UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE GRADUAÇÃO EM CIÊNCIA E ENGENHARIA DA COMPUTAÇÃO

DISCIPLINA DE CLASSIFICAÇÃO E PESQUISA DE DADOS
PROF. LEANDRO KRUG WIVES

Alunos:
Ian Kersz Amaral - 00338368
Marcos Luiz Kurth Reckers - 00315653

TRABALHO FINAL DA DISCIPLINA:

Aplicativo de Pesquisa de Jogos retirado da Steam.

Porto Alegre
2023

Sumário

Visão Geral

A Steam é a maior plataforma de distribuição digital de jogos do mundo, com mais de 30 mil títulos disponíveis e milhões de usuários ativos. No entanto, a sua interface de busca de jogos pode ser considerada complexa e confusa, pois exige que o usuário filtre por diversos critérios, como gênero, categoria, preço, avaliação, data de lançamento, entre outros, porém nem todos os campos estão disponíveis ou de fácil acesso para o usuário.

Nós buscamos fazer uma aplicação bem rudimentar e simples, que simula a busca na Steam. A nossa aplicação não tem a intenção de substituir ou competir com a Steam, mas apenas de demonstrar os conceitos básicos de um sistema de buscas. A aplicação é apenas um protótipo didático e experimental, que serve para aprendermos sobre o funcionamento e os desafios de um sistema de buscas e sistemas de arquivos e banco de dados.

Nossa aplicação carinhosamente chamada de “VAPOR” não oferece as mesmas funcionalidades e vantagens da Steam, como a recomendação personalizada, a descoberta de novos jogos, a integração com a comunidade, entre outras. É apenas um exercício acadêmico e uma homenagem à Steam. Entretanto, nos possibilitou aprender não só os requisitos da cadeira de Classificação e Pesquisa de Dados, como outras tecnologias e ferramentas que vamos abordar adiante nos tópicos subsequentes.

Para esse trabalho, utilizamos então um banco de dados pronto em CSV disponível no site Kaggle que continha a base de dados de jogos da Steam. Esse banco de dados é uma versão mais compacta e simplificada do banco de dados total disponível na API da Steam, porém ele já nos fornece muitas informações sobre todos os jogos da Steam, tais como: Nome, Data de lançamento, Desenvolvedor, Publicadora, Avaliação, Preço entre outras.

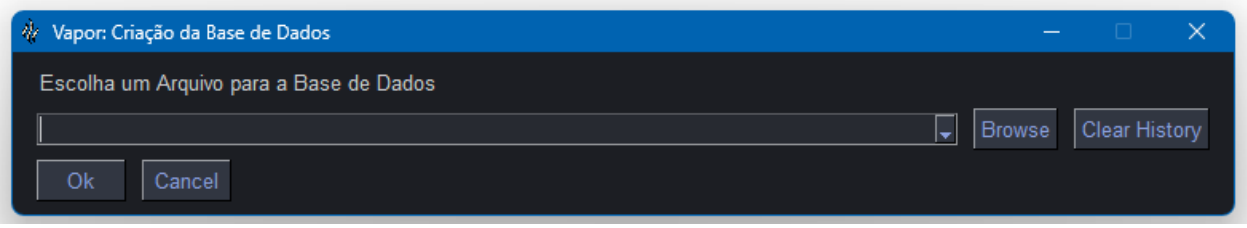
Sendo assim, o VAPOR é um sistema de buscas que preza pela simplicidade, mas, sem abrir mão das funcionalidades, ele conta com uma interface gráfica que dispõe de diversos filtros aplicáveis, além da possibilidade de adicionar novos jogos à base de dados que possui e ordenar os resultados. Abordaremos com detalhes, sobre as funcionalidades no tópico de “Guia de Uso”.

O programa pode ser acessado fazendo download diretamente do repositório GitHub, no seguinte link: https://github.com/Marcos-Reckers/CPD-Trab-Final

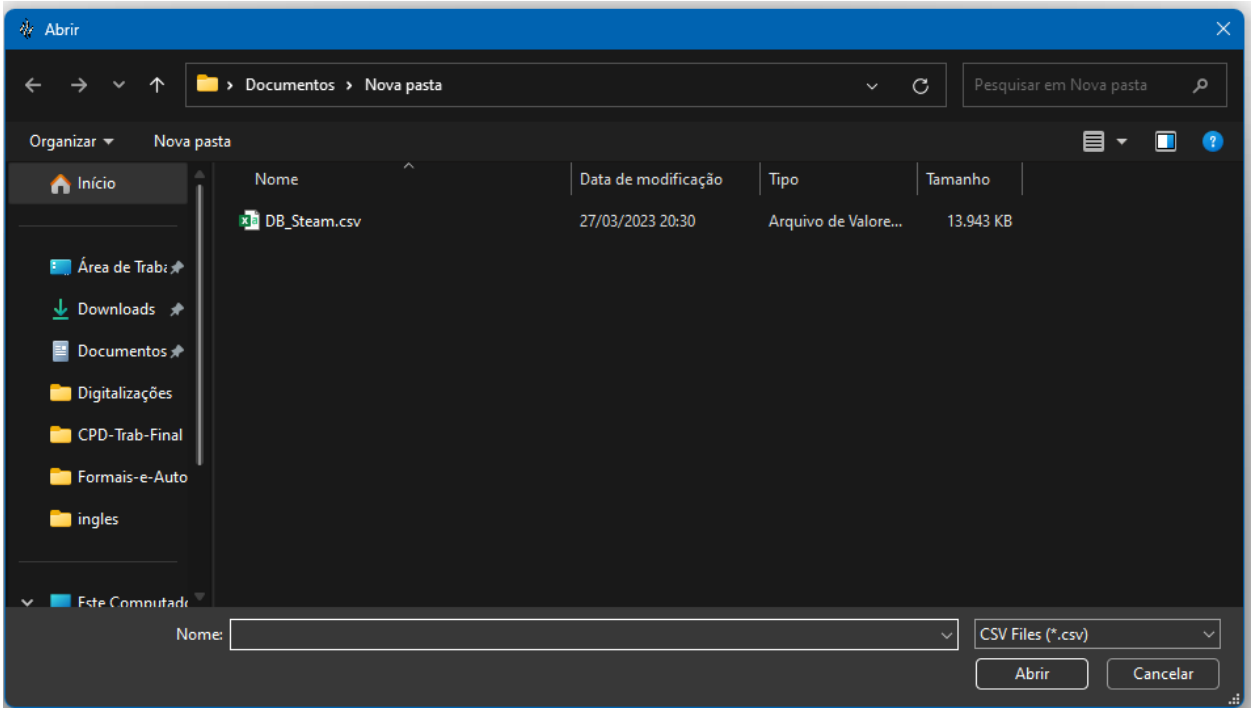
Guia de Uso

Como mencionado no tópico anterior, buscamos ao máximo a simplicidade dentro da nossa aplicação, portanto tentamos deixar tudo da forma mais intuitiva possível.

1. ***Primeiros Passos***
    a. Ao abrir o executável “VAPOR.exe”, ele automaticamente irá verificar se já existe uma pasta chamada “data” que contém as estruturas criadas para a busca. Caso não encontre, uma janela aparecerá solicitando que o usuário coloque o caminho para a base de dados em CSV



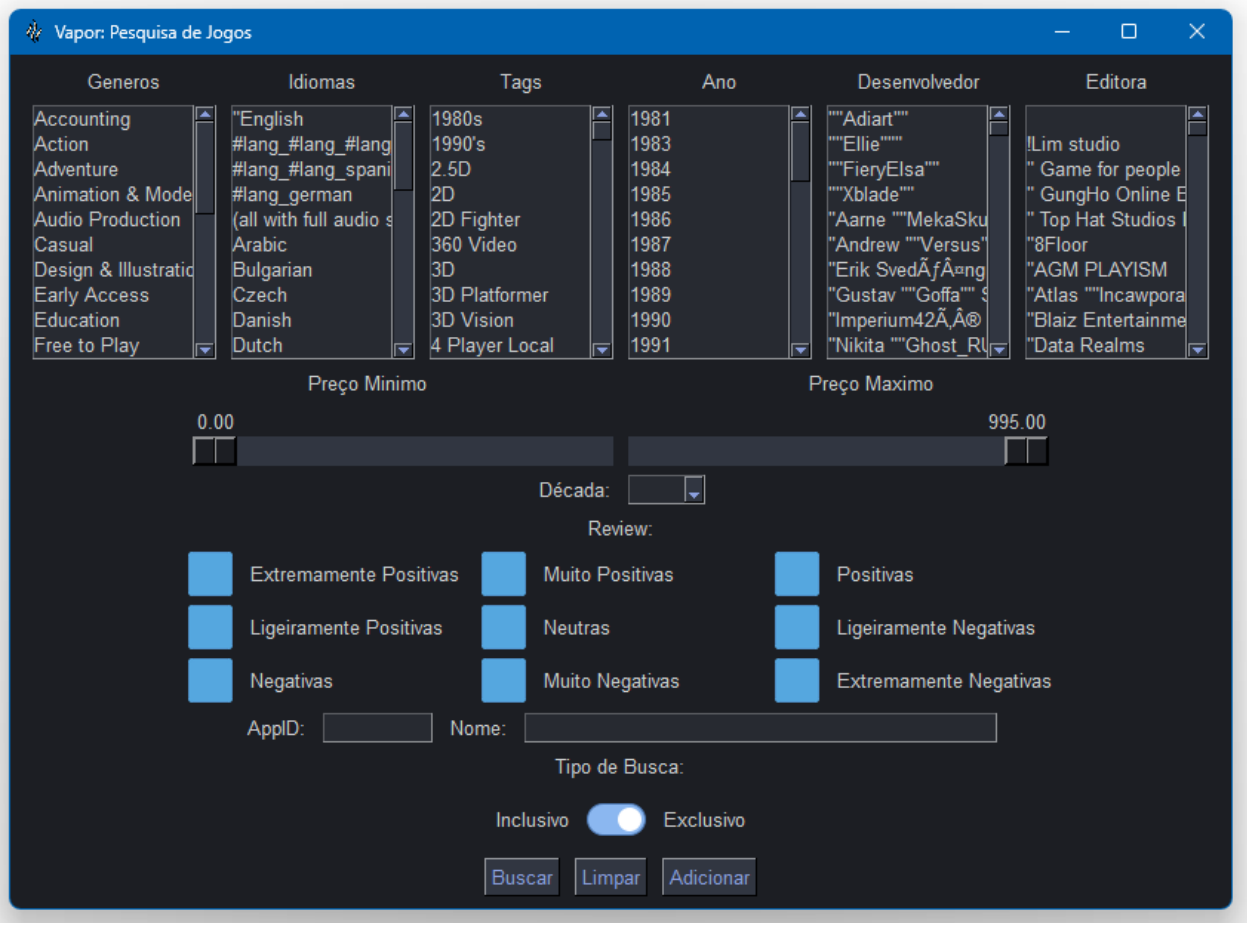
    b. O usuário pode optar tanto por inserir diretamente o caminho, como por buscar o arquivo no explorador do sistema clicando no botão “Browse”
        - OBS: A base de dados em CSV necessária para rodar o programa é uma versão modificada da presente no Kaggle anteriormente linkado, visto que esta continha diversos caracteres fora do padrão ASCII, os quais causam problemas para o programa.



c. Uma vez selecionada a base de dados, o programa criará todas as estruturas necessárias na pasta “data”

2. *Tela Inicial*

a. Uma vez que tudo esteja carregado a tela inicial irá aparecer:



b. Como mostrado na figura acima, todas as opções já estão na tela inicial.
c. Na primeira linha temos os filtros de Gênero, Idiomas, Tags, Ano de lançamento, Desenvolvedora e Editora (Publisher)
d. Na linha de baixo temos os sliders de preço mínimo e preço máximo bem como a seleção de intervalo de décadas.
e. Mais abaixo no terceiro bloco, temos as opções de seleção de Avaliações, indo de “Extremamente Positivas” até “Extremamente Negativas”
f. Por fim temos dois campos de texto onde podem ser inseridos um AppID de algum jogo específico e um nome ou prefixo para busca.
g. Todas essas informações podem ser selecionadas em conjunto e, no rodapé da tela, temos um switch que seleciona entre busca inclusiva ou exclusiva dos filtros.
h. Os três botões ao final da tela são os botões de realizar a busca “Buscar”, Limpar todos os filtros selecionados “Limpar” ou ir para a tela de adição de

novos jogos a base de dados “Adicionar” (Abordaremos melhor sobre a adição mais pra frente no documento)

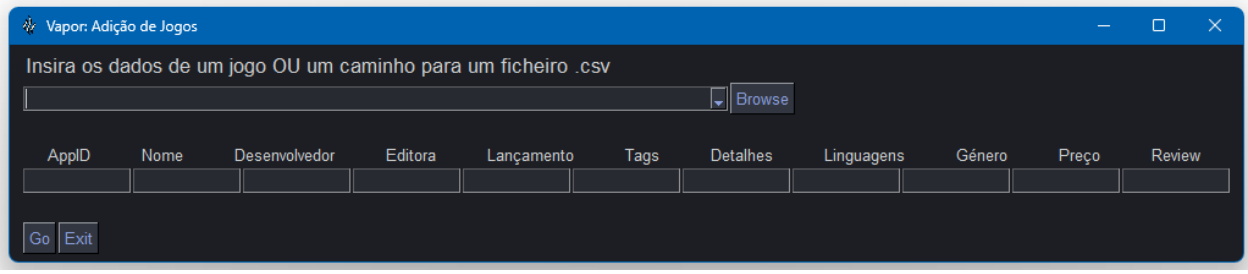
3. Realizando a Busca

a. Ao clicar no botão “Buscar” uma nova tela irá abrir com uma tabela contendo todos os jogos que batem com os filtros selecionados, bem como todas as informações sobre eles:

b. Nessa tela, será possível destacar um dos jogos clicando sobre ele, bem como ordenar de ordem crescente ou decrescente cada uma das colunas, apenas clicando sobre o nome da coluna.

4. Adicionando Informações

a. Ao clicar no botão “Adicionar” uma nova tela será exibida, solicitando que o usuário selecione um arquivo em CSV, ou digite manualmente as informações.



b. Ao optar por inserir novos dados, deve-se atentar para alguns detalhes
   - AppID deve ser um ”int”
   - Lançamento deve ser uma data no formato mmm DD, AAAA ou apenas um ano sendo os meses em inglês ex: apr, feb…
   - Preço deve conter um “$” antes do valor e ser separado por “.” seguindo o padrão americano.

c. O programa então adiciona todos os dados para a base de dados e fecha a janela de adição.

Considerações Adicionais

- Ao fazer a adição de novos jogos, caso o jogo já esteja na base de dados, as informações serão sobrescritas, ou seja, a informação mais recente é a que vai ser registrada.
- Alguns elementos podem aparecer mais de uma vez, por exemplo em idiomas temos “English e English, isso se deve ao fato de que a Steam deixa o preenchimento dos dados de Idioma, Dev, Publisher e alguns outros a critério de quem publica o jogo na plataforma, o que acarreta no fato de que algumas pessoas escrevem as informações de jeitos diferentes, como o exemplo citado acima.
- A base de dados utilizada originalmente continha caracteres não ASCII, o que levava a diversos erros de leitura, como quebras de linha e espaços em branco ou caracteres fantasmas. Tivemos então que realizar uma normalização nos campos com problema. Esses dados ainda estão presentes, como podemos observar se rolarmos todo o campo de devs ou publishers, vários nomes com caracteres estranhos aparecem.
- A data de lançamento também sofreu algumas alterações, visto que existe uma grande variedade no tipo de informação. Por exemplo, encontramos coisas como, TBA (To Be Announced), Coming Soon, entre outros. Para contornar isso, deixamos todas as datas que não seguem um padrão de mmm DD, AAAA em branco, pois esses valores também não serviriam para filtrar na busca por datas.

Compilação

Antes de explicarmos como compilar o código fonte, vamos abordar um pouco sobre as ferramentas e tecnologias que usamos, indo desde a biblioteca para a construção da interface gráfica, o módulo de integração entre C e Python, chamado Cython, até o compilador de python para .exe.

- ***REQUISITOS MINIMOS***
  - Sistema operacional Windows 10 ou acima
  - Microsoft Visual Studio Build Tools 2022
  - Python 3.11
  - PySimpleGUI
  - Cython
  - Auto-Py-To-Exe

1. ***Python 3.11***

   A instalação do Python 3.11 no Windows é bem simples, basta ir até a Microsoft Store e buscar por Python. Ao achar a opção desejada, é só clicar em adquirir e depois em instalar. Pronto, já está com Python na sua máquina. Para testar basta escrever ***Python -v*** no terminal. Esse comando deve retornar qual a versão do Python que está rodando no seu computador.

2. ***Biblioteca Gráfica***

   Mesmo sabendo que o programa poderia rodar em modo texto no terminal, buscamos uma forma de fazer uma interface mais amigável ao usuário. Para isso, decidimos utilizar uma biblioteca chamada PySimpleGUI.

   Essa biblioteca é uma forma simples e direta de fazer interfaces utilizando Python. Embora sua aparência seja bem simples e datada (aos moldes do saudoso windows 2000), existem muitas funções de elementos gráficos já prontos, o que facilitou muito a implementação, visto que nenhum de nós tinha experiência com interfaces antes.

   Para fazer a instalação é bastante simples. Desde que já tenha o Python instalado, basta digitar no terminal ***pip install PySimpleGUI***

3. ***Cython***

   O Cython é uma biblioteca python que utiliza as ferramentas de compilação do Microsoft Visual Studio para transformar programas e arquivos em C e C++ em novas bibliotecas Python, fazendo a interface entre as chamadas de função na linguagem de baixo nível para as chamadas de função em Python.

   Para instalar o Cython, é necessário rodar o comando ***pip install Cython*** que irá instalar todas as dependências do Python para compilar a nova biblioteca.

Para a biblioteca Cython realizar a transformação de um arquivo C ou C++ em Python, é necessário que o sistema tenha o MSVC (Microsoft Visual Studio Code) Build Tools 2022 instalado.

4. ***Microsoft Visual Studio 2022***

   Para a instalação do MSVC Build Tools, basta acessar o site da Microsoft, descendo na página até encontrar All Downloads, expandido a aba “Tools for Visual Studio” e então clicando no botão de Download associado à “Build Tools for Visual Studio 2022”.

   Caso você já tenha o Visual Studio Installer em seu sistema, é possível fazer download do Build Tools internamente, indo na aba de Available e então instalando o pacote de Desenvolvimento de C++ para Desktop.

5. ***Compilador Python para Exe***

   Aqui temos outra ferramenta bem simples mas que traz uma versatilidade incrível, o Auto-Py-To-Exe é um pacote Python, que cria um servidor web local na máquina do usuário e permite transformar qualquer arquivo .py em um arquivo executável, compilando todos os arquivos auxiliares e imagens ou outros assets necessários.

   Sua interface web é bem simples e intuitiva, permitindo várias configurações inclusive a adição de um ícone para o executável e se deseja utilizar o terminal ou escondê-lo.

   Para instalar o pacote é bem simples, assim como o PySimpleGUI, se já estiver com o Python instalado, basta digitar no terminal ***pip install Auto-Py-To-Exe***

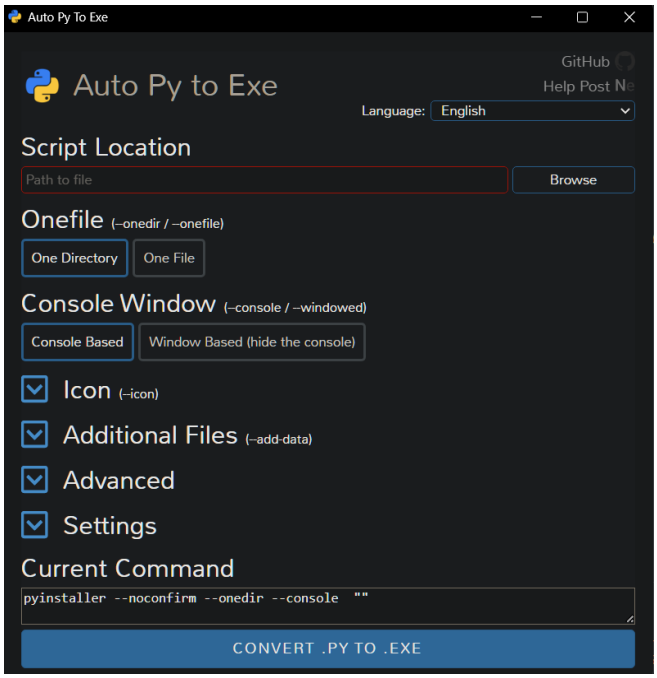
   Após instalado, basta rodar o comando ***Auto-Py-To-Exe*** no terminal, se tudo tiver sido instalado corretamente, uma página web deve abrir com a interface de compilação.

Realizando a Compilação Completa

Para juntar o programa do zero, é necessário entrar com o terminal na pasta “Code” disponível dentro do repositório. Após averiguar que todas as dependências estão instaladas, rode o seguinte comando: ***python setup.py build_ext --inplace*** que irá começar a transformar as funções em C++ em uma biblioteca Python.

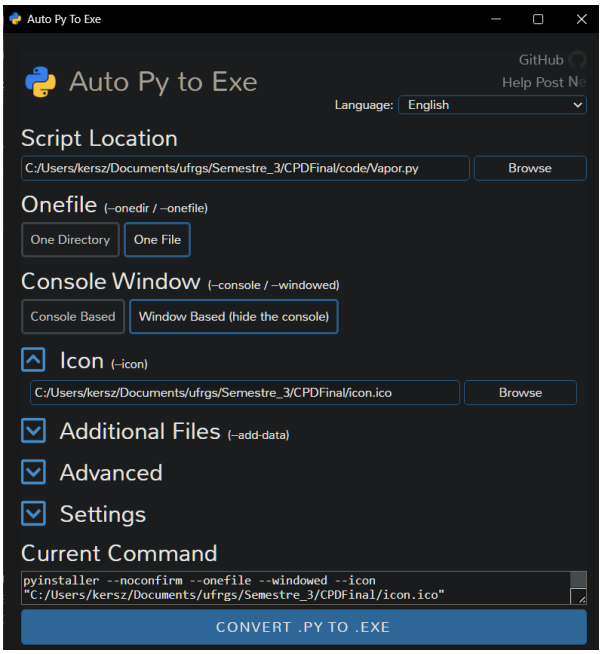
Depois que a execução do comando terminar de forma sucedida, basta rodar ***python vapor.py*** para abrir a interface gráfica.

Para realizar a compilação para .exe, no mesmo terminal, digite ***auto-py-to-exe*** onde então uma tela similar à esta deve abrir no navegador padrão:



Para efetuar a compilação, em “Script Location” digite o caminho para o Arquivo vapor.py, em “Onefile” selecione a opção One File, em “Console Window” selecione Window Based, após isso, talvez seja desejável selecionar o Ícone, simplesmente clicando na aba “Icon” e então digitando o caminho para o arquivo .ico disponível dentro do repositório.

Após todos os passos, a janela deve ser similar à esta:



Então, clique em “Convert .Py to .Exe” e o arquivo .exe completo deve estar disponível numa pasta chamada “output” dentro da pasta em que o comando foi chamado, seguindo o exemplo deste tutorial, a pasta deve ter sido criada dentro da pasta “code”.

Considerações Adicionais

- Devido ao fato do programa executar alguns comandos no terminal e mexer com o explorador de arquivos, o programa executável pronto pode ser erroneamente reconhecido como um vírus. Caso isso aconteça, basta compilar você mesmo o executável utilizando o Auto-Py-To-Exe ou desabilitar o antivírus para esse programa.
- Já deixamos o programa pré-compilado, isso significa que para transformar o Vapor.py em Vapor.exe basta rodar o Auto-Py-To-Exe sem a necessidade de baixar o ***Microsoft Visual Studio 2022*** e compilar as bibliotecas em C++.

Estruturas de Dados e Arquivos

Estrutura do Código

O código do programa está armazenado na pasta “Code” dentro do repositório, dentro dela há mais duas pastas, que diferenciam as partes do programa escritas em C++ e em Python. Devido a isso, cada parte será explicada separadamente.

C++

O programa em C++ tem como seu arquivo principal o “database.hpp”, onde as declarações das funções que serão utilizadas no programa Python estão armazenadas. Em conjunto temos “dabatase.cpp” que contém a implementação destas funções, que juntas formam a base do programa e operações na base de dados.

Dentro da pasta “classes”, é possível encontrar os arquivos de todas as funções majoritárias utilizadas, bem como suas implementações sendo: árvores, classes de jogos, tabelas e derivados. Cada um separado em suas pastas, dentro da subpasta "datastructures".

Python

A parte de python faz uso de uma estrutura muito mais simples, não contendo nenhuma subpasta.

Dentro da pasta principal, é possível encontrar os arquivos utilizados para Cython, sendo eles “Setup.py”, que faz a compilação da biblioteca, “DB.pyx”, que diz quais funções devem ser importadas, “DB.cp311-win_amd64.pyd”, que é a versão já compilada da bilioteca, e “database.py”, que faz o link entre as funções declaradas em Cython e seu uso nos arquivos Python.

Os arquivos Python puros são “images.py”, que contem as imagens utilizadas no programa, “GUI.py”, o qual armazena todas as funções para montar o menu interativo, e “Vapor.py”, o qual declara e inicia o programa em si, realizando todas os testes de segurança para minimizar eventuais crashes.

Estrutura de Dados

As estruturas de dados utilizadas na base de dados podem ser divididas em duas categorias: Armazenamento e Organização.

Armazenamento:

As estruturas utilizadas para armazenar dados durante o processo de conversão foram: uma versão modificada entre a árvore TRIE e Patrícia e uma tabela Hash. Inicialmente chegamos a pensar em utilizar uma tabela fixa e uma árvore B+, mas visto que a tabela hash era rápida o suficiente para suprir os benefícios que a tabela fixa poderia trazer, e a nossa base de dados não tinha um tamanho tão exacerbado a ponto de tirar muito proveito de uma árvore B+, essas duas foram abandonadas.

A busca de informações se dá de forma muito rápida, o maior gargalo está na ingestão dos dados para a interface gráfica, ou, output para o console.

Organização:

Para organizar o código da base de dados, foi criado uma classe chamada “Game”, a qual armazena o ID do jogo (int), seu nome, desenvolvedor, editora, tags (strings), data de lançamento, avaliações e preço. Cada uma possui suas estruturas de dados próprias para facilitar seu armazenamento.

Além disso, a classe faz uso da orientação a objetos para criar funções que auxiliam na manipulação desses dados, tal como, salvar e ler de arquivos, imprimir no terminal e passar todos seus parâmetros para um formato padrão entre os programas.

As avaliações foram criadas como um Enum, onde cada nome corresponde a um número, dessa forma é possível salvar de forma compacta todos os dados. Apesar disso, devido à forma que Enums funcionam em C++, diversas funções de suporte foram criadas, para converter entre Enum e Strings.

Para o preço, a classe “Price” foi criada, onde ela transforma e salva o preço, normalmente denotado em uma string na forma fracionária, em um número inteiro. Dessa forma mantemos sua precisão e diminuímos o espaço utilizado. Além disso, a conversão entre a classe e uma string de entrada foi criada, tão quanto uma função para re-converter o número inteiro em string.

A data de lançamento tem sua própria classe, a qual armazena o dia, ano e década de lançamento como números inteiros e o mês de lançamento como um Enum auxiliar. O Enum novamente necessita da criação de funções adicionais para a conversão entre ele e strings. Para facilitar o desenvolvimento, o construtor da classe foi criado para aceitar diferentes tipos de strings encontrados nos arquivos, tentando adequar e armazenar as datas corretamente. Caso não seja possível ou não encontre na string os elementos necessários, salva como “NaN”, dessa forma não parando a execução do programa.

Estrutura de Arquivos

Após a base de dados ser convertida pela primeira vez, o sistema de arquivos da aplicação é construído, e pode ser totalmente localizado na pasta “data”.

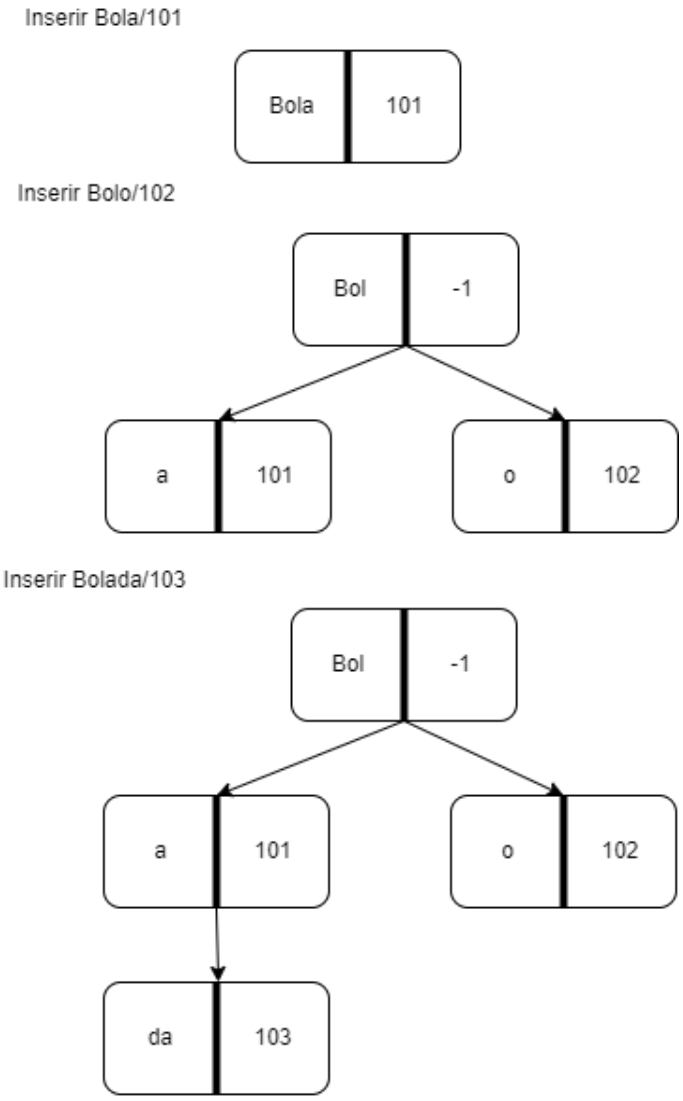
Dentro dela, se encontram diversos arquivos que auxiliam na pesquisa dos diversos critérios utilizados pela aplicação. Majoritariamente, são utilizados Arquivos Reversos, onde poucas palavras chaves contém uma lista de todos os itens que fazem seu uso, também chamado de postings. Esses arquivos contêm as extensões “.date”, “.dev”, “.genre”, “.lang”, “.price”, “.pub”, “.review” e “.tag”.

Para as pesquisas por nome, são utilizados os arquivos “.pat”, os quais armazenam a estrutura da árvore PATRÍCIA. Já no arquivo “.pat.str” são armazenadas as strings utilizadas, possibilitando uma reconstrução muito rápida.

O arquivo “.ids”, armazena a correlação entre AppIds, basicamente nosso dicionário, os quais são os identificadores unitários dos jogos, e o seu indice no arquivo primário “Steam.db”, o qual armazena todos os jogos de forma serializada em blocos de 1300 Bytes.

Considerações Adicionais:

- Nossa versão da TRIE/PATRICIA consiste em uma árvore N-ária, a qual tem nós onde sua chave principal é parte parcial da string e o valor que está contido no nó sendo -1 caso não tenha nenhum jogo que termine naquela substring ou, o AppID do jogo, caso exista um jogo com aquele ID. Dessa forma, o Zig-Zag é eliminado como em uma PATRÍCIA normal, mesmo não sendo uma árvore binária. Uma imagem descritiva pode ser vista abaixo.



Considerações Finais

Ambos nos divertimos muito e gostamos de realizar o trabalho, gostamos muito de programar e ter trabalhos na faculdade que possibilitam exercitar e melhorar nossas habilidades são sempre mais interessantes. Mas nem tudo é um mar de flores. Ter que conciliar todas as cadeiras, provas e outros trabalhos, vida profissional e pessoal durante esse semestre foi um desafio e tanto. Com isso em mente, decidimos pegar um conceito mais “simples” como a pesquisa na Steam, visto que gostaríamos de ter certeza que conseguimos entregar um MVP (Minimum Viable Product) bom o suficiente.

Ainda assim, foi um desafio bem complexo. Nos propusemos a aprender a mexer em interfaces gráficas e mexer em uma linguagem de programação que não estávamos acostumados. Então, além de realizar o trabalho, acabamos por aprender várias tecnologias novas e nos aprofundar em algumas áreas que já conhecemos. As aventuras por C++, Python, e, mais para o final do trabalho, Cython, foram ao mesmo tempo desafiadoras e muito satisfatórias, visto que conseguimos nos ver entendendo cada vez mais de cada linguagem, suas peculiaridades e casos de uso.

Porém, o tempo ainda foi o nosso maior inimigo, tendo que conciliar todo o caos do final de semestre, gostaríamos de ter tido um pouco mais de tempo livre para polir ainda mais a nossa aplicação, corrigir alguns problemas de layout, melhorar um pouco o tempo de resposta da aplicação, deixar ela mais bonitinha, entre outras coisas. Contudo, acreditamos que o trabalho ficou bem decente e robusto. Ter usado uma interface ajudou bastante a mantermos tudo modularizado. Ter descoberto como transformar C++ em Python também facilitou muito a integração entre os módulos. Acabamos por ter a facilidade de usar Python na GUI e a performance de C++.

Com a conclusão do trabalho e fazendo uma recapitulação geral para escrever esse relatório, percebemos várias coisas que podiam ser diferentes, como ter feito a obtenção dos dados diretamente da API da Steam, teríamos muito mais flexibilidade para trabalhar com os dados, ou então utilizar uma biblioteca que integrasse HTML e CSS para fazer a interface, talvez ficasse mais agradável aos olhos.

Além disso, ter utilizado a classe de strings com Unicode de C++, poderia ter nos poupado muita dor de cabeça que tivemos com os vários erros de caracteres não identificados. Junto a isso, um melhor tratamento nas exceções de data ajudariam muito a deixar a aplicação com um ar menos de “beta”, mas devido ao volume extremo de variações, isso se tornou uma atividade maçante, as quais foram então deixadas de lado, visto que, a maioria dos jogos não necessitam desse tratamento.

Referências

ANTONOV, A. Steam games complete dataset. Disponível em: <https://www.kaggle.com/datasets/trolukovich/steam-games-complete-dataset>. Acesso em: 28 fev. 2023.

auto-py-to-exe. Disponível em: <https://pypi.org/project/auto-py-to-exe/>. Acesso em: 3 abr. 2023.

cppreference.com. Disponível em: <https://en.cppreference.com/w/>. Acesso em: 16 mar. 2023.

Cython: C-Extensions for Python. Disponível em: <https://cython.org/>. Acesso em: 27 mar. 2023.

DOS, C. Árvore Patricia. Disponível em: <https://pt.wikipedia.org/wiki/%C3%81rvore_Patricia>. Acesso em: 23 mar. 2023.

Flaticon. Disponível em: <https://www.flaticon.com/br/>. Acesso em: 4 abr. 2023.

Hash Table Data Structure. Disponível em: <https://www.programiz.com/dsa/hash-table>. Acesso em: 16 mar. 2023.

MICROSOFT. Visual Studio Code. Disponível em: <https://code.visualstudio.com/>. Acesso em: 27 fev. 2023.

PySimpleGUI. Disponível em: <https://www.pysimplegui.org/en/latest/>. Acesso em: 13 mar. 2023.

Stack Overflow - Where Developers Learn, Share, & Build Careers. Disponível em: <https://stackoverflow.com/>. Acesso em: 13 mar. 2023.

Visual Studio: IDE and Code Editor for Software Developers and Teams. Disponível em: <https://visualstudio.microsoft.com/>. Acesso em: 27 mar. 2023.

Welcome to Python.org. Disponível em: <https://www.python.org/>. Acesso em: 13 mar. 2023.

WIKIPEDIA CONTRIBUTORS. Hash table. Disponível em: <https://en.wikipedia.org/wiki/Hash_table>. Acesso em: 16 mar. 2023.