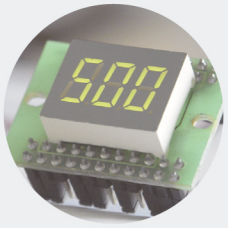
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Unificação e Expansão da Plataforma U=RIsolve Academy: Projeto e Implementação de um Editor/Simulador de Circuitos

ÂNGELO JOSÉ COUTO PINHEIRO
novembro de 2025

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Unificação e Expansão da Plataforma U=RIsolve Academy: Projeto e Implementação de um Editor/Simulador de Circuitos

Ângelo Pinheiro

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Novembro, 2025

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Automação e Sistemas.

Candidato: Ângelo Pinheiro, Nº 1190398, 1190398@isep.ipp.pt

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

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

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

Novembro, 2025

Agradecimentos

Dedico esta página para acomodar os devidos agradecimentos a todos os que deixaram o seu contributo no decorrer do trabalho.

Agradeço ao Professor Mário Alves e ao Professor André Rocha pela orientação de excelência e apoio a nível, não só académico, como também pessoal. Foram decerto a principal fonte de conhecimento e guia para o sucesso do projeto apresentado nesta dissertação, bem como continuarão certamente a sê-lo no futuro do U=RIsolve Academy.

Agradeço à minha namorada pelo apoio nos momentos mais difíceis, principalmente quando ponderei desistir do projeto por completo. Este suporte foi claramente um alicerce essencial e permitiu-me acumular a força necessária para persistir e continuar, resolvendo todos os problemas imagináveis pelo caminho.

Por fim, mas não menos importante, agradeço a todos os participantes dos testes práticos à aplicação, docentes, alunos e ex-colegas de curso, que fortaleceram o conjunto de sugestões e problemas a resolver, resultando numa aplicação mais robusta, completa e madura.

A todos os mencionados, um muitíssimo obrigado.

Resumo

Esta dissertação documenta o processo de desenvolvimento e melhoria da plataforma U=RIsolve Academy, uma ferramenta de aprendizagem e análise de circuitos elétricos, focando em dois objetivos principais: em primeiro lugar é relatado um trabalho de unificação da plataforma, isto é, a transformação de diversos módulos realizados por sujeitos distintos e com características distintas numa ferramenta coesa e homogénea; em segundo lugar é apresentado o processo de estudo, projeto, e desenvolvimento de um editor/simulador de circuitos elétricos, que juntamente com o restante da plataforma visam proporcionar ao utilizador uma fonte de ensino/aprendizagem com diversas funcionalidades integradas, levando a uma experiência de utilização apelativa.

Por fim, a modo a verificar o bom funcionamento da plataforma e do editor de circuitos, foram efetuados testes por um conjunto selecionado de *beta-tester's*, especialistas/docentes. Este conjunto de pessoas respondeu a um extenso questionário concebido especificamente para avaliar a experiência de utilização em diversas dimensões, nomeadamente em termos de funcionalidades, intuitividade e *design* gráfico da aplicação.

Assim, este projeto contribui para o avanço da plataforma U=RIsolve Academy, que por sua vez tem o potencial de melhorar o processo de ensino/aprendizagem acerca de circuitos elétricos.

A nova plataforma encontra-se disponível em urisolve.pt/academy.

Palavras-Chave: U=RIsolve, Unificação, Editor de Circuitos, Didático, Apelativo, Análise de Circuitos Elétricos, Simulador de Circuitos Elétricos, Interface de Utilizador, Experiência de Utilizador

Abstract

This dissertation details the development and enhancement of the U=RIsolve Academy platform, a tool for learning and analyzing electrical circuits. The first objective is to unify the platform by integrating modules created by different developers into a cohesive system. The second objective is to design and develop an electrical circuit editor and simulator that, combined with the platform, offers users a comprehensive and engaging learning environment.

To validate the platform and circuit editor, a group of specialists and instructors served as beta testers. They completed a detailed questionnaire assessing user experience in terms of functionality, intuitiveness, and graphical design.

Thus, this project contributes to the advancement of the U=RIsolve Academy platform, which in turn has the potential to improve the teaching and learning process related to electrical circuits.

The new platform is available at urisolve.pt/academy

Keywords: U=RIsolve, Unification, Circuit Editor, Educational, Engaging, Electrical Circuit Analysis, Circuit Simulator, User Interface, User Experience

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Glossário

backend

Parte de um sistema de software que lida com a lógica de negócio, processamento de dados e comunicação com bases de dados, geralmente não visível para o utilizador final

beta-tester

Utilizador que testa uma aplicação em fase de desenvolvimento, com o objetivo de identificar erros e fornecer feedback sobre a usabilidade e funcionalidade do software

frontend

Parte de um sistema de software que interage diretamente com o utilizador final, incluindo a interface gráfica e a experiência do utilizador

User experience

Experiência do utilizador; conjunto de perceções e respostas de uma pessoa resultantes da utilização e/ou antecipação da utilização de um produto, sistema ou serviço

User interface

Interface de utilizador; conjunto de elementos gráficos e mecanismos de interação que permitem ao utilizador comunicar e interagir com uma aplicação ou sistema informático

Lista de Acrónimos

API	*Application Programming Interface*
CA	Corrente Alternada
CC	Corrente Contínua
CSS	*Cascading Style Sheets*
DiCES	Didactic Circuit Editor and Simulator
HTML	*HyperText Markup Language*
iGen	Inteligent circuit Generator
JSON	*JavaScript Object Notation*
MCM	Método da Corrente nas Malhas
MCR	Método da Corrente nos Ramos
MTN	Método das Tensões nos Nós
MVC	*Model-View-Controller*
XML	*eXtensible Markup Language*

Lista de Símbolos

Símbolo	Descrição	Unidades
C	Capacitância	F
f	Frequência	Hz
I	Corrente	A
L	Indutância	H
P	Potência	W
R	Resistência	Ω
U	Tensão	V

Capítulo 1

Introdução

A capacidade de resolver um circuito elétrico, isto é, descobrir o potencial em todos os nós e as correntes em todos os ramos de um circuito elétrico, é fundamental para a aprendizagem da engenharia eletrotécnica. Esta é também, no entanto, uma grande dificuldade para iniciantes na área, quer porque as resoluções manuais são, por vezes, extensas, quer porque as ferramentas de edição e análise de circuitos tendem a ser complexas e pouco intuitivas. [1]

Conclui-se assim, a ausência de ferramentas simples e vocacionadas para a aprendizagem, em prol de ferramentas profissionais com leques extensos de funcionalidades, justificando-se o desenvolvimento de uma nova que pretende combater e preencher esta lacuna.

A plataforma U=RIsolve Academy tem como objetivo melhorar a interface gráfica de utilizador, facilitando o emprego de funcionalidades vocacionadas para o ensino de Engenharia Eletrotécnica, atenuando as dificuldades do aluno, levando a uma construção de conhecimentos mais fluída e intuitiva.

1.1 Contextualização

A plataforma U=RIsolve Academy tem vindo a ser alicerçada e desenvolvida a partir de projetos de Licenciatura e Mestrado em Engenharia Eletrotécnica e de Computadores, maioritariamente de estudantes do Instituto Superior de Engenharia do Porto.

Esta visão web da aplicação U=RIsolve começou por suportar apenas a análise de circuitos elétricos pelo Método das Correntes nas Malhas (MCM), submetendo-lhe para isso um ficheiro do tipo **Netlist**. Era, porém, ainda muito prematura, quer na interface, quer na eficiência do método de resolução.

Foi assim desenvolvida uma nova versão, que integrava uma interface, com o utilizador (UI), mais apelativa, principalmente na apresentação dos resultados, e onde era integrado o **Método das Tensões nos Nós** (MTN) [2].

Após esta versão, foram desenvolvidos em paralelo outros métodos de resolução, nomeadamente: nova versão do **Método das Correntes nas Malhas** (MCM) [3], **Método das Correntes nos Ramos** (MCR) [4] e **Teorema da Sobreposição** (TSP) [5], que foram implementados e integrados na aplicação existente.

De forma simultânea com este desenvolvimento, foi também desenvolvida uma primeira versão de um editor de circuitos, cuja intenção seria a sua integração também na aplicação, criando uma plataforma de aprendizagem auto-suficiente. Esta plataforma começou a ser vista de forma mais abrangente, adicionando uma camada de abstração superior, tendo sido nomeada de: **U=RIsolve Academy** [6].

1.2 Definição do Problema

O desenvolvimento simultâneo de múltiplos módulos e funcionalidades numa única plataforma de ensino/aprendizagem oferece vantagens claras, com a principal sendo a celeridade do progresso da implementação. No entanto, esta abordagem tem limitações, especialmente em contexto académico: o facto de diferentes indivíduos trabalharem na mesma aplicação pode originar heterogeneidade entre os seus módulos.

Um caso particular de um módulo cuja implementação teve de ser completamente refeita (de raiz) é o do editor de circuitos: a sua primeira versão [7] revelou possuir características de implementação que impossibilitavam o seu posterior desenvolvimento e aperfeiçoamento, bem como a inclusão de novas funcionalidades.

Este desenvolvimento desigual leva a uma necessidade de existir uma fase de unificação de todos os trabalhos, tornando-os equivalentes em termos de qualidade e funcionamento, um dos objetivos deste projeto/tese.

1.3 Objetivos

Deste modo, enumeram-se os diversos objetivos para este trabalho, dividindo-os em quatro blocos principais:

1. Harmonização de código-fonte da plataforma *web* U=RIsolve Academy
    (a) Reestruturação da arquitetura de *software* do U=RIsolve Academy;

(b) Integração de métodos de análise previamente concebidos;

(c) Melhoria da qualidade gráfica da interface do utilizador (UI);

2. Desenvolvimento de um novo editor de circuitos

(a) Estudo da arte sobre editores e editores/simuladores de circuitos, nomeadamente o anteriormente desenvolvido nesta framework [7];

(b) Atualização e unificação do modelo de dados para representação do circuito elétrico, quer a nível lógico, quer a nível gráfico;

(c) Projetar e implementar um novo editor/simulador de circuitos elétricos;

(d) Analisar a qualidade do editor/simulador através de testes com vários utilizadores.

1.4 Plano de Trabalho

O plano de trabalho estipulado para o desenvolvimento do projeto inclui todo o processo de análise do estado da arte, desenvolvimento da plataforma base, desenvolvimento do editor/simulador, e testes funcionais. Inclui também a escrita deste mesmo documento.

Assim, é representado na Tabela 1.1 uma divisão temporal das tarefas como forma de organização do trabalho. Consiste numa tabela concebida tendo por base dois princípios básicos, acreditando que estes contribuem para uma gestão eficiente do tempo: decisão do que fazer e não fazer, e decisão entre o que fazer primeiro e fazer depois.

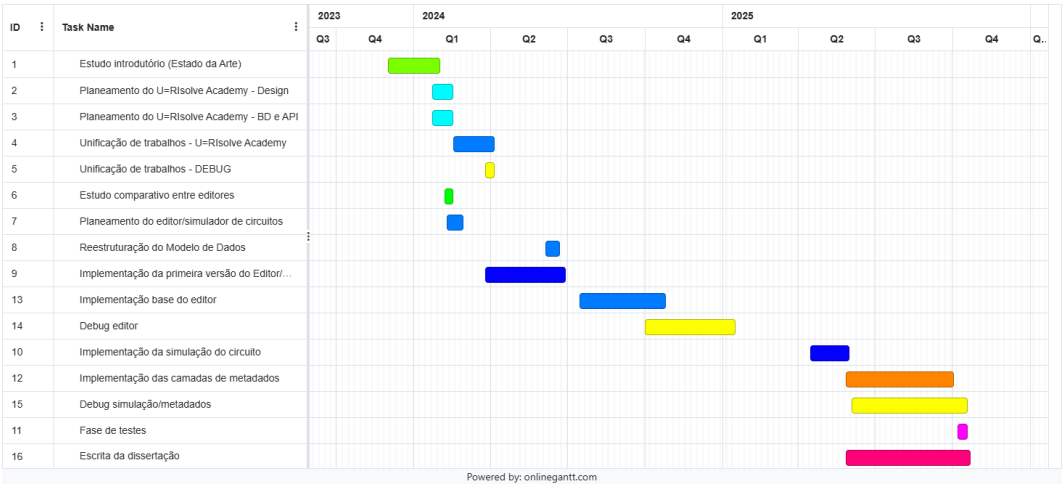


Figura 1.1: Tabela de escalonamento do projeto

Nota: consoante o decorrer do projeto, este escalonamento teve de ser devidamente atualizado de modo a rever com a máxima exatidão a janela temporal de todas

as tarefas. A tabela anterior alberga o planeamento-resultado de várias modificações ao planeamento original.

1.5 Contribuições

As principais contribuições deste trabalho são as seguintes:

- Desenvolvimento de um editor/simulador de circuitos elétricos interativo, integrado na plataforma U=RIsolve Academy, permitindo aos utilizadores criar, editar e analisar circuitos diretamente na mesma;
- Implementação de um modelo de dados em formato JSON para representar circuitos elétricos, facilitando a manipulação e armazenamento eficiente dos dados dos circuitos;
- Criação de algoritmos para a gestão das conexões entre componentes e fios de ligação, garantindo a integridade do circuito durante as operações de edição;
- Desenvolvimento de camadas de metadados que fornecem informações adicionais sobre os componentes do circuito, como valores e indicações gráficas de potenciais e correntes elétricas;
- Integração do editor/simulador com o analisador de circuitos do U=RIsolve Academy, permitindo a análise passo-a-passo dos circuitos criados pelos utilizadores, segundo os métodos de análise disponíveis [1]
- Realização de testes e validação do editor/simulador de circuitos, por parte da equipa (autor e orientadores), assegurando a sua funcionalidade, usabilidade e desempenho. Análise dos resultados/comentários de 13 beta-tester's, que responderam a um questionário especificamente desenvolvido para o efeito.

1.6 Organização da Dissertação

No Capítulo 1 é feita uma introdução ao tema do projeto, apresentando o contexto, objetivos, contribuições, plano de trabalho e organização da dissertação.

No Capítulo 2 é apresentada, e re-projetada a arquitetura da plataforma U=RIsolve Academy, onde o editor de circuitos será integrado.

No Capítulo 3 é desenvolvido e descrito o modelo de dados dos circuitos elétricos, que servirá de base para a implementação do editor/simulador de circuitos (DiCES).

No Capítulo 4 é descrita a implementação do DiCES, detalhando as diferentes funcionalidades implementadas, bem como as decisões de design tomadas.

[1] Método das Correntes nas Malhas (MCM) [3] e Método das Correntes nos Ramos (MCR) [4]. O Método das Tensões Nodais (MTN) [2] e Teorema da Sobreposição (TSP) [5] ainda necessitam de ser readaptados

No Capítulo 5 é abordado o desenvolvimento das camadas de metadados do DiCES.

No Capítulo 6 são apresentados os testes realizados ao DiCES, bem como a validação do mesmo através de um questionário aplicado a diversos utilizadores/beta-tester's.

No Capítulo 7 são apresentadas as conclusões do trabalho realizado, bem como sugestões para trabalhos futuros.

Capítulo 2

U=RIsolve Academy

O primeiro passo da implementação do projeto implicou criar uma estrutura de software de raiz, proporcionando um alicerce estável onde todos os módulos assentarão. Nesta fase foi ainda possível fazer um *re-design* à aplicação, dando-lhe um aspeto mais moderno e apelativo. Assim, tal como qualquer aplicação *web*, o U=RIsolve Academy foi planeado quanto aos seus possíveis atores e casos de uso, tendo ainda ficado definida uma possível estrutura de base de dados, quando forem implementadas funcionalidades de *backend*, *a posteriori*.

Por último, foram desenvolvidas algumas páginas de teste de *design* cuja função foi dar a possibilidade de visualizar o aspeto estimado da aplicação no seu estado final e poder alinhavar aspetos gerais sem a necessidade de programar, validando o novo *design* criado.

É sempre de salientar que esta abordagem à arquitetura de software de um sistema deste género permite a sua futura expansão, facilitando melhorias e adições de funcionalidades.

2.1 Contexto tecnológico

O U=RIsolve Academy pretende ser uma plataforma de ensino/auto-aprendizagem integrada para um estudante de engenharia eletrotécnica. Consiste numa aplicação *web* bem estruturada, agrupando várias ferramentas ao dispor do utilizador, tais como a personalização e geração de *Quizzes* e pontuações. Do Academy, faz também parte integrante o Analisador de Circuitos, uma ferramenta capaz de receber

um ficheiro do tipo *netlist* e fornecer uma análise passo-a-passo do circuito segundo um dos métodos disponíveis [8]. O seguinte diagrama de blocos ilustra a arquitetura geral da aplicação U=RIsolve Academy.

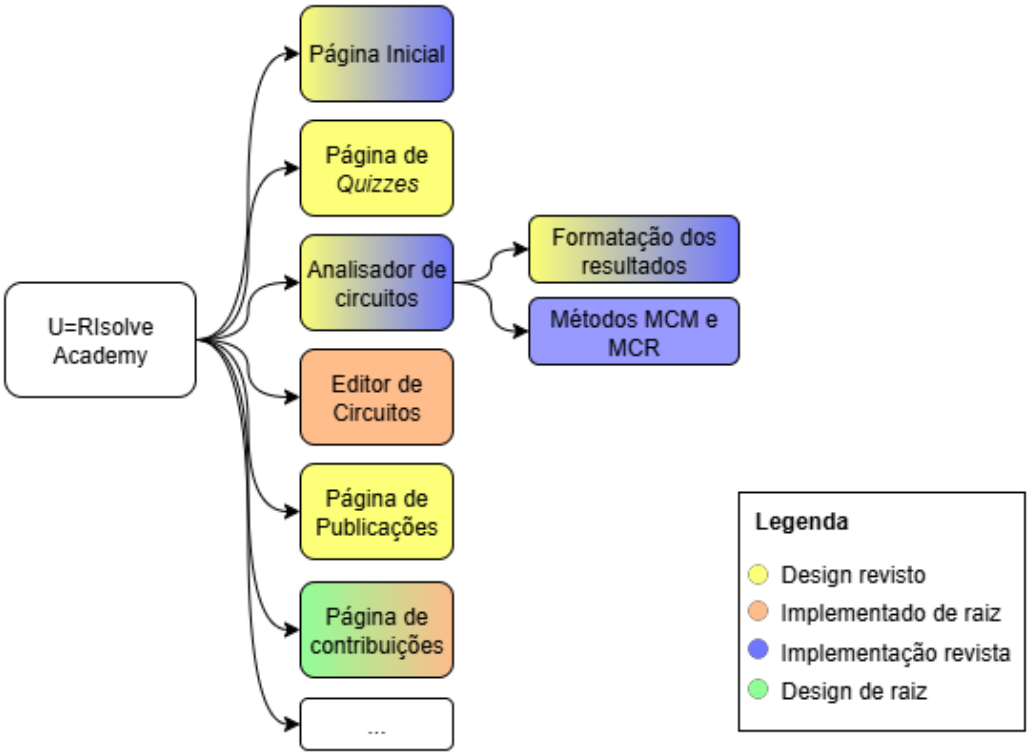


Figura 2.1: Diagrama da arquitetura geral da aplicação U=RIsolve Academy

Pela legenda da Figura 2.1, é possível perceber ainda em que blocos este projeto teve intervenção.

Análise estrutural

Numa perspetiva mais abrangente, esta plataforma permite a criação de *Quizzes*: uma sequência ordenada de perguntas sobre um determinado tema (ou abordando temas aleatórios). Os temas incluem as seguintes matérias, apresentadas como abas/opções no separador de treino:

- **Métodos de análise** — permite ao estudante escolher entre análise CC ou CA, o método de resolução (MTN, MCR ou MCM) e o nível de dificuldade das questões;
- **Estudo por temas** — segmenta o conteúdo em tópicos específicos, como nós, ramos, malhas, equações e correntes/tensões;
- **Estudo aleatório** — fornece uma seleção de questões dispersas.

Pontos fortes e limitações

Entre os principais pontos fortes da plataforma destacam-se:

- **Ambiente prático com *feedback* imediato** -– permite que o estudante execute exercícios e simulações, recebendo um retorno lógico sobre as respostas sem necessidade de recorrer imediatamente ao professor, facilitando a autoaprendizagem;
- **Personalização** — os *quizzes* e estudo por temas permitem adaptar o estudo ao nível de cada estudante, oferecendo flexibilidade pedagógica e possibilitando focar nos tópicos onde sente maior dificuldade;
- **Acessibilidade** — a aplicação é acessível dentro e fora do ISEP, permitindo ao estudante consultar conteúdos e executar exercícios em qualquer lugar com ligação à Internet, sem depender de *software* instalado localmente.

Por outro lado, algumas limitações foram identificadas na versão original:

- **Interface e experiência do utilizador** — algumas secções apresentavam navegação pouco intuitiva, dificultando a exploração eficiente dos conteúdos;
- **Flexibilidade e manutenção** — o código monolítico e pouco modular dificulta a implementação de novas funcionalidades e a adaptação a diferentes cenários de uso;
- **Dependências externas** – o simulador passo-a-passo integrado na plataforma depende de um editor de circuitos externo, que consiga fornecer uma *netlist* para análise, limitando a usabilidade e dificultando o *workflow* do utilizador;
- **Utilização da *netlist*** – o simulador passo-a-passo utiliza a *netlist* como única forma de entrada. Consequentemente, a aplicação não tem informação sobre a geometria do circuito, o que limita a capacidade de fornecer *feedback* visual ao utilizador, como por exemplo: desenho das malhas ou identificação dos nós sobre o esquema do circuito.

A organização e re-implementação da aplicação *web* U=RIsolve Academy surgiu como uma oportunidade estratégica de modernizar a plataforma, corrigindo as limitações identificadas na versão original. Embora os pontos fracos, como a dependência excessiva de processamento do lado do *frontend*, problemas de escalabilidade e dificuldade de manutenção, já justifiquem a atualização, a re-implementação permitiu ir além: criar uma base tecnológica mais robusta, modular e flexível, capaz de suportar novos requisitos pedagógicos e evoluções futuras.

Esta abordagem oferece ainda a possibilidade de repensar a arquitetura e o *design* da plataforma, incorporando tecnologias mais atuais, como *Node.js*[9] para

desenvolvimento do *backend*, servidor *MariaDB*[10] para gestão de base de dados, e *HandleBars*[11] como *template engine* no *frontend*. Para além disto, a utilização da arquitetura MVC garante performance, escalabilidade e manutenção simplificada.

Assim, a nova versão da ferramenta U=RIsolve Academy não só corrige os problemas anteriores, mas também estabelece um ambiente moderno e sustentável, pronto para futuras expansões, integração de novas funcionalidades e adaptação às necessidades dos estudantes de engenharia eletrotécnica no ISEP e fora dele.

2.2 Atores e Casos de uso

Esta plataforma servirá de ferramenta de ensino/autoaprendizagem de engenharia eletrotécnica, principalmente no meio da comunidade de estudantes do ISEP. Por este motivo, fez sentido dividir os atores da aplicação entre os seguintes, cada um com as suas funções e permissões, como indicado de seguida:

- Utilizador não registado (tipo 0):
    1. Ver publicações;
    2. Ver contactos;
    3. Mudar linguagem;
    4. Ver informação acerca do projeto;
    5. Criar conta;
- Utilizador do tipo **Aluno** (tipo 1):
    1. (Todas as funcionalidades do utilizador não registado);
    2. Entrar / Sair / Recuperar senha;
    3. Área de treino;
    4. Simular e analisar um circuito (via ficheiro ou DiCES);
    5. Criar/editar um circuito;
    6. Ver estatísticas da sua aprendizagem;
    7. Salvar circuitos na sua conta;
- Utilizador do tipo **Docente** (tipo 2):
    1. (Todas as funcionalidades do utilizador Aluno);
    2. Editor de publicações;
    3. Editor de Quizzes;
    4. Área de aprendizagem de todos os alunos (ou de uma das suas turmas);
- Utilizador do tipo **Administrador** (tipo 3):

1. (Todas as funcionalidades do utilizador Docente);
2. Gestão do tipo de utilizador atribuído a cada conta;
3. Gestão das funcionalidades permitidas a cada tipo de utilizador;

Deste modo fica ainda definido o diagrama de casos de uso, que sumariza visualmente a informação anterior:

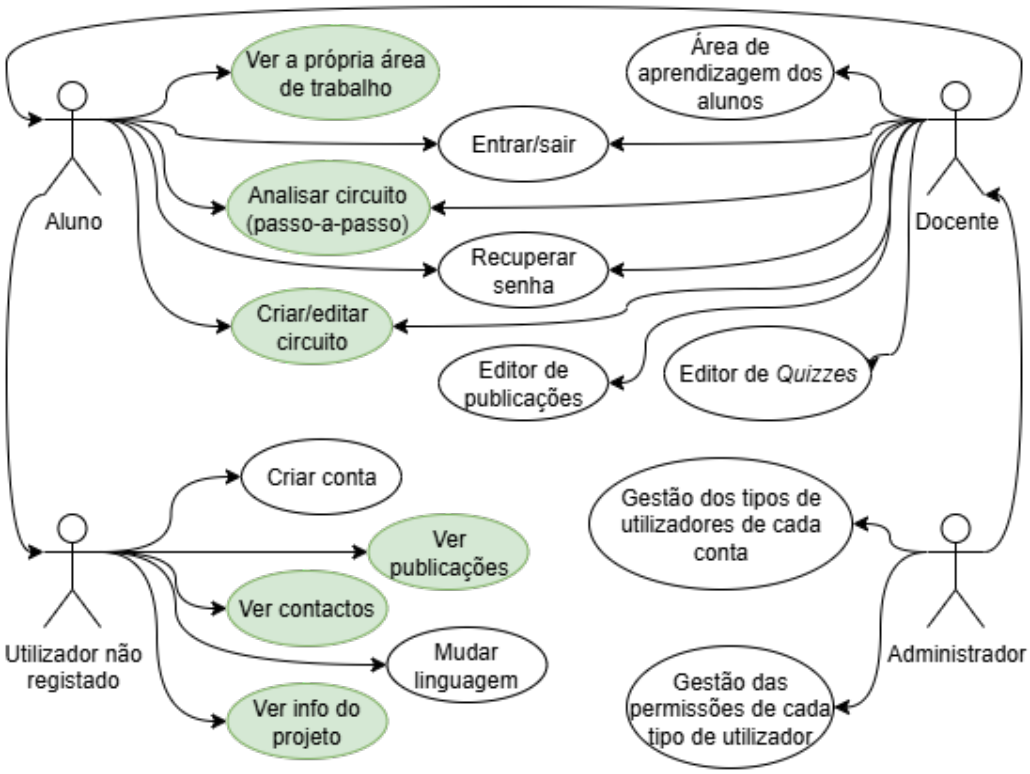


Figura 2.2: Diagrama de casos de uso

Apesar de a maioria destas funcionalidades não terem sido implementadas neste projeto, a sua definição prévia foi importante para compreender o contexto geral, e estruturar corretamente a aplicação U=RIsolve Academy. No diagrama da figura 2.2, é possível perceber quais os casos de uso que foram implementados neste projeto, assinalados com a cor verde [1].

2.3 Base de dados

Como indicado anteriormente, a base de dados foi desenvolvida utilizando a tecnologia *MariaDB* [10], por ser o que já está em utilização no servidor destino, e pela sua *performance*, adequada para o projeto.

[1] Uma vez que ainda não foi implementado nenhuma funcionalidade de , todas as funcionalidades implementadas estão acessíveis ao utilizador não registado.

Após a definição dos atores e casos de uso da aplicação, foi criada uma base de dados escalável, capaz de representar e guardar toda a informação necessária. Desde o tipo de utilizador aos circuitos que ele possa guardar, desde as publicações até aos Quizzes que certos tipos de utilizadores criam: tudo deve ter um lugar na base de dados.

Foi, então, definida uma possível estrutura que albergue toda a informação, moldada para funcionar corretamente com a tecnologia MariaDB. Dá-se o exemplo da criação de algumas estruturas de base de dados, nomeadamente:

Estrutura do utilizador

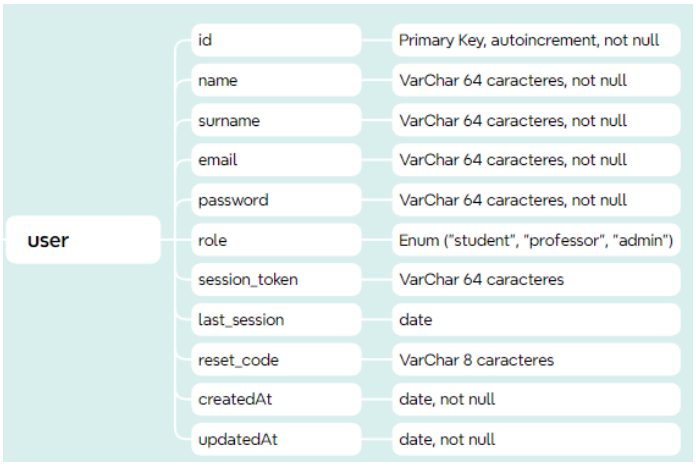


Figura 2.3: Possível estrutura de base de dados de um utilizador

Estrutura da questão

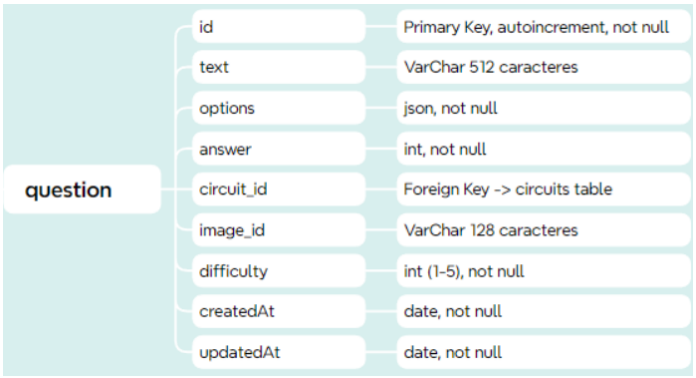


Figura 2.4: Possível estrutura de base de dados de uma questão

Estrutura do *Quiz*

Figura 2.5: Possível estrutura de base de dados de um quiz

Estrutura do circuito

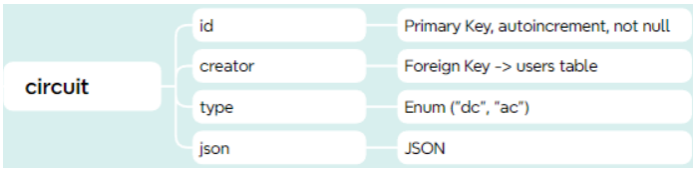


Figura 2.6: Possível estrutura de base de dados de um circuito elétrico

De notar que, a cada elemento de cada estrutura é automaticamente atribuído um identificador, *private key*, que pode servir como propriedade de ligação entre outros elementos de outras tabelas, sendo utilizado como *foreign key*. Dá-se o exemplo da estrutura do *Quiz*, que inclui um campo "questions" destinado a uma lista de *foreign key's*, identificadores das questões na estrutura das questões, *private key's*.

2.4 Planeamento da API

Tendo como planeada a estrutura da base de dados, foi desenvolvida uma API de comunicação entre o servidor e os clientes da aplicação. Este é um módulo de código executado no servidor e que serve como interface entre os clientes e a base de dados. Esta estrutura foi pensada de forma a que a transmissão de informação seja completa e permita realizar todas as operações que o projeto assim requer. De seguida, é apresentado um diagrama que ilustra todas estas rotas:

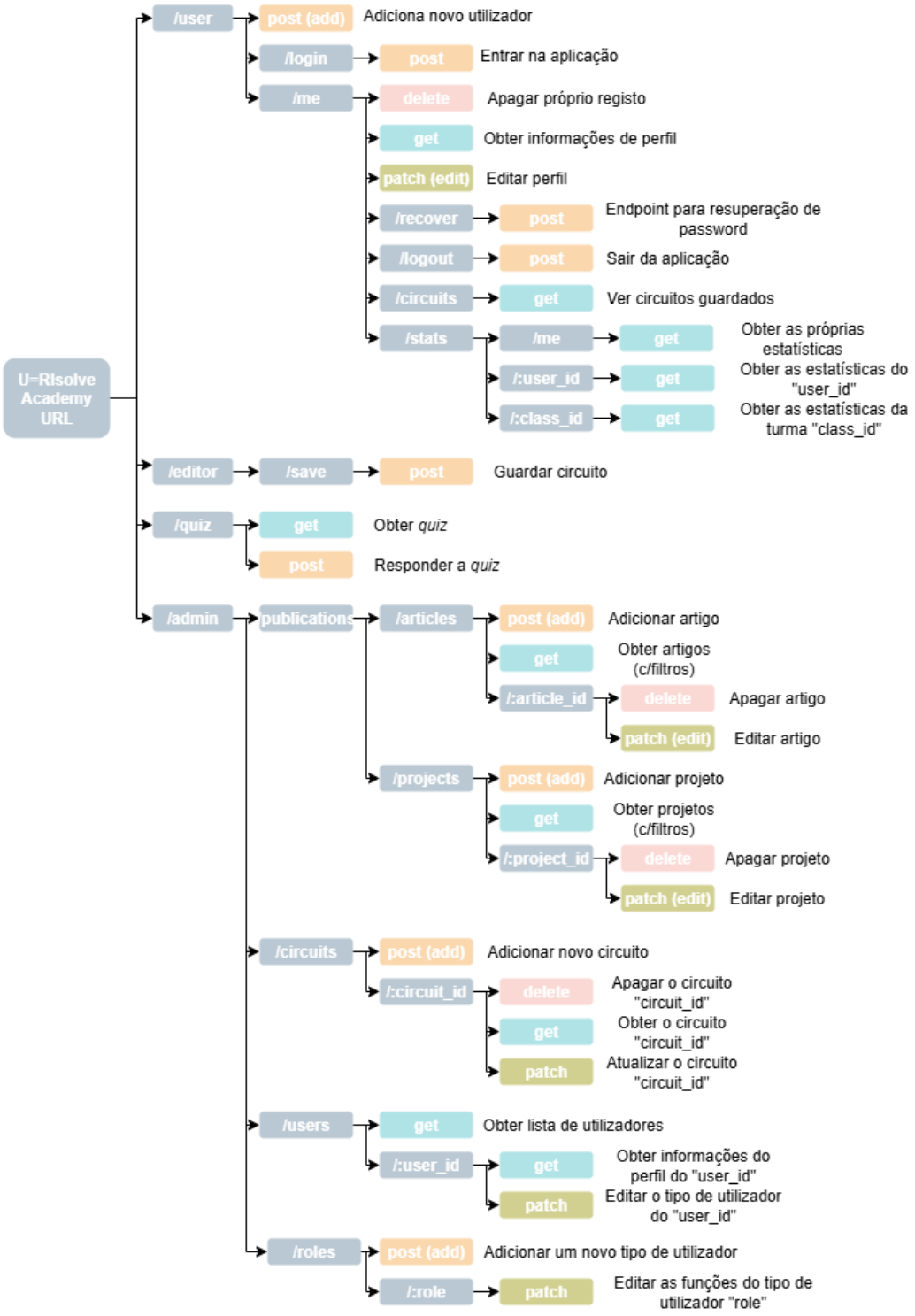


Figura 2.7: Diagrama de uma possível API

Cada uma destas rotas representa um ponto de acesso à API, que pode ser invocado por um cliente, para realizar uma operação específica. Estas operações podem ser de leitura, escrita, atualização ou eliminação de dados na base de dados.

2.5 Estruturação da aplicação

Por fim, foi necessário (re)estruturar todas as páginas principais da aplicação *web*.

Para isto foi adotado um padrão de estrutura de *software* conhecido por MVC [12], isto é, *Model-View-Controller*. Garantindo que a programação decorre de acordo com este padrão, o resultado é um módulo de *software* altamente escalável e de fácil manutenção para o futuro. Tal como o nome sugere, este padrão é composto por três principais componentes:

Model

O ***Model*** representa diretamente a estrutura da base de dados e é responsável por interagir com a mesma. Nesta camada, o conhecimento do mundo exterior é nulo, isto é, a camada não interage diretamente com o utilizador.

View

A ***View*** consiste na parte da aplicação que se encarrega de apresentar os dados ao utilizador e fazer a interação com o mesmo.

Controller

O ***Controller*** serve de elo de ligação entre as duas camadas anteriores. Numa API, o seu papel passa por perceber que pedidos foram feitos pelo utilizador e providenciar a resposta correta.

Um diagrama ilustrativo do padrão MVC é apresentado na Figura 2.8.

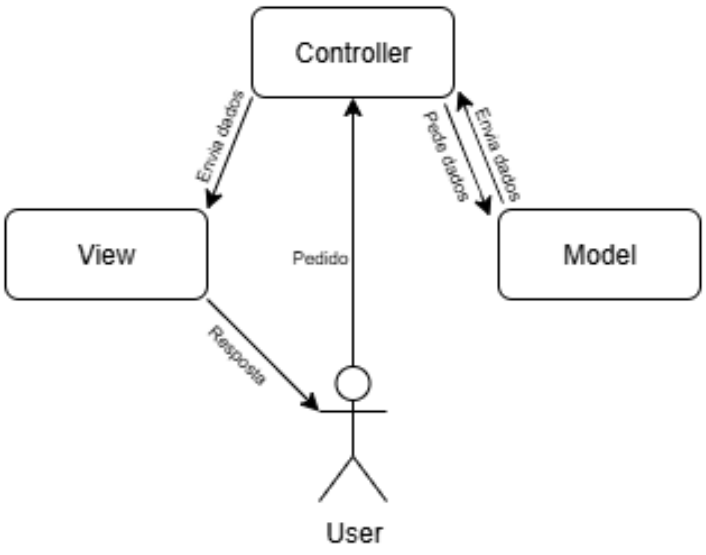


Figura 2.8: Diagrama MVC [12]

Neste trabalho foram desenvolvidas algumas rotas para efeitos de prova de conceito, que seguiram esta estrutura. Um exemplo prático, é a rota de gestão de

utilizadores, cuja implementação completa se encontra presente no anexo .1, nomeadamente o *Controller* e o *Model*

2.6 Novo *design*

Por último, foram desenvolvidas algumas páginas de teste de *design*, com o objetivo de se definir o aspeto final da aplicação de forma clara e consistente. Este passo foi fundamental, pois permitiu experimentar diferentes layouts de forma rápida, sem necessidade de programação, evitando alterações visuais significativas após o início do desenvolvimento.

Primeiramente, apresenta-se a versão anterior do U=RIsolve Academy, que serviu de base para o novo *design*. A Figura 2.9 ilustra o *layout* e a interface do portal original, destacando elementos estruturais, disposição de menus e organização de conteúdo que orientaram a nova conceção:

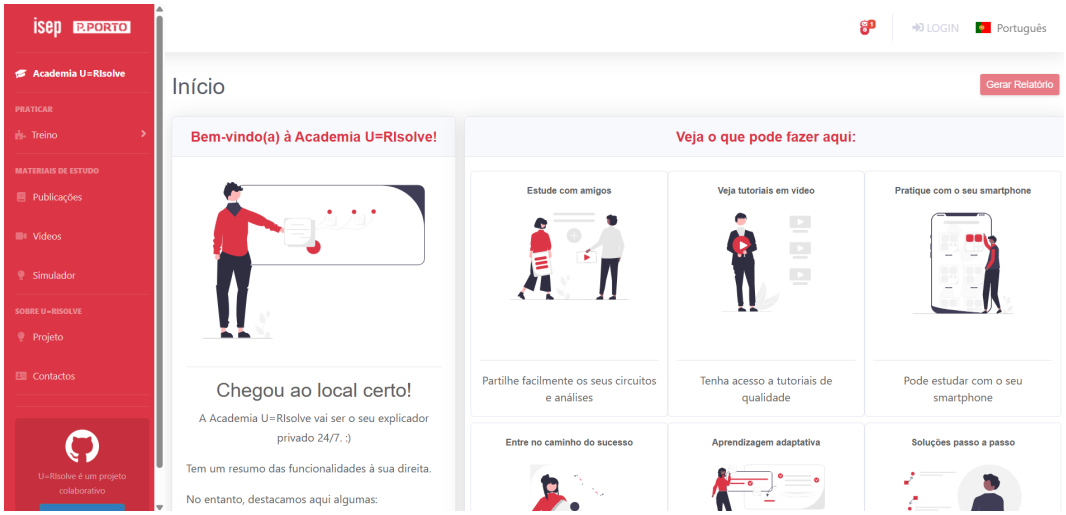


Figura 2.9: Página principal - *design* antigo

Posteriormente, a Figura 2.10 apresenta o novo *design* do U=RIsolve Academy, onde já se contempla o aspeto da página inicial da plataforma. As mudanças incluem uma disposição mais intuitiva dos menus, maior clareza visual nas seções de quizzes e analisador de circuitos, e consistência estética entre diferentes páginas, contribuindo para uma experiência de utilização mais fluida e agradável.

Figura 2.10: Página principal - novo *design*

Foi feito ainda o *(re)design* da página de *Quizzes*, que é uma das principais funcionalidades da plataforma. A Figura 2.11 apresenta o *design* antigo, que serviu de base para o novo *design*.

Figura 2.11: Página de *Quizzes* - *design* antigo

A Figura 2.12 apresenta o novo *design* da página de *Quizzes*. Nesta página destaca-se a mudança do menu lateral de escolha do tema para a própria página de treino, centralizando todo o conteúdo e facilitando a navegação.

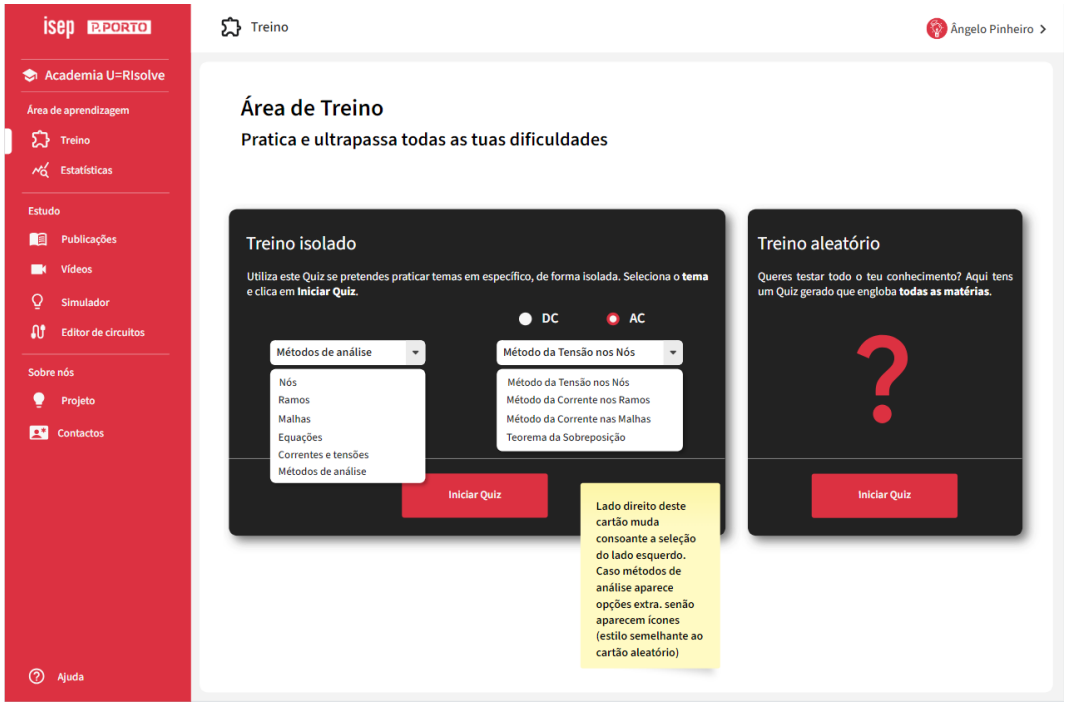


Figura 2.12: Página de *Quizzes* - novo *design*

Adicionalmente, a Figura 2.13 apresenta o novo *design* preliminar, onde já se contempla o aspeto inicial do editor e dos principais componentes da interface.

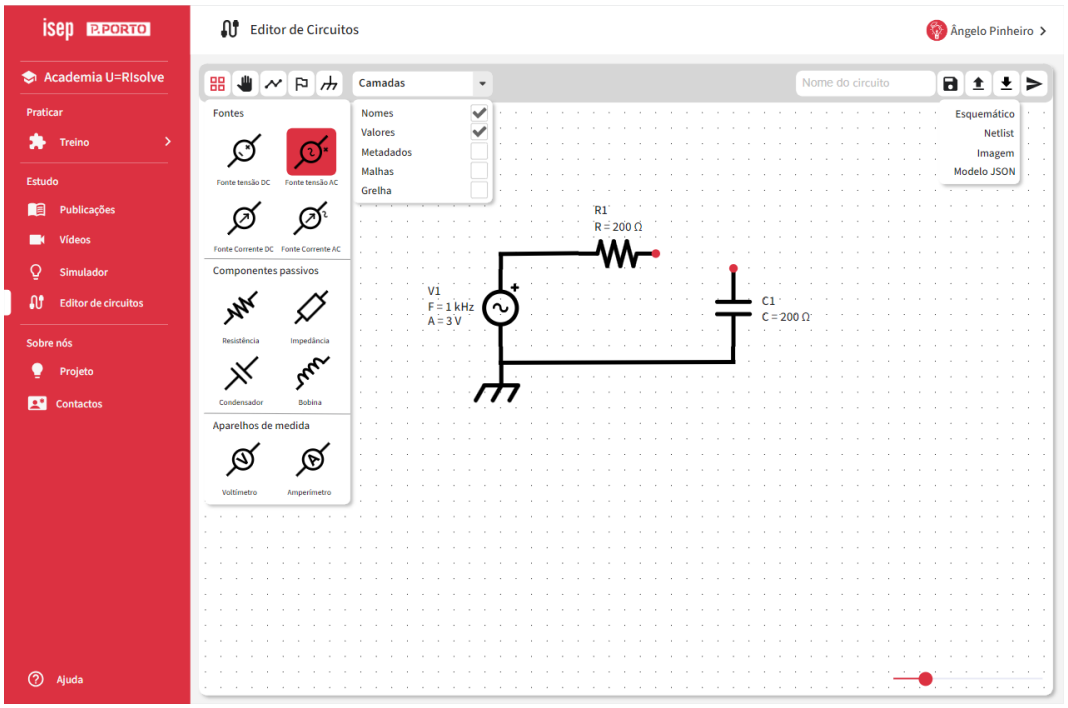


Figura 2.13: Página do editor - *design* desenvolvido

A prototipagem do *design* do editor de circuitos foi particularmente importante,

uma vez que este componente é central para a experiência do utilizador. Ao criar e testar protótipos do editor, foi possível identificar e resolver problemas de usabilidade antes da implementação, economizando tempo e recursos no desenvolvimento posterior.

Capítulo 3

Modelo de dados

Antes da era digital, a única forma de representar circuitos elétricos era através de esquemas desenhados manualmente, utilizando símbolos padronizados para cada componente. Esta é ainda uma prática comum em contextos educacionais e de engenharia, onde diagramas esquemáticos são desenhados para ilustrar circuitos e sistemas elétricos. O desenho à mão, quer seja em suporte papel (folha/caderno) ou digital (PC/tablet), é absolutamente crucial para desenvolver massa crítica [13] em termos de visualização mental dos circuitos elétricos (representação dos componentes, instrumentos de medição, fontes de alimentação, tensões e correntes, com os símbolos adequados, e da sua identificação e valor (se aplicável), ligações e pontos de interligação/nós, etc.) e de transposição entre esquemas e montagens (e vice-versa).

Com o advento da computação, tornou-se possível automatizar essa representação, mas isso depende de transformar os conceitos visuais e intuitivos em estruturas de dados formais, que os programas consigam manipular.

Para atingir o objetivo de desenvolver um editor de circuitos, o primeiro passo, e o mais fundamental, é justamente definir um modelo de dados capaz de representar um circuito de forma completa e consistente. Este modelo precisa incluir não apenas os elementos individuais, resistências, fontes, condensadores, etc, mas também as suas posições, as ligações entre eles, a topologia do circuito e todas as propriedades elétricas relevantes.

Desenvolver este modelo é essencial, pois todas as funcionalidades do editor, desde desenhar componentes no ecrã até simular o comportamento do circuito, dependem da existência de uma representação digital estruturada. Com um modelo

de dados bem definido, o computador consegue interpretar, validar e manipular os circuitos, permitindo não só a edição visual, mas também cálculos automáticos e integração com outras ferramentas de análise.

3.1 Contexto tecnológico

A representação de circuitos elétricos é um passo fundamental tanto para a análise manual quanto para a manipulação computacional. Ao longo dos anos, diferentes modelos têm sido desenvolvidos, cada um com propósitos distintos e níveis variados de detalhamento. Entre os mais utilizados destacam-se a **netlist** e o **Schematic (SCH) do Qucs** [14].

A ***netlist*** é uma representação textual do circuito que lista todos os componentes e as suas conexões. Cada linha descreve um componente, os nós aos quais está ligado e os seus parâmetros elétricos. Este modelo é amplamente usado em simulação automática, sendo a base de softwares como os da popular família "SPICE", e.g. PSpice [15] ou Ngspice [16]. No contexto deste trabalho, a ***netlist*** é particularmente relevante, pois é uma representação do circuito restrita ao que é importante para a análise de circuitos: um conjunto de componentes e a informação de interligação entre eles. Assim, permite-se que algoritmos interpretem o circuito, executem cálculos automáticos e simulem o seu comportamento elétrico sem necessidade da sua representação gráfica. No entanto, uma *netlist* não contém a informação da geometria do circuito.

Dá-se o exemplo da seguinte *netlist*, que descreve um circuito simples com resistências, fontes de tensão e corrente, e um controlador de simulação DC, referente ao circuito da Figura 3.1:

```
1
2  # Qucs 0.0.19 C:/exemplo.sch
3
4  R:R1 gnd C R="9 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
5  Vdc:V1 A _net0 U="3 V"
6  R:R2 gnd _net0 R="3 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
7  R:R3 gnd A R="1 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
8  R:R6 _net2 A R="5 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
9  Vdc:V2 _net2 B U="15 V"
10 R:R4 _net3 A R="7 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
11 R:R8 A B R="5 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
12 R:R5 _net1 gnd R="1 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
13 Vdc:V3 B _net1 U="6 V"
14 R:R7 B _net4 R="2 Ohm" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
15 Vdc:V4 gnd _net4 U="8 V"
```

```
16 Idc:I1 C A I="2 A"
17 Idc:I2 _net3 B I="4 A"
18 .DC:DC1 Temp="26.85" reltol="0.001" abstol="1 pA" vntol="1 uV"
       saveOPs="no" MaxIter="150" saveAll="no" convHelper="none"
       Solver="CroutLU"
```

Listagem 3.1: Exemplo de uma netlist

De forma análoga, o **Schematic (SCH) do QUCS** também descreve o circuito, mas desta vez incluindo a informação geométrica e visual do mesmo, incluindo não só a informação dos componentes existentes e da sua posição, mas também do percurso dos fios de ligação presentes. Apresenta-se o exemplo de um ficheiro SCH do Qucs, que descreve o mesmo circuito que a netlist anterior, mas com a informação gráfica adicional:

```
1  <Qucs Schematic 0.0.19>
2  <Properties>
3    <View=0,-60,1078,800,1,0,0>
4    <Grid=10,10,1>
5    <DataSet=exemplo.dat>
6    <DataDisplay=exemplo.dpl>
7    <OpenDisplay=1>
8    <Script=exemplo.m>
9    <RunScript=0>
10   <showFrame=0>
11   <FrameText0=Titulo>
12   <FrameText1=Autor:>
13   <FrameText2=Data:>
14   <FrameText3=Revisao:>
15 </Properties>
16 <Symbol>
17 </Symbol>
18 <Components>
19   <R R1 1 310 390 15 -26 0 1 "9 Ohm" 1 "26.85" 0 "0.0" 0 "0.0" 0
         "26.85" 0 "US" 0>
20   <Vdc V1 1 480 270 18 -26 0 1 "3 V" 1>
21   <R R2 1 480 350 15 -26 0 1 "3 Ohm" 1 "26.85" 0 "0.0" 0 "0.0" 0
         "26.85" 0 "US" 0>
22   <R R3 1 630 330 15 -26 0 1 "1 Ohm" 1 "26.85" 0 "0.0" 0 "0.0" 0
         "26.85" 0 "US" 0>
23   <R R6 1 810 80 -26 -53 0 2 "5 Ohm" 1 "26.85" 0 "0.0" 0 "0.0" 0
         "26.85" 0 "US" 0>
24   <Vdc V2 1 930 80 -26 -56 0 2 "15 V" 1>
```

```
25   <R R4 1 820 200 -26 -53 0 2 "7 Ohm" 1 "26.85" 0 "0.0" 0 "0.0" 0
         "26.85" 0 "US" 0>
26   <R R8 1 840 240 -26 15 0 0 "5 Ohm" 1 "26.85" 0 "0.0" 0 "0.0" 0
         "26.85" 0 "US" 0>
27   <Idc I1 1 310 300 -55 -26 0 3 "2 A" 1>
28   <Idc I2 1 930 200 -26 -56 0 2 "4 A" 1>
29   <Vdc V3 1 930 430 -26 18 0 0 "6 V" 1>
30   <GND * 1 630 550 0 0 0 0>
31   <R R7 1 960 550 -26 -53 0 2 "2 Ohm" 1 "26.85" 0 "0.0" 0 "0.0" 0
         "26.85" 0 "US" 0>
32   <Vdc V4 1 810 550 -26 -56 0 2 "8 V" 1>
33   <R R5 1 790 430 -26 -53 0 2 "1 Ohm" 1 "26.85" 0 "0.0" 0 "0.0" 0
         "26.85" 0 "US" 0>
34   <.DC DC1 1 340 620 0 39 0 0 "26.85" 0 "0.001" 0 "1 pA" 0 "1 uV" 0
         "no" 0 "150" 0 "no" 0 "none" 0 "CroutLU" 0>
35 </Components>
36 <Wires>
37   <480 300 480 320 "" 0 0 0 "">
38   <480 240 630 240 "" 0 0 0 "">
39   <630 240 630 300 "" 0 0 0 "">
40   <630 360 630 430 "" 0 0 0 "">
41   <480 430 630 430 "" 0 0 0 "">
42   <480 380 480 430 "" 0 0 0 "">
43   <630 200 630 240 "" 0 0 0 "">
44   <630 240 810 240 "" 0 0 0 "">
45   <1020 240 1020 430 "" 0 0 0 "">
46   <870 240 1020 240 "" 0 0 0 "">
47   <630 200 790 200 "" 0 0 0 "">
48   <1020 200 1020 240 "" 0 0 0 "">
49   <960 200 1020 200 "" 0 0 0 "">
50   <850 200 900 200 "" 0 0 0 "">
51   <630 80 630 200 "" 0 0 0 "">
52   <630 80 780 80 "" 0 0 0 "">
53   <840 80 900 80 "" 0 0 0 "">
54   <960 80 1020 80 "" 0 0 0 "">
55   <1020 80 1020 200 "" 0 0 0 "">
56   <310 200 630 200 "" 0 0 0 "">
57   <310 200 310 270 "" 0 0 0 "">
58   <310 330 310 360 "" 0 0 0 "">
59   <960 430 1020 430 "" 0 0 0 "">
60   <1020 430 1020 550 "" 0 0 0 "">
61   <310 420 310 550 "" 0 0 0 "">
62   <630 430 630 550 "" 0 0 0 "">
63   <310 550 630 550 "" 0 0 0 "">
```

```
64    <990 550 1020 550 "" 0 0 0 "">
65    <840 550 930 550 "" 0 0 0 "">
66    <630 550 780 550 "" 0 0 0 "">
67    <630 430 760 430 "" 0 0 0 "">
68    <820 430 900 430 "" 0 0 0 "">
69    <630 200 630 200 "A" 660 170 0 "">
70    <1020 240 1020 240 "B" 1060 210 0 "">
71  </Wires>
72  <Diagrams>
73  </Diagrams>
74  <Paintings>
75  </Paintings>
```

Listagem 3.2: Exemplo de um ficheiro esquemático

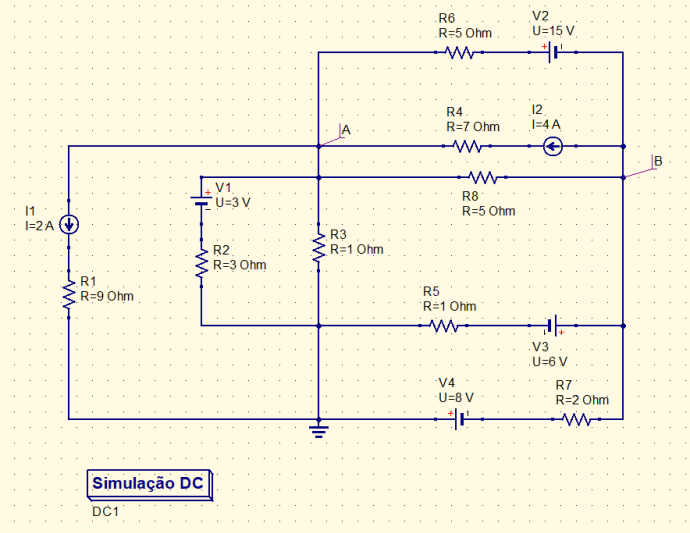


Figura 3.1: Circuito descrito nas representações netlist e SCH do QUCS anteriores

Além destes, existem outros modelos de representação de circuitos, como diagramas de nós ou topológicos, que representam o circuito como um grafo de nós e ramos, focando na conectividade entre componentes; modelos em JSON ou XML, que armazenam informações sobre componentes, ligações, propriedades elétricas e posições na interface gráfica; formatos proprietários CAD/E-CAD, de softwares como Eagle [17], KiCad [18] ou Altium [19], que contêm tanto o esquema gráfico quanto informações de layout; e representações matemáticas ou baseadas em grafos, que utilizam matrizes de incidência e de adjacência para descrever a topologia do circuito, principalmente para algoritmos de análise e verificação. No analisador de circuitos passo-a-passo, já são utilizados estes conceitos de teoria de grafos para obter as malhas que constituem um determinado circuito [20] [21].

No âmbito deste projeto, a atenção principal recai sobre o **SCH do QUCS** e a **netlist**, por serem os modelos mais diretamente aplicáveis à criação de um editor de circuitos. O *SCH* fornece a base para a interface visual do editor, enquanto a netlist permite a análise elétrica e integração com simulações.

No entanto, a criação de um editor de circuitos a ser executado no *browser* exige uma representação mais estruturada e eficiente do que um simples ficheiro de texto. Formatos tradicionais, como *netlist* ou ficheiros *schematic*, foram concebidos para simulação e armazenamento estático, não para manipulação dinâmica em ambiente *web*. A tecnologia JSON apresenta-se como a escolha mais indicada para este propósito, pois é nativamente orientada a objetos e perfeitamente integrada com linguagens utilizadas no *frontend*, como *JavaScript*. Além disso, a sua estrutura hierárquica permite representar relações complexas, como componentes, nós e ramos, de forma clara e facilmente manipulável pelo código.

Partindo do modelo JSON de base [1], capaz de representar corretamente um circuito elétrico, procedeu-se a uma revisão e extensão que permitiu acomodar todos os requisitos funcionais do editor de circuitos desenvolvido. Este modelo revisto é detalhado na Secção 3.3.

3.2 Parametrização do modelo

Como já foi mencionado, o modelo de dados deve conseguir representar qualquer circuito. Para este efeito, é necessário compreender exatamente o que é um circuito e os elementos que o constituem, tanto a nível conceptual, como gráfico. Podemos isolar três tipos de constituintes distintos:

- **Componentes**: nesta categoria incluem-se não só resistências, condensadores e bobinas, mas também fontes de tensão, de corrente, voltímetros, amperímetros, etc...
- **Fios de ligação**: esta categoria consiste na descrição de elementos de ligação entre componentes.
- **Pontos de conexão**: consistem em pontos do circuito em que três ou mais fios de ligação ou componentes se unem.

Numa segunda fase, podemos definir outros dois tipos de entidades elétricas, que podemos considerar secundárias, já que são construídas com recurso às anteriores.

[1] O modelo-base foi especificado teoricamente [22] por André Rocha, coorientador deste trabalho, no âmbito do seu doutoramento. A especificação em JSON utilizada neste relatório constitui uma versão revista e expandida, derivada do conhecimento empírico da implementação prática realizada neste trabalho.

- **Ramos**: eletricamente definimos um ramo como um conjunto de componentes e fios de ligação, ligados em série, e com início e fim em dois nós distintos. A definição no nosso modelo não é diferente, dado que teremos um conjunto de componentes e fios de ligação para a definição de ramo;

- **Nós**: eletricamente, definimos um nó como a interceção de três ou mais ramos. No nosso modelo definimos como nó um conjunto de Fios, Pontos de conexão e Terminais de componentes diretamente interligados, definindo uma zona equipotencial. [2]

Uma vez que estas estruturas são consideradas secundárias, dado que são construídas com recurso a estruturas mais simples, podemos albergá-las numa única propriedade do futuro modelo de dados.

3.3 Modelo representativo de um circuito elétrico

Dados os elementos identificados anteriormente, podemos definir a estrutura de dados completa, fazendo também incluir parâmetros que informam sobre posições, cores, geometria, grandezas elétricas, etc...

Componente (ou *Component*)

Apresenta-se de seguida a estrutura do modelo de dados para representar o componente:

[2] Um nó elétrico pode conter vários Pontos de Conexão

Nome da propriedade			Descrição	Tipo
id			Identificador único do componente	String
type			Tipo do componente	String
position	**x**		Posição do centro do componente (em unidades de grelha)	Número
	y			Número
	z		Camada do componente em relação a outros componentes	Número
	rotation		Rotação do componente (em graus)	Número
terminals (vetor)	**terminalId**		Identificador único do terminal do componente	String
	position	**xOffset**	Posição relativa do terminal ao centro do componente (em unidades de grelha)	Número
		xOffset		Número
		yOffset		Número
	terminalRole		Papel do terminal no componente (ex: positivo, negativo, undefined)	String
	wire	**id**	Identificador do fio ligado	String
		terminalId	Identificador do terminal do fio conectado	String
	connectionPoint	**id**	Identificador do ponto de interligação conectado	String
	component	**id**	Identificador do componente conectado	String
		terminalId	Identificador do terminal do componente conectado	String
label	**value**		Valor da label (texto)	String
	position	**xOffset**	Posição relativa da label ao centro do componente (em píxel)	Número
		yOffset		Número
		rotation	Rotação da label (em graus)	Número
	visible		Visibilidade da label	Booleano
attributes (vetor)	**name**		Nome da grandeza física / propriedade	String
	value		Valor numérico da grandeza / propriedade	String
	unit		Unidade da grandeza física / propriedade	String
	type		Tipo da propriedade (main ou sub)	String
	visible		Visibilidade da propriedade na label	Boolean
visible			Visibilidade global do componente	Boolean

Tabela 3.1: Tabela da estrutura de dados - Componente

Com este modelo será possível representar qualquer tipo de componente elétrico, bastando especificar o seu tipo, terminais, e os atributos que o definem.

Fio de ligação (ou *Wire*)

Apresenta-se de seguida a estrutura do modelo de dados para representar o fio de ligação:

Nome da propriedade				Descrição	Tipo
id				Identificador único do fio de ligação	String
properties	**stroke**			Espessura do fio de ligação (em píxel)	Número
	color			Cor do fio de ligação (código HEX)	String
	decoration	**startPoint**	**type**	Tipo do elemento decorativo (seta, ponto, ...)	String
			color	Cor do elemento decorativo (código HEX)	String
		endPoint	**type**	Tipo do elemento decorativo (seta, ponto, ...)	String
			color	Cor do elemento decorativo (código HEX)	String
startTerminal	**terminalId**			Identificador único do terminal do fio de ligação	String
	position	**x**		Posição absoluta do terminal do fio de ligação	Number
		y		(em unidades de grelha)	Number
	component	**id**		Identificador único do componente conectado	String
		terminalId		Identificador único do terminal do componente conectado	String
	connectionPoint	**id**		Identificador único do ponto de conexão conectado	String
endTerminal	**terminalId**			Identificador único do terminal do fio de ligação	String
	position	**x**		Posição absoluta do terminal do fio de ligação	Number
		y		(em unidades de grelha)	Number
	component	**id**		Identificador único do componente conectado	String
		terminalId		Identificador único do terminal do componente conectado	String
	connectionPoint	**id**		Identificador único do ponto de conexão conectado	String
vertices	**x**			Pontos de quebra no fio de ligação (em unidades de grelha)	Number
	y				Number
attributes (vetor)	**name**			Nome da grandeza física / propriedade	String
	value			Valor numérico da grandeza / propriedade	String
	unit			Unidade da grandeza física / propriedade	String
	type			Tipo da propriedade (main ou sub)	String
	visible			Visibilidade da propriedade na label	Boolean

Tabela 3.2: Tabela da estrutura de dados - Fio de ligação

Ponto de conexão (ou *Connection Point*)

Apresenta-se de seguida a estrutura do modelo de dados para representar o ponto de conexão:

Nome da propriedade		Descrição	Tipo
id		Identificador único do ponto de interligação	String
properties	**stroke**	Diâmetro do ponto de interligação	String
	color	Cor do ponto de interligação	String
position	**x**	Posição absoluta do ponto de interligação	Número
	y	(em unidades de grelha)	Número
wires [ ]	**id**	Identificador único do fio de ligação conectado	String
	terminalId	Identificador único do terminal do fio de ligação conectado	String
components[ ]	**id**	Identificador único do componente conectado	String
	terminalId	Identificador único do terminal do componente conectado	String
label	**value**	Valor do nome do ponto de interligação	String

Tabela 3.3: Tabela da estrutura de dados - Ponto de conexão

Apresenta-se de seguida a estrutura do modelo de dados para representar os metadados do circuito:

Circuito

Por fim, é possível definir uma estrutura de dados do circuito, que faz uso das estruturas anteriores. Note-se também o campo *metadata*, onde são definidas as estruturas secundárias descritas na secção anterior, bem como outras informações relevantes acerca do circuito, tais como as tensões nodais e correntes nos ramos.

Nome da propriedade	Descrição	Tipo
editorProperties	Propriedades do editor e janela visível	Objeto
components	Lista de componentes existentes no circuito	Vetor
wires	Lista de fios de ligação existentes no circuito	Vetor
connectionPoints	Lista de pontos de interligação existentes no circuito	Vetor
metadata	Objeto com informação complementar do circuito	Objeto

Tabela 3.4: Tabela da estrutura geral do modelo JSON

3.4 Exemplo de um circuito simples

De forma a melhor ilustrar a materialização do modelo de dados, segue um exemplo de um circuito simples, representado na Figura 3.2.

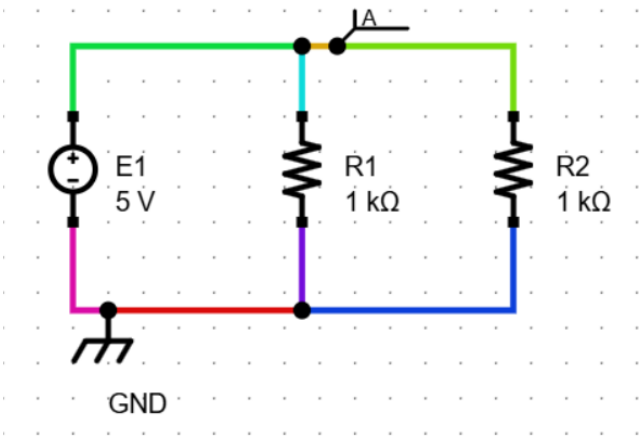


Figura 3.2: Exemplo de circuito simples

O circuito apresentado é composto por cinco tipos de elementos: Fonte de tensão E1, Resistências R1 e R2, Massa GND e Etiqueta A. Apresenta ainda 8 fios de ligação (a cores diferentes) e 4 pontos de conexão (círculos pretos). Em termos de elementos secundários, o circuito é composto por 3 ramos e 2 nós.

Exemplo do componente E1

Segue o modelo de dados relativo à fonte de tensão E1, do circuito:

```
{
    "id": "f384b4c919a070fa747",
    "position": {
    "rotation": 270,
    "x": -25,
    "y": 6.5,
    "z": 0
    },
```

```
    "type": "VDC",
    "terminals": [
    {
        "terminalId": "73d12bc719a070fa747",
        "position": {
        "xOffset": 0,
        "yOffset": 1.5
        },
        "terminalRole": "negative",
        "wire": {
        "id": "2deae93e19a07103b57",
        "terminalId": "77714d2719a07103b57"
        }
    },
    {
        "terminalId": "fee1e25419a070fa747",
        "position": {
        "xOffset": 0,
        "yOffset": -1.5
        },
        "terminalRole": "positive",
        "wire": {
        "id": "af68eaf519a071012c5",
        "terminalId": "a1045b2a19a071012c5"
        }
    }
    ],
    "attributes": [
    {
        "name": "name",
        "value": "E1",
        "visible": true,
        "type": "main"
    },
    {
        "name": "voltage",
        "value": 5,
        "unit": "V",
        "visible": true,
        "type": "main"
    },
    {
        "name": "ideal",
        "value": true,
```

```
53          "visible": false,
54          "type": "sub"
55      },
56      {
57          "name": "internal_resistance",
58          "value": 0,
59          "unit": "ohm",
60          "visible": false,
61          "type": "sub"
62      }
63      ],
64      "label": {
65      "position": {
66          "xOffset": 31.200000000000006,
67          "yOffset": 0,
68          "rotation": 0
69      },
70      "origin": {
71          "xOrigin": -650.0000000000002,
72          "yOrigin": 169.00000000000006
73      }
74      }
75  }
```


Listagem 3.3: Modelo de dados da fonte de tensão E1

É possível observar que o componente E1 possui diversas propriedades, tais como a sua posição no circuito (posição e rotação), o seu tipo (VDC), os seus terminais (com as respetivas posições, papéis e fios ligados), os seus atributos (nome, tensão, resistência interna, etc...) e a sua label (com a posição relativa ao componente e a origem absoluta no circuito). Note-se que cada atributo tem ainda o seu próprio nome, valor, unidade, visibilidade (no desenho do circuito), e tipo (*main* ou *sub*, referente a ser uma propriedade principal ou adicional).

Exemplo do fio de ligação

Segue o modelo de dados relativo ao fio de ligação, assinalado a azul no esquema elétrico do circuito.


```
1  {
2      "id": "dff7a9e91995c823844",
3      "properties": {
4      "stroke": 3.5,
5      "dashed": false,
6      "color": "#3207f1ff",
```

```
7       "decoration": {}
8       },
9       "attributes": [
10      {
11          "name": "internal_resistance",
12          "value": 0,
13          "unit": "ohm",
14          "visible": false
15      },
16      {
17          "name": "temperature",
18          "value": 20,
19          "unit": "C",
20          "visible": false
21      },
22      {
23          "name": "material",
24          "value": "default",
25          "visible": false
26      },
27      {
28          "name": "length",
29          "value": 0,
30          "unit": "m",
31          "visible": false
32      },
33      {
34          "name": "cross_section",
35          "value": 0,
36          "unit": "mm2",
37          "visible": false
38      },
39      {
40          "name": "resistivity",
41          "value": 0,
42          "unit": "ohmmm2/m",
43          "visible": false
44      },
45      {
46          "name": "temperature_coefficient",
47          "value": 0,
48          "unit": "ohm/C",
49          "visible": false
50      }
```

```
51      ],
52      "startTerminal": {
53      "terminalId": "97bed4381995c823844",
54      "position": {
55          "x": 4,
56          "y": -1
57      },
58      "component": {
59          "id": "0f210aca1995c820594",
60          "terminalId": "35a4999b1995c820594"
61      }
62      },
63      "endTerminal": {
64      "terminalId": "2e8e693b1995c823844",
65      "position": {
66          "x": -1.5,
67          "y": 1.5
68      },
69      "connectionPoint": "064c9e221995c8250c7"
70      },
71      "vertices": [
72      {
73          "x": 4,
74          "y": 1.5
75      }
76      ]
77  }
```

Listagem 3.4: Modelo de dados do fio de ligação

É possível observar que o fio de ligação possui diversas propriedades, tais como as suas propriedades visuais (cor, espessura, estilo), os seus atributos elétricos (resistência interna, temperatura, material, etc...), o terminal inicial (com a sua posição e o componente ligado) e o terminal final (com a sua posição e o ponto de conexão ligado). Apresenta ainda um vetor de vértices que define o seu percurso geométrico no circuito.

Exemplo do ponto de conexão

Segue o modelo de dados relativo ao ponto de conexão, ao qual está conectada a massa do circuito.

```
1  {
2      "id": "37f96a781995c8250c7",
3      "properties": {
```

```
    "color": "black",
    "stroke": 5
    },
    "position": {
    "x": -7.5,
    "y": 1.5
    },
    "wires": [
    {
        "id": "b2efa9e21995c824750",
        "terminalId": "61c42aef1995c824750"
    },
    {
        "id": "571ba7e11995c824750",
        "terminalId": "4e35e1d51995c824750"
    }
    ],
    "components": [
    {
        "id": "6d30b96f1995c824111",
        "terminalId": "9102b9ee1995c824111"
    }
    ]
}
```

Listagem 3.5: Modelo de dados do ponto de conexão

É possível observar que o ponto de conexão possui diversas propriedades, não só visuais (cor, espessura, estilo), mas também geométricos (posição) e elétricos (informação do que está ligado ao ponto: terminais de componentes, e de fios de ligação).

Exemplo de um ramo

Segue o modelo de dados relativo ao ramo em que está inserida a resistência R2, do circuito.

```
{
    "id": "f1ceb0f81995c825a5b",
    "components": [
        "0f210aca1995c820594"
    ],
    "wires": [
        "dff7a9e91995c823844",
        "553e4aa11995c8250c7"
```

```
    ],
    "path": [
        {
        "x": -1.5,
        "y": 1.5
        },
        {
        "x": 4,
        "y": 1.5
        },
        {
        "x": 4,
        "y": -6
        },
        {
        "x": -0.5,
        "y": -6
        }
    ]
}
```

Listagem 3.6: Modelo de dados de um ramo do circuito

Exemplo nó

Segue o modelo de dados relativo ao nó superior do circuito.

```
{
    "id": "9a00d5621995c825a5b",
    "wires": [
        "c84dfb1f1995c82249f",
        "b0243b671995c82249f",
        "0bcd18541995c8250c7",
        "553e4aa11995c8250c7"
    ],
    "connectionPoints": [
        "9e77c2b61995c8250c7",
        "ffab0b751995c8250c7"
    ],
    "components": [
        {
        "id": "1daf71441995c81ec06",
        "terminalId": "2dc8237d1995c81ec06"
        },
        {
```

```
19          "id": "5052da0f1995c820146",
20          "terminalId": "772b92431995c820146"
21          },
22          {
23          "id": "0f210aca1995c820594",
24          "terminalId": "fdf27c431995c820594"
25          },
26          {
27          "id": "77c8752d1995c8250c5",
28          "terminalId": "a82fc2591995c8250c5"
29          }
30      ],
31      "net": "A",
32      "type": "real"
33  }
```

Listagem 3.7: modelo de dados de um nó do circuito

Tanto o modelo de dados do ramo, como do nó conistem em conjuntos dos elementos mais simples, descritos anteriormente.

A totalidade do modelo de dados relativo a este circuito exemplo encontra-se descrito no anexo .2

Capítulo 4

Editor de circuitos

Neste capítulo é abordado todo o projeto de planeamento, desenvolvimento e implementação do novo editor/simulador de circuitos, que demominamos de Didactic Circuit Editor and Simulator (DiCES), da plataforma U=RIsolve. A apresentação é feita de forma simplificada, focando-se nos aspetos mais relevantes do projeto, e não entrando em pormenores exageradamente técnicos que não sejam essenciais para a compreensão do mesmo.

4.1 Contexto tecnológico

4.1.1 Estudo comparativo de editores/simuladores de circuitos

Um dos objetivos deste projeto foi o desenvolvimento de um novo editor/simulador de circuitos, o que implica perceber o que já existe para conseguir produzir algo de útil e mais adequado aos nossos propósitos: o ensino/autoaprendizagem dos fundamentos da análise de circuitos elétricos em CC e CA.

Assim foi realizado um estudo comparativo entre diversas ferramentas de edição gráfica e simulação de circuitos elétricos populares na comunidade, nomeadamente os sequintes:

- QUCS [23]
- Falstad [24]
- EveryCircuit [25]

- EasyEDA [26]
- CircuitMaker [27]
- LTSpice [28]
- Editor antigo da plataforma U=RIsolve Academy [29]
- Diagrams.net [30]
- Visio [31]

Cada ferramenta foi analisada quanto às seguintes funcionalidades e características, tanto de User interface, como de User experience:

- Adição de componentes (ou blocos)
- Adição/desenho de conexões
- Comportamento dos componentes/blocos (quando são arrastados, rodados, etc...)
- Comportamento das conexões (quando são arrastadas, rodados, etc...)
- Criação automática de nós
- Atalhos (de utilização)
- Multi-seleção (seleção de múltiplos componentes/ligações)
- Legendas (dos componentes)
- Desgin gráfico geral

As funcionalidades de "simulador" não foram consideradas nesta análise comparativa, porque as características pretendidas para o novo editor/simulador de circuitos são muito específicas para o contexto de ensino/autoaprendizagem dos métodos de análise de circuitos, para quem está a aprender do zero.

Este estudo foi feito assumindo um espírito crítico e imparcial, fruto de experiência pessoal com diversos editores e do senso comum proveniente da utilização constante de tecnologias digitais.

Posteriormente, foram definidas as características do novo editor/simulador, com base nesta análise. Para já, apresentam-se os mesmos de forma condensada na Tabela 4.1, onde cada ponto anterior é classificado com uma pontuação de 1 a 3 estrelas:

	Qucs	Falstad	Every Circuit	Easy EDA	Circuit Maker	LTSpice	Editor antigo	Diagrams .net	Visio
Adição de componentes	★★★	★	★★	★★	★★★	★	★★	★★★	★★★
Adição de conexões	★★★	★	★★	★★	★★★	★	★	★★★	★★★
Comportamento dos componentes	★★★	★	★	★★	★★★	★	★★	★★★	★★★
Comportamento das conexões	★★★	★	★★	★★	★★★	★	★	★★★	★★★
Criação automática de nós	★★★	-	-	★★★	★★★	★	★	-	-
Multi-seleção	★★★	★★★	★★★	★★★	★★★	★★★	★★★	★★★	★★★
Legendas	★★★	★	-	★★★	★★	★★	★★★	★★★	★★
Aspeto gráfico geral	★	★★	★★	★★★	★★★	★	★★	★★★	★★★
Global	★★★	★	★★	★★	★★★	★	★★	★★★	★★★

Figura 4.1: Tabela comparativa entre editores gráficos

Embora a tabela apresente uma vista geral e resumida, é possível perceber quais editores se destacam em cada aspeto, e, por sua vez, quais devemos valorizar mais no projeto do novo editor. Para além disso, a versão completa da tabela encontra-se presente no Anexo .4.

4.1.2 Enquadramento do editor/simulador DiCES

Como inserir um editor de circuitos numa *framework* web já existente? Primeiro há que entender o funcionamento da plataforma U=RIsolve, bem como os dados envolvidos no processo de análise de circuitos.

Analise-se o seguinte diagrama de blocos, que, de forma simplificada, retrata os passos que o utilizador deveria realizar para analisar um circuito na versão antiga da plataforma U=RIsolve.

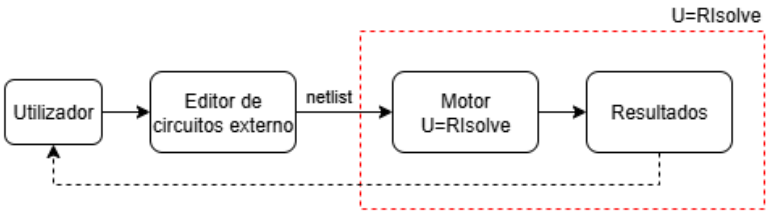


Figura 4.2: *Workflow* antigo

De forma mais detalhada, os passos para a análise de um circuito eram os seguintes:

1. O utilizador faz uso de um editor de circuitos externo para construir o circuito a ser analisado (e.g. Qucs);

2. O utilizador exporta um ficheiro representativo desse circuito (esquemático (SCH) ou *netlist*);
3. O utilizador submete o ficheiro, acompanhado de uma imagem (opcional) do circuito, na plataforma U=RIsolve;
4. O utilizador escolhe o método de resolução que pretende (MTN, MCM, MCR ou TSP) e clica em "analisar";
5. O U=RIsolve faz os cálculos necessários à análise e retorna os resultados ao utilizador, com uma análise passo-a-passo (segundo o método escolhido).

A conclusão é que este método de trabalho é pouco eficiente, depende completamente de um editor externo. Este foi um ponto comentado por diversas vezes em trabalhos anteriores e pela generalidade dos utilizadores desta primeira versão da plataforma U=RIsolve.

Em contraste, o novo diagrama de blocos retrata o método de trabalho proposto com a inclusão do novo editor de circuitos, mantendo o resto da plataforma inalterada.

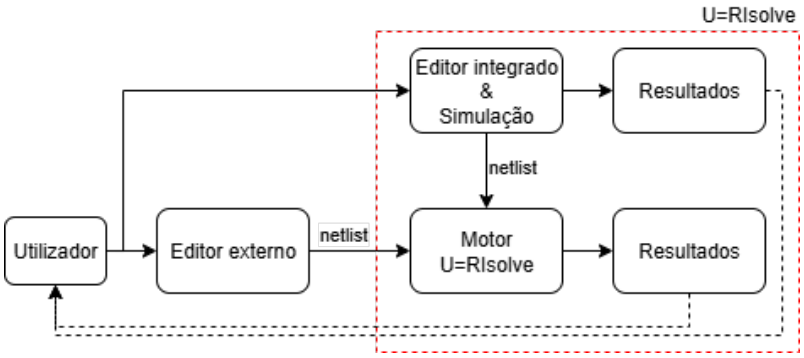


Figura 4.3: Novo *workflow*

Agora o utilizador dispõe de várias opções:

- Continuar a utilizar um editor externo, como anteriormente;
- Utilizar o novo editor de circuitos, que se encontra integrado na plataforma, simulando o circuito diretamente, e obtendo resultados imediatos;
- Utilizar o novo editor de circuitos para criar o circuito, mas utilizar o mesmo motor de análise anterior, escolhendo o método de resolução que pretende, e obtendo os resultados de forma detalhada.

No futuro serão ainda possíveis novas interações, como por exemplo:

1. Integração do módulo iGen, um módulo capaz de gerar modelos de circuitos elétricos de forma automática e customizável;

2. Utilização de esquemas elétricos de circuitos representados no editor como material para a criação de *Quizzes*, do lado do U=RIsolve Academy;

Esta nova abordagem é muito mais eficiente, e permite ao utilizador criar e analisar circuitos de forma rápida e intuitiva, sem abandonar a plataforma.

4.1.3 Ferramentas, tecnologias e estilos de programação

Quanto a ferramentas e tecnologias, o editor/simulador de circuitos DiCES mereceu uma análise entre diversas opções. A opção que se destacou mais acabou também por ser a mais simples, e consistiu em programar o DiCES totalmente do zero, sem recurso a nenhuma biblioteca ou *framework* externa, utilizando apenas *JavaScript* puro, HTML e CSS. Esta decisão foi tomada com base em diversos fatores, tais como:

- A redução de dependências externas, o que torna o projeto mais leve e fácil de manter;
- A flexibilidade total na implementação, permitindo adaptar o código exatamente às necessidades do projeto;
- A facilidade de integração com a plataforma U=RIsolve, que já utilizava estas tecnologias;
- O nível de ensino médio que o projeto se propõe, tornando o código mais acessível a futuros estudantes que possam vir a trabalhar no mesmo.
- A simplicidade do editor, que não requer funcionalidades avançadas que justifiquem o uso de bibliotecas complexas.

Existe ainda a questão de implementação técnica do DiCES, mesmo dentro o âmbito do *JavaScript* puro. A decisão aqui recaiu entre utilizar elementos DOM [32] diretamente, ou recorrer a um elemento CANVAS [33] para desenhar o editor [34]. A escolha recaiu sobre a utilização do elemento CANVAS, devido a:

- Facilidade em desenhar elementos gráficos, como componentes e conexões, de forma dinâmica e eficiente;
- Melhor desempenho em operações de desenho e atualização, especialmente para editores gráficos;
- Aspeto visual com maior qualidade, permitindo efeitos gráficos mais avançados;
- Maior controlo sobre o desenho e manipulação dos elementos gráficos.

Quanto ao estilo de programação, o DiCES foi desenvolvido utilizando um paradigma orientado a objetos, onde cada componente, conexão e nó é representado por uma classe específica, com propriedades e métodos próprios. Este estilo de programação facilita a organização do código, tornando-o mais modular e reutilizável. Para além disso, é gerada naturalmente uma estrutura hierárquica de objetos muito similar à estrutura de dados utilizada para representar o circuito, o que facilita a manipulação e atualização do mesmo. A abordagem geral foi muito semelhante à utilizada em [35], onde é explicado como criar um editor gráfico simples utilizando o elemento CANVAS e programação orientada a objetos em *JavaScript*.

4.2 Projeto do DiCES

Tendo como base o estudo comparativo realizado no capítulo anterior, e novas necessidades discutidas previamente, é possível definir um conjunto de requisitos e funcionalidades que o novo editor/simulador de circuitos, DiCES, deve possuir, bem como características específicas de *User Interface* e *User Experience* que devem ser tidas em conta no desenvolvimento do projeto.

Reunem-se aqui os pontos mais importantes a considerar no projeto do DiCES:

- deve ser intuitivo e fácil de usar, mesmo para utilizadores sem experiência prévia em edição de circuitos;
- a interface deve ser limpa e organizada, evitando a sobrecarga visual.
- deve permitir a adição rápida e eficiente de componentes, a partir de uma aba lateral (semelhante à de outras aplicações, como o QUCS), isto é, selecionar o componente com um clique, e posicioná-lo no esquema com outro clique, sendo possível ajustar a orientação em passos de 90 graus;
- deve suportar os seguintes tipos de componentes elétricos básicos:
  - **Fontes**: incluem fontes de tensão contínua (VDC), fontes de corrente contínua (IDC), fontes de tensão alternada (VAC), e fontes de corrente alternada (IAC).
  - **Componentes passivos**: incluem resistências, condensadores e bobinas.
  - **Instrumentos de medição**: incluem voltímetros, amperímetros, ohmímetros e wattímetros [1].

[1] De notar que apenas os voltímetros e amperímetros serão de facto implementados nesta fase, mas ficam adiantados em *design* os outros instrumentos para futuras versões do editor/simulador.

  - ***Meta-Componentes***: neste grupo incluem-se a massa (GND) e as etiquetas de nomeação de nós. [2]

- a adição de fios de ligação deve ser simples, com um clique para iniciar e outro para terminar a ligação. A ligação é feita por dois segmentos em L, e a orientação superior ou inferior deve ser escolida pelo utilizador;
- deve suportar a criação automática de nós ao conectar múltiplos fios/componentes num mesmo ponto;
- a multi-seleção deve ser possível para facilitar a manipulação de vários elementos simultaneamente;
- deve permitir a edição das propriedades dos componentes e fios de ligação diretamente na interface, através de um menu de propriedades;
- deve ter capacidades de simulação integradas, permitindo ao utilizador validar e simular o circuito na mesma janela da interface;
- quanto aos dados resultado da simulação, estes devem ser apresentados sobre o circuito, de forma clara, podendo ser ativados ou desativados conforme a preferência/necessidade do utilizador.
- quanto a dados resultado segundo um método específico, estes devem ser apresentados numa aba lateral, que seja colapsável, para não interferir com o esquema;
- deve permitir a exportação do circuito em diversos formatos, incluindo *netlist*, *schematic* (SCH), JSON e imagem;
- deve permitir a importação de circuitos a partir de diversos formatos, incluindo *schematic* (SCH) e JSON.

Nota: para já, não é possível importar *netlists*, pois estas não contêm informação gráfica suficiente para reconstruir o esquema, nomeadamente a sua geometria.

No entanto, esse trabalho já foi desenvolvido noutro projeto, que visa a geração automática de esquemáticos e irá ser integrado no futuro, permitindo também partir de um ficheiro *netlist* e obter uma possível representação gráfica.

[2] A consideração da massa como um componente ou não componente foi uma decisão a tomar no editor/simulador, dado que é um elemento que altera o circuito (na medida em que define o potencial como 0 V no ponto onde estiver colocada), mas é apenas uma referência de potencial. Por este motivo, optou-se por considerá-la como um *meta-componente*, fazendo essa distinção. Do ponto de vista do modelo de dados, a massa continua a ser representada utilizando a classe *Component*.

4.3 Implementação

Nesta seccção, são apresentados alguns dos pontos chave da implementação do editor/simulador de circuitos, DiCES, assim como alguns detalhes técnicos mais relevantes.

4.3.1 Base do editor/simulador de circuitos

O funcionamento do DiCES assenta na utilização de um elemento HTML5 `<canvas>`, que permite desenhar gráficos de forma interativa, utilizando *JavaScript*. Assim, o DiCES consiste num ciclo de atualização contínuo, onde cada iteração é responsável por limpar o `canvas` e fazer refletir o conteúdo atual do modelo de dados do circuito no ecrã, desenhando os componentes, ligações, pontos de conexão, e outros elementos visuais relevantes.

O código responsável por este ciclo de atualização é o seguinte:

```
1  const circuitEditor = new CircuitEditor({
2      canvas,
3      circuit: new Circuit(),
4      textCanvas
5  });
6
7  function animate(){
8      circuitEditor.clear();
9      circuitEditor.display();
10     requestAnimationFrame(animate);
11 }
12
13 animate();
```

Listagem 4.1: Ciclo de atualização do DiCES

Note-se a utilização função *requestAnimationFrame* [36], nativa do *JavaScript*, que permite criar animações suaves e eficientes, sincronizando o ciclo de desenho com a taxa de atualização do ecrã.

4.3.2 Desenho do editor/simulador de circuitos

O desenho do editor/simulador de circuitos é levado a cabo através da função `display()`, desenvolvida neste projeto, que é responsável por desenhar todos os elementos visuais do editor, incluindo a grelha de fundo, os componentes, as ligações, os pontos de conexão, e outros elementos relevantes. A função é a seguinte:

```
1     display() {
2         this.clear();
3
4         //move the whole canvas by the viewport offset
5         this.ctx.translate(this.editorProperties.viewport.position.x,
              this.editorProperties.viewport.position.y);
6
7         this.drawBackground(this.ctx,
              this.editorProperties.window.grid.visible,
              this.editorProperties.window.origin.visible);
8
9         this.circuit.draw(this.ctx);
10
11        // draw the selected components
12        this.selectedComponents.forEach(component => {
13            component.drawBoundingBox(this.ctx);
14        });
15
16        // draw the selected wires
17        this.selectedWires.forEach(wire => {
18            wire.drawBoundingBox(this.ctx);
19        });
20
21        if (this.componentPreview) {
22            this.componentPreview.draw(this.ctx);
23        }
24
25        if(this.mode === 'connect'){
26            // draw cross
27            if(this.connectionCross){
28                this.connectionCross.draw(this.ctx);
29            }
30            if(this.connectionPreview){
31                this.connectionPreview.draw(this.ctx);
32            }
33        }
34
35        if(this.mode === 'boxselect'){
36            this.boxSelect.draw(this.ctx);
37        }
38
39        //move the whole canvas back
```

```
40          this.ctx.translate(-this.editorProperties.viewport.position.x,
                -this.editorProperties.viewport.position.y);
41
42          // draw the metadata on the text canvas
43          this.textCtx.clearRect(0, 0,
                this.editorProperties.viewport.width,
                this.editorProperties.viewport.height);
44
45              if (
46                  this.circuit.metadata?.simulation &&
47                  (
48                      Object.keys(this.circuit.metadata.simulation?.currents
                            || {}).length > 0 ||
49                      Object.keys(this.circuit.metadata.simulation?.voltages
                            || {}).length > 0
50                  )
51              )
52              {
53
54              if(this.drawPolarizarion){
55                  // add a semi transparent background, with a slight blur
56                  this.textCtx.fillStyle = "rgba(255, 255, 255, 0.6)";
57                  this.textCtx.fillRect(0, 0,
                        this.editorProperties.viewport.width,
                        this.editorProperties.viewport.height);
58
59                  this.textCtx.translate(this.editorProperties.viewport.position.x,
                        this.editorProperties.viewport.position.y);
60
61                  if(this.drawPolarizarion === 'nodes')
                        this.circuit.drawPolarizationNodeText(this.textCtx);
62                  if(this.drawPolarizarion === 'branches')
                        this.circuit.drawPolarizationBranchText(this.textCtx);
63                  if(this.drawPolarizarion === 'flags')
                        this.circuit.drawPolarizationFlagText(this.textCtx);
64                  if(this.drawPolarizarion === 'allNodes')
                        this.circuit.drawPolarizationAllNodeText(this.textCtx);
65                  if(this.drawPolarizarion === 'instruments')
                        this.circuit.drawPolarizationInstrumentText(this.textCtx);
66
67                  this.textCtx.translate(-this.editorProperties.viewport.position.x,
                        -this.editorProperties.viewport.position.y);
68              }
69
```

```
70          }
71      }
```

Listagem 4.2: Função de desenho do circuito elétrico no editor/simulador

Nesta função, podemos destacar alguns aspetos de implementação:

- Ideia de grelha infinita: A grelha de fundo é desenhada de forma a cobrir toda a área visível do editor/simulador, em torno no ponto de origem, que é o centro do `canvas`, e com um *offset* tanto na horizontal como na vertical, de forma a permitir o movimento do *viewport* sem que a grelha desapareça.
- Vários modos de utilização: O editor/simulador possui vários modos de utilização, como o modo de conexão, onde o utilizador pode ligar componentes entre si, e o modo de seleção por caixa, onde o utilizador pode selecionar múltiplos componentes desenhando uma caixa de seleção. Todos estes modos implicam um desenho de componentes visuais no editor/simulador, que prestam auxílio ao utilizador.
- Desenho de metadados: O editor/simulador permite desenhar metadados diretamente sobre o circuito. No entanto, repare-se na utilização de um segundo `canvas`, que é sobreposto ao primeiro, e que é utilizado exclusivamente para desenhar os metadados. Desta forma, o desenho dos metadados é independente do desenho do circuito, facilitando a sua gestão e atualização.

4.3.3 Descrição das ferramentas disponíveis

O DiCES dispõe de várias ferramentas que permitem ao utilizador interagir com o circuito de forma eficiente. Estas ferramentas encontram-se apresentadas na barra superior do editor/simulador, Figura 4.5, e incluem:

1. **Gaveta de componentes:** Permite ao utilizador selecionar e adicionar componentes ao circuito, como resistências, fontes de tensão, etc. O funcionamento da adição de componentes é realizado através da seleção do componente com um clique, e posterior adição ao esquema ao circuito com um novo clique. É ainda utilizado o botão direito para alterar a orientação do componente de vertical para horizontal, e vice-versa. Foram implementados os seguintes componentes:

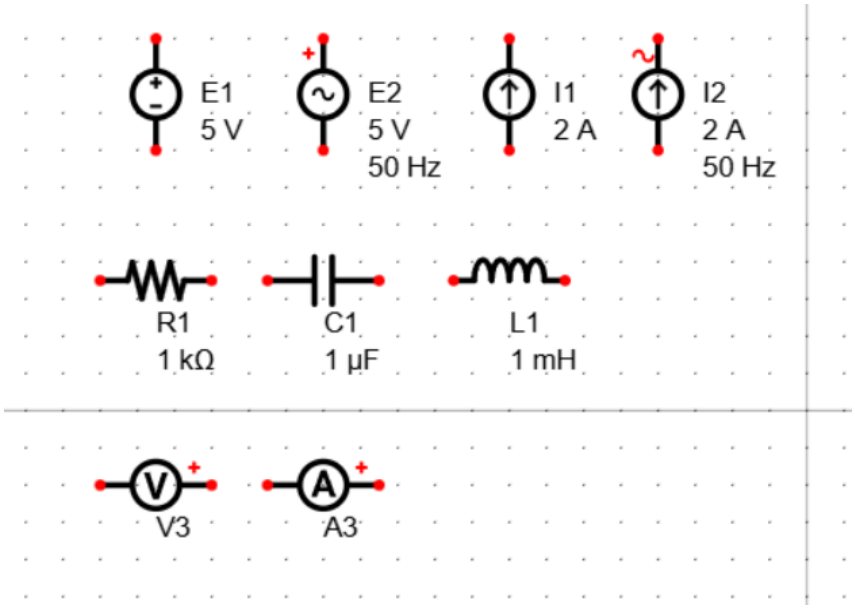


Figura 4.4: Componentes implementados no DiCES

2. **Conectar:** Permite ao utilizador adicionar ligações ao circuito. Para isso, após a seleção da ferramenta, o utilizador deve clicar no ponto inicial da ligação, ficando com uma ligação fantasma em L. De seguida, deve arrastar o rato e clicar novamente, desta vez no ponto final da ligação, para a adicionar ao esquema. Enquanto a ligação fantasma existe é possível alterar a sua orientação, de *upper elbow* ("joelho superior") para *lower elbow* ("joelho inferior") e vice-versa, utilizando o botão direito do rato.

3. **Adicionar etiqueta:** Permite ao utilizador adicionar etiquetas ao circuito, para identificar nós e respetivo potencial (depois de simular e ativar os metadados). O funcionamento é semelhante ao da adição de componentes, onde o utilizador clica no botão da ferramenta, e posteriormente clica no ponto do circuito onde pretende adicionar a etiqueta.

4. **Massa (elétrica):** Permite ao utilizador adicionar uma massa de referência ao circuito, garantindo que esse ponto fica ao potencial de 0V. O funcionamento é novamente semelhante ao da adição de componentes ou etiquetas.

5. **Centrar:** Permite ao utilizador centrar o circuito no *viewport*, ajustando a posição e *zoom* do *viewport* de forma a que o circuito fique visível e centrado.

6. **Eliminar selecionado:** Permite ao utilizador eliminar os componentes ou ligações selecionados no circuito. A seleção pode ser feita através de clique individual (CTRL + clique), ou através da ferramenta de seleção por caixa. A tecla "delete" também tem ingual funcionalidade.

7. **Ferramentas de zoom:** Permitem ajustar manualmente o nível de *zoom* do *viewport*, aproximando ou afastando a vista do circuito. O botão "-" reduz o *zoom*, enquanto o botão "+" aumenta o *zoom*. É ainda possível definir o zoom diretamente, em percentagem, na caixa de texto entre os dois botões.

8. **Desfazer/refazer (*undo/redo*):** Permitem ao utilizador desfazer ou refazer a última ação realizada no circuito, facilitando a correção de erros ou a experimentação com diferentes configurações do circuito.

9. **iGen - Intelligent circuit Generator:** Ferramenta generativa que permite ao utilizador criar automaticamente um circuito simples, com base em parâmetros definidos pelo próprio. Esta ferramenta é uma integração do Gerador Automático de Modelos de Esquemáticos (GAME) [37].

10. **Abrir no analizador:** Gera a *netlist* do circuito atual, e abre página do analizador de circuitos do U=RIsolve Academy, carregando automaticamente o circuito para análise segundo um dado método.

11. **Validar circuito:** Executa o algorimto de validação do circuito, retornando erros e avisos, se aplicável.

12. **Simular circuito:** Executa o algoritmo de simulação do circuito. Caso existam erros de validação, a simulação não é executada, e o utilizador é notificado. Caso contrário, o circuito é simulado e os resultados/metadados ficam disponíveis para visualização.

13. **Metadados de simulação:** Permite ao utilizador escolher quais os metadados que pretende visualizar sobre o circuito, como as correntes nos ramos, as tensões nos nós, etc.

14. **Analizar segundo um método (MCR, MCM):** Permite ao utilizador analisar o circuito segundo um determinado método, fazendo uso no analizador de circuitos do U=RIsolve Academy. Difere da ferramenta "Abrir no analizador" por ser integrado no próprio editor/simulador, sem necessidade de abrir uma nova página.

15. **Novo circuito:** Permite ao utilizador criar um novo circuito, limpando o circuito atual e iniciando um novo.

16. **Configurações:** Janela modal que permite ajustar configurações do editor/-simulador, como o esquema de controlo, o número de algarismos significativos dos metadados, ou alternar entre os modos de visualização de correntes (setas ou partículas).

17. **Importar:** Permite importar circuitos a partir de ficheiros JSON, carregando o modelo do circuito no editor/simulador, ou a partir de ficheiros de esquemáticos (SCH), convertendo-os para o modelo do editor/simulador.

18. **Exportar:** Permite exportar o circuito atual para ficheiros JSON, guardando o modelo do circuito, ou para imagens em formato *.png*, exportando uma imagem do circuito desenhado. É ainda possível exportar a *netlist* do circuito no formato *.txt*.

Figura 4.5: Barra de ferramentas do editor/simulador de circuitos

4.3.4 Menus de propriedades

O DiCES disponibiliza vários menus de propriedades, que permitem ao utilizador editar as propriedades dos componentes, ligações e da grelha de fundo. Estes menus são apresentados em janelas, que são abertas quando o utilizador clica com o botão direito do rato sobre um componente, ligação ou grelha de fundo.

Menu de propriedades da grelha de fundo

Este menu é o mais simples, e permite alternar a visibilidade da grelha de fundo e da origem do editor/simulador.

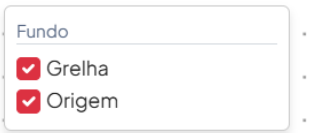


Figura 4.6: Menu de propriedades da grelha de fundo

Menu de propriedades dos componentes

O menu de propriedades dos componentes permite editar as propriedades específicas de cada componente, como o valor da resistência, a tensão da fonte, etc. O exemplo seguinte mostra o menu de propriedades de uma fonte de tensão alternada.

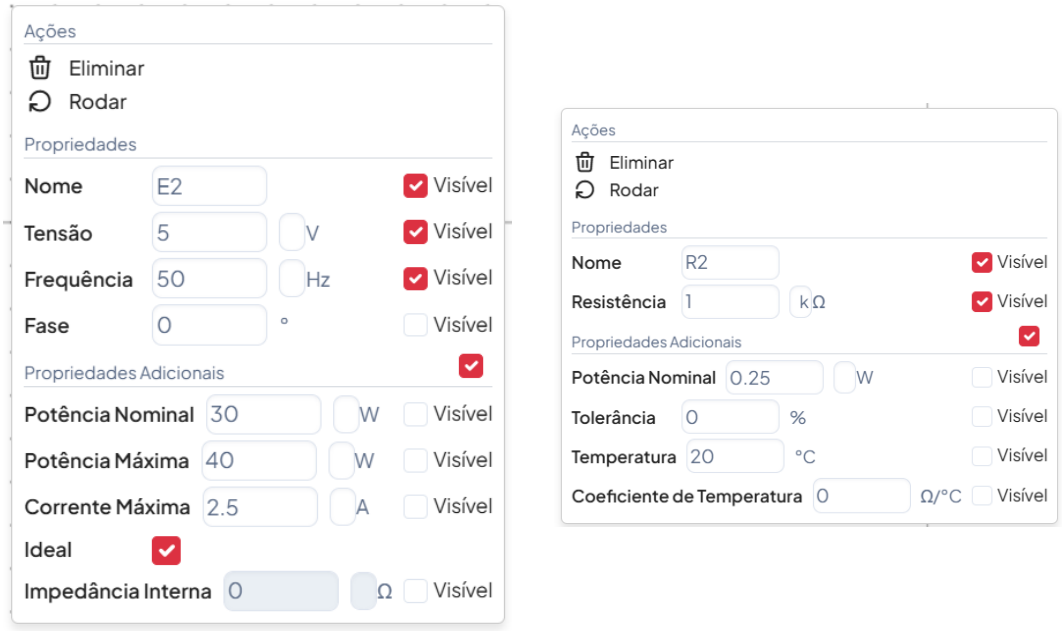


Figura 4.7: Menus de propriedades de uma fonte de tensão AC e Resistência (R)

Menu de propriedades das ligações

Por fim, o menu de propriedades das ligações permite editar as propriedades específicas de cada ligação, como a espessura da linha, a cor, etc. Destaca-se ainda a possibilidade de definir uma resistência interna do condutor, ou de definir o material, comprimento, e secção do condutor, calculando automaticamente a resistência com base nesses parâmetros. De momento, este valor de resistência interna não é utilizado na resolução do circuito, mas fica assinalada a possibilidade de o fazer em futuras versões do editor/simulador.

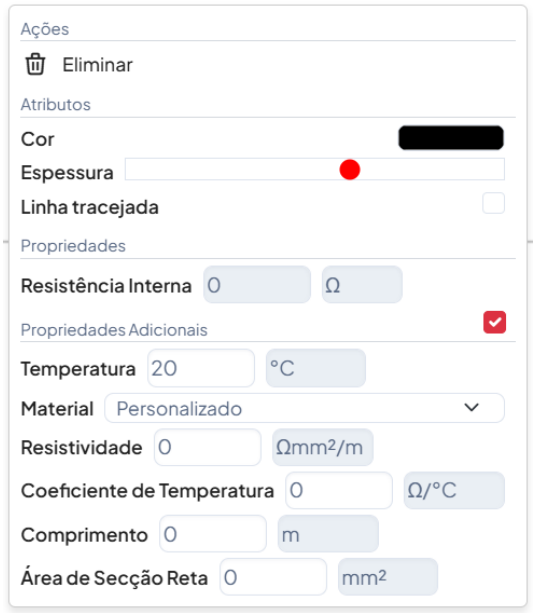


Figura 4.8: Menu de propriedades de uma ligação

4.4 Validação do circuito

Como já referido anteriormente, o editor/simulador de circuitos inclui um algoritmo de validação do circuito, que é executado sempre que o utilizador clica no botão "Validar Circuito", ou automaticamente antes de executar a simulação do circuito (botão "Simular"). O objetivo do algoritmo é perceber se vão existir problemas na sua resolução, e notificar o utilizador sobre esses problemas, para que possam ser corrigidos antes de avançar.

Note-se a diferença entre erros e avisos: os erros são problemas que impedem a resolução do circuito, enquanto os avisos são problemas que podem afetar a qualidade dos resultados, mas não impedem a resolução. Quando o utilizador valida o circuito, o editor/simulador notifica ambos os erros e avisos, mas quando o utilizador simula o circuito, apenas são notificados os erros.

Lista de avisos implementados:

- **Circuito sem massa.** – Um circuito sem massa de referência leva a resultados de potenciais flutuantes, o que é indesejável do ponto de vista prático, mas útil do ponto de vista didático.
- **O componente X tem um terminal desconectado.** – Componentes com terminais desconectados podem levar a resultados incorretos, uma vez que não estão a contribuir para o circuito.

- **Há terminais de condutores desconectados.** – Condutores com terminais desconectados podem levar a resultados incorretos, uma vez que não estão a contribuir para o circuito.

Na Figura 4.9, encontra-se um exemplo do primeiro warning: a falta de massa num circuito.

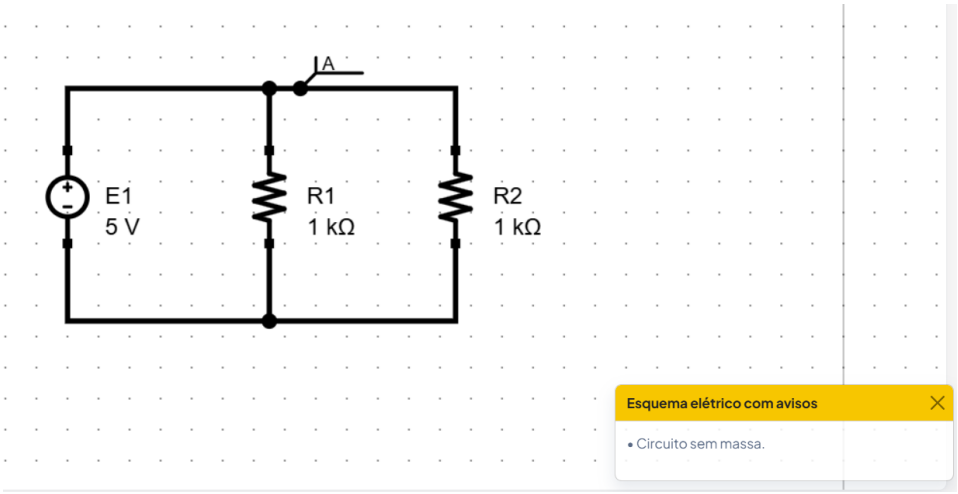


Figura 4.9: Exemplo de circuito sem massa - Warning

Lista de erros implementados:

- **Desenha um circuito primeiro.** – Não é possível resolver um circuito vazio.
- **Circuito sem componentes.** – Não é possível resolver um circuito sem componentes.
- **Circuito sem fontes de tensão ou corrente, ou omímetro.** – Não é possível resolver um circuito sem fontes de energia ou instrumentos de medição.
- **Circuito contém Amperímetros: remover amperímetros para poder simular.** [3]
- **Circuito contém Voltímetros: remover voltímetros para poder simular.** [4]
- **Nomes de componentes duplicados: X.** – Nomes de componentes duplicados podem levar a ambiguidades na resolução do circuito.

Na Figura 4.10, encontra-se um exemplo do último erro: erros duplicados.

[3] Erro temporário, até ser implementada a resolução com amperímetros
[4] Erro temporário, até ser implementada a resolução com voltímetros

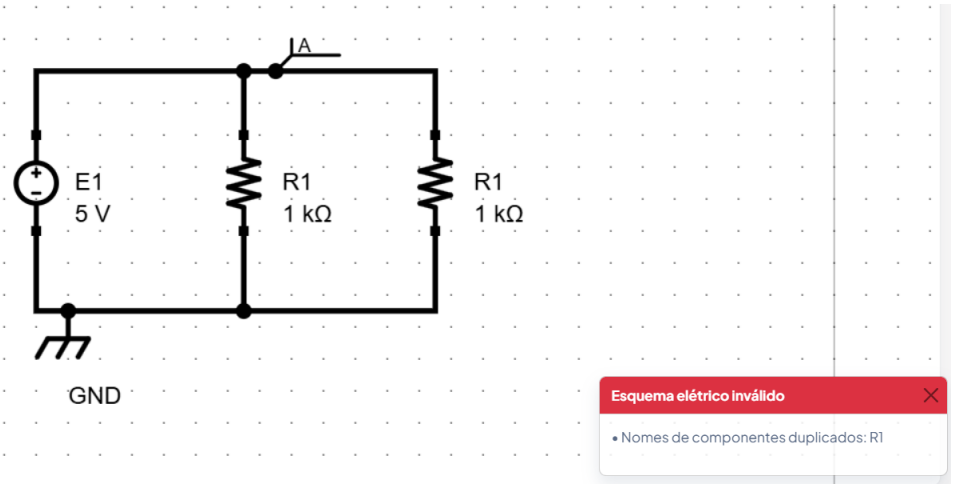


Figura 4.10: Exemplo de circuito com nomes de componentes duplicados - Erro

4.5 Ajuda ao utilizador

Para ajudar o utilizador nos passos iniciais de utilização do editor/simulador de circuitos, foi implementado um sistema de ajuda interativo, que guia o utilizador através das principais funcionalidades da aplicação, com o objetivo de criar e simular um circuito de raiz.

São apresentados 7 passos ao utilizador:

- **Passo 1:** Bem-vindo ao Editor de Circuitos!
- **Passo 2:** Adicionar componentes
- **Passo 3:** Conectar componentes
- **Passo 4:** Adicionar etiquetas
- **Passo 5:** Adicionar massa
- **Passo 6:** Simular circuito
- **Passo 7:** Parabéns!

Apesar de simples, este sistema consegue guiar o utilizador numa fase inicial, deixando ainda espaço para exploração livre da aplicação após a conclusão do tutorial.

Dá-se o exemplo do passo número 2 - Adicionar componentes:

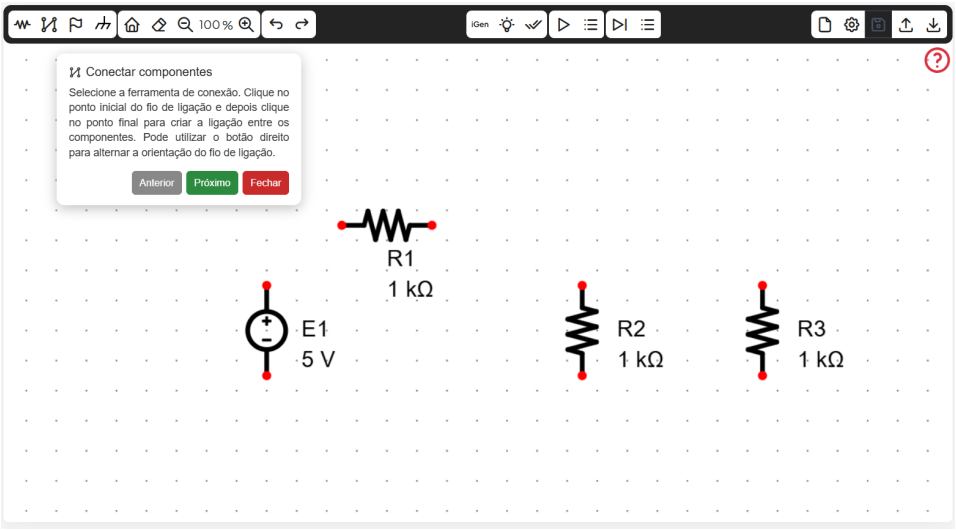


Figura 4.11: Exemplo do sistema de ajuda ao utilizador - Passo 2

4.6 Lógica de deteção de conexões e criação de pontos de conexão

Sempre que o utilizador adiciona, move, ou remove um componente ou fio de ligação, o DiCES atualiza automaticamente as conexões entre os elementos do circuito. Este processo envolve a deteção de quais componentes e fios condutores estão conectados entre si, e a re-criação de pontos de conexão conforme necessário. Assim, foi preciso conceber um altoritmo cuja função é receber a lista de componente e fios de ligação do circuito, no formato do modelo de dados acordado anteriormente, e devolver esse mesmo modelo atualizado, contendo a informação correta das conexões entre os elementos.

O algoritmo, em formato de fluxograma, encontra-se representado na Figura 4.12.

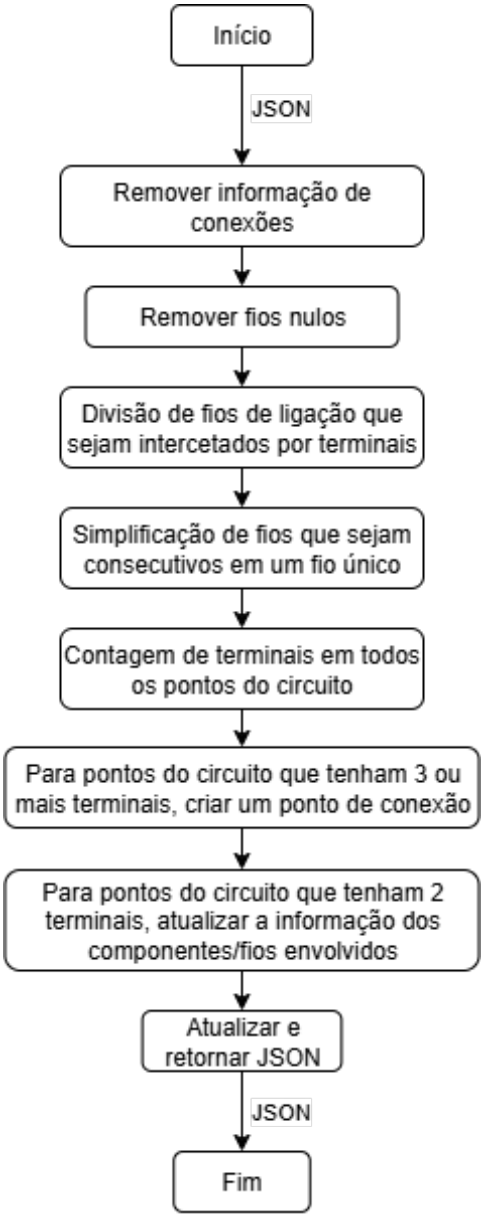


Figura 4.12: Fluxograma do algoritmo de deteção de conexões

De forma mais simplificada, o processo do algoritmo é o seguinte:

1. Receção do modelo de dados, contendo a lista de componentes e fios de ligação do circuito;
2. Limpeza de toda a informação de conexões existentes no circuito;
3. Limpeza de fios que sejam nulos, ou seja, que tenham o mesmo ponto inicial, final, e os seus vértices;
4. Deteção e divisão de fios que se cruzem com terminais de componentes ou outros fios;
5. Junção de fios que sejam consecutivos, ou seja, que partilhem um ponto final/inicial, tornando-os num único fio;
6. Perceber que pontos do circuito são pontos de conexão, contando quantos terminais de componentes e fios passam por cada ponto;
7. Criação dos pontos de conexão nos pontos do circuito que tenham mais do que dois terminais de componentes/fios a passar por eles;
8. Atualização da informação de conexões dos componentes e fios, para que cada terminal de componente e cada fio saiba a que pontos de conexão está ligado.

Capítulo 5

Camadas de metadados

Tendo o circuito desenhado no editor/simulador de circuitos, DiCES, o próximo passo lógico é permitir a sua simulação, e apresentação dos resultados diretamente no editor, sem necessidade de outras aplicações. Para tal, o DiCES é capaz de gerar a *netlist* do circuito, que é o formato de dados utilizado para a resolução do mesmo, e recorrer a um método de análise de circuitos, para calcular os potenciais nos nós do circuito, e as correntes nos ramos. Por fim, apresenta os resultados de uma forma clara e intuitiva, recorrendo a camadas de metadados sobrepostas ao próprio esquema elétrico, que podem ser ativadas ou desativadas pelo utilizador.

5.1 Resolução do circuito no DiCES

5.1.1 Geração da netlist

Tal como mencionado anteriormente, a *netlist* é o formato de dados que contém a informação necessária e suficiente para a resolução do circuito, isto é, a lista de componentes e a informação de como estão interligados. Assim, o primeiro passo é obter este ficheiro (.txt) partindo do modelo de dados JSON do circuito.

Este processo é relativamente simples, consistindo em percorrer a lista de componentes do circuito e, para cada componente, escrever uma linha no documento de *netlist* com a informação relevante, como o tipo de componente, os seus terminais, e os seus valores (resistência, tensão, etc.).

Como exemplo, considere-se o seguinte circuito simples:

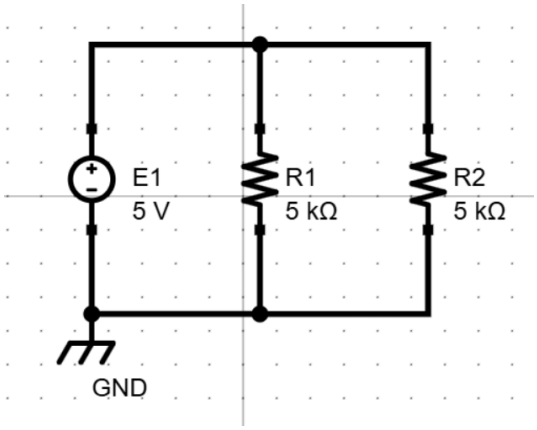


Figura 5.1: Circuito simples de exemplo

A linha da netlist, por exemplo, do componente E1, que é uma fonte de tensão, seria a seguinte:

```
1    Vdc:E1 node_a gnd E="5 V" node_a gnd
```

A linha da netlist relativa à resistência R1 seria a seguinte:

```
1    R:R1 gnd node_a R="5 kOhm" Temp="20"
```

E assim sucessivamente para todos os componentes do circuito. No caso do circuito-exemplo da Figura 5.1, a totalidade da netlist (completa) gerada pelo DiCES é a seguinte:

```
1    # U=RIsolve circuit Editor
2
3    R:R1 gnd node_a R="5 kOhm" Temp="20"
4    R:R2 gnd node_a R="5 kOhm" Temp="20"
5    Vdc:E1 node_a gnd E="5 V" node_a gnd
```

Inclusão de resistências internas

Alguns componentes, nas suas propriedades, permitem definir resistências internas, que devem ser consideradas na resolução do circuito, ou seja, devem constar na *netlist*. Assim, durante o processo da sua criação, o algoritmo deve verificar se o componente em questão possui algum destes atributos, e atualizar a *netlist* em conformidade, adicionando as resistências internas como componentes adicionais.

Por exemplo, no caso do circuito anterior, se a fonte de tensão E1 possuir uma resistência interna de 1 Ohm, passamos a ter a seguinte netlist do circuito:

```
# U=RIsolve circuit Editor

R:R1 gnd node_a R="5 kOhm" Temp="20"
R:R2 gnd node_a R="5 kOhm" Temp="20"
Vdc:E1 node_a _net_source_0 E="5 V" node_a _net_source_0
R:R_E1 _net_source_0 gnd R="1 Ohm" Temp="20"
```

Note-se que foi adicionado um novo componente de resistência, `R_E1`, que representa a resistência interna da fonte de tensão E1, e que foi criado um novo nó virtual, chamado `_net_source_0`, entre a fonte de tensão e a sua resistência interna. Embora este nó **não exista** no circuito real, ele tem de fazer parte do modelo do circuito e da *netlist* e é necessário para conseguir fazer incluir a resistência interna na resolução do circuito.

Este conceito de manupulação da *netlist* será aplicado a todos os componentes que possuam resistências internas, como fontes de corrente, indutores, condensadores, amperímetros, voltímetros, de acordo com a sua natureza série ou paralelo.

5.1.2 Simulação/Resolução do circuito

Numa fase inicial, o DiCES utiliza o Método das Correntes de Malha (MCM). No entanto, fica já assinalada a nota futura para a possibilidade de utilizar um método mais leve, como o Modified Nodal Analysis (MNA) [38], cuja natureza matricial é de mais simples e rápida execução, o que permitiria a resolução de circuitos em tempo real, e até enquanto são editados, proporcionando uma experiência mais fluida, interativa e síncrona para o utilizador. Para além disso, o MNA é um método mais completo, uma vez que os resultados incluem não só as correntes nos ramos, mas também as tensões nos nós, o que não acontece com o MCM. Isto é particularmente útil pois evitaria a necessidade do passo seguinte, que é o cálculo dos potenciais em todos os pontos do circuito (com base nas correntes nos ramos).

Neste ponto, a utilização do MCM é de fácil implementação, uma vez que o algoritmo já se encontra desenvolvido e testado (é um dos métodos disponíveis para a análise passo-a-passo), e a sua integração no editor é direta, bastando apenas fornecer o documento de *netlist* gerado anteriormente, e receber os resultados calculados, que neste caso, são as correntes nos ramos do circuito.

5.1.3 Cálculo de potenciais em todos os pontos do circuito

Tendo obtido as correntes nos ramos do circuito, o próximo passo é calcular os potenciais em todos os pontos do circuito, para que possam ser apresentados ao utilizador. Para este efeito, foi desenvolvido um algoritmo específico, cuja função é receber o modelo JSON do circuito, bem como as correntes nos ramos, e calcular os

potenciais utilizando a Lei de Ohm, de forma sucessiva, até que todos os potenciais sejam calculados.

A totalidade do algoritmo, isto é, todo o processo de resolução do circuito, encontra-se representado no diagrama de blocos da Figura 5.2.

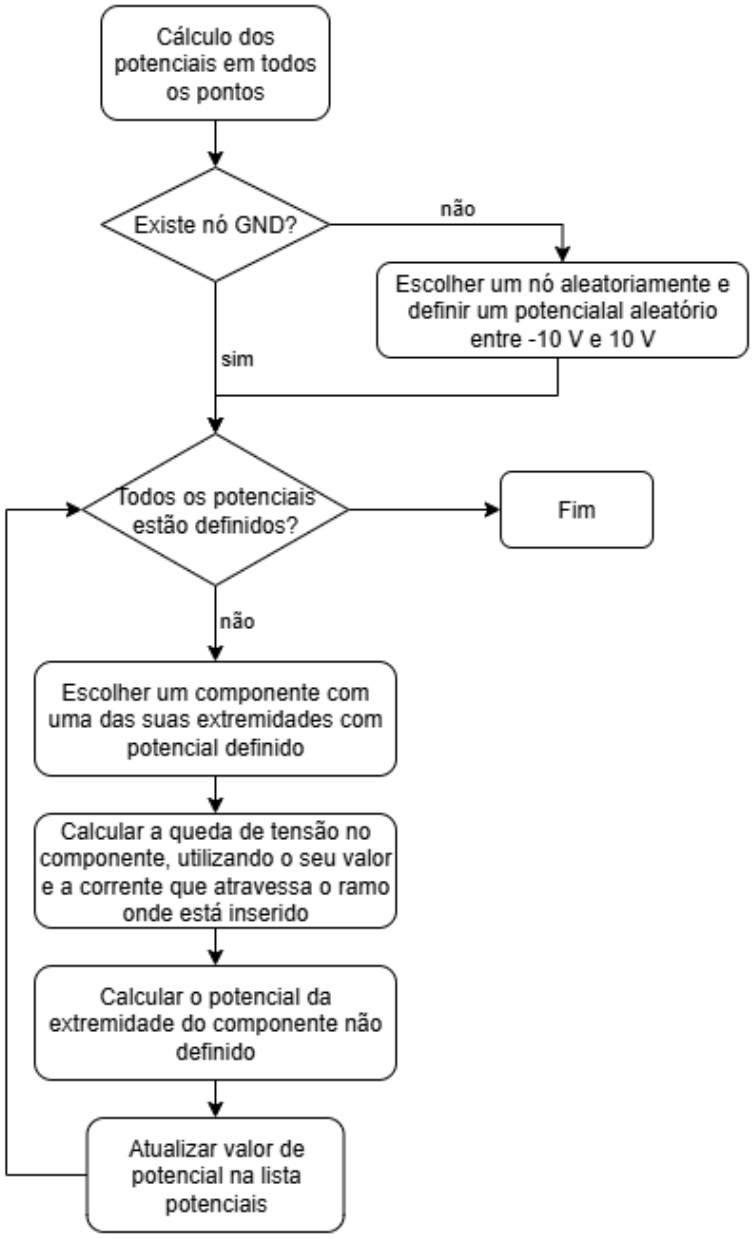


Figura 5.2: Diagrama de blocos do algoritmo de cálculo de potenciais nos nós do circuito

Destaca-se ainda a possibilidade de simular potenciais flutuantes, ou seja, circuitos que não possuam um nó de referência (massa). Para isso é apenas necessário, não incluir nenhuma massa elétrica no circuito. Assim, no início, o algoritmo escolhe um nó aleatório como nó de referência, atribuindo-lhe o potencial aleatório (no caso

foi utilizado um valor entre -10 V e 10 V), e prosseguir com o cálculo dos potenciais nos restantes nós do circuito. Esta funcionalidade é particularmente útil, uma vez que permite ao utilizador compreender melhor o funcionamento dos circuitos reais, e a importância de um nó de referência.

5.2 Apresentação dos resultados

Para a apresentação dos resultados determinados na fase de simulação do circuito, existem várias opções possíveis, cada uma com as suas vantagens e desvantagens. No entanto, a opção que neste caso fez mais sentido foi a de apresentar os resultados diretamente sobre o circuito, de forma clara e intuitiva, fazendo uso de elementos gráficos simples, tal como setas para as correntes, ou utilizando cores diferentes para salientar os nós. Desta forma, o utilizador pode percepcionar mais rapidamente quais os valores de tensão e corrente em cada ponto do circuito, sem necessidade de recorrer a tabelas ou gráficos adicionais. Para além disso, é possível adicionar outras funcionalidades, como a possibilidade de ativar ou desativar a apresentação dos resultados, ou de escolher quais os dados a apresentar, de forma a não sobrecarregar a interface do utilizador.

Foram implementadas as seguintes camadas de metadados:

- **Potenciais nos nós**: Cada nó do circuito é colorido usando uma cor distinta, e o valor do potencial é apresentado junto do mesmo;
- **Correntes nos ramos**: Cada ramo do circuito é representado por uma seta, cuja direção indica o sentido da corrente, e o valor da corrente é apresentado junto da mesma; Alternativamente, é possível representar a corrente através de partículas que se movem ao longo do ramo, cuja velocidade é proporcional ao valor da corrente; Esta escolha é feita no menu de configurações;
- **Potenciais em todos os pontos**: Semelhante à camada de potenciais nos nós, mas inclui todos os pontos do circuito, não apenas nós elétricos;
- **Potenciais nas etiquetas**: Semelhante à camada de potenciais nos nós, mas apenas inclui pontos que tenham sido identificados com uma etiqueta;
- **Valores medidos em instrumentos**: Cada instrumento de medição (voltímetro e/ou amperímetro) apresenta o valor medido diretamente sobre o mesmo. [1]

Seguem-se os exemplos das camadas de metadados simples mais relevantes.

[1] Esta funcionalidade não ficou habilitada por complicações a nível de geração de netlist, nomeadamente quando se incluem as resistências internas dos aparelhos. Apesar disto, é possível apresentar o aspeto desta camada de metadados (Figura 5.6)

Tensões nos nós

Apresenta-se a imagem com a camada de tensões nos nós ativa:

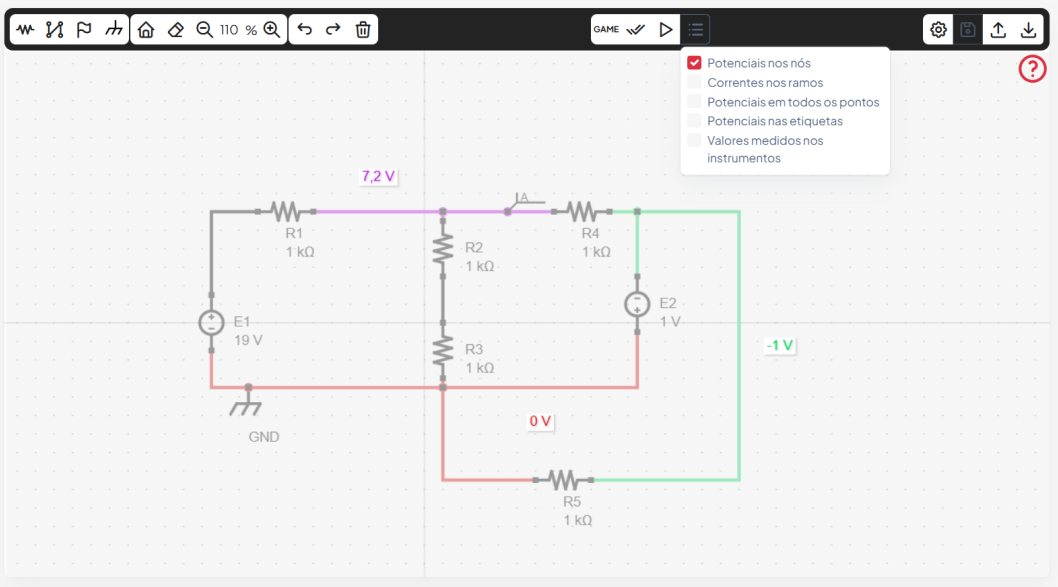


Figura 5.3: Camada de Metadados - Potenciais nos nós (exemplo)

Correntes nos ramos

Apresenta-se a imagem com a camada de correntes nos ramos ativa (versão com setas):

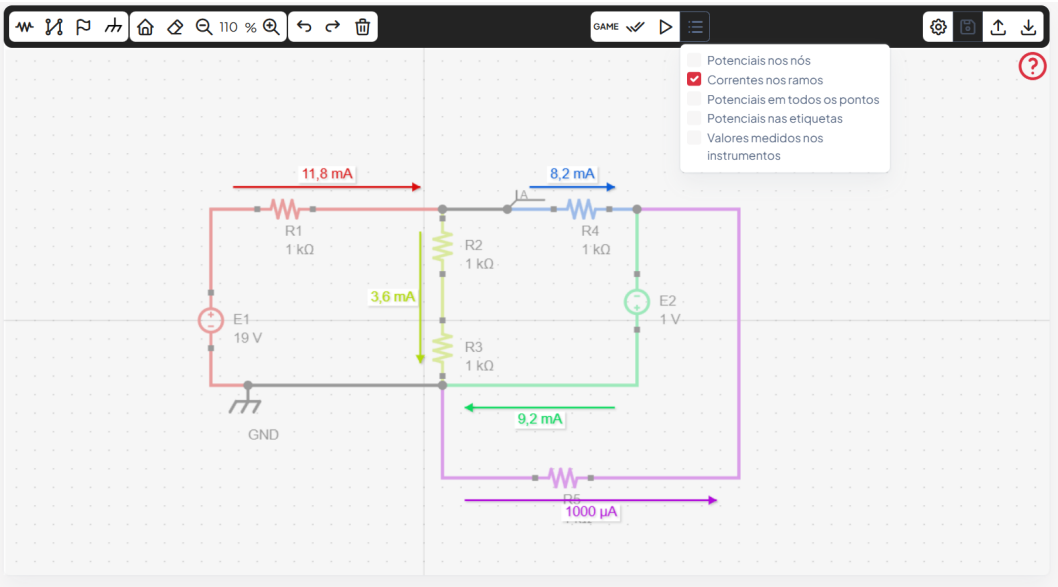


Figura 5.4: Camada de Metadados - Correntes nos ramos (exemplo)

Apresenta-se a imagem com a camada de correntes nos ramos ativa (versão com partículas):

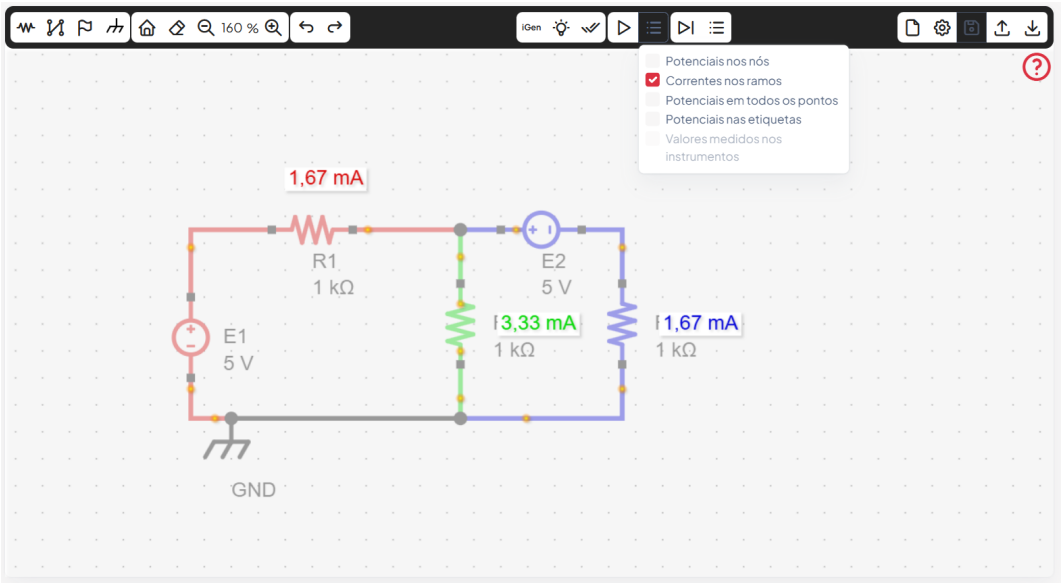


Figura 5.5: Camada de Metadados - Correntes nos ramos (exemplo com partículas)

Valores medidos em instrumentos

Apresenta-se a imagem com a camada de valores medidos em instrumentos ativa:

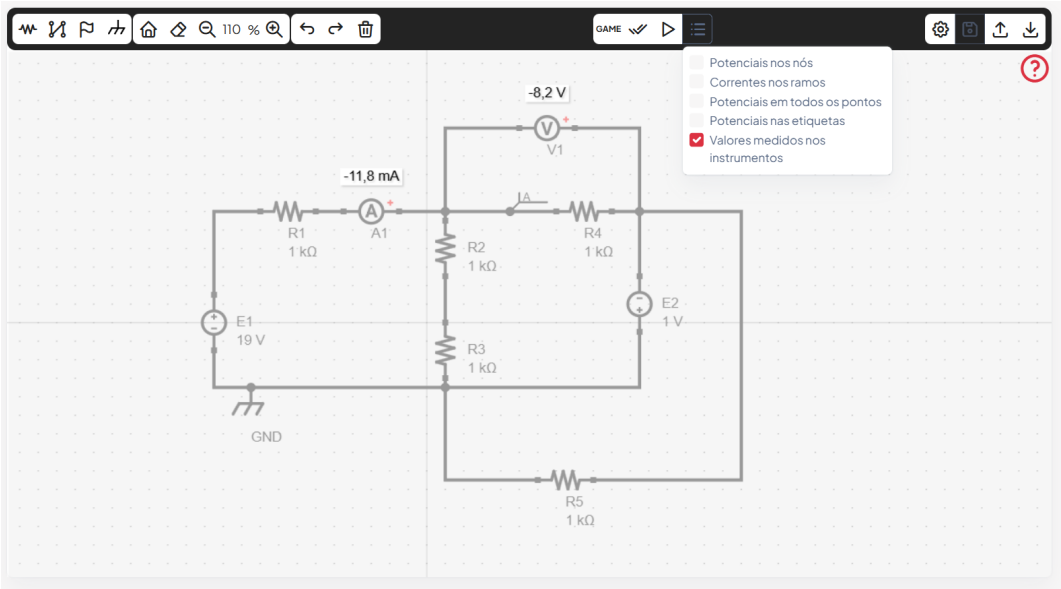


Figura 5.6: Camada de valores medidos em instrumentos

Capítulo 6

Experiência de utilização

Este capítulo tem como objetivo a obtenção de algumas opiniões relativamente à aplicação *web* U=RIsolve Academy, nomeadamente ao Editor/Simulador de circuitos DiCES. Para este efeito, foi realizada uma fase formal de testes, e ao preenchimento de um formulário, por parte de um grupo de utilizadores selecionados, incluindo ex-alunos e docentes da área de engenharia. Este grupo, totalizando 13 pessoas, fez ainda inclusão de várias faixas etárias de forma a obter uma variedade de perspetivas e experiências mais alargada.

6.1 Questionário

Para validar e obter opiniões sobre a aplicação desenvolvida, foi criado um formulário de testes ao editor/simulador de circuitos.

O formulário encontra-se disponível online no seguinte link: `https://forms.gle/tPVpyyBuqBetuiCP7` e está ilustrado no anexo .3.

A estrutura final do questionário foi a seguinte:

- Perguntas sobre o inquirido (identificação, experiência prévia com editores de circuitos);
- Perguntas sobre a experiência do utilizador (facilidade de uso, satisfação geral, funcionalidades mais/menos úteis);
- Perguntas sobre a interface do utilizador (facilidade de navegação, clareza dos ícones, organização das ferramentas);

- Perguntas abertas para sugestões e comentários adicionais.

Adianta-se já uma breve apresentação dos sujeitos que participaram nos testes, nomeadamente a sua experiência prévia com editores de circuitos elétricos.

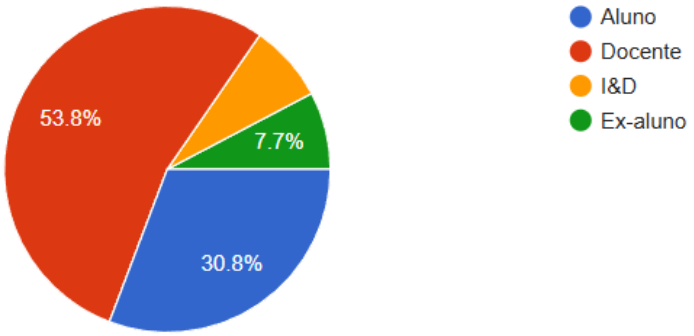


Figura 6.1: Dados do utilizador: aluno, docente, etc...

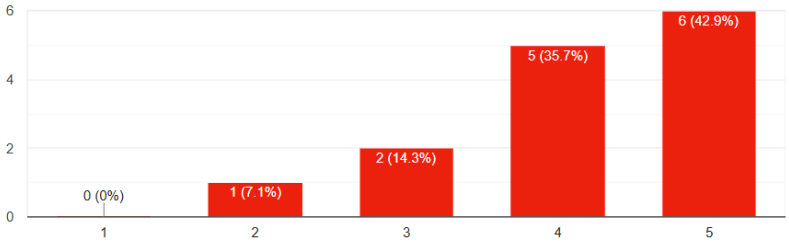


Figura 6.2: Dados do utilizador: experiência prévia com editores de circuitos

Analisando as figuras apresentadas, nomeadamente a Figura 6.2, é possível concluir que a maioria dos utilizadores possui experiência prévia com ferramentas semelhantes ao DiCES, o que é benéfico para a obtenção de feedback relevante e fundamentado. No entanto, também foram incluídos utilizadores com menos experiência, o que permite avaliar a facilidade de utilização incial do editor/simulador. Curiosamente, o utilizador que reportou a menor experiência prévia foi também aquele que atribuiu as melhores classificações ao DiCES em quase todas as categorias avaliadas, principalmente no que diz respeito a começar a utilizar a ferramenta.

6.2 User Experience (UX)

A *User Experience* (UX) refere-se à experiência geral do utilizador na interação com a aplicação DiCES, em questões como a facilidade de uso, o bom funcionamento, a satisfação quanto às funcionalidades, entre outros aspetos.

No questionário desenvolvido, foram incluídas perguntas que abordam estes tópicos, que são apresentadas de seguida, em conjunto com uma breve análise das respostas obtidas.

- **Quão fácil foi começar a utilizar o editor?**
  A maioria dos utilizadores reportou uma facilidade relativa a começar a utilizar o editor, com 60% a indicarem uma facilidade de 4, numa escala de 1 a 5.

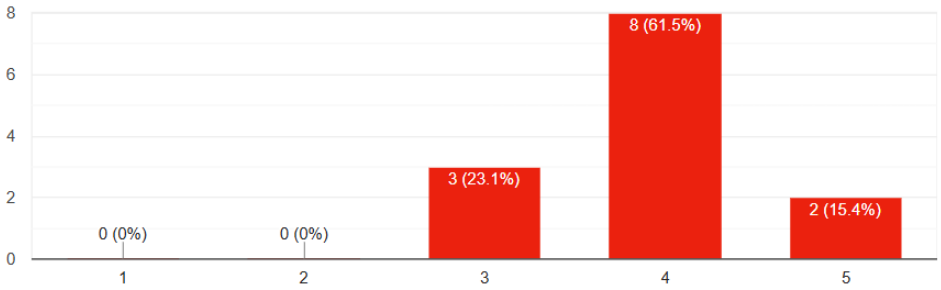


Figura 6.3: Facilidade de começar a utilizar o editor

- **Teve dificuldades? Quais foram?**
  As respostas a esta questão foram variadas. A maioria das respostas indicaram pequenos erros e problemas que podem ser resolvidos com facilidade. Outras respostas realçaram pontos de implementação já anotados para implementação futura. Alguns exemplos de respostas foram:

  - "Clicar no "esc"não desseleciona o componente e por vezes dava jeito";
  - "Inicialmente, não entendi muito bem o que fazia o botão simular. Ao clicar nos metadados da simulação percebi que tem as diferentes análises, contudo não consigo utilizar valores medidos nos instrumentos. Recebo mensagem de "Esquema elétrico inválido: Circuito contém voltímetros: remover voltímetros para poder simular."";
  - "Mover mais do que um componente em simultâneo";
  - "Análise do circuito com uma malha - não existe nenhum aviso sobre a impossibilidade de fazer a simulação.";

- **Quão fácil foi perceber a função de cada botão?**
  A maioria dos utilizadores reportou uma grande facilidade a perceber a função dos botões do editor/simulador de circuitos, com 84% a indicarem uma facilidade de 4 ou 5, numa escala de 1 a 5.

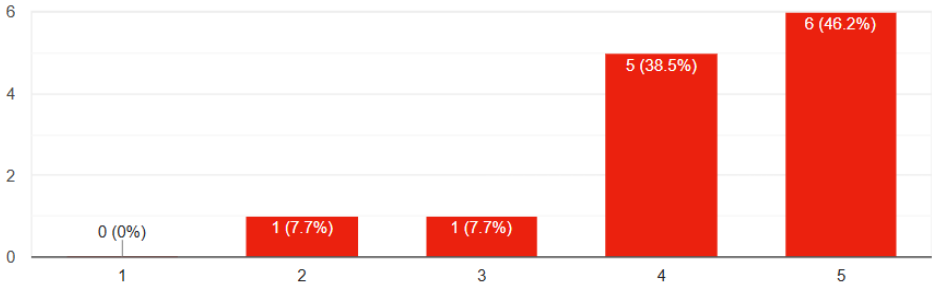


Figura 6.4: Facilidade de perceber a função de cada botão

- **Existe alguma funcionalidade que procurou e não encontrou?**
  As respostas a esta questão foram novamente diversas. Algumas respostas indicaram funcionalidades já planeadas para implementação futura, outras realçaram pontos de melhoria na interface, e outras ainda indicaram funcionalidades que não estavam planeadas, mas que podem ser consideradas para o futuro. Alguns exemplos de respostas foram:
  - "Duplo clique para editar um componente";
  - "Transistores e similares, implementação de instrumentação de medida, achei estranho estarem presentes, mas n serem possíveis de se usar";
- **Houve algum momento em que tenha ficado "preso" sem saber o que fazer a seguir?**
  Embora a maioria dos utilizadores tenha indicado que não, algumas respostas revelaram momentos de dúvida ou confusão, bem como novos erros. Algumas respostas foram:
  - "Fiz um circuito em CA e quando carreguei em simular (sem ter carregado no validar previamente por esquecimento) não me apareceu mensagem nenhuma e fiquei perdido. Porém, ao carregar no validar, apareceu-me então o pop-up a mencionar que faltava posicionar a massa."
  - "Tentei simular um circuito com uma fonte de tensão AC e uma fonte de corrente DC e não simulou. Não deu qualquer mensagem de erro."
- **Como classifica a rapidez e fluidez do editor?**
  A maioria dos utilizadores reportou uma rapidez e fluidez satisfatórias no editor, com 76% a indicarem uma classificação de 5, numa escala de 1 a 5.

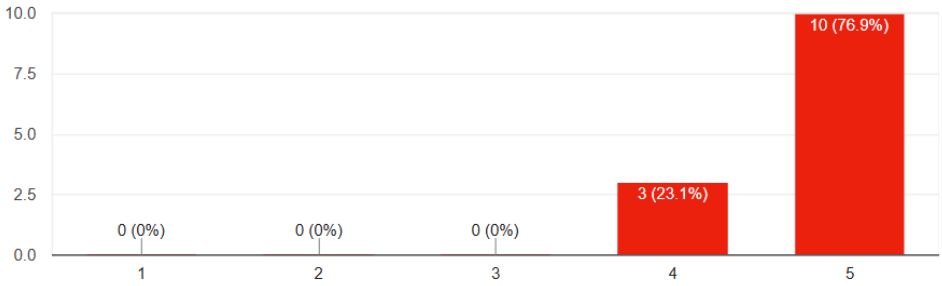


Figura 6.5: Classificação da rapidez e fluidez do editor

- **Sentiu lentidão? Onde?**
  Nenhum dos utilizadores reportou sentir lentidão no editor.

- **Detectou alguma falha/bug no que toca a conexões/pontos de interligação?**
  Alguns utilizadores reportaram falhas ou bugs relacionados com o comportamento das conexões e pontos de interligação. Algumas respostas foram:

  – "Sim, no cruzamento de duas ligações, é criado um curto-circuito no componente, e depois não se consegue eliminar esse curto. Nem fazendo CTRL-Z.";

  – "Não dá para arrastar linhas individuais, não é propriamente uma falha, mas dá jeito ocasionalmente.";

  – "Permite componentes iguais sobrepostos que ficam ligados entre si – confuso para, p. ex, dois GND sobrepostos.".

6.3 User Interface (UI)

A *User interface* (UI) refere-se à interface gráfica da aplicação DiCES, incluindo a disposição dos elementos, a estética visual e a facilidade de navegação. Uma boa UI é fundamental para garantir uma experiência de utilizador positiva.

No questionário desenvolvido, foram incluidas várias perguntas que abordam especificamente a UI do editor de circuitos, que são apresentadas de seguida, juntamente com as respostas obtidas.

- **Como classifica o aspeto visual do editor?**
  A maioria dos utilizadores reportou uma avaliação positiva do aspeto visual do editor, com 70% a indicarem uma classificação de 5, numa escala de 1 a 5.

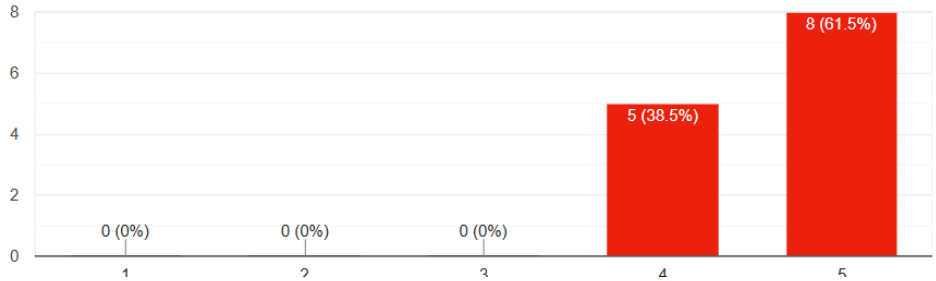


Figura 6.6: Classificação do aspeto visual do editor

- **Como classifica a disposição dos elementos na interface do U=RIsolve Academy, por exemplo, o menu lateral esquerdo, o menu superior do editor de circuitos, etc...** A maioria dos utilizadores reportou uma avaliação positiva na disposição dos elementos na interface da aplicação, com 69% a indicarem uma classificação de 5, numa escala de 1 a 5.

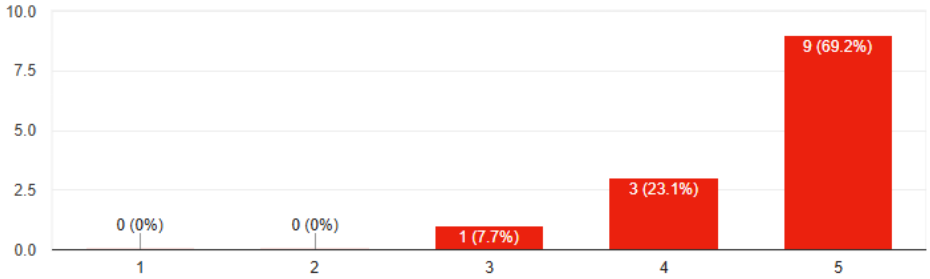


Figura 6.7: Classificação da disposição dos elementos na interface

- **Como classifica a intuitividade dos botões?**
  A maioria dos utilizadores reportou uma avaliação positiva da intuitividade dos botões do editor, com 23% a indicarem uma classificação de 3, 38% a indicarem uma classificação de 4, e 31% a indicarem uma classificação de 5, numa escala de 1 a 5.

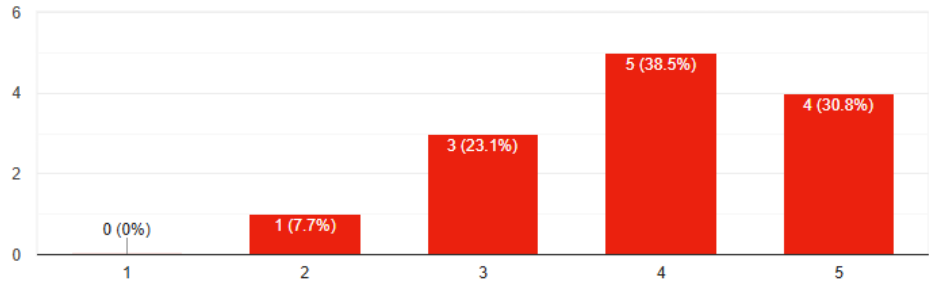


Figura 6.8: Classificação da intuitividade dos botões

- **Como classifica a escolha da paleta de cores?**
  A maioria dos utilizadores reportou uma avaliação positiva da escolha da paleta de cores do editor, com 54% a indicarem uma classificação de 5, numa escala de 1 a 5.

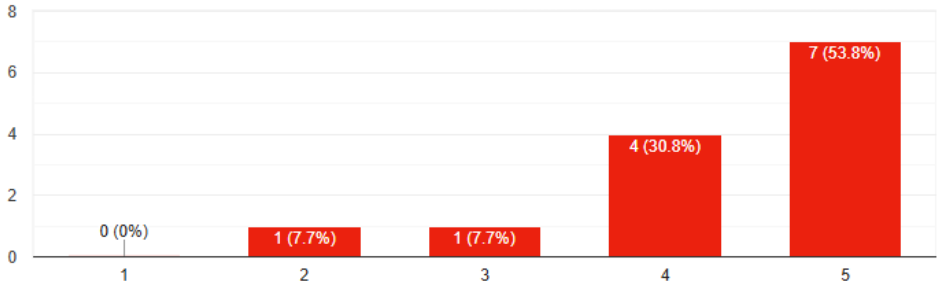


Figura 6.9: Classificação da escolha da paleta de cores

- **Como é a qualidade visual do circuito desenhado na janela do editor (grid, componentes, ligações, etc...)?**
  A maioria dos utilizadores reportou uma avaliação positiva da qualidade visual do circuito desenhado na janela do editor, com 77% a indicarem uma classificação de 5, numa escala de 1 a 5.

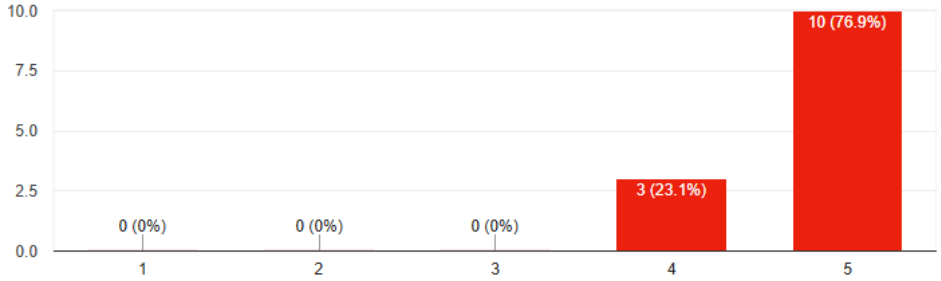


Figura 6.10: Classificação da qualidade visual do circuito

- **Como é a qualidade visual dos Metadados desenhados na janela do editor (nós, correntes, etc...)?**
  A maioria dos utilizadores reportou uma avaliação positiva da qualidade visual dos Metadados desenhados na janela do editor, com 75% a indicarem uma classificação de 5, numa escala de 1 a 5.

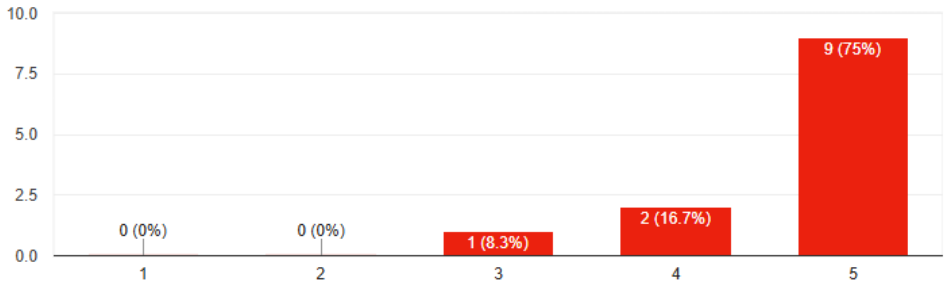


Figura 6.11: Classificação da qualidade visual dos Metadados

- **Em termos de interface, o que pode ser melhorado?**
  As respostas a esta questão, para além de variadas, foram bastante enriquecedoras, uma vez que realçaram pontos de melhoria que não tinham sido considerados inicialmente, e que podem ser implementados em futuras versões do editor. Alguns exemplos de respostas foram:
  - "Haver uma "toolbar"onde estejam os últimos N componentes inseridos no circuito, para ser mais rápido inserir mais componentes de um mesmo tipo."
  - "Quando se efetua a análise dos Metadados desenhados na janela do editor, o circuito fica "greyed out". Acho boa ideia colocar diferentes cores nos diferentes ramos, mas talvez fosse interessante reduzir o efeito de "greyed out"."
  - "A gaveta de componentes podia mostrar logo todos os componentes e tendo apenas o header, em vez de utilizar técnica dropdown. Talvez fosse mais prático uma vez que são poucos componentes. Por outro lado, esta técnica faz com que a tab não fique demasiado grande quando aberta."
  - "O Menu de ajuda deve ser melhorado e toda a ajuda deve ser contextual. Sugiro um mode de primeira utilização, que dá dicas."

6.4 Sugestões e opinião geral

Nesta secção, são apresentadas algumas das respostas a perguntas abertas sobre o que foi mais ou menos apelativo para cada utilizador, bem como o que pode vir a ser implementado no futuro.

- **O que mais gostou?**
  Reportam-se algumas respostas:
  - "Da facilidade da simulação."
  - "O simular e mostrar as cores, bem como valores no circuito, é espetacular. Testei alguns circuitos DC e funcionou com todos."

  - "Nota-se um esforço muito grande em apresentar diversas funcionalidades (pormenores como escolher como exportar a imagem do circuito). Acho que conseguiu pegar nas ideias que lhe foram passadas no início do trabalho e também capturar o melhor da iteração anterior do editor de circuitos. Acrescentou qualidade, com boas decisões ao nível da implementação de software."
  - "Do aspecto visual."
  - "Do facto de ser bastante intuitivo, com uma curva de aprendizagem muito baixa. Dos atalhos para componentes, acho que acelera bastante a edição."
  - "Da apresentação de meta-dados, muito bom para os alunos analisarem rapidamente."

- **O que menos gostou?**
  Alguns dos pontos menos positivos relatados foram:
  - "De ainda apresentar alguns bugs que precisam de ser trabalhados."
  - "A falta de um guia no início, para um utilizador sem experiência, pode ficar confuso começar."
  - "Erros sem informação suficiente para perceber."
  - "Da edição de propriedades de componentes, penso que se fosse possível editar diretamente sem ter de ir à janela de propriedades seria mais rápido."

- **Sugestões de funcionalidades que gostaria de ver implementadas**
  Algumas respostas a esta questão foram:
  - "Tutorial. Circuitos exemplo."
  - "Ao clicar na borracha, fixar a função de apagar, e poder ser utilizada de uma vez em varios pontos. Neste momento, é necessário selecionar a borracha multiplas vezes. E, tal como referido anteriormente, duplo clique para editar componentes."
  - "Exportação rápida dos resultados de simulação, ou visualização numa tabela para depois exportar em imagem."
  - "Instrumentos de medida."
  - "A cooligação da funcionalidade de geração automática de circuitos com o editor, até porque, numa componente de testes como esta, tive de ir pesquisar uns circuitos, pois não me ocorria nada à cabeça sobre o que montar (porém este deve ser um problema mais pessoal devido à minha falta de maestria com circuitos elétricos e o gerador automático iria me permitir explorar mais facilmente)."

6.5 Conclusão

Após a realização dos testes e da respetiva análise dos resultados obtidos, é possível inferir algumas conclusões relevantes sobre a utilização do DiCES. A maioria dos utilizadores reporta um balanço positivo nas respostas quantitativas ao questionário, destacando a facilidade de uso, a clareza da interface e a utilidade das funcionalidades implementadas. Para além disso, foram reportados alguns problemas (que foram devidamente registados para resolução futura) e sugestões de melhoria de alto valor, que poderão ser consideradas em futuras versões do editor/simulador de circuitos.

De modo geral, os resultados obtidos indicam que o DiCES é uma ferramenta promissora para a criação e análise de circuitos elétricos, com potencial para ser melhorada e expandida continuamente. Fica também evidente a importancia de realizar testes com utilizadores reais, de forma a obter *feedback* valioso que permita orientar o desenvolvimento futuro da aplicação.

Capítulo 7

Conclusões

7.1 Contribuições efetuadas

Pontos implementados

Este projeto teve como objetivo principal dar um recomeço de fresco a toda a aplicação U=RIsolve Academy, desde o design, à arquitetura, ao modelo de dados, fazendo incluir também o desenvolvimento de um editor/simulador de circuitos.

Quanto ao design, este foi reformulado e reprototipado, tendo sido implementadas algumas páginas de teste.

A arquitetura de toda a aplicação foi revista e reimplementada, de modo a ser o mais modular possível, e a incluir serviços de *backend* que permitam a futura expansão da aplicação.

O modelo de dados foi praticamente constuído de raiz para conseguir, não só acomodar o novo editor/simulador de circuitos, DiCES, mas também para ser o mais flexível possível, evitando ser necessário fazer alterações estruturais no futuro.

O DiCES foi projetado e desenvolvido de raiz, com base no estudo comparativo entre editores gráficos existentes, e tendo em conta as necessidades específicas impostas inicialmente, e inúmeras iterações na sequência de testes exaustivos por parte dos orientadores.

Deste modo, enumeram-se os diversos objetivos cumpridos neste trabalho:

1. Harmonização de código-fonte da plataforma *web* U=RIsolve Academy
   (a) Reestruturação da arquitetura de *software* do U=RIsolve Academy;

(b) Integração de métodos de análise previamente concebidos, nomeadamente o Método da Corrente nas Malhas (MCM) [3] e o Método da Corrente nos Ramos (MCR) [4];

(c) Melhoria da qualidade gráfica da interface do utilizador (UI);

2. Desenvolvimento do editor/simulador de circuitos DiCES

(a) Estudo da arte sobre editores e editores/simuladores de circuitos;

(b) Atualização e unificação do modelo de dados para representação do circuito elétrico, quer a nível lógico, quer a nível gráfico;

(c) Projeto e implementação do DiCES;

(d) Análise da qualidade do DiCES através de testes com vários utilizadores.

7.2 Sugestões para o futuro

Existem diversas melhorias e expansões que podem ser feitas, tanto à aplicação U=RIsolve Academy, como ao editor de circuitos em particular, nomeadamente algumas já identificadas previamente pela equipa e outras salientadas pelos beta-tester's.

A curto prazo, melhorias que podem ser feitas são:

- **Correção de erros:** Apesar de se poder considerar que o DiCES está numa fase estável, é possível que sejam encontrados erros.
- **Implementação do método MNA:** Atualmente, o editor utiliza o método MCM para a resolução dos circuitos. No futuro, seria interessante implementar o método MNA (Modified Node Analisys) [38], que é mais leve e rápido, permitindo a resolução de circuitos mais complexos, bem como a resolução em tempo real enquanto o circuito é editado [1];
- **Propriedades dos fios de ligação:** Os fios de ligação dispõem de uma lista de propriedades que pode ser editada, mas que atualmente não tem qualquer efeito na simulação do circuito. Seria interessante que estas propriedades se fizessem sentir na simulação, possibilitanto a inclusão de perdas nos condutores, ou verificação do comportamento de diversos materiais condutores;
- **Resultados segundo um método de análise específico:** De momento, os resultados da simulação de circuitos que são apresentados sobre o editor resumem-se apenas a potenciais e correntes. No entanto seria interessante integrar o analisador de circuitos ao editor, de modo a permitir a análise segundo um método específico, usufruindo assim do grafismo do editor. Dá-se

[1] Este projeto já está a ser desenvolvido por um estudante de PESTA/LEEC

o exemplo da resolução segundo o método das correntes de malha, em que é possível ter uma aba lateral com as malhas escolhidas, e respetivas equações, e sobreposto ao circuito existir a indicação e desenho de cada malha, numa nova camada de metadados.

Algumas das melhorias mais relevantes a longo prazo são:

- **Melhorar a usabilidade do DiCES:** Apesar de já ter sido feito um esforço considerável para obter uma aplicação o mais intuitívo e fácil de usar possível, existem sempre melhorias que podem ser feitas, de acordo com a massificação da sua utilização, particularmente por parte de alunos do 1º ano.
- **Adicionar mais componentes:** Atualmente, o DiCES possui um conjunto básico de componentes (componentes passivos e fontes de alimentação). No futuro é possível adicionar mais componentes à biblioteca, tal como díodos, transístores, transformadores, ohmímetro, etc. Isto deixaria o editor mais completo e com a possibilidade de criar circuitos mais complexos;
- **Adicionar mais funcionalidades de simulação:** O DiCES apenas tem a capacidade de fazer simulação de circuitos em regime permanente. Numa próxima fase, seria interessante adicionar a capacidade de simular circuitos em regime transitório;
- **Adicionar osciloscópio:** Uma vez que o DiCES possui capacidades de simulação CA, a inclusão de um osciloscópio permitiria visualizar e medir grandezas temporais (período, frequência, desfasamento) e de tensões (valores de pico, pico-a-pico, eficaz). Tendo implementado o ponto anterior, a adição de um osciloscópio virtual permitiria ainda avaliar todas estas grandezas em regime transitório.

Referências

[1] F. Fayyaz and C. Trueman, "Persistent mistakes in learning basic circuit analysis," *Proceedings of the Canadian Engineering Education Association (CEEA)*, Nov. 2022. [Citado na página 1]

[2] L. Sousa, "U=ri*s*olve: a web-based tool for teaching and self-learning the nodal voltage method," master's thesis, Instituto Superior de Engenharia do Porto, 2020. [Citado nas páginas 2 e 4]

[3] A. Pinheiro, "Project, implementation and integration of the mesh current method in the u=RIsolve application.," bachelor's thesis, Instituto Superior de Engenharia do Porto, 2021. [Citado nas páginas 2, 4 e 78]

[4] H. Carvalho, "Application for generating the methodology/equations of the mesh current method in the branches, using qucs.," bachelor's thesis, Instituto Superior de Engenharia do Porto, 2023. [Citado nas páginas 2, 4 e 78]

[5] G. Zenha, "Software application for generating the methodology/equations of the superposition theorem, using qucs.," bachelor's thesis, Instituto Superior de Engenharia do Porto, 2023. [Citado nas páginas 2 e 4]

[6] L. Sousa, A. Rocha, M. Alves, and F. Pereira, "U=RIsolve: A web-based application for learning electrical circuit analysis [education]," *IEEE Circuits and Systems Magazine*, vol. 21, no. 3, pp. 66–95, 2021. [Citado na página 2]

[7] P. Morim, "Circuit editor: Design, implementation and integration into the u=RIsolve web application.," bachelor's thesis, Instituto Superior de Engenharia do Porto, 2022. [Citado nas páginas 2 e 3]

[8] A. Castedo, "U=RIsolve academy: an online learning framework for learning circuit analysis.," bachelor's thesis, Instituto Superior de Engenharia do Porto, 2023. [Citado na página 8]

[9] O. Foundation, "Node.js: Javascript runtime built on chrome's v8 engine," 2009. Server-side JavaScript runtime environment. [Citado na página 9]

[10] M. Foundation, "Mariadb: Open source relational database," 2009. Community-developed fork of MySQL. [Citado nas páginas 10 e 11]

[11] Y. Katz and contributors, "Handlebars.js: Minimal templating on steroids," 2010. Logicless templating engine for JavaScript. [Citado na página 10]

[12] J. Deacon, "Model-view-controller (mvc) architecture," *Online][Citado em: 10 de março de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf*, vol. 28, p. 61, 2009. [Citado nas páginas vii e 15]

[13] A. Martín Erro, M. L. Martínez Muñeta, and A. A. Rodríguez Sevillano, "Exploring freehand drawing skills of engineering students as a support of visualization," *Education Sciences*, vol. 14, no. 6, 2024. [Citado na página 21]

[14] M. E. Brinson and S. Jahn, "Qucs: Schematic file format — quite universal circuit simulator (version 0.0.19 internal documentation)," 2013. Internal document describing schematic file format of Qucs. [Citado na página 22]

[15] M. Corporation and C. D. Systems, "Pspice: Personal simulation program with integrated circuit emphasis," 1984. Proprietary SPICE-based analog and mixed-signal circuit simulator by Cadence. [Citado na página 22]

[16] N. Team, "Ngspice: Open source spice simulator for electric and electronic circuits," 1999. Based on SPICE3f5, Cider1b1 and Xspice — part of the gEDA project. [Citado na página 22]

[17] I. Autodesk, "Eagle: Pcb design software," 2016. Schematic capture, PCB layout and routing tool. [Citado na página 25]

[18] K. Developers, "Kicad eda: Open source software suite for electronic design automation," 1992. Includes schematic editor, PCB layout, 3D viewer, and Gerber export. [Citado na página 25]

[19] A. LLC, "Altium designer: Pcb design and electronics cad software," 2005. Professional PCB design suite for schematic capture, simulation, and manufacturing. [Citado na página 25]

[20] A. Rocha, "Meshes-Finder library: Efficiently find meshes in electric circuits." `https://github.com/txroot/meshes-finder`, 2022. GitHub repository. [Citado na página 25]

[21] A. Rocha, M. Alves, L. Sousa, and F. Pereira, "U=RIsolve: An interactive web-based application for teaching and self-learning loop current and branch current methods." Manuscript in preparation., 2023. [Citado na página 25]

[22] A. Rocha, "Electric circuit analysis: Computational tools for learning." Report presented at the Planned Individual Study (PIS) course, 2021. [Citado na página 26]

[23] M. E. Brinson and S. Jahn, “Qucs: A gpl software package for circuit simulation, compact device modelling and circuit macromodelling from dc to rf and beyond,” 2009. Quite Universal Circuit Simulator (Qucs). [Citado na página 39]

[24] P. Falstad, “Falstad circuit simulator,” 2000. Interactive web-based circuit simulator. [Citado na página 39]

[25] I. MuseMaze, “Everycircuit: Animated interactive circuit simulator,” 2025. Online and mobile circuit simulation tool. [Citado na página 39]

[26] D. He and E. Cui, “Easyeda: Web-based eda tool suite,” 2013. Schematic capture, PCB layout and simulation online. [Citado na página 40]

[27] A. LLC, “Circuitmaker,” 2016. Free PCB and schematic tool for makers. [Citado na página 40]

[28] M. Engelhardt, “Ltspice: High-performance spice simulator, schematic capture and waveform viewer,” 1999. Freeware from Analog Devices / Linear Technology. [Citado na página 40]

[29] U.-R. Academy, “Editor antigo da plataforma u=risolve academy,” n.d. Editor antigo da plataforma “U=RIsolve Academy”. [Citado na página 40]

[30] J. Ltd., “diagrams.net (formerly draw.io),” 2011. Diagramming tool (flowcharts, schematics, etc.). [Citado na página 40]

[31] M. Corporation, “Microsoft visio,” 1992. Diagramming and flowchart tool. [Citado na página 40]

[32] W. W. W. C. (W3C), “Document object model (dom) level 4 specification,” 2021. Specification for interacting with HTML and XML documents via a structured object model. [Citado na página 43]

[33] W. W. W. C. (W3C), “Html standard: The canvas element,” 2024. Specification for the HTML ‘<canvas>‘ element for drawing graphics via scripting. [Citado na página 43]

[34] D. Abstract, “When to use a javascript canvas library or framework,” 2020. Discusses when to use a Canvas framework versus traditional HTML, CSS, and JavaScript or a DOM framework. [Citado na página 43]

[35] C. Train, “A virtual world - javascript course: Lesson 1 / 11 [spatial graph],” 2022. YouTube video, accessed 21 October 2025. [Citado na página 44]

[36] W. W. W. C. (W3C), “Windoworkerglobalscope.requestanimationframe() — web api,” 2024. Defines the ‘requestAnimationFrame()‘ method for scheduling animations in web browsers. [Citado na página 46]

[37] R. Ferrero, “Game - gerador automático de modelos de esquemáticos,” bachelor’s thesis, Instituto Superior de Engenharia do Porto, 2024. [Citado na página 51]

[38] A. Montagne, “Modified nodal analysis,” 2023. Capítulo do livro *Structured Electronics Design*, abordando a técnica de Análise Nodal Modificada para análise de redes elétricas. [Citado nas páginas 61 e 78]

Anexos

.1 Anexo A

Modelo do utilizador

```
const User = sequelize.define('User', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  surname: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  email: {
    type: DataTypes.STRING,
    allowNull: false,
    unique: true,
    validate: {
      isEmail: true,
    },
  },
  password: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  role: {
    type: DataTypes.ENUM('student', 'professor', 'admin'),
    allowNull: false,
    defaultValue: 'student',
  },
  session_token: {
    type: DataTypes.STRING,
    allowNull: true,
  },
  last_session: {
    type: DataTypes.DATE,
    allowNull: true,
  },
```

```
  reset_code: {
    type: DataTypes.STRING,
    allowNull: true,
  },
  createdAt: {
    type: DataTypes.DATE,
    allowNull: false,
    defaultValue: DataTypes.NOW,
  },
  updatedAt: {
    type: DataTypes.DATE,
    allowNull: false,
    defaultValue: DataTypes.NOW,
  },
});
```

***Controlador* referente ao utilizador**

```
// Define controller methods
const UserController = {
    addUser: async (req, res) => {
        try {
            const userData = req.body;

            // Check if user already exists in the database
            const existingUser = await User.findOne({ where: { email:
                userData.email } });
            if (existingUser) {
                return res.status(400).json({ error: 'User already
                    exists.' });
            }

            // Create new user
            const newUser = await User.create(userData);

            res.status(201).json({
                message: 'User created successfully.',
                data: newUser
            });
        } catch (error) {
            console.error('Error creating user:', error);
            res.status(500).json({error: 'An error occurred while
                creating the user.'});
```

```
23          }
24      },
25      getUsers: ... ,
26      updateUser: ... ,
27      deleteUser: ... ,
28  };
```

.2 Anexo B

Modelo de dados referente ao circuito da Figura 3.2

```
{
  "components": [
    {
      "id": "f384b4c919a070fa747",
      "position": {
        "rotation": 270,
        "x": -25,
        "y": 6.5,
        "z": 0
      },
      "type": "VDC",
      "terminals": [
        {
          "terminalId": "73d12bc719a070fa747",
          "position": {
            "xOffset": 0,
            "yOffset": 1.5
          },
          "terminalRole": "negative",
          "wire": {
            "id": "2deae93e19a07103b57",
            "terminalId": "77714d2719a07103b57"
          }
        },
        {
          "terminalId": "fee1e25419a070fa747",
          "position": {
            "xOffset": 0,
            "yOffset": -1.5
          },
          "terminalRole": "positive",
          "wire": {
            "id": "af68eaf519a071012c5",
            "terminalId": "a1045b2a19a071012c5"
          }
        }
      ],
      "attributes": [
        {
```

```
          "name": "name",
          "value": "E1",
          "visible": true,
          "type": "main"
        },
        {
          "name": "voltage",
          "value": 5,
          "unit": "V",
          "visible": true,
          "type": "main"
        },
        {
          "name": "nominal_power",
          "value": 30,
          "unit": "W",
          "visible": false,
          "type": "sub"
        },
        {
          "name": "max_power",
          "value": 40,
          "unit": "W",
          "visible": false,
          "type": "sub"
        },
        {
          "name": "max_current",
          "value": 2.5,
          "unit": "A",
          "visible": false,
          "type": "sub"
        },
        {
          "name": "ideal",
          "value": true,
          "visible": false,
          "type": "sub"
        },
        {
          "name": "internal_resistance",
          "value": 0,
          "unit": "ohm",
          "visible": false,
```

```
        "type": "sub"
      }
    ],
    "label": {
      "position": {
        "xOffset": 31.200000000000006,
        "yOffset": 0,
        "rotation": 0
      },
      "origin": {
        "xOrigin": -650.0000000000002,
        "yOrigin": 169.00000000000006
      }
    }
  },
  {
    "id": "21a9394219a070fcca1",
    "position": {
      "rotation": 90,
      "x": -18.5,
      "y": 6.5,
      "z": 1
    },
    "type": "R",
    "terminals": [
      {
        "terminalId": "18840cb919a070fcca1",
        "position": {
          "xOffset": 0,
          "yOffset": 1.5
        },
        "wire": {
          "id": "a5ec938f19a07102b01",
          "terminalId": "d3d6d54519a07102b01"
        }
      },
      {
        "terminalId": "b8327ef019a070fcca1",
        "position": {
          "xOffset": 0,
          "yOffset": -1.5
        },
        "wire": {
          "id": "2bff555419a071012c5",
```

```
          "terminalId": "0198c82919a071012c5"
        }
      }
    ],
    "attributes": [
      {
        "name": "name",
        "value": "R1",
        "visible": true,
        "type": "main"
      },
      {
        "name": "resistance",
        "value": 1,
        "unit": "kohm",
        "visible": true,
        "type": "main"
      },
      {
        "name": "nominal_power",
        "value": 0.25,
        "unit": "W",
        "visible": false,
        "type": "sub"
      },
      {
        "name": "tolerance",
        "value": 0,
        "unit": "%",
        "visible": false,
        "type": "sub"
      },
      {
        "name": "temperature",
        "value": 20,
        "unit": "C",
        "visible": false,
        "type": "sub"
      },
      {
        "name": "temperature_coefficient",
        "value": 0,
        "unit": "ohm/C",
        "visible": false,
```

```
        "type": "sub"
      }
    ],
    "label": {
      "position": {
        "xOffset": 31.200000000000006,
        "yOffset": 0,
        "rotation": 0
      },
      "origin": {
        "xOrigin": -481.0000000000001,
        "yOrigin": 169.00000000000006
      }
    }
  },
  {
    "id": "52552c5c19a070fd356",
    "position": {
      "rotation": 90,
      "x": -12.5,
      "y": 6.5,
      "z": 2
    },
    "type": "R",
    "terminals": [
      {
        "terminalId": "60ce05cd19a070fd356",
        "position": {
          "xOffset": 0,
          "yOffset": 1.5
        },
        "wire": {
          "id": "edeb352919a07102aff",
          "terminalId": "707e49ec19a07102aff"
        }
      },
      {
        "terminalId": "98fa753e19a070fd356",
        "position": {
          "xOffset": 0,
          "yOffset": -1.5
        },
        "wire": {
          "id": "5cf1bf6819a07107aa5",
```

```
          "terminalId": "03f9a22219a07107aa5"
        }
      }
    ],
    "attributes": [
      {
        "name": "name",
        "value": "R2",
        "visible": true,
        "type": "main"
      },
      {
        "name": "resistance",
        "value": 1,
        "unit": "kohm",
        "visible": true,
        "type": "main"
      },
      {
        "name": "nominal_power",
        "value": 0.25,
        "unit": "W",
        "visible": false,
        "type": "sub"
      },
      {
        "name": "tolerance",
        "value": 0,
        "unit": "%",
        "visible": false,
        "type": "sub"
      },
      {
        "name": "temperature",
        "value": 20,
        "unit": "C",
        "visible": false,
        "type": "sub"
      },
      {
        "name": "temperature_coefficient",
        "value": 0,
        "unit": "ohm/C",
        "visible": false,
```

```
        "type": "sub"
      }
    ],
    "label": {
      "position": {
        "xOffset": 31.200000000000006,
        "yOffset": 0,
        "rotation": 0
      },
      "origin": {
        "xOrigin": -325.0000000000001,
        "yOrigin": 169.00000000000006
      }
    }
  },
  {
    "id": "6a7ce23019a07103b55",
    "position": {
      "rotation": 0,
      "x": -24,
      "y": 12,
      "z": 3
    },
    "type": "GND",
    "terminals": [
      {
        "terminalId": "32a6264419a07103b55",
        "position": {
          "xOffset": 0,
          "yOffset": -1.5
        },
        "connectionPoint": "6c2d60a719a07107aa5"
      }
    ],
    "attributes": [
      {
        "name": "name",
        "value": "GND",
        "visible": true,
        "type": "main"
      }
    ],
    "label": {
      "position": {
```

```
          "xOffset": 0,
          "yOffset": 31.200000000000006,
          "rotation": 0
        },
        "origin": {
          "xOrigin": -624.0000000000002,
          "yOrigin": 312.0000000000001
        }
      }
    },
    {
      "id": "a6a50ffe19a07107aa3",
      "position": {
        "rotation": 0,
        "x": -17.5,
        "y": 3,
        "z": 4
      },
      "type": "FLAG",
      "terminals": [
        {
          "terminalId": "b1da84b619a07107aa3",
          "position": {
            "xOffset": 0,
            "yOffset": 0
          },
          "connectionPoint": "7d0f5c5719a07107aa5"
        }
      ],
      "attributes": [
        {
          "name": "name",
          "value": "A",
          "visible": true,
          "type": "main"
        }
      ]
    }
  ],
  "wires": [
    {
      "id": "af68eaf519a071012c5",
      "properties": {
        "stroke": 3.5,
```

```
348         "dashed": false,
349         "color": "hsl(855, 90%, 45%)",
350         "decoration": {}
351       },
352       "attributes": [
353         {
354           "name": "internal_resistance",
355           "value": 0,
356           "unit": "ohm",
357           "visible": false
358         },
359         {
360           "name": "temperature",
361           "value": 20,
362           "unit": "C",
363           "visible": false
364         },
365         {
366           "name": "material",
367           "value": "default",
368           "visible": false
369         },
370         {
371           "name": "length",
372           "value": 0,
373           "unit": "m",
374           "visible": false
375         },
376         {
377           "name": "cross_section",
378           "value": 0,
379           "unit": "mm2",
380           "visible": false
381         },
382         {
383           "name": "resistivity",
384           "value": 0,
385           "unit": "ohmmm2/m",
386           "visible": false
387         },
388         {
389           "name": "temperature_coefficient",
390           "value": 0,
391           "unit": "ohm/C",
```

```
392           "visible": false
393         }
394       ],
395       "startTerminal": {
396         "terminalId": "a1045b2a19a071012c5",
397         "position": {
398           "x": -25,
399           "y": 5
400         },
401         "component": {
402           "id": "f384b4c919a070fa747",
403           "terminalId": "fee1e25419a070fa747"
404         }
405       },
406       "endTerminal": {
407         "terminalId": "9e035fbe19a071012c5",
408         "position": {
409           "x": -18.5,
410           "y": 3
411         },
412         "connectionPoint": "8e21164719a07107aa5"
413       },
414       "vertices": [
415         {
416           "x": -25,
417           "y": 3
418         }
419       ]
420     },
421     {
422       "id": "2bff555419a071012c5",
423       "properties": {
424         "stroke": 3.5,
425         "dashed": false,
426         "color": "hsl(900, 90%, 45%)",
427         "decoration": {}
428       },
429       "attributes": [
430         {
431           "name": "internal_resistance",
432           "value": 0,
433           "unit": "ohm",
434           "visible": false
435         },
```

```
        {
          "name": "temperature",
          "value": 20,
          "unit": "C",
          "visible": false
        },
        {
          "name": "material",
          "value": "default",
          "visible": false
        },
        {
          "name": "length",
          "value": 0,
          "unit": "m",
          "visible": false
        },
        {
          "name": "cross_section",
          "value": 0,
          "unit": "mm2",
          "visible": false
        },
        {
          "name": "resistivity",
          "value": 0,
          "unit": "ohmmm2/m",
          "visible": false
        },
        {
          "name": "temperature_coefficient",
          "value": 0,
          "unit": "ohm/C",
          "visible": false
        }
      ],
      "startTerminal": {
        "terminalId": "83ea1ddc19a071012c5",
        "position": {
          "x": -18.5,
          "y": 3
        },
        "connectionPoint": "8e21164719a07107aa5"
      },
```

```
      "endTerminal": {
        "terminalId": "0198c82919a071012c5",
        "position": {
          "x": -18.5,
          "y": 5
        },
        "component": {
          "id": "21a9394219a070fcca1",
          "terminalId": "b8327ef019a070fcca1"
        }
      },
      "vertices": []
    },
    {
      "id": "edeb352919a07102aff",
      "properties": {
        "stroke": 3.5,
        "dashed": false,
        "color": "hsl(945, 90%, 45%)",
        "decoration": {}
      },
      "attributes": [
        {
          "name": "internal_resistance",
          "value": 0,
          "unit": "ohm",
          "visible": false
        },
        {
          "name": "temperature",
          "value": 20,
          "unit": "C",
          "visible": false
        },
        {
          "name": "material",
          "value": "default",
          "visible": false
        },
        {
          "name": "length",
          "value": 0,
          "unit": "m",
          "visible": false
```

```
524         },
525         {
526           "name": "cross_section",
527           "value": 0,
528           "unit": "mm2",
529           "visible": false
530         },
531         {
532           "name": "resistivity",
533           "value": 0,
534           "unit": "ohmmm2/m",
535           "visible": false
536         },
537         {
538           "name": "temperature_coefficient",
539           "value": 0,
540           "unit": "ohm/C",
541           "visible": false
542         }
543       ],
544       "startTerminal": {
545         "terminalId": "ddf99f9a19a07102aff",
546         "position": {
547           "x": -18.5,
548           "y": 10.5
549         },
550         "connectionPoint": "ce64fd5c19a07107aa5"
551       },
552       "endTerminal": {
553         "terminalId": "707e49ec19a07102aff",
554         "position": {
555           "x": -12.5,
556           "y": 8
557         },
558         "component": {
559           "id": "52552c5c19a070fd356",
560           "terminalId": "60ce05cd19a070fd356"
561         }
562       },
563       "vertices": [
564         {
565           "x": -12.5,
566           "y": 10.5
567         }
```

```
      ]
    },
    {
      "id": "a5ec938f19a07102b01",
      "properties": {
        "stroke": 3.5,
        "dashed": false,
        "color": "hsl(990, 90%, 45%)",
        "decoration": {}
      },
      "attributes": [
        {
          "name": "internal_resistance",
          "value": 0,
          "unit": "ohm",
          "visible": false
        },
        {
          "name": "temperature",
          "value": 20,
          "unit": "C",
          "visible": false
        },
        {
          "name": "material",
          "value": "default",
          "visible": false
        },
        {
          "name": "length",
          "value": 0,
          "unit": "m",
          "visible": false
        },
        {
          "name": "cross_section",
          "value": 0,
          "unit": "mm2",
          "visible": false
        },
        {
          "name": "resistivity",
          "value": 0,
          "unit": "ohmmm2/m",
```

```
          "visible": false
        },
        {
          "name": "temperature_coefficient",
          "value": 0,
          "unit": "ohm/C",
          "visible": false
        }
      ],
      "startTerminal": {
        "terminalId": "4505ab2319a07102b01",
        "position": {
          "x": -18.5,
          "y": 10.5
        },
        "connectionPoint": "ce64fd5c19a07107aa5"
      },
      "endTerminal": {
        "terminalId": "d3d6d54519a07102b01",
        "position": {
          "x": -18.5,
          "y": 8
        },
        "component": {
          "id": "21a9394219a070fcca1",
          "terminalId": "18840cb919a070fcca1"
        }
      },
      "vertices": []
    },
    {
      "id": "2deae93e19a07103b57",
      "properties": {
        "stroke": 3.5,
        "dashed": false,
        "color": "hsl(1035, 90%, 45%)",
        "decoration": {}
      },
      "attributes": [
        {
          "name": "internal_resistance",
          "value": 0,
          "unit": "ohm",
          "visible": false
```

```
656         },
657         {
658           "name": "temperature",
659           "value": 20,
660           "unit": "C",
661           "visible": false
662         },
663         {
664           "name": "material",
665           "value": "default",
666           "visible": false
667         },
668         {
669           "name": "length",
670           "value": 0,
671           "unit": "m",
672           "visible": false
673         },
674         {
675           "name": "cross_section",
676           "value": 0,
677           "unit": "mm2",
678           "visible": false
679         },
680         {
681           "name": "resistivity",
682           "value": 0,
683           "unit": "ohmmm2/m",
684           "visible": false
685         },
686         {
687           "name": "temperature_coefficient",
688           "value": 0,
689           "unit": "ohm/C",
690           "visible": false
691         }
692       ],
693       "startTerminal": {
694         "terminalId": "77714d2719a07103b57",
695         "position": {
696           "x": -25,
697           "y": 8
698         },
699         "component": {
```

```
700           "id": "f384b4c919a070fa747",
701           "terminalId": "73d12bc719a070fa747"
702         }
703       },
704       "endTerminal": {
705         "terminalId": "cea5610819a07103b57",
706         "position": {
707           "x": -24,
708           "y": 10.5
709         },
710         "connectionPoint": "6c2d60a719a07107aa5"
711       },
712       "vertices": [
713         {
714           "x": -25,
715           "y": 10.5
716         }
717       ]
718     },
719     {
720       "id": "d779b91819a07103b57",
721       "properties": {
722         "stroke": 3.5,
723         "dashed": false,
724         "color": "hsl(1080, 90%, 45%)",
725         "decoration": {}
726       },
727       "attributes": [
728         {
729           "name": "internal_resistance",
730           "value": 0,
731           "unit": "ohm",
732           "visible": false
733         },
734         {
735           "name": "temperature",
736           "value": 20,
737           "unit": "C",
738           "visible": false
739         },
740         {
741           "name": "material",
742           "value": "default",
743           "visible": false
```

```
        },
        {
          "name": "length",
          "value": 0,
          "unit": "m",
          "visible": false
        },
        {
          "name": "cross_section",
          "value": 0,
          "unit": "mm2",
          "visible": false
        },
        {
          "name": "resistivity",
          "value": 0,
          "unit": "ohmmm2/m",
          "visible": false
        },
        {
          "name": "temperature_coefficient",
          "value": 0,
          "unit": "ohm/C",
          "visible": false
        }
      ],
      "startTerminal": {
        "terminalId": "62dd68e919a07103b57",
        "position": {
          "x": -24,
          "y": 10.5
        },
        "connectionPoint": "6c2d60a719a07107aa5"
      },
      "endTerminal": {
        "terminalId": "c9dabb0819a07103b57",
        "position": {
          "x": -18.5,
          "y": 10.5
        },
        "connectionPoint": "ce64fd5c19a07107aa5"
      },
      "vertices": []
    },
```

```
    {
      "id": "f28d7eb019a07107aa5",
      "properties": {
        "stroke": 3.5,
        "dashed": false,
        "color": "hsl(1125, 90%, 45%)",
        "decoration": {}
      },
      "attributes": [
        {
          "name": "internal_resistance",
          "value": 0,
          "unit": "ohm",
          "visible": false
        },
        {
          "name": "temperature",
          "value": 20,
          "unit": "C",
          "visible": false
        },
        {
          "name": "material",
          "value": "default",
          "visible": false
        },
        {
          "name": "length",
          "value": 0,
          "unit": "m",
          "visible": false
        },
        {
          "name": "cross_section",
          "value": 0,
          "unit": "mm2",
          "visible": false
        },
        {
          "name": "resistivity",
          "value": 0,
          "unit": "ohmmm2/m",
          "visible": false
        },
```

```
832         {
833           "name": "temperature_coefficient",
834           "value": 0,
835           "unit": "ohm/C",
836           "visible": false
837         }
838       ],
839       "startTerminal": {
840         "terminalId": "9a153c8419a07107aa5",
841         "position": {
842           "x": -18.5,
843           "y": 3
844         },
845         "connectionPoint": "8e21164719a07107aa5"
846       },
847       "endTerminal": {
848         "terminalId": "accb099a19a07107aa5",
849         "position": {
850           "x": -17.5,
851           "y": 3
852         },
853         "connectionPoint": "7d0f5c5719a07107aa5"
854       },
855       "vertices": []
856     },
857     {
858       "id": "5cf1bf6819a07107aa5",
859       "properties": {
860         "stroke": 3.5,
861         "dashed": false,
862         "color": "hsl(1170, 90%, 45%)",
863         "decoration": {}
864       },
865       "attributes": [
866         {
867           "name": "internal_resistance",
868           "value": 0,
869           "unit": "ohm",
870           "visible": false
871         },
872         {
873           "name": "temperature",
874           "value": 20,
875           "unit": "C",
```

```
          "visible": false
        },
        {
          "name": "material",
          "value": "default",
          "visible": false
        },
        {
          "name": "length",
          "value": 0,
          "unit": "m",
          "visible": false
        },
        {
          "name": "cross_section",
          "value": 0,
          "unit": "mm2",
          "visible": false
        },
        {
          "name": "resistivity",
          "value": 0,
          "unit": "ohmmm2/m",
          "visible": false
        },
        {
          "name": "temperature_coefficient",
          "value": 0,
          "unit": "ohm/C",
          "visible": false
        }
      ],
      "startTerminal": {
        "terminalId": "1181e2a919a07107aa5",
        "position": {
          "x": -17.5,
          "y": 3
        },
        "connectionPoint": "7d0f5c5719a07107aa5"
      },
      "endTerminal": {
        "terminalId": "03f9a22219a07107aa5",
        "position": {
          "x": -12.5,
```

```
          "y": 5
        },
        "component": {
          "id": "52552c5c19a070fd356",
          "terminalId": "98fa753e19a070fd356"
        }
      },
      "vertices": [
        {
          "x": -12.5,
          "y": 3
        }
      ]
    }
  ],
  "connectionPoints": [
    {
      "id": "6c2d60a719a07107aa5",
      "properties": {
        "color": "black",
        "stroke": 5
      },
      "position": {
        "x": -24,
        "y": 10.5
      },
      "wires": [
        {
          "id": "d779b91819a07103b57",
          "terminalId": "62dd68e919a07103b57"
        },
        {
          "id": "2deae93e19a07103b57",
          "terminalId": "cea5610819a07103b57"
        }
      ],
      "components": [
        {
          "id": "6a7ce23019a07103b55",
          "terminalId": "32a6264419a07103b55"
        }
      ],
      "label": null
    },
```

```
964     {
965       "id": "7d0f5c5719a07107aa5",
966       "properties": {
967         "color": "black",
968         "stroke": 5
969       },
970       "position": {
971         "x": -17.5,
972         "y": 3
973       },
974       "wires": [
975         {
976           "id": "5cf1bf6819a07107aa5",
977           "terminalId": "1181e2a919a07107aa5"
978         },
979         {
980           "id": "f28d7eb019a07107aa5",
981           "terminalId": "accb099a19a07107aa5"
982         }
983       ],
984       "components": [
985         {
986           "id": "a6a50ffe19a07107aa3",
987           "terminalId": "b1da84b619a07107aa3"
988         }
989       ],
990       "label": null
991     },
992     {
993       "id": "8e21164719a07107aa5",
994       "properties": {
995         "color": "black",
996         "stroke": 5
997       },
998       "position": {
999         "x": -18.5,
1000        "y": 3
1001      },
1002      "wires": [
1003        {
1004          "id": "2bff555419a071012c5",
1005          "terminalId": "83ea1ddc19a071012c5"
1006        },
1007        {
```

```
          "id": "f28d7eb019a07107aa5",
          "terminalId": "9a153c8419a07107aa5"
        },
        {
          "id": "af68eaf519a071012c5",
          "terminalId": "9e035fbe19a071012c5"
        }
      ],
      "components": [],
      "label": null
    },
    {
      "id": "ce64fd5c19a07107aa5",
      "properties": {
        "color": "black",
        "stroke": 5
      },
      "position": {
        "x": -18.5,
        "y": 10.5
      },
      "wires": [
        {
          "id": "edeb352919a07102aff",
          "terminalId": "ddf99f9a19a07102aff"
        },
        {
          "id": "a5ec938f19a07102b01",
          "terminalId": "4505ab2319a07102b01"
        },
        {
          "id": "d779b91819a07103b57",
          "terminalId": "c9dabb0819a07103b57"
        }
      ],
      "components": [],
      "label": null
    }
  ],
  "metadata": {
    "simulation": {},
    "nodes": [],
    "branches": []
  }
```

```
1052 }
```

.3 Anexo C

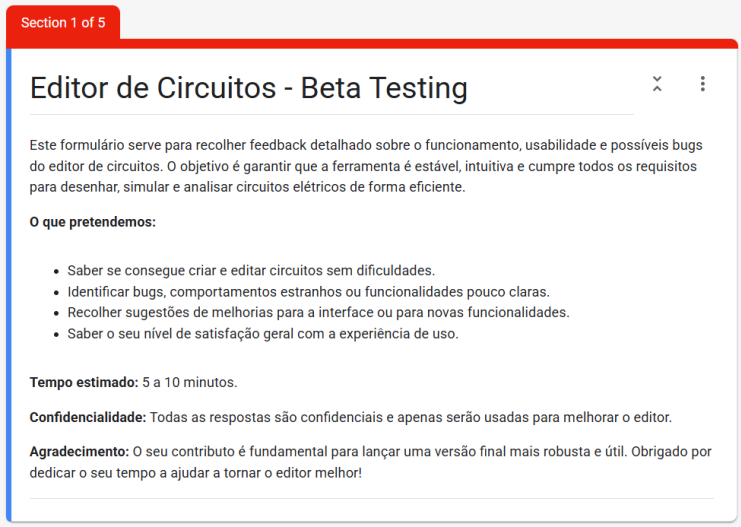
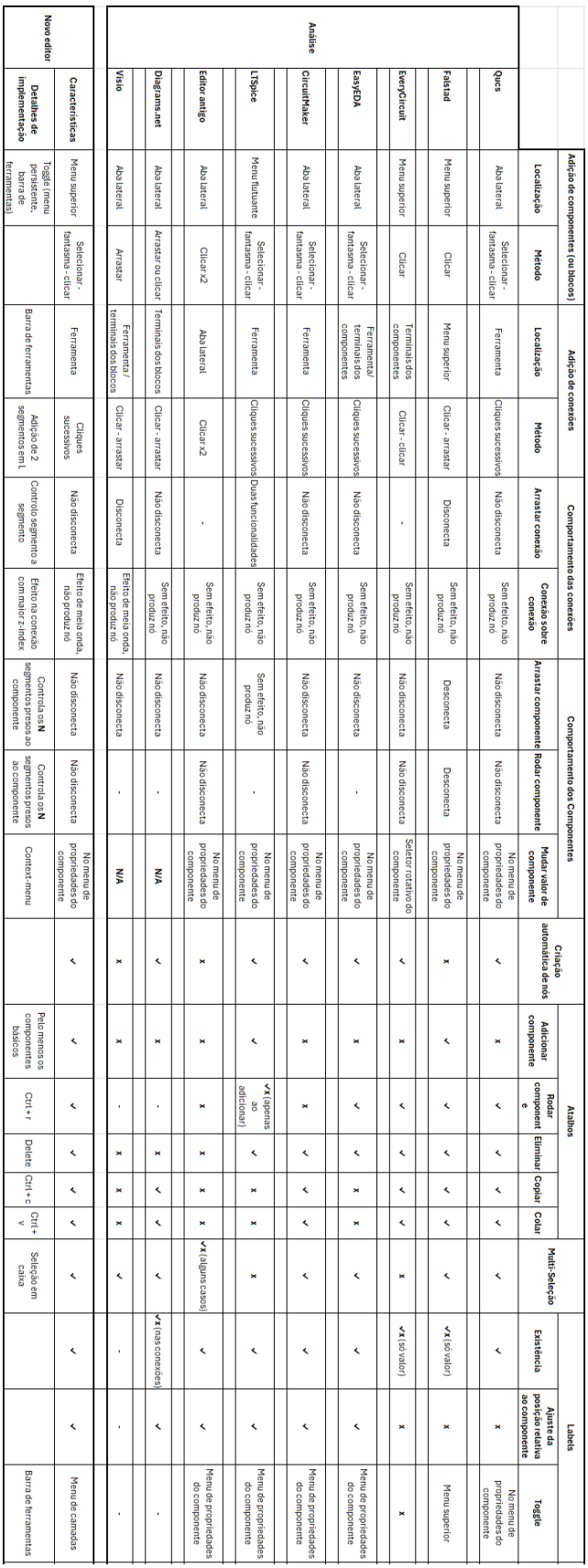


Figura 1: Cabeçalho do formulário de UI e UX desenvolvido

.4 Anexo D



		Adição de componentes (ou blocos)		Adição de conexões		Comportamento das conexões		Comportamento dos Componentes			Atalhos					Multi-Seleção	Labels		
		Localização	Método	Localização	Método	Arrastar conexão	Conexão sobre conexão	Arrastar componente	Rodar componente	Mudar valor de componente	Criação automática de nos	Adicionar componente	Rodar componente	Eliminar	Copiar	Colar	Seleção em caixa	Existência	Ajuste da posição relativa ao componente	Toggle
Análise	Qucs	Aba lateral	Selecionar - fantasma - clicar	Ferramenta	Cliques sucessivos	Não disconecta	Sem efeito, não produz nó	Não disconecta	Não disconecta	No menu de propriedades do componente	✓	x	✓	✓	✓	✓	✓	✓	x	No menu de propriedades do componente
	Falstad	Menu superior	Clicar	Menu superior	Clicar - arrastar	Disconecta	Sem efeito, não produz nó	Desconecta	Desconecta	No menu de propriedades do componente	x	✓	✓	✓	✓	✓	✓	✓x1 (só valor)	x	Menu superior
	EveryCircuit	Menu superior	Clicar	Terminais dos componentes	Clicar - clicar	-	Sem efeito, não produz nó	Não disconecta	Não disconecta	Seletor rotativo do componente	✓	x	✓	✓	✓	x	✓x1 (só valor)	x	x	
	EasyEDA	Aba lateral	Selecionar - fantasma - clicar	Ferramenta/ terminais dos componentes	Cliques sucessivos	Não disconecta	Sem efeito, não produz nó	Não disconecta	-	No menu de propriedades do componente	✓	x	✓	x	x	✓	✓	✓	✓	Menu de propriedades do componente
	CircuitMaker	Aba lateral	Selecionar - fantasma - clicar	Ferramenta	Cliques sucessivos	Não disconecta	Sem efeito, não produz nó	Não disconecta	-	No menu de propriedades do componente	✓	x	✓	✓	✓	✓	✓	✓	✓	Menu de propriedades do componente
	LTSpice	Menu flutuante	Selecionar - fantasma - clicar	Ferramenta	Cliques sucessivos	Duas funcionalidades	Sem efeito, não produz nó	Sem efeito, não produz nó	-	No menu de propriedades do componente	✓	✓x1 (apenas ao adicionar)	✓	✓	✓	✓	✓	✓	✓	Menu de propriedades do componente
	Editor antigo	Aba lateral	Clicar x2	Aba lateral	Clicar x2	-	Sem efeito, não produz nó	Não disconecta	Não disconecta	No menu de propriedades do componente	x	x	x	x	x	x	✓x1 (alguns casos)	✓	✓	Menu de propriedades do componente
	Diagrams.net	Aba lateral	Arrastar ou clicar	Terminais dos blocos	Clicar - arrastar	Não disconecta	Sem efeito, não produz nó	Não disconecta	-	N/A	✓	x	-	x	✓	✓	✓x1 (nas conexões)	✓	✓	-
	Visio	Aba lateral	Arrastar	Ferramenta/ terminais dos blocos	Clicar - arrastar	Disconecta	Efeito de meia onda, não produz nó	Não disconecta	-	N/A	x	x	-	x	x	✓	-	-	-	-
Novo editor		Menu superior	Selecionar - fantasma - clicar	Ferramenta	Cliques sucessivos	Não disconecta	Efeito de meia onda, não produz nó	Não disconecta	Não disconecta	No menu de propriedades do componente	✓	✓	✓	✓	✓	✓	✓	✓	✓	Menu de camadas
	Detalhes de implementação	Toggle (menu persistente, barra de ferramentas)		Barra de ferramentas	Adição de 2 segmentos em L	Controlo segmento a segmento	Efeito na conexão com maior z-index	Controla os N segmentos presos ao componente	Controla os N segmentos presos ao componente	Context-menu		Pelo menos os componentes básicos	Ctrl + r	Delete	Ctrl + c	Ctrl + v				Barra de ferramentas
	Características																			

Figura 2: Tabela comparativa completa dos diferentes editores de circuitos

		Qucs		Falstad		EveryCircuit		easyEDA		CircuitMaker		LTSpice		Editor antigo		Novo editor	
		Existência	Funcionamento	Existência	Funcionamento	Existência	Funcionamento	Existência	Funcionamento	Existência	Funcionamento	Existência	Funcionamento	Existência	Funcionamento	Existência	Funcionamento
Componentes passivos	Resistência	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	Clicar 2x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
	Reactância	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	-	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	Clicar 2x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
	Condensador	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	Clicar 2x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
	Bobina	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	Clicar 2x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
	Díodo	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	Clicar 2x	-	Menu superior	Selecionar - fantasma - clicar
Componentes ativos	Transistor (BJT)	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	-	Menu superior	-
	Transistor (FET)	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	-	Menu superior	-
	Transistor (MOSFET)	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	-	Menu superior	-
	AMPOP	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	-	Menu superior	-
	LED	x	-	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	-	Menu superior	-
Fontes	Tensão CC	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
	Tensão CA	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
	Corrente CC	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
	Corrente CA	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
Aparelhos	Voltímetro	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
	Amperímetro	No menu lateral	Selecionar - fantasma - clicar	No menu superior	Selecionar e colocar no circuito	No menu superior	Clicar 2x	No menu lateral	Selecionar - fantasma - clicar	No menu lateral	Selecionar - fantasma - clicar	Selecionar - fantasma - clicar	No menu lateral	x	Clicar 2x	Menu superior	Selecionar - fantasma - clicar
	Osciloscópio	x	-	✓x	Existe-lhe no menu de propriedades do componente. Gráfico de tensão e corrente no componente. Opção de "ver na bancada" ou no circuito	Com o componente selecionado, existe na parte inferior do editor	Clicar no símbolo de visualizar abre um osciloscópio na parte superior do editor	Nas ferramentas de	Selecionar - fantasma - clicar	x	Aparece após executar a simulação	Executar simulação e adicionar a grandeza ao gráfico para ser visualizada	-	-	-	Barra de ferramentas	As formas de onda aparecerão no osciloscópio, cujas portas de prova se movimentam para o ponto do circuito
	Dipolo	x	-	x	-	x	-	x	-	x	-	-	Selecionar - fantasma - clicar	x	-	Barra de ferramentas	Selecionar - fantasma - clicar
	Etiqueta	Na barra de ferramentas	Selecionar e clicar num nó	x	-	x	-	x	Selecionar - fantasma - clicar	x	Menu flutuante	Selecionar - fantasma - clicar	No menu lateral	Clicar 2x	-	Barra de ferramentas	Selecionar - fantasma - clicar

Figura 3: Tabela comparativa completa dos diferentes editores de circuitos