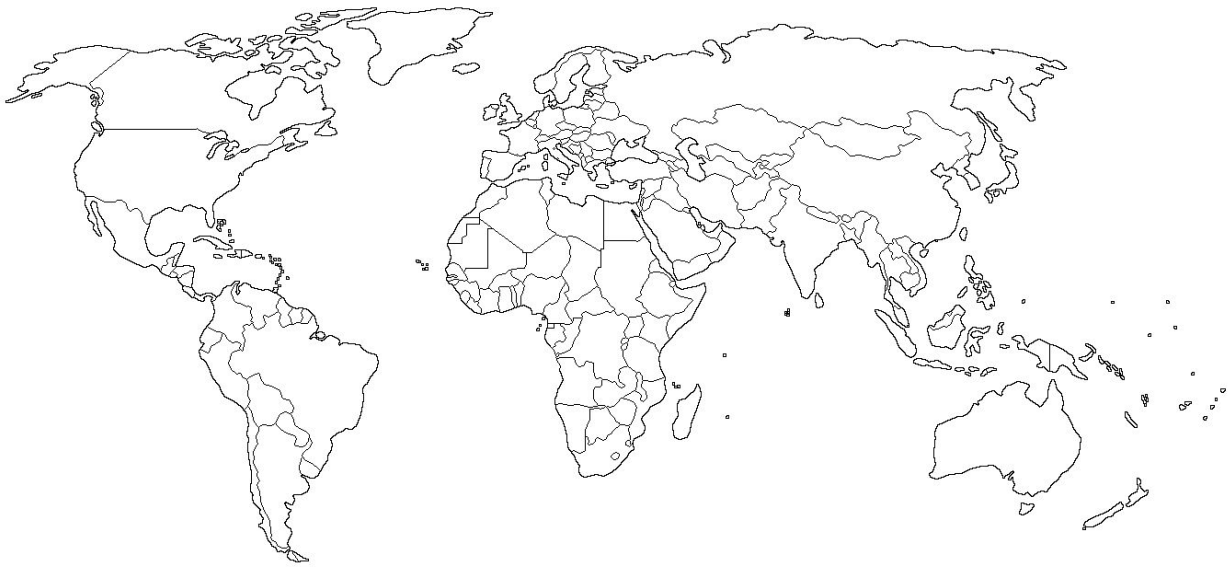
INF 01124 - **Classificação e Pesquisa de Dados**

Professor Leandro Krug Wives

Turma A

GEOINDICADORES - CONSULTA, LISTAGEM E COMPARAÇÃO



Grupo Kebab:

Germano Martinelli - 216684

Henrique Silva - 262508

Lauren Rolan - 262517

Premissa:

Coletar e analisar dados geográficos e geopolíticos disponibilizados pelas APIs PyCountry, Wikipedia e pelos arquivos .csv encontrados na Internet, disponibilizando uma gama de opções aos usuários sobre como observar e interpretar as informações coletadas, numa interface de uso gráfica *rudimentar* porém *agradável*.

Implementação e Considerações Técnicas:

Utilizamos Python como linguagem de programação, por seu foco em produtividade, limpeza e fluidez no processo de desenvolvimento. Além disso, na fase inicial de planejamento, utilizamos a plataforma Trello, a fim de organizar e otimizar o processo de desenvolvimento da aplicação.

Como carro-chefe da fase de desenvolvimento em si, utilizamos o GitHub, que garante a possibilidade de manter o código sempre atualizado e facilmente editável para todos os membros, além de oferecer um possível "rollback" em casos de erros extremos.

Durante o processo de exploração (tanto manual quanto por meio de APIs) em busca dos dados, constatamos que muitas fontes tinham dados incompletos (por exemplo, a falta de indicadores de IDH de muitos países) ou ainda dados supérfluos (informações sobre territórios não-soberanos, como muitos arquipélagos, cujos dados já estavam contemplados em seu país administrador).

Isso é uma situação muito comum - é a norma, aliás - quando se trabalha com coleta e análise de dados. Resta aos desenvolvedores lidar com isso da melhor forma possível.

Experimentamos alguns APIs, como Wikipedia e Pycountry. Deles, retiramos alguns dados - mas a fonte principal veio a ser o website *www.worlddata.info* e o website http://data.worldbank.org/, que forneciam arquivos .csv para download com os indicadores desejados. Fizemos o download e criamos funções para tratar os arquivos, usando a própria biblioteca *csv* do Python.

Outros dados, coletamos completamente a mão: os mapas e as bandeiras de cada país.

Desde o início do planejamento, concordamos que seria necessária uma interface gráfica, ainda que simples - pois ainda temos pouca experiência com elas - para que o trabalho ficasse apresentável. Decidimos pelo uso da package *TkInter*, pois foi, dentre as experimentadas, a que consideramos mais clara e simples de se programar.

Conforme o enunciado do trabalho pedia, escolhemos uma das estruturas de dados estudadas na área 2 da disciplina para servir de alicerce para nosso banco de dados. A escolhida foi a árvore B. Pesquisamos um objeto BTree já implementado, mas realizamos

alguns acréscimos, necessários para que tivéssemos todas as funcionalidades necessárias para cumprir nosso objetivo. A mais notável dessas modificações foi a implementação de caminhamentos - crescente e decrescente - na árvore B, para implementar a funcionalidade Listing (ordena os países de forma crescente ou decrescente de acordo com um quesito - por exemplo, IDH) no programa final.

Como introdução informal aos futuros estudos de Orientação a Objeto, criamos um *objPais* como tipo de dado elementar do banco de dados. Alguns membros do grupo lançaram uso de experiência básica prévia para isso, e outros aprenderam rapidamente os rudimentos do uso de objetos.

Para cumprir a outra exigência do enunciado - o uso de arquivos binários -, utilizamos a package *Pickle*: um serializador. Tivemos que usá-la porque a manipulação de arquivos em baixo nível é difícil em Python: a linguagem foi feita pensando no alto nível. Até mesmo o uso de ponteiros é extremamente dificultado, de forma que precisamos usar o *work-around* de trabalhar com índices na árvore B.

Durante a execução do aplicativo, também optamos por carregar a árvore B inteira para a memória durante a inicialização. Essa decisão foi tomada pois não conseguíamos salvar ou carregar o arquivo em blocos - dadas as dificuldades e limitações do Python em si, ao tentar trabalhar em baixo nível, e do serializador Pickle. Isso suscita uma discussão saudável sobre vantagens e desvantagens de cada linguagem de programação, e sobre a importância e as consequências dessa escolha para um projeto.

Uma consequência da forma com que conduzimos esse projeto é que ele é completamente offline. A única interação cabível com a web seria a troca do .csv do www.worlddata.info por uma versão mais atualizada - visto que o site mantém os indicadores em dia -, mas optamos por não implementar essa funcionalidade.

Além disso, fizemos o projeto em inglês. A principal razão foi técnica: no início do desenvolvimento, tivemos várias complicações com codificação Unicode ao tentar trabalhar em português no Python, então, em vez de gastar tempo depurando detalhadamente esse problema, optamos pelo inglês. Apesar disso, utilizamos o módulo Unidecode em alguns tratamentos de entrada.

Guia de Uso:

O programa tem três funcionalidades:

- **Display:** dado o nome de um país, que deve ser escrito na caixa de texto, o aplicativo apresenta todos os dados sobre ele.

- **Listing:** a partir de um dado numérico, ou seja, um dado para o qual faça sentido uma ordenação, escrito na caixa de texto, apresenta uma lista em ordem crescente ou decrescente (a critério do usuário) dos países, ordenados em relação a esse dado.
- **Comparison:** dados os nomes de dois países, apresenta seus dados lado a lado e compara os numéricos, destacando o maior.

O sistema de entrada é um pouco rudimentar por causa da ainda pequena experiência dos membros do grupo em interface gráfica, o que implica uma menor *user-friendliness*. Dessa forma, esse breve manual se faz necessário para que um usuário não-envolvido no processo de desenvolvimento possa aproveitar o aplicativo.

Para os campos em que se pede um país, é fundamental que o país seja escrito em inglês, e com a caixa correta das letras.

Exemplos: "Brazil", "United States" (e não United states), "Poland", "Paraguay", "Russian Federation" (e não Russia).

Alguns países com nomes irregulares: "Korea, Democratic People's Republic of" (Norte) e "Korea, Republic of" (Sul)

Devido a algumas dificuldades na elaboração da busca dentro da árvore B - envolvendo tanto a estrutura em si quanto o uso de strings em Python - a busca de países pode apresentar ocasionais erros. Contudo, funciona na grande maioria dos testes. Caso aconteça algum erro, o usuário deve buscar outro país.

Para os campos em que se pede um dado não há *sensibilidade de caixa*, uma vez que a entrada é padronizada usando Unidecode - mas é necessário que o dado seja suportado pelo programa, caso contrário, um erro é gerado. (*"There is no such data"*)

São suportados os segintes dados:
- *Population*
- *Coast Area*
- *Area*
- *GINI*
- *HDI*
- *GDP*
- *Life Expectancy*
- *Currency Exchange*
- *Tourism*

- *Inflation*

Considerações Finais:

Este trabalho serviu, para o grupo, como uma experiência empírica de vários conceitos diversos da Computação:

- **Programação orientada a objeto:** Nesse trabalho, utilizamos OOP tanto ao elaborar um objeto que representa um país, quanto também ao utilizar códigos de terceiros.
- **Geoprocessamento:** O tema do trabalho foi influenciado e inspirado pelo interesse dos membros do grupo nessa área do conhecimento. Os GIS são uma importante união de duas profissões distintas: a Computação e a Geografia - e estão presentes tanto nas aplicações mais críticas da Humanidade quanto nas mais corriqueiras - desde a NASA até o Foursquare.
- **Busca e análise de dados:** Esse trabalho foi uma experiência de como trabalhos reais com dados funcionam: dados insuficientes, dados supérfluos, e as decisões que isso implica. Muitas vezes, em ambiente de aula, nos são fornecidos datasets “perfeitos” - que não correspondem à realidade do trabalho com dados.
- **Banco de dados:** As estruturas de dados aprendidas na disciplina e utilizadas no trabalho são a base para estruturas bem mais complexas, que compõem os bancos de dados atuais.

Além disso, algumas outras habilidades fundamentais, não estritamente técnicas mas sim “transversais”, foram desenvolvidas e exercitadas:

- **Uso de repositórios de código:** O GitHub mostrou ser uma ferramenta utilíssima. Até este trabalho, os membros do grupo sempre haviam trabalhado de forma ‘arcaica’, trocando arquivos de código entre si, e muitas vezes perdendo versões pelo caminho. O uso de repositórios torna o processo muito mais organizado
- **Uso de plataformas de comunicação e planejamento pela equipe:** Aplicações como o Trello permitem uma organização fácil e prática das fases do trabalho - são melhores que organização apenas oral ou em listas manuscritas, pois são de fácil atualização e consulta.
- **Identificação de referências confiáveis:** Todo programador utiliza materiais de referência - apenas programadores sobre-humanos seriam capazes de

decorar cada minúcia conceitual ou sintática que precisarão usar em um projeto, ou sabê-las todas de antemão. O bom programador é aquele que sabe discernir entre bons e maus materiais de referência, e também entender os princípios e funcionamento do conceito que ora busca, ao invés de usá-lo cegamente.

- **Adaptação de código de terceiros:** Uma habilidade importantíssima, ubícua na Computação, é saber interpretar, usar, e às vezes até mesmo modificar o código feito por outros. Este vem em diferentes níveis de clareza - devemos estar conscientes disso, e, a cada dificuldade que temos quando interpretamos código alheio, nos conscientizar da importância de escrever um código limpo e bem comentado.

Referências:

http://www.pythonlearn.com/book.php - Professor Charles Severance, da Universidade de Michigan. Seu livro Python for Informatics fundamentou a base teórica necessária para a realização do trabalho, e foi um fator decisivo para a escolha da linguagem de programação

Indicadores e .csv's:
https://www.oanda.com/lang/pt/currency/table
http://data.worldbank.org/indicator
https://github.com/datasets/country-codes/blob/master/data/country-codes.csv
https://datahub.io/dataset/iso-3166-1-alpha-2-country-codes/resource/9c3b30dd-f5f3-4bbe-a3cb-d7b2c21d66ce
https://www.worlddata.info/downloads/
http://hdr.undp.org/en/composite/HDI

Documentações dos módulos utilizados:
BTree - https://gist.github.com/teepark/572734
Pickle - https://docs.python.org/2/library/pickle.html
TkInter - https://docs.python.org/2/library/tkinter.html
Unidecode - https://pypi.python.org/pypi/Unidecode

Documentações das APIs utilizadas:
Wikipedia - https://www.mediawiki.org/wiki/API:Main_page
Pycountry - https://pypi.python.org/pypi/pycountry