UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

JOÃO PEDRO DE SOUZA OLIVO TARDIVO

RELATÓRIO TÉCNICO SOBRE O SIMULADOR DE GRAMÁTICA REGULAR

APUCARANA – PR
2023

JOÃO PEDRO DE SOUZA OLIVO TARDIVO

RELATÓRIO TÉCNICO SOBRE O SIMULADOR DE GRAMÁTICA REGULAR

Trabalho apresentado à disciplina de Linguagens Formais Autômatos e Computabilidade, do curso de Bacharelado em Ciência da Computação.

Professor: Guilherme Nakahata

APUCARANA – PR
2023

SUMÁRIO

INTRODUÇÃO

No domínio das linguagens formais e da teoria dos autômatos, os simuladores desempenham um papel fundamental na elucidação do intrincado funcionamento de vários tipos de gramática. Dentre elas, as gramáticas regulares ocupam uma posição fundamental devido à sua simplicidade e aplicabilidade em diversos cenários do mundo real. A compreensão e manipulação de gramáticas regulares são habilidades essenciais para cientistas da computação, auxiliando em diversas áreas como design de compiladores, processamento de linguagem natural e reconhecimento de padrões.

Este relatório investiga o desenvolvimento e análise de um simulador dedicado a gramáticas regulares. O simulador serve como uma ferramenta computacional que permite a exploração, validação e visualização de gramáticas regulares, abrangendo suas regras, derivações e linguagens. Ao examinar meticulosamente as nuances das gramáticas regulares através deste simulador, pretende-se melhorar a compreensão dos seus princípios subjacentes e mecanismos operacionais.

O objetivo deste esforço é fornecer uma exploração abrangente de gramáticas regulares, elucidando sua importância no domínio das linguagens formais e dos autómatos. Através de análise meticulosa, experimentação e apresentação de resultados, este relatório se esforça para contribuir com insights valiosos para a implementação prática e utilização de gramáticas regulares. Além disso, o relatório pretende servir como um recurso fundamental para estudantes e investigadores, promovendo uma compreensão mais profunda das gramáticas regulares e do seu significado computacional no campo da ciência da computação.

CAPÍTULO 1

OBJETIVOS, MOTIVAÇÃO E RECURSOS UTILIZADOS DURANTE A IMPLEMENTAÇÃO

O objetivo principal deste relatório é aprofundar-se de forma prática no domínio das gramáticas regulares através do desenvolvimento de um simulador. Através de exploração e análise meticulosas, este estudo visa compreender de forma abrangente os meandros das gramáticas regulares e suas aplicações no campo das linguagens formais e dos autómatos.

Este estudo visa projetar e implementar uma ferramenta de simulação capaz de avaliar e demonstrar gramáticas regulares. Isso envolve a utilização de algoritmos computacionais e estruturas de dados para criar uma interface eficiente e fácil de usar.

Além disso, o relatório se esforça para analisar a funcionalidade do simulador através de diversos casos de teste, explorando diferentes tipos de gramáticas regulares e suas linguagens correspondentes. O objetivo aqui é avaliar a capacidade do simulador de lidar com diversas estruturas gramaticais e produzir derivações linguísticas corretas, sejam elas GLD, GLUD, GLE, GLUE ou GLC.

Desta forma, este relatório procura oferecer uma exploração das gramáticas regulares e da sua simulação, promovendo uma compreensão diferenciada dos seus fundamentos teóricos e aplicações práticas no domínio da ciência da computação.

1.1 LINGUAGEM DE PROGRAMAÇÃO PYTHON

Python foi selecionado como linguagem de programação para desenvolver o simulador de gramática regular devido à sua versatilidade, legibilidade e extensas bibliotecas. A simplicidade e facilidade de uso do Python permitem prototipagem rápida e desenvolvimento eficiente, tornando-o uma escolha ideal para projetos acadêmicos. Além disso, o rico ecossistema de bibliotecas e estruturas do Python facilita a integração de vários componentes, permitindo a criação de um simulador rico em recursos. Sua

sintaxe limpa e expressiva garante clareza no código, facilitando a parte de apresentação do projeto.

1.2 PACOTE PYQT6 PARA INTERFACE GRÁFICA

A estrutura PyQt6 foi empregada para projetar a interface gráfica do usuário (GUI) do simulador de gramática regular. PyQt6, um conjunto de ligações Python para a estrutura de aplicação Qt, foi escolhido por sua robustez, compatibilidade entre plataformas e extensa documentação. A flexibilidade do Qt permite a criação de interfaces de usuário intuitivas e visualmente atraentes, melhorando a experiência do usuário. O suporte ativo da comunidade do PyQt6 garante acesso a uma variedade de recursos, tutoriais e fóruns, permitindo a rápida resolução de problemas e a melhoria contínua da interface do simulador.

1.3 PACOTE PYINSTALLER PARA BUILD E DISTRIBUIÇÃO

PyInstaller foi utilizado para empacotar o simulador de gramática regular em executáveis independentes para vários sistemas operacionais. PyInstaller simplifica o processo de implantação convertendo scripts Python em arquivos executáveis, eliminando a necessidade dos usuários instalarem o Python ou quaisquer dependências adicionais. Esta escolha agiliza a experiência do usuário, tornando o simulador acessível a um público mais amplo, sem exigir que eles naveguem em procedimentos complexos de instalação. A compatibilidade do PyInstaller com as plataformas Windows, macOS e Linux garante que o simulador possa ser facilmente distribuído e executado em diversos ambientes, maximizando seu alcance e usabilidade.

1.4 WSL (WINDOWS SUBSYSTEM FOR LINUX) COM PYINSTALLER PARA BUILD E DISTRIBUIÇÃO NA PLATAFORMA

No contexto das plataformas Windows, a integração do Windows Subsystem for Linux (WSL) foi fundamental para garantir a compatibilidade perfeita com o sistema operacional. WSL forneceu um ambiente que imitava uma distribuição Linux dentro do Windows, permitindo que a ferramenta PyInstaller criasse executáveis compatíveis com Linux diretamente em um sistema Windows. Essa integração preencheu a lacuna entre o Windows e o

Linux, permitindo o desenvolvimento e o empacotamento do simulador de gramática regular para distribuições Linux sem a necessidade de uma partição Linux separada.

1.5 ORACLE VM VIRTUALBOX E UBUNTU PARA TESTES

Para validar a funcionalidade e compatibilidade da versão Linux do simulador de gramática regular, foi utilizada uma máquina virtual Oracle VM VirtualBox rodando Ubuntu Linux. Esse ambiente virtualizado forneceu um espaço controlado e isolado para testes com a plataforma, permitindo que o desenvolvedor avaliasse o desempenho do simulador, a interface do usuário e a funcionalidade geral em um sistema Linux genuíno.

Ao empregar o Oracle VM VirtualBox junto com o Ubuntu, o desenvolvedor pode simular cenários de uso do mundo real, garantindo que o simulador se comporta conforme esperado em uma plataforma Linux. A combinação de WSL para integração com PyInstaller e Oracle VM VirtualBox com Ubuntu para testes garantiu a criação e validação bem-sucedidas de uma versão Linux do simulador de gramática regular.

1.6 ESTRUTURA DA INTERFACE GRÁFICA

Neste projeto, uma abordagem de layout empilhado é adotada para gerenciar diferentes telas ou layouts de forma eficiente. A aplicação inicia com uma tela de boas-vindas e, à medida que o usuário interage com a GUI, novos layouts são carregados, adicionados a uma pilha e exibidos com base em suas escolhas.

Essa abordagem oferece vários benefícios como a modularidade, onde cada tela ou layout é desenvolvido de forma independente, facilitando o foco em um aspecto do aplicativo por vez.

Ademais, tem-se uma melhor eficiência de memória, pois a medida que o usuário avança, novos layouts são adicionados à pilha. No entanto, se o usuário optar por voltar, o layout anterior será removido da pilha e efetivamente “destruído”. Isso ajuda a economizar memória e recursos, garantindo que o aplicativo permaneça responsivo e eficiente mesmo durante fluxos de trabalho complexos.

Finalmente, temos uma separação relativa de funções, onde cada layout pode encapsular sua própria lógica e funcionalidade, promovendo uma divisão mais clara de interesses na base de código do aplicativo.

1.7 ESTRUTURAS DE DADOS PARA A LÓGICA DA APLICAÇÃO

O simulador depende intrinsecamente de estruturas de dados bem projetadas, que são fundamentais para armazenamento, recuperação e manipulação eficientes de regras gramaticais. Neste simulador, uma estrutura semelhante a um HashMap, implementada por meio de dicionários Python, serve como base da lógica subjacente. Essa estrutura permite a representação de variáveis em uma gramática regular, mapeando-as para suas regras de produção correspondentes, que são armazenadas em conjuntos (Set). A utilização de dicionários e conjuntos neste contexto é uma escolha criteriosa, proporcionando um equilíbrio harmonioso entre eficiência computacional e coerência lógica.

Os dicionários Python foram escolhidos como estrutura de dados primária por sua complexidade de tempo médio O(1) para operações de inserção e recuperação. No contexto do simulador de gramática regular, os dicionários servem como um mecanismo ideal para mapear variáveis (símbolos não terminais) para suas respectivas regras de produção. As chaves do dicionário representam as variáveis, garantindo um mapeamento único para cada variável. Esta singularidade inerente reflete as características das variáveis numa gramática regular, onde cada variável deve ter regras de produção distintas. Ao utilizar dicionários, o simulador otimiza o armazenamento e recuperação de regras gramaticais, facilitando o acesso rápido a regras de produção específicas associadas a uma variável durante os processos de análise e derivação.

As regras de produção para variáveis em uma gramática regular são armazenadas como conjuntos no dicionário. Os conjuntos foram escolhidos pela sua capacidade de impor exclusividade, garantindo que regras de produção idênticas não sejam repetidas para uma variável específica. Numa gramática regular, cada variável deve ter regras de produção distintas, um requisito crucial para gerar derivações e linguagens inequívocas. Os conjuntos, com a propriedade inerente de conterem elementos únicos, alinham-se

perfeitamente com este requisito, garantindo a ausência de duplicatas nas regras de produção de uma variável. Isto não apenas mantém a integridade da gramática, mas também aumenta a precisão do simulador na geração e análise de derivações.

O uso de dicionários em Python aproveita um mecanismo de hash nos bastidores. Este mecanismo melhora significativamente a eficiência das operações que envolvem grandes conjuntos de dados. Ao acessar regras de produção para uma variável específica, a função hashing calcula o valor hash da chave da variável, permitindo acesso direto e rápido às regras de produção correspondentes. Esse mecanismo de recuperação baseado em hash garante que, mesmo em cenários que envolvem regras gramaticais extensas, o simulador possa recuperar e manipular rapidamente os dados necessários. Consequentemente, o simulador pode lidar com gramáticas regulares complexas com diversas variáveis e regras de produção, oferecendo uma ferramenta versátil para fins educacionais.

Em suma, a utilização estratégica de dicionários e conjuntos Python, semelhante a uma estrutura HashMap, capacita o simulador de gramática regular com a base necessária para uma representação gramatical precisa, eficiente e escalável. Esta escolha reflete uma consideração da eficiência computacional e da precisão lógica, garantindo a robustez do simulador no tratamento de diversas gramáticas regulares e permitindo exploração e análise abrangentes no domínio das linguagens formais e da teoria dos autómatos.

1.8 ESTRUTURA LÓGICA DA SIMULAÇÃO

O coração do simulador de gramática regular está em seu processo de derivação recursiva, um algoritmo projetado para explorar sistematicamente a vasta árvore de derivações possíveis. Esta função recursiva recebe uma série de parâmetros: o dicionário gramatical que representa a gramática regular, a variável inicial, a palavra alvo a ser construída, a palavra atualmente construída (inicialmente vazia), a profundidade da iteração atual (começando em 1) e uma lista (array Python) dos resultados (inicialmente vazia). Através de uma série de chamadas recursivas, a função constrói e avalia meticulosamente possíveis derivações, empregando lógica para navegar efetivamente na árvore gramatical.

A recursão inicia com a variável inicial e uma string vazia como a palavra construída. A cada iteração, a função avalia múltiplas condições. Primeiramente, verifica se a palavra construída corresponde à palavra alvo, significando uma derivação bem-sucedida. Alternativamente, se a palavra construída ultrapassar o comprimento da palavra alvo, a função conclui que o ramo atual da árvore gramatical não tem êxito e interrompe a recursão para este ramo.

A função recursiva explora então as regras de produção associadas à variável atual. Na iteração inicial, a função identifica a variável inicial e itera através de suas regras de produção. Para cada regra de produção, é feita uma chamada recursiva com a palavra construída atualizada, permitindo que a função se aprofunde na árvore gramatical. A recursão continua em profundidade, explorando exaustivamente um ramo antes de passar para o próximo, garantindo uma busca abrangente de todas as derivações possíveis.

Durante cada iteração, a função examina a palavra construída, identificando variáveis dentro dela. Se uma variável for encontrada, a função a substituí pelas regras de produção correspondentes, gerando novas strings para explorar. Este mecanismo de substituição recursiva constrói meticulosamente derivações potenciais, passo a passo. A abordagem em profundidade garante que a função explore a totalidade de um ramo antes de voltar atrás e explorar caminhos alternativos.

Ao longo do processo recursivo, a função preenche uma lista de resultados com as regras de produção utilizadas em cada etapa. Este registro abrangente de etapas permite que o simulador capture o caminho de derivação bem-sucedido quando encontra a palavra alvo. Se um caminho válido for identificado, a lista de resultados encapsula a sequência de regras de produção, fornecendo um registro detalhado das etapas executadas para construir a palavra alvo. Essas informações são importantes para exibir ao usuário o processo de derivação, possibilitando uma visualização transparente do comportamento da gramática.

A recursão termina sob duas condições, ou um caminho válido que leva à palavra alvo é descoberto, caso em que a simulação é bem-sucedida, ou todas as possibilidades são exploradas exaustivamente sem construir a palavra

alvo. No último cenário, a função termina normalmente, sinalizando a ausência de uma derivação válida dentro da gramática regular dada.

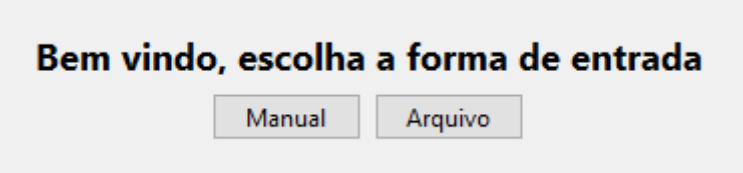
Em essência, o processo de derivação recursiva do simulador de gramática regular resume uma exploração meticulosa das possibilidades da gramática. Através da sua abordagem sistemática, a função encapsula a essência das gramáticas regulares, incorporando a natureza determinística e não ambígua destas estruturas gramaticais. Essa metodologia recursiva serve como base do simulador, permitindo gerar, explorar e analisar diversas gramáticas regulares com precisão e exatidão.

CAPÍTULO 2

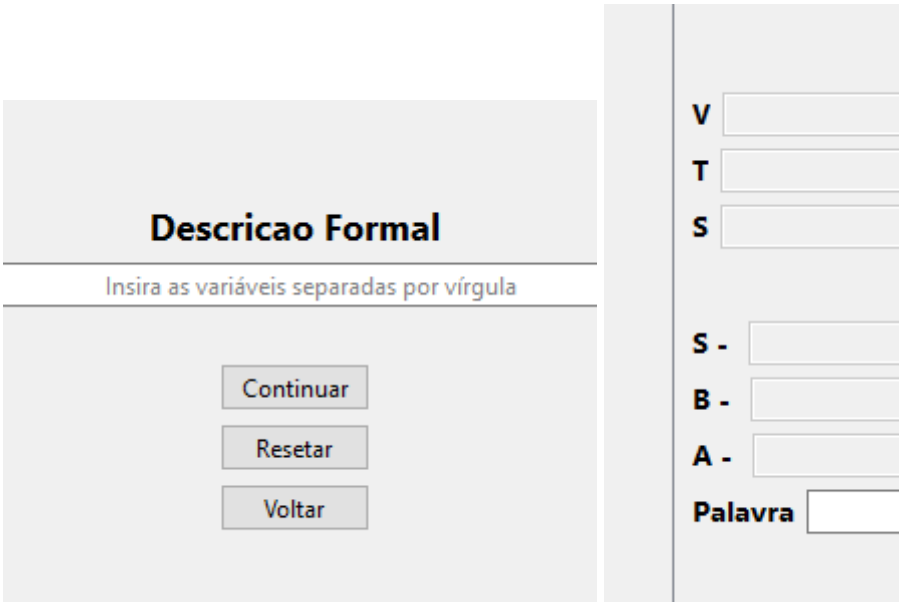
RESULTADOS

O simulador de gramática regular, projetado como um aplicativo de desktop, combina funcionalidade com interface amigável, fornecendo uma plataforma intuitiva para explorar gramáticas regulares estilo GLD, GLUD, GLE, GLUE e GLC. A experiência do usuário é elaborada garantindo uma navegação tranquila e instruções claras em cada etapa. O aplicativo possui um fluxo de trabalho eficiente, permitindo aos usuários interagir com o simulador sem esforço.

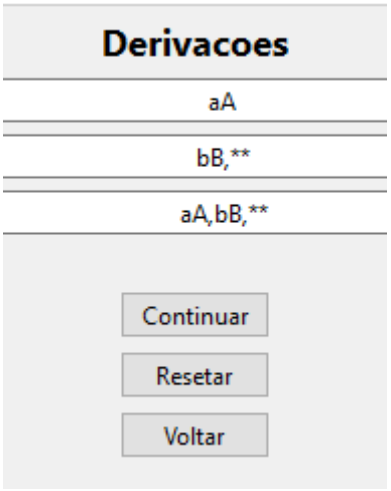
Ao iniciar o aplicativo, os usuários são recebidos por uma interface acolhedora. A tela de boas-vindas apresenta duas opções distintas: entrada manual e entrada de arquivo. Os usuários podem escolher seu método de entrada preferido, preparando o terreno para uma experiência de exploração gramatical.



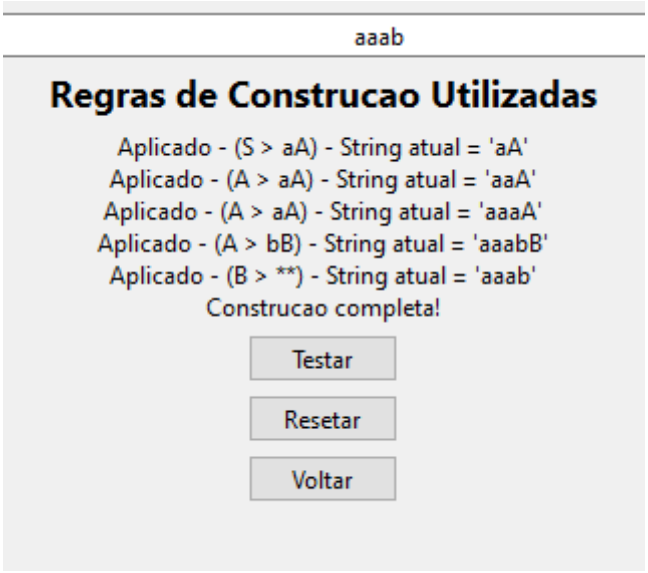
Para usuários que optam pela entrada manual, a interface os orienta através de um processo passo a passo de definição da gramática regular. A interface solicita que os usuários insiram variáveis, garantindo exclusividade e formatação correta. Da mesma forma, os terminais são inseridos, servindo como alfabeto da gramática. O mecanismo de validação intuitivo fornece feedback instantâneo, garantindo que os usuários cumpram os requisitos formais da gramática.



A interface continua a orientar os usuários na especificação da variável inicial e na definição de regras de produção para cada variável. Os usuários são solicitados a inserir regras de produção usando variáveis válidas, terminais ou o símbolo de terminal vazio "**". Cada etapa de entrada é acompanhada por mensagens e avisos úteis, garantindo que os usuários construam uma gramática regular válida.

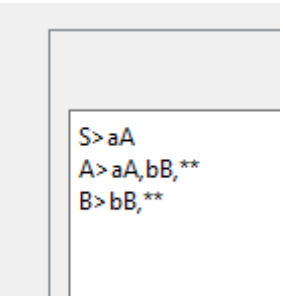


Uma vez definida a gramática, os usuários podem inserir uma palavra de teste, iniciando o processo de simulação. O simulador constrói a palavra meticulosamente, exibindo as regras de produção específicas utilizadas na derivação. Alternativamente, se a palavra não puder ser construída, a aplicação fornece uma mensagem clara a respeito.



A interface também oferece opções convenientes, permitindo aos usuários redefinir a entrada, permitindo-lhes começar de novo ou retornar à tela de boas-vindas, oferecendo flexibilidade na exploração de múltiplas gramáticas.

Para usuários que preferem entrada de arquivos, a interface simplifica o processo. Os usuários podem colar a gramática diretamente no campo de texto designado, seguindo o formato prescrito descrito no anexo. A abordagem simplificada reduz o esforço manual, permitindo aos usuários inserir rapidamente gramáticas complexas para simulação.



Semelhante ao método de entrada manual, os usuários podem inserir uma palavra de teste e iniciar o processo de simulação. O aplicativo fornece feedback, exibindo as etapas de derivação ou indicando a incapacidade de construir a palavra, mantendo a consistência na experiência do usuário em ambos os métodos de entrada.

Em todo o aplicativo, botões intuitivos facilitam a navegação. O botão “continuar” orienta os usuários em cada etapa, garantindo a sua progressão sequencial. Mensagens claras e concisas fornecem feedback instantâneo,

orientando os usuários na correção de erros e garantindo uma definição gramatical precisa. A transformação do botão “continuar” em botão “teste” significa a prontidão para iniciar a simulação, melhorando a compreensão do usuário.

A inclusão do botão “reset” permite aos usuários redefinir suas entradas, facilitando a experimentação com diversas gramáticas. Além disso, o botão “voltar” permite que os usuários retornem à tela de boas-vindas, fornecendo uma rota de fuga caso desejem mudar os métodos de entrada ou começar do zero.

Em suma, o aplicativo fornece uma ferramenta poderosa para estudantes, pesquisadores e entusiastas mergulharem no fascinante mundo das gramáticas regulares nas linguagens formais e dos autómatos.

CONCLUSÃO

No domínio das linguagens formais e da teoria dos autômatos, as gramáticas regulares permanecem como construções fundamentais, essenciais na compreensão de processos computacionais e estruturas linguísticas. Este relatório embarcou numa exploração de gramáticas regulares, culminando no desenvolvimento de uma aplicação desktop dedicada à sua simulação. Através de planejamento, utilização estratégica de recursos e design algorítmico, o simulador de gramática regular emergiu como uma ferramenta educacional e um testemunho da sinergia entre a teoria e a implementação prática na ciência da computação.

A jornada através deste relatório começou com uma introdução completa ao significado das gramáticas regulares na ciência da computação. Sua simplicidade e natureza determinística os tornam blocos de construção fundamentais, encontrando aplicações em diversos campos, como design de compiladores, processamento de linguagem natural e reconhecimento de padrões. Os objetivos delineados neste relatório orientaram o processo de desenvolvimento, enfatizando a necessidade de uma compreensão profunda das gramáticas regulares, da sua implementação prática e da sua relevância no mundo real.

A escolha do Python como linguagem de programação, aliada à utilização de bibliotecas e frameworks essenciais, forneceu uma base robusta para o simulador de gramática regular. A seleção das estruturas de dados, meticulosamente explicada nas seções anteriores, garantiu representação e manipulação eficientes das regras gramaticais. A lógica de derivação recursiva, detalhada em profundidade, facilitou a exploração de diversas derivações, incorporando a essência da natureza determinística e não ambígua das gramáticas regulares.

O próprio simulador, apresentado na seção de resultados, é uma prova de design e abordagem centrada no usuário. Sua interface intuitiva, métodos de entrada estruturados e mecanismos de feedback claros permitem que os usuários explorem gramáticas complexas sem esforço. A integração de opções de entrada manual e de arquivo adiciona versatilidade, acomodando usuários

com preferências e níveis de conhecimento variados. Através do simulador, os usuários podem testemunhar o intrincado processo de derivações gramaticais regulares, obtendo informações valiosas sobre os princípios computacionais subjacentes.

Em termos mais amplos, este relatório sublinha a natureza interdisciplinar da ciência da computação, onde os conceitos teóricos se traduzem perfeitamente em aplicações práticas. As gramáticas regulares, antes símbolos abstratos na teoria da linguagem formal, encontram vida na forma de um simulador fácil de usar, preenchendo a lacuna entre a teoria e a prática. Este esforço reafirma a importância do conhecimento teórico na definição de soluções inovadoras, enfatizando a relação simbiótica entre fundações acadêmicas e aplicações do mundo real.

REFERÊNCIAS

Python. Disponível em: https://www.python.org/. Acesso em: 23 jul. 2023.

Qt. Disponível em: https://www.qt.io/. Acesso em: 23 jul. 2023.

PyQt6 - PyPI. Disponível em: https://pypi.org/project/PyQt6/. Acesso em: 23 jul. 2023.

PyQt - Riverbank Computing. Disponível em: https://www.riverbankcomputing.com/software/pyqt/. Acesso em: 23 jul. 2023.

PyInstaller - PyPI. Disponível em: https://pypi.org/project/pyinstaller/. Acesso em: 23 jul. 2023.

PyInstaller Documentation. Disponível em: https://pyinstaller.org/en/stable/. Acesso em: 23 jul. 2023.

Windows Subsystem for Linux Installation Guide. Disponível em: https://learn.microsoft.com/en-us/windows/wsl/install. Acesso em: 23 jul. 2023.

Ubuntu. Disponível em: https://ubuntu.com/download. Acesso em: 23 jul. 2023.

VirtualBox. Disponível em: https://www.virtualbox.org/. Acesso em: 23 jul. 2023.