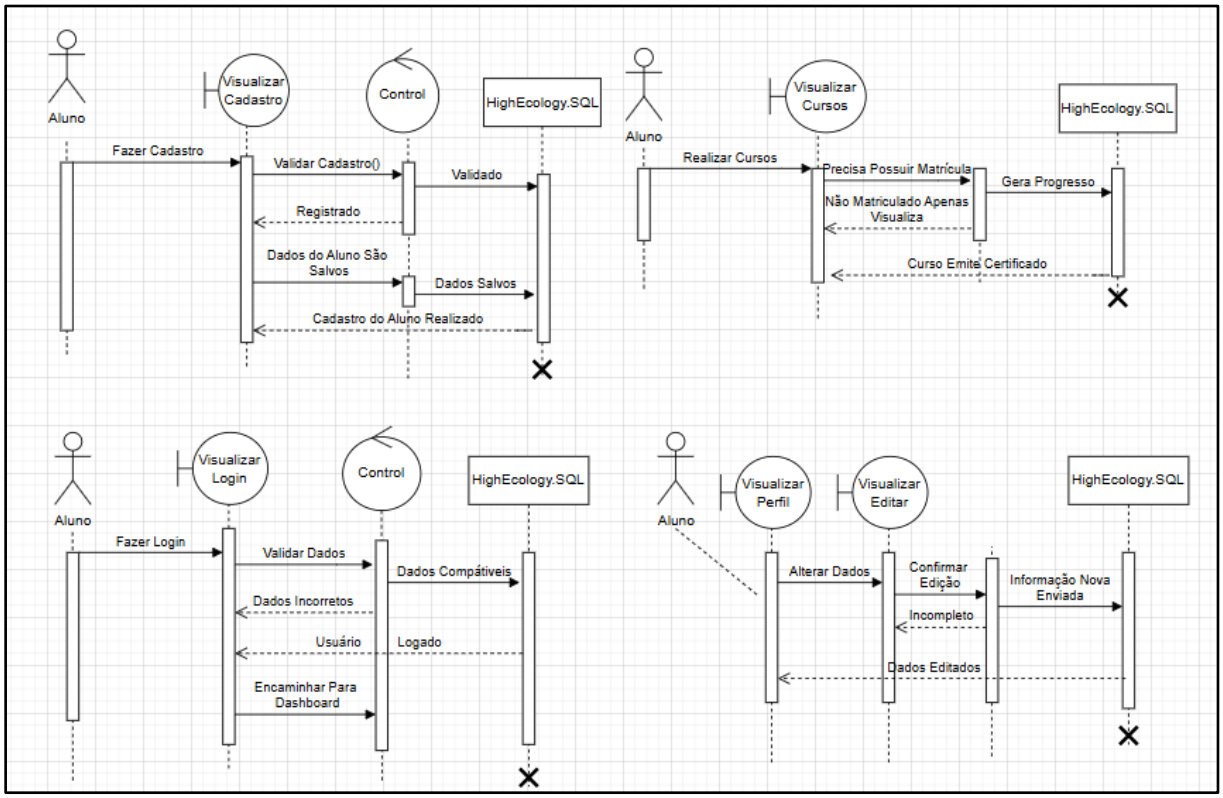
Fonte: Autoria Própria, 2025.

2.2.6.4 Diagrama de Sequência

Fowler (2005), ressalta que o diagrama de sequência regularmente detém o comportamento de um cenário singular, também necessita da existência de um Caso de Uso, pois diversos objetos e mensagens passadas neste cenário são extraídas do caso de uso.

Diagrama de Sequência é recomendado para ilustrar a visão dinâmica de um sistema dando ênfase a ordem passageira de mensagens como um conjunto de papéis representados pelas mensagens que são enviadas e recebidas, como demonstrado por Booch (2012).

Figura 22 - Exemplo de Diagrama de Sequência

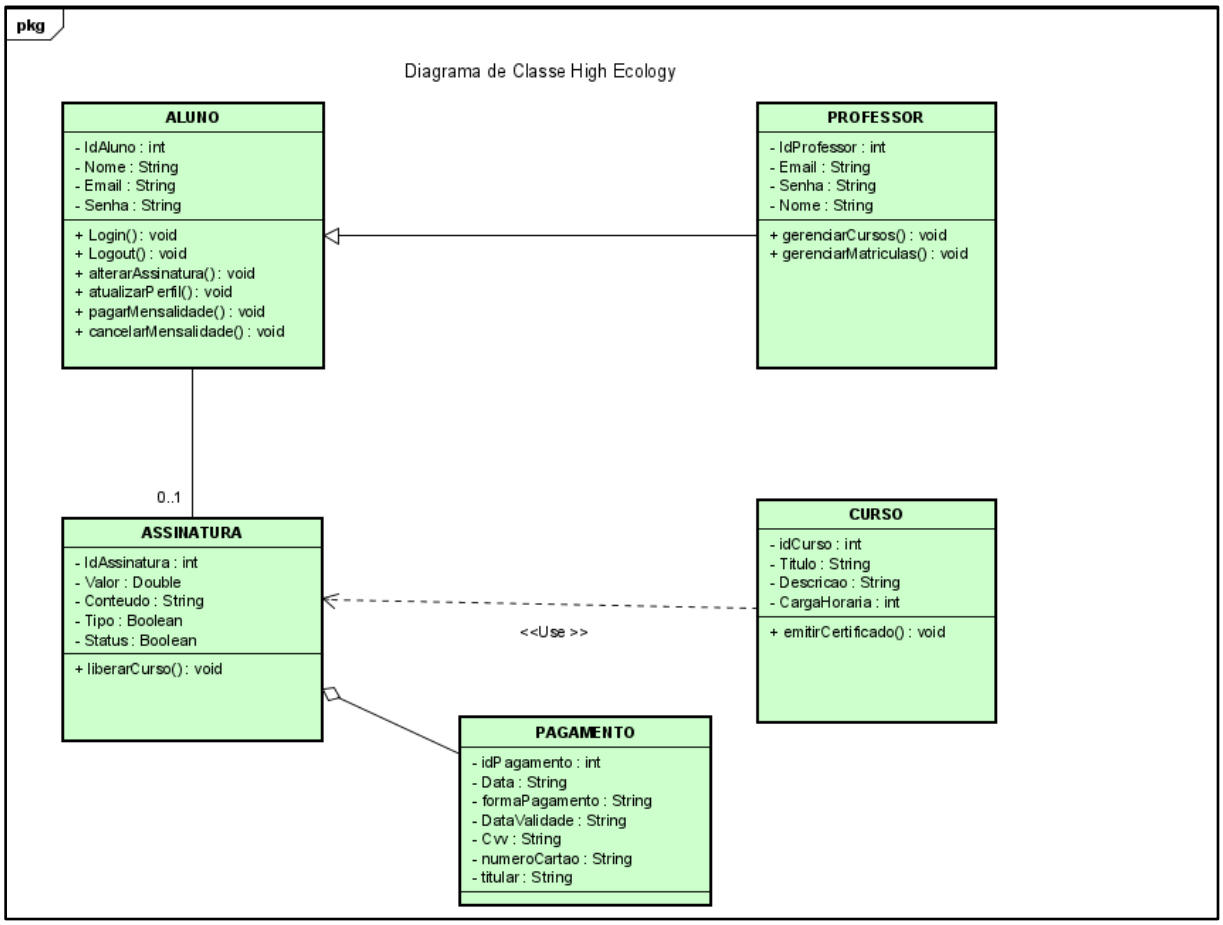


Fonte: Autoria Própria, 2025.

2.2.6.5 Diagrama de Classe

Na visão de Fowler (2005), o diagrama de classes retrata as variações de objetos existentes no sistema e seus tipos de relacionamento estático, ademais mostram as operações e propriedades de classe. Análise de Guedes (2018), indica que o diagrama é essencialmente importante por servir de apoio para construção de outros diagramas UML, evidenciando a ordem e definindo a estrutura lógica.

Figura 23 - Exemplo de Diagrama de Classe



Fonte: Autoria Própria, 2025.

2.2.7 Prototipagem

Como destaca Grilo (2019), o projeto da interface, com disposição no desktop ou no mobile, a tipografia e ícones utilizados, faz parte da composição visual, como projetar telas, layout, organizar os elementos e pensar nos comportamentos destes são práticas técnicas, executadas por designers gráficos (UI designers).

A UX faz parte de tudo que o humano vivencia sendo um usuário, desde rede social, e até mesmo aplicativos básicos**,** tais como o alarme e o banco, quando você usa

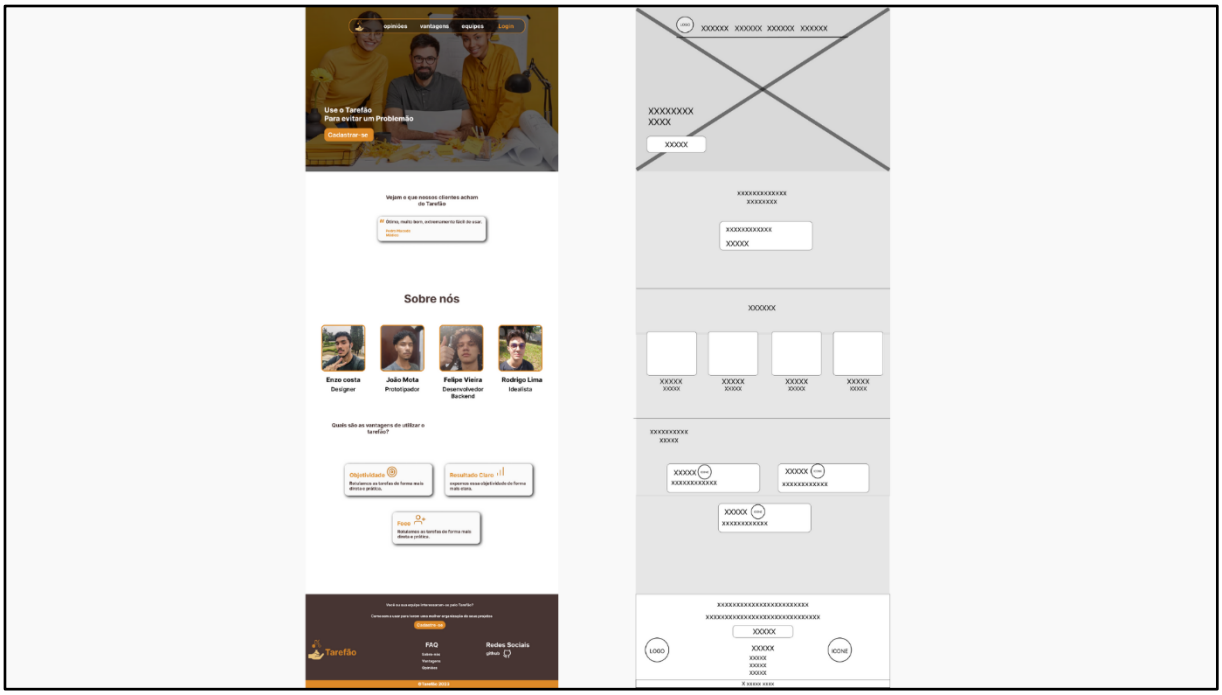
alguns desses conceitos você tem uma experiência, e o UX é o estudo de como aprimorar a percepção do usuário em suas experiências como observado por Teixeira (2014).

Nos estudos de Grilo (2019), o cientista Donald Norman é mencionado como o responsável por popularizar o termo de UX, para Norman desenvolver aplicações mobile ou páginas web não estão só incluídos na UX, mas são apenas uma pequena parte dela.

Conforme aponta Miro (2025), os Wireframes são protótipos simples para visualizar a sua aplicação, seja no desktop ou no mobile, podendo existir em mais de um formato, como wireframes de baixa e alta fidelidade, neles são aplicados os conhecimentos de UX e UI, mostrados anteriormente.

Assim o wireframe pode ser descrito como o esqueleto de toda a aplicação, sendo o de baixa um conteúdo básico, sem cores, sem imagens e com formas simples, a seguir o wireframe de alta possui uma complexidade maior, com cores e imagens, conforme os apontamentos de GoDaddy (2024).

Figura 24 - Exemplo de Wireframes de Alta e Baixa Fidelidade

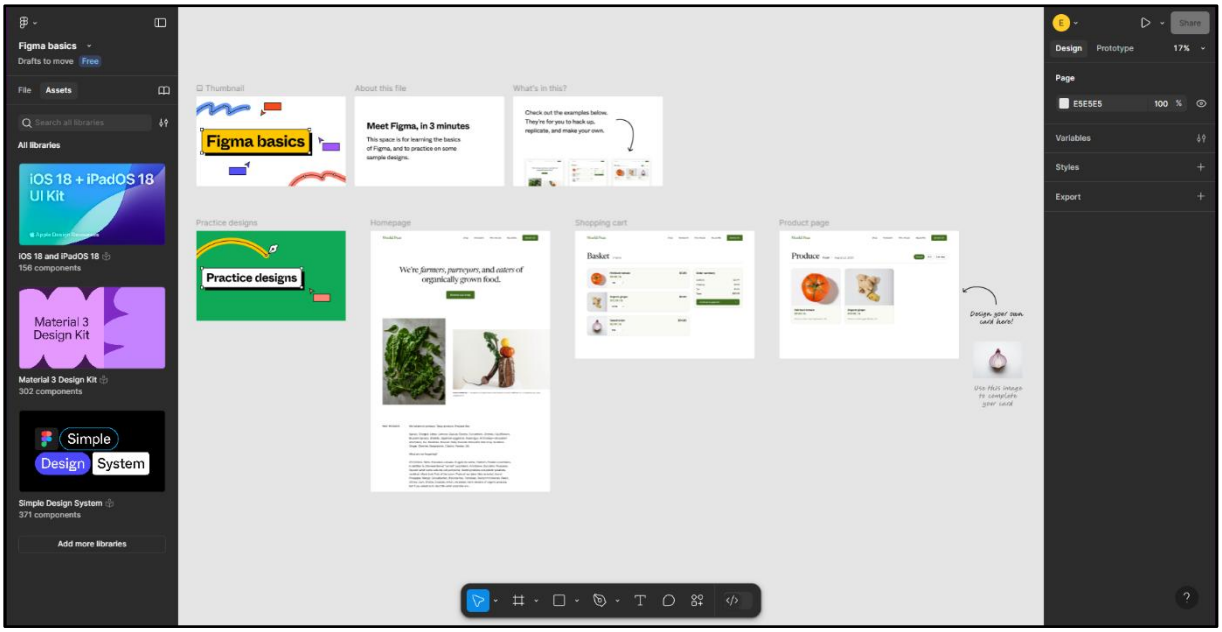


Fonte: Autoria Própria, 2025.

Tendo em mente os conceitos explicados anteriormente tudo se reúne ao figma, a principal ferramenta de prototipagem.

Tal como observa Oliveira (2022), o Figma obteve uma grande comunidade pela sua capacidade de criar um bom espaço público de compartilhamento de dados entre os usuários, promovendo também a oportunidade de tornar o design acessível a todos, que suporta diversos sistemas operacionais. A análise de Alura (2022) indica que a ferramenta aproveita de sua comunidade que por sua vez produz sistemas de design, plugins e guias de estilo, para auxiliar as gamas de usuários.

Figura 25 - Exemplo de interface para uso do figma



Fonte: Autoria Própria, 2025

2.3 Desenvolvimento Web

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 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.

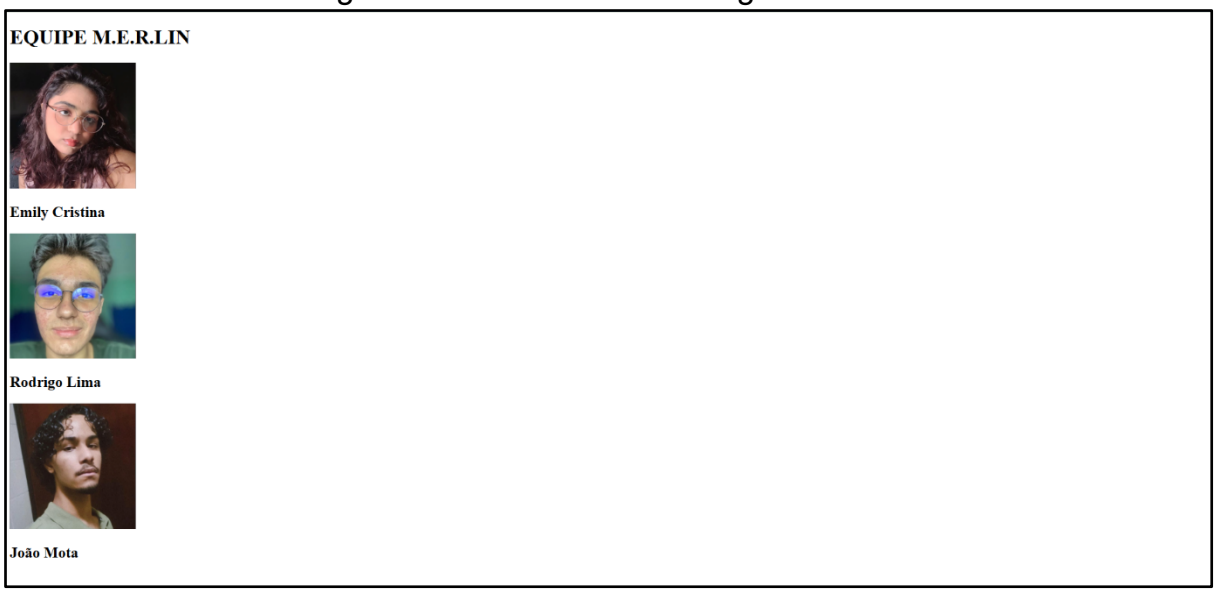
“<h1>”: Possuindo uma hierarquia de relevância na página, numerada de “<h1>” até “<h6>”, sendo essa segunda a de menor impacto, esta tag é utilizada em textos presentes na página.

“<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 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 link para o css

Fonte: Autoria Própria, 2025.

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

Figura 30 - Resultado do css com html

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 JavaScript

```javascript
const cards = document.querySelectorAll('.card');

cards.forEach(card => {
  card.addEventListener('click', () => {
    const nome = card.getAttribute('data-nome');
    alert(`Você Clicou na Foto de : ${nome}`);
  });
});
```

Fonte: Autoria Própria, 2025.

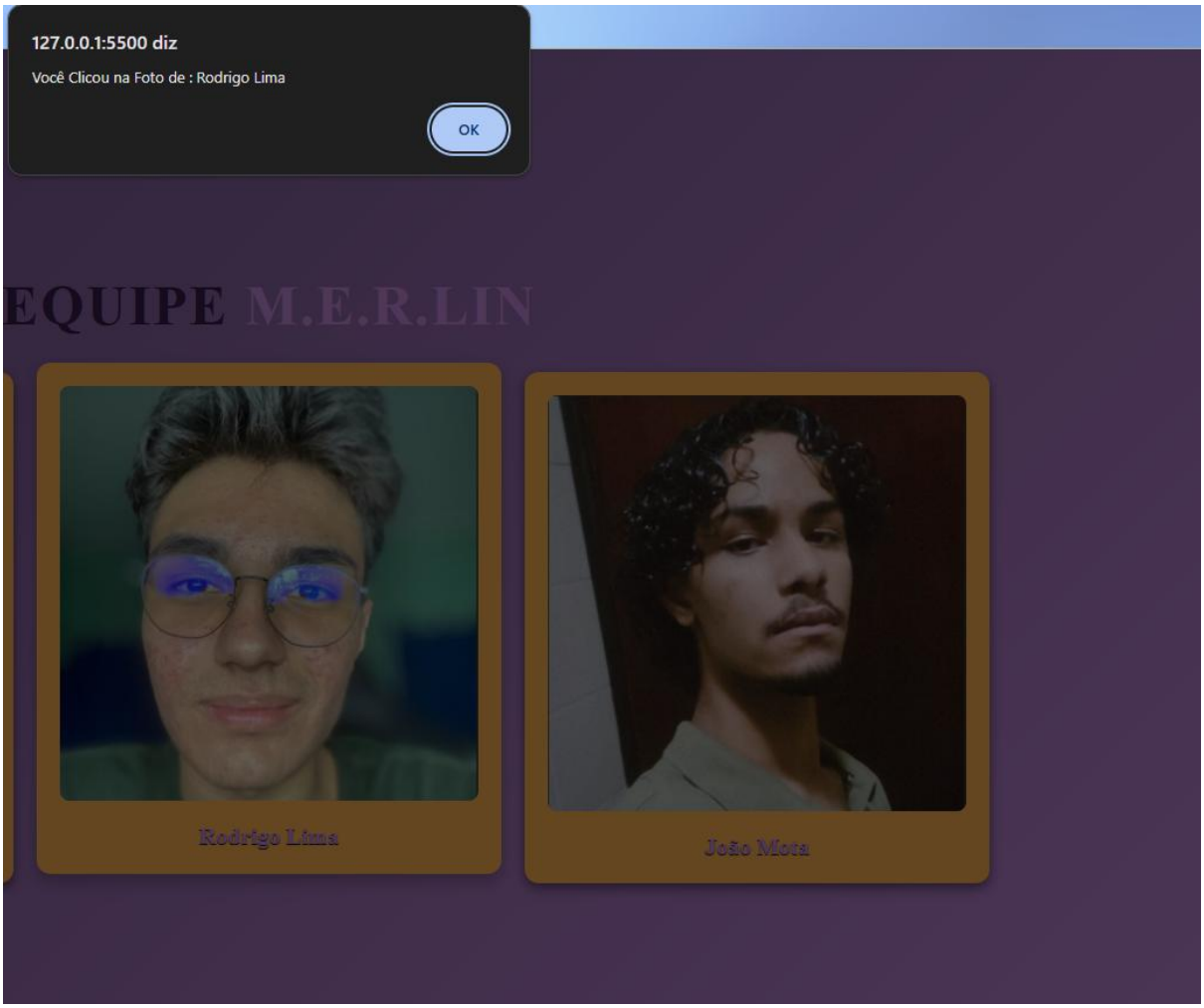
Na figura há uma demonstração de codificação em JavaScript. O objetivo desse código é exibir uma mensagem quando o usuário clicar em uma das 3 fotos presentes na tela.

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 JavaScript



Fonte: Autoria Própria, 2025.

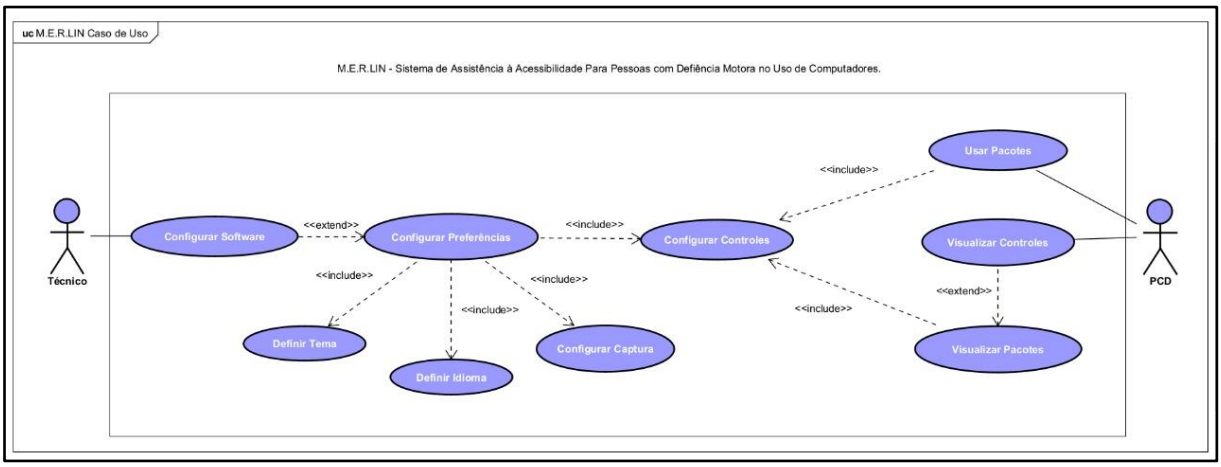
3 DESENVOLVIMENTO

O presente capítulo será documentado e apresentado o desenvolvimento do software M.E.R.LIN seguindo os conceitos da UML. A apresentação contará com protótipos das telas, conceito da marca e Tabela de documentação, mostrando o processo de desenvolvimento do sistema e sua marca.

3.1 Diagrama de Caso de Uso

No diagrama de Caso de Uso é possível visualizar detalhadamente as ações que os atores do software poderão realizar no produto final de forma simples e clara. Contendo o Ator, PCD, principal utilizador do software e, Responsável, o qual fará as configurações necessárias para o uso adequado do sistema para o PCD. O diagrama possui 16 elipses de Caso de Uso.

Figura 33 - Caso de Uso 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 disponibilizará os conceitos não visuais do caso de uso, como: Requisitos Funcionais (RF), Requisitos Não Funcionais (RNF), Regras de Negócio (RN).

O Requisito Funcional é responsável por definir a ação que o sistema deverá executar para cumprir com o seu propósito de desenvolvimento, sendo essencial para qualquer aplicação que conta com algum objetivo e propósito. Os Requisitos Não Funcionais são os responsáveis por definir como deverá ser operada a aplicação, como seu desempenho, usabilidade, segurança, confiabilidade, entre outros; não sendo essenciais para o funcionamento das suas ações como o RF. Já as Regras de Negócios fazem parte da marca, é o processo jurídico e político privado da aplicação, na qual fazem parte de: diretrizes, condições, finanças e serviços; todo negócio desde marcas e empresas precisa ter sua RN interna sobre as operações de seus 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

O diagrama de atividade é necessário para ver o fluxo de ações dentro de um sistema, que descreve os comportamentos do sistema, sendo paralelo ou simultâneo em decisões e atividades que podem ocorrer durante o uso de ambos os atores.

3.4 Diagrama de Sequência

O diagrama de Sequência mostra visualmente a troca de mensagens do sistema, de forma temporal, deixando detalhado a sequência de ações de um caso de uso dos atores de cada cenário único. Podendo no diagrama ter a presença, do ator, interface, classe, banco de dados, que ajudam a mostrar as interações necessárias para a realização de uma ação do sistema.

3.5 Diagrama de Classe

O diagrama de classe faz o mapeamento a estrutura estática, mostrando seus atributos, operações e relacionamentos entre classes, descrevendo a estrutura dos objetos e definindo as classes do sistema e o que elas podem fazer. Sendo um diagrama essencial para a construção de outros diagramas da aplicação.

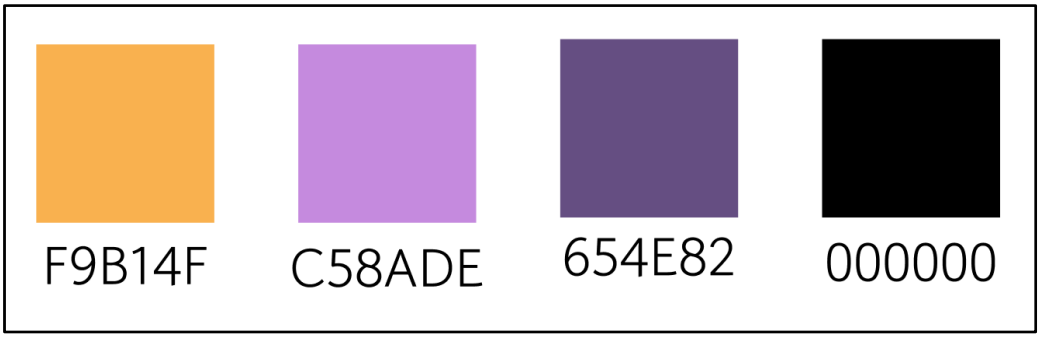
3.6 Marca

O capítulo explicará os conceitos pensados pelos desenvolvedores para a criação da marca M.E.R.LIN, na qual mostrará o motivo da escolha do nome, as cores, logo e tipografia para o desenvolvimento do software da marca.

O nome da marca M.E.R.LIN diz respeito as três iniciais dos integrantes, sendo M de Mota, E de Emily e R de Rodrigo, além de que M.E.R.LIN refere-se ao Mago Merlin da Mitologia celta, trazendo assim um ar mágico ao conceito da marca.

O software possuí 4 cores principais, dois tons de roxo, um tom de amarelo e preto.

Figura 42 - Cores do software



Fonte: Autoria Própria, 2025.

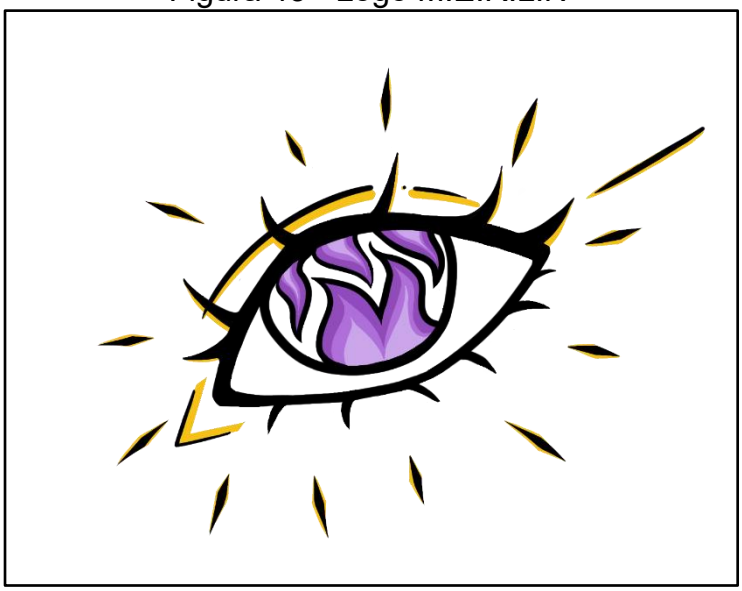
O amarelo remete a otimismo, desenvolvimento, clareza, e encoraja a comunicação, além de estimular os processos mentais; essa cor dentro do software é usada para os detalhes, sendo mais como um brilho e sombra.

O roxo remete a imaginação, sabedoria, sendo animado ao mesmo tempo que acalma a mente; no software o roxo é a principal cor, onde os blocos dos wireframes, e a cor predominante na logo será o roxo.

O preto remete a autoridade, e poder, também evocando um mistério implementando o conceito da marca; O preto é a cor na qual contorna os elementos da logo, e é a cor fonte.

A logo do software, é um olho com brilho amarelo complementando o contorno preto, com a íris do olho sendo dois tons de roxo que imitam chamas, fazendo discretamente um formato de M do nome da marca.

Figura 43 - Logo M.E.R.LIN

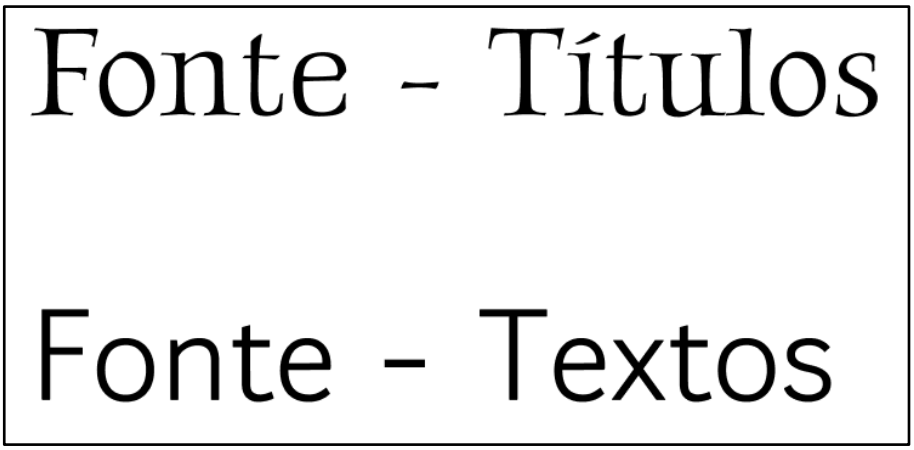


Fonte: Autoria Própria, 2025.

A logo da marca possuí esse formato de olho pois o principal fator é o uso do olho para executar as ações que o PCD deseja no computador, e as chamas são o conceito da marca de magia, pois executar ações através do olho é similar a magia.

Na tipografia foi escolhido duas fontes, uma fonte para títulos e uma fonte para textos a fonte para títulos é a Gideon Roman e a de textos Gowun Dodum.

Figura 44 - Fontes do M.E.R.LIN



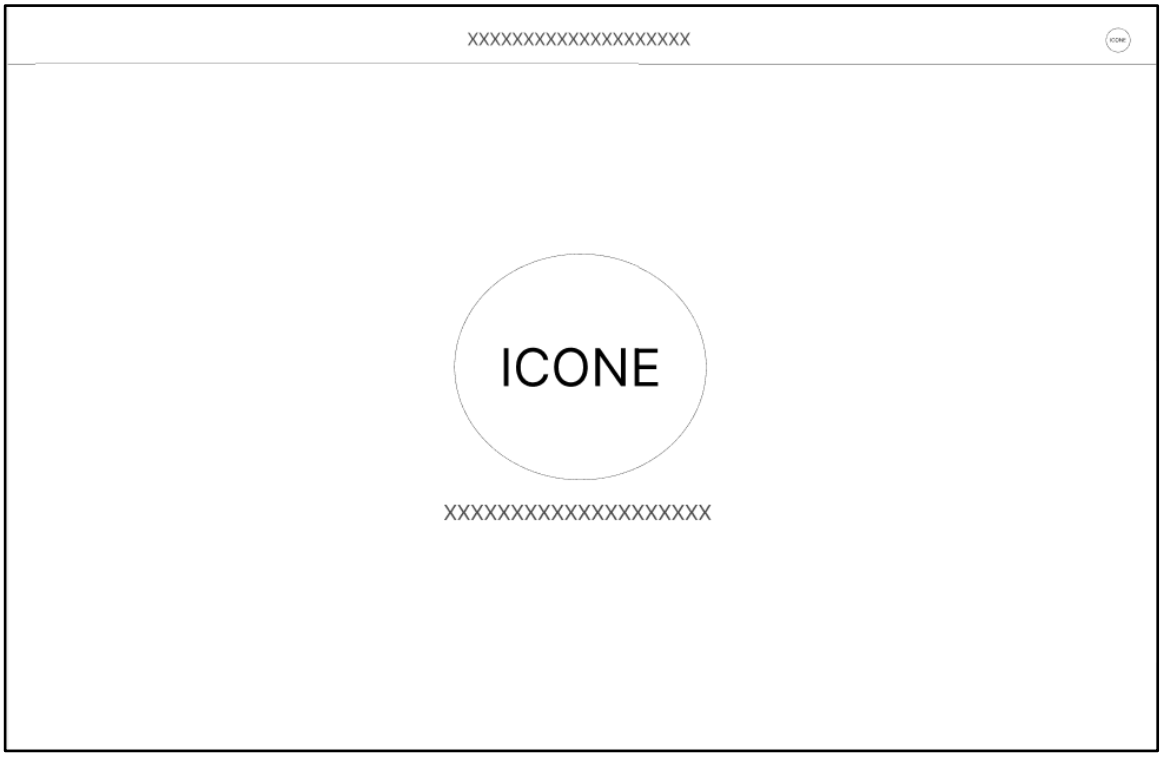
Fonte: Autoria Própria, 2025.

3.7 Prototipação das interfaces do sistema

Neste capítulo será mostrado as interfaces que compõem o sistema, contendo wireframes de baixa e alta qualidade sobre as telas do software e do site, importante citar que o site, é apenas uma página web vitrine, para apresentar e informar o funcionamento dos serviços do M.E.R.LIN. O software contém 8 páginas de wireframe sobre a configuração do aplicativo; já o website contém apenas uma página rolável de exibição.

A primeira página do software, é a tela de inicialização para configuração, a primeira página a ser vista assim que o aplicativo é instalado na máquina, a tela exibe a estrutura na qual será montada a página, sendo os “X” onde possui algo escrito, e o círculo com ícone, onde terá alguma ilustração ou símbolo.

Figura 45 - Página de Inicialização software baixa qualidade



Fonte: Autoria Própria, 2025.

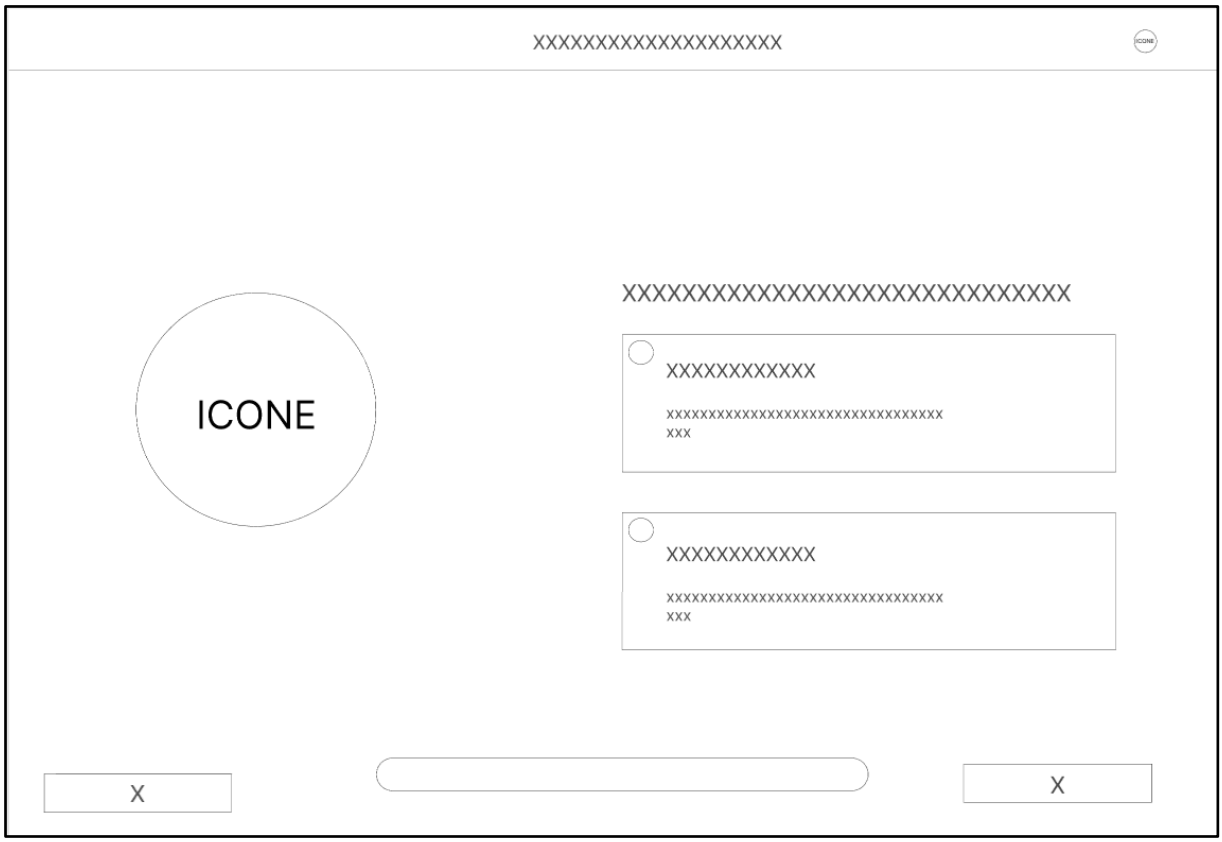
A seguir o wireframe de alta qualidade da mesma página de inicialização, assim sucessivamente.

Figura 46 - Página de Inicialização software alta qualidade

Fonte: Autoria Própria, 2025.

Após a tela inicial temos a tela de seleção do modo de configuração, pelo wireframe de baixa qualidade, título na navbar acompanhado de um ícone ao lado, abaixo a logo da marca, e duas box de escolha, a escolha rápida e a escolha personalizada, no footer a opção de voltar, a barra de progresso da configuração, e opção de avançar.

Figura 47 - Modo de configuração de baixa qualidade



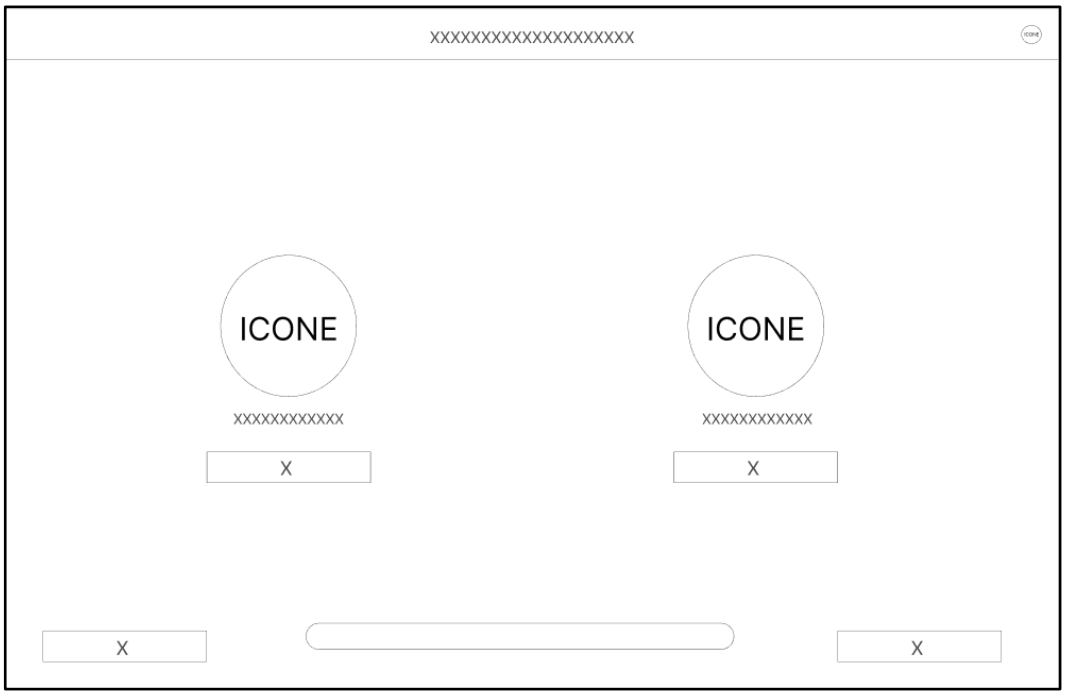
Fonte: Autoria própria, 2025.

Figura 48 - Modo de configuração de alta qualidade

Fonte: Autoria própria, 2025.

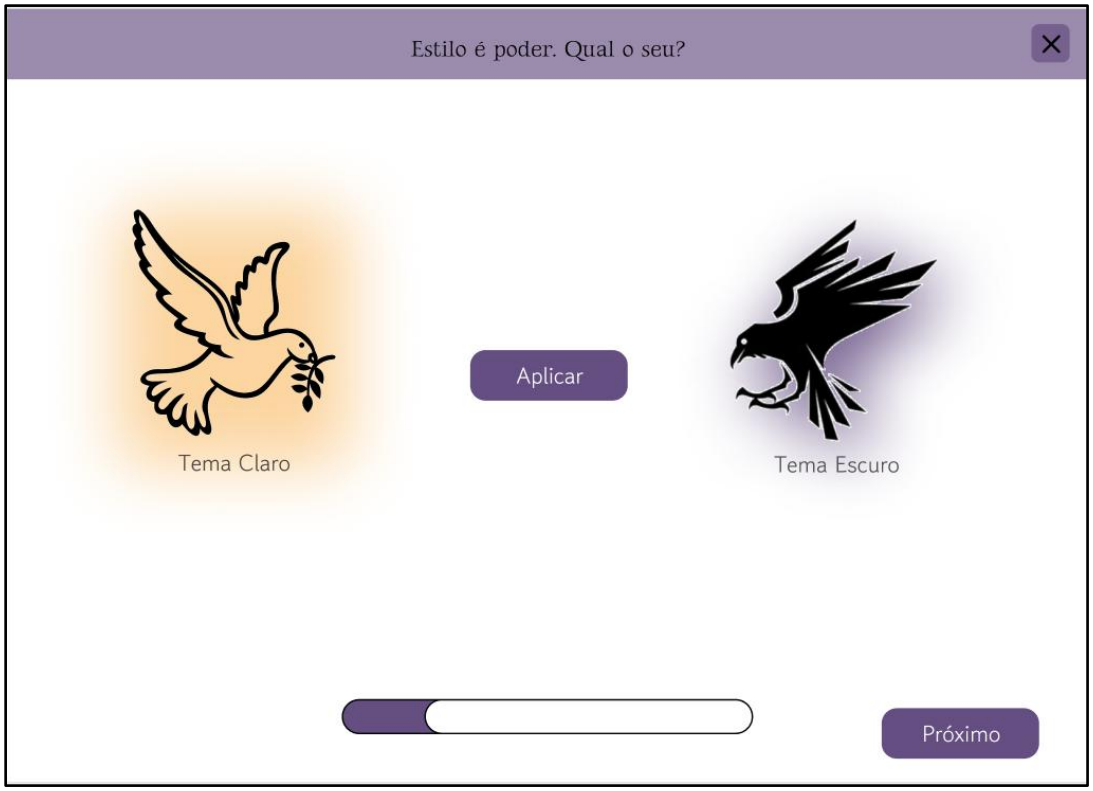
Caso a escolha do responsável tenha sido a configuração rápida se encerra a configuração do software, indo para o uso, mas se eventualmente a escolha tenha sido a personalizada, o responsável será direcionado para a página de escolha do tema, tendo apenas duas opções, escuro ou claro, seguindo o mesmo padrão da navbar e footer, o centro de interesse temos dois ícones representando os temas, e um botão central de aplicar a escolha.

Figura 49 - Tema do Software baixa qualidade



Fonte: Autoria própria, 2025.

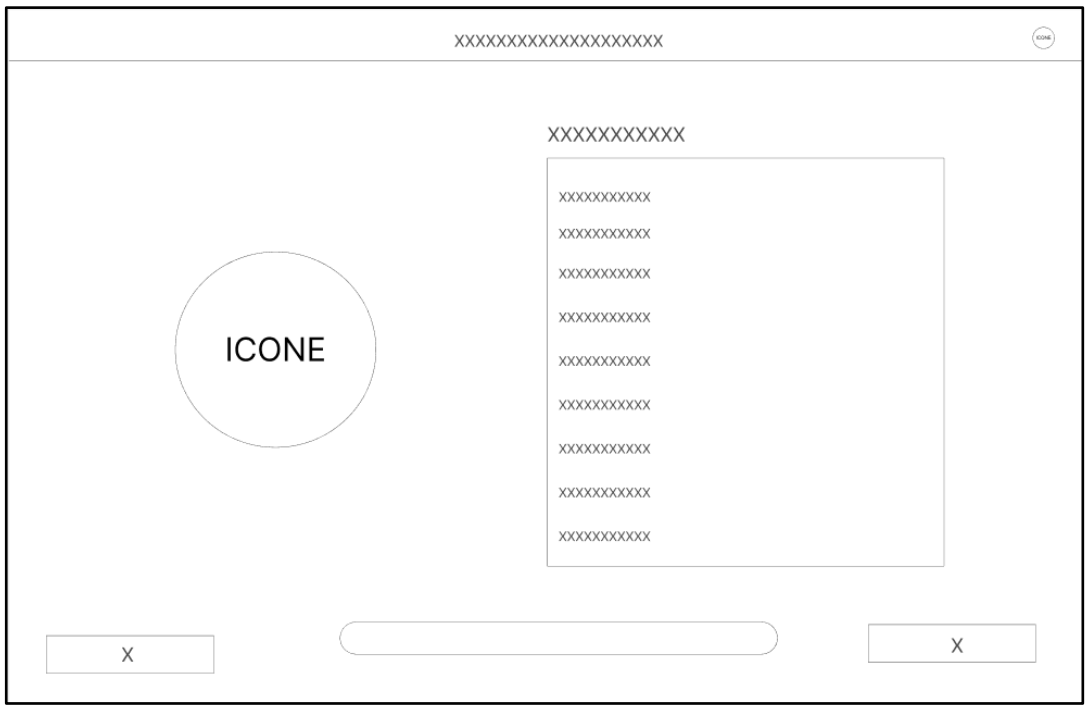
Figura 50 - Tema do Software alta qualidade



Fonte: Autoria própria, 2025.

A seguir escolherá a idioma da aplicação, tendo uma abrangência maior, na página a navbar e o footer seguem o mesmo padrão, já o centro da tela, tem a logo e o catálogo de escolhas do idioma.

Figura 51 - Escolha do Idioma baixa qualidade



Fonte: Autoria própria, 2025.

Figura 52 - Escolha de Idioma alta qualidade

Fonte: Autoria própria, 2025.

E escolha dos ajustes da câmera, o wireframe a ser apresentado é a página de Ajustes, no qual pode ser definido as configurações da câmera como: qualidade, FPS (Frames por segundo), e a luz de alerta da câmera que notifica se ela está ligada ou não, na tela de ajustes também pode redefinir a linguagem do software e ver os termos de uso.

Figura 53 - Ajustes de configuração baixa qualidade

Fonte: Autoria própria, 2025.

Figura 54 - Ajustes de configuração alta qualidade

Fonte: Autoria própria, 2025.

Assim enfim iremos para definições das coletâneas de rotina, a página possui a logo e nome do sistema, botão de voltar, a parte central principal tem cinco campos com o nome da coletânea e botões de ver comando, editar e excluir.

Figura 55 - Configuração das coletâneas alta qualidade

Fonte: Autoria própria, 2025.

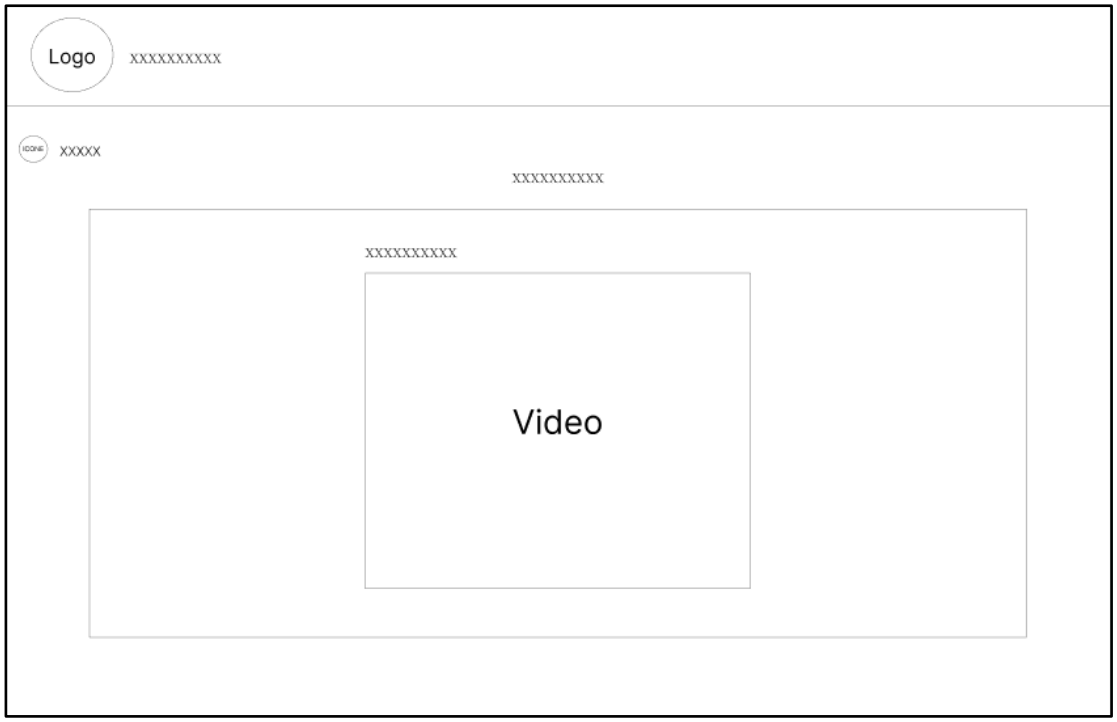
Caso você escolha editar a sua coletânea, será direcionado para outra página de comandos da coletânea, página contém mesma logo, nome, e botão de retorno, no centro terá cinco espaços selecionáveis de aplicativos, assim o aplicativo o aplicativo de dentro da coletânea é ajustável.

Figura 56 - Configuração dos comandos da coletânea alta qualidade

Fonte: Autoria própria, 2025.

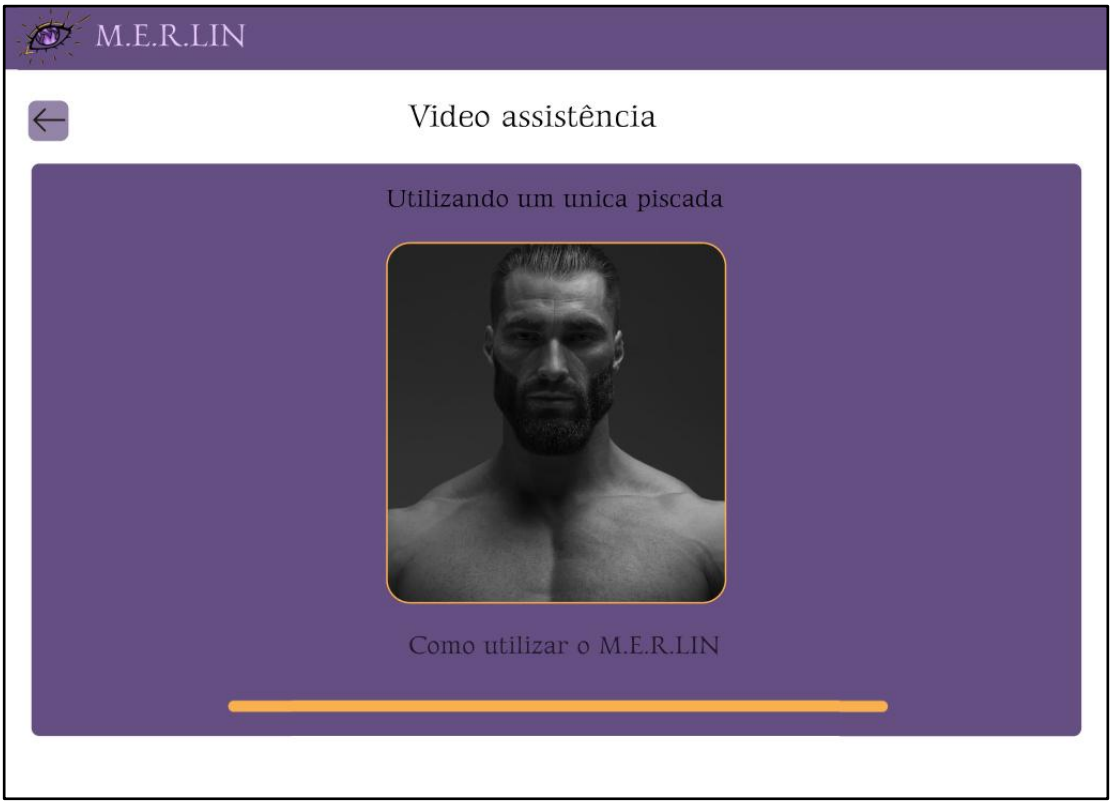
Ou se a opção escolhida ver o comando, a tela exibida terá um vídeo de assistência o qual mostrará como executar o movimento para executar tal ação.

Figura 57 - Vídeo de assistência baixa qualidade



Fonte: Autoria própria, 2025.

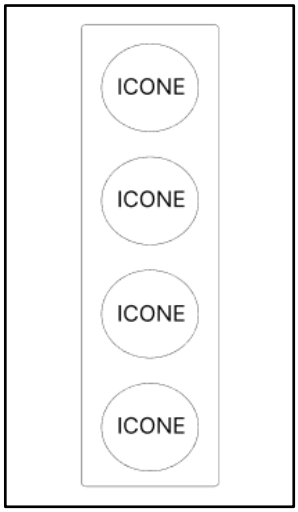
Figura 58 - Vídeo assistência alta qualidade



Fonte: Autoria própria, 2025.

A configuração inicial se encerra sobrando apenas os seus complementos como o Dock, um painel que ficará sempre ao lado da tela ilustrando os botões que podem ser ativados pelo comando ocular no software.

Figura 59 - Dock baixa qualidade



Fonte: Autoria própria, 2025.

Aqui se encerra as telas do software e inicia o wireframe do website de exibição. O website possuí apenas uma tela rolável que exibe a descrição, funcionamento, desenvolvedores, e produto baixável. As imagens dos wireframes serão dividas por cada seção do site, que serão seis, passando por cada etapa pela sua ordem na navbar.

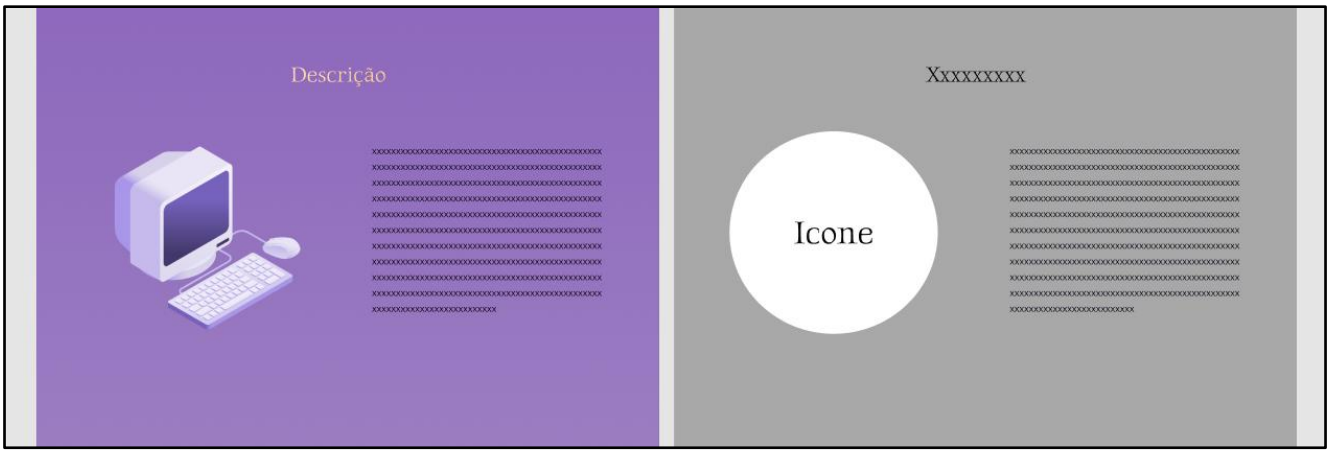
A primeira seção se dá na home, na qual exibe a navbar com logo, nome e mais quatro links de direcionamento na página, abaixo uma mensagem de texto e a logo do website.

Figura 60 - Home do site alta e baixa qualidade

Fonte: Autoria própria, 2025.

A segunda seção será a descrição, na qual dará uma explicação geral do projeto, como qual a finalidade do desenvolvimento dele. A tela possui um título central, uma imagem e texto abaixo.

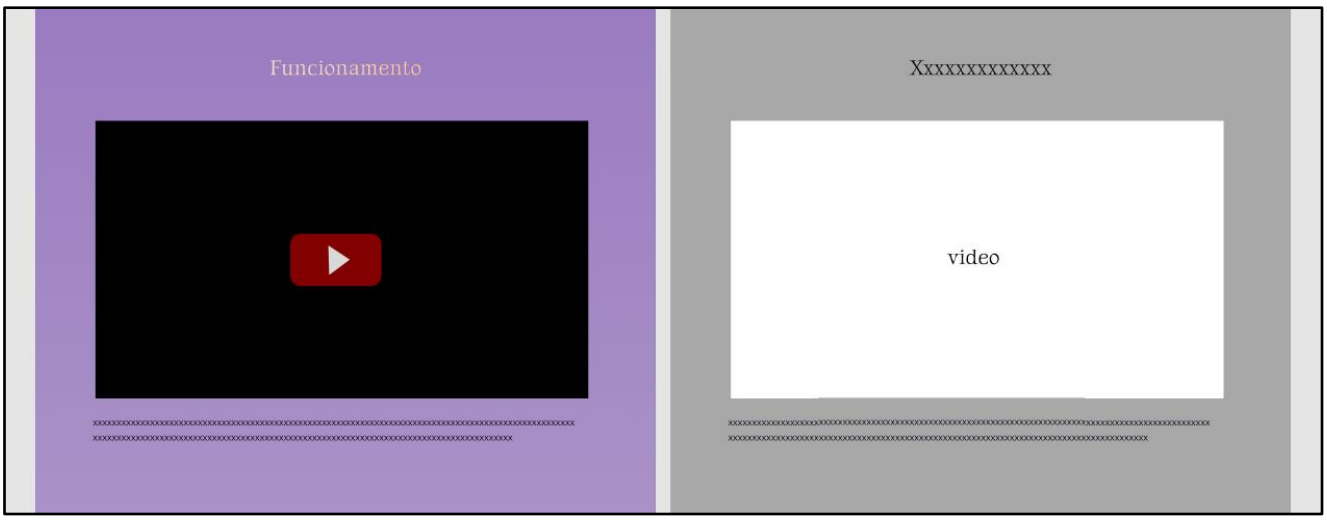
Figura 61 - Descrição do site alta e baixa qualidade



Fonte: Autoria própria, 2025.

A terceira seção mostra por trás de um vídeo o funcionamento do software antes de que o cliente tenha que baixá-lo. A tela possui um título central, um vídeo e texto abaixo.

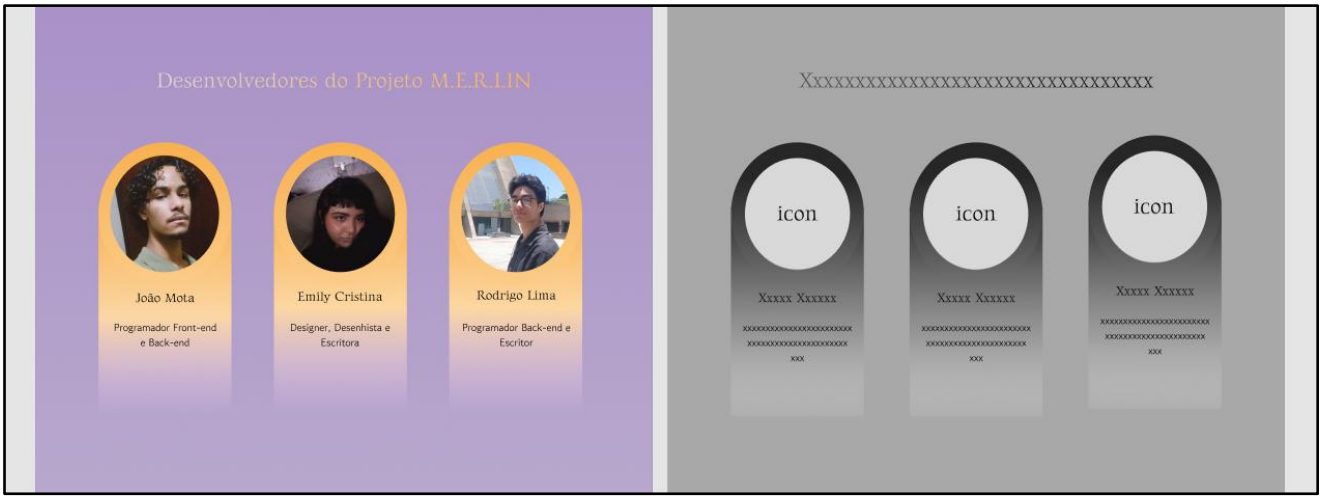
Figura 62 - Funcionamento site baixa e alta qualidade



Fonte: Autoria própria, 2025.

A quarta seção mostra os desenvolvedores do projeto assim como as suas respectivas funções durante o seu desenvolvimento. A tela possui um título central, três cards com a foto dos desenvolvedores por adentro o nome dos tais e sua função.

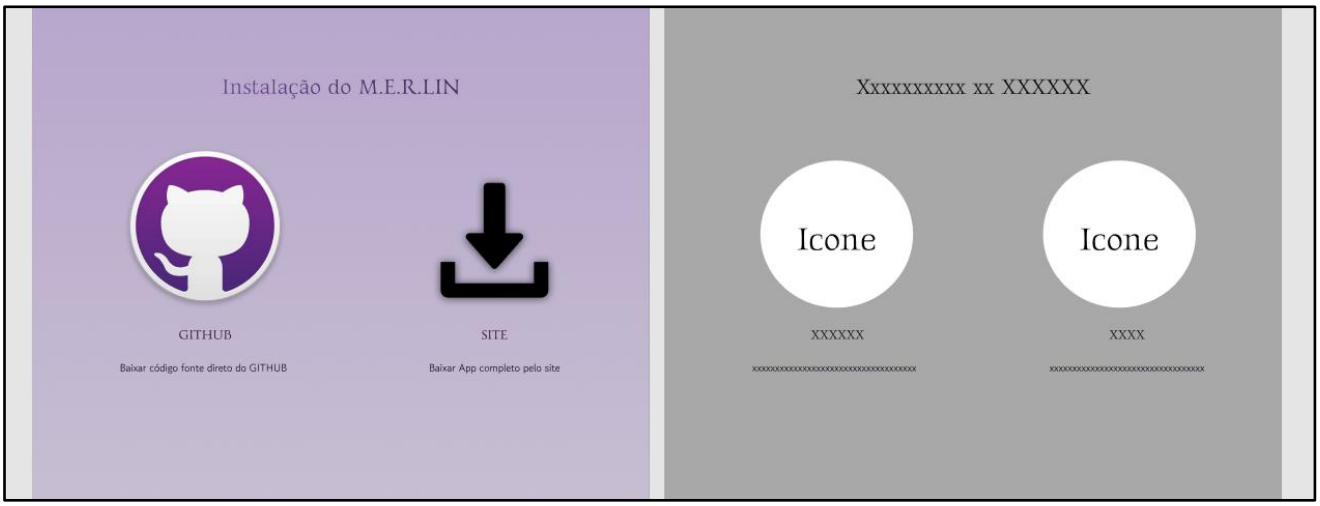
Figura 63 - Desenvolvedor site baixa e alta qualidade



Fonte: Autoria própria, 2025.

A quinta seção mostra os métodos de instalação do software. A tela possui um título central, dois ícones e texto um a cada lado.

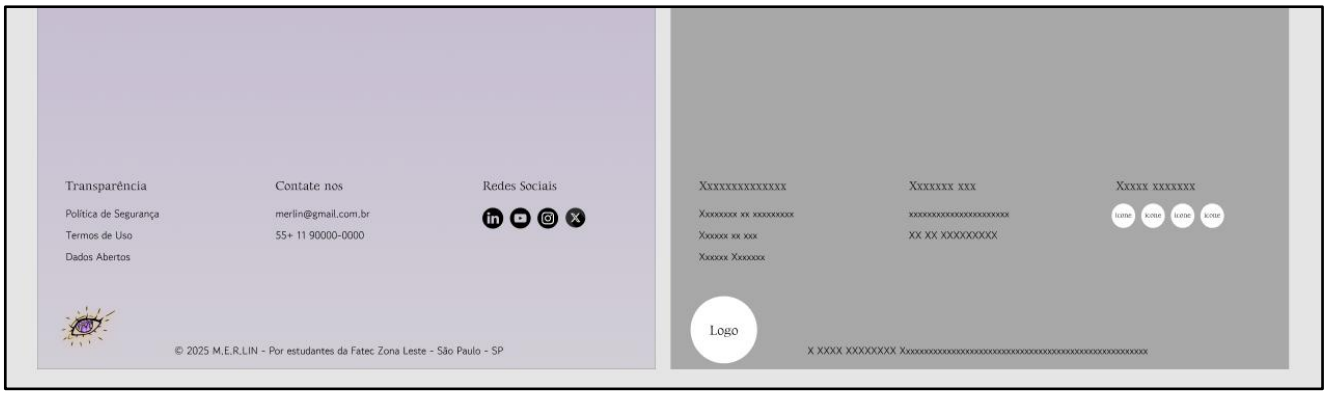
Figura 64 - Instalação site alta e baixa qualidade



Fonte: Autoria própria, 2025.

Por fim a última seção é o footer do site que exibe informações de política, contato e redes sociais. A parte possui 3 seções de texto, abaixo a logo e origem.

Figura 65 - Footer do site alta e baixa qualidade



Fonte: Autoria própria, 2025.

4 CONSIDERAÇÕES FINAIS

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.

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.

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.