SEE APP - Aplicação de Uso Pedagógico para Disciplinas de Subestações de Energia Elétrica

Laís Brum Menezes [*] **Cristiane Cauduro Gastaldini** [**]
Laura Lisiane Callai dos Santos [***]

Engenharia Elétrica campus Cachoeira do Sul, Grupo de Processamento de Energia e Sistemas de Potência - GPSIS, Universidade Federal de Santa Maria

[*] *lais.brum@acad.ufsm.br.*
[**] *cristiane.gastaldini@ufsm.br*
[***] *laura.santos@ufsm.br*

Abstract: The present work aims at the development of a didactic tool that serves as support to the academics of the discipline of Electric Power Substations. The application will provide the user with demonstrations of maneuvers in the most common busbar arrangements, facilitating and providing an understanding of the different equipment that make up a substation. It was decided to use the Tkinter package, which comes with the official distribution of the Python free license interpreter, being responsible for the graphical user interface. It is intended to carry out a quality assessment of the application with the students of the discipline. The application will be made available for download, for personal and/or academic use, through the GitHub.com hosting platform, where it provides access control and various collaboration features.

Resumo: O presente trabalho objetiva o desenvolvimento de uma ferramenta didática que sirva como suporte aos acadêmicos da disciplina de Subestações de Energia Elétrica. A aplicação proporcionará ao usuário demonstrações de manobras em arranjos de barramentos mais comuns, facilitando e proporcionando o entendimento dos diferentes equipamentos que compõem uma subestação. Decidiu-se a utilização do pacote Tkinter, que acompanha a distribuição oficial do interpretador de licença livre Python, sendo responsável pela interface gráfica com o usuário. Pretende-se realizar uma avaliação de qualidade da aplicação com os discentes da disciplina. O aplicativo será disponibilizado para *download*, para uso pessoal e/ou acadêmico, através da plataforma de hospedagem GitHub.com, onde fornece controle de acesso e vários recursos de colaboração.

Keywords: Electric Power Substations; Maneuvers in Buses; Pedagogical application; Python language; Tkinter package;

Palavras-chaves: Subestações de Energia Elétrica; Manobras em Barramentos; Aplicação pedagógica; Linguagem Python; Pacote Tkinter;

1. INTRODUÇÃO

Segundo Macedo et al. (2012), comunicação e educação estão entrelaçadas no mundo digital. Portanto, professores e estudantes precisam aproveitar adequadamente os recursos dessas novas tecnologias, explorar seu potencial pedagógico e fazer uso positivo desses novos ambientes de ensino e aprendizagem.

As ferramentas pedagógicas geralmente servem para facilitar o processo de aprendizagem e esse termo depende da intenção e propósito de quem as utiliza e contribui para a educação efetiva do estudante. Uma das formas é a utilização de softwares educacionais que corroboram na compreensão de conceitos desenvolvidos em aula.

O objetivo deste trabalho é desenvolver um software educacional que atenda a disciplina de Subestações de Energia Elétrica. Atualmente, a utilização de *softwares* de simulação não didáticos, como o *MGA Power Simulator*, ou que não são desenvolvidos para esta finalidade, como o *CADe SIMU*; trouxe a motivação para a implementação de uma ferramenta pedagógica que atendesse as necessidades básicas desta disciplina. Verificou-se em pesquisa sobre o tema, o fato das tecnologias nesta área serem escassas e/ou não atenderem às necessidades da disciplina.

A seguir, são apresentados alguns *softwares* didáticos já existentes.

1.1 Ferramenta Didática de Subestações Elétricas SUEL

Dada a importância do aprendizado, entendimento e a necessidade de realizar um projeto de subestação, foi criada por Holanda (2016) a *Ferramenta Didática Subestação Elétrica SUEL*, que visa agregar de forma prática e fácil as informações relevantes sobre os diferentes tipos de arranjos de subestações dessas plantas. O programa *Adobe Animate CC 2015*, produzido pela empresa norte-americana *Adobe*

Systems Incorporated, foi utilizado para desenvolver a ferramenta didática.

De acordo com Holanda (2016), *Adobe Animate CC* é uma plataforma de desenvolvimento baseada em linha do tempo para criar animações vetoriais, conteúdo multimídia, aplicativos e jogos; Possui um ambiente gráfico com ferramentas de desenho e ilustração e um ambiente de programação que permite agregar interatividade e manipulação de dados ao conteúdo desenvolvido.

1.2 FEUPowerTool: ferramenta pedagógica para manobras em subestações

Segundo Ramos (2010), a criação de um aplicativo didático para simulação de manobras de subestações decorre do fato de que as tecnologias de auxílio e base para treinamento nesta área são escassas e desatualizadas.

Havia a necessidade de desenvolver uma ferramenta que apresentasse mais liberdade tanto ao usuário quanto ao programador. Essa liberdade só pode ser alcançada com uma ambiente de desenvolvimento integrado (IDE - *Integrated Development Environment*). Assim, é possível desenvolver uma interface gráfica do usuário (GUI - *Graphical User Interface*) que possibilita a criação de qualquer circuito, que distingue um seccionador de um disjuntor e detecta as manobras efetuadas durante a simulação (RAMOS, 2010).

De acordo com a página oficial da Lazarus and Free Pascal Team, *Lazarus* é um IDE compatível com multiplataforma *Delphi* para *Free Pascal*. *Pascal* é um compilador de Licença Pública Geral (GLP - *General Public License*) projetado para ser capaz de entender e compilar a sintaxe *Delphi*, que é uma Programação Orientada a Objetos (OOP - *Object-Oriented Programming*). Foi este o programa escolhido, para o desenvolvimento da aplicação (RAMOS, 2010).

1.3 Simulador de uma subestação elétrica para ensino de princípios básicos de eletricidade

O trabalho de Silva (2017) propõe o desenvolvimento de um simulador com base no sistema de transmissão de energia da Eletrobrás/Eletronorte, onde o estudante poderá, através de uma tela simulada do Sistema Aberto de Gerenciamento de Energia (SAGE), fazer operações com os disjuntores, com abertura e fechamento de carga. Está integrado em uma plataforma *Arduino*, onde interpreta os comandos do *software* do simulador e os converte em sinais analógicos, que acionam o diodo emissor de luz (LED - *Light Emitting Diode*) que representa a continuidade da carga, em caso de ativação, ou não acionam, em caso de desativação para um centro urbano específico.

A ferramenta utilizada para o desenvolvimento do projeto foi o ambiente 2D da plataforma *Unity*, que segundo Silva (2017), possibilita a criação de jogos e simuladores em 2D, oferecendo uma interface muito simples e amigável, com o objetivo de facilitar o desenvolvimento de jogos ou simuladores de diversos tipos, bem como outros sistemas de visualização.

1.4 Aplicação Informática para Dimensionamento de Barramentos em Subestações

O trabalho de Tavares (2015) propõe a construção de um aplicativo computacional para realizar automaticamente os cálculos necessários para validar os barramentos selecionados pelo usuário.

Utilizou-se o *software Microsoft Access* para a construção da aplicação de dimensionamento dos barramentos. Segundo Tavares (2015), esta escolha deve-se ao fato de as características do *software* irem exatamente ao encontro dos objetivos propostos para a aplicação, pois trata-se de um *software* de criação e gestão de base de dados, amplamente utilizado no mercado, de fácil interação com o utilizador. A configuração de formulários e programas de ações é realizada em linguagem *Visual Basic for Applications* (VBA) e *Structured Query Language* (SQL).

1.5 Automação de Manobras em Subestações de Transmissão de Energia Elétrica

O trabalho de Dias (2017) propõe uma estratégia baseada na automação de etapas da tarefa para minimizar o erro durante a realização de manobras em um Sistema Elétrico de Potência (SEP), implementada através do desenvolvimento de uma ferramenta de *software*.

Para o desenvolvimento das funcionalidades da interface foi utilizada a linguagem de programação de alto nível *Python*, que se fundamenta na abordagem orientada a objetos, sendo esta escolha motivada pela simplicidade dos códigos e por ser uma linguagem nativa do sistema operacional *Linux*, este utilizado como sistema de suporte ao supervisório SAGE (DIAS, 2017).

2. SEE APP

A proposta deste trabalho, SEE APP, utilizará a linguagem de programação *Python*, sendo um desenvolvimento totalmente comunitário e sem fins lucrativos. Diferentemente das ferramentas citadas, a SEE APP estará disponível para *download* na plataforma *GitHub.com*, este sendo um provedor de hospedagem na *Internet* para desenvolvimento de *software* e controle de versão, fornecendo controle de acesso e vários recursos de colaboração, como rastreamento de defeitos (*bugs*), solicitações de recursos, gerenciamento de tarefas, integração contínua e uma publicação de hipertexto editada de forma colaborativa para cada projeto.

A interface com o usuário contará com uma barra de menus contendo os itens necessários para a navegação, tais como: abrir arquivos, interagir com os modelos de barramento ou requisitar ajuda. Para isso, foram realizadas pesquisas exploratórias em ferramentas computacionais já existentes sobre o tema, estas citadas nas sessões anteriores, além do suporte bibliográfico sobre subestações e sobre as manobras em barramentos de subestações. Decidindo-se a utilização do pacote *Tkinter*, que acompanha a distribuição oficial do interpretador de licença livre *Python* sendo responsável pela interface gráfica com o usuário. As manobras serão analisadas através de funções criadas por *script* no *Python*.

Com esta aplicação, espera-se que os estudantes tenham um maior entendimento sobre o assunto tratado na disciplina de subestações, além de ter uma visualização de como realizar uma manobra de substituição de um equipamento em uma subestação. Além disso, caso ocorra alguma manobra/ação indevida, o usuário será notificado e, um aviso escrito aparecerá na tela relatando uma breve justificativa sobre o passo errôneo.

2.1 Linguagem de Programação Python

A linguagem de programação escolhida foi o *Python*. É uma linguagem de programação interpretada, ou seja, ao escrever um algoritmo, este não será traduzido para uma linguagem de máquina, e sim decodificado por outro programa, denominado interpretador (SOUBHIA et al. 2019).

Além disso, *Python* é uma linguagem Orientada a Objetos, um paradigma que facilita entre outras coisas o controle sobre a estabilidade dos projetos quando estes começam a tomar grandes proporções (LABAKI AND WOISKI, 2003).

Outro fator que também influenciou na escolha foi que a linguagem inclui diversas estruturas de alto nível (listas, dicionários, data/hora, complexos e outras) e uma vasta coleção de módulos prontos para uso, além de *frameworks* de terceiros que podem ser adicionados (BORGES, 2014).

2.2 Interface Gráfica

As GUIs são bastante populares no uso de *softwares* em geral, e os programadores devem estar aptos a trabalhar com a criação de interfaces, já que torna o uso mais fácil além de aumentar a produtividade (OTAVIO, 2016).

Para quem trabalha com desenvolvimento em *Python*, existem diversos *frameworks* e ferramentas que permitem a criação de interfaces gráficas Otavio (2016). Algumas ferramentas proeminentes de aplicativos de *Python* para desenvolvimento de GUI são *PyQt, Tkinter, wxWidgets, Python GTK*, e *Kivy* (INFO, 2021).

Alguns conceitos muito comuns abordados em GUIs são *Container, Widget, Event Handler e Event Loop*

- Container significa recipiente, e tem como objetivo organizar e guardar objetos na janela.
- Widget significa ferramenta, podendo ser um componente qualquer na tela, como um botão, uma caixa de seleção, uma caixa de texto, etc.
- Event Handler significa manipulador de eventos, e serve para executar algum tipo de rotina, como uma ação ao clicar em um botão.
- Event Loop significa ciclo de eventos, e verifica constantemente se outro evento foi acionado, executando uma rotina correspondente, caso a hipótese seja verdadeira.

3. METODOLOGIA

A metodologia utilizada neste trabalho, objetiva o projeto de uma aplicação pedagógica voltada à disciplina de subestações de energia elétrica. Toda a aplicação será desenvolvida utilizando a linguagem de programação *Python*, pois fornece um kit de ferramentas de *widget*, sendo o pacote *Tkinter*, responsável por proporcionar a implementação da GUI. Para o desenvolvimento do *software* será utilizado o IDE *PyCharm Community Edition*, lançada sob a licença de *software* livre de autoria da *Apache Software Foundation* (ASF).

A janela principal do aplicativo conterá uma barra de menus, anexada ao topo desta janela, logo abaixo da barra de títulos. Os itens da barra proverão funcionalidades a janelas específicas ou à aplicação, tais como: abrir arquivos, interagir com os modelos de barramento, ou requisitar ajuda, conforme Figura 1.

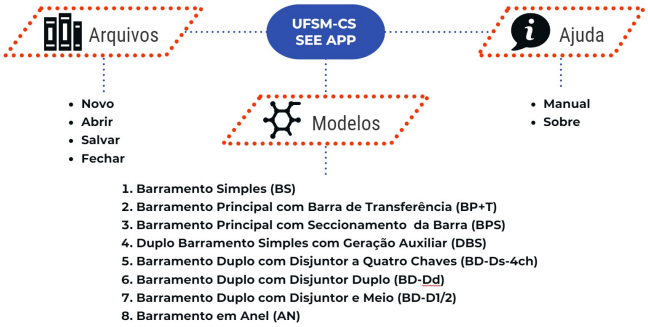

Figura 1. Fluxograma da Aplicação Educacional
Fonte: Autora (2022)

No *Front-End*, ou seja, tudo que envolve a parte visível da aplicação, ao selecionar o tipo de barramento que será realizado a manobra, o usuário terá a visualização do diagrama unifilar do barramento escolhido, além de uma tela informativa, uma caixa de combinação (*Combobox*) e botões de comandos. No *Back-End*, ou seja, os bastidores da aplicação, será realizado os procedimentos de checagem necessários para retornar ao usuário se a manobra de proteção foi bem sucedida ou não.

Os diagramas unifilares utilizados no aplicativo foram desenvolvidos em um editor de gráficos vetoriais gratuito e de código aberto, *LibreOffice Draw*, e as imagens geradas serão importadas para o aplicativo.

O aplicativo desenvolvido será disponibilizado para *download*, tanto para uso pessoal e/ou acadêmico quanto para contribuições do repositório remoto, através da plataforma de hospedagem na internet *GitHub.com*, onde fornece controle de acesso e vários recursos de colaboração.

4. IMPLEMENTAÇÃO

A seguir será descrito os passos de implementação, tanto do *Front-End* quanto do *Back-End*, do projeto em questão.

4.1 Front-End

Para a implementação da interface principal deve-se importar todos os componentes do módulo *Tkinter*, depois instanciar a classe ***Tk()***, através da variável ***app***, permitindo que os *widgets* possam ser utilizados, e para finalizar deve-se chamar o método ***.mainloop()***, sendo o *event loop* que irá manter a janela principal estática, caso contrário a interface não será exibida.

O usuário irá se deparar com uma janela contendo como componentes um ícone e uma barra de títulos, criados através dos métodos ***.iconbitmap()*** e ***title()***, respectivamente. Para fornecer comandos ao usuário, criou-se a classe ***barraMenu()*** e com seu método ***.criaBarraMenu()***. criou-se, também, uma barra de status com o *widget* ***Label()***, para que informe qual a janela em que o usuário está trabalhando. A janela principal pode ser vista na Figura 2.

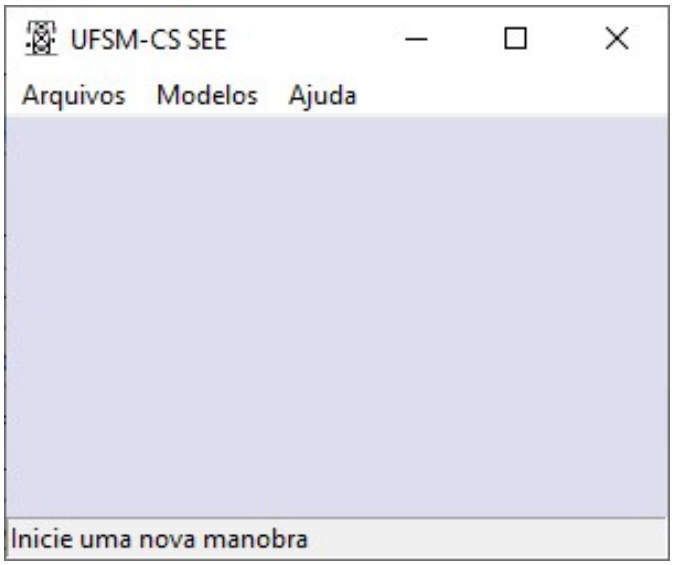

Figura 2. Primeiro contato com a GUI
Fonte: Autora (2022)

A classe ***barraMenu()*** contém o método ***.criaBarraMenu()***, este método tem um *widget* para a criação da barra de menus. Farão parte desta barra os submenus Arquivos, Modelos, e Ajuda.

- **Arquivos:** Este submenu contém os comandos Novo, Abrir, Salvar, e Fechar. Sendo responsáveis pela interação do usuário com as janelas da aplicação.
- **Modelos:** Este submenu contém oito comandos, com os nomes dos arranjos de barramentos mais usados nas subestações.
- **Ajuda:** Neste submenu contém os comandos Manual e Sobre, onde o usuário poderá abrir o manual contendo instruções da aplicação, e também terá um resumo sobre a aplicação, respectivamente.

As janelas dos barramentos seguem o mesmo princípio de criação e características da janela principal, contendo título e ícone. Conterá dois *containers* principais, sendo eles: o Diagrama do barramento e a Comunicação com o usuário.

- **Diagrama do barramento:** Este *container* contém o *widget* da imagem do barramento que será exibida na janela.
- **Comunicação com o usuário:** Este *container* contém três *widgets*, sendo eles: uma Tela informativa, uma Caixa de combinação, e os Botões da janela.
  - **Tela informativa:** Este *widget* será responsável por se comunicar com o usuário, solicitando informações e/ou retornando o *feedback* das ações realizadas pelo usuário.
  - **Caixa de combinação:** Este *widget*, sendo uma combinação dos *widgets* ***Entry*** e ***drop-down***, deve permitir a visualização ao usuário, ao clicar na seta do lado esquerdo, um menu suspenso mostrando todas as opções, e ao selecionar uma delas, substituirá o conteúdo atual do ***Entry***.
  - **Botões da janela:** Este *widget* conterá os botões necessários para cada passo, a quantidade pode variar de acordo com o decorrer da realização da manobra.

4.2 Back-End

A implementação das janelas dos barramentos são similares entre si, o que irá diferenciá-las será a comunicação com o usuário, pois cada um terá uma quantidade específica de elementos, e uma quantidade específica de passos para a realização das manobras.

O *widget* ***Label()*** tem muitas opções que permitem personalizar sua aparência, ele é o *widget* principal para todos os componentes do *Front-End*. Através do *widget* ***PhotoImage()***, é possível exibir a imagem do diagrama unifilar do barramento escolhido, bastando enviar a informação em uma variável. Para a sua exibição na janela, basta utilizar o método ***.pack()***

A tela informativa irá, inicialmente, apresentar um texto informando o tipo de arranjo de barramento exibido, além de solicitar em qual equipamento será realizado a manobra. Para isto, uma variável recebe a *String* e sua exibição se dá, também, pelo método ***.pack()***. Para a personalização da aparência, basta enviar como parâmetro as opções para ***Label()***, podendo, por exemplo, ser a definição da cor de fundo, o alinhamento e ancoragem do texto, o estilo e tamanho da fonte, a largura da borda, entre outras. Para ajustar a *widget* da tela informativa no *container*, basta enviar parâmetros das opções para ***.pack()***, caso nada seja enviado, a tela interativa se ajustará no topo e centralizado.

O *widget* ***Combobox()*** é responsável por listar os equipamentos que o usuário terá que selecionar, para realizar a manobra. Por padrão, o primeiro elemento da lista é mostrado, porém é possível alterá-lo utilizando o método ***.current()***. Também, por padrão, o seu estado normal o torna editável, logo, basta alterar o estado para *readonly* que o *combobox* se tornará "Somente leitura".

Inicialmente conterá dois botões, sendo eles "Iniciar Manobra" e "Sair". Para "Iniciar Manobra", ao pressioná-lo, dará início aos procedimentos de análise, e para "Sair", ao pressioná-lo, a janela do barramento irá fechar. Os botões podem ser criados utilizando o *widget* ***Button()***, que receberá um texto como parâmetro, além de outras opções para sua personalização. Assim como na tela informativa, utiliza-se o método ***.pack()*** para sua exibição e posição na janela principal.

A seguir, será apresentado o conjunto de *widgets*, mencionados anteriormente, na Figura 3.

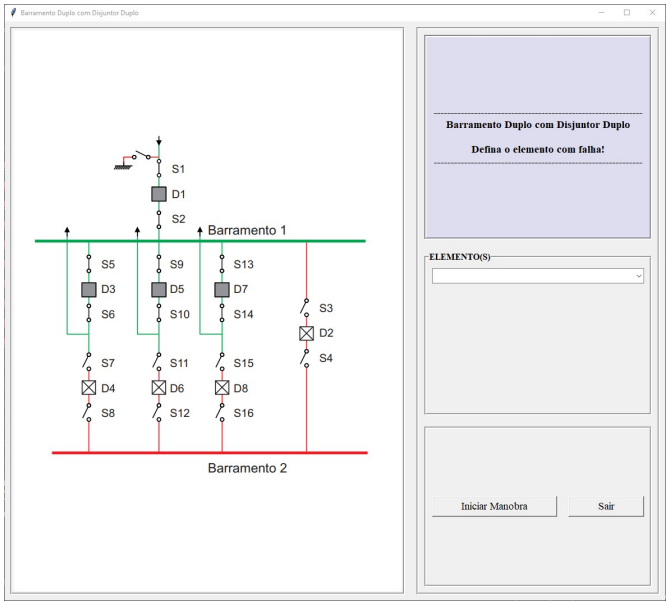

Figura 3. Janela de simulação
Fonte: Autora (2022)

Para a análise das manobras, uma função criada irá receber o valor do elemento selecionado pelo usuário, através do *combobox*, com a utilização de funções condicionais como *if/else*. Para melhor entendimento do processo de análise, será apresentado os passos iniciais das manobras para o arranjo barramento duplo com disjuntor duplo, a falha simulada será no disjuntor (D3). E para fins didáticos, foi adicionado um disjuntor de transferência (D2) neste arranjo.

Passo 1 Neste passo será apresentado para o usuário as seguintes informações:

- **Diagrama:** A imagem que aparecerá será o diagrama unifilar do arranjo "Barramento Duplo com Disjuntor Duplo", tendo o disjuntor em falha com uma indicação em formato de raio.
- **Tela informativa:** Conterá o texto: "Falha em D3 - Selecione o próximo passo para a manutenção!".
- **Caixa de combinação:** Conterá opções como abrir seccionadoras fechadas (Abrir S1-S2; Abrir S5-S6; ...), fechar seccionadoras abertas (Fechar S3-S4; Fechar S15- 16; ...), desligar disjuntores (Desligar D1; Desligar D3; ...), e ligar disjuntores (Ligar D2).
- **Botões da janela:** Os dois botões serão atualizados para "Próximo passo" e "Sair".

Pode-se visualizar a janela de simulação para o "Passo 1" na Figura 4.

Sabe-se que, para o "Passo 1", a opção a ser escolhida é "Fechar S3-S4". Com as funções condicionais, se o usuário selecionar "Fechar S3-S4" e pressionar o botão "Próximo Passo", segue-se para o "Passo 2", se não, uma mensagem irá aparecer na tela informativa, avisando-o sobre o erro cometido.

Passo 2 Neste passo será apresentado para o usuário as seguintes informações:

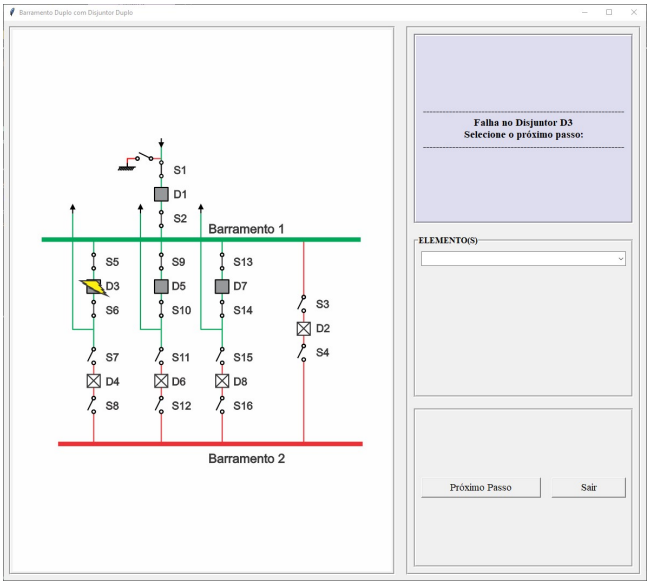

Figura 4. Janela Passo 1
Fonte: Autora (2022)

- **Diagrama:** A imagem que aparecerá será o mesmo diagrama do "Passo 1", porém com as seccionadoras S3 e S4 fechadas.
- **Tela informativa:** Conterá o texto: "Seccionadoras S3 e S4 fechadas!"
- **Caixa de combinação:** Conterá opções como abrir seccionadoras fechadas (Abrir S1-S2; Abrir S3-S4; ...), fechar seccionadoras abertas (Fechar S7-S8; Fechar S15-16; ...), desligar disjuntores (Desligar D1; Desligar D3; ...), e ligar disjuntores (Ligar D2).
- **Botões da janela:** Conterá os dois botões "Próximo passo" e "Sair".

Pode-se visualizar a janela de simulação para o "Passo 2" na Figura 5.

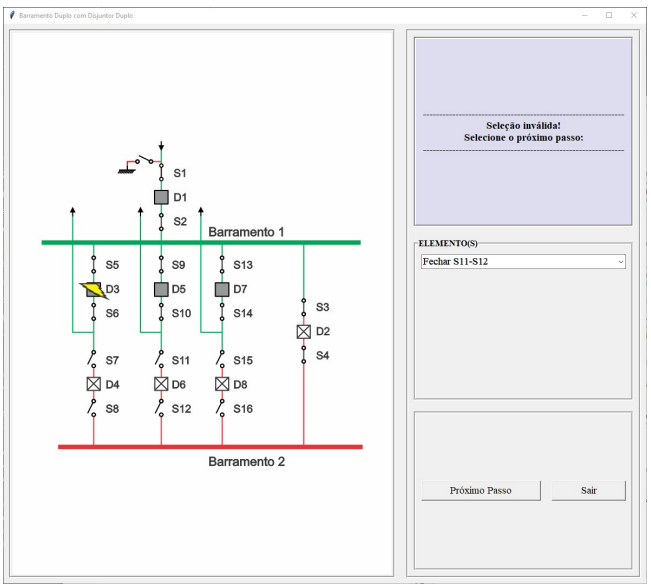

Figura 5. Janela Passo 2 com mensagem de erro
Fonte: Autora (2022)

Assim como no "Passo 1", do "Passo 2" em diante serão utilizadas as funções condicionais para avaliar a manobra

realizada, e a cada passo os *widgets* serão atualizados. A manobra final será quando o disjuntor com falha estiver trocado e as seccionadoras e o disjuntor de transferência não estiverem em uso. O usuário será informado com a mensagem: "Manobra de manutenção realizada!".

5. CONCLUSÃO

Ademais, o estudo e o desenvolvimento da ferramenta SEE APP só intensifica o quão a tecnologia pode e deve ser utilizada para fins educativos. O uso de aplicações didáticas fazem com que os acadêmicos consigam acompanhar, de maneira simples e objetiva, o que é visto em teoria em sala de aula. A utilização de interfaces gratuitas (*PyCharm Community Edition, LibreOffice Draw, GitHub*) para o desenvolvimento do aplicativo só reforçam e motivam o estudante no processo de criação de novas ferramentas didáticas.

Até o momento tem-se vencido algumas etapas do cronograma estipulado, como a definição do ambiente de aprendizagem, o planejamento e desenvolvimento da interface gráfica do usuário e o desenvolvimento dos diagramas unifilares correspondentes aos tipos de barramentos que serão utilizados na aplicação. Alguns barramentos já estão implementados e já foram testados com os discentes, do 1º semestre de 2022, da disciplina de Subestações de Energia Elétrica, da Universidade Federal de Santa Maria - Campus de Cachoeira do Sul.

Após a utilização da aplicação, os discentes responderam um formulário, adaptado do questionário de Ssemugabi, sendo este um instrumento avaliativo de softwares educacionais. O questionário aplicado conteve vinte e cinco perguntas, tendo, até o momento, nove respostas. Para uma primeira análise, 66,7% dos resultados afirmaram estarem satisfeitos com a aplicação SEE APP, o que serve de subsídio para modificações necessárias desta ferramenta.

Para o mês de Setembro/2022 espera-se que a conclusão do restante dos arranjos de barramento e a realização de uma nova avaliação de qualidade da aplicação. No mais, espera-se que em breve, o aplicativo seja disponibilizado aos usuários conforme o cronograma.

REFERÊNCIAS

Borges, L.E. (2014). *Python para desenvolvedores: Aborda Python 3.3*. Novatec, Rio de Janeiro, 3ª edition.

DIAS, S.E.C. (2017). *Automação de manobras em subestações de transmissão de energia elétrica.* Mestrado em engenharia elétrica, Universidade Federal de Campina Grande, Campina Grande.

HOLANDA, I.A.G.T.d. (2016). Suel: ferramenta didática de subestações elétricas. *Universidade Federal de Campina Grande*.

Info, W. (2021). 7 principais aplicações práticas de python e dicas para iniciar uma carreira na área, 2021. Disponível em: <https://br.atsit.in/archives/110078/>. Acessado em 04 Jan. 2022.

Labaki, J. and Woiski, E. (2003). Introdução a python–módulo a. *Grupo Python, UNESP*.

Macedo, R.J., Duarte, M.d.A., and Teixeira, N.G. (2012). Novas metodologias de ensino e aprendizagem aplicadas ao curso de engenharia elétrica: o foco do ensino no século xxi. Artigo COBENGE, Rio de Janeiro.

Otavio, J. (2016). Tkinter: Interfaces gráficas em python, 2016. Disponível em: <https://www.devmedia.com.br/tkinter-interfaces-graficas-em-python/33956/>. Acesso em: 04 Jan. 2022.

Ramos, J.F.M.D. (2010). *FEUPowerTool: ferramenta pedagógica para manobras em subestações*. Mestrado integrado em engenharia electrotécnica e de computadores, Faculdade de Engenharia da Universidade do Porto, Porto.

SILVA, H.A.B.d. (2017). *Simulador de uma subestação elétrica para ensino de princípios básicos de eletricidade*. Mestrado em computação aplicada, Universidade Federal do Pará, Marabá.

Soubhia, D.A.L., da Costa, E.T., Freitas, F.L.M., Menezes, L.B., dos Santos, M.A., and Maran, V. (2019). Python 101. *Universidade Federal de Santa Maria*.

Tavares, F.A.M. (2015). *Aplicação informática para dimensionamento de barramentos em subestações*. Mestrado em engenharia eletrotécnica, Instituto Superior de Engenharia de Lisboa, Lisboa.