Um Sistema Gerenciador de Tarefas

Igor Figueredo Soares[1]

[1] UEFS – Universidade Estadual de Feira de Santana
Av. Transnordestina, s/n, Novo Horizonte
Feira de Santana – BA, Brasil – 44036-900

ifs54@hotmail.com

Resumo. *O Departamento de Ciências Exatas (DEXA) da Universidade Estadual de Feira de Santana (UEFS) solicitou aos estudantes do módulo integrador Algoritmos, o desenvolvimento de um programa gerenciador de tarefas. O sistema gerenciador de tarefa apresenta o objetivo de auxiliar os estudantes da universidade na organização de suas tarefas. Para a implementação deste programa foi utilizada a linguagem e programação Python em sua versão 3.7.4.*

1. Introdução

Gerenciadores de tarefas são softwares utilizados para organização de afazeres e otimização de tempo. Esse tipo de aplicação consiste, de modo geral, em organizar tarefas inseridas pelo usuário, permitindo que sejam definidas algumas características, tais como, prioridade, descrição e alguns desses sistemas ainda permitem a definição de um intervalo de tempo em que as tarefas devem ser feitas.

Reconhecendo a deficiência de alguns estudantes do curso de Engenharia de Computação da Universidade Estadual de Feira de Santana (UEFS) em organizar suas ocupações e deveres, a Coordenação do Colegiado do Curso de Engenharia de Computação recomendou o uso de um sistema gerenciador de tarefas para os estudantes a fim de amenizar tal problema. Foi definida que a ferramenta de gerenciamento deveria ser de livre acesso e simples, deste modo, a proposta em questão foi encaminhada ao Departamento de Ciências Exatas (DEXA), o qual delegou a responsabilidade do desenvolvimento deste programa aos estudantes do módulo integrador de Algoritmos.

O software gerenciador de tarefas proposto deve seguir alguns requisitos e especificações, tais como, apresentar um menu principal que disponha das opções: Cadastrar novo usuário, onde é possível efetuar o cadastro de novos utilizadores do sistema; Logar no sistema, opção que permite usuários entrarem em sua conta; e Sair do sistema: que promove o encerramento do programa. Além disso, ao usuário logar no sistema, um menu secundário deve ser apresentado, este menu deve oferecer as seguintes opções para o usuário: cadastrar nova tarefa, visualizar tarefas, alterar tarefa, excluir tarefa e sair da conta do usuário.

Deste modo, exposta a sintetização supramencionada, o presente relatório tem como objetivo discorrer sobre o processo de implementação deste programa gerenciador de tarefas, o qual foi desenvolvido na linguagem de alto nível Python. Contudo, para a solução do problema (desenvolvimento do software) alguns questionamentos foram surgindo e precisaram ser respondidos, sendo eles: 1) O que são classes e como implementa-las no problema; 2) Como armazenar as informações de login dos usuários; 3) Como verificar as informações para o login dos usuários; 4) Como armazenar as tarefas dos usuários; 5) Como efetuar a visualização de tarefas respeitando o critério determinado; 6) Como editar uma tarefa; 7) Como efetuar a exclusão de uma tarefa?

2. Desenvolvimento

Nesta seção de desenvolvimento procura-se elucidar como o código se encontra organizado e também responder os questionamentos levantados na introdução deste relatório.

O que são classes e como implementa-las no problema? Classes são estruturas fundamentais do paradigma de programação orientada a objetos. Uma classe representa um tipo do objeto, ou seja, uma espécie de molde para a criação de objetos. Já os objetos são abstrações para a representação de entidades e elementos, os quais dispõe de características (atributos) e ações que podem executar (métodos) [Borges 2010]. As classes permitem modularizar o código de uma forma mais eficiente e além de facilitar o desenvolvimento de aplicações complexas.

Na linguagem Python, classes podem ser criadas a partir da palavra-chave *class*, seguida do nome da classe, que por convenção, começa com letra maiúscula. Novos objetos em Python são construídos através da classe por meio de atribuições [Borges 2010]. O método *__init__* é definido logo após a criação da classe. O *__init__* é um método especial, chamado quando um objeto é instanciado [Downey 2016]. Além disso, em todo método de um objeto deve ser passada de forma explicita uma variável que represente o objeto, por convenção é utilizada a variável *self* para esse procedimento, porém o nome desta variável pode ser alterado por alguma outro (porém, não ser considerada uma boa prática de programação) [Borges 2010].

No problema em questão, foram utilizadas duas classes, sendo elas: Usuario e Tarefa. A classe Usuario é responsável por instanciar os novos usuários cadastrados, recebendo como atributos o nome de usuário e a senha. Além disso, a classe Usuario apresenta os métodos de cadastrar usuário, onde executa o processo de armazenamento dos dados do usuário e o método fazer login, que possibilita o usuário entrar em sua conta no sistema, utilizando seu nome de usuário e senha. Já a classe Tarefa instancia novas tarefas, recebendo como atributos da tarefa: sua prioridade, título, descrição e ID. Dessa forma, cada tarefa criada é um objeto que apresenta suas características. Além do mais, a classe Tarefa dispõe do método alterar tarefa, o qual é responsável por efetuar qualquer modificação dos atributos: prioridade, título e descrição de determinada tarefa.

Como armazenar as informações de login dos usuários? É necessário que novos usuários se cadastrem no sistema de gerenciamento de tarefas. O utilizador deve inserir um nome de usuário e uma senha, posteriormente, o sistema deve verificar se o nome de usuário ainda não se encontra cadastrado no sistema, caso isso seja verdade, o cadastro é efetuado com sucesso, caso contrário um novo nome de usuário deve ser informado.

A linguagem Python permite a criação e manipulação de arquivos, sobretudo arquivos de texto, durante a execução do programa. Para isso é utilizado o comando *open(),* atribuído uma variável, o qual recebe como parâmetros o nome do arquivo e o modo de abertura do arquivo. Além disso, o Python dispõe também do módulo *hashlib* que apresenta diversos métodos para a criação de valores hash de determinado elemento. O hash, de modo geral, permite através de seus algoritmos de mapeamento converter uma grande quantia de dados em pequenas informações. Além disso, o hash permite a criptografia de informações, visto

que o valor gerado por meio de seus algoritmos é praticamente não invertível, ou seja, não é possível chegar aos dados originais que originaram tal hash.

No programa em questão, para o armazenamento das informações dos usuários foi utilizado um arquivo de texto, o qual guarda o nome de usuário e a senha correspondente a esse usuário. Para efetuar tal procedimento, o arquivo de texto é aberto em modo de escrita, o utilizando o comando 'a', esse modo permite adicionar elementos ao final do arquivo, porém caso o arquivo não exista, ele é criado e o elemento é adicionado no arquivo.

Para atribuir segurança ao sistema, a senha dos usuários foi salva em forma de *hash* hexadecimal. Isso foi feito utilizando os métodos da biblioteca supracitada *hashlib*. Quando o usuário insere a senha que deseja e o nome de usuário é verificado como válido, sua senha é convertida em hash. Para isso, foi usado o algoritmo *sha256* presente na biblioteca *hashlib*. Desta maneira, quando as informações dos usuários são salvas no arquivo de texto, sua senha não é armazenada e sim o seu hash junto ao nome de usuário.

Como verificar as informações para o login dos usuários? Os utilizadores do software de gerenciamento, após cadastrados deve poder efetuar login em suas contas, permitindo que tenham acesso ao menu de tarefas. Desta maneira, os usuários podem cadastrar novas tarefas, visualiza-las e exclui-las.

Para que o usuário consiga ter acesso a sua conta, ele precisa inserir seus dados (nome de usuário e senha) de forma correta. A verificação desses dados é feita por meio de uma busca no arquivo de texto que contém os dados de todos os usuários cadastrado. Esse processo de verificação é iniciado com a obtenção do hash, através do mesmo processo utilizado na seção de cadastro, da senha inserida pelo usuário durante o login. Posteriormente, o arquivo de texto é aberto em modo de leitura, por meio do comando 'r' e todo conteúdo do arquivo é percorrido com um laço for, o qual percorre cada linha, verificando se o nome de usuário e o hash da senha inseridos coincidem com algum dos cadastrados. Por fim, caso os dados coincidam o login é efetuado sem problemas, porém caso não coincidam são solicitados novos dados para o login do usuário.

Como armazenar as tarefas dos usuários? As tarefas salvas pelo usuário devem ser capazes de serem acessadas sempre que o utilizador efetuar seu login no sistema. Dessa forma, é necessário armazena-las em um arquivo, permitindo que os dados sejam armazenados de forma persistente.

Arquivos binários são arquivos que apresentam seu conteúdo armazenado na forma de binário. Os arquivos binários permitem que elementos inteiros, como por exemplo, listas, vetores e objetos sejam armazenados neles. Desta forma, a leitura desses elementos a partir do arquivo é, relativamente, mais simples, pois o elemento é lido diretamente.

No programa foram utilizados arquivos binários para o armazenamento das tarefas dos usuários. Cada usuário dispõe um arquivo binário único. Além disso, é solicitado que cada tarefa tenha um ID único que as identifique.

Dito isso, para o armazenamento de uma tarefa no sistema, inicialmente deve ser gerado o seu ID. As tarefas são inseridas em uma lista que contém três dicionários, o dicionário de índice 0 (zero) armazena as tarefas de alta prioridade, o de índice 1 (um) de

média prioridade e o de índice 2 (dois) de baixa prioridade. O ID das tarefas são utilizados como chaves dos dicionários para a identificação das tarefas.

O processo de armazenamento inicia-se com a execução de uma função para gerar o ID da tarefa, após os dados da tarefa serem inseridos. A função que gera o ID tenta abrir o arquivo binário que tem como nome o próprio nome de usuário do utilizador que se encontra logado. Para isso é utilizado o comando *open(),* recebendo como paramentos o nome do usuário seguido da indicação da extensão do arquivo (*.dat*) e o modo de abertura do arquivo 'rb', em que o r indica que deve ser aberto em modo de leitura e o b indica que o arquivo é binário.

Desta maneira, caso o arquivo de tarefas do usuário não exista, uma exceção de *FileNotFoundError* é disparada, indicando que o usuário nunca cadastrou alguma tarefa, então o arquivo binário do usuário é criado e o ID da tarefa é automaticamente 1 (o primeiro). Porém, caso o arquivo de tarefas do usuário já exista, tenta-se ler as chaves (ID) dos dicionários de cada tarefa e armazenar a maior de cada dicionário em uma lista. Posteriormente, o maior ID dessa lista das maiores é obtida e somada a ela +1 (mais um), gerando dessa maneira o ID da nova tarefa cadastrada. Por fim, o arquivo é carregado e a nova tarefa é adicionada no dicionário – contido na lista armazenada no arquivo binário – correspondente a sua prioridade.

Como efetuar a visualização de tarefas respeitando o critério determinado? Os usuários devem ser capazes de visualizar suas tarefas em uma determinada ordem, a fim de permitir uma melhor organização. Neste sistema de gerenciamento de tarefas foram estabelecidos que o critério para a visualização das tarefas seriam: primeiro as tarefas de alta prioridade seguindo a ordem crescente de ID dessas tarefas, segundo as tarefas de média prioridade seguindo também a ordem crescente de ID desse tipo de tarefas e terceiro as tarefas de baixa prioridade, que assim como as outras, deve seguir a ordem crescente de ID.

Para a implementação dessa regra, armazenar as tarefas em dicionários dentro de uma lista foi um grande facilitado. Inicialmente o arquivo de tarefas do usuário é aberto, caso o arquivo não exista é informado para o usuário que não existem tarefas. Porém se o arquivo existir, a lista com os dicionários de tarefas é copiada, por meio desta cópia, as chaves dos dicionários de cada prioridade de tarefa são obtidas e armazenadas ordenadamente, por meio do método *sorted()*, em três listas distintas, uma para cada prioridade. Posteriormente, um loop percorre as listas contendo o ID de cada prioridade, permitindo a exibição das informações das tarefas de cada ID e prioridade. Além disso, é verificado o comprimento das três lista de contendo os ID, caso a soma do tamanho delas seja igual a zero, indica que não há tarefas cadastradas e é informado para o usuário que não existe tarefas a serem exibidas.

Como editar uma tarefa? Usuários podem sentir a necessidade de alterar alguma informação de suas tarefas, como o nome, prioridade e a descrição. Para isso o usuário deve informar o ID da tarefa que deseja editar.

Neste software, para efetuar a edição de tarefas foi necessário abrir o arquivo de tarefas do usuário e efetuar uma cópia da lista com os dicionários de suas tarefas. Depois disso, é feita uma busca do ID informado pelo usuário em cada um dos dicionário presentes na lista copiada do arquivo com o método *.pop()*. Se ao verificar no primeiro dicionário e não for encontrado o ID referido, uma exceção do tipo *KeyError* é emitida e passa para o próximo

dicionário, caso também não encontre não segundo dicionário o ID, a mesma exceção é relatada e passa para o último dicionário e se por ventura também não encontre a exceção é disparada novamente, então é perguntado ao usuário se ele deseja tentar novamente ou voltar ao menu de tarefas.

Por outro lado, caso o ID seja encontrado em algum dos dicionários, a tarefa é retirada do dicionário que pertence e armazenada em uma variável. Como a tarefa em si é um objeto, é utilizado um método da própria tarefa para modificar suas informações. Depois, quando terminada as alterações, a tarefa é inserida novamente no dicionário em que foi retirada. Esse dicionário pertence a lista copiada do arquivo. Por sim, a lista copiada e agora atualizada, é inserida no arquivo, substituindo a antiga.

Como efetuar a exclusão de uma tarefa? Com o decorrer do tempo, as tarefas dos usuários são concluídas, o que requer que sejam excluídas do quadro de tarefas do usuário. Assim como para editar uma tarefa, para excluir deve ser informado o ID tarefa que é desejada excluir.

Para e exclusão de uma tarefa o procedimento é semelhante, até certo ponto, com o implementado na edição de tarefas. Da mesma forma, o arquivo de tarefa do usuário é aberto e uma cópia da lista de dicionários é feita. Então de forma semelhante, utilizando o método *.pop()* é feita a busca do ID da tarefa informada para ser excluída nos dicionários presentes na cópia da lista do arquivo. Caso o ID seja encontrado a tarefa é retirada do dicionário e o arquivo de tarefas é atualizado, substituindo a lista presente nele pela lista copiada anteriormente, a qual agora é a lista atualizada. Porém se não for encontrado o ID é retornada uma mensagem informando ao usuário que o ID inserido não é válido.

2.1. Organização do código

A estrutura do código está organizada da seguinte maneira:

- Métodos e bibliotecas importados;
- Inicialização de algumas variáveis, listas e dicionários;
- Funções que desempenham rotinas mais especificas e módulos que executam as opções do sistema;
- Programa principal.

3. Conclusões

Foi descrito neste relatório, um conjunto de passos do desenvolvimento de um software gerenciador de tarefas, desenvolvido na linguagem de alto nível Python em sua versão 3.7.4. O problema proposto conseguiu ser resolvido e para isso, foi necessária a utilização de elementos como listas, dicionários, estruturas de repetições, estruturas condicionais, bibliotecas, funções, métodos, variáveis, classes e objetos. O produto foi testado através do terminal do sistema operacional Windows e Linux, apresentando o funcionamento de forma adequada, sem apresentar problemas.

Os questionamentos abordados na introdução procuraram ser respondidos ao longo do desenvolvido do relatório, permitindo seguir uma lógica linear para a resolução do problema proposto.

Mesmo com o sistema de gerenciamento de tarefa funcionando de forma adequada, melhorias podem ser feitas e implementadas. Uma das melhorias que poderiam ser incluída

no software, seria uma interface gráfica que facilitaria a organização das tarefas e proporcionaria uma melhor experiência aos usuários.

O relatório apresenta como pontos fortes o levantamento de questões pertinentes para a solução do problema proposto e a explicação de como do código se encontra estruturado. Contudo, pontos fracos podem ser observados, tais como o uso de termos repetidamente e a não explicação detalhada do funcionamento de algumas estruturas utilizadas no software.

Referências

Borges, L. E. (2010) “Python para Desenvolvedores”, 2ª edição, https://ark4n.files.wordpress.com/2010/01/python_para_desenvolvedores_2ed.pdf. Acessado em 23 mar. 2020.

Downey, A. B. (2016) “Pense em Python”, 2ª edição, https://pense-python.caravela.club/introducao.html. Acessado em 23 mar. 2020.