pista: periódico interdisciplinar
[sociedade tecnologia ambiente]

DESENVOLVIMENTO DE UM APLICATIVO DE TRIAGEM MÉDICA BASEADO NO PROTOCOLO DE MANCHESTER*

DEVELOPMENT OF A MEDICAL TRIAGE APPLICATION BASED ON THE MANCHESTER PROTOCOL

João Vítor Mendes Soares[1]
Nathan Ribeiro Ferreira Pinto[2]
Magali Rezende Gouvêa Meireles[3]

Submetido em: 25/03/2023
Aprovado em: 27/05/2023

RESUMO

O Protocolo de Manchester é um método de avaliação do estado de saúde de pacientes que permite a definição de uma prioridade para o seu atendimento, de modo que pacientes em situação mais grave sejam atendidos primeiro. O protocolo define que o tempo entre a che- gada do paciente ao hospital até a conclusão da avaliação não deve ultrapassar 10 minutos. Os profissionais que, geralmente, realizam essa avaliação são enfermeiros, mesmo que, em alguns casos, não tenham recebido treinamento adequado. Isso pode levar a avaliações in- corretas ou a tempos de avaliação superiores aos recomendados pelo protocolo. O objetivo do trabalho é o desenvolvimento de uma aplicação web que auxilia os enfermeiros na tarefa de avaliação dos pacientes. A aplicação reduz o tempo de avaliação, aumenta a precisão das priorizações e pode ser utilizada por profissionais de qualquer nível de experiência com o protocolo. A aplicação usa Redes Neurais Artificiais para classificar os pacientes de acordo com os fluxogramas do Protocolo de Manchester. Durante o desenvolvimento da aplicação foram feitos testes unitários.

Palavras-chave: Protocolo de Manchester. Aplicação Web. ASP.NET Core. Redes Neurais Artificiais.

ABSTRACT

The Manchester Protocol is a method of assessing the health status of patients that allows the definition of a priority for their care, so that patients in a more serious situation are treated first. The protocol defines that the time between the patient's arrival at the hospital and the completion of the assessment should not exceed 10 minutes. The professionals who generally carry out this assessment are nurses, in some cases, they have not received adequate training. This can lead to incorrect assessments or longer assessment times than recommended by the protocol. The objective of the work is the development of a web appli- cation that helps nurses in the patient assessment task. The application reduces evaluation time, increases the accuracy of prioritization and can be used by professionals of any level of experience with the Protocol. The application uses Artificial Neural Networks to classify patients according to Manchester Protocol flowcharts. During the application development, unit tests were carried out.

*Artigo apresentado à disciplina TCC 2

[1] Graduando de Sistemas de Informação da PUC Minas, Brasil – Contato: jvmsoares@sga.pucminas.br
[2] Graduando de Sistemas de Informação da PUC Minas, Brasil – Contato: nathan.pinto@sga.pucminas.br
[3] Professor and Research Coordinator of the Information Systems course at Pontifícia Universidade Católica de Minas Gerais - PUC Minas. Listed among the most productive and influential researchers in the world, according to the World Scientist and University Rankings, 2023. PhD in Information Science from UFMG, Brazil – Contato: magali@pucminas.br.

Keywords: Manchester Protocol. Web Application. ASP.NET Core. Artificial Neural Networks.

1 INTRODUÇÃO

O Protocolo de Manchester surge da necessidade dos hospitais de ordenar a fila de pacientes em espera de acordo com o seu estado de saúde, de modo que os pacientes em situação mais grave sejam atendidos primeiro. Ao chegar no hospital, o paciente, normalmente, é avaliado por um profissional de saúde que obtém detalhes de sua situação e define uma prioridade de atendimento, de acordo com o Protocolo de Manchester. Segundo Silva et al. (2019a), é recomendado que o tempo entre a chegada do paciente ao hospital e a sua classificação não ultrapasse 10 minutos.

Anziliero et al. (2016) concluíram, em um estudo com pacientes admitidos em uma unidade de pronto atendimento do sul do Brasil, que os pacientes esperam em média sete minutos até serem classificados pelo protocolo e a classificação dura em média dois minutos. O tempo total da chegada do paciente até o término da classificação chega a 10 minutos em média. Houve registro de um caso em que o paciente esperou ao todo 44 minutos.

Alguns enfermeiros entrevistados pela pesquisa de Roncalli et al. (2017) relataram que o treinamento oferecido para a realização da classificação não era suficiente, sendo necessário um tempo de prática para adquirir maturidade com o protocolo. Para Silva et al. (2019b), a maioria dos enfermeiros que atuam na classificação de risco não recebeu nenhum treinamento. E aqueles que recebem o treinamento reconhecem que é necessário experiência para aplicar a classificação com confiança e precisão.

A automatização do processo de classificação do Protocolo de Manchester é uma alternativa valiosa que, aliada a uma aplicação que proporciona uma interface amigável e de fácil aprendizado, pode ajudar na redução do tempo gasto para classificação da prioridade de atendimento de pacientes. Um benefício da adoção da classificação automática utilizando Rede Neural Artificial (RNA), é a redução do tempo gasto para a capacitação dos usuários. Os profissionais da saúde podem focar seus esforços na interação com o paciente, considerando-se que não é mais necessário navegar pelos fluxogramas do Protocolo de Manchester, manualmente, pois a classificação é realizada pelo algoritmo.

Custodio (2021) automatizou o sistema de classificação definido pelo Protocolo de Man- chester ao desenvolver um modelo utilizando RNA. De acordo com a autora, o modelo apresentou resultados satisfatórios quando comparados a outros trabalhos disponíveis na literatura. Seguindo a sugestão de trabalhos futuros de Custodio (2021), este trabalho desenvolveu uma aplicação que serve de interface entre o modelo de classificação e o usuário, permitindo que a tecnologia seja aplicada em um cenário real.

Este trabalho possui como objetivo geral o desenvolvimento de uma aplicação web que permita que o usuário registre uma série de informações relativas a um determinado paciente. O sistema processa, utilizando um modelo de RNA desenvolvido por Custodio (2021), as in- formações recolhidas pelo usuário responsável por preencher o questionário de atendimento do paciente para a definição da prioridade de atendimento. A aplicação retorna para o usuário a prioridade de atendimento do paciente de acordo com o Protocolo de Manchester. A aplicação foi hospedada em um serviço de nuvem[4] e pode ser acessada utilizando dispositivos móveis e computadores. A aplicação é acessível e de fácil utilização, devendo responder rapidamente às interações do usuário.

Os objetivos específicos deste trabalho são:

• Desenvolvimento de um *front-end* responsivo, que se adapte aos formatos e resoluções de telas mais comuns do mercado. Essa responsividade é garantida por uma funcionalidade chamada *media queries* e é disponibilizada pela linguagem de estilo Cascading Style Sheet (CSS), que é responsável por toda a estilização da aplicação. As *media queries* permitem definir diferentes estilizações dos componentes apresentados ao usuário de acordo com a resolução de tela do mesmo.

• Desenvolvimento de uma API que permita o uso do modelo de RNA desenvolvido por Custodio (2021) no back-end da aplicação.

• Realização de testes unitários, utilizando a ferramenta de testes xUnit. Esta ferramenta permite a realização de testes unitários para garantir o funcionamento dos elementos individuais da aplicação.

O Referencial Teórico é apresentado na próxima seção, abordando os conceitos de classificação, RNA, Protocolo de Manchester, .Net e ASP.Net Core, padrão

[4] https://filamanchester.azurewebsites.net

arquitetural *Model-View-Controller* (MVC) e testes de *software*. Alguns trabalhos relacionados são apresentados na Subseção 2.7. A Seção 3 apresenta a metodologia utilizada no desenvolvimento da aplicação. A Seção 4 registra os resultados obtidos durante o desenvolvimento do trabalho.

2 REFERENCIAL TEÓRICO

Nesta seção, são apresentados os principais conceitos relacionados a este trabalho. A Subseção 2.1 aborda o método de classificação. Na Subseção 2.2 é apresentada a definição das RNA. O Protocolo de Manchester é abordado na Subseção 2.3. A Subseção 2.4 trata do ecossistema de desenvolvimento de *software* da Microsoft, o .Net. A Subseção 2.5 discute o *framework* de desenvolvimento *web*, o ASP Net Core. A Subseção 2.6 aborda o padrão arquitetural MVC e testes de *software* são discutidas na Subseção 2.7. Finalmente, a Subseção 2.8 trata dos trabalhos relacionados.

2.1 Método de Classificação

A classificação é um processo de tomada de decisão frequentemente utilizado por seres humanos. Segundo Xavier (2008) um dos precursores da análise do processo de classificação foi Aristóteles, quando definiu a classificação em dez gêneros: substância, quantidade, qualidade, relação, lugar, tempo, posição, posse, ação e paixão. Xavier (2008) conclui que, esses gêneros têm por função possibilitar a ordenação do pensamento, fornecendo elementos para a perfeita caracterização do objeto a ser estudado.

Já em uma definição voltada para classificação no ambiente da Inteligência Artificial (IA) Freitas (2019) relata que um problema de classificação ocorre quando um objeto precisa ser atribuído a um grupo ou classe predefinida com base em uma série de atributos observados relacionados a esse objeto.

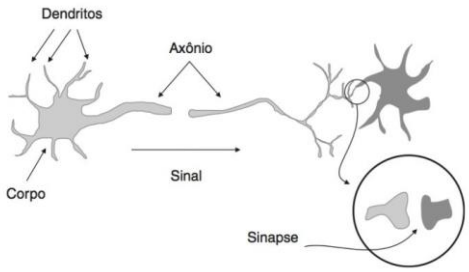
Muitos problemas nos negócios, na ciência, na indústria e na medicina podem ser tratados como problemas de classificação. Isso comprova como a classificação é uma ferramenta importante para a resolução de problemas da sociedade e na melhora da qualidade de vida do ser humano (FREITAS, 2019).

2.2 Redes Neurais Artificiais

A história das RNAs remete à década de 40, quando os cientistas da computação utilizaram o cérebro humano como inspiração para desenvolver uma máquina inteligente (FACELI et al., 2011).

O cérebro é o centro de comando do sistema nervoso, que possui células cruciais para a tomada de decisões. Essas células são os neurônios que existem em grande quantidade no cérebro, na ordem de 10 a 500 bilhões. Segundo Faceli et al. (2011) os neurônios possuem prolongamentos especializados na recepção de sinais elétricos de outros neurônios ou do ambiente chamados dendritos. Após o recebimento do sinal, o mesmo é processado internamente e enviado para outro neurônio por meio dos axônios, outros prolongamentos dos neurônios que em contraste com os dendritos são especializados no envio de sinais. Pode-se observar a representação de um neurônio na Figura 1.

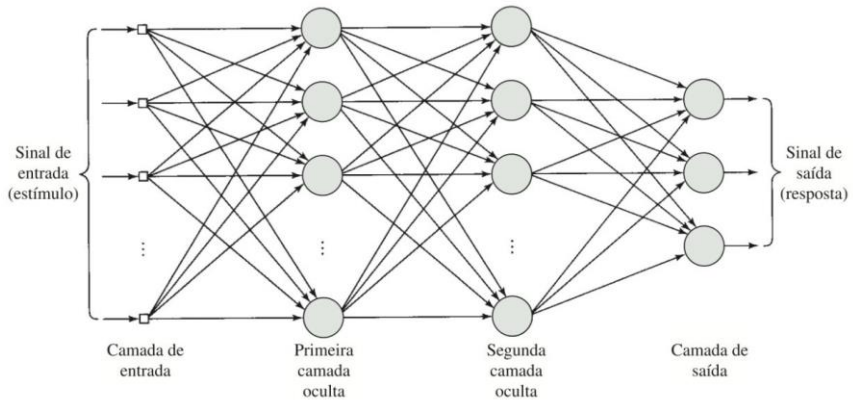
Figura 1 – Neurônio biológico simplificado



Fonte: Faceli et al. (2011).

As RNAs possuem neurônios artificiais, também conhecidos como nós. Da mesma forma que os neurônios do cérebro são interligados com outros neurônios, os nós das RNAs também se comunicam. Eles podem ser distribuídos em camadas, responsáveis pelo processa- mento de informações, por meio de funções matemáticas, gerando uma saída que é enviada a um conjunto de nós da próxima camada. Depois de passar por várias camadas a RNA apresenta uma resposta (HAYKIN, 2003). A Figura 2 ilustra uma RNA multicamadas.

Figura 2 – Grafo arquitetural de uma rede neural multicamadas



Fonte: Haykin (2003).

2.3 Protocolo de Manchester

De acordo com o GBCR (2021), o Protocolo de Manchester permite identificar a prioridade de atendimento clínico e ajuda a estipular o tempo de espera recomendável para avaliação médica de um paciente. O protocolo foi usado pela primeira vez em 1997 na Manchester Royal Infirmary na cidade de Manchester e é composto por 52 fluxogramas.

Segundo Amthauer e Cunha (2016) uma enfermeira fica encarregada de selecionar o fluxograma mais adequado para cada caso, baseado em fatores como a principal queixa do paciente, histórico médico, sinais e sintomas apresentados que venham a ser notados pelo profissional. Realizando o atendimento para a triagem, um discriminador é encontrado e após percorrer o fluxograma o paciente é classificado em uma das cinco categorias representadas pelas seguintes cores:

• Vermelho (Emergência), atendimento imediato por um médico.

• Laranja (Muito urgente), atendimento o mais rápido possível. É tolerável até 10 minutos de espera.

• Amarelo (Urgente), atendimento rápido, mas o paciente pode aguardar até 60 minutos.

• Verde (Pouco urgente), o paciente pode aguardar atendimento por 120 minutos, ou pode ser encaminhado para outros serviços de saúde.

• Azul (Não urgente), o paciente pode aguardar atendimento por até 240 minutos, ou pode ser encaminhado para outros serviços de saúde.

A Figura 3 ilustra as cinco categorias do Protocolo de Manchester.

Figura 3 – Classificações de risco do Protocolo de Manchester

Fonte: Ministério da Educação (2016).

2.4 .Net e ASP.Net Core

De acordo com Microsoft (2020), .NET ou dotNet é uma plataforma de código aberto para desenvolvimento de software, que oferece um ambiente de execução, diversas bibliotecas e APIs. Pode ser utilizada para construir diversos tipos de aplicativos, quais sejam aplicações *web*, aplicações para dispositivos móveis, aplicações para computadores pessoais, microsserviços, jogos, aplicações de aprendizado de máquina, aplicações para nuvem e aplicações para dispositivos IoT. As linguagens de programação que podem serem utilizadas na plataforma são: C#, Visual Basic e F#.

A plataforma .Net possui 4 diferentes implementações, sendo elas:

• .NET 5 e versões posteriores (e .NET Core).

• .NET Framework.

• Mono.

• UWP.

A plataforma possui um *framework* específico para desenvolvimento de aplicações *web*, chamado de ASP.NET.

Segundo Freeman (2020), o ASP.NET contém ferramentas de baixo nível para manipulação de requisições HTTP, middlewares (como o ORM Entity Framework Core ou o *middleware* de autenticação *Identity*) e componentes principais como roteamento de URLs e o *motor* de renderização das páginas Razor.

Ainda de acordo com Freeman (2020) as páginas Razor aceleram o desenvolvimento de páginas web pois código (C#) e conteúdo (HTML) ficam juntos em um mesmo arquivo. As páginas Razor podem ser usadas em conjunto com o framework MVC, sendo recomendável utilizá-las pra funções secundárias.

2.5 Model-View-Controller

O padrão de arquitetura MVC foi desenvolvido no final da década de 1970 pelo cientista da computação norueguês Trygve Reenskaug com o objetivo de proporcionar uma arquitetura com ênfase na independência entre os módulos do sistema, permitindo que equipes distintas trabalhem simultaneamente no mesmo sistema (FOWLER et al., 2006).

Segundo Aniche et al. (2016) o padrão MVC se tornou muito popular na indústria de desenvolvimento *web*, sendo o núcleo de vários *frameworks* de desenvolvimento, como o Spring MVC (Java), ASP.NET MVC (.NET), Ruby on Rails (Ruby), e Django (Python).

O sistema desenvolvido de acordo com a arquitetura MVC é dividido em três camadas.

Kalelkar et al. (2014) explicam essas camadas como:

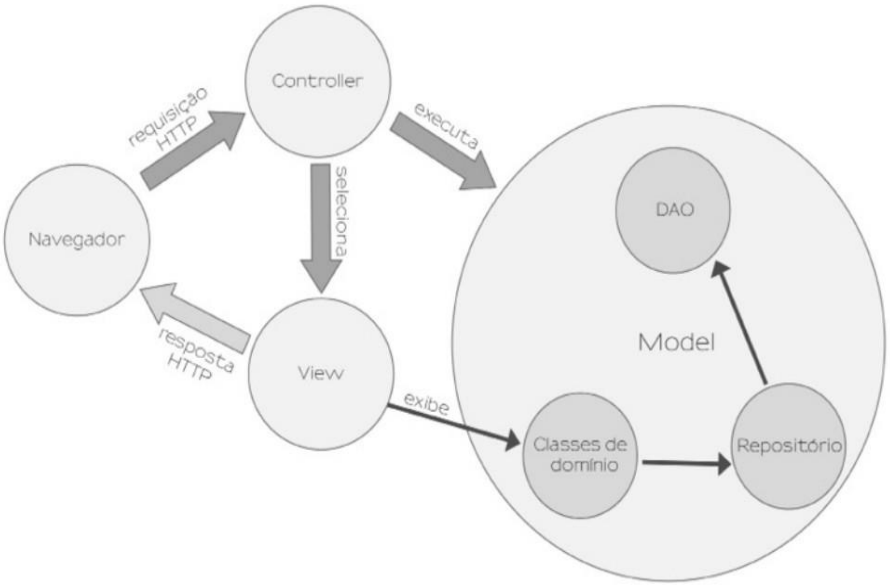
• Model (Modelo): a camada de modelo contém as funcionalidades do sistema, regras de negócio e acesso aos dados armazenados no banco de dados.

• View (Visão): a camada de visão é responsável por apresentar e atualizar os dados exibi- dos ao usuário de acordo com as instruções do controlador.

• Controller (Controlador): a camada de controle recebe as requisições HTTP do *front-end*. essas requisições carregam as instruções geradas pelas ações do usuário e disparam as funções do sistema que processam os dados do modelo e atualizam a visão do sistema.

A Figura 4 mostra como o padrão pode ser implementado.

Figura 4 – Uma possível implementação de MVC para aplicações



Fonte: Silveira (2012).

2.6 Testes de *software*

Teste de *software* pode ser definido como: "uma atividade na qual um sistema ou componente é executado sob condições especificadas. Os resultados são observados ou registrados e uma avaliação é feita de algum aspecto do sistema ou componente."(IEEE. . . , 2014)

Para Sommerville (2019) o teste de software tem dois objetivos: determinar se o *software* atende aos requisitos definidos (teste de validação) e encontrar entradas que façam com que o software se comporte de maneira errada (teste de defeitos). Os testes não conseguem, no entanto, garantir que um software está livre de defeitos ou que vai se comportar conforme as especificações em qualquer circunstância.

Para Pressman (2016) a estratégia de testes para aplicações web pode ser resumida em 10 passos:

1. Revisão do modelo de conteúdo.

2. Revisão do modelo de interface, com o objetivo de garantir a implementação de todos os casos de uso.

3. Revisão do modelo de projeto em busca de erros na navegação pela aplicação.

4. Teste de interface em busca de erros de apresentação e/ou navegação.

5. Testes unitários.

6. Testes de navegação abrangendo toda a arquitetura do sistema.

7. Testes em diferentes ambientes.

8. Testes de segurança.

9. Testes de desempenho.

10. Testes com um grupo controlado de usuários para avaliar as interações em busca de problemas que possam ter passado pelas etapas anteriores e também avaliar a usabilidade.

2.7 Trabalhos relacionados

Nesta subseção, são apresentados cinco trabalhos relacionados que contribuíram para a fundamentação deste trabalho. Os trabalhos de Riquetta (2018) e de Silva e Hild (2019) serviram de referência quanto às particularidades do desenvolvimento de aplicações na área de saúde relacionadas ao atendimento e à triagem de pacientes. Silva et al. (2019b) apontaram uma deficiência no treinamento de pessoas para a realização da classificação do Protocolo de Manchester. Isso reforça a necessidade de uma ferramenta automática que possa ser utilizada por pessoas com pouco ou nenhum treinamento. Já os trabalhos de Custodio (2021) e Fonseca (2019) foram importantes para aprimorar o entendimento dos autores sobre o funcionamento de RNA e facilitar a integração do modelo de classificação desenvolvido por Custodio (2021) com a aplicação desenvolvida neste trabalho.

Riquetta (2018) desenvolveu um aplicativo utilizando Xamarin, uma plataforma para aplicativos móveis do ambiente de desenvolvimento .NET. Esse aplicativo possui o objetivo de auxiliar pacientes de pronto atendimento por meio de informações como a localização de Unidades de Pronto Atendimento (UPAs) próximas, informando também a gravidade de casos que estão aguardando atendimento em cada UPA. Riquetta (2018) chegou à conclusão que, durante os testes de usabilidade, o aplicativo desenvolvido se mostrou efetivo ao cumprir o objetivo de orientar os pacientes para as UPAs menos movimentadas, equilibrando as filas de espera de atendimento.

Silva et al. (2019b) realizaram um estudo sobre a utilização do Protocolo de Manchester como ferramenta para classificação de prioridade da fila de atendimento a gestantes. Concluído que os enfermeiros se mostraram satisfeitos com a adoção do protocolo. Mas os enfermeiros se queixaram de várias dificuldades durante o atendimento, como a falta de fluxogramas para classificação de pacientes gestantes, o tempo de espera para que o quadro clínico das gestantes seja classificado e a falta de treinamento adequado para a realização das classificações de risco.

Silva e Hild (2019) realizaram um trabalho em que foi desenvolvido um sistema web com a finalidade de auxiliar gestores da área da saúde nas tomadas de decisões. O back-end do sistema foi desenvolvido utilizando ferramentas da plataforma .NET como a linguagem de programação C#. E o front-end se baseou na biblioteca bootstrap. Os autores concluíram que o sistema foi implementado com sucesso. E com base nos testes de usabilidade realizados com usuários voluntários, foram feitas sugestões de novas funcionalidades do sistema a serem implementadas no futuro.

Fonseca (2019) utiliza dois modelos de RNA treinados pela biblioteca OpenNN. O primeiro modelo utiliza uma base de dados de biomarcadores de pacientes com Transtorno Bipolar ou esquizofrenia e o outro modelo usa uma base de pacientes com depressão transtorno bipolar e indivíduos sem transtorno. Os indivíduos foram classificados pelo modelo treinado com o objetivo de diferenciar cada indivíduo quanto ao seu transtorno. Foi obtido um grau de acerto de mais de 90% no primeiro modelo e mais de 80% no segundo.

Custodio (2021) desenvolveu um modelo de classificação utilizando RNA. A classificação segue o Protocolo de Manchester e o modelo da RNA foi treinado utilizando a base de dados de Seiger et al. (2014), que reuniu dados do departamento de emergência de hospitais pediátricos da Europa. Custodio (2021) ressalta que nem todos os pacientes da base de dados tiveram as informações do prontuário preenchidas completamente, o que limitou à 50% dos dados da base utilizada no treinamento do modelo.

Após a realização de experimentos para validar o modelo, Custodio (2021) concluiu que a acurácia obtida variou de 77% a 86%, sendo maior que de outros modelos encontrados na literatura e ainda ressalta que esses resultados são significativamente afetados pela quantidade de dados faltantes na base de dados de treinamento.

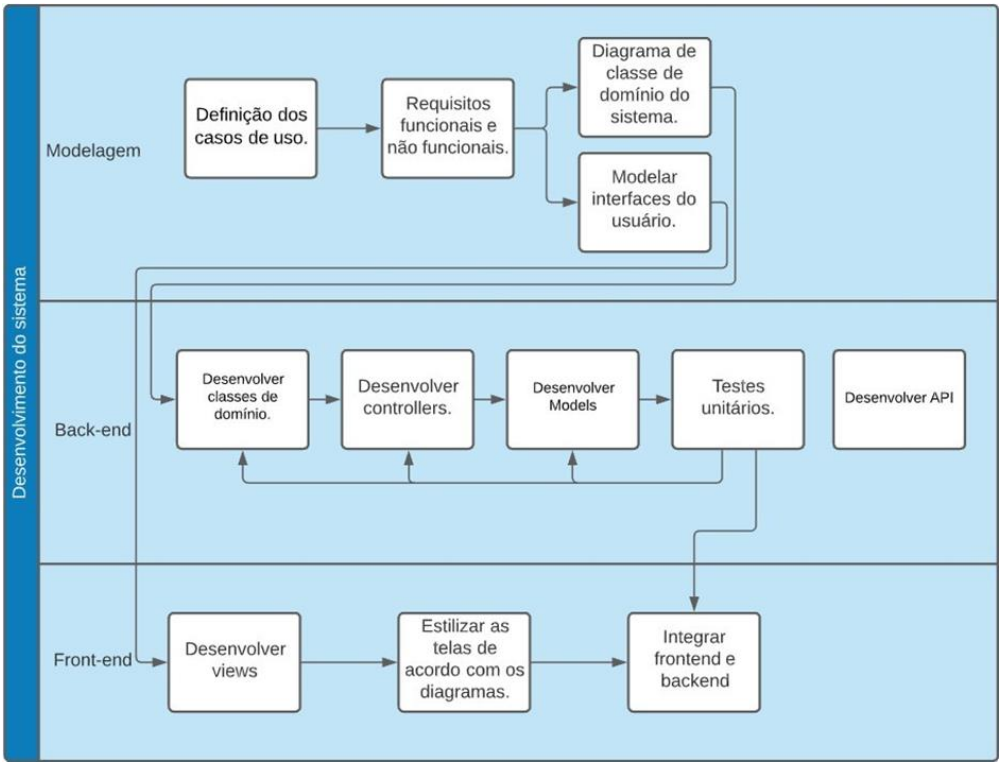
3 METODOLOGIA

As principais etapas da metodologia deste trabalho são a modelagem, o desenvolvimento do *back-end*, o desenvolvimento do *front-end* e os testes unitários da aplicação.

A primeira etapa, consistiu em definir casos de uso e estabelecer os requisitos funcionais e não funcionais do projeto. Os modelos de interface do usuário e diagramas de classe foram modelados com a finalidade de cumprir os requisitos da aplicação. Os requisitos funcionais foram estabelecidos com a finalidade de coletar os dados necessários para o funcionamento do modelo desenvolvido por Custodio (2021) e também, para exibir as informações coletadas de uma forma organizada e funcional para o usuário. Os requisitos não funcionais foram ela- borados levando em consideração as características do tipo de aplicação desenvolvida. Como uma aplicação WEB não necessita de instalação prévia, o usuário precisa possuir uma conexão estável com a Internet.

A persistência dos dados da aplicação foi realizada com o recurso de local storage do navegador, que permite o armazenamento de dados localmente no dispositivo do usuário.

Figura 5 – Fluxograma do processo de desenvolvimento do projeto



Fonte: Elaborada pelos autores.

A etapa do desenvolvimento do back-end foi dividida em três partes:

1. Desenvolver classes de domínio: essas classes representam a abstração de objetos reais com atributos e comportamentos estabelecidos de acordo com a modelagem de requisitos e casos de uso.

2. Desenvolver controllers: os controllers possuem o papel de ponte entre o back-end e o front-end.

3. Testes unitários: como é ilustrado no fluxograma apresentado na Figura 5, ao atingir os testes unitários é possível retornar às etapas de desenvolvimento para realizar correções de bugs que podem surgir durante o teste.

A etapa de desenvolvimento do front-end foi formada por quatro sub etapas:

1. Desenvolver *views*: as visões contêm as estruturas das páginas e atualizam as informações exibidas ao usuário.

2. Estilizar as telas de acordo com os diagramas: nesta sub etapa, os diagramas de interface do usuário serão utilizados para guiar a estilização das páginas realizada com a linguagem de estilo CSS.

3. Integrar o *front-end* e o *back-end*: nesta sub etapa, a aplicação estará funcionando. Os comandos recebidos pela interface do usuário serão processados e um resultado e devolvido e exibido na tela.

3.1 Desenvolvimento de uma *API* para o modelo de RNA

Foi desenvolvida uma API em Python para obter a prioridade de atendimento do paciente a partir de seu prontuário. Os seguintes dados são passados para a API: idade, frequência cardíaca, frequência respiratória, saturação de oxigênio, sintomas e temperatura. O proces- samento destes dados é feito com o modelo de Custodio (2021) que retorna a prioridade de atendimento. Nenhum dos parâmetros é obrigatório e caso um ou mais não sejam passados para a API é possível obter uma prioridade de atendimento com um nível menor de confiança.

3.2 Desenvolvimento do *back-end*

O desenvolvimento do *back-end* da aplicação foi iniciado com a criação e configuração do projeto ASP .NET Core MVC. Em seguida, a partir da modelagem, foram desenvolvidas as classes do domínio da aplicação e os *Controllers* para realizar as operações básicas (criar, ler, atualizar e deletar) para cada entidade e comunicar com a respectiva *View*.

Foi desenvolvida uma *Controller* especial para a interação com a *API* da RNA. Esta *API* executa o modelo de RNA criado por Custodio (2021) e é implementada como uma Azure *Function* (serviço de computação *Serverless* do Azure). A *API* é consumida pelo *Controller* da aplicação que envia, por meio de uma chamada POST, os dados necessários do paciente a ser classificado. A *API*, em seguida, faz o processamento dos dados com a RNA, o resultado deste processamento é a prioridade de atendimento que retorna à *Controller*.

3.3 Desenvolvimento do front-end

A interface foi desenvolvida a partir do projeto ASP .NET Core MVC. Cada página da aplicação é formada por um arquivo .cshtml, que é uma extensão especial do ASP .NET, que define o tipo dos arquivos que possuem a Linguagem de Marcação de HiperTexto (HTML) das páginas combinado com alguns elementos da linguagem de programação C#. Esses arqui- vos contêm a estrutura das páginas, como posições de elementos da interface os tipos destes elementos. Outro elemento importante do *front-end* são os arquivos no formato (CSS). Esses arquivos são responsáveis por estilizar as páginas. É um arquivo no formato .cs, que define as funcionalidades dos elementos HTML e que também possui as regras de negócio da aplicação.

3.4 Testes

Junto do desenvolvimento da aplicação foram desenvolvidos testes unitários utilizando a biblioteca xUnit para facilitar a identificação de erros e inconsistências.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Esta seção contém os resultados provenientes do desenvolvimento da aplicação. O período de desenvolvimento ocorreu de agosto de 2021 até novembro de 2021.

A primeira etapa do processo de desenvolvimento foi a modelagem do sistema. Esta etapa consistiu na definição dos casos de uso, na definição de requisitos funcionais, de requisitos não funcionais e dos protótipos das telas da aplicação. Foram analisados quais dados eram necessários para a classificação da prioridade de atendimento pelo modelo de Custodio (2021). Com isso em mente, os fluxos das ações a serem performadas pelo usuário foram estabelecidos e documentados nos casos de uso disponíveis nos Apêndices A e B. Os requisitos funcionais foram estabelecidos com base nos casos de uso, com a finalidade de viabilizar uma interface entre o modelo desenvolvido e o usuário. Os requisitos não funcionais foram elaborados com base nas características da aplicação *WEB*, que necessita de uma conexão constante com a in- ternet para funcionar de forma apropriada. O RNF3 foi elaborado levando em consideração os navegadores mais utilizados por usuários para acessar páginas *WEB* no Brasil. Segundo Stat- counter (2021) no ano de 2020, cerca de 83.94% dos usuários de Internet utilizaram o navegador Chrome, 4.99% utilizaram o Firefox, 2.45% utilizaram o Edge e os outros 8.63% dos usuários ficaram divididos entre outros navegadores. Os Quadros 1 e 2 apresentam os requisitos funci- onais e não funcionas respectivamente. Cada requisito possui um código único, uma descrição e uma prioridade que determina a importância daquele requisito. Os protótipos das interfaces da aplicação foram elaborados com a finalidade de destacar informações importantes como a prioridade de atendimento, com a utilização das cores do Protocolo de Manchester para indicar o nível de urgência da situação do paciente.

Quadro 1 - Requisitos Funcionais

Requisitos	Descrição	Prioridade
RF1: Coletar dados do paci- ente.	Cadastro de sintomas do paciente, coletando informações sobre: temperatura, saturação do oxigênio, frequência respiratória, idade, discri- minador positivo (sintoma) e tipo de sintoma.	Alta
RF2: Classificar prioridade de atendimento do paciente.	Utilizar os dados coletados para classificar a prioridade do paciente.	Alta
RF3: Listar classificações.	Listar classificações de pacientes realizadas no passado.	Alta
RF4: Ordenar listagem de classificação de pacientes.	A lista de classificação de pacientes poderá ser ordenada de acordo com a data de classificação e de acordo com a prioridade de atendimento.	Média

Fonte: Elaborado pelos autores.

Quadro 2 - Requisitos Não-Funcionais

Requisitos	Descrição	Prioridade
RNF1: Estabilidade.	Por se tratar de uma aplicação *WEB*, a estabilidade da aplicação será dependente da qualidade da conexão do usuário com a Internet.	Alta
RNF2: Usabilidade.	A interface de usuário deverá ser de fácil entendimento, dispensando treinamento prévio.	Média
RNF3: Compatibilidade.	A aplicação deverá ser executada nos *browsers*: Google Chrome, Mozila Firefox e Microsoft Edge.	Alta

Fonte: Elaborado pelos autores.

A Subseção 4.1 apresenta os validadores da aplicação. Em seguida a Subseção 4.2 apresenta os testes unitários programados para garantir a qualidade do software. Já a Subseção 4.3 mostra as interfaces finais da aplicação.

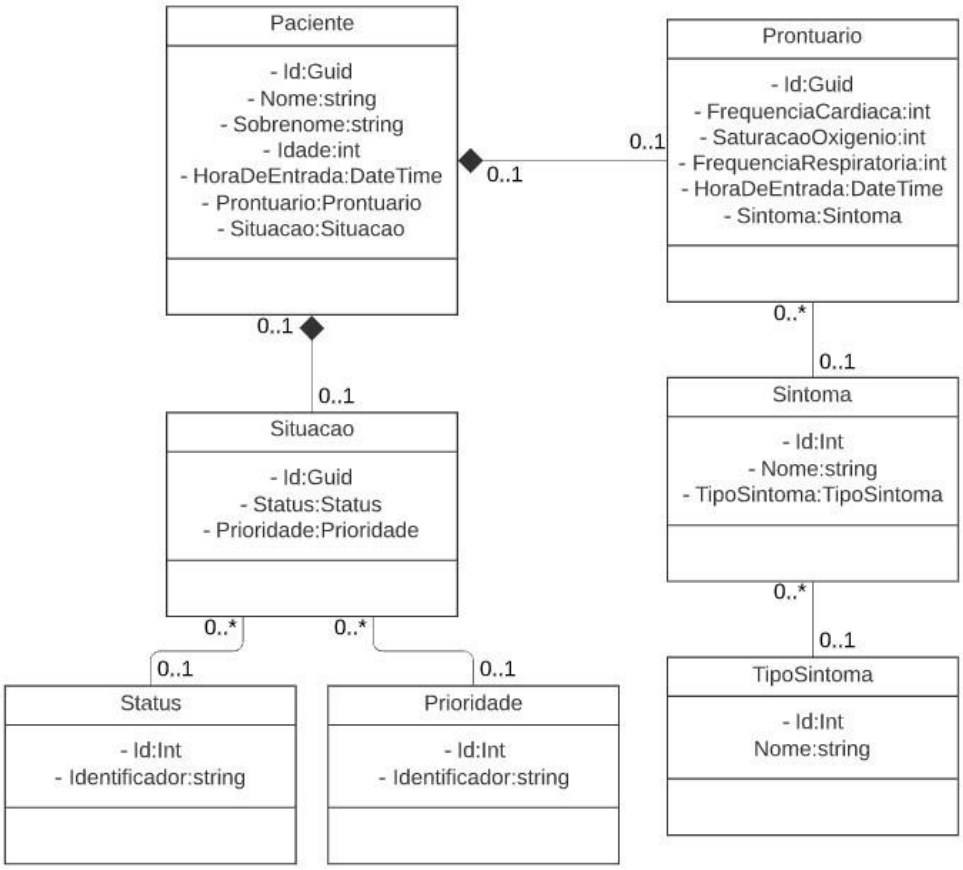
4.1 Detalhamento da aplicação

Cada classe de domínio da aplicação possui um atributo Id. As classes de Paciente, de Prontuário e de Situação possuem identificadores do tipo *Guid*. Isso permite que cada objeto possua um identificador único gerado automaticamente. Como ilustrado na Figura 6, as classes de Prontuário e de Situação possuem um relacionamento de associação por composição com a classe de Paciente, significando que quando um paciente é excluído da aplicação o seu prontuário e a sua situação também são excluídos e quando um novo paciente é cadastrado, o seu prontuário e situação também são instanciados.

A classe de Situação registra o *status* de atendimento do paciente, ou seja, se ele já foi atendido. Essa classe também registra a prioridade do paciente de acordo com os níveis de prioridade de atendimento do Protocolo de Manchester.

A classe de Prontuário armazena os dados referentes à situação física do paciente. Essa classe também representa o principal sintoma do paciente utilizando um objeto da classe Sintoma. Por sua vez, a classe Sintoma possui um atributo *TipoSintoma* que disponibiliza tipos de sintoma pré estabelecidos, facilitando o processo de entrada de dados para o usuário da aplicação.

Figura 6 – Diagrama de classes de domínio



Fonte: Elaborada pelos autores.

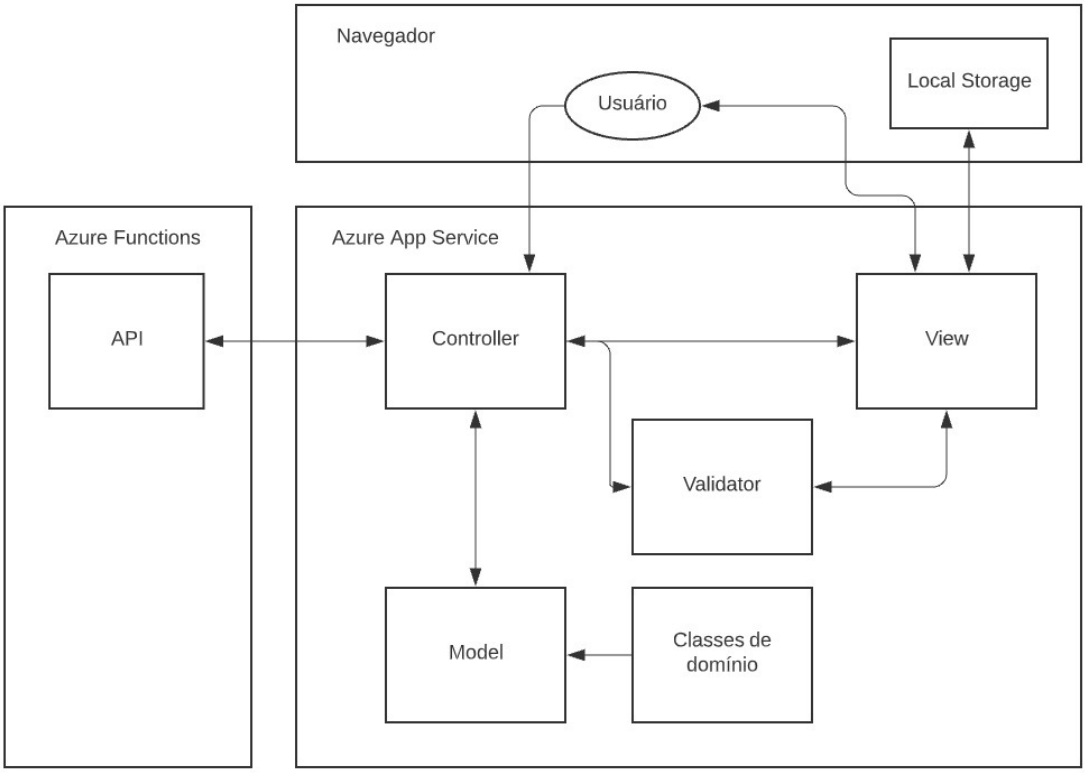
A Figura 7 é um diagrama de arquitetura que ilustra os principais componentes da aplicação. A *API* está hospedada no serviço Azure Functions[5]. e recebe uma requisição HTTP do *controller*. Essa requisição é enviada quando o usuário da aplicação pressiona o botão de classificação do paciente. Após receber a requisição HTTP, a RNA processa os dados do paciente, gerando uma saída que consiste em uma classificação de prioridade de atendimento. A *API* retorna a classificação do paciente ao controller que por sua vez, altera os demais componentes da aplicação.

Após receber a resposta da *API*, com a prioridade de atendimento do paciente que está aguardando a triagem, o controller executa um método que altera o atributo de prioridade do paciente na classe de Situação. O controller também atualiza a *view*, que por sua vez, atualiza a interface do usuário exibindo os dados alterados. Outro componente que interage com o con- troller é o validador (validator). Este componente é responsável por garantir que não ocorram erros causados por dados que violam as regras

[5] https://predictmanchester.azurewebsites.net/api/Predict

de negócio. Um exemplo de regra de negócio aplicada pelo validador é o número mínimo de caracteres necessários para o cadastro de um nome no sistema.

Figura 7 – Diagrama de arquitetura



Fonte: Elaborada pelos autores.

4.2 Testes unitários

Os testes unitários foram desenvolvidos utilizando a biblioteca xUnit. Foram testadas diretamente as validações das classes de Paciente, de Situação e de Prontuário. Foram testadas as validações de cadastros com múltiplos erros, erros em campos específicos e cadastros bem sucedidos.

Cada método de cenário de testes é definido pelo atributo *Fact*, que é responsável por indicar que o método demarcado precisa ser executado na rotina de testes.

Cada teste possui três componentes principais:

- Instanciar objeto com atributos essenciais: inicialmente é preciso definir o estado da uni- dade da aplicação a ser testada. Isso ocorre por meio da instanciação de objetos com os dados necessários para executar o cenário do teste.
- Instanciar validador: após a instanciação dos atributos essenciais, um objeto da classe de validador referente à classe sendo testada é instanciado. Ou seja, se a

classe de Prontuário está sendo testada, cada método de teste precisa ter acesso à instância de um objeto da classe de validador de Prontuário.

- Executar uma função *Assert*: a classe *Assert* é disponibilizada pela biblioteca xUnit. Esta classe concede acesso a algumas funções, como *False*, *True, Equal* e etc. Estas funções recebem o retorno do validador, que, por sua vez, recebem os objetos instanciados com os atributos essenciais para a realização do teste. Se o retorno do validador for o esperado pela função *Assert*, o teste retorna como sucesso. Em caso contrário, o teste retorna como falhado. Nos testes desenvolvidos, foram utilizados os Asserts *True* e *NotEmpty*.

A biblioteca utilizada para a realização das validações, FluentValidation, fornece algu- mas funções para realizar validações em cenários de teste, como a função *TestValidate*, que retorna o resultado de uma validação e a função *ShouldHaveValidationErrorFor*, que filtra os erros na validação por uma propriedade específica. Na Figura 8, consta a implementação de um teste da validação da classe Situação. Nesse teste, não se espera obter nenhum erro de validação.

Figura 8 – Teste de validação de Situação

```
[Fact]
| 0 referências | João Vítor, há 8 dias | 1 autor, 1 alteração
public void SituacaoValida()
{
    var situacao = new Situacao
    (
        id: Guid.NewGuid(),
        status: new Status(1, "Situacao"),
        prioridade: new Prioridade(0, "Urgente")
    );
    var validator = new SituacaoValidator();

    Assert.True(validator.TestValidate(situacao).IsValid);
}
```

Fonte: Elaborada pelos autores.

Na Figura 9, consta o teste da validação da propriedade Idade do Paciente, em que espera-se erros especificamente nessa propriedade.

Figura 9 – Teste de validação de Idade

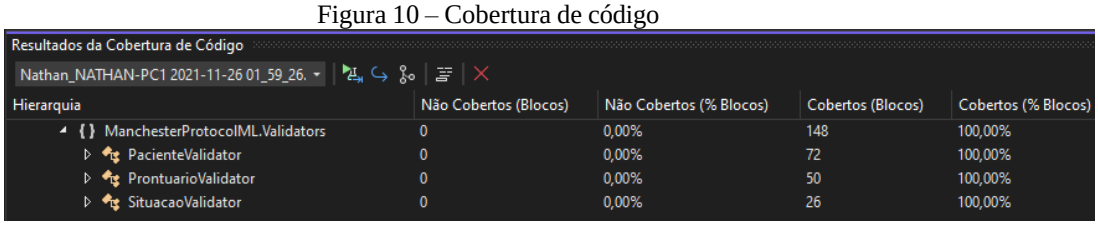
```
[Fact]
| 0 referências | Nathan, há 2 dias | 1 autor, 2 alterações
public void CadastroIdadeInvalido()
{
    var paciente = new Paciente
    (
        id: Guid.NewGuid(),
        nome: "Tom",
        sobrenome: "Hagem",
        idade: -1,
        horaDeEntrada: DateTime.Now,
        prontuario: new Prontuario
        (
            id: Guid.NewGuid(),
            frequenciaCardiaca: 59,
            saturacaoOxigenio: 98,
            frequenciaRespiratoria: 89,
            temperatura: 37,
            sintomas: new List<Sintoma>()
        ),
        situacao: new Situacao(Guid.NewGuid(), Statuses.EmConsulta, Prioridades.Urgente)
    );
    var validator = new PacienteValidator();

    Assert.NotEmpty(validator.TestValidate(paciente).ShouldHaveValidationErrorFor("Idade"));
}
```

Fonte: Elaborada pelos autores.

Indiretamente, foram testadas as classes de Prioridade, de Sintoma, de Status e de Tipo de Sintoma. Um exemplo de testes de classes indireto pode ser encontrado nos testes da classe de Situação. Esta classe possui atributos do tipo *Status* e do tipo Prioridade. Dessa maneira, ao se testar a classe de Situação, as classes de *Status* e de Prioridade são testadas indiretamente.

No relatório de cobertura de código do Visual Studio 2022, na Figura 10, é possível observar que os validadores estão 100% cobertos por testes unitários.

Figura 10 – Cobertura de código



Fonte: Elaborada pelos autores.

Foram realizados dezenove testes. O Quadro 3 apresenta o Plano de testes. A primeira coluna contém os testes que foram executados. A segunda coluna descreve quais cenários foram testados. Em seguida, a terceira coluna descreve o tipo do teste, podendo ser classificado como funcional ou não funcional. A última coluna descreve o objetivo do teste.

Quadro 3 - Plano de testes

Teste	Cenários testados	Tipo	Objetivo

Validar instancia do objeto de paciente.	1. Cadastro inválido. 2. Cadastro válido	Funcional	Testar o cadastro válido de um novo paciente
Validar campo de hora de chegada.	1. Hora válida 2. Hora inválida	Funcional	Testar o horário de classificação de prioridade igual ou superior ao horário atual.
Validar o prontuário do paciente	. Prontuário válido . Prontuário inválido	Funcional	Testar se o paciente possui um prontuário
Validar situação do paciente	1. Situação válida 2. Situação inválida	Funcional	Testar se o paciente possui uma prioridade e status definidos.
Validar campo de nome	1. Nome válido 2. Nome inválido	Funcional	Testar se o nome possui a quantidade mínima de caracteres.
Validar campo de sobrenome	1. Sobrenome valido 2. Sobrenome inválido	Funcional	Testar se o sobrenome possui a quantidade mínima de caracteres.
Validar campo de idade	1. Idade válida 2. Idade inválida	Funcional	Testar se a idade está dentro do intervalo de zero à cento e vinte e três anos.
Validar funcionamento em Chrome, Edge e Firefox	1. Testar Chrome 2. Testar Edge 3. Testar Firefox	Não funcional	Testar se os elementos da interface não são quebrados nos navegadores testados.
Validar a estabilidade da aplicação	1. Conexão estável 2. Sem conexão	Não funcional	Testar o comportamento da aplicação com uma conexão de internet estável e sem uma conexão de internet.

Fonte: Elaborado pelos autores.

4.3 Interfaces da aplicação

A Figura 11 representa a tela de cadastro de sintomas do paciente na aplicação. Desen- volvida com o objetivo de cobrir o RF1, esta tela é composta por oito campos de input. Os dois primeiros campos coletam o nome e o sobrenome do paciente que está sendo atendido. O terceiro campo coleta a idade do paciente. O quarto campo é do tipo data e este campo é preenchido automaticamente pela aplicação com a data e a hora em

que o processo de cadastro foi iniciado. Porém, isto não impede que ele seja alterado manualmente. Os campos de frequên- cia cardíaca, temperatura, saturação do oxigênio e frequência respiratória só recebem valores numéricos positivos. O campo de tipo de sintoma apresenta uma lista de tipos de sintomas pre- viamente cadastrados. O usuário deve escolher os tipos de sintomas que melhor representa a situação do paciente. Quando o usuário pressiona o botão de salvar, a prioridade do paciente é classificada automaticamente, cumprindo o RF2.

Figura 11 – Tela móvel de classificar pacientes

Fonte: Elaborada pelos autores.

A Figura 12 mostra a tela de listagem de pacientes classificados anteriormente, cum- prindo com o RF3. Está tela consiste de uma lista de classificação realizada pelo usuário. Cada item desta lista contém o nome do paciente, a prioridade de atendimento e se o paciente já foi atendido. Outra funcionalidade desta tela é a ordenação da listagem de acordo com o nível de urgência dos atendimentos, atendendo o RF4 do Quadro 1.

Figura 12 – Tela móvel de listar classificações anteriores

Fonte: Elaborada pelos autores.

Nos Apêndices A e B, estão descritos dois casos de uso desenvolvidos pelos autores. O caso de uso UC01 descreve o fluxo em que o atendente informa os sintomas do paciente na aplicação e o modelo classifica a prioridade de atendimento do paciente. Já o caso de uso UC02 é referente ao fluxo de execução da funcionalidade em que o atendente quer listar os casos classificados.

5 CONSIDERAÇÕES FINAIS

O Protocolo de Manchester é utilizado para determinar o tempo de espera de pacientes de acordo com a gravidade dos sintomas que devem ser tratados. Após um estudo com pacientes admitidos em uma unidade de pronto atendimento no Brasil, Anziliero et al. (2016) concluíram que o tempo médio de espera de um paciente durante a triagem, para determinar a prioridade de atendimento é de em média sete minutos. Este tempo pode variar de acordo com fatores como lotação da unidade de atendimento e com a experiência do profissional da saúde que está utilizando o Protocolo de Manchester. Segundo Silva et al. (2019b), a maioria dos enfermeiros que atuam na classificação da prioridade de atendimento não recebem nenhum treinamento com o protocolo. A automatização deste processo de classificação pode mitigar os efeitos causados pela falta de maturidade com o Protocolo de Manchester, gerando um impacto na média do tempo necessário para realização da classificação de prioridade. Estes poucos minutos economizados pela automatização podem representar horas de uma equipe de uma unidade de pronto atendimento durante o decorrer de um dia de trabalho, o que pode auxiliar os profissionais de saúde nos processos de tomada de decisão e beneficiar os pacientes que estejam em situação de risco e necessitem de uma intervenção imediata no Pronto Atendimento. Devido à limitação de tempo para o desenvolvimento do aplicativo, os autores deste trabalho não conseguiram validar a eficácia da aplicação quanto à proposta de diminuição do tempo de classificação. Também não foi possível validar a implementação do RNF2 de usabilidade, descrito no Quadro 2.

O *framework* ASP.NET foi utilizado para o desenvolvimento da aplicação. O projeto utilizou o padrão arquitetural MVC. Uma API foi desenvolvida para executar o modelo de classificação desenvolvido por Custodio (2021), que sugeriu a criação de uma API como con- tinuidade do seu trabalho. O modelo recebe os dados do paciente e

retorna uma prioridade de atendimento. A interface da aplicação é simples e intuitiva, para atender a expectativa de diminuição da curva de aprendizado de novos usuários.

Foram realizados testes unitários para garantir o funcionamento correto da aplicação. Os validadores se certificam que os campos de entrada de dados só recebem valores relevantes para a classificação da prioridade de atendimento do paciente.

Uma sugestão para trabalhos futuros é a aplicação de testes de usabilidade da aplicação desenvolvida neste trabalho. É sugerido que os testes sejam realizados com a participação de usuários da área de saúde, com o objetivo de validar as funcionalidades da aplicação e o impacto causado na alteração do tempo de classificação de prioridade de atendimento de pacientes.

Outra sugestão de trabalhos futuros é a pesquisa e desenvolvimento de uma API que possua um modelo de classificação que suporta requisições com um número maior de parâmetros de saúde do paciente. Caso está API seja desenvolvida, será necessária a adaptação do *controller* da aplicação para se conectar à nova API. Por fim, também é sugerido como trabalho futuro a adição de novas funcionalidades a está aplicação, por exemplo, a integração da fila de atendimento da aplicação com um sistema externo, representando o hospital.

REFERÊNCIAS

AMTHAUER, Camila; CUNHA, Maria Luzia Chollopetz da. Sistema de triagem de manchester: principais fluxogramas, discriminadores e desfechos dos atendimentos de uma emer- gência pediátrica. **Revista Latino-Americana de Enfermagem**, scielo, v. 24, 00 2016. ISSN 0104-1169. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid= S0104-11692016000100402&nrm=iso>.

ANICHE, Maurício et al. A validated set of smells in model-view-controller architectures. In: **2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)**. [S.l.: s.n.], 2016. p. 233–243.

ANZILIERO, Franciele et al. Sistema manchester: tempo empregado na classificação de risco e prioridade para atendimento em uma emergência. **Revista Gaúcha de Enfermagem**, SciELO Brasil, v. 37, n. 4, 2016. Disponível em: <https://www.scielo.br/scielo.php?pid= S1983-14472016000400417&script=sci_arttext&tlng=pt>. Acesso em: 10 de mai. 2021.

CUSTODIO, Luise Kelles. **Utilização de Redes Neurais Artificiais na Classificação de Risco e Prioridade em Pronto Atendimentos por meio do Protocolo de Manchester**. 2021. 36f. Monografia (Conclusão de curso) — Pontifícia Universidade Católica de

Minas Gerais.

FACELI, Katti et al. **Inteligência artificial uma abordagem de aprendizado de máquina**. 1. ed. Rio de Janeiro: LTC — Livros Tecnicos e Cientificos Editora Ltda, 2011.

FONSECA, MATEUS BECK. **Classificação do Transtorno Bipolar, Esquizofrenia e Depres- são Utilizando Redes Neurais Artificiais**. 2019. Tese (Doutorado) — Universidade Católica de Pelotas.

FOWLER, Martin et al. **Padrões de arquitetura de aplicações corporativas**. 1. ed. Porto Alegre: Bookman Editora, 2006.

FREEMAN, Adam. **Pro ASP.NET Core 3**. 8. ed. Califórnia: Apress, 2020. 1080 p. ISBN 978-1-4842-5439-4.

FREITAS, Igor Wescley Silva de. **Um estudo comparativo de técnicas de detecção de outliers no contexto de classificação de dados**. 2019. Dissertação (Mestrado) — Universidade Federal Rural do Semi-Árido, Programa de Pós-Graduação em Ciência da Computação, Mossoró.

GBCR. **Grupo Brasileiro de Classificação de Risco**. 2021. Disponível em: <http://gbcr.org. br>. Acesso em: 24 de abr. 2021.

HAYKIN, Simon. **Redes neurais: princípios e prática**. 2. ed. Porto Alegre: Bookman Editora, 2003.

IEEE Standard for Software Quality Assurance Processes. **IEEE Std 730-2014 (Revision of IEEE Std 730-2002)**, p. 1–138, 2014.

KALELKAR, Medha; CHURI, Prathamesh; KALELKAR, Deepa. Implementation of model- view-controller architecture pattern for business intelligence architecture. **International Jour- nal of Computer Applications**, IJCA, v. 102, 09 2014. ISSN 0975 – 8887. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.800.1517&rep=rep1&type=pdf>.

MICROSOFT. **Documentação do .NET**. 2020. Disponível em: <https://docs.microsoft.com/ pt-br/dotnet/fundamentals>. Acesso em: 24 de abr. 2021.

Ministério da Educação. **HU-Univasf adotará novo sistema de classificação para aten- dimentos**. 2016. Disponível em: <https://www.gov.br/ebserh/pt-br/comunicacao/noticias/ hu-univasf-adotara-novo-sistema-de-classificacao-para-atendimentos>. Acesso em: 25 de abr. 2021.

PRESSMAN, Roger S. **Engenharia de Software: Uma Abordagem Profissional**. 8. ed. Porto Alegre: AMGH, 2016. 968 p. Disponível em: <https://integrada.minhabiblioteca.com. br/books/9788580555349>. Acesso em: 25 de abr. 2021.

RIQUETTA, Atalaha Carvalho Barcellos. **Aplicativo para dispositivo móvel direcionado aos usuários das Unidades de Pronto Atendimento (UPA) 24 horas de Curitiba**. 2018. Disser- tação (Mestrado) — Universidade Tecnológica Federal do

Paraná.

RONCALLI, Aline Alves et al. Protocolo de manchester e população usuária na classifica- ção de risco: visão do enfermeiro. **Revista Baiana de Enfermagem**, v. 31, n. 2, 2017. Dis- ponível em: <https://cienciasmedicasbiologicas.ufba.br/index.php/enfermagem/article/view/ 16949>. Acesso em: 10 de mai. 2021.

SEIGER, N. et al. Improving the manchester triage system for pediatric emergency care: an international multicenter study. **PloS one**, v. 9, n. 1, 2014. Disponível em <https://journals.plos. org/plosone/article?id=10.1371/journal.pone.0083267>. Acesso em: 07 de set. 2021.

SILVA, Alessandra Dias Costa e et al. Caracterização dos atendimentos de um pronto-socorro público segundo o sistema de triagem de manchester. **Revista Mineira de Enfermagem**, reme, v. 23:e-1178, 02 2019. ISSN 2316-9389. Disponível em: <http://www.dx.doi.org/10.5935/1415-2762.20190026>.

SILVA, Maria Jamile Evangelista da; CEZÁRIO, Tanise de Lima; PEREIRA, Divinamar. Uti- lização do protocolo de manchester na classificação de risco no centro obstétrico. 2019. Dis- ponível em: <https://dspace.uniceplac.edu.br/handle/123456789/89>. Acesso em: 27 de abr. 2021.

SILVA, Paulo Henrique Pieczarka da; HILD, Tony Alexander. Desenvolvimento de uma inter- face comunicativa utilizando molic para auxiliar gestores da saude no monitoramento da situaç ao das famılias. 2019.

SILVEIRA, Paulo. **Introdução à arquitetura e design de software : uma visão sobre a pla- taforma Java.** 1. ed. São Paulo: Elsevier, 2012. ISBN 9788535250299.

SOMMERVILLE, Ian. **Engenharia de Software**. 10. ed. São Paulo: Pearson, 2019. 768 p. Disponível em: <https://plataforma.bvirtual.com.br/Leitor/Publicacao/168127/pdf/0>. Acesso em: 24 de abr. 2021.

Statcounter. **Desktop, Tablet Console Browser Market Share Brazil**. 2021. Dis-ponível em: <https://gs.statcounter.com/browser-market-share/desktop-tablet-console/brazil/ #yearly-2020-2020-bar>. Acesso em: 25 de nov. 2021.

XAVIER, Beatriz Rêgo. As categorias de aristóteles e o conhecimento científico. **Pensar-Revista de Ciências Jurídicas, v. 13, n. 1**, p. 57–64, 2008.

Apêndice A CASO DE USO 1

1. Id do caso de uso: UC01.
2. Atores: Atendente.
3. Sumário: O atendente entra os sintomas do paciente na aplicação e o modelo classifica a prioridade de atendimento do paciente.
4. Pré-condições: O atendente entra no formulário de cadastro de sintomas.
5. Fluxo de eventos: O atendente clica em cadastrar.

5.1 Fluxo principal:
1 - O atendente informa os 7 sintomas;
2 - O modelo processa os dados;
3 - O nível de prioridade é exibido na tela;
4 - O atendente confirma a operação.
5.2 Fluxos alternativos:
5.2.1 Pré-condição: O atendente cancelou a rotina.
1 – A aplicação pergunta se o usuário deseja cancelar o cadastro.
2 – Se sim, a aplicação retorna a página inicial. Se não, a aplicação permanece
na página atual.
6. Pós condições: A aplicação retorna para tela inicial.

Apêndice B CASO DE USO 2

1. Id do caso de uso: UC02.
2. Atores: Atendente.
3. Sumário: O atendente quer listar os casos classificados (CRUD).
4. Pré-condições: O atendente entra na opção de listar casos.
5. Fluxo de eventos: O atendente clica em listar casos classificados.
5.1 Fluxo principal:
1 – A aplicação lista todos os casos classificados em ordem cronológica;
2 – Primeiramente são exibidos resumos com o nome e prioridade do caso;
3 – O atendente pode clicar no caso e expandi-lo para exibir mais informações;
4 – O atendente aperta o botão de voltar/sair.
5.2 Fluxos alternativos:
5.2.1 Pré-condição: O atendente reordenou a ordem de exibição dos casos.
1 – O atendente clica na opção de “ordenar por”.
2 – A aplicação exibe os critérios de ordenação por ordem cronológica, prio- ridade e alfabética;
3 – O atendente seleciona uma das opções;
4 – A aplicação recarrega a lista listando os casos de acordo com o critério selecionado.
5.2.2 Pré-condição: O atendente quer excluir um caso.
1 – O atendente clica no ícone da lixeira no card do caso;
2 – A aplicação exibe a mensagem de confirmação de exclusão;
3 – O atendente confirma;
4 – A aplicação exclui o card.
5.3 Fluxos de exceções:
5.3.1 Pré-condição: Não existem casos cadastrados.
1 – A aplicação exibe a mensagem de “sem casos cadastrados”.
6. Pós condições: A aplicação retorna para tela inicial.