Projeto de Estrutura de Dados I - UDF

Ciência da Computação N1_-_4142_ (Not) _RRR_UDF

Catálogo de Filmes em Python com aplicação de Estruturas de Dados, algoritmos de Ordenação e busca

Alunos:
Asafe Rodrigues Marinho - RGM: 37272110
Mauro Kauã de Lima de Souza - RGM: 37754246
Rayanne Nathália de Souza Matos - RGM 37171615

Brasília DF, 28 de maio de 2025

Introdução

A disciplina de Estrutura de Dados tem como objetivo fornecer os fundamentos teóricos e práticos para a organização eficiente de informações em programas de computador. Para consolidar esses conhecimentos, foi proposto o desenvolvimento de um projeto prático que envolvesse a aplicação direta de estruturas como listas, dicionários e algoritmos clássicos de busca e ordenação.

Diante disso, a equipe optou por desenvolver um sistema de catálogo de filmes em linguagem Python, utilizando como base um conjunto de dados reais extraídos do Kaggle. A escolha do tema “Filmes” foi motivada pela familiaridade e interesse geral no assunto, o que facilitou a análise e manipulação dos dados durante o desenvolvimento.

O projeto teve como foco principal a implementação de funcionalidades que permitissem ao usuário listar, buscar e ordenar filmes com base em diferentes critérios, aplicando algoritmos como busca linear e Quick Sort. A estrutura modular do código e o uso de boas práticas de programação também foram priorizados, visando facilitar a manutenção e a escalabilidade do sistema.

Além da interface em linha de comando (CLI), o sistema foi expandido com uma interface web simples, utilizando o microframework Flask, proporcionando uma experiência mais interativa e moderna. A proposta representa uma ponte entre teoria e prática, aproximando o aluno de situações reais de desenvolvimento de software e reforçando a importância do uso adequado de estruturas de dados para a construção de sistemas eficientes.

Repositório GitHub: github.com/RayanneMatos/catalogo-filmes-python

Escolha do tema

Com base nos exemplos fornecidos em sala de aula e disponibilizados no BlackBoard, a equipe decidiu fazer o projeto de busca e ordenação com o tema “Filmes”.

Criação de repositório no GitHub:

Após a escolha do tema a ser elaborado no projeto de Estruturas de Dados, o primeiro passo foi a criação do repositório GitHub com todos os colaboradores e a estrutura básica contendo os arquivos:

- dados
  - movies_catalog_clean.csv
- frontend
  - index.html
- listar.py
- busca.py
- catalogo.py
- main.py
- ordenacao.py
- utils.py
- app.py
- Readme.md

Arquivo CSV:

1. Extração de CSV:

O arquivo bruto **'The Movies Dataset'** foi extraído do site Kaggle, onde o arquivo principal de metadados de filmes contém informações sobre 45.000 filmes apresentados no conjunto de dados Full MovieLens. Os recursos incluem pôsteres, fundos, orçamento, receita, datas de lançamento, idiomas, países de produção e empresas.

2. Limpeza de Dados com Python:

Utilizando o Google Colab integrado ao Google Drive, um dos colaboradores do projeto desenvolveu um código simples com o objetivo de extrair uma amostra do arquivo original.

A partir do conjunto bruto de dados, foi criada uma versão reduzida contendo apenas os 1000 primeiros registros de filmes.

Esse novo arquivo foi salvo no ambiente local do Colab e disponibilizado para download, facilitando a manipulação e análise inicial dos dados.

3. Montagem do Google Drive no Colab:

```
[ ]  1  from google.colab import drive
     2  drive.mount('/content/drive')
     3
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

4. Manipulação de Dados com a Biblioteca Pandas:

Utilizando a biblioteca Pandas, realizamos a manipulação dos dados seguindo as etapas abaixo:

a. Definimos o caminho do arquivo movies_metadata.csv, armazenado no Google Drive.
b. Carregamos o conteúdo do arquivo para um DataFrame chamado df.
c. Extraímos as 1000 primeiras linhas de df, criando um novo DataFrame com essa amostra.
d. Salvamos este subconjunto em um novo arquivo chamado *movies_metadata_small.csv*, no ambiente local do Google Colab.
e. Exibimos uma mensagem de confirmação informando que o processo foi concluído com sucesso.
f. Utilizamos a função *files.download()* para disponibilizar o novo arquivo CSV para download direto no computador do usuário.

```
[ ]  1  import pandas as pd
     2
     3  caminho = "/content/drive/MyDrive/3_semestre/EstruturaDados1-kadija/movies_metadata.csv"
     4
     5  df = pd.read_csv(caminho, low_memory=False)
     6
     7  df_small = df.head(1000)
     8  df_small.to_csv("movies_metadata_small.csv", index=False)
     9
    10  print("Arquivo menor criado com sucesso!")
    11
```

```
Arquivo menor criado com sucesso!
```

```
     1  from google.colab import files
     2  files.download("movies_metadata_small.csv")
     3
```

Scripts Python:

A seguir iremos explicar cada arquivo do nosso repositório:

1. catalogo.py

Este código foi desenvolvido com o objetivo de carregar e processar os dados do arquivo Comma-Separated Values (CSV) chamado movies_catalog_clean.csv. Para isso, utilizamos a biblioteca padrão csv do Python, que facilita a leitura e escrita de arquivos nesse formato.

Primeiramente, definimos o caminho do arquivo a ser lido, localizado na pasta dados. Em seguida, criamos a função *carregar_dados(),* que é responsável por abrir o arquivo, processar seu conteúdo e retornar os dados já preparados para uso.

Dentro da função, inicializamos uma lista chamada filmes, que armazenará cada registro de filme. O arquivo é aberto utilizando a codificação UTF-8, garantindo a leitura correta de caracteres especiais.

Para ler as linhas do arquivo, usamos o csv.DictReader, que transforma cada linha em um dicionário, onde as chaves correspondem aos nomes das colunas.

Para evitar erros nas etapas posteriores do processamento, realizamos conversões específicas para assegurar que os dados estejam nos formatos corretos. Por exemplo, os campos vote_average, runtime, budget e revenue são convertidos para float ou int, conforme apropriado. Nos casos em que algum campo está em branco, é atribuído o valor padrão zero.

Além disso, o campo genres_names, que originalmente armazena os gêneros do filme como uma única string separada por vírgulas, é transformado em uma lista de gêneros. Isso é feito utilizando a função split(","), seguida do método strip() para remover espaços em branco em torno de cada gênero.

Após esse tratamento, o dicionário com os dados do filme é adicionado à lista filmes. Ao final da execução da função, essa lista é retornada, contendo todos os filmes com seus dados devidamente tratados e prontos para análise.

2. listar.py

Este código tem como objetivo listar de forma detalhada as informações dos filmes carregados a partir do arquivo CSV tratado pelo módulo *catalogo.py*. Para isso, ele importa a função *carregar_dados* do arquivo catalogo.py para obter os dados dos filmes.

Primeiramente, o código define uma função chamada l*istar_filmes_detalhado(filmes)*, que recebe como parâmetro uma lista de filmes, onde cada filme é representado por um dicionário com seus respectivos atributos.

Dentro da função, há uma verificação inicial para identificar se a lista está vazia. Caso não haja nenhum filme cadastrado, a função exibe a mensagem "Nenhum filme cadastrado para listar." e encerra sua execução.

Logo após, é definido um dicionário chamado idiomas_map, que faz o mapeamento das siglas dos idiomas (códigos ISO 639-1) para seus nomes completos em inglês. Isso facilita

a apresentação dos nomes dos idiomas em formato mais amigável ao usuário. Caso o idioma não esteja listado, o código exibe a sigla em letras maiúsculas, e se o campo estiver vazio, é mostrada a palavra "Desconhecida".

A função então imprime um cabeçalho indicando que os filmes cadastrados serão listados. Em seguida, percorre cada filme da lista, exibindo suas informações detalhadas. Para cada filme, são mostrados:

- O título do filme, ou “Título Desconhecido” caso o título não esteja disponível.
- O ano de lançamento, ou “N/A” (não disponível) se o dado estiver ausente.
- Os gêneros do filme, formatados como uma lista separada por vírgulas, ou “N/A” caso não haja essa informação.
- A média de votos, apresentada com uma casa decimal.
- Uma descrição ou resumo do filme, ou “N/A” caso não esteja disponível.
- A duração em minutos.
- O idioma original do filme, exibido em seu nome completo conforme o dicionário de mapeamento.
- O orçamento e a receita do filme, formatados com vírgulas como separadores de milhares para melhor legibilidade.

Por fim, o script contém um bloco condicional *if __name__ == "__main__":* que garante que, se o arquivo for executado diretamente, os dados dos filmes sejam carregados por meio da função *carregar_dados()* e então listados utilizando a função *listar_filmes_detalhado().*

Dessa forma, o código permite uma visualização clara e organizada dos dados dos filmes, facilitando a análise e exploração das informações presentes no catálogo.

3. ordenacao.py

Este módulo implementa uma função de ordenação para um catálogo de filmes utilizando o algoritmo Quick Sort, um método eficiente e amplamente usado para ordenar listas.

O código importa a função listar_filmes_detalhado do módulo listar.py, que pode ser usada para exibir os filmes após a ordenação.

A função principal do módulo é quick_sort(lista, chave, inicio=0, fim=None). Ela recebe uma lista de filmes, a chave pela qual deseja ordenar (como título, ano ou nota), e os índices de início e fim do trecho da lista a ser ordenado. Se o índice fim não for fornecido, a função considera o último elemento da lista.

O algoritmo Quick Sort funciona dividindo a lista em partes menores por meio de uma função auxiliar chamada particao. Essa função escolhe um elemento pivô (neste caso, o último elemento da sublista) e reorganiza a lista de forma que todos os elementos menores ou iguais ao pivô fiquem antes dele, e os maiores fiquem depois. Depois, o Quick Sort é chamado recursivamente para ordenar as partes à esquerda e à direita do pivô.

A função comparar(a, b, chave) é responsável por comparar dois filmes com base na chave de ordenação especificada. Para isso, utiliza a função extrair_valor(filme, chave), que extrai

o valor correto do filme para comparação, garantindo que os dados sejam interpretados corretamente conforme o tipo:

- Se a chave for "title", retorna o título do filme em letras minúsculas para comparação alfabética.
- Se a chave for "ano", tenta extrair o ano da data de lançamento (os quatro primeiros caracteres da string release_date), convertendo para inteiro. Se o campo estiver vazio ou inválido, retorna 0.
- Se a chave for "nota", retorna a média de votos (vote_average) como número de ponto flutuante, tratando possíveis erros e valores ausentes.

Caso uma chave inválida seja passada, a ordenação padrão será pelo título do filme.

Por fim, a função ordenar_catalogo(catalogo, chave_ordenacao) é a interface que deve ser usada para ordenar o catálogo. Ela verifica se a lista está vazia e se a chave de ordenação é válida, exibindo mensagens de aviso caso contrário. Em seguida, chama a função quick_sort para ordenar a lista conforme a chave escolhida.

Este módulo permite ordenar o catálogo de filmes de forma flexível e eficiente, com base em diferentes critérios, facilitando a análise e visualização organizada dos dados.

4. busca.py

Este módulo foi criado para realizar buscas em um catálogo de filmes, permitindo localizar títulos específicos ou filmes de determinado gênero. Ele implementa duas funções principais que utilizam buscas lineares — uma estratégia simples e eficiente para conjuntos de dados de pequeno a médio porte.

4.1. Função *buscar_por_titulo(catalogo, titulo)*

Essa função percorre todo o catálogo de filmes e retorna uma lista contendo todos os filmes cujo título contém o termo buscado. A busca é case insensitive, ou seja, não faz distinção entre letras maiusculas e minúsculas. Isso é feito convertendo tanto o título do filme quanto o termo buscado para letras minúsculas com o método .lower().

Por exemplo, se o usuário procurar por "Batman", a função retornará filmes como "Batman Begins" e "The Lego Batman Movie".

4.2. Função buscar_por_genero*(catalogo, genero)*

Esta função permite buscar filmes que pertençam a um determinado gênero, como "Action", "Drama" ou "Comedy". A entrada do usuário (o nome do gênero) também é convertida para minúsculas para garantir uma busca insensível a maiúsculas/minúsculas.

A função percorre o catálogo e utiliza uma *list comprehension* com o método any(), verificando se o gênero informado está presente em algum dos gêneros associados ao filme (armazenados na lista *genres_names*).

Assim, se um filme tiver os gêneros ["Action", "Adventure"], ele será incluído nos resultados de busca por "action" ou "adventure", independentemente da capitalização das letras.

5. main.py

Em suma, o arquivo main.py é o controlador da aplicação, orquestrando a interação com o usuário e integrando os diferentes módulos para permitir um uso simples e funcional do catálogo de filmes via terminal.

O arquivo main.py implementa o ponto de entrada do programa de gerenciamento do catálogo de filmes. Ele é responsável por apresentar um menu interativo ao usuário, permitindo navegar entre as funcionalidades principais: listar filmes, realizar buscas, ordenar os filmes e sair do sistema.

O código importa funções essenciais de outros módulos do projeto:

- carregar_dados do módulo catalogo: função que carrega e retorna os dados do arquivo CSV de filmes;
- listar_filmes_detalhado do módulo listar: função que exibe os filmes com informações completas;
- menu_busca e menu_ordenacao do módulo utils: funções que implementam os menus interativos para busca e ordenação dos filmes, respectivamente.

A função menu() exibe no terminal as opções disponíveis para o usuário:

1. Listar todos os filmes;
2. Buscar filmes;
3. Ordenar filmes;
4. Sair do programa.

Após a exibição, aguarda a entrada do usuário para escolher uma opção, que é então retornada para controle do fluxo principal.

Já ao iniciar, a função main() chama carregar_dados() para ler e preparar a lista de filmes, armazenando-a na variável catalogo. Isso garante que o catálogo estará disponível para todas as operações subsequentes.

Em seguida, inicia-se um loop infinito onde o menu é exibido repetidamente, e a escolha do usuário é processada conforme segue:

1. Se o usuário escolher a opção 1, o programa chama listar_filmes_detalhado(catalogo) para imprimir a lista completa de filmes com seus detalhes;
2. Se for a opção 2, o controle é passado para o submenu de busca (menu_busca), que permite filtrar filmes conforme critérios desejados;
3. Se for a opção 3, o controle vai para o submenu de ordenação (menu_ordenacao), onde o usuário pode ordenar o catálogo conforme título, ano ou nota;

4. Se for a opção 4, o programa exibe uma mensagem de despedida e encerra o loop, finalizando a execução;
5. Para qualquer outra entrada inválida, o programa exibe uma mensagem de erro e volta a exibir o menu principal.

A linha if __name__ == "__main__": assegura que a função main() será executada apenas quando o arquivo main.py for rodado diretamente, evitando que o código principal seja executado caso o módulo seja importado por outro script.

6. utils.py

O módulo utils.py reúne funções auxiliares que fornecem interfaces interativas para duas funcionalidades principais do sistema: busca e ordenação de filmes. Ele organiza essas operações em menus dedicados que são chamados a partir do programa principal (main.py), facilitando a interação do usuário com o catálogo de filmes.

O módulo utils.py fornece uma camada de interação fundamental para o funcionamento do sistema. Ao encapsular os menus de busca e ordenação, ele organiza a lógica de interface com o usuário, separando-a da lógica de processamento de dados, o que torna o código mais modular, reutilizável e fácil de manter.

As funções utilizadas neste módulo dependem de outros módulos do projeto:

- buscar_por_titulo e buscar_por_genero do módulo busca: executam buscas específicas no catálogo de filmes;
- ordenar_catalogo do módulo ordenacao: aplica o algoritmo de ordenação (Quick Sort) com base na chave escolhida;
- listar_filmes_detalhado do módulo listar: imprime a lista de filmes ordenada com todos os detalhes.

Função menu_busca(catalogo)

Esta função exibe um menu de busca que permite ao usuário procurar filmes por dois critérios:

- Título: Realiza uma busca linear e case insensitive, comparando a string digitada com os títulos dos filmes;
- Gênero: Verifica se o gênero informado está presente na lista de gêneros (genres_names) de cada filme.

Caso o usuário selecione uma dessas opções, a função solicita o termo de busca, executa a busca correspondente e exibe os resultados de forma resumida, com o título e o ano de lançamento (release_date) de cada filme encontrado.

Se não houver resultados, é exibida uma mensagem informando que nenhum filme foi encontrado.

A opção 3 permite retornar ao menu principal, enquanto qualquer entrada inválida é tratada com uma mensagem de aviso.

Função menu_ordenacao(catalogo)

A função menu_ordenacao apresenta um menu para o usuário escolher um critério de ordenação do catálogo. As opções disponíveis são:

- Título: Ordena os filmes em ordem alfabética (A-Z);
- Ano: Ordena os filmes do mais antigo para o mais recente, com base no ano de lançamento;
- Nota: Ordena os filmes da menor para a maior nota média;
- Voltar ao Menu Principal.

Quando o usuário escolhe um critério válido, a função chama ordenar_catalogo, que executa a ordenação com base na chave informada. Em seguida, exibe uma mensagem de confirmação e chama listar_filmes_detalhado para apresentar os filmes ordenados. Após isso, a função interrompe o laço (break) e retorna ao menu principal.

Caso o usuário escolha a opção 4, a função apenas retorna ao menu principal, sem executar nenhuma ordenação.

Entradas inválidas são tratadas com uma mensagem de erro, solicitando nova tentativa.

7. app.py

O arquivo app.py é responsável por disponibilizar o catálogo de filmes como uma API web utilizando o microframework Flask. Ele define a estrutura principal de uma aplicação web que expõe rotas específicas para listar todos os filmes ou realizar buscas por título ou gênero. Dessa forma, permite que outras aplicações, como interfaces web, possam se comunicar com o sistema e acessar os dados do catálogo de forma prática e organizada.

Na inicialização do sistema, os dados do catálogo são carregados uma única vez e armazenados em memória, o que garante eficiência durante as requisições. A aplicação fornece uma rota principal que serve o arquivo HTML da interface localizada na pasta frontend, permitindo a integração com uma camada visual do sistema.

Além da rota inicial, o arquivo define uma rota para retornar todos os filmes em formato JSON, bem como rotas específicas para buscar filmes por título ou por gênero. Essas buscas são realizadas de forma insensível a maiusculas e minúsculas, tornando a experiência do usuário mais flexível.

A aplicação é executada em modo de desenvolvimento (debug), o que facilita a identificação de erros e a atualização automática durante a construção do sistema. Em resumo, app.py é o ponto de entrada do servidor e atua como intermediário entre a lógica interna do catálogo de filmes e o consumo externo via web.

Frontend:

1. index.html

O arquivo HTML apresentado define a interface web do Catálogo de Filmes, permitindo ao usuário interagir com as funcionalidades do sistema de forma intuitiva e visual. Ele utiliza o framework Bootstrap para garantir um layout responsivo e organizado, com botões que acionam diferentes funcionalidades da aplicação.

A interface oferece quatro ações principais: listar todos os filmes, buscar por título, buscar por gênero e ordenar filmes por título. Quando o usuário clica em uma dessas opções, a aplicação envia requisições para a API Flask definida no backend, obtendo os dados em formato JSON e exibindo-os dinamicamente na página.

A listagem e os resultados das buscas são apresentados em uma área específica da página, com os títulos e gêneros dos filmes formatados em uma lista. Caso não haja resultados para uma busca, uma mensagem informativa é exibida. A funcionalidade de ordenação é realizada diretamente no navegador, utilizando a lista de filmes já carregada e organizando-os alfabeticamente pelo título.

Por fim, o HTML também inclui um estilo básico para modais, mesmo que eles não estejam em uso nesta versão. Em resumo, este arquivo representa a camada de apresentação do sistema, conectando-se ao backend para fornecer uma experiência interativa e acessível ao usuário final.

Melhorias Futuras

Apesar das funcionalidades desenvolvidas atenderem aos objetivos principais da disciplina, identificamos algumas oportunidades de aprimoramento que podem ser exploradas em versões futuras do projeto:

1. CRUD completo: Atualmente, o sistema oferece funcionalidades de leitura, busca e ordenação dos filmes, mas não contempla as operações de criação, atualização e exclusão de registros. A inclusão dessas funcionalidades permitirá que o sistema se torne um gerenciador de catálogo mais completo e dinâmico.
2. Internacionalização dos dados: Os dados foram mantidos no idioma original (inglês), preservando o padrão do dataset extraído do Kaggle. Como sugestão de melhoria, propõe-se a tradução de campos como nomes de gêneros, idiomas e datas para o português, melhorando a acessibilidade e usabilidade para o público local.
3. Validações e tratamento de erros: Implementar verificações adicionais de entrada do usuário e mensagens de erro personalizadas pode tornar o sistema mais robusto e intuitivo.
4. Testes automatizados: A inclusão de testes unitários, utilizando bibliotecas como pytest, poderá garantir maior confiabilidade das funcionalidades e facilitar a manutenção do sistema.

5. Exportação de dados filtrados: Funcionalidades que permitam exportar os resultados de buscas ou ordenações para arquivos .csv ou .json dariam maior utilidade prática ao sistema.
6. Aprimoramento da interface web: A interface atual, embora funcional, pode ser expandida com o uso de frameworks mais robustos para front-end, além de melhorias no design responsivo e na interação com o usuário.

Conclusão

O projeto “Catálogo de Filmes” proporcionou à equipe uma oportunidade prática de aplicar conceitos fundamentais da disciplina de Estrutura de Dados, como manipulação de listas e dicionários, busca linear, e ordenação com Quick Sort. Além disso, o trabalho envolveu a leitura e tratamento de arquivos CSV reais, integração com bibliotecas como Pandas, modularização do código Python e criação de uma interface web com Flask e Bootstrap.

A experiência permitiu consolidar conhecimentos teóricos, desenvolver habilidades práticas de programação e compreender a importância da organização de código em projetos reais. O sistema está funcional e pronto para uso básico, e as sugestões de melhorias futuras abrem caminho para sua evolução em direção a um produto mais completo, interativo e profissional.