INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS AVANÇADO LAJES
CURSO TÉCNICO INTEGRADO EM INFORMÁTICA

YANN CARLOS SILVA DE MORAIS

MAIS SAÚDE: PLATAFORMA MÓVEL DOS SERVIÇOS DIGITAIS DAS UNIDADES BÁSICAS DE SAÚDE LOCAIS

LAJES/RN
2021

YANN CARLOS SILVA DE MORAIS

MAIS SAÚDE: PLATAFORMA MÓVEL DOS SERVIÇOS DIGITAIS DAS UNIDADES BÁSICAS DE SAÚDE LOCAIS

Relatório de Prática Profissional apresentado ao Curso Técnico Integrado em Informática do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, Campus Avançado Lajes, em cumprimento às exigências legais como requisito parcial à obtenção do título de Técnico em Informática.

Orientador (a): Prof. Fernando H. L. Soares

LAJES/RN

2021

RESUMO

A Secretaria Municipal de Saúde é o órgão responsável pelo planejamento, organização, coordenação e execução dos programas, projetos e atividades voltadas para a implantação das políticas de saúde do Município de Lajes/RN. Ao total, existem 5 (cinco) unidades básicas de saúde no município que oferecem atendimento à população baseado na distribuição de fichas impressas e cedidas por ordem de chegada ao local de distribuição. Isto por vezes causa conflito, em relação a horários, entre os habitantes da região. Com base nessas informações, construímos este projeto a fim de propor um sistema *web* para facilitar o acesso à informação e promover um melhor atendimento de saúde à população do município de Lajes/RN.

Palavras-chave: Aplicativo. Postos de Saúde. Cidadãos de Lajes/RN.

SUMÁRIO

1 INTRODUÇÃO

Distanciando-se 125 km da capital do Estado do Rio Grande do Norte – Natal, o Município de Lajes encontra-se inserido regionalmente na Mesorregião Central Potiguar, mais especificamente na Microrregião de Angicos. Com uma área total de 665,7km^2 (equivalente a 1,25% da superfície estadual), Lajes limita-se a Norte com os Municípios de Jandaíra e Pedra Preta, ao Sul com Cerro-Corá e São Tomé, a Leste com Jardim de Angicos, Caiçara do Rio dos Ventos e Pedra Preta, e a Oeste com Fernando Pedroza, Pedro Avelino e Angicos, estando a sede municipal situada a uma altitude média de 199m. Sua população estimada, segundo dados do IBGE, é de 11.208 para o ano de 2018 (IBGE, 2019).

A cidade possui um hospital no qual funciona, também, como pronto socorro e maternidade e que presta serviços de urgência, emergência, internamentos, partos e pequenas cirurgias tanto para à população da cidade como também aos visitantes. A cidade também possui três clínicas médicas particulares (sendo duas delas clínicas odontológicas), nas quais, uma delas dispõe convênio, e 5 (cinco) postos de saúde que atendem a população e seus visitantes com os serviços de clínico geral, odontologia e enfermagem em geral. Todos os centros médicos estão em um bom estado de conservação. A unidade Mariana Gomes trabalha com a equipe dos PSF's (Programa de Saúde da Família), do ESF (Programa Estratégico Social da Família) e do NASF (Núcleo de Apoio à Saúde da Família) que tem os serviços de nutricionista, fisioterapia, fonoaudiologia e psicologia, sendo quatro na zona urbana e um na zona rural na comunidade de Firmamento. E foram localizadas cinco farmácias, sendo uma pertencente à franquia Unifarma (PREFEITURA DE LAJES/RN, 2019).

De acordo com entrevista realizada com Sâmara Bridget Botelho de Figueiredo, Secretária Municipal de Saúde, o atendimento à população é feito da seguinte forma: cada unidade de saúde disponibiliza 20 atendimentos diários (fichas) para médicos e 10 fichas para dentistas e o número de enfermeiras varia de acordo com a demanda de pacientes. Além desses atendimentos, a equipe da NASF (Núcleo de Assistência à Família) oferece outros tipos de serviço, como por exemplo psicólogos, assistentes sociais e fonoaudiólogo (PREFEITURA DE LAJES/RN, 2019).

Segundo pesquisa realizada com pacientes nos postos de saúde do município de Lajes, há insatisfação com o atendimento nas unidades de saúde da cidade, todos

relataram a má desenvoltura que as unidades de saúde têm. Uma das reclamações é a falta de organização na distribuição de fichas. A comunidade se dirige aos postos com frequência e muita antecedência, mas nem sempre conseguem ser atendidos. Também foi relatado que existe uma dificuldade de acesso às informações sobre as unidades de saúde.

Neste contexto, acreditamos que é importante democratizar o acesso aos serviços de saúde disponíveis no país, por isto este projeto propõe implementar um sistema *web* que promova melhorias de saúde no município de Lajes/RN.

2 DADOS GERAIS DA EXTENSÃO

Título do projeto: Mais Saúde: Plataforma Móvel dos Serviços Digitais das Unidades Básicas de Saúde Locais.

Período de realização: de 11/06/2019 a 30/03/2020.

Total de horas: mínimo de 340 horas.

Orientador: Prof. Fernando Helton Linhares Soares.

Função: Professor.

Formação profissional: Mestre.

Quadro 1 – Síntese das Atividades do Aluno no Projeto.

CARGA HORÁRIA	**ATIVIDADES DESENVOLVIDAS**	**RESULTADOS ALCANÇADOS**
15h	Projeto de *design*	O projeto foi pensado de uma maneira que o seu resultado agradasse a sociedade. Estando simples e prático ao manuseio.
15h	Elaboração do questionário	O questionário foi elaborado com 14 questões básicas sobre atividades que os pacientes executam para conseguir uma fixa e perguntas para saber se eles usariam o sistema *web*.
15h	Aplicação do questionário	Conseguimos aplicar o questionário de uma maneira *online*, onde mandamos os *links* pelo *WhatsApp* e os entrevistados respondiam.
15h	Consolidação das respostas do questionário.	A partir das respostas obtidas no questionário, percebemos o quão grave é a situação da população. Onde a maioria passa mais de cinco horas esperando uma ficha de atendimento.

15h	Implementação	O sistema *web* foi implementado, e como resultado foi um sistema *web* onde o paciente consegue facilmente pegar e olhar suas fichas, assim como ver as informações básicas da sua UBS.
15h	Divulgação do projeto nos postos de saúde	Não realizado devido a pandemia do COVID-19.

Fonte: autoria própria (2019).

3 FUNDAMENTAÇÃO TEÓRICA

3.1 DOCUMENTO DE VISÃO DE *SOFTWARES*

De acordo com o *site* da *IBM (International Business Machines Corporation)*, o documento de visão define o escopo de alto nível e o propósito de um programa, produto ou projeto. O documento de Visão é criado no início das atividades de concepção e serve como base para a modelagem dos casos de uso de um sistema (IBM, 2020). Esse possui geralmente a seguinte estrutura dividida em oito tópicos, listados abaixo:

1. **Introdução:** fornece uma visão geral de todo o documento, incluindo o propósito, escopo, definições, acrônimos, abreviações, referências e visão geral de todo o documento (*IBM*, 2020);
2. **Descrições da Parte Interessada e do Usuário:** fornece um perfil das partes interessadas e usuários que estão envolvidos no projeto. Além de fornecer, ela também identifica os principais problemas e quais soluções propostas deva tratar (*IBM*, 2020);
3. **Visão Geral do Produto:** Esta seção fornece uma visualização de alto nível das capacidades do produto, *interfaces* para outros aplicativos e configurações dos sistemas. Esta seção, em geral, consiste em outras três subseções: Perspectiva do Produto; Funções do Produto; e Suposições e Dependências (*IBM*, 2020);
4. **Recursos do Produto:** lista e descreve resumidamente os recursos do produto. Os recursos são capacidades de alto nível do sistema que são necessários para entregar benefícios aos usuários (*IBM*, 2020).
5. **Restrições:** analisa todas as restrições que possam vir a surgir no programa final, tais como restrições de *design*, restrições externas, ou outras dependências (*IBM*, 2020).
6. **Faixas de Qualidade:** define as faixas de qualidade para desempenho, robustez, tolerância a falhas, usabilidade e características similares que o conjunto de recursos não descreve (*IBM*, 2020).
7. **Precedência e Prioridade:** define quais são as prioridades dos recursos do sistema (*IBM*, 2020).
8. **Outros Requisitos do Produto:** em um alto nível, lista os padrões aplicáveis, os requisitos de *hardware* ou plataforma, os requisitos de desempenho e os

requisitos ambientais (*IBM*, 2020).

3.2 DESENVOLVIMENTO *WEB*

O advento e ascensão da *Internet* trouxe consigo um crescimento exponencial do número de *sites* criados nos últimos anos. Graças a isso, a área de desenvolvimento *web* passou de um pequeno detalhe, para um pilar de grande importância na hora da construção de um *site*. Este fato deve-se a sua fácil aplicação e portabilidade, pois as páginas *web* são muito vistas como alternativas, ou até, como complementação de um aplicativo ou *software* que pode ser instalado em uma máquina. Por estarem hospedadas na rede mundial de computadores, o seu acesso é simples, pois não há necessidade de baixar nenhum arquivo em seu dispositivo, dando assim, uma boa vantagem a esta área.

O desenvolvimento *web* se divide em duas partes principais:

- **Front-*end***

O *Front-end* funciona como uma porta de entrada para qualquer *site*, pois é nele que é mostrado todo o *design* e preocupação dos desenvolvedores com a atratividade do *site*. O desenvolvimento *front-end* é baseado em três principais linguagens, a *Hypertext Markup Language* (*HTML*), a *Cascading Style Sheet* (*CSS*) e a *Javascript* (*JS*).

- ***Back-end***

O *Back-end* é a parte escondida e funcional do *site*, é nela que ficam guardadas todas as informações programadas. O banco de dados, as informações de *login*, como seu nome de usuário e senha, além dos sistemas de processamento de dados.

Segundo o *site Scriptcase* (2020) “Com a chegada do desenvolvimento *Web*, um novo conceito foi possível criar aplicações muito mais inteligentes. Aproveitando todos os recursos que a *Internet* disponibiliza.”.

Logo, o desenvolvimento *web* por sua vez tende a evoluir cada vez mais, abrindo os nossos horizontes para uma praticidade e comodidade bem maior do que já foi visto.

3.2.1 *HTML (Hypertext Markup Language)*

O *HTML* é uma linguagem de marcação muito utilizada para o desenvolvimento de *sites*. A sigla *HTML* significa *Hypertext Markup Language*. Por ser de fácil entendimento tanto para seres humanos, quanto para máquinas, o *HTML* torna-se uma linguagem muito difundida no mundo inteiro. Criado por Tim Berners-Lee, foi inicialmente utilizado para troca de informações entre ele e um grupo de amigos. A base do *HTML* são as *tags* que são utilizadas para dizer ao navegador o que é cada informação. Geralmente, as *tags* vêm em pares, como e , sendo respectivamente utilizadas para abrir e fechar a *tag* (*TABLELESS*, 2011).

Segundo o livro Criando *Sites* com *HTML* de Maurício Samy Silva:

> HTML é a sigla em inglês para *Hypertext Markup Language*, que, em português, significa linguagem para marcação de hipertexto. O conceito de hipertexto admite um sem-número de considerações e discussões que fogem ao escopo deste livro. Para o bom entendimento das definições, podemos resumir hipertexto como todo o conteúdo inserido em um documento para *web* e que tem como principal característica a possibilidade de se interligar a outros documentos da *web*. O que torna possível a construção de hipertextos são os links, presentes nas páginas dos *sites* que estamos acostumados a visitar quando entramos na internet (SILVA, 2008).

Fonte: autoria própria, 2020.

```
1 <p> Isso é uma parágrafo. </p>
```

Figura 1: exemplo da *tag* <p>.

Fonte: autoria própria, 2020.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title></title>
5 </head>
6 <body>
7
8 </body>
9 </html>
```

Figura 2: início padrão do código.

A linguagem possui centenas de *tags* para centenas de possibilidades oferecidas, como *tags* para criar tabelas, formulários, parágrafos, títulos, inserir imagens etc.

3.2.2 *CSS (Cascading Style Sheet)*

CSS é uma linguagem muito utilizada para definir a aparência de páginas *HTML*. É ela que define como serão mostrados os elementos que estão inseridos no código, sendo o navegador responsável por interpretá-los. Por toda essa interação entre as ferramentas, hoje é bastante incomum fazer um arquivo *HTML* e não pensar no seu estilo. O acrônimo *CSS*, significa *Cascading Style Sheet* ou Folhas de Estilo em Cascata. Atualmente, a grande maioria dos *sites* possuem elementos conhecidos do *CSS*, como menus, botões personalizados, cabeçalhos e rodapés (*HOSTINGER*, 2019).

Para poder escrever um código *CSS*, é preciso seguir uma regra básica de sintaxe. Essa regra é composta por três partes, um seletor que é o alvo da regra de estilo, uma propriedade que define o que será estilizado e um valor que define quanto e como será estilizado. Veja o exemplo a seguir, h1 (seletor), *color* e *text-align* (propriedades), *green* e *center* (valores):

Fonte: autoria própria, 2020.

```
1 h1{
2     color: green;
3     text-align: center;
4 }
```

Figura 3: exemplo de código *CSS*.

Os códigos *CSS* tanto podem ser implementados no mesmo arquivo *HTML*, apenas precisando abrir e fechar uma *tag (<style> </style>)*, como também podem ser escritos em um documento separado, sendo preciso conectar os dois arquivos em outro momento.

Fonte: autoria própria 2020.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title></title>
5  <style>
6      h1{
7          color: green;
8          text-align: center;
9      }
10 </style>
11 </head>
12 <body>
13
14 </body>
15 </html>
```

Figura 4: exemplo de *CSS* na página *HTML*.

Fonte: autoria própria, 2020.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title></title>
5      <link rel="stylesheet" type="text/css" href=
       "1.css">
6  </head>
7  <body>
8
9  </body>
10 </html>
```

Figura 5: exemplo de *CSS* separado do *HTML*.

3.3 BANCO DE DADOS (*MYSQL*)

De forma sugestiva, um banco de dados organiza informações (dados) estruturadas em um sistema de computador. Para poder usufruir melhor da ferramenta é preciso de um sistema de gerenciamento de banco de dados (SGBD). Os dados mais comuns atualmente são arquitetados em linhas e colunas de várias tabelas, dessa forma, torna-se mais fácil e eficiente a consulta das informações (dados) necessárias. Tal consulta só é possível pela linguagem *SQL* que é muito utilizada por vários sistemas de gerenciamento (SGBD) para manipular e definir o acesso.

Segundo o livro *MySQL* - Guia do Programador, o *MySQL* é um sistema de gerenciamento de banco de dados relacional de licença dupla, sendo uma dessas *open source*. Desenvolvido inicialmente para aplicações de pequeno e médio porte, mas que hoje atende grandes aplicações. Este SGBD é bastante conhecido por ser o *software* livre que tem maior capacidade de concorrer com outros programas similares (MILANI, 2007, p. 22).

> O *MySQL* é uma ferramenta gráfica para trabalhar com servidores e bancos de dados *MySQL*. Permite criar e gerenciar conexões com servidores de banco de dados. Além de permitir que você configure parâmetros de conexão, o *MySQL Workbench* oferece a capacidade de executar consultas *SQL* nas conexões do banco de dados usando o Editor *SQL* embutido (*MySQL*, 2020).

3.4 *SPRING BOOT*

O *Spring Boot* é um *framework* Java que facilita todo o processo de construção de um *software* ou sistema. Ele torna a programação mais rápida, segura e simplificada, pois faz diversas operações que antes seriam feitas pelo programador. A classe de ligação entre código e o banco, que por vezes é uma classe que demanda bastante tempo por ser grande, é simplificada em apenas dois métodos. Além disso, essa ferramenta Java é recheada com bibliotecas flexíveis de dependências, que podem ser utilizadas por todos os usuários, e para todos os serviços, como *streaming* de TV, carros conectados, compras *online*, etc. O *Spring* também recebe contribuições dos grandes nomes da tecnologia, tais como Amazon, Google, Microsoft, etc. Atualmente, o *Spring* se coloca como o *framework* Java mais popular do mundo (SPRING, 2020).

3.5 *ECLIPSE IDE*

Segundo o *site* oficial do *Eclipse*, o Projeto *Eclipse* foi inicialmente criado pela a empresa IBM *(International Business Machines Corporation)* em novembro de 2001 e continuado por um consórcio de fornecedores de *software*. Logo após sua criação, o Eclipse ganhou incrível notoriedade, por ser um *software* de código livre, ou, *open source*. Graças a isso, a *Eclipse Foundation* foi criada em janeiro de 2004 como uma corporação sem fins lucrativos para atuar como administradora da comunidade *Eclipse* (*ECLIPSE FOUNDATION*, 2020).

> A *Eclipse Foundation* fornece um ambiente maduro, escalonável e favorável aos negócios para colaboração e inovação em *software* de código aberto. A fundação abriga o *IDE Eclipse* e mais 350 projetos de código aberto. Além de ferramentas e estruturas para uma ampla gama de domínios de tecnologia, como Internet das Coisas, automotivo, geoespacial, engenharia de sistemas e muitos outros (*ECLIPSE FOUNDATION*, 2020).

O *Eclipse* é um *IDE* (Ambiente de Desenvolvimento Integrado) para desenvolvimento de várias linguagens com programação, como Java, *PHP*, *Python*, C, C++, etc. Umas das principais vantagens do *Eclipse* é o uso do *SWT (Standard Widget Toolkit)*, e a forte orientação ao desenvolvimento baseado em *plugins*, ampliando o suporte do desenvolvedor com centenas deles, que procuram atender as diferentes necessidades (DEVMEDIA, 2012). E mais, podemos citar novamente que o *Eclipse* é um *software open source*, ou, livre de patentes. Ele também é bastante portátil, o que possibilita o funcionamento em vários ambientes.

3.6 *BOOTSTRAP* 4

O *Bootstrap* é um *framework* gratuito para o desenvolvimento *HTML (Hypertext Markup Language)*, *CSS (Cascading Style Sheet)* e *JS (Javascript)*. Originalmente criado por Mark Otto e Jacob Thornton, em 2011, tornou-se um dos mais populares *frameworks front-end* do mundo. Atualmente, o *Bootstrap* é mantido por uma equipe reduzida de desenvolvedores no *GitHub*, que se empenha em criar e manter *plug-ins* de *JavaScript* personalizados e melhorando os processos de criação de ferramentas para código *front-end* (BOOTSTRAP, 2020).

O *Bootstrap* nos permite como ferramenta, otimizar o tempo e o trabalho que gastaríamos para personalizar uma página *web*. Isso acontece porque temos acesso a vários componentes padrões na documentação, tais como: *buttons*, *cards*, *carousels*, *forms*, *dropdowns*, *navbars*, etc. Além disso, podemos editar todos os componentes de qualquer forma. Podemos mudar a cor, o tamanho, a fonte, os ícones, etc.

Existem duas formas mais conhecidas de instalar o *framework*. A primeira forma é baixando a biblioteca compactada já pronta, com todas as tags e com todos os códigos *Javascript*. A segunda, conhecida como "*CDN*", não precisa baixar nada, pois é utilizado um código que deverá ser colado no arquivo *HTML* que deseja personalizar. O problema dessa última forma é que uma conexão à *Internet* é necessária, pois a biblioteca não foi baixada no seu computador.

4 METODOLOGIA

Este projeto consiste num trabalho original explicativo de natureza mista, conforme definido em Wazlawick (2014). A abordagem *Design Thinking* para Educadores (IDEO, 2010) foi utilizada como inspiração para definição desta metodologia. Neste projeto serão seguidos os seguintes procedimentos metodológicos:

Etapa 1: Descoberta e Interpretação	Realizar revisão bibliográfica sobre estudos primários relacionados ao uso de sistemas de gerenciamento de atendimento médico;
	Realizar mapeamento sobre tecnologias utilizadas no desenvolvimento de sistemas para saúde.
	Definir modelo de domínio do sistema a ser construído.
Etapa 2: Ideação e Experimentação	Realizar sessões sistemáticas de *brainstorming* e triagem de ideias;
	Propor protótipo descartável de um sistema para suporte ao acesso à informação e agendamento de atendimentos médicos no âmbito do município de Lajes/RN;
	Realizar estudo de caso do protótipo proposto;
	Implementar o sistema proposto;
	Realizar estudo de caso do sistema desenvolvido.
	Sistematizar e analisar resultados deste projeto de extensão;

Etapa 3: Publicação e Entrega de Relatório.	Comparar resultados do projeto de extensão com trabalhos relacionados;
	Promover resultados desse projeto de extensão junto à comunidade de Lajes;
	Entregar relatório final deste trabalho.

5 CARACTERIZAÇÃO DAS ATIVIDADES DESENVOLVIDAS

5.1 REQUISITOS

Foram utilizadas duas técnicas na elicitação de requisitos. Estudo de sistemas semelhantes e entrevistas.

No primeiro caso, foram analisados sistemas semelhantes ao desta proposta para identificar informações possíveis de serem utilizadas no projeto de elicitação de requisitos. Os sistemas utilizados na pesquisa foram Aplicativo Banco do Brasil e MeuDigiSUS.

E, no segundo caso, a entrevista foi utilizada com os pacientes das unidades básicas e com a secretaria de saúde.

As duas técnicas possibilitaram a coleta de boa parte das informações necessárias para o desenvolvimento dos requisitos do sistema, como fluxo de trabalho, e detalhes de formulários utilizados em clínicas.

5.1.1 Requisitos Funcionais

Os requisitos funcionais descrevem a funcionalidade ou os serviços que se espera que o sistema realize em benefício dos usuários (PAULA FILHO, 2000):

- **RF001 - Acessar o Sistema:** o sistema deve permitir aos usuários terem acesso a suas respectivas áreas de acesso. Os **usuários avulsos** (não cadastrados) terão a possibilidade de ver a tela de cadastro, tela de login e as informações sobre as UBS's. Os **usuários pacientes** (já cadastrados no sistema), terão acesso a solicitação de fichas e as informações sobre as UBS's. Os **usuários agentes** (administradores) terão acesso às páginas de solicitações de fichas feitas, pacientes cadastrados, cadastro de UBS e cadastro de especialidades.
- **RF002 - Cadastro de Paciente:** o sistema deve permitir a usuários não identificados se cadastrar no sistema.
- **RF003 - Gerenciar UBS:** o sistema deve permitir aos agentes de saúde cadastrar e deletar as informações de uma UBS.
- **RF004 - Gerenciar Distribuição de Fichas da UBS:** o sistema deve permitir aos agentes de saúde confirmar ou negar as solicitações de fichas realizadas.
- **RF006 - Solicitar Ficha:** o sistema deve permitir aos pacientes solicitar uma

ficha de atendimento a partir da definição da UBS, especialidade e dia de atendimento.

5.1.2 Requisitos Não Funcionais

Requisitos não funcionais são os requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas (Wikipédia, 2020).

Este parágrafo apresenta a descrição dos requisitos não funcionais do produto, ou seja, a maneira que o sistema deve se comportar.

- **RNF001 - Limitação de caracteres:** o sistema deve limitar os campos de texto até 255 caracteres.
- **RNF002 - Acesso Restrito:** o sistema deverá permitir o acesso dos usuários às áreas restritas conforme suas permissões de segurança.

Este parágrafo apresenta a descrição dos requisitos não funcionais organizacionais.

- **RNF003 - Linguagem de Programação:** a implementação do sistema deve utilizar a linguagem Java, HTML, CSS e JS.
- **RNF004 - Banco de Dados:** a implementação do sistema deve empregar o MySQL Server como servidor de banco de dados.

5.2 CASOS DE USO

Antes da introdução conceitual sobre os casos de uso, é preciso conhecer um pouco mais sobre o que é a linguagem *UML*.

Segundo o livro *UML*: Guia do Usuário, a *UML (Unified Modeling Language)* é uma linguagem-padrão para elaboração da estrutura de projetos de *software*. Ela poderá ser empregada para visualização, para especificação, para construção e para documentação de artefatos que façam uso de sistemas complexos de *software* (BOOCH; RUMBAUGH; JACOBSON, 2006, p. 13). Em outras palavras, a UML é usada para reproduzir graficamente os requisitos e o funcionamento do site, auxiliando assim, no seu desenvolvimento. A *UML* é uma linguagem e, portanto, é somente uma parte de um método para desenvolvimento de *software*. A *UML* é independente do processo, apesar de ser perfeitamente utilizada em processo orientado a casos de usos, centrado na arquitetura, iterativo e incremental (BOOCH; RUMBAUGH; JACOBSON, 2006, p. 13).

Os casos de uso, nada mais são que uma forma para representar os requisitos do programa, suas funcionalidades e como irá funcionar. Ele representa uma utilização do sistema por um usuário, que aqui é chamado de ator. Além disso, os casos de uso podem auxiliar no levantamento de requisitos do programa, facilitando trabalhos futuros.

O Mais Saúde é um sistema *web* que permite aos usuários (pacientes) fazerem solicitações de fichas para atendimento nas devidas UBS diretamente de casa. Com ele será possível a otimização do tempo e do trabalho, que antes seria gasto com filas de espera para apenas conseguir uma ficha. As funcionalidades do Mais Saúde serão exibidas no diagrama de casos de uso abaixo:

Fonte: autoria própria, 2020.

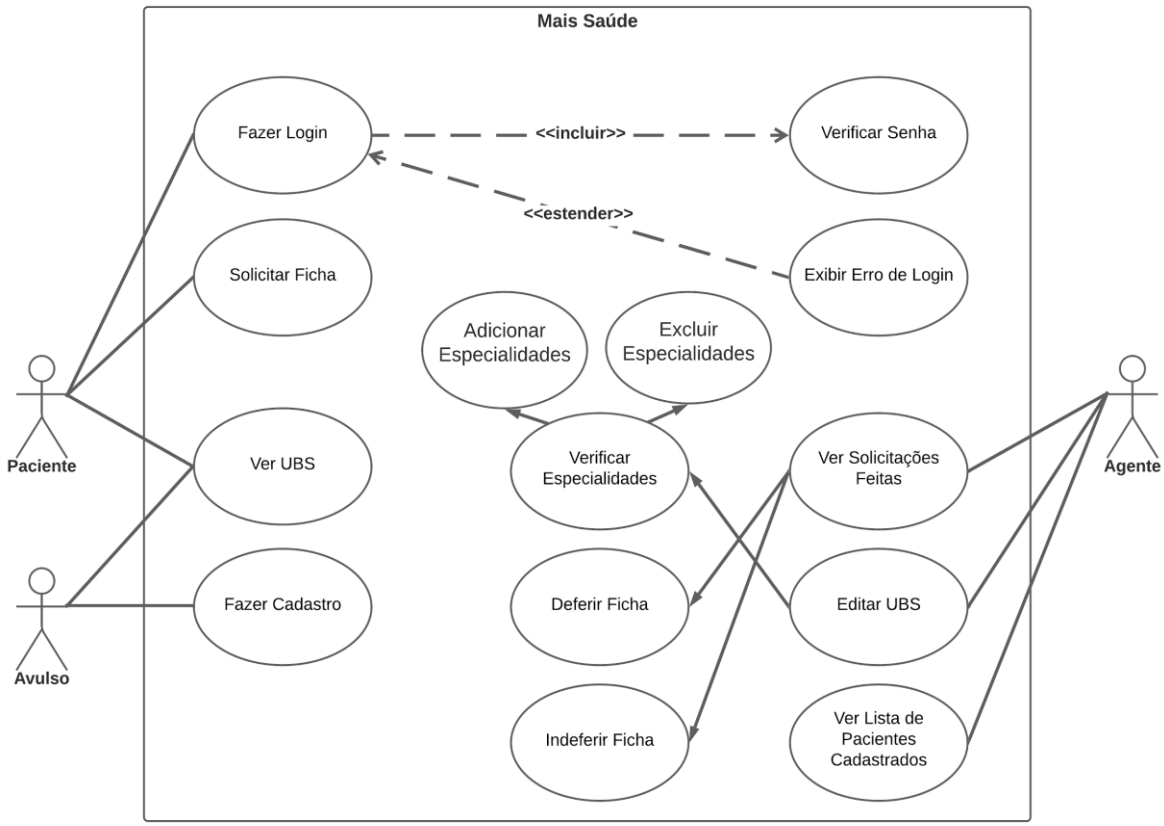

Figura 6: Diagrama de casos de uso Mais Saúde.

Como exposto no diagrama acima, podemos perceber os diferentes níveis de acesso ao sistema. Cada nível chamaremos de módulo, sendo assim, temos o módulo avulso, o módulo paciente e por fim, o agente.

O usuário avulso possui duas funcionalidades representadas por duas elipses com os nomes “Ver UBS” e “Fazer Cadastro”. Assim que um usuário avulso faz cadastro no sistema, ele automaticamente torna-se um paciente e é redirecionado para a tela inicial para poder fazer *login*.

Já o paciente possui três funcionalidades representadas por três elipses com os nomes “Fazer Login”, “Solicitar Ficha” e “Ver UBS”. Assim que o usuário preencher os dados de *login* e clicar em “entrar”, esses dados preenchidos por ele serão verificados, se estiverem corretos, ele entrará com sucesso, se não, será exibido um erro de *login*. A diferença entre <<incluir>> e <<estender>> é que o <<incluir>> sempre irá acontecer, já que é uma medida de segurança. Já o <<estender>>, só irá acontecer caso algum dado de *login* esteja errado.

Por fim, o agente possui três funcionalidades representadas por três elipses

com os nomes “Ver Solicitações Feitas”, “Editar UBS” e “Ver Lista de Pacientes Cadastrados”. Na funcionalidade “Ver Solicitações Feitas” o agente terá a possibilidade de deferir ou não a ficha solicitada pelo o paciente. Na funcionalidade “Editar UBS” será possível ver as informações das unidades básicas de saúde e gerenciar as especialidades para aquela UBS: adicioná-las ou excluí-las. Já na última funcionalidade, o agente poderá ver uma lista com todos os pacientes cadastrados, caso ele queira saber alguma informação, ou resolver algum problema.

5.2.1 Diagrama de Classes

Basicamente, o diagrama de classes representa as estruturas e os relacionamentos entre as classes modelo de um projeto. Para a sua concepção é necessário antes saber como se dar um diagrama de classes na linguagem UML *(Unified Modeling Language)*. Nessa linguagem, uma classe é representada por um retângulo, que geralmente, é dividida em 3 (três) partes: o nome da classe, os atributos e seus tipos e, por fim, seus métodos.

Fonte: autoria própria, 2020.

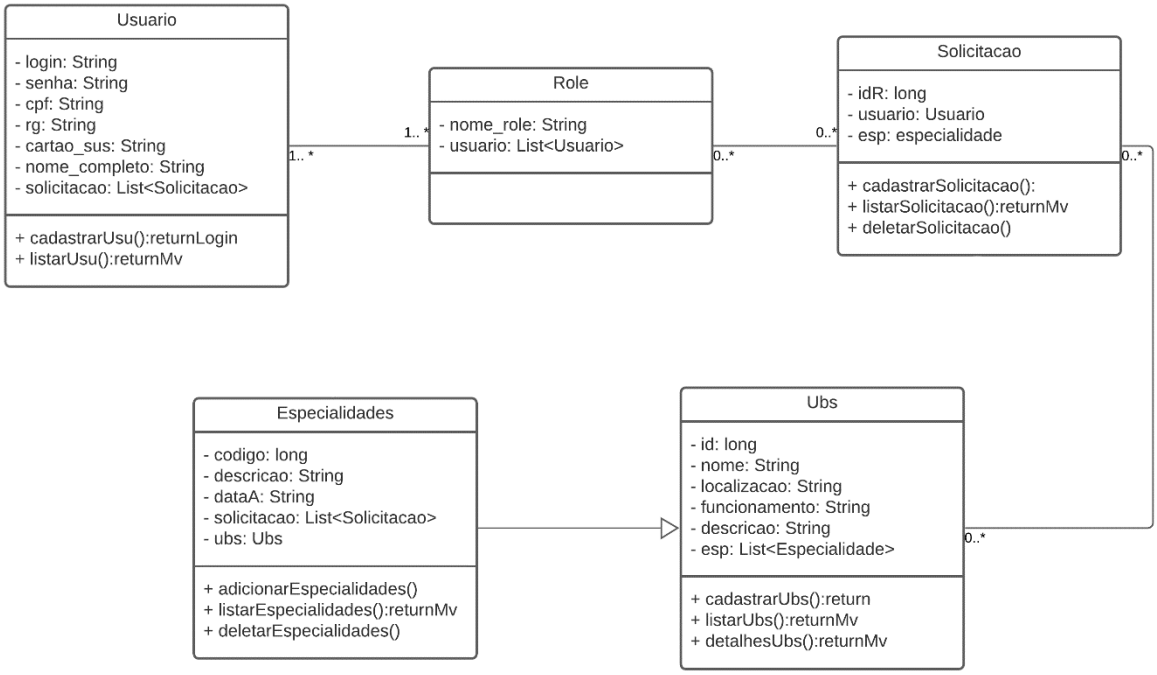

Figura 7: diagrama de classes.

No diagrama de classes do projeto Mais Saúde há a presença de 5 (cinco) classes modelo: Usuario, Role, Solicitacao, Especialidades e Ubs. A comunicação e relação entre elas é representada por linhas e setas. A ligada por seta, significa que é uma classe herdeira da qual a seta aponta. Já as outras ligações, são de associação, que é um tipo de relacionamento, entre vários que existem.

5.3 IMPLEMENTAÇÃO DA APLICAÇÃO

O Mais Saúde é um projeto que visa solucionar os problemas de logística das Unidades Básicas de Saúde (UBS). Com ele, as UBS dão um grande passo para o futuro, pois passam a contar com um programa que melhora a organização das fichas de atendimento e que otimizam esse processo, tornando-o mais rápido. Além disso, o Mais Saúde também pode ser usado para obter mais informações sobre as UBS existentes.

O projeto nasceu por consequência de uma dificuldade real e muito conhecida por toda a população lajense. Dificuldade essa que acontecia frequentemente em todas as unidades de saúde. Por falta de automatização do processo, era comum pessoas acordarem às 03h00min da madrugada para irem ao os postos e esperarem horas para conseguir uma ficha. Por via das vezes, havia confusões relacionadas com a ordem de chegada e com a pouca quantidade de fichas disponibilizadas.

Assim, com base em outros aplicativos, levantamos alguns requisitos funcionais e não funcionais que seriam importantes para guiar-nos na construção do projeto. O Mais Saúde deve oferecer a possibilidade de os usuários acessarem o sistema fazendo login, com nome de usuário e senha. Além disso, cadastrar-se no sistema e solicitar ficha também são requisitos. Para o agente, a aplicação deve ter o gerenciamento das unidades e o gerenciamento da distribuição de fichas.

De acordo com o diagrama de casos de uso, temos três usuários que possuem funcionalidades diferentes. O Avulso pode “Fazer Cadastro” e “Ver UBS”, o Paciente pode “Fazer Login”, “Solicitar Ficha” e “Ver UBS”. E por fim, o Agente pode “Ver Solicitações Feitas”, “Editar UBS”, “Ver Lista de Pacientes Cadastrados”, “Verificar Especialidades”, “Adicionar ou Excluir Especialidades”, “Deferir Ficha” e “Indeferir Ficha”.

5.3.1 Funcionamento da Aplicação

Fonte: autoria própria, 2020.

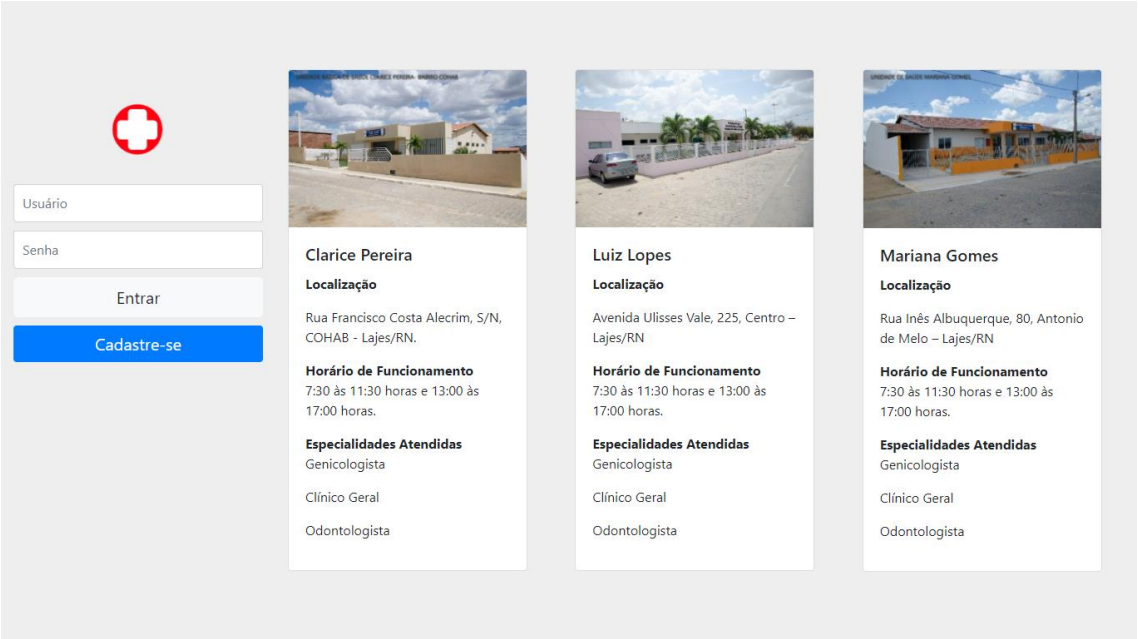

Figura 8: página inicial (avulso).

Esta é a página inicial da aplicação. É nela que os usuários poderão se cadastrar, fazer login e consultar informações das unidades básicas de saúde. Essa tela, assim como todas, foi diagramada por nós, componentes, e depois foi desenvolvida com o auxílio do framework Bootstrap. Aqui foi utilizada a ferramenta de cards, que são muito usados para passar informações. Nesse caso, informações de localização, horário de funcionamento e especialidades atendidas nos postos de saúde da cidade.

Quem ainda não é usuário, deverá clicar em “Cadastre-se” para ser redirecionado a página de cadastro. Quem já for cadastrado, basta digitar o nome de usuário, senha e clicar em “Entrar”.

A página de início demonstra uma clara identidade visual que será levada por todo o site. Com uma paleta de cores simples e neutras, com poucas informações e bem divididas, o Mais Saúde carrega consigo uma fácil usabilidade, uma boa comunicação visual e um *design* que foca no minimalismo.

Fonte: autoria própria, 2020.

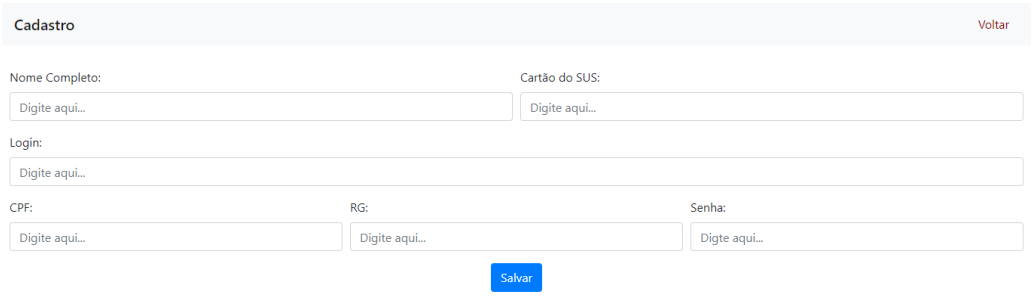

Figura 9: página de cadastro (avulso).

Esta é a tela de cadastro de usuário. É aqui que os visitantes poderão se cadastrar no sistema e assim liberar o acesso a funcionalidade de solicitação de fichas. São pedidos alguns dados, como nome completo, número do cartão do SUS, CPF e RG. Além desses, o visitante deverá criar um nome de usuário e digitar sua senha. Essa é a segunda tela para quem não é um usuário cadastrado. Assim que clicar em “Salvar”, o visitante, agora paciente, será redirecionado à página inicial.

Fonte: autoria própria, 2020.

Figura 10: menu da aplicação (módulos paciente e agente).

A tela de menu hospeda todas as páginas que cada tipo de usuário pode acessar. O paciente pode acessar apenas duas páginas, a de “Solicitação” e a de “Ver UBS”. Já o agente pode acessar as seguintes páginas: “Solicitações” (onde poderá ver as solicitações feitas pelo o paciente), “Pacientes Cadastrados”, “Cadastrar UBS”, “Editar UBS”, “Editar Especialidades” que só poderá ser acessada através de “Editar UBS”.

Fonte: autoria própria, 2020.

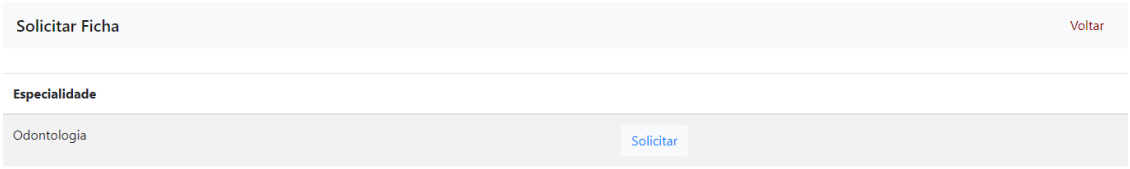

Figura 11: página de solicitação (módulo paciente).

Esta é página de Solicitar Ficha, como o próprio nome já diz, aqui o paciente poderá solicitar sua ficha para a especialidade disponível. De um lado, atua o agente que irá adicionar ou excluir as especialidades. Do outro, o paciente que utiliza o sistema para fazer a solicitação de sua ficha. Para facilitar, essa página faz parte do módulo paciente, portanto, o agente não tem acesso a ela.

Fonte: autoria própria, 2020.

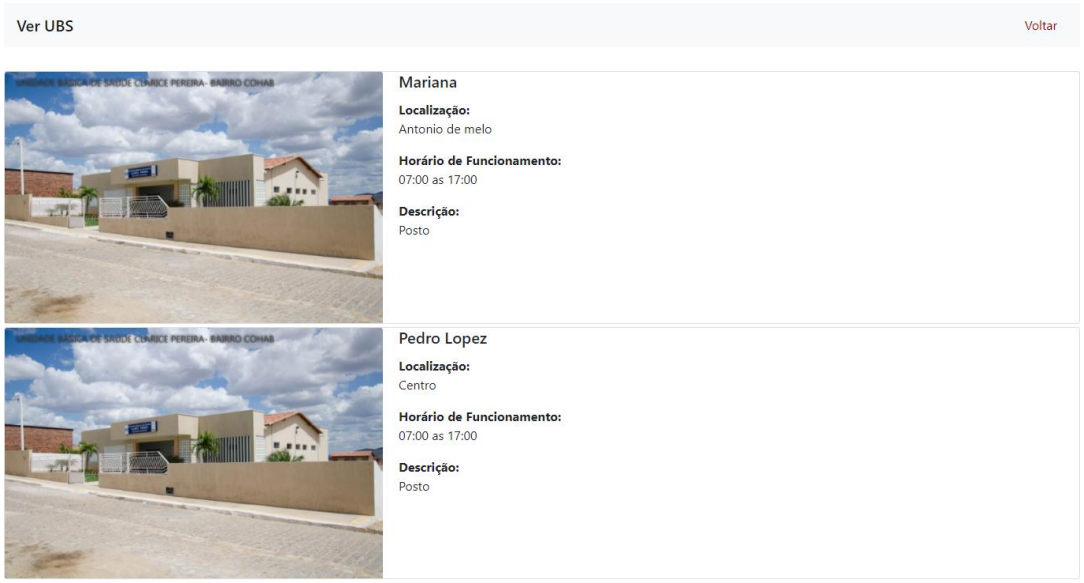

Figura 12: informações das UBS (módulo paciente).

Aqui o paciente (usuário cadastrado no sistema) poderá consultar algumas informações muito importantes, tais como, localização, horário de funcionamento e a descrição do posto de saúde. Nessa página também foram usados *cards* que são componentes disponibilizados pelo o *framework front-end*, Bootstrap 4. Essa página também faz parte do módulo paciente, ou seja, o agente não pode acessar.

Fonte: autoria própria, 2020.

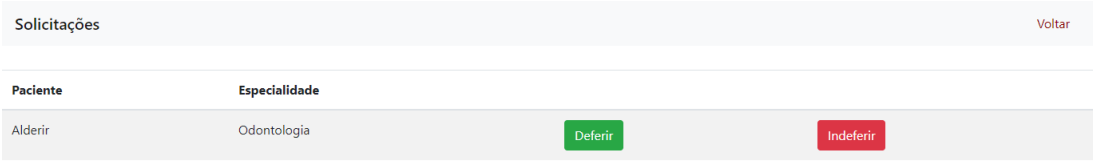

Figura 13: tela de solicitações (módulo agente).

Esta é a tela onde chegam todas as solicitações feitas pelos pacientes. É aqui que o agente de saúde irá deferir ou indeferir o pedido das fichas solicitadas. Como podemos ver, o Mais Saúde também consegue comunicar-se visualmente, isso acontece porque utilizamos cores que já têm um significado implícito para as pessoas. Sendo assim, mesmo que a pessoa não leia, ela vai saber que o verde é para deferir a ficha e que o vermelho é para indeferir.

Fonte: autoria própria, 2020.

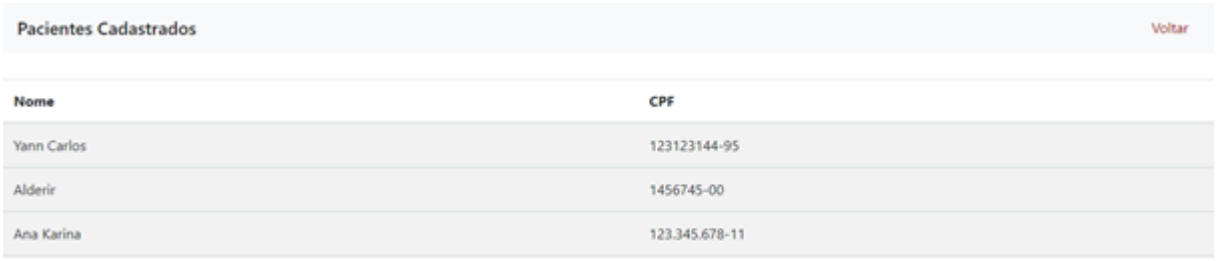

Figura 14: tela de pacientes cadastrados (módulo agente).

Esta tela lista todos os pacientes cadastrados, com suas características principais (nome e CPF). É onde também o agente de saúde poderá ver todos os usuários que existem no sistema.

Fonte: autoria própria, 2020.

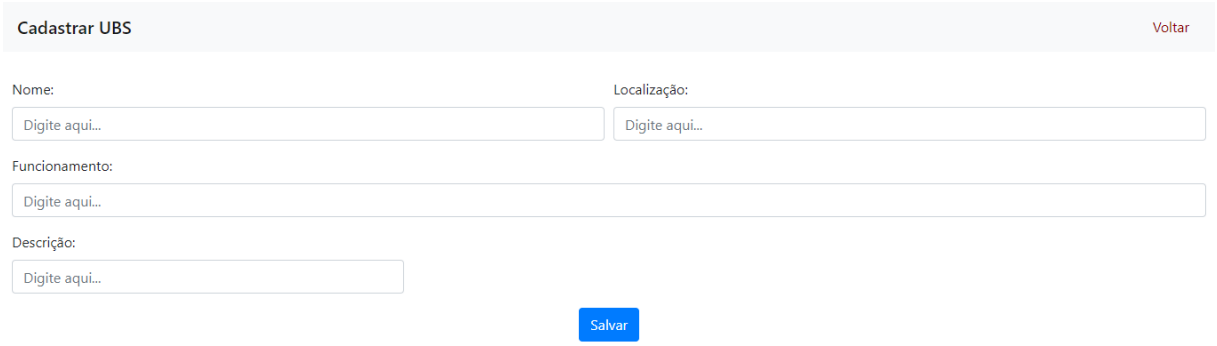

Figura 15: tela de cadastro de UBS (módulo agente).

Esta é a tela onde o agente de saúde vai cadastrar as UBS presentes na cidade. Para isso, ele usará as informações básicas que a definem, como nome, localização, horário de funcionamento e alguma descrição. Assim que todas as características forem digitadas, o agente deverá clicar no botão azul para salvar. Como podemos perceber, o botão tem a mesma cor do botão “cadastre-se” na página inicial. Essa cor, por sua vez, é mais uma forma de comunicação visual, onde todos os botões de cadastrar ou de salvar alguma informação será na cor azul.

Fonte: autoria própria, 2020.

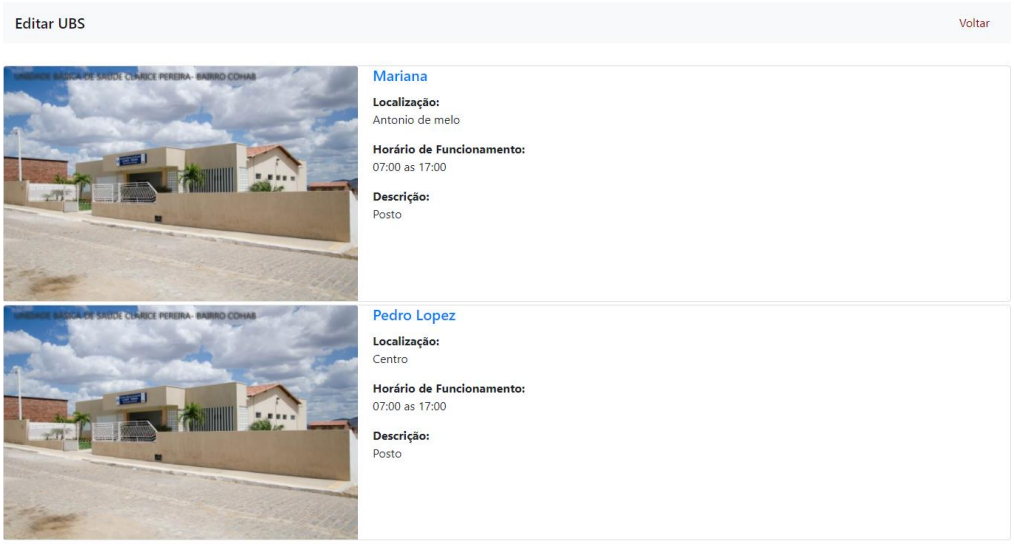

Figura 16: tela de edição das UBS (módulo agente).

Esta é a tela onde o agente de saúde vai poder editar as especialidades das UBS, seja por erro ou por ter sido alterado no decorrer do tempo. A diferença dessa tela para a outra que foi exposta mais acima é que aqui o agente poderá detalhar a UBS e assim que detalhada, as especialidades irão aparecer em listas. Perceba que os nomes das Unidades Básica de Saúde estão azuis, isso porque, agora, os nomes estão como *hiperlinks*. Basicamente, eles estão funcionando como um botão que nos redireciona para outra tela.

Fonte: autoria própria, 2020.

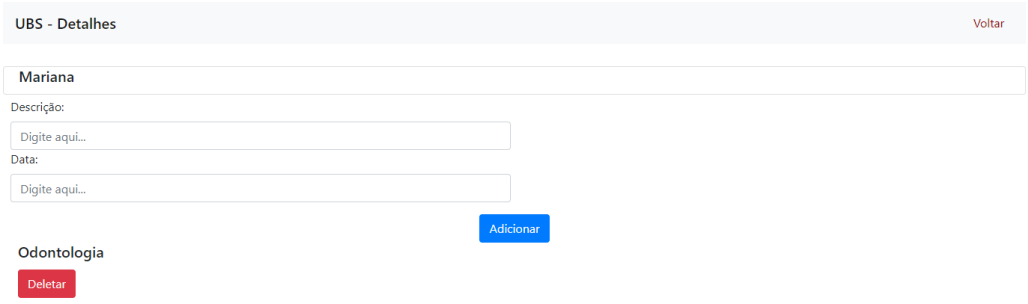

Figura 17: tela de detalhes das UBS (módulo agente).

Na tela “Detalhes”, serão adicionadas ou excluídas as especialidades presentes em cada UBS, com a descrição (que pode ser o nome) e a data prevista para o próximo atendimento. Aqui poderá ser adicionada quantas especialidades precisar. Caso as fichas para aquela área sejam esgotadas, o agente poderá apagar apenas clicando no botão vermelho. Caso queira mais áreas, é necessário digitar os dados e clicar no botão azul.

5.3.2 Implementação do Caso de Uso

Contextualizando, o projeto de extensão Mais Saúde é desenvolvido por três discentes, onde a implementação do programa, como a escrita deste relatório é fragmentada entre os membros do grupo. Sendo assim, de acordo com o diagrama de casos de uso, apresentado nos subtópicos anteriores, eu fiquei responsável por desenvolver os seguintes casos de uso: Fazer *Login*, Solicitar Ficha, Adicionar Especialidades, Excluir Especialidades, Deferir Ficha e Indeferir Ficha. Dos casos de uso desenvolvidos por mim, irei detalhar e exemplificar o funcionamento do “Solicitar Ficha”:

O caso de uso ou a funcionalidade “Solicitar Ficha” é uma das principais funções do sistema Mais Saúde e é a única que tem uma comunicação direta com outro módulo. Como já citado, o sistema possui dois módulos, paciente e agente. Sendo assim, a funcionalidade de solicitação está disponível apenas para o paciente, que se comunica diretamente com o agente.

Tendo isso em vista, entenda agora passo a passo como a funcionalidade age:

Já logado no sistema, o usuário paciente selecionará a página de solicitação de ficha, ele poderá fazer essa seleção a partir da página de “Menu”, onde todos os *links* das funções do sistema estão hospedados. Esses *links*, são ativados através da propriedade “*href*” que são pertencentes a tag *<a>* da linguagem de marcação de texto, *HTML*. Assim que selecionar, a página “Solicitar Ficha” será carregada e ele será redirecionado.

Na tela “Solicitar Ficha”, terá uma lista com todas as opções de atendimento disponíveis à solicitação, que foram cadastradas por um agente no módulo correspondente a ele. Para pedir a ficha, o usuário deverá clicar no botão “Solicitar” na opção em que ele desejar atendimento. Assim, uma requisição será enviada à página “Fichas Solicitadas” no módulo agente. Após isso, o paciente deverá aguardar sua ficha ser deferida ou indeferida.

Já no módulo do administrador, o agente deverá selecionar a página “Fichas Solicitadas” que também está no “Menu” através de *links*. Assim que selecionada, a página será carregada e aberta. Essa página, assim como “Solicitar Ficha”, também terá uma lista, mas dessa vez, essa lista terá todas as solicitações feitas por pacientes. Sendo assim, em cada pedido de atendimento, há dois botões, um “Deferir” e outro “Indeferir”.

Caso queira liberar uma ficha, o agente deverá clicar no botão de cor verde, escrito “Deferir”. Assim que clicar, a requisição que foi iniciada pelo o paciente em seu módulo, será finalmente concluída. Desse jeito, o processo de solicitação de uma ficha estará completo. Caso não queira liberar, o agente clicará no botão de cor vermelha, escrito “Indeferir”. Quando clicar, a requisição que foi enviada pelo paciente será apagada.

Para uma melhor compreensão do caso de uso “Solicitar Ficha”, veja o diagrama de sequência abaixo:

Fonte: autoria própria, 2021.

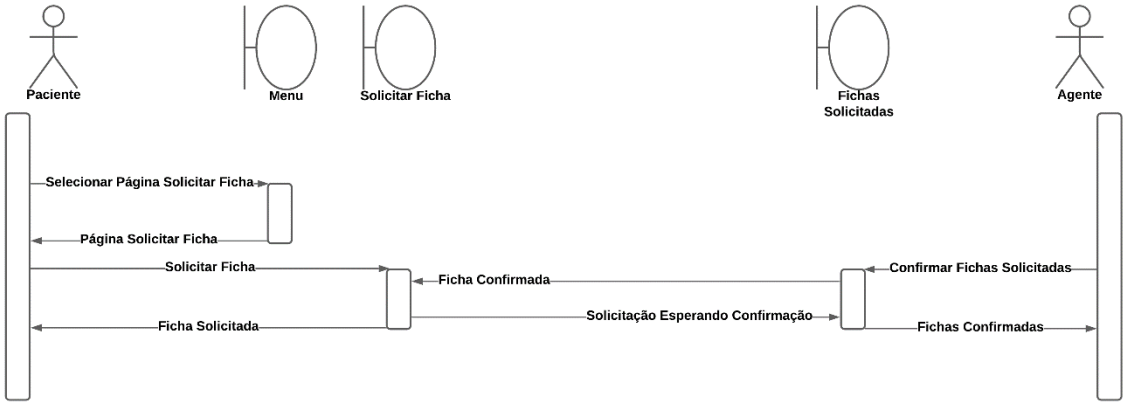

Figura 18: diagrama de sequência da funcionalidade solicitar ficha.

Com base no que foi explicado e no diagrama de sequência, podemos enfim partir para a implementação do caso de uso “Solicitar Ficha”. Mas, para melhor entendermos o processo, dividiremos em três partes. A primeira parte falará dos métodos usados na página “Solicitar Ficha”, a segunda parte falará como se dá a comunicação entre os módulos; e a terceira e última parte, tratará sobre os métodos da página “Fichas Solicitadas”.

Fonte: autoria própria, 2021.

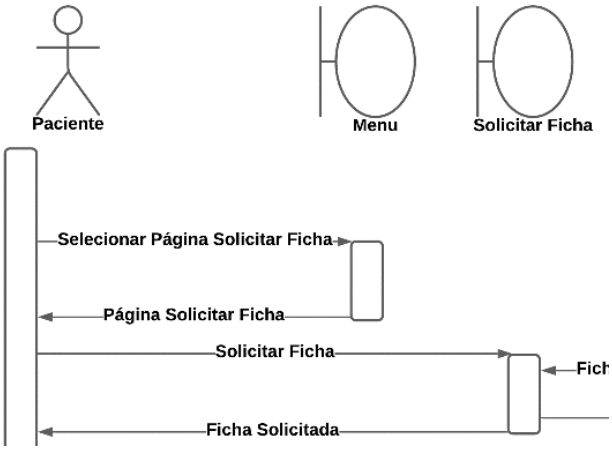

Figura 19: primeira parte.

Fonte: autoria própria, 2021.

```
//// ESPECIALIDADES ////

    @RequestMapping("/solicitar")
    public ModelAndView listaespecialidades() {
        ModelAndView mv = new ModelAndView("web/solicitarficha");
        Iterable<Especialidades> espe = esp.findAll();
        mv.addObject("espe", espe);
        return mv;
    }
```

Figura 20: método /solicitar no *controller* Ubs.java.

De acordo com a figura 19, a primeira parte do diagrama, o paciente selecionará a página para solicitação diretamente do menu, assim que acessar, ele verá uma lista de especialidades disponíveis e poderá solicitar sua ficha. Mas, para que isso aconteça precisamos criar um método que faça esse trabalho de listagem.

Como podemos ver, o método com nome “listaespecialidades” possui uma abordagem *ModelAndView*, que serve para retornar informações de modelo e visualização. Veja também que o parâmetro da nova instância do objeto é a página *HTML* “solicitarficha” que está dentro da pasta “*web*” e que será renderizada. Para a realização de um método de listagem, geralmente precisamos utilizar o *Iterable*, pois é com ele que conseguiremos criar uma lista de Especialidades. Isso, apenas será válido com ajuda do *repository* que possui a função “*findAll*” para a busca dessa lista.

Para podermos passar essa lista de especialidades para a página *web*, precisaremos utilizar o objeto “mv” que foi criado nas linhas anteriores. O objeto junto com a propriedade *addObject*, fará com que a página seja renderizada de acordo com os dados das especialidades que foram adicionadas. Por fim, é preciso criar uma requisição utilizando a anotação *@RequestMapping*. Como parâmetro dessa

anotação, colocaremos a *url* “/solicitar”, que será utilizada para quando quisermos acessar a página “Solicitar Ficha”. É essa *url* que utilizamos na página “Menu”, para fazer com que ela seja aberta com apenas um clique.

Fonte: autoria própria, 2021.

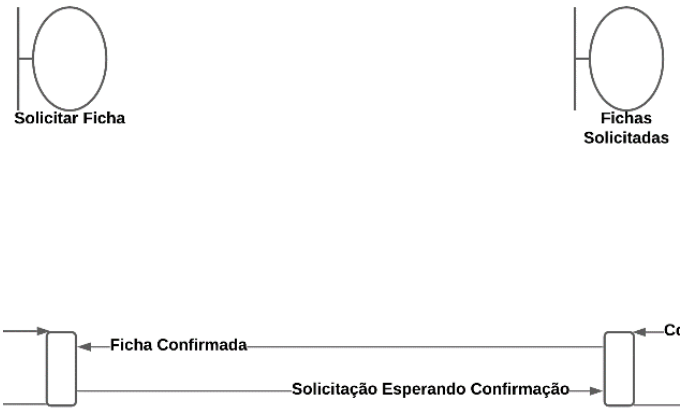

Figura 21: segunda parte.

Ainda na página “Solicitar Ficha”, mas já na segunda parte, teremos uma lista de especialidades disponíveis, onde cada item dessa lista terá um botão “Solicitar”. Para podermos realizar essa solicitação apenas clicando nesse botão, teremos que criar uma classe de controle que consiga coletar os dados do usuário, pois na hora em que a ficha for solicitada, os dados como nome de usuário e qual especialidade ele escolheu terá que ser enviada para a página “Fichas Solicitadas” no módulo do agente para ser avaliada.

Essa dinâmica poderia ser feita através de um *repository,* mas não é viável ao nosso caso, pois o *repository* envia apenas um dado por vez e queremos enviar dois dados, o nome de usuário e a especialidade escolhida. Dessa forma, a classe de controle importará a classe modelo “Usuario”, as *interfaces* “*Authentication*” e “*SecurityContext*”. Além das classes “*SecurityContextHolder*” e “*User*” que é associada a *interface* “*UserDetailsSevice*”.

Fonte: autoria própria, 2021.

```
public UsuarioController(){
    usuario = new Usuario();
    SecurityContext context = SecurityContextHolder.getContext();
    if(context instanceof SecurityContext)
    {
        Authentication authentication = context.getAuthentication();
        if(authentication instanceof Authentication)
        {
             usuario.setLogin(((User)authentication.getPrincipal()).getUsername());
        }
    }

}

public Usuario getUsuario() {
    return usuario;
}

public void setUsuario(Usuario usuario) {
    this.usuario = usuario;
}
```

Figura 22: classe de controle Usuario*Controller*.java.

Como já dito, ela servirá para mandar as informações dos usuários que solicitarem uma ficha. As informações que serão coletadas irão servir apenas para identificação do solicitante, não sendo possível ver outras informações, com exceção do nome de usuário que é exatamente a única que está sendo coletada no código.

Fonte: autoria própria, 2021.

```
//// SOLICITACAO ////

    @RequestMapping(value="/solicitacao", method=RequestMethod.GET)
    public String cadsolicitacao(long codigo) {

        Solicitacao solicitacao = new Solicitacao();

        UsuarioController uc = new UsuarioController();

        String usu = uc.getUsuario().getLogin();
        Usuario usuario = ur.findByLogin(usu);

        solicitacao.setUsuario(usuario);

        Especialidades espe = esp.findByCodigo(codigo);
        solicitacao.setEsp(espe);

        sol.save(solicitacao);
        return "redirect:/solicitar";
    }
```

Figura 23: método /solicitação no *controller* Ubs.java.

Este é o método “cadsolicitacao”, ele serve para cadastrar as solicitações de fichas feitas por pacientes. Para isso, ele utiliza a classe de controle “UsuarioController” e as classes modelo “Solicitacao” e “Especialidade” que possuem um relacionamento “@ManyToOne”. Ele também obtém os dados do usuário e das especialidades que estão salvos no banco, como *login* e código de cada especialidade. No final, ele salva o pedido em solicitação e redireciona à *url* “/solicitar”, que corresponde a página “Solicitar Ficha”.

Fonte: autoria própria, 2021.

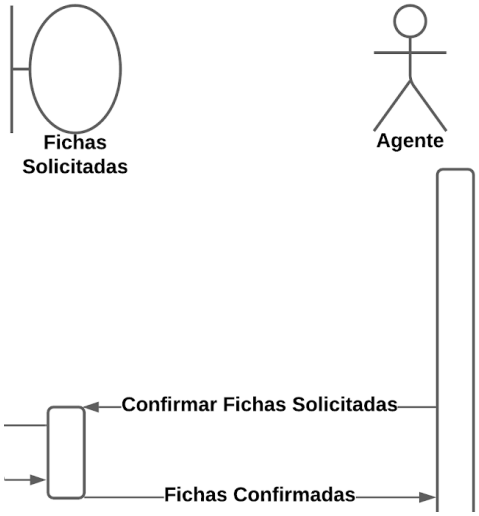

Figura 24: terceira parte.

Fonte: autoria própria, 2021.

```
@RequestMapping("/indeferi")
public String deletarSol(long idR){
    Solicitacao solicitacao = sol.findByIdR(idR);
    sol.delete(solicitacao);

    return "redirect:/solicitacaoList";
}
```

Figura 25: método indeferir ficha.

Este é o último método da última parte do caso de uso “Solicitar Ficha”. Ele configura a função de deletar e é correspondente ao botão indeferir da página “Fichas Solicitadas”. Para sua implementação: precisamos fazer a instância da classe modelo “Solicitacao”, onde, logo em seguida será buscado no banco, através de seu *repository*. Depois, será usado o método “*delete*” para excluir a solicitação, novamente, utilizando o *repository*, que funciona como uma porta para o banco. Por fim, retornamos a página “Fichas Solicitadas” para que a lista de fichas seja atualizada.

Para uma melhor compreensão de todo o sistema, acesse: https://github.com/carlosyann/maisaude

6 ANÁLISE E DISCUSSÃO DOS RESULTADOS

A partir de estudos realizados nos postos de saúde do município, em relação a dificuldade da população para obter uma ficha, foi proposto o desenvolvimento de um sistema que pudesse otimizar, digitalizar e democratizar esse acesso dos cidadãos, usuários do sistema de saúde, às fichas de atendimento. Diariamente são distribuídas manualmente apenas 20 (vinte) fichas nas Unidades Básicas de Saúde (UBS), uma pequena quantidade para uma grande demanda.

Essa insuficiência acaba acarretando alguns problemas como, filas grandes, alto tempo de espera, deslocamento desnecessário de pessoas, problemas com ordem de chegada, problemas de falta de atendimento, etc. Com a adoção do sistema *web* Mais Saúde, a maioria desses problemas serão sanados, pois o processo de distribuição de ficha será otimizado. Ficará mais cômodo para a população, pois não será necessário sair de casa de madrugada, expondo-se a perigos como antes.

Posto isto, após o desenvolvimento do sistema *web*, verificamos que os requisitos funcionais e os requisitos não-funcionais, foram efetivados com sucesso. O sistema web, enfim, pode receber novos usuários e dar-lhes o acesso a solicitação e as informações necessárias a ele. Ademais, foi alcançada a possibilidade de separar o sistema *web* em módulos, já que teríamos mais de um tipo de usuário acessando, ao mesmo tempo, o Mais Saúde.

Desde o começo, tentamos seguir uma proposta visual, que fosse bonita, mas também que fosse simples, de fácil usabilidade, fácil entendimento, que fosse confortável aos olhos e que tivesse páginas *web* expansivas. Com isso, resolvemos utilizar o *framework front-end* Bootstrap, que foi de grande importância para alcançarmos esse objetivo. Graças a ele, conseguimos resolver alguns problemas de criatividade que estavam como um empecilho anteriormente.

Com a mesma importância, o *Spring Boot* foi essencial para a implementação do sistema *web* Mais Saúde. Com ele, conseguimos, não apenas criar as classes modelo, os *controllers* e as *views*, bem como toda arquitetura *MVC*. Mas também conseguimos realmente entender todos os processos que aconteciam com as classes, sejam elas de controle, configuração ou de modelo.

Dessa forma, concluiremos com imagens que expõem o funcionamento do sistema fragmentado, bem como, o seu *design* minimalista e algumas de suas funções realizadas individualmente. É válido ressaltar que, por estar pronto, ele já pode ser

usado. Já que não apresenta nenhum problema estrutural, seja por códigos errados ou por falta de conexão com as ferramentas utilizadas.

7 CONSIDERAÇÕES FINAIS

Durante a realização do sistema *web* Mais Saúde, assumimos o desafio de buscar o envolvimento de profissionais da área em diferentes níveis hierárquicos, todos participando de maneira direta ou indireta. É esperado que o sistema *web* Mais Saúde minimize as dificuldades enfrentadas diariamente por todos os cidadãos que utilizam do Sistema Único de Saúde (SUS) na cidade de Lajes. A atividade em questão é de suma importância para desenvolvimento tecnológico da saúde lajense e para o bem-estar da população. Acreditamos que a implantação do nosso sistema *web* irá melhorar a qualidade de vida das pessoas, diminuir o trabalho nas Unidades Básicas de Saúde e além de tudo, diminuir a quantidade de papel gasto com a fabricação de fichas.

REFERÊNCIAS

IBGE. IBGE Cidades. Disponível em: <https://cidades.ibge.gov.br/brasil/rn/lajes/panorama>. Acesso em: 09 jun. 2019.

PREFEITURA DE LAJES. Prefeitura de Lajes - Saúde. Disponível em: <https://lajes.rn.gov.br/saude/>. Acesso em: 09 jun. 2019.

IBM. Documento de Visão. Disponível em: https://www.ibm.com/support/knowledgecenter/pt-br/SSWMEQ_4.0.6/com.ibm.rational.rrm.help.doc/topics/r_vision_doc.html. Acesso em: 9 jun. 2019.

SCRIPTCASE. Desenvolvimento WEB. Disponível em: http://www.scriptcase.com.br/desenvolvimento-web/. Acesso em: 12 jun. 2019.

SILVA, Maurício Samy. Criando Sites Com HTML: Sites de Alta Qualidade com HTML e CSS. 1. ed. São Paulo: Novatec Editora, 2008. p. 26-26.

KINGLY STUDIO. Mas, o que é desenvolvimento web mesmo?. Disponível em: <http://www.kinglystudio.com.br/descomplica/mas-o-que-e-desenvolvimento-web-mesmo/>. Acesso em: 09 jun. 2019.

TABLELESS. O básico: o que é HTML?. Disponível em: <https://tableless.com.br/o-que-html-basico/>. Acesso em: 09 jun. 2019.

HOSTINGER BLOG. O que é CSS? guia básico para iniciantes. Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/>. Acesso em: 09 jun. 2019.

MILANI, André. MySQL: Guia do Programador. 1. ed. São Paulo: Novatec Editora, 2007. p. 22-22.

SPRING. Why Spring?. Disponível em: https://spring.io/why-spring. Acesso em: 11 jun. 2019.

ECLIPSE FOUNDATION. About Us. Disponível em: https://www.eclipse.org/org/. Acesso em: 2 nov. 2020.

DEVMEDIA. Conhecendo o Eclipse - Uma apresentação detalhada da IDE. Disponível em: https://www.devmedia.com.br/conhecendo-o-eclipse-uma-apresentacao-detalhada-da-ide/25589. Acesso em: 2 nov. 2020.

BOOTSTRAP. Sobre o BOOTSTRAP. Disponível em: https://getbootstrap.com.br/docs/4.1/about/overview/. Acesso em: 12 jun. 2019.

ANEXO A – FORMULÁRIO DE IDENTIFICAÇÃO

Dados do Relatório Científico	
Título e subtítulo: **MAIS SAÚDE:** PLATAFORMA MÓVEL DOS SERVIÇOS DIGITAIS DAS UNIDADES BÁSICAS DE SAÚDE LOCAIS	
Tipo de relatório: Projeto de Extensão	Data: 11/06/2019
Título do projeto/ programa/ plano: Mais Saúde	
Autor(es): Alderir Anselmo da Silva, Ana Karina Silva Albuquerque e Yann Carlos Silva de Morais.	
Instituição e endereço completo: Instituto Federal de Educação, Ciência e Tecnologia. BR-304, Km 120, s/n - Centro, Lajes - RN, 59535-000.	
Resumo: A Secretaria Municipal de Saúde é o órgão responsável pelo planejamento, organização, coordenação e execução dos programas, projetos e atividades voltadas para a implantação das políticas de saúde do Município de Lajes/RN. Ao total, existem 5 (cinco) unidades básicas de saúde no município que oferecem atendimento à população baseado na distribuição de fichas impressas e cedidas por ordem de chegada ao local de distribuição. Isto por vezes causa conflito, em relação a horários, entre os habitantes da região. Com base nessas informações, construímos este projeto a fim de propor um aplicativo para facilitar o acesso à informação e promover um *software* de atendimento de saúde a população do município de Lajes/RN.	
Palavras-chave/descritores: Aplicativo. Postos de Saúde. Cidadãos de Lajes/RN.	
Nº de páginas: 49	
Jornada de trabalho: 5 dias	Horas semanais: 15h
Total de horas: 340 horas	
Observações/notas:	