
*Trabalho de Conclusão de Curso*

# Sistema de Gestão e Controle de Atividades em Unidade de Saúde

de **Wagner Williams Barros Ferreira Filho**

**orientado por**
**Prof. Dr. Erick de Andrade Barboza**

Universidade Federal de Alagoas
Instituto de Computação
Maceió, Alagoas
01 de Fevereiro de 2022

UNIVERSIDADE FEDERAL DE ALAGOAS
Instituto de Computação

# SISTEMA DE GESTÃO E CONTROLE DE ATIVIDADES EM UNIDADE DE SAÚDE

Trabalho de Conclusão de Curso submetido ao Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do grau de Engenheiro de Computação.

Wagner Williams Barros Ferreira Filho

***Orientador: Prof. Dr. Erick de Andrade Barboza***

**Banca Avaliadora:**

| | |
|---|---|
| Leandro Dias da Silva | Prof. Dr., UFAL |
| Thiago Damasceno Cordeiro | Prof. Dr., UFAL |

Maceió, Alagoas
01 de Fevereiro de 2022

**Catalogação na Fonte**
**Universidade Federal de Alagoas**
**Biblioteca Central**
**Divisão de Tratamento Técnico**

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

F383s Ferreira Filho, Wagner Williams Barros.
Sistema de gestão e controle de atividades em unidade de saúde / Wagner Williams Barros Ferreira Filho. – 2021.
43 f. : il.

Orientador: Erick de Andrade Barboza.
Monografia (Trabalho de conclusão de curso em Engenharia de Computação) - Universidade Federal de Alagoas, Instituto de Computação. Maceió.

Bibliografia: f. 42-43.

1. Sistema de Saúde - Brasil. 2. Protocolo de Manchester. 3. Análise de dados. 4. Sistema web. I. Título.

CDU: 004.6:61

# Dedicatória

Dedico este trabalho a minha avó Izabel, que sempre esteve ao meu lado apoiando e me incentivando a aprimorar minha educação e a nunca desistir dos meus sonhos.

*Wagner Williams Barros Ferreira Filho*

# Agradecimentos

Aos meus pais e irmãos por todo apoio e incentivo ao longo dos anos da graduação, sempre dando todo o suporte sem o qual eu jamais teria chegado até aqui.

A minha namorada incrível, Amanda, que esteve comigo do início ao fim dessa dura caminhada, mesmo passando também por grandes dificuldades você sempre esteve comigo, dando apoio, atenção, carinho e amor. Compartilhamos juntos momentos de tristezas e alegrias, crises de estresse e de risos, por isso, dedico além deste trabalho todo meu amor a você.

Aos grandes amigos que fiz no decorrer do curso, por sempre estarem juntos nas rodas de estudo, nos almoços do RU, lanches a tarde e sobretudo, pelas incontáveis risadas que sem dúvida amenizaram a dificuldade de tantos obstáculos pelos quais passamos juntos.

Aos professores que acompanharam a minha jornada acadêmica de perto, deram muito apoio, incentivo e mostraram grande parceria dentro e fora da sala de aula. Em especial ao professor Dr. Leandro Dias que abriu para mim as portas do mundo da pesquisa e ao professor e orientador, Dr. Erick de Andrade Barboza pela oportunidade de trabalhar em sua pesquisa e por me orientar e conduzir o trabalho com paciência e dedicação, sempre disponível a me auxiliar e compartilhar toda sua experiência e conhecimento.

Esse TCC é de todos vocês!

09 de janeiro de 2022, Maceió - AL *Wagner Williams Barros Ferreira Filho*

# Resumo

A saúde no Brasil há muitos anos vem sendo um dos grandes problemas do Brasil, muito por conta do sucateamento de suas unidades públicas de emergência e pronto socorro, por falta de investimentos, falta de modernização e também falta de capacidade de gestão. Com todos esses problemas, e nos tempos modernos em que vivemos é imprescindível contar com a tecnologia para atuar como parceira para ajudar a amenizar ou solucionar alguns deles. Empresas dos mais diversos ramos do mercado apostam na modernização de seus modelos de negócios para incluir sistemas informatizados, inteligentes, que ajudam a processar informações dos serviços realizados na empresa e no mercado a sua volta e assim recomendar aos donos e analistas dessa empresa soluções e tomadas de decisões inteligentes. Na saúde isso não é diferente, no entanto, ainda existe no nosso país um grande atraso na informatização de setores públicos de saúde. Este trabalho propõe uma plataforma base para controle e monitoramento de atividades corriqueiras realizadas em Unidades de Pronto Atendimento (UPA), com foco na execução do protocolo manchester aplicado nas triagens.

***Palavras-chave*: *Sistema de Saúde*; *Protocolo Manchester*; *Sistema Web*; *Análise de Dados*.**

# Abstract

Health in Brazil has been one of Brazil's biggest problems for many years, largely due to the scrapping of its public emergency and emergency care units, lack of investment, lack of modernization and also lack of management capacity. With all these problems, and in the modern times we live in, it is essential to have technology to act as a partner to help alleviate or solve some of them. With all these problems, and in the modern times we live in, it is essential to have technology to act as a partner to help alleviate or solve some of them. In health this is no different, however, in Brazil there is still a great delay in the computerization of public health sectors. This work proposes a base platform for the control and monitoring of common activities carried out in Emergency Care Units (UPA), with a focus on the implementation of the Manchester protocol applied in screenings.

# Lista de Figuras

# Lista de Abreviaturas

**API** *Application Programming Interfaces*.

**CHUC** Centro Hospitalar Universitário de Coimbra.

**HTTP** *Hyper Text Transfer Protocol* (Protocolo de Transferência de Hipertexto).

**IC** Instituto de Computação.

**MTV** *Model Template View*.

**MVC** *Model View Controller*.

**REST** *Representational State Transfer* (Transferência Representacional de Estado).

**SAMU** Serviço de Atendimento Móvel de Urgência.

**POST** *Model View Controller*.

**STM** Sistema de Triagem de Manchester.

**UI** *User Interface* (Interface de usuário).

**UPA** Unidade de Pronto Atendimento.

# Sumário

# Capítulo 1

# Introdução

É inegável que em nosso país, o Brasil, um dos maiores problemas públicos é a promoção à saúde. Apesar de, segundo o [UnaSus, 2021], termos o maior sistema público de saúde do mundo, que atende desde exames de rotina e consultas, nas Unidades Básicas de Saúde, até atendimentos de alta complexidade hospitalar, em Pronto-Socorros e Unidades Hospitalares, falta investimento em recursos financeiros e também capacidade de gerenciamento para todas as demandas que esses locais exigem. As Unidades de Pronto Atendimento (UPAs), atuam como principal componente fixo de urgência pré-hospitalar para evitar que casos de complexidade intermediária sejam levados a rede hospitalar. Assim como os serviços de atendimento móvel de urgência (SAMU), elas surgiram no início dos anos 2000 como uma política nacional de atenção às urgências a partir de um decisão que pode-se dizer que foi acertada dado que haviam experiências nacionais e internacionais mostrando impacto positivo desse sistema.

As UPAs funcionam 24 horas por dia, [MinistériodaSaúde, ], com atendimento de média e alta complexidades, atendimentos laboratoriais e infecções de urgência, pequenos traumas e febres altas. No entanto, o artigo [Souza, 2021] salienta que elas não são unidades hospitalares, o paciente só pode ficar no máximo por 24 horas em observação, onde caso não estabilize o seu quadro clínico, ele deve ser encaminhado ao serviço hospitalar adequado. As UPAs são bem equipadas, fazem classificação de risco e todo paciente na UPA é atendido por um médico, garantindo assim, que não haverá dispensa ou encaminhamento ao hospital sem um diagnóstico médico. Os principais desafios de uma UPA englobam a gestão profissional e a "internação"dos pacientes.

Os primeiros momentos do paciente em hospitais e unidades de saúde são imprescindíveis para a garantia de um atendimento eficiente, sem piora do quadro clínico e com menos riscos de transtornos médicos. É por isso que muitos hospitais aplicam o Protocolo de Manchester que é um método para triagem de pacientes logo após sua chegada na unidade, onde eles são classificados por cores conforme a gravidade de cada caso. Como o próprio nome sugere, o protocolo foi criado na cidade de Manchester, na Inglaterra, em 1997. Segundo o artigo, [Rosa, 2021], esse método se mostrou tão eficaz que hoje

é aplicado em diversas instituições de saúde, visando melhorar e agilizar o atendimento ao dar prioridade a casos mais graves, além de aperfeiçoar substancialmente os processos em geral. Estudos recentes, como o artigo [Souza et al., 2015], apontam bons níveis de confiabilidade, sensibilidade e especificidade do protocolo. Além disso ele também de mostrou-se bom preditor da necessidade de internação e de mortalidade hospitalar.

Contudo, no cenário atual muitos atrasos ao atendimento vem ocorrendo de modo que alguns podem acarretar em uma piora do quadro clínico do paciente quando foi classificado na triagem para o momento da consulta com o médico. O acúmulo de atrasos podem refletir em grandes prejuízos para os pacientes que complicam o seu quadro e para a própria UPA que poderia evitar gastar com medicamentos ou com internação de um paciente.

## 1.1 Objetivos

### 1.1.1 Objetivo geral

Construir uma plataforma de gerenciamento de ações, visualização e análise de dados para auxiliar o gestor de uma unidade de saúde no controle de atendimentos de pacientes triados e classificados seguindo o protocolo manchester.

### 1.1.2 Objetivos específicos

1. Disponibilizar informações de atendimentos realizados na unidade.
2. Disponibilizar informações das triagens realizadas na unidade, seguindo as orientações do procolo Manchester.
3. Viabilizar interface de gerenciamento do atendimento ao paciente na sala de recepção.

## 1.2 Justificativa

A principal motivação para o desenvolvimento do trabalho é prover uma solução que ajude a amenizar os problemas que um gestor encontra ao administrar uma UPA. Um sistema automatizado dedicado as tarefas do dia a dia de uma UPA pode agilizar as atividades, armazenar informações úteis ao gestor, aos pacientes e a até mesmo a pesquisas em estudos na área da saúde. A página Dashboard será uma alternativa inteligente de análises das atividades realizadas na UPA, tais como, atendimentos, triagens, consumo de medicamentos, dentro outras e será capaz de a médio e longo prazo atuar no planejamento futuro

da unidade estimando demandas de insumos, detectando grupos atingidos por determinada doença e assim gerar estudos demográficos epidemiológicos dos pacientes e também atuar como assistente virtual em planejamentos logísticos e econômicos. Além do mais, o trabalho pode servir como um ponto de partida para uma série possibilidades de estudos relacionados a serviços praticados em Unidades de Saúde. Este último ponto é muito importante, pois existe ainda, no nosso país, uma escassez de trabalhos multidisciplinares englobando saúde e computação, que falem de temas como gestão e informatização de outros serviços em unidades de saúde.

## 1.3 Estrutura do texto

No Capítulo 2 apresenta-se a fundamentação teórica usada como referência neste trabalho. No Capítulo 3 são descritos os trabalhos relacionados encontrados. Descreve-se no capítulo 4 a metodologia executada no desenvolvimento do projeto. O Capítulo 5 mostra os resultados obtidos na construção da aplicação. Por último, no Capítulo 6 são dadas as conclusões e os possíveis trabalhos futuros.

# Capítulo 2

# Fundamentação Teórica

Este capítulo apresenta a base teórica relevante ao entendimento deste trabalho, explanando sobre o Protocolo Manchester, como ele funciona, por que é utilizado, quais as características de cada cor de classificação e quais os problemas na sua utilização aqui no Brasil. Apresenta ainda uma breve descrição dos conceitos e tecnologias necessários para construção desse trabalho.

## 2.1 Protocolo Manchester

Trata-se de um protocolo de seleção de pacientes adotado no mundo todo que foi aplicado pela primeira vez na cidade de Manchester, na Inglaterra, em 1997, por isso deu-se seu nome. Ele permite a identificação de prioridade e a definição do tempo alvo recomendado até a avaliação médica caso a caso. Para isso, o protocolo Manchester funciona com base em uma escala adotada pela instituição de saúde, geralmente, dividida em cores. Ao chegar à unidade, o paciente é examinado por um profissional de nível superior, Médico ou Enfermeiro que avalia seu quadro clínico geral, por meio da anamnese [1] e checagem dos sinais vitais. A classificação é feita de acordo com as queixas e sintomas e com os principais fatores, ou seja, aqueles que impactam o tempo em que o paciente pode esperar, tais como: risco de morte, escala de dor, hemorragia, nível de consciência, temperatura, glicemia, entre outros.

Segundo o artigo [Redec, 2019], os profissionais de saúde responsáveis por fazer a triagem devem realizar uma avaliação sobre o quadro clínico em que o paciente se encontra para colocar nele uma pulseira com a cor correspondente à gravidade do caso. Nesta etapa não se deve buscar um diagnóstico, somente a identificação do risco daquele quadro. Essa triagem é feita por um profissional de nível superior, Médico ou Enfermeiro, que possua boa comunicação, capacitação e conhecimento clínico. As cores estão associadas as seguintes características:

[1] A anamnese é uma entrevista conduzida pelo médico em consultório com o objetivo de identificar os sintomas do paciente e chegar ao diagnóstico de uma doença.

- **Vermelha:** Emergência

  É atribuída aos pacientes que se encontram em estado gravíssimo e com risco de morte, os quais necessitam de atendimento imediato, como quadros de queimadura em mais de 25% do corpo, problemas respiratórios, dor no peito relacionada à falta de ar, crises de convulsão, trauma cranioencefálico, tentativa de suicídio, parada cardiorrespiratória, hemorragias incontroláveis, entre outros.

- **Laranja:** Muito Urgente

  É atribuída aos casos considerados muito urgentes e com risco significativo de morte. O tempo de espera aproximado é de até 10 minutos. Abrange casos, como arritmia cardíaca sem apresentação de sinais de instabilidade, cefaleia intensa com rápida progressão, dores severas, etc.

- **Amarela:** Urgente

  É destinada aos casos urgentes de gravidade moderada com necessidade de atendimento médico, mas sem riscos imediatos. O tempo médio de espera é de até 60 minutos e classifica casos, como desmaios, dor moderada, vômito intenso, crises de pânico, hemorragia moderada, picos de hipertensão, alteração dos sinais vitais, entre outros quadros clínicos.

- **Verde:** Pouco Urgente

  Abrange casos considerados menos graves. O tempo de espera pode ser de até 2 horas e abrange pacientes com dores leves, torcicolo, enxaqueca, estado febril sem a presença de alterações vitais, resfriados e viroses, náuseas e tonturas, hemorragia controlada, asma não diagnosticada como quadro de crise, etc.

- **Azul:** Não Urgente

  Por último, a cor azul representa a classificação mais simples para casos que o paciente pode aguardar atendimento ou ser encaminhado para outra unidade de saúde. O tempo de espera pode ser de até 4 horas e envolve pacientes com queixas de dores crônicas, aplicação de medicação com receita, troca de sondas, entre outros.

Entretanto quando os prazos não são respeitados e os atrasos acontecem, o quadro clínico desses pacientes podem se agravar, sobretudo em casos de triados entre cor amarela e vermelha, o mínimo atraso de tempo ao atendimento é capaz de piorar o seu estado de saúde. O impacto de um cenário onde acontecem muitos atrasos pode ser muito grande, tendo em vista que mais e mais vezes o médico vai estar atendendo pacientes em estado de muito urgência, necessitando de internação e carecendo de medicação. Nesse caso a consequência se reflete em mais consumo de insumos na UPA e mais salas de internações ocupadas e isso significa em longo prazo em problemas de falta de medicamentos e também

falta de leitos de internações. Infelizmente, a demanda dos atendimentos é alta e muitas unidades de pronto atendimento não conseguem suprir as demandas, seja por corpo médico insuficiente, ou por falta de salas, ou por ter poucos equipamentos para determinados exames, ou até mesmo falta de capacidade de um gestor para conduzir e organizar as atividades.

## 2.2 Conceitos e Tecnologias para desenvolvimento

O desenvolvimento de um projeto WEB é uma tarefa complexa que engloba uma série de conceitos de programação e necessita que todos atuem bem em conjunto. No mundo da programação existem inúmeras linguagens de programação mas nenhuma é perfeita, cada uma tem suas vantagens e desvantagens. Por conta disso, não existe um consenso de melhor ou pior linguagem, mas sim, em qual tipo de aplicação as linguagens são mais indicadas para se utilizar. Por exemplo, para construção de backends de projetos WEB, o Python é a mais popular e mais utilizada. No entanto, para construção de frontends de sistema WEB, o JavaScript é a mais adequada, assim como Swift e Java são mais indicadas para aplicações Mobile, etc.

Este presente trabalho foi produzido utilizando-se de várias linguagens de programação, destacando-se a Python no backend e JavaScript no frontend, mas além das linguagens o trabalho não seria construído sem grandes conceitos vistos nas disciplinas de graduação do Curso de Engenharia de Computação, tais como, banco de dados, modelos relacionais, sistemas distribuídos, ciências de dados, probabilidade e estatística, interface homem máquina, etc.

# Capítulo 3

# Trabalhos Relacionados

Durante as pesquisas no processo de desenvolvimento deste trabalho foram encontrados alguns trabalhos com temáticas que fazem parte do tema do projeto. Dentre eles, dois se destacaram e contribuíram muito com ideias e informações que nortearam muitas funcionalidades implementadas na solução. Vale salientar que esta foi uma grande oportunidade de se trabalhar com auxílio de outras pesquisas vindas de profissionais de outras áreas que não são da computação.

## 3.1 Associação entre sinais vitais e Sistema de Triagem de Manchester: estudo observacional retrospectivo

Os autores desse artigo [Martins et al., 2017] são: José Carlos Amado Martins, Helisamara Mota Guedes, Cristiane Chaves Souza e Tânia Couto Machado Chianca. O artigo teve como objetivo avaliar a associação entre os sinais vitais coletados na entrada do paciente ao departamento de emergência e os níveis de risco do Sistema de Triagem de Manchester (STM), através de método observacional retrospectivo, cuja amostra foi de 154.714 pacientes. O artigo conseguiu produzir muitas informações estatísticas a respeito das triagens realizadas no serviço de urgência do Centro Hospitalar Universitário de Coimbra (CHUC), em Portugal. A partir delas, os autores montaram três tabelas.

1. Tabela 01: Queixas principais apresentadas pelos pacientes no momento da triagem. (Figura 3.1).

2. Tabela 02: Estatística descritiva das medições dos sinais vitais dos pacientes que compareceram ao pronto-socorro do cenário do estudo. (Figura 3.2).

3. Tabela 03: Média e desvio padrão de cada sinal vital avaliado nos pacientes classificados nos diferentes níveis de risco do Sistema de Triagem de Manchester e

| Queixa | (n) | % |
|---|---|---|
| Indisposição no adulto | 22.679 | 14,66 |
| Problemas nos membros | 18.810 | 12,16 |
| Problemas oftalmológicos | 12.150 | 7,85 |
| Obstetrícia/Ginecologia | 11.253 | 7,27 |
| Dispneia | 9.234 | 5,97 |
| Dor abdominal | 8.830 | 5,71 |
| Dor torácica | 6.780 | 4,38 |
| Dor lombar | 6.504 | 4,2 |
| Outros | 58.474 | 37,79 |
| Total | 154.714 | 100,00 |

Figura 3.1: Dados da pesquisa. (Fonte: autor)

| Sinal vital | Número de aferições | |
|---|---|---|
| | n | % |
| Dor | 119.625 | 77,32 |
| Temperatura (°C) | 54.404 | 35,16 |
| Frequência Cardíaca (bpm) | 49.227 | 31,82 |
| Glicemia (mg/dl) | 11.208 | 7,24 |
| Frequência respiratória | 1.357 | 0,87 |
| Pressão arterial (sistólica/ diastólica) | 1.230 | 0,79 |

Figura 3.2: Dados da pesquisa. (Fonte: autor)

associação entre os grupos de cores da classificação. (Figura 3.3).

Esse trabalho constrói uma serie de informações estatísticas a respeito da relação entre sinais vitais e classificações no protocolo manchester onde é possível criar uma base de dados de pacientes fictícios e aleatórios com classificações manchester e dados de sinais vitais coerentes com um cenário real.

## 3.2 Monitoramento da execução do Protocolo de Manchester: uma proposta tecnológica para o hospital universitário de Florianópolis

Os autores desse artigo [Laureano, 2019] são: Laureano, Guilherme dos Santos. Nesse artigo foi desenvolvido uma ferramenta capaz de monitorar a execução do Protocolo de Manchester no pronto atendimento do Hospital Universitário Polydoro Ernani de São Thiago. A solução do autor foi a construção de um hardware que foi posicionado estrategicamente no corredor de acesso da sala de recepção para os consultórios para fazer a leitura da posição de pulseiras colocadas em pacientes após a triagem através de raios in-

| Sinal Vital | | Nível de risco | | | | | Testes estatísticos | |
|---|---|---|---|---|---|---|---|---|
| | | Vermelho | Laranja | Amarelo | Verde | Azul | Teste de Levene | Kruskall Wallis |
| T1 (ºC) | Média | 36,42 | 36,85 | 36,63 | 36,53 | 36,27 | F = 497,6 | H = 406,7 |
| | DP | 1,30 | 1,21 | 0,78 | 0,57 | 0,42 | (p<0,0000) | (p<0,0000) |
| FR2 (irpm) | Média | 25,70 | 24,58 | 22,93 | 21,32 | — | F = 4,1 | H = 17,42 (p = |
| | | | | | | | (p=0,0064) | 0,001) |
| | DP | 9,60 | 9,39 | 8,58 | 8,07 | — | | |
| FC3 (bpm) | Média | 160,74 | 93,17 | 86,24 | 84,45 | 89,38 | F = 1677,4 | H = 508,6 |
| | | | | | | | (p<0,0000) | (p<0,0000) |
| | DP | 132,61 | 42,53 | 19,74 | 17,22 | 17,74 | | |
| GC4 (mg/dl) | Média | 252,08 | 193,89 | 149,39 | 151,31 | 121,85 | F = 39,5 | H = 307,64 |
| | DP | 266,25 | 168,40 | 102,87 | 85,23 | 43,78 | (p<0,0000) | (p<0,0000) |
| PAS5 | Média | 152,48 | 125,88 | 125,15 | 121,69 | 118,43 | F = 35,8 | H = 47,8 |
| (mmHg) | | | | | | | (p<0,0000) | (p<0,0000) |
| | DP | 54,35 | 28,87 | 20,52 | 17,64 | 10,86 | | |

Figura 3.3: Dados da pesquisa. (Fonte: autor)

fravermelhos. A partir das localizações dos pacientes em tempo real, foi possível registrar o momento em que eles saem da sala de espera para serem atendidos em um consultório e daí, a partir do registro do tempo em que ele começou a ser monitorado, é calculado o intervalo de tempo de espera do atendimento e se ele se enquadra em um caso de atraso ou não, em relação a cor da sua classificação. Os dados processados pelo dispositivos são enviados via cabo USB para um computador e são exibidos em terminal cmd.

O trabalho do Laureano teve um foco maior na construção do hardware de monitoramento e registro de informações, não foi desenvolvido uma ferramenta sofisticada para visualização e estudo dos resultados. O trabalho aqui proposto possui um foco maior na visualização dessas informações, através de um software moderno, com páginas Dashboard. Dessa forma, o usuário poderá visualizar melhor as informações de monitoramento dos atendimentos.

# Capítulo 4

# Metodologia

Neste capítulo é apresentada a metodologia deste trabalho. Inicia-se com a descrição do processo de desenvolvimento da proposta da plataforma. Em seguida, é demonstrada a infraestrutura do sistema, como as classes se relacionam no banco de dados e como foi construído o algoritmo gerador da base de dados. Prosseguindo, apresenta-se as tecnologias utilizadas para construir o sistema WEB, tais como, a escolha do framework Django para atuar no backend do sistema e Quasar para atuar no frontend.

Finaliza com a etapa em que o projeto passou de esboços e definições da interface e funcionalidades do sistema.

## 4.1 Processo de desenvolvimento

No início, o projeto passou por uma etapa de pesquisa por trabalhos relacionados ao escopo do projeto, na qual foram utilizadas as palavras chaves: manchester, triagem e computação. Destacou-se nessa etapa o artigo [Martins et al., 2017] , que futuramente foi de suma importância para a construção de um algoritmo capaz de gerar de uma base de dados coerente e diversificada para testes e visualizações do funcionamento do Dashboard do sistema. Após essa etapa dedicou-se um período para estudar e definir as funcionalidades do sistema, esboçar o diagrama de software (Figura 4.1), definir as as classes que irão compor o sistema e como elas se relacionarão no banco de dados. Por final, definiu-se quais seriam as tecnologias a serem utilizadas para construir o trabalho, pensando principalmente em usar frameworks atuais, de código aberto, comunidade ativa e também com alguma experiência de utilização.

## 4.2 Infraestrutura

A partir do estudo e elaboração das funcionalidades do sistema foi possível traçar as classes que abarcarão o backend. De modo a organizar a demanda das classes, foram

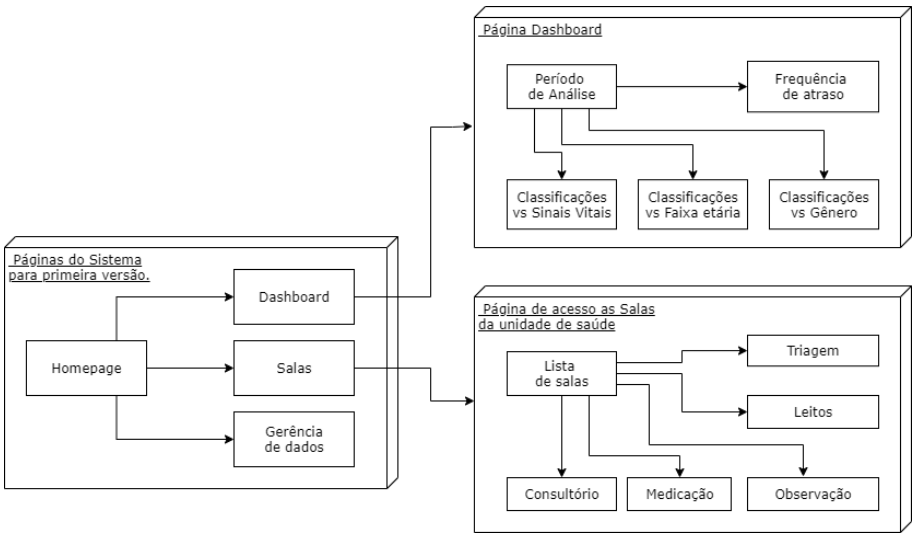


Figura 4.1: Diagrama de Software do sistema. (Fonte: autor.)

criadas três camadas (Figura 4.2), que relacionam-se entre si, sendo a primeira a camada *Comunidade* constituída de classes que representam as informações associadas a parte humana da aplicação, tais como, *Paciente*, *Endereço*, *Profissional*, *Cargo* e *Especialidade*. Seguindo, a próxima camada é a *Saúde*, que representa o coração do sistema, pois assina todas as classes associadas ao campo de saúde da aplicação, tais como, *Sintoma*, *Sinais Vitais*, *Relatório*, *Triagem*, *Sala* e *Centro Médico*. Por final, mas não menos importante, a *Temporizadores* é a camada dedicada a cuidar das classes que armazenam um ciclo temporal, com informações de data e hora, do início e/ou do final de atividades realizadas na unidade de saúde, como por exemplo o tempo de espera para ser chamado ao atendimento após o paciente realizar a triagem.

## 4.3 Base de dados

Às frequências de atrasos nos chamados ao atendimento e às classificações de níveis de urgência dadas aos pacientes, após a triagem, fazem parte do panorama inicial proposto de análise no Dashboard. Para isso é necessário que a base possua informações como, data e horário do momento em que foi realizado a triagem, assim como o registro temporal do momento em que o paciente foi chamado ao atendimento, dados dos sinais vitais do paciente registrados na triagem, lista de sintomas relatados, além de outras informações do paciente, como idade e sexo. Para que por meio delas seja possível traçar paralelos e fazer projeções, como identificar grupos de risco, demandas sazonais, deficiências técnicas da instituição, etc.

É importante saber que, na literatura atual, não existe um critério ou regra definida que aponte a classificação no protocolo manchester a um paciente de acordo com determinado valor de temperatura corporal, pressão arterial, ou índice de glicemia detectado

no momento da triagem. Tudo é feito com base na experiência e no *feeling* do profissional. Sendo assim, é possível que ocorram falhas na decisão de em qual nível de urgência de atendimento o paciente está. Por causa disso, o Dashboard proposto atuaria como uma ferramenta importante no estudo e visualização de como vem sendo feito essas classificações na clínica e possivelmente projetar alguns critérios a serem seguidos a fim de normalizar as decisões, até porque as triagens são feitas por profissionais diferentes, em períodos diferente do dia.

Prosseguindo, apesar da procura, não foi encontrada uma base de dados dentro dos moldes desejados e como solução foi proposto criar uma base de dados própria. Com grande ajuda do artigo [Martins et al., 2017], que busca avaliar a associação entre os sinais vitais coletados na entrada do paciente ao departamento de emergência e os níveis de risco do Sistema de Triagem de Manchester (STM), foi possível construir modelos de distribuição gaussiana na geração dos valores que abarcam a tabela da base de dados ao longo do Algoritmo.

### 4.3.1 Algoritmo gerador da base de dados de cargos e especialidades

O banco de dados da aplicação contempla uma tabela de profissionais de saúde, além de tabelas de cargos e especialidades que estão relacionadas entre si. Para alimentar essas tabelas foram retiradas informações da Classificação Brasileira de Ocupações, do site [MinistérioDaSaúde, ] e também do site [DadosGov, ]. Assim, criou-se uma planilha com essas informações e um código de leitura dessa planilha para dinamicamente criar e alocar no banco de dados as informações.

```
import pandas as pd
import random
from comunidade.models import *

df = pd.read_excel('area-saude.xlsx', header=None)

count = 0
for line in df.values:
    if count >= 1:
        cargo, _ = Cargo.objects.get_or_create(nome=line[1])
        if(len(str(line[1])) > 1):
            especialidade, _ = Especialidade.
                objects.get_or_create(nome=line[2])
    count += 1
```

### 4.3.2 Algoritmo gerador da base de dados de triagens

O artigo [Martins et al., 2017] utilizou-se de uma base de dados de um estudo observacional retrospectivo, realizado no serviço de urgência do Centro Hospitalar Universitário de Coimbra (CHUC), Portugal, com 154.714 pacientes entre os dias 01 de janeiro e 31 de dezembro do ano de 2012, triados por enfermeiro com uso do STM. Nele, foram ranqueados os sintomas mais relatados na triagem, quantitativos por gênero, faixa etária, classificações dadas, além de uma tabela construída com valores de média e desvio padrão de cada tipo de sinal vital avaliado nos pacientes classificados em diferentes níveis de risco do Sistema de Triagem de Manchester e associação entre os grupos de cores da classificação. As informações extraídas para a construção da base de dados foram:

- A maioria (56,43%) dos pacientes são do sexo feminino.
- As idades variando entre 0 e 112 anos (média: 53,65 + 21,18 desvio padrão).
- Quanto a prioridade clínica, os pacientes triados distribuem-se em:

| | |
|---|---|
| Azul | 6,54% |
| Verde | 29,16% |
| Amarelo | 54.21% |
| Laranja | 9,85% |
| Vermelho | 0.24% |

- Tabela com T1°C (temperatura), FR2 irmp (Frequência respiratória, em incursões respiratórias por minuto), FC3 bpm (Frequência cardíaca, em batimentos por minuto). GC4 mg/dl (Glicemia capilar, miligramas/decilitro), PAS5 (Pressão arterial sistólica, em milímetros de mercúrio).

O algoritmo foi desenvolvido na plataforma Google Colab Notebook [1], em linguagem Python [2], disponível em repositório no Github [3]. O código consiste essencialmente em incrementar dinamicamente, seguindo as informações citadas anteriormente, um Dicionário que será usado para converter em um DataFrame e assim, facilmente exportado para o formato CSV.

I - **A maioria (56,43%) dos pacientes são do sexo feminino.**

A primeira etapa declara:

- Um dicionário, apenas com a chave *Sex* que possui uma lista vazia.
- Tamanho que a base de dados portara, nesse caso será de tamanho 10 mil.

[1] https://colab.research.google.com/notebooks/
[2] https://www.python.org/
[3] https://github.com/wagnerfilho1995/Generator-Manchester-Classification

| Sinal Vital | | Nível de risco | | | | | Testes estatísticos | |
|---|---|---|---|---|---|---|---|---|
| | | Vermelho | Laranja | Amarelo | Verde | Azul | Teste de Levene | Kruskall Wallis |
| T1 (ºC) | Média | 36,42 | 36,85 | 36,63 | 36,53 | 36,27 | F = 497,6 | H = 406,7 |
| | DP | 1,30 | 1,21 | 0,78 | 0,57 | 0,42 | (p<0,0000) | (p<0,0000) |
| FR2 (irpm) | Média | 25,70 | 24,58 | 22,93 | 21,32 | — | F = 4,1 | H = 17,42 (p = |
| | | | | | | | (p=0,0064) | 0,001) |
| | DP | 9,60 | 9,39 | 8,58 | 8,07 | — | | |
| FC3 (bpm) | Média | 160,74 | 93,17 | 86,24 | 84,45 | 89,38 | F = 1677,4 | H = 508,6 |
| | | | | | | | (p<0,0000) | (p<0,0000) |
| | DP | 132,61 | 42,53 | 19,74 | 17,22 | 17,74 | | |
| GC4 (mg/dl) | Média | 252,08 | 193,89 | 149,39 | 151,31 | 121,85 | F = 39,5 | H = 307,64 |
| | DP | 266,25 | 168,40 | 102,87 | 85,23 | 43,78 | (p<0,0000) | (p<0,0000) |
| PAS5 | Média | 152,48 | 125,88 | 125,15 | 121,69 | 118,43 | F = 35,8 | H = 47,8 |
| (mmHg) | | | | | | | (p<0,0000) | (p<0,0000) |
| | DP | 54,35 | 28,87 | 20,52 | 17,64 | 10,86 | | |

Figura 4.2: Média e desvio padrão de cada sinal avaliado nos pacientes classificados nos diferentes níveis de urgência do Sistema de Triagem de Manchester e associação entre os grupos de cores da classificação. (Fonte: [Martins et al., 2017].)

- Quantidade de pessoas do sexo feminino e do sexo masculino de acordo com a informação extraída.

A partir daí, segue dois simples loopings iterando sobre as quantidades definidas e incrementando a lista no dicionário com os caracteres $F$, para feminino e $M$ para pessoas do sexo masculino. Ao final, é executada a função *shuffle* para embaralhar a lista de caracteres.

```
import random

dict_pessoas = {
  'Sex': []
}

num_pessoas = 10000

qtd_feminino = int(0.56 * num_pessoas)
qtd_masculino = int(0.44 * num_pessoas)

for i in range(qtd_feminino):
  dict_pessoas['Sex'].append('F')

for i in range(qtd_masculino):
  dict_pessoas['Sex'].append('M')

random.shuffle(dict_pessoas['Sex'])
```

II - **As idades variando entre 0 e 112 anos (média: 53,65 + 21,18 desvio padrão).**

Dispondo dessas informações é possível construir um modelo probabilístico de distribuição normal que faz com que a curva de dispersão das idades geradas seja satisfatória. Nesse trecho, o código cria a chave *Data_nascimento* e a inicia com uma lista vazia, além disso, é utilizada a função randn, da biblioteca *random*, onde cria-se uma lista de tamanho 10 mil, preenchida com valores flutuantes aleatórios amostradas a partir de uma distribuição uniforme Gaussiana de média 0 e variância 1. Os valores dessa lista são iterados em um *looping* e utilizados junto com a média e desvio padrão informados anteriormente para calcular um valor de idade. Além disso, em cada iteração é preciso converter a idade em uma data, para ser salva como data de nascimento, portanto, optou-se por subtrair a idade calculada pelo ano atual (2021), dado ano, o dia e o mês para essa data nascimento foram gerando aleatoriamente. Ao final de cada iteração a data calculada é adicionada na lista do dicionario, na chave *Data_nascimento*.

```
dict_pessoas['Data_nascimento'] = []
values = randn(num_pessoas)

for val in values:
  scaled_value = int(53.65 + val * 21.18)
  mes = random.randint(1, 12)
  dia = random.randint(1, 29)
  if mes == 2 and dia > 28:
    dia = 28
  ano = datetime.now().year - scaled_value
  data_nascimento = date(ano, mes, dia)
  dict_pessoas['Data_nascimento'].append(data_nascimento)
```

III - **Quanto à prioridade clínica, 54,21% dos pacientes foram triados como amarelo, 29,16% como verde, 9,85% como laranja, 6,54% como azul e 0,25% como vermelho.**

Nessa etapa do algoritmo é possível utilizar a função *choice* da biblioteca python, que retorna uma lista gerada após sortear N vezes a escolha de valores de um conjunto de resposta. É possível inserir como parâmetro na função a probabilidade de sortear cada valor desse conjunto. Ou seja, o dicionário do algoritmo, na nova chave *manchester*, recebe uma lista, de tamanho 10 mil, onde valores são strings com nomes das cores das classificações do protocolo manchester.

```
dict_pessoas['manchester'] = np.random.choice(
  ['blue', 'green', 'yellow', 'orange', 'red'],
  num_pessoas,
  p=[0.0654, 0.2916, 0.5421, 0.0985, 0.0024]
)
```

IV - **Tabela de informações dos Sinais Vitais**

Nessa etapa do algoritmo optou-se por construir a tabela de informações (Figura 4.2) em forma de dicionário para facilitar a leitura e atribuições durante as iterações dos loopings. De forma que, cada coluna de cor será uma chave no dicionário, e cada uma dessas possuem outro dicionário interno, com chaves nomeadas de acordo com os nomes dos sinais vitais dados na tabela. Os valores internos nas chaves de sinais vitais são o resultado do cruzamento entre cor e sinal vital, dado por uma dupla, onde o primeiro valor representa a média e o segundo valor representa o desvio padrão.

```
    sinais_vitais = {
    'red': {
        'temperatura': [36.42, 1.30],
        'frequencia_respiratoria': [25.70, 9.60],
        'frequencia_cardiaca': [160.74, 132.61],
        'glicemia_capilar': [252.08, 266.25],
        'pressao_sistolica': [152.48, 54.35]
    },
    'orange': {
        'temperatura': [36.85, 1.21],
        'frequencia_respiratoria': [24.58, 9.39],
        'frequencia_cardiaca': [93.17,42.53],
        'glicemia_capilar': [193.89, 168.40],
        'pressao_sistolica': [125.88, 28.87]
    },
    'yellow': {
        'temperatura': [36.63, 0.78],
        'frequencia_respiratoria': [22.93, 8.58],
        'frequencia_cardiaca': [86.24, 19.74],
        'glicemia_capilar': [149.39, 102.87],
        'pressao_sistolica': [125.15, 20.52]
    },
    'green': {
        'temperatura': [36.53, 0.57],
        'frequencia_respiratoria': [21.32, 8.07],
        'frequencia_cardiaca': [84.45, 17.22],
        'glicemia_capilar': [151.31, 85.23],
        'pressao_sistolica': [121.69, 17.64]
    },
    'blue': {
        'temperatura': [36.53, 0.57],
        'frequencia_respiratoria': [21.32, 8.07],
        'frequencia_cardiaca': [84.45, 17.22],
        'glicemia_capilar': [151.31, 85.23],
        'pressao_sistolica': [121.69, 17.64]
    },
}
```

A partir daí, o código prossegue declarando uma chave para cada tipo de sinal vital,

originando-as com listas vazias. Além delas, as chaves *Prioridade* e *Espera (mins)* também foram criadas, iniciando-as com lista vazia, para facilitar as atribuições quanto ao nível de urgência e o tempo em minutos que cada um deve esperar de acordo com esse nível. Para auxiliar nas atribuições dentro do loopings dois arrays foram declarados também, um contendo os termos em *String* dos possíveis níveis de urgência que serão atribuídos e o outro contendo os valores em minutos que cada, respectivamente, cada nível estará relacionado.

```
    dict_pessoas['Temperatura (°C)'] = []
    dict_pessoas['Frequencia_respiratoria (irpm)'] = []
    dict_pessoas['Frequencia_cardiaca (bpm)'] = []
    dict_pessoas['Glicemia_capilar (mg/dl)'] = []
    dict_pessoas['Pressao_sistolica (mmHg)'] = []

    dict_pessoas['Prioridade'] = []
    dict_pessoas['Espera (mins)'] = []

    prioridade = ['Não urgente', 'Pouco urgente', 'Urgente',
        'Muito urgente', 'Emergencia']
    espera = [240, 120, 50, 10, 0]
```

Continuando, o looping de leitura da tabela vai iterar sobre o próprio dicionário final, na chave *manchester*, a qual foi preenchida na etapa anterior, ou seja, serão 10 mil iterações. Dentro do looping, novamente, se utilizara da fórmula para modelos de distruibuição normal, onde serão usados valores da lista *values* gerando em seções anteriores, com a função randn, além dos valores que média e desvio padrão dados na tabela de informações. Para iterar sobre essa lista de valores, foi criado uma variável $i$ inicia em zero que será incrementado a cada loop. Os valores gerados são arrendondados para duas casas decimais antes de cada atribuição.

```
i = 0
for cor in dict_pessoas['manchester']:
  c = sinais_vitais[cor]['temperatura'][0] + values[i]
  * sinais_vitais[cor]['temperatura'][1]
  dict_pessoas['Temperatura (°C)'].append(round(c, 2))

  irpm = sinais_vitais[cor]['frequencia_respiratoria'][0] +
  values[i] * sinais_vitais[cor]['frequencia_respiratoria'][1]
  dict_pessoas['Frequencia_respiratoria (irpm)']
    .append(round(irpm, 2))

  bpm = sinais_vitais[cor]['frequencia_cardiaca'][0] +
  values[i] * sinais_vitais[cor]['frequencia_cardiaca'][1]
  dict_pessoas['Frequencia_cardiaca (bpm)'].append(round(bpm, 2))

  mgdl = sinais_vitais[cor]['glicemia_capilar'][0] +
  values[i] * sinais_vitais[cor]['glicemia_capilar'][1]
  dict_pessoas['Glicemia_capilar (mg/dl)'].append(round(mgdl, 2))

  mmhg = sinais_vitais[cor]['pressao_sistolica'][0] +
  values[i] * sinais_vitais[cor]['pressao_sistolica'][1]
  dict_pessoas['Pressao_sistolica (mmHg)'].append(round(mmhg, 2))

  i += 1

  if cor == 'blue':
    dict_pessoas['Prioridade'].append(prioridade[0])
    dict_pessoas['Espera (mins)'].append(espera[0])
  elif cor == 'green':
    dict_pessoas['Prioridade'].append(prioridade[1])
    dict_pessoas['Espera (mins)'].append(espera[1])
  elif cor == 'yellow':
    dict_pessoas['Prioridade'].append(prioridade[2])
    dict_pessoas['Espera (mins)'].append(espera[2])
  elif cor == 'orange':
    dict_pessoas['Prioridade'].append(prioridade[3])
    dict_pessoas['Espera (mins)'].append(espera[3])
  elif cor == 'red':
    dict_pessoas['Prioridade'].append(prioridade[4])
    dict_pessoas['Espera (mins)'].append(espera[4])
```

V - **Numeração SUS**

Posteriormente, para auxiliar na identificação do usuário no sistema e sem comprometer com informações pessoais, como CPF ou Nome Completo, optou-se por adicionar uma última coluna na tabela final, produzindo a chave *SUS* iniciada com uma lista vazia. essa coluna representa o número na carteirinha do nosso Sistema Único de Saúde, o qual é sempre pedido e utilizado nos atendimentos, ele possui um total de 15 dígitos. O looping deve iterar 10 mil vezes, gerando e atribuindo a string em cada volta. É frequente que nos 3 primeiros dígitos desse cartão tenhamos a participação do algarismo zero, portando, deu-se preferência, ao gerar randomicamente os três primeiros dígitos com uma probabilidade maior de conter números zeros e os demais dígitos restantes foram gerados aleatoriamente, com mesma probabilidade para cada algarismo.

```
dict_pessoas['SUS'] = []

for i in range(num_pessoas):
  meu_sus = ''
  # bloco de 3 dígitos
  bloco = np.random.choice(
    ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'],
    3,
    p=[0.5, 0.1, 0.1, 0.1, 0.1, 0.1, 0, 0, 0, 0]
  )
  meu_sus += ''.join(bloco)
  meu_sus += ' '
  for j in range(3):
    # blocos de 4 dígitos
    bloco = np.random.choice(
      ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'],
      4,
      p=[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
    )
    meu_sus += ''.join(bloco)
    meu_sus += ' '
  dict_pessoas['SUS'].append(meu_sus)
```

VI - **Exportação do Resultado**

Ao final do algoritmo, é feita a exportação para uma planilha excel que será utilizada pelo backend do sistema para alimentação do banco de dados. O dicionário final

é convertido para formato DataFrame e assim utilizada a função *to_excel* para exportar o arquivo.

```
df = pd.DataFrame(dict_pessoas)
file_name = 'TriagemData.xlsx'
df.to_excel(file_name)
```

## 4.4 Tecnologias utilizadas na construção do Sistema WEB

WEB framework é um conjunto de componentes que ajuda a desenvolver sites de forma mais rápida e fácil, eles oferecem inúmeras funcionalidades, desde a conexão com banco de dados até gerar automaticamente a interface que vai ser utilizada para gerenciar o site, de forma a evitar que os programadores precisem reinventar a roda tendo trabalho extra e gastando tempo criando esses componentes. Por essa razão, o sistema usufruiu de uma série de frameworks no seu desenvolvimento, destacando-se no backend o framework Django e no frontend o Quasar Framework.

### 4.4.1 Backend

I - **Django**

Django [4] é um *framework* para aplicações WEB, gratuito e de código aberto, escrito em Python que permite desenvolvimento rápido de sites seguros e de fácil manutenção. Atualmente é um dos *frameworks* mais utilizados do mercado. Tem como pontos positivos uma comunidade próspera e ativa, com profissionais que estão sempre dispostos a trocar ideias e compartilhar soluções que já usaram para resolver problemas relacionados ao Django, ótima documentação, muitas opções de suporte gratuito e pago, além de oferecer uma interface administrativa, que torna mais simples o gerenciamento de todo o conteúdo. Acrescentando a esses fatores a escolha do Django para atuação no backend do sistema foi feita por familiaridade, tendo feito com ele em alguns trabalhos anteriormente.

O Django possui uma arquitetura que segue o padrão de projeto MTV *(Model, Template, View)*. Para que ele funcione, todas essas camadas estão interligadas e conversam entre si. Ou seja, há interdependência entre as partes para que a ação solicitada pelo usuário seja posta em prática. Ainda assim, um aspecto importante sobre o padrão MTV é a separação das responsabilidades de uma aplicação em camadas, o que torna a leitura do código mais legível e ainda melhor a organização. A

[4] https://www.djangoproject.com/

camada *Model* é responsável pelo gerenciamento de dados, a *Template* por gerenciar as entradas e saídas e por fim, a camada *View* é responsável por processar as requisições vindas dos usuários, formar uma resposta e enviá-la de volta ao usuário, é nela que residem as regras de negócio (Figura 4.3). Vale apontar que essa arquitetura é bastante parecida e muitas vezes até mesmo tratada como equivalente ao padrão MVC *(Model, View, Controller)* utilizado por diversos frameworks. O projeto Django desse trabalho foi criado seguindo as orientações do artigo [Girls, 2021].

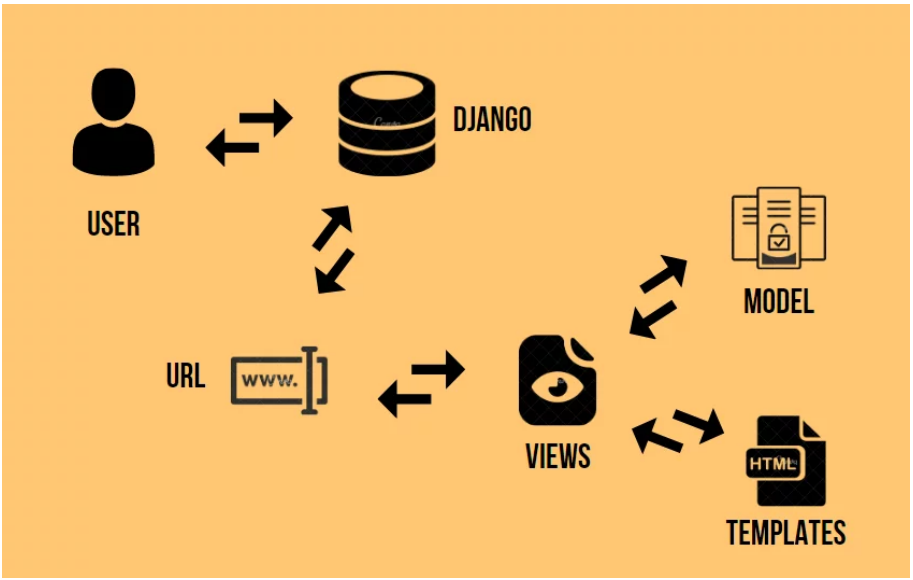


Figura 4.3: Arquitetura do Django em padrão MTV. (Fonte: techvidvan.com)

II - **Django REST Framework**

Diante dos dados modelados e gerenciados pelo Django é necessário que exista uma ponte por onde as informações das requisisões de backend e frontend possam transitar, para isso, torna-se necessária a criação de uma *API (Application Programming Interfaces)*. O Django Rest Framework [5] é uma biblioteca para o Framework Django que disponibiliza funcionalidades para implementar APIs Rest de forma extremamente rápida. Uma *REST (Representational State Transfer)* API é o conjunto de boas práticas utilizadas nas requisições HTTP realizadas por uma API em uma aplicação web. Uma das vantagens ao utilizar um site com REST API é a facilidade de comunicação com outras aplicações. As interfaces permitem adicionar funcionalidades ou informações ao site de forma simples, rápida e segura.

III - **Swagger**

Por já ter familiriade e para facilitar a construção das documentações e requisições da api, utilizou-se o Swagger que é um framework composto por diversas ferramentas que, independente da linguagem, auxilia a descrição, consumo e visualização de

[5] https://www.django-rest-framework.org/

serviços de uma API REST. Com o Swagger UI, a partir da especificação da API, podemos criar documentações elegantes e acessíveis ao usuário, permitindo assim uma compreensão maior da API, pois além de poder ver os endpoints e modelos das entidades com seus atributos e respectivos tipos, o módulo de UI possibilita que os usuários da API interajam intuitivamente com a API usando uma sandbox.

### 4.4.2 Frontend

Neste trabalho, o Vue.js foi escolhido para ser utilizado em conjunto ao Quasar framework na construção das interfaces gráficas do sistema, tanto por familiaridade com as tecnologias, quanto pela praticidade e grande biblioteca de componentes que agilizam e tornam toda aplicação responsiva a diversas plataformas.

I - **VueJs**

O Vue.js [6] (comumente conhecido como Vue) é um framework, feito em JavaScript, de código aberto para a construção de interfaces de usuário. A integração em projetos que usam outras bibliotecas de JavaScript é facilitada com o Vue pois ele foi projetado para ser adotado de forma incremental. O Vue também pode funcionar como uma estrutura de aplicativos web capaz de alimentar aplicativos avançados de uma única página.

II - **Quasar Framework**

Por conta dessa característica incremental do Vue surgiram muitos ”filhos”, sendo um deles o próprio Quasar Framework [7]. Ele atua como uma poderosa ferramenta para desenvolvimento híbrido com um incrível suporte a criação de aplicações para múltiplas plataformas, como Web, Mobile e Desktop. Outro ponto positivo é a estrutura UI (*User Interface*) baseada em *Material Design* responsivo com muitas bibliotecas de componentes pré-moldados que Quasar fornece ao desenvolvedor, sendo eles constantemente atualizados e incrementados pela comunidade. Sua documentação é organizada e de fácil leitura. A construção do projeto quasar desse trabalho segue as orientações do artigo [with Danny, 2019].

## 4.5 Interface do sistema

O Quasar fornece uma série de layouts que podem ser utilizados para dar uma cara inicial ao seu sistema, ajudando a começar mais rápido a construção do frontend. Dentre os modelos encontra-se layouts de sites conhecidos, como Youtube, Whatsapp Web, Google Photos, Google play e Github. O layout do Google photos foi escolhido e ele atuará como

[6] https://vuejs.org/
[7] https://quasar.dev/

esqueleto base de todo o sistema (Figura 4.4). A partir daí, remove-se, modifica-se e incrementa componentes afim de moldar o template para atender as funcionalidades do sistema. Tanto a barra superior (*AppBar*) quando a coluna na lateral esquerda, serão elementos sempre visíveis ao usuário, estando ele em qualquer página do sistema. A priori, na primeira versão entregue desse trabalho, a AppBar não vai implementar barras de busca nem sistema de login, portanto, remove-se os elementos contidos nela, deixando apenas um texto para título do sistema. Na coluna lateral, os ícones funcionam como botões que redirecionam o usuário para outra página. As páginas em questão, para essa primeira versão, serão a página Dashboard, a página de Dados e a das Salas do sistema. Portanto, é necessário trocar os ícones para fazer parte dos contextos e também os nomes abaixo de cada um. Também optou-se por colocar um botão e o início de redirecionamento para a página inicial.

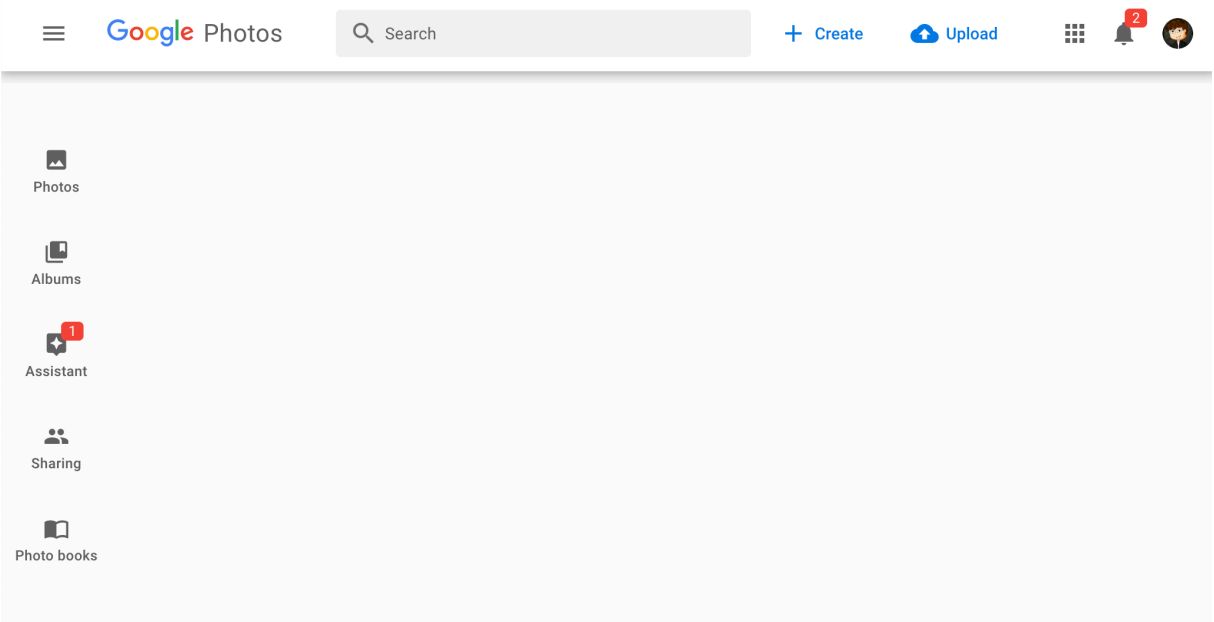


Figura 4.4: Google Photos Layout contruído com componentes do Quasar Framework. (Fonte: https://quasar.dev/layout/gallery/google-photos)

### 4.5.1 Página inicial

A página inicial, a priori, possui um desenho simples, com poucas funcionalidades, sendo elas:

- Lista de paciente triados e classificados aguardando atendimento.

  Nessa lista é importante destacar no paciente a quanto tempo ele está esperando o atendimento e sua respectiva cor de classificação, que representa o nível clínico de urgência que ele se encontra.

- Botão ”Nova Triagem”de redirecionamento para página de cadastro de nova triagem.

### 4.5.2 Dashboard

A página Dashboard passou por uma etapa de pesquisa por exemplos de sistemas que usam desse recurso, listou-se alguns e a partir daí foi possível esboçar uma layout inicial.

I - **Período de análise**

O modelo contém, no topo da página, uma entrada inicial para usuário inserir o período que se deseja analisar as informações, por meio de um intervalo de datas. O componente é construído com componentes do Quasar framework, e as entradas capturadas são tratadas em JavaScript para serem enviadas, em formato nativo *Date*, para o backend através de um endpoint POST da API. O Django recebe a requisição vinda da API em sua camada *View* e através da função associada ao endpoint, em sua camada *Controller*, faz a filtragem no banco de dados para retornar apenas os dados no intervalo desejado.

II - **Análise de atraso**

Ao inserir o período de análise, o frontend realiza uma operação GET na API que envia para o backend a informação das duas datas que formam o intervalo. Todas as operações da API de procurar, filtrar, manipular e modelar os dados foram construídas no backend individualmente dedicados a atender a necessidade de cada tipo de gráfico produzido no Dashboard. Dentro das Classes Viewsets, no backend, são criados *WEB Services* chamados de *Endpoints* que são as URLs por onde os serviços são acessados de uma aplicação cliente (Figura 4.5).

O Django interpreta os sinais da API, por meio dos *Endpoints*, e é redirecionado para a Classe Viewset onde as funções desses micro serviços estão localizadas. A partir daí, as funções trabalham para modelar os dados de objetos da classe *Timer*, filtrando pelos que possuem datas de entrada e saída no intervalo de tempo dado. Em seguida, declara-se um dicionário vazio aciona um looping que itera sob cada um dos objetos registrando a cada volta uma série de informações no dicionário. Para essa ocasião de análise de atrasos, é importante calcular o tempo em minutos da diferença entre horário de entrada na triagem e horário de saída da sala de espera. Essa diferença representa o tempo que o paciente levou para ser atendido e com ela será possível comparar com o tempo em que, baseado no sua classificação no protocolo manchester, ele deveria ter sido chamado, ou seja, ocorrendo atraso ou não. Caso seja um caso de chamado de atraso, o algoritmo registra o quanto em minutos foi o atraso, para que com essa informação seja possível traçar faixas de frequências dessas ocorrências. As faixas de frequências de atrasos são definidas para dentro de intervalos de 5 em 5 minutos. Ao final, a API deve retornar para o frontend todo o resultado do algoritmo em formato JSON.

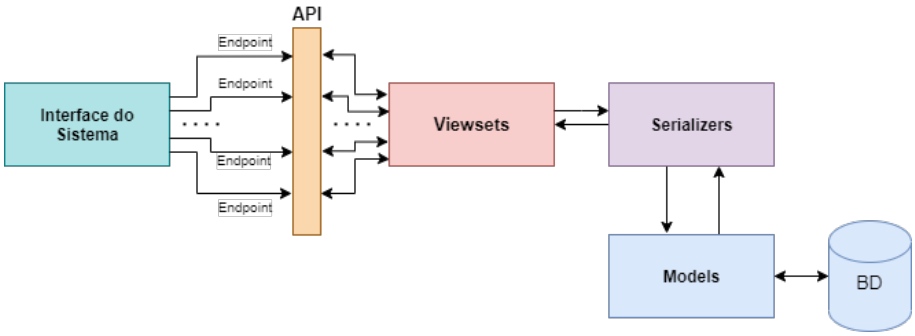


Figura 4.5: Transição das informação entre frontend e backend. (Fonte: autor)

III - **Análise de classificações**

Outra abordagem do Dashboard é a de buscar uma visão geral do cenário das cores atribuídas na triagem em relação a algumas variáveis. Dessa forma, com os dados disponíveis era possível relacionar as classificações em relação a: Faixas etárias, sexo e sinais vitais. Sendo assim, é possível identificar, por exemplo, grupos de riscos que estão sendo atingidos por alguma doença na região dado a alta frequência de casos sendo triados em maiores níveis de urgência. Também é possível buscar uma relação entre faixas de temperatura, pressão ou outros sinais vitais registrados no momento da triagem, com as classificações dadas e assim, a partir de um grande histórico definir regras e padrões para definir qual cor deve ser classificado o paciente que apresentar tal medida de tal sinal vital.

IV - **Gráficos**

O frontend da aplicação é construído em VueJs por meio do Quasar Framework, eles trabalham com linguagem JavaScript [8], portanto, para elaborar os gráficos pesquisou-se quais as bibliotecas nessa linguagem trabalham com esses recursos. Existe uma gama de bibliotecas mas dentre elas optou-se pela Chart.js [9] por ser uma biblioteca simples, leve, open source e possui uma boa documentação e uma grande e ativa comunidade. A Chart.js possui todos os tipos de gráficos necessários para o dashboard, dentre eles envolvem, gráficos de linhas, de pontos, Doughnut e de Barras.

### 4.5.3 Dados

Essa página tem como objetivo apenas exibir, em forma de tabela, todos os dados cadastrados no banco de dados da aplicação. O Quasar framework possui componentes em formato de Tabela. É necessário realizar um GET na api através de um

[8]https://www.javascript.com/
[9]https://www.chartjs.org/

*Endpoint* responsável por retornar os objetos da Classe Timer. O próprio componente Q-Table do Quasar possui, por padrão, a capacidade de paginação dos dados, assim, apesar da grande quantidade de objetos guardados no banco, a tabela vai exibir as informações de 5 em 5 linhas, sendo possível variar essa quantidade por página (Figura 4.6).

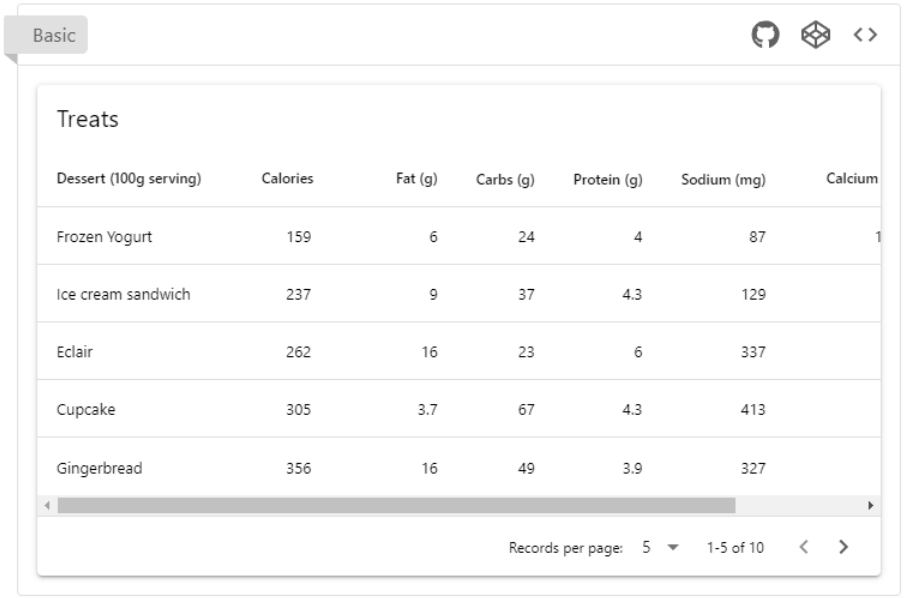


Figura 4.6: Exemplo de componente Table do Quasar Framework (Fonte: https://quasar.dev/vue-components/table)

### 4.5.4 Salas

O sistema gerencia e faz o controle das atividades realizadas na unidade de saúde. Em sua estrutura existem Classes chamadas Salas, que fazem o papel de representar o espaço físico na unidade, como sabemos, cada uma tem um objetivo, como por exemplo a Sala de Triagem, para realizar as triagens dos pacientes, a Sala de Espera, onde ficam as pessoas triadas aguardando atendimento, o Consultório, que é uma sala onde acontece o atendimento médico e aí nesse caso, uma unidade de saúde pode ter um ou vários consultórios. Isso posto, essa página tem como objetivo exibir uma lista com todas as salas do centro médico, onde cada item da lista é um link de acesso a página específica dedicada a essa sala.

A lista é construída utilizando o componente *Q-List* [10] e outros elementos do Quasar e através de um *Endpoint* que faz uma requisição to tipo GET para buscar no backend uma lista com todas as salas, cada uma com seu ID de identificação, tipo de sala (Espera, Triagem, Consultório, Leito, Observação, Medicamento, Internação,

[10] https://quasar.dev/vue-components/list-and-list-items

Amarela ou Vermelha). Cada elemento da q-list, os *Q-List-Item*, são elementos clicáveis e usam o ID de cada sala para compor a URL da página que o usuário é redirecionado ao clicar nela. Daí a página para onde ele for terá conteúdo dedicado aquela sala de determinado ID descrito na URL. A priori, a versão do trabalho desenvolveu apenas as Salas de Triagem e de Espera.

### 4.5.5 Sala de Espera

A sala de espera já foi descrita nesse documento pois ela é a própria página inicial do sistema, que contém a lista de pacientes triados aguardando atendimento e um botão de acesso a página da Sala de Triagem.

### 4.5.6 Sala de Triagem

A página de Sala de Triagem contém uma formulário para enviar ao backend e cadastrar no sistema um novo objeto do tipo Triagem. Essa etapa exige uma pesquisa de quais são as informações pedidas pacientes e quais são os sinais vitais medidos na examinação. Nem todas as triagem levantam os mesmos dados, sobretudo os dados relacionados ao Sinais Vitais, por problemas de falta de equipamento, mas no geral, as informações retiradas nesse processo são:

- Paciente: Nome, Sexo, Data de nascimento e Número do cartão do SUS do paciente.
- Sinais Vitais: Frequências Cardíaca e Respiratória, temperatura, pressão sistólica, Hemoglicoteste (HGT) e teste de Glicemia capilar.
- Além disso, pergunta-se quais sintomas o paciente esta sentindo a partir de uma lista pré-definida de sintomas cadastrados no sistema.

Ao final, após todas essas informações serem levantadas o profissional que realizou a triagem anota do formulário qual classificação dentro do padrão do protocolo Manchester o paciente deve estar classificado e finaliza a operação. Ao clicar no botão de finalizar triagem, o frontend realiza uma operação do tipo POST na API acionando o *Endpoint* dedicado a enviar a enviar o pacote com as informações que serão tratadas no backend. A partir daí, as informações do pacote são utilizadas para criar classes que compõem um Objeto da classe Triagem e também, se o usuário nao possuir cadastro, criar um Objeto da Classe Paciente. Ao final, também cria-se um objeto da Classe Timer, para associar essa triagem, a data do momento em que foi feita essa operação e o Paciente envolvido para que dessa forma detenhamos registro do horário que o paciente começou a aguardar seu atendimento. No momento em

que ele for chamado ao atendimento, esse objeto Timer é chamado novamente para atribuir o horário de saída.

# Capítulo 5

# Resultados

Neste capítulo são apresentados os resultados deste trabalho. Inicia-se com a consolidação da construção da infraestrutura do sistema, o banco de dados definido e todo desenvolvimento do backend com o Django Framework. Logo após, apresenta-se o resultado do algoritmo gerador da base de dados utilizada no projeto. Em seguida é demonstrado a concepção do frontend, sua estrutura definida com componentes do Quasar por meio de imagens das páginas do sistema e suas funcionalidades implementadas.

## 5.1 Backend

O Django possui um utilitário que gera automaticamente a estrutura básica de diretório de uma aplicação, dando liberdade para o desenvolvedor focar apenas em escrever código em vez de ficar criando diretórios. A infraestrutura deste trabalho foi composta de 3 domínios que interagem entre si (Figura 5.1). Cada domínio foi construído a partir do comando *startapp* do Django que cria o diretório com arquivos e suas configurações internas automaticamente. As classes são criadas no arquivos Models.py dos respectivos Apps e estão definidas seguindo o Diagrama UML da figura 5.2.

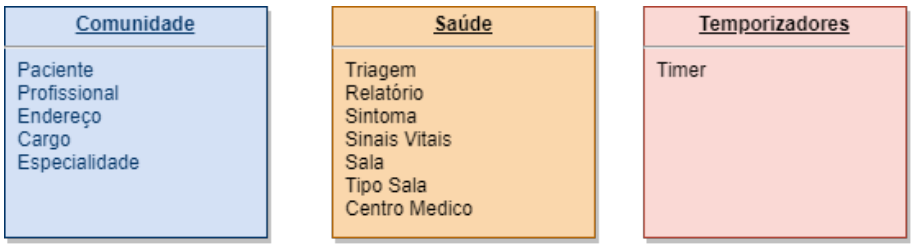


Figura 5.1: Domínios da aplicação (Fonte: autor)

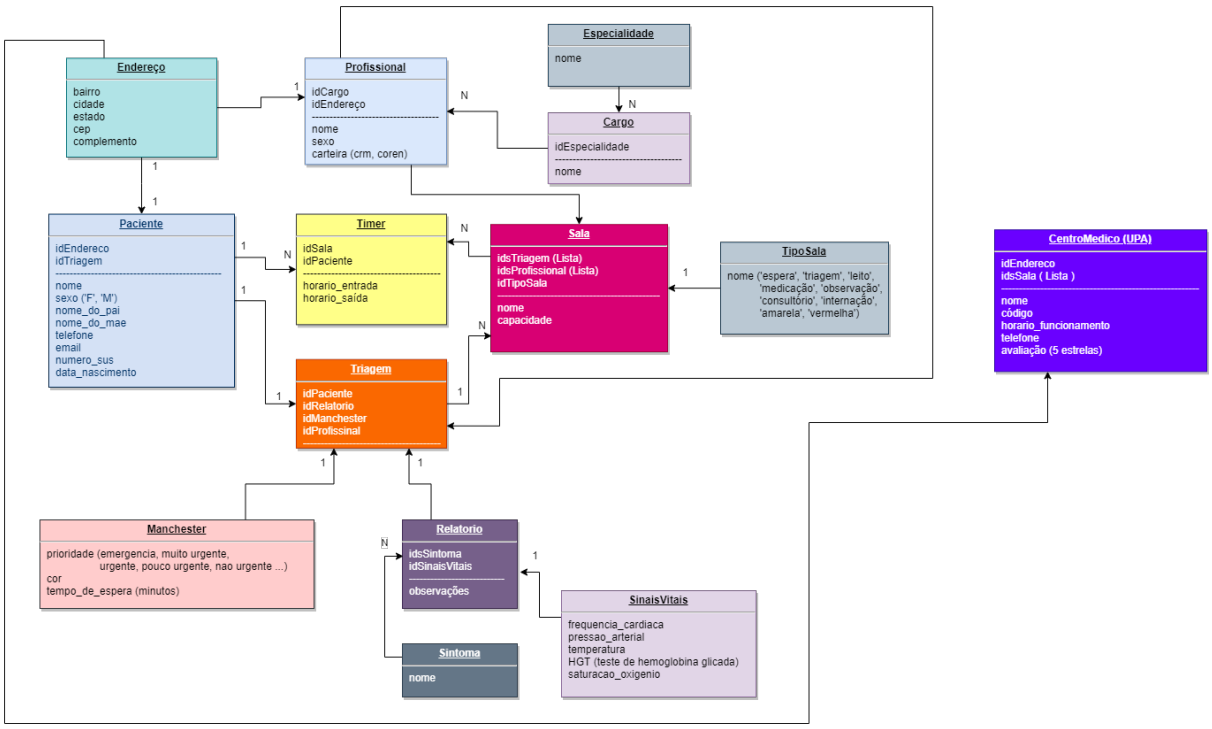


Figura 5.2: Diagrama UML das classes do Sistema (Fonte: autor)

### 5.1.1 Base de Dados

Durante algumas etapas do algoritmo foram utilizados modelos de Distribuição Gaussiana na geração de dados aleatórios, como por exemplo na etapa de faixas etárias em que a média era de 53.65 anos e o desvio padrão era de 21.68 (Figura 5.3). Em seguida, na etapa de definição das cores relacionadas a prioridade de urgência dos pacientes, onde 54,21% dos pacientes foram triados como amarelo, 29,16% como verde, 9,85% como laranja, 6,54% como azul e 0,25% como vermelho tal como pode ser visto na Figura 5.4. Ao final do algoritmo é gerado um Dataframe (Figura 5.5) que é exportado como planilha para alimentar o banco de dados do sistema.

### 5.1.2 Interface API

A API foi construída por meio do Django REST Framework e sua interface por meio do Swagger, onde é possível realizar testes e identificar rapidamente todos os endpoints dos micro serviços do sistema (Figura 5.7). Ao clicar em uma das camadas, por exemplo na camada Timers, será possíver ver a lista de Endpoints associados aquela camada, onde os serviços CRUD da classe Timer, tais como GET, POST, PUT e DELETE, são criados automaticamente e os outros endpoints com tarefas específicas, foram criados manualmente no backend.

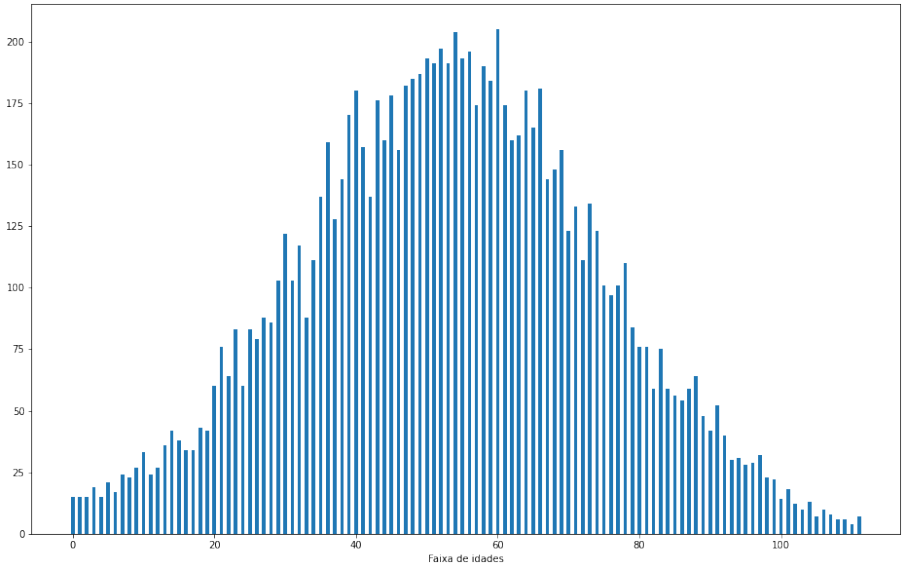


Figura 5.3: Distribuição Normal (Gaussiana) das idades (Fonte: autor)


Figura 5.4: Classificações Manchester (Fonte: autor)

| | Sex | Data_nascimento | manchester | Temperatura (°C) | Frequencia_respiratoria (irpm) | Frequencia_cardiaca (bpm) | Glicemia_capilar (mg/dl) | Pressao_sistolica (mmHg) | Prioridade | Espera (mins) | SUS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | F | 1933-12-21 | yellow | 37.90 | 36.87 | 118.30 | 316.49 | 158.48 | Urgente | 50 | 020 6625 8350 5373 |
| **1** | M | 1981-05-10 | blue | 36.18 | 16.38 | 73.92 | 99.17 | 110.90 | Não urgente | 240 | 115 3717 3311 4856 |
| **2** | F | 1979-01-21 | orange | 36.21 | 19.62 | 70.71 | 104.95 | 110.63 | Muito urgente | 10 | 332 1454 7231 4556 |
| **3** | F | 1991-04-04 | green | 35.92 | 12.66 | 65.97 | 59.86 | 102.76 | Pouco urgente | 120 | 000 3457 3985 4937 |
| **4** | M | 1950-04-27 | green | 37.02 | 28.30 | 99.35 | 225.07 | 136.96 | Pouco urgente | 120 | 040 7144 4235 7258 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9995** | M | 1966-05-12 | green | 36.57 | 21.91 | 85.71 | 157.53 | 122.98 | Pouco urgente | 120 | 402 8956 8072 9157 |
| **9996** | F | 1957-12-29 | yellow | 37.01 | 27.13 | 95.89 | 199.70 | 135.19 | Urgente | 50 | 520 2218 2724 1048 |
| **9997** | F | 1989-07-07 | green | 35.95 | 13.14 | 66.99 | 64.87 | 103.80 | Pouco urgente | 120 | 200 3565 7347 2099 |
| **9998** | M | 1969-04-19 | blue | 36.49 | 20.81 | 83.37 | 145.97 | 120.58 | Não urgente | 240 | 042 3567 0449 7774 |
| **9999** | M | 1998-07-14 | yellow | 35.51 | 10.59 | 57.86 | 1.48 | 95.64 | Urgente | 50 | 001 0513 8209 0547 |

10000 rows × 11 columns

Figura 5.5: Dataframe gerado para Base de Dados (Fonte: autor)

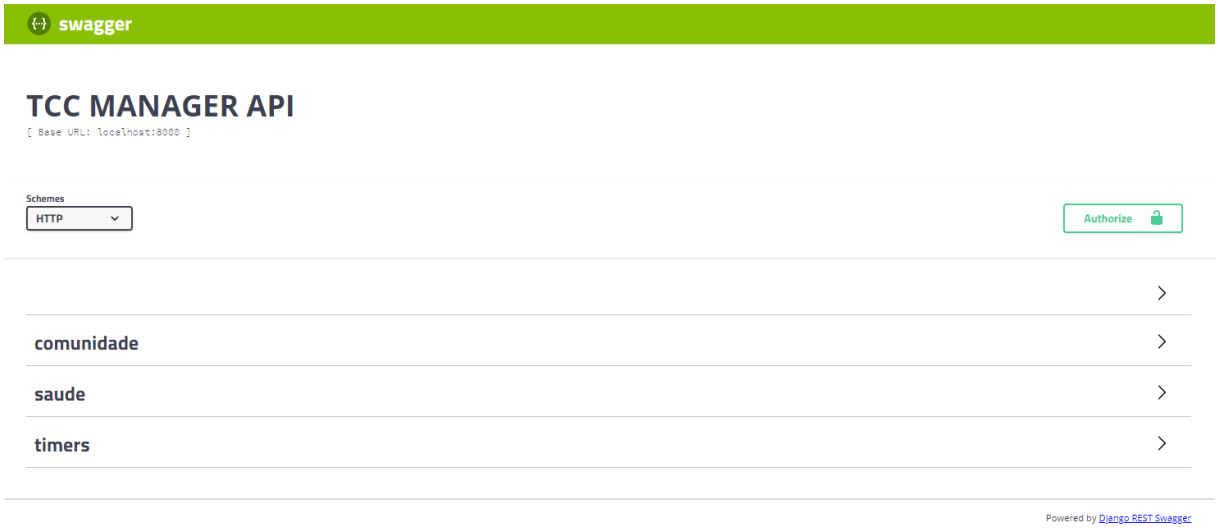


Figura 5.6: Interface da API com acesso aos microserviços dos 3 Domínios da aplicação (Fonte: autor)

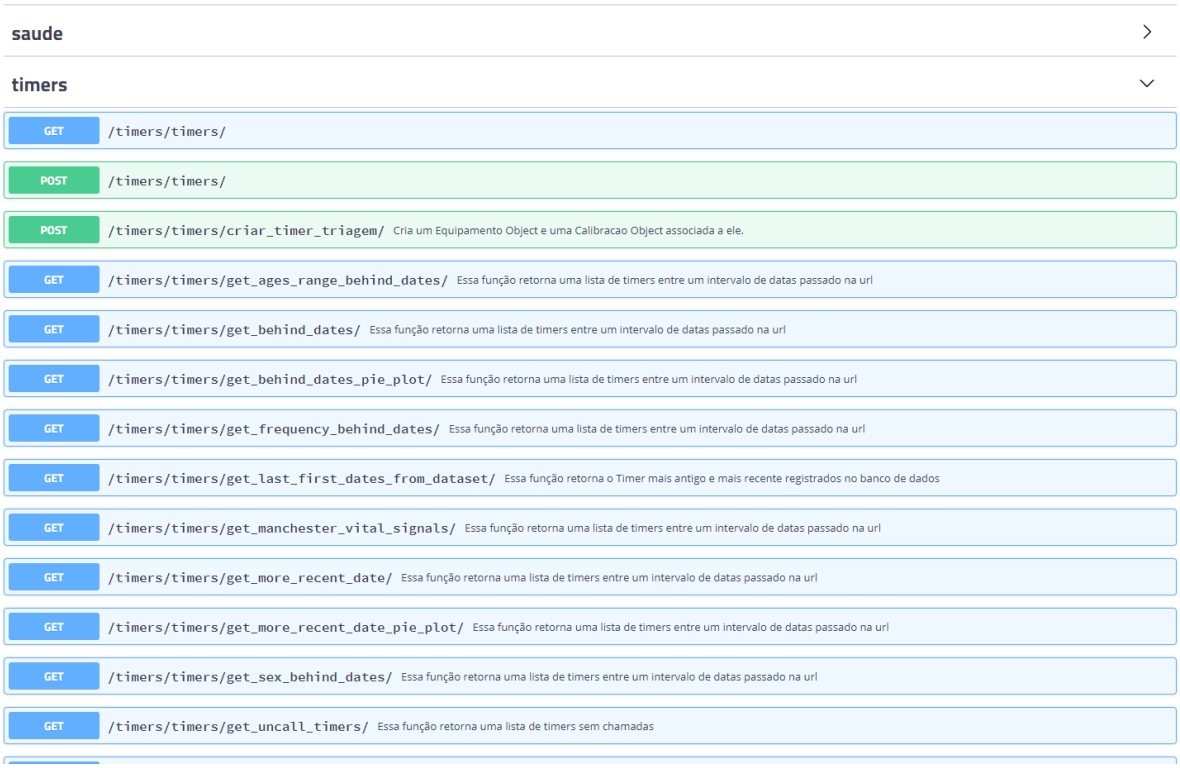


Figura 5.7: Microserviços do domínio Timers da aplicação (Fonte: autor)

## 5.2 Interface do Sistema

### 5.2.1 Página inicial

A página inicial do sistema exibe uma lista de pacientes já triados que estão aguardando atendimento (Figura 5.8). Os elementos da lista são clicáveis e ao clicar em qualquer um deles o sistema interpreta que ele está sendo chamado ao atendimento. Além disso, existe

um botão para redirecionamento para página de cadastro de uma nova triagem.

É possível notar que a esquerda da página inicial existem alguns botões em forma de ícones que facilitam a navegação entre as outras páginas do sistema. O primeiro botão, início, aponta para a rota da própria página inicial. O segundo botão, Dashboard, direciona para a página do Dashboard da aplicação. O terceiro botão, Dados, aponta para a página onde exibem-se os dados do banco de dados e o quarto botão redireciona o usuário para página responsável pelas Salas do sistema.


Figura 5.8: Página inicial do sistema (Fonte: autor)

### 5.2.2 Página Dashboard

Os gráficos da página Dashboard são produzidos nos limites do período de análise inserido pelo usuário. O período é definido através de duas datas, ao inserir e pressionar o botão ao lado dos campos de texto a requisição é mandada ao backend e após sua resposta os gráficos se atualizam automaticamente.

#### Análise de Atraso

A seção de análise dos atrasos foi construída dentro de um Q-card [1]. Os dados do período de análise são divididos em dias e o Card possui no canto superior direito botões de passagem entre os dias, onde com eles é possível navegar para frente ou para trás na linha do tempo. Além disso, no canto superior direito o Q-Card registra em relação ao dia quantos e a média por hora dos atendimentos realizados naquele dia (Figura 5.10).

Prosseguindo, no Q-card ainda são produzidos dois gráficos. O primeiro é um gráfico de pontos que possui dois conjuntos de informação. O eixo-x do gráfico representa cada hora do dia e o eixo-y a quantidade de atendimentos. O conjunto de dados na cor vermelha detém a informação dos atendimentos que foram realizados em atraso em relação ao tempo que o paciente triado em determinada cor deveria ser atendido. O conjunto de dados na

[1] https://quasar.dev/vue-components/card

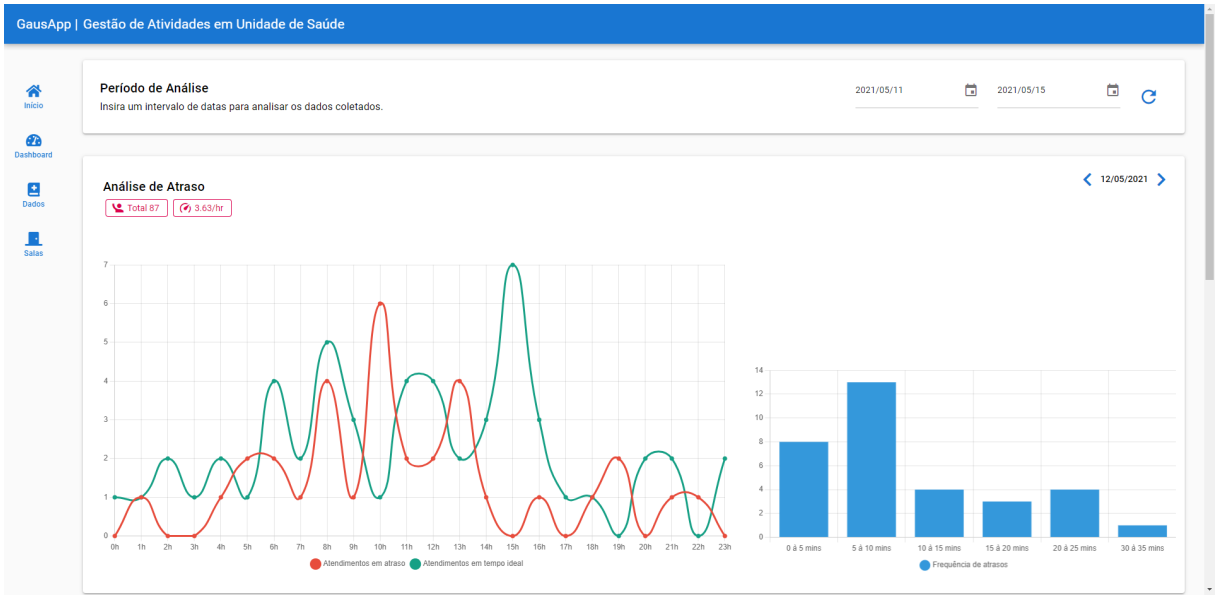


Figura 5.9: Página Dashboard (Fonte: autor)


Figura 5.10: Elementos da faixa superior do Q-Card de análise de atraso (Fonte: autor)

cor verde relata a informação dos pacientes que foram chamados ao atendimento antes do tempo limite de espera para a sua cor na triagem (Figura 5.11).

O segundo gráfico emite a análise de frequências dos chamados em atraso. Nele os atrasos são contabilizados em faixas de 5 em 5 minutos. Ou seja, é possível observar os intervalos mais frequentes de tempo em que ocorreram os atrasos. Por exemplo, na Figura 5.12 é possível notar que, dentre os atendimentos em atraso, a maior parte dos pacientes foram chamados em torno de 10 minutos atrasados em relação ao tempo em que deveriam ser atendidos.

### Classificações das triagens

Utilizou-se da base de dados que alimentou o sistema para construir gráficos que mostram as densidades das cores de classificação dadas aos pacientes na etapa de triagem em relação a algumas variáveis. Dentre elas, temos a data de nascimento, que foi utilizada para analisar como foram triados os grupos de faixas etárias diferentes. As faixas etárias foram definidas em Jovens, para pacientes de 0 a 19 anos, Adultos para pacientes de 20 a 59 anos e Idosos de 60 em diante.

Outra questão levantada para análise foi a das triagens em relação ao sexo dos paci-

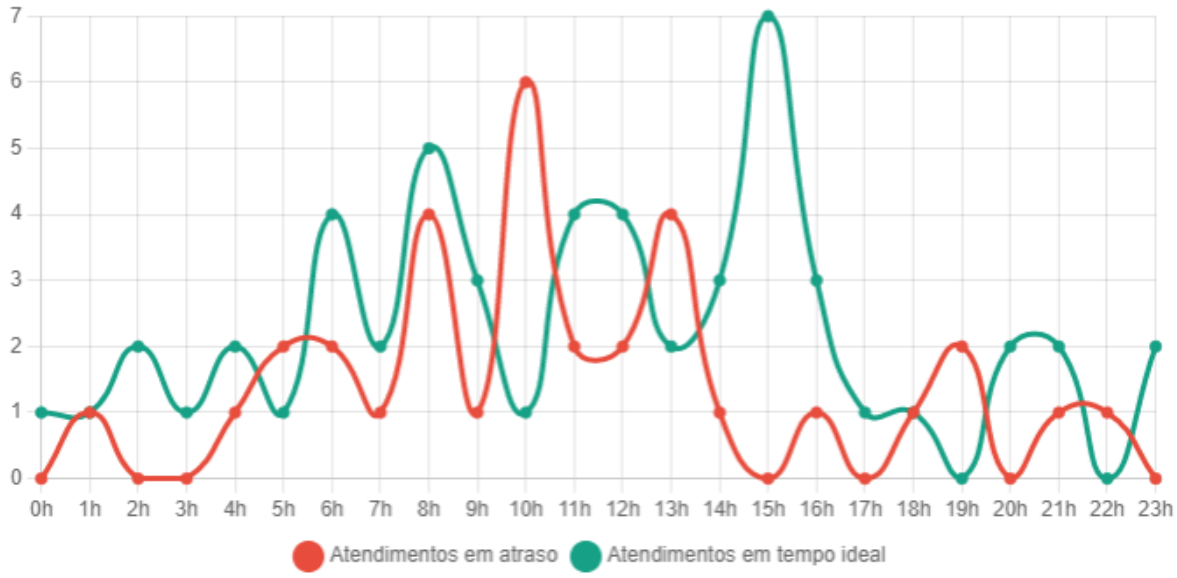


Figura 5.11: Gráfico de pontos dos chamados ao atendimento em atraso e em tempo ideal (Fonte: autor).

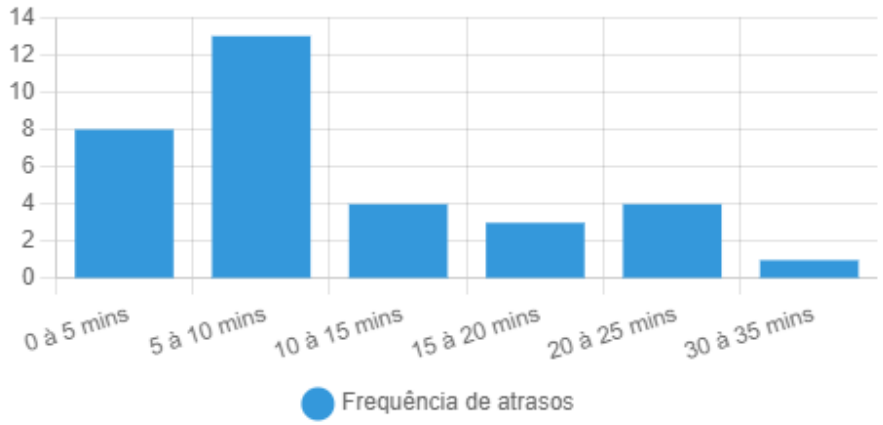


Figura 5.12: Gráfico de frequência dos atrasos nos chamados ao atendimento (Fonte: autor).

entes. Como pode ser visto na Figura 5.12.

Por final, foi produzida uma análise dos sinais vitais analisados e as cores atribuídas aos valores (Figura 5.15).

### 5.2.3 Página de Dados

A página de Dados foi construída com o objetivo de listar em componente *Q-Table* [2] os dados de Triagens guardados no banco de dados. A tabela é flexível quanto a quantidades de linhas exibidas por página. Os dados estão em modo de apenas leitura, mas numa

[2]https://quasar.dev/vue-components/table

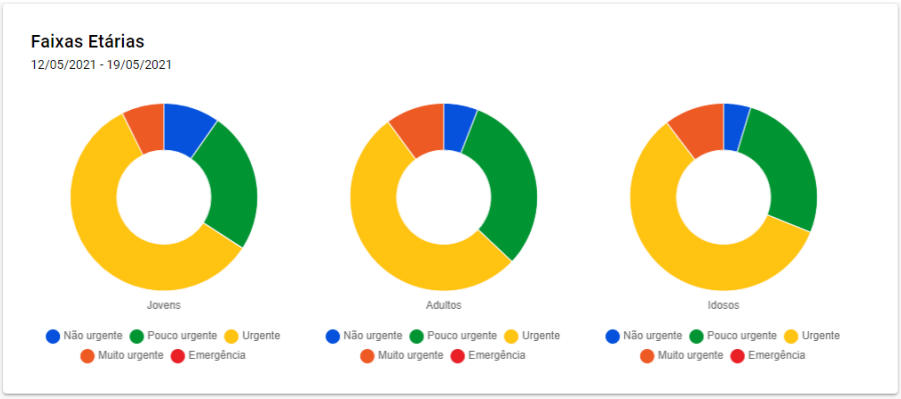


Figura 5.13: Gráfico de Cores vs Faixas etárias (Fonte: autor).

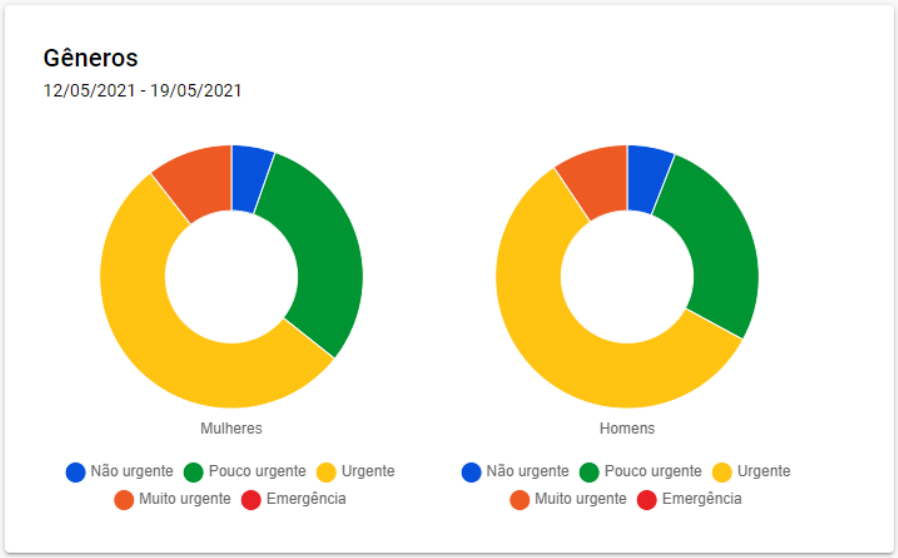


Figura 5.14: Gráfico de Cores vs Gêneros (Fonte: autor).

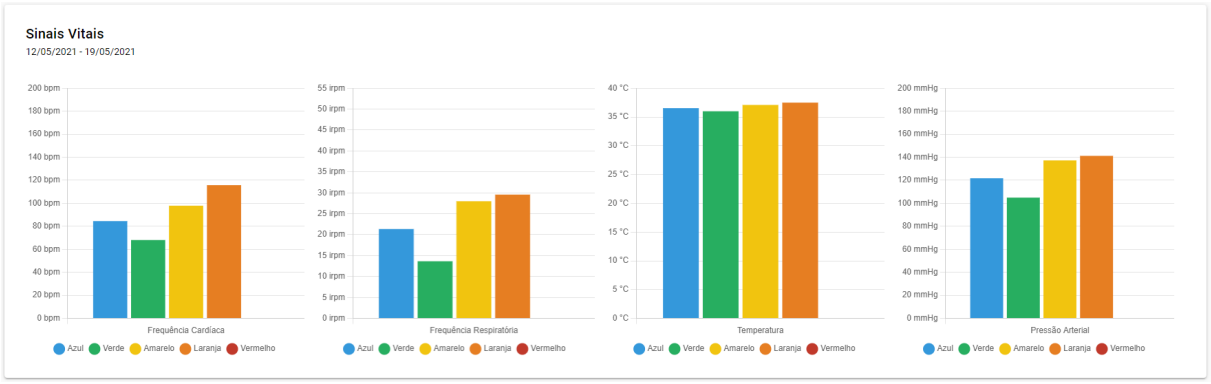


Figura 5.15: Gráfico de Cores vs Gêneros (Fonte: autor).

próxima versão será possível por meio dessa tabela editar ou remover cada informação (Figura 5.16). Além de trabalhar com outras tabelas de informações, como por exemplo medicamentos, insumos e funcionários da unidade.

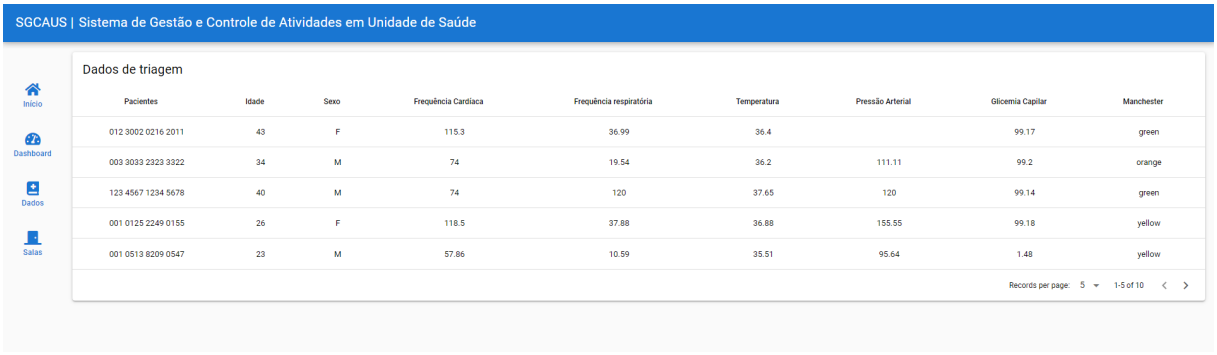


Figura 5.16: Página de Dados da Unidade de Saúde (Fonte: autor).

### 5.2.4 Página de Salas

A página de Salas foi construída com o objetivo de listar as diferentes salas que a unidade possui e fornecer um link de acesso a cada uma delas. A lista da sala foi construída com componentes *Q-Cards* que mostram o nome do tipo de Sala e a quantidade de salas que daquele tipo que o centro médico possui, além de um botão de *Entrar* que possui um redirecionamento para uma página dedicada especialmente ao tipo de sala do *Card* (Figura 5.17).


Figura 5.17: Página de acesso as salas da Unidade de Saúde (Fonte: autor).

### 5.2.5 Página da Sala de Triagem

A página da Sala de Triagem tem como objetivo cadastrar no sistema uma nova triagem realizada na unidade de saúde (Figura 5.19). O Q-Card contém um componente *Q-Form* [3], onde os campos possuem regras em suas entradas. Por exemplo, os campos onde serão inseridos dados dos sinais vitais não permitem caracteres diferentes de dígitos. Dessa forma as entradas são controladas e evita-se possíveis triagens com dados errados que comprometam as análises.

[3] https://quasar.dev/vue-components/form

O formulário registra informações pessoais do paciente, tais como nome, gênero, data de nascimento e número da carteirinha do SUS. Além disso, registra dados dos sinais vitais medidos por um profissional de Saúde na sala. É importante salientar que os campos não são obrigatórios pois nem toda unidade de saúde tem a capacidade de inferir todos os tipos de sinais vitais de um paciente. Ao final do formulário é possível salvar um ou uma lista de sintomas relatados pelo paciente e definir a cor referente a sua prioridade clínica no sistema do protocolo Manchester (Figura 5.19). Ao clicar em salvar os dados são enviados ao backend para serem guardados e o usuário é redirecionado de volta a página inicial.

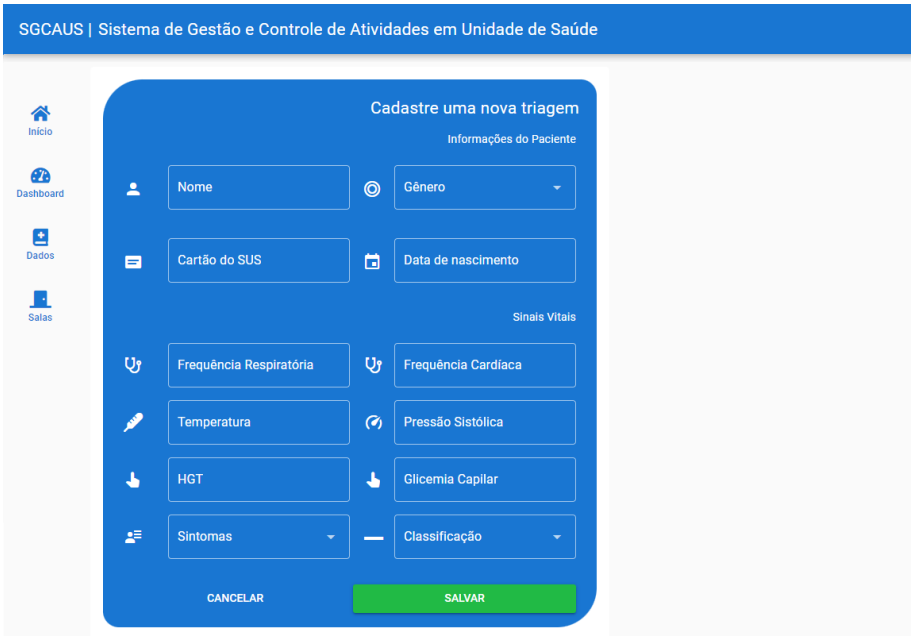


Figura 5.18: Página da sala de Triagem (Fonte: autor).


Figura 5.19: Formulário de Triagem preenchido (Fonte: autor).

# Capítulo 6

# Conclusão

A transformação digital é uma tendência mundial do mercado atual que consiste na adoção de novas tecnologias e inovações como parceiro do seus negócios, otimizando processos e atividades. A partir de boas escolhas, mudanças como essa podem ter uma grande impacto positivo.

A adoção de projetos de inovação e tecnologia por parte na gestão pública de uma UPA é importante para melhorar o desempenho dos processos e atividades realizadas. Isso significa benefícios para o gestor, para a cidade e seus habitantes. Além de que, em um mundo hiper conectado de hoje onde as pessoas estão cada vez mais familiarizadas com processos digitais deveria ser natural que tais serviços públicos acompanhassem o ritmo.

A solução proposta neste trabalho buscou contribuir para o processo de modernização visando aspectos como: redução de gastos, sugerir alterações na estrutura da equipe e melhorar o controle operacional. Com uma grande base de dados, o software pode ajudar no planejamento do gestor. A partir de um histórico de atendimentos, é possível traçar horário de mais e de menos frequência de pacientes, podendo reduzir carga horaria do corpo médico em determinados períodos ou aumentar o número de profissionais em outros espaços de tempo. Versões futuras do sistema podem ter a opção de salvar o histórico de consumo de insumos e assim tornar possível traçar estratégias de compras e evitar desperdícios. Outra possibilidade com auxilio do Dashboard é ter uma visão geral das épocas do ano em que determinadas doenças estão mais em evidência do que outras e daí traçar campanhas de combate a elas. Dessa maneira, são muitas as possibilidades de trabalhos futuros que a plataforma pode funcionar como base para início dessas ideias.

# Bibliografia

[DadosGov, ] DadosGov. Porta brasileiro de dados abertos. Accessed: 2021-03-23.

[Girls, 2021] Girls, D. (2021). Seu primeiro projeto django!

[Laureano, 2019] Laureano, G. d. S. (2019). Monitoramento da execução do protocolo de manchester: uma proposta tecnológica para o hospital universitário de florianópolis. Master's thesis.

[Martins et al., 2017] Martins, J. C. A., Guedes, H. M., de Souza, C. C., and Chianca, T. C. M. (2017). Associação entre sinais vitais e sistema manchester de triagem: estudo observacional retrospectivo. Master's thesis.

[MinistérioDaSaúde, ] MinistérioDaSaúde. Ministério da saúde - cbo 2002. Accessed: 2021-03-28.

[MinistériodaSaúde, ] MinistériodaSaúde. Unidades de pronto atendimento (upas): características da gestão às redes de atenção no paraná. Accessed: 2021-03-28.

[Redec, 2019] Redec (2019). A importância do protocolo de manchester.

[Rosa, 2021] Rosa, C. S. (2021). Hospital são josé de jaraguá do sul adota o protocolo de manchester no pronto socorro. Accessed: 2021-05-24.

[Souza et al., 2015] Souza, C. C. d., Araújo, F. A., and Chianca, T. C. M. (2015). Scientific literature on the reliability and validity of the manchester triage system (mts) protocol: A integrative literature review.

[Souza, 2021] Souza, M. S. (2021). Qual a finalidade das upas 24 horas. Accessed: 2021-03-28.

[UnaSus, 2021] UnaSus (2021). Maior sistema público de saúde do mundo, sus completa 31 anos. Accessed: 2021-07-21.

[with Danny, 2019] with Danny, M. A. (2019). Why quasar?