Instituto Federal de Mato-grosso
Cel. Octayde Jorge da Silva - Campus cuiabá

Engenharia da Computação

OTONIEL SERAFIM SILVA

PEDRO FERNANDES BARBOSA COSTA

RELATÓRIO TÉCNICO DE DESENVOLVIMENTO: SIMULADOR DE CIRCUITOS DE CORRENTE ALTERNADA (DESKTOP 2.0)

Relatório apresentado como requisito parcial para avaliação na disciplina de Circuitos Elétricos II.

CUIABÁ-MT

2025

Simulador e Calculadora de Circuitos CA Hierárquicos....3
1. Resumo Executivo (Abstract)....3
2. Tecnologias Empregadas....3
3. Desafios de Visualização e Interface Gráfica....3
3.1. Integração Matplotlib em GUI (Execução Non-blocking)....3
3.2. Disparidade de Escala em Diagramas Fasoriais....5
3.3. Interatividade e Sobreposição de Rótulos (Draggable Labels)....6
4. Desafios de Lógica e Modelagem de Dados....6
4.1. Precisão Numérica e Validação de Entradas....6
4.2. Gerenciamento de Topologias Hierárquicas (Composite Pattern)....7
5. Refinamento de Interface e Documentação....7
5.1. Geração de Relatórios PDF....7
5.2. Representação Visual Estrutural (ASCII)....8
5.3. Modernização e Distribuição....9
6. Casos de Uso....9
6.1. UC-01 – Montar Circuito....9
6.2. UC-02 – Calcular Impedância Total....10
6.3. UC-03 – Gerar Diagrama Fasorial....10
6.4. UC-04 – Exportar Relatório em PDF....10
6.5. UC-05 – Resetar Circuito....11
6.1. UC-01 – Montar Circuito (Detalhado)....11
6.2. UC-02 – Calcular Impedância Total (Detalhado)....11
6.3. UC-03 – Gerar Diagrama Fasorial (Detalhado)....12
6.4. UC-04 – Exportar Relatório em PDF (Detalhado)....12
6.5. UC-05 – Resetar Circuito (Detalhado)....12
7. Cenários de Teste....13
8. Requisitos Funcionais e Não-Funcionais....14
8.1. Requisitos Funcionais (RF)....14
8.2. Requisitos Não-Funcionais (RNF)....15
9. Arquitetura do Sistema....15
9.1. Visão Geral....15
9.2. Módulos Principais....16
9.3. Diagrama de Classes (Representação Textual Simplificada)....16
9.4. Fluxo Interno de Cálculo....16
9.5. Fluxo de Renderização de Fasores....16
9.6. Justificativa da Arquitetura....17
10. Trabalhos Futuros....17
11. Limitações Atuais....18
12 – Glossário Técnico Expandido (Com Funções e Defasagens)....18
13. Conclusão....21

Simulador e Calculadora de Circuitos CA Hierárquicos

1. Resumo Executivo (Abstract)

Este documento apresenta o ciclo de desenvolvimento e melhoria do projeto apresentado no primeiro semestre de 2025 na disciplina de circuitos 2, os desafios técnicos enfrentados e as soluções adotadas durante a criação de uma ferramenta desktop para análise de circuitos de Corrente Alternada (CA). O software foi construído em Python 3.12 e emprega uma arquitetura orientada a objetos baseada no Padrão Composite, permitindo a resolução de impedâncias complexas em topologias arbitrárias série-paralelo. Além do cálculo numérico, o sistema oferece visualização gráfica interativa — incluindo diagramas fasoriais e curvas de Bode — e geração automática de relatórios técnicos em PDF.

2. Tecnologias Empregadas

- Linguagem: Python 3.12
- Interface Gráfica: Tkinter + ttkbootstrap (tema Darkly)
- Cálculo Numérico: NumPy e cmath
- Visualização: Matplotlib (backend TkAgg)
- Documentação: FPDF2

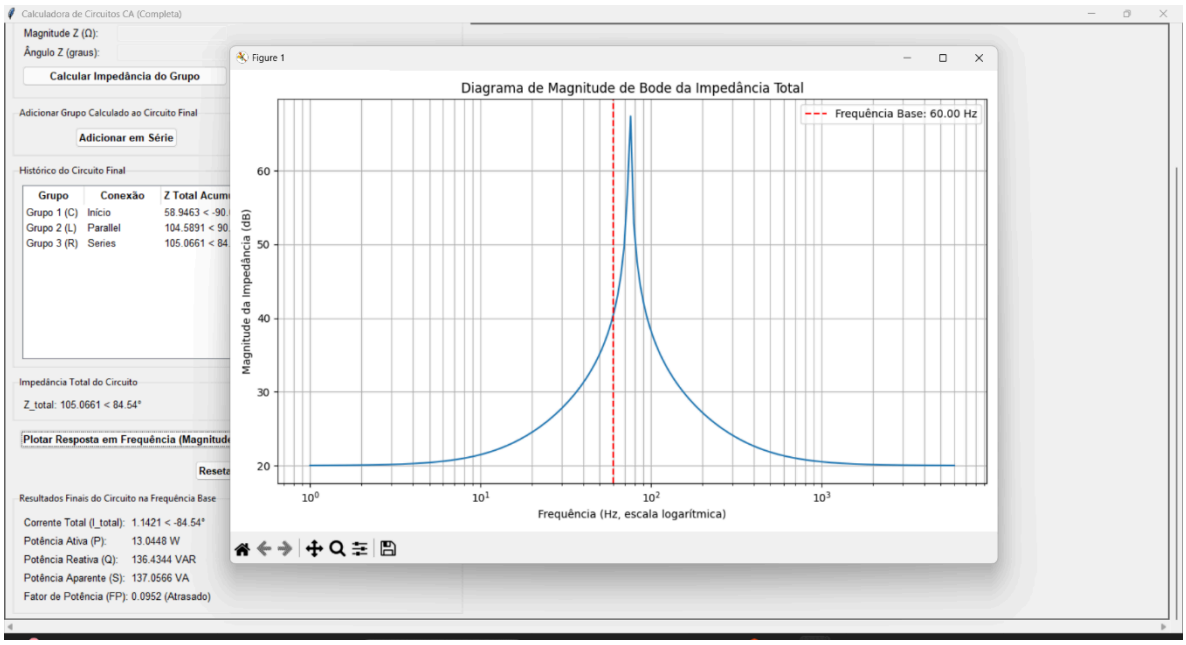
---

3. Desafios de Visualização e Interface Gráfica

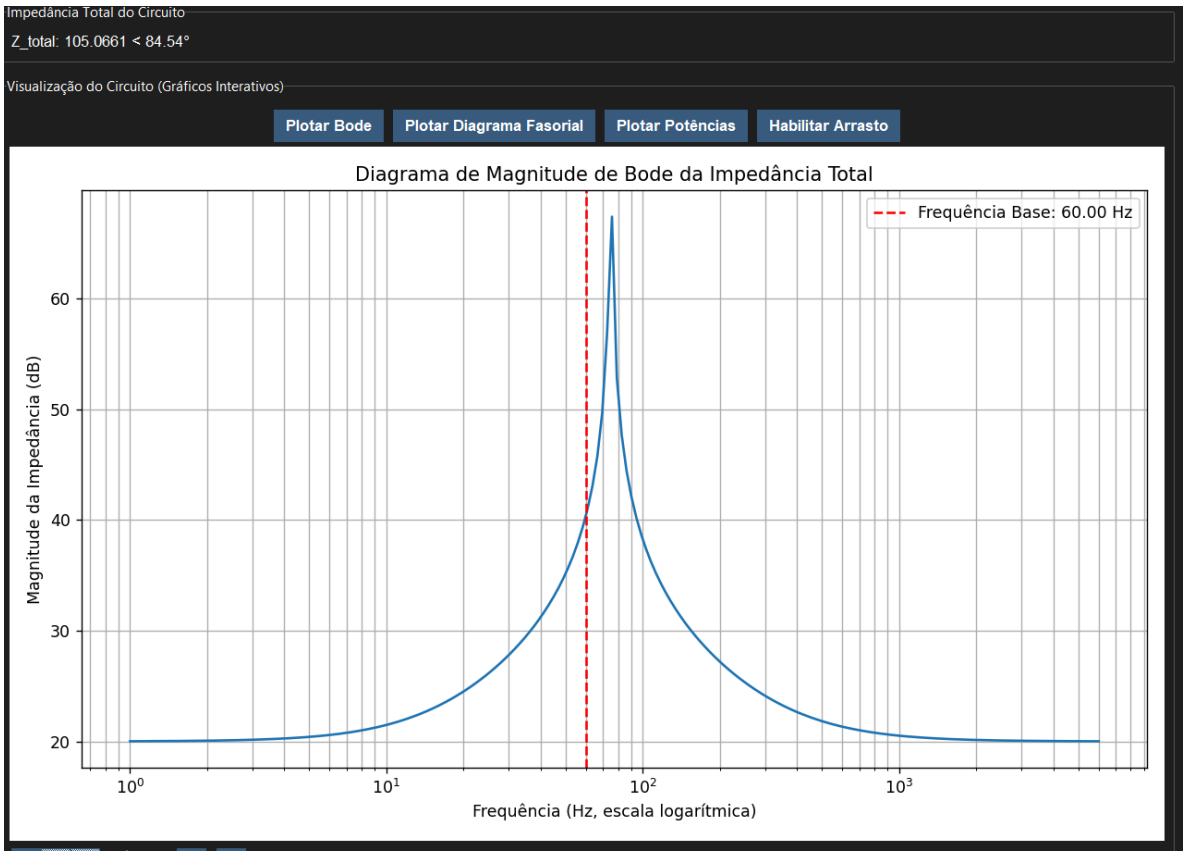
A implementação dos gráficos — especialmente os diagramas fasoriais — representou uma das etapas mais complexas do projeto devido a limitações inerentes ao Matplotlib e aos desafios de integração com o Tkinter.

3.1. Integração Matplotlib em GUI (Execução Non-blocking)

Problema: A utilização padrão de plt.show() bloqueava o mainloop do Tkinter, impedindo qualquer interação com a interface enquanto a janela do gráfico estivesse aberta.

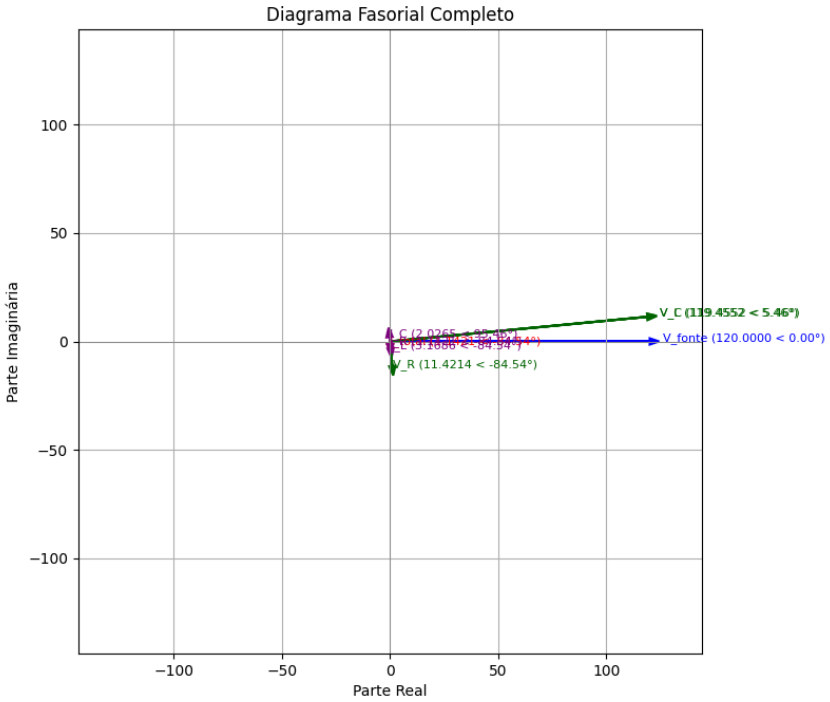


Solução: Substituição de plt.show() pela inserção direta da figura no GUI usando FigureCanvasTkAgg. Essa abordagem tornou o sistema responsivo, permitindo que o usuário interagisse livremente com os controles da interface enquanto visualizava gráficos em tempo real.



3.2. Disparidade de Escala em Diagramas Fasoriais

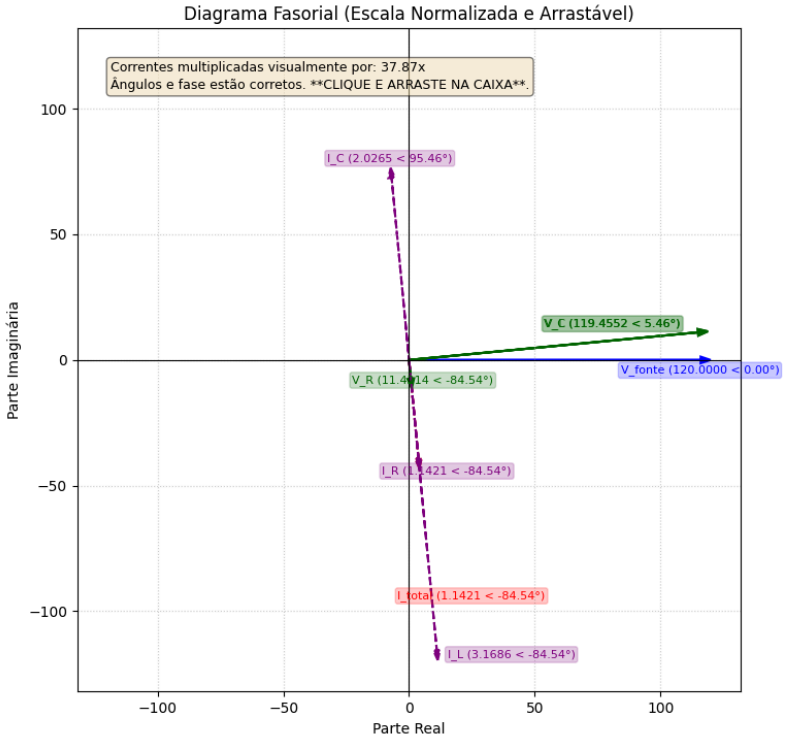
Problema: Em cenários reais, tensões típicas possuem magnitude significativamente maior que correntes correspondentes, tornando vetores de corrente invisíveis.



Solução: Implementação de um algoritmo de Normalização Visual Dinâmica, aplicando o fator:

k = Vmax / Imax

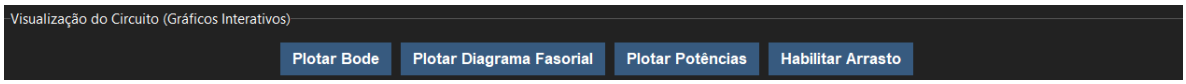
Isso ajusta o comprimento dos vetores exclusivamente para visualização, preservando ângulos e relações de fase.



3.3. Interatividade e Sobreposição de Rótulos (Draggable Labels)

Problema: Rótulos se sobrepunham quando fasores tinham ângulos similares, além de conflitos entre cliques e modos Pan/Zoom do Matplotlib.

Solução Final: Criação da classe DraggableAnnotation, com: - picker=10 - bbox translúcido - manipulação dos eventos de mouse (press, move, release) - botão “Habilitar Arrasto” que desabilita pan/zoom da toolbar



---

4. Desafios de Lógica e Modelagem de Dados

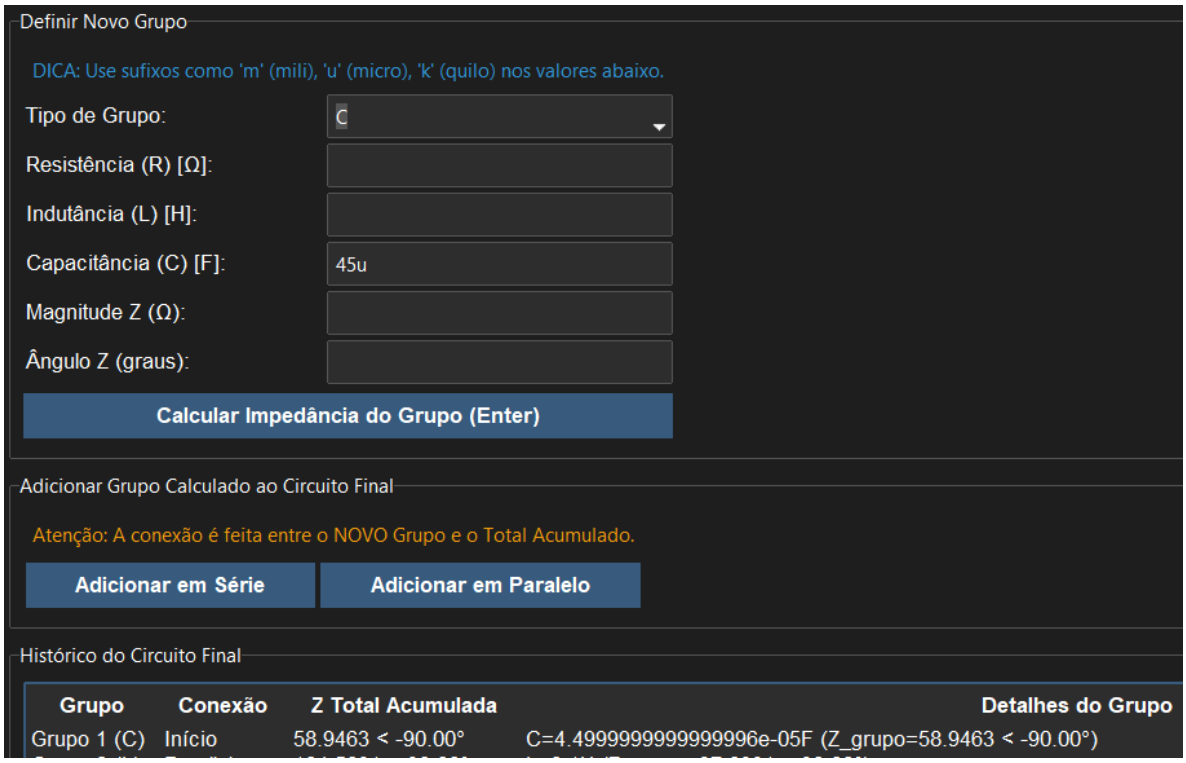
4.1. Precisão Numérica e Validação de Entradas

Problema: Durante testes em um circuito RLC Série, foram observadas discrepâncias significativas entre cálculos manuais e resultados do simulador.

Diagnóstico: Logs de depuração confirmaram que a matemática interna estava correta.

Causa Raiz: Erro de entrada — valor de capacitância digitado incorretamente.

Solução: Implementação da função parse_unit_input, permitindo valores como 47u, 10k, 100m e reduzindo erros humanos.



4.2. Gerenciamento de Topologias Hierárquicas (Composite Pattern)

Desafio: Necessidade de suportar topologias arbitrárias sem impedâncias inválidas.

Solução: Aplicação do padrão Composite com classes Component e CircuitGroup, além de um reset_circuit robusto para limpar estados anteriores.

---

5. Refinamento de Interface e Documentação

5.1. Geração de Relatórios PDF

Problema: FPDF2 falhava ao renderizar símbolos Unicode como Ω, μ e ∠.

Solução: Criação de safe_content, convertendo símbolos para equivalentes textuais seguros (Ω → Ohm, μ → u).

```
================================================================
                RELATÓRIO DE CÁLCULO DE CIRCUITO CA
================================================================
Data e Hora: 2025-11-23 17:34:03
----------------------------------------------------------------
PARÂMETROS DE ENTRADA
Tensão da Fonte (V_rms): 120.0 V @ 0.0deg
Frequência Base (f): 60.0 Hz
Frequência Angular (w): 376.9911 rad/s
----------------------------------------------------------------
HISTÓRICO DE REDUÇÃO (PROVA DO CÁLCULO)
  1. Grupo: Grupo 1 (C)
     - Elementos: C=4.4999999999999996e-05F
     - Conexão: Início
     - Z do Grupo: 58.9463 < -90.00deg
     - Z TOTAL Acumulada: 58.9463 < -90.00deg
  2. Grupo: Grupo 2 (L)
     - Elementos: L=0.1H
     - Conexão: Parallel
     - Z do Grupo: 37.6991 < 90.00deg
     - Z TOTAL Acumulada: 104.5891 < 90.00deg
  3. Grupo: Grupo 3 (R)
     - Elementos: R=10.0Ohm
     - Conexão: Series
     - Z do Grupo: 10.0000 < 0.00deg
     - Z TOTAL Acumulada: 105.0661 < 84.54deg
----------------------------------------------------------------
RESULTADOS FINAIS NA FREQUÊNCIA BASE
Z Total Final: 105.0661 < 84.54deg Ohm
I Total: 1.1421 < -84.54deg A
Potência Ativa (P): 13.0448 W
Potência Reativa (Q): 136.4344 VAR (Indutivo)
Potência Aparente (S): 137.0566 VA
Fator de Potência (FP): 0.0952
```

5.2. Representação Visual Estrutural (ASCII)

Tentativa: Criar diagramas ASCII automáticos.

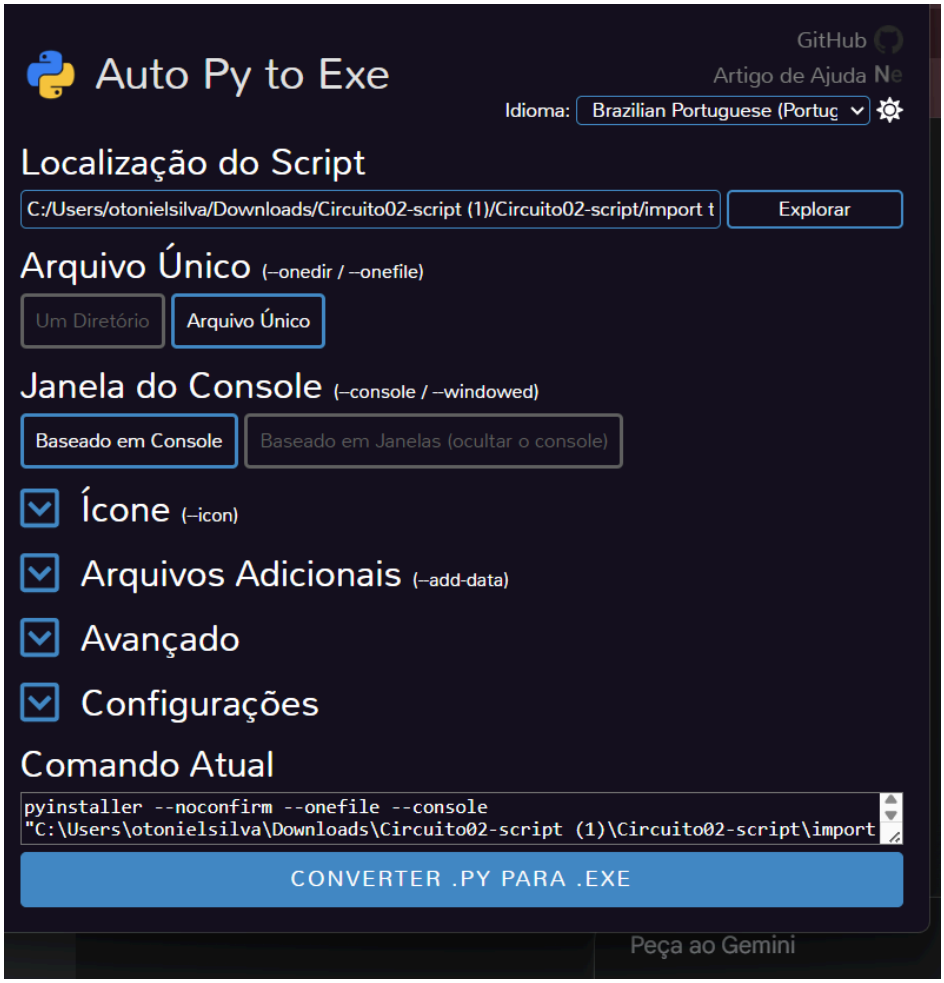
Obstáculo: Topologias recursivas profundas inviabilizaram o recurso.

Decisão: Usar ASCII apenas para casos simples; usar Histórico de Redução passo a passo para circuitos complexos.

5.3. Modernização e Distribuição

Interface: Migração completa para ttkbootstrap com tooltips e legendas coloridas.

Distribuição: Empacotamento via PyInstaller, gerando um único executável portátil.



---

6. Casos de Uso

6.1. UC-01 – Montar Circuito

Ator: Usuário

Objetivo: Construir um circuito CA arbitrário utilizando componentes básicos (R, L, C) e agrupamentos série/paralelo.

Fluxo Principal: 1. Usuário seleciona “Adicionar Componente”. 2. Escolhe R, L ou C. 3. Insere valor (com suporte a unidades: 10k, 47u, etc.). 4. Seleciona “Adicionar ao Grupo”. 5. Usuário pode criar grupos Série ou Paralelo. 6. O sistema atualiza a árvore do circuito.

Exceções: - Valor inválido → sistema exibe erro de validação. - Topologia inconsistente → sistema bloqueia a conexão.

6.2. UC-02 – Calcular Impedância Total

Ator: Usuário

Objetivo: Executar o cálculo de Z total do circuito.

Fluxo Principal: 1. Usuário clica “Calcular Impedância”. 2. O sistema percorre a estrutura Composite. 3. Calcula reatâncias e combinações série/paralelo. 4. Exibe Z complexa, módulo e fase.

Exceções: - Circuito vazio → erro. - Grupo incompleto → impedir cálculo.

6.3. UC-03 – Gerar Diagrama Fasorial

Ator: Usuário

Objetivo: Visualizar fasores de tensão e corrente.

Fluxo Principal: 1. Usuário seleciona “Gerar Fasores”. 2. Sistema normaliza vetores. 3. Matplotlib renderiza diagrama. 4. Usuário pode arrastar rótulos.

Exceções: - Sem cálculo prévio → solicitar cálculo.

6.4. UC-04 – Exportar Relatório em PDF

Ator: Usuário

Objetivo: Exportar relatório técnico com resultados.

Fluxo Principal: 1. Usuário aciona “Exportar PDF”. 2. Sistema sanitiza conteúdo. 3. PDF é gerado via FPDF2. 4. Sistema confirma exportação.

Exceções: - Conteúdo incompatível → correção automática.

6.5. UC-05 – Resetar Circuito

Ator: Usuário

Objetivo: Limpar montagem atual e iniciar nova.

Fluxo Principal: 1. Usuário clica “Resetar Circuito”. 2. reset_circuit limpa a árvore. 3. Interface é atualizada.

6.1. UC-01 – Montar Circuito (Detalhado)

Ator: Usuário

Objetivo: Construir um circuito arbitrário utilizando componentes R, L, C e agrupamentos hierárquicos.

Pré-condições: - A aplicação deve estar aberta. - O circuito atual pode estar vazio ou parcialmente montado.

Pós-condições: - A árvore de circuito é atualizada com o novo componente ou grupo.

Fluxo Principal: 1. Usuário seleciona “Adicionar Componente”. 2. Escolhe o tipo (R, L, C). 3. Insere o valor com ou sem sufixo (ex: 10k, 47u). 4. Seleciona “Adicionar ao Grupo Atual”. 5. Opcionalmente cria grupos Série ou Paralelo. 6. Sistema atualiza e exibe a estrutura do circuito.

Fluxos Alternativos: - 3A. Valor inválido → o sistema exibe erro e solicita correção. - 5A. Grupo inconsistente → sistema bloqueia a conexão e sugere correção.

6.2. UC-02 – Calcular Impedância Total (Detalhado)

Ator: Usuário

Objetivo: Calcular a impedância total do circuito montado.

Pré-condições: - Deve existir ao menos um componente na árvore.

Pós-condições: - Z total é computada e exibida.

Fluxo Principal: 1. Usuário clica em “Calcular Impedância”. 2. Sistema percorre a árvore Composite. 3. Calcula impedâncias individuais. 4. Aplica combinações série/paralelo. 5. Exibe Z complexa, módulo e fase.

Fluxos Alternativos: - 1A. Circuito vazio → sistema exibe erro. - 2A. Grupo incompleto → sistema solicita ajustes.

6.3. UC-03 – Gerar Diagrama Fasorial (Detalhado)

Ator: Usuário

Objetivo: Exibir fasores de tensão e corrente com interatividade.

Pré-condições: - A impedância já deve ter sido calculada.

Pós-condições: - Janela com fasores exibida.

Fluxo Principal: 1. Usuário seleciona “Gerar Fasores”. 2. Sistema calcula amplitudes e ângulos. 3. Aplica normalização visual. 4. Renderiza diagrama no Matplotlib. 5. Habilita arraste de anotações.

Fluxos Alternativos: - 1A. Nunca calculou impedância → solicitar cálculo.

6.4. UC-04 – Exportar Relatório em PDF (Detalhado)

Ator: Usuário

Objetivo: Exportar relatório técnico contendo todos os dados.

Pré-condições: - Deve existir cálculo realizado ou circuito montado.

Pós-condições: - PDF válido é salvo.

Fluxo Principal: 1. Usuário clica em “Exportar PDF”. 2. Sistema sanitiza símbolos. 3. Gera PDF via FPDF2. 4. Exibe confirmação.

Fluxos Alternativos: - 2A. Conteúdo inválido → sanitização automática.

6.5. UC-05 – Resetar Circuito (Detalhado)

Ator: Usuário

Objetivo: Limpar completamente o circuito atual.

Pré-condições: - Aplicação aberta.

Pós-condições: - Circuito limpo, árvore reiniciada.

Fluxo Principal: 1. Usuário clica “Resetar Circuito”. 2. Sistema executa reset_circuit. 3. Interface é atualizada.

---

7. Cenários de Teste

7.1. CT-01 – Circuito Simples (R apenas)

Objetivo: Validar cálculo básico de impedância real.

Entrada: R = 100Ω

Resultado Esperado: Z = 100 + j0 Ω

Critério de Aprovação: Sistema calcula sem erros.

7.2. CT-02 – Circuito RL Série

Entrada: R = 10Ω, L = 0.2H, f = 60Hz

Resultado Esperado: Z = R + jωL

Critério de Aprovação: Módulo e fase corretos.

7.3. CT-03 – Circuito RC Paralelo

Entrada: R = 1kΩ, C = 10μF

Resultado Esperado: Z obtida pela combinação paralela.

Critério de Aprovação: Sem divisão por zero.

7.4. CT-04 – Série contendo Paralelo Aninhado

Topologia: R série (L | C)

Objetivo: Validar árvore Composite.

Critério: Sistema percorre todos os níveis corretamente.

7.5. CT-05 – Circuito Altamente Aninhado (5 níveis)

Entrada: Grupo série contendo múltiplos paralelos internos.

Objetivo: Testar estabilidade em circuitos complexos.

Critério: Sem falhas de recursão; desempenho aceitável.

7.6. CT-06 – Erro de Unidade (Input Malformado)

Entrada: “0.0006” quando o usuário quis dizer “60uF”.

Objetivo: Testar parse_unit_input.

Critério: Sistema identifica e rejeita ou converte corretamente.

7.7. CT-07 – Diagrama Fasorial com Escalas Extremas

Entrada: Correntes muito pequenas; tensões grandes.

Objetivo: Validar normalização visual.

Critério: Vetores visíveis e ângulos preservados.

7.8. CT-08 – Exportação PDF com Símbolos

Entrada: Conteúdo contendo Ω, μ, ∠.

Objetivo: Garantir sanitização adequada.

Critério: PDF abre sem erros.

7.9. CT-09 – Teste de Interatividade (Draggable Labels)

Entrada: Múltiplos fasores com ângulos próximos.

Objetivo: Testar arraste sem conflito com Pan/Zoom.

Critério: Rótulos se movem corretamente.

7.10. CT-10 – Reset Completo

Entrada: Circuito complexo seguido de Reset.

Objetivo: Garantir limpeza total da árvore.

Critério: Nenhum resíduo de topologia permanece.

---

8. Requisitos Funcionais e Não-Funcionais

8.1. Requisitos Funcionais (RF)

RF-01 — Montagem de Circuitos
O sistema deve permitir ao usuário criar circuitos arbitrários usando resistores, indutores e capacitores, incluindo agrupamento série e paralelo.

RF-02 — Validação de Entradas
O sistema deve validar valores numéricos, suportar unidades simplificadas (10k, 47u, 100m etc.) e impedir valores inválidos.

RF-03 — Cálculo de Impedância Total
O sistema deve calcular a impedância equivalente de qualquer topologia hierárquica válida.
RF-04 — Visualização de Fasores
O sistema deve gerar diagramas fasoriais com vetores normalizados e rótulos arrastáveis.
RF-05 — Plot de Resposta em Frequência (Bode)
O sistema deve permitir visualizar magnitude e fase da impedância em função da frequência.
RF-06 — Histórico de Redução
O sistema deve exibir passo a passo as reduções série/paralelo quando solicitado.
RF-07 — Exportação de Relatórios
O sistema deve gerar relatórios técnicos em PDF contendo impedâncias, tabelas, parâmetros e observações.
RF-08 — Reset do Circuito
O sistema deve permitir resetar toda a árvore do circuito e iniciar uma nova montagem.

8.2. Requisitos Não-Funcionais (RNF)

RNF-01 — Precisão Numérica
A simulação deve operar com precisão compatível com cálculos de engenharia, usando ponto flutuante de dupla precisão.
RNF-02 — Usabilidade
A interface deve ser intuitiva, responsiva e compatível com telas HD.
RNF-03 — Performance
Operações de montagem e cálculo devem ocorrer em menos de 200 ms para circuitos de até 300 componentes.
RNF-04 — Portabilidade
O aplicativo deve funcionar em Windows 10+ sem instalação de bibliotecas externas (graças ao uso de PyInstaller).
RNF-05 — Confiabilidade
O sistema deve emitir mensagens claras de erro e impedir topologias inconsistentes.
RNF-06 — Interoperabilidade Visual
Gráficos devem ser integrados ao Tkinter sem travar a interface (uso obrigatório de FigureCanvasTkAgg).
RNF-07 — Manutenibilidade
A arquitetura do código deve seguir padrões OOP (Composite + Strategy) para facilitar extensão e testes.

9. Arquitetura do Sistema

9.1. Visão Geral

A arquitetura do simulador segue um modelo modular orientado a objetos, com separação clara entre interface gráfica, núcleo de cálculos elétricos, utilitários e geração de relatórios.

O uso do padrão Composite permite a criação de circuitos hierárquicos arbitrários, enquanto o padrão Strategy assegura flexibilidade na implementação de cálculos específicos.

9.2. Módulos Principais

- **UI (Interface Gráfica):** Responsável pelos elementos de interação do Tkinter + ttkbootstrap, inserção dos gráficos e controle de eventos.
- **Core (Motor de Cálculo):** Implementa classes de componentes (R, L, C) e grupos série/paralelo via Composite.
- **Graphics:** Lida com geração de fasores, diagramas Bode e manipulação de rótulos arrastáveis.
- **Utils:** Funções auxiliares como parse_unit_input, validações e sanitização de conteúdo.
- **Reporting:** Geração de PDF, conversão de símbolos e formatação de conteúdos técnicos.

9.3. Diagrama de Classes (Representação Textual Simplificada)

Component (abstract)
├── Resistor
├── Inductor
├── Capacitor
└── CircuitGroup
    ├── SerieGroup
    └── ParallelGroup

9.4. Fluxo Interno de Cálculo

1. Usuário monta a árvore do circuito.
2. O método get_impedance(f) é chamado no nó raiz.
3. Cada nó delega o cálculo para seus filhos (Composite).
4. Grupos Série → soma direta das impedâncias.
5. Grupos Paralelo → combinação pela fórmula:
   Zeq=(1/Z1+1/Z2+...)−1Z_{eq} = (1/Z_1 + 1/Z_2 + ...)^{-1}Zeq=(1/Z1+1/Z2+...)−1
6. O valor final retorna ao controlador, que o exibe para o usuário.

9.5. Fluxo de Renderização de Fasores

1. Sistema coleta tensões e correntes calculadas.
2. Normaliza amplitudes (Vmax/Imax).
3. Gera figura Matplotlib.
4. Passa para UI via FigureCanvasTkAgg.
5. Anotações são tornadas arrastáveis pelo usuário.

9.6. Justificativa da Arquitetura

- Composite permite suportar qualquer profundidade e complexidade de circuito.
- Separação de módulos facilita manutenção, testes e reuso.
- Integração Matplotlib/Tkinter exigiu camadas distintas para desacoplamento

10. Trabalhos Futuros

10.1. Implementação de Diagramas de Bode

Embora parte da estrutura já esteja preparada, a visualização completa de Bode (magnitude + fase em função da frequência) pode ser integrada ao módulo Graphics.

Recursos previstos:

- Escalas logarítmicas automáticas.
- Varredura de frequência configurável.
- Marcação automática de frequência de ressonância.

10.2. Otimização de Desempenho

Para circuitos muito profundos (10+ níveis), podem ser exploradas técnicas adicionais:

- *Memoization* de subcircuitos repetidos
- Uso opcional de NumPy vetorizado na varredura de frequência
- Conversão parcial para Cython em pontos críticos

10.3. Exportação em Outros Formatos

Além do PDF, pode ser adicionada exportação para:

- **SVG** (diagramas fasoriais vetoriais)
- **JSON** (estrutura completa do circuito)
- **LaTeX** (relatórios acadêmicos)

10.4. Biblioteca de Modelos Prontos

Criar uma *Library* que inclua padrões:

- Filtros passa-alta/baixa
- Filtros RLC sintonizados
- Divisores de tensão
- Circuitos equivalentes famosos (Thevenin/Norton)

Usuário poderia arrastar modelos prontos para o circuito principal.

10.5. Modo Escuro Inteligente e Temas Customizáveis

Estender o sistema de UI para permitir:

- Seleção de temas (Claro, High-Contrast, Azul, etc.)
- Detecção automática do tema do sistema operacional

- Ajuste dinâmico de cores nos gráficos

10.6. Testes Automatizados e Integração Contínua

Implementar uma suíte de testes com **pytest** e habilitar CI via GitHub Actions.

Cobertura recomendada:

- parse_unit_input
- cálculos de impedância
- composição série/paralelo
- geração de PDF
- normalização de fasores

10.7. Módulo Educacional (Modo Didático)

Um modo especial que exibe:

- Passo a passo do cálculo simbólico
- Explicações automáticas sobre o comportamento do circuito
- Destaque de fasores com cores didáticas
- Animações mostrando fases e amplitudes variando

Voltado para ensino de circuitos básicos e AC.

10.8. Segurança, Logs e Tratamento de Falhas

Mesmo sendo um software local, sistemas de engenharia se beneficiam de:

- Logs estruturados (níveis: INFO, WARNING, ERROR)
- Stack trace amigável para o usuário
- Arquivo de log persistente em caso de falha crítica
- Modo de recuperação (fallback)

Por que é relevante?

Permite depuração mais fácil e documentação de instabilidades

11. Limitações Atuais

Seção simples, mas essencial:

- Não suporta transitório
- Não suporta fontes dependentes
- Não simula potência harmônica
- Normalização visual não reflete grandezas reais
- Não há layouts exportáveis de circuito

12 – Glossário Técnico Expandido (Com Funções e Defasagens)

Componentes Elétricos

Resistor (R)

Função: Dissipar energia em forma de calor.
Comportamento em CA:

- Não atrasa nem adianta a corrente (defasagem = 0°).
- Tensão e corrente permanecem em fase.
  Para que serve no sistema:
- Cálculo de impedância real.
- Construção do diagrama fasorial (parte real).
- Determinação de potência ativa (P).

Capacitor (C)
Função: Armazenar energia no campo elétrico.
Comportamento em CA:

- A corrente **adianta** a tensão em 90°.
- Impedância diminui com a frequência ($Xc = 1/(\omega C)$).
  Para que serve no sistema:
- Cálculo de reatância capacitiva.
- Fasores com componente imaginária negativa.
- Contribuição para potência reativa **capacitiva** (Qc).

Indutor (L)
Função: Armazenar energia no campo magnético.
Comportamento em CA:

- A corrente **atrasa** a tensão em 90°.
- Impedância cresce com frequência ($Xl = \omega L$).
  Para que serve no sistema:
- Cálculo de reatância indutiva.
- Fasores imaginários positivos.
- Contribuição para potência reativa **indutiva** (Ql).

Fonte CA
Função: Fornecer tensão alternada ao circuito.
Comportamento:

- Define amplitude e frequência para todos os cálculos.
  Para que serve no sistema:
- Geração dos fasores.
- Cálculo das impedâncias dependentes de frequência.
- Geração de gráficos (como Bode).

Conceitos de Conexão
Série
O que significa:
Corrente é igual para todos os componentes.
Impacto no sistema:

- Impedâncias somam diretamente.

- Estrutura mais simples no parser.

Paralelo
O que significa:
Tensão é igual para todos os ramos.
Impacto no sistema:

- Impedâncias combinam como inverso da soma dos inversos.
- Necessário validar ramos e sub-ramos no parser.

Elementos da Análise
Impedância (Z)
O que é:
Combinação entre resistência (R) e reatância (X).
Representação: número complexo R + jX
Função no sistema:

- Base de todos os cálculos elétricos.
- Define fasores, potência e gráficos.

Fasor
O que é:
Representação vetorial de grandezas em CA (módulo e ângulo).
Função no sistema:

- Gera o **diagrama fasorial**.
- Mostra defasagens entre tensão e corrente.
- Facilita entendimento visual dos efeitos de L e C.

Gráficos e Ferramentas de Análise
Gráfico de Bode (Módulo e Fase)
O que é:
Representação da resposta em frequência do circuito.
Mostra na prática:

- **Módulo:** como a magnitude da tensão/corrente muda com frequência.
- **Fase:** como o circuito atrasa ou adianta sinais.

Para que serve no sistema:

- Analisar comportamentos de filtros (mesmo que simples).
- Ver transição entre comportamento indutivo e capacitivo.
- Diagnosticar ressonância.

Diagrama Fasorial
O que é:
Gráfico que mostra tensões e correntes como vetores.
Serve para:

- Visualizar relações de defasagem.
- Entender efeitos combinados de R, L e C.

- Avaliar potência real/reac tiva/total.

Triângulo de Potências
O que é:
Representação geométrica da potência ativa (P), reativa (Q) e aparente (S).
No sistema:
- P → potência ativa (resistiva)
- Q → potência reativa (indutiva ou capacitiva)
- S → potência total
- FP = P / S (fator de potência)

Serve para:
- Ver como R, L e C influenciam no consumo.
- Avaliar indutância e capacitância predominantes.
- Demonstrar equilíbrio ou desequilíbrio do circuito.

Outros Termos da Arquitetura do Sistema
Árvore de Circuito
Estrutura hierárquica usada para representar combinações de série e paralelo.
Ramo
Segmento específico num paralelo.
Parser
Interpreta a estrutura de texto e monta o circuito internamente.
Validação
Confere se o circuito tem formato correto e conexões completas.
Complexidade
Classificação automática do seu sistema (baixa, média, alta).
Relatório
Documento final contendo componentes, estrutura e gráficos.

13. Conclusão

O projeto de melhoria do simulador de circuito desktop apresentado no primeiro semestre de 2025 resultou em uma ferramenta estável, precisa e profissional. Os principais desafios — interatividade gráfica e validação de entrada — foram superados com engenharia orientada a objetos, tratamento de eventos customizados e refinamentos de UX. A aplicação entrega resultados confiáveis para simulação de circuitos CA em regime permanente.