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:

* 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:
    - Preferências 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 |

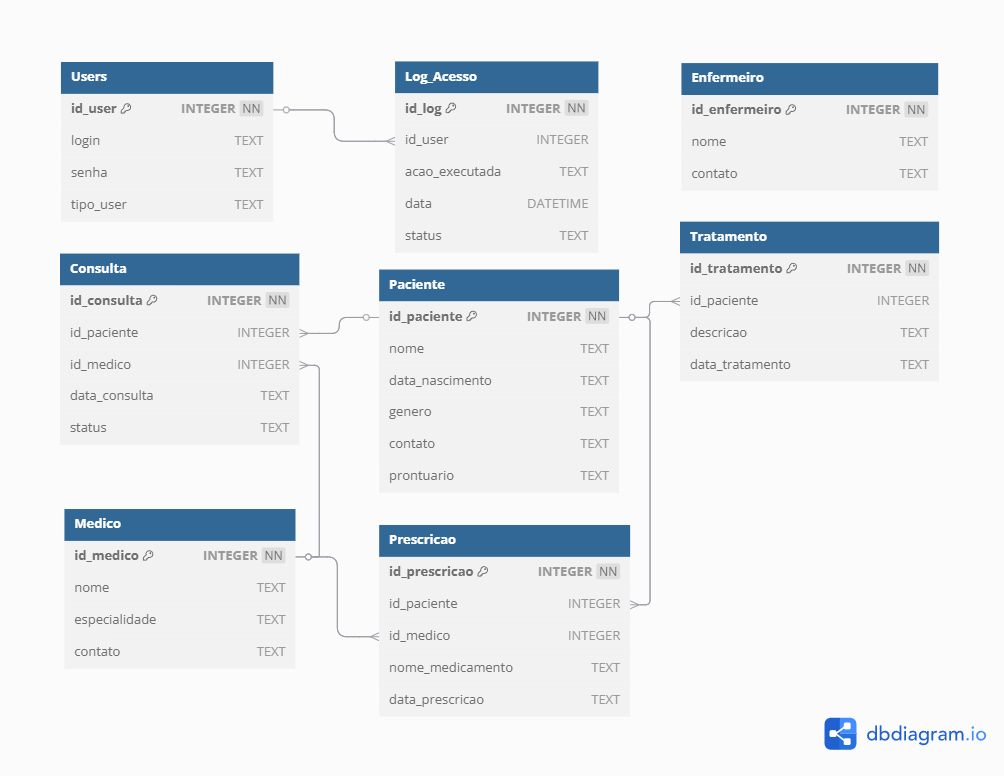
## Diagrama Entidade-Relacionamento (ER)

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 guardadas e manipulados. Para garantir a confidencialidade, integridade e controle de acesso, forma implementadas as seguintes de segurança medidas no sistema:

## Autenticação de Utilizador

O sistema exige **autenticação obrigatória** por nome de utilizador e palavra-passe antes de qualquer operação.

* A entrada da palavra-passe é feita com o módulo ‘*getpass’* impedidndo que seja exibida no terminal.
* Os utilizadores estão registados na tabela ‘*users’* com os campos: ‘*login’*, ‘*senha’*, ‘*tipo\_user'*.

Existem diferentes tipos de utilizadores, cada um com permissões específicas definidas:

|  |  |
| --- | --- |
| Tipo de Utilizador | Permissões |
| Administrador | Acesso total. Pode adicionar, editar e excluir qualquer registo. Visualiza logs, todas as tabelas e dados. |
| Medico | Acesso apenas às suas consultas, prescrições e pacientes. Não pode editar outros utilizadores. |
| Enfermeiro | Acesso aos seus dados pessoais e às consultas dos pacientes. Sem permissões de edição global. |
| Paciente | Acesso apenas aos próprios dados: pode visualizar e editar contacto, ver tratamentos e consultas. |

## Controle de Permissões

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

Permissões específicas:

Apenas o administrador pode:

* Visualizar o log de acessos de todos os utilizadores
* Modificar os contactos de qualquer utilizador (paciente, medico ou enfermeiro)
* Visualizar o conteúdo completo de qualquer tabela
* Excluir registos (paciente, medico ou enfermeiro)

Os médicos podem:

* Visualizar os seus próprios pacientes.
* Visualizar e filtrar as suas prescrições.
* Ver apenas as consultas onde são o médico responsável.

Enfermeiros podem:

* Visualizar os seus próprios dados.
* Editar o próprio contacto.
* Visualizar as consultas dos pacientes.

Pacientes podem apenas:

* Ver as suas próprias consultas.
* Ver os seus tratamentos.
* Alterar o seu contacto.
* Visualizar os seus próprios dados (incluindo o prontuário, já decifrado).

## Proteção de Dados Sensíveis

Para garantir a confidencialidade dos dados pessoais e clínicos dos pacientes:

* O prontuário médico é armazenado de **forma cifrada na base de dados**, utilizando uma técnica de reversão de texto (texto[::-1]). Embora seja uma cifra simples, cumpre o requisito de impedir leitura direta dos dados sensíveis em caso de acesso não autorizado ao ficheiro ‘*.db’*.
* A interface impede que utilizadores não autorizados acedam a dados de outros pacientes ou médicos.
* As senhas dos utilizadores são, nesta versão, armazenadas em texto plano na base de dados (‘*Users.senha’*).

## Registos de Log

O sistema mantém um **registo automático (log)** de todas as ações relevantes realizadas pelos utilizadores, com o objetivo de rastrear atividades e garantir responsabilidade.

Cada registo contém:

* ID do utilizador
* Ação executada
* Data e hora (timestamp automático)
* Status da operação (ex. sucesso ou falha)

Os registos são armazenados na tabela *‘Log\_acesso’*

As regras de acesso ao registo de logs são;

* Apenas o administrador pode visualizar o log
* A visualização pode ser feita:
  + Por período (data de início e de fim)
  + Por utilizador (pelo ID)

# Interface do utilizador

O sistema foi desenvolvido para funcionar inteiramente por interação em consola (por linha de comandos), utilizando menus interativos e mesagens informativas para guiar a navegação. Posteriormente, a biblioteca rich foi integrada de modo a melhorar a estética e legibilidade da interface com cores e painéis, tornando a experiência do utilizador melhor.

## Menu Principal

Após o login, o utilizador tem acesso ao menu principal que apresenta várias opções, dependendo do tipo de utilizador. O menu é dinâmico adaptando-se automaticamente ao utilizador autenticado.

## Navegação

A navegação no sistema é feita por entrada de números (opções do menu). O utilizador interage com o sistema de forma sequencial:

1. Efetua o login com user e senha
2. Visualiza o menu adaptado ao seu perfil
3. Seleciona uma opção escrevendo o número que pretende
4. Recebe as mensagens de confirmação, erro ou instrução
5. O sistema pausa com **‘Clique ENTER para continuar’**… antes de limpar a consola e voltar novamente ao menu.

## Tipos de Utilizador

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

INSERIR FOTO DO MENU DE CADA UTILIZADOR

ADMIN

MEDICO

ENFERMEIRO

PACIENTE

## Mensagens e Feedback

O sistema fornece respostas visuais claras e imediatas ao utilizador após cada ação, utilizando cores e facilitando a navegação e o entendimento.

Confirmações de sucesso

ATTACH IMAGE

Erros e exceções

ATTACH IMAGE

Validações de entrada

* O sistema inclui validações básicas ao solicitar informações como **IDs, datas, nomes ou contatos**.
* Embora **não exista validação rigorosa de formato**, entradas mal formatadas (ex: datas inválidas) são **geridas com try/except**, evitando falhas críticas no programa

ATTACH IMAGE

Experiência fluida

Após cada operação, o sistema pausa com a mensagem:

ATTACH IMAGE – clique enter para continuar

Isto garante tempo para ler as mensagens antes de o ecrã ser limpo e o menu reapresentado.

A interface evita repetições e mantem o utilizador sempre informado do que esta a acontecer.

# Implementação técnica

## Organização do Código

O sistema foi desenvolvido em **Python**, com uma separação clara entre a interface do utilizador e a lógica de acesso à base de dados:

* **main.py** (compilado como main.exe): contém a interface de consola, autenticação, menus dinâmicos e navegação. Cada funcionalidade do sistema é ativada através de funções definidas.
* **sqlitecommands.py**: contém as funções responsáveis por interagir com a base de dados (inserções, consultas, atualizações, exclusões), além das funções de registo de log.

A separação do código em módulos distintos foi aplicada como boa prática de programação especialmente recomendada para projetos desta dimensão. Esta abordagem aumenta a clareza, facilita alterações e manutenções futuras e ajuda a evitar a duplicação de código. Além disso, funções críticas são protegidas com blocos **try/except** para prevenir falhas inesperadas durante a execução.

## Bibliotecas Utilizadas

Foram utilizadas principalmente bibliotecas padrão do Python, com uma única dependência externa:

* sqlite3: conexão e manipulação da base de dados relacional.
* getpass: entrada segura de palavras-passe (sem exibição no terminal).
* datetime: gestão de datas em consultas, prescrições e logs.
* os e sys: gestão de caminhos e compatibilidade com diferentes ambientes.
* typing: uso de tipos para maior clareza no código.
* rich: biblioteca externa usada para melhorar a estética da interface com cores, tabelas e mensagens visuais.

A escolha de bibliotecas simples e eficientes não só contribui para a portabilidade e leveza do sistema, como também demonstra a aplicação consciente dos recursos da linguagem, reforçando boas práticas de desenvolvimento e garantindo que o sistema esteja de acordo com os objetivos estabelecidos.

## Justificação Técnica das Decisões

Base de dados SQLite3: escolhida por ter sido trabalhada em aula, por ser leve, integrada ao Python e não requerer instalação de servidor. Ideal para sistemas locais, trabalhos académicos e desenvolvimento em equipa.

Separação de ficheiros (main.py e sqlitecommands.py): permite reutilização de funções, facilita testes e manutenção, e segue boas práticas de programação modular.

Interface em consola com ‘rich’: mantém a simplicidade do terminal sem dependências de interfaces gráficos complexas, mas melhora significativamente a experiência visual do utilizador com cores e formatação.

Compilação com PyInstaller: o script principal foi convertido em main.exe, facilitando a distribuição e execução do programa em sistemas que não tenham Python instalado, garantindo acessibilidade.

Tratamento de dados sensíveis: apesar de não utilizar cifragem forte, o uso de reversão do texto no prontuário demonstra preocupação com a privacidade dos dados e cumpre com os requisitos mínimos de segurança exigidos no contexto do projeto.

Registo de logs: foi implementado um sistema de logging completo que regista todas as ações relevantes. Esta funcionalidade garante rastreabilidade, transparência e responsabilidade – algo particularmente relevante em contextos onde a integridade e confidencialidade da informação são **essenciais**.

## Estrutura da entrega

A estrutura final da entrega foi organizada da seguinte forma. Este arranjo facilita a portabilidade, utilização e verificação do projeto.

— /ProjetoFinal

— db/

— dbHospital.db → Ficheiro principal da base de dados

— dbHospital.sqbpro → Projeto auxiliar

— sqlitecommands.py → Lógica e operações da base de dados

— main.exe → Programa compilado para execução

— requirements.txt → Para quem desejar instalar manualmente

# Exemplos de testes e resultados

A seguir, apresentam-se testes realizados ao sistema, com destaque para os casos abaixo.

Testes:

Login com credenciais validas – deve dar

Login com credenciais invalidas – deve falhar

Log depois de uma entrada bem sucedida

Acão falhada e o respetivo log

Medico a tentar ver o log acesso, o qual não tem permissão

Paciente tenta ver o eu prontuário medico (exibe cifrado?)

Verificação de cifragem no .bd? os dados devem estar cifrados

# Conclusão

NÃO ESQUECER DE ESCREVER CONCLUSÃO

# Bibliografia

[1] Proposta projeto 2 - *Sistema de Gestão Hospitalar utilizando Python e SQLite3*. Site Moodle do IPS, consultado em maio de 2025.

Link: “https://moodle.ips.pt/2425/pluginfile.php/185475/mod\_resource/content/1/PROJETO2-FINAL-GESTAO-HOSP-PROPOSTA.pdf”.

[2] Python Software Foundation.*sqlite3 – interface para base de dados sqlite*.

Documentação oficial. Consultado em maio de 2025.

Link: “https://docs.python.org/3/library/sqlite3.html”

[3] Python Software Foundation. *getpass – entrada de senha Segura*.

Documentação oficial. Consultado em maio 2025.

Link: “https://docs.python.org/3/library/getpass.html”

[4] Will McGugan. *Rich — Biblioteca Python para formatação de terminal*.

Consultado em maio de 2025.

Link: https://rich.readthedocs.io/

[5] dbdiagram.io — *Ferramenta online para criação de diagramas ER*.

Consultado em maio de 2025.

Link: “https://dbdiagram.io”

[6] PyInstaller — *Compilador de scripts Python para executáveis standalone*.

Consultado em junho de 2025.

Link: “https://pyinstaller.org/”