Relatório do Projeto ATP 2025/2026

🏥 Simulação de uma Clínica Médica

Autores

- Afonso Pires a107161
- Andreia Cardoso a107279
- Beatriz Neiva a107241

Projeto desenvolvido no âmbito da UC **Algoritmos e Técnicas de Programação** Licenciatura em Engenharia Biomédica — Universidade do Minho

1. Introdução

Este projeto consiste no desenvolvimento de uma **simulação de eventos discretos** que modela o funcionamento de uma clínica médica, permitindo estudar o impacto de diferentes parâmetros operacionais no desempenho do sistema.

O sistema foi desenvolvido em Python e recorre a **processos estocásticos** para modelar:

- a chegada de doentes;
- o tempo de consulta;
- a utilização dos médicos;
- a evolução da fila de espera.

O principal objetivo é compreender como alterações nos parâmetros (ex.: taxa de chegada ou número de médicos) influenciam métricas como o tempo de espera, ocupação dos médicos e saturação do sistema.

---

2. Estrutura do Projeto

O projeto encontra-se organizado de forma modular, separando a interface gráfica, a simulação e a análise de resultados:

```
ATP_VSC/
├── menu_principal.py        # Interface gráfica principal
├── tab1.py                  # Simulação base da clínica
├── tab2.py                  # Cálculo de estatísticas
├── tab3.py                  # Visualização e simulação avançada
├── tab4.py                  # Estudos paramétricos
├── tab5.py                  # Relatórios e exportação
├── medicos.json             # Dados dos médicos
├── pessoas.json             # Dataset de doentes
├── equipamentos.json        # Recursos da clínica
├── relatorio1.json          # Resultados de simulações
├── relatorio3.json          # Resultados comparativos
```

Esta organização facilita a manutenção do código e a extensão futura do sistema.

---

3. Setup e Execução

3.1 Requisitos

- Python 3.10 ou superior
- Bibliotecas utilizadas:
  - `numpy`
  - `matplotlib`
  - `FreeSimpleGUI`
  - `json`

3.2 Execução

Para iniciar a aplicação basta executar:

```
python menu_principal.py
```

É aberta uma **interface gráfica** que permite selecionar diferentes módulos da aplicação, alterar parâmetros e visualizar resultados.

---

4. Funcionamento Geral do Sistema

4.1 Modelo de Simulação

O sistema baseia-se numa **simulação de eventos discretos**, onde o tempo avança de evento em evento e não de forma contínua.

Os principais eventos são:

- **Chegada de doente**
- **Saída de doente** (fim de consulta)

A lógica geral do sistema é a seguinte:

1. Um doente chega à clínica;
2. Se existir um médico livre, a consulta inicia-se de imediato;
3. Caso contrário, o doente entra na fila de espera;
4. Quando uma consulta termina, o médico fica livre e atende o próximo doente da fila;
5. O sistema regista estatísticas ao longo de toda a simulação.

---

5. Geração Estocástica

5.1 Chegada de Doentes

As chegadas seguem uma **distribuição de Poisson**, sendo o tempo entre chegadas gerado de forma exponencial.

Exemplo de código:

```
import numpy as np

def gera_intervalo_chegada(lambda_rate):
    return np.random.exponential(60 / lambda_rate)
```

Este modelo é adequado para representar chegadas aleatórias e independentes, comuns em sistemas reais de atendimento.

5.2 Tempo de Consulta

O tempo de consulta pode seguir diferentes distribuições, configuráveis pelo utilizador:

- Exponencial
- Normal
- Uniforme

Exemplo simplificado:

```
import random

def gera_tempo_consulta(tipo, media):
    if tipo == "exponential":
        return random.expovariate(1 / media)
    elif tipo == "normal":
        return max(1, random.gauss(media, media * 0.2))
    elif tipo == "uniform":
        return random.uniform(media * 0.5, media * 1.5)
```

---

6. Gestão da Fila de Espera

Quando todos os médicos estão ocupados, os doentes são colocados numa **fila FIFO (First-In First-Out)**.

Exemplo de lógica:

```
if medico_livre:
    iniciar_consulta(doente)
else:
    fila.append((doente, tempo_chegada))
```

Esta abordagem garante justiça no atendimento e reflete o funcionamento típico de uma clínica.

---

7. Parâmetros de Configuração

Os parâmetros são definidos no ficheiro `config.json`:

Parâmetro	Descrição
`lambda_rate`	Taxa de chegada de doentes (doentes/hora)
`num_doctors`	Número de médicos
`service_distribution`	Distribuição do tempo de consulta
`mean_service_time`	Tempo médio de consulta (min)
`simulation_time`	Duração da simulação (min)

A alteração destes valores permite simular diferentes cenários de carga do sistema.

---

8. Métricas Recolhidas

Durante a simulação são calculadas automaticamente:

- Tempo médio de espera dos doentes
- Tempo médio de consulta
- Tempo médio total na clínica
- Tamanho médio e máximo da fila
- Taxa de ocupação dos médicos
- Número total de doentes atendidos

Estas métricas são armazenadas em ficheiros JSON para análise posterior.

---

9. Resultados e Análise Gráfica

São gerados gráficos que mostram:

- Evolução do tamanho da fila ao longo do tempo;
- Evolução da ocupação dos médicos;
- Relação entre taxa de chegada e tamanho médio da fila;
- Comparação entre diferentes distribuições de tempo de consulta.

Estes gráficos permitem identificar situações de saturação e avaliar a eficiência da clínica.

---

10. Discussão dos Resultados

Os resultados obtidos demonstram que:

- O aumento da taxa de chegada provoca crescimento acentuado da fila de espera;

- A ocupação dos médicos aproxima-se rapidamente de 100% em cenários de elevada procura;
- Distribuições com maior variância no tempo de consulta aumentam a instabilidade do sistema.

Este comportamento está de acordo com sistemas reais de atendimento.

---

10. Interface Gráfica e Interação com o Utilizador

A aplicação dispõe de uma **interface gráfica desenvolvida com FreeSimpleGUI**, que constitui o principal meio de interação com o utilizador. Esta interface foi pensada para ser intuitiva e para separar claramente diferentes tipos de funcionalidades.

10.1 Menu Principal

O ficheiro `menu_principal.py` implementa o menu inicial da aplicação. A partir deste menu, o utilizador pode escolher entre diferentes áreas do sistema, nomeadamente:

- **Área de Simulação** – focada na execução e análise das simulações;
- **Área de Gestão Clínica** – dedicada à administração dos recursos da clínica.

Esta separação reflete dois perfis de utilização distintos e melhora a organização global da aplicação.

10.2 Interface da Simulação

Na área de simulação, o utilizador pode:

- Configurar os parâmetros da simulação (taxa de chegada, número de médicos, distribuição do tempo de consulta, duração da simulação);
- Iniciar e executar a simulação;
- Visualizar resultados estatísticos e gráficos associados;
- Comparar diferentes cenários de funcionamento.

Cada uma destas funcionalidades encontra-se distribuída pelas diferentes *tabs* (`tab1.py` a `tab4.py`), permitindo uma navegação clara entre:

- execução da simulação base;
- visualização da evolução da fila e ocupação dos médicos;
- estudos paramétricos com variação da taxa de chegada.

A interface permite assim que o utilizador explore o comportamento do sistema sem necessidade de alterar diretamente o código.

10.3 Interface de Gestão Clínica

Para além da simulação, foi desenvolvido um conjunto de **funcionalidades extra de gestão clínica**. Nesta área, o utilizador pode:

- **Adicionar, remover e consultar médicos**, recorrendo ao ficheiro `medicos.json`;
- **Adicionar e gerir doentes**, utilizando dados reais armazenados em `pessoas.json`;
- **Gerir equipamentos da clínica**, definidos em `equipamentos.json`;

- Atualizar dinamicamente os recursos disponíveis na clínica, que podem ser usados nas simulações.

Estas funcionalidades aproximam o sistema de uma aplicação real de gestão hospitalar e demonstram uma preocupação com a extensibilidade e realismo do modelo.

10.4 Separação de Perfis de Utilizador

A existência de dois grandes conjuntos de funcionalidades permite distinguir implicitamente dois perfis de utilizador:

- **Utilizador Analista**: interessado em executar simulações, estudar resultados e analisar métricas;
- **Utilizador Gestor**: focado na administração da clínica, gestão de recursos humanos e materiais.

Esta separação contribui para uma arquitetura mais limpa e facilita futuras extensões, como controlo de permissões ou autenticação de utilizadores.

---

11. Conclusão

O projeto permitiu aplicar conceitos fundamentais de:

- Processos estocásticos;
- Simulação de eventos discretos;
- Análise de desempenho de sistemas;
- Desenvolvimento de aplicações modulares com interface gráfica.

A simulação desenvolvida constitui uma ferramenta útil para apoio à decisão no planeamento de recursos clínicos.
