Uma imagem com texto, Tipo de letra, Gráficos, logótipo

Descrição gerada automaticamente

**Fundamentos, Administração e**

**Gestão de Base da Dados**

CTeSP em Cloud e Cibersegurança

Projeto nº 2:

**Sistema de Gestão Hospitalar**





Realizado por:

Amira Babkir nº2024126219

Vicente Gonçalves nº2024122708

Docente: Prof. Fábio Sampaio

ESTSetúbal/IPS, ano letivo 2024/2025

**Índice**

[1. Introdução 3](#_Toc200038522)

[2. Requisitos e Funcionalidades 3](#_Toc200038523)

[2.1. Requisitos funcionais 3](#_Toc200038524)

[2.2. Requisitos não funcionais 6](#_Toc200038525)

[3. Estrutura da base de dados 8](#_Toc200038526)

[3.1. Tabelas 8](#_Toc200038527)

[3.2. Relações 11](#_Toc200038528)

[3.3. Diagrama Entidade-Relacionamento (ER) 11](#_Toc200038529)

[4. Segurança e acesso 12](#_Toc200038530)

[4.1. Autenticação de Utilizador 12](#_Toc200038531)

[4.2. Controle de Permissões 13](#_Toc200038532)

[4.3. Proteção de Dados Sensíveis 13](#_Toc200038533)

[4.4. Registos de Log 13](#_Toc200038534)

[5. Interface do utilizador 14](#_Toc200038535)

[5.1. Menu Principal 14](#_Toc200038536)

[5.2. Navegação 14](#_Toc200038537)

[5.3. Tipos de Utilizador 14](#_Toc200038538)

[5.4. Mensagens e Feedback 14](#_Toc200038539)

[6. Implementação técnica 14](#_Toc200038540)

[6.1. Estrutura do Código 14](#_Toc200038541)

[6.2. Bibliotecas Utilizadas 15](#_Toc200038542)

[6.3. Criação e Utilização da Base de Dados 15](#_Toc200038543)

[6.4. Controle de Acesso 15](#_Toc200038544)

[6.5. Encriptação de Dados sensíveis 15](#_Toc200038545)

[6.6. Registo de Logs 16](#_Toc200038546)

[7. Exemplos de testes e resultados? 16](#_Toc200038547)

[8. Conclusão 16](#_Toc200038548)

[Bibliografia 16](#_Toc200038549)

# Introdução

A informatização da gestão hospitalar é fundamental para garantir eficiência, organização e segurança no tratamento de dados clínicos e administrativos. Este projeto tem como objetivo o desenvolvimento de um sistema de gestão hospitalar funcional e seguro, utilizando Python e a base de dados SQLite3, com interface em consola.

O tema foi proposto pelo professor da UC e tem como finalidade aplicar os conhecimentos teóricos em um contexto prático. O sistema abrange funcionalidades como a gestão de pacientes, médicos, consultas, tratamentos, prescrições e funcionários, além de mecanismos de segurança como o controle de acesso por tipo de usuário e o registo de ações por meio de um log.

O projeto permite consolidar os conteúdos abordados em aula e desenvolver competências práticas como o trabalho em grupo, o raciocínio lógico, a organização do código e a documentação técnica. Essas habilidades são fundamentais na área de tecnologia da informação, especialmente em contextos que lidam com dados sensíveis, como os da área da saúde.

# Requisitos e Funcionalidades

## Requisitos funcionais

O sistema de gestão hospitalar implementado oferece as seguintes funcionalidades, todas acessíveis por meio de uma interface em consola com menus dinâmicos conforme o tipo de utilizador autenticado:

1. Autenticação e Controle de Acesso

* Login obrigatório com nome de utilizador e palavra-passe (sem exibição no terminal).
* Diferenciação automática de permissões por tipo de utilizador:
  + Administrador: acesso total ao sistema.
  + Médico: acesso restrito às suas consultas, prescrições e pacientes.
  + Enfermeiro: acesso às suas informações e às consultas dos pacientes.
  + Paciente: pode visualizar e modificar apenas os próprios dados, tratamentos e consultas.

2. Gestão de Pacientes

* Adicionar novo paciente, incluindo:
  + Nome, data de nascimento, gênero, contato e prontuário médico.
  + O prontuário é armazenado cifrado no banco de dados.
* Buscar pacientes por nome ou por contato.
* Editar dados (ex.: atualizar contato).
* Excluir paciente.
* Visualizar lista de pacientes (apenas para admins e médicos).

3. Gestão de médicos

* Adicionar médico com nome, especialidade e contato.
* Buscar médicos por nome.
* Editar dados do médico (nome, especialidade, contato).
* Excluir médico.
* Visualizar lista de médicos.

4. Gestão de enfermeiros

* Adicionar enfermeiro com nome e contato.
* Buscar enfermeiros por nome.
* Editar dados do enfermeiro.
* Excluir enfermeiro.
* Visualizar lista de enfermeiros.

5. Consultas

* Agendar novas consultas entre pacientes e médicos, informando data e hora.
* Visualizar consultas:
  + Por data específica.
  + Por intervalo de datas (semana, mês, etc.).
* Visualização adaptada ao tipo de utilizador:
  + Médico: apenas suas consultas.
  + Paciente: apenas suas consultas.
  + Enfermeiro: todas as consultas dos pacientes.

6. Tratamentos

* Adicionar tratamentos para pacientes com limite de 1024 caracteres.
* Visualizar todos os tratamentos de um paciente, com data e nome do médico que prescreveu, via JOIN com a tabela de prescrições.
* Observação: a relação entre tratamento e prescrição é feita por data e paciente, e não por chave estrangeira direta — funcional, mas limitada.

7. Prescrições

* Adicionar nova prescrição (médico, paciente, medicamento e data).
* Visualizar prescrições:
  + Por médico e período de tempo.
  + Por médico, período de tempo e faixa etária dos pacientes.
* Médicos só veem suas prescrições; admins podem consultar qualquer uma.

8. Gestão do próprio perfil (paciente/enfermeiro)

* Ver dados pessoais.
* Modificar o próprio contato.
* Paciente: visualizar suas consultas, tratamentos e prontuário (decifrado).
* Enfermeiro: visualizar suas consultas e editar dados.

9. Logs de acesso

* Registro automático de ações no sistema com:
  + ID do utilizador, ação executada, data/hora e status (sucesso/falha).
* Visualização do log:
  + Por período ou por utilizador.
  + Acesso exclusivo ao administrador.

10. Visualização de tabelas

* Administrador pode visualizar o conteúdo completo de qualquer tabela da base de dados.

11. Segurança e Proteção de Dados

* O prontuário médico é armazenado cifrado na base de dados (função ‘cifrar()’).
* Acesso às funcionalidades controlado por tipo de utilizador.
* Senhas são armazenadas em texto simples — ponto a melhorar para versões futuras.
* Acesso ao log restrito ao perfil administrador.
* Interface segura com ‘getpass’.

12. Interface e Navegação

* Todo o sistema é operado via consola (terminal).
* Menus interativos e dinâmicos por tipo de utilizador.
* Mensagens claras de erro, sucesso e instruções ao utilizador.
* Função ‘espera()’ garante leitura antes de limpar a tela.

## Requisitos não funcionais

Aqui pretende-se escrever o que funciona no programa (não em termos de funcionalidades), segue-se um exemplo em seguida: EDITAR TEXTO

* Persistência de Dados: O sistema utiliza a base de dados para armazenar e gerir todas as informações sobre pacientes, médicos, consultas, prescrições, tratamentos, utilizadores e registos de acesso.  
  Todos os dados são mantidos entre diferentes sessões de uso, garantindo assim persistência e integridade sem a necessidade de servidor externo.
* Interface: A aplicação foi desenvolvida para funcionar em consola (linha de comandos), com menus organizados e interação orientada por texto.  
  A biblioteca ‘rich’ foi utilizada para melhorar a estética e legibilidade da interface, com cores e tabelas formatadas, proporcionando uma melhor experiência de utilizador.
* Segurança: O sistema implementa:
  + Autenticação obrigatória com nome de utilizador e palavra-passe (oculta no terminal via ‘getpass’).
  + Controle de permissões por tipo de utilizador (admin, médico, enfermeiro e paciente).
  + Armazenamento cifrado do prontuário médico usando uma função de reversão como prova de conceito (‘cifrar()’).
  + Registo de logs de todas as ações realizadas no sistema, incluindo utilizador, ação, data/hora e status da operação.
  + Restrição do acesso ao log de ações apenas para administradores.
* Portabilidade: O sistema foi desenvolvido em Python, utilizando apenas bibliotecas padrão (sqlite3, os, getpass, datetime, sys, etc.) e a biblioteca externa rich (de fácil instalação).  
  Pode ser executado em qualquer sistema que tenha Python instalado, com dependências do ficheiro ‘requirements.txt’ que é de fácil instalação. Este é instalado com o comando abaixo

dentro da pasta onde se encontra o ficheiro.

pip install -m requirements.txt

* Desempenho: Durante os testes, o sistema apresentou desempenho satisfatório em todas as operações executadas como por exemplo de inserção, edição, consulta e exclusão de dados.  
  Tirando o tempo de espera inicial do programa, o tempo de resposta manteve-se praticamente instantâneo mesmo com várias dezenas de registos, validando a adequação da escolha por SQLite para ambientes de pequeno porte.
* Organização e Manutenção: O código foi organizado da seguinte maneira:
  + Main.exe: programa executável com interface, menus e interação com o utilizador.
  + SqliteCommands.py: contem funções de acesso a base de dados e logica do programa.
  + dbHospital.bd: Ficheiro principal da base de dados usado. Ele contém todas as tabelas, dados inseridos, relações e índices.
  + Dbhospital.sqbpro: Ficheiro de projeto que armazena:
    - Preferencias de visualização
    - Histórico de queries executadas
    - Guias abertas

# Estrutura da base de dados

O sistema utiliza o SQLite3 como mecanismo de base de dados relacional, com **seis tabelas principais** interligadas, de acordo com os requisitos do projeto. Abaixo são descritas as tabelas, os campos e os relacionamentos definidos.

## Tabelas

Abaixo estão descritas as tabelas principais com os respetivos campos e tipos de dados.

**Paciente**

|  |  |
| --- | --- |
| id\_paciente | Intenger, PK |
| nome | Text |
| data\_nascimento | Text |
| genero | Text |
| contacto | Text |
| prontuario | Text |

**Medico**

|  |  |
| --- | --- |
| id\_medico | Integer, PK |
| nome | Text |
| especialidade | Text |
| contacto | Text |

**Enfermeiro**

|  |  |
| --- | --- |
| id\_enfermeiro | Integer, PK |
| Nome | Text |
| contacto | Text |

**Consulta**

|  |  |
| --- | --- |
| id\_consulta | Inetger; PK |
| id\_paciente | Integer, FK |
| id\_medico | Integer, FK |
| data\_consulta | Text |
| status | Text |

**Tratamento**

|  |  |
| --- | --- |
| id\_tratamento | Integer, PK |
| id\_paciente | Integer, FK |
| descrição | Text |
| data\_tratamento | Text |

**Prescricao**

|  |  |
| --- | --- |
| id\_prescrição | Integer, PK |
| id\_paciente | Integer, FK |
| id\_medico | Integer, FK |
| nome\_medicamento | Text |
| data\_prescricao | Text |

**Log\_acesso**

|  |  |
| --- | --- |
| id\_log | Integer, PK |
| id\_user | Integer, FK |
| acao\_executada | Text |
| data | Datatime |
| status | Text |

**Users**

|  |  |
| --- | --- |
| id\_user | Integer, PK |
| login | Text |
| senha | Text |
| tipo\_user | Text |

## Relações

As principais relações da base de dados são:

|  |  |  |
| --- | --- | --- |
| **Tabela** | **Relação** | **Tabela** |
| Users | 1:N | Log\_acesso |
| Paciente | 1:N | Consulta |
| Paciente | 1:N | Tratamento |
| Paciente | 1:N | Prescrição |
| Medico | 1:N | Prescrição |
| Medico | 1:N | Consulta |

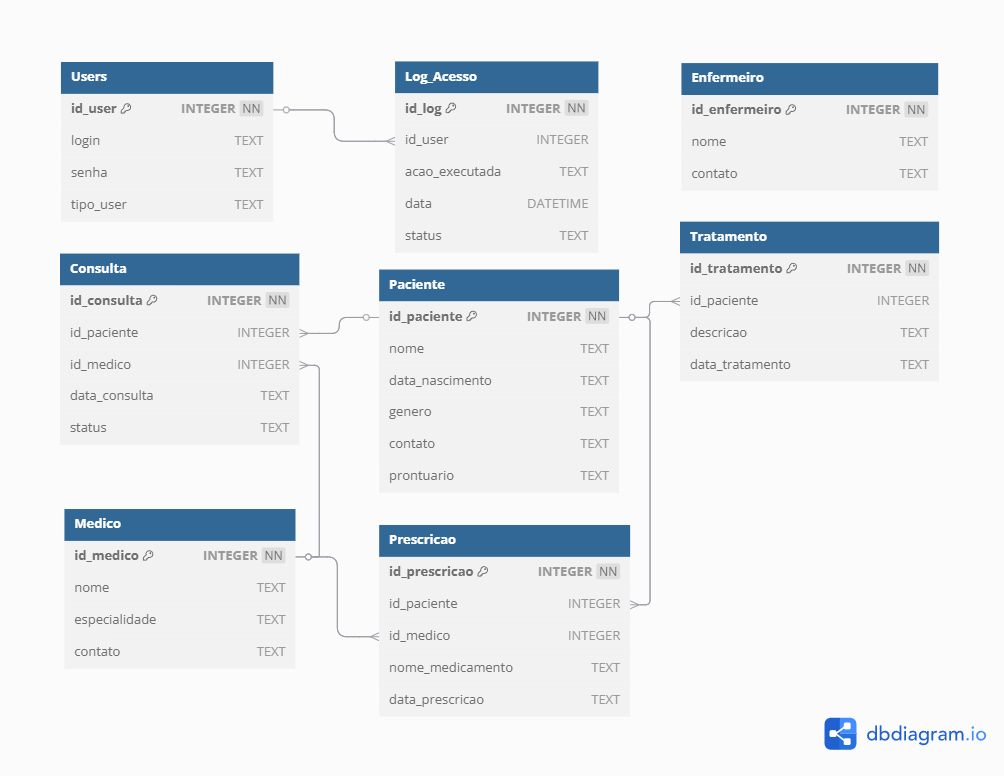
## Diagrama Entidade-Relacionamento (ER)

Na imagem abaixo encontra-se o Diagrama ER que representa a estrutura da base de dados utilizada no sistema.

Este diagrama inclui:

* As tabelas principais como entidades.
* As chaves primárias (identificadas por ícone de chave).
* As chaves estrangeiras e as ligações entre tabelas, representando os relacionamentos definidos no modelo lógico.
* Os campos e os tipos de dados de cada entidade.

O diagrama foi gerado com a ferramenta online **dbdiagram.io**, a partir do código que descreve a estrutura e as relações da base de dados.



# Segurança e acesso

A segurança da informação é um elemento necessário no contexto hospitalar onde dados sensíveis como históricos clínicos, dados pessoais e prescrições são manipulados. Para garantir a confidencialidade, integridade e controle de acesso, forma implementadas as seguintes medidas no sistema:

## Autenticação de Utilizador

**EXEMPLO ABAIXO:**

O sistema exige autenticação por nome de utilizador e palavra-passe para qualquer operação.

Existem diferentes tipos de usuário, cada um com permissões específicas:

Administrador: acesso total ao sistema, incluindo visualização do log de acessos e alteração de dados de qualquer usuário.

Funcionários (médicos/enfermeiros): acesso limitado às funcionalidades relacionadas às suas funções.

Pacientes: acesso restrito aos seus próprios dados, podendo visualizar e editar informações pessoais.

## Controle de Permissões

**EXEMPLO ABAIXO:**

Cada funcionalidade do menu é validada com base no tipo de usuário autenticado.

Apenas o administrador pode:

Ver o log de acessos

Modificar dados de outros usuários

Visualizar todas as tabelas

Pacientes podem apenas:

Ver suas consultas, prescrições e tratamentos

Alterar o próprio contato

## Proteção de Dados Sensíveis

**EXEMPLO ABAIXO:**

O histórico médico dos pacientes é armazenado de forma cifrada na base de dados, de modo a garantir a confidencialidade em caso de acesso indevido ao banco.

As senhas dos usuários também são armazenadas de forma cifrada.

## Registos de Log

**EXEMPLO ABAIXO:**

Todas as ações relevantes no sistema são registadas em um arquivo de log, contendo:

ID do usuário

Ação executada

Data e hora

Status da operação (sucesso, erro, etc.)

Esse log só pode ser consultado por usuários com perfil de administrador.

# Interface do utilizador

## Menu Principal

COLOCAR:

Lista de principais opções do utilizador após o login

Colocar que o menu é dinâmico variando conforme o tipo de utilizador

## Navegação

Explicar como o utilizador interage com o menu

Mencionar a validação de entradas e mensagens de erro

## Tipos de Utilizador

Resumir as permições de cada perfil – admin, funcionário (medico/enfermeiro) e paciente

## Mensagens e Feedback

Informar que o sistema dá respostas claras ao utilizador como confirmações e erro.

# Implementação técnica

## Estrutura do Código

Como o programa foi desenvolvido – num único python ou em módulos;

As funcionalidades e como foi organizada, se foi em funções separadas para melhor legibilidade e reutilização.

Se o código esta comentado e os blocos de tratamento de erros para evitar falhas

## Bibliotecas Utilizadas

Bibliotecas tais como

sqlite3: para conexão, criação de tabelas, inserção e consulta de dados.

getpass: para entrada segura de palavras-passe, sem exibição no terminal.

datetime: para trabalhar com datas (consultas, prescrições, logs).

hashlib ou cryptography (caso tenham usado): para cifragem de palavras-passe ou dados sensíveis.

## Criação e Utilização da Base de Dados

Se ainda não existir a base de dados é criada automaticamente.

Que comandos foram utilizados e porque

A integridade foi respeitada com fk entre as tabelas

## Controle de Acesso

Função de login que identifica o tipo de utilizador com base no login

Depois do login o sistema esta limitado a cada privilegio de cada user

## Encriptação de Dados sensíveis

Histórico medical dos pacientes foi encriptado antes de ser armazenado

As palavras passes dos utilizadores também são armazenadas para evitar o armazenamento em texto plano

## Registo de Logs

Ações importantes realizadas no sistema são registadas numa tabela de log, com identificação do usuário, tipo de ação, data/hora e status (sucesso ou erro).

Esse log é acessível apenas ao administrador.

# Exemplos de testes e resultados?

Para garantir a funcionalidade do sistema, foram realizados alguns testes. Como seguido de exemplo:

Teste a) adição de paciente

Teste b – agendamento de consulta

Teste c – visualização de prescrições por período e medico

Teste d – teste de login com acesso restrito

# Conclusão

NÃO ESQUECER DE ESCREVER CONCLUSÃO

# Bibliografia

[1] Guia do Laboratório 1 - Configurações Básicas de Routers, *site Moodle do IPS*, consultado em abril de 2023. Link: “https://moodle.ips.pt/2223/mod/resource” “/view.php?id=79165”.