**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS**

**ESCOLA POLITÉCNICA**

**ENGENHARIA DE SOFTWARE**

Caio Cezar Gandara dos Santos 23015616

David Hengstmann 23007064

Filipe Daniel Medeiros T. Mota 23002322

**RELATÓRIO DE PROJETO:**

**SISTEMA DE CONTROLE DE VOOS**

**CAMPINAS - 2023**

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS**

**ESCOLA POLITÉCNICA**

**ENGENHARIA DE SOFTWARE**

Caio Cezar Gandara dos Santos 23015616

David Hengstmann 23007064

Filipe Daniel Medeiros T. Mota 23002322

**RELATÓRIO DE PROJETO:**

**SISTEMA DE CONTROLE DE VOOS**

Relatório de Projeto de Sistema, apresentado no componente curricular Algoritmos de Programação, Projetos e Computação, do curso de Engenharia de Software, da Escola Politécnica da Pontifícia Universidade Católica de Campinas

Orientador: Profa. Lucia Filomena de Almeida Guimarães

**CAMPINAS - 2023**

# introduçao

O projeto do Sistema de Controle de Voos desenvolvido pelo grupo tem como o objetivo principal o controle dos voos dos clientes que adquiriram a passagem aérea com a empresa.

Os requisitos, que serão discutidos mais profundamente nos próximos tópicos, foram algumas funcionalidades para facilitar o dia a dia dos funcionários da empresa cuja função é o controle de voos. São eles:

- A inclusão de novos voos, contendo suas principais informações;

- A alteração das informações de determinado voo, caso ocorra algum erro;

- A exclusão das informações de determinado voo, caso o cliente faça o cancelamento;

- A determinação da quantidade de voos que saem de uma cidade origem escolhida no sistema;

- A busca do voo com menor número de escalas a partir da escolha de cidade de origem e cidade destino;

- A busca de todos os voos que chegam na cidade destino escolhida no sistema.

A partir dessas informações, discutiremos adiante as soluções encontradas pela equipe para cada um destes requisitos.

# Dificuldades durante o desenvolvimento

Na primeira etapa do projeto, onde o desenvolvimento principal do sistema foi feito apenas utilizando as estruturas de dado dicionário (para registrar os voos) e listas (para armazenar as cidades de escalas), a equipe teve grande facilidade e conseguiu realizar o projeto nas duas primeiras semanas, durante as aulas dadas em laboratório.

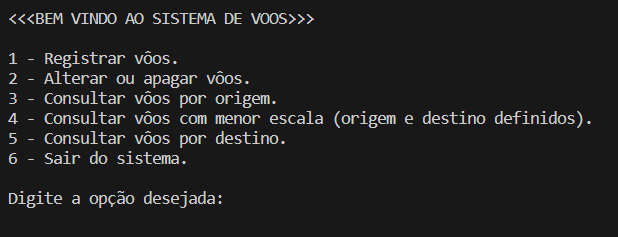
Após a mudança para a segunda parte do projeto, onde o projeto teria que ser remodelado com a utilização de funções em cada requisito, as dificuldades começaram a aparecer. Os integrantes ficaram confusos com o novo aprendizado e não sabiam muito bem aplicar o novo conhecimento. A parte de “chamar” a função foi bem compreendida, porém, o principal problema se encontrava em como passar o valor desejado para dentro da função.

Com todos esses problemas, utilizamos o tempo extra classe para estudar, aplicar, testar e corrigir erros para aprender a utilizar melhor as funções, e assim, conseguimos desenvolver o restante do projeto.

A principal fonte de estudos foi um artigo no site DevMedia, um fórum relacionado ao estudo de programação.

# Apresentação do projeto

Como solicitado, a equipe construiu um menu com todas as opções de escolha disponíveis ao usuário:



O código responsável pelo menu:



Explicação das escolhas do código:

A codificação do menu se encontra na parte principal do programa, ou seja: fora das funções (que serão abordadas nos próximos códigos).

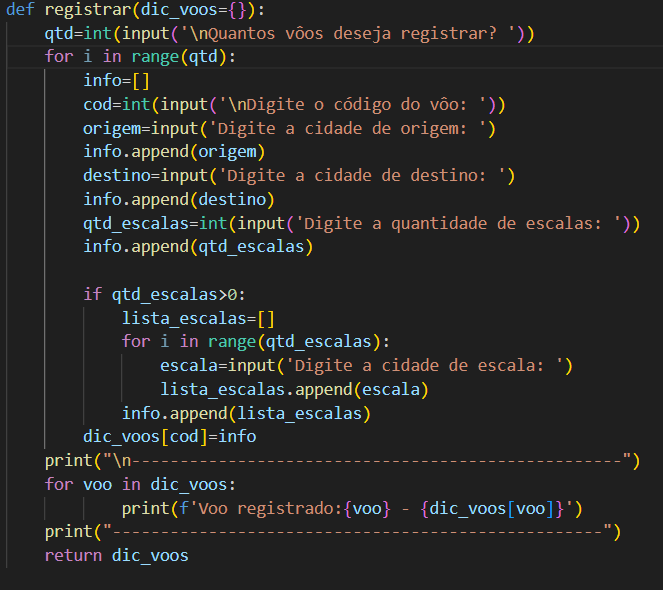
Tudo se encontra dentro de um “while True”, para que o menu seja exibido novamente sempre que usuário finalizar uma ação.

A exibição do menu se dá pelos prints com cada uma das opções, de 1 a 6, seguidas de um input que pede um valor para a variável “opcao”. Foi realizado o tratamento de exceções para que o único valor aceito nesse input seja o de números inteiros, e também utilizado um if para caso os números inteiros sejam negativos ou superiores a 6, que é a última opção do menu.

Seguindo adianta, podemos ver uma sequencia com um if, 4 elifs e um else. O if e os elifs, quando acionados por seus respectivos valores atribuídos à variável “opcao”, fazem a chamada da função referente ao processo que o usuário deseja realizar. O else, por sua vez, só será acionado caso “opcoes” seja o valor 6, já que os números de 1 a 5 possuem um if próprio e os valores negativos e número maiores que 6 não poderão ser atribuídos a variável, pois, como visto antes, o “if opcao <1 or opcao >6” não deixarão passar estes valores.

Por fim, o else é composto de dois prints: um com uma contagem regressiva para o final do programa, que imprime uma mensagem a cada segundo (foi utilizado o for e a biblioteca “time”), e também o print de finalização do sistema.

Código para registro de voos:



Foi criada uma função que possui como entrada, um dicionário vazio.

A primeira interação do usuário nessa função é digitar quantos voos ele deseja registrar, número inteiro este que será atribuído à variável qtd.

Após a obtenção da variável, a equipe optou pelo for, tendo como range a quantidade de voos, para que o programa repita o registro quantas vezes for solicitado.

Dentro deste for, uma lista é aberta. É importante ressaltar que a lista é aberta dentro do for pois é necessário ter uma lista para armazenar informações de cada voo. Juntamente com a lista, são solicitadas as informações: código, origem, destino e quantidade de escalas. Exceto o código, os inputs de todas as informações são seguidas do .append, para que a informação já seja armazenada na lista.

Após este processo, optamos por utilizar um if para a condição de o voo ter alguma escala: if qtd\_escalas>0. Então, caso tenha alguma escala, é aberta uma outra lista para que os nomes das cidades de escala sejam adicionados. O for que aparece adiante tem como range a quantidade de cidades de escala, para que sejam perguntadas ao usuário (input) e adicionadas na lista (append) quantas vezes forem necessárias.

Quando encerrado este for, a lista contendo o nome de todas as cidades é adicionada à lista principal, tornando-se um elemento da mesma.

Então, com a lista completa, temos a seguinte estrutura:

info[0] -> cidade de origem

info[1] -> cidade de destino

info[2] -> quantidade de escalas

info[3] -> lista com as cidades de escala

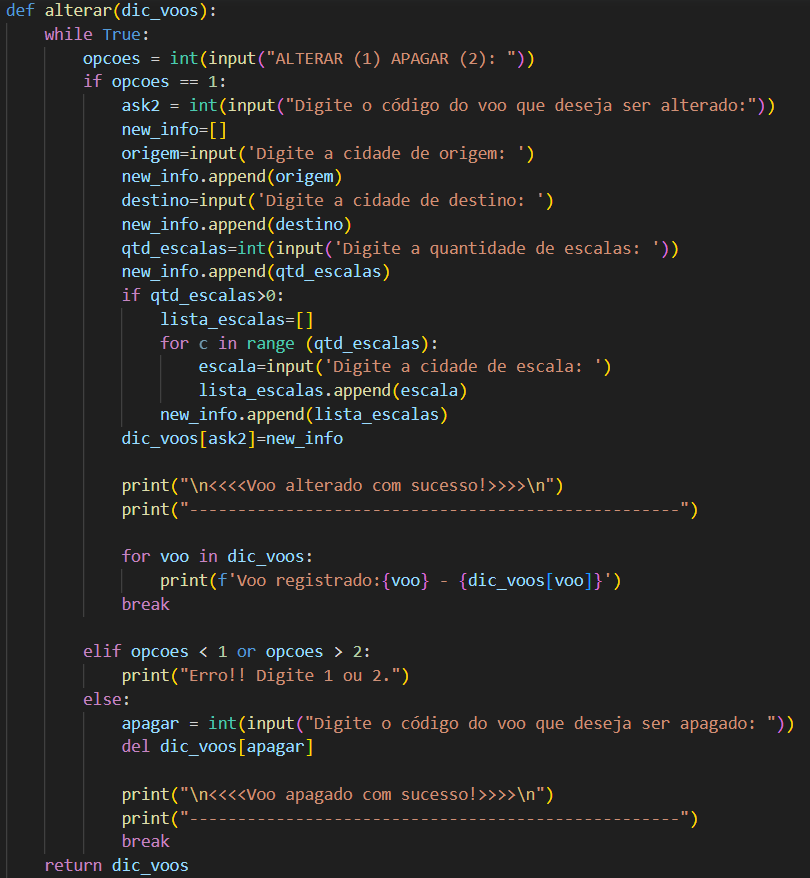
OBS: info[3] só existe se existir alguma cidade de escala no voo

Após todos estes processos, serão adicionados ao dicionário de entrada 1-o valor da variável “cod”, que será o código do voo (key) e a lista de informações (values) através do código dic\_voos[cod]=info.

Foi utilizado também um for para percorrer o dicionário e imprimir suas informações, para que o usuário tenha a confirmação de que os voos foram registrados.

Por fim, essa função nos retorna o dicionário, contendo todas as informações armazenadas durante a execução da função de registro.

Código para alterar/apagar informações de voo:



O dicionário já não entra vazio na função, ou seja: algum registro deve ter sido feito para que esta função possa ser executada.

Dentro da função, temos um while True, que será responsável por repetir o input da variável opcoes toda vez que o usuário digitar algo diferente de 1 (alterar) e 2 (apagar). A parte do código responsável por esta validação é: elif opcoes <1 or opcoes > 2, print (Erro).

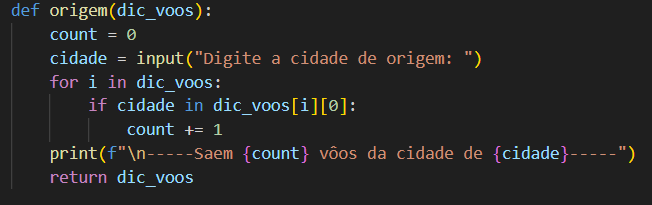
Caso o valor de opcoes seja 1, o processo de alteração no voo será iniciado. Ele é idêntico ao processo de registro, porém substituirá um registro já existente. Após a finalização da alteração, temos um dos dois possíveis breaks para o while True. Isso significa o encerramento da função e a volta ao menu principal.

O else se refere unicamente ao valor 2, atribuído à opção de apagar um voo. Já que o valor 1 está associado à mudança no registro e valores menores que 1 ou maiores que 2 não passarão pela validação, o else será acionado unicamente pelo valor 2. O input de um valor para a variável “apagar” será solicitado, que deve ser o valor do código de voo. Ele será utilizado no código del dic\_voos[apagar], que tem como objetivo apagar o respectivo voo.

Após a exclusão do voo, temos o segundo dos dois possíveis breaks para o while True da função. Então, assim como anteriormente, significa o encerramento da função e a volta para o menu principal.

Independente da ação de apagar o alterar o voo, a função nos retorna o dicionário com as alterações realizadas.

Código para consulta por origem:



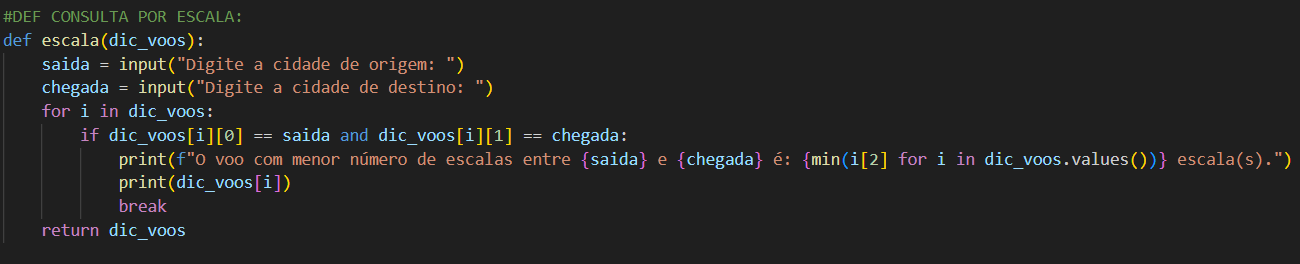
Como na opção de apagar/alterar voo, na opção de consulta por origem o dicionário já deve conter alguma informação, já que não entra mais vazio na função.

A variável “count” é adicionada com valor 0. Ela será utilizada para contar os voos da cidade de origem, que deve ser digitada no input que a sucede.

Após a declaração da variável e a obtenção do nome da cidade, é utilizado um for. Como vimos anteriormente, a cidade de origem se encontra na posição 0 da lista de valores referentes ao código (key) do voo. Então, este for serve para percorrer o dicionário, encontrar a lista de valores, e localizar o valor 0. Caso a variável “cidade” (cidade escolhida pelo usuário para consulta) seja a cidade presente no valor 0 da lista, é adicionada à variável count o valor +1. Após a finalização do for, o print foi formatado para exibir a quantidade de voos (count) que saem da cidade digitada (cidade).

Ao final, a função retorna o mesmo dicionário que entrou no início.

Código para consulta por menor números de escala:



Novamente, o dicionário entra na função já contendo registros.

São solicitadas ao usuário o nome das cidade de origem e destino, através dos inputs de “saida” e “chegada”.

Como visto anteriormente, na lista de valores do dicionário a cidade de origem ocupa a posição 0 e e a cidade de destino ocupa a posição 1.

O for foi utilizado para percorrer o dicionário e a lista de valores, e o if para então acessar os valores 0 e 1 da lista e os comparar respectivamente com as variáveis “saida” e “chegada”. Se forem iguais, significa que o sistema encontrou um voo que tem as cidades de origem e destino que o usuário digitou.

Como visto anteriormente, o número referente a quantidade de escalas do voo ocupada a posição 2 da lista de valores do dicionário.

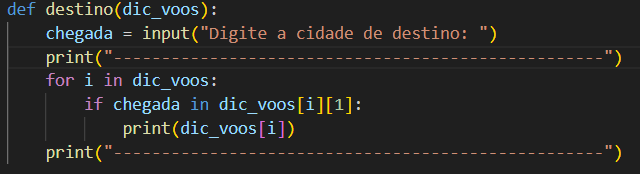
Com isso em mente, a equipe decidiu formatar o print utilizando “min” para essa posição da lista de valores, para assim, identificar o voo que contenha o menor valor na posição 2 da lista de valores do dicionário.

Após esta ação, o print da sequência imprime todas as informações do voo descrito acima.

O break foi utilizado para que após a impressão dos valores deste voo, o for não siga imprimindo o valor de outros voos.

A função retorna o dicionário da mesma maneira que entrou, já que ele não sofreu modificações.

Código para consulta por destino:



O dicionário entra na função já contendo informações.

O input solicita ao usuário o nome da cidade destino, que será atribuído à variável “chegada”.

O for é utilizado para percorrer o dicionário, e o if condiciona o print às informações dos dicionários que possuírem o valor da posição 1 da lista de valores (que é a cidade destino) equivalente à variável chegada.

# referência

DevMedia. Padrões de Qualidade do Ar. devmedia.com.br, 10/06/2023 Disponível em: https://www.devmedia.com.br/funcoes-em-python/37340