Documentação de um Produto de Software

versão 1.0

autores: Mayara Lima,
Levi Passos e
Irapuam junio

27 de Junho de 2025

SUMÁRIO

1. Introdução

O Sistema de Gerenciamento de Filas de Pronto-Socorro Prioriza+ foi desenvolvido para otimizar o fluxo de atendimento em unidades de pronto-socorro, proporcionando uma gestão eficiente das filas de pacientes com base na classificação de risco. O sistema implementa o Protocolo de Manchester, um método de triagem amplamente utilizado em serviços de emergência médica para priorizar o atendimento de pacientes de acordo com a gravidade de seus quadros clínicos. A superlotação e os longos tempos de espera são problemas comuns em prontos- socorros, muitas vezes resultando em experiências negativas para os pacientes e sobrecarga para os profissionais de saúde. O Prioriza+ busca mitigar esses problemas através da automação e otimização do processo de triagem e gerenciamento de filas, garantindo que os pacientes com condições mais graves sejam atendidos prioritariamente, enquanto os demais recebam atendimento dentro de prazos adequados. Este documento fornece uma visão abrangente do sistema, incluindo sua arquitetura, funcionalidades, instruções de instalação e uso, além de considerações sobre segurança e manutenção.

2. Visão Geral do Sistema

O Prioriza+ é uma aplicação web completa que permite o gerenciamento eficiente de filas em prontos-socorros. O sistema abrange todo o fluxo de atendimento, desde o cadastro inicial do paciente até a finalização do atendimento médico.

Principais Funcionalidades:

1. Cadastro de Pacientes: Registro completo de informações dos pacientes, incluindo dados pessoais e de contato.

2. Triagem e Classificação de Risco: Implementação do Protocolo de Manchester para classificação de pacientes em cinco níveis de prioridade:

- Vermelho (Emergência): Atendimento imediato
- Laranja (Muito Urgente): Atendimento em até 10 minutos
- Amarelo (Urgente): Atendimento em até 60 minutos
- Verde (Pouco Urgente): Atendimento em até 120 minutos
- Azul (Não Urgente): Atendimento em até 240 minutos

3. Gerenciamento de Filas: Visualização e controle das filas de atendimento, com organização automática por prioridade e tempo de espera.

4. Chamada de Pacientes: Sistema para chamar o próximo paciente da fila, com registro de local de atendimento e profissional responsável.

5. Monitoramento em Tempo Real: Dashboard com informações atualizadas sobre o status das filas, tempos de espera e atendimentos em andamento.

6. Controle de Acesso: Sistema de autenticação e autorização para garantir que apenas usuários autorizados acessem funcionalidades específicas.

Benefícios do Sistema:

. Redução do tempo de espera para pacientes em estado crítico.
. Melhor distribuição dos recursos médicos de acordo com a demanda .
. Maior transparência no processo de atendimento Redução de conflitos relacionados à ordem de atendimento,
. Coleta de dados para análise e melhoria contínua do serviço.
. Melhoria na experiência do paciente e na satisfação dos profissionais de saúde.

3. Arquitetura do Sistema

O Prioriza+ foi desenvolvido seguindo uma arquitetura de três camadas, com separação clara entre frontend, backend e banco de dados. Esta abordagem proporciona modularidade, facilitando a manutenção e evolução do sistema.

4. Banco de Dados:

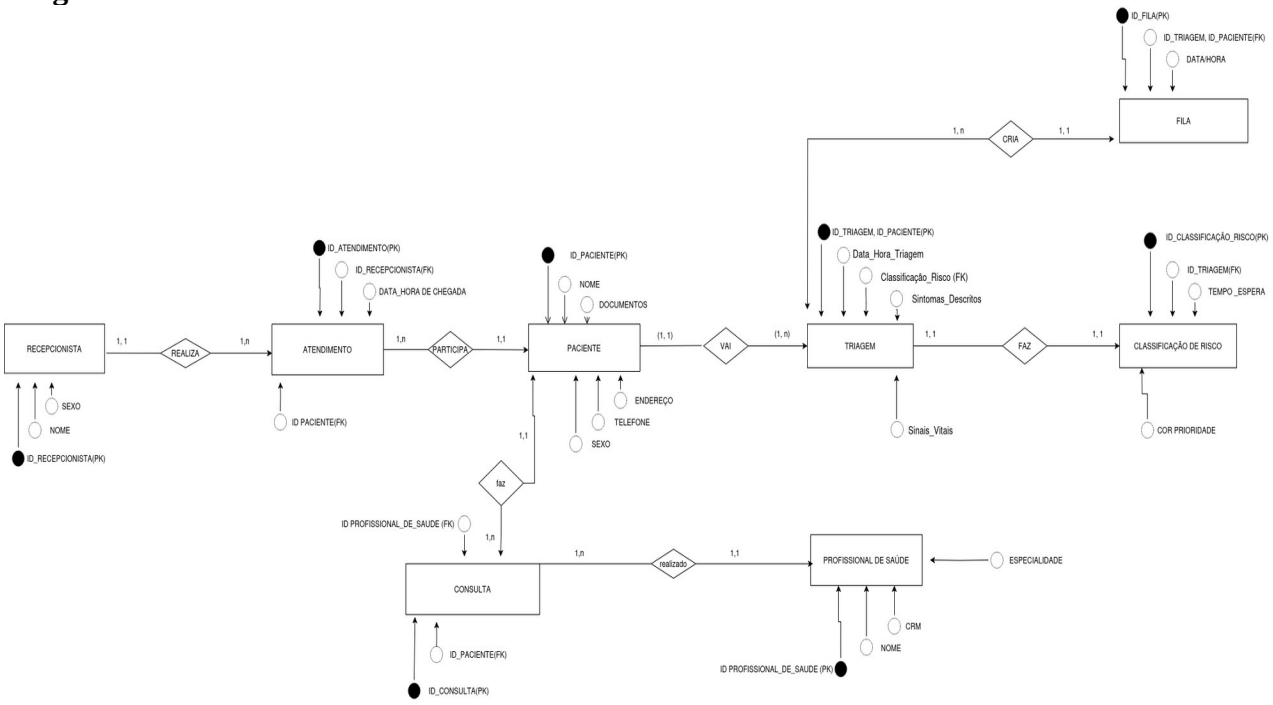
O sistema utiliza um banco de dados relacional SQLite para armazenamento persistente dos dados. A estrutura do banco foi projetada para garantir integridade referencial e otimizar as consultas mais frequentes.

<u>Modelo de Entidade-Relacionamento:</u>

O banco de dados é composto pelas seguintes entidades principais:

1. Pacientes: Armazena informações dos pacientes
2. Atendimentos: Registra cada visita ao pronto-socorro
3. Triagens: Armazena dados da classificação de risco
4. Filas: Gerencia as filas de atendimento
5. Profissionais: Informações dos profissionais de saúde
6. Usuários: Gerencia os usuários do sistema
7. Chamadas: Registra as chamadas de pacientes

Diagrama de Entidade-Relacionamento



Descrição das Tabelas

- Entidade: Recepcionista

Atributo	Tipo de Dado	Chave	Descrição
id_recepcionista	Inteiro	PK	Identificador do usuário.
nome	string	-	Nome completo.
sexo	string	-	Feminino ou masculino.

- Entidade: Atendimento

Atributo	Tipo de Dado	Chave	Descrição
id_atendimento	Inteiro	PK	Identificador do usuário
id_recepcionista	inteiro	FK	Dados do profisional de saúde.
data_hora de chegada	date		Data do atendimento do paciente.
id_paciente	inteiro	FK	Dados do paciente.

- Entidade: Paciente

Atributo	Tipo de Dado	Chave	Descrição
id_paciente	Inteiro	PK	Identificador único do paciente
nome	string		Nome completo do paciente
documentos	string		Data de nascimento, cpf, rg etc
sexo	string		Sexo do paciente (M, F, O)
endereço	string		Endereço do paciente
telefone	inteiro		Número de telefone

- Entidade: Triagem

Atributo	Tipo de Dado	Chave	Descrição
id_triagem,id_paciente	Inteiro	PK	Identificador da triagem
classificação_risco	string	FK	Cinco cores de classificação de risco
sinais_vitais	string		Resumo dos sinais vitais
Sintomas_ descritos	string		Sintomas relatados pelo paciente
data/hora_triagem	date		Data e hora da triagem

- Entidade: Classificação_risco

Atributo	Tipo de Dado	Chave	Descrição
id_classificação_risco	Inteiro	PK	Identificador da classificação_risco.
id_triagem	Inteiro	FK	Referência a triagem.
tempo_espera	inteiro		Tempo de espera para o atendimento.
cor_prioridade	string		Cor do nível do risco.

- Entidade: fila

Atributo	Tipo de Dado	Chave	Descrição
id_fila	Inteiro	PK	Identificador do usuário
id_triagem, id_paciente(fk)	inteiro	FK	Identificador do paciente.
data/hora	inteiro		Data/hora da entrada na fila.

- Entidade: Profissional_saúde

Atributo	Tipo de Dado	Chave	Descrição
id_Profissional_saúde	Inteiro	PK	Identificador do profissional de saúde.
nome	string		Nome completo do profissional.
especialidade	string		Especialidade do profissional.
crm	string		É o número de registro do médico no Conselho Regional de Medicina do estado onde atua.

- Entidade: Consulta

Atributo	Tipo de Dado	Chave	Descrição
id_consulta	Inteiro	PK	Identificador do usuário
id_Paciente	inteiro	FK	Identificador do paciente.
id_profissional_saúde	inteiro	FK	Data do atendimento do paciente.

- Relacionamentos

Entidades Envolvidas	Tipo de Relacionamento	Descrição
Paciente — Triagem	1 : N	Um paciente pode passar por várias triagens.
Paciente — Atendimento	1 : N	Um paciente pode passar por vários atendimentos.
Recepcionista — Atendimento	1 : N	Uma recepcionista pode fazer vários atendimentos.
Triagem — Classificação_risco	1 : 1	Uma triagem só pode fazer uma classificação.
Triagem — fila	1:1	Uma triagem só pode gerar uma única fila.
Paciente — Consulta	1: n	Um paciente pode participar de muitas consultas.
Consulta — Profissional de saúde	1:1	Uma consulta só pode ser realizada por um único profissional de saúde.

5. Backend e frontend

O frontend e backend do sistema foram desenvolvidos utilizando o framework Express em Typescript, HTML, CSS e Javascript e com a biblioteca de estilos BOOTSTRAP 5.0. A API fornece endpoints para todas as operações necessárias do sistema.

```
pronto_socorro_api/
├── src/ # Código fonte do backend
│   ├── Register.ts
│   ├── Pacient.ts
│   ├── Triage.ts
│   ├── Heap.ts
│   ├── PriorityControl.ts
│   ├── utils.ts
│   ├── wrapper.ts
│   └── app.ts
├── public/ # Páginas HTML do sistema
│   ├── index.html # Página de login do sistema
│   ├── menu-administrador.html
│   ├── menu-recepcionista.html
│   ├── triagem.html
│   └── fila.html
├── tsconfig.json # Configurações do compilador
└── package.json # Configurações do projeto
├── node_modules/ # bibliotecas do sistema
└── databases/ # Base de dados do sistema
```

Endpoints da API
A API expõe os seguintes endpoints principais:

Autenticação

Método	Endpoint	Descrição
POST	/login	Realiza login com base em usuário e senhas cadastradosno sistema

Cadastro

Método	Endpoint	Descrição
POST	/cadastrar-funcionario	Realiza o cadastro de um funcionario no sistema
PUT	/excluir-funcionario	Exclui um funcionario especifico da base de dados do sistema
DELETE	/api/pacientes/:id	Remove um paciente

Sistema

Método	Endpoint	Descrição
POST	/pesquisar-funcionarios	Retorna todos os funcionarios na base de dados com base em um parâmetro
POST	/cadastrar-paciente	Realiza o cadastro de um paciente no sistema
PUT	/modificar-paciente	Modifica os dados de um paciente
DELETE	/excluir-paciente	Exclui um paciente especifico da base de dados do sistema
POST	/pesquisar-pacientes	Retorna todos os pacientes na base de dados com base em um parâmetro

Triagens

Método	Endpoint	Descrição
POST	/triagem-marcar	Marca a triagem de um paciente especifico e o coloca na fila de espera para triagem
POST	/triagem	Realiza o registro de uma triagem de um paciente que está na fila de espera para triagem

Filas

Método	Endpoint	Descrição
GET	/fila	Retorna a fila de pacientes a espera de atendimento médico atualmente
POST	/fila/proximo	Realiza a chamada do próximo paciente a ser atendido com base nos critérios de prioridade

Autenticação e Autorização
1. O usuário fornece credenciais (login e senha)
1. 2. A API valida as credenciais e redireciona para a página correspondente ao cadastro do usuário.

Tratamento de Erros
A API implementa tratamento padronizado de erros, retornando respostas com códigos HTTP apropriados e mensagens descritivas:
200: Sucesso
201: Recurso criado com sucesso
400: Requisição inválida
401: Não autorizado
403: Acesso proibido
404: Recurso não encontrado
500: Erro interno do servidor

Principais Páginas
1. Login: Página de autenticação para acesso ao sistema
2. Filas: Gerenciamento das filas de atendimento
3. Menu da Recepção: Listagem e cadastro de pacientes assim como marcação de triagens
4. Menu da Administração: Gerenciamento dos registros de paciente e funcionários
5. Triagem: Realização de triagem e classificação de risco

Tecnologias Utilizadas
- React: Biblioteca para construção de interfaces
- Vite: Ferramenta de build rápida para desenvolvimento
- React Router: Gerenciamento de rotas
- Axios: Cliente HTTP para comunicação com a API
- Tailwind CSS: Framework CSS utilitário
- Shadcn/UI: Componentes de UI reutilizáveis
- React Toastify: Notificações toast
- JWT Decode: Decodificação de tokens JWT

Fluxo de Autenticação
1. O usuário acessa a página de login
2. Após autenticação bem-sucedida, os tokens são armazenados no localStorage
3. O usuário é redirecionado para o dashboard
4. Todas as requisições subsequentes incluem o token de acesso
5. Se o token expirar, o sistema tenta renovar automaticamente usando o token de atualização
6. Se a renovação falhar, o usuário é redirecionado para a página de login

Responsividade
A interface foi projetada para funcionar bem em diferentes tamanhos de tela:
Desktop: Layout completo com todas as funcionalidades
Tablet: Layout adaptado com menus colapsáveis
Mobile: Layout simplificado com foco nas funcionalidades essenciais

6. Instalação e Configuração
Esta seção descreve os passos necessários para instalar e configurar o sistema em um ambiente de produção.
- Requisitos de Sistema Operacional: Ter um sistema operacional que consiga rodar sqlite3, compilador typescript e node.js

Instalação
1. Clone o repositório: bash git clone https://github.com/Yrahpuan/projeto-interdisciplinar.git
2. Vá para o local do repositório clonado
3. Abrir um terminal ou prompt de comando
4. digitar npm run server se estiver no windows e npm run serveri se estiver no linux
5. o servidor já estará rodando e o sistema pode ser acessado usando o link de localhost que aparece no prompt de comando

7. Guia do Usuário
Esta seção fornece instruções para os usuários finais sobre como utilizar o sistema.
Acesso ao Sistema
1. Abra um prompt de comando no diretório onde foi clonado o repositório.
2. Digite npm run server se estiver no windows e npm run serveri se estiver no linux
3. O servidor já estará rodando e o sistema pode ser acessado usando o link de localhost que aparece no prompt de comando.
4. A primeira tela que aparecerá é a de login, logue com nome de usuário e senha para acessar as outras páginas do sistema como:

- Cadastro:
1. Acesse o sistema como administrador para poder realizar o cadastro tanto de funcionários como de pacientes, um usuário administrador possue privilégios de modificar os dados de funcionários já registrados no sistema e de excluí-los. Acesse o sistema como recepcionista para poder realizar o cadastro de pacientes e marcar-triagens.

- Registro de Triagens:
1. Acesse o sistema como triador para poder realizar o registro de triagens dos pacientes

- Vizualização da fila de prioridade:
1 Acesse o sistema como um médico para poder visualizar a fila assim como os pacientes que estão nela. Você poderá apertar em um botão Chamar Próximo para que o paciente com a maior prioridade seja retirado da fila.

Observações: O sistema não foi feito com o intuito de ser usado como uma ferramenta profissional para
resolver o problema, seu fim é somente educacional e para ganho de experiência dos envolvidos. Há várias falhas de segurança como o acesso sem autenticação as páginas destinadas a acessos específicos no sistema, o banco de dados ser local e de fácil manipulação, não haver registro de usuário ou sessão e manipulação de dados direta e sem uso de transações no banco de dados.

8. Testes
O sistema só foi testado manualmente para descoberta de bugs, erros e quebras de sequência assim como para melhorar a experiência de uso do sistema.
O sistema inclui testes automatizados para garantir a qualidade e a confiabilidade do código.

Testes de Backend:
Os testes do backend foram implementados usando o framework pytest e cobrem:
. Testes Unitários: Testam funções e métodos isoladamente
. Testes de Integração: Testam a interação entre diferentes componentes.
- Testes de API: Testam os endpoints da API

Para executar os testes do backend: cd pronto_socorro_api
source venv/bin/activate pytest

Testes de Frontend:
Os testes do frontend foram implementados usando Jest e React Testing Library e cobrem:
. Testes de Componentes: Testam o comportamento dos componentes React
. Testes de Integração: Testam a interação entre diferentes
componentes . Testes de Renderização: Testam a renderização correta
da interface.

Para executar os testes do frontend: cd pronto_socorro_frontend
npm test

Testes Manuais:
Além dos testes automatizados, recomenda-se a realização de testes manuais para verificar:
. Usabilidade: Facilidade de uso da interface
. Responsividade: Comportamento em diferentes tamanhos de tela
. Desempenho: Tempo de resposta e carregamento
. Compatibilidade: Funcionamento em diferentes navegadores.

9. Considerações de Segurança
O sistema implementa várias medidas de segurança para proteger os dados e garantir o uso adequado:

Autenticação e Autorização
. Autenticação baseada em JWT com tokens de curta duração
. Senhas armazenadas com hash seguro (bcrypt)
. Controle de acesso baseado em perfis de usuário
. Proteção contra ataques de força bruta

Proteção de Dados
. Comunicação criptografada via HTTPS
. Validação de entrada para prevenir injeção de SQL e XSS
. Sanitização de dados antes do armazenamento
. Logs de auditoria para ações críticas

Conformidade com LGPD
O sistema foi projetado considerando os requisitos da Lei Geral de Proteção de Dados (LGPD):
. Coleta apenas dados necessários para o funcionamento do sistema
. Implementa controles de acesso para proteger dados sensíveis
. Permite a exclusão de dados quando apropriado
. Mantém registros de operações realizadas com dados pessoais

10. Manutenção e Suporte
Esta seção fornece orientações para a manutenção contínua do sistema.
Como esse sistema foi feito somente para ganho de experiência, provavelmente não receberá atualizações ou manutenções, mas você pode contatar os envolvidos para se precisar de ajuda com relação ao sistema.

11. Referências
1. Protocolo de Manchester. Grupo Brasileiro de Classificação de Risco. Disponível em: https://gbcr.org.br/
2. Lei Geral de Proteção de Dados (LGPD). Disponível em: https://www.gov.br/
3. cidadania/pt-br/acesso-a-informacao/lgpd