**Ambientes de desenvolvimento colaborativo**

UAlg – ISE – LESTI

**Documentação do Projeto de Gestão de Consultas Médicas**

90398 – Rodrigo Marques – a90398@ualg.pt

89825 – Rodrigo Fraga – [a89825@ualg.pt](mailto:a89825@ualg.pt)

# 

# **1. Introdução**

O presente projeto tem como finalidade o desenvolvimento de uma aplicação de gestão de consultas médicas, focada em facilitar a comunicação e a organização entre médicos e pacientes.

A aplicação permite o registo de utilizadores, a marcação de consultas, a gestão de agendas médicas e a visualização de eventos num calendário interativo.

Foi utilizada a linguagem de programação Python, uma base de dados SQLite e a framework NiceGUI garantindo assim uma solução leve e eficiente para contextos como clínicas de pequena e média dimensão.

# 

# 

# **2. Objetivo**

O principal objetivo deste projeto é criar uma plataforma intuitiva que permita:

* O registo de pacientes e médicos, atribuindo-lhes um identificador único (cartão).
* A marcação de consultas, de forma prática e rápida.
* A gestão da disponibilidade dos médicos através de agendamentos.
* A visualização de consultas futuras por parte dos pacientes e médicos.

# 

# **3. Tecnologias usadas**

Para o desenvolvimento deste projeto, foram utilizadas as seguintes tecnologias:

* **Python 3.10 ou superior**: Linguagem de programação principal utilizada para a construção da aplicação.
* **SQLite3**: Base de dados relacional leve, integrada diretamente no Python, usada para armazenar informações de pacientes, médicos, consultas e calendário.
* **NiceGUI**: Framework moderna para criação de interfaces web interativas utilizando Python.

Estas tecnologias foram escolhidas pela sua facilidade de integração, leveza e adequação ao desenvolvimento rápido de protótipos e aplicações de pequena a média escala.

# 

# 

# **4. Requisitos de Instalação**

* Python 3.10 ou superior
* Biblioteca nicegui 2.15
* Biblioteca sqlite3 (incluída no Python por defeito)

### **Instalação das dependências**

pip install nicegui

# 

# **5. Estrutura de Ficheiros**

| **Ficheiro** | **Descrição** |
| --- | --- |
| main.py | Inicializa a base de dados e popula com dados de teste. |
| features.py | Funções auxiliares para agendamento de consultas via terminal. |
| gui.py | Interface web com sistema de login, registo, painel de utilizador e calendário interativo. |
| data.db | Base de dados SQLite que armazena todos os dados. |
| data.sql | Script de criação das tabelas necessárias. |

# **6. Modelação da Base de Dados**

A estrutura da base de dados **data.db** foi concebida de forma a garantir a relação eficiente entre pacientes, médicos, consultas e o respetivo agendamento.  
 A modelação segue o esquema relacional tradicional, onde cada entidade é representada por uma tabela.

**Tabelas principais:**

* **pacientes**
  + id: Identificador único do paciente (chave primária).
  + nome: Nome do paciente.
  + email: Email de contacto.
  + cartao\_paciente: Número único de cartão (até 8 dígitos) usado para autenticação.
* **medicos**
  + id: Identificador único do médico (chave primária).
  + nome: Nome do médico.
  + especialidade: Especialidade médica.
  + cartao\_medico: Número único de cartão (até 8 dígitos) usado para autenticação.
* **consultas**
  + id: Identificador único da consulta (chave primária).
  + id\_paciente: Chave estrangeira referenciando o paciente.
  + id\_medico: Chave estrangeira referenciando o médico.
* **calendario**
  + id: Identificador único do evento de calendário (chave primária).
  + day: Dia da consulta.
  + month: Mês da consulta.
  + year: Ano da consulta.
  + id\_consulta: Chave estrangeira referenciando a consulta.

**Relacionamentos:**

* Um paciente pode ter várias consultas.
* Um médico pode atender várias consultas.
* Cada consulta está associada a uma única entrada no calendário.

Esta estrutura permite facilmente consultar, agendar, reagendar e visualizar todas as interações entre médicos e pacientes.

# **7. Iniciar a Aplicação**

1. Certifique-se que o ambiente está ativado e todas as dependências instaladas.
2. Execute o seguinte comando para preparar a base de dados:

python main.py

1. Inicie o servidor NiceGUI:

python gui.py

1. A aplicação estará disponível por defeito em http://localhost:8080

# 

# **8. Funcionalidades do Calendário**

## **8.1 Integração com FullCalendar**

O sistema utiliza a biblioteca externa [FullCalendar](https://fullcalendar.io/) para uma visualização dinâmica dos eventos de consulta.

### **Destaques:**

* Dias com consultas são marcados com eventos visuais (bolas verdes).
* Ao clicar num evento, é apresentado um alert com os detalhes da consulta (médico, especialidade ou paciente).

## **8.2 Fonte dos Eventos**

Os eventos do calendário são carregados diretamente da base de dados data.db, a partir das tabelas consultas, calendario, pacientes e medicos.

## **8.3 Comportamento**

* Pacientes veem as suas consultas.
* Médicos veem as consultas dos seus pacientes.

# **9. Detalhes Técnicos Importantes**

* **Segurança:** Não existem atualmente mecanismos de encriptação de passwords. Recomenda-se implementar hashing (ex: bcrypt).
* **Validação:** A validação dos dados inseridos pelos utilizadores é feita principalmente no lado do cliente.
* **Compatibilidade:** O projeto foi desenhado para funcionar em navegadores modernos (Chrome, Firefox, Edge).
* **Performance:** Ideal para pequenas clínicas ou protótipos. Em ambientes de produção, poderá ser necessário migrar para bases de dados mais robustas.

# **10. Conclusão**

* O projeto de Gestão de Consultas Médicas apresentado cumpre os requisitos propostos, oferecendo uma solução funcional para o agendamento e organização de consultas entre médicos e pacientes.  
   Através da integração de tecnologias modernas como NiceGUI e FullCalendar, é possível garantir uma experiência de utilização agradável e intuitiva.
* A utilização de SQLite como sistema de base de dados permitiu uma implementação rápida e eficiente, adequada a pequenas clínicas ou a projetos de prototipagem.

**FIM DA DOCUMENTAÇÃO**

**GITHUB**

https://github.com/rodft2/adc-tp2