Politécnico do Porto
Instituto Superior de Engenharia do Porto

U=RIsolve Geração Automática de Modelos de Circuitos Elétricos

Rodrigo Ferrero

LEEC
Licenciatura em Engenharia Eletrotécnica e de Computadores

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Fevereiro, 2025

Candidato: Rodrigo Ferrero, Nº 1211629, 1211629@isep.ipp.pt

Orientação Científica: André Rocha, anr@isep.ipp.pt

Coorientação Científica: Mário Alves, mjf@isep.ipp.pt

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Fevereiro, 2025

Agradecimentos

Declaro nesta secção os meus agradecimentos a todos os que contribuíram para o desenvolvimento do projeto:
Aos professores André Rocha e Mário Alves, cuja orientação foi indispensável para a realização deste projeto.
Aos meus pais, pelo constante apoio e dedicação ao longo de toda a minha jornada. Agradeço pela paciência, pela motivação e por acreditarem em mim em todos os momentos.
Ao meu amigo Ricardo Ferreira, pela ajuda e orientação na aprendizagem das linguagens de programação utilizadas neste projeto.

Resumo

Este relatório documenta o projeto do desenvolvimento de uma aplicação interativa destinada a apoiar os estudantes de Engenharia Eletotécnica na auto-aprendizagem da análise e compreensão de circuitos elétricos. Esta necessidade foi sendo apercebida ao longo dos anos pelos docentes das disciplinas de Fundamentos da Engenharia Electrotécnica e Teoria de Circuitos. Assim, iniciou-se a criação de uma ferramenta que auxiliasse os estudantes a desenvolver as suas aptidões nestas temáticas e que os profissionais na área também pudessem usufruir de uma aplicação de análise de circuitos. Esta aplicação é denominada *U=RIsolve Academy* e já teve várias versões, mas encontra-se ainda em desenvolvimento, para futuramente disponibilizar aos estudantes uma versão robusta, mais completa (em termos de funcionalidades) e de utilização mais amigável.

O objetivo deste projeto foi desenvolver um módulo/aplicação de *software*, chamada Geração Automática de Modelos de circuitos Elétricos (GAME), a qual tenciona coexistir com a plataforma *U=RIsolve Academy*, complementando o Editor de Circuitos que se encontra em fase de desenvolvimento. Assim, o objetivo principal foi proporcionar uma ferramenta que automatiza a geração de modelos de circuitos com base em parâmetros definidos pelo utilizador, como o número de componentes, tipo de circuito (*Direct Current* (DC) ou *Alterning Current* (AC)) e topologia desejada. A arquitetura desta aplicação *web* foi desenhada com enfoque na acessibilidade e simplicidade de utilização, suportando múltiplos navegadores. A interface de utilizador foi implementada com *HyperText Markup Language* (HTML), Cascading Style Sheets (CSS) e *JavaScript*, enquanto os cálculos e algoritmos subjacentes foram desenvolvidos em *Python*. Os testes realizados confirmaram a eficácia do sistema em cenários diversos, desde circuitos simples a configurações complexas, o que provou que esta aplicação *web* se tornou numa ferramenta útil na geração de circuitos. Estes circuitos gerados podem ser usados para aprendizagem dos métodos de análise, ou em exercícios onde o estudo dos nós ou dos ramos é relevante.

Palavras-Chave: U=RIsolve Academy, Editor de Circuitos, GAME, ligação, ponto de conexão, simulador de circuitos, modelo JSON, autoaprendizagem, e-learning, modelo analítico

Abstract

This report documents the development of an interactive application designed to support Electrical Engineering students in self-learning the analysis and understanding of electrical circuits. Over the years, the need for such a tool has been recognized by the faculty of the Fundamentals of Electrical Engineering and Circuit Theory courses. As a result, the development of a tool that assists students in enhancing their skills in these subjects was initiated, while also providing professionals in the field with an application for circuit analysis. This application is called U=RIsolve Academy and has undergone several versions; however, it is still under development to provide students with a more robust, feature-rich, and user-friendly version in the future.

The goal of this project was to develop a software module/application, referred to as Automatic Generation of Electrical Circuit Models, which is intended to coexist with the U=RIsolve Academy platform, complementing the Circuit Editor that is currently under development. The primary objective was to provide a tool that automates the generation of circuit models based on user-defined parameters, such as the number of components, circuit type (*Direct Current* (DC) or *Alterning Current* (AC)), and desired topology.

The architecture of this web application was designed with a focus on accessibility and ease of use, supporting multiple browsers. The user interface was implemented using *HyperText Markup Language* (HTML), Cascading Style Sheets (CSS), and JavaScript, while the underlying calculations and algorithms were developed in Python. The tests conducted confirmed the system's effectiveness across various scenarios, ranging from simple circuits to complex configurations. These results demonstrated that this web application has become a valuable tool for generating circuits that can be used for learning analysis methods or in exercises where studying nodes or branches is relevant.

Keywords: U=RIsolve Academy, Circuit Editor, GAME, connection, connection point, circuit simulator, JSON model, self-learning, e-learning, analytical model

Índice

Lista de Figuras

Lista de Tabelas

Glossário

componente

Qualquer elemento individual elétrico de um circuito com dois terminais que pode ser conectado a uma ligação.

grafo

Representação visual da interligação entre nós e ligações [1][2].

ligação

Conexão entre os terminais de dois elementos.

nó

Ponto de convergência de três ou mais ramos, composto por um ou mais pontos de conexão [3].

ponto de conexão

Interligação entre três ou mais ligações, sejam elas ramos ou não.

ramo

Conjunto de um ou mais componentes que pertencem a uma ligação [3].

netlist

Ficheiro de texto que representa um circuito, gerada através de um simulador de circuitos [4].

schematic

Ficheiro que representa esquematicamente as ligações, os componentes e as interligações entre eles, gerada através de um simulador de circuitos [4].

spanning tree

Caminho possível de um grafo que mantém todos os vértices conectados sem haver caminhos fechados, sendo composto pelo número mínimo de ligações necessárias para isso [1][2].

Lista de Acrónimos

AC	*Alterning Current*
AI	*Artificial Intelligence*
CSS	*Cascading Style Sheets*
DC	*Direct Current*
GAME	Geração Automática de Modelos de circuitos Elétricos
HTML	*HyperText Markup Language*
ISEP	Instituto Superior de Engenharia do Porto
JSON	*JavaScript Object Notation*
LEEC	Licenciatura em Engenharia Eletrotécnica e de Computadores
NLP	*Natural Language Processing*
PDF	*Portable Document Format*
QUCS	*Quite Universal Circuit Simulator*

Lista de Símbolos

Símbolo	Descrição	Unidades
L	Indutância magnética	H
C	Capacidade de um condensador	F
I	Corrente elétrica	A
U	Tensão elétrica	V
f	Frequência	Hz
R	Resistência elétrica	Ω

Capítulo 1

Introdução

1.1 Contextualização

Durante o primeiro ano da Licenciatura em Engenharia Eletrotécnica e de Computadores (LEEC) no Instituto Superior de Engenharia do Porto (ISEP), existem alunos com conhecimentos praticamente nulos de circuitos elétricos, algo que é fundamental para um futuro engenheiro eletrotécnico. Esta realidade destaca a necessidade de ferramentas que possam auxiliar o aluno a superar essas dificuldades.

Foi neste contexto que surgiu a ideia de desenvolver uma aplicação inovadora, capaz de fornecer uma ferramenta interativa que permita aos alunos reforçar os conhecimentos adquiridos nas aulas, ao mesmo tempo em que promove uma aprendizagem mais prática e eficaz.

Este projeto parte de uma aplicação/portal de *eLearning* em desenvolvimento, denominada *U=RIsolve Academy*, que irá disponibilizar brevemente um Editor de Circuitos, onde o utilizador poderá desenhar circuitos elétricos de maneira simples e dentro do próprio ambiente *web* da aplicação.

Esta aplicação resulta de um projeto/linha de investigação que contou com a colaboração de muitos colegas/alunos do ISEP, estando sob a coordenação dos professores Mário Alves e André Rocha.

1.2 Ferramentas Disponíveis

1.2.1 *U=RIsolve Academy*

O *U=RIsolve Academy*[1] é uma aplicação desenvolvida para facilitar e ajudar na aprendizagem da análise de circuitos elétricos. Através desta ferramenta, os utilizadores poderão estudar e compreender conceitos fundamentais da "Teoria de Circuitos", utilizando exemplos práticos e interativos. A plataforma permite a resolução passo-a-passo, e o utilizador pode assim explorar e aprender a resolução dos mesmos. A plataforma permite que o utilizador explore a análise de circuitos, solucionando problemas com o auxílio de uma interface simples e intuitiva.
A Figura 1.1 mostra a área de treino da aplicação (atualmente em desenvolvimento) onde mostra as funcionalidades da mesma.

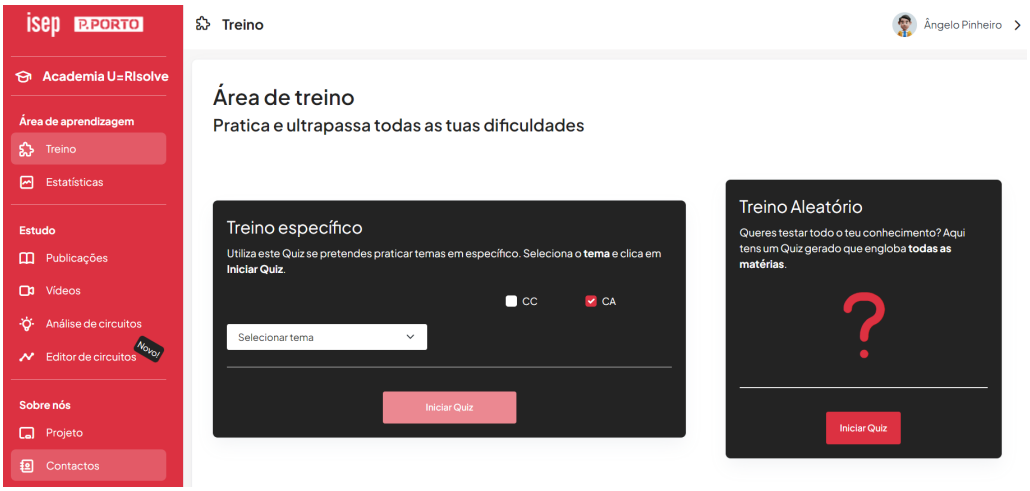


Figura 1.1: Área de treino do *U=RIsolve Academy*

A aplicação foi projetada principalmente para estudantes de LEEC que desejam consolidar ou expandir os seus conhecimentos sobre circuitos elétricos.
O *U=RIsolve Academy* dá continuidade ao *U=RIsolve Simulator*, que é uma aplicação explicitamente para análise de circuitos elétricos que se encontra atualmente online e disponível para uso, embora esteja em fase de reestruturação e consolidação. O simulador é capaz de realizar a análise passo-a-passo de circuitos elétricos usando diferentes métodos: Método da Tensão dos Nós, Método das Correntes nos Ramos e Método das Correntes nas Malhas. Estes métodos são cruciais para a devida compreensão de circuitos elétricos e esta ferramenta mostra todos os passos para analisar um circuito de acordo com o método escolhido.

[1] A aplicação está disponível através do seguinte *link* (temporário): http://cloud.microlumin.com:3020/home.

A Figura 1.2 mostra um exemplo da análise de um circuito, com recurso ao *U=RIsolve Simulator*, onde se nota a explicação detalhada dos primeiros passos para analisar devidamente o circuito escolhido, utilizando o Método da Tensão nos Nós.

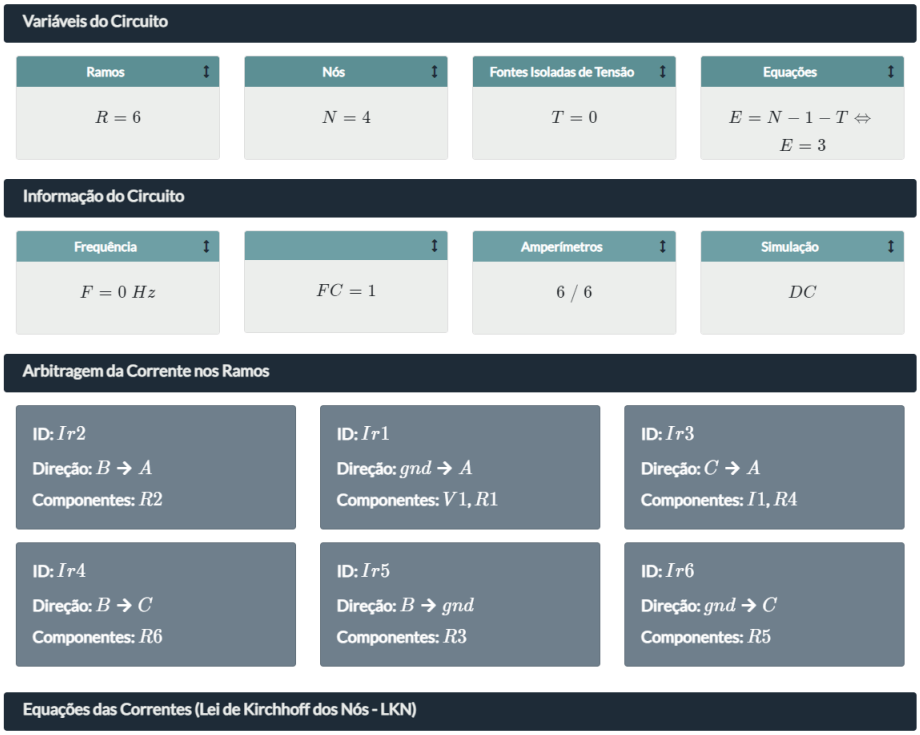


Figura 1.2: Circuito analisado pelo Método da Tensão nos Nós no *U=RIsolve Simulator*

1.2.2 Editor de Circuitos

O Editor de Circuitos[1] é uma ferramenta complementar disponível no *U=RIsolve Academy*, destinada ao desenho e à simulação de circuitos elétricos simples. A interface do editor permite que o utilizador crie esquemas elétricos personalizados, adicionando componentes como resistências, condensadores, bobinas, fontes de tensão e fontes de corrente. A ferramenta fornece um ambiente simples, especialmente para iniciantes.

A Figura 1.3 apresenta a planilha do Editor de Circuitos com um exemplo de um circuito desenhado. Esta planilha é estruturada num sistema de coordenadas cartesianas x,y, onde cada ponto representa uma posição específica no espaço de trabalho. Atualmente, a planilha possui dimensões de sessenta unidades no eixo x (comprimento) e trinta unidades no eixo y (largura).

[1] O editor está disponível através do seguinte *link* (temporário): http://cloud.microlumin.com:3020/editor.

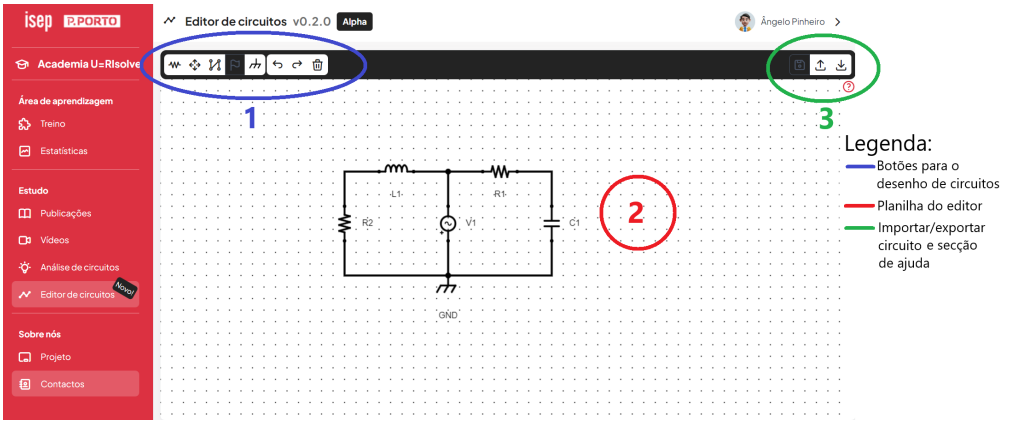


Figura 1.3: Página do Editor de Circuitos do *U=RIsolve Academy*

Esta aplicação *web* oferece várias opções de modo a facilitar a utilização do editor, que podem ser sucintamente explicadas na Figura 1.3 deste modo:

1. Rodeado a azul, são apresentados um conjunto de botões explicitamente para facultar o desenho de circuitos no editor. É possível colocar e movimentar componentes, interligar componentes com fios de ligação e eliminar componentes.

2. A planilha do Editor de Circuitos é o espaço disponível para desenhar os circuitos desejados. Na versão atual (Versão Alpha 0.2.0), as coordenadas cartesianas (x,y) são limitadas, onde não é possível desenhar circuitos demasiado extensivos. Está planeado tornar a planilha infinita futuramente, ou seja, sem limites cartesiando.

3. Rodeado a verde, são disponibilizados botões que permitem a importação de circuitos que utilizem o modelo de dados em uso pela aplicação, exportar/fazer o *download* dos circuitos desenhados e aceder a uma secção de ajuda (No ícone da ponto de interrogação vermelho) que orienta o utilizador no editor.

Deste modo, podemos concluir que a aplicação *web* proporciona uma interface intuitiva e funcional para a criação e edição de circuitos.

1.3 Descrição do Projeto e Metodologia

1.3.1 Objetivos

O objetivo principal da aplicação a conceber é a Geração Automática de Modelos de circuitos Elétricos (GAME), onde os utilizadores especificam parâmetros relacionados a circuitos elétricos, como o número de ramos, o número de resistências ou até mesmo o tipo de circuito (Corrente contínua ou Corrente alternada).

A aplicação deverá complementar o *U=RIsolve Academy*, pelo que utilizará os modelos de dados já desenvolvidos pelo Professor André Rocha. Esta integração tem o propósito de gerar circuitos-exemplo, de modo a complementar a versão atual estática, onde um utilizador teria de recorrer a outra aplicação, designada de *QUCS*, para desenhar e gerar o circuito elétrico pretendido.
Por fim, esta aplicação deverá ser capaz de gerar *data sets* para treino de algoritmos de *Artificial Intelligence* (AI), em particular, pelo uso de técnicas *Natural Language Processing* (NLP).
Em última análise, a aplicação a desenvolver será um importante instrumento para auxiliar na análise de circuitos elétricos, nomeadamente, permitindo a geração automática de modelos de circuitos elétricos lineares de corrente contínua e de corrente alternada, disponibilizando uma aplicação *web* interativa e que permita a seleção do circuito a gerar, a partir da seleção de parâmetros relevantes e referentes à topologia e composição do circuito, como o número de ramos, nós, tipo de circuito e o número e tipo de componentes (Resistências, condensadores, bobinas, fontes de tensão e fontes de corrente).
Com os objetivos devidamente definidos, foi possível dividir o trabalho de forma organizada.

1.3.2 Metodologia

A abordagem adotada neste projeto foi estruturada em várias etapas, desde a definição inicial de requisitos e planeamento, até ao desenvolvimento e testes.
Numa fase inicial, foi feito um estudo das ferramentas disponíveis para a realização do projeto, deste modo, foi analisado o modelo de dados [1] utilizado no *U=RIsolve Academy* [5], com o objetivo de garantir que o trabalho desenvolvido respeitasse o mesmo. Isto permitiu perceber como abordar o problema numa fase inicial e concretizar de modo eficaz os objetivos traçados.

De seguida, foram estudados algoritmos e métodos de análise de circuitos/redes, recorrendo a vários estudos sobre teoria de circuitos, como grafos [6, pp. 12–13] [7] e *spanning trees* [8] com o objetivo de construir o esquema de implementação da geração e desenho de circuitos.
Com as bases definidas, a abordagem foi criar uma aplicação *web* e interativa. Para este propósito, foi necessário escolher uma linguagem de programação para desenvolver o *frontend* e outra linguagem para o *backend*. Para o *frontend*, foi selecionado *HyperText Markup Language* (HTML) com *JavaScript* e *Cascading Style Sheets* (CSS), dado que é uma linguagem intuitiva e de resultados rápidos; para o *backend*, no desenvolvimento do algoritmo, foi utilizado *Python*, dado ser amplamente utilizada e considerando a facilidade de desenvolvimento de código.

[1] Modelo analítico que representa a lógica e a topologia de cada circuito (em formato JSON)

Assim, e após a escolha das linguagens, houve necessidade de aprofundar conhecimentos sobre as mesmas, dando especial atenção ao *Python*.
Seguidamente, foi elaborada uma aplicação simples para recolha de parâmetros base de circuitos elétricos. Com esta recolha, foi possível iniciar o desenvolvimento do algoritmo de geração automática de modelos de circuitos. Entendeu-se que era fundamental dividir em fases o desenvolvimento do algoritmo, para facilitar a progressão do trabalho, de modo a chegar a um resultado final, eficaz e sem imprecisões.
Foram assim testados diferentes métodos de geração de circuitos, o que permitiu ter uma visão mais abrangente de como esta é executada e, após uma fase extensiva de testes e melhoramentos, foi selecionado o método mais eficaz para alcançar o objetivo do projeto, que será explicado ao longo deste relatório.
Com a escolha do método feita, iniciou-se o desenvolvimento do algoritmo, juntamente com a aplicação. Esta foi sendo melhorada à medida que era testada, de modo a atender a todos os requisitos pretendidos na geração de circuitos.
Assim, e ao longo da criação do algoritmo, foi percecionado que a escolha das linguagens e do faseamento do projeto tinham sido uma boa opção.

1.3.3 Calendarização

A Figura 1.4 ilustra cronologicamente as fases descritas no subcapítulo anterior.

GAME	Outubro 2024			Novembro 2024				Dezembro 2024				Janeiro 2024			
Tarefa/Semana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Estudo da aplicação															
Algoritmos de geração															
Estudo das linguagens de programação															
Implementação e avaliação de diferentes algoritmos para GAME															
Implementação do algoritmo e da aplicação GAME															
Teste e melhoramentos na aplicação GAME															
Redação do relatório															

Figura 1.4: Calendarização do Projeto GAME

1.4 Organização do Relatório

Este relatório é composto por cinco capítulos, possibilitando ao leitor conhecer, de forma mais elucidativa, as várias etapas do desenvolvimento do projeto. No primeiro capítulo "Introdução", apresenta-se a contextualização do problema, as ferramentas utilizadas, a descrição do projeto e a abordagem adotada. Em seguida, no capítulo "Conteúdo Tecnológico", são discutidos os conceitos fundamentais necessários para a compreensão do trabalho, a estrutura do Editor de Circuitos e o modelo de dados utilizado, além de serem apresentados trabalhos relacionados que fundamentam o projeto.

O Capítulo 3, "Projeto *Software*", foca-se na descrição das opções de arquitetura e desenvolvimento do *software*, aplicando teoria de circuitos, explicando as etapas de implementação do algoritmo e o processo de parametrização e geração dos resultados.
O Capítulo 4, "Testes e Validação", apresenta as análises realizadas para verificar o funcionamento do sistema.
Por fim, o relatório conclui com uma análise geral dos resultados obtidos e propostas de Trabalho Futuro, destacando possíveis extensões e melhorias para o sistema.

Capítulo 2

Conteúdo tecnológico

Neste capítulo, são explicados alguns conceitos fundamentais sobre circuitos elétricos, juntamente com métodos de geração de circuitos.

2.1 Terminologia e Conceitos Fundamentais

Um circuito elétrico pode ser configurado de diferentes formas, e compreender esses conceitos é fundamental para a correta interpretação e análise de projetos elétricos. A familiarização com a terminologia utilizada permite não apenas entender o funcionamento dos circuitos, mas também comunicar-se de maneira eficaz em contextos académicos e profissionais.

Os circuitos elétricos são compostos por diversos elementos, como fontes de tensão e corrente, resistências, condensadores e bobinas, cada um desempenhando um papel específico na operação do sistema. Além disso, conceitos como nós, ramos e malhas são essenciais para a análise de circuitos.

A Figura 2.1 ilustra um circuito exemplo que será utilizado para esclarecer as diversas terminologias.

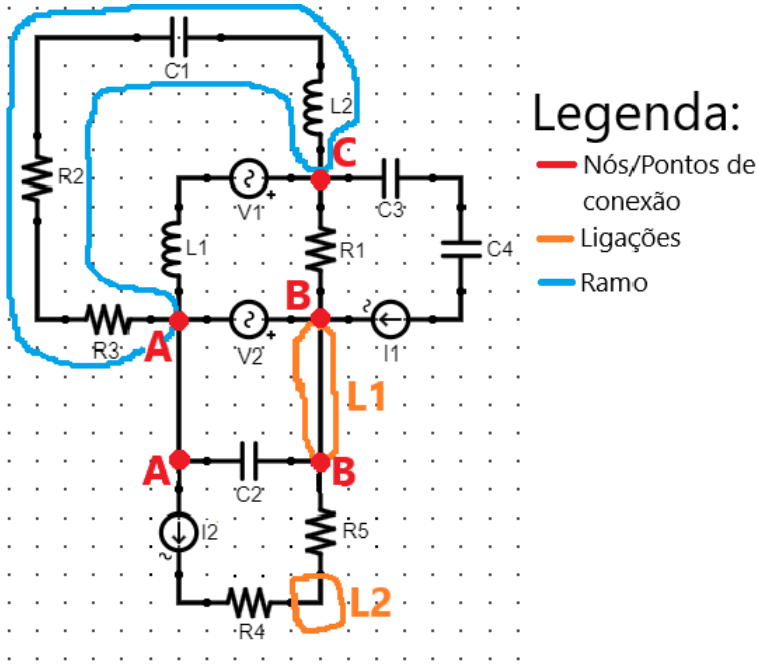


Figura 2.1: Circuito exemplo com legenda dos diferentes conceitos fundamentais

A seguir, são elencados os principais conceitos e definições essenciais para a compreensão deste relatório.

2.1.1 Sobre Circuitos Elétricos

- **Ligação** - Conexão entre os terminais de dois elementos. Esta interligação, no contexto deste projeto, é representado por uma linha. Rodeado a laranja, é apresentado na Figura 2.1 dois exemplos de uma ligação (L1 e L2).
- **Componente** - Qualquer elemento individual elétrico de um circuito com dois terminais que pode ser conectado a uma ligação. Na figura 2.1 é possível visualizar exemplos de componentente, como uma resistência (R1), um condensador (C1) e uma fonte de tensão *Direct Current* (DC) (V1).
- **Ramo** - Conjunto entre componentes e ligações que formam uma parte contínua do circuito [3], no entanto, existem circuitos com apenas um ramo, onde os terminais são, na verdade, o mesmo. A Figura 2.1 mostra um exemplo de um ramo com 4 componentes (R1).
- **Ponto de conexão** - Interligação entre três ou mais ligações, sejam elas ramos ou não. No circuito exemplo da Figura 2.1, são apresentados 5 pontos de conexão (2 identificados como "A", outros 2 como "B" e um como "C").

- **Nó** - Ponto de convergência de três ou mais ramos, composto por um ou mais pontos de conexão [3]. É possível visualizar 3 nós na Figura 2.1, identificados como "A", "B" e "C". Onde "A" e "B" são nós compostos por 2 pontos de conexão.

2.1.2 Sobre Teoria de Grafos

A análise de redes consiste em grafos e *spanning trees*/árvores geradoras que representam uma visão topológica de um circuito através de ligações e nós, demonstrando possíveis maneiras de desenhar o circuito.

- **Grafo** - Representação visual da interligação entre nós e ligações [1] [2] [9]. Formalmente, um grafo G, seguindo a terminologia deste relatório no Subcapítulo 2.1.1, pode ser definido como $G = (N, L)$, onde N é o conjunto de nós e L é o conjunto de ligações. A teoria dos grafos [1] [2] [9], que faz parte da matemática discreta, estuda as propriedades e aplicações dessas estruturas. Neste contexto, os grafos fornecem a base para modelar as relações e conexões necessárias para o desenvolvimento do algoritmo que foi utilizado neste projeto.
  É possível visualizar na Figura 2.2 um grafo com cinco nós e oito ligações.

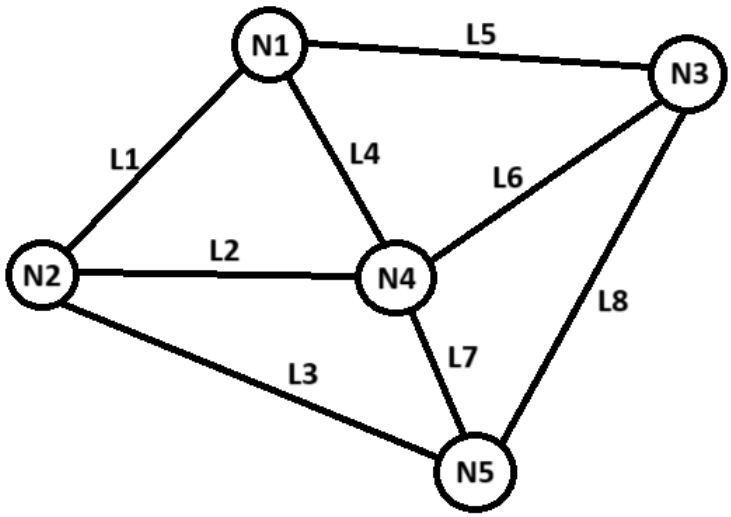


Figura 2.2: Exemplo de um grafo

- ***Spanning tree*** - Caminho possível de um grafo que mantém todos os vértices conectados sem haver caminhos fechados, sendo composto pelo número mínimo de ligações necessárias para isso [1] [2] [9].

Se for considerado o número de nós como "N", uma *spanning tree* fica feita com N-1 ligações.
Na Figura 2.3 é possível visualizar uma possibilidade de criação de uma *spanning tree* utilizando o grafo da Figura 2.2.

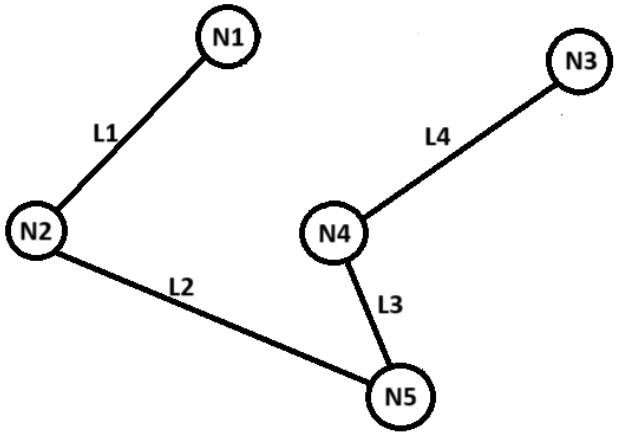


Figura 2.3: Escolha de uma *spanning tree*, para o grafo da Figura 2.2

2.2 Editor de Circuitos e Modelo de Dados

O *Editor de Circuitos* e o modelo de dados utilizados desempenharam um papel central no desenvolvimento deste projeto, servindo como a base estrutural sobre a qual o trabalho foi conduzido. O editor permitiu a criação e manipulação gráfica de circuitos elétricos, enquanto o modelo de dados, no formato *JavaScript Object Notation* (JSON), forneceu uma representação organizada e acessível desses circuitos. Nos pontos seguintes detalha-se a forma como essas ferramentas foram utilizadas, destacando-se os seus principais recursos, funcionamento e a forma como orientaram o desenvolvimento das funcionalidades do projeto.

2.2.1 Editor de Circuitos

O Editor de Circuitos[1] presente na aplicação *U=RIsolve*, descrito no Subcapítulo 1.2.2, foi a ferramenta desenvolvida pelo Professor André Rocha e implementada no servidor pelo colega Ângelo Pinheiro. Esta é utilizada para desenhar circuitos elétricos simples, mas, posteriormente, tem como objetivo analisar devidamente os circuitos desenhados. Para este projeto, esta ferramenta foi usufruída simultaneamente para compreender o modelo de dados disponível (também desenvolvido pelo Professor

[1]O Editor de Circuitos pode ser acedido no seguinte endereço (temporário): http://cloud.microlumin.com:3020/editor.

André Rocha) e para testar os circuitos a serem gerados. Deste modo, a familiarização com esta ferramenta, que atualmente está na versão *Alpha* 0.2.0, foi crucial para a realização do projeto.
Esse sistema de coordenadas permite ao utilizador posicionar elementos com precisão, facilitando a organização e o alinhamento dos componentes no editor.
Este editor oferece os elementos necessários para desenhar circuitos elétricos simples. Os elementos podem ser descritos como:

- **"Gaveta de Componentes"** - Acessível através do botão com a figura de uma resistência, onde os componentes que podem ser selecionados são:

  1. **Fontes:**
     - Fonte de Tensão DC
     - Fonte de Corrente DC
     - Fonte de Tensão *Alterning Current* (AC)
     - Fonte de Corrente AC
  2. **Componentes passivos:**
     - Resistências
     - Condensadores
     - Bobinas
  3. **Instrumentos de medição:**
     - Voltímetro
     - Amperímetro

  A Figura 2.4 ilustra os ícones de todos os componentes disponíveis na aplicação, organizados de forma lógica e categorizada:

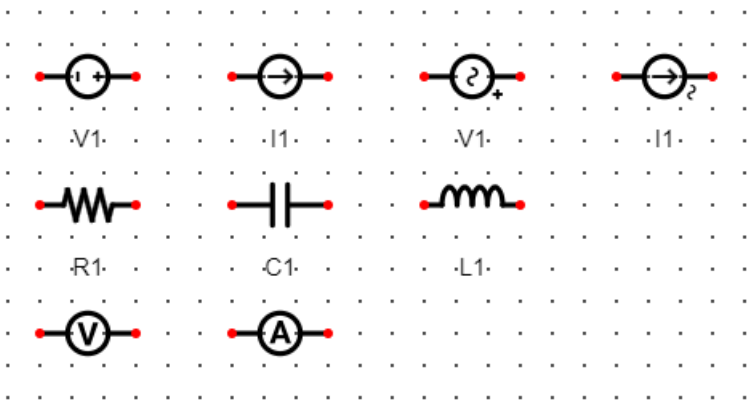


Figura 2.4: Ícones dos componentes presentes no Editor de Circuitos

  – **Fontes:** Na primeira linha, da esquerda para a direita estão os ícones que representam as fontes de energia, incluindo a Fonte de Tensão DC, a Fonte de Corrente DC, a Fonte de Tensão AC e a Fonte de Corrente AC.

  - **Componentes passivos:** Na segunda linha, da esquerda para a direita, estão apresentados os Componentes passivos na seguinte ordem: Resistência, Condensador e Bobina.
  - **Instrumentos de medição:** Com os ícones do Voltímetro e do Amperímetro.

- **"Mover planilha"** - Botão que permite ao utilizador selecionar elementos ou um conjunto de elementos específicos do circuito desenhado, sendo possível depois movimentar o selecionado. Este botão está pré-definido como *default*, ou seja, quando não é selecionado nenhum botão em específico este é o que está selecionado.

- **"Conectar"** - Para fazer as ligações entre componentes e, eventualmente, formar ramos e nós.

- **"Massa"** - Colocar a massa do circuito, que é um elemento cujo objetivo é ter um ponto no circuito onde o potencial elétrico é nulo ou quase nulo por razões de segurança ou para a conveniência do estudo de sistemas elétricos [10].

- **"Desfazer"** - Representado por um ícone de seta curvada para a esquerda e permite reverter a última ação realizada. É essencial pela eficácia que traz ao editor, pois permite a rápida correção de erros(pode ser utilizado o atalho "$CTRL + Z$" para este efeito).

- **"Refazer"** - Representado por um ícone de seta curvada para a direita, este botão é essencialmente o oposto do botão "Desfazer", onde restaura ações que foram desfeitas anteriormente, possibilitando recuperar mudanças desfeitas por engano(pode ser utilizado o atalho "$CTRL + Y$" para este efeito).

- **"Eliminar"** - Ao utilizar o botão "Mover planilha"para selecionar um elemento, é possível utilizar este botão para remover esse elemento do desenho.

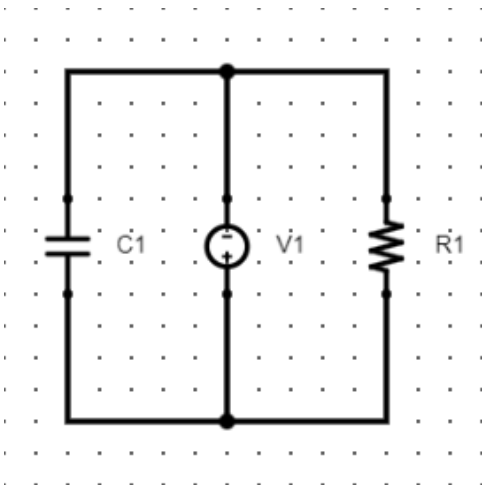
- **"Importar"** - Permite importar ficheiros compatíveis com os modelos de dados disponíveis no editor. Atualmente, a única forma de importação suportada é através de ficheiros no formato JSON, apesar de existir visualmente(não é possível selecionar) a opção de importar com um "Esquemático". No âmbito do projeto esta foi a opção utilizada para testar os circuitos gerados.

- **"Exportar"** - Para exportar o circuito desenhado com o modelo de dados pretendido. Igual à opção de "Exportar", só é possível em ficheiro JSON, apesar de existir visualmente a opção "*Netlist*" e "*Image*". Esta funcionalidade foi crucial para compreender a estrutura do modelo de dados em uso.

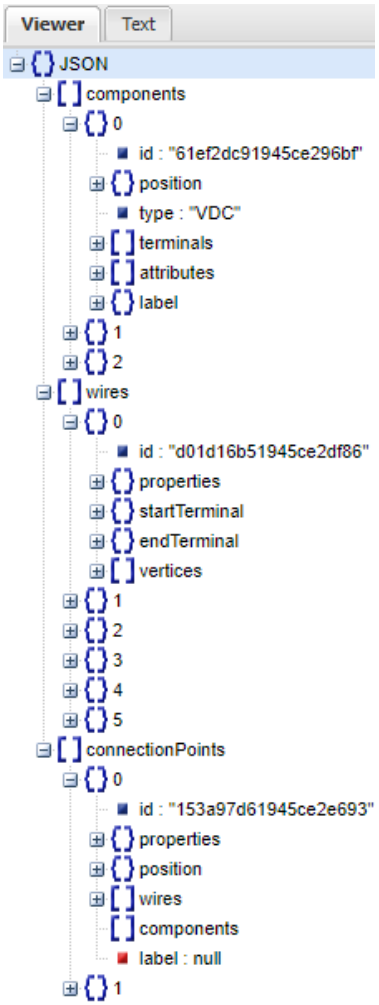
- **"Ajuda"** - Todos os botões explicados até agora são explicados em pormenor no botão representado com um ponto de interrogação "?", de forma a ajudar o aluno a utilizar o Editor de Circuitos sem dificuldade.
- **Pormenores dos elementos** - Ao clicar com o botão direito do rato sobre um elemento, seja um ramo ou um componente do circuito, uma janela de configuração é aberta. Esta janela permite ao utilizador especificar os parâmetros do elemento selecionado.
    1. **Ramo** - Ao clicar com o botão direito do rato num ramo do circuito, é possível modificar a cor desse ramo, colocar esse ramo a tracejado e/ou eliminar o ramo.
    2. **Componente** - As opções disponíveis num componente permitem a especificação do valor em número e em unidade, modificar a *label* do componente, rodar ou eliminar o mesmo.

2.2.2 Modelo de Dados

O modelo de dados adotado para este projeto está no formato JSON, que é amplamente utilizado devido à sua simplicidade e capacidade de estruturar informações de forma hierárquica e legível. Esse formato é especialmente útil para armazenar e transmitir dados entre diferentes sistemas, garantindo compatibilidade e facilidade de manipulação. No contexto deste projeto, o uso do JSON permite que os circuitos sejam representados de modo organizado, facilitando a sua leitura.
Este modelo foi projetado pelo Professor André Rocha e foi adaptado para representar de forma clara e eficiente os componentes e conexões de um circuito elétrico no editor. A estrutura do JSON armazena detalhes como o tipo de componente, as suas propriedades (como resistência, tensão ou corrente) e a forma como estão interligados, garantindo a correta importação para o editor.
Na Figura 2.5a é apresentado um exemplo de um circuito elétrico desenhado no Editor de Circuitos.
Na Figura 2.5b, após clicar no botão "Exportar"em formato JSON, é mostrado o conteúdo do ficheiro exportado do respetivo circuito desenhado.



(a) Exemplo de um circuito elétrico desenhado no Editor de Circuitos.



(b) Conteúdo do ficheiro JSON exportado do respetivo circuito desenhado.

Figura 2.5: Exemplo de um circuito elétrico desenhado no Editor de Circuitos (a) e o respetivo conteúdo do ficheiro JSON exportado (b)

2.3 Trabalhos Relacionados

Este projeto divide-se em duas vertentes principais, a primeira centra-se na geração automática de circuitos elétricos a partir de parâmetros definidos pelo utilizador, permitindo a criação de circuitos com base em regras e características previamente especificadas. A segunda vertente está relacionada aos modelos de dados de circuitos elétricos no disponíveis. Este modelo de dados fornece uma representação clara e acessível dos circuitos, facilitando a manipulação dos dados em diferentes aplicações.

Serão pesquisadas aplicações e ferramentas que lidam com a geração automatizada de circuitos elétricos e também serão vistos os modelos de dados que as aplicações utilizam. Esta análise permitirá identificar as soluções existentes, seus pontos fortes e limitações, bem como posicionar o presente trabalho no contexto dessas abordagens.

2.3.1 autoCircuits

O *autoCircuits* [11] é uma aplicação *web* projetado para a geração automática de problemas de teoria de circuitos elétricos. A ferramenta utiliza um motor de cálculo em tempo real que, com parametrizações definidas pelo utilizador, gera um grafo com nós e ligações apropriadas. O grafo é então preenchido com elementos de circuito que recebem valores numéricos ou simbólicos. A solução do circuito é calculada usando *Modified Nodal Analysis* [12] para análises DC, AC, simbólica (*Laplace*) e até transitória.
A Figura 2.6 apresenta a página *web* do *autoCircuits* onde se pode visualizar as diferentes opções de parametrização disponíveis.

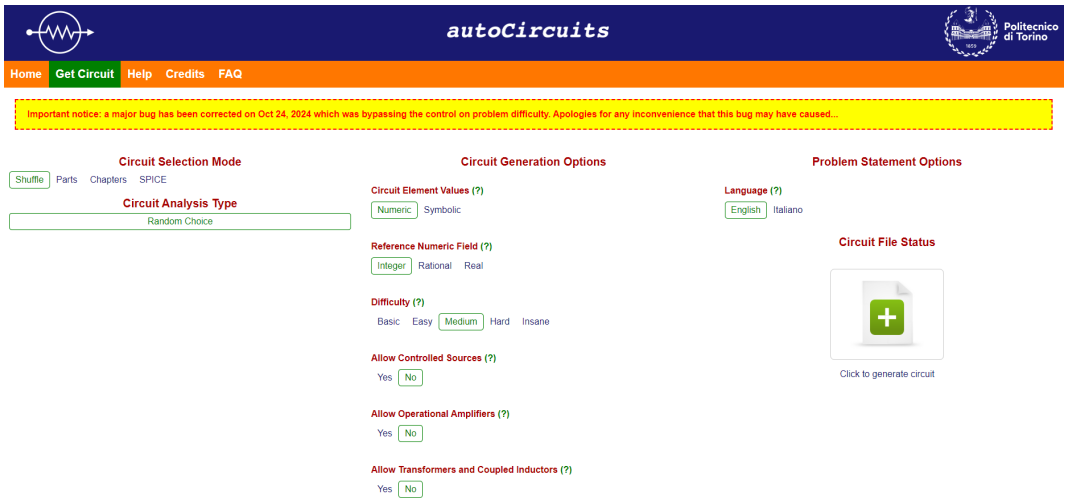


Figura 2.6: Página *web* do *autoCircuits*

A aplicação gera um esquema do circuito e um arquivo *Portable Document Format* (PDF) com o enunciado, esquema e solução, disponível para *download* por tempo limitado.
A Figura 2.7 apresenta um exemplo de *output* da aplicação, onde é possível visualizar o problema com o esquema do circuito, o enunciado e a solução do problema.

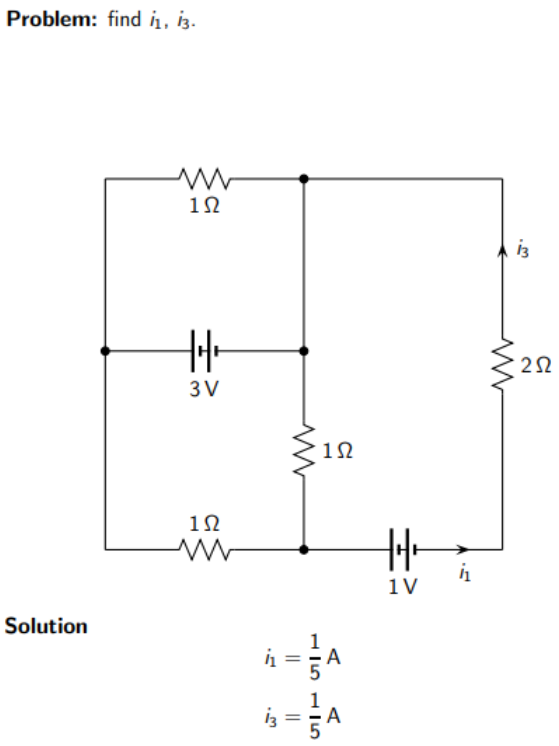


Figura 2.7: Exemplo de *Output* da aplicação autoCircuits

2.3.2 QUCS

O *Quite Universal Circuit Simulator* (QUCS) [13] é uma aplicação dedicada à simulação de circuitos elétricos. Oferece uma interface gráfica intuitiva para criar esquemáticos de circuitos e realizar simulações de diversos tipos, que incluem análise DC, AC e transiente. O QUCS suporta uma ampla gama de componentes de circuitos, incluindo resistências, diodos, amplificadores operacionais e fontes de sinal. Por este motivo, o QUCS é maioritariamente utilizado para circuitos mais complexos.

O modelo de dados usado pelo QUCS para representar esquemáticos de circuitos é o *schematic* que é um ficheiro do tipo ".sch"que representa as ligações, os componentes e as interligações entre eles, gerado através deste simulador de circuitos [4]. A Figura 2.8 apresenta um *schematic* de um circuito-exemplo desenhado no QUCS.

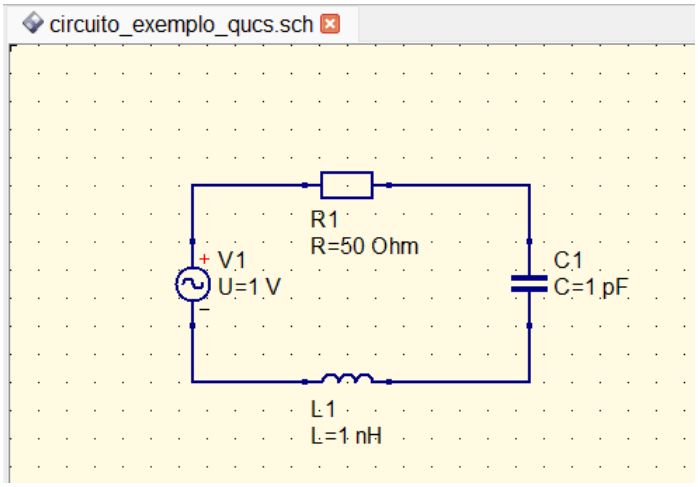


Figura 2.8: Ficheiro tipo .sch "*Schematic*" de um circuito elétrico

Juntamente com o *schematic*, o QUCS inclui outra representação de um circuito elétrico através de texto. Esta representação chama-se *netlist* e o modelo de dados presente neste tipo de ficheiros é utilizado no *U=RIsolve Simulator* para a resolução de circuitos elétricos.

É possível visualizar na Figura 2.9 a *netlist* referente ao circuito-exemplo da Figura 2.8, onde se nota a declaração dos componentes, a conexão com ligações entre os componentes e a configuração da simulação (Corrente alternada).

circuito_exemplo_qucs.sch | netlist.txt

```
R:R1 _net0 _net1 R="50 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
L:L1 _net2 _net1 L="1 nH" I=""
C:C1 _net3 _net2 C="1 pF" V=""
Vac:V1  net0  net3 U="1 V" f="1 GHz" Phase="0" Theta="0"
```

Figura 2.9: *Netlist* do circuito elétrico da Figura 2.8

Capítulo 3

Projeto de Software

3.1 Enquadramento

O desenvolvimento do projeto seguiu um plano que começou por implementar o algoritmo de geração de ramos, onde são calculadas as posições geométricas(x, y) dos nós e dos ramos de forma a criar um *layout* do circuito. Depois de ter o *layout* do circuito, resta apenas alocar os componentes nas posições dos ramos já definidos.
Finalmente, criou-se o ficheiro seguindo o modelo de dados descrito no Subcapítulo 2.2.2.
Na Figura 3.1 é possível observar a arquitetura e planeamento do projeto realizado, onde se visualizam os passos efetuados para gerar o modelo do circuito.

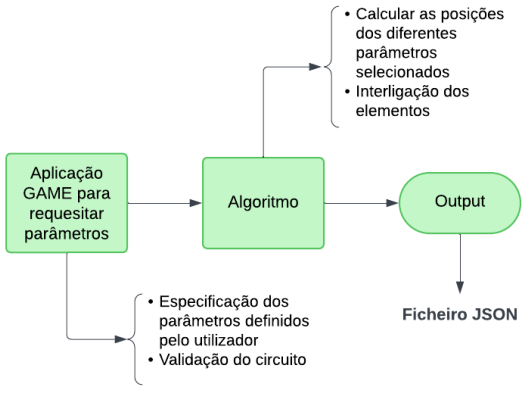


Figura 3.1: Diagrama de blocos da arquitetura de *software*

O diagrama mostrado garante o funcionamento dos algoritmos utilizados e pode ser explicado como:

- **"Aplicação GAME para requisitar parâmetros"** - Aplicação simples para permitir ao utilizador especificar o circuito que pretende gerar, denominada Geração Automática de Modelos de circuitos Elétricos (GAME).
- **"Algoritmo"** - Algoritmo criado, que foi utilizado para gerar o modelo de circuito elétrico.
- **"*Output*"** - É feito o *download* do ficheiro JSON gerado.

3.2 Arquitetura de *Software*

A arquitetura de *software* é representada na Figura 3.2, que ilustra as interações entre o utilizador, a aplicação e os diferentes componentes tecnológicos utilizados para o seu funcionamento.

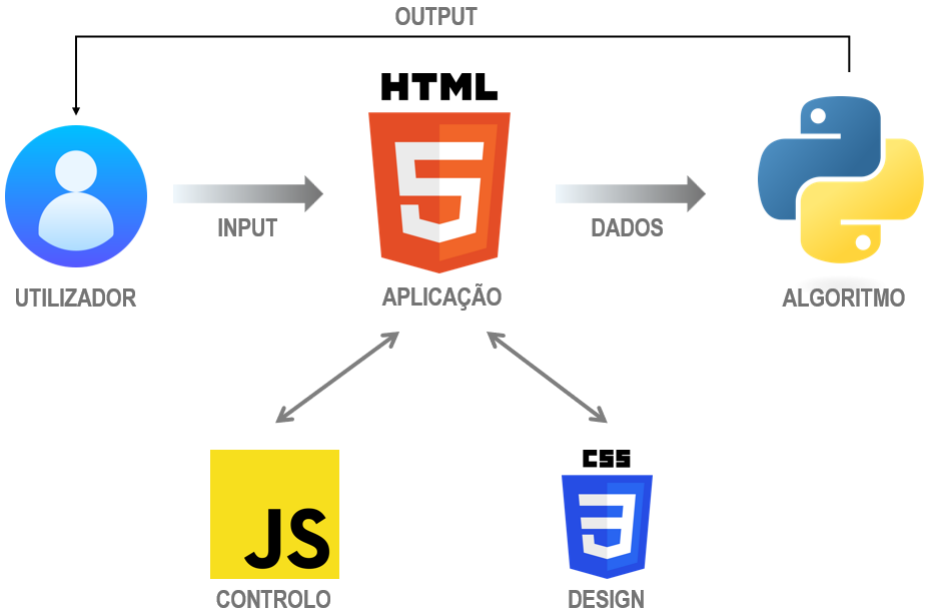


Figura 3.2: Arquitetura do software.

A interação com o sistema inicia-se pelo utilizador, que fornece *input* através da interface da aplicação desenvolvida em HTML. Esta interface constitui a camada de apresentação, permitindo ao utilizador introduzir dados ou interagir com os elementos visuais.
O HTML, em conjunto com o CSS, é responsável pelo *design* da aplicação, definindo a estrutura e a aparência visual. Por outro lado, o *JavaScript* desempenha um papel essencial no controlo da interação, processando eventos do utilizador e gerindo dinamicamente o comportamento da aplicação.

Os dados introduzidos pelo utilizador são enviados para o algoritmo, desenvolvido em *Python*, que processa a informação e executa cálculos e lógica avançada. O resultado deste processamento é então devolvido à aplicação e apresentado ao utilizador como *output*.
Esta arquitetura modular e interativa permite uma comunicação eficiente entre o utilizador e a aplicação, proporcionando uma experiência fluida e intuitiva.

3.3 Limitações e Interdependências entre Parâmetros dos Circuitos

Existem certas limitações que definem a estrutura correta de circuitos elétricos. Por exemplo, não é possível existir um circuito com dois ramos ou um circuito com quatro ramos e quatro nós [14] [15]. Adicionalmente, no caso do número de nós ser zero, o único valor do número de ramos é, obrigatoriamente, um. Neste subcapítulo, foi elaborado um estudo empírico sobre essas restrições, explanado de seguida. Uma das limitações mais importantes está na definição do número mínimo de ramos para um certo número de nós, onde foram feitas experiências recorrendo ao desenho de circuitos exaustivamente. Na Tabela 3.1 visualiza-se a relação descrita.

Tabela 3.1: Número mínimo de ramos para um circuito com um dado número de nós.

Nº Nós	**Nº Ramos Mínimos**
0	1
X	X
2	3
3	5
4	6
5	8
6	9
7	11
8	12
9	14

Para chegar à relação apresentada na Tabela 3.1, foi determinada a seguinte equação, que descreve o número mínimo de ramos ($R_{\text{mín}}$) necessários:

$$R_{\text{mín}} = \lfloor 1.5 \times N_{\text{nós}} + 0.5 \rfloor \tag{3.1}$$

A equação 3.1 baseia-se no fato de que, para que o circuito seja funcional, os ramos devem conectar adequadamente os nós de forma a garantir a continuidade elétrica, evitando configurações inválidas.

Por exemplo, com dois nós, o número mínimo de ramos necessário é três, e com quatro nós, são necessários pelo menos seis ramos. A equação utiliza a função piso (*floor function*), que arredonda para baixo os valores calculados [16], garantindo a adequação ao número de ramos inteiros.
Além dessas restrições relacionadas aos ramos e nós, outras limitações relacionadas com a teoria dos circuitos devem ser consideradas ao definir os mesmos, tais como:

- **Garantia na presença de pelo menos uma fonte de energia:** Um circuito elétrico para ser válido deve incluir pelo menos uma fonte de tensão ou de corrente. A ausência de uma fonte de energia impossibilita o funcionamento do circuito, pois não há diferença de potencial ou fluxo de corrente necessários para o comportamento elétrico.

- **Restrição de componentes em circuitos DC:** Para o objetivo da aplicação *U=RIsolve*, não é permitido o uso de componentes que dependem de variações de frequência para operar, ou seja, as bobinas e os condensadores não foram considerados em circuitos DC. Esses componentes, quando presentes em circuitos DC, são modelados como curtos-circuitos ou circuitos abertos, comprometendo a funcionalidade.

- **Número total de componentes não pode ser menor que o número de ramos:** Esta restrição existe porque cada ramo em um circuito deve conter, no mínimo, um componente. Caso não houvesse componentes suficientes para o número de ramos, de acordo com a explicação de um ramo no Subcapítulo 2.1.1, seria impossível formar os ramos todos.

Essas restrições foram implementadas principalmente na aplicação(*frontend*), no entanto, também foram implementadas no *backend* em *Python* por uma questão de redundância e de modo a garantir que os circuitos sejam configurados corretamente.

3.4 Aplicação/Formulário

Para recolher as especificações escolhidas pelo utilizador, foi desenvolvida uma aplicação[1] que facilita a seleção dos parâmetros necessários para a geração de circuitos. Este subcapítulo apresenta uma explicação detalhada sobre a aplicação, incluindo as opções disponíveis para parametrização e as limitações implementadas e explicadas no Subcapítulo 3.3 para garantir a criação de circuitos válidos.

[1] A Aplicação GAME pode ser acedida no seguinte endereço (temporário): http://cloud.microlumin.com:3022.

3.4.1 Opções de Parametrização

O formulário para seleção de parâmetros foi dividido em várias secções para auxiliar o utilizador a especificar o circuito que pretende.

- **Opção do Circuito:** Permite escolher entre "Personalizado", que possibilita a especificação de cada componente, ou "Aleatório", caso não seja necessário especificar os componentes individualmente. A Figura 3.3 ilustra as opções descritas e como é feita a sua seleção. Também é mostrado a opção "Modelo IA"que irá ser desenvolvida futuramente, logo a sua seleção é impossível.

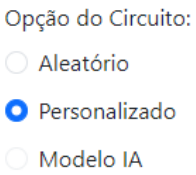


Figura 3.3: Seleção da opção do circuito na aplicação de geração automática de circuitos

- **Tipo de Circuito:** Define se o circuito será em DC (corrente contínua) ou AC (corrente alternada). A seleção do tipo de circuito influencia diretamente os componentes disponíveis.
  É apresentado na Figura 3.4 como é feita a seleção do tipo de circuito disponível (AC ou DC).

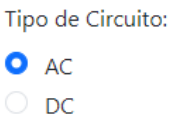


Figura 3.4: Seleção do tipo de circuito na aplicação de geração automática de circuitos

- **Selecione os Componentes:** Se for selecionado "Personalizado" na Opção do Circuito, para cada componente, o utilizador pode optar por definir valores exatos (Fixo) ou permitir que a aplicação escolha aleatoriamente (Aleatório). Também é possível definir um limite inferior e superior dos valores de cada componente. Aqui o utilizador pode especificar a faixa de valores que pretende utilizando o controlo deslizante disponível. Estes valores podem variar de 1 a 1000 para resistências, bobinas e condensadores e, para as fontes de tensão e corrente, variam entre 1 e 30.
  Nesta secção, o utilizador, caso escolha Fixo num dos componentes, pode especificar:

- Quantidade de resistências.
- Quantidade de condensadores[1].
- Quantidade de bobinas[1].
- Quantidade de fontes de tensão[2].
- Quantidade de fontes de corrente[2].

A Figura 3.5 apresenta os diferentes componentes parametrizáveis, onde se verificam também os limites superior e inferior para a especificação de valores de cada componente.

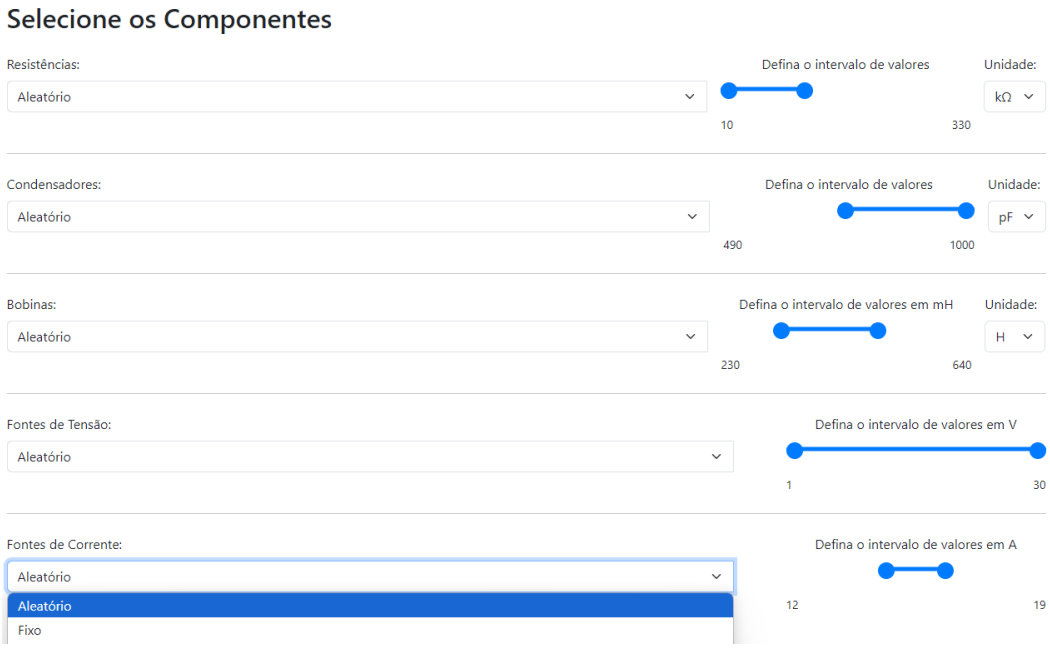


Figura 3.5: Parametrização dos componentes e dos seus valores na aplicação de geração automática de circuitos

À direita do controlo deslizante, existe uma caixa de seleção onde é possível especificar o submúltiplo da unidade base do componente em escolha. Os submúltiplos disponíveis para cada componente foram definidos da seguinte maneira:

- Resistências: mΩ, Ω, kΩ, MΩ
- Condensadores: pF, nF, uF, mF
- Bobinas: nH, uH, mH, H
- Fontes de tensão: V
- Fontes de corrente: A

[1]Opção disponível apenas quando o Tipo de Circuito selecionado é AC.
[2]Este componente é diretamente afetado pelo Tipo de Circuito (AC ou DC)

Atualmente, as fontes de tensão e corrente são representadas apenas pelas suas unidades no Sistema Internacional (V para volts e A para amperes). Essa escolha foi feita porque, no contexto atual da aplicação, não se considerou necessário incluir outras unidades ou variações. No entanto, essa decisão poderá ser revisitada no futuro, caso surjam novas necessidades ou requisitos que justifiquem a inclusão de diferentes escalas, como milivolts (mV) ou microamperes (µA), dependendo da aplicabilidade da ferramenta.
A Figura 3.6 mostra como é possível selecionar diferentes submúltiplos para as resistências.

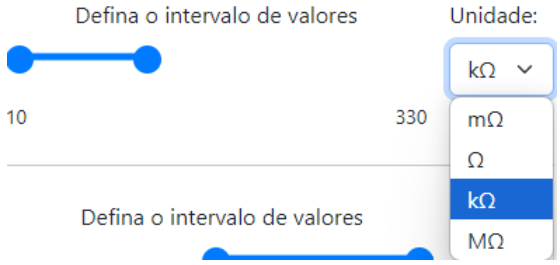


Figura 3.6: Submúltiplos disponíveis para resistências

- **Selecione o número de nós e de ramos:** Permite configurar a quantidade de nós e ramos do circuito. O utilizador pode definir valores fixos ou aleatórios para ambas as opções, como é possível verificar na Figura 3.7.

Figura 3.7: Parametrização dos nós e dos ramos na aplicação

No final do formulário, existe um botão denominado "Gerar Circuito"(visível no final da Figura 3.7), que executa o algoritmo de geração do circuito com base nos parâmetros selecionados. Após a execução, é automaticamente feito o *download* do ficheiro com o circuito gerado.
Adicionalmente, a aplicação também dispõe de duas hiperligações, uma para abrir automaticamente um separador do *browser* com o Editor de Circuitos e outra para preenchimento de um formulário para reporte de erros na aplicação.

3.4.2 Atribuição das Limitações

As limitações descritas no Subcapítulo 3.3 foram implementadas tanto no design da aplicação em *HTML* e, maioritariamente, *JavaScript*, quanto no *backend* do projeto, de forma a garantir a geração de circuitos válidos. A seguir, cada limitação é explicada no contexto da sua aplicação prática:

- **Número mínimo de ramos:** A equação 3.1, que determina o número mínimo de ramos necessários para um dado número de nós, foi incorporada diretamente na lógica do formulário. Ao definir um número de nós, o sistema calcula automaticamente o número mínimo de ramos permitido, atribuindo o valor de ramos mínimo e impossibilitando a seleção de valores inválidos/menores.
- **Garantia de pelo menos uma fonte de energia:** O sistema começa por atribuir automaticamente uma fonte de tensão e nenhuma fonte de corrente, onde, caso o utilizador tente remover a fonte de tensão, primeiro tem de ser adicionada uma fonte de corrente e vice-versa.
- **Restrições de componentes em circuitos DC:** Quando a opção DC é selecionada no formulário, as opções de configuração de bobinas e condensadores são desativadas. Essa funcionalidade assegura que componentes dependentes de variação de frequência não sejam utilizados em circuitos DC. Caso o tipo de circuito seja alterado para AC, essas opções tornam-se disponíveis novamente.
- **Número total de componentes não pode ser menor que o número de ramos:** Na submissão do formulário, o sistema verifica se o número total de componentes especificado é suficiente para preencher todos os ramos do circuito. Caso o número total de componentes for menor que o número de ramos, o formulário impedirá o envio dos dados e exibirá uma mensagem indicando a necessidade de ajuste.

As limitações implementadas no *frontend* e *backend* da aplicação garantem que o utilizador possa apenas configurar circuitos válidos, reduzindo erros e inconsistências na geração dos mesmos. Cada restrição foi cuidadosamente projetada para respeitar as regras fundamentais da teoria de circuitos, enquanto proporciona uma experiência de uso intuitiva.
Assim, a aplicação assegura que os parâmetros definidos pelo utilizador estejam em conformidade com as exigências de projeto, resultando em circuitos funcionalmente corretos e otimizados para os objetivos pretendidos.

3.5 Implementação do Algoritmo

Neste subcapítulo, é explicado o algoritmo de geração de circuitos concebido e implementado, onde é possível visualizar na Figura 3.8 um diagrama de blocos capaz de explicar o processo.

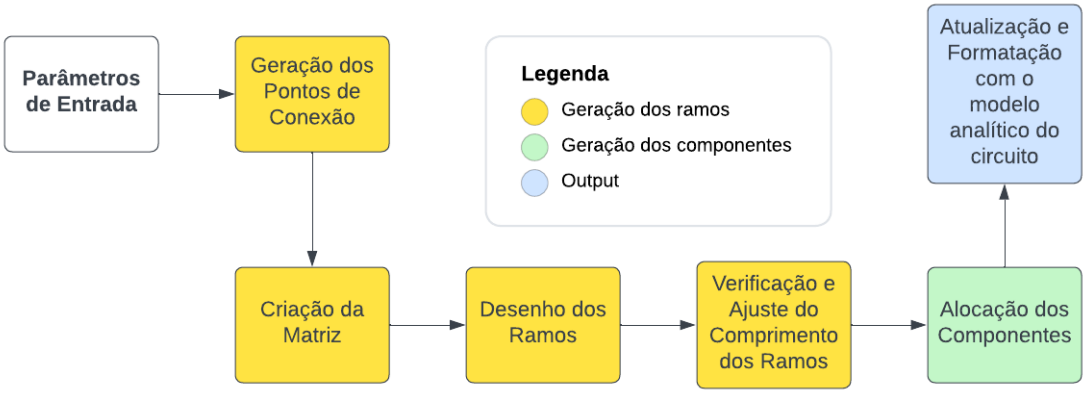


Figura 3.8: Diagrama de blocos referente ao algoritmo de geração de circuitos

Como primeiro passo, o algoritmo de geração de circuitos começa por recolher os diferentes parâmetros enviados através da aplicação. Estes parâmetros definem o funcionamento do algoritmo, pois é através deles que sabemos a "Opção do Circuito", o "Tipo de Circuito", o número de cada tipo de componente, o número de ramos e o número de nós.
Ao longo da explicação do algoritmo, vão sendo apresentadas figuras de um circuito-exemplo para complementar a compreensão do algoritmo. O circuito-exemplo em questão tem de ter três nós e cinco ramos.

3.5.1 Geração dos Pontos de Conexão

Na geração dos pontos de conexão, foi escolhida uma abordagem dependente da planilha do editor de circuitos. O primeiro ponto a ser criado é posicionado no centro da planilha e este serve como base para a criação dos restantes pontos de conexão. Para os pontos de conexão subsequentes, a posição é calculada a partir de um ponto aleatório já existente. Este processo utiliza direções predefinidas (*esquerda*, *direita*, *cima* ou *baixo*) e deslocamentos fixos, de forma a garantir a distribuição geométrica dos pontos de conexão na planilha. Antes de adicionar um novo ponto, é realizada uma verificação para evitar sobreposições, assegurando a validade da estrutura gerada. Os pontos de conexão aqui criados têm o objetivo de, no futuro, se tornarem nós, respeitando a definição, onde vão ser intersetados três ou mais ramos neste ponto de conexão.

A Figura 3.9 apresenta as posições dos pontos de conexão gerados para o circuito-exemplo. Esta abordagem garante que os nós sejam criados de forma dinâmica.

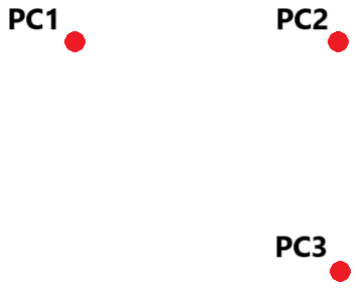


Figura 3.9: Geração dos nós para o circuito-exemplo

3.5.2 Criação da Matriz

Para representar as ligações do circuito, utiliza-se uma matriz de adjacência bidimensional, onde cada elemento da matriz indica a existência (ou ausência) de uma ligação direta entre dois pontos de conexão do circuito. O processo começa com a identificação de todos os pontos de conexão do circuito. Cada ponto recebe um identificador único, que é utilizado para estruturar as linhas e colunas da matriz. A dimensão da matriz é definida pelo número total de pontos de conexão do circuito, sendo P×P, onde P é o número de pontos.
Inicialmente, a matriz é preenchida com zeros em todas as suas posições, indicando que nenhum ponto de conexão tem uma ligação. Em seguida, as ligações definidas no circuito são utilizadas para atualizar a matriz. Quando dois nós estão conectados, o valor da posição correspondente A[i][j] na matriz é incrementado. Como a matriz de adjacência é bidimensional e simétrica, também se incrementa A[j][i].
A Tabela 3.2 demonstra a matriz de adjacência para o circuito-exemplo.

Tabela 3.2: Criação da matriz de adjacência para o circuito-exemplo

PC	**PC1**	**PC2**	**PC3**
PC1	0	0	0
PC2	0	0	0
PC3	0	0	0

Esta matriz irá ser atualizada à medida que são feitas ligações entre pontos de conexão, permitindo a fácil visualização de ligações no circuito. Para saber quantas ligações tem cada ponto de conexão, basta somar a linha ou a coluna referente ao ponto escolhido.

3.5.3 Desenho dos Ramos

O processo de geração de ramos num circuito segue um ciclo que consiste em três etapas principais. Estas etapas garantem que os ramos sejam criados de forma ordenada e consistente, conectando adequadamente os nós do circuito. As etapas podem ser descritas da seguinte forma:

- **Seleção dos Nós:** São escolhidos dois nós no circuito que deverão ser conectados com um ramo. Esses nós podem ser definidos aleatoriamente ou seguindo uma lógica previamente estabelecida que acompanha uma série de condições ordenadas:

  1. Foi determinado quantos pontos de conexão é que são verdadeiramente nós, ou seja, que têm três ou mais ligações, recorrendo à matriz mencionada no Subcapítulo 3.5.2
  2. O primeiro ponto de conexão a ser escolhido é aleatório e não pode ser um nó, exceto quando todos os pontos de conexão já tiverem três ou mais ligações, onde nesse caso, o ponto de conexão tem de ser um nó que esteja num dos extremos do circuito, de forma a que o desenho do ramo não sobreponha outros.
  3. O segundo ponto de conexão a ser escolhido segue a mesma regra que o primeiro, com a adição de uma regra particular. Que este ponto de conexão não seja igual ao escolhido no primeiro.

- **Determinação do Sentido:** Após a escolha dos pontos de conexão, o sentido do ramo é determinado. O sentido indica a posição relativa do segundo ponto de conexão em relação ao primeiro e pode ser um sentido simples, como *esquerda*, *direita*, *cima* ou *baixo* ou um sentido combinado, como por exemplo, simultaneamente *direita* e *cima*.
  Esse passo é fundamental para estabelecer a orientação geométrica do ramo a desenhar.

- **Desenho do Ramo:** Finalmente, pegando nos pontos de conexão escolhidos e no sentido determinado, o processo para desenhar o ramo começa. O desenho do ramo é diretamente orientado pelo sentido previamente estabelecido, ou seja, é desenhado do primeiro ponto de conexão até ao segundo ponto de conexão através do sentido determinado.
  Essa abordagem não apenas garante que o ramo seja visualmente claro e respeite a orientação geométrica definida pelo algoritmo, mas também implementa verificações adicionais para evitar sobreposição de ramos já existentes no circuito.
  Antes de desenhar o ramo, o algoritmo verifica se há espaço disponível para o

novo traço sem interferir com os elementos existentes. Caso uma sobreposição seja detectada, uma lógica de adaptação é ativada:

- Se o sentido determinado for composto, como direita e cima, o algoritmo tenta desenhar o ramo primeiramente na direção especificada (direita e depois cima). Caso isso não seja possível devido a obstáculos ou restrições geométricas, o algoritmo inverte a ordem, tentando primeiro na direção cima e depois direita. Essa flexibilidade aumenta a probabilidade de encontrar um caminho válido para o ramo.
- Se, mesmo após essas tentativas, não for possível desenhar mais nenhum ramo respeitando as condições locais, um ramo "externo" é adicionado. Nesse caso, dois nós previamente selecionados como extremos (conforme explicado anteriormente) são utilizados. Desta maneira, existe a garantia que é sempre possível adicionar um ramo, tendo em consideração todas as possibilidades.

Essa estratégia combina verificação geométrica e adaptabilidade para garantir que o circuito gerado seja coerente e tecnicamente válido.

- **Atualização da matriz:** Com a adição do ramo, é necessário atualizar a matriz para incluir a nova ligação estabelecida. Isto é feito incrementando o elemento da matriz que interseta os dois pontos de conexão escolhidos. Como a matriz é bidimensional, dois elementos da matriz vão ser incrementados.

  Este processo é crucial para organizar o número de ligações estabelecidas, pois para futuras ligações, a escolha dos pontos de conexão pode variar.

O ciclo descrito acima é repetido até que todos os pontos de conexão estejam devidamente conectados e que o número de ramos especificado pelo utilizador seja atendido. Durante esse processo, o algoritmo verifica continuamente se as ligações estabelecidas respeitam as regras de conectividade e garantem a integridade estrutural do circuito. Além disso, é essencial que as conexões sigam as condições mínimas para que o circuito seja funcional, conforme as restrições apresentadas no Subcapítulo 3.3.

Garantir um *layout* bem estruturado é fundamental para evitar inconsistências no circuito gerado, como componentes isolados ou caminhos elétricos interrompidos. Dessa forma, a distribuição dos ramos e pontos de conexão é feita de maneira a assegurar que todos os elementos estejam corretamente interligados.

A Figura 3.10 demonstra o circuito-exemplo com todas as ligações efetuadas, ilustrando como o algoritmo finaliza a estruturação do circuito. Esse exemplo destaca a disposição dos ramos/ligações, evidenciando a organização resultante do processo automatizado. Com essa abordagem, o sistema garante que o circuito gerado seja coerente e funcional, atendendo às especificações do utilizador de forma eficiente.

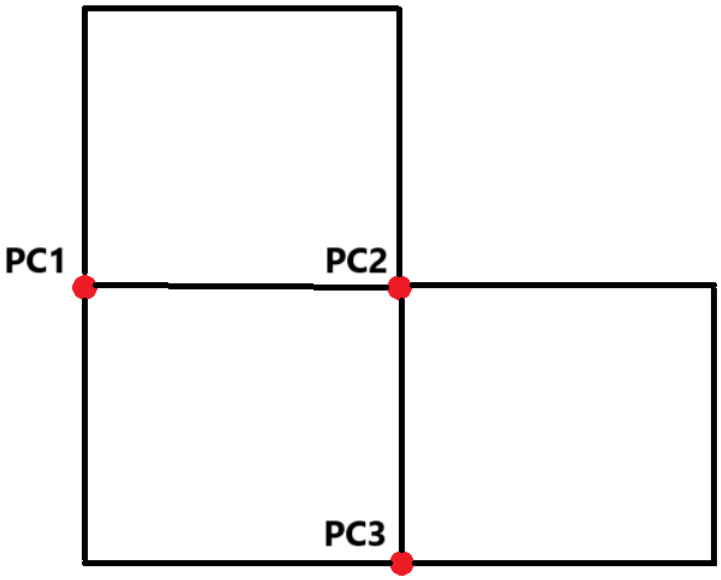


Figura 3.10: Ligações efetuadas para o circuito-exemplo

Estas ligações foram registadas na matriz de adjacência e pode ser apresentada na Tabela 3.3 onde se verifica de forma coerente as ligações que cada ponto de conexão tem.

Tabela 3.3: Matriz de adjacência do circuito-exemplo atualizada com as ligações efetuadas

PC	**PC1**	**PC2**	**PC3**
PC1	0	2	1
PC2	2	0	2
PC3	1	2	0

3.5.4 Verificação e Ajuste do Comprimento dos Ramos

Como o desenho dos ramos tem comprimentos fixos, foi considerado o caso de não haver espaço suficiente disponível para o número de componentes requisitados. Deste modo, caso não haja espaço, é iniciado um processo que aumenta o comprimento dos ramos de forma uniforme.

A técnica que foi utilizada para realizar este processo envolve expandir o ramo para o sentido oposto ao do centro do circuito, ou seja, caso seja selecionado um ramo que esteja no extremo de baixo do circuito, o resto do circuito estaria em cima do ramo escolhido, logo, esse ramo evoluiria para baixo, onde não existe circuito desenhado ainda, de forma a não sobrepor ramos ou pontos de ligação já desenhados.

Para este efeito, foi feita uma filtragem dos ramos que estão nos extremos do circuito (extremo da *esquerda*, *direita*, *cima*, *baixo*) e depois o processo de aumento de comprimento foi executado.

3.5.5 Alocação dos Componentes

Após a definição dos ramos e dos pontos de conexão, o processo de alocação dos componentes torna-se simples, pois o *layout* do circuito já está completamente estruturado. Com os ramos delineados e os pontos de conexão definidos, basta posicionar os componentes nas respetivas localizações, associando-os aos ramos de forma consistente com o circuito gerado. Essa abordagem garante que os componentes sejam integrados de maneira eficiente e alinhados à topologia previamente estabelecida.
Depois, é necessário conhecer que ligações são consideradas como ramos, visto que existem ligações entre dois pontos de conexão, onde esses são, na verdade, o mesmo nó. Para resolver isto, cada ponto de conexão tem atribuído uma etiqueta a dizer qual nó é que pertence a este ponto.
A Figura 3.11 apresenta dois pontos de conexão **A** e **B** que, como não existe nenhum componente entre eles, são representados como um único nó (rodeado a vermelho). Neste caso, seria impossível alocar um ramo nesse segmento.

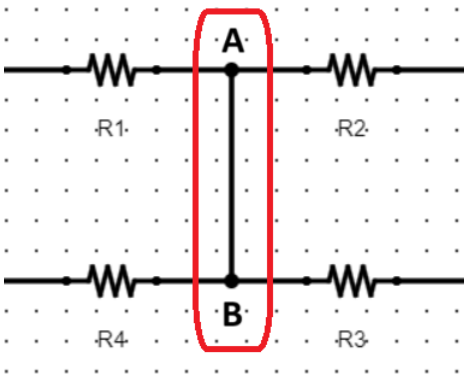


Figura 3.11: Exemplo de dois pontos de conexão que formam um nó

Depois de feita esta filtragem, o processo de alocação dos componentes começa com a atribuição a cada ramo com o número de componentes de maneira uniforme, assim, é possível saber quantos componentes é que cada ramo irá ter. Agora, com a disposição dos componentes feita, um ciclo é iniciado para efetivamente desenhar os componentes. Este ciclo pode ser descrito da seguinte maneira:

1. Escolha de um ramo aleatório
2. Divisão do ramo em segmentos de reta horizontais e verticais. Este processo é feito para tratar dos ramos que têm vértices, ou seja, ramos que mudam de direção num certo ponto. A Figura 3.12 apresenta um ramo com dois vértices, rodeados a vermelho.

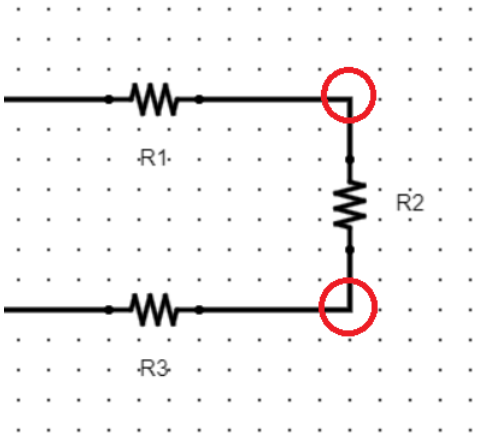


Figura 3.12: Exemplo de um ramo com dois vértices

3. Distribuição uniforme dos componentes para cada segmento de reta

4. Desenho dos componentes para as posições calculadas respetivas e atribuição dos valores para cada componente.

5. Remoção do ramo. Isto é um passo que tem de ser feito, pois caso o ramo inicial não fosse removido, haverá sobreposição do ramo com os componentes.

6. Interligação dos componentes. Após remover o ramo original, tem de ser desenhado um novo ramo que não sobreponha os componentes anteriormente desenhados. Para isto, foram estabelecidas ligações entre os componentes existentes, de forma a redesenhar o ramo removido com a inclusão dos componentes.

Esse processo de alocação dos componentes assegura que todos os elementos do circuito sejam posicionados de forma adequada, seguindo o *layout* previamente estabelecido. O posicionamento correto dos componentes é essencial para manter a organização do circuito, evitar sobreposições e garantir que todas as conexões sejam feitas corretamente. Durante essa etapa, o algoritmo verifica a disposição espacial dos elementos, assegurando que não haja cruzamento indevido de ligações e que cada componente esteja integrado ao restante da estrutura.
Além disso, cada componente é integrado de modo funcional, garantindo que o circuito final seja válido. Isso significa que todas as ligações obedecem às regras elétricas e estruturais definidas, evitando erros que poderiam comprometer o funcionamento do circuito. O objetivo principal desse processo é oferecer um modelo visualmente compreensível e concordante com as especificações do utilizador, permitindo que o circuito gerado seja não apenas tecnicamente correto, mas também intuitivo e fácil de interpretar.
Na Figura 3.13, é apresentado o circuito-exemplo gerado na fase final, com todos os componentes devidamente posicionados. Nota-se que os pontos de conexão definidos anteriormente foram agora consolidados também como nós do circuito.

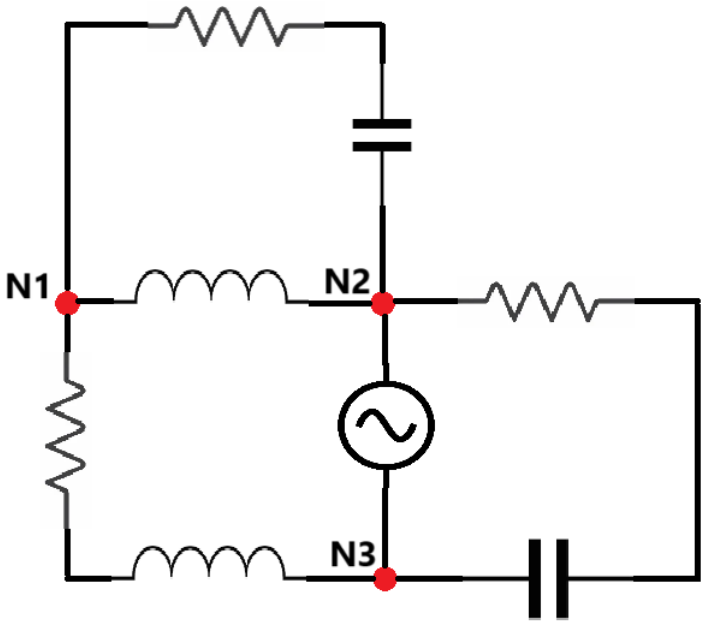


Figura 3.13: Geração total do circuito-exemplo

3.6 Modelo analítico do circuito (*Output*)

À medida que o algoritmo corre, a informação dos ramos, dos componentes e dos pontos de conexão foi sendo utilizada para a criação do ficheiro. No final da execução do algoritmo, este ficheiro é formatado em JSON para que seja adaptado ao modelo de dados existente. O tipo de informação armazenada no ficheiro gerado varia de acordo com o tipo de parâmetro associado ao circuito, seja ele um ramo, um ponto de conexão ou um componente. No entanto, há uma característica comum a todos os parâmetros: a posição $(x,\ y)$ na planilha, que indica onde cada elemento está localizado.
Abaixo, foram enumerados os diferentes tipos de parâmetros e as informações específicas mais importantes associadas a cada um:

1. **Ramo**:
    - Pontos de ligação conectados
    - Componentes conectados
    - Posições $x,\ y$ dos vértices

2. **Ponto de conexão**

    - Ramos conectados
    - Componentes conectados[1]

3. **Componente**

    - Rotação, medida em graus
    - Ramos conectados
    - Pontos de ligação conectados[1]
    - Valor e unidade de medida do componente especificado
    - Caso o componente seja uma fonte de tensão/corrente num circuito AC, é necessário colocar o valor da frequência do sinal em Hz.

Ao seguir esta estrutura, o modelo de dados mantém a coerência e facilita na compreensão dos dados e elementos interligados. Assim, além de funcional, ele é intuitivo, tornando sua utilização eficaz tanto para aplicações automatizadas quanto para revisões/estudos manuais.
Com o ficheiro concordante com o modelo de dados, é efetuado o *download* deste, onde a importação do mesmo no editor de circuitos é bem-sucedida.

[1]Isto não é utilizado, pois seguindo o algoritmo, nunca seria possível um componente estar diretamente conectado a um ponto de conexão, sem haver uma ligação entre eles.

Capítulo 4

Testes e Validação

Neste capítulo, tenta-se sumarizar os testes realizados para verificar o correto funcionamento do algoritmo (e da aplicação *web*). Desta maneira, é possível validar ou não se está tudo de acordo com o que foi planeado.
Para fazer um teste, é fundamental executar a aplicação e depois, para visualizar o circuito gerado, é necessário, após o *download* do ficheiro JSON, fazer a importação do mesmo através do botão "Importar", explicado no Subcapítulo 2.2.1 no Editor de Circuitos[1], que pode ser visitado através deste.
Os testes realizados e escolhidos para explicação neste documento tentam ser o mais abrangentes possível em termos de tipo, modo e complexidade dos circuitos gerados, de acordo com a seguinte metodologia:

1. **Teste com Circuito Mais Simples** - Circuito com apenas 1 ramo e 0 nós em modo DC. Este é o circuito mais simples possível no âmbito deste projeto e que a aplicação pode gerar.

2. **Teste com Circuito de Baixa Complexidade** - Circuito com 3 nós e 5 ramos em modo DC. Circuito simples com nada de extraordinário de modo a demonstrar a execução do algoritmo que provavelmente vai ser mais utilizada, em modo "Personalizado". Também foi adicionado um exemplo complementar com o mesmo número de nós e de ramos, mas com um elevado número de componentes, de modo a manifestar o algoritmo referente ao Subcapítulo 3.5.4.

[1]O Editor de Circuitos pode ser acedido através do seguinte endereço (temporário); http://cloud.microlumin.com:3020/editor.

3. **Teste com Circuito de Elevada Complexidade** - Circuito com 5 nós e 10 ramos em modo AC. Verifica-se o funcionamento em modo AC juntamente com uma complexidade elevada.

4. **Teste com Circuito Parcialmente Aleatório** - Circuito com 4 nós e 6 ramos em modo AC com o número e tipo de elementos aleatórios. Este tipo de seleção dos parâmetros é de certa forma um regularizador de complexidade, que pode ser controlada através do número de ramos e de nós. Com este modo foi demonstrado o funcionamento da "Opção do Circuito"na opção "Aleatório".

5. **Teste com Circuito Totalmente Aleatório** - Circuito onde todos os parâmetros, à exceção da seleção do tipo de circuito, são aleatórios. É demonstrado o número de nós e ramos em modo aleatório, juntamente com a opção "Aleatório"no parâmetro "Opção do Circuito".

Para cada teste, foi mostrada uma figura correspondente ao circuito gerado, no entanto, é importante denotar que, na maioria dos casos, será muito difícil replicar os circuitos demonstrados nas figuras, visto que o modo de geração dos circuitos é diferente em cada execução da aplicação.
Adicionalmente, devido às limitações cartesianas(x, y) da planilha descritas no Subcapítulo 2.2.1, a geração de circuitos com um elevado número de parâmetros pode ultrapassar as fronteiras. Não houve consideração deste pormenor, pois é esperado como objetivo futuro a remoção dessas limitações, de modo a que a planilha seja infinita (descrito pelo Professor André Rocha).

4.1 Circuito Mais Simples

De modo a confirmar a execução básica do algoritmo, o primeiro teste realizado foi o de menor complexidade possível, com apenas 1 ramo e nenhum nó adicional. Este caso inicial foi escolhido para validar se a aplicação é capaz de gerar um circuito com configurações mínimas, testando assim a base do algoritmo de geração.
Parâmetros configurados:

- Opção do Circuito: Personalizado
- Tipo de Circuito: DC
- Resistências: 7
- Fontes de Tensão: 1
- Fontes de Corrente: 0
- Número de nós: 0
- Número de ramos: 1

Resultado do teste:
A Figura 4.1 apresenta o circuito gerado para este teste.

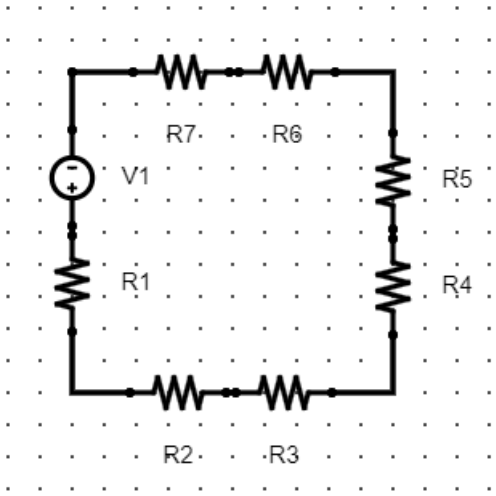


Figura 4.1: Resultado do teste com Circuito Mais Simples

Análise dos resultados

- O número de componentes está correto.
- Não houve geração de nós.
- O modo DC foi corretamente configurado, e a fonte de tensão está no estado adequado.

4.2 Circuito de Baixa Complexidade e Ajuste do Comprimento dos Ramos

Para avaliar o desempenho do algoritmo em situações práticas, foi testado um circuito de baixa complexidade, contendo 3 nós e 5 ramos. Este tipo de configuração reflete cenários simples, mas que podem ocorrer frequentemente em aplicações reais. Este teste também permite verificar a execução de algoritmos específicos, como o de distribuição de componentes e expansão dos ramos, validando sua funcionalidade básica em condições mais típicas de uso. Além disso, foi realizado um teste complementar com o mesmo número de nós e ramos, mas incluindo um número elevado de componentes. Esse exemplo adicional visou verificar o desempenho da aplicação ao lidar com uma maior densidade de elementos, conforme descrito no Subcapítulo 3.5.4.

4.2.1 Circuito de Baixa Complexidade

Parâmetros configurados:

- Opção do Circuito: Personalizado
- Tipo de Circuito: DC
- Resistências: 8
- Fontes de Tensão: 2
- Fontes de Corrente: 1
- Número de nós: 3
- Número de ramos: 5

Resultado do teste:
A Figura 4.2 apresenta o circuito gerado para este teste.

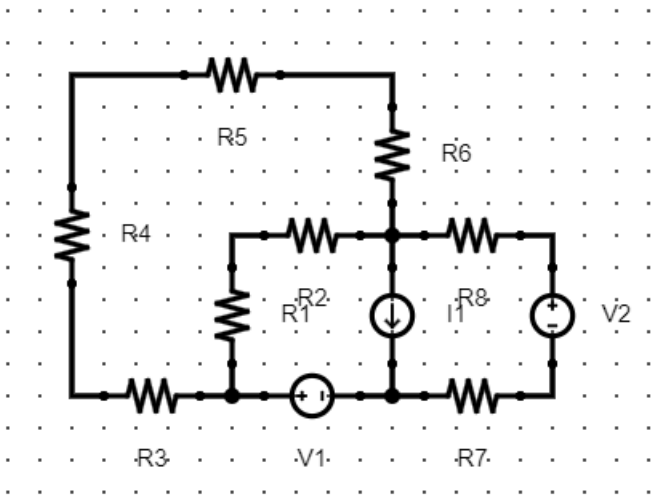


Figura 4.2: Resultado do teste com Circuito de Baixa Complexidade

Análise dos resultados

- O número de nós e ramos está correto.
- Componentes foram divididos pelos ramos de maneira coerente
- A fonte de corrente está devidamente configurada

4.2.2 Ajuste do Comprimento dos Ramos

Parâmetros configurados:

- Opção do Circuito: Personalizado
- Tipo de Circuito: DC
- Resistências: 10
- Fontes de Tensão: 3
- Fontes de Corrente: 3
- Número de nós: 3
- Número de ramos: 5

Resultado do teste:
A Figura 4.3 apresenta o circuito gerado para este teste.

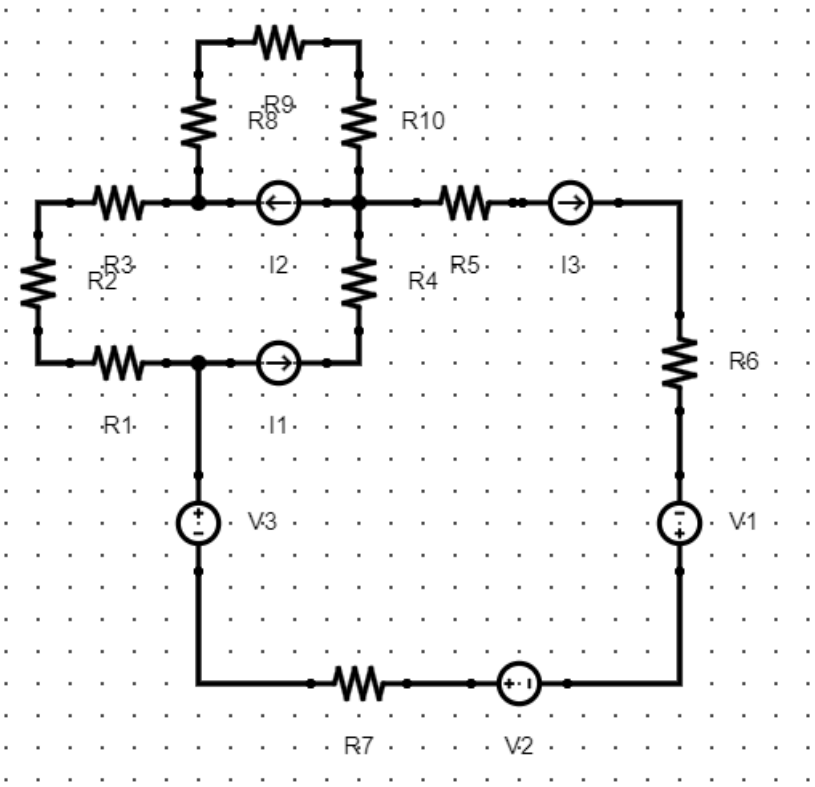


Figura 4.3: Resultado do teste com Ajuste do Comprimento dos Ramos

Análise dos resultados

É de notar que o ramo que contém os componentes *R5, I3, R6, V1, V2, R7* e *V3* foi ajustado de modo a caberem mais componentes. O mesmo aconteceu com o ramo que contém os componentes *R1, R2* e *R3*, onde, se não fosse realizada a verificação do comprimento dos ramos, seria um ramo com apenas 1 segmento de reta vertical, com espaço para somente 1 componente.

- O número de componentes está correto.
- O comprimento dos ramos foram ajustados.

4.3 Circuito de Elevada Complexidade e Atribuição de Valores

Para validar a funcionalidade do algoritmo em cenários desafiadores, foi realizado um teste com um circuito de alta complexidade. Este circuito foi configurado para incluir 5 nós e 10 ramos, utilizando todos os tipos de componentes disponíveis e a operar em modo AC. Esse tipo de teste é essencial para verificar a capacidade da aplicação de lidar com circuitos complexos.

O uso do modo AC adiciona um nível extra de dificuldade, já que a lista dos componentes aumenta[1] e é necessário considerar o valor da frequência nas fontes de tensão e corrente. Além disso, ao incluir todos os componentes, o teste avalia a capacidade do *software* de alocar corretamente os elementos em um circuito com elevada diversidade.

Juntamente, foi testado a atribuição de valores aos componentes, onde foi realizado um teste específico, assegurando que cada componente recebe um valor dentro dos parâmetros estabelecidos. Esse teste é essencial para verificar a precisão do *software* na configuração dos valores e garantir que todos os elementos do circuito operem conforme esperado. A correta atribuição de valores é especialmente importante no modo AC, pois a presença de componentes como condensadores e bobinas requer a consideração de impedâncias e da frequência das fontes.

[1] No modo AC, além de resistências, fontes de corrente e de tensão, são adicionados bobinas e condensadores.

4.3.1 Circuito de Elevada Complexidade

Parâmetros configurados:

- Opção do Circuito: Personalizado
- Tipo de Circuito: AC
- Resistências: 7
- Condensadores: 5
- Bobinas: 5
- Fontes de Tensão: 1
- Fontes de Corrente: 1
- Número de nós: 5
- Número de ramos: 10

Resultado do teste:
A Figura 4.4 apresenta o circuito gerado para este teste.

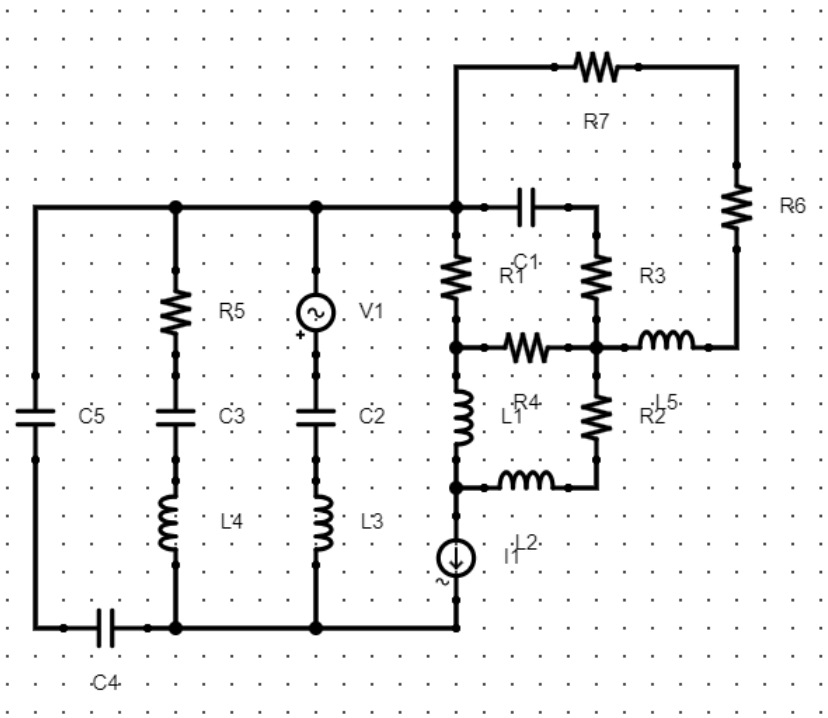


Figura 4.4: Resultado do teste com Circuito de Elevada Complexidade

Análise dos resultados

- O número de componentes está correto
- Todos os tipos de componentes foram gerados
- O modo AC foi confirmado, pois o símbolo da fonte de tensão e da fonte de corrente mudaram para incluírem frequência.

4.3.2 Atribuição de Valores

O circuito utilizado no subcapítulo anterior, foi já parametrizado com os limites superior e inferior, que podem ser agrupados e explicados na Tabela 4.1.

Tabela 4.1: Limites superiores e inferiores definidos para cada componente.

Componente	Limite Inferior	Limite Superior	Unidade
Resistência	200	600	Ω
Condensador	1	1000	nF
Bobina	900	1000	mH
Fonte de Tensão	5	25	V
Fonte de Corrente	10	15	A

Resultado do teste:
A Figura 4.5 apresenta os valores de cada componente.

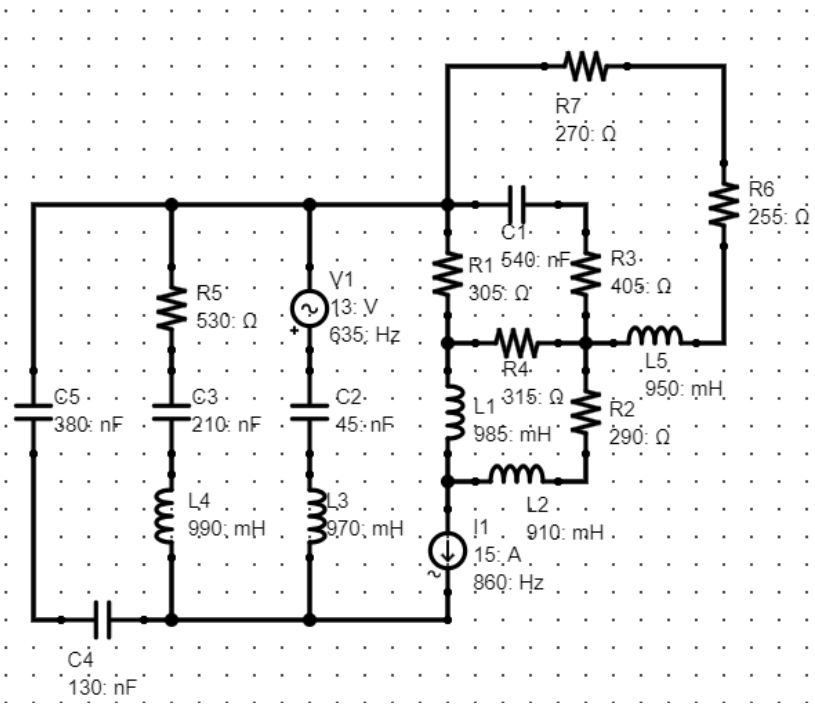


Figura 4.5: Atribuição dos valores para cada componente

A Tabela 4.1 e os valores de cada componente foram recolhidos e agrupados num gráfico para melhor compreensão, como é possível visualizar na Figura 4.6.

Figura 4.6: Valores de cada componente

Análise dos resultados

Os valores atribuídos a todos os componentes foram verificados, e foi constatado que estão dentro dos limites predefinidos, conforme especificado na Tabela 4.1. Em detalhe:

- Resistências: Os valores gerados para as resistências situaram-se no intervalo de 200 Ω a 600 Ω.
- Condensadores: Os valores atribuídos aos condensadores ficaram dentro do intervalo de 1 nF a 1000 nF.
- Bobinas: Os valores das bobinas geradas respeitaram o intervalo entre 900 mH e 1000 mH.
- Fontes de Tensão: O valor atribuído à fonte de tensão situa-se entre 5 V e 25 V.
- Fontes de Corrente: O valor atribuído à fonte de corrente ficou dentro do intervalo de 10 A a 15 A.

4.4 Circuito Parcialmente Aleatório

Este teste teve como objetivo verificar a capacidade do algoritmo em gerar um circuito utilizando parâmetros parcialmente aleatórios mantendo, no entanto, um controle sobre a complexidade por meio da definição do número de ramos e nós. Para este caso, foram definidos 4 nós e 6 ramos em modo AC, com os tipos e quantidades de componentes configurados de forma aleatória.
A escolha deste teste permite avaliar a flexibilidade do algoritmo em acomodar diferentes configurações sem um padrão pré-definido.

Parâmetros configurados:

- Opção do Circuito: Aleatório
- Tipo de Circuito: AC
- Número de nós: 4
- Número de ramos: 6

Resultado do teste:
A Figura 4.7 apresenta o circuito gerado para este teste.

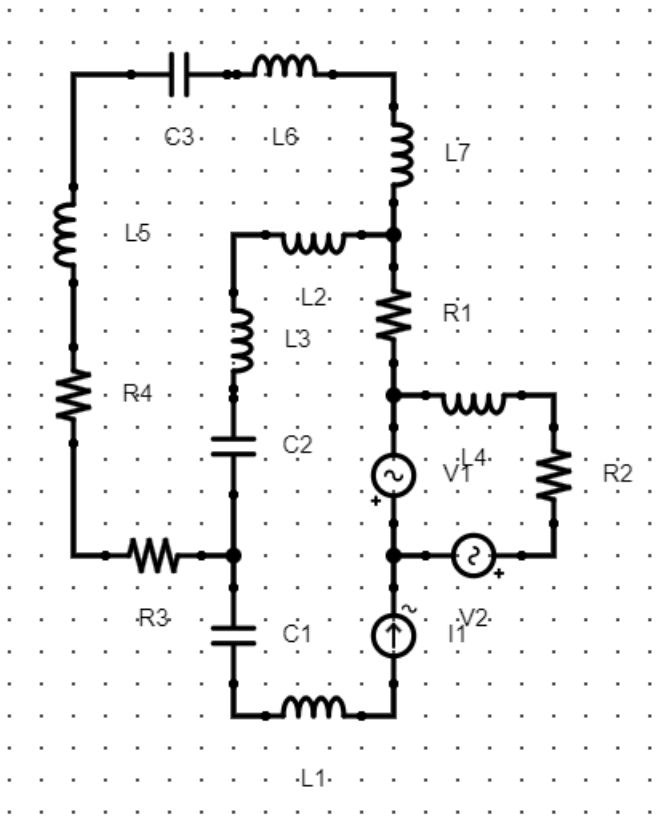


Figura 4.7: Resultado do teste com Circuito Parcialmente Aleatório

Análise dos resultados

- Número de nós e ramos estão corretos
- O número de cada tipo de componente está aleatório
- Não houve erros na geração do circuito

4.5 Circuito Totalmente Aleatório

O teste com o circuito totalmente aleatório foi projetado para avaliar a capacidade do algoritmo de lidar com a máxima incerteza, em que todos os parâmetros, exceto a seleção do tipo de circuito (neste teste foi selecionado o modo AC), foram definidos aleatoriamente.
Este teste apresenta o maior grau de variabilidade, desafiando o algoritmo a operar de forma eficiente mesmo sem restrições ou padrões predefinidos. Ele também permite verificar como o sistema responde a combinações potencialmente complexas ou improváveis, o que permitiu avaliar o funcionamento do *software*.
Parâmetros configurados:

- Opção do Circuito: Aleatório
- Tipo de Circuito: AC
- Número de nós: Aleatório
- Número de ramos: Aleatório

Resultado do teste:
A Figura 4.8 apresenta o circuito gerado para este teste.

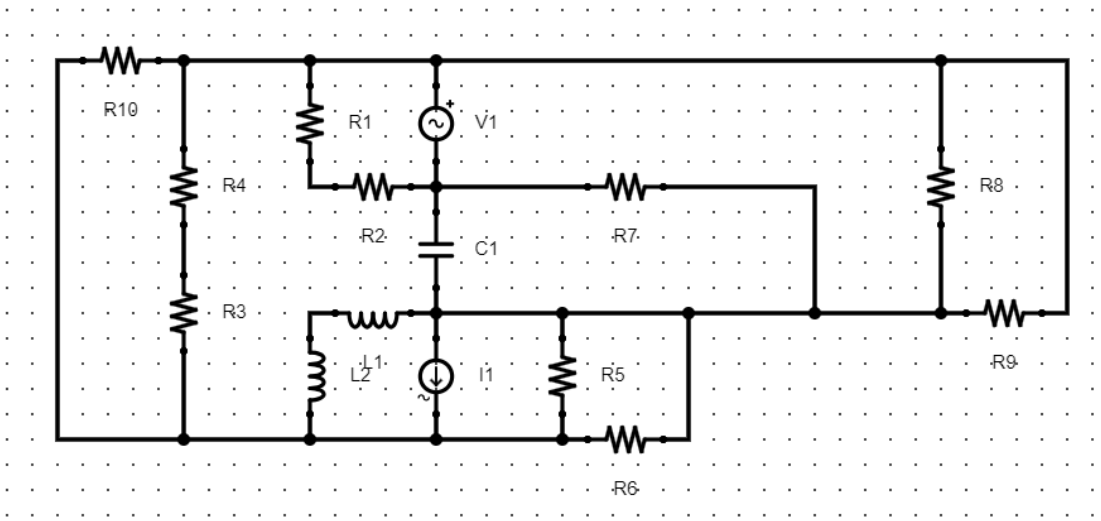


Figura 4.8: Resultado do teste com Circuito Totalmente Aleatório

Análise dos resultados

Conforme ilustrado na Figura 4.8, o circuito gerado apresenta 4 nós, 12 ramos e um total de 15 componentes. Nota-se que o algoritmo executou corretamente a geração, sem apresentar quaisquer erros, resultando num circuito válido e funcional. Este caso extremo é essencial para testar a robustez e adaptabilidade do sistema, garantindo que, mesmo em cenários caóticos, o algoritmo consegue gerar um circuito funcional e coerente.

4.6 Análise Final

Os testes realizados demonstraram a capacidade da aplicação de gerar e parametrizar circuitos de diversas complexidades e níveis de aleatoriedade. Os testes realizados confirmaram que o algoritmo corre devidamente, sem erros e como pretendido, garantindo:

- **Flexibilidade:** Suporte para diferentes tipos de circuitos e modos de operação.
- **Confiabilidade:** Geração consistente e precisa de topologias e valores dentro dos limites definidos.
- **Escalabilidade:** Capacidade de lidar com configurações simples e complexas, sem perda de funcionalidade.

Os resultados obtidos são promissores, consolidando a base para futuros aprimoramentos do sistema, no entanto, houve também desafios/problemas que dificultaram a progressão do projeto, mas que abriram conhecimentos para a elaboração do mesmo. Um exemplo destes desafios foi o próprio algoritmo, na fase de geração de ramos e nós, onde foram feitos vários testes para verificar qual seria a abordagem melhor. Inicialmente, foi testada a possibilidade de geração do circuito, mas em vez de começar com a geração dos nós, começava com a geração dos ramos. Após desenvolver grande parte do código para essa tentativa, começaram a existir várias exceções ao algoritmo, o que retirava a própria definição da palavra "algoritmo". Logo, esta tentativa foi descartada, no entanto, tempo que poderia ser utilizado para melhorias na aplicação/algoritmo, foi utilizado a desenvolver este código errado.
Com os resultados promissores, o próximo passo será abordar as limitações identificadas e explorar possibilidades de otimização e expansão.

Capítulo 5

Conclusões

O desenvolvimento deste algoritmo/aplicação permite disponibilizar um importante bloco funcional para a aplicação *U=RIsolve Academy*, uma ferramenta inovadora e prática para apoiar os estudantes do curso da Licenciatura em Engenharia Eletrotécnica e de Computadores (LEEC) no Instituto Superior de Engenharia do Porto (ISEP), especialmente os que iniciam a sua jornada no estudo e análise de circuitos elétricos. Desde o início, foi identificada a necessidade de uma solução que não apenas complementasse o Editor de Circuitos (em fase de conclusão) da plataforma *U=RIsolve Academy*, mas que também introduzisse um método eficiente para gerar automaticamente modelos de circuitos personalizados, de modo a complementar a versão atual .

A abordagem adotada baseou-se numa arquitetura acessível e de fácil utilização, aproveitando tecnologias amplamente suportadas como *HTML*, *JavaScript* e *Python*. O resultado é uma aplicação funcional em maior parte dos navegadores/*browsers*, que permite a criação de circuitos com parametrizações ajustáveis, promovendo uma experiência de aprendizagem prática e adaptável às necessidades individuais dos utilizadores.

Durante o desenvolvimento, diversos desafios técnicos foram superados, como a implementação do algoritmo de Geração Automática de Modelos de circuitos Elétricos (GAME), a adaptação dos circuitos para o modelo de dados em utilização no editor e a implementação das limitações impostas pelos critérios de complexidade na aplicação. Os testes realizados validaram a eficácia da ferramenta em diferentes cenários, desde circuitos simples até configurações de elevada complexidade e modelos

aleatórios.
Adicionalmente, a integração harmoniosa com o Editor de Circuitos existente demonstrou o potencial da aplicação para complementar as capacidades do *U=RIsolve Academy*, reforçando o objetivo de fornecer aos estudantes um ambiente de aprendizagem robusto e intuitivo.

5.1 Trabalho Futuro

Apesar do sucesso alcançado, existem oportunidades para expandir e melhorar a aplicação. Algumas sugestões incluem:

- Adicionar uma opção na aplicação que adapte o número de componentes a gerar de acordo com o número de ramos.
- Escolha de complexidade na aplicação, onde é feita a geração de um circuito aleatório dependente do nível da complexidade.

A sugestão de adaptar o número de componentes conforme o número de ramos, por exemplo, exigiria uma análise cuidadosa sobre como distribuir adequadamente os componentes no circuito gerado, o que demandaria mais tempo de implementação e ajustes contínuos. Já a adição de uma opção de complexidade implicaria a criação de um novo módulo para geração de circuitos aleatórios, o que também envolveria um processo mais complexo de codificação.
Além disso, ambas as sugestões envolveriam uma fase extensiva de testes para garantir que as novas funcionalidades não impactassem negativamente o funcionamento do algoritmo/aplicação existente. Esses testes são fundamentais para validar o comportamento do sistema em diferentes cenários, o que demandaria tempo adicional para a coleta e análise dos resultados.
Portanto, embora essas sugestões sejam relevantes e poderiam trazer melhorias para a aplicação, a sua implementação foi além da capacidade de tempo disponível e da extensão do trabalho imposto por cada uma delas.

Referências

[1] J. Bondy and U. Murty, *Graph Theory*. London: Springer, graduate texts in mathematics, vol. 244 ed., 2008. [Citado nas páginas xi e 11]

[2] J. Vrbik, "Solving electrical circuits via graph theory," *Applied Mathematics*, vol. 13, pp. 77–86, 2022. [Citado nas páginas xi e 11]

[3] A. F. Dorado, *Electrical Circuits: Analysis, Terminology, and Basic Concepts*. Boca Raton, FL: CRC Press, 2021. [Citado nas páginas xi, 10 e 11]

[4] The Qucs Team, "Schematic File Format," 2025. [Citado nas páginas xi e 18]

[5] A. Rocha, *Descrição do Modelo de Dados: U=RIsolve Academy*. Instituto Superior de Engenharia do Porto, Oct. 2024. [Citado na página 5]

[6] A. T. da Rocha, "Análise de circuitos elétricos: Ferramentas computacionais para aprendizagem." Planned Individual Study. up202003372@fe.up.pt. [Citado na página 5]

[7] A. Rocha, M. Alves, L. Sousa, and F. Pereira, "Métodos gerais de análise de circuitos elétricos," tech. rep., Instituto Superior de Engenharia do Porto, Porto, Portugal, Oct. 2022. Resumo. [Citado na página 5]

[8] H. Zhu, S. K. Lim, and D. Z. Pan, "Power distribution network design for VLSI," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 3, pp. 170–183, 2005. [Citado na página 5]

[9] B. Bollabás, *Graph Theory*. New York: Springer-Verlag, 1979. [Citado na página 11]

[10] O. S. Peters, *Ground Connections for Eletrical Systems*. Hoboken, NJ: Technologic Papers of the Bureau of Standards, 1918. Referência específica: página 7. [Citado na página 14]

[11] AutoCircuits, "Autocircuits: Automated circuit generator," 2025. [Citado na página 17]

[12] C.-W. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, 1975. [Citado na página 17]

[13] The Qucs Team, "Quite Universal Circuit Simulator (Qucs)," 2025. [Citado na página 18]

[14] K. S. Kumar, *Electric circuits and networks.* Pearson Education India, 2008. [Citado na página 23]

[15] C. Desoer and E. Kuh, *Basic Circuit Theory.* McGraw-Hill, 1969. [Citado na página 23]

[16] X. Wang, "Brief summary of frequently-used properties of the floor function," *IOSR Journal of Mathematics (IOSR-JM)*, vol. 13, pp. 46–48, September–October 2017. Department of Mechatronics, Foshan University, PR China. [Citado na página 24]