Sistema de chamados (Python + SQL)

Introdução

Este projeto é um sistema de help desk para gerenciamento de chamados de suporte técnico, desenvolvido em Python e utilizando SQL Server como banco de dados. Ele permite registrar, consultar, atualizar e excluir chamados de forma simples e eficiente.

Visão Geral do Projeto

O sistema é composto por uma interface de linha de comando (CLI) que interage com o usuário, uma camada de lógica de negócios e uma camada de acesso ao banco de dados.

Funcionalidades

- **C - Criar novos chamados:** Cadastro de solicitações de suporte.
- **R - Listar todos os chamados:** Visualização de todos os chamados registrados.
- **U - Atualizar status de chamados:** Alteração do status para "Em Andamento" ou "Resolvido".
- **D - Deletar chamados:** Remoção de chamados do sistema.
- *** - Buscar chamados por ID:** Consulta detalhada de um chamado específico.

Estrutura do Projeto

```
ChamadosELT/
│
├── main.py          # Interface principal (CLI)
├── manager.py       # Lógica de negócios
├── database.py      # Operações de banco de dados
```

```
├── requirements.txt  # Dependências Python
├── .env              # Configurações de ambiente (não versionado)
└── README.md         # Documentação
```

Requisitos

- Python 3.x
- SQL Server
- Bibliotecas Python:
  - pyodbc
  - python-dotenv
  - datetime

Instale as dependências com:

```
pip install -r requirements.txt
```

Configuração do Ambiente

1. **Clone o repositório:**

   ```
   git clone <https://github.com/seu-usuario/ChamadosELT.git>
   cd ChamadosELT
   ```

2. **Configure o arquivo `.env` com os dados do seu banco:**

   ```
   DB_SERVER=seu_servidor
   DB_USER=seu_usuario
   DB_PASSWORD=sua_senha
   DB_NAME=ChamadosELT
   ```

3. **Crie a tabela no SQL Server:**

   ```
   CREATE TABLE Chamados (
       ID INT IDENTITY(1,1) PRIMARY KEY,
   ```

```
    Titulo VARCHAR(100) NOT NULL,
    Descricao TEXT NOT NULL,
    Solicitante VARCHAR(50) NOT NULL,
    Status VARCHAR(20) DEFAULT 'Aberto',
    DataCriacao DATETIME DEFAULT GETDATE()
);
```

Banco de Dados

O sistema utiliza uma tabela chamada `Chamados` com os seguintes campos:

Campo	Tipo	Descrição
ID	int (auto-increment)	Identificador único do chamado
Titulo	varchar(100)	Título do chamado
Descricao	text	Descrição detalhada
Solicitante	varchar(50)	Nome do solicitante
Status	varchar(20)	Status do chamado
DataCriacao	datetime	Data/hora de criação

Guia de Uso

1. **Execute o sistema:**

```
python main.py
```

2. **Menu principal:**

```
1. Abrir um novo chamado
2. Visualizar todos os chamados
3. Buscar um chamado por ID
4. Atualizar Status de um Chamado
5. Deletar um chamado
6. Sair
```

3. **Fluxo de operações:**

- Siga as instruções do menu para criar, consultar, atualizar ou excluir chamados.
- Para atualizar o status, utilize apenas os valores permitidos: "Em Andamento" ou "Resolvido".
- Para deletar, confirme a operação quando solicitado.

Arquitetura da Aplicação

- **main.py:** Responsável pela interação com o usuário e navegação do menu.
- **manager.py:** Implementa a lógica de negócios, validações e coordena as operações.
- **database.py:** Gerencia a conexão e as operações SQL com o banco de dados.

Gestão de acesso de usuário

Perfis:

- `Usuário Comum`
- `Técnico (Admin)`

Casos de Uso:

1. **Abrir Novo Chamado:**
   - Perfis: `Usuário Comum` , `Técnico (Admin)`
2. **Visualizar Todos os Chamados:**
   - Perfis: `Técnico (Admin)`
3. **Buscar Chamado por ID:**
   - Perfis: `Usuário Comum` , `Técnico (Admin)`
4. **Atualizar Status do Chamado:**
   - Perfis: `Técnico (Admin)`
5. **Deletar Chamado:**
   - Perfis: `Técnico (Admin)`

Fluxo de dados

Criação de um novo usuário

1. **Usuário (Camada de Interface):** O usuário seleciona a opção "1" no menu apresentado pelo `main.py`.
2. `main.py` **(Camada de Interface):** A aplicação solicita ao usuário os dados necessários (`título`, `descrição`, `solicitante`) através da função `input()`.
3. **Chamada para a Lógica:** O `main.py` chama o método `gestor.criar_novo_chamado()`, passando os dados coletados como argumentos.
4. `manager.py` **(Camada de Lógica):** O método `criar_novo_chamado()` recebe os dados, realiza validações (verifica se campos obrigatórios não estão vazios) e, se tudo estiver correto, chama a função `database.criar_chamado()`.
5. `database.py` **(Camada de Dados):** A função `criar_chamado()` estabelece a conexão com o banco de dados e executa o comando SQL `INSERT`, passando os dados de forma segura para persistir o novo chamado na tabela.
6. **Retorno:** O sucesso da operação no banco é retornado para a camada de lógica, que por sua vez informa a camada de interface, que finalmente exibe uma mensagem de sucesso ao usuário.

Tratamento de Erros

- Entradas inválidas são tratadas com mensagens amigáveis.
- IDs inexistentes retornam mensagens de não encontrado.
- Erros de conexão ou operação no banco são capturados e exibidos ao usuário.
- O sistema previne SQL Injection utilizando parâmetros nas queries.
