Sistema de Atendimento Ambulatorial

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. *A Universidade Estadual de Feira de Santana (UEFS), solicitou a equipe de informática da universidade o desenvolvimento de um software de atendimento. Este software deve auxiliar o atendimento dos pacientes do Ambulatório da UEFS, permitindo o controle de entrada e saída dos pacientes dos consultórios, além de dispor dos tempos médios de atendimento e espera dos pacientes. Para a implementação deste sistema foi utilizada a linguagem e programação Python em sua versão 3.7.4.*

1. Introdução

Ambulatórios médicos são ambientes em que recebem um grande número de pacientes, deste modo, existe uma necessidade de organizar a estrutura de atendimento, afim de garantir que todo o processo seja ágil e eficiente. Cogitada a inauguração do ambulatório da Universidade Estadual de Feira de Santana (UEFS), a Assessoria Especial de Informática da UEFS (AEI) foi contata pela reitoria da Universidade para a implementação de um sistema de atendimento aos pacientes no ambulatório. A AEI levou a proposta do projeto ao DAEComp, os quais decidiram solicitar o desenvolvimento do sistema aos estudantes de Engenharia de Computação participantes do Módulo Integrador Algoritmos.

Este sistema de atendimento permite que seja cadastrado pacientes gerando uma senha exclusiva para cada um deles, possibilitando identifica-los se são pacientes comuns ou preferenciais e em qual consultório será sua consulta. Além disso, neste software, ainda é possível chamar um paciente para o atendimento em determinado consultório, encerrar uma consulta e retirar da fila de espera desistentes do atendimento. No sistema ainda existe a possibilidade de verificar quais pacientes estão nas filas de esperas dos consultórios, pacientes atendidos por cada médico, tempo médio das pessoas nas filas de espera e o tempo médio de atendimento nos consultórios.

Assim, visto a sintetização supracitada, esse relatório se apresenta com o objetivo de discorrer acerca desse software de atendimento para o ambulatório da UEFS, desenvolvido na linguagem de alto nível Python. Porém, para que o problema fosse concluído alguns questionamentos foram levantados e precisaram ser respondidos, sendo eles: 1) Como modularizar o código?; 2) Como ler horários, armazena-los e manipula-los; 3) Como cadastrar pacientes e gerar suas senhas?; 4) Como chamar pacientes para o atendimento respeitando a política 50-50 e suas exceções?; 5) Como remover uma pessoa da fila de espera caso haja desistência?; 6) Como encerrar uma consulta realizada?; 7) Como exibir os pacientes na fila de espera e atendidos?; 8) Como contabilizar os tempos médios de atendimento e de espera dos pacientes?

2. Desenvolvimento

Nesta seção de desenvolvimento procura-se elucidar como é a organização do código e também responder os questionamentos levantados na introdução deste relatório.

Como modularizar o código? Com o intuito de poder reutilizar partes do código em um futuro, foi-se feita a modularização do mesmo. A modularização de código consiste em dividir ele em pequenas partes que executam determinadas funções. Além disso, a vantagem de ter um código modular é tornar o código menos repetitivo e consequentemente menor [Downey 2016].

Na linguagem Python podem ser criadas funções a partir da palavra-chave *def* seguida do nome da função e parênteses [Downey 2016]. Dentro dos parênteses estão os parâmetros da função. Esses parâmetros permitem que dados sejam inseridos na função e ela execute determinada tarefa, porém parâmetros não são obrigatórios em todas as funções. Para mais, funções podem retornar alguma informação ou simplesmente executar suas ações internas sem qualquer retorno.

As funções apresentam um escopo de variável local, ou seja, se passado uma variável como parâmetro em uma função, uma cópia dessa variável é utilizada na execução da função e toda modificação feita a essa cópia da variável não é refletida na variável original. Por outro lado, se forem inseridas listas as modificações associadas a elas fletem nas listas originais.

Assim, por meio das funções foi possível deixar o software de atendimento do ambulatório modular. Sendo divido em módulos principais que executam cada um uma opção do menu principal e em funções que executam tarefas mais especificas dentro destes módulos principais.

Como ler horários, armazena-los e manipula-los? Em determinados trechos do programa é necessário fazer a leitura de horários e armazena-los. Esses horários são utilizados para determinar o tempo de espera e o tempo de atendimento dos pacientes.

A linguagem Python dispõe do módulo *datetime* que traz consigo diversos métodos e classes relacionadas a datas e horários. Um desses métodos é o *datetime.strptime* que permite converter dados de horário do tipo *str* para *datetime.datetime*, proporcionando o reconhecimento e manipulação do horário no programa. Já uma classe importante neste módulo é a t*imedelta* que permite a manipulação de horários, possibilitando subtrair um horário de outro e encontrar o tempo entre eles.

Desta maneira, para a recepção das horas no software, é lido os dados fornecidos pelo usuário como *str* e posteriormente utilizando o método *datetime.strptime* é transformado no formato reconhecido pela linguagem Python como horário.

Já para a determinação do tempo de espera na fila e do tempo de atendimento dos pacientes foram os utilizados os recursos da classe *timedelta*. Subtraindo o horário de saída do consultório de atendimento do horário de entrada é obtido o tempo de atendimento. Enquanto que subtraindo o horário de entrada no consultório do horário de entrada na fila se tem o tempo de espera dos pacientes em atendimento ou já atendidos. Por outro lado, para pacientes ainda na fila de espera, é subtraída a hora em que é solicitada a visualização da fila de espera da hora em que o paciente entrou na mesma, obtendo desta maneira o tempo de espera do paciente.

Como cadastrar pacientes e gerar suas senhas? É necessário inserir pacientes no sistema de atendimento e cada paciente deve ter uma senha única. O usuário do programa deve informar ao cadastrar um novo paciente se ele é comum ou preferencial e se será atendido no consultório de Dermatologia, Endocrinologia ou Ortopedia.

Para gerar a senha é preciso dispor de um contador de pacientes de cada tipo em cada consultório que foram cadastrados para o atendimento. Foi utilizada uma lista para contar essas informações. Listas são objetos que podem conter qualquer tipo de elemento dentro delas. Esses objetos são manipuláveis, desta forma, elementos presentes neles podem ser acrescentados, retirados e modificados [Borges 2010].

A lista utilizada para armazenar esses dados foi organizada da seguinte maneira: o primeiro elemento e o segundo, respectivamente armazenam o número de pacientes cadastrados comuns e preferenciais do consultório de dermatologia; os próximos dois elementos seguem a mesma ordem, pacientes comuns e preferenciais, porém do consultório de endocrinologia e; os dois últimos elementos também seguem a ordem, contundo armazenam o número de pacientes do consultório de ortopedia.

Após as informações do paciente serem fornecidas a senha é gerada de acordo com o tipo de paciente, o consultório de atendimento e a quantidade de pacientes cadastrados que apresentam essas mesmas características. Então, a senha do paciente é inserida em uma lista de espera correspondente ao seu tipo (preferencial ou comum) e ao consultório.

Por fim, seus dados são adicionados a um dicionário que armazena as informações de todos os pacientes, de determinado consultório. Os dicionários funcionam em um sistema, *chave: valor*, onde é possível acessar o valor do objeto contido no dicionário através de sua chave. No caso em questão, a chave que armazena as informações do paciente (*valor*) é a própria senha gerada para ele.

Como chamar pacientes para o atendimento respeitando a política 50-50 e suas exceções? A política de atendimento requer que os pacientes de cada fila sejam atendidos de forma alternada, ou seja, um paciente prioritário é atendido, depois um paciente comum e segue alternando desta maneira. Porém há uma exceção nessa política: caso um paciente preferencial tenha um tempo de espera maior que o paciente comum, o preferencial pode ser atendido no lugar do comum, mesmo que o paciente anterior tenha sido preferencial.

Para a implementação desse sistema de atendimento, foi utilizada uma variável para cada consultório com intuído de verificar se o paciente a ser chamado é comum ou preferencial. Essas variáveis verificadoras são iniciadas com o valor 1, onde indica que o paciente a ser atendido é preferencial. Ao solicitar a chamada do paciente para o atendimento em determinado consultório, a variável verificadora deste é checada. Caso o verificador tenha valor 1 é chamada uma pessoa da fila preferencial e a variável verificadora recebe o valor 2. Por outro lado, se essa variável analisada é igual a 2, também é analisado se o tempo de espera da primeira pessoa na fila comum é maior do que a da pessoa na fila preferencial, caso seja verdade, é chamado o paciente comum, senão é chamado o paciente preferencial.

Toda essa análise é feita em todos os consultórios ao chamar uma nova pessoa para o atendimento, afim de garantir o pleno cumprimento da política.

Como remover uma pessoa da fila de espera caso haja desistência? A desistência do atendimento por parte dos pacientes não é algo incomum. Por isso, se faz necessário a

possibilidade de remoção de pacientes e todos seus dados do sistema, caso ocorra esse tipo de situação.

Em Python a utilização de um loop *for* é maneira mais fácil de percorrer uma lista. A sintaxe dessa estrutura de repetição para efetuar tal procedimento é a seguinte: *for...in* [Downey 2016].

Neste trecho do programa, foi utilizada duas variáveis contadoras para auxiliar no processo de remoção de paciente da fila de espera. A variável contador1, com o objetivo de representar os índices dos pacientes para escolha e a variável contador2 para a correção do valor inserido pelo usuário, caso haja pacientes em atendimento na fila.

O loop *for...in* foi utilizado para percorrer a lista de espera de determinado tipo de paciente (comum ou preferencial) do consultório selecionado, exibindo na tela os pacientes. Então uma variável recebe o índice equivalente ao paciente que deseja ser removido.

Na linguagem Python, existe o método *.pop()* utilizado para a remoção de elementos de listas e dicionários, além disso, ao remover os elementos ele retorna o item removido, possibilitando guarda-lo em uma variável e imprimir na tela.

Então, a variável que recebeu o índice é acrescida do contador2 e subtraída da mesma o valor 1, afim de selecionar a posição certa do paciente na fila. Desta maneira, é utilizado o método *.pop()* recebendo com parâmetro o índice indicado pelo usuário para a remoção da senha do paciente da fila de espera. A senha removida da fila de espera é passa como parâmetro novamente no método *.pop(),* utilizado para a remoção dos dados do paciente do dicionário que contém seus dados.

Como encerrar uma consulta realizada? Para que seja chamado um novo paciente para o atendimento em determinado consultório, é necessário encerrar a consulta do último paciente atendido.

Neste software é utilizada a estrutura condicional *if* para verificar se há algum paciente comum ou preferencial em atendimento no consultório escolhido, caso haja é solicitado uma confirmação de encerramento de consulta. Após obter a confirmação, a senha do paciente é removida com o método *.pop()* e as informações do paciente são passadas para uma lista que armazena os dados das pessoas do mesmo tipo (preferencial ou comum) atendidas do consultório em questão.

Como exibir os pacientes na fila de espera e atendidos? É comum a necessidade ter informações sobre quantas pessoas ainda faltam atender e quantas pessoas foram atendidas em cada consultório. Isso permite ter uma perspectiva da demanda do ambulatório e do ritmo de atendimento dos pacientes.

Neste sistema, foram utilizadas estruturas de repetição do tipo *for...in* para percorrer a lista de pacientes em espera do consultório escolhido, mostrando a senha do paciente, o nome, as informações de tempo de espera (obtido como descrito na seção 2.2), e se o paciente está em atendimento – visto que o paciente só é movido para a lista de pacientes atendidos após o termino na consulta – ou em espera.

Em paralelo, para a exibição dos pacientes atendidos em cada consultório, foi-se feito o uso, novamente, da estrutura *for...in.* Com a finalidade de exibir o nome das pessoas atendidas, o tempo na fila para ser atendido, o tempo de atendimento (descrito na seção 2.2), o consultório de atendimento e o médico responsável pelo atendimento.

Como contabilizar os tempos médios de atendimento e de espera dos pacientes? De posse das informações dos tempos de espera nas filas dos consultórios e do tempo de atendimento dos pacientes, é possível contabilizar os tempos médios de espera e de atendimento.

Nesse sistema é pretendido exibir o tempo médio de espera do ambulatório como um todo, além do tempo médio de espera das filas comuns e preferenciais do ambulatório todo e de cada consultório. A mesma premissa se aplica para o tempo médio de atendimento.

No software em questão, foram utilizadas duas variáveis acumuladoras de tempo para cada consultório. Essas variáveis são iniciadas utilizando o método *datetime.timedelta* atribuindo um tempo nulo (igual a zero), com o intuito de não trazer consigo restos, visto que são acumuladoras de tempo utilizados para dados estatísticos. Além disso, foram utilizadas duas variáveis contadoras de pacientes para cada consultório.

De maneira a facilitar o processo de computação das médias, em cada consultório, os diferentes acumuladores guardam a soma dos tempos de atendimento dos pacientes preferenciais e comuns. Além disto, são contabilizadas a quantidade de pacientes em cada fila dos consultórios.

Deste modo, os tempos de espera de cada fila nos consultórios é calculado dividindo-se a soma do tempo de espera pelo número de pacientes daquele tipo. Já para o cálculo do tempo médio de espera dos filas comuns e preferenciais do ambulatório como um todo, os tempos comuns e preferenciais de todos consultórios são divididos pelas respectivas quantidades de pacientes atendidos. Então, para cálculo da média do tempo de atendimento de todo ambulatório, são somados os tempos de espera de todas as filas e divido pela quantidade total de pacientes já atendidos.

Por fim, os tempos médios de atendimento são obtidos seguindo a mesma estrutura dos tempos médios de espera.

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
- Módulos principais que executam cada opção do sistema
- Programa principal.

3. Conclusões

Foi descrito neste relatório, uma série de passos do desenvolvimento de um software de atendimento ambulatorial, desenvolvido na linguagem Python em sua versão 3.7.4. O problema proposto conseguiu ser sanado, e para tal, se fez necessário a utilização de elementos como listas, dicionários, estruturas de repetições, estruturas condicionais, bibliotecas, funções, métodos e variáveis, os quais alguns tiveram seu funcionamento explicado no decorrer do relatório. O produto foi testado através do terminal do sistema operacional Windows e funcionou de forma adequada, sem apresentar quaisquer problemas.

As questões levantadas na introdução foram sendo respondidas ao longo da seção de desenvolvimento do relatório, desta maneira a solução do problema pôde ser construída seguindo uma certa linearidade.

Apesar do sistema de atendimento funcionar perfeitamente e atender aos requisitos solicitados, algumas melhorias poderiam ser implementadas. Poderia ser implantado um banco de dados para armazenar os tempos médios de espera e atendimento do ambulatório, permitindo que seja possível fazer comparações futuras entre os dados. Além disso, os horários poderiam ser obtidos de forma automática através de ferramentas disponíveis na linguagem Python

O relatório apresenta como pontos fortes a descrição do funcionamento de diversas estruturas utilizadas no código e o levantamento de questões pertinentes para a solução do problema proposto. Em contrapartida, existe o uso repetido de diversos termos e palavras que podem empobrecer a argumentação, também pode existir inconsistências na conexão entre raciocínios que dificultariam a compreensão do leitor em certos trechos.

Referências

Borges, L. E. (2010) “Python para Desenvolvedores”, 2ª edição, https://ark4n.files.wordpress.com/2010/01/python_para_desenvolvedores_2ed.pdf. Acessado em 28 Jan. 2020.

Downey, A. B. (2016) “Pense em Python”, 2ª edição, https://pense-python.caravela.club/introducao.html. Acessado em 28 Jan. 2020.