

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso de Bacharelado em Sistemas de Informação

Saulo Daniel Medeiros

SISTEMA DE CONTROLE DE VACINAÇÃO PESSOAL - CVPSIS

Belo Horizonte
2018

Saulo Daniel Medeiros

SISTEMA DE CONTROLE DE VACINAÇÃO PESSOAL - CVPSIS

Monografia apresentada ao programa de Curso de Bacharelado em Sistemas de Informação da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: João Caram Santos de Oliveira

Belo Horizonte
2018

Saulo Daniel Medeiros

SISTEMA DE CONTROLE DE VACINAÇÃO PESSOAL - CVPSIS

Monografia apresentada ao programa de Curso de Bacharelado em Sistemas de Informação da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

João Caram Santos de Oliveira

1o avaliador

2o avaliador

Belo Horizonte, 15 de Dezembro de 2018

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema de controle de vacinação pessoal. Quando ainda estamos na infância a vacinação é algo muito importante para que haja uma prevenção contra possíveis doenças, porém quando alcançamos a vida adulta esse cuidado é em sua maioria deixado de lado. Nos dias de hoje o controle de vacinação pessoal é realizado pelo cartão de vacina, que se trata de um papel que pode se perder em meio aos acontecimentos do dia a dia. Para este trabalho foi utilizado o modelo de desenvolvimento de *software* em cascata onde cada etapa é descrita neste documento. Foi utilizada a modelagem UML na criação dos diagramas para o entendimento das funcionalidades e requisitos do sistema. Ao final, é apresentado o *software* desenvolvido.

Palavras-chave: Vacinação. Saúde. Controle de vacinas.

LISTA DE FIGURAS

FIGURA 1 – Os clientes chamam o servidor individual	15
FIGURA 2 – Componentes MVC	17
FIGURA 3 – Servidor Node	22
FIGURA 4 – MEAN <i>Stack</i>	23
FIGURA 5 – Diagrama de Processo Consulta Cartão de Vacina	25
FIGURA 6 – Diagrama de Processo Cadastro de Vacinas	26
FIGURA 7 – Diagrama de Processo Consulta de Postos de Saúde	26
FIGURA 8 – Diagrama de Processo Cadastro de Dependentes	27
FIGURA 9 – Diagrama de Casos de Uso do CPVSIS	28
FIGURA 10 –Diagrama de Classes do CPVSIS	29
FIGURA 11 –Tela de Login	30
FIGURA 12 –Modal Esqueceu Senha	31
FIGURA 13 –Tela de Registro de Usuário 1	31
FIGURA 14 –Tela de Registro de Usuário 2	32
FIGURA 15 –Menu lateral	33
FIGURA 16 –Menu lateral	34
FIGURA 17 – <i>Form</i> de consulta de cartão de vacina	34
FIGURA 18 –Resultado de consulta de cartão de vacina	35
FIGURA 19 –Cadastro de Vacinas	35
FIGURA 20 –Lista de vacinas cadastradas	36
FIGURA 21 –Cadastro Dependentes	36
FIGURA 22 –Lista de Dependentes Cadastrados	37
FIGURA 23 –Consulta Posto de Saúde	37
FIGURA 24 –Informações Posto de Saúde	38
FIGURA 25 –Informações todos os Postos de Saúde	39
FIGURA 26 –Lista Solicitações Médicas Pendente	39
FIGURA 27 –Lista Solicitações Médicas Aceita	40
FIGURA 28 –Mensagem Solicitação Médica	40
FIGURA 29 –Menu Lateral Médico	40

FIGURA 30 –Form Consulta Paciente	41
FIGURA 31 –Lista Vacinas em Atraso	41
FIGURA 32 –Form Cadastro Acesso	42
FIGURA 33 –Lista Acessos Solicitados	42
FIGURA 34 –Menu Lateral Administrador	43
FIGURA 35 –Form Cadastro Médico	43
FIGURA 36 –Form Cadastro Notícia	44
FIGURA 37 –Lista Notícias	44
FIGURA 38 –Login Válido	45
FIGURA 39 –Login inválido	46
FIGURA 40 –Mensagem excluir vacina com sucesso	46
FIGURA 41 –Editar dados da vacina	47

LISTA DE TABELAS

TABELA 1 – Requisitos levantados	25
--	----

LISTA DE SIGLAS

ACID – Atomicidade, Consistência, Isolamento e Durabilidade

API – *Application Program Interface* (Interface de Programação de Aplicativos)

CAP – *Consistency, Availability e Partition tolerance* (Consistência, Disponibilidade e Tolerância à partição)

CSS – *Cascading Style Sheets* (Folhas de estilo em cascata)

CVPSIS – Sistema de Controle de Vacinação Pessoal

HTML – *Hypertext Markup Language* (Linguagem de marcação de hipertexto)

JSON – *JavaScript Object Notation* (Notação de objeto JavaScript)

MEAN – Mongo, Express, Angular e Node

MVC – *Model View Controller* (Modelo Visão Controlador)

NoSQL – *Not Only SQL* (Não apenas SQL)

OO – Orientado a Objetos

SGBD – Sistema de Gerenciamento de Banco de Dados

SPA – *Single Page Application* (Aplicativo de página única)

SUMÁRIO

1	INTRODUÇÃO	11
1.1	Motivação	11
1.2	Objetivos	12
1.2.1	Geral	12
1.2.2	Específicos	12
1.3	Organização do texto	12
2	REFERENCIAL TEÓRICO	13
2.1	Engenharia de <i>Software</i>	13
2.2	Arquitetura de <i>Software</i>	14
2.3	Aplicações Distribuídas	14
2.4	Aplicações Web	16
2.5	Arquitetura MVC	16
2.6	JavaScript	18
2.7	NoSQL - Not Only SQL	18
2.8	MEAN <i>Stack</i>	20
2.8.1	MongoDB	20
2.8.2	ExpressJS	21
2.8.3	Angular	21
2.8.4	NodeJS	21
2.8.5	O <i>Stack</i>	22
3	METODOLOGIA	24
3.1	Levantamento de Requisitos	24
3.2	Projeto de Sistema	24
3.2.1	Diagramas de Processo	25
3.2.2	Casos de Uso	27
3.2.3	Diagrama de Classes	28
4	IMPLEMENTAÇÃO	30
4.1	Tela de Login	30
4.2	Tela de Registro de Usuário	31
4.3	Tela principal	33
4.4	Tela Consulta de Cartão de Vacina	34
4.5	Tela Gestão de Dependentes	36
4.6	Tela Consulta Posto de Saúde	37
4.7	Tela Acesso Médico	39
4.8	Menu perfil Médico	40
4.9	Tela Consulta Paciente	41
4.10	Tela Consulta Vacinas em Atraso	41
4.11	Tela Acesso Paciente	42
4.12	Menu perfil Administrador	42
4.13	Tela Cadastro Médico	43

4.14 Tela Gestão de Notícias	44
5 TESTES	45
5.1 Login	45
5.1.1 <i>Logar no sistema com dados válidos</i>	45
5.1.2 <i>Logar no sistema com dados inválidos</i>	45
5.2 Cartão de Vacinas	46
5.2.1 <i>Cadastrar Vacinas</i>	46
5.2.2 <i>Excluir Vacinas</i>	46
5.2.3 <i>Editar Vacinas</i>	47
6 CONCLUSÃO E TRABALHOS FUTUROS	48
6.1 Dificuldades Encontradas	48
6.2 Trabalhos Futuros	48
REFERÊNCIAS	49

1 INTRODUÇÃO

Nos dias atuais a rotina corrida das pessoas tentando conciliar o tempo entre trabalho, família e na maioria das vezes estudos, faz com que não se tenha a devida atenção para alguns pontos importantes, dentre eles a saúde, mais especificamente o controle de vacinação. Enquanto estamos na infância os pais tendem a ser mais cautelosos quanto às vacinas a serem tomadas, e em qual data devem ser aplicadas, mas quando se chega na vida adulta a vacinação é deixada de lado. De acordo com SILVEIRA (2007) a vacinação infantil é muito importante para a prevenção de doenças imunopreveníveis além de evitar a ocorrência de surtos epidêmicos. Segundo MORAES (2018) a infectologista Rosana Richtmann do Hospital e Maternidade Santa Joana, em São Paulo, relata que em relação as crianças o Brasil está muito bem com a vacinação, mas quando se trata de adultos o cenário é diferente pois existe uma lacuna entre o que é oferecido e o que é realmente feito. Porém esse ano o cenário da vacinação infantil foi um pouco diferente.

A campanha de vacinação no Brasil não tem obtido os resultados esperados para o ano de 2018. No início deste ano o Sarampo, uma doença que já havia sido erradicada do país desde 2016, voltou a contaminar a população. Segundo G1 (2018) a vacinação contra o Sarampo foi abaixo do esperado em todo o país, e em 2017 21 estados não alcançaram a meta de imunizar 95% das crianças. Em 2018 foram registrados mais de mil casos de sarampo, sendo que a última vez que houve um registro de mesma proporção foi em 1999 com 908 casos confirmados.

1.1 Motivação

O controle pessoal de vacinação nos dias de hoje é realizado através do cartão de vacinação, que é uma cartela com as informações sobre as vacinas tomadas pelo indivíduo. A sua perda pode causar a inconsistência de informações pois na maioria das vezes a pessoa não se recorda das vacinas já administradas. No caso de perda do cartão de vacinas, as recomendações são procurar o posto de saúde no qual o indivíduo esta habituado a consultar e solicitar a segunda via, porém pode ocorrer de o histórico de vacinação não estar atualizado e não ser possível recuperar essas informações, sendo assim necessário colocar todas as vacinas em dia de acordo com a faixa etária (BRASIL, 2018).

Ainda sobre a perda do cartão de vacinas, caso o indivíduo não saiba qual a localização do posto de saúde, a obtenção dessa informação não se encontra disponível de maneira fácil no site do governo, tornando a recuperação das informações sobre o cartão de vacinas um processo mais demorado.

1.2 Objetivos

1.2.1 Geral

Criar uma aplicação Web com um foco em controle de vacinação pessoal, e fornecer informações úteis da área da saúde para o usuário.

1.2.2 Específicos

- Disponibilizar consulta online de cartão de vacinação por usuário.
- Criar um mecanismo de notificação para o usuário sobre vacinas em atraso.
- Disponibilizar consulta de dados do usuário para equipe médica autorizada.
- Disponibilizar serviço de consulta da localização postos de saúde de Belo Horizonte e região metropolitana.
- Criar um *feed* de notícias relacionadas com a área da saúde.

1.3 Organização do texto

O restante dessa monografia está organizada da seguinte forma: o capítulo 2 apresenta o referencial teórico. A metodologia é descrita no capítulo 3 e a implementação no capítulo 4. O capítulo 5 descreve os testes e o capítulo 6 descreve as conclusões e trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Engenharia de *Software*

A Engenharia de *Software* é uma área que estuda os processos envolvidos no desenvolvimento de um sistema e quais as formas de otimizá-lo. Segundo SOMMERVILLE (2007) é uma área que estuda as possibilidades de melhorar a qualidade de um *software*, e um *software* de qualidade é aquele que é fácil de usar, possui uma boa manutenibilidade, possui integridade dos dados, entre outras características relacionadas com a satisfação do usuário.

Segundo SOUZA (2015) a Engenharia de *Software* busca selecionar o método mais apropriado para um conjunto de circunstâncias e uma abordagem mais criativa e menos formal pode ser mais eficaz em outras circunstâncias. Tendo em vista que, quando se trata de desenvolvimento, a qualidade e produtividade devem estar unidas para que o sistema se torne algo viável no meio empresarial, e para que isso seja possível os Engenheiros de *Software* tendem a procurar uma solução ágil para o desenvolvimento do produto.

De acordo com FERNANDES (2004) dentro da Engenharia de *Software* existem 10 áreas com seus respectivos assuntos, são elas: Gerência de Configuração de *Software*, Gerência de Engenharia de *Software*, Processo de Engenharia de *Software*, Ferramentas e Métodos, Qualidade de *Software*, Requisitos de *Software*, Design de *Software*, Construção de *Software*, Teste de *Software*, Manutenção de *Software*.

A combinação dessas áreas ajudam no processo de criação de um *software*, como por exemplo a área de Requisitos de *Software* onde será realizada toda a parte de levantamento e análise de requisitos produz insumo para a área de Construção de *Software* onde irá, através de codificação, criar programas funcionais e com o auxílio de métodos definidos na área de Ferramentas e Métodos, com o objetivo de tornar a atividade de desenvolvimento de *software* sistemática e mais propensa ao sucesso. Segundo SOMMERVILLE (2011) os requisitos de um sistema são a descrição da tarefa a ser executada pelo sistema, e suas restrições funcionais. A partir dos requisitos deve-se ser capaz de executar as tarefas propostas.

Ao se utilizar técnicas de modelagem de processos é possível obter uma visão melhor do contexto onde o sistema será desenvolvido e como o mesmo deve funcionar, gerando assim a identificação de requisitos que reflitam as necessidades do negócio (BAKER, 2001).

2.2 Arquitetura de *Software*

De acordo com IEEE (2000) Arquitetura é a organização fundamental de um sistema incorporado em seus componentes, seus relacionamentos uns aos outros, e ao meio ambiente, e os princípios que guiam o seu design e evolução.

Segundo VERGILIO (2004) a Arquitetura de *Software* é importante para o processo de desenvolvimento de sistemas pois facilita o entendimento do projeto, uma vez que esta irá filtrar e formatar a informação, ou seja, é uma área da Engenharia de *Software* que trata de produzir as estruturas de *software*, visando a reduzir complexidade.

Ainda de acordo com VERGILIO (2004) a Arquitetura de *Software* auxilia na estimação de custos do projeto e gerência de complexidade do sistema, além de reduzir o intervalo entre especificação e implementação, e também reduz os riscos relacionados ao processo de construção do *software*.

Alguns padrões arquiteturais são conhecidos e utilizados amplamente pelos desenvolvedores e de acordo com VERGILIO (2005) são eles:

- MVC - *Model-View-Controller*
- Camadas
- *Microkernel*
- Chamada e Retorno
- Sequenciais *Batch*
- *Pipes & Filters*
- Centrado em Dados

2.3 Aplicações Distribuídas

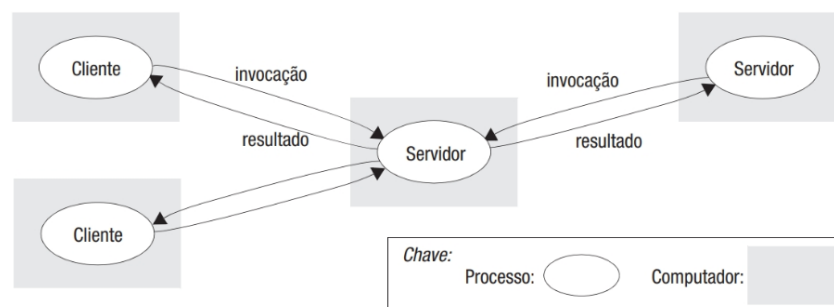
As aplicações distribuídas permitem que seus componentes estejam separados em vários ambientes de maneira heterogênea. Segundo TANENBAUM e STEEN (2008) podemos definir

as aplicações distribuídas como uma coleção de computadores independentes que se comporta perante o usuário como um único sistema consistente.

Segundo COULOURIS (2013) a arquitetura de um sistema é sua estrutura em termos de componentes especificados separadamente. O modelo de arquitetura de um sistema distribuído primeiro simplifica e abstrai as funções dos componentes individuais do sistema e depois considera o posicionamento dos componentes em uma rede de computadores, buscando determinar padrões para a distribuição dos dados e da carga de trabalho, e considera também o inter-relacionamento entre os componentes, ou seja, seus padrões funcionais e os padrões de comunicação.

A arquitetura Cliente-Servidor é a mais utilizada quando se trata de sistemas distribuídos. Nesse modelo os processos clientes integram com os processos servidores, que podem estar em computadores distintos, e acessam recursos compartilhados. Os servidores podem também assumir o papel de cliente, visto que podem realizar requisições para outros servidores como pode ser visto na figura 1. Embora o modelo cliente-servidor ofereça uma estratégia direta e relativamente simples para o compartilhamento de dados e de outros recursos, ele não é flexível em termos de escalabilidade.

Figura 1 – Os clientes chamam o servidor individual



Fonte: (COULOURIS, 2013)

Existem outros modelos de arquitetura que podem ser utilizados para a construção de sistemas distribuídos, mas em cada um haverá seus pontos positivos e negativos, cabendo aos desenvolvedores a tarefa de escolher aquele que melhor se adequa para a solução proposta.

2.4 Aplicações Web

As aplicações Web possuem seu espaço dentro da área da computação, devido ao sucesso e crescimento do uso ao longo dos anos, e também pelo aumento de recursos disponíveis nesse contexto. Essas aplicações utilizam tecnologias e linguagens mundialmente conhecidas como HTML(*Hypertext Markup Language*), JavaScript e CSS(*Cascading Style Sheets*), que servem como base estrutural para o ambiente Web.

Segundo FRATERNALLI e PAOLINI (1998) pelo fato de utilizarem da infra-estrutura Web, as aplicações Web passam a possuir características específicas, e devem considerar particularidades relacionadas às dimensões:

- Estrutural (conceitual): define a organização das informações a serem tratadas pela aplicação e os seus relacionamentos.
- Navegacional: representa como as informações serão acessadas através da aplicação.
- Apresentação: descreve como as informações e o acesso a essas serão apresentados ao usuário da aplicação.

Essas dimensões podem definir diferentes visões para o projeto da aplicação Web.

”Processos de desenvolvimento para aplicações de software Web devem produzir representações para projeto de aspectos de aplicações tradicionais, como estrutura e funcionalidades; e também para aspectos orientados para Web, como navegação e apresentação (com recursos Web)”(MENDES; CONTE; TRAVASSOS, 2005). Existem diferentes tipos de metodologias voltadas para o desenvolvimento de aplicações Web, e a construção de modelos específicos se torna uma tarefa relevante, entretanto não se pode definir um modelo padrão.

2.5 Arquitetura MVC

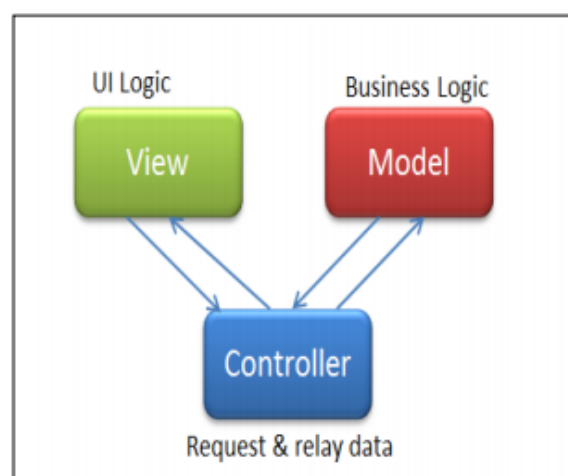
A Arquitetura MVC (*Model-View-Controller*) faz uma abordagem para a estruturação de uma aplicação, separando em módulos ou camadas suas partes principais, possibilitando um sistema com uma alta coesão e um baixo acoplamento entre as classes. A coesão dentro da visão do paradigma O.O. (Orientada a Objetos), fala que para que uma classe possua uma alta

coesão a mesma deve possuir responsabilidades e propósitos bem definidos. Já o acoplamento, também dentro da visão O.O. fala sobre o grau de dependência entre as classes de um sistema, quanto maior a dependência de uma classe sobre os métodos de outra, maior será esse grau de acoplamento e vice-versa.

Segundo ASTOLFI (2012) o modelo MVC possui três camadas principais:

- Camada de Visão: é o estágio transitório entre a interface e o sistema. É a camada que interage com o usuário, permitindo que o mesmo obtenha acesso às informações geradas pelo sistema, e para isso estão presentes nessa camada as páginas, as funcionalidades e protocolos de transferência de dados.
- Camada de Serviço: também conhecida como camada de controle, é o estágio que são tratadas as regras de negócio da aplicação. É de responsabilidade dessa camada conhecer as entidades do sistema, e caso haja algum tipo de erro a camada de controle deve tratá-lo da maneira correta. É a responsável pela comunicação entre a camada de visão e a camada de modelo.
- Camada de Modelo: É responsável por garantir a integridade dos dados, o modelo de relacionamento entre entidades e os atributos que compõem cada uma delas. É a base do sistema, é nessa camada que são declaradas todas as estruturas de dados, que pertencem ao sistema, permitindo então uma boa estruturação do código.

Figura 2 – Componentes MVC



Fonte: (SARKER; APU, 2014)

A figura 2 demonstra de uma maneira geral como é feita a comunicação entre as camadas da arquitetura MVC.

2.6 JavaScript

Segundo STIPKOVIC e BIANCHINI (2015) o JavaScript é uma linguagem de programação interpretada, que foi criada com o intuito de ajudar os navegadores Web a executar tarefas do lado cliente. Foi criada como parte do *Web Browser Netscape Navigator*. Quando terminou sua implementação a Netscape (atualmente Fundação Mozilla) submeteu em novembro de 1996 a especificação da linguagem JavaScript, que foi aceita e teve a definição do padrão chamada de ECMAScript. De acordo com MDN (2018) algumas das características do JavaScript são:

- Sem distinção entre tipos e objetos. A herança é feita através do protótipo e as propriedades e métodos podem ser adicionadas a qualquer objeto dinamicamente.
- No JavaScript a tipagem é realizada através da atribuição de valores, permitindo que uma mesma variável possa ser de diferentes tipos, ou seja, a mesma variável pode ser um inteiro mas posteriormente se tornar em uma *string*.
- Utiliza um mecanismo de protótipos para tratar heranças entre classes, com isso é possível se assemelhar com as características da programação orientada a objetos.
- A interação com o usuário é feita pelo HTML, através de eventos disparados pelos componentes da própria página Web.

2.7 NoSQL - Not Only SQL

O NoSQL é uma abordagem de banco de dados diferente dos SGBDs (Sistema de Gerenciamento de Banco de Dados) convencionais que já são conhecidos por todos. Essa abordagem é capaz de armazenar diferentes tipos de estruturas. Segundo BONFIOLI (2006) os bancos de dados NoSQL são ótimos para trabalhar com grande volume de dados distribuídos pois de acordo com LÓSCIO (2011) os bancos de dados NoSQL apresentam algumas características fundamentais que os diferenciam dos modelos de bancos de dados relacionais, permitindo que armazenem um grande volume de dados não estruturados ou semi-estruturados. Essas características são:

- Escalabilidade horizontal: com o aumento do volume de dados a necessidade de aumento de desempenho relacionado a banco de dados é inevitável, e como solução para essa situação temos a escalabilidade horizontal onde *threads*/processos são criados para gerenciar e distribuir o processamento de dados. Nesse contexto um banco de dados relacional não se encaixaria muito bem, pois os diferentes processos realizando requisições sobre o mesmo conjunto de dados causaria uma alta concorrência o que prejudicaria o desempenho da aplicação. Os banco de dados NoSQL não possuem essas barreiras pelo fato de utilizarem técnicas que permitem a distribuição dos dados, sendo uma delas é o *Sharding* que consiste em dividir os dados em múltiplas tabelas a serem armazenadas ao longo de diversos nós de uma rede. A utilização dessa técnica transmite a complexidade de juntar os dados que foram separados para a aplicação, sendo necessário a utilização de *joins* para conseguir realizar uma consulta completa.
- Ausência de esquema ou esquema flexível: os bancos de dado NoSQL tem ausência completa ou quase total do esquema que define a estrutura dos dados modelados, facilitando em relação a escalabilidade e aumento da disponibilidade, porém não garante a integridade dos dados armazenados.
- Suporte nativo a replicação: o processo de replicação reduz o tempo gasto para recuperar as informações armazenadas. Existem dois tipos de abordagens para a replicação a *Master-Slave*(Mestre-Escravo) e a *Multi-Master*. Na abordagem *Master-Slave* cada escrita no banco resulta em N escritas no total, onde N é o número de nós escravos, e a escrita é realizada apenas no nó mestre que e refeita nos nós escravos pelo nó mestre. Já na abordagem *Multi-Master* assumimos que existem vários nós mestres, diminuindo o gargalo em casos de grande número de acessos em um conjunto de dados.
- API simples para acesso aos dados: os bancos de dados NoSQL possuem o foco em recuperação e acesso rápido e eficiente dos dados armazenados, e para que isso seja possível as APIs devem ser desenvolvidas para facilitar esse acesso, permitindo que qualquer tipo de aplicação seja capaz de acessá-la.
- Consistência eventual: pelo fato de que os dados são armazenados de maneira distribuída em diferentes nós de uma rede, a consistência desses dados nem sempre é mantida em todos os nós. Isso se baseia em no teorema CAP (*Consistency, Availability e Partition tolerance*) onde fala que, em um determinado momento, só é possível garantir duas de três

propriedades entre consistência, disponibilidade e tolerância à partição. Em contrapartida as propriedades ACID(Atomicidade, Consistência, Isolamento e Durabilidade) não podem ser obedecidas simultaneamente. Tendo em vista esse ponto, a aplicação deve ser estruturada para tolerar inconsistência de dados com o intuito de garantir a disponibilidade.

2.8 MEAN Stack

Segundo BONFIM e LIANG (2014) o nome MEAN é um acrônimo das quatro tecnologias que compõe o MEAN Stack, MongoDB que é um banco de dados NoSQL, o *framework backend* ExpressJS, o *framework frontend* Angular e o servidor NodeJs. O MEAN surgiu em um debate de usuários em um grupo do LinkedIn.

2.8.1 MongoDB

O MongoDB é um banco de dados NoSQL de código aberto que é orientado a documentos. Esses documentos possuem o formato de JSON (*JavaScript Object Notation*) ou BSON que seria o JSON em formato binário, que são formatos muito utilizados para a troca de informações entre sistemas.

Segundo DB-Engines (2018) o MongoDB é o quinto banco de dados mais aceito pela comunidade de desenvolvimento, e como se trata de um banco de dados NoSQL possui um bom desempenho em relação a manipulação de grande volume de dados. Outros pontos fortes que justificam esse nível de aceitação são o fato de que trata diretamente com o formato JSON, o que facilita na configuração para comunicação *backend* da aplicação, e conseguir executar consultas em JavaScript.

2.8.2 *ExpressJS*

Segundo EXPRESS (2018) o Express é um *framework* para aplicativo da web do NodeJS mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos Web e móvel. Possui diversos métodos HTTP o que auxilia na criação de APIs.

2.8.3 *Angular*

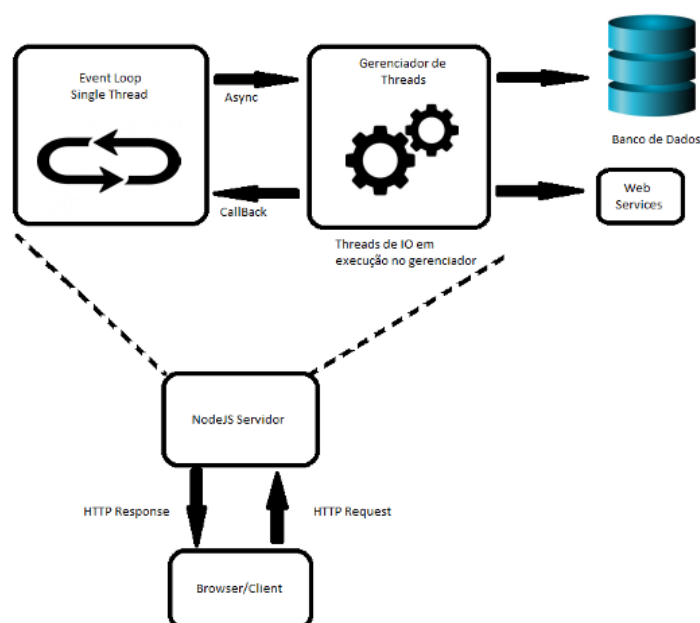
Segundo BONFIM e LIANG (2014) O Angular é um *framework frontend* que segue o modelo MVC e oferece um conjunto de ferramentas para a toda a criação e estilização de uma aplicação. Esse *framework* utiliza uma abordagem de SPA (*Single Page Application*), onde toda a aplicação é exibida em uma única página e seu conteúdo é carregado de acordo com as solicitações do usuário. Essa abordagem oferece um ganho no desempenho da aplicação, tendo em vista que não há a necessidade de recarregar toda a página a cada requisição feita, apenas partes da página são recarregados o que por sua vez diminui o tráfego de dados entre a aplicação e o servidor.

Outra abordagem utilizada pelo Angular é o *Two-Way Data Biding*, que facilita na exibição e manipulação dos dados carregados em tela, através de uma sincronização entre a tela e a camada de controle.

2.8.4 *NodeJS*

De acordo com PFÜTZENREUTER (2015) o NodeJs foi desenvolvido em 2009 com o intuito de executar programas JavaScript fora do contexto do *browser*. A comunidade de desenvolvedores já almejava que o JavaScript fosse uma linguagem de uso geral e o Node conseguiu atingir essa expectativa.

Figura 3 – Servidor Node



Fonte: Elaborado pelo autor

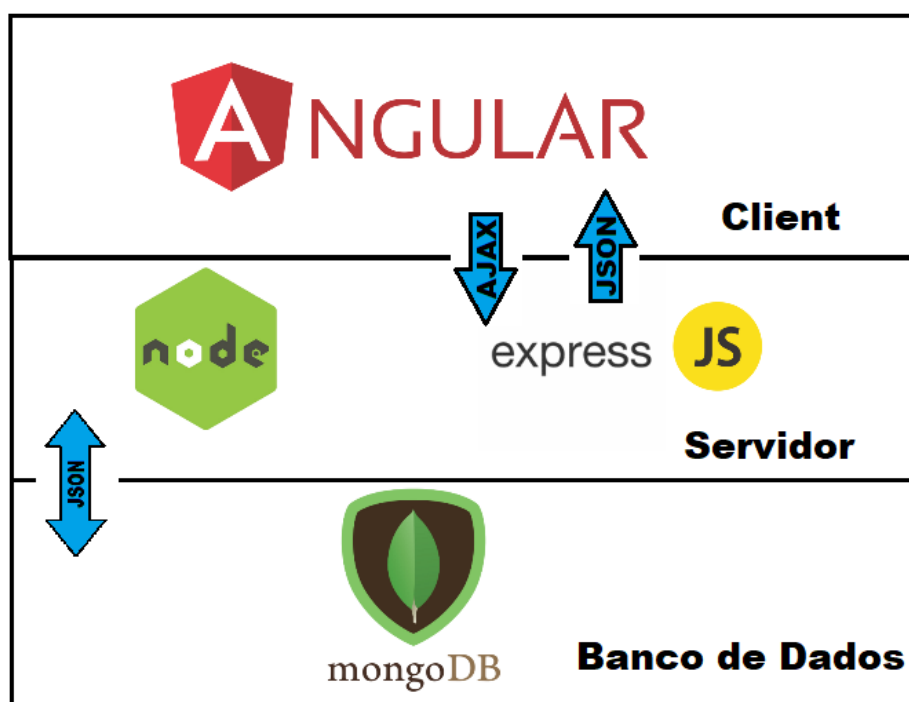
A figura 3 demonstra como é o funcionamento de um servidor Node. De acordo com CAMARGO (2018) o Node utiliza um modelo *single-thread* onde todas as requisições realizadas para o servidor são tratadas como eventos e de forma assíncrona, e são gerenciadas por uma única *thread*. Isso permite que dentro do servidor o Node gerencie as múltiplas requisições através de chamadas de E/S (entrada e saída) não-bloqueantes, ou seja as operações de entrada e saída (ex: acesso a banco de dados e leitura de arquivos do sistema) são assíncronas e não bloqueiam a *thread*. Devido a sua arquitetura um servidor Node possibilita um grande número de requisições concorrentes.

2.8.5 O Stack

O MEAN *Stack* utiliza essas tecnologias para criar toda a estrutura da aplicação, desde o lado *client* contendo o *layout* das páginas e as funcionalidades acessadas pelo usuário, até a camada de persistência de dados. A nomenclatura *Stack* é atribuída pela forma como é realizada a comunicação entre os componentes da aplicação, onde o *client* o servidor e o banco de dados se comunicam de uma forma hierárquica que lembra uma pilha.

A figura 4 mostra a estrutura do *MEAN Stack*. Através das interfaces, o usuário interage com a aplicação realizando requisições ao servidor Node, que por sua vez através do Express realiza a comunicação com o banco de dados. Durante essa comunicação é transportado objetos JSON entre o servidor e o banco de dados, seja pelo fato de ser realizado uma consulta, ou uma inserção. Após essa troca de informações com o banco de dados o servidor retorna um objeto JSON para a interface, que é exibido para o usuário.

Figura 4 – MEAN Stack



Fonte: Elaborado pelo autor

3 METODOLOGIA

O CVPSIS (Sistema de Controle de Vacinação Pessoal) foi desenvolvido utilizando o modelo cascata com as seguintes etapas: levantamento de requisitos, criação do projeto e arquitetura do sistema, implementação e validação através de testes. Essas etapas estão descritas a seguir.

O modelo cascata é um processo em sequencia onde a próxima etapa só é executada quando a etapa anterior é finalizada. Segundo SOMMERVILLE (2011) cada uma das etapas pode possuir documentos que serão utilizados na execução da próxima fase. O modelo consiste em 5 etapas: Definição de Requisitos, Projeto de Sistema, Implementação, Validação e Entrega e Manutenção.

Durante a execução dessas etapas segundo SOMMERVILLE (2011) podem ocorrer problemas nos resultados obtidos nas etapas anteriores, como problemas no levantamento de requisitos, problemas durante a implementação, o que pode levar a uma modificação do produto obtido entre as etapas, refletindo na execução da etapa seguinte.

Nas seções seguintes, serão detalhadas as etapas seguidas no CPVSIS.

3.1 Levantamento de Requisitos

Este trabalho apresenta a modelagem de requisitos utilizando notação UML. A seguir são apresentados os requisitos levantados nessa etapa.

3.2 Projeto de Sistema

Segundo SOMMERVILLE (2011) um projeto de *software* deve descrever a estrutura do *software* a ser implementado, os tipos de dados utilizados pelo sistema, seus componentes e em alguns casos os algoritmos utilizados. A seguir serão apresentadas as estruturas utilizadas para a execução deste trabalho.

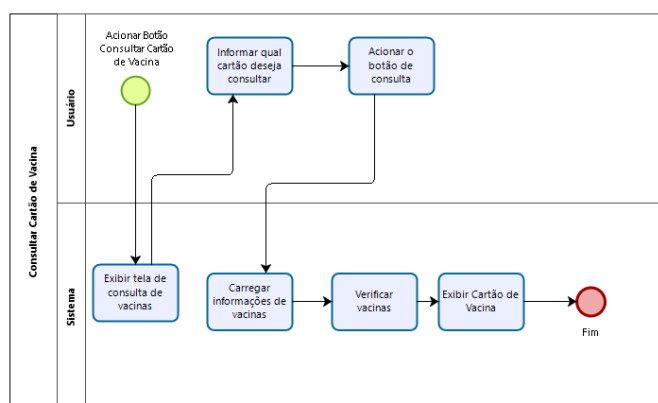
Tabela 1 – Requisitos levantados

Análise de Requisitos levantados	
Requisito	Descrição
Login	- O usuário deve logar no sistema para poder utilizá-lo.
Cartão de Vacina	- O usuário deve ser capaz de consultar o cartão de vacina cadastrado. - O usuário deve ser capaz de consultar o cartão de vacina cadastrado de seus dependentes. - O usuário deve ser capaz de gerenciar as vacinas suas próprias vacinas. - O usuário deve ser capaz de gerenciar as vacinas de seus dependentes.
Dependentes	- O usuário deve ser capaz de gerenciar os seus dependentes.
Postos de Saúde	- O usuário deve ser capaz de consultar informações e localização dos postos de saúde da região metropolitana de Belo Horizonte.
Notícias	- O usuário Administrador deve ser capaz de gerenciar o <i>feed</i> de notícias.
Consulta Paciente	- O usuário Médico deve ser capaz de consultar o cartão de vacinas de um Paciente autorizado. - O usuário Médico deve ser capaz de consultar Pacientes com vacinas em atraso.

Fonte: Elaborado pelo autor

3.2.1 Diagramas de Processo

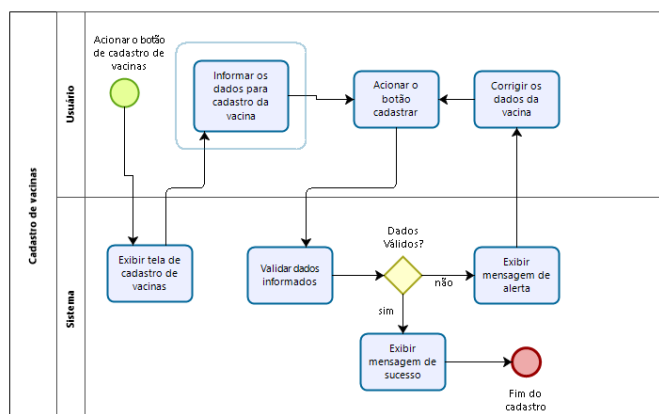
A figura 5 apresenta o fluxo executado pelo sistema no processo de consulta do cartão de vacinas do usuário. Ao acionar o botão de consulta de cartão de vacinas, o usuário é redirecionado para a tela de consulta. O usuário informa se deseja consultar o próprio cartão de vacinas ou se deseja consultar o cartão de um dependente, e então executa a consulta que retorna as vacinas já cadastradas.

Figura 5 – Diagrama de Processo Consulta Cartão de Vacina

Fonte: Elaborado pelo autor

A figura 6 apresenta o fluxo executado pelo sistema no processo de cadastro de vacina. Ao acionar o botão de cadastro de vacinas, o sistema exibe o formulário para preenchimento, após acionar o botão cadastrar o sistema valida os dados informados, caso estejam inválidos é exibido uma mensagem de alerta para o usuário, caso contrário é exibido a mensagem de sucesso do cadastro.

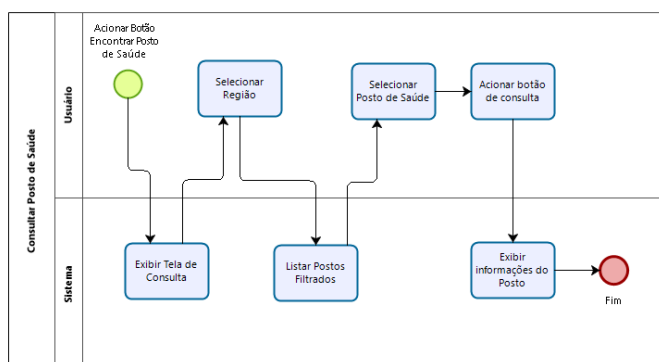
Figura 6 – Diagrama de Processo Cadastro de Vacinas



Fonte: Elaborado pelo autor

A figura 7 apresenta o fluxo de consulta de posto de saúde. Ao acionar o botão de consulta de postos de saúde o sistema exibe a tela de consulta, o usuário informa a região que deseja pesquisar, então o sistema carrega os postos de saúde referente a mesma. Ao selecionar o posto de saúde e acionar o botão de consulta, o sistema exibe as informações do mesmo.

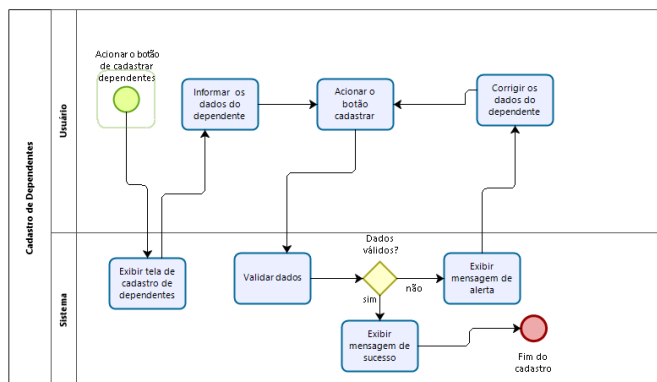
Figura 7 – Diagrama de Processo Consulta de Postos de Saúde



Fonte: Elaborado pelo autor

A figura 8 apresenta o fluxo executado pelo sistema no processo de cadastro de dependentes. Ao acionar o botão de cadastro de dependentes, o sistema exibe o formulário para preenchimento, após acionar o botão cadastrar o sistema valida os dados informados, caso estejam inválidos é exibido uma mensagem de alerta para o usuário, caso contrário é exibido a mensagem de sucesso do cadastro.

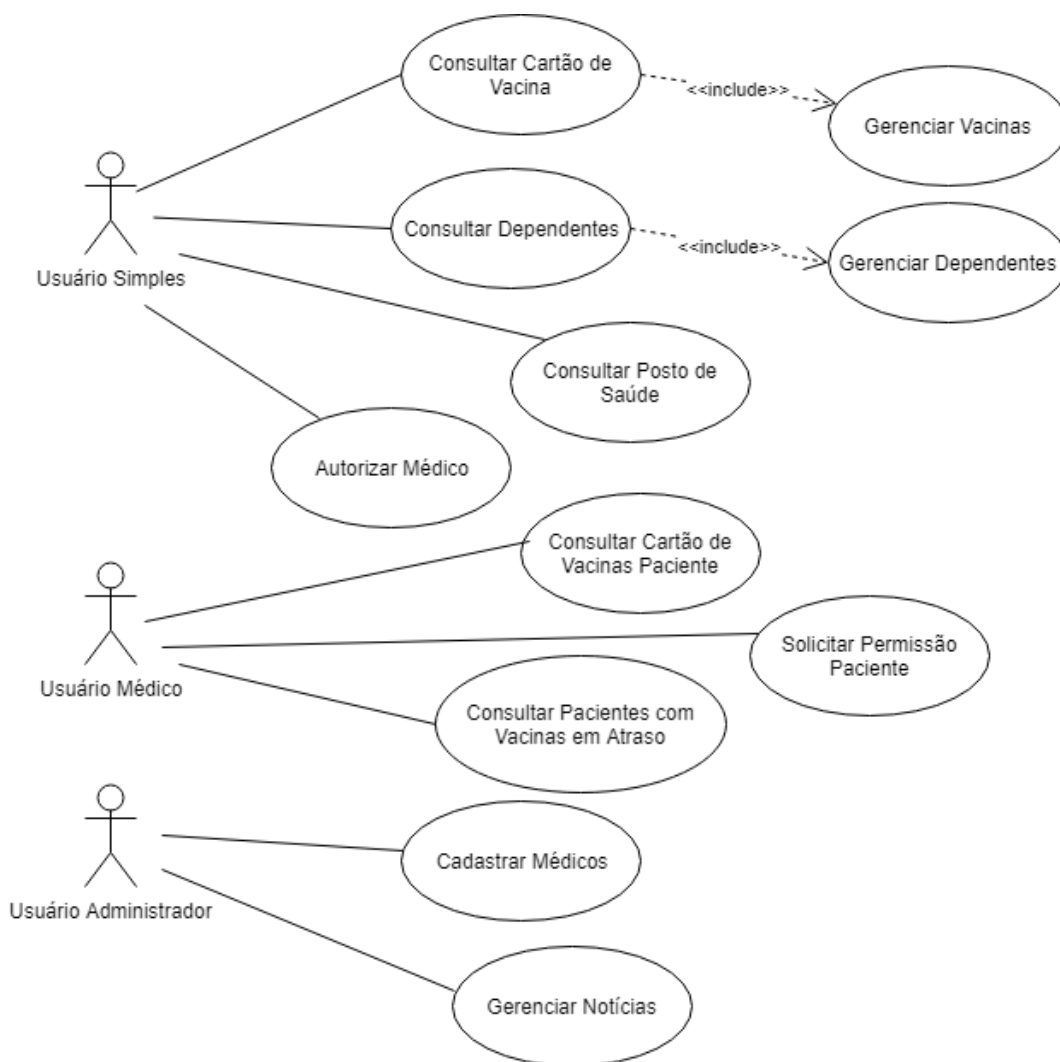
Figura 8 – Diagrama de Processo Cadastro de Dependentes



Fonte: Elaborado pelo autor

3.2.2 Casos de Uso

A figura 9 exibe os principais casos de uso e os atores que interagem com o CVPSIS. O Usuário Simples é o usuário que se cadastrou no sistema para poder utilizar suas funcionalidades de gestão de vacinas, e consulta de informações dos postos de saúde da região metropolitana de Belo Horizonte, e também é considerado um paciente para o Usuário Médico. Esse diagrama mostra os casos de uso de Consultar Cartão de Vacina, Gerenciar Vacinas, Consultar Dependentes, Gerenciar Dependentes, Consultar Posto de Saúde e Autorizar Médico, sendo todos acessados pelo Usuário Simples. O Usuário Médico é o usuário cadastrado por um Usuário Administrador e acessa o sistema para realizar a consulta das vacinas dos pacientes autorizados e interage com os casos de uso de Consultar Cartão de Vacinas Paciente, Solicitar Permissão Paciente e Consultar Pacientes com Vacinas em Atraso. O Usuário Administrador acessa o sistema para gerenciar as notícias cadastradas e cadastrar Usuários Médicos e interage com os casos de uso de Cadastrar Médicos e Gerenciar Notícias.

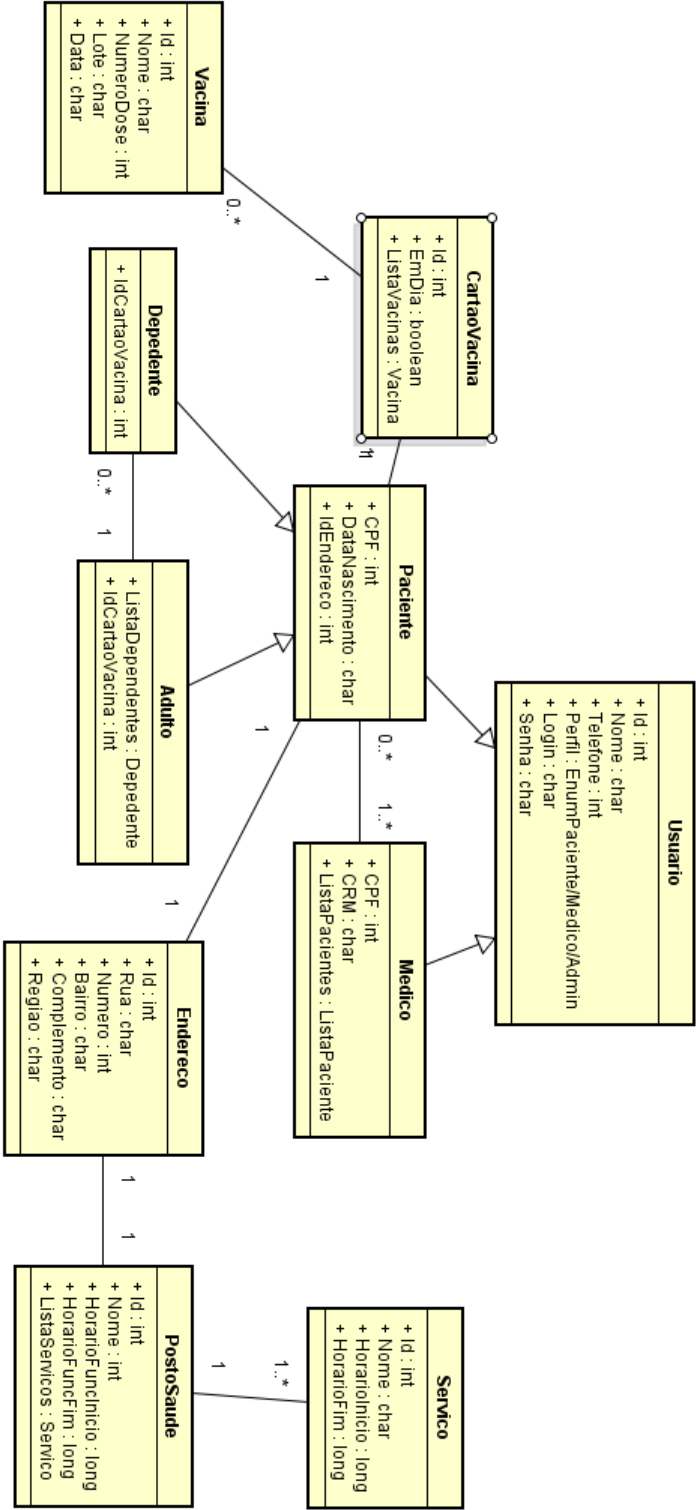
Figura 9 – Diagrama de Casos de Uso do CPVSI

Fonte: Elaborado pelo autor

3.2.3 Diagrama de Classes

A figura 10 mostra as entidades utilizadas pelo sistema. A entidade Usuário representa o usuário que acessa o sistema, com seus dados de acesso. As entidades Paciente e Médico são especializações da classe usuário e cada uma possui seus atributos específicos. A entidade CartaoVacina representa o cartão de vacina de cada paciente que contém uma lista de vacinas representadas pela entidade Vacina. A entidade PostoSaude representa os postos de saúde que podem ser consultados dentro do sistema, que possui uma lista de serviços representado pela entidade Servico. A entidade Endereco é compartilhada entre a entidade PostoSaude e a entidade Paciente, pois ambos possuem um endereço associado.

Figura 10 – Diagrama de Classes do CPVSIS



Fonte: Elaborado pelo autor

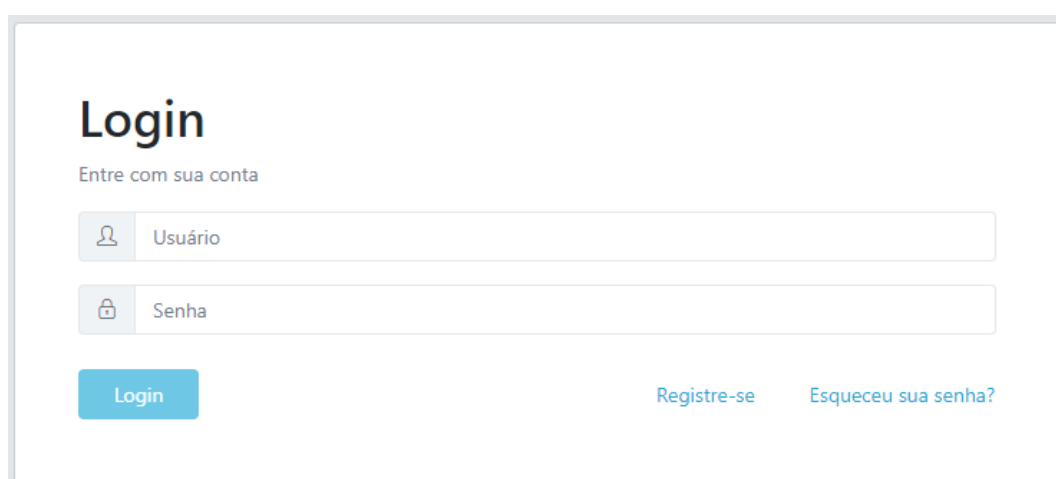
4 IMPLEMENTAÇÃO

Neste capítulo são apresentadas as telas do sistema e um descritivo de como foi elaborada cada interface.

4.1 Tela de Login

Quando o sistema é carregado, ou quando o usuário aciona o botão “Deslogar”, é apresentado a tela de login conforme a figura 11.

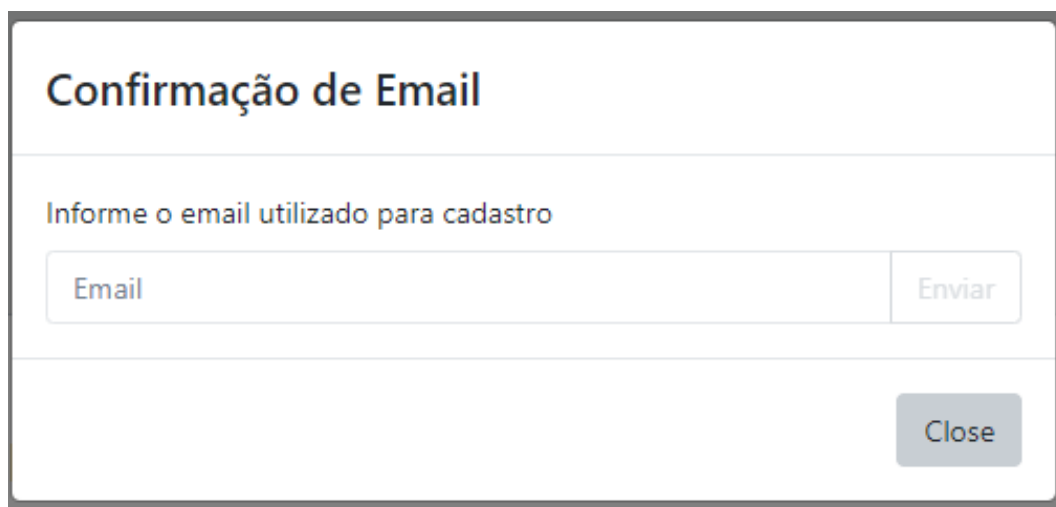
Figura 11 – Tela de Login

A imagem mostra a interface de login de um sistema web. No topo, o título "Login" é exibido em uma fonte grande e escura. Abaixo dele, o texto "Entre com sua conta" aparece em uma fonte menor e cinza. Há dois campos de entrada: o primeiro, rotulado "Usuário" com um ícone de pessoa, e o segundo, rotulado "Senha" com um ícone de cadeado. Abaixo dos campos, há um botão azul com o texto "Login". À direita do botão, há dois links em azul: "Registre-se" e "Esqueceu sua senha?".

Fonte: Elaborado pelo autor

A tela de login é dividida em 3 etapas, seguindo as seguintes interações:

- O usuário informa login e senha cadastrados na base de dados do sistema e aciona o botão “Login”, é realizado a validação e redireciona para a tela principal do sistema. Caso seja informado login ou senha inválidos, é exibido um alerta no navegador informado o usuário que os dados estão incorretos.
- O usuário aciona o botão “Esqueceu sua senha?” e é exibido uma modal solicitando o e-mail cadastrado no sistema para envio de instruções de recuperação de senha conforme a figura 12.
- O usuário aciona o botão “Registrar-se” que redireciona para a tela de Registro de Usuário.

Figura 12 – Modal Esqueceu Senha

A screenshot of a modal window titled "Confirmação de Email". Inside the modal, there is a heading "Informe o email utilizado para cadastro". Below this, there is a text input field with the placeholder text "Email" and a button labeled "Enviar" to its right. At the bottom right of the modal, there is a button labeled "Close".

Fonte: Elaborado pelo autor

4.2 Tela de Registro de Usuário

Quando o usuário aciona o botão “Registrar-se” na tela de login é exibido a tela de registro de usuário conforme a figura 13

Figura 13 – Tela de Registro de Usuário 1

A screenshot of a registration form titled "Registrar-se". Below the title is the subtitle "Crie sua conta". The form contains four input fields, each with an icon on the left: a person icon for "Nome de usuário", an @ symbol for "Email", a lock icon for "Senha", and a lock icon for "Repita a senha". At the bottom of the form is a large green button labeled "Continuar".

Fonte: Elaborado pelo autor

A tela de registro de usuário é dividida em 2 etapas, executadas em ordem e com as seguintes interações:

- O usuário informa os dados solicitados pelo sistema, e aciona o botão “Continuar”, o sis-

tema valida se o nome de usuário informado está disponível e se a senha e a confirmação da senha estão iguais. Caso o nome de usuário esteja indisponível, é exibido um alerta informando o usuário que o nome de usuário já está sendo utilizado. Caso a senha e a confirmação da senha não estejam iguais, é exibido um alerta informando o usuário que a senha e a confirmação da mesma não conferem.

- O usuário informa os dados solicitados pelo sistema e aciona o botão “Registrar-se” conforme a figura 14. Caso algum campo não seja informado, o sistema exibe um alerta informando o usuário que um campo não foi preenchido. Quando o registro é completado é exibido um alerta informando o usuário que o registro foi realizado com sucesso e redireciona para a tela de login.

Figura 14 – Tela de Registro de Usuário 2



O formulário, intitulado "Informe seus Dados", é dividido em seções para coleta de dados pessoais e de endereço. A seção "Dados pessoais" contém campos para Nome, Sobrenome, CPF e Data de Nascimento (formato dd/mm/aaaa). A seção "Endereço" contém campos para Celular, Rua, Bairro, Número, Complemento e uma lista suspensa. Um botão verde "Registrar-se" está no final do formulário.

Informe seus Dados

Dados pessoais

Nome

Sobrenome

CPF

Data de Nascimento:

dd/mm/aaaa

Celular

Endereço

Rua

Bairro

Número

Complemento

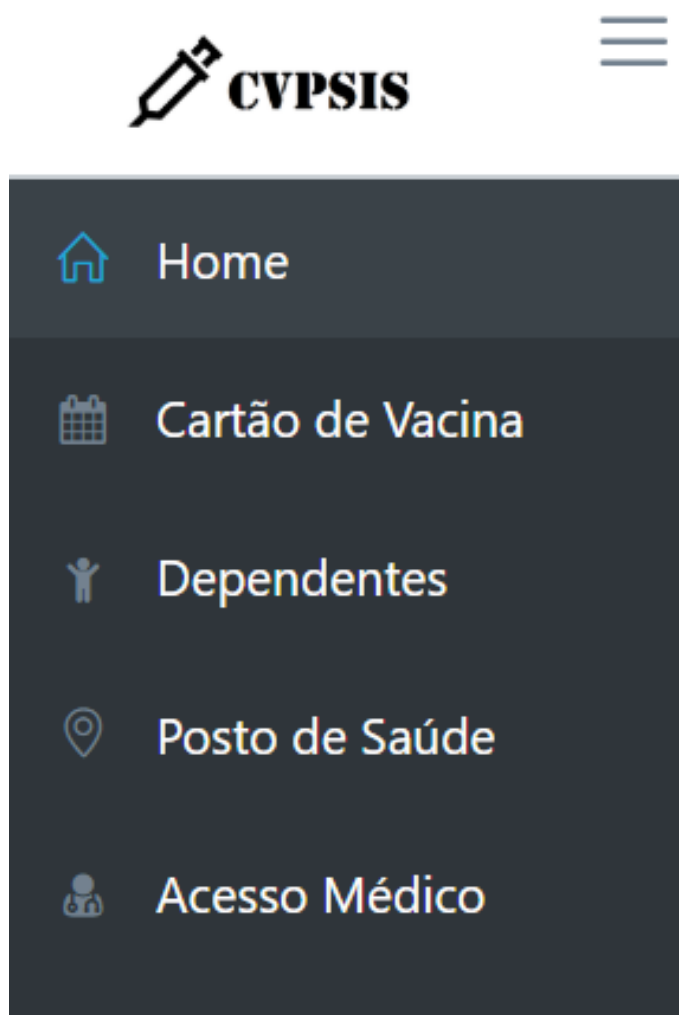
Registrar-se

Fonte: Elaborado pelo autor

4.3 Tela principal

Quando o usuário entra na tela principal é exibido o menu lateral que permite a navegação dentro do sistema de acordo com o perfil de usuário logado conforme mostra a figura 15.

Figura 15 – Menu lateral



Fonte: Elaborado pelo autor

Ao acionar um botão no menu o usuário é redirecionado para a tela respectiva da funcionalidade.

Na tela principal do sistema há um *feed* de notícias onde é exibido as notícias cadastradas pelo Administrador conforme mostra a figura 16.

Figura 16 – Menu lateral

Fonte: Elaborado pelo autor

4.4 Tela Consulta de Cartão de Vacina

Ao acionar no menu lateral o botão “Cartão de Vacina” o usuário é redirecionado para a tela de consulta do cartão de vacina. A tela possui um form de consulta que permite o usuário consultar o próprio cartão de vacina ou o de seus dependentes conforme a figura 17.

Figura 17 – Form de consulta de cartão de vacina

O formulário de consulta de cartão de vacina possui dois campos principais: "Eu mesmo?" com um botão de alternância (checkbox) e "Dependente:" com um menu suspenso rotulado "Selecione o dependente". Abaixo desses campos, há dois botões: "Consultar" e "Cadastrar".

Fonte: Elaborado pelo autor

O form de consulta possui 3 passos com as seguintes interações:

- O usuário seleciona o *checkbox* informando que deseja consultar o próprio cartão de vacina, e aciona o botão “Consultar”, então é apresentado o resultado com as vacinas do usuário conforme mostra a figura 18. Caso o usuário possua vacinas que não foram cadastradas, essas serão exibidas na cor vermelha.
- O usuário desmarca o *checkbox* e seleciona no combo informando que deseja consultar o cartão de vacina de um dependente e aciona o botão “Consultar”, então é apresentado o resultado com as vacinas do dependente. Caso o dependente possua vacinas que não foram cadastradas, essas serão exibidas na cor vermelha. Caso o usuário não possua

dependentes cadastrados, o *checkbox* fica desabilitado sendo possível apenas a consulta de cartão de vacinas do próprio usuário.

- O usuário aciona o botão “Cadastrar” e é exibido o *form* para cadastrar uma vacina conforme a figura 19. O cadastro é de acordo com a seleção do *checkbox* ou do combo de dependentes. Caso o usuário já possua vacinas cadastradas é exibido uma lista permitindo a edição e exclusão das vacinas conforme a figura 20. O *dropdown* de vacinas contém apenas 10 tipos de vacinas que estão cadastradas no banco de dados.

Figura 18 – Resultado de consulta de cartão de vacina

Nome Paciente: teste teste			
Nome: Antimeningocócica C conjugada Lote: ABC123 Dose: 1 Data: 01/10/2018	Nome: Antipneumocócica 10 valente conjugada Lote: Dose: Data:	Nome: BCG-ID Lote: Dose: Data:	Nome: DTP(tríplice bacteriana) Lote: Dose: Data:
Nome: Febre Amarela Lote: Dose: Data:	Nome: Hepatite B Lote: Dose: Data:	Nome: SRC (tríplice viral, MMR) Lote: Dose: Data:	Nome: Tetravalente (DTP + Hib) Lote: Dose: Data:
Nome: VOP(vacina oral contra a poliomielite - Sabin) Lote: Dose: Data:	Nome: VORH (vacina oral contra rotavírus humano) Lote: Dose: Data:		

Fonte: Elaborado pelo autor

Figura 19 – Cadastro de Vacinas

Selecione a vacina

Data de vacinação:
dd/mm/aaaa

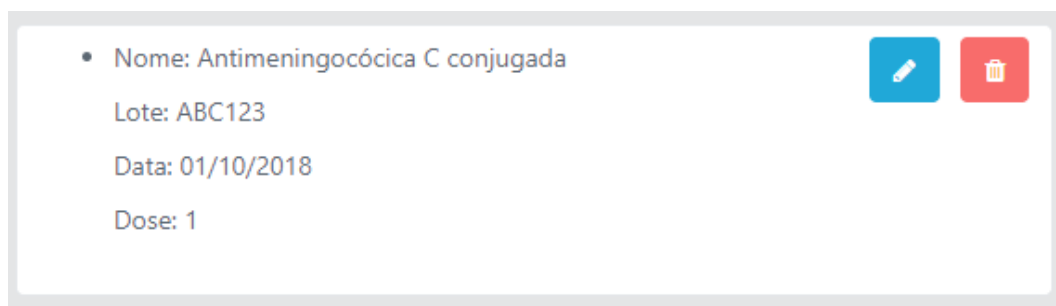
Lote:
Lote vacina

Dose:
Dose vacina

Caso seja dose única informe 0.

Salvar

Fonte: Elaborado pelo autor

Figura 20 – Lista de vacinas cadastradas

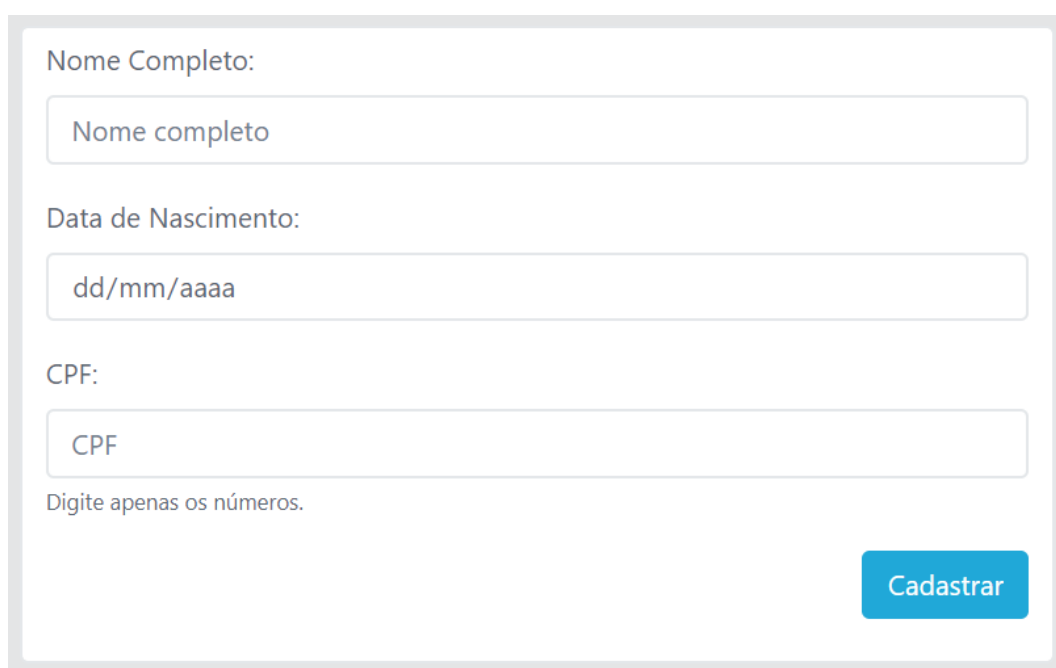
A screenshot of a vaccine list entry. It shows a single item with the following details: Name: Antimeningocócica C conjugada, Lot: ABC123, Date: 01/10/2018, and Dose: 1. To the right of the text are two icons: a blue square with a white pencil (edit) and a red square with a white trash can (delete).

- Nome: Antimeningocócica C conjugada
- Lote: ABC123
- Data: 01/10/2018
- Dose: 1

Fonte: Elaborado pelo autor

4.5 Tela Gestão de Dependentes

Ao acionar o botão “Dependentes” o usuário é redirecionado para a tela de Gestão de Dependentes. A tela possui um *form* de cadastro de dependentes que permite o usuário vincular um dependente a ele mesmo conforme mostra a figura 21.

Figura 21 – Cadastro Dependentes

A screenshot of a dependent registration form. It contains four input fields: 'Nome Completo:' with placeholder text 'Nome completo', 'Data de Nascimento:' with placeholder text 'dd/mm/aaaa', and 'CPF:' with placeholder text 'CPF'. Below the CPF field is a note: 'Digite apenas os números.' At the bottom right is a blue button labeled 'Cadastrar'.

Nome Completo:

Nome completo

Data de Nascimento:

dd/mm/aaaa

CPF:

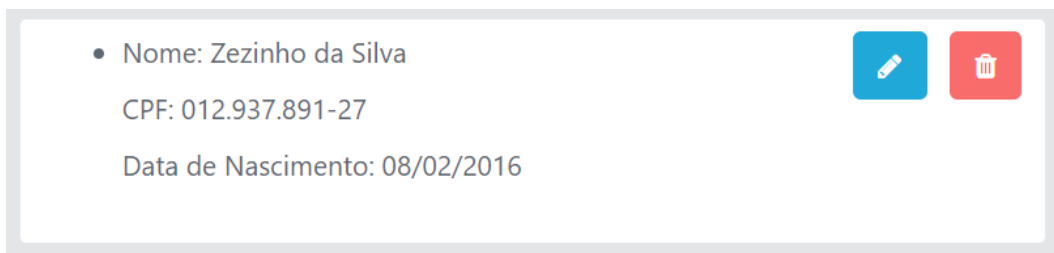
CPF

Digite apenas os números.

Cadastrar

Fonte: Elaborado pelo autor

Ao acionar o botão do *form* “Cadastrar” o sistema vincula o dependente ao usuário e lista logo a baixo as informações do dependente conforme a figura 22

Figura 22 – Lista de Dependentes Cadastrados

• Nome: Zezinho da Silva

CPF: 012.937.891-27

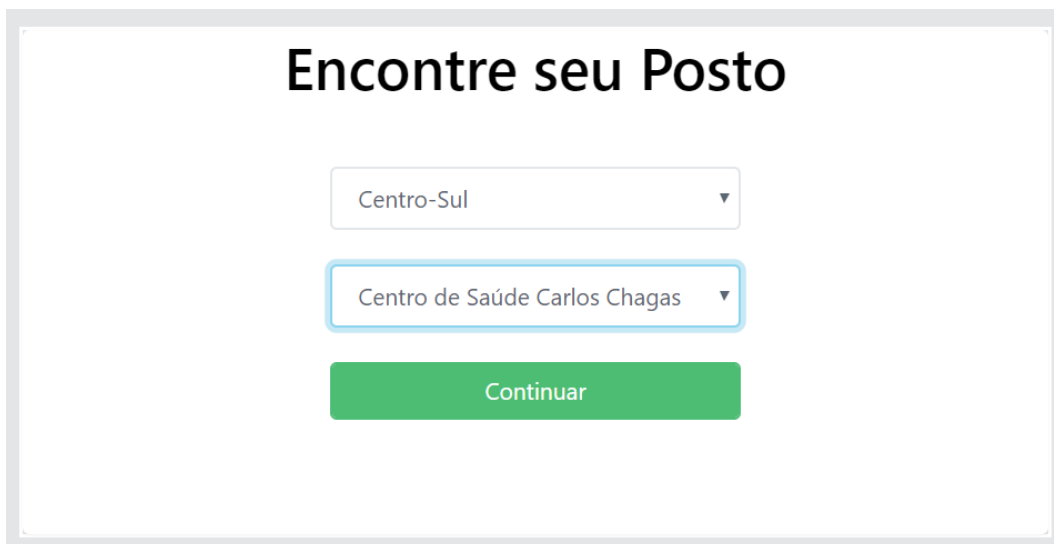
Data de Nascimento: 08/02/2016

Fonte: Elaborado pelo autor

Na lista de dependentes é possível o usuário editar os dados do dependente e excluir o mesmo.

4.6 Tela Consulta Posto de Saúde

Ao acionar o botão “Posto de Saúde” o usuário é redirecionado para a tela de consulta de posto de saúde, onde é exibido um *form* para a consulta contendo um *dropdown* das regiões de Belo Horizonte e um *dropdown* com os nomes dos postos de saúde conforme a figura 23.

Figura 23 – Consulta Posto de Saúde

Encontre seu Posto

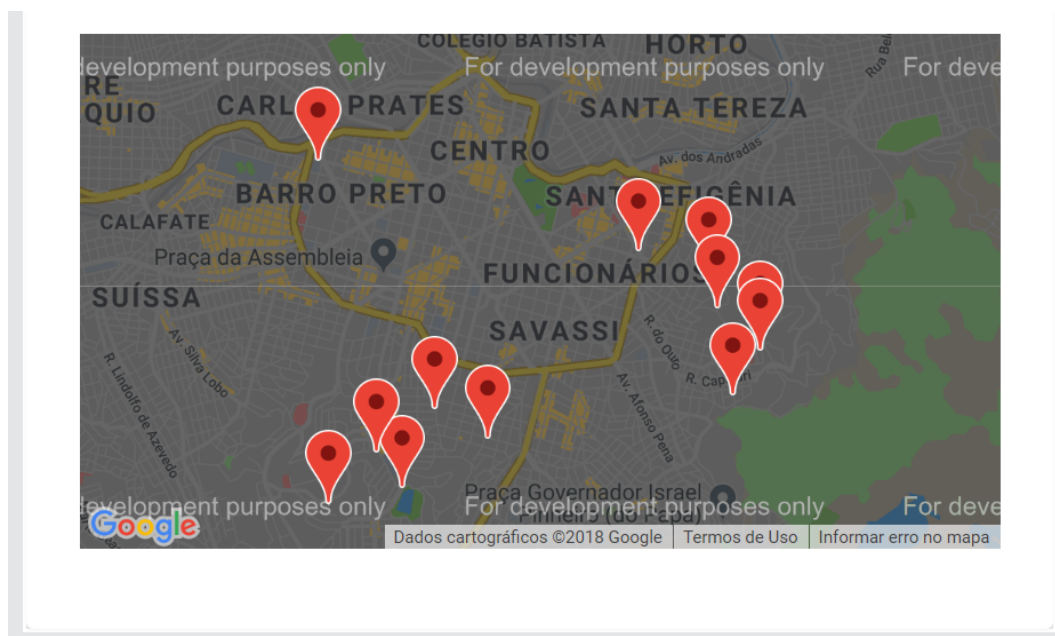
Centro-Sul ▼

Centro de Saúde Carlos Chagas ▼

Continuar

Fonte: Elaborado pelo autor

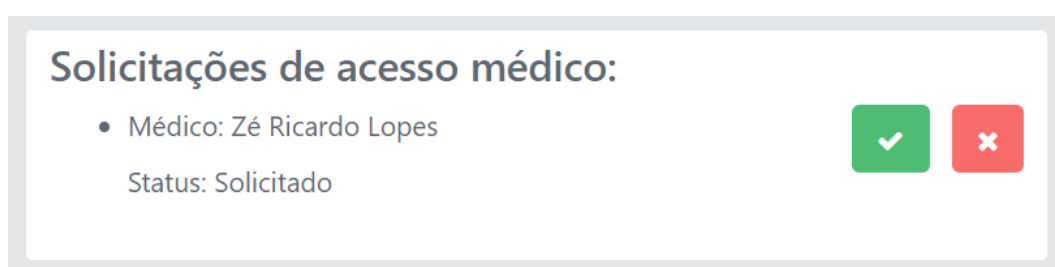
O *dropdown* com os nomes dos postos de saúde só é habilitado após a seleção da região, e o botão “Continuar” só é habilitado após a seleção da região e do posto de saúde. Após acionar o botão “Continuar” é exibido informações sobre o posto de saúde selecionado conforme mostra a figura 24.

Figura 25 – Informações todos os Postos de Saúde

Fonte: Elaborado pelo autor

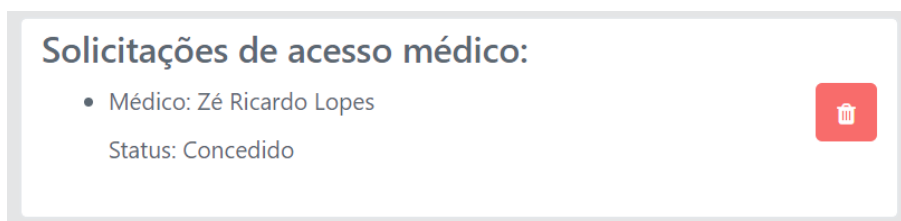
4.7 Tela Acesso Médico

Ao acionar o botão “Acesso Médico” o usuário é redirecionado para a tela de gestão de acesso médico. Na tela é exibido uma lista com as solicitações de acesso médico cadastradas conforme a figura 26.

Figura 26 – Lista Solicitações Médicas Pendente

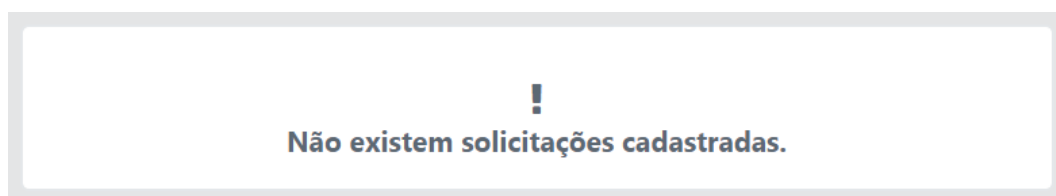
Fonte: Elaborado pelo autor

O usuário pode aceitar a solicitação ou recusar, caso seja aceita é cadastrado no banco a permissão para o médico acessar as informações do usuário, e caso seja recusada a solicitação é excluída. Após o usuário aceitar a solicitação o status da mesma é alterado e é exibido o botão de exclusão, caso o usuário deseje retirar a permissão de acesso do médico conforme mostra a figura 27.

Figura 27 – Lista Solicitações Médicas Aceita

Fonte: Elaborado pelo autor

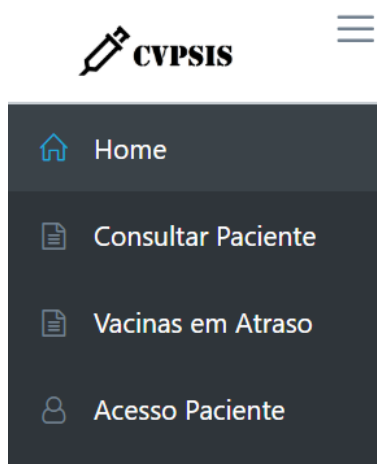
Caso não haja solicitações cadastradas será exibido uma mensagem para informar o usuário conforme a figura 28.

Figura 28 – Mensagem Solicitação Médica

Fonte: Elaborado pelo autor

4.8 Menu perfil Médico

Quando o usuário logado for um médico é exibido um menu diferente conforme a figura 29.

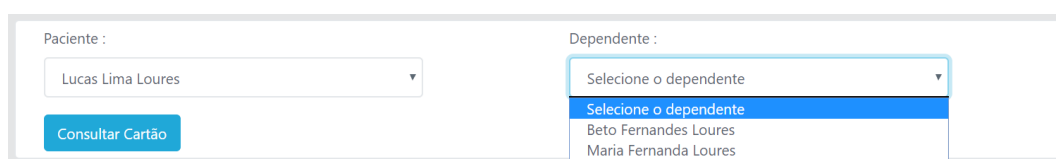
Figura 29 – Menu Lateral Médico

Fonte: Elaborado pelo autor

4.9 Tela Consulta Paciente

Ao acionar o botão “Consultar Paciente” o usuário é redirecionado para a tela de consultar o cartão de vacinas. As informações do cartão de vacina são de pacientes previamente autorizados. A consulta é feita pelo *form* de conforme a figura 30.

Figura 30 – Form Consulta Paciente



The image shows a web form titled "Form Consulta Paciente". It contains two dropdown menus. The first, labeled "Paciente:", has "Lucas Lima Loures" selected. The second, labeled "Dependente:", has "Selecione o dependente" selected, and a dropdown menu is open showing three options: "Selecione o dependente" (highlighted in blue), "Beto Fernandes Loures", and "Maria Fernanda Loures". Below the "Paciente:" dropdown is a blue button labeled "Consultar Cartão".

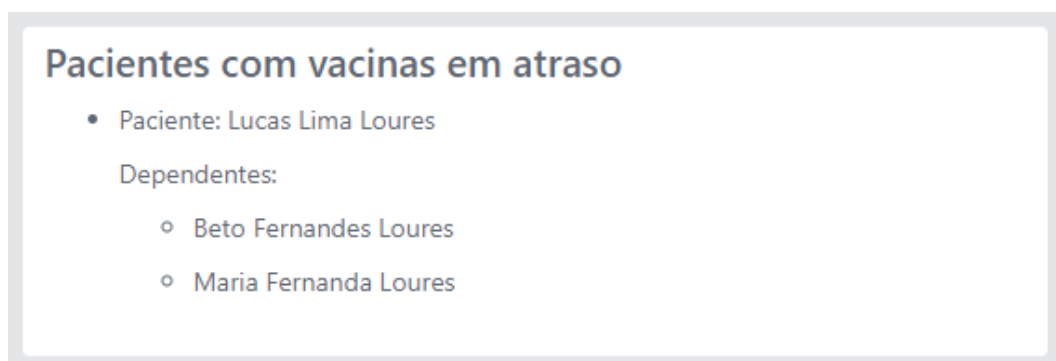
Fonte: Elaborado pelo autor

Após selecionar qual paciente deseja consultar é exibido o mesmo cartão de vacina quando o próprio paciente realiza a consulta conforme a figura 18.

4.10 Tela Consulta Vacinas em Atraso

Ao acionar o botão “Vacinas em Atraso” o usuário é redirecionado para a tela de consulta de pacientes autorizados que possuem vacinas em atraso. Ao acessar a tela é exibido uma lista com os nomes dos pacientes e seus dependentes, caso estes possuam vacinas em atraso conforme mostra a figura 31.

Figura 31 – Lista Vacinas em Atraso



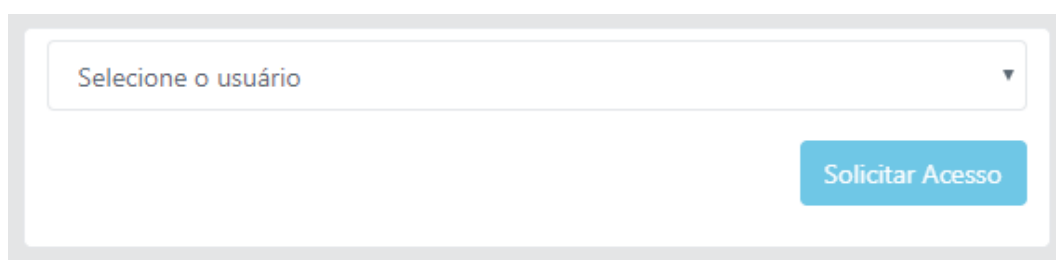
The image shows a web page titled "Pacientes com vacinas em atraso". It contains a list of patients and their dependents. The first patient is "Paciente: Lucas Lima Loures". Below this, under the heading "Dependentes:", there are two entries: "Beto Fernandes Loures" and "Maria Fernanda Loures".

Fonte: Elaborado pelo autor

4.11 Tela Acesso Paciente

Ao acionar o botão “Acesso Paciente” o usuário é redirecionado para a tela de solicitação de acesso aos pacientes. O usuário seleciona um paciente que ainda não cadastrou uma solicitação conforme a figura 32.

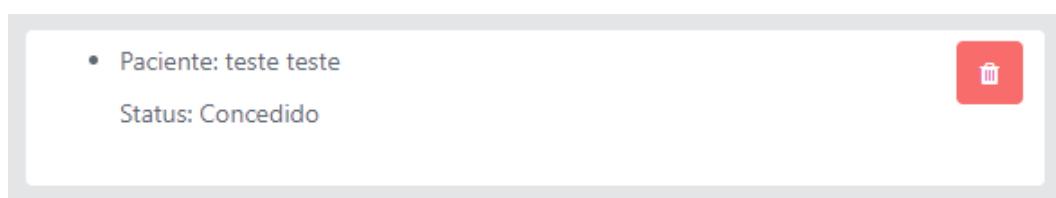
Figura 32 – Form Cadastro Acesso

A interface de usuário para o formulário de cadastro de acesso. Ela contém um campo de seleção com o texto "Selecione o usuário" e uma seta para baixo no canto direito. Abaixo do campo, há um botão azul com o texto "Solicitar Acesso".

Fonte: Elaborado pelo autor

É exibido também uma lista com as solicitações cadastradas e o status da solicitação, onde o usuário pode excluir as mesmas conforme a figura 33.

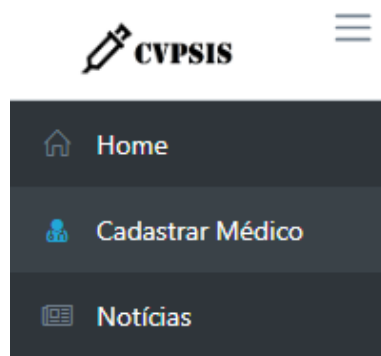
Figura 33 – Lista Acessos Solicitados

A interface de usuário para a lista de acessos solicitados. Ela mostra uma única entrada com o texto "Paciente: teste teste" e "Status: Concedido". À direita da entrada, há um ícone de lixeira em um botão vermelho para excluir a solicitação.

Fonte: Elaborado pelo autor

4.12 Menu perfil Administrador

Quando o usuário logado for um Administrador é exibido outro tipo de menu conforme a figura 34.

Figura 34 – Menu Lateral Administrador

Fonte: Elaborado pelo autor

4.13 Tela Cadastro Médico

Ao acionar o botão “Cadastrar Médico” o usuário é redirecionado para a tela de cadastro médico, onde é exibido o *form* para inserir as informações de cadastro conforme a figura 35.

Figura 35 – Form Cadastro MédicoA imagem mostra o formulário de cadastro médico. O formulário é dividido em duas seções: "Dados da conta" e "Dados pessoais e médicos". A seção "Dados da conta" contém campos para "Nome de usuário", "Email", "Senha" e "Repita a senha". A seção "Dados pessoais e médicos" contém campos para "Nome", "Sobrenome", "CRM", "CPF" e "Celular". No final do formulário, há um botão verde com o texto "Cadastrar".

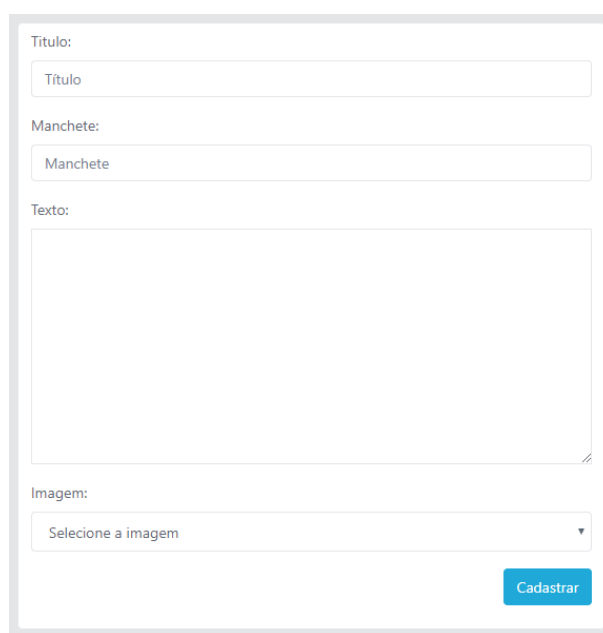
Fonte: Elaborado pelo autor

Ao acionar o botão “Cadastrar” os dados informados são inseridos no banco de dados.

4.14 Tela Gestão de Notícias

Ao acionar o botão “Notícias” o usuário é redirecionado para a tela de gestão de notícias. É exibido o *form* para o cadastro de notícias, solicitando título, manchete, texto e imagem para o cadastro conforme a figura 36.

Figura 36 – Form Cadastro Notícia

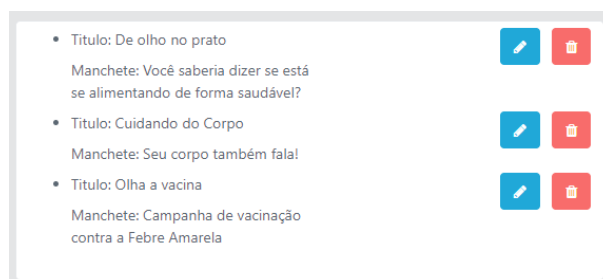


O formulário de cadastro de notícia é composto por quatro campos de entrada e um botão de ação. Os campos são: 'Título' (campo de texto curto), 'Manchete' (campo de texto curto), 'Texto' (campo de texto longo com uma barra de rolagem) e 'Imagem' (menu suspenso com o texto 'Selecione a imagem'). O botão 'Cadastrar' é azul e está localizado no canto inferior direito do formulário.

Fonte: Elaborado pelo autor

É exibido também uma lista com as notícias cadastradas, permitindo a edição ou a exclusão das mesmas conforme a figura 37.

Figura 37 – Lista Notícias



A lista de notícias cadastradas é exibida em um formato de lista com ícones de edição e exclusão. Cada item da lista contém o título, a manchete e os ícones de ação. Os ícones de edição são azuis e os de exclusão são vermelhos.

Título	Manchete	Ações
• Título: De olho no prato	Manchete: Você saberia dizer se está se alimentando de forma saudável?	[Ícone de edição] [Ícone de exclusão]
• Título: Cuidando do Corpo	Manchete: Seu corpo também fala!	[Ícone de edição] [Ícone de exclusão]
• Título: Olha a vacina	Manchete: Campanha de vacinação contra a Febre Amarela	[Ícone de edição] [Ícone de exclusão]

Fonte: Elaborado pelo autor

5 TESTES

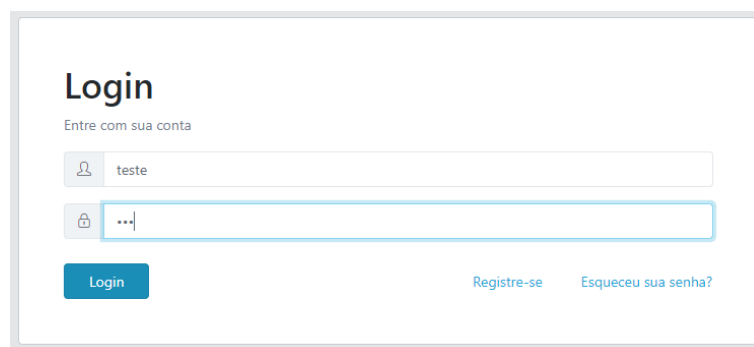
Neste capítulo foram executados testes funcionais de algumas funcionalidades com o objetivo de identificar se as mesmas foram implementadas corretamente.

5.1 Login

5.1.1 Logar no sistema com dados válidos

Dado que o usuário está na tela de Login, o mesmo digita usuário e senha válidos e aciona o botão “Login”, então é redirecionado para a tela principal do sistema conforme mostra a figura 38.

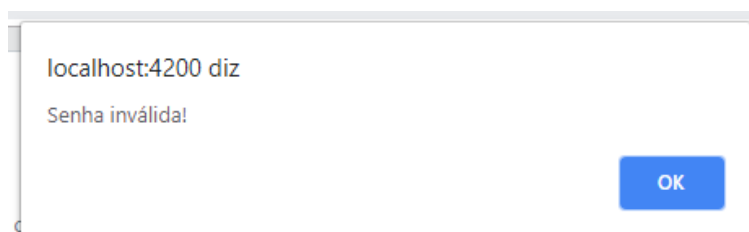
Figura 38 – Login Válido



Fonte: Elaborado pelo autor

5.1.2 Logar no sistema com dados inválidos

Dado que o usuário está na tela de Login, o mesmo digitar o usuário e/ou senha inválidos, ao acionar no botão “Login” é exibido um alerta de erro informando que os dados estão inválidos conforme mostra a figura 39.

Figura 39 – Login inválido

Fonte: Elaborado pelo autor

5.2 Cartão de Vacinas

5.2.1 Cadastrar Vacinas

Dado que o usuário está na tela de Cadastro de Vacina, e deseja cadastrar uma nova vacina, após informar todos campos obrigatórios e acionar o botão "Salvar" o sistema grava as informações e exibe a vacina preenchida no cartão de vacinas, conforme a figura 18.

5.2.2 Excluir Vacinas

Dado que o usuário está na tela de Cadastrado de Vacina e existe vacinas cadastradas, ao acionar o botão "Cadastrar" é exibido a lista de vacinas, onde ao acionar o botão "Excluir" o sistema exclui a vacina cadastrada e exibe um alerta de sucesso conforme a figura 40.

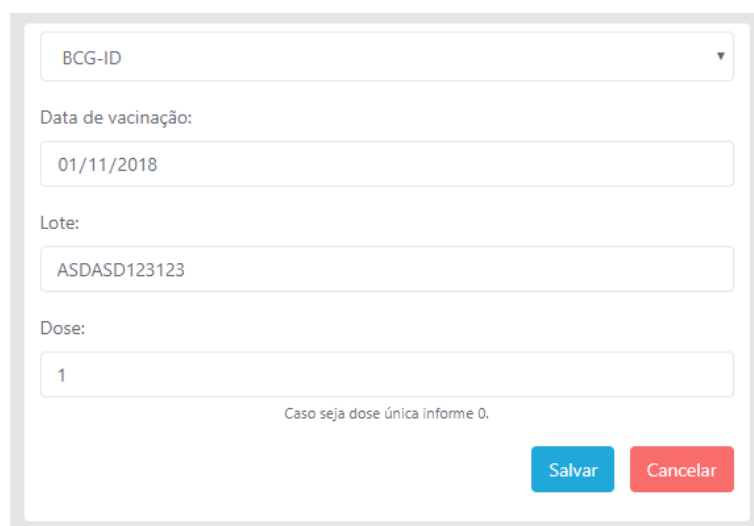
Figura 40 – Mensagem excluir vacina com sucesso

Fonte: Elaborado pelo autor

5.2.3 Editar Vacinas

Dado que o usuário está na tela de Cadastro de Vacina, e existe vacinas cadastradas ao acionar o botão "Cadastrar" é exibido a lista de vacinas, onde ao acionar o botão "Editar" localizado na listagem de Vacinas então o sistema exibe as informações da vacina cadastradas conforme a figura 41. Quando alterar os dados desejados e acionar no botão "Salvar", então o sistema salva as alterações e exibe o cartão de vacinas com as informações alteradas. Caso acione o botão "Cancelar" então o sistema retorna para o cartão de vacinas sem realizar alterações.

Figura 41 – Editar dados da vacina



O formulário de edição de dados da vacina apresenta os seguintes campos e elementos:

- Um menu suspenso no topo com o texto "BCG-ID" e uma seta para baixo.
- Um rótulo "Data de vacinação:" seguido por um campo de entrada contendo a data "01/11/2018".
- Um rótulo "Lote:" seguido por um campo de entrada contendo o texto "ASDASD123123".
- Um rótulo "Dose:" seguido por um campo de entrada contendo o número "1".
- Um texto de instrução em cinza: "Caso seja dose única informe 0."
- Dois botões de ação no canto inferior direito: "Salvar" (azul) e "Cancelar" (vermelho).

Fonte: Elaborado pelo autor

6 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foi desenvolvido um sistema Web com o objetivo de realizar o controle de vacinação pessoal utilizando a tecnologia do MEAN Stack.

6.1 Dificuldades Encontradas

No início de projeto foram encontradas dificuldades para adquirir a localização e telefones dos postos de saúde, pois isso não é de fácil acesso nos portais de saúde, e foi necessário realizar uma pesquisa mais detalhada para obter tais informações.

De uma maneira geral houve uma dificuldade para lidar com a nova versão do Angular, pois essa foi lançada em Abril deste ano e alguns componentes tiveram alterações em seu comportamento. O componente do Google *Maps* é um exemplo pois foi necessário realizar uma pesquisa em fóruns da comunidade desenvolvedora, para descobrir como realiza o *bind* das localizações.

6.2 Trabalhos Futuros

Existem funcionalidades que não foram mapeadas no processo de levantamento de requisitos, mas que podem complementar o sistema. Uma dessas funcionalidades é a de consulta de hospitais de Belo Horizonte, contendo a localização e informações úteis, como por exemplo as especialidades de cada hospital. Outra funcionalidade seria o envio de *e-mail* para os usuários, caso haja vacinas em atraso e quando for se aproximando data de vacina dos dependentes.

Existem também melhorias para as funcionalidades já implementadas, como o cadastro de novos tipos de vacinas na base de dados, possibilitando o cadastro e controle das mesmas, e a montagem dinâmica do cartão de vacinas no momento das consultas.

REFERÊNCIAS

- ASTOLFI, B. A arquitetura mvc no desenvolvimento de jogos para navegadores. 2012.
- BAKER, B. Business modeling with uml: The light at the end of the tunnel. 2001.
- BONFIM, F. L.; LIANG, M. Aplicações escaláveis com mean stack. 2014.
- BONFIOLI, G. F. Banco de dados relacional e objeto-relacional: Uma comparação usando postgresql. 2006.
- BRASIL, G. do. *Postos de saúde disponibilizam vacinas gratuitas durante todo o ano*. 2018. Julho 6, 2018. Disponível em: <<http://www.brasil.gov.br/noticias/saude/2018/07/postos-de-saude-disponibilizam-vacinas-gratuitas-durante-todo-o-ano>>.
- CAMARGO, L. T. E. *Node.js – O que é, como funciona e quais as vantagens*. 2018. Disponível em: <<https://www.opus-software.com.br/node-js/>>.
- COULOURIS, G. e. a. *SISTEMAS DISTRIBUÍDOS Conceitos e Projeto*. 5. ed. Av. Jerônimo de Ornelas, 670, Santana, Porto Alegre, RS: Bookman, 2013. ISBN 9780132143011.
- DB-ENGINES. *The DB-Engines Ranking ranks database management systems according to their popularity*. 2018. Setembro 15, 2018. Disponível em: <<https://db-engines.com/en/ranking>>.
- EXPRESS. *Express Framework web rápido, flexível e minimalista para Node.js*. 2018. Disponível em: <<https://expressjs.com/pt-br/>>.
- FERNANDES, J. H. C. *As 10 Áreas da Engenharia de Software, Conforme o SWEBOK*. 2004. Disponível em: <[http://files.engenharia-de-software7.webnode.com/200000016-0a7e60b780/As 10 Áreas da Engenharia de Software, Conforme o SWEBOK.pdf](http://files.engenharia-de-software7.webnode.com/200000016-0a7e60b780/As%2010%20Áreas%20da%20Engenharia%20de%20Software,%20Conforme%20o%20SWEBOK.pdf)>.
- FRATERNALLI, P.; PAOLINI, P. A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications. 1998.
- G1, S. *Estados enfrentam surto de sarampo, que volta a ameaçar o Brasil*. 2018. Julho 7, 2018. Disponível em: <<http://g1.globo.com/jornal-nacional/noticia/2018/07/estados-enfrentam-surto-de-sarampo-que-volta-ameacar-o-brasil.html>>.
- IEEE, C. S. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. 2000. Setembro, 2000. Disponível em: <<http://cabibbo.dia.uniroma3.it/ids/altrui/ieee1471.pdf>>.
- LÓSCIO, B. F. e. a. Nosql no desenvolvimento de aplicações web colaborativas. 2011.
- MDN, M. D. N. *Introdução - JavaScript*. 2018. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Introduction>>.

MENDES, E.; CONTE, T.; TRAVASSOS, G. H. Processos de desenvolvimento para aplicações web: Uma revisão sistemática. 2005.

MORAES, A. L. *Adultos também têm que tomar vacinas*. 2018. Julho 30, 2018. Disponível em: <<https://saude.abril.com.br/medicina/adultos-tambem-tem-que-tomar-vacinas/>>.

PFÜTZENREUTER, E. *Node.js: ferramenta e filosofia*. 2015. Setembro 16, 2018. Disponível em: <<https://epxx.co/artigos/nodejs1.html>>.

SARKER, I. H.; APU, K. Mvc architecture driven design and implementation of java framework for developing desktop application. 2014.

SILVEIRA, A. S. d. A. e. a. *Controle de vacinação de crianças matriculadas em escolas municipais da cidade de São Paulo*. 2007. Junho, 2007. Disponível em: <http://www.scielo.br/scielo.php?pid=S0080-62342007000200018&script=sci_arttext>.

SOMMERVILLE, I. *Engenharia de software*. 8. ed. [S.l.]: Pearson Addison-Wesley, 2007.

SOMMERVILLE, I. *Engenharia de software*. 9. ed. [S.l.]: Pearson Education do Brasil Ltda, 2011. ISBN 9788579361081.

SOUZA, F. P. e. a. Estudo de viabilidade do uso de arquitetura orientada a microsserviços para maximizar o reaproveitamento de código. 2015.

STIPKOVIC, C.; BIANCHINI, C. d. P. Computação paralela em javascript. 2015.

TANENBAUM, A.; STEEN, M. *Sistemas distribuídos: princípios e paradigmas*. 2. ed. [S.l.]: Pearson Prentice Hall, 2008. ISBN 9788576051428.

VERGILIO, S. R. *Arquitetura de Software*. 2004. Disponível em: <<http://www.inf.ufpr.br/andrey/ci163/IntroduzArquiteturaAl.pdf>>.

VERGILIO, S. R. *Arquitetura de Software*. 2005. Disponível em: <<http://www.inf.ufpr.br/andrey/ci163/PadroesFrameworksAl.pdf>>.