Ifes Campus Serra
BSI – Bacharelado de Sistemas de Informação
Programação II

Trabalho de Programação Avaliação 2

Tema: Tipos Abstratos de Dados com Dicionarios, Listas e Arquivos

Importante: construa o seu código exatamente de acordo com os enunciados.

Regras para o trabalho: * utilize apenas os comandos python vistos em sala de aula; * para manipulação de arquivos utilizar apenas open, close, readline, write; * não utilizar expressões de lista; * não utlizar dados com tipos; * na construção do trabalho, utilizar apenas os arquivos de dados fornecidos pelo Professor, SEM modificações; * para construir os códigos, utilize os nomes de arquivos .py e nomes de funções fornecidos nos enunciados; * todas as figuras exibem apenas exemplos para auxiliar na interpretação do enunciado.

I. Introdução

Crie o protótipo de um Sistema de Reservas de Passagens – SRP - especificado nas seções a seguir.

O SRP é formado por 3 componentes:

- O TAD Banco de Dados – tadbd: contém 3 tabelas com os dados a serem processados pelo sistema.
- O Servidor SRP: um processo servidor xmlrpc-Python que fornece para clientes da rede uma API baseada na exposição da iterface do tadbd.
- O Cliente SRP: um processo cliente xmlrpc-Python que disponibiliza para o usuário final uma série de funcionalidades do SRP através de chamadas ao Servidor.

II. TAD Banco de Dados

O TAD Banco de Dados deve ser implementado no arquivo *tadbd_srp.py* e disponibilizar a seguinte interface:

a) *def open_bd(tabnamePass, tabnameRes, tabnameVoos)*: função que recebe os nomes dos arquivos csv (ver figura 1) que contém/conterão os dados de passageiros, reservas e vôos, respectivamente.

   A partir dos nomes, *open_db* lê cada um dos arquivos transformando-os em um dicionário de dados JSON. Após processar os 3 arquivos, retorna o tad bd estruturado como o dicionário mostrado na figura 2.

b) *def salva_bd(bd, tabnamePass, tabnameRes, tabnameVoos)*: função que recebe um tad dicionario bd (figura 2) e os nomes dos arquivos que receberão os dados de passageiros, reservas e voos no formato csv. Ver figura 1.

c) *def adPassageiro(bd, idpass, nome, email, fone)*: cadastra, usando o formato JSON, os dados do passageiro no respectivo dicionario em bd. Retorna bd (o banco de dados atualizado com o novo passageiro). Ver figura 2.

d) *def adReserva(bd, idres, data, status, assento, idpass, idvoo)*: cadastra, usando o formato JSON, os dados da reserva no respectivo dicionario em bd e retorna o código zero. Ver figura 2.

A reserva só é cadastrada se: o passageiro existir, o voo existir e tiver vaga, o assento estiver desocupado. Caso o passageiro não exista, retornar o código 1. Caso o vôo não exista, retornar o código 2. Caso o vôo exista mas não tem vaga, retornar o código 3. Caso o vôo exista e tem vaga, se o assento solicitado já está ocupado, retornar o código 4.

e) *def adVoo(bd, idvoo, numvoo, origem, destino, dtpartida, dtchegada)*: cadastra, usando o formato JSON, os dados do vôo no respectivo dicionario em bd. Ver figura 2. Retorna bd (o banco de dados atualizado com o novo vôo)

f) *def vooExiste(bd, idvoo)*: retorna True se o vôo existe em bd. Retorna False caso contrário.

g) *def passExiste(bd, idpass)*: retorna True se o passageiro existe em bd. Retorna False caso contrário.

h) *def vagasVoo(bd, idvoo)*: retorna a quantidade de vagas disponiveis no voo. Caso o vôo não exista, retorna o código -1.

i) *def assentoLivre(bd, idvoo, assento)*: retorna False se o vôo não existir ou se o vôo existir e o assento já esteja ocupado. Retorna False caso contrário.

III. Servidor SRP

O servidor SRP deve ser construído seguindo-se as referências encontradas nas aulas de PROG2 e nos documentos “Laboratório TAD/API em Python” e “Código Clente/Servidor Protótipo JDV”, ambos encontrados na sala AVA da disciplina.

O servidor SRP deve:

- Importar a biblioteca *xmlrp* apropriada;
- Importar o tad banco de dados, *tadbd_srp.py*;
- Definir as funções do servidor: tanto as que serão visiveis aos clientes (funções registradas) quanto as privadas do servidor, se houverem na implementação adotada.
- As funções que precisarem fazer uso do tad banco de dados deverão trabalhar com a variável global que representa o banco e que foi definida na função *main*.
- Possuir uma função *main*, onde o servidor é atividado e o banco de dados do sistema é criado e disponibilzado como uma variável global ao módulo servidor.

API do Servidor SRP (funções registradas para uso do cliente):

a) *def adPassageiro(idpass, nome, email, fone)*: invoca a correspondente função do *tadbd_srp.py* para cadastrar um passageiro utilizando os dados enviados pelo cliente.

b) *def adReserva(idres, data, status, assento, idpass, idvoo)*: invoca a correspondente função do *tadbd_srp.py* para cadastrar uma reserva utilizando os dados enviados pelo cliente.

c) *def adVoo(idvoo, numvoo, origem, destino, dtpartida, dtchegada)*: invoca a correspondente função do *tadbd_srp.py* para cadastrar um vôo utilizando os dados enviados pelo cliente.

d) *def getPassageiro(id)*: invoca a correspondente função do *tadbd_srp.py* para resgatar e retornar o passageiro do banco de dados que possui o id fornecido pelo cliente.

e) *def getVoo(id)*: invoca a correspondente função do *tadbd_srp.py* para resgatar e retornar o vôo do banco de dados que possui o id fornecido pelo cliente.

f) *def getReserva(id)*: invoca a correspondente função do *tadbd_srp.py* para resgatar e retornar a reserva do banco de dados que possui o id fornecido pelo cliente.

g) *def vooExiste(idvoo)*: invoca a correspondente função do *tadbd_srp.py* para verificar e retornar a existência, ou não, do vôo (no banco de dados) cujo id foi enviado pelo cliente.

h) *def passExiste(idpass)*: invoca a correspondente função do *tadbd_srp.py* para verificar e retornar a existência, ou não ,do passageiro (no banco de dados) cujo id foi enviado pelo cliente.

i) *def vagasVoo(idvoo)*: invoca a correspondente função do *tadbd_srp.py* para verificar e retornar a quantidades de lugares disponiveis no vôo (no banco de dados) cujo id foi enviado pelo cliente.

j) *def assentoLivre(idvoo, assento)*: invoca a correspondente função do *tadbd_srp.py* para verificar e retornar a ocupação, ou não, do assento no vôo identificados pelos ids enviados pelo cliente.

Função *main()*:

A função *main()* deve:

- Criar o objeto servidor (ver material de referência);
- Enviar mensagem para a tela indicando que o servidor será atividado;
- Criar o tad banco de dados a partir dos arquivos de dados fornecidos pelo Professor: “tabpassageiros.txt", "tabreservas.txt", "tabvoos.txt”.
- Disponibilizar o tad banco de dados na forma de uma variável global ao módulo servidor;
- Registrar as funções que serão visiveis ao módulo cliente;
- Ativar o processo servidor;
- Antes de encerrar, salvar o banco de dados.

IV. Cliente SRP

O cliente SRP deve ser construído seguindo-se as referências encontradas nas aulas de PROG2 e nos documentos “Laboratório TAD/API em Python” e “Código Clente/Servidor Protótipo JDV”, ambos encontrados na sala AVA da disciplina.

O cliente SRP deve:

- Importar a biblioteca *xmlrp* apropriada;
- Exibir um título da aplicação:
- Exemplo: “SISTEMA RESERVA DE VÔOS VERSÃO 1.0”
- Criar o objeto que representa o servidor do sistema SRV no cliente;
- Exibir o menu mostrado na figura 3;
- Gerenciar as escolhas do menu feitas pelo usuário e as respectivas respostas entregues pelo servidor;
- No caso da opção 1 do menu, “Fazer Reserva”, desenvolver o seguinte fluxo de ações:
    - Ler o id do vôo que se pretende embarcar; Se o vôo não existir (no banco de dados), enviar mensagem para o usuário, encerrar a opção 1 e retornar para o menu principal; Se o vôo existir, verificar se o vôo possui assentos disponíveis; Se o vôo estiver lotado, enviar mensagem para o usuário, encerrar a opção 1 e retornar para o menu principal; Caso haja assento disponível, solicitar ao usuário o número do assento; Verificar se o assento desejado está desocupado; Se o assento estiver ocupado, enviar mensagem para o usuário, encerrar a opção 1 e retornar para o menu principal; Caso o assento desejado esteja disponível, solicitar o id do passageiro. Caso o passageiro não exista (no banco de dados), enviar mensagem para o usuário, encerrar a opção 1 e retornar

para o menu principal; Caso o passageiro exista, então a reserva é confirmada e os dados da reserva, vôo e passageiro devem ser exibidos na tela, de forma organizada.

- No caso das demais opções do menu principal, os procedimentos devem ser facilmente inferidos.

V. Figuras Ilustrativas

tabpassageiros.txt

```
1,João Silva,joao.silva@example.com,+55 11 91234-5678
2,Maria Souza,maria.souza@example.com,+55 21 98765-4321
3,Carlos Pereira,carlos.pereira@example.com,+55 31 99876-5432
4,Ana Oliveira,ana.oliveira@example.com,+55 41 98765-1234
5,Pedro Lima,pedro.lima@example.com,+55 51 91234-8765
```

Figura 1: Exemplo de arquivo csv disponibilizado para o trabalho. O conteúdo é apenas exemplo.

```
{
'passageiros': {'1': {'id': '1', 'nome': 'João Silva', 'email': 'joao.silva@example.com', 'tel': '+55 11 91234-5678'},
                '2': {'id': '2', 'nome': 'Maria Souza', 'email': 'maria.souza@example.com', 'tel': '+55 21 98765-4321'},
                '3': {'id': '3', 'nome': 'Carlos Pereira', 'email': 'carlos.pereira@example.com', 'tel': '+55 31 99876-5432'},
                '4': {'id': '4', 'nome': 'Ana Oliveira', 'email': 'ana.oliveira@example.com', 'tel': '+55 41 98765-1234'},
                '5': {'id': '5', 'nome': 'Pedro Lima', 'email': 'pedro.lima@example.com', 'tel': '+55 51 91234-8765'}},

'reservas': {'10': {'id': '1', 'data': '2025-11-10', 'status': 'Confirmada', 'assento': '12A', 'idpass': '1', 'idvoo': '101'},
             '20': {'id': '2', 'data': '2025-11-11', 'status': 'Pendente', 'assento': '7C', 'idpass': '2', 'idvoo': '101'},
             '30': {'id': '3', 'data': '2025-11-12', 'status': 'Cancelada', 'assento': '14B', 'idpass': '3', 'idvoo': '101'},
             '40': {'id': '4', 'data': '2025-11-13', 'status': 'Confirmada', 'assento': '21D', 'idpass': '4', 'idvoo': '101'},
             '50': {'id': '5', 'data': '2025-11-14', 'status': 'Confirmada', 'assento': '3F', 'idpass': '5', 'idvoo': '101'}},

'voos': {'101': {'id': '101', 'numVoo': 'AZ1234', 'origem': 'São Paulo', 'destino': 'Rio de Janeiro',
                 'dataPda': '2025-11-15 08:30', 'dataChe': '2025-11-15 09:45', 'vagas': 5},
         '102': {'id': '102', 'numVoo': 'LA5678', 'origem': 'Belo Horizonte', 'destino': 'Brasília',
                 'dataPda': '2025-11-16 14:20', 'dataChe': '2025-11-16 16:10', 'vagas': 5},
         '103': {'id': '103', 'numVoo': 'TP9012', 'origem': 'Curitiba', 'destino': 'Porto Alegre',
                 'dataPda': '2025-11-17 18:00', 'dataChe': '2025-11-17 19:30', 'vagas': 5}}
}

{'id': '3', 'nome': 'Carlos Pereira', 'email': 'carlos.pereira@example.com', 'tel': '+55 31 99876-5432'}
```

Figura 2: Exemplo da implementação do tad bd reservas: dicionário de dicionário de dados json.

```
== Cliente XML-RPC ===
1 - Fazer Reserva
2 - Cadastrar Passageiro
3 - Cadastrar Voo
4 - Sair
```

Figura 3: Exemplo do menu principal do cliente SRV.

Bons estudos!!