



Universidade do Minho
Escola de Engenharia

Relatório de Projeto Laboratorial

Simulação de uma Clínica Médica

Algoritmos e Técnicas de Programação

Ano Letivo 2025/2026

Bárbara Jacques, a109722

Esmeralda Freitas, a109932

Martim Rocha, a112252

Docentes:

José Carlos Leite Ramalho

Luís Filipe Costa Cunha

Braga, 29 de dezembro de 2025 (2026?)

Resumo

Este projeto consiste no desenvolvimento de uma aplicação em Python para a simulação de uma clínica médica, utilizando a metodologia de Simulação de Eventos Discretos (DES). O objetivo principal foi modelar o fluxo de atendimento de doentes, desde a sua chegada (baseada num processo de Poisson) até à alta médica, passando pela triagem e consulta.

A aplicação foi estruturada de forma modular para garantir a escalabilidade e a manutenção do código. O núcleo da simulação (`principal.py` e `manipulacao.py`) gere uma lista de eventos prioritária, processando chegadas e saídas cronologicamente¹¹¹¹¹. Os doentes são encaminhados para filas de espera específicas (Cardiologia, Pneumologia, Clínica Geral) com base em regras de triagem aplicadas a um dataset real de utentes (`peessoas.json`).

Para a interação com o utilizador, desenvolveu-se uma Interface Gráfica (GUI) robusta utilizando a biblioteca `FreeSimpleGUI3`. Esta interface permite a configuração dinâmica dos parâmetros da simulação (número de médicos, taxa de chegada λ , distribuição estatística do tempo de consulta) e oferece visualização em tempo real do progresso e dos eventos.

Adicionalmente, o projeto integrou um módulo de Análise de Dados e Visualização⁵. Após a simulação, o sistema gera relatórios estatísticos detalhados (tempos médios de espera, taxas de ocupação, dimensão das filas) e gráficos interativos (`Matplotlib`) que ilustram a evolução do sistema ao longo do tempo⁶. Foi também implementada a funcionalidade de exportação destes resultados para ficheiros externos (JSON e TXT), permitindo a persistência e análise posterior dos dados.

Em suma, este trabalho demonstra a aplicação prática de algoritmos complexos e estruturas de dados na resolução de problemas de engenharia biomédica, oferecendo uma ferramenta funcional para o estudo e otimização de recursos em ambiente hospitalar.

Índice

1	Introdução	1
2	Metodologia e Algoritmos	3
2.1	Modelo Matemático e Distribuições	3
2.2	Gestão de Eventos e Filas	3
3	Arquitetura da Solução	5
3.1	Descrição dos Módulos	5
3.1.1	Módulo main.py	5
3.1.2	Módulo interface.py	5
3.1.3	Módulo principal.py	6
3.1.4	Módulo manipulacao.py	6
3.1.5	Módulo analisar.py	7
3.1.6	Módulo graficos.py	7
	Referências	8
	Anexos	9

Lista de Figuras

Lista de Tabelas

1 Introdução

1. Enquadramento e Motivação: A gestão eficiente de unidades de saúde é um dos desafios centrais na Engenharia Biomédica moderna. O desequilíbrio entre a procura (fluxo de doentes) e a oferta (disponibilidade de médicos e recursos) resulta frequentemente em longos tempos de espera e na saturação dos serviços, comprometendo a qualidade do atendimento prestado. Neste contexto, a simulação computacional emerge como uma ferramenta poderosa, permitindo modelar sistemas complexos, prever comportamentos e testar cenários de otimização sem os custos ou riscos associados a testes em ambiente real.

2. Objetivos do Projeto: O presente trabalho, desenvolvido no âmbito da Unidade Curricular de Algoritmos e Técnicas de Programação da Licenciatura em Engenharia Biomédica, tem como objetivo principal o desenvolvimento de uma aplicação em Python para a simulação do funcionamento de uma clínica médica. O sistema proposto visa modelar o ciclo completo de atendimento do doente, desde a sua chegada à clínica até à conclusão da consulta, permitindo a monitorização de indicadores-chave de desempenho (KPIs), tais como o tempo médio de espera, a dimensão das filas e a taxa de ocupação dos recursos médicos.

3. Abordagem Metodológica: Para representar a natureza estocástica e dinâmica de uma clínica real, adotou-se a metodologia de Simulação de Eventos Discretos (Discrete Event Simulation - DES). O modelo matemático implementado baseia-se em processos estocásticos conhecidos: a chegada de utentes é modelada através de um processo de Poisson (com intervalos entre chegadas a seguir uma distribuição exponencial), enquanto os tempos de serviço (consultas) podem seguir distribuições exponencial, normal ou uniforme, configuráveis pelo utilizador.

A aplicação integra ainda estruturas de dados fundamentais, como filas de espera (queues) baseadas na política FIFO (First-In-First-Out) e listas de prioridade para a gestão cronológica dos eventos. Adicionalmente, o projeto inclui uma interface gráfica para interação com o utilizador e módulos de análise estatística para a interpretação dos resultados gerados.

4. Estrutura do Relatório: Este relatório encontra-se organizado da seguinte forma: o Capítulo 2 descreve a arquitetura do sistema e os algoritmos implementados;

o Capítulo 3 detalha as funcionalidades da interface e as opções de configuração; o Capítulo 4 apresenta a análise dos resultados obtidos em diferentes cenários de simulação; e, por fim, o Capítulo 5 apresenta as conclusões e sugestões de trabalho futuro.

2 Metodologia e Algoritmos

A implementação da simulação seguiu o paradigma de *Discrete Event Simulation* (DES), onde o estado do sistema muda apenas em instantes discretos no tempo, correspondentes à ocorrência de eventos (chegadas ou partidas).

2.1 Modelo Matemático e Distribuições

De acordo com os requisitos do projeto, a modelação estocástica baseou-se nas seguintes premissas:

- **Chegada de Doentes:** Segue um processo de Poisson. Para implementação computacional, isto traduz-se em gerar os intervalos de tempo entre chegadas sucessivas utilizando uma distribuição exponencial com parâmetro λ (taxa de chegada)[cite: 12, 27].
- **Tempo de Atendimento:** O tempo de duração das consultas é gerado aleatoriamente. O sistema permite configurar a distribuição subjacente, suportando distribuições Exponencial, Normal ou Uniforme, permitindo assim testar diferentes cenários de variabilidade clínica[cite: 13, 30].

2.2 Gestão de Eventos e Filas

O núcleo do algoritmo baseia-se numa lista de eventos (*Event Queue*), ordenada cronologicamente. O algoritmo principal processa esta lista sequencialmente:

1. O evento com o menor tempo (t_{min}) é retirado da lista.
2. O relógio da simulação avança para t_{min} .
3. O tipo de evento é processado:
 - Se for **CHEGADA**: O doente é triado. Se houver médico disponível na sua especialidade, inicia-se a consulta imediatamente (gerando um futuro evento de SAÍDA). Caso contrário, o doente entra numa fila de espera FIFO (*First-In-First-Out*)[cite: 29].

- Se for **SAÍDA**: O médico fica livre. Se houver doentes na fila de espera dessa especialidade, o próximo doente é atendido imediatamente.

Para a implementação computacional, recorreu-se à biblioteca **numpy** para a geração de números pseudoaleatórios[cite: 165].

3 Arquitetura da Solução

A aplicação foi desenvolvida em Python, adotando uma arquitetura modular para separar a lógica de negócio, a interface gráfica e a manipulação de dados.

3.1 Descrição dos Módulos

main.py: Ponto de entrada da aplicação. Inicializa a interface gráfica.

interface.py: Implementa a GUI utilizando a biblioteca `FreeSimpleGUI`. É responsável pela recolha dos parâmetros de configuração (número de médicos, λ , etc.) e pela visualização dos resultados em tempo real (barra de progresso e *logs*). Inclui validações de *input* para garantir a robustez do sistema.

principal.py: Contém o ciclo principal da simulação (*Main Loop*). Orquestra a interação entre a lista de eventos, os recursos (médicos) e as filas de espera.

manipulacao.py: Módulo utilitário que encapsula as funções de baixo nível, como a gestão da lista de prioridade (`enqueue`, `dequeue`) e a geração de tempos aleatórios baseados nas distribuições estatísticas.

analisar.py e graficos.py: Módulos dedicados ao pós-processamento. O `analisar.py` processa o *dataset* de utentes (`personas.json`), enquanto o `graficos.py` utiliza a biblioteca `matplotlib` [cite: 166] para gerar visualizações gráficas da evolução das filas e ocupação.

3.1.1 Módulo main.py

Este ficheiro atua como o ponto de entrada da aplicação. A sua função é unicamente instanciar a interface gráfica, garantindo que o ciclo de vida da aplicação começa de forma controlada. Mantém o código limpo e segue o princípio de separação de responsabilidades.

3.1.2 Módulo interface.py

Este módulo é responsável por toda a interação com o utilizador. Desenvolvido com a biblioteca `FreeSimpleGUI`, implementa as seguintes funcionalidades críticas:

- **Validação de Dados:** Funções auxiliares (`validar_float` e `validar_int`) que atuam como barreiras de segurança ("Safety Car"), impedindo a inserção de valores inválidos (ex: taxas negativas ou letras em campos numéricos).
- **Janela de Configuração:** Permite definir parâmetros como o número de médicos, a taxa de chegada (λ) e o tipo de distribuição estatística para as consultas.
- **Feedback em Tempo Real:** Utiliza *callbacks* para atualizar uma barra de progresso e uma janela de *logs* durante a simulação, sem bloquear a execução do programa principal.
- **Persistência:** Inclui menus para exportar os resultados finais em formatos JSON e TXT.

3.1.3 Módulo principal.py

O ficheiro `principal.py` contém o "motor" da simulação, encapsulado na função `simula`. [cite_{start}Algicabaseia—senumciclowhilequeprocessaumalistadeeventosdiscretosenquanto

Processamento de Eventos: Distingue entre eventos de CHEGADA e SAÍDA (fim de consulta).

Gestão de Recursos: Verifica a disponibilidade dos médicos e atribui doentes ou coloca-os em fila de espera específica por especialidade (Cardiologia, Pneumologia, Clínica Geral).

Recolha de Métricas: Regista timestamps de entrada e saída para calcular posteriormente os tempos médios de espera e a ocupação dos consultórios.

3.1.4 Módulo manipulacao.py

Este módulo de baixo nível fornece as ferramentas matemáticas e estruturais necessárias ao motor de simulação. destaca-se pela implementação de:

- **Fila de Prioridade:** As funções `enqueue` e `dequeue` gerem a lista de eventos, garantindo que estes são sempre processados por ordem cronológica e não pela ordem de criação.

- **Geração Estocástica:** Utiliza a biblioteca `numpy` para gerar tempos aleatórios. Implementa a distribuição exponencial para os intervalos de chegada (Processo de Poisson) e permite selecionar entre distribuição normal, uniforme ou exponencial para a duração das consultas.
- **Lógica Biomédica:** A função `atribuir_especialidade` cruza dados do utente (idade, hábitos tabágicos) para determinar a especialidade médica adequada.

3.1.5 Módulo `analisar.py`

Focado na Ciência de Dados, este módulo carrega e trata o *dataset* de utentes (`peessoas.json`). Contém funções estatísticas que permitem caracterizar a população atendida, calculando médias de idades, distribuição por distritos e prevalência de fatores de risco (como tabagismo), enriquecendo o relatório final da simulação.

3.1.6 Módulo `graficos.py`

Responsável pela visualização de dados, recorrendo à biblioteca `matplotlib` integrada na interface gráfica. Gera gráficos de:

- Evolução temporal do tamanho das filas de espera (grafico de degraus/*step plot*).
- Taxa de ocupação dos médicos ao longo do tempo.
- Distribuição percentual de doentes por especialidade (diagrama circular/*pie chart*).

O módulo inclui verificações de segurança para evitar erros de execução caso os dados da simulação sejam insuficientes para gerar gráficos.

Referências

Anexos

Anexo A