Relatório do trabalho em Python - Simulação de uma Clínica Médica

Data: 2026-01-11

Unidade Curricular: Algoritmos e Técnicas de Programação

Docentes: José Carlos Ramalho e Luís Filipe Cunha

Grupo : Cláudia Reis(a103262), Daniela Faria(a107228) e Pedro Oliveira(a100625)

Neste relatório iremos abordar o funcionameto das funções criadas e usadas no ficheiro "main.py", e "simulacao.py" que são então importadas para o ficheiro "interface.py". Por fim também iremaos colocar neste relatório a documentação detalhada de como utlizar o sistema. Este projeto permite-nos fazer pesquisas e análises de uma base de dados fornecido no nome de "pessoas.json". Decidimos incorporar três ficheiros no desenvolvimento do projeto de forma a facilitar a interação entre as funções e a interface gráfica de forma a organizar e permitir uma melhor perceção do funcionamento do sistema.

1. simulacao.py

O dicionário DOENCA_TO_ESP estabelece a correspondência entre diferentes doenças e as respetivas especialidades médicas. Este mapeamento permite ao sistema encaminhar automaticamente cada paciente para o serviço mais apropriado, contribuindo para uma simulação mais realista do funcionamento de um hospital.

A classe Paciente representa a entidade central do sistema, modelando cada indivíduo atendido. No seu construtor são definidos os principais dados do paciente, como identificador interno, número de identificação, nome, idade, profissão e prioridade clínica. É utilizada a estrutura kwargs para permitir a inclusão flexível de outras informações relevantes, como sexo, morada, descrição clínica, atributos pessoais, religião e atividades desportivas. O método repr define a representação textual de um objeto da classe Paciente, sendo utilizado sempre que o objeto é impresso ou exibido no ecrã. Neste caso, apresenta de forma simples o nome do paciente e a sua prioridade clínica, facilitando a leitura e interpretação dos resultados da simulação.

A função carregar_pacientes_json é responsável pela leitura e criação de objetos do tipo Paciente a partir de um ficheiro no formato JSON. O seu objetivo principal é transformar dados externos armazenados em ficheiro em estruturas internas que podem ser utilizadas pelo sistema de simulação.

Inicialmente, a função verifica se o ficheiro indicado existe no sistema. Caso não exista, é apresentado um aviso ao utilizador e é devolvida uma lista vazia, prevenindo erros durante a execução do programa. De seguida, o ficheiro é aberto em modo de leitura com codificação UTF-8, sendo o seu conteúdo convertido para uma estrutura de dados Python através da função json.load. Qualquer erro ocorrido durante este processo é tratado por um bloco de exceção, garantindo que o programa não termina abruptamente.

Após a leitura bem-sucedida, o programa percorre sequencialmente os registos existentes no ficheiro. Para cada entrada, é analisada a profissão do indivíduo, de modo a identificar e excluir registos correspondentes a profissionais de saúde, nomeadamente médicos, que não devem ser considerados como pacientes no contexto da simulação.

Em seguida, é atribuído a cada paciente um identificador interno e é aplicada uma lógica rigorosa para obter o número de identificação real, priorizando o campo “CC”, depois “BI” e, em último recurso, “cc”. Caso nenhuma destas informações esteja disponível, é utilizado o valor “N/A”, garantindo assim a consistência dos dados.

Para cada registo válido é criado um novo objeto da classe Paciente, sendo preenchidos os seus principais atributos, como nome, idade, profissão, prioridade clínica e outros dados pessoais relevantes, incluindo sexo, morada, características pessoais, religião e práticas desportivas. Todos os objetos criados são armazenados numa lista de pacientes.

Após o carregamento completo, a lista de pacientes é baralhada aleatoriamente, simulando uma ordem de chegada não determinística ao hospital. Caso seja fornecido um limite máximo de pacientes, a lista é truncada para esse número. Por fim, é apresentada uma mensagem de sucesso indicando quantos pacientes foram carregados e a lista final é devolvida para utilização pelo sistema.

A função gera_intervalo_tempo_chegada tem como objetivo calcular o intervalo de tempo entre chegadas consecutivas de pacientes, utilizando um modelo probabilístico. O seu funcionamento baseia-se numa distribuição exponencial, frequentemente usada para modelar processos de chegada aleatória, como fluxos de pessoas num hospital.

A função recebe como parâmetro a taxa de chegadas taxa, que representa o número médio de chegadas por hora, e um gerador de números aleatórios opcional (rng). Inicialmente, o valor do intervalo de tempo é definido como infinito, garantindo que, no caso de uma taxa inválida ou nula, não sejam geradas chegadas.

Quando a taxa é superior a zero, esta é convertida para uma taxa por minuto, permitindo trabalhar com unidades de tempo mais adequadas à simulação. Em seguida, é gerado um valor aleatório segundo uma distribuição exponencial, cujo parâmetro é o inverso da taxa por minuto, representando assim o tempo esperado até à próxima chegada. Caso seja fornecido um gerador de números aleatórios específico, este é utilizado para garantir reprodutibilidade da simulação; caso contrário, é utilizado o gerador padrão da biblioteca NumPy.

O valor calculado é devolvido pela função, correspondendo ao intervalo de tempo, em minutos, até à próxima chegada de um paciente, permitindo assim controlar de forma realista o ritmo de entradas no

sistema hospitalar.

A função gera_tempo_consulta é responsável por gerar a duração de uma consulta médica de forma probabilística, permitindo modelar o tempo de atendimento de cada paciente de acordo com diferentes distribuições estatísticas. Esta abordagem torna a simulação mais realista, uma vez que, na prática, o tempo de consulta varia de paciente para paciente.

A função recebe como parâmetros o valor médio da duração da consulta (media), o tipo de distribuição estatística a utilizar (distribuicao) e, opcionalmente, um gerador de números aleatórios (rng). Inicialmente, o tempo de consulta é definido como zero e, de seguida, é calculado de acordo com a distribuição selecionada.

Quando a distribuição escolhida é exponencial, o tempo de consulta é gerado através de uma distribuição exponencial com média igual ao valor fornecido, sendo esta uma escolha adequada para modelar tempos de serviço em sistemas de filas de espera. Caso a distribuição seja normal, o tempo é gerado em torno do valor médio, com um desvio padrão correspondente a 20% desse valor, sendo imposto um valor mínimo de 0,1 para evitar durações negativas ou irrealistas. Para a distribuição uniforme, o tempo de consulta é escolhido aleatoriamente dentro de um intervalo compreendido entre metade e uma vez e meia o valor médio, garantindo variabilidade controlada.

Se for fornecido um tipo de distribuição desconhecido, a função utiliza por defeito a distribuição exponencial, assegurando assim a robustez do sistema. No final, o valor gerado é devolvido, representando a duração estimada da consulta médica a ser utilizada na simulação.

A função calcular_estatisticas tem como objetivo recolher, processar e sintetizar os principais indicadores de desempenho da simulação, tanto a nível individual de cada médico como a nível global do sistema. Estes indicadores permitem avaliar a eficiência do funcionamento da clínica e o impacto das decisões de alocação de recursos.

Inicialmente, a função percorre a lista de médicos presentes na simulação. Para cada médico são recolhidos os tempos de consulta realizados, o número de pacientes atendidos e o tempo total de ocupação. A partir destes valores, é calculada a duração média das consultas e a percentagem de ocupação do médico ao longo do período total da simulação, garantindo que este valor nunca excede 100%. Estas estatísticas individuais são armazenadas numa estrutura própria, permitindo uma análise detalhada do desempenho de cada profissional de saúde.

De seguida, a função processa os principais indicadores globais da simulação. São construídos vetores contendo os tempos de espera dos pacientes, os tempos de consulta e a evolução do tamanho da fila ao longo do tempo. Através destes dados, são calculados o tempo médio de espera, o tempo médio de consulta, o tamanho médio e máximo da fila e o número total de doentes atendidos.

A ocupação global dos médicos é igualmente analisada, utilizando uma linha temporal que regista o número de médicos ocupados em cada instante da simulação. Este valor é normalizado pelo número total de médicos disponíveis, permitindo obter a percentagem média de ocupação global do sistema, também limitada a um máximo de 100%, garantindo coerência estatística.

Além disso, é determinado o tempo médio total que um paciente permanece na clínica, desde a sua chegada até à saída. Por fim, a função devolve um dicionário contendo todas estas métricas, incluindo variâncias dos tempos de espera e de consulta, estatísticas globais e estatísticas individuais por médico. Este conjunto de resultados constitui a base para a análise do desempenho e da qualidade do serviço prestado pelo sistema simulado.

A classe SimulacaoClinica representa o núcleo do sistema de simulação, sendo responsável por configurar, executar e armazenar toda a informação relacionada com o funcionamento de uma clínica ao longo do tempo. O seu construtor recebe diversos parâmetros que permitem personalizar o comportamento da simulação, como a taxa média de chegadas de pacientes, o número de médicos disponíveis, o tipo de distribuição estatística dos tempos de atendimento, a duração média das consultas, o tempo total de simulação e a semente utilizada para a geração de números aleatórios, garantindo reprodutibilidade dos resultados.

Para além destes parâmetros principais, o construtor permite ainda definir o padrão de chegadas dos pacientes, perfis de chegada personalizados, a lista inicial de pacientes e as especialidades atribuídas a cada médico. Após a leitura de todos os parâmetros, é invocado o método reset, que inicializa o estado interno da simulação.

O método reset prepara todas as estruturas de dados necessárias para uma nova execução da simulação. São criadas listas para armazenar tempos de espera, tempos de consulta, tempo total dos pacientes na clínica, evolução do tamanho das filas e ocupação dos médicos. São igualmente inicializadas estruturas para registo de eventos, associação entre pacientes e médicos, bem como contadores internos e um gerador de números aleatórios com base na semente fornecida.

Adicionalmente, é criada a estrutura que representa os médicos disponíveis na clínica. Cada médico é caracterizado por um identificador, estado de disponibilidade, tempo previsto de término da consulta atual, especialidade, histórico de eventos, número de pacientes atendidos e tempos individuais de consulta. Caso não seja especificada uma especialidade para um determinado médico, é atribuída uma especialidade padrão, garantindo consistência no sistema.

Por fim, são inicializadas as filas de espera por especialidade, o contador interno de pacientes e um vetor que acompanha, ao longo do tempo, o número de médicos ocupados. Este conjunto de componentes forma a base operacional da simulação, permitindo controlar com precisão o comportamento do sistema clínico e recolher métricas fundamentais para posterior análise e avaliação do seu desempenho.

O método gera_intervalo_chegada_homogeneo é responsável por determinar o intervalo de tempo entre chegadas consecutivas de pacientes, assumindo um padrão de chegadas homogéneo, isto é, com uma taxa constante ao longo de todo o período da simulação. Este modelo representa uma situação em que os pacientes chegam à clínica de forma regular e aleatória, sem variações significativas de procura ao longo do tempo.

Inicialmente, o valor do intervalo de chegada é definido como infinito, garantindo que, na ausência de pacientes disponíveis para serem gerados, não ocorram novas chegadas. Quando existe uma lista de pacientes válida, o método invoca a função gera_intervalo_tempo_chegada, fornecendo a taxa média

de chegadas e o gerador de números aleatórios interno da simulação. Esta função calcula, através de uma distribuição exponencial, o tempo esperado até à próxima chegada, produzindo assim um comportamento estocástico realista do fluxo de pacientes.

O método gera_chegadas_naohomogeneo é responsável por gerar eventos de chegada de pacientes segundo um modelo de chegadas não homogéneo, no qual a taxa de chegadas varia ao longo do tempo. Este tipo de modelação permite representar de forma mais realista o funcionamento de uma clínica, uma vez que a procura tende a ser diferente ao longo do dia.

Caso exista uma lista de pacientes disponível, o método começa por obter o perfil de chegadas definido pelo utilizador. Se nenhum perfil for fornecido, é utilizado um perfil padrão composto por vários intervalos de tempo, cada um com uma taxa média de chegadas distinta. Cada bloco do perfil é caracterizado por um intervalo temporal e por uma taxa de chegadas associada.

Para cada bloco, o método simula as chegadas dos pacientes através de uma distribuição exponencial, utilizando a taxa correspondente ao intervalo considerado. À medida que o tempo avança dentro de cada bloco, são gerados novos eventos de chegada até ao final do intervalo ou até que todos os pacientes tenham sido agendados.

Sempre que ocorre uma nova chegada válida, é atribuído um identificador único ao paciente, o instante de chegada é registado e o evento é inserido numa estrutura de prioridade (fila de eventos), garantindo que os eventos são processados pela ordem cronológica correta. Em simultâneo, é mantida a associação entre o identificador do paciente e a respetiva posição na lista de pacientes.

No final do processo, o contador interno de pacientes é atualizado, assegurando a continuidade e consistência da simulação. Este mecanismo permite ao sistema reproduzir variações de procura ao longo do tempo, como períodos de maior afluência ou momentos de menor movimento, aumentando significativamente o realismo do modelo.

O método gera_chegadas_homogeneo é responsável por gerar os eventos de chegada de pacientes de acordo com um padrão homogéneo, em que a taxa de chegadas é constante ao longo de toda a simulação. Este método simula um fluxo de pacientes regular e aleatório, ideal para cenários em que se assume uma procura estável ao longo do tempo.

Inicialmente, o método obtém o intervalo até à primeira chegada, utilizando a função _gera_intervalo_chegada_homogeneo. Em seguida, é inicializado um contador de identificadores de pacientes e um índice para percorrer a lista de pacientes disponíveis. O método entra num ciclo que continua enquanto o tempo da próxima chegada seja inferior ao tempo total da simulação e existam pacientes por agendar.

Dentro do ciclo, cada paciente recebe um identificador único, o instante de chegada é registado e o evento correspondente é inserido numa fila de eventos priorizada, garantindo que os eventos são processados na ordem cronológica correta. É mantida também a associação entre o identificador do paciente e a sua posição na lista de pacientes.

Após cada inserção, o tempo da próxima chegada é atualizado, somando um novo intervalo de chegada gerado de forma estocástica. No final, o contador interno de pacientes é atualizado,

garantindo consistência na simulação e permitindo que futuras chegadas continuem a ser numeradas de forma sequencial.

O método detectar_doenca_e_prioridade tem como objetivo identificar a doença do paciente, determinar a prioridade do atendimento e registar notas clínicas relevantes. Inicialmente, assume-se uma doença padrão (“virose”) e prioridade normal. Caso o paciente possua informações detalhadas, o método extrai a doença a partir do campo “doenca” ou, se indisponível, do campo “descrição”, convertendo o valor para minúsculas para uniformizar o tratamento. Adicionalmente, são verificadas características específicas do paciente, como restrições religiosas (“Testemunhas de Jeová”) ou hábitos de saúde (“fumador”), que são registadas como notas clínicas. No final, a função devolve a doença identificada, a prioridade e um resumo das notas clínicas.

O método doenca_para_especialidade traduz o diagnóstico do paciente para a especialidade médica adequada. Primeiro, procura uma correspondência exata no dicionário DOENCA_TO_ESP. Caso não encontre, tenta identificar correspondências parciais com os nomes das doenças. Se ainda assim não houver correspondência, são aplicadas regras heurísticas simples, como encaminhar doenças relacionadas com o coração para “cardiologia” ou problemas musculoesqueléticos para “ortopedia”. Caso nenhuma regra se aplique, é utilizada a especialidade padrão definida pelo sistema.

O método registar_ocupacao_timeline permite atualizar a ocupação dos médicos ao longo do tempo. Recebe o minuto de início da consulta e a duração, convertendo-os em um intervalo de minutos discretos. Para cada minuto dentro do intervalo, incrementa-se o contador de médicos ocupados, garantindo que a informação sobre a utilização dos recursos ao longo do dia seja registada de forma precisa. Esta informação é crucial para calcular métricas de ocupação e eficiência da clínica.

A função run é responsável por executar todos os eventos ao longo do tempo e calcular as métricas de desempenho. Inicialmente, a simulação é reinicializada através do método reset, garantindo que todas as estruturas de dados estão limpas e prontas para um novo ciclo. Caso não existam pacientes, a execução é interrompida.

Dependendo do padrão de chegadas definido, os eventos de chegada são gerados por meio dos métodos _gera_chegadas_homogeneo ou _gera_chegadas_nonhomogeneous, criando uma fila de eventos priorizada que organiza todas as chegadas e saídas de pacientes ao longo do tempo. Para cada especialidade é criada uma fila de espera, incluindo a fila padrão de clínica geral.

A simulação processa os eventos de forma ordenada pela hora de ocorrência, utilizando uma fila de prioridade (heapq). Quando ocorre um evento de chegada, o sistema identifica o paciente correspondente e determina a doença, a prioridade e as notas clínicas através do método _detectar_doenca_e_prioridade. A doença é então mapeada para a especialidade apropriada usando _doenca_para_especialidade. É verificada a disponibilidade de médicos, seguindo uma ordem de prioridade: primeiro especialistas na doença, depois médicos de clínica geral, e finalmente qualquer médico disponível. Se houver um médico livre, é gerado o tempo de consulta com base na distribuição estatística definida, o paciente é associado ao médico, a ocupação é registada e um evento de saída é agendado. Caso nenhum médico esteja disponível, o paciente entra na fila correspondente.

Quando ocorre um evento de saída, o médico associado ao paciente é marcado como livre e o sistema verifica se existem pacientes na fila à espera. Se houver, o próximo paciente é imediatamente atendido, garantindo a continuidade do fluxo. Durante toda a simulação, são registados os tempos de chegada, início, duração e saída, bem como a evolução das filas e a ocupação dos médicos minuto a minuto.

Após a execução de todos os eventos, são calculadas métricas detalhadas, incluindo tempos de espera, tempos de consulta, tempo total na clínica e ocupação média dos médicos. Estas informações são armazenadas para posterior análise estatística, sendo o método calcular_estatisticas invocado para compilar as estatísticas globais e individuais por médico.

Este processo permite simular de forma realista a dinâmica de atendimento numa clínica, fornecendo dados úteis para avaliação de desempenho, planeamento de recursos e análise de eficiência do serviço.

2. main.py

A função load_initial_config é responsável por carregar a configuração inicial da simulação da clínica, definindo os parâmetros essenciais para o seu funcionamento. Inicialmente, são atribuídos valores padrão a diversos parâmetros, incluindo a taxa média de chegadas de pacientes (lambda_rate), o número de médicos disponíveis, o tipo de distribuição estatística dos tempos de atendimento, a duração média das consultas, o tempo total de simulação, o padrão de chegadas dos pacientes, o ficheiro de dados com os registos dos pacientes e as especialidades dos médicos.

Posteriormente, a função verifica a existência de um ficheiro de configuração externo (CONFIG_FILE). Caso o ficheiro exista, o seu conteúdo é carregado e os valores fornecidos são utilizados para atualizar os parâmetros padrão. Para garantir consistência, valores numéricos são convertidos para o tipo adequado quando necessário. Qualquer erro na leitura do ficheiro é tratado de forma controlada, recorrendo aos valores padrão definidos inicialmente, assegurando que a simulação pode ser executada sem interrupções.

Se o ficheiro de configuração não for encontrado, é apresentada uma mensagem informativa e os valores padrão são utilizados. No final, a função devolve um dicionário com a configuração completa da simulação, garantindo que todos os parâmetros necessários estão disponíveis para inicializar a execução da clínica.

A função parse_cli_arguments utiliza a biblioteca argparse para definir argumentos opcionais correspondentes a cada parâmetro relevante da simulação, como a taxa média de chegada de pacientes por hora (lambda_rate), o número de médicos disponíveis, o tipo de distribuição dos tempos de consulta, o tempo médio de atendimento, a duração total da simulação, o padrão de chegadas dos pacientes e o ficheiro de dados a ser utilizado.

Após a definição dos argumentos, a função processa os valores fornecidos pelo utilizador e atualiza o dicionário de configuração inicial (config), substituindo os valores padrão pelos valores especificados na linha de comandos, caso existam. Este procedimento garante que a simulação pode ser personalizada de forma dinâmica, sem necessidade de intervenção manual nos ficheiros de configuração.

3.interface.py

A interface gráfica do utilizador diz respeito à camada de apresentação do sistema de simulação clínica, sendo responsável por permitir a interação entre o utilizador e o motor de simulação desenvolvido no módulo “simulacao.py”. O seu principal objetivo é disponibilizar uma forma clara e intuitiva de configurar parâmetros, executar simulações, acompanhar a evolução do sistema em tempo real e analisar os resultados obtidos. A função embed_plot_on_frame tem como objetivo incorporar figuras do Matplotlib dentro de um frame Tkinter. Esta função remove quaisquer elementos previamente existentes no frame e insere um novo objeto FigureCanvasTkAgg, permitindo a atualização dinâmica dos gráficos sempre que necessário. Esta abordagem garante que os gráficos são corretamente renderizados dentro da interface gráfica e não em janelas externas. De seguida, são implementadas várias funções específicas para a geração de gráficos, cada uma responsável por visualizar um aspeto distinto da simulação:

A função grafico_distritos_bar constroi um gráfico de barras que apresenta os distritos com maior número de pacientes, permitindo analisar a proveniência geográfica da procura.

A função grafico_tempo_espera_frame apresenta um histograma dos tempos de espera dos pacientes, fornecendo uma visão clara da variabilidade do sistema.

A função grafico_tempo_total_frame ilustra a distribuição do tempo total que cada paciente permanece na clínica, desde a chegada até à saída.

A função grafico_ocupacao_frame apresenta a evolução temporal da taxa de ocupação dos médicos ao longo da simulação.

A função grafico_fila_frame permite observar a dinâmica da fila de espera, identificando períodos de maior congestionamento.

A função grafico_fila_vs_taxa_frame representa o impacto da variação da taxa de chegada (λ) no tamanho médio da fila, permitindo estudar o comportamento do sistema sob diferentes cenários.

A função grafico_ocupacao_medicos_bar apresenta um gráfico de barras com a percentagem de ocupação de cada médico, facilitando a análise individual do desempenho.

A classe App herda diretamente de tk.Tk e representa a janela principal da aplicação. Nesta são inicializados todos os parâmetros essenciais. Durante a inicialização são definidos os parâmetros iniciais da simulação, é configurado o número inicial de médicos e respetivas especialidades, são criadas variáveis de estado para armazenar pacientes, dados da simulação e animações e é invocado o método “_build_unique_ui”, responsável por construir toda a interface gráfica.

O painel direito da interface concentra todos as ferramentas do sistema, permitindo ao utilizador: definir parâmetros da simulação (taxa de chegada, número de médicos, duração), iniciar e parar a simulação, aceder à análise gráfica dos resultados, gerir especialidades médicas, carregar datasets externos e também pesquisar utentes.

Por sua vez, o painel esquerdo da interface inclui um canvas gráfico onde é representado o tamanho da fila de espera, o estado de cada gabinete médico, o paciente que está atualmente a ser atendido e ainda, indicadores textuais que mostram o número de pacientes em fila e o minuto atual da simulação.

A função config_specs abre uma janela secundária que permite ao utilizador associar uma especialidade a cada médico. Esta funcionalidade é particularmente importante para simulações mais realistas, pois influencia diretamente a forma como os pacientes são encaminhados e atendidos.

A função load_json permite ao utilizador carregar um ficheiro JSON contendo dados dos pacientes. Após o carregamento, os dados são armazenados internamente e o nome do ficheiro e o número de registos são apresentados na interface.

A função open_search implementa um sistema completo de pesquisa de pacientes, permitindo filtrar resultados por nome, número de identificação, idade e sexo. Os resultados são apresentados numa tabela interativa.

Ao selecionar um paciente, é apresentada uma ficha detalhada contendo dados pessoais, informação religiosa, triagem clínica e observações clínicas detalhadas. Quando os dados clínicos são inexistentes ou inválidos, o sistema gera automaticamente descrições coerentes.

A função start_sim é responsável por iniciar a simulação, validando previamente a existência de um dataset carregado.