**TOTEN ASSIST**

**INTEGRANTES:**

**1ESPA**

Guilherme Augusto Caseiro - RM: 559765

Leonardo Fernandes Mesquita - RM: 559623

Marco Antonio Caires Freire - RM: 559256

***TOTEN ASSIST: TOTEM DE AUTOATENDIMENTO***

***HOSPITAL INFANTIL SABARÁ***

**São Paulo**

*Junho de 2025*

**1. DESCRIÇÃO DO PROJETO** O projeto consiste no desenvolvimento de um protótipo funcional de totem de autoatendimento em Python, utilizando os frameworks Kivy e KivyMD, com foco no ambiente do Hospital Infantil Sabará. O objetivo é facilitar e humanizar o processo de triagem inicial dos pacientes , automatizando a coleta de informações cadastrais e a avaliação de sintomas para uma posterior classificação por prioridade de atendimento. A aplicação conta com uma interface moderna e intuitiva, e os dados da fila de atendimento são salvos de forma persistente.

**2. OBJETIVOS**

* Automatizar o processo de triagem inicial.
* Melhorar a eficiência do atendimento e potencialmente reduzir o tempo de espera.
* Identificar e priorizar casos urgentes de forma automática.
* Melhorar a experiência e a comunicação com o paciente desde o primeiro contato.

**3. TECNOLOGIAS UTILIZADAS**

* Python 3
* Framework Kivy
* Framework KivyMD (para componentes de Material Design)

**4. ESTRUTURA DO SISTEMA** A aplicação foi desenvolvida de forma modular, separando a lógica da interface para maior organização e manutenibilidade. A estrutura é composta por:

* **Três telas principais:** O fluxo do usuário passa por uma tela de boas-vindas (Splash Screen), uma tela de cadastro de dados e uma tela de autoavaliação de sintomas.
* **Interface (front-end):** Definida no arquivo totem.kv, que descreve a aparência e o layout de todos os componentes visuais.
* **Lógica (back-end):** Contida no arquivo main.py, que gerencia o comportamento das telas, a validação de dados, a lógica de classificação e a manipulação de arquivos.
* **Persistência de Dados:** As informações da fila de atendimento são salvas no arquivo patient\_queue.json, garantindo que os dados não se percam ao fechar o aplicativo.

**5. LÓGICA DE CLASSIFICAÇÃO** O sistema classifica os pacientes com base na gravidade dos sintomas informados:

* **Muito Emergente:** Atribuído a pacientes com dor extrema (nível 9-10), febre alta (acima de 40°C) ou presença de vômito.
* **Urgente:** Pacientes com dor forte (nível 7-8) ou febre moderada (acima de 39°C) com náusea.
* **Emergente:** Casos com sintomas moderados, como dor de nível 5-6 ou febre acima de 38°C.
* **Estável:** Pacientes sem sintomas críticos ou com dores leves.

A fila de atendimento é então ordenada automaticamente com base nessa prioridade.

**6. DIAGRAMA DE FUNCIONAMENTO** O fluxo de interação do sistema segue a seguinte ordem:

Tela de Abertura (Splash) → Entrada de Dados → Autoavaliação → Classificação por Sintomas → Geração/Atualização da Fila Prioritária → Resultado e Instruções ao Paciente

**7. OBSERVAÇÕES TÉCNICAS**

* **Estruturas de Dados:** Utilização de listas para a fila de atendimento e dicionários para organizar os dados de cada paciente.
* **Manipulação de Arquivos:** Implementação de rotinas para salvar (escrita) e carregar (leitura) a fila de pacientes em um arquivo JSON, garantindo a persistência dos dados.
* **Tratamento de Exceções:** Uso de blocos try-except para validação de entradas numéricas (grau da dor, temperatura), prevenindo crashes por dados inválidos.
* **Modularidade:** O código foi separado em classes para cada tela e o layout foi desacoplado para um arquivo .kv, seguindo as melhores práticas do Kivy.
* **Interface Gráfica:** A interface foi construída com componentes do KivyMD para uma experiência de usuário amigável e moderna, adequada para interação por toque.

**8. CONSIDERAÇÕES FINAIS**  **e POSSÍVEIS MELHORIAS.**

Este protótipo atende com sucesso a todos os requisitos funcionais propostos, entregando uma solução completa e bem estruturada para o problema da triagem hospitalar. A separação entre lógica e interface, o tratamento de erros e a persistência de dados demonstram a aplicação de conceitos fundamentais de engenharia de software.

**Possíveis Melhorias Futuras:**

* **Integração com Banco de Dados:** Substituir o arquivo JSON por um banco de dados relacional (como SQLite ou PostgreSQL) para maior escalabilidade e segurança.
* **Painel Administrativo:** Criar uma tela separada, protegida por senha, para que a equipe do hospital possa visualizar e gerenciar a fila de atendimento em tempo real.
* **API de Integração:** Desenvolver uma API para que o totem possa se comunicar com outros sistemas do hospital, como o Prontuário Eletrônico do Paciente (PEP).
* **Internacionalização:** Adicionar suporte para outros idiomas, como inglês e espanhol.