

# Documentação Do Sistema De Gerenciamento De Fila De Prioridade Do IntegraMed

---

**Integrantes: Cristian Brandão Tavares e Luis Felipe Ferreira Martins**

link do Github: <https://github.com/luis-felipe-ferreira/Sistema-integrador.git>

## 1.Introdução

---

Este documento descreve o funcionamento e a arquitetura do sistema de gerenciamento de fila de prioridade em uma aplicação web full-stack usando Typescript,html e sqlite. O sistema cobre o fluxo desde o cadastro de pacientes, passando pelo registro de triagem com atribuição de prioridade, até a gestão de uma fila de atendimento para médicos.

## 2.Tecnologias Usadas

---

**HTML5**

**CSS3**

**live-server:** Servidor de desenvolvimento para os arquivos estáticos.

**Node.js:** Ambiente de execução do JavaScript no servidor.

**Express.js:** Framework para a construção da API e do servidor web.

**TypeScript:** Linguagem principal para o desenvolvimento do backend.

**ts-node-dev:** Ferramenta para rodar o servidor em modo de desenvolvimento com recarregamento automático.

**CORS:** Middleware para permitir a comunicação entre o frontend e o backend.

**SQLite:** Banco de dados relacional leve e baseado em arquivo.

**Prisma:** ORM (Mapeador Objeto-Relacional) moderno para interagir com o banco de dados de forma segura e com tipagem forte.

### **3.Requisitos E Funcionalidades Implementadas**

---

**Autenticação por Perfil:** Sistema de login para três tipos de usuários: Recepcionista, Enfermeira e Médico.

#### **Cadastro de Pacientes:**

A recepcionista pode cadastrar novos pacientes no sistema.

Validação de CPF: O sistema impede o cadastro de CPFs duplicados e com tamanho incorreto (diferente de 11 dígitos).

Busca Dinâmica de Pacientes: Na tela de triagem, a enfermeira pode buscar pacientes por nome ou CPF em tempo real (autocomplete) para agilizar o atendimento.

Registro de Triagem: A enfermeira registra os sinais vitais do paciente e atribui um nível de prioridade (de 1 a 5).

#### **Fila de Atendimento Persistente:**

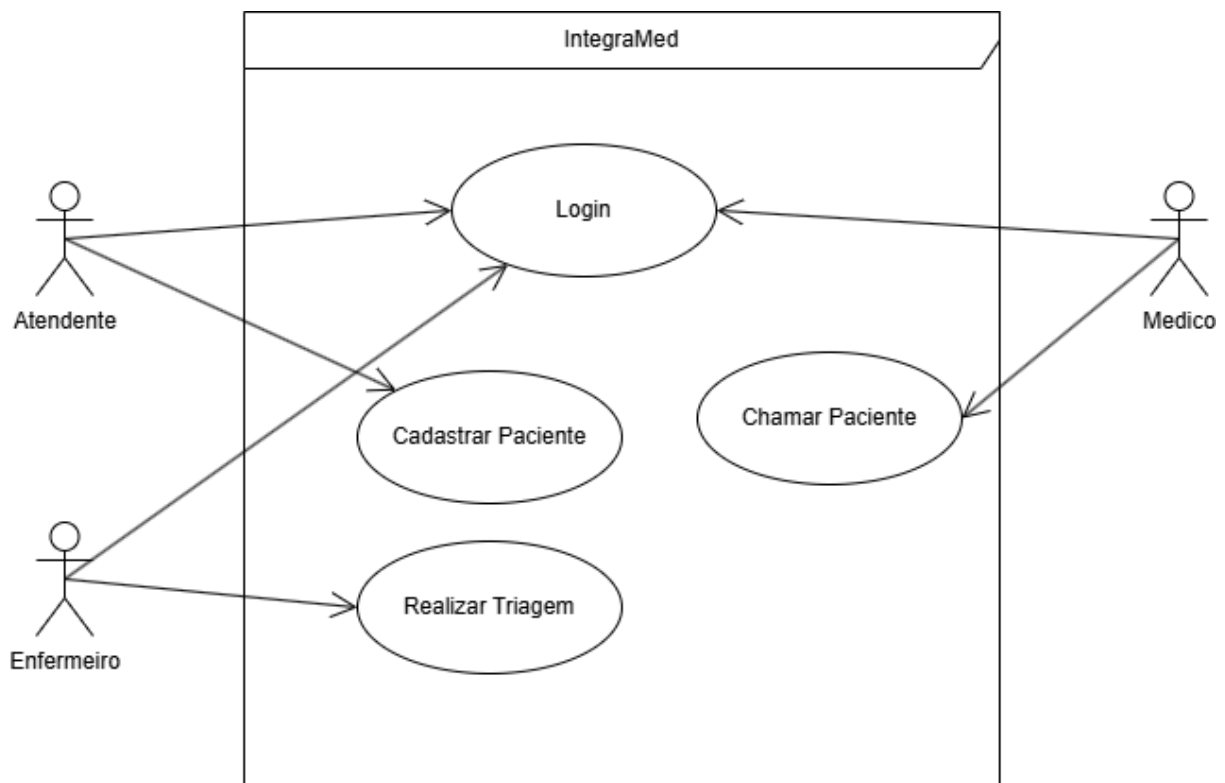
Pacientes triados entram em uma fila que é salva no banco de dados, garantindo que os dados não se percam.

Validação de Fila: O sistema impede que um paciente que já está na fila seja adicionado novamente.

#### **Painel do Médico:**

O médico visualiza a fila de atendimento ordenada por prioridade (menor número é mais urgente) e, para prioridades iguais, por ordem de chegada.

Ele pode "chamar" o próximo paciente, que é então removido da fila e tem seus dados de triagem exibidos em detalhe.



## 4.Estrutura Do Sistema

---

### TypeScript:

[minHeap.ts](#): estrutura da fila de prioridade.

[cadastrar\\_paciente.ts](#): código responsável por cadastrar o paciente.

[login.ts](#), [enfermeira.ts](#), [recepcionista.ts](#): código responsável por fazer login.

[medico.ts](#): responsável por fazer login, visualizar a fila e chamar o paciente para ser atendido.

[triagem.ts](#): código responsável por fazer triagem do paciente.

[authService.ts](#): código responsável por autenticação do sistema.

[dataService.ts](#): gerencia a interação com o banco de dados.

[server.ts](#): código que implementa a API

[types.ts](#): responsável pela implementação das interfaces

### Html:

[login.html](#): responsável pelo login do usuário.

[recepcionista.html](#), [cadastrar\\_paciente.html](#): responsável pela tela de cadastro do paciente

enfermeira.html,triagem.html: responsável pela tela triagem do paciente  
medico.html: responsável pela tela da fila e chamar o paciente

## 5. Instalação E Configuração

---

Siga estes passos para configurar o ambiente do zero.

Instale as Dependências do Frontend:

Abra um terminal na pasta raiz do projeto (.../sistema\_hospitalar\_projeto/).

Execute o comando:

Bash

```
npm install
```

Instale e Configure o Backend:

Ainda no terminal, navegue para a pasta backend:

Bash

```
cd backend
```

Instale as dependências do servidor:

Bash

```
npm install
```

Execute a migração do Prisma para criar o arquivo do banco de dados (dev.db) e suas tabelas:

Bash

```
npx prisma migrate dev --name init
```

Gere o Prisma Client, o código que permite a comunicação com o banco:

Bash

```
npx prisma generate
```

Volte para a pasta raiz para os próximos passos:

Bash

cd ..

## 6. Como Rodar A Aplicação

---

Existem duas maneiras de rodar a aplicação: um modo de desenvolvimento (dois terminais) e um modo de produção simplificado (um terminal).

Modo de Desenvolvimento (Ideal para programar)


Terminal 1 - Backend:

Navegue até a pasta backend.

Inicie o servidor de desenvolvimento. Ele reiniciará automaticamente a cada alteração no código.

Bash

npm run dev

Você verá a mensagem:  Servidor backend rodando na porta 3001. Deixe este terminal aberto.

Terminal 2 - Frontend:

Navegue até a pasta raiz do projeto.

Compile o frontend para a pasta dist.

Bash

npm run build

Navegue para a pasta dist e inicie o live-server.

Bash

cd dist

live-server

Acesse o site em <http://127.0.0.1:8080/html/login.html>.

## 7. Como Usar o Sistema

---

Para fazer o login dos perfis:

Perfil: recepcionista, Senha: 123

Perfil: enfermeira, Senha: 456

Perfil: médico, Senha: 789

Após fazer login:

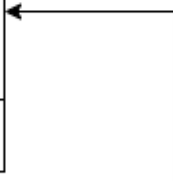
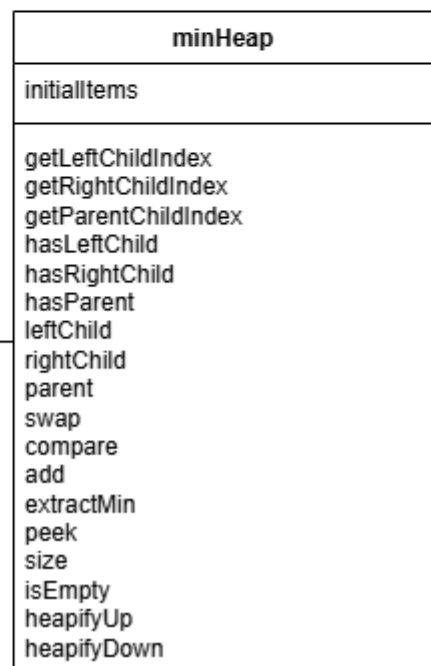
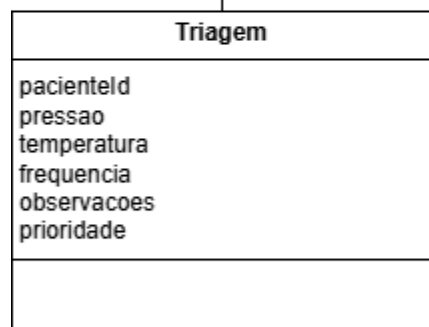
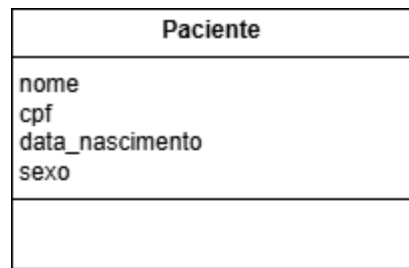
Recepcionista: terá acesso a uma tela com o botão cadastrar paciente que ao clicar irá abrir o formulário para cadastrar o paciente onde todos os campos são obrigatórios. Após o preenchimento do formulário basta clicar no botão cadastrar que realizará o cadastro do paciente

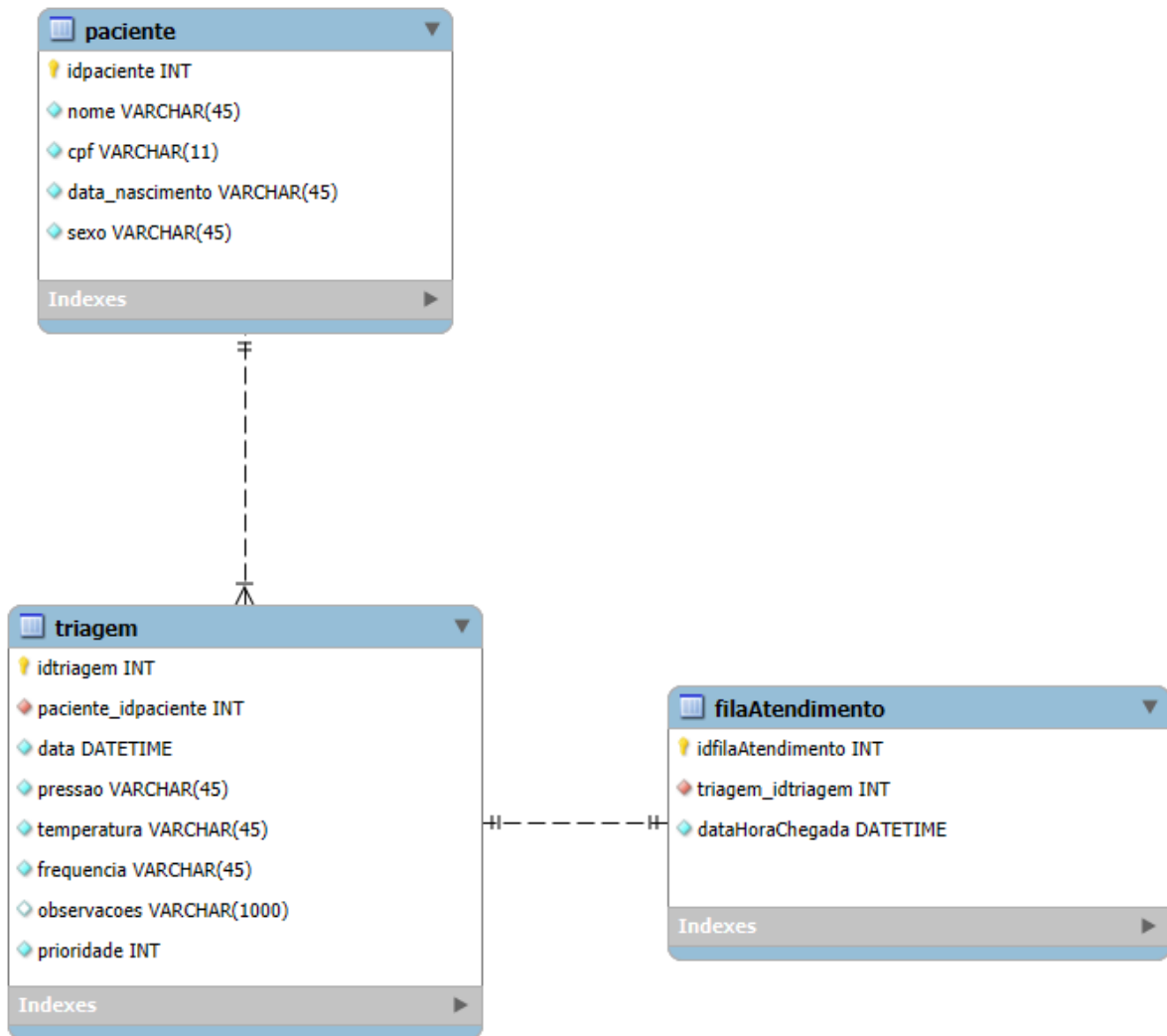
Enfermeira: terá acesso a uma tela com o botão realizar nova triagem que ao clicar irá abrir um formulário onde você seleciona o paciente e preenche os campos. O campo observações adicionais é opcional o preenchimento. Após o preenchimento do formulário basta clicar no botão salvar triagem que realizará a triagem do paciente

Médico: terá acesso a uma tela com a fila dos pacientes e um botão para chamar o próximo paciente da fila

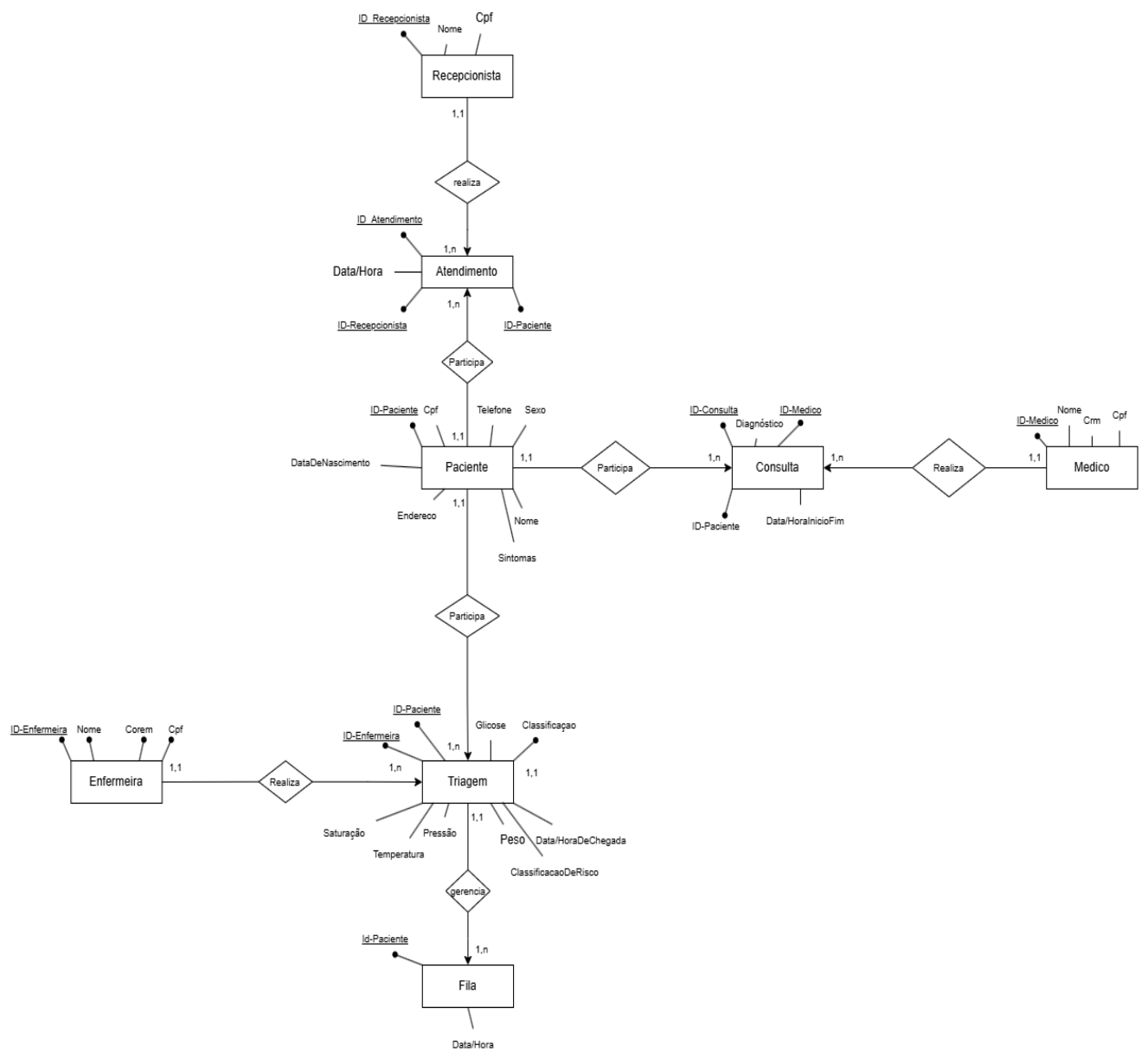
## **8. Diagramas e Benchmark**

---









Critérios	Código grupo	Código aberto
Linguagem	TypeScript	C
Estrutura	Min-Heap	Lista Encadeada Simples
Lógica de priorização	Baseada nos níveis de prioridade SUS. Operações de inserção e extração do prioritário são $O(\log n)$ .	A lógica para manter a ordem ou encontrar o próximo prioritário pode ser $O(n)$ .
Complexidade	→ Adicionar Paciente (com prioridade): $O(\log n)$ - Muito eficiente. A nova prioridade sempre sobe para sua posição correta no heap. → Chamar Próximo Paciente (extrair o de maior prioridade): $O(\log n)$ - Muito eficiente. O elemento raiz é removido e o heap é reorganizado. → Ver Próximo Paciente (sem remover): $O(1)$ - Simplesmente olha o elemento raiz.	→ Adicionar Paciente: $O(n)$ no pior caso, pois pode ser necessário percorrer a lista para encontrar a posição correta de inserção para manter a ordem. → Chamar Próximo Paciente do início: $O(1)$ . → A complexidade da inserção torna-se um problema.

fonte: <https://github.com/Mahelchandupa/Hospital-Queue-Management-System>

## 9.Link do GitHub

Github: <https://github.com/luis-felipe-ferreira/Sistema-integrador.git>

Integrantes: Cristian Brandão Tavares e Luis Felipe Ferreira Martins