

Gustavo Calce – RM:550638

Gabriel Rocha – RM:550788

Link para vídeo: https://www.loom.com/share/6e0d7ba52e5e4f32b58c3ed620ead9cd?sid=428f5275-6220-4d62-b58f-810d188d3b33

# Problema

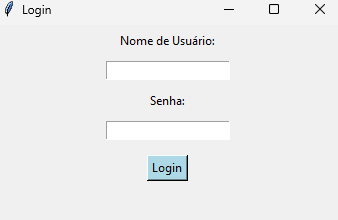
Em diversos estabelecimentos hospitalares, enfrenta-se o desafio notório do tempo de espera para atender pacientes, somado à ausência de uma análise preditiva em determinadas circunstâncias. O sistema de ERP desenvolvido pela MediMor, líder no setor, destina-se a mitigar substancialmente o tempo de espera e viabilizar a análise preditiva. Esse propósito é efetivado por meio de uma inovadora pulseira que monitora informações vitais do paciente, tais como batimentos cardíacos, níveis de oxigenação e temperatura corporal.

O diferencial distintivo do sistema ERP da MediMor reside na integração inovadora de uma pulseira de monitoramento de dados vitais. Ao contrário de soluções convencionais, essa abordagem permite uma coleta em tempo real e contínua de informações cruciais, como batimentos cardíacos, níveis de oxigenação e temperatura corporal. Essa conectividade em tempo real proporciona não apenas uma resposta mais ágil às necessidades do paciente, mas também alimenta um robusto sistema de análise preditiva. Assim, o sistema da MediMor não apenas reduz o tempo de espera, mas também oferece uma visão proativa e preditiva para aprimorar a eficiência e a qualidade do atendimento médico.

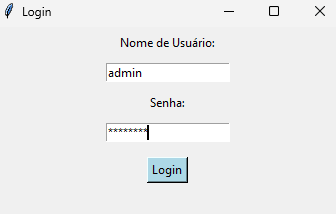
# Códigos

**Login.py**

Implementa a tela de login. A autenticação é baseada em um conjunto pré-definido de nomes de usuário e senhas. Após o login bem-sucedido, o usuário pode acessar o sistema.

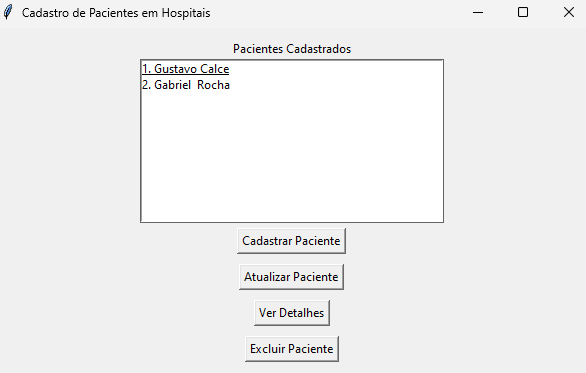


Com o usuário “admin” e senha “password”:



**HospitalApp.py**

Este script é o coração da aplicação. Ele cria a interface principal onde a lista de pacientes é exibida. As principais funcionalidades incluem cadastrar, atualizar, excluir e visualizar detalhes de pacientes. A aplicação também se conecta a um servidor MQTT para receber dados em tempo real.



**CadastrarDialog.py**

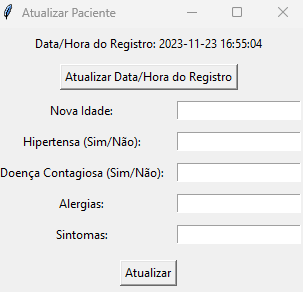
Este script implementa a janela de cadastro de pacientes. Ele contém uma classe `CadastrarDialog` que é chamada para criar uma nova janela de cadastro. A janela possui campos para informações do paciente, como nome, sobrenome, CPF (o CPF deve ser válido, o sistema entende a validação de CPFs), idade, sexo, condições de saúde (hipertensão, doença contagiosa, alergias) e sintomas. Após o cadastro, as informações são validadas, e o paciente é adicionado à lista de pacientes do sistema.



Após a receber dados do campo “Sintomas”, os dados serão enviados automaticamente para o médico que atenderá o paciente, gerando assim uma maior produtividade e diminuindo o tempo de espera dos pacientes, será possível analisar se os sintomas do paciente se enquadram em quadros de emergência, caso sim, o paciente deve ser levado para a ala emergencial do hospital.

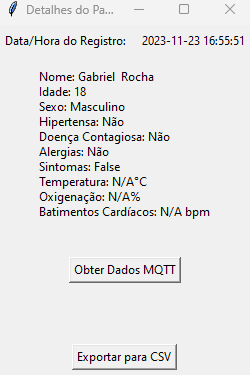
**AtualizarDialog.py**

Semelhante ao `CadastrarDialog.py`, este script implementa a janela de atualização de pacientes. Ele permite a atualização das informações de um paciente existente. Se o paciente não existir, um novo paciente será criado.



**DetalhesDialog.py**

Este script implementa a janela de detalhes do paciente. Ele exibe informações detalhadas do paciente, incluindo dados vitais monitorados pela pulseira. Além disso, há a opção de obter dados em tempo real através do protocolo MQTT e exportar os dados do paciente para um arquivo .CSV, o que pode ser usado em relatórios futuros. Essa janela será exibida para o médico e enfermeiros responsáveis pelo atendimento do tal paciente.



**Paciente.py**

Este módulo define a classe `Paciente`, que representa as informações de um paciente. Cada paciente possui atributos como nome, sobrenome, CPF, idade, etc. A classe também tem métodos para registrar consultas e é responsável por manter o histórico médico do paciente.

**MainApp.py**

Este é o ponto de entrada principal da aplicação. Ele inicia a interface gráfica e controla a transição entre a tela de login e a aplicação principal.

**Outros Arquivos**

- Hospital\_app.log: Registra mensagens de erro para fins de depuração.

- pacientes.json: Arquivo JSON para armazenar os dados dos pacientes.

**Como Executar pelo CMD(Windows)**

**Abra o Prompt de Comando:**

* Pressione **Win + R** para abrir a caixa de diálogo "Executar".
* Digite **cmd** e pressione Enter.

**Navegue até o Diretório do Código:**

* Use o comando **cd** para navegar até o diretório onde está o seu código. Por exemplo:

cd caminho/do/codigo

**Execute o código:**

python MainApp.py

**Executar pelo terminal da IDE:**

**Abra o Projeto na sua IDE**

* Abra a IDE e clique em "File" (Arquivo) > "Open Folder" (Abrir Pasta).
* Navegue até o diretório do seu projeto e clique em "Selecionar Pasta" para abri-lo.

**Abra o Arquivo do Código**

* Na sua IDE, navegue até o arquivo do seu código no painel lateral esquerdo.

**Execute o Código**

* Use o terminal integrado ou abra um terminal separado e digite:

python MainApp.py