

Global Solutions – Computational Thinking with Python

FIAP – Engenharia de Software

DownSkilling

Guilherme Satler Macedo – RM 563330

Matheus Freitas Vieira – RM 566198

1. Introdução

1.1. A ideia

DownSkilling, o nosso termo, conceito e possível trabalho de futuro, se refere a simplificação de trabalhos com a utilização de tecnologia. Com constantes inovações tecnológicas, cada vez mais avançadas, o baixo nível de educação e escolaridade do Brasil não permite que todos os cidadãos consigam usufruir dessas tecnologias. Ferramentas que poderiam facilmente melhorar suas qualidades de vida dentro e fora do trabalho, muitas vezes não são utilizadas pois não possuem conhecimento suficiente para conseguirem utilizar estas ferramentas.

Com isso, nossa missão é criar softwares e ferramentas para facilitar e ajudar o cidadão Brasileiro a executar tarefas que esteja além do seu nível de conhecimento. A partir de softwares e ferramentas extremamente intuitivas para que qualquer pessoa seja capaz de utilizar. Junto de um parceiro de Inteligência Artificial para guiar e explicar qualquer parte do processo que o usuário precise entender.

Ressaltando que *DownSkilling* não é uma única ferramenta ou software, mas uma categoria de software, já que não seria prático e eficiente criar uma única solução para englobar diversas indústrias, principalmente quando o objetivo é simplificar. Assim as ferramentas de *DownSkilling* seriam feitas com uma única indústria em mente, para que assim possam ser aperfeiçoadas a executar certas tarefas.

Dessa forma nossa ferramenta pode ser interpretada como uma ferramenta de *ReSkilling*, já que ajudará o usuário aprenderá a realizar tarefas mais avançadas na prática.

1.2. Projeto de Python

Para o projeto de Python, decidimos criar uma ferramenta exemplificando parte de como nossas soluções poderiam funcionar.

Utilizando um problema real do meu trabalho, criamos uma solução para resolver este problema, a falta de coleta de dados. Criamos um menu simples para coletar, visualizar e alterar dados. Com funções para checar o *input* do usuário para garantir que não haja erros na coleta de dados e quando possível, a própria coleta de dados foi automatizada.

2. Utilização

2.1. Utilização

Como mencionado anteriormente, a nossa solução será utilizada para a coleta de dados, neste caso em específico, para a coleta de dados dos Gastos de uma empresa. O usuário responsável pela coleta de dados utilizara este programa, onde será apresentado com um menu, com opções para registrar, visualizar e alterar dados. O usuário utilizara principalmente a função de registrar dados, onde ao escolher está opção, será encaminhado a outro menu, um submenu com as opções para registrar dados.

Sua primeira opção será o registro automatizado, onde o programa tira os dados dos arquivos .XML que são gerados junto das notas fiscais, extraindo esses dados ele registra no arquivo .CSV utilizado para salvar os dados. Porém nem todas as notas fiscais são geradas com um arquivo .XML auxiliar, como notas de serviços e certas despesas, como eletricidade e água. São nestes casos que as opções 2 e 3 serão utilizadas, para que o usuário insira os dados necessários manualmente. A diferença entre as duas opções é apenas sua classificação, entre despesa e serviço, para registrar o dado já o classificando. Em ambas as opções o usuário é requisitado por informações da nota fiscal que quer registrar. Ao inserir todos os dados corretamente o programa salva as informações inseridas no arquivo ou retorna o usuário ao menu caso alguma informação tenha sido inserida incorretamente.

No menu principal, a segunda opção dada ao usuário é de visualizar os dados, onde ele será apresentado com um outro submenu, com opções para visualizar todos os dados, visualizar os dados filtrado por categoria ou salvar todos os dados em um arquivo Excel estilizado. Assim o usuário tem a opção de visualizar os dados dentro do programa (com paginação para deixar mais acessível a leitura de dados) ou visualizar os dados em um arquivo Excel (uma opção mais familiar para aqueles que já trabalham com a ferramenta).

A terceira e última opção apresentada no menu principal é a de alterar dados, escolhendo está opção o usuário é apresentado novamente com outro submenu, onde o usuário terá diversas opções para procurar pela nota fiscal que gostaria de alterar, podendo procurar a partir do número da nota fiscal, pelo nome do fornecedor, data de emissão ou vencimento. Em qualquer uma dessas opções, caso exista mais de uma nota fiscal que corresponda a pesquisa, o usuário é apresentado com uma lista contendo essas notas fiscais, onde então ele seleciona a nota fiscal desejada a partir de seus índices marcados na lista.

Então o usuário é requisitado a inserir todos os dados a serem alterados, podendo pular aqueles que deseja manter.

Dentro do submenu de alterar dados, existe também a opção de deletar dados, com o mesmo modo de pesquisa e confirmação que as opções de alterar, porém com uma confirmação a mais após a nota fiscal for selecionada, para evitar que o usuário apague por acidente.

O submenu de alterar dados, embora não se espere que seja utilizado frequentemente, é essencial para alterar dados em caso de alteração ou cancelamento de notas fiscais.

Voltando ao menu principal, embora a terceira opção seja a última opção que seria inclusa no programa (excluindo a última opção de sair do programa), uma quarta opção foi inserida por questões acadêmicas, a opção de *DEBUG*, uma opção para ligar e desligar um debug incluso no código. Algo que foi incluso para facilitar o teste do código. Por questão acadêmica o código foi salvo com o *DEBUG* ativo, então existe uma mensagem a mais ao inserir datas no programa, exibindo ao usuário qual dia da semana a data inserida corresponde e ainda mais importante, em momentos que o programa fosse fazer alterações ao arquivo .CSV para salvar os dados, o programa cria e faz alterações em um arquivo clone marcado como debug, para que assim ao fechar o programa, nenhum arquivo 'oficial' utilizado para armazenar os dados tenha sido realmente alterado. Mas um arquivo clone é criado para que o usuário possa ver quais alterações teriam sido feitas.

Então por padrão, ao fechar e abrir o programa, se o *DEBUG* não for desativado, as alterações feitas aos dados não serão salvas e ao iniciar o programa novamente os dados exibidos serão os dados antigos antes de ser atualizado.

3. Funções Invisíveis

3.1. Funções de funcionamento do Menu invisíveis ao usuário

3.1.1. MENU PRINCIPAL

O programa inicia executando a função *main()*, onde o programa carrega os dados e abre o **menu principal**, que controla toda a navegação.

main()

É o ponto de entrada do programa.

Executa duas ações fundamentais:

1. Carrega os dados do CSV para o DataFrame global (*carregar_dados()*).
2. Inicia o menu principal (*main_menu()*).

main_menu()

Limpa a tela, exibe o título artístico da empresa e mostra as opções principais do sistema. Utilizando as subfunções *exibir_titulo(titulo_empresa)* e *exibir_opcoes()*. Depois chama *escolher_opcao()* para aguardar a escolha do usuário.

exibir_opcoes()

Mostra as opções do menu principal:

- Registrar custos
- Visualizar despesas
- Alterar dados
- Alternar debug
- Sair

escolher_opcao()

Interpreta o número digitado e redireciona para o submenu correto:

- 1 → *menu_registrar_dados()*
- 2 → *menu_visualizar_dados()*
- 3 → *menu_alterar_dados()*
- 4 → *alternar_debug()*
- 9 → *finalizar_app()*

Em caso de erro ou input inválido: retorna ao menu principal.

3.1.2. SUBMENU: REGISTRAR

Este submenu permite registrar gastos manualmente ou automaticamente via XML.

menu_registrar_dados()

Exibe o título artístico da seção e lista as opções de registro.

Utilizando as subfunções *exibir_titulo(registrar_titulo)* e *exibir_opcoes_registrar()*.

Depois chama *escolher_opcao_registrar()*.

exibir_opcoes_registrar()

Lista as opções:

1. Registro automático por XML
2. Registro manual (Despesas)
3. Registro manual (Serviços)
4. Voltar ao menu principal

escolher_opcao_registrar()

Interpreta a opção digitada:

- 1 → *registro_automatico()*
- 2 → *registro_manual('D')*
- 3 → *registro_manual('S')*
- 9 → retorna ao menu principal

registro_automatico()

Percorre todos os arquivos XML na pasta indicada, extrai dados fiscais e registra automaticamente as notas usando:

- datas
- fornecedor
- valores
- tributos

Depois chama *registrar_gasto()* para cada nota encontrada.

registro_manual(tipo)

Fluxo completo para registrar um gasto manualmente:

1. Fornecedor
2. Data de emissão
3. Data de vencimento
4. Número da NF

5. Valores (total, ICMS, COFINS, PIS, IPI)

Valida todas as entradas e, no final, registra com *registrar_gasto()*.

registrar_gasto(...)

Recebe um dicionário com todos os campos do gasto e adiciona ao DataFrame global. Depois salva as alterações no CSV (ou Debug CSV).

Funções auxiliares usadas no registro

- ***request_date()*** → válidas diferentes formatos de data.
- ***input_valor_monetario()*** → válida valores monetarios.

3.1.3 SUBMENU: VISUALIZAR

Aqui o usuário pode ver todos os gastos, filtrar ou gerar Excel.

menu_visualizar_dados()

Exibe o título da seção e as opções, chamando utilizando as subfunções *exibir_titulo(visualizar_titulo)* e *exibir_opcoes_visualizar()*.

Depois executa *escolher_opcao_visualizar()*.

exibir_opcoes_visualizar()

Opções:

1. Visualizar todos os dados
2. Visualizar por categoria
3. Salvar arquivo Excel
4. Voltar ao menu principal

escolher_opcao_visualizar()

Redireciona:

- 1 → *visualizar_dados()*
- 2 → *visualizar_por_categoria()*
- 3 → *salvar_arquivo_excel()*
- 9 → menu principal

visualizar_dados()

Exibe todo o DataFrame usando paginação.

Após visualizar, retorna ao menu de visualização.

visualizar_por_categoria()

Solicita um tipo (C, D, S), filtra o DataFrame e usa paginação para exibição.

salvar_arquivo_excel()

Gera o arquivo Excel final para o usuário, utilizando o template anual.

Inclui:

- formatação
- validação
- escrita nas abas mensais

Usa internamente ***salvar_dados_em_excel()***.

Funções auxiliares

- ***filtrar_dados()*** → filtra DataFrame por tipo
- ***paginacao()*** → exibe dados em páginas de 10 linhas
- ***exibir_pagina()*, *converter_input_numero_pagina()*** → suporte da paginação

3.1.4 SUBMENU: ALTERAR

Menu dedicado a modificar ou excluir registros já existentes.

menu_alterar_dados()

Exibe o título e as opções, utilizando as subfunções ***exibir_titulo(alterar_titulo)*** e ***exibir_opcoes_alterar()***. Executando em seguida ***escolher_opcao_alterar()***.

exibir_opcoes_alterar()

Opções:

1. Alterar por número da NF
2. Alterar por fornecedor
3. Alterar por data de emissão
4. Alterar por data de vencimento
5. Apagar por número da NF
6. Voltar

escolher_opcao_alterar()

Redireciona:

- 1 → ***alterar_por_nNF()***

- 2 → alterar_por_fornecedor()
- 3 → alterar_por_data_emissao()
- 4 → alterar_por_data_vencimento()
- 5 → apagar_por_nNF()
- 9 → menu principal

alterar_por_nNF()

Permite alterar um registro a partir da NF.

Se houver duplicidade, o usuário escolhe o índice.

Chama alterar_dados().

alterar_por_fornecedor()

Busca todos os registros de um fornecedor e permite escolher qual alterar.

alterar_por_data_emissao() / alterar_por_data_vencimento()

Filtra por data, mostra todos os registros e permite selecionar via NF.

apagar_por_nNF()

Exclui um registro definitivamente.

Requer confirmação do usuário.

Depois salva o CSV novamente.

alterar_dados(indice)

A função central das alterações:

Para cada campo, oferece a opção de trocar o valor ou manter o antigo.

Ao final, salva todas as alterações.

3.1.5 FUNÇÃO DE ALTERAR O DEBUG

alternar_debug()

Alterna entre:

- **modo debug ativo** → salva em arquivo separado
- **modo normal** → salva no CSV principal

Mostra o novo status e retorna ao menu principal.

3.1.6 SAIR DO PROGRAMA

finalizar_app()

Exibe o subtítulo “Finalizando app” e encerra a execução.

(Como o programa é um loop de menus, o encerramento acontece naturalmente após essa função.)

4. Resumo

Este é um pequeno exemplo, uma pequena parte de uma grande solução que gostaríamos de criar. Focamos no controle de input de dados, pois esse controle é essencial para o funcionamento do resto da nossa ferramenta, já que com a entrada de dados incompatíveis na ferramenta, geraria erros, dificultando o funcionamento autônomo da ferramenta para possibilitar o auxílio do usuário em suas tarefas como esperado.

Por isso é essencial que o programa verifique tudo que o usuário insere no sistema, para garantir o próprio funcionamento e dessa maneira, garantindo o auxílio ao usuário.