INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Campus São João da Boa Vista

Trabalho Final de Curso

4º ano – Curso Técnico em Informática

Prof. Breno Lisi Romano e Prof. Luiz Angelo Valota Francisco

PROCESSO DE DESENVOLVIMENTO DAS ITERAÇÕES DO MÓDULO 2

Aluno: Pedro Henrique da Costa Abreu

Prontuário: 1620363

São João da Boa Vista – SP

2019

Resumo

Este documento tem o papel de descrever o processo de desenvolvimento das funcionalidades do módulo 2 do projeto Gerações, deste o levantamento de requisitos e elaboração de protótipos até o desenvolvimento propriamente dito, introduzindo conceitos básicos e apresentando o método adotado de desenvolvimento de software, apontando as dificuldades e aprendizados ao longo do processo de elaboração do projeto.

Sumário

Índice de ilustrações

Índice de Tabelas

1 Introdução

1.1 Contextualização/motivação

Com o propósito inicial de prover educação aos "deserdados da fortuna e menores marginalizados", nasceu a Escola de Aprendizes e Artífices de São Paulo em 1910 [1], que deu origem ao CEFET-SP (Centro Federal de Educação Tecnológica) e só em 2008, com a lei federal nº 11.892, 31 CEFETs, 75 unidades descentralizadas de ensino (UNEDs), 39 escolas agrotécnicas, 7 escolas técnicas federais e 8 escolas vinculadas a universidades deixaram de existir para receber o atual nome, Instituto Federal de Educação, Ciência e Tecnologia.

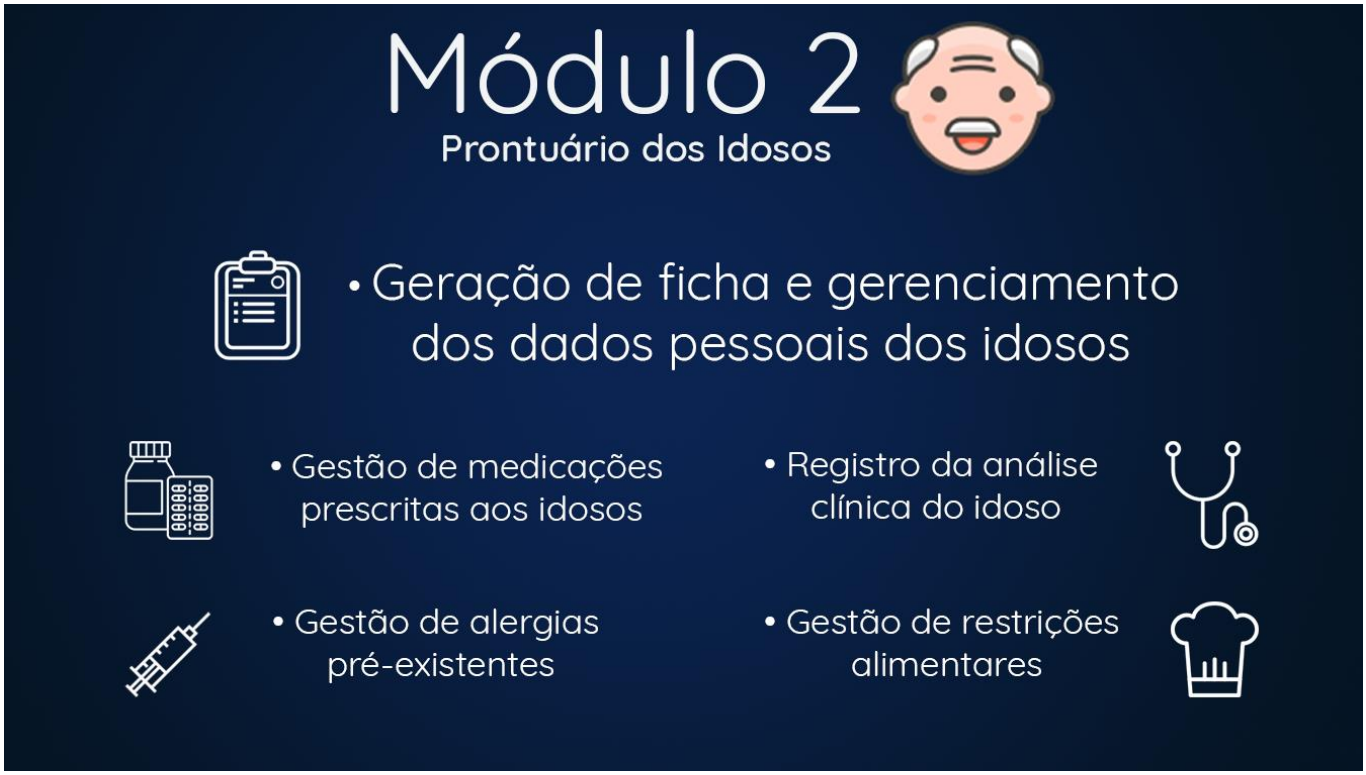
O Campus de São João da Boa Vista teve sua estreia em 2007, ainda sob o título de CEFET e com somente dois cursos técnicos disponíveis, hoje a unidade não se restringe aos cursos técnicos, disponibilizando também cursos de tecnologias, engenharias, licenciaturas, bacharelados e pós-graduações, fornecendo assim altíssima qualidade de ensino que resulta em mão de obra qualificada para toda região.

O munícipio de São João da Boa Vista tem uma população estima pelo IBGE de 91.211 habitantes (2019) [2], sendo mais de 8.700 com mais de 60 anos de acordo com o censo de 2010 [3], e é colocada como a melhor cidade do país para idosos [4], motivados por esses fatos, os alunos do 4º ano do curso integrado de informática do IFSP foram encorajados a desenvolver um sistema que suprisse as necessidades das casas de longa permanência oferecendo um *software* que permitisse o total gerenciamento da instituição, além de permitir os familiares acompanharem seus parentes fornecendo dados sobre seus medicamentos, patologias, entre outros. O projeto busca integrar os alunos e incentivar o trabalho em equipe, os preparando para mercado de trabalho, pré-estabelecendo prazos para a entrega das atividades e assim promovendo um senso de responsabilidade ao aluno.

Durante o desenvolvimento desse sistema os alunos foram divididos em módulos com objetivos e funções diferentes e dentro de cada módulo cada aluno possuía sua função especifica, analistas, responsáveis pelas atividades de documentação, análise e testes de protótipos de desenvolvimento; DBAs, responsável pela criação, gerenciamento e integração do banco de dados do projeto e desenvolvedores, responsáveis pelo desenvolvimento de protótipos, *templates* e iterações do sistema.

O módulo 2 ficou responsável pelo cadastro de idosos e gerenciamento de seus dados pessoais.

Figura 1 - perspectiva do módulo 2



1.2 Objetivo Geral

O objetivo geral deste trabalho compreende a tudo que se diz respeito sobre o desenvolvimento das iterações do módulo 2, desde o desenvolvimento inicial de protótipos até o das iterações em si, apresentando etapas, processos, adaptações e dificuldades nessa importante parcela do sistema.

1.3 Objetivos específicos

Os objetivos específicos versados neste trabalho são:

- Desenvolvimento das iterações;
- Planejamento das iterações;
- Dificuldades na refatoração do código;
- Adaptações necessárias e a reutilização de código durante o desenvolvimento.

2 Desenvolvimento

2.1 – Levantamento Bibliográfico

2.1.1 Breve História das Linguagens de Programação

Na década de 1940, com a invenção dos primeiros computadores, surgiu a necessidade de um meio de comunicação e expressão que promovesse o contato entre máquina e homem, é a partir desta época que surgem as primeiras linguagens de programação como a linguagem de máquina e a linguagem *assembly*. [5]

Com o passar dos anos a crescente facilidade ao acesso a computadores somado ao cada vez mais amplo uso da computação nas mais diversas áreas de conhecimento fez surgir a demanda de linguagens que se adequassem a suas necessidades, tal demanda incentivou a criação de cada vez mais linguagens de programação que satisfariam a carência de diversos profissionais, seja no desenvolvimento de um *website*, *videogame* ou na realização de cálculos complexos.

Figura 5 - Exemplo de linguagens de programação muito utilizadas

Cada linguagem exige conhecimento especifico para seu uso adequado, possuindo sintaxe, semântica e vocabulário próprio, além de possuir diversas maneiras de se organizarem, podendo ser orientadas a objeto, estruturadas, modular ou lineares.

2.1.2 Orientação a Objetos

A Programação Orientada a Objetos (POO) é um padrão de desenvolvimento de *softwares* amplamente utilizado em muitas linguagens de programação atuais, como Java, C#, PHP, Python, C++, entre outras.

Nesta forma de estruturação de programação, são criados objetos com atributos e comportamentos próprios. Tais objetos interagem entre si para assim promover o resultado esperado.

Nesta seção iremos introduzir conceitos básicos da POO.

2.1.2.1 Classe

Classe é, de certa forma, um conjunto de objetos similares, ela define os dados (atributos) de um objeto e seus comportamentos (métodos), formando um arquétipo a ser seguido, por exemplo, se fossemos criar uma classe "Pessoa" ela teria os atributos: nome, idade, CPF, RG, endereço, etc. e os métodos: andar(), comer(), dirigir(), etc.

2.1.2.2 Objeto

Os objetos são as entidades por trás de todo sistema através de sistemas estruturados a partir de orientação a objetos, eles são instâncias de classes, por exemplo, utilizando a classe "Pessoa" que utilizamos no capítulo anterior podemos criar o objeto "João" e especificar seus atributos: nome: João da Silva, idade: 73, CPF:XXX.XXX.XXX-XX e assim sucessivamente.

2.1.2.3 Encapsulamento

O encapsulamento surge com o propósito de segurança, escondendo os atributos privados da classe em questão, permitindo sua utilização através dos métodos especiais *get e set* [7].

2.1.2.4 Abstração

Abstração é o ato de transformar uma entidade do mundo real em um objeto de nosso sistema, desconsiderando os atributos menos importantes, tornando assim o objeto menos complexo e mais prático.

2.1.2.5 Herança

Assim como no mundo real, o conceito de herança vem de herdar, onde uma classe-filha (subclasse) herda atributos e métodos de uma classe-pai (superclasse), para isso utilizamos a palavra reservada *extends* ao criar a subclasse, por exemplo: `public class Gerente extends Funcionario,` a partir disto criamos uma classe Gerente que além de seus atributos e métodos

próprios irá herdar os atributos e métodos comuns a todo funcionário. Tal funcionalidade é extremamente útil quando pensamos na reutilização de código, uma das principais vantagens da programação orientada a objetos.

2.1.2.6 Polimorfismo

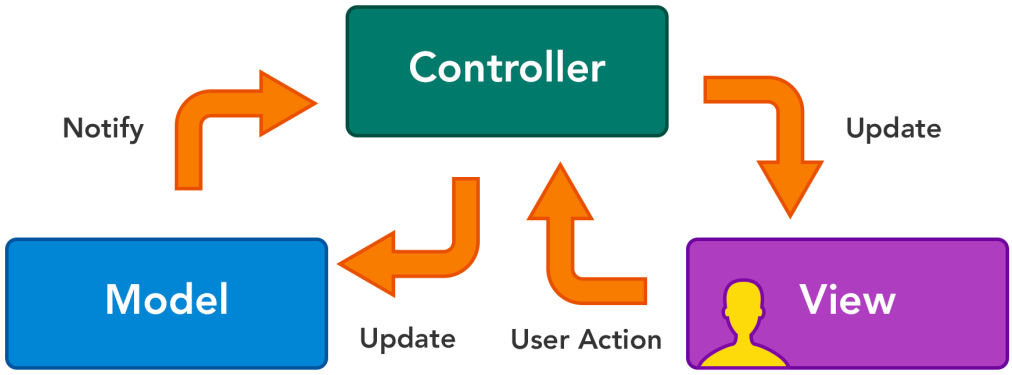
A herança apesar de ser extremamente útil e prática possui uma certa lacuna, pois nem sempre os métodos necessários na classe filha terão de ter comportamentos idênticos ao da classe mãe, por exemplo, o método que calcula o salário de um vendedor não terá o mesmo comportamento do método que calcula o salário de um gerente, apesar de ambos serem classes filhas da classe funcionário. É aí que entra o conceito de polimorfismo, permitindo que o mesmo método tenha comportamentos diferentes dependendo da subclasse.

2.1.2.7 Arquitetura MVC

A arquitetura de projetos Model-View-Controller (MVC) divide a aplicação em três partes:

- Modelo;
- Visão;
- Controlador.

Figura 9 - Fluxo da arquitetura MVC



A camada do "view" é a interface que o usuário terá acesso, o *cliente-side* como vimos anteriormente, ela representa os dados e os exibe. Já a camada do modelo (model) é responsável pela manipulação, leitura, escrita e verificação de dados. O controlador lida com todas as demandas do usuário, essa camada decide qual modelo e qual *view* a aplicação irá utilizar.

O fluxo de um *login* em um programa estruturado em MVC pode ser exemplificado da seguinte maneira:

1. O usuário tem acesso a interface através do *view*, e a partir dela decide realizar o *login*;
2. Então ele entra com seus dados no formulário, a partir disso o *controller* recebe as informações do usuário e chama a função de *login*;
3. Logo após os dados são verificados no *model* que, se validados, retornam ao *controller* que retornará a tela requisitada pelo usuário ao *view*. [12]

2.1.3 Desenvolvimento Web

Como vimos anteriormente o desenvolvimento de *software* está em constante evolução e, hodiernamente, grande parte dele se concentra no desenvolvimento *web*, já que sua adaptabilidade em diversos dispositivos e a baixa demanda de *hardware* em sua execução o tornam mais acessível e prático.

Quando pensamos no desenvolvimento dessa área podemos dividi-lo em duas partes principais:

- *Front-end*: parte da codificação responsável por aquilo que o cliente vê, o *layout* da aplicação, geralmente em HTML, JavaScript, CSS, etc.
- *Back-end*: codifica a comunicação da aplicação com o servidor, estabelecendo a conexão de *APIs* e banco de dados, além do uso de linguagens de programação para estruturar as funcionalidades da página. Exemplos de linguagens são: PHP, Perl, Python, etc.

Nesta seção iremos introduzir algumas das linguagens de marcação e de programação utilizadas no projeto Gerações.

2.1.3.1 HTML

Quando pensamos em desenvolvimento *web* o HTML (*HyperText Markup Language)* pode ser considerado a base de tudo, desenvolvida nos primórdios da *web*, no início da década de 1990, os documentos HTML podem ser interpretados pelos navegadores *web*.

Figura 13 - Exemplo de código HTML

```
<html lang="pt-br">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <h1>Título 1</h1>
    <h2>Título 2</h2>
    <h3>Título 3</h3>
    <h4>Título 4</h4>
    <h5>Título 5</h5>
    <h6>Título 6</h6>

  </body>
</html>
```

A demarcação do código no HTML se dá através de *tags* – palavras marcadas por chevrons (< >) que formatam o documento para que o *browser* interprete a informação de maneira correta, por exemplo, tudo que estiver entre as *tags* <head> e </head> serão definidas como cabeçalho do documento HTML – dentro dessas demarcações podemos atribuir um *id* (exclusivo da *tag*) ou uma *class* (que pode ser designado a infinitas *tags*) que são vitais na utilização do JavaScript ou Bootstrap como veremos mais à frente em suas respectivas seções, além desses atributos de identificação, podemos também inserir atributos exclusivos de uma determinada *tag*, como por exemplo os atributos *href e type* da *tag* <a>, utilizada na definição de *hyperlinks*.

2.1.3.2 CSS

Com o passar dos anos a simplicidade de uma página de HTML puro tornou-se obsoleta e fazer com que os *sites* ficassem mais atraentes aos usuários converteu-se em uma prioridade, a partir desta demanda foi desenvolvido o Cascading Style Sheets (CSS), sua proposta era utilizar HTML somente para estruturar o *site* e o estilo adotado por ele ficaria no arquivo contendo o código com extensão .css ou no próprio HTML demarcado pelas *tags* através do atributo *style*.

2.1.3.3 JavaScript

Javascript é uma linguagem *cliente-side* que manipula e controla os elementos de HTML e CSS, permitindo programar o comportamento de uma página web a partir da ocorrência de um evento jQuery – biblioteca muito utilizada de JavaScrpit – por exemplo. [11]

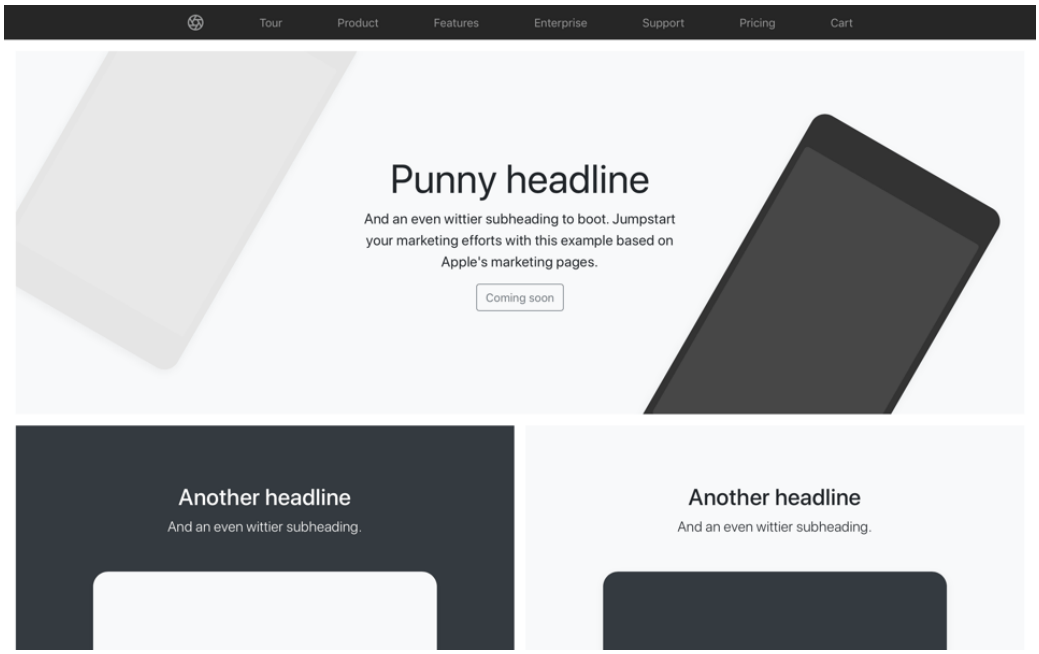
Figura 17 - Exemplo de código JavaScript utilizando jQuery.

```
<script>
jQuery(document).ready(function($) {
    var janelaCriada = false;

    $('.modal').click(function(e) {
        if (janelaCriada) {
            $('#fundoModal').show();
            $('#imagem').attr("src",$(this).attr("href"));
        } else {
            var janelaModal =
            '<div id="fundoModal">' +
            '<div id="janela">' +
            '<h2>Janela Modal</h2><img id="imagem" src="'+$(this).attr("href")+'">' +
            '<p><a href="#" id="fecharModal">Clique aqui para fechar</a>' +
            '</div></div>';
            $('body').append(janelaModal);

            $('#fecharModal').click( function(e) {
                $('#fundoModal').hide();
            });

            janelaCriada = true;
        }

        return false;
    });

});
</script>
```

2.1.3.4 Bootstrap

O Bootstrap é um *framework* – um aglomerado de códigos que provem uma usabilidade genérica mais fácil, para que o desenvolvedor possa focar na resolução do problema e não gaste tempo reescrevendo código – que utiliza HTML, CSS e JavaScript para promover um site mais responsivo e amigável ao usuário, suas funcionalidade são aplicadas através de classes nas *tags* de HTML, entre elas podemos encontrar: *modal, dropdown, button, collapse, carousel e typeahead,* por exemplo, além de permitir a divisão da página em *containers,* colunas e linhas com *breakpoints* que facilita muito o desenvolvimento de aplicações responsivas. [8]

Figura 21 – Exemplo de página que utiliza Bootstrap.



2.1.3.5 PHP

Diferente das ferramentas que vimos anteriormente – HTML, CSS, JavaScript, todas *client-side* – o PHP irá trabalhar no *back-end* da aplicação, fazendo a comunicação entre o servidor e o programa. O PHP surge em 1994, criado pelo programador canadiano-dinamarquês Rasmus Lerdorf, que o utilizava *a priori* para o desenvolvimento de páginas dinâmicas nas quais ele exibia seu currículo, em 1995 Rasmus liberou o código-fonte do PHP, o que instaurou muitos adeptos à esta linguagem, hoje uma das principais no ambiente *web*.[7]

Aplicações em PHP geralmente contam com a extensão *.php*, e seu código pode ser escrito dentro de páginas HTML, demarcado em seu início com "*<?php*" e seu final com "*?>*"

Figura 25 – Exemplo de código PHP dentro de uma página

2.2 Etapas para o Desenvolvimento da Pesquisa

Nas próximas seções discorreremos, através das etapas anteriormente estabelecidas, o desenvolvimento deste trabalho de pesquisa.

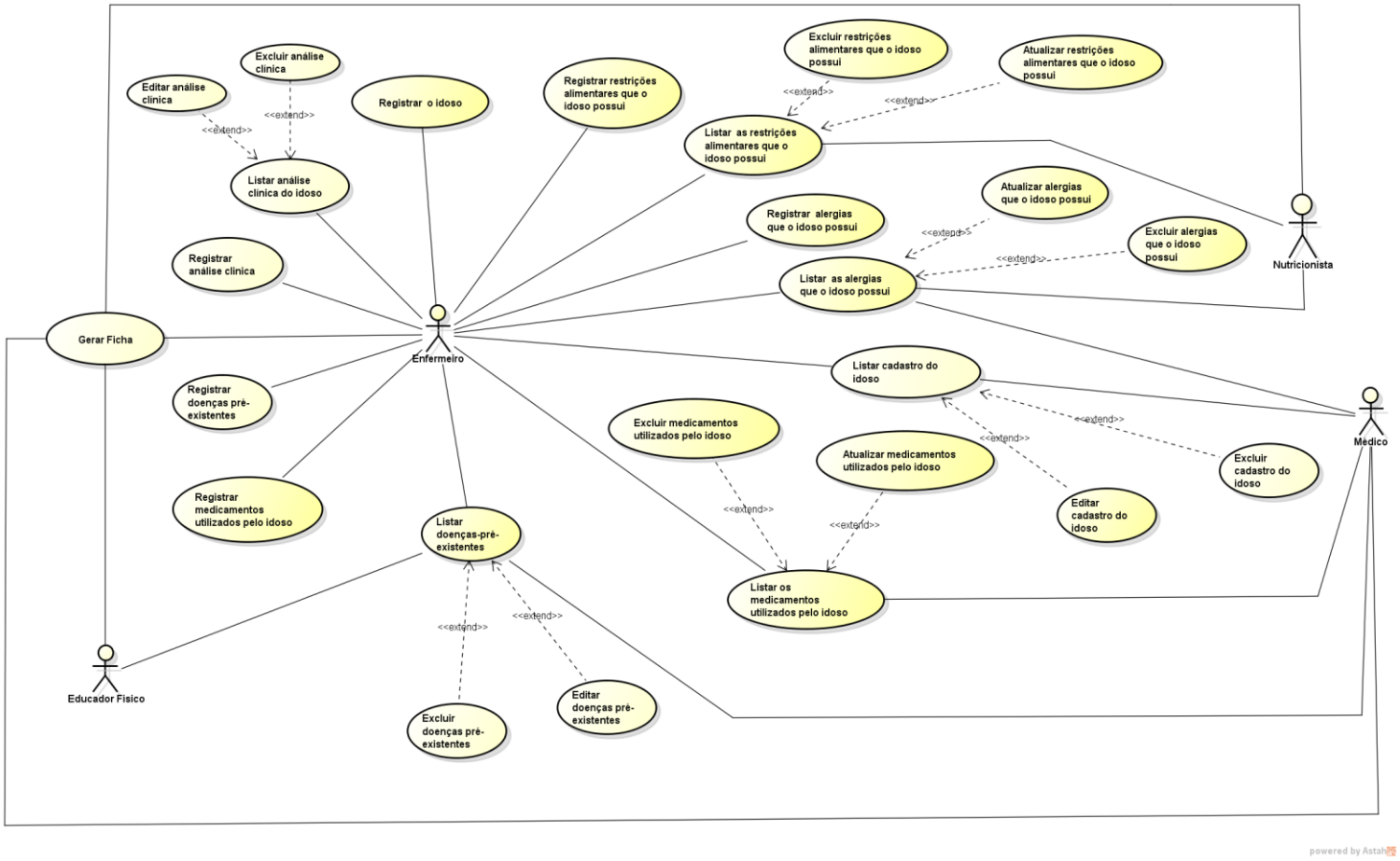
2.2.1 Requisitos do Módulo 02

Quando pensamos em desenvolver um *software* não basta sentarmos na cadeira e escrever o código sem qualquer tipo de orientação, é necessário um analista, alguém que o enxergue como um todo, escreva suas funcionalidades, elabore seus diagramas, etc., tal documentação é vital para o planejamento e integração entre os módulos do projeto.

Nesta seção será apresentado os requisitos do módulo 2 – as funcionalidades obrigatórias a serem desenvolvidas – necessárias para o compreendimento das próximas fases deste trabalho.

A imagem a seguir apresenta um diagrama que representa os atores do sistema (agentes) – representados por bonecos – e o que eles podem fazer (funcionalidades) – representadas por balões de texto.

Figura 29 – Diagrama de casos de uso do módulo 2



Além da imagem anterior podemos tomar de orientação ao desenvolvimento os requisitos funcionais do módulo representados na tabela a seguir, que ditam quais as funcionalidades essências a serem desenvolvidas e as especificam de forma detalhada.

Tabela 1 - Requisitos funcionais do módulo 2

Identificador	Descrição do Requisito
RF #01	O sistema deve gerar prontuários automaticamente gerando uma sequência numérica pré-definida, começando pelo número 01 para o cadastro do idoso. O prontuário deve aparecer em cima da tela de cadastro, se cancelarem o cadastro, esse prontuário não será atribuído a nenhum idoso.
RF #02	O Sistema deve cadastrar o idoso, com as seguintes informações: • Nome Completo (varchar 255) – Obrigatório • RG (varchar 9) – Obrigatório • CPF (char 12 XXX.XXX.XXX-XX) – Obrigatório – deve ser validado • Sexo (M ou F) – Obrigatório • Data de nascimento (DD/MM/AAAA) – Obrigatório • Estado do idoso (Ativo/Inativo), inicialmente ativo – Obrigatório • Fralda ou forrinho (S/N) - Obrigatório

	• Código do responsável (O código deve ser pesquisado a partir do nome do responsável e/ou CPF, se o responsável for cadastrado, será selecionado e será designado ao idoso em questão. Se não houver responsável cadastrado (Botão "Não tenho responsável" (direcionar para a funcionalidade do Módulo 01), o site deverá ir para a página de cadastro de responsável) – obrigatório, não pode ser alterado. • Para atualizar será necessário fazer uma pesquisa antes procurando pelo nome ou cpf do idoso e irá aparecer uma tabela com o nome ou parte do nome dos idosos e nessa tabela conterá o nome, o cpf, o prontuário do idoso, e no final de cada linha de informações conterá um botão de editar onde se clicado levará para página de cadastro do idoso com todas as informações dele contida, onde poderá ser editado todos os campos, com exceção do prontuário. • O cadastro irá funcionar da seguinte forma: o funcionário pesquisa pelo nome ou cpf do idoso, caso não existir cadastro irá aparecer uma mensagem de cadastro inexistente, onde o funcionário irá cadastrar o idoso com todas as informações pedidas, e no final irá ter um botão de salvar, se os campo cpf estiver errado vai aparecer um erro para corrigir, pois esse campo deve ser validado, se tudo estiver certo após salvar, vai aparecer a mesma tela com todas os campos preenchidos, onde a pessoa que está cadastrando poderá modificar algum campo caso esteja errado, após isso terá o botão confirmar onde o cadastro vai ser concluído. • Após o cadastro ser concluído voltará para a tela inicial de pesquisa, e o funcionário deverá pesquisar pelo nome, cpf ou prontuário do idoso para conferir se realmente foi cadastrado, se sim, aparecerá o nome do idoso na tabela. • Todas as vezes que forem realizadas pesquisas de idosos, selecionando o nome ou CPF do mesmo, uma tabela irá listar todos os idosos que respeitem os parâmetros de pesquisa, possuindo os seguintes atributos: Prontuário, Nome e CPF. Na frente de cada um dos idosos, será mostrado opções para continuar o seu cadastro de entrada na Instituição de Longa Permanência. As opções que serão apresentadas são: o Registro de Análise Clínica: Inserir, Editar e Visualizar (RF#04); o Doenças: Inserir, Editar e Visualizar (RF#05); o Medicações: Inserir, Editar e Visualizar (RF#06); o Restrições Alimentares: Inserir, Editar e Visualizar (RF#07); o Alergias: Inserir, Editar e Visualizar (RF#08).
RF #03	O Sistema deve mostrar a listagem de todos os idosos, a partir do nome ou cpf digitado, só o funcionário responsável pelo cadastro terá acesso a essa listagem. Também deverá ter uma lista completa de todos os idosos cadastrados na instituição, aparecendo ao lado do nome o seu código e se está ativo ou não. Essa lista será outra escolha que terá na página inicial na opção de meu idoso. Essa opção terá 3 escolhas sendo elas: pesquisa e cadastro de idosos, ficha completa dos idosos, lista de idosos cadastrado sendo essa última um select da tabela idoso.
RF #04	O sistema deve registrar a analise clínica do idoso ao entrar na casa. Esse registro deve conter: • Alergias (S/N) • Tipo sanguíneo (Lista exibindo os seguintes tipos: A+, A-, B+, B-, AB+, AB-, O+ e O-) • Situação atual idoso (Saudável/doenças leves/Enfermo) • Altura e Peso (XXX centímetros, XXX,XXX Quilos) • Após concluído o cadastro do idoso, ao final de cada linha na listagem terá outra tabela com 3 botões(registrar a analise clínica, editar e visualizar) se clicar no botão de registro vai aparecer uma página com todas os campos citados acima, preenchendo todos, no botão de editar irá aparecer a mesma página com os campos já preenchidos podendo editar quaisquer campos, no botão de visualizar só será mostrado a analise clínica do idoso em pdf.
RF #05	O sistema deve registrar todas as doenças pré-existentes do idoso e atualiza-las de acordo com a situação clínica do idoso. Em outra tabela terá outro botão para registrar as doenças existentes onde abrirá uma página com algumas possíveis doenças pré-cadastradas no banco de dados (como diabete, anemia, pressão baixa, entre outras), com a opção de escolher sim ou não para aquele problema, e se caso o idoso tiver uma

	doença não presente na página, haverá um campo para digitar quais doenças o idoso tem, e terá um botão de editar e visualizar em pdf as doenças do idoso. Tendo este registro exibido a todos os interessados por cada área da instituição.
RF #06	O sistema deve registrar as medicações prescritas dos idosos especificando quais medicamentos e sua quantidade diária. Ao final de cada linha na listagem terá outra tabela de medicações do lado da analise clinica com 3 botões (registrar medicações, editar e visualizar) se clicar no botão de registro vai aparecer uma página com a opção de escolher se sim para quem tem medicações e não para quem não usa nada, se sim vai aparecer um campo medicações onde será descrito que tipo de medicação, para que finalidade ele toma o medicamento e outro para especificar a quantidade diária. Vale destacar que essas 3 informações devem ser inseridas para cada uma das medicações tomadas pelo idoso. No botão de editar irá aparecer a mesma página com os campos já preenchidos podendo editar quaisquer campos, no botão de visualizar só será mostrado as medicações, finalidade e quantidade. Tendo este registro exibido a todos os interessados por cada área da instituição.
RF #07	O sistema deve registrar as restrições alimentares do idoso especificando sua dieta especifica para suas necessidades nutricionais. Da mesma forma que medicações e analise clinica haverá outra tabela de restrições alimentares com os mesmos botões, onde no botão de registrar haverá uma tabela com possíveis restrições alimentares demarcando com "x" para as devidas restrições. Essas restrições alimentares serão apresentadas a partir de uma lista pré-cadastrada no Banco de Dados. No botão de editar irá aparecer a mesma página com os campos já preenchidos podendo editar quaisquer campos, no botão de visualizar só será mostrado as restrições alimentares. Tendo este registro exibido a todos aqueles que interessarem.
RF #08	O sistema deve registrar os tipos de alergias que o idoso possui, especificando o tipo de medicamento correto para cada caso. Em uma tabela como os demais, terá os mesmos botões que os anteriores, e quando registrar deverá ser especificado quais tipos de alergias têm, explicando seus cuidados diários e definindo os medicamentos corretos para ela. Podem ser criadas quantas alergias forem necessárias, mas estes 3 campos devem ser sempre informados para cada um. No botão de editar irá aparecer a mesma página com os campos já preenchidos podendo editar quaisquer campos, no botão de visualizar só será mostrado as alergias, seus cuidados diários e as respectivas medicações. Tendo este registro exibido a todos os aqueles que interessarem.
RF #09	O sistema deve gerar uma ficha completa do idoso em pdf, sendo detalhados todos os dados registrados anteriormente, sendo eles a analise clínica do idoso, as doenças preexistentes, as medicações, as alergias, e as restrições alimentares, dando acesso aos responsáveis de cada área medica. Ao entrar na opção de ficha completa do idoso, no menu "Meu Idoso", aparecerá uma área de pesquisa pelo nome ou prontuário do idoso, o funcionário ao pesquisar e selecionar o nome estará logado e se não estiver, quando selecionar para gerar a ficha será levado para a tela de login, depois de autenticado voltara à página de pesquisa onde terá um botão de gerar ficha de determinado idoso, onde será gerada as informações detalhadas da situação do mesmo. A ficha deverá ser apresentada da seguinte forma: Dados Gerais: • Nome Completo • Prontuário • CPF • RG • Data de Nascimento • Sexo

	• Status • Responsável • Usa Fralda / Forrinho Análise Clínica: • Alergias • Tipo sanguíneo • Situação atual idoso • Altura e Peso Doenças Pré-Existentes: • Lista com os Nomes da Doenças Medicações: • Nome do Medicamento • Finalidade • Quantidade Restrições Alimentares: • Lista com os Nomes das Restrições Alergias: • Nome da Alergia • Cuidados Diários • Medicação

2.2.2 Planejamento das iterações

Entregar um projeto deste tamanho com um prazo estipulado necessita de muita organização e compromisso por parte dos envolvidos, a partir dessa necessidade um cronograma de entregas foi elaborado, definindo a data, qual caso de uso e a equipe responsável dentro do módulo.

Tabela 2 – Planejamento de entrega de iterações.

Iteração	Data Prevista	Casos de Uso	Equipe Responsável
#01	02/07/2019	Registrar o cadastro do idoso	Pedro Abreu (Desenvolvedor) e Verônica Forte (Desenvolvedora)
#02	23/08/2019	Listar cadastro do idoso	Grupo 01: Filipe Muniz (Analista), Verônica Forte (Desenvolvedor) e Marina Mello (DBA)
		Editar Cadastro do idoso	Grupo 01: Filipe Muniz (Analista), Verônica Forte (Desenvolvedor) e Marina Mello (DBA)
		Excluir cadastro do idoso	Grupo 01: Filipe Muniz (Analista), Verônica Forte (Desenvolvedor) e Marina Mello (DBA)

		Registrar Medicamentos utilizados pelo idoso	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)
		Listar Medicamentos utilizados pelo idoso	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)
		Atualizar medicamentos utilizados pelo idoso	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)
		Excluir Medicamentos	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)
#03	20/09/2019	Registrar alergias	Grupo 01: Filipe Muniz (Analista), Verônica Forte (Desenvolvedor) e Marina Mello (DBA)
		Listar alergias	Grupo 01: Filipe Muniz (Analista), Verônica Forte (Desenvolvedor) e Marina Mello (DBA)
		Editar alergias	Grupo 01: Filipe Muniz (Analista), Verônica Forte (Desenvolvedor) e Marina Mello (DBA)
		Excluir alergias	Grupo 01: Filipe Muniz (Analista), Verônica Forte (Desenvolvedor) e Marina Mello (DBA)
		Registrar análise clínica	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)
		Editar análise clínica	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)
		Excluir análise clínica	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)
		Listar análise clínica do idoso	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)
#04	18/10/2019	Registrar restrições alimentares do idoso	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)
		Listar restrições alimentares do idoso	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli
		Editar restrições alimentares do idoso	Grupo 01: Filipe Muniz (Analista), Verônica Forte (Desenvolvedor) e Marina Mello (DBA)

		Excluir restrições alimentares do idoso	Grupo 01: Filipe Muniz (Analista), Verônica Forte (Desenvolvedor) e Marina Mello (DBA)
#05	08/11/2019	Gerar ficha do idoso	Grupo 02: Beatriz Basilio (Analista), Pedro Abreu (Desenvolvedor) e Henrique Bissoli (DBA)

2.2.3 Desenvolvimento dos Casos de Uso

Nesta seção será apresentado o processo de desenvolvimento de alguns casos de uso do módulo 2, os casos selecionados são:

1. Listar cadastro dos idosos;
2. Excluir medicamentos utilizados pelos idosos;
3. Registrar análise clínica do idoso.

Os casos de uso selecionados serão apresentados da seguinte maneira: uma breve explicação suas funcionalidades, código por trás da aplicação, interface gráfica e seu fluxo.

1. Listar cadastro dos idosos

Uma das páginas mais importantes do módulo, que permite acesso as outras funcionalidades desenvolvidas. Apresenta o nome do idoso com ações a serem tomadas.

A partir da tela inicial do sistema, um usuário logado como enfermeiro – através do menu prontuário e em seguida do botão listar cadastro – tem acesso a tela como mostra a figura 10, onde ele pode acionar diversas funcionalidades desenvolvidas pelo módulo 2, como listar suas informações de cadastro através do botão “Listar” que desencadeará uma janela pop-up com os registros do idoso – como a figura 11 – além disso o usuário pode optar por acessar os medicamentos, doenças, análises clínicas, alergias e restrições alimentares do idoso, editá-los, registrar novos e excluí-los.

Figura 37 – Código por trás da aplicação.

```
<div class="table-responsive">
  <table class="table table-bordered" id="" width="100%" cellspacing="">
    <thead>
      <tr>
        <th>Nome</th>
        <th>Prontuário</th>
        <th>Status</th>
        <th>Ações</th>
      </tr>
    </thead>

    <tbody>
    <?php
        foreach ($idosos as $idoso) {
    ?>
        <tr>
            <td><?php echo $idoso->__get('nome'); ?></td>
            <td><?php echo $idoso->__get('prontuario'); ?></td>
            <td><?php if($idoso->__get('estado') === "1"){echo "Ativo";}else{echo "Inativo";} ?></td>
            <td><a onclick="UpdateInfo('<?php echo $idoso->__get('prontuario'); ?>', '<?php echo $idoso->__get('nome'); ?>', '
                <?php echo $idoso->__get('rg'); ?>', '<?php echo $idoso->__get('cpf'); ?>', '<?php echo $idoso->__get('sexo');
                ?>', '<?php echo $idoso->__get('data_nasc'); ?>', '<?php echo $idoso->__get('data_ingresso'); ?>','<?php echo $
                idoso->__get('estado'); ?>', '<?php echo $idoso->__get('fralda_ou_forrinho'); ?>', '<?php echo $idoso->__get('
                cod_resp'); ?>');" data-toggle="modal" data-target="#modalQuickView" class="text-primary btn btn-outline-primary
            ">Listar</a>
                <a href="medicamentos?id=<?php echo $idoso->__get('prontuario'); ?>" class="btn btn-outline-primary ml-1">
                Medicamentos </a>
                <a href="doencas?id=<?php echo $idoso->__get('prontuario'); ?>" class="btn btn-outline-primary ml-1">Doenças</a>
                <a href="analiseclinica?id=<?php echo $idoso->__get('prontuario'); ?>" class="btn btn-outline-primary ml-1">
                Análise Clínica </a>
                <a href="listar_alergia?id=<?php echo $idoso->__get('prontuario'); ?>" class="btn btn-outline-primary ml-1">
                Alergias</a>
                <a id="id_rest" onclick="listarRestricoes('<?php echo $idoso->__get('prontuario'); ?>')" class="btn
                btn-outline-primary ml-1 text-primary">Restrições Alimentares</a>

        </tr>
    <?php
        }
    ?>
    </tbody>
  </table>
</div>
```

Figura 33 - Interface da listagem de idosos.

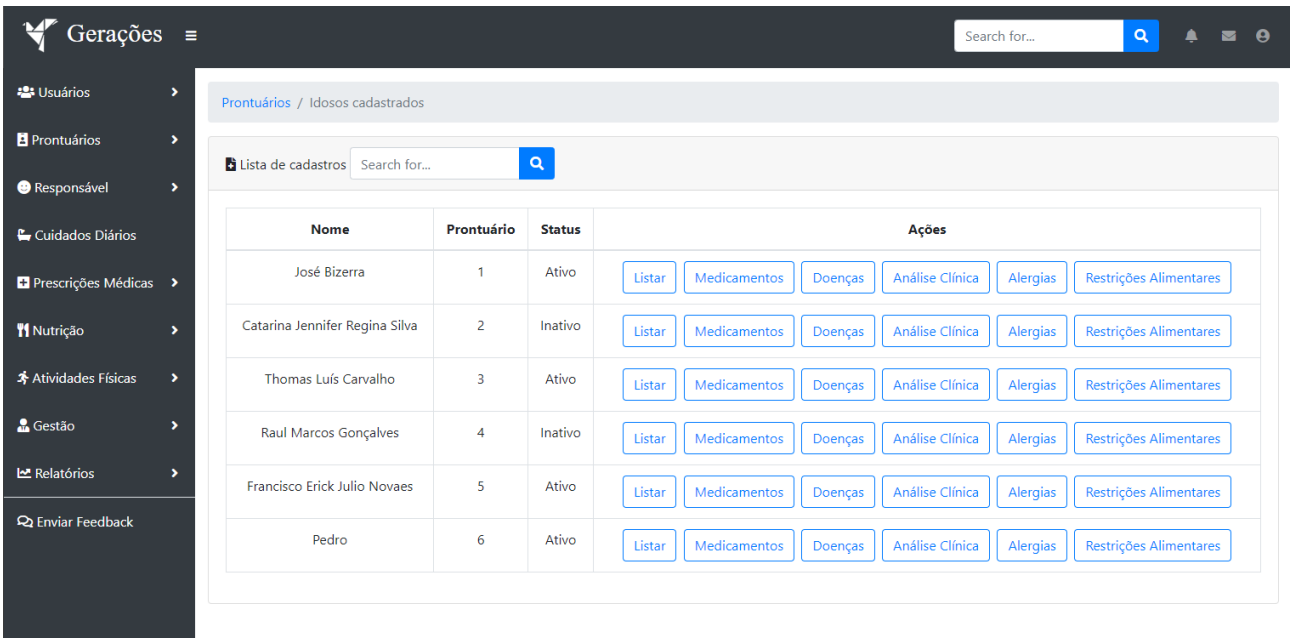
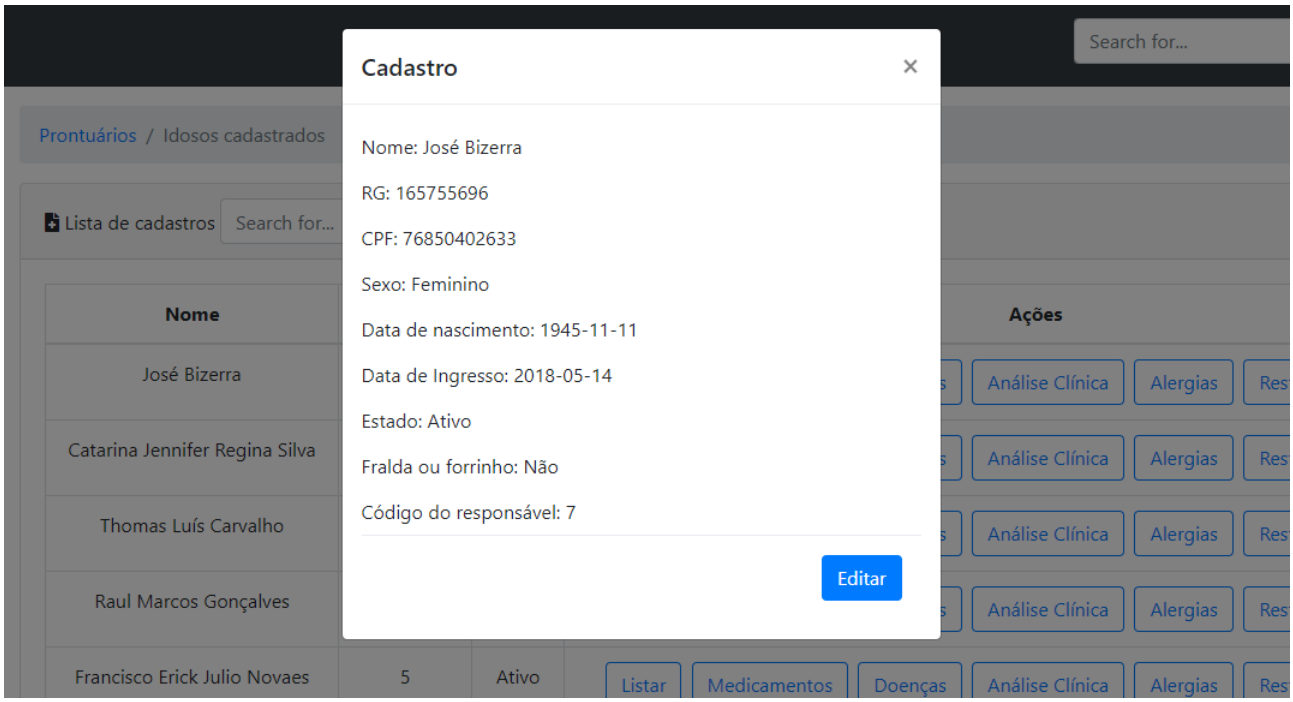


Figura 41 - Janela modal que exibe dados registrados do idoso.



2. Excluir medicamentos utilizados pelos idosos

Assim como o caso de uso anterior, para ter acesso a funcionalidade de excluir medicamentos do idoso é necessário estar logado como enfermeiro, para que assim usuários que não dizem a respeito de tal ação não possam realiza-la.

A partir da tela da listagem de idosos que vimos anteriormente na figura 10, o enfermeiro clica no botão “medicamentos” que aciona a função “medicamentos()” no controlador do módulo 2 e passa o *id* do idoso selecionado, esse *id* possibilita a chamada da função “listarPorId_Idoso()” que lista os medicamentos ligados ao idoso com o *id* correspondente e os exibe na tabela do página medicamentos.

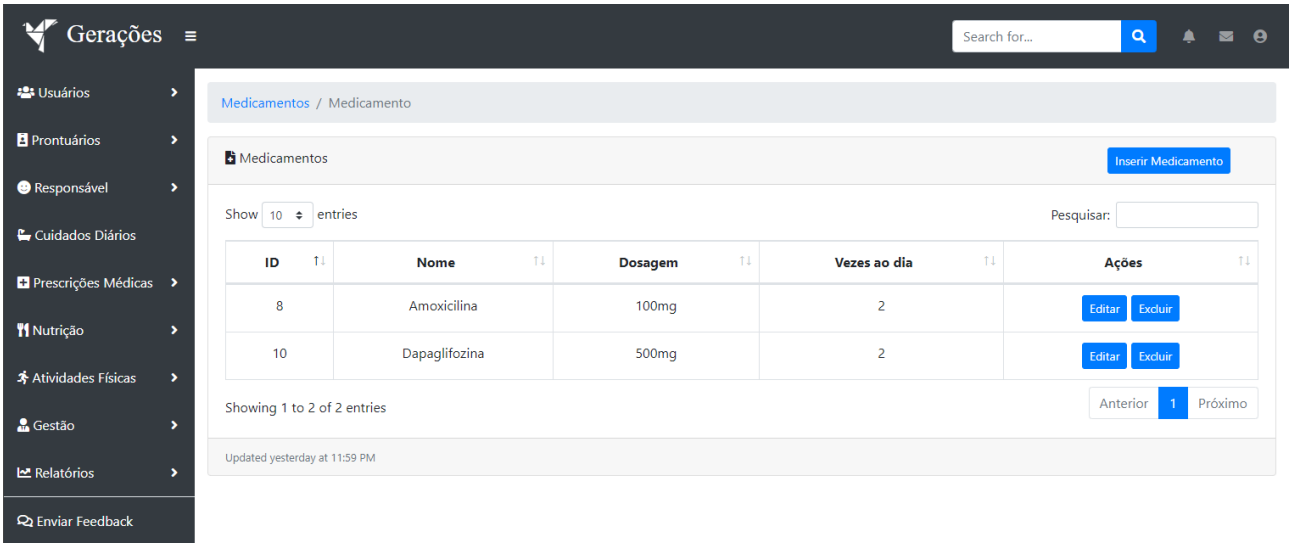
Figura 45 - Código da função medicamentos no controlador do módulo 2.

```
class Modulo02Controller extends Action{
    public function medicamentos(){
        $idoso_medsDAO = new Idoso_medicamentoDAO();
        $pront_medDAO = new pront_medDAO();
        $idoso_medicamento = $idoso_medsDAO->listarPorId_Idoso($_GET['id']);

        $this->getView()->dados = $idoso_medicamento;
        $this->getView()->prontMed = $pront_medDAO;
        $this->render('medicamentos', 'dashboard', '../');
    }
```

Figura 53 - Código da página medicamentos no view do módulo 2.

```
<div class="card-body">
    <div class="table-responsive">
        <table class="table table-bordered text-center" id="dataTable" width="100%" cellspacing="0">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Nome</th>
                    <th>Dosagem</th>
                    <th>Vezes ao dia</th>
                    <th>Ações</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($this->getView()->dados as $idoso_meds) { ?>
                    <tr>
                        <td class="idMedicamento"><?php echo $idoso_meds->__get('id_medicamento'); ?></td>
                        <td ><?php echo $this->getView()->prontMed->buscarPorId($idoso_meds->__get('id_medicamento'))->__get
                          ('nome'); ?></td>
                        <td><?php echo $this->getView()->prontMed->buscarPorId($idoso_meds->__get('id_medicamento'))->__get(
                          'dosagem'); ?></td>
                        <td><?php echo $this->getView()->prontMed->buscarPorId($idoso_meds->__get('id_medicamento'))->__get(
                          'quantdade'); ?></td>
                        <td>
                            <a class="btn btn-primary btn-sm text-white" onclick="editarModal('<?php echo $idoso_meds->__get
                              ('id_medicamento'); ?>')"> Editar </a>
                            <a class="btn btn-primary btn-sm text-white delete" onclick="excluirModal('<?php echo $
                              idoso_meds->__get('id'); ?>')">Excluir </a>
                        </td>
                    </tr>
                <?php } ?>
            </tbody>
        </table>
    </div>
</div>
```

Figura 49 - Interface da página medicamentos do módulo 2.



Através da página de medicações o usuário pode excluir ou editar o medicamento em questão. Ao selecionar a ação excluir é chamado uma função em JavaScript que exibe um modal (pop-up) e se o enfermeiro confirmar a exclusão, essa função direciona o usuário a função excluirMedicamento que, através do *id* do medicamento chama a função excluir que apaga a medicação do banco de dados.

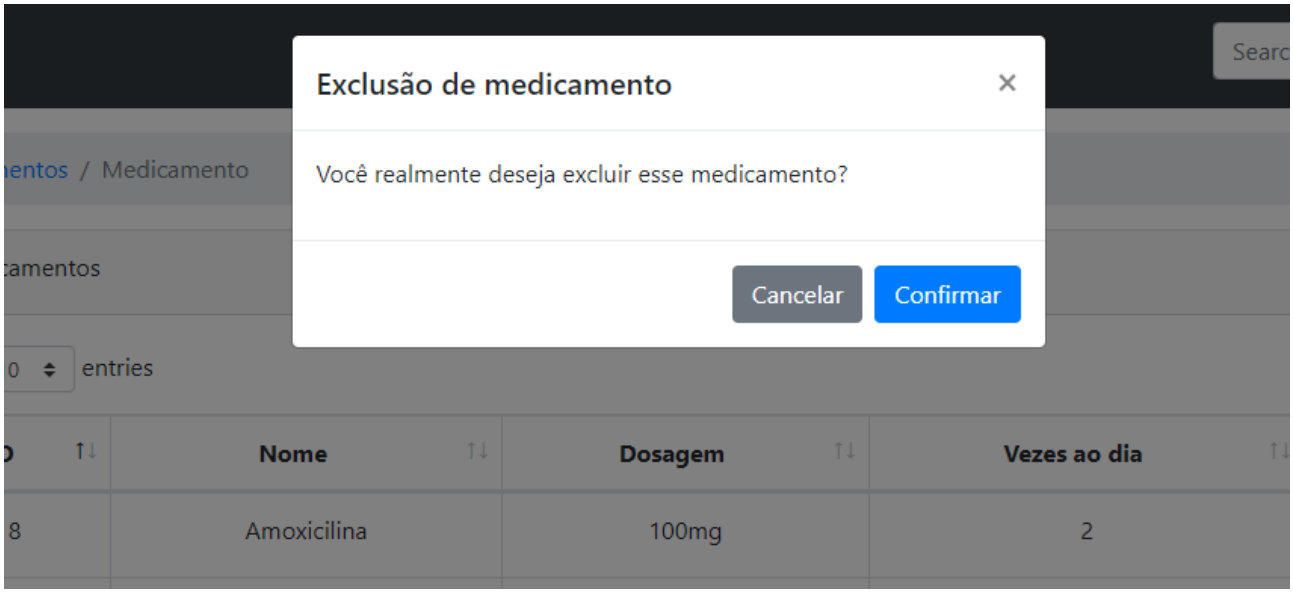
Figura 65 – Código da função em JavaScript responsável pela exclusão do medicamento.

```
function excluirModal(id)
{
    $('a.delete-yes').attr('href', '/md2/excluirMedicamento?ID=' +id); // mudar dinamicamente o link, href do botão confirmar da modal
    $('#confirmar').modal('show'); // modal aparece
}
```

Figura 61 - Código da função desencadeada pelo JavaScript.

```
public function excluirMedicamento(){
    $pront_medDAO = new pront_medDAO();
    $pront_medDAO->excluir($_GET['ID']);
    header('Location: /md2/listar_cadastro');
    die();
}
```

Figura 57 - Interface do modal de exclusão de medicamentos.



3. Registrar análise clínica do idoso

Partindo do mesmo princípio dos casos anteriores, para a realização do registro da análise clínica do idoso, o usuário acessa a página de listagem de internos (figura 10) e seleciona a função análise clínica, esse botão levará o usuário uma página que possui a última análise clinica registrada do idoso, se não houver nenhuma somente o botão de inserir estará disponível, caso contrário as funcionalidades de edição e exclusão se tornaram visíveis.

Se o idoso não possuir análise clínica e o enfermeiro responsável decidir registrá-la ele deve clicar em “Inserir Análise Clínica”, tal botão redirecionará o usuário a uma nova página com um formulário de cadastro de análise, com os campos: peso, altura, sintomas e observações. Ao enviar as informações desejadas a função “cadastrarAnalise()” é requerida e as registra no banco de dados.

Figura 69 - Interface da listagem da análise clínica quanto inexistente.

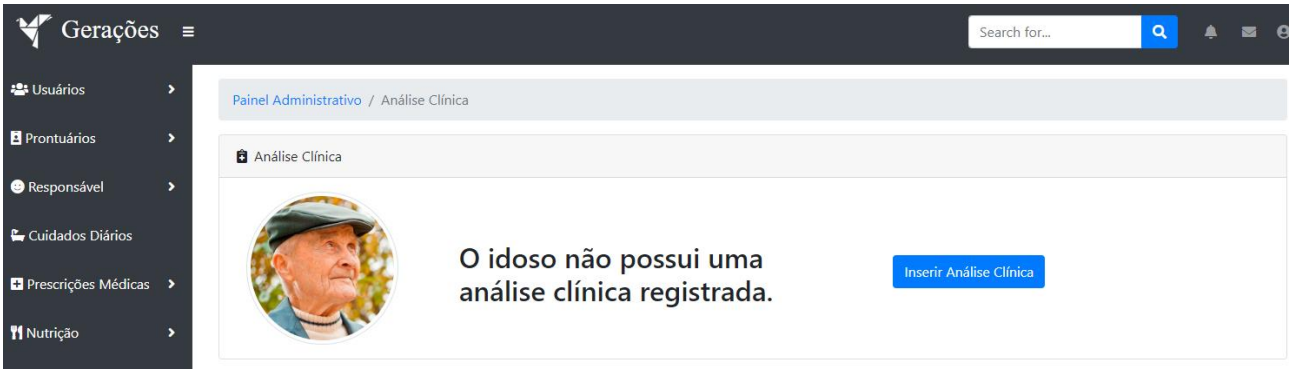


Figura 73 - Código do formulário de registro de análise.

```
<div class="card-body">
    <form id="login-form" class="form" action="cadastroAnalise?ID=<?php echo $id; ?>" method="post">
        <h3 class="text-center text-dark text-dark display-4">Inserir Análise Clínica</h3>
        <div class="text-center">
            <p class="titulo"><i class="fas fa-notes-medical fa-5x mt-3"></i></p>
        </div>
        <div class="form-group">
            <div class="row">
                <div class="col-2 ml-auto">
                        <label for="peso" class="text-dark text-dark display-5">Peso:</label><br>
                        <input type="text" name="peso" id="peso" class="form-control" required>
                </div>
                <div class="col-2 mr-auto">
                        <label for="altura" class="text-dark text-dark display-5">Altura:</label><br>
                        <input type="text" name="altura" id="altura" class="form-control" required>
                </div>
            </div>
        </div>
            <div class="form-group">
                <div class="row">
                    <div class="col-4 m-auto">
                        <label for="vezes_dia" class="text-dark text-dark display-5">Sintomas:</label><br>
                        <input type="text" name="sintomas" id="sintomas" class="form-control" required>
                    </div>
                </div>
            </div>
            <div class="form-group">
                <div class="row">
                    <div class="col-4 m-auto">
                        <label for="vezes_dia" class="text-dark text-dark display-5">Observações:</label><br>
                        <textarea rows='5' name="obs" id="obs" class="form-control"></textarea>
                    </div>
                </div>
            </div>
        </div>
        </br>
        <div class="form-group text-center">
            <button type="submit" class="btn btn-dark btn-lg">Inserir</a>
        </div>
    </form>
</div>
```

Figura 77 - Interface do formulário de registro de análise.

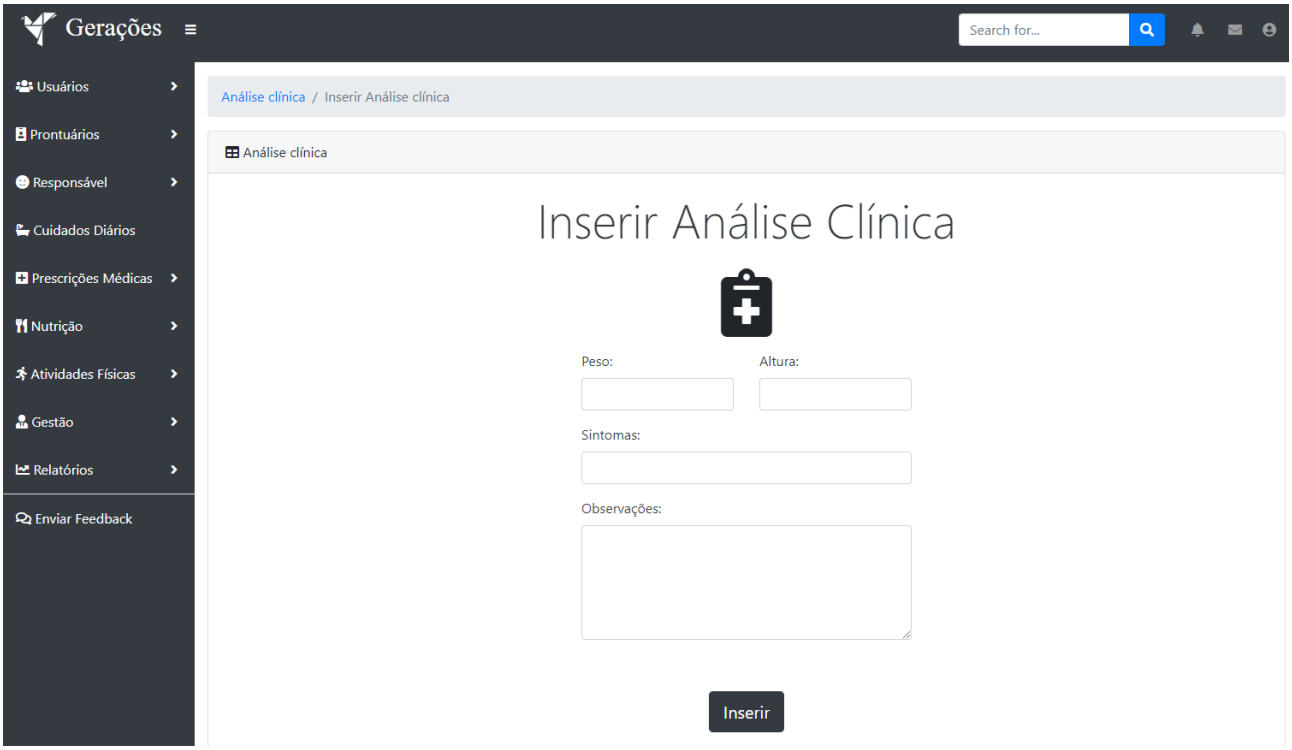
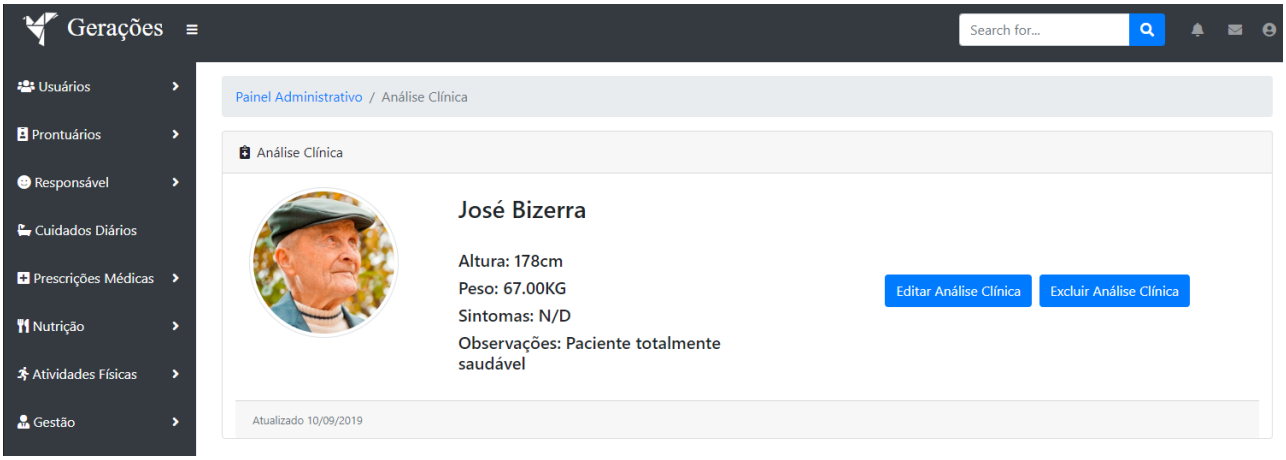


Figura 81 - Interface da listagem de análise após o registro.



2.2.4 Dificuldades na refatoração do código

Apesar de o desenvolvimento do projeto ser dividido entre módulos, chega uma hora na qual é necessário a integração entre eles.

A refatoração do código é um dos maiores desafios do projeto, afinal cada desenvolvedor de cada módulo possui seu modo de pensar, escrever e organizar sua parte da aplicação, e se desenvolver em conjunto com seu módulo já é um tanto quanto difícil, imagine integrá-lo com os

restantes, é a partir desse obstáculo que a equipe de desenvolvedores acaba por se mobilizar como um todo para superá-lo e, a partir disso, estabelecer uma maior união.

2.2.5 Adaptações necessárias e reutilização de código durante o desenvolvimento

Provavelmente uma das maiores vantagens do MVC e da programação orientada a objetos, a reutilização de código é de indubitável importância permitindo os desenvolvedores pouparem horas de trabalho em um projeto de tamanha complexidade e tamanho. Essa vantagem foi muito explorada durante o desenvolvimento do Gerações. Algo que a grande maioria dos módulos possuía era a necessidade da exibição de tabelas, o datatables – plugin de jQuery – facilitou, e muito, esse processo, e foi utilizado em praticamente todos os módulos.

Nem sempre o desenvolvimento se sai de acordo com o levantamento de requisitos, são necessárias adaptações, e isso não é diferente no módulo 2, grande parte do que foi desenvolvido foi modificado, seja por questões de compatibilidade ou por impossibilidades ocasionadas por erros técnicos.

Um exemplo disso é muitas informações que, a priori, foram julgadas como essenciais, porém com o decorrer do projeto se tornaram desnecessárias e foram excluídas da aplicação.

3 Conclusões e Recomendações

Em 2019, a matéria de projeto de desenvolvimento de sistemas propôs aos quartos anos de informática, a elaboração de um sistema que facilitaria os a integração entre idosos internados em instituições de longa permanência e seus parentes, o Gerações, algo muito coerente, já que São João da Boa Vista é colocado como um dos melhores lugares para idosos viverem.

Este trabalho nasce com o intuito de documentar e apresentar os métodos de desenvolvimento adotados no módulo 2 do projeto, responsável pelo gerenciamento de dados dos idosos, seus medicamentos, alergias, restrições alimentares e análises clínicas, além disso introduzir o leitor a conceitos básicos de ferramentas utilizadas.

O desenvolvimento de um sistema desse tamanho – mais de 50 pessoas trabalhando – exige muita organização por parte de todos, é aí que o levantamento de requisitos entra em campo, servindo de norte a todos a equipe de desenvolvedores, nosso módulo levantou 9 requisitos funcionais.

Alicerçado nesse levantamento, é dado o início do desenvolvimento com a prototipação de interfaces homem-máquina, que serviriam de subsídio para o desenvolvimento propriamente dito, quando essa etapa foi concluída, começa a refatoração do código, que visa integrar os módulos e estabelecer padrões de desenvolvimento.

A etapa de refatoração serviu como um obstáculo que acabou por beneficiar o grupo como um todo, estabelecendo uma relação de equipe, apesar de adversidades.

Este trabalho de pesquisa me permitiu rever o esforço de toda a sala em conjunto, principalmente dos desenvolvedores com quem mantive mais contato durante todo o ano, e fazê-lo foi verdadeiramente recompensador. Revisar cada iteração e protótipo e revisitar linhas de código do começo do desenvolvimento, reforçaram o quão complexo e extenso foi esse projeto e que, apesar disso, conseguimos entregar um bom resultado. Por outro lado, a confecção desta pesquisa não possui somente pontos positivos, minha falta de experiência na elaboração de trabalhos deste nível acabou servindo como empecilho, acabei por gerir mal meu tempo e procrastinar em diversas situações, acumulando muito o que se fazer em pouco tempo disponível.

Como recomendações em trabalhos futuros coloco a inserção dos desenvolvedores do projeto na documentação desde o início, para corrigir possíveis erros na análise do sistema e debates diferentes métodos de execução do desenvolvimento.

Referências Bibliográficas

[1] IFSP Câmpus São João da Boa Vista. **IFSP Câmpus São João da Boa Vista.** 2017. Disponível em: <https://www.sbv.ifsp.edu.br/sobre-campus>. Acesso em: 10 out. 2019.

[2] IBGE. **São João da Boa Vista.** 2019. Disponível em: <https://www.ibge.gov.br/cidades-e-estados/sp/sao-joao-da-boa-vista.html>. Acesso em: 10 out. 2019.

[3] IBGE. **São João da Boa Vista.** 2019. Disponível em: <https://cidades.ibge.gov.br/brasil/sp/sao-joao-da-boa-vista/pesquisa/23/24304>. Acesso em: 10 out. 2019.

[4] SÃO JOÃO DA BOA VISTA. PREFEITURA. São João é também a melhor cidade do país para idosos. 2017. Disponível em: <http://saojoao.sp.gov.br/home/ler_noticia.php?id=2312>. Acesso em: 10 out. 2019. [5] TUCKER, Allen; NOONAN, Robert. **Linguagens de Programação - Princípios e Paradigmas.** 2. ed. São Paulo: AMGH Editora, 2010.

[6] SINTES, Anthony. **Aprenda Programação Orientada a Objetos em 21 dias.** São Paulo: Pearson Makron Books, 2002.

[7] DALL'OGLIO, Pablo. **PHP: Programando com Orientação a Objetos.** 2ª Edição. São Paulo: Editora Novatec, 2009.

[8] TEAM, Bootstrap. **Sobre.** Disponível em: <https://getbootstrap.com.br/docs/4.1/about/overview/>. Acesso em: 09 out. 2019.

[9] EIS, Diego. **Uma breve história do CSS.** 2006. Disponível em: <https://tableless.com.br/uma-breve-historia-do-css/>. Acesso em: 10 out. 2019.

[10] ALTIERI. **A origem do CSS, um pouco da história.** 2009. Disponível em: <https://www.devmedia.com.br/a-origem-do-css-um-pouco-da-historia/15195>. Acesso em: 10 out. 2019.

[11] TEAM, Javascript. **JavaScript.** 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 10 out. 2019.

[12] RAMOS, Allan. **O que é MVC?** 2015. Disponível em: <https://tableless.com.br/mvc-afinal-e-o-que/>. Acesso em: 10 out. 2019.