INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA
DE PERNAMBUCO - CAMPUS RECIFE

DEPARTAMENTO ACADÊMICO DE SISTEMAS, PROCESSOS E CONTROLES ELETRO-ELETRÔNICOS

CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

VANESSA LIMA DA COSTA FERREIRA

QUALIDADE DE SOFTWARE NO SISTEMA HEALTHNET: Elaborando e Automatizando Testes para o Módulo Segunda Opinião

Recife
2021

FICHA CATALOGRÁFICA

F383q
Ferreira, Vanessa Lima da Costa.
Qualidade de software no sistema Healthnet: elaborando a automatizando testes para o módulo segundo opinião / Vanessa Lima da CostaFerreira; orientador Prof. Dr. Ramide Augusto Sales - Recife, 2021.
63f.; il.
Trabalho de Conclusão de Curso (Tecnológico em Análise eDesenvolvimento de Sistemas) – IFPE – campus Recife.
. Inclui Referências.

1. Software 2. Qualidade. 3. Automação 4. Engenharia de software I.Ferreira, Vanessa Lima da Costa. II. IFPE. III. Título.

Catalogação na fonte elaborada pelo Bibliotecário Thiago Leite Amaro da Silva - CRB4/ 2.021

VANESSA LIMA DA COSTA FERREIRA

QUALIDADE DE SOFTWARE NO SISTEMA HEALTHNET: Elaborando e Automatizando Testes para o Módulo Segunda Opinião

Trabalho de Conclusão de Curso apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas, como pré-requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco IFPE, Campus Recife.

Orientador: Prof. Dr. Ramide Augusto Sales Dantas

Trabalho de Conclusão de Curso apresentado pela estudante **Vanessa Lima da Costa Ferreira** à coordenação de Análise e Desenvolvimento de Sistemas, do Instituto Federal de Pernambuco, sob o título de "***Qualidade de Software no Sistema HealthNet: Elaborando e Automatizando Testes para o Módulo Segunda Opinião*****"**, orientado pelo **Prof. Dr(a). Ramide Augusto Sales Dantas** e aprovado pela banca examinadora formada pelos professores[1]:

Recife, 16 de abril de 2021.

_____

Prof. Dr(a). Ramide Augusto Sales Dantas
CTADS/DASE/IFPE

_____

Prof. Dr(a). Vilmar Santos Nepomuceno
CTADS/DASE/IFPE

_____

Prof(a). Ms. Danilo Farias Soares da Silva
UNISÃOMIGUEL/UNINASSAU/IESO

_____

Aluno: Vanessa Lima da Costa Ferreira

[1] Excepcionalmente, a presente folha foi assinada única e exclusivamente pelo orientador do Trabalho de Conclusão de Curso (TCC), presidente da Banca Examinadora, em razão da necessidade de distanciamento social como uma das medidas de proteção e enfrentamento da emergência frente àpandemia – Covid-19.

Agradecimentos

A Deus que está sempre comigo.

Aos meus pais por estarem sempre ao meu lado, dando-me forças para superar minhas limitações e os desafios da vida. A minha família pelo apoio em todas as horas.

Ao meu orientador pela paciência, críticas e sugestões; pela confiança depositada e pela oportunidade que me proporcionou.

RESUMO

A qualidade de software é de grande importância para as empresas, sendo um desafio enfrentado por muitos gestores. Sempre que o sistema é atualizado, é necessário ter atenção redobrada para não gerar erros decorrentes das funcionalidades recém-criadas, ou fazer com que uma função já existente deixe de funcionar. Testes automatizados ganharam grande importância nos últimos tempos, principalmente no contexto das metodologias ágeis, pois prometem rapidez com menor intervenção humana, permitindo a reaplicação consistente dos testes, o que é importante para verificar se novos erros foram introduzidos, consequentemente aumentando o grau de confiança no software. Com base nisso, este trabalho se propôs a elaborar e aplicar testes automatizados no modulo do Segunda Opinião do sistema de telemedicina HealthNet, desenvolvido pelo NUTES/UFPE, visando facilitar o reteste caso novas modificações fossem realizadas. Para elaboração dos testes, este trabalho baseou-se nos requisitos funcionais já implementados no sistema. Para a realização dos testes foi usada como ferramenta de automação o Protractor, uma vez que o sistema usa Angular como *framework* de desenvolvimento para o modulo de Segunda Opinião. Ao final da bateria de testes realizados, o sistema se comportou de forma condizente com o esperado, atendendo a todos os requisitos levantados.

Palavras-chave: Qualidade de software. Testes automatizados. Engenharia de software.

ABSTRACT

Software quality is of great importance to companies, being a challenge faced by many managers. Whenever the system is updated, it is necessary to pay extra attention not to introduce errors resulting from the newly created functionalities or cause an existing function to stop working. Automated tests have gained great importance in recent times, mainly in the context of agile methodologies, as they promise speed with less human intervention, allowing the consistent reapplication of tests, which is important to verify if new errors were introduced, consequently increasing the degree of confidence in the software. Based on this, this work proposed to elaborate and apply automated tests in the Second Opinion module of the HealthNet telemedicine system, developed by NUTES / UFPE, aiming to facilitate the retest if new modifications were carried out. To prepare the tests, this work was based on the functional requirements already implemented in the system. To perform the tests, Protractor was used as an automation tool, since the system uses Angular as a development framework for the Second Opinion module. At the end of the battery of tests performed, the system behaved in a way that was expected, meeting all the requirements raised.

Keywords: Software quality. Automated tests. Software Engineering.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Os últimos anos viram a ascensão de aplicativos móveis, sites responsivos, serviços em nuvem, micro serviços, Big Data, inteligência artificial e Internet das Coisas. E nesse período de transformações, existe o desafio de garantir a resiliência das operações nos negócios. Nesse contexto, a qualidade do software deixa de ser uma opção para se tornar um requisito essencial para o sucesso do produto desenvolvido. (BARTIÉ, 2002).

A garantia de qualidade de um software é consequência da aplicação de testes que buscam encontrar erros que comprometam suas funcionalidades, sejam esses softwares comerciais ou não. Teste de software vem ganhando grande atenção nas empresas de Tecnologia da Informação (TI), as quais perceberam que a garantia de qualidade não é um gasto e sim um investimento(DIAS, 2008).

As empresas de software estão interessadas em melhorar cada vez mais a qualidade de seus sistemas, pois erros que são identificados e corrigidos antes dechegar aos clientes aumentam em satisfação do produto. Porém, testes mais eficazes pressupõem ciclos de testes mais longos e completos, o que tornao produto final mais caro o que podem inviabilizar o tempo de disponibilização do software no mercado e o custo do projeto. A automação de testes tem grande importância nesse contexto uma vez que, no longo prazo, os mesmos diminuem os custos com execução e melhoram o próprio processo de controle de qualidade do software.

Empresas do ramo de saúde também objetivam a qualidade de seus softwares, já que algumas tem por foco de atuação a medicina diagnostica,

Sendo revolucionarias em processos de diagnóstico e tratamento de doenças. A

medicina diagnóstica evoluiu bastante, com aumento na precisão nos diagnósticos, possibilitando uma maior exatidão na escolha do tratamento mais adequado para determinada patologia.

O Núcleo de Telessaúde Universidade Federal de Pernambuco (NUTES-UFPE) desenvolveu o sistema HealthNet para o projeto RedeNutes, e é uma plataforma de telemedicina que dá suporte ao Telediagnóstico e à Segunda Opinião Médica. Buscando melhorar a qualidade do seu software, o NUTES-UFPE vem efetuando contínuas melhorias. A versão atual (V.3) visa melhorar o desempenho do sistema efetuando uma mudança da linguagem usada no desenvolvimento do *front-end* do modulo Segunda Opinião.

Este trabalho tem por objetivo analisar e aprimorar os procedimentos de Testes e Qualidade de Software no módulo Segunda opinião da plataforma do Telessaúde HealthNet. Os objetivos específicos desse trabalho são: avaliar o processo dos testes na plataforma, mais precisamente no módulo Segunda opinião; realizar melhorias no processo, gerando casos de testes para serem executados; automação dos casos de teste; e por fim, execução dos testes e medição da qualidade do software.

Este trabalho está dividido em 5 capítulos:

- Capítulo 1 – Introdução: apresenta a introdução, problema, os objetivos e a justificativa do trabalho.
- Capítulo 2 – Revisão bibliográfica: este capítulo apresenta temas relacionados à Qualidade e Teste de Software, Casos de Testes e testes automatizados.
- Capítulo 3 – Conhecendo o sistema HealthNet.

- Capítulo 4 – Desenvolvimento dos casos de testes e apresentação da ferramenta utilizada (Protractor).
- Capítulo 5 – Apresenta as conclusões deste trabalho e sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Qualidade de Software

Existem muitas definições sobre qualidade que podem ser aplicadas diretamente a qualidade de software, então precisamos entender o conceito de qualidade e posteriormente o conceito de qualidade de software. Segundo a NBR ISO 9000:2005, "qualidade é o grau no qual um conjunto de características inerentes satisfaz aos requisitos".

A premissa básica sobre qualidade é que a qualidade do produto depende da qualidade do processo de desenvolvimento, desta forma, é comumque a busca por um software de maior qualidade passe necessariamente por uma melhoria no processo de desenvolvimento.

Segundo Bartié (2002): "Qualidade de software é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos".

Portanto, é necessário um planejamento adequado para que a qualidade de software seja atingida, conforme a definição de qualidade que deverá ser alcançada. Para isso são necessários modelos, padrões, procedimentos e técnicas para atingir essas metas de qualidade propostas, ou seja, seguir o ciclo de vida de um software. Para tanto, todas as etapas do ciclo de vida de engenharia de software devem ser contempladas com atividades que visam garantir a qualidade tanto do processo quanto do produto.

2.2 Ciclo de vida de software

Ciclo de vida de um projeto pode ser entendido como a divisão desse projeto em fases, do início até seu fim, para facilitar a realização complexa do todo. Com o crescimento do mercado de software as empresas passaram a adotar praticas que deram certo no processo de criação de software, modelos passaram a serem seguidos, assim surgiu a Norma ISO/IEC 12207, que entende ciclo de vida como “Estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema, desde a definição de seus requisitos até o término de seu uso.”

Arruda (2006) escreve que o objetivo da norma ISO/IEC 12207 é estabelecer uma estrutura comum para os processos de ciclo de vida de software. Os ciclos de vida se comportam de maneira sequencial (fases seguem determinada ordem) e/ou incremental (divisão de escopo) e/ou iterativa (retroalimentação de fases) e/ou evolutiva (software é aprimorado).

Com a movimento ágil inserido nas empresas desenvolvedoras de softwares, afetou diretamente a forma de entendimento do conceito e aplicação do ciclo de vida dos softwares, trouxe para os projetos mais flexibilidade, resultados e colaboração ao final de cada fase há uma reavaliação do trabalho que será e já foi realizado, sendo mais fácil e rápido diagnosticar e corrigir erros assim há um melhor gerenciamento dos projetos.

Dois dos sistemas mais utilizados atualment são o Kanban e o Scrum. O primeiro mais centrado no fluxo, são estabelecidas limitações muito claras para a aprovação das atividades (Work in Progress) e no último as iterações são mais

rápidas (entre 1 e 4 semanas) entre os sprints, ou seja, o trabalho que é realizado.

2.3 Testes de Software

O teste do software é uma das fases do processo de Engenharia de Software que visa atingir um nível de qualidade de produto. Existem várias definições para o conceito de teste de software.

Segundo Sommerville (2011, pag.144), por exemplo, “[o] teste é destinado a mostrar que um programa faz o que é proposto e para descobrir os defeitos do programa antes do uso. Quando se testa o software, o programa é executado usando dados fictícios. Os resultados do teste são verificados à procura de erros, anomalias ou informações sobre os atributos não funcionais do programa.”

Já Bartié (2002, pag.22) entende que: “Teste é um processo sistemático e planejado que tem por finalidade única a identificação de erros”. Nishimura (2011, pág.17) define: “teste de software abrange um conjunto de técnicas utilizadas para investigar defeitos em um sistema.” De uma forma simples, testar um software significa verificar através de uma execução controlada se o seu comportamento está de acordo com o especificado.

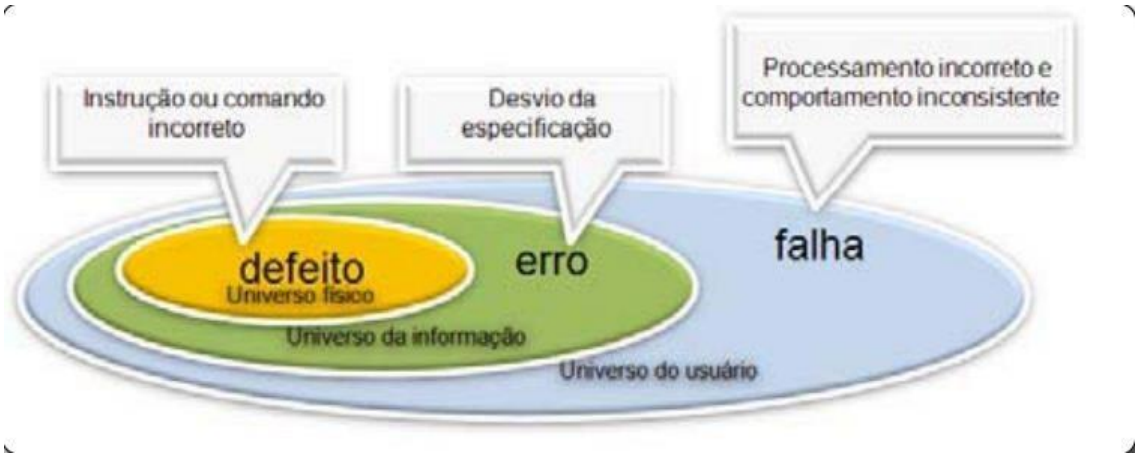
Mas para entender bem o processo de teste de software é preciso compreender as diferenças entre Defeitos, Erros e Falhas**,** pois são conceitos que se confundem ao longo do processo. Segundo o conceito padrão da engenharia de software do *Institute of Electrical and Electronics Engineers* – (IEEE 610, 1990):

- **Erro** é fruto da ação humana, que produz um resultado incorreto, como

uma falha na escrita de um código;

- **Defeito**, também conhecido como *bug,* é o resultado de um erro no código, gerando uma anomalia no funcionamento no sistema;
- **Falha**, por sua vez, é resultado da execução de um defeito no código.

Figura 1 *- Defeito, Erro e Falha*



Fonte: Dias Neto, 2008

A Figura 1 busca explicar visualmente a diferença entre esses conceitos. Defeitos fazem parte da aplicação propriamente dita (universo físico) e são causados por pessoas, por exemplo, através do uso incorreto do software. Defeitos podem gerar erros em um produto, ou seja, a construção de um software de forma diferente ao que foi especificado (universo de informação). Por fim, os erros geram falhas, que são comportamentos inesperados em um software que afetam diretamente o usuário final da aplicação (universo do usuário) e podem inviabilizar a utilização de um software.

As empresas têm grandes dificuldades na área de testes de softwares por vários fatores como prazos apertados, falta de mão de obra qualificada disponível, entre outros.

Segundo Libânio (2020), as empresas desenvolvedoras de softwares não

estão totalmente comprometida em garantir a qualidade de seus produtos digitais porque consideram isso um gasto e não um investimento. A qualidade é reduzida a apenas uma etapa do processo, que em geral é a realização de testes (manuais ou automatizados) apenas no fim do processo de desenvolvimento.

Os testes servem para verificar se o sistema conseguiu atingir determinadas especificações, descobrindo o maior número de defeitos possíveis. Porém o poder de decisão das companhias está muito nas mãos das áreas de negócio e como estão olhando constantemente para o mercado e querem inovar com velocidade. Neste contexto, o entendimento costuma ser que olhar para a qualidade é “perda de tempo”; é um gasto e não um investimento.

2.4 Atividades de Teste

Segundo Pressman(1997), Técnicas de verificação, validação e teste são atividades que estão intimamente relacionadas, as quais buscam garantir a qualidade do software. Aatividade de teste fornece evidências da confiabilidade do software em complemento a outras atividades, como por exemplo o uso de revisões, de técnicas formais e rigorosas de especificação e de verificação. É uma atividade relevante para a identificação e eliminação de erros que persistem.

Para garantir a qualidade dos testes, estes devem ser feitos de forma sistemática incluindo planejamento dos testes, projeto de casos de teste, execução e avaliação dos resultados dos testes. Essas atividades devem ser desenvolvidas ao longo do processo de desenvolvimento de software, e em geral, concretizam-se em três fases de teste: de unidade, de integração e de sistema.

Para este projeto foi implementado testes de sistemas sendo utilizados como validar os criterios de aceitação que foram definidos inicialmente no projeto e mantidos para o modulo Segunda Opção.

2.5 Níveis de Testes

Testes devem ser executados em diferentes níveis para avaliar o software em diferentes perspectivas de acordo com o produto gerado em cada fase do ciclo de vida de desenvolvimento de um software. Para cada fase do ciclo de vida do produto uma técnica de teste melhor se apresenta. Os principais níveis de testes são:

2.5.1 Teste de Unidade

Conhecido também como teste unitário, concentra-se em encontrar erros de lógica de programação e implementação em cada parte do software separadamente. Por lidar diretamente com o código, estes testes são executados pelos próprios desenvolvedores.

O teste de unidade enfoca a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente. Esse tipo de teste pode ser conduzido em paralelo para diversos componentes (Pressman, 2006).

2.5.2 Teste de Integração

Refere-se a integração das partes do que formam o todo do software, ou seja, a estrutura do programa testes com as interfaces dos módulos do software.

Para BARTIÉ (2002, p.149)

“O objetivo do teste de integração é validar as interfaces entre componentes da mesma arquitetura tecnológica. Essas interfaces representam as trocas dinâmicas de informação que ocorrem entre os componentes durante a execução de um software”.

Teste de integração para Pressman(2006) é uma técnica sistêmica para construir a arquitetura do software e ao mesmo tempo, conduz testes para descobrir falhas associadasa interface. Possui abordagem não incremental e incremental.

A não incremental é uma abordagem caótica, um conjunto de falhas é identificada e a correção é difícil, pois o isolamento das causas é complicado. Se contrapondo a isso temos a abordagem incremental, onde o software é testado em pequenos incrementos, sendo mais fácil a identificação e correção dos erros.

2.5.3 Teste de Sistema

O teste de sistema é realizado após a integração do sistema, busca identificar erros de funções e características de desempenho que não estejam de acordo com a especificação dos requisitos.

Rios e Moreira (2013, p. 172) explicam que: “o teste end-to-end é um tipo de Teste de Sistema que visa colocar o sistema à prova de uma forma mais completa a partir da simulação de um ambiente real”. O sistema é testado por completo ou uma parte dele, dentro de um ambiente controlado afim de validar a exatidão da execução de funções.

2.5.4 Teste de Aceitação

Os testes de aceitação validam se o software está pronto para ir para produção. Para avaliar isso, normalmente um cliente ou testador especializado, avalia se a aplicação está se comportando como esperado e pode ser considerada “pronta” (Rocha et al., 2001). Usualmente os testes de aceitação são feitos somente após a consolidação de todo o produto, ou seja, após todos os testes funcionais e não-funcionais terem sido executados e os erros encontrados no processo, corrigidos.

2.5.5 Teste de Regressão

Para Rocha (2001) teste de regressão tem por objetivo checar se as mudanças não causaram algum problema no software. Esse tipo de teste é realizado após modificações, atualizações, inclusão de funcionalidades ou ainda correções de defeitos. A cada nova versão do software ou a cada novo ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do software.

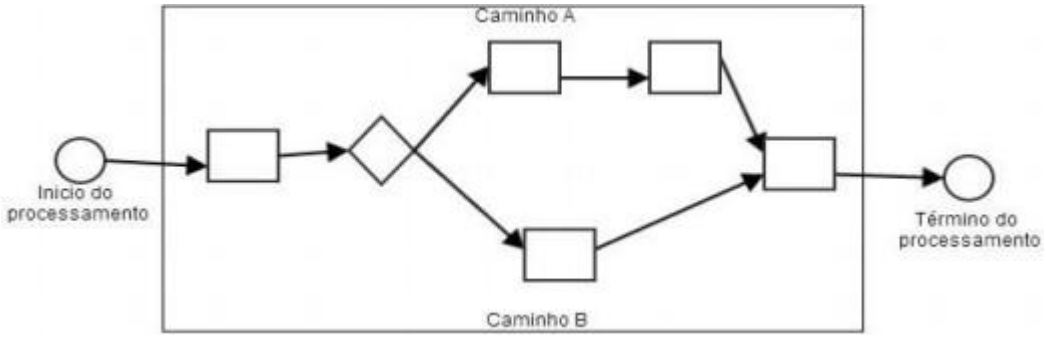
2.5.6 Teste Funcional

Técnicas de teste são classificadas de acordo com a origem das informações utilizadas para estabelecer os requisitos de teste contemplando diferentes perspectivas do software e obrigam a estabelecer uma estratégia de

teste que englobe as vantagens e os aspectos complementares dessas técnicas. Existem duas principais técnicas, a estrutural e a funcional.

A técnica de teste estrutural também chamada de teste Caixa Branca (figura 2). Essa técnica trabalha diretamente sobre o código fonte do componente de software para avaliar aspectos tais como: teste de condição, teste de fluxo de dados, teste de ciclos e teste de caminhos lógicos.

Figura 2 *- Técnica de teste Caixa Branca*



Fonte: Baseado em BARTIÉ (2002)

A técnica de teste funcional também chamada teste Caixa Preta são “[...] técnicas para garantir que os requisitos do sistema são plenamente atendidos pelo software que foi construído. Não é seu objetivo verificar como ocorrem internamente os processamentos no software, mas se o algoritmo inserido no software produz os resultados esperados.” (BARTIÉ, 2002, pag. 105).

O teste caixa baseia-se somente nos requisitos e especificações objetivando validar o comportamento do software em relação as funcionalidades de negócios documentadas e as especificações.

Figura 3 *- Visão sobre os testes de caixa preta*

Fonte: Baseado em BARTIÉ (2002)

Podemos entender da figura 3, os dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado conhecido previamente, o testador não tem necessariamente acesso ao código fonte do software. Esse tipo de teste reflete a ótica do usuário interessado em utilizar o programa, sem levar em conta os detalhes de sua construção.

2.6 Casos de Teste

Um caso de teste é um documento que descreve uma entrada, ação ou evento e uma resposta esperada, a fim de determinar se uma característica da aplicação está executando corretamente ou não. Um caso de teste deve conter identificação, objetivos, condições de entrada, sequência de passos e resultados esperados (Inthurn, 2001, pag. 78).

Sommerville (2011), acrescenta que os casos de testes são elaborados para exercitar adequadamente as estruturas do programa. Os casos de teste são utilizados em diversos seguimentos, os principais são:

- Na criação e implementação de esquemas de scripts de testes, fornecendo os pontos chaves de acordo com as implementações

dos requisitos no momento da codificação do software;

- Utilização de scripts para identificação dos melhores testes, tanto para testes manuais quanto para testes automáticos;
- Num controle dos números de teste específicos para abranger todo o software.

A qualidade de um caso de teste é descrita através de atributos como capacidade de encontrar defeitos, capacidade em exercitar mais de um aspecto reduzindo assim a quantidade de casos de teste requeridos, baseado no custo necessário para a realização do caso de teste incluindo o esforço de desenho, execução e análise dos resultados de teste e esforço de manutenção necessário sobre o caso de teste a cada alteração do sistema.

2.7 Testes Automatizados

Segundo Bartié (2002, pag. 196), para testes automatizados é feita “[...] a utilização de ferramentas de testes que possibilitem simular usuários ou atividades humanas de forma a não requerer procedimentos manuais no processo de execução dos testes.”

Automação dos testes consiste no uso de algum apoio computacional, ferramentas, para controlar a execução dos testes, a comparação dos resultados e comportamentos obtidos com a execução dos testes em relação aos resultados e comportamentos esperados, a configuração das pré-condições dos testes e outras atividades do controle dos testes e relato de seus resultados.

Existem diversas vantagens para a utilização dos testes automatizados,

dentre elas destacam-se menor tempo na execução dos testes, aumento da consistência e abrangência, redução dos custos e principalmente alcançar melhor qualidade de software.

Para cada software existe um tipo de automação de teste que melhor se adequa, os tipos de automação de testes são agrupados de acordo com a forma como que interagem com a aplicação. Em geral, os tipos de automação são agrupados em duas abordagens: baseados na interface gráficae na lógica do negócio.

2.7.1. Abordagem baseada em interface gráfica

Abordagem baseada em interface gráfica é quando os testes automatizados interagem com a interface gráfica do sistema simulando as ações de um usuário. Estas ações são gravadas por uma ferramenta com isso scripts são gerados e posteriormente podem ser reproduzidos. Existem três tipos de testes automatizados baseados na interface gráfica:

- Gravação/Execução (Capture/Playback) – os testes automatizados simulam usuários usando o sistema. Estas ações são gravadas e scripts são criados e depois reproduzidos.
- Dirigido a dados (Data-Driven) – são as mesmas ações dos testes anteriores, mas com uma variação nos dados, a cada novo teste novos dados são usados.
- Dirigido à palavra-chave (Keyword-Driven) - os testes são baseados em palavras-chaves (*keywords*). Cada palavra-chave é um comando em alto nível (praticamente em linguagem nativa) que representa uma ação do

usuário. Dessa forma, os testes são facilmente entendidos (e até escritos) pelos usuários finais em virtude do alto nível de abstração.

2.7.2. Abordagem baseada em lógica de programação

Os testes automatizados executam as funcionalidades da aplicação sem interagir com a interface gráfica. Os usuários interagem com a aplicação por meio de um prompt ou shell do sistema operacional.

2.8 Automação de testes

Automação de testes significa fazer o uso de um software que controlem a execução dos casos de teste. Segundo Bartié (2002, pag. 196), os testes automatizados são definidos como “[...] a utilização de ferramentas de testes que possibilitem simular usuários ou atividades humanas de forma a não requerer procedimentos manuais no processo de execução dos testes.”.

Na plataforma HealthNet a automação, baseada em interface grafica do tipo playback simulando ações do usuario, trará o benefício da agilidade em testar implementadas na versão 3. Estas alterações manterão a alta qualidade que o sistema exige sem adição de grandes recursos financeiros e de pessoal.

3 SISTEMA HEALTHNET

Segundo Barbosa *et al. (2003), o sistema* HealthNet surgiu como parte de um projeto maior, o Recife ATM, que tinha por objetivo a instalação, manutenção e estudo de uma rede de alta velocidade ATM no Recife. O Recife ATM visa, também, o desenvolvimento e estudo de aplicativos que utilizem esta rede. Dentre as áreas de aplicações encontram-se a Educação a Distância, Sistemas de geoprocessamento e Sistemas de Telemedicina.

O HealthNet é uma aplicação de telemedicina que dá suporte ao Telediagnóstico e à Segunda Opinião Médica. A melhoria da prestação de serviços de saúde em áreas distantes e carentes é uma de suas metas, além de permitir implantar um processo de cooperação médica entre grandes centros especialistas.

Estão envolvidos diretamente neste projeto os grupos de telemedicina e de gerência de redes do Recife ATM, o Centro de Informática (UFPE), o Setor de Tecnologias da Informação em Saúde (TIS) do Laboratório de Imunopatologia Keizo Asami (LIKA), o Hospital Das Clínicas da UFPE e o Real Hospital Português (RHP) de Beneficência em Pernambuco.

3.1 HealthNet - Módulo telediagnóstico

O serviço de telediagnóstico permite que médicos residentes em locais distantes e de poucos recursos possam interagir com médicos especialistas a fim de chegarem a um diagnóstico sobre seus pacientes. Através do HealthNet telediagnóstico, o médico solicitante recolhe um conjunto mínimo de informações

de seu paciente, que inclui informações textuais, imagens e vídeos, escolhe o médico ou setor de um dos Hospitais da Rede Integrada ao qual deseja encaminhar seu pedido de telediagnóstico e envia os dados, que ficam armazenados na base de dados do Hospital escolhido para o telediagnóstico.

Figura 4 *-Página inicial Sistema HealthNet*

Fonte: http://www.nutes.ufpe.br/produtos-e-servicos/(2018)

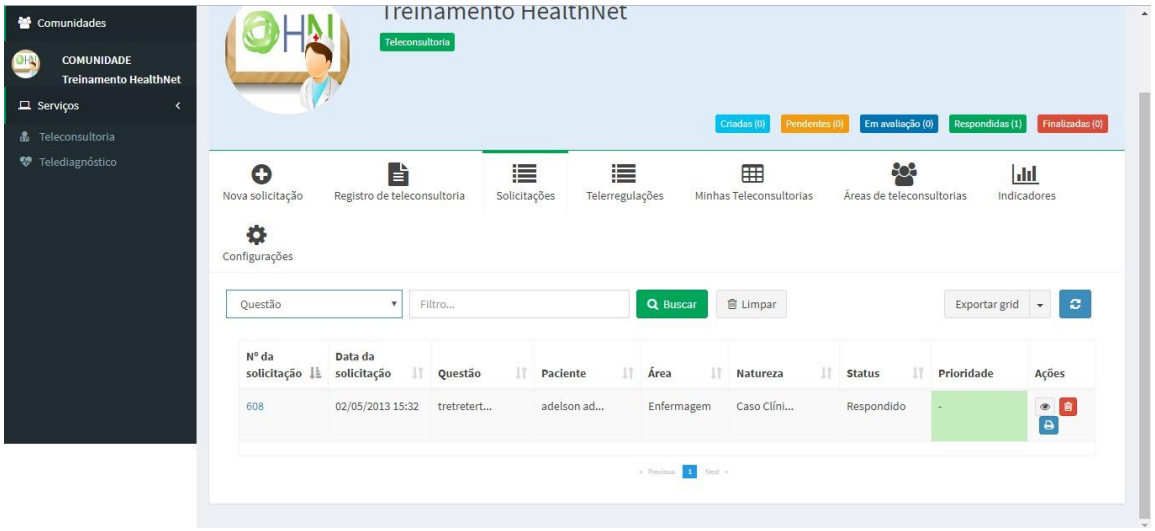
O médico especialista acessa a base de dados, avalia os dados do paciente e emite o seu parecer. O médico solicitante é notificado e estará apto a acessar o diagnóstico emitido pelo médico.

3.2 HealthNet - Módulo Segunda Opinião

Através do serviço Segunda Opinião, médicos especialistas podem discutir a respeito de casos de seus pacientes advindos do serviço de telediagnóstico ou criar seus próprios casos. Geralmente são colocados em discussão casos raros, de difícil diagnóstico ou que englobem mais de uma

especialidade médica. A figura 5, mostra uma solicitação no modulo segunda Opinião respondida.

Figura 5 *- Solicitação cadastrada no modulo Segunda Opinião*



Fonte: http://www.nutes.ufpe.br/hn3/(2018)

Um médico especialista, ao requisitar os serviços de segunda opinião, inicialmente escolherá os participantes daquela cooperação (médicos disponíveis que estejam participando da Rede Integrada). Posteriormente, os dados daquele paciente ficarão disponíveis a todos os participantes, os quais poderão interagir através de fórum de discussão, chats e videoconferência até chegar a uma conclusão sobre o caso, tendo uma camada de distribuição responsável por garantir a funcionalidade do serviço de segunda opinião.

4 CASOS DE TESTES

Este trabalho propõe aperfeiçoar a qualidade do HealthNet para melhor atender aos usuários e colaboradores não só para versão atual do site, mas também para futuras alterações implementadas. Pois a Rede Nutes nao possui uma equipe de qualidade de software que realize teste na plataforma, tal atividade é desempenhada pelos desenvolvedores.

Os testes automatizados foram executados no ambiente de desenvolvimento, naversão 3.0 atualizada disponível para uso por todos que tem acesso a plataformaHealthNet. Os testes foram executados apenas no modulo Segunda Opinião nacomunidade treinamento HealthNet onde os testes puderam ser realizados, semque houvesse interferência dos usuários do sistema, mas que fosse possível a simulação do ambiente real.

4.1 Criação dos Casos de testes

Segundo Bastos et al. (2007, pag. 154) os casos de teste, “[...] costumam derivar de uma especificação formal (caso de uso etc.). É necessário desenvolver casos de teste para cada cenário de caso de uso”. A fim de evitar um caso de teste para cada cenário foram usadas técnicas de refinamento para os casos de testes.

As técnicas de refinamento servem para identificar o menor número de cenários de testes que consigam cobrir a maior parte das situações. Aumentando assim, a eficiência do processo, uma vez que podem ser eliminados casos de

teste que seriam redundantes em uma situação em particular. (BARTIÉ, 2002, pag. 134).

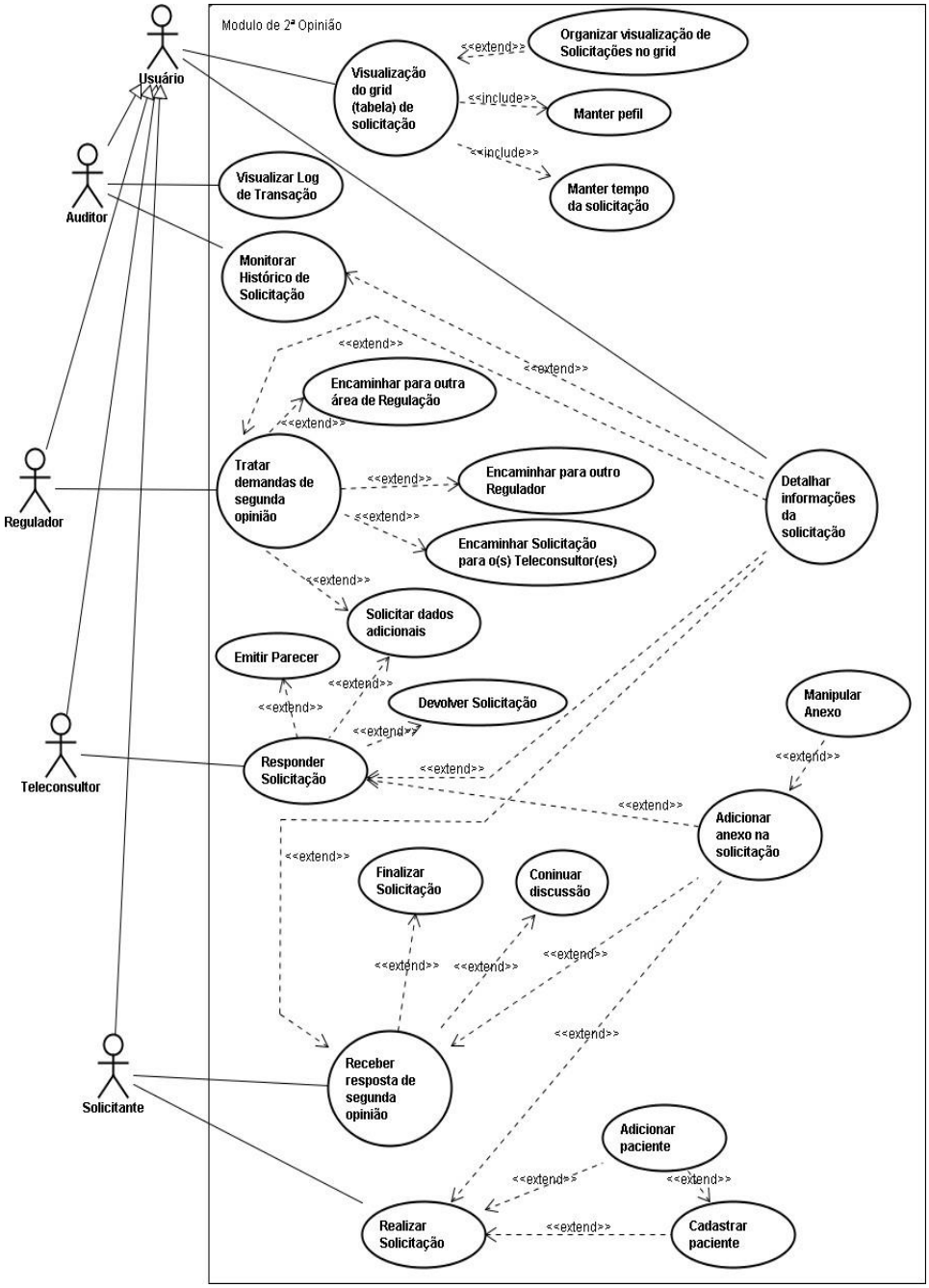
Para este trabalho foram usadas as técnicas de refinamento por partição de equivalência e análise de valor limite. Segundo Bartié (2002, pag. 134), essa técnica pode ser definida como “[...] um método que divide o domínio de entrada de dados em classes (grupos de valores). Cada classe representa um possível erro a ser identificado, permitindo que os casos de testes redundantes de cada classe sejam eliminados sem que a cobertura dos cenários existentes seja prejudicada.”

A análise de valor limite, por sua vez, é uma técnica que complementa a participação de equivalência, já que os casos de teste são selecionados nos limites de suas fronteiras de domínio. Além disso, a diferença é que essa técnica não somente foca a análise das condições de entrada como também das condições de saída. (BARTIÉ, 2002).

Os casos de testes foram desenvolvidos para validar as novas alterações existentes no sistema. Como descrito anteriormente, o Sistema HealthNet passou por uma alteração na linguagem usada para implementar o seu *front-end*, ou seja, a parte do sistema que é exibida ao usuário final. Os testes têm por finalidade encontrar erros e inconsistências do modulo de Segunda Opinião, sendo que as requisições de teste geradas devem ser feitas de forma automatizada a fim de reduzir o baixo custo com a execução de testes, além de facilitar o reteste.

A figura 6 apresenta o diagrama de caso de uso do fluxo que descreve o fluxo de atividades do sistema. A partir disso foram criados os casos de teste funcionais baseados do módulo Segunda Opinião.

Figura 6 *- Diagrama de Caso de Uso Segunda Opinião*



Fonte:Barbosa(2011)

Os papeis de cada tipo de usuário na figura 5 são os seguintes:

1. Auditor: responsável por fazer auditoria no sistema. Ele tem acesso ao histórico de dados e atividades realizada.
2. Regulador: responsável por regular as solicitações, encaminhando

para o Teleconsultor mais adequado.

3. Solicitante: realiza uma solicitação de segunda opinião.
4. Teleconsultor: é responsável por responder as solicitações.

4.2. Casos de testes

A seguir são descritos os casos de testes e condicionais de execução. Os casos de teste seguem uma sequência lógica do caminho típicos de ações realizado por um usuário básico, o Solicitante.

Quadro 1 – Caso de teste: confirmar teleconsultor logado

CT001	Confirma teleconsultor logado(Massa de teste :usuario com perfil teleconsultor e solicitante)
Pré-condição	Usuário deve ter um perfil (de solicitante) associado à segunda opinião;
Descrição	1. Usuário acessa link da aplicação digita usuário e senha;
Resultado Esperado	1. Sistema verifica usuário e senha; 2. Sistema exibe tela inicial do sistema.

Fonte: Ferreira(2019)

Quadro 2 - Caso de teste: Visualização do tabela de solicitação

CT002	Visualização do grid (tabela) de solicitação
Pré-condição	Usuário deve estar autenticado na aplicação como SOLICITANTE; Usuário deve ter um perfil (de solicitante) associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;
Descrição	1. O usuário seleciona o módulo de segunda opinião - Teleconsultoria.
Resultado Esperado	1. O sistema apresenta a tela de gerenciamento de segunda opinião através de uma tabela(‘Grid’) e a coloração (própria do sistema) em relação ao tempo e status. As solicitações devem vir ordenadas por data de solicitação, respeitando as restrições de tempo da solicitação.

Fonte: Ferreira(2019)

Quadro 3 - Caso de teste: Filtrar solicitação existente

CT003	Filtrar solicitações existentes
Pré-condição	Usuário deve estar autenticado na aplicação como SOLICITANTE; Usuário deve ter um perfil(de solicitante) associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;
Descrição	1. O usuário seleciona o módulo de segundas opiniões - Teleconsultoria. 2. O Usuário filtra os dados da tabela(‘grid’) inserindo informações no filtro sobre qualquer campo da tabela executando uma consulta.
Resultado Esperado	1. O sistema exibe tabela de solicitações; 2. O sistema apresenta a listagem de itens que satisfazem os critérios informados.

Fonte: Ferreira(2019)

Quadro 4 - Caso de teste: Ordenar solicitação

CT004	Ordenar solicitações
Pré-condição	Usuário deve estar autenticado na aplicação; Usuário deve ter um perfil associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;
Descrição	1. O usuário poderá ordenar os dados da tabela(‘grid’) de acordo com os campos, clicando no topo da coluna.
Resultado Esperado	1.O sistema ordena as solicitações de acordo com o campo selecionado.

Fonte: Ferreira(2019)

Quadro 5 - Caso de teste: Verificar exitencia de solicitação

CT005	Verificar existência de solicitações
Pré-condição	Usuário deve estar autenticado na aplicação; Usuário deve ter um perfil associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;
Descrição	1. O usuário seleciona modulo segunda opinião; 2. Visualiza a existência de itens na tabela
Resultado Esperado	1. O sistema exibe grid de solicitações.

Fonte: Ferreira(2019)

Quadro 6 - Caso de teste: Detalhar solicitação

CT006	Detalhar solicitação
Pré-condição	Usuário deve estar autenticado na aplicação - USUARIO REGULADOR; Usuário deve ter um perfil associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;
Descrição	1. O usuário seleciona o módulo de segundas opiniões. 2. O usuário visualizando a lista de solicitações no grid (tabela)clica no botão Visualizar solicitação dando um clique na solicitação Desejada
Resultado Esperado	1. O sistema valida a permissão do usuário segundo seu perfil solicitante e exibi as informações da solicitação conforme seu perfil.

Fonte: Ferreira(2019)

Quadro 7 - Caso de teste: Realizar solicitação

CT007	Realizar Solicitação
Pré-condição	Usuário deve estar autenticado na aplicação - USUARIO REGULADOR; Usuário deve ter um perfil associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;
Descrição	1. O usuário seleciona o módulo de segundas opiniões. O usuário visualizando a lista de solicitações no grid (tabela) clica no botão Visualizar solicitação dando um clique na solicitaçãodesejada
Resultado Esperado	2. 1. O sistema valida a permissão do usuário segundo seu perfil solicitante e exibi as informações da solicitação conforme seu perfil.

Fonte: Ferreira(2019)

Quadro 8 - Caso de teste: Não vincular paciente a solicitação

CT008	Não vincular paciente a solicitação.
Pré-condição	Usuário deve estar autenticado na aplicação; Usuário deve ter um perfil de Solicitante associado à segunda opinião; O usuário deve estar criando uma solicitação.
Descrição	1. O usuário seleciona o módulo de segundas opiniões. 2. O usuário visualizando a lista de solicitações no grid (tabela) clica no botão Visualizar solicitação dando um clique na solicitação desejada Clicar no botão vincular paciente não associar paciente.
Resulta Esperado	Visualizar solicitação sem paciente vinculado.

Fonte: Ferreira(2019)

Quadro 9 - Caso de teste: Vincular paciente através de uma busca

CT009	Vincular paciente através de uma busca
Pré-condição	Usuário deve estar autenticado na aplicação; Usuário deve ter um perfil de Teleconsultor associado à segunda opinião; Usuário deve estar criando ou respondendo uma solicitação.
Descrição	1. O usuário seleciona o módulo de segundas opiniões. 2. O usuário visualizando a lista de solicitações no grid (tabela) clica no botão Visualizar solicitação dando um clique na solicitação desejada; 3. Clicar no botão vincular paciente, efetuar busca por paciente e associar paciente.
Resultado Esperado	Buscar paciente em solicitação já existente.

Fonte: Ferreira(2019)

Quadro 10 - Caso de teste: Vincular e cadastrar novo paciente

CT010	Vincular e cadastrar novo paciente
Pré-condição	Usuário deve estar autenticado na aplicação; Usuário deve ter um perfil de Teleconsultor associado à segunda opinião; Usuário deve estar criando uma solicitação.
Descrição	Usuário deve criar uma solicitação e vincular um paciente.
Resultado Esperado	Paciente associado com sucesso.

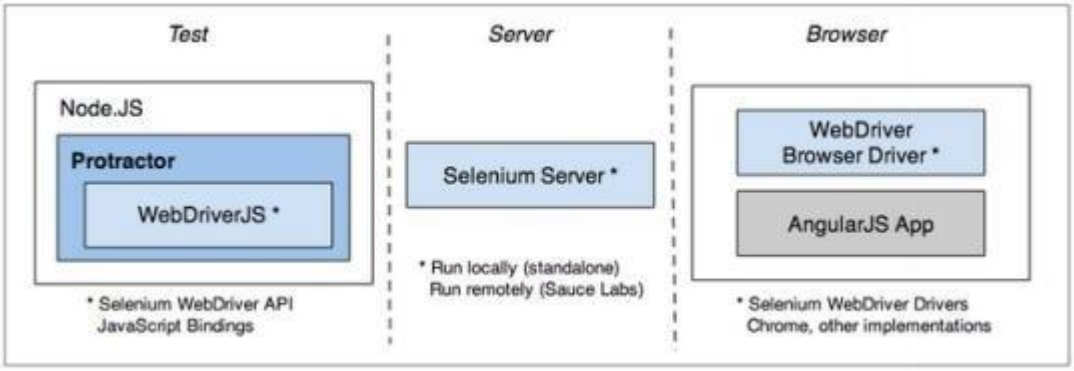
Fonte: Ferreira(2019)

4.3 Automação com o Protractor

O Protractor[1] foi o *framework* escolhido,dentre outras no mercado, como ferramenta para os testes aserem feitos no site responsivo do HealthNet. A escolha se deu devido a aplicaçãoser desenvolvida em AngularJS, *framework* Web para o qual o Protractor melhor se destina, além de ser de fácil manuseio e aprendizagem. Trata-se de um *framework* de testes funcionais *end-to-end* em JavaScript e funciona com a combinação de várias ferramentas como NodeJS[2], Webdriver, Selenium e outrascomo podemos melhor visualizar na figura 7.

[1] Site oficial do Protractor: https://www.protractortest.org/#/
[2] Site oficial do NodeJs: https://nodejs.org/en/

***Figura 7** -Infraestrutura provida pelo Selenium e Protractor*



Fonte: https://www.protractortest.org/#/infrastructure(2019)

O Protractor empacota WebDriverJS, que são ligações JavaScript para o API do SeleniumWebDriver. Comandos do WebDriverJS são assíncronos, ou seja, eles são programados no fluxo de controle e retornam um resultado prometido, e não valores primitivos. Os scripts de testes são enviados para o Selenium Server, que se comunica com o driver dos navegadores.

4.4 Execução dos Casos de Testes

Para executar um teste, o Protractor faz uso de dois arquivos:

- Arquivo de configuração (*configuration file*, conf.js);
- Arquivo de especificação (*spec file,* spec.js);

No arquivo de configuração é onde colocamos todo o tipo de configuração que vamos utilizar nos testes. O arquivo de configuração especifica o *framework* de testes utilizado, onde o servidor Selenium está (URL, e.g.: *http://localhost:4444/wd/hub*), qual é o arquivo de teste a ser executado (spec.js), o browser, a URL padrão que vamos acessar, a pasta na qual estão nossos testes, a pasta onde estão as evidências, que estão os prints de tela de cada

caso de teste que podem ser observados no apêndice ao final deste trabalho, e o relatório com os resultados dos testes. Um exemplo de arquivo de configuração pode ser observado a seguir:

Figura 8 *-Arquivo de configuração, configuration file*

```
var HtmlScreenshotReporter = require('protractor-jasmine2-screenshot-reporter');
var reporter = new HtmlScreenshotReporter({  dest: './prints',    filename: 'my-
report.html', userCss: 'my-report-styles.css'});
exports.config = {
        seleniumAddress: 'http://localhost:4444/wd/hub',
        capabilities: {
         browserName: 'chrome'
         },
//baseUrl: 'https://150.161.30.35:8443/hn3/login.html',      framework:jasmine2',
jasmineNodeOpts: { // opções passada para o Jasmine - Node
showColors: true, //Utiliza cores na linha de comando para reportar
defaultTimeoutInterval: 400000 //Tempo de espera antes do sistema apontar TimeOut},
useAllAngular2AppRoots: true, //Para o protractor identifcar os comandos do Angular 2
        specs: ['spec.js'],
         logLevel: 'WARN',
        onPrepare: function () {
var SpecReporter = require('jasmine-spec-reporter').SpecReporter;

        jasmine.getEnv().addReporter(new SpecReporter ({
        displayFailureSummary: true,
        displayFailedSpec: true,
        displaySuiteNumber: true,
        displaySpecDuration: false,
        displayStacktrace: true
        }));
        },
        onPrepare: function () {
                jasmine.getEnv().addReporter(reporter);
                        }
        }
```

Fonte: protractor (2019)

O arquivo *spec.js* é onde está descrita a suíte de testes. Ele inicia com os testes com o *describe* (do Jasmine), onde é definido uma suíte de testes, passando como primeiro parâmetro uma descrição, título do teste e o segundo parâmetro é uma função de *callback* onde vão estar os testes daquele *describe*. Na função *beforeEach* do Jasmine é declarado tudo que deve ser executado

antes de cada *it.* Um *it* (do Jasmine) é o teste em si que vai ser executado. Ele recebe dois parâmetros também, uma descrição para o teste e um *callback*.

Figura 9 *- Arquivo de especificação, spec file*

```
/**
   @file spec.js*/

browser.ignoreSyncronization = true;
useAllAngular2AppRoots: true; //Para o protractor identifcar os comandos do Angular 2

describe ('HealthNet', function () {
        beforeEach (function(){
         browser.ignoreSynchronization = true;
         // A página de login não é desenvolvida em angular, então é necessário para
         // a sicronização para o protractor não ficar procurando as tags do Angular
         browser.get('https://150.161.30.35:8443/hn3/login.html');
         var EC = protractor.ExpectedConditions;

        //Espera até o elemento de login aparecer para poder continuar
         browser.wait(EC.presenceOf(element(by.id('cpf'))));

        // localiza input com ID de Login
         var login = browser.driver.findElement(by.id('cpf'));
        // Localizando input de ID Password
         var senha = browser.driver.findElement(by.id('password'));
         // localização botão com texto de Entrar
         var btnEntrar = element(by.className('btn-login'));
         // login de acesso e // senha do sistema
         login.sendKeys('AB123458');senha.sendKeys('nutes123');
         btnEntrar.click().then(function(){
         // espera o aside da outra página aparecer
         browser.wait(EC.presenceOf(element(by.css('aside'))));
         });
        });

       beforeEach (function(){
browser.get('https://150.161.30.35:8443/hn3/#/comunidade/140/servico/teleconsultoria');
        });

        it ('confima teleconsultor logado', function(){
                browser.driver.sleep(2000);
                var usuario_logado = $('.user-panel');
                expect(usuario_logado.getText()).toContain('Teleconsultor Treinamento');
         });

        it ('Visualização do grid de solicitação',function(){
                let list = element.all(by.css('#solicitacoes-teleconsultoria td a'));
                expect(list.isPresent()).toBe(true);
                //expect(list.isPresent()).toBeTruthy();
        });

        it('Filtrar solicitações existentes',function(){//
                var btnBuscar = element(by.className('btn btn-success'));
                expect(element(by.cssContainingText('option', 'Natureza'))).click();
        });

        it('Verificar existencia de solicitações',function(){
                let  dadosSolicitacao  =  element(by.cssContainingText('.box-header  .box-
title', 'DADOS'));
                let   btnvisualizar   =   element.all(by.css('#solicitacoes-teleconsultoria
tbody tr td button .fa .fa-eye'));
                btnvisualizar.click().then(function(){

        browser.get('https://150.161.30.35:8443/hn3/#/comunidade/140/servico/teleconsulto
ria/608');
         });
        expect(dadosSolicitacao.isPresent()).toBe(true);});
        it('Realizar Solicitação',function(){
                let newSolicitacao = element(by.partialLinkText('nova'));
                newSolicitacao.click();
                browser.driver.sleep(2000);});});
```

Fonte: protractor (2019)

4.5 Relatório de Execução dos Testes

A seguir é visto um resumo da execução dos testes pelo Protractor cada linha é um caso de teste, “✓”check que sinaliza que o teste passou com sucesso e “✗”evidenciando que o teste não passou, apresenta erro.

Figura 10 *– Resultado de execução script*

HealthNet (239 s)

- ✓confima teleconsultor logado(41 s)
- ✓Visualização do grid de solicitação(22 s)
- ✗Filtrar solicitações existentes(22 s)
  - Failed: expect(...).click is not a function [stack]
- ✓Verificar existencia de solicitações(24 s)
- ✗Realizar Solicitação(22 s)
  - Failed: No element found using locator: By(partial link text, nova) [stack]
- ✓Não vincular paciente a solicitação(22 s)
- ✓Vincular paciente através de um busca(27 s)
- ✓Vincular e cadastrar novo paciente(20 s)
- ✓Receber resposta de segunda opinião(20 s)
- ✓Adicionar anexo na solicitação(20 s)

Toggle Configuration

4.5.1. Análise dos Resultados

Como podemos ver o caso teste 3 apresentou uma falha no *expect*, mas é apenas um erro de compilação, causada pela lentidão da internet no momento da execução do teste. Os testes automatizados obtiveram um alto índice de aproveitamento, devido ao curto tempo de execução como exibido no relatório, gerado pela própria automação, apenas 239 segundos, ou 3 minutos e meio.

O sistema HealthNet em sua versão 3.0 tem se preocupado muito com a qualidade tendo em vista a complexidade dos dados aos quais o sistema manipula. Apesar do baixo tempo de execução dos testes, isso não pode ser comparado ao seu tempo de desenvolvimento dos casos de teste. Foi despendido um longo tempo para o desenvolvimento dos scripts de teste, e o framework utilizado para elaboração e execução dos testes nao dispoem de um vasto acerto de materiais para apoio já que não há uma equipe de suporte para possiveis questionamentos, sendo este debito tecnico um fator de dificuldade.

Somando todos esses empecilhos ainda se pode perceber os grandes benefícios que traria para o sistema HealthNet testes automatizados, quantos benefícios para qualidade seriam possíveis de serem alcançados.

Antes das considerações finais alguns itens merecem ser pontuados. Automatizar os testes somente em classes que façam sentido para o projeto, como as funcionalidades principais, foco do negócio. Automações necessitam de manutenção e há um custo em automatizar. Utilização de padrões de projeto para os testes, como Page Objects, pois assim como os padrões de projeto são interessantes para resolver problemas em classes de produção,

independentemente do nível de teste, cada caso de teste precisa ser o mais autônomo possível, pois um caso de teste pode ser executado independente da ordem que foi chamado em uma classe.

Para plataforma HealthNet ainda é necessário a aplicação dos testes automatizados em outras partes do sistema como por exemplo o modulo de tele atendimento, porém já é possível visualizar o quão beneficente a automação seria para os testes de inserção, alteração e exclusão no sistema, pois demandaria um tempo reduzido com testes funcionais manuais. E assim poder continuar com a preocupação em manter a usabilidade associada com alta qualidade e um menor tempo de resposta para as solicitações de segunda opiniões que venham a surgir.

5 CONCLUSÃO E TRABALHOS FUTUROS

Este Trabalho de Conclusão de Curso procurou destacar a importância da qualidade em software como forma de manter as empresas que os desenvolvem sempre competitivas no mercado, enfatizando que a garantia de qualidade não deve ser vista como um custo a mais no projeto de desenvolvimento de um software, mas sim como um investimento.

Para obter um software de qualidade é preciso se estar sempre atento a questão dos Testes de Software. A ausência dos testes ou sua realização de maneira equivocada pode custar caro a um projeto de software, pois os sistemas são desenvolvidos por pessoas e pessoas são suscetíveis a erros. Existe uma grande variedade de tipos de testes que utilizam técnicas projetadas para diferentes partes de um software, desde a sua estrutura até suas funcionalidades. Este trabalho abordou principalmente os testes funcionais por serem testes que avaliam se o produto final realiza o que foi proposto inicialmente, contudo sem deixar de ressaltar a importância dos testes estruturais e não-funcionais.

Considerando que os testes de software são um dos requisitos para se obter um software de boa qualidade, a revisão bibliográfica buscou evidenciar a importância dos testes em diversas fases e níveis de um projeto de desenvolvimento. O conhecimento levantado então foi aplicado no estudo da qualidade do Sistema HealthNet, módulo Segunda opinião. Os casos de testes produzidos a partir dos casos de uso do sistema, e executados usando a ferramenta Protractor. Estes testes automatizados serviram como forma de

viabilizar o processo de teste além de possibilitar seu reaproveitamento em trabalhos futuros.

Os resultados dos testes realizados no módulo Segunda Opinião identificaram uma pequena falha de compilação e não erro de funcionalidades. Um erro encontrado no momento da extração de relatório deste, um erro não esperado, reafirmando a importância dos testes que mesmo sob um ambiente monitorado ainda assim surgem erros. De toda forma, foi possível perceber que o sistema apresentou um nível alto de qualidade, ficando os testes funcionais desenvolvidos servindo de base para futuros aperfeiçoamentos do sistema.

Como trabalho futuro, vislumbra-se a adequação dos testes para *Page Objects* objetivando a redução de código duplicado. Isso trará ao projeto benefícios como o desenvolvimento de código limpo, a reutilização de código e para tornar os testes em si mais fáceis de serem lidos.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ARRUDA, Sérgio. **ISO/IEC 12207 Processos Fundamentais**. Disponível em:http://www.plugmasters.com.br/sys/materias/539/1/ISO%7B47%7DIEC-12207-Processos-Fundamentais.Acesso em: 03 maio 2021.

BARBOSA A. K. P.; NOVAES M. A.; VASCONCELOS A. L. A Web Application to Support Telemedicina Services in Brazil. In: AMERICAN MEDICAL INFORMATICS ASSOCIATION ANNUAL SYMPOSIUM(AMIA), 2003, Washington. **Proceedings**, 2003. p. 56-60.

BARTIÉ, Alexandre. **Garantia da qualidade de software:** adquirindo maturidade organizacional. Rio de Janeiro: Campus, 2002.

DIAS NETO, A**. Introdução a Teste de Software**. Engenharia de Software Magazine, n. 01, p. 54-55, 2018.

FARIAS, S. S. D.; DE SÁ, C. F. A.; NOVAES, D. A. M.;GOMES, A. Uma proposta Web para melhor formação de profissionais de saúde a distância. In: CONGRESSO BRASILEIRO DE ENGENHARIA BIOMÉDICA(CBEB), 22., 2010, Tiradentes, MG. [**Anais**...] Tiradentes: CBEB, 2010. Disponivel em: http://cbeb.linkedej.com.br. Acesso em: 30 Dez. 2018.

LIBÂNIO, Grace. Os desafios de se implementar uma política de quality assurance dentro do desenvolvimento. **Computerworld**,2020. Disponível em:https://computerworld.com.br/negocios/os-desafios-de-se-implementar-uma-politica-de-quality-assurance-dentro-do-desenvolvimento/. Acesso em: 04 mai. 2021

NISHIMURA GONÇALVES, H. **Geração de testes Automatizados Utilizando o Selenium**. Trabalho de conclusao de curso (Bacharel em Ciência da Computação) Universidade de Pernambuco, 2011.

PRESSMAN, R. **Engenharia de Software**. 6.ed. Rio de Janeiro : McGraw-

Hill, 2006.

RIOS, Emerson; MOREIRA, Filho; TRAYAHÚ, R . **Teste de Software**. 3. ed. Rio de Janeiro: Alta Books, 2013.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. et al. **Qualidade de software – Teoria e prática**. São Paulo: Prentice Hall, 2001.

SOMMERVILLE, Ian**. Engenharia de software**. 9a edição. São Paulo: Pearson Prentice Hall, 2011.

STAMFORD, P.; Barbosa, A.; Novaes, M.; Belian, R. Um Sistema de Telediagnóstico para a Rede Recife ATM. In: CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE(CBIE), 8., 2018, Fortaleza. [**Anais...]** Fortaleza:SBC, 2018. Disponivel em: https://www.sbc.org.br/calendario-de-eventos/evento/241/vii-congresso-brasileiro-de-informatica-na-educacao-cbie-2018. Acesso em: 09 abr. 2019.

TEIXEIRA, T. 5 ferramentas de automação de testes para projetos digitais. **Tiinside.** Disponível em: http://tiinside.com.br/tiinside/services/14/10/2016/5- ferramentas-de-automacao-de-testes-para-projetos-digitais/. Acesso em:10 jan. 2019.

7 APÊNDICE

A seguir constam a planilha de testes bem como as evidências geradas por cada caso de teste pela automação durante a execução dos testes.

7.1 PLANO DE TESTES

Planilha com casos de testes que foram criados baseados nos casos de uso.

Quadro A-1 – Planilha com casos de teste

ID	Modulo	Cenário	Pré-condição	Descrição	Resultado Esperado
CT01	Segunda opinião - SOLICITANTE	Confirma teleconsultor logado	Usuário deve ter um perfil(de solicitante) associado à segunda opinião;	1. Usuário acessa link da aplicação digita usuário e senha ;	1.Sistema verifica usuário e senha; 2.Sistema exibe tela inicial do sistema.
CT02	Segunda opinião - SOLICITANTE	Visualização do grid (tabela) de solicitação	Usuário deve estar autenticado na aplicação Como SOLICITANTE; Usuário deve ter um perfil(de solicitante) associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;	1.O usuário seleciona o módulo de segunda opinião - Teleconsultoria.	1.O sistema apresenta a tela de gerenciamento de segunda opinião através de uma tabela('Grid') e a coloração em relação ao tempo e status. As solicitações devem vir ordenadas por data de solicitação, respeitando as restrições de tempo da solicitação.

CT03	Segunda opinião - SOLICITANTE	Filtrar solicitações existentes	Usuário deve estar autenticado na aplicação Como SOLICITANTE; Usuário deve ter um perfil(de solicitante) associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;	1.O usuário seleciona o módulo de segundas opiniões - Teleconsultoria. 2. O Usuário filtra os dados da tabela('grid') inserindo informações no filtro sobre qualquer campo da tabela executando uma consulta.	1. sistema exibe tabela de solicitações; 2.O sistema apresenta a listagem de itens que satisfazem os critérios informados.

CT04	Segunda opinião	Verificar existência de solicitações	Usuário deve estar autenticado na aplicação; Usuário deve ter um perfil associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;	1.O usuário seleciona modulo segunda opinião; 2.Visualiza a existência de itens na tabela	1.O sistema exibe grid de solicitaçõe.
CT05	Segunda opinião	Detalhar solicitação	Usuário deve estar autenticado na aplicação - USUARIO REGULADOR; Usuário deve ter um perfil associado à segunda opinião; Existir pelo menos uma solicitação no perfil visualizado;	1.O usuário seleciona o módulo de segundas opiniões. 2.O usuário visualizando a lista de solicitações no grid (tabela) Clica no botão Visualizar solicitação dando um clique na solicitação desejada	1.O sistema valida a permissão do usuário segundo seu perfil **solicitante** e exibi as informações da solicitação conforme seu perfil ..

CT06	Segunda opinião	Realizar Solicitação		1.O usuário clicar em “Nova Solicitação”, 2.O usuário no campo “Dados do Paciente” seleciona “sim” para vincular paciente; 3. O usuário na tela “Associar um Paciente” e clica em “Listar Todos”; 4.O usuário seleciona um paciente apresentado na lista e clicar em “Adicionar”; 5.O usuário clica em “Sim”; 6. O usuário no campo “Dados da Solicitação” deve preencher todos os campos obrigatório [RN-04-004], podendo anexar arquivos para melhora o entendimento da solicitação. Após o preenchimento da solicitação, o usuário clica em “Enviar”.	1.O sistema apresenta um *panel* (Aba) “Nova Solicitação” com os campos preenchidos de “Dados do Solicitante” e os campos de “Dados do Paciente” e “Dados da Solicitação” a ser preenchido; 2. O sistema apresenta a tela “Associar um Paciente” com botões para poder listar todos os pacientes, cadastrar um novo paciente e buscar paciente através de seus parâmetros; 3. O sistema apresenta a listagem de itens com todos os pacientes cadastrados; 4. O sistema apresenta um pop up de aviso com mensagem; 5.O sistema retorna a tela “Nova Solicitação” com os dados do paciente cadastrado nos campo “Dados do Paciente”, precisando preencher os campos de “Dados da Solicitação” para terminar a solicitação; 6. O sistema verifica se os campos obrigatórios foram devidamente preenchidos;

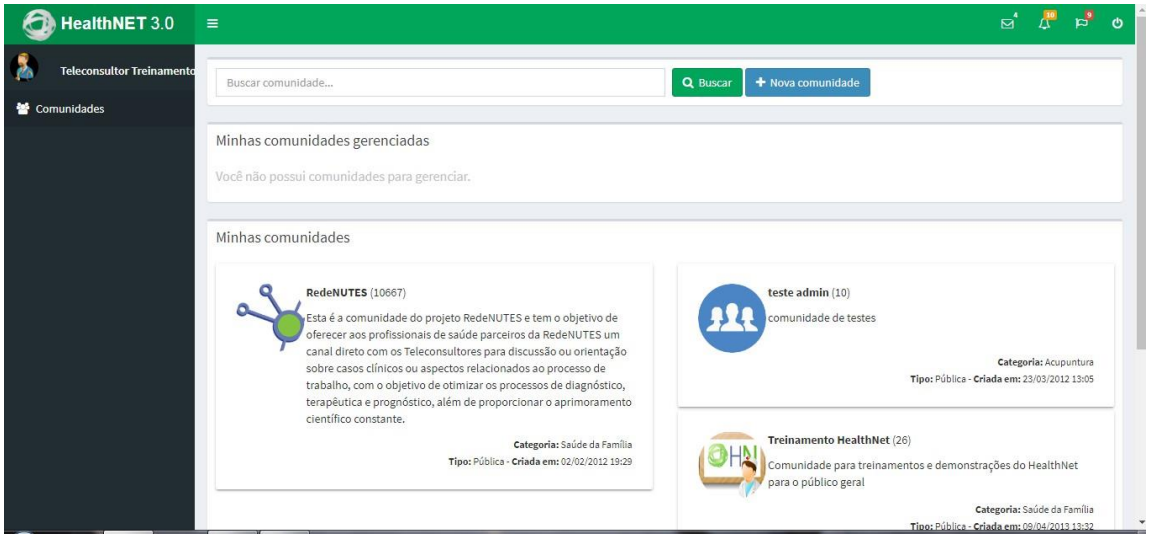
					7.O sistema emiti a solicitação para os reguladores, realizando a criação da solicitação com a definição do status e inicialização do tempo de resposta.
CT07	Segunda opinião	Não vincular paciente a solicitação		1.O usuário no campo “Dados do Paciente” seleciona “não” para vincular paciente e no campo “Dados da solicitação” o usuário deve preencher todos os campos obrigatório, podendo anexar arquivos para melhora o entendimento da solicitação. Após o preenchimento da solicitação, o usuário clica em “Enviar”.	

CT08	Segunda opinião	Vincular paciente através de um busca		1.O usuário na tela “Associar um Paciente” preenche os campos para realizar a busca [RN-04-008] e clica em “Buscar”; 2.O usuário seleciona um paciente apresentado na lista e clicar em “Adicionar”.	1.O sistema verifica os campos inseridos de acordo com os campos apresentado; 2. O sistema apresenta a listagem de itens que satisfazem os critérios informados. 3.O sistema apresenta um pop up de aviso com mensagem
CT09	Segunda opinião	Vincular e cadastrar novo paciente		1.O usuário na tela “Associar um Paciente” clicar no botão “Novo Paciente”. 2. O usuário preenche todos os campos obrigatórios , e clicar em “Salvar”. 3.O usuário clica em “sim”.	1.O sistema apresenta a tela “Manter Dados do Paciente” com uns campos obrigatórios para cadastrar um novo paciente; 2. O sistema apresenta um pop up de aviso com mensagem 3.O sistema cadastra o novo paciente e retorna a tela “Nova Solicitação” com os dados do paciente cadastrado nos campo “Dados do Paciente”, precisando preencher os campos de “Dados da Solicitação” para terminar a solicitação.

Fonte: Ferreira(2019)

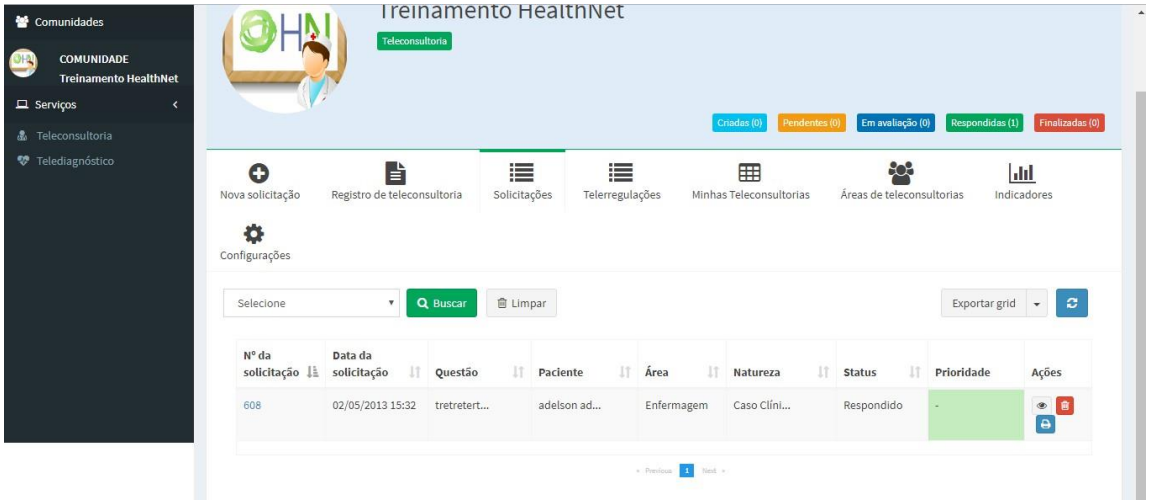
7.2 EVIDÊNCIAS DOS CASOS DE TESTES

Figura A-1 – confirma Teleconsultor logrado



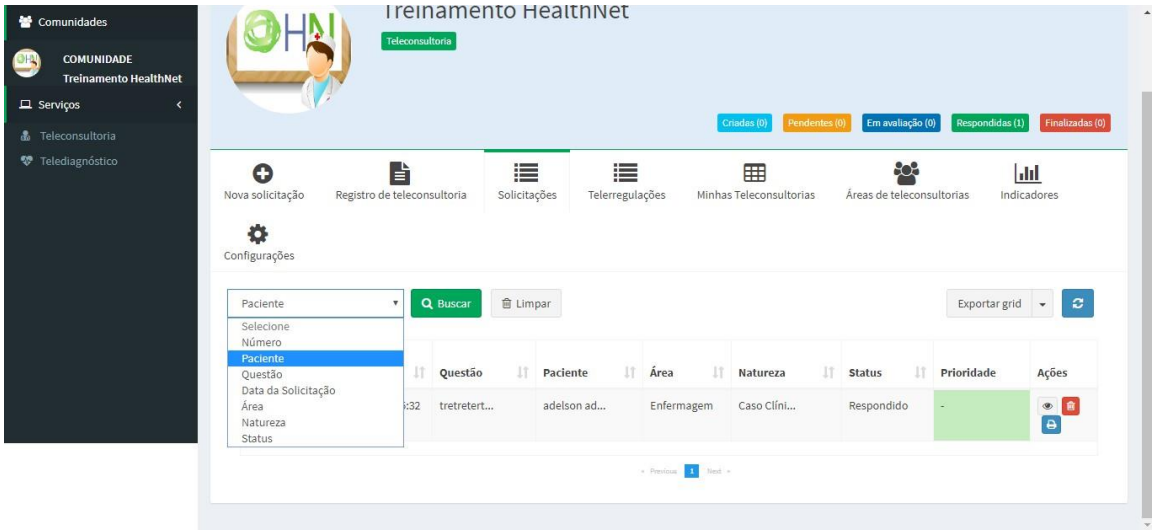
Fonte: Ferreira(2019)

Figura A-2 - Visualização do grid de solicitação



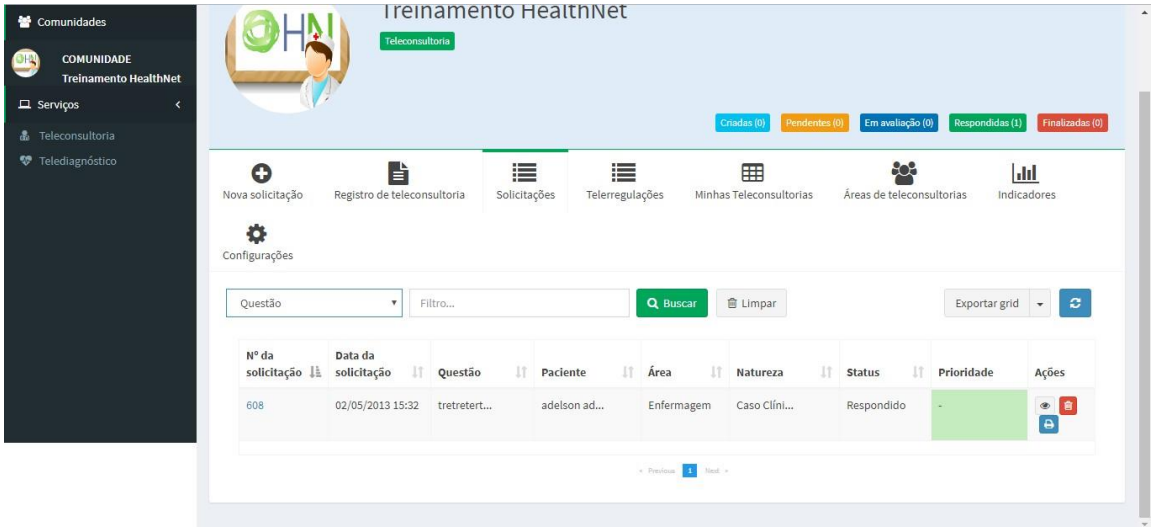
Fonte: Ferreira(2019

Figura A-3 - Filtrar solicitações existentes



Fonte: Ferreira(2019

Figura A-4 - Verificar existência de solicitações

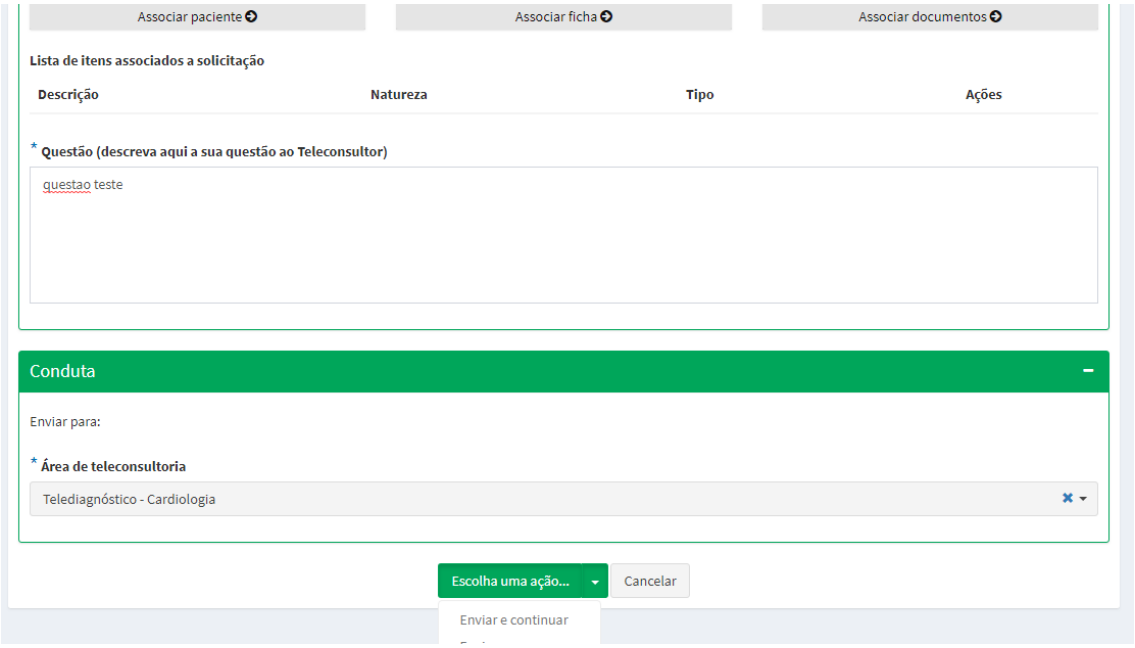


Fonte: Ferreira(2019

Figura A-5 - Realizar Solicitação

Fonte: Ferreira(2019

Figura A-6 - Não vincular paciente a solicitação



Fonte: Ferreira(2019)