UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO CIÊNCIA DA COMPUTAÇÃO

WESLEY ADAMO COSTA DOS SANTOS

SISMED: UM SISTEMA DE CONTROLE DE CONSULTÓRIO MÉDICO

MOSSORÓ
2019

WESLEY ADAMO COSTA DOS SANTOS

SISMED: UM SISTEMA DE CONTROLE DE CONSULTÓRIO MÉDICO

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Paulo Gabriel Gadelha Queiroz, Prof. Dr.

MOSSORÓ

2019

S237s Santos, Wesley Adamo Costa dos.

SisMED: Um sistema de controle de consultório médico / Wesley Adamo Costa dos Santos. - 2019.
52 f. : il.

Orientador: Paulo Gabriel Gadelha Queiroz.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência da Computação, 2019.

1. Internet. 2. Sistema Web. 3. Medicina. 4. Consultório médico. I. Queiroz, Paulo Gabriel Gadelha ,orient. II. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecasda Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

WESLEY ADAMO COSTA DOS SANTOS

SISMED: UM SISTEMA DE CONTROLE DE CONSULTÓRIO MÉDICO

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência da Computação.

Defendida em: 21 / 03 / 2019.

BANCA EXAMINADORA

Paulo Gabriel Gadelha Queiroz, Prof. Dr. (UFERSA)
Presidente

Angélica Félix de Castro, Prof. Dr. (UFERSA)
Membro Examinador

Daniel Faustino Lacerda de Souza, Prof. Dr. (UFERSA)
Membro Examinador

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois durante todos esses anos de graduação tem me guiado e suportado em todos os obstáculos enfrentados e, sem Ele, não seria possível a realização deste sonho, o título de graduado em Ciência da Computação.

Agradeço também a minha família, que tem sido o meu alicerce e me apoiado desde o início do curso. Inúmeras vezes surgiram decepções, desânimo, porém, eles sempre estiveram juntos dando apoio e encorajamento para seguir meus objetivos.

Agradeço ao Orientador Dr. Paulo Gabriel por aceitar o desafio de orientar este trabalho e por todos os conselhos e dedicação na orientação.

Agradeço a Banca Examinadora por ajudar na realização deste trabalho e na avaliação do mesmo.

Agradeço a Jemima Dias por todo o apoio, paciência e incentivo durante toda a graduação.

Agradeço ao meu amigo Jonas Firmino, aos meus amigos e companheiros de curso, que foram essenciais em toda minha caminhada na UFERSA e aos demais amigos que de alguma forma ajudaram nesses anos de graduação.

Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes.

Martin Luther King, Jr.

RESUMO

Com os avanços da Internet, sobretudo, nas últimas décadas, os serviços disponibilizados têm se tornado cada vez mais presente na vida das pessoas. Para isto, são criadas aplicações a fim suportar as mais diversas atividades, sejam profissionais, acadêmicas ou pessoais. Essa importância se justifica pelas vantagens proporcionadas, a saber: praticidade, meio de comunicação e informação, *E-Commerce*. Dentre os serviços, destaca-se a *Web*. O presente trabalho tem como objetivo demonstrar a criação de um sistema Web para controle de consultório médico. A partir da necessidade de um sistema web para o consultório médico da UFERSA, verificou-se a necessidade da produção deste trabalho. Para tanto, foi utilizada a prototipação junto à professora de medicina da UFERSA com o objetivo de levantar os requisitos necessários para o desenvolvimento da aplicação. Além disso, foi escolhida a plataforma de desenvolvimento JAVA juntamente com a metodologia de desenvolvimento ágil *eXtremeProgramming* (XP). Com este trabalho, disponibiliza-se aos profissionais do consultório médico uma ferramenta de apoio para o atendimento ao paciente e o gerenciamento do consultório e, proporciona aos alunos de medicina, orientados pelos professores, a prática do conhecimento adquirido durante o curso por meio do atendimento ao paciente apoiado pelo sistema desenvolvido.

Palavras-chave: Internet. Sistema web. Medicina. Consultório médico.

ABSTRACT

Advances of the Internet, especially in the last decades, made the services available more and more present in people's lives. For this, applications are created in order to support the most diverse activities, between labor, academic or personal. This importance is justified by the advantages provided, namely: practicality, means of communication and information, E-Commerce. Among the services, the Web stands out. The objective of this work is to demonstrate the creation of a Web system for the control of a doctor's office. Based on the need for a web system for *Universidade Federal Rural do Semi-Árido - UFERSA*'s medical practice, the production of this work was verified. For that, prototyping was used with the *UFERSA* medical professor in order to raise the necessary requirements for the development of the application. In addition, the JAVA development platform was chosen along with the agile development methodology eXtreme Programming (XP). With this work, the medical practice professionals provide a support tool for patient care and office management, and provides medical students, guided by teachers, the practice of the knowledge acquired during the course through care to the patient supported by the developed system.

Keywords: Internet. Web system. Medicine. Doctor'soffice.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

WWW	*World Wide Web*
XP	*EXtremeProgramming*
CERN	*ConseilEuropéenpourlaRechercheNucléaire*
HTML	*HyperTextMarkupLanguage*
XHTML	*EXtensibleHyperTextMarkupLanguage*
HTTP	*Hypertext TransferProtocol*
CSS	*CascadingStyleSheets*
DOM	*DocumentObjectModel*
JSF	*JavaServer Faces*
IDE	*IntegratedDevelopmentEnvironment*
MVC	*Model-View-Controller*
DAO	*Data Access Object*

SUMÁRIO

1 INTRODUÇÃO

O advento da Internet provocou mudanças significativas no cotidiano da sociedade. A comunicação, o acesso à informação, o ensino e o aprendizado, as relações de trabalho, a vida social, etc. jamais foram os mesmos desde tal acontecimento.

A Internet foi oriunda de um projeto militar planejado por volta da década de 70 (CLEMENTE, 2014). Dentre os vários serviços disponibilizados na Internet, destaca-se a *Web*. Esse sistema, criado por Tim Berners-Lee, surgiu com a finalidade de interligar as universidades entre si, permitindo que os trabalhos e pesquisas acadêmicas fossem utilizados entre as universidades. Berners-Lee também é responsável pelo desenvolvimento da linguagem HTML e do protocolo HTTP. (BARWINSKI, 2009).

Com o desenvolvimento da *Web* e a popularização dos computadores, principalmente nas últimas décadas, houve um grande esforço no desenvolvimento de soluções computacionais para apoiar o homem nas mais diversas atividades, sejam pessoais, profissionais ou acadêmicas. Essas soluções computacionais são justificadas devido às vantagens proporcionadas, por exemplo: a otimização de tarefas.

Diante da implantação do curso de medicina e do consultório médico na Universidade Federal Rural do Semi-Árido - UFERSA, surgiu a necessidade de um sistema de controle para o consultório médico visando apoiar o atendimento aos pacientes e as demais necessidades dos profissionais envolvidos, aspectos que serviram de motivação para elaboração deste projeto de conclusão de curso. A relevância deste trabalho se efetiva na contribuição para os profissionais que utilizam o consultório médico, para os estudantes do curso de medicina que utilizarão o sistema com a supervisão dos professores, como também para os estudos relacionados ao desenvolvimento de sistemas *Web*.

1.1 Problema de pesquisa

A partir das considerações mencionadas acerca da Internet e da *Web*, o presente trabalho visa responder ao seguinte problema de pesquisa: Como desenvolver um sistema *web* para apoiar um consultório de medicina?

1.2 Objetivos

1.2.1 Objetivo geral

O presente trabalho tem como finalidade desenvolver um sistema *Web* para controle do consultório médico da UFERSA.

1.2.2 Objetivos específicos

- ✓ Aplicar a metodologia de desenvolvimento ágil XP.
- ✓ Apresentar as etapas envolvidas no desenvolvimento *web*.

1.3 Metodologia

Inicialmente, foram realizadas reuniões para definir o problema, o escopo e os requisitos funcionais do sistema, gerando como artefato uma lista de requisitos. A cada reunião, além de surgirem novos requisitos, os requisitos definidos em reuniões anteriores passavam por um refinamento. Como os requisitos mudavam com certa frequência e não estavam bem definidos, o XP, motivado por seus valores, como o *feedback* constante do cliente e a codificação como atividade principal, foi definido como abordagem de desenvolvimento por se adequar ao contexto de desenvolvimento do sistema.

Para auxiliar o processo de desenvolvimento e conduzir as reuniões, foram construídos protótipos funcionais e apresentados aos *stakeholders*. De acordo com Vazquez e Simões (2016), a prototipação é uma técnica interativa que permite identificar possíveis problemas relacionados aos requisitos antes que a versão final do produto esteja finalizada.

No SisMed, a prototipação foi realizada desde o início do processo de desenvolvimento, com o objetivo de conhecer bem o problema e validar os requisitos.

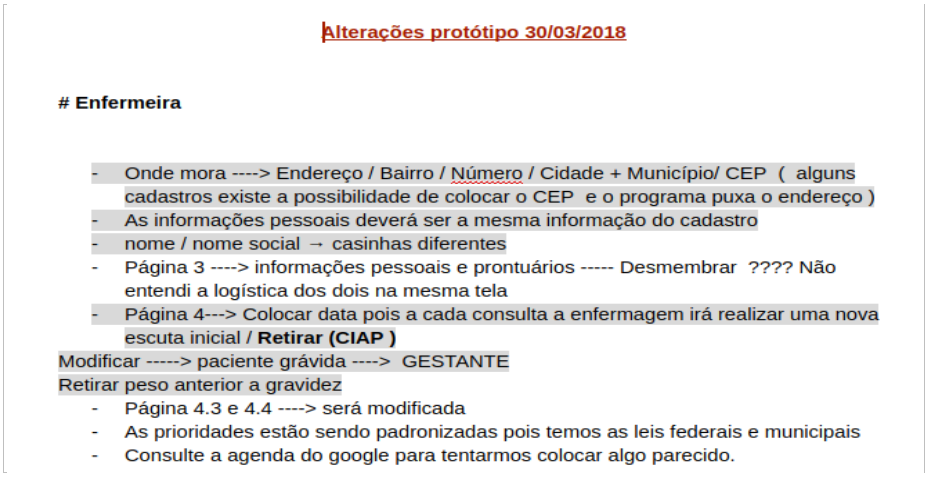
Após a definição de alguns requisitos junto à professora de medicina Ana Flávia, a enfermeira Graciella Jales ficou responsável por fornecer todos os requisitos restantes e validar o sistema. A partir da definição dos requisitos foi escolhida a estratégia de prototipagem de baixa fidelidade, que, segundo Vazquez e Simões (2016), caracteriza-se por protótipos, também alcunhados de *mockups* ou *wireframes*, desenhados sem priorizar a estética do sistema e com baixo nível de detalhamento, demonstrado na figura 1.

Figura 1-*Wireframe* inicial da tela do prontuário médico.

Fonte: Autor.

A etapa de construção do protótipo foi apoiada pela ferramenta *Balsamiq*, devido sua praticidade e facilidade na criação dos *wireframes*. Após a criação dos protótipos, a validação era realizada por meio de uma reunião, na qual os protótipos eram apresentados e analisados pela professora. O *feedback* resultante da fase de validação era realizado por meio de alterações sugeridas de forma presencial bem como de uma lista de alterações enviada via *e-mail*, como pode ser observado na figura 2.

Figura 2-Lista de alterações e adição de requisitos solicitados pelo cliente.



Alterações protótipo 30/03/2018

 # Enfermeira

- Onde mora ----> Endereço / Bairro / Número / Cidade + Município/ CEP (alguns cadastros existe a possibilidade de colocar o CEP e o programa puxa o endereço)
- As informações pessoais deverá ser a mesma informação do cadastro
- nome / nome social → casinhas diferentes
- Página 3 ----> informações pessoais e prontuários ----- Desmembrar ???? Não entendi a logística dos dois na mesma tela
- Página 4---> Colocar data pois a cada consulta a enfermagem irá realizar uma nova escuta inicial / **Retirar (CIAP)**

Modificar -----> paciente grávida ----> GESTANTE

Retirar peso anterior a gravidez

- Página 4.3 e 4.4 ----> será modificada
- As prioridades estão sendo padronizadas pois temos as leis federais e municipais
- Consulte a agenda do google para tentarmos colocar algo parecido.

Fonte: Autor.

Diante da necessidade de alterações e acréscimos de requisitos, um novo processo de prototipação era planejado, com o objetivo de validar as alterações nos requisitos solicitados e dos requisitos acrescentados. Dessa forma, os requisitos eram avaliados até que estivessem aptos suficientes para serem implementados.

Por fim, foi realizada uma validação do sistema por meio de um formulário respondido pelo assistente e pela enfermeira do ambulatório, com a finalidade de atestar que o sistema cumpre com as funções para as quais foi designado.

1.4 Organização do trabalho

O restante deste trabalho está organizado da seguinte forma. No capítulo 2, apresenta-se o referencial teórico. No capítulo 3, apresenta-se o desenvolvimento do sistema, descrevendo as tecnologias e ferramentas utilizadas, o digrama de caso de uso, os diagramas de classe, a modelagem do sistema, os padrões de projetos utilizados e o sistema desenvolvido, com imagens das telas do sistema. No capítulo 4, apresenta-se a validação do sistema. Por fim, no capítulo 5, apresentam-se as considerações finais, em seguida, as referências bibliográficas.

2 REFERENCIAL TEÓRICO

2.1 *Web*

Cada vez mais presentes no cotidiano das pessoas, a *Web* e suas aplicações se tornaram indispensáveis nos dias atuais. Seu surgimento é atribuído ao pesquisador britânico Tim Berners-Lee, na data de 12 de março de 1989 em um relatório para o CERN, descrevendo o protocolo de transferência de hipertextos, transformando em *World Wide Web* um ano depois. (CLEMENTE, 2014).

A evolução da *web* é definida em três fases: *web* 1.0, *web* 2.0 e *web* 3.0. De acordo com Aquarela (2015), a *web* 1.0 foi caracterizada pela apresentação de conteúdo de forma estática e pela baixa interação. Já para Primo (2007), a *web* 2.0 foi caracterizada pelo compartilhamento de informações, além de proporcionar o aumento das interações entre os participantes. Por fim, segundo Rohr (2018), a *web* 3.0 é caracterizada pelo compartilhamento de dados entre computadores, objetivando atribuir sentido a grandes quantidades de dados.

2.2 Padrões de projetos

2.2.1 Introdução

O conceito de padrões de projetos é originado do campo da arquitetura, quando, em 1977, Christopher Alexander, divulgou um catálogo com 250 padrões para a arquitetura civil. (LEITE, 2005).

De acordo com Gamma et al. (2000), os melhores projetistas não resolvem os problemas do zero. Em vez disso, reutilizam soluções que funcionaram em projetos passados. Por sua vez, quando uma solução é boa, eles passam a utilizar em projetos futuros. Sendo assim, um padrão de projeto apresenta-se como solução cuja finalidade é resolver problemas específicos em projetos, tornando-os mais flexíveis e, em alguns casos, reutilizáveis.

Em geral, um padrão de projeto está estruturado em quatro elementos: um nome, que representa uma descrição do problema de projeto; o problema, que descreve o problema, o contexto e em que situação aplicar o padrão; uma solução, que descreve os participantes do padrão de projeto, além de suas responsabilidades e contribuições. Por fim, as consequências

de se aplicar o padrão de projeto, com resultados, vantagens e desvantagens oriundas de seu uso. (GAMMA et al., 2000).

2.2.2 Padrões *GoF*

Em 1995, Erich Gamma, Richard Helm, Palph Johnson e John Vlissides lançaram o livro "*Design Patterns: Elements of Reusable Object-Oriented Software*" no qual discutiram 23 padrões de projetos. Diante da relevância do livro, os quatros foram chamados de *Gang of Four* (*GoF*) e os padrões de *GoF Patterns* (Padrões *GoF*).(LEITE, 2005).

Observa-se na figura 3 que os padrões *GoF* são categorizados de acordo com o problema ao qual são aplicados, a saber: criacional, estrutural e comportamental.

Os padrões de criação objetivam abstrair a instanciação de objetos. Sendo assim, ao ser solicitado um objeto de determinado tipo, uma instância de tal objeto será fornecida, sem que o solicitante se preocupe e tenha detalhes do processo de criação do objeto solicitado. (GAMMA et al., 2000).

De acordo com Gamma et al. (2000), os padrões de projetos estruturais são aqueles em que há a preocupação na forma como as classes e os objetos do sistema são compostos.

Já os padrões de projetos comportamentais são focados nas interações e distribuição de responsabilidade entre classes ou objetos, descrevendo os padrões de comunicação entre eles. (GAMMA et al., 2000).

Figura 3- Padrões *GoF* e suas classificações. [1]

		Propósito		
		De criação	**Estrutural**	**Comportamental**
Escopo	**Classe**	Factory Method	Adapter (class)	Interpreter Template Method
	Objeto	Abstract Method Builder Prototype Singleton	Adapter (object) Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsability Command Iterator Mediator Memento Observer State Strategy Visitor

Fonte: Fernando Henrique Ferreira, 2012.

[1] Disponível em: <https://ferhenriquef.com/2012/08/27/design-patterns/>. Acesso em: 25 de jul. 2018.

2.2.3 Padrões J2EE

2.2.3.1 Definição

J2EE[2] é uma arquitetura que fornece escalabilidade, acessibilidade e gerenciamento para o desenvolvimento de aplicações corporativas de multicamadas. As soluções multicamadas implementadas com o Java EE geralmente são divididos em três camadas: máquinas dos clientes, servidor Java EE e o banco de dados ou máquinas legadas no *back-end*. (ORACLE, 2017).

Para Lupianhez e Lucrédio (2012) as camadas são definidas como:

✓ Camada de apresentação: camada onde ficam as páginas JSP/JSF e que acessarão os objetos da camada de negócio.

✓ Camada de negócio: camada que implementa as regras de negócios do sistema.

✓ Camada de integração: camada responsável pela persistência de objetos em banco de dados, por exemplo, como também o acesso aos dados disponíveis em sistemas legados, entre outras fontes externas.

2.2.3.2 Catálogo de padrões Java EE

Na tabela 1 são apresentados os padrões Java EE definidos de acordo com cada camada da Java EE.

Tabela 1-Padrões de projeto Java EE.[3]

Camada de apresentação	Camada de negócio	Camada de integração
InterceptingFilter	Business Delegate	Data Access Object
ContextObject	Service Locator	Service Activator
Front Controller	SessionFacade	Domain Store
ApplicationController	Application Service	Web Service Broker
ViewHelper	Business Object	
CompositeView	CompositeEntity	
DispatcherView	TransferObject	
Service ToWorker	T O Assembler	
	ValueListHandler	

Fonte: Core J2EE Pattern, 2018.

[2] Por uma questão de marketing, o nome correto hoje é Java EE.
[3]Disponível em: <http://www.corej2eepatterns.com/index.htm>. Acesso em: 25 de jul. 2018.

2.2.4 Como usar os padrões de projetos

De acordo com Gamma et. al (2000) para utilizar um padrão de projeto é necessário:

✓ Ler o padrão inteiro para obter uma visão geral do padrão e assegurar que o padrão é correto para o problema que se deseja aplicá-lo.

✓ Compreender as classes e objetos do padrão e como se relacionam entre si.

✓ Olhar exemplos de código do padrão.

✓ Utilizar nomes dos participantes do padrão que tenha sentido no contexto da aplicação.

✓ Definir as classes, bem como suas *interfaces* e os seus relacionamentos de herança. Definir as variáveis que representam dados e referências a objetos. É necessário também, identificar e modificar as classes que serão afetadas decorrentes da aplicação do padrão de projeto.

✓ Definir nomes específicos que dependem da aplicação para as operações do padrão.

✓ Implementar as operações assegurando as responsabilidades e as colaborações presentes nos padrões.

2.3 Métodos ágeis

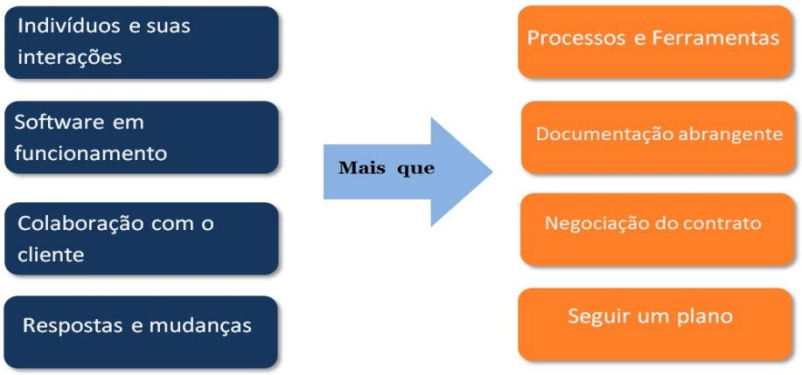
2.3.1 Definição

De acordo com Sommerville (2011) a insatisfação com as abordagens pesadas (tradicionais) da engenharia de *software*, levou, na década de 90, um grande número de desenvolvedores a proporem novos métodos de desenvolvimento de *software*, surgindo os métodos ágeis. Nessa nova abordagem, foca-se no desenvolvimento do *software* ante sua concepção e documentação. A especificação, o desenvolvimento e a entrega do *software* baseiam-se na perspectiva incremental. Os métodos ágeis são adequados quando os requisitos mudam rapidamente durante o desenvolvimento do sistema, visando uma entrega rápida do *software* ao cliente, e esse poderá propor alterações que serão adicionadas nas iterações seguintes.

2.3.2 Manifesto ágil

Em uma reunião ocorrida em 2001, no estado norte-americano de Utah, os principais nomes do desenvolvimento de *software* se reuniram para criar o manifesto ágil. Como pode se observar na figura 4 e na tabela 2, o manifesto ágil é composto por quatro valores e doze princípios, representando a filosofia dos métodos ágeis.

Figura 4- Valores do manifesto ágil. [4]



Fonte: ASR consultoria, 2018.

[4] Disponível em: <https://scrumasr.wordpress.com>. Acesso em: 20 de jun. 2018.

Tabela 2- Os doze princípios do manifesto ágil. [5]

Item	Definição
1	Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
2	Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
3	Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
4	Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
5	Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
6	Método mais eficiente e eficaz de transmitir informações para, e por dentro de um Time de desenvolvimento, é através de uma conversa cara a cara.
7	Software funcional é a medida primária de progresso.
8	Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
9	Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
10	Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
11	As melhores arquiteturas, requisitos e designs emergem de Times auto-organizáveis.
12	Em intervalos regulares, o Time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Fonte: Imasters, 2018.

2.3.3 *eXtremeProgramming*

Em relação ao XP, Bonato (2002, p. 3) declara:

> Extreme Programming, ou XP, é uma metodologia leve para equipes de tamanho pequeno e médio desenvolverem software face a requisitos vagos ou que mudem muito rapidamente. Criada em 1996 por Kent Beck, Ron Jeffries e Ward Cuningham a partir de um projeto piloto realizado na Daymler-Crysler, a XP vem ganhando cada vez mais espaço no mercado. (BONATO, 2002, p. 3).

Segundo Medeiros (2013), as práticas do XP são fundamentadas nos seguintes valores: comunicação, *feedback*, simplicidade e coragem. É a figura do *coach* responsável por garantir que os valores estão sendo aplicados na prática.

✓ Comunicação: a comunicação é fundamental para que as práticas do XP possam ocorrer. Sendo assim, é necessário que a equipe esteja em comunicação com o cliente

[5] Disponível em:<https://imasters.com.br/devsecops/utilizacao-de-checklist-para-validacao-de-requisitos-de-software>. Acesso em: 20 de jun. 2018.

e entre a própria equipe de desenvolvimento. Dessa forma, a comunicação favorece a eliminação de documentos.

✓ *Feedback*: O *feedback* é fundamental para que o *software* evolua. É necessário que o *feedback* seja o mais cedo possível para que seja garantido junto ao cliente que a equipe está fazendo a coisa correta e dentro do prazo planejado.

✓ Simplicidade: a simplicidade tenta fazer o mais simples possível, caso seja necessário, algo mais complexo é tentado. Assim, evita-se de fazer algo completo e que depois não será utilizado ou necessário.

✓ Coragem: em XP, é necessário ter coragem para fazer mudanças e deixar o cliente ciente das mudanças.

Por fim, segundo Medeiros (2013), no XP, o foco está na codificação, os testes são utilizados como especificação de requisitos e é essencial a comunicação entre os desenvolvedores.

3 DESENVOLVIMENTO

Nesta seção, são apresentadas as tecnologias e ferramentas utilizadas no desenvolvimento do sistema, o diagrama de caso de uso, a modelagem de dados, os diagramas de classes e os padrões de projetos utilizados no desenvolvimento do sistema.

3.1 Tecnologias e ferramentas

3.1.1 Tecnologias

Além do Java, foram utilizadas as seguintes tecnologias no desenvolvimento do sistema: HTML, CSS, JavaScript.

O HTML, que em português se apresenta como linguagem de marcação de hipertexto, é uma das linguagens utilizadas no desenvolvimento de sites. Uma das características da linguagem é ser facilmente entendida pelas pessoas e pelas máquinas. Foi criada por Tim Berners-Lee com a finalidade de auxiliar ele e seu grupo de colegas na comunicação e compartilhamento de pesquisas (EIS, 2011). Atualmente, encontra-se na versão nomeada de HTML5.

Em relação ao CSS (*Cascading Style Sheets*), trata-se de uma linguagem direcionada para a visualização do conteúdo da pagina *web* desenvolvida com linguagem de marcação, separando o código de formatação de estilo do conteúdo da página. (PEREIRA, 2009).

Por fim, foi utilizada a linguagem *JavaScript*. A linguagem foi criada por Brendan Eich em 1995, então funcionário da *Netscape Communications Corporation*, para inicialmente ser utilizada no *Netscape Navigator*. Sua consolidação efetivou-se após a *Microsoft* incorporar a linguagem em seu navegador, um ano após seu lançamento. (SILVA, 2015).

Antes projetada para ser executada em navegadores *web*, hoje é possível executar códigos *JavaScript* também em servidores *web*. Do lado do cliente, a linguagem possibilita o controle do navegador e do DOM. Já do lado do servidor, permite que a aplicação acesse banco de dados, manipule arquivos no servidor, etc. (MOZILLA, 2017).

3.1.2 Ferramentas

As ferramentas que apoiaram o desenvolvimento foram: *JSF*, *PrimeFaces*, *Hibernate*, *MySQL*, *Eclipse IDE*, *Spring Security* e o *Apache TomCat*.

O *JSF* (*JavaServer Faces*) é definida como uma especificação para um *framework* utilizado no desenvolvimento de aplicações *web* em Java. As implementações mais conhecidas do *JSF* são a *Mojarra* e o *MyFaces* da *Apache*. Com essas implementações é possível utilizar os recursos padrões do *JSF* (formulários, tabelas, etc.) além de permitir a comunicação com classes Java, chamadas de *Managed Beans*.(LUCKOW; MELO, 2015).

No que se refere ao *PrimeFaces*, trata-se de uma biblioteca de componentes para *JSF*. Atualmente, contêm mais de 100 componentes gráficos para as aplicações *web* baseadas em *JSF*. Como possível vantagem do uso do *PrimeFaces*, há o aumento da produtividade, bem como da experiência por parte do usuário com o sistema.(SCHIECK, 2015).

Em relação à persistência de dados, foi utilizado o *Hibernate*. O *Hibernate* é um *framework* de mapeamento objeto/relacional (ORM) para ambiente Java. A técnica ORM persiste objetos de classes persistentes da aplicação em tabelas em um banco de dados relacional, utilizando tecnologias como o *Hibernate*. (LUCKOW; MELO, 2015).

De acordo com Silva (2009), além da persistência de objetos, o *Hibernate* provê mecanismos para a consulta e para a busca de dados que reduz consideravelmente o tempo de desenvolvimento.

O banco de dados utilizado na aplicação foi o *MySQL*. O *MySQL* é um dos principais bancos de dados de código-fonte aberto do mundo. Possui duas possibilidades de uso: gratuito e pago. Para se utilizar gratuitamente, a aplicação que faz uso não pode obter lucro. (LUCKOW; MELO, 2015).

A proteção do sistema foi desenvolvida utilizando o *framework* de código aberto *Spring*, mais precisamente um de seus projetos, o *Spring Security*.

Utilizando o *Spring Security* foi realizada a autenticação dos usuários e a segurança das pastas do sistema. Sendo assim, quando um usuário acessa o sistema, suas credenciais são verificadas pelo *framework* bem como sua permissão de usuário, que será utilizada para permitir ou não o acesso às pastas do sistema, com suas respectivas páginas.

O *Apache Tomcat* é um *container Web* baseado em Java para executar aplicações *Web* que utilizam as tecnologias *Servlets* e *Java Servlet Pages* (*JSP*). No desenvolvimento do

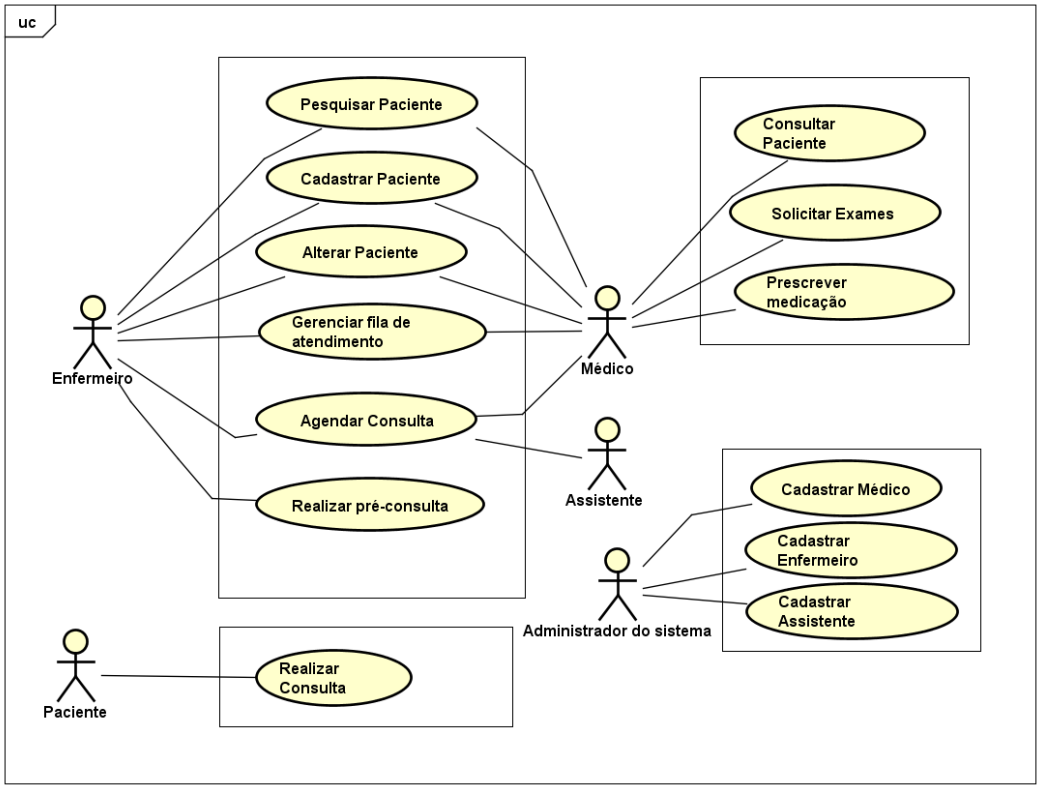
sistema, foi utilizada a versão 8, que implementa a especificação *Servlet* 3.1, permitindo o funcionamento do *JSF*. (LUCKOW; MELO, 2015).

Por fim, foi utilizada a ferramenta Eclipse IDE, do inglês *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, para desenvolvimento Java. Assim como o *MySQL*, também é uma ferramenta de código-fonte aberto. (LUCKOW; MELO, 2015).

3.2 Diagrama de caso de uso

Como artefato oriundo da etapa de especificação dos requisitos, foi elaborado o diagrama de caso de uso (figura 5), descrevendo as principais funcionalidades do sistema do ponto de vista dos usuários e suas interações com eles.

Figura 5- Diagrama de casos de uso do sistema.

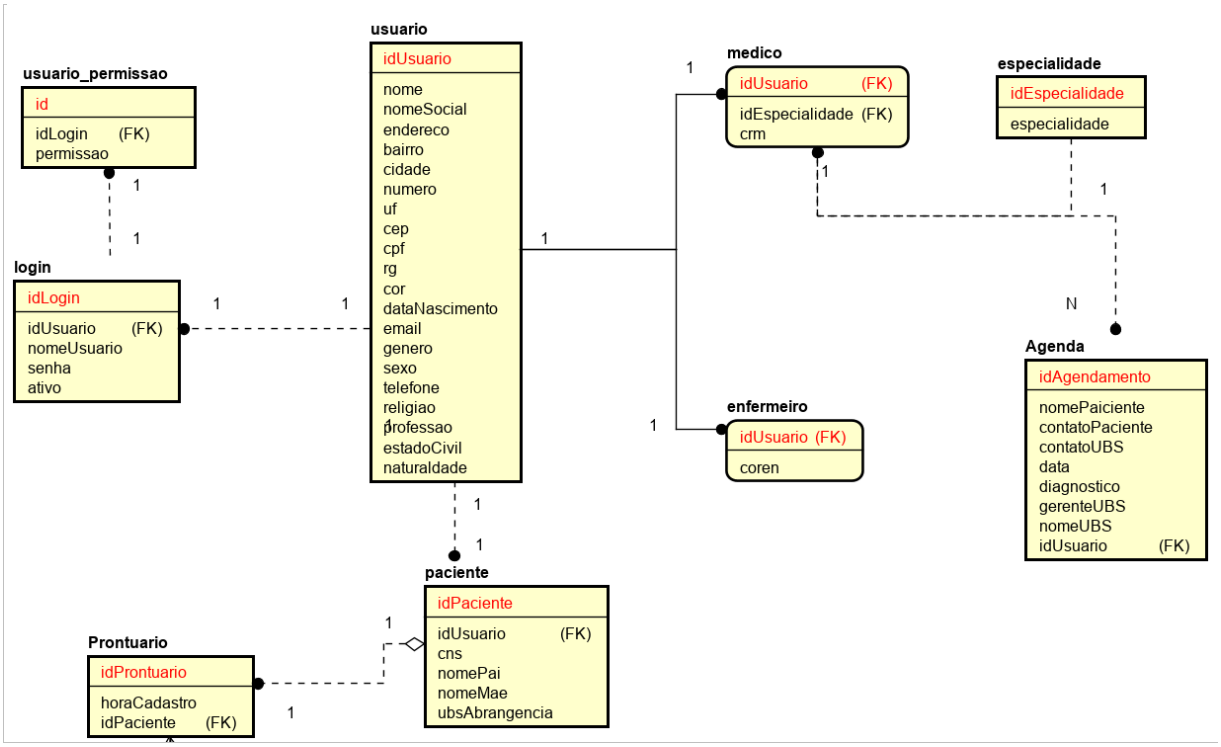


Fonte: Autor.

3.3 Modelagem de dados

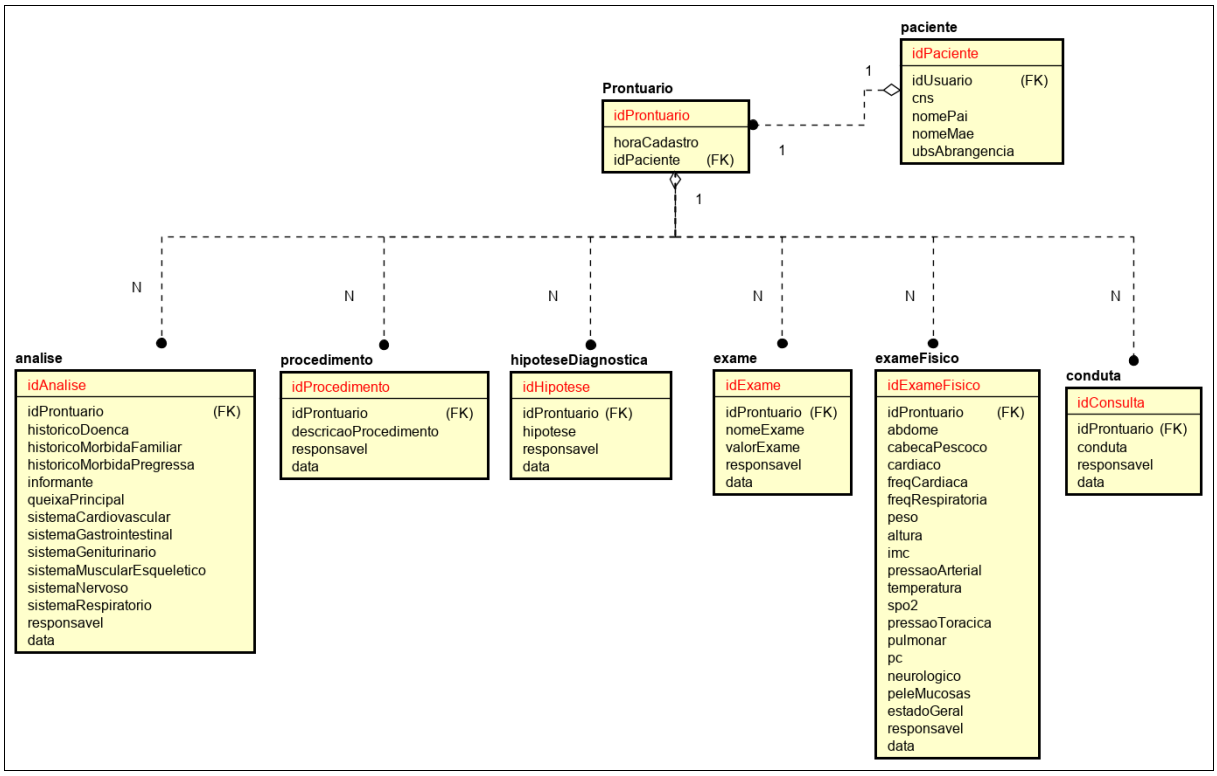
Nas figuras 6 e 7 é apresentada a modelagem de dados do sistema, detalhando suas tabelas e os seus relacionamentos.

Figura 6-Modelagem de dados dos usuários do sistema.



Fonte: Autor.

Figura 7- Modelagem de dados do prontuário médico.



Fonte: Autor.

3.4 Digramas de classes

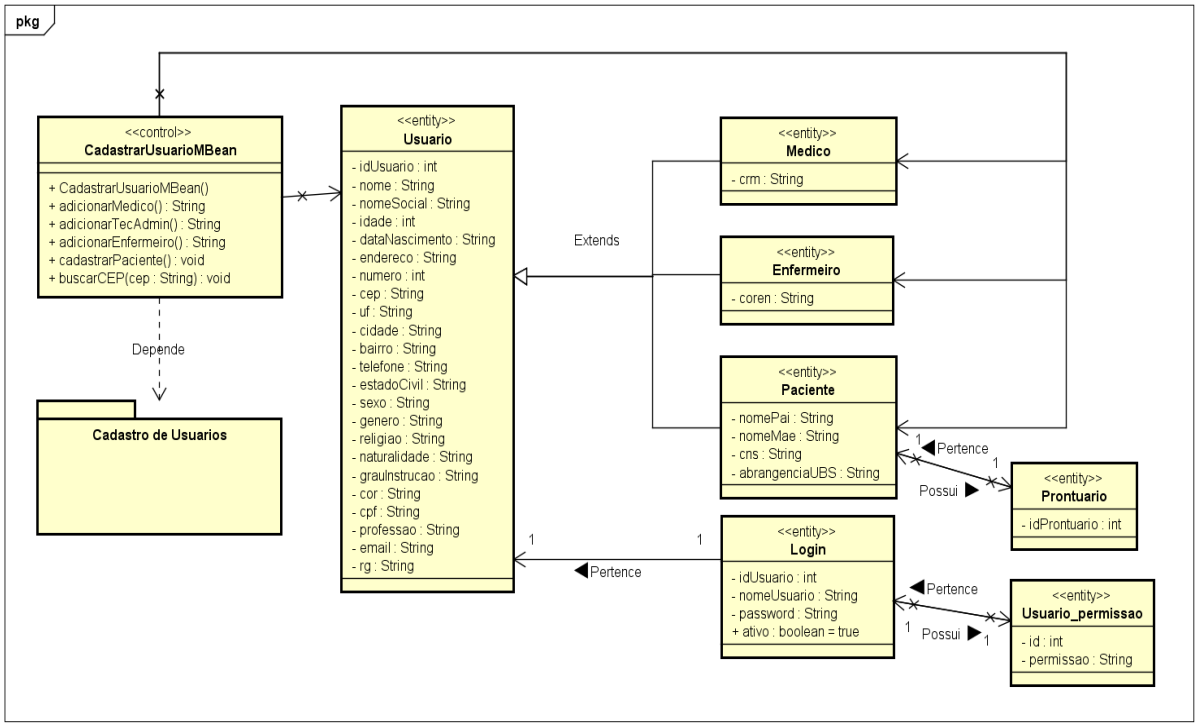
Os diagramas de classes demonstrados abaixo representam as principais classes e relacionamentos das principais funcionalidades do sistema.

3.4.1 Cadastro de Usuários

Na figura 8 é apresentado o digrama de classes para o cadastro de usuários. O cadastro de médicos, enfermeiros e assistentes do ambulatório é realizado pelo administrador do sistema. Ao cadastrar os usuários mencionados, é associada uma conta de *login* para o acesso ao sistema e uma permissão de usuário, que será utilizada pelo *Spring Security* para verificar quais as funcionalidades permitidas ao usuário. Em relação ao cadastro de paciente, esse é realizado por médicos e enfermeiros. Ao cadastrar um paciente, um único prontuário é atribuído a ele.

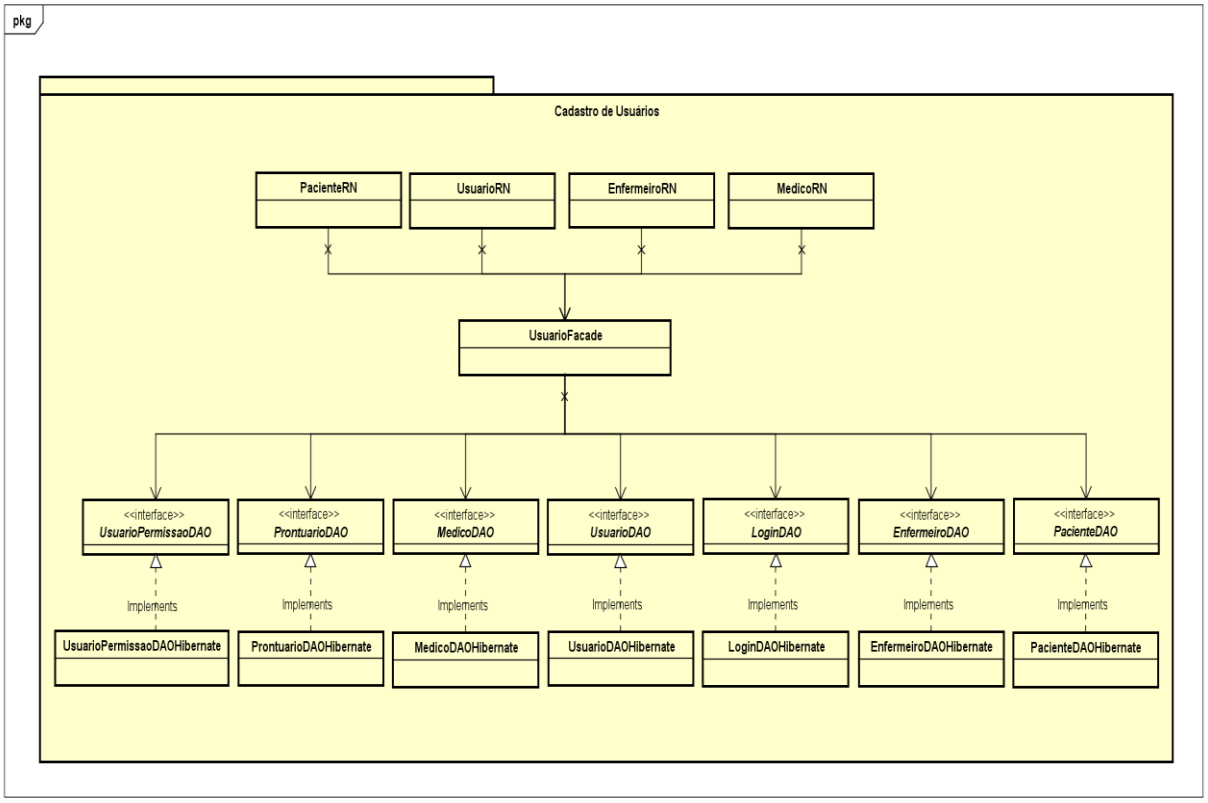
A fim de simplificar o diagrama e o seu entendimento, o diagrama de classes representado pelo *package* “Cadastro de usuários”, na figura 9, utilizado pelo controle *CadastroUsuarioMBean*, é apresentado separadamente. Na figura 9, a classe *UsuarioFacade* é responsável por simplificar a utilização das classes de persistências, encapsular as etapas para a inserção de um usuário, remoção, etc. a verificação de dados duplicados e fornecer também uma interface única para interação com essas classes.

Figura 8- Diagrama de classes para o cadastro de usuários



Fonte: Autor.

Figura 9-*Package* cadastro de usuários.

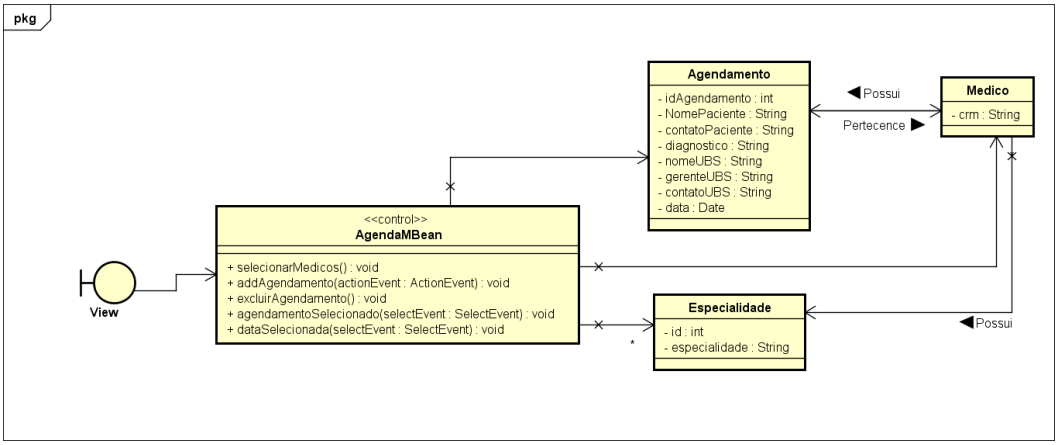


Fonte: Autor.

3.4.2 Agenda

O diagrama de classe para o agendamento de consultas é apresentado na figura 10. Como observado, para cada agendamento de consulta há um médico associado, informações do paciente e do gerente da UBS solicitante.

Figura 10-Diagrama de classes para o agendamento de consultas.

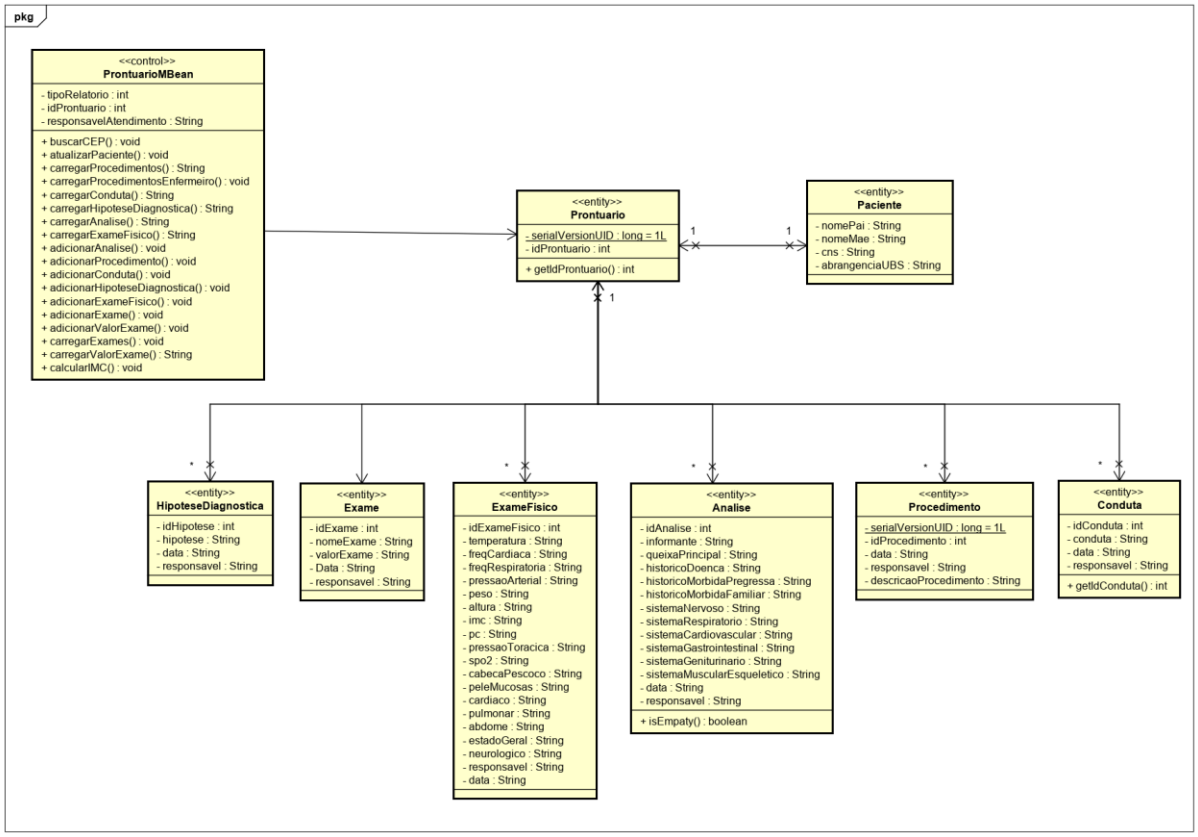


Fonte: Autor.

3.4.3 Prontuário

Na figura 11 é apresentado o digrama de classes do prontuário médico. O prontuário médico é composto por: hipótese diagnóstica, exame(s), exame físico, análise, procedimento(s), conduta e pertencente a um único paciente.

Figura 11- Diagrama de classes do prontuário médico.

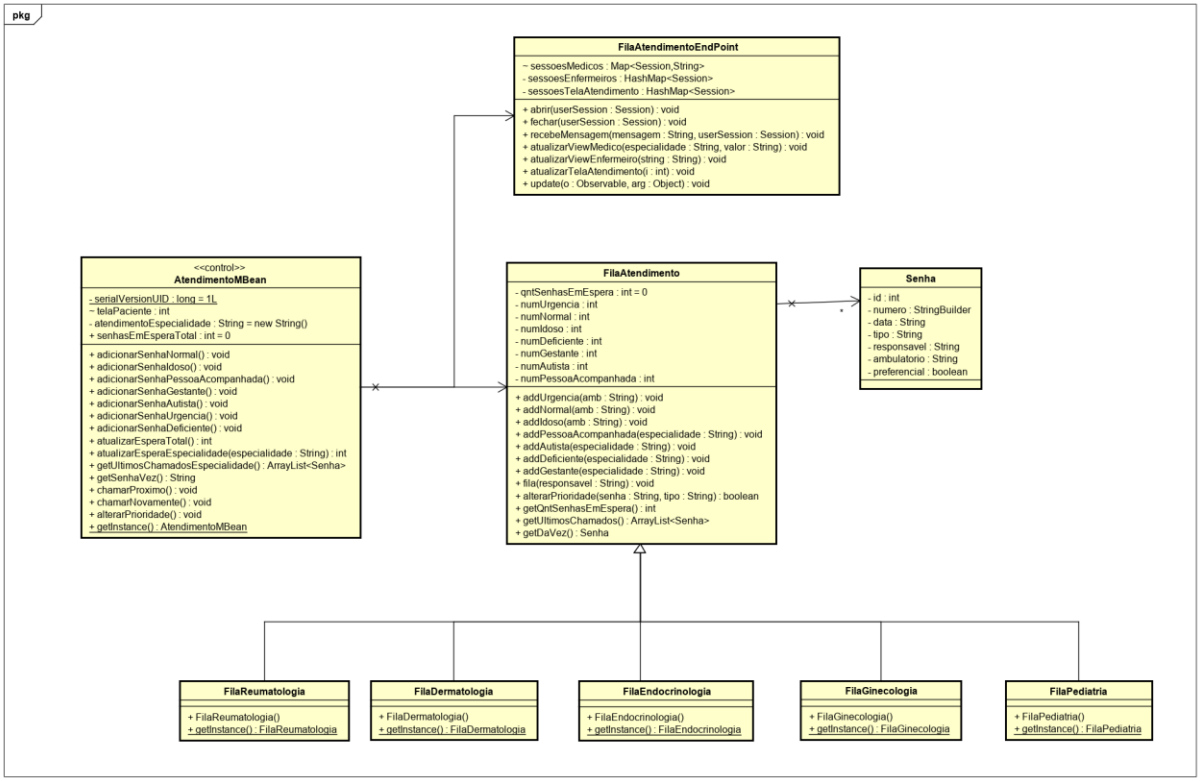


Fonte: Autor.

3.4.4 Fila de atendimento

No diagrama de classe da figura 12 estão as classes e relacionamentos que compõem o gerenciamento da fila de atendimento dos pacientes.

Figura 12-Diagrama de classes do gerenciamento da fila de atendimento.



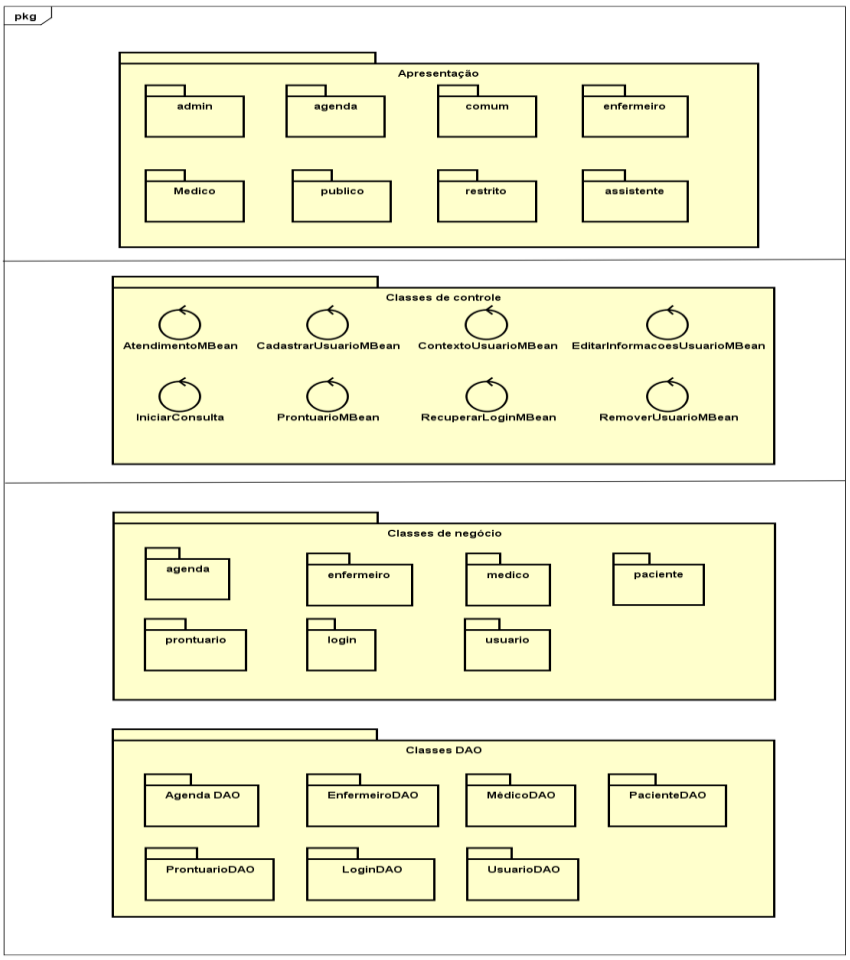
Fonte: Autor.

3.5 Padrões utilizados

3.5.1 MVC

O padrão MVC (*Model- View- Controller*) foi utilizado para separar o sistema em três camadas, como pode ser observado na figura 13.

Figura 13-Utilização do padrão MVC no sistema.



Fonte: Autor.

3.5.2 *Façade*

O padrão de projeto *Façade* foi utilizado para fornecer as classes de negócio responsáveis pelo gerenciamento dos usuários (inserção, remoção, atualização e busca) uma interface para persistir e acessar os dados. Sendo assim, todas as etapas necessárias para a persistência e recuperação dos dados estão encapsuladas na classe *UsuarioFacade*. Na figura 9 apresentada anteriormente, nota-se que a classe *UsuarioFacade* está localizada entre a camada de negócio e a camada de persistência.

3.5.3 *Singleton*

O *Singleton* foi utilizado no gerenciamento da fila de atendimento por meio da anotação *@Singleton*, garantindo assim que apenas uma instância da fila de atendimento é

criada. Na figura 14 é demonstrada a utilização da anotação *@Singleton* na classe de controle *AtendimentoMBean*, que tem como atributo a fila de atendimento utilizada pelos clientes. Dessa forma, é garantida que apenas uma classe de controle é instanciada e utilizada por todos os clientes.

Figura 14-Utilização do padrão *Singleton*.

```
@ManagedBean(name = "atendimentoMBean")
@ApplicationScoped
@Singleton

public class AtendimentoMBean extends Observable implements Serializable {
```

Fonte: Autor.

3.5.4 *DAO*

Com o intuito de separar a camada de regra de negócio da lógica de persistência de dados, foi utilizado o padrão *DAO*. As classes *DAO* são responsáveis por encapsular a lógica de persistência dos dados. Sendo assim, ao utilizar o padrão *DAO*, a alteração na forma de persistir os dados não afeta a lógica de negócio. Para as classes de negócio, basta conhecer a interface disponibilizada pelas classes *DAO*, sem ser necessário conhecer a tecnologia de persistência e os mecanismos para a troca de informação entre a aplicação e o SGBD.

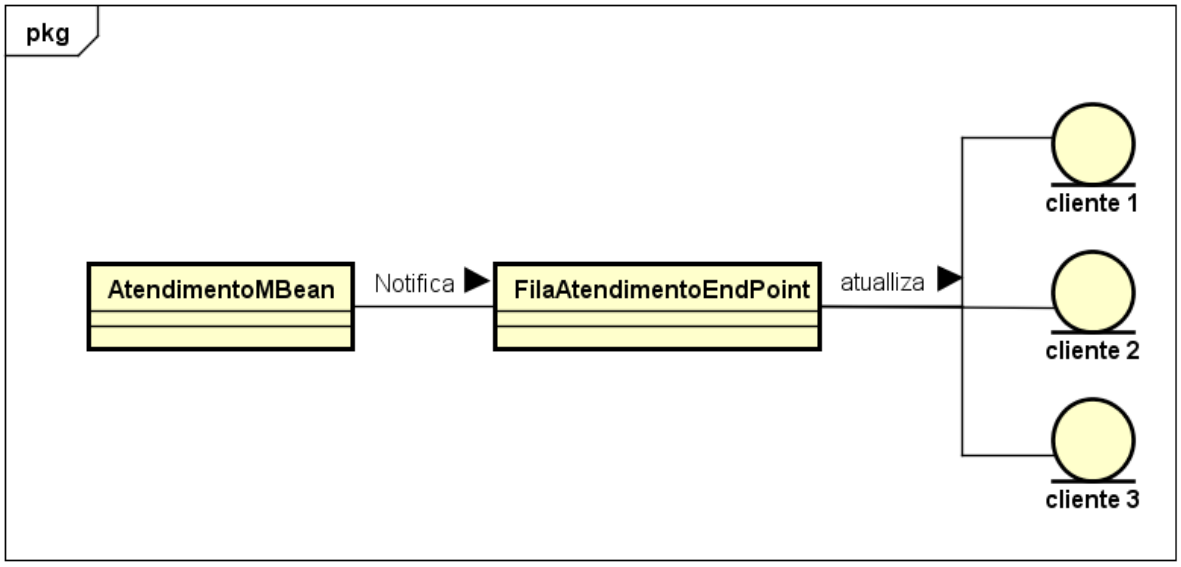
Na figura 9 apresentada anteriormente, observa-se como estão dispostas as classes de negócio e as classes de persistência dos dados.

3.5.5 *Observer*

O padrão *observer* foi utilizado no gerenciamento da fila de atendimento. A cada alteração na fila de atendimento, a classe *FilaAtendimentoEndPoint* é notificada. A Classe *FilaAtendimentoEndPoint* é um *Web Socket* que armazena e atualiza as sessões dos clientes que utilizam a fila de atendimento (médicos, enfermeiros e a tela visível aos pacientes).

Na figura 15, a classe *AtendimentoMBean* representa a classe observada, chamada de *Subject* no padrão *Observer*. Já a classe *FilaAtendimentoEndPoint* representa a classe observadora, chamada de *Observer*, que é notificada a cada alteração no estado da classe observada.

Figura 15- Utilização do padrão *Observer*.



Fonte: Autor.

3.5.6 *Interception Filter*

A cada requisição que chega ao servidor, uma sessão *Hibernate* é aberta e finalizada ao final do processamento da requisição, com o objetivo de evitar que uma nova conexão seja aberta a cada operação realizada no banco de dados, evitando muitas conexões abertas e um grande consumo de recursos. Sendo assim, todas as operações realizadas durante uma requisição ao servidor estão dentro de uma sessão *hibernate*. Para isso, é necessário interceptar as requisições que chegam ao servidor.

Na figura 16 é apresentada a classe responsável por filtrar as requisições que chegam ao servidor e implementar o padrão *Interception filter*. Na linha 24, a sessão *hibernate* é criada a partir da classe *HibernateUtil*. A classe *HibernateUtil* é uma fábrica de sessões. Na linha 36 é realizada a inicialização de uma transição no banco de dados. Após isso, a requisição é passada adiante para execução normal (linha 38). Por fim, após todo o processamento, a transação com o banco de dados é concluída (linha 40) e a sessão é fechada (linha 43).

Figura 16- Utilização do padrão *Interception filter*.

```
@WebFilter(urlPatterns = { "*.jsf" })
public class ConexaoHibernateFilter implements Filter {
    private SessionFactory sf;

    public void init(FilterConfig config) throws ServletException {
        this.sf = HibernateUtil.getSessionFactory();
    }

    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain chain)
            throws ServletException, ConstraintViolationException {

        Session currentSession = this.sf.getCurrentSession();

        Transaction transaction = null;

        try {

            transaction = currentSession.beginTransaction();

            chain.doFilter(servletRequest, servletResponse);

            transaction.commit();

            if (currentSession.isOpen()) {
                currentSession.close();
            }

        } catch (Throwable ex) {
            try {
                if (transaction.isActive()) {
                    transaction.rollback();
                }
            } catch (Throwable t) {
                t.printStackTrace();
            }
            throw new ServletException(ex);
```

Fonte: Autor.

3.5.7 *Factory*

A classe *DAOFactory* é responsável por ser o único ponto para a instanciação das classes *DAO*. É na *DAOFactory* onde as classes *DAO* obtêm a sessão *hibernate* aberta. Na figura 17 exemplifica-se a instanciação de duas classes *DAO* utilizadas no sistema. Observa-se que as classes que implementam as interfaces *PacienteDAO* e *MedicoDAO* são instanciadas e atribui-se a sessão *hibernate* ao atributo de sessão das respectivas classes.

Figura 17-Utilização do padrão *Factory*.

```
38 public class DAOFactory {
39
40     public static PacienteDAO criarPacienteDAO() {
41         PacienteDAOHibernate pacienteDAO = new PacienteDAOHibernate();
42         pacienteDAO.setSession(HibernateUtil.getSessionFactory().getCurrentSession());
43         return pacienteDAO;
44     }
45
46     public static MedicoDAO criarMedicoDAO() {
47         MedicoDAOHibernate medicoDAO = new MedicoDAOHibernate();
48         medicoDAO.setSession(HibernateUtil.getSessionFactory().getCurrentSession());
49         return medicoDAO;
50     }
```

Fonte: Autor.

3.5.8 *Composite View*

O padrão *Composite View* foi utilizado na criação das *views*. Um *template* foi elaborado para ser utilizado em todas as *views* do sistema, evitando a duplicação de código, como cabeçalhos, importação de arquivos de estilos, *scripts*, etc.

Para a utilização do padrão *Composite View* foi utilizado o *Facelets*. O *Facelets* foi criado com o intuito de fornecer uma linguagem de *templates* para telas do *JSF* por meios de *tags*, permitindo o reuso de padrões de telas.

Na figura18 é demonstrado o uso da *tag insert*. A *tag insert* é responsável por definir uma área que poderá ser substituída, por meio da *tag define*, nas telas que utilizam o *template*.

Figura 18-Utilização da *tag insert*.

```
<div id="page-wrapper">
    <div class="container-fluid">
        <div class="row">
            <div class="col-lg-12">
                <ui:insert name="conteudo" />


            </div>
        </div>
    </div>
</div>
```

Fonte: Autor.

Já na figura 19 é demonstrada a utilização da *tag composition*. A *tag composition* é utilizada sempre que deseja utilizar um *template* criado.

Figura 19-Utilização da *tag Composition*.

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:p="http://primefaces.org/ui"
    xmlns:pe="http://primefaces.org/ui/extensions">

<ui:composition template="template.xhtml">
```

Fonte: Autor.

Por fim, na figura 20 é demonstrada a utilização *das tags define e include*. A *tag define* é utilizada para inserir conteúdo que foi delimitado pela *tag insert* no *template*. Já a *tag include* insere um arquivo, por exemplo, outra página contendo outros conteúdos.

Figura 20- Utilização das *tags define e include*.

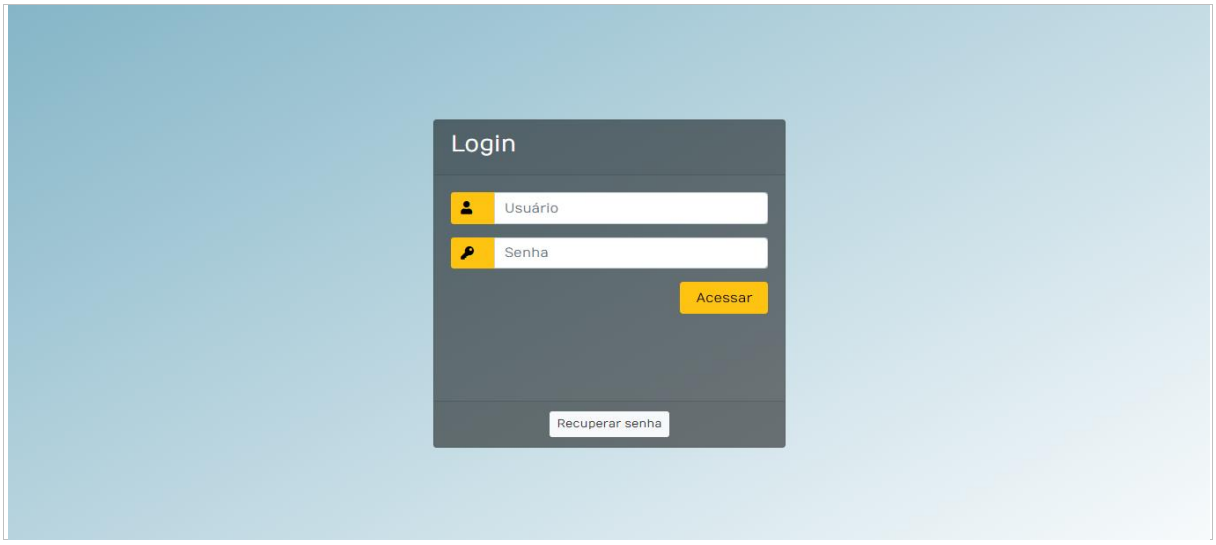
```
<ui:define name="conteudo">

<ui:include src="cabecalho-paciente.xhtml"></ui:include>


    <h:form id="formtabela">
        <button type="button"
            class="btn btn-primary btn-md adicionarProcedimento"
            data-toggle="modal" data-target="#modalAnalise" data-whatever="@mdo"
            data-backdrop="static" data-keyboard="false">
            <i class="fas fa-plus"></i> Adicionar
        </button>
```

Fonte: Autor.

3.6 Sistema desenvolvido

Nas figuras 21 e 22 são apresentadas as telas de login e de recuperação de senha de acesso ao sistema.

Figura 21- Tela de *login*.



Fonte: Autor.

Figura 22- Tela de recuperação de senha.

Fonte: Autor.

Na figura 23 é apresentada a tela exibida após a autenticação do médico no sistema. Como observado, é solicitada a escolha do ambulatório de atendimento do médico que será utilizado no gerenciamento da fila de atendimento. Na figura 24, é exibida a tela seguinte à seleção do ambulatório de atendimento.

Figura 23- Tela de seleção do ambulatório de atendimento.

Fonte: Autor.

Figura 24- Tela inicial do médico.

Fonte: Autor.

Já na figura 25, é apresentado o prontuário do paciente em consulta com todas as suas informações pessoais e as opções de adição e visualização dos dados do prontuário do paciente.

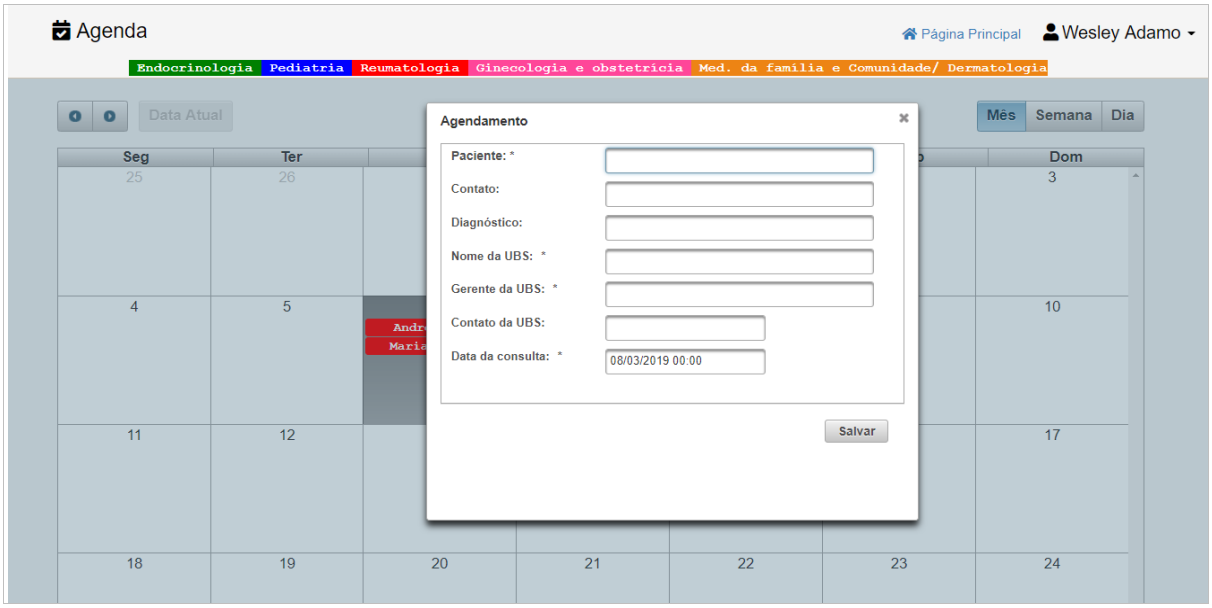
Figura 25- Tela do prontuário visível ao médico.

Fonte: Autor.

Nas figuras 26 e 27, são apresentadas as telas da agenda do médico. A tela comum aos médicos e enfermeiro para o cadastro de pacientes pode ser observada na figura 28. Já a figura

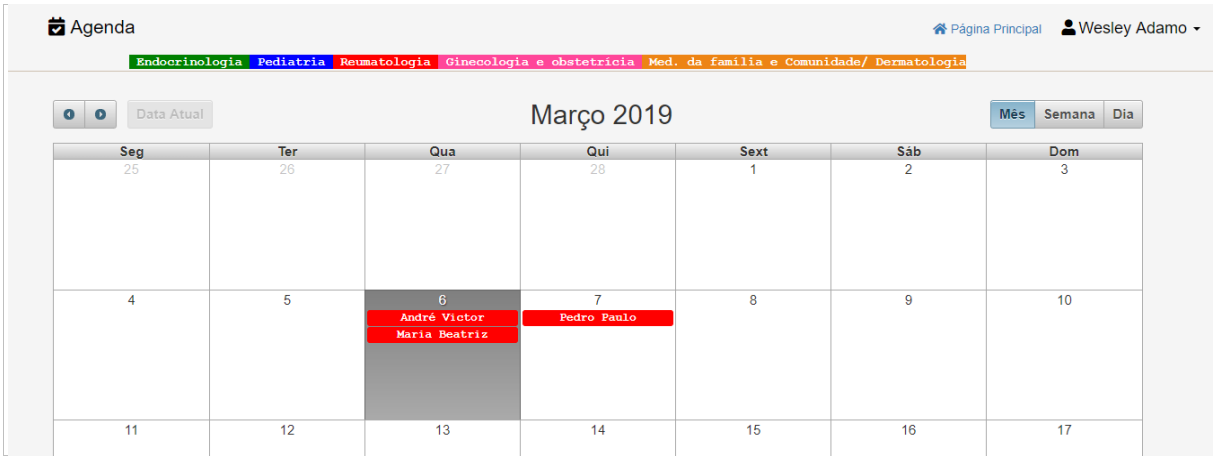
29, é referente ao gerenciamento da fila de atendimento por parte do médico. Por fim, encerrando as telas do médico, é apresentada a tela (figura 30) para alteração dos dados cadastrais do médico, semelhante para os demais usuários do sistema.

Figura 26- Tela de entrada de dados do agendamento de consulta visível ao médico.



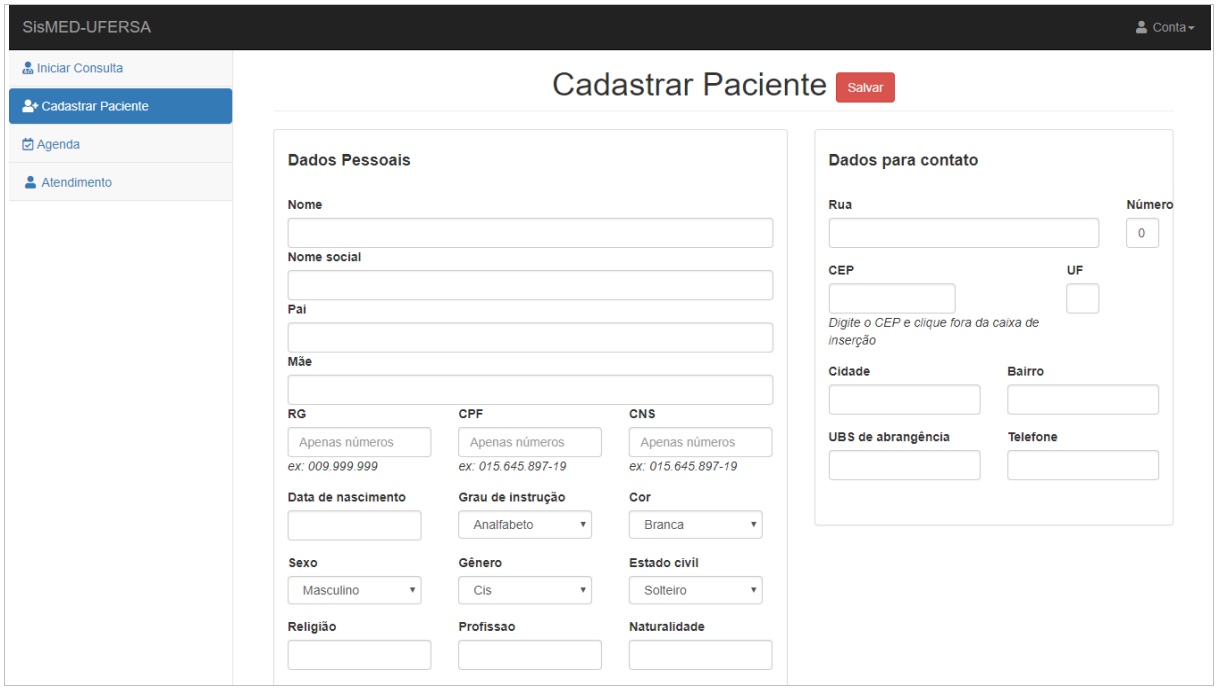
Fonte: Autor.

Figura 27- Tela da agenda do médico.



Fonte: Autor.

Figura 28- Tela comum aos enfermeiros e médicos de cadastro de paciente.



Fonte: Autor.

Figura 29- Tela do médico para o gerenciamento da fila de atendimento.

Fonte: Autor.

Figura 30- Tela de alteração de dados pessoais do médico.

Fonte: Autor.

Em relação às telas do enfermeiro, na figura 31 é apresentada a tela inicial do enfermeiro, após o *login* no sistema. As figuras seguintes, 32 e 33, são referentes à pré-consulta realizada pelo enfermeiro, com a adição de dados que serão utilizados pelo médico durante a consulta. Já na figura 34, é apresentada a tela de geração de senha de atendimento. Encerrando a apresentação das telas do enfermeiro, nas figuras 35 e 36 são apresentadas as telas de gerenciamento da agenda do consultório, comum aos enfermeiros e os assistentes do ambulatório.

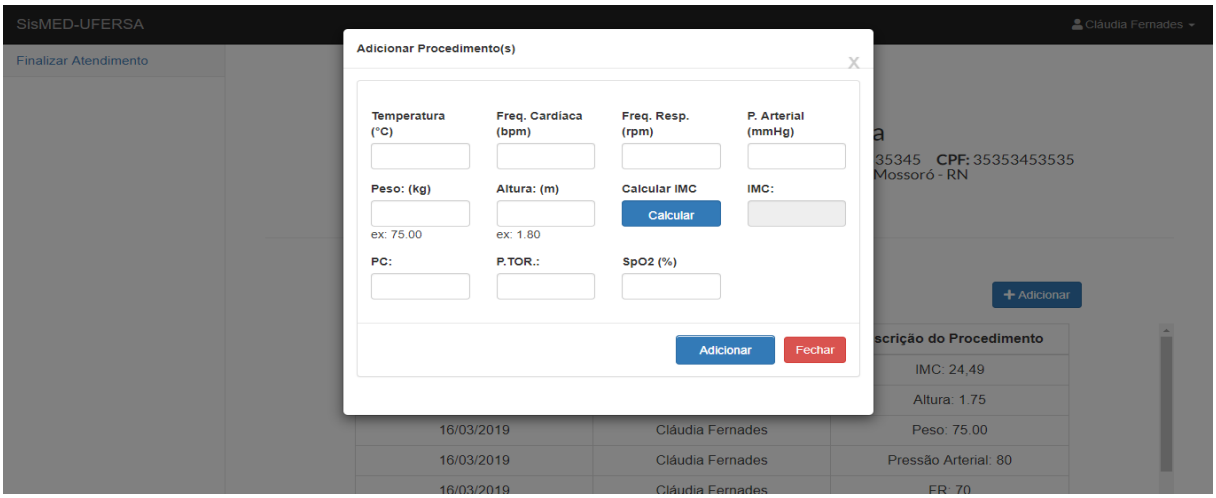
Figura 31- Tela inicial do enfermeiro.

Fonte: Autor

Figura 32- Tela de procedimento do enfermeiro.

Fonte: Autor.

Figura 33- Tela de adição de procedimento do enfermeiro.

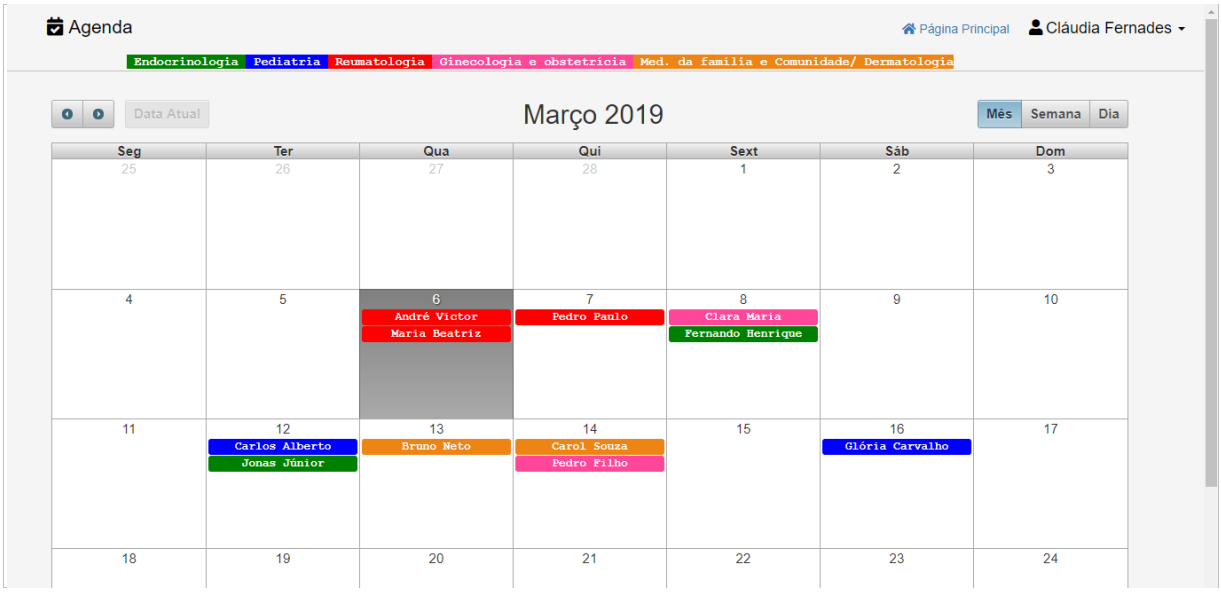


Fonte: Autor.

Figura 34- Tela do enfermeiro para geração de senha de atendimento.

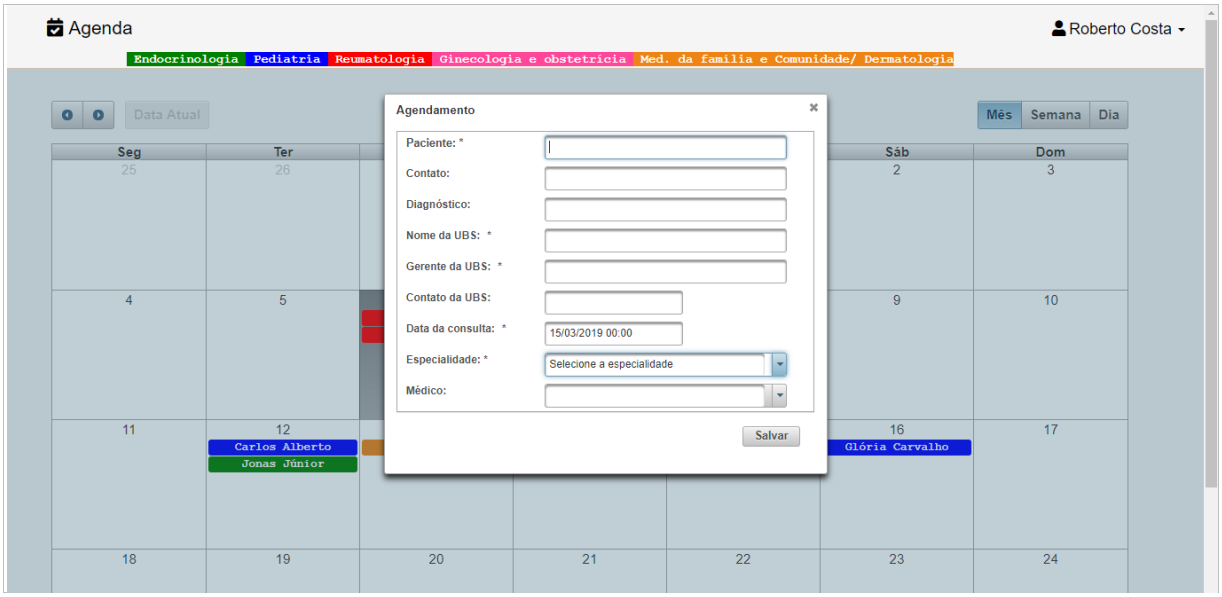
Fonte: Autor.

Tela 35- Tela da agenda de consultas comum aos enfermeiros e assistentes.



Fonte: Autor.

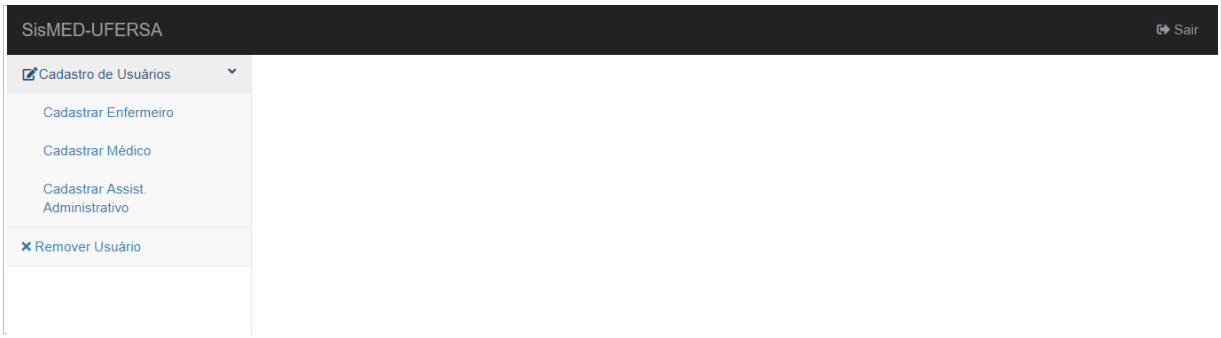
Figura 36- Tela de entrada de dados do agendamento de consulta comum aos enfermeiros e assistentes do ambulatório.



Fonte: Autor.

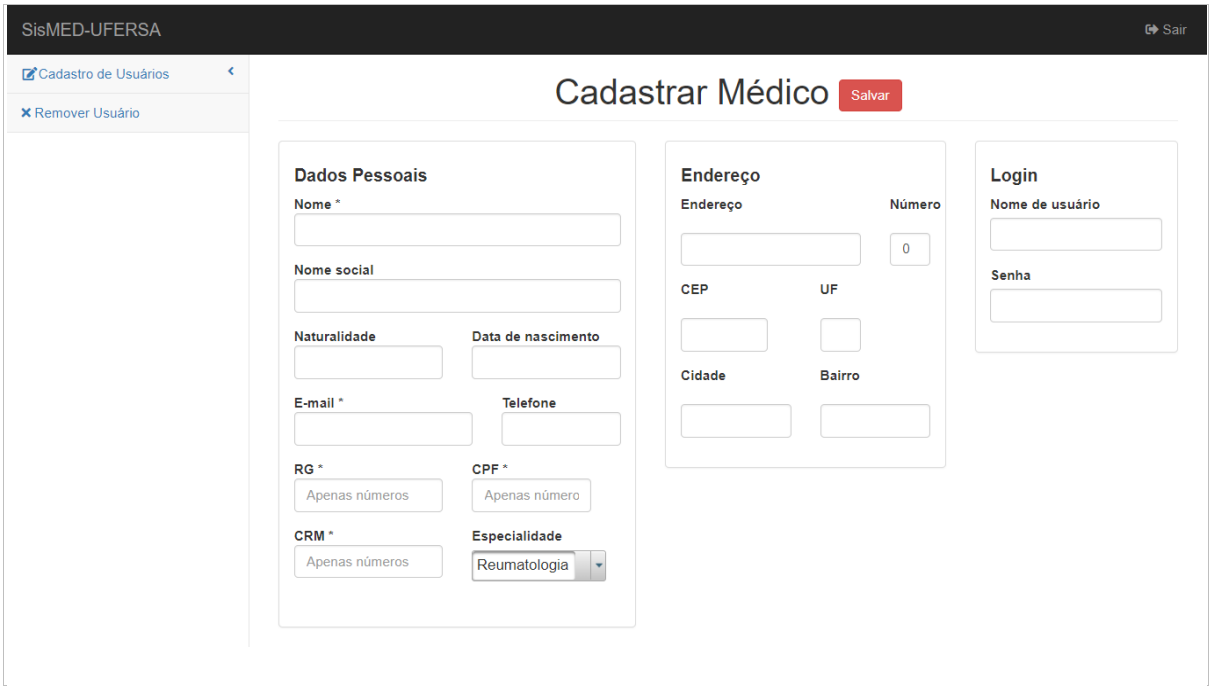
Por fim, é apresentada a tela inicial do administrador do sistema (figura 37), a tela para cadastro do médico (figura 38) e a tela visível aos pacientes com as senhas chamadas (figura 39).

Figura 37- Tela inicial do administrador do sistema.



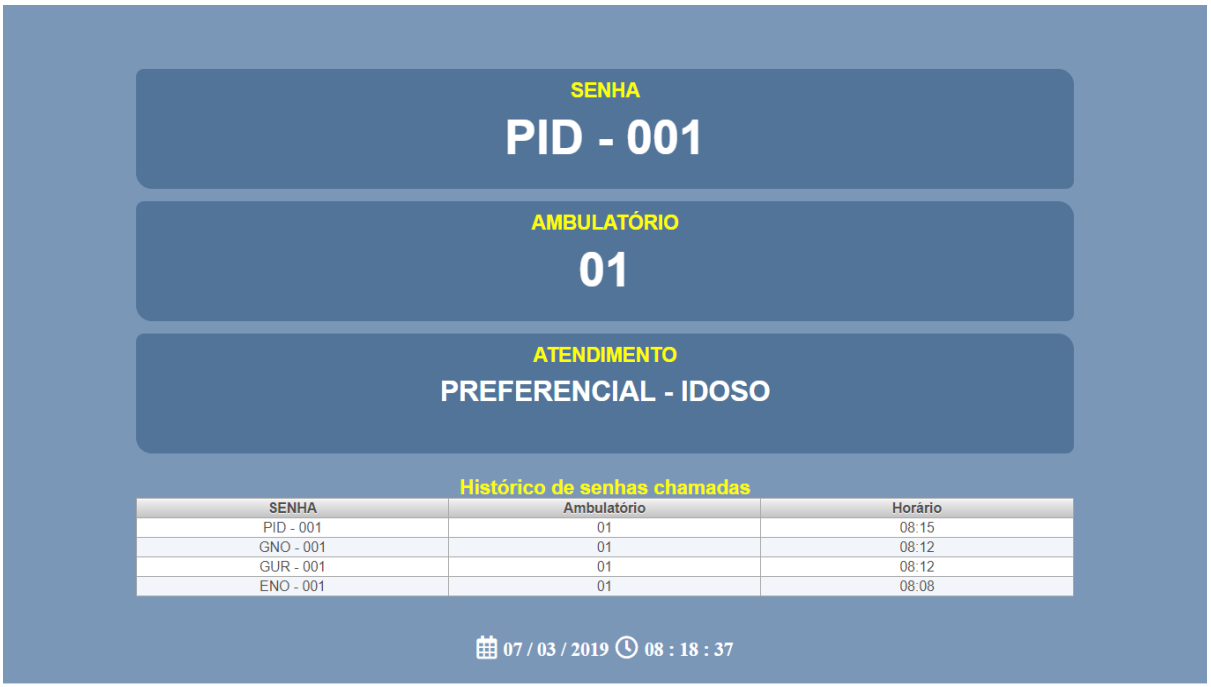
Fonte: Autor.

Figura 38- Tela de cadastro de médico.



Fonte: Autor.

Figura 39- Tela de chamada de senha visível aos pacientes.



Fonte: Autor.

4 VALIDAÇÃO

4.1. Introdução

A validação do sistema foi realizada por meio de um formulário disponibilizado digitalmente aos usuários do sistema.

4.2 Formulário de validação

O formulário de validação foi elaborado com 18 perguntas acerca da usabilidade, funcionalidades e satisfação com o sistema, conforme figura 40.

Figura 40- Formulário de validação do sistema.

1. Função exercida	1- Médico ○		2- Enfermeiro ○		3- Assistente ○
2. Leitura de caracteres na tela	1- Muito difícil ○	2 ○	3 ○	4 ○	5- Muito fácil ○
3. Disposição dos objetos na tela	1- Mal distribuídos ○	2 ○	3 ○	4 ○	5- Bem distribuídos ○
4. Informações disponíveis na tela	1- Totalmente Inadequadas ○	2 ○	3 ○	4 ○	5- Totalmente adequadas ○
5 - Terminologia utilizada no sistema	1- Totalmente Inadequada ○	2 ○	3 ○	4 ○	5- Totalmente adequada ○
6- Campo para entrada de texto na tela	1- Totalmente confuso ○	2 ○	3 ○	4 ○	5- Totalmente claro ○
7- As informações apresentadas no sistemas estão corretas?	1- Totalmente incorretas ○	2 ○	3 ○	4 ○	5- Totalmente corretas ○
8. Os recursos gráficos (imagens, ícones, etc) são adequados?	1- Totalmente Inadequados ○	2 ○	3 ○	4 ○	5- Totalmente adequados ○
9- As mensagens de erros apresentadas ajudam?	1- Sim ○		2 - Não ○		3- Às vezes ○
10. Aprender a operar o sistema	1- Muito difícil ○	2 ○	3 ○	4 ○	5- Muito fácil ○
11. Quão amigável é a interface do sistema?	1- Nada amigável ○	2 ○	3 ○	4 ○	5- Muito amigável ○
12. O tempo de resposta do sistema é adequado para a realização das atividades?	1- Totalmente Inadequado ○	2 ○	3 ○	4 ○	5- Totalmente adequado ○
13. O usuário pode se deslocar de uma tela para a outra rapidamente	1- Discordo totalmente ○	2 ○	3 ○	4 ○	5- Concordo totalmente ○
14. Quão bem-sucedido é o sistema na realização das funções a que ele se propõe a fazer?	1 - Mal Sucedido ○	2 ○	3 ○	4 ○	5- Bem sucedido ○
15- Qual a funcionalidade que você teve mais dificuldade?	Campo para resposta curta				
16. De forma geral, quão satisfeito está com o sistema?	1- Pouco satisfeito ○	2 ○	3 ○	4 ○	5- Muito satisfeito ○
17. Recomendaria o sistema para outra pessoa?	1- Sim ○		2 - Não ○		3- Talvez ○
18- Quais as sugestões de melhorias que você propõe?	Campo para resposta curta				

Fonte: Autor.

4.3 Resultados

Os principais resultados obtidos com a aplicação do questionário são apresentados a seguir:

Figura 41- Principais resultados obtidos da validação.

1. Função exercida	Médico	Enfermeiro	Assistente		
	0%	50%	50%		
2. Leitura de caracteres na tela	1- Muito difícil	2	3	4	5- Muito fácil
	0%	0%	0%	0%	100%
3. Disposição dos objetos na tela	1- Mal distribuídos	2	3	4	5- Bem distribuídos
	0%	0%	0%	0%	100%
4. Informações disponíveis na tela	1- Totalmente Inadequadas	2	3	4	5- Totalmente adequadas
	0%	0%	0%	0%	100%
5. Os recursos gráficos (imagens, ícones, etc) são adequados?	1- Totalmente Inadequados	2	3	4	5- Totalmente adequados
	0%	0%	0%	0%	100%
6. Aprender a operar o sistema	1- Muito difícil	2	3	4	5- Muito fácil
					100%
7. Quão amigável é a interface do sistema?	1- Nada amigável	2	3	4	5- Muito amigável
	0%	0%	0%	0%	100%
8. O tempo de resposta do sistema é adequado para a realização das atividades?	1- Totalmente Inadequado	2	3	4	5- Totalmente adequado
	0%	0%	0%	50%	50%
9. Quão bem-sucedido é o sistema na realização das funções a que ele se propõe a fazer?	1 - Mal Sucedido	2	3	4	5- Bem sucedido
	0%	0%	0%	50%	50%
10. De forma geral, quão satisfeito está com o sistema?	1- Pouco satisfeito	2	3	4	5- Muito satisfeito
				50%	50%
11. Recomendaria o sistema para outra pessoa?	1- SIM	2 - NÃO	3- Talvez		
	100%	0%	0%		

Fonte: Autor.

Um risco para a validade do formulário foi o baixo número de entrevistados, em decorrência da dificuldade em realizar a demonstração do uso do sistema e da validação com os médicos dos ambulatórios, sendo possível apenas a validação com a enfermeira e o assistente do ambulatório.

5 CONSIDERAÇÕES FINAIS

A elaboração do presente trabalho possibilitou o desenvolvimento de um sistema *Web* para gerenciamento de consultório médico. Além disso, permitiu demonstrar algumas das tecnologias, ferramentas e etapas do processo de desenvolvimento de aplicações *Web*.

Partindo do objetivo de desenvolver um sistema para o gerenciamento de consultório médico da UFERSA, verificou-se por meio da aplicação do formulário que o sistema desenvolvido atende às necessidades dos profissionais do ambulatório médico, como verificado na figura 41. Além do mais, reforça-se a importância do assunto apresentado no trabalho para os estudantes que desejam conhecer e aprofundar o conhecimento a respeito do desenvolvimento *Web*.

As limitações existentes no trabalho, devido ao número de usuários disponíveis para testar e avaliar o sistema, impossibilitou uma análise mais avançada dos dados e informações.

Por fim, considerando as limitações e os resultados obtidos com o presente trabalho, indica-se, para futuras pesquisas, a utilização do sistema e validação com mais usuários, com a finalidade de melhorar a qualidade do sistema e o aumento da amostra de dados, como também uma análise das melhorias proporcionadas em decorrência da utilização do sistema por parte dos usuários.

REFERÊNCIAS

AQUARELA. **O que é a web 3.0? Qual sua importância para os negócios?**. [S. l.], 2015. Disponível em: <https://www.aquare.la/web-3-0-e-sua-importancia-nos-negocios/>. Acesso em: 2 jul. 2018.

BARWINSKI, Luísa. **A WorldWide Web completa 20 anos, conheça como ela surgiu.** 2009. Disponível em: <https://www.tecmundo.com.br/historia/1778-a-world- wide-web-completa-20-anos-conheca-como-ela-surgiu.htm>. Acesso em: 12 jun. 2018.

BONATO, Antonio Sergio Ferreira. **Extreme programming e qualidade de software**. 2002. 15 f. Artigo. Universidade de São Paulo, São Paulo, 2002. Disponível em: <http://ftp://ftp.ufv.br/dpi/mestrado/XP/A%20-%20XP%20e%20Qualidade/Bonato,%20A%20-%20XP%20e%20Qualidade.pdf>. Acesso em: 03 ago. 2018.

CLEMENTE, Rafael. **A WorldWide Web completa 25 anos**: Em 12 de março de 1989 o britânico Tim Berners-Lee descreveu o protocolo de transferências de hipertextos. Barcelona. 2014. Disponível em: <https://brasil.elpais.com/brasil/2014/03/11/tecnologia/1394554623_973239.html>. Acesso em: 02 jul. 2018.

EIS, Diego. **O básico: o que é HTML?**:Entenda o HTML básico, saiba o que significa tags do HTML e entenda como fazer. 2011. Disponível em: <https://tableless.com.br/o-que-html-basico/>. Acesso em: 02 ago. 2018.

GAMMA, Erich et al. **Padrões de projeto**: Soluções Reutilizáveis de Software Orientado a Objetos. 1. ed. Brasil: Bookman, 2000. 368 p.

LEITE, Alessandro Ferreira. **Conheça os padrões de projeto**. 2005. Disponível em: <https://www.devmedia.com.br/conheca-os-padroes-de-projeto/957>. Acesso em: 26 jun. 2018.

LUCKOW, Décio Heinzelmann; MELO, Alexande Altair de. **Programação Java para a Web**. 2. ed. São Paulo: Novatec, 2015. 680 p.

LUPIANHEZ, Leandro; LUCRÉDIO, Daniel. **Utilizando padrões de Projeto JEE no desenvolvimento de aplicações web: Um estudo de caso**. **T.I.S**, São Carlos, v. 1, n. 1, p. 09-19, jul. 2012. Disponível em: <http://revistatis.dc.ufscar.br/index.php/revista/article/viewFile/10/11>. Acesso em: 05 ago. 2018.

MEDEIROS, Higor. **Introdução ao ExtremmingProgramming (XP)**: Veja neste artigo o que são métodos ágeis, o que é Extremming Programming, como surgiu e quais são as suas principais características e seus valores. 2013. Disponível em: <https://www.devmedia.com.br/introducao-ao-extreme-programming-xp/29249>. Acesso em: 21 jul. 2018.

MOZILLA. **Introdução**. 2017. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Introduction>. Acesso em: 07 ago. 2018.

ORACLE. **Java Platform, Enterprise Edition (Java EE) 8**: The Java EE Tutorial. 1. 2017. Disponível em: <https://javaee.github.io/tutorial/overview.html>. Acesso em: 05 ago. 2018.

PEREIRA, Ana Paula. **O que é CSS?**. [S. l.], 9 set. 2009. Disponível em: <https://www.tecmundo.com.br/programacao/2705-o-que-e-css-.htm>. Acesso em: 31 jul. 2019

PRIMO, Alex. **O aspecto relacional das interações na Web 2.0**. E- Compós (Brasília), v. 9, p. 1-21, 2007. Disponível em: <http://www.ufrgs.br/limc/PDFs/web2.pdf>. Acesso em: 02 ago. 2018.

ROHR, Altieres. **Como a 'web 3.0' criou um caos na privacidade**. 2018. Disponível em: <http://g1.globo.com/tecnologia/blog/seguranca-digital/post/como-web-30-criou-um-caos-na-privacidade.html>. Acesso em: 02 ago. 2018.

SCHIECK, Rodrigo. **Introdução ao PrimeFaces**. [S. l.], 2015. Disponível em: <https://www.devmedia.com.br/introducao-ao-primefaces/33139>. Acesso em: 10 nov. 2018.

SILVA, Giancarlo. **O que é e como funciona a linguagem JavaScript?**. 2015. Disponível em: <https://canaltech.com.br/internet/O-que-e-e-como-funciona-a-linguagem-JavaScript/>. Acesso em: 07 ago. 2018.

SILVA, Izalmo Primo da. **Desenvolvendo com Hibernate**. [S. l.], 2009. Disponível em: <https://www.devmedia.com.br/desenvolvendo-com-hibernate/14756>. Acesso em: 10 nov. 2018.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. Brasil: Pearson Education do Brasil, 2011. 544 p.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira. **Engenharia de requisitos:** Software orientado ao negócio. 1. ed. [S.l.]: BRASPORT Livros e Multimídia Ltda, 2016. 328 p. v. 652.8.