

MEC-SETEC
INSTITUTO FEDERAL MINAS GERAIS - *Campus* Formiga
Curso de Ciência da Computação

**PROTÓTIPO DE *SINGLE PAGE APPLICATION* (SPA) PARA
GERENCIAMENTO DE SORVETERIAS E DISPOSITIVO PARA
MONITORAMENTO DA TEMPERATURA**

Déborah Aparecida Resende

Orientador: Prof. Dr. Bruno Ferreira

FORMIGA- MG

2018

DÉBORAH APARECIDA RESENDE

**PROTÓTIPO DE *SINGLE PAGE APPLICATION* (SPA) PARA
GERENCIAMENTO DE SORVETERIAS E DISPOSITIVO PARA
MONITORAMENTO DA TEMPERATURA**

Trabalho de Conclusão de Curso apresentado ao
Instituto Federal Minas Gerais - *Campus*
Formiga, como requisito parcial para a obtenção
do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Bruno Ferreira.

FORMIGA- MG

2018

004	<p>Resende, Déborah Aparecida.</p> <p>Protótipo de single page application (SPA) para gerenciamento De sorveterias e dispositivo para monitoramento de temperatura / Déborah Aparecida Resende. -- Formiga : IFMG, 2018. 103p. : il.</p> <p>Orientador: Prof. Dr. Bruno Ferreira Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – <i>Campus</i> Formiga.</p> <p>1. Vue. 2. MongoDB. 3. NodeJs. 4. SPA. 5. Raspberry Pi, IoT. I. Título.</p> <p style="text-align: right;">CDD 004</p>
-----	---

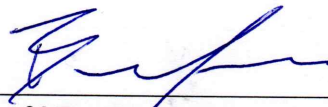
DÉBORAH APARECIDA RESENDE

**PROTÓTIPO DE SINGLE PAGE APPLICATION(SPA) PARA
GERENCIAMENTO DE SORVETERIAS E DISPOSITIVO PARA
MONITORAMENTO DA TEMPERATURA**

Trabalho de Conclusão de Curso apresentado ao
Instituto Federal de Minas Gerais-Campus Formiga,
como Requisito parcial para obtenção do título de
Bacharel em Ciência da Computação.

Aprovado em: 21 de novembro de 20 18.

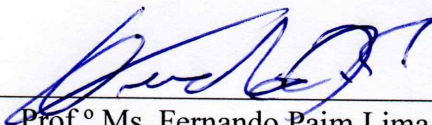
BANCA EXAMINADORA



Prof.º Dr. Bruno Ferreira



Prof.ª Dra. Paloma Maira de Oliveira



Prof.º Ms. Fernando Paim Lima

Agradecimentos

Primeiramente agradeço a Deus por estar sempre iluminando os meus caminhos. Aos meus pais, Simone e Genivaldo, que sempre estiveram ao meu lado, me apoiando, me incentivando e ajudando em todos os momentos e que tanto contribuíram para minha formação.

A minha irmã Clarisse, por acreditar que este sonho um dia se realizaria.

Ao meu namorado Diego, que foi amigo, conselheiro e sempre me encorajou a seguir em frente.

Ao meu orientador Bruno Ferreira, por compartilhar seus conhecimentos e pelo apoio durante o trabalho.

Aos meus amigos e familiares, que sempre torceram por mim.

Aos professores do curso Ciência da Computação, pelos conhecimentos sabiamente transmitidos.

“O sucesso é a soma de pequenos esforços repetidos dia após dia.”

Robert Collier

Resumo

Este trabalho apresenta o desenvolvimento de um protótipo de um sistema Web *Single Page Application*(SPA) para auxiliar no gerenciamento de sorveterias, principalmente aquelas que possuem filiais. Ele propõe funcionalidades tanto para o funcionário quanto para o administrador, além de realizar o monitoramento da temperatura ideal dos *freezers*. O sistema foi desenvolvido utilizando os componentes do MEVN: MongoDB, Express, Vue e NodeJs. O sensor DHT22 e o Raspberry Pi 3 Model B foram usados para efetivar a leitura da temperatura e o ThingSpeak foi responsável por armazenar os dados.

Palavras-chave: Vue, MongoDB, NodeJs, SPA, Raspberry Pi, IoT.

Abstract

This thesis presents the development of a prototype of a Single Page Application(SPA) Web system to assist in the management of ice cream shops, especially those with branches. It offers functionalities for both the employee and the administrator, as well as monitoring the ideal temperature of the freezers. The system was developed using the components of the MEVN: MongoDB, Express, Vue and NodeJs. The DHT22 sensor and the Raspberry Pi 3 Model B were used to make the temperature read and the ThingSpeak was responsible for storing the data.

Keywords: Vue, MongoDB, NodeJs, SPA, Raspberry Pi, IoT.

Lista de ilustrações

Figura 1 – Associação dos métodos HTTP com operações CRUD.	32
Figura 2 – Gráfico demonstrando os <i>downloads</i> do NPM de julho de 2017 a janeiro de 2018.	33
Figura 3 – Número de <i>downloads</i> do NPM de janeiro de 2017 a janeiro de 2018 e sua respectiva mudança percentual.	34
Figura 4 – Sensor DHT22.	44
Figura 5 – Raspberry Pi 3 Model B.	51
Figura 6 – Raspberry Pi 3 GPIO <i>Header</i>	52
Figura 7 – Fases do PU.	53
Figura 8 – Diagrama de Caso de Uso para administrador.	55
Figura 9 – Diagrama de Caso de Uso para funcionário.	56
Figura 10 – GUI <i>Login</i> do sistema.	57
Figura 11 – GUI <i>Login</i> do sistema com erro de autenticação.	57
Figura 12 – GUI Cadastro de Administrador.	58
Figura 13 – GUI Cadastro de Administrador com dados inválidos nos campos.	58
Figura 14 – Menu Lateral para administradores.	59
Figura 15 – Menu Lateral para funcionários.	60
Figura 16 – Barra de Ferramentas.	61
Figura 17 – GUI Lojas.	61
Figura 18 – Cadastrando Nova Loja.	62
Figura 19 – Editando Loja.	62
Figura 20 – Confirmação para excluir loja.	63
Figura 21 – Filtrando lojas.	63
Figura 22 – GUI Funcionários.	64
Figura 23 – GUI Produtos.	65
Figura 24 – GUI Estoque com caixa fechado.	66
Figura 25 – Adicionando quantidade de determinado produto ao estoque.	66
Figura 26 – GUI Abrir Caixa.	67
Figura 27 – GUI Realizar Reforço de Caixa.	67
Figura 28 – GUI Realizar Sangria.	68
Figura 29 – GUI Fechamento de Caixa.	68
Figura 30 – GUI Vendas.	69
Figura 31 – GUI Vendas pesquisando produtos.	69
Figura 32 – GUI Vendas selecionando tipo de pagamento.	70
Figura 33 – Adicionando novo sensor de temperatura.	70
Figura 34 – GUI monitoramento da temperatura dos <i>freezers</i>	71

Figura 35 – GUI Fluxo de Caixa Diário.	71
Figura 36 – GUI Fluxo de Caixa Diário da data 03/11/2018.	72
Figura 37 – GUI Gráfico de Comparação de Vendas das Lojas.	72
Figura 38 – GUI Gráfico de Comparação de Vendas das Lojas gerando gráfico do intervalo: 25/10 - 30/10.	73
Figura 39 – Conexão Mongoose e método <i>connect</i>	74
Figura 40 – <i>Schema</i> usuário.	74
Figura 41 – Exemplo de um documento usuário.	75
Figura 42 – <i>Schema</i> loja.	75
Figura 43 – <i>Schema</i> produto.	76
Figura 44 – Exemplo de um documento loja.	77
Figura 45 – Modelo documento movimentação.	78
Figura 46 – Modelo documento Produto e Quantidade.	78
Figura 47 – Exemplo de um documento movimentação após abertura de caixa. . .	78
Figura 48 – Exemplo de um documento movimentação após vendas, operações e fechamento de caixa.	79
Figura 49 – Configurando ThingSpeak.	80
Figura 50 – Esquemático da montagem do circuito utilizando a ferramenta Fritzing. 80	
Figura 51 – Montagem do circuito.	81
Figura 52 – <i>Field Charts</i> do Thingspeak do canal 606462 após receber alguns dados. 83	
Figura 53 – Diagrama de Caso de Uso para administrador.	93
Figura 54 – Diagrama de Caso de Uso para funcionário.	93

Lista de abreviaturas e siglas

AJAX	JavaScript e XML assíncronos
API	Interface de Programação de Aplicações
BJSON	JSON Binário
CRUD	Criar, Consultar, Atualizar e Deletar
CSRF	Falsificação de Solicitação entre Sites
CSS	Folhas de Estilo em Cascata
CSU	Caso de Uso
GUI	Interface Gráfica com o usuário
HTML	Linguagem de Marcação de Hipertexto
HTTP	Protocolo de Transferência de Hipertexto
MIT	Instituto de Tecnologia de Massachusetts
NPM	Gerenciador de Pacotes do Node
PDV	Ponto de Venda
PU	Processo Unificado
REST	Transferência de Estado Representacional
SPA	Aplicativo de Página Única
JSON	Notação de Objetos JavaScript
JWT	JSON <i>Web Token</i>
UML	Linguagem de Modelagem Unificada
URI	Identificador de Recurso Uniforme

Sumário

1	INTRODUÇÃO	25
1.1	Justificativa	25
1.2	Objetivos	26
1.2.0.1	Objetivo Geral	26
1.2.0.2	Objetivo Específico	26
1.2.0.3	Estrutura do Trabalho	27
2	FUNDAMENTAÇÃO TEÓRICA	29
2.1	Empresas do Ramo	29
2.1.1	Operações de Caixa	30
2.1.2	Fluxo de Caixa	30
2.2	Desenvolvimento de Aplicações Web	30
2.3	REST	32
2.4	Linguagem JavaScript e Frameworks Relacionados	32
2.4.1	Vue	34
2.4.1.1	Vuetify	35
2.4.1.2	Vue Router	36
2.4.1.3	Vue Navigator Online	36
2.4.1.4	ESLint Vue	36
2.4.1.5	Vue Charts	36
2.4.2	NodeJs	37
2.4.2.1	NPM	37
2.4.2.2	Express	37
2.4.2.3	Mongoose	37
2.4.2.4	Nodemon	38
2.4.2.5	Passport	38
2.4.2.5.1	Bcrypt	38
2.4.3	Webpack	38
2.4.4	Axios	38
2.4.5	Babel	39
2.5	Bases de Dados	39
2.5.1	MongoDB	41
2.6	Single-Page Applications	41
2.6.1	JWT	42
2.7	Dispositivo para Monitoramento da Temperatura	42

2.7.1	Sensor DHT22	43
2.7.2	Raspberry Pi	44
2.7.3	Python	45
2.7.4	ThingSpeak	45
2.8	Ferramentas auxiliares	46
2.8.1	Sublime Text	46
2.8.2	Postman	46
2.8.3	Lucidchart	46
2.8.4	Fritzing	46
2.9	Sistemas Similares	47
2.9.1	MarketUP	47
2.9.2	Consumer	47
2.9.3	CTP Sorveteria	47
3	MATERIAIS E MÉTODOS	49
3.1	Raspberry Pi 3 Model B	50
3.1.1	Raspbian	52
3.2	Metodologia	52
3.2.1	Concepção	53
3.2.2	Elaboração	53
3.2.3	Construção	54
3.2.4	Transição	54
3.2.5	Macro Entregas do Projeto	54
4	PROJETO E DESENVOLVIMENTO	55
4.1	Modelagem	55
4.1.1	Diagrama de Caso de Uso	55
4.1.1.1	Administrador	55
4.1.1.2	Funcionário	56
4.2	Interfaces	56
4.2.1	GUI Login	56
4.2.2	GUI Cadastro Administrador	57
4.2.3	Menu	59
4.2.4	Barra de Ferramentas	60
4.2.5	GUI Lojas	61
4.2.6	GUI Funcionários	63
4.2.7	GUI Produtos	64
4.2.8	GUI Estoque	65
4.2.9	GUI Abrir Caixa	66
4.2.10	GUI Reforço	67

4.2.11	GUI Sangria	67
4.2.12	GUI Fechamento	68
4.2.13	GUI Vendas	68
4.2.14	GUI Monitoramento da Temperatura	70
4.2.15	GUI Fluxo de Caixa	71
4.2.16	GUI Gráfico de Comparação de Vendas das Lojas	72
4.3	Modelo de Dados	73
4.3.1	Usuário	74
4.3.2	Loja	75
4.3.3	Movimentação	77
4.4	Monitoramento da Temperatura	79
5	CONSIDERAÇÕES FINAIS	85
	REFERÊNCIAS	87
	APÊNDICE A – DIAGRAMA E EXPANSÕES DE CASOS DE USO	93

1 INTRODUÇÃO

Uma das sobremesas mais populares, o sorvete é um grande sucesso entre o público consumidor brasileiro, ocupando a 10^a posição mundial no consumo de sorvetes (SEBRAE, 2017). Segundo Azevedo (2016), o sorvete é uma iguaria valorizada pela população, o mercado de sorveterias é consolidado e o setor ainda tem bastante espaço para crescer, o que pode significar boas oportunidades de negócios.

Um estudo realizado por Duarte (2017), afirma que existem aproximadamente 8 mil empresas que se encaixam no setor de sorvete no Brasil. Assim como em qualquer outro negócio, fatores como controle e organização são alguns dos principais responsáveis pelo sucesso de um empreendimento. Inevitavelmente, a tecnologia vêm ganhando cada vez mais espaço no meio empresarial. Independente do tamanho da empresa, adotar *softwares* que auxiliem no gerenciamento é uma importante iniciativa que traz benefícios a várias áreas da empresa, promovendo um melhor controle e análise de dados, o que contribuirá para a tomada de decisões importantes e economia de tempo.

O crescimento através da *franchising*¹ tem se mostrado uma estratégia viável para um grande número de empresas. Mas os casos de sucesso relatam um trabalho minucioso de planejamento e gestão (ESTADÃO, 2017).

O gerenciamento através de plataformas Web vem ganhando espaço e preferência de muitos gestores, principalmente com empresas que possuem filiais ou que são gerenciadas a distância. Comumente possuem um bom custo-benefício, sendo uma ótima ferramenta inclusive para pequenas ou microempresas. Por serem *online* possuem: mobilidade, escalabilidade, instantaneidade, integração, flexibilidade e facilidade de entrega: instalação e manutenção (SILVA, 2017).

Para auxiliar no estudo sobre necessidades e problemas de empresas desse ramo com filiais, contou-se com o auxílio de duas sorveterias situadas em Alfenas e Varginha, ambas gerenciadas pelo mesmo administrador em Divinópolis. O protótipo de *software* desenvolvido irá atender tanto o funcionário da filial quanto o administrador da empresa.

1.1 Justificativa

Com o crescimento do consumo de sorvetes no Brasil e conseqüentemente crescimento dessas companhias (SEBRAE, 2017), existem empresas que não contam com

¹ Um tipo de arranjo comercial no qual a franqueadora concede ao franqueado uma franquía do seu negócio, com o objetivo de replicar o sucesso anterior. O *franchising* “clona”, em diferentes locais, o mesmo tipo modelo de negócio aliado ao conceito da marca (RUSCHEL, 2017).

sistemas que auxiliam adequadamente na gestão de empresas com filiais nesse ramo, uma delas é Sorveteria Sol e Verão. Ela possui duas unidades situadas em Alfenas e Varginha, e é gerenciada por um administrador em Divinópolis.

Uma das principais justificativas para a realização deste projeto é melhorar a gestão administrativa da empresa, tanto para funcionários quanto para o administrador. O administrador da empresa em questão não possui um único *software* para gerenciar ambas as lojas, ele utiliza um sistema para cada uma.

Empresas desse ramo que possuem um grande número de filiais têm seu próprio sistema de gestão administrativa, mas empresas com filiais pequenas enfrentam alguns problemas:

- Existem poucos sistemas desse tipo no mercado;
- Apresentam funcionalidades que não se encaixam nas regras de negócios da empresa;
- Alto custo de implantação e manutenção.

Assim, o sistema adequado para empresa ocasionará uma melhor gestão, atendendo as necessidades específicas da organização e conseqüentemente redução considerável de tempo.

1.2 Objetivos

1.2.0.1 Objetivo Geral

Este trabalho tem como objetivo o desenvolvimento de um protótipo de um sistema Web que auxilie, tanto o funcionário quanto o administrador, no gerenciamento e monitore a temperatura ideal dos *freezers* de empresas no ramo de sorveteria.

1.2.0.2 Objetivo Específico

São objetivos específicos deste trabalho:

- Estudo de caso, afim de verificar possíveis situações e problemas, e assim adequar o sistema;
- Realizar a modelagem do sistema;
- Pesquisar ferramentas e tecnologias para atender o projeto;
- Utilizar tecnologias e ferramentas mais atuais para o desenvolvimento do *software*;

- Fornecer um ambiente que permita ao funcionário manter informações sobre a sorveteria, realizar vendas e ter acesso a relatórios financeiros;
- Prover um ambiente que permita aos administradores acesso a informações relevantes sobre suas lojas.

1.2.0.3 Estrutura do Trabalho

Este trabalho é composto por cinco capítulos, sendo que este é o primeiro, onde fez-se as apresentações das ideias iniciais do projeto, com os objetivos e a justificativa.

No segundo capítulo é apresentado a fundamentação teórica sobre empresas do ramo, desenvolvimento de aplicações Web, linguagem Javascript e *Frameworks* relacionados, base de dados e dispositivo para monitoramento da temperatura.

No terceiro capítulo apresenta os materiais e métodos usados para desenvolver a aplicação, assim como a metodologia.

O quarto capítulo expõe a modelagem do sistema, interfaces e modelagem do banco de dados.

Por fim, o quinto capítulo contém as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta o referencial teórico utilizado para entender os conceitos necessários para a criação do protótipo do sistema Web e o dispositivo que realizará a medição da temperatura.

2.1 Empresas do Ramo

Muitas informações indicam que o sorvete foi criado pelos chineses, há cerca de três mil anos atrás. Naquela época, o sorvete começou a ser feito através de frutas, mel e neve. Mas só chegou ao conhecimento dos brasileiros por volta de 1835 (MIGUEL, 2010). Com o passar do tempo, sofreu diversas modificações até chegar à consistência e o sabor conhecidos atualmente.

Ainda longe dos índices registrados nos Estados Unidos e em alguns países da Europa, o consumo de sorvete no Brasil cresce a cada ano. Segundo o SEBRAE (2017), o Brasil ocupa a 10ª posição mundial no consumo de sorvetes, que corresponde a 3,1% no cenário mundial. O mercado de sorvetes brasileiro faturou R\$ 14,9 bilhões em 2016. O país tem, atualmente, cerca de 8 mil empresas dedicadas à produção e comercialização de sorvetes, o segmento responde por 75 mil empregos diretos e 200 mil indiretos.

Uma sorveteria é um comércio que tem como seu principal produto de vendas o sorvete mas atualmente, o mercado de açaí cresce a passos largos, fazendo com que quase todas as sorveterias ofereçam açaí em seu portfólio de venda. Outros produtos agregadores podem ser vendidos também como bebidas (água, refrigerantes, sucos), picolés, vitaminas entre outros.

O monitoramento da temperatura é extremamente importante durante todo o processo de fabricação e armazenamento de sorvetes (DURSO, 2012). É esse controle que vai garantir a consistência certa e a qualidade dos produtos. Quando armazenado de maneira irregular, fora da temperatura ideal, o sorvete pode sofrer algumas alterações irreversíveis. Entre as principais estão a arenosidade, o aparecimento de cristais de gelo, a perda de cor e a separação de fases. Segundo Maroma (2016), no balcão expositor, no local de venda dos sorvetes, a temperatura deve manter-se o mais constante possível, entre -12 e -17°C.

O maior problema do mercado de sorvetes fica a cargo da sazonalidade. Existe maior consumo do produto durante o verão e baixa durante o inverno, por isso a empresa precisa realizar uma gestão administrativa eficiente, agilizando a tomada de decisões e apontando opções para a melhoria do desempenho para conseguir se manter no mercado.

O processo de uma sorveteria inclui basicamente, segundo [Tassi \(2014\)](#), as atividades de atendimento e venda, operações de caixa, e gestão administrativa como compras, contas a pagar e fluxo de caixa. As operações de caixa e fluxo de caixa serão melhores abordados nas seções [2.1.1](#) e [35](#), respectivamente.

2.1.1 Operações de Caixa

Algumas operações são fundamentais para o correto funcionamento de um caixa, entre elas está a abertura, o suprimento, sangria e o fechamento de caixa.

Na abertura de caixa é declarado o valor inicial de reserva, também conhecido como fundo de troco ou valor de encaixe. Esse valor, normalmente composto por cédulas e moedas, servirá para que o operador possa dar troco, ou pagamento de despesas no decorrer do período.

Suprimento, também conhecido como reforço de caixa, é todo valor que entra no caixa. Consiste em suprir o caixa em situações como uma eventual falta de troco.

A sangria é o contrário do suprimento, ou seja, é todo valor que sai do caixa, permitindo que o usuário efetue a retirada de acordo com o valor disponível em caixa.

O termo fechar o caixa diz respeito as ações realizadas com intuito de se verificar se o valor presente no caixa condiz com as movimentações financeiras efetuadas durante o período em que o caixa esteve aberto. Além disso o fechamento de caixa organiza o ambiente financeiro tornando viável a retomada das atividades financeiras no dia seguinte ([ÁVILA, 2015](#)).

2.1.2 Fluxo de Caixa

[Erbano et al. \(2014\)](#) define o fluxo de caixa como um controle de entrada e saída do dinheiro de um dado período, que permite a análise da geração dos meios financeiros e da sua utilização num determinado período de tempo.

"É composto por dados obtidos dos controles de contas a pagar, contas a receber, de vendas, de despesas, de saldos de aplicações e de todos os demais elementos que representem as movimentações de recursos financeiros da empresa" ([SEBRAE, 2008](#)).

2.2 Desenvolvimento de Aplicações Web

O desenvolvimento crescente de tecnologias traz a necessidade de realizar tarefas de maneira cada vez mais simples e informatizadas, de modo que seja exigido o menor esforço possível por parte das pessoas ([SILVA, 2017](#)). As aplicações em plataforma Web são

exemplos dessa informatização, além de melhorar a mobilidade e a facilidade de entrega: instalação e atualização.

Uma aplicação Web é um programa implementado para ser executado por meio de um navegador, na Internet ou em redes privadas. Seu desenvolvimento tem muito haver com a necessidade de simplificar a utilização e manutenção, mantendo o código fonte em um mesmo local, de onde é acessado pelos diferentes usuários (AREND, 2013).

Em sua dissertação, Duarte (2015) expõe que o desenvolvimento de aplicações Web consiste na construção de dois grandes componentes: cliente e servidor. Esse tipo de aplicação requer um poder de processamento extremamente veloz, para que a interação em tempo real entre cliente e servidor seja eficaz.

No lado do cliente são desenvolvidas páginas através da utilização de linguagens de programação como HTML, CSS e JavaScript. Estas linguagens são interpretadas no navegador de internet do utilizador da aplicação. Por sua vez, no servidor é desenvolvida a lógica de negócio, usando por exemplo linguagens como PHP, NodeJs e .NET.

As instâncias clientes realizam requisições de dado para o servidor da aplicação que está sendo executada e aguardam o retorno da resposta. O servidor disponível para a aplicação pode aceitar as requisições, processá-las e retornar o resultado para o cliente.

Hypertext Transfer Protocol (HTTP) é um protocolo utilizado para enviar e receber informações na Web, do tipo *stateless*, ou seja, ele não guarda o estado do cliente. Um protocolo é um conjunto de regras que determinam que tipo de informações podem ser trocadas, e que mensagens são apropriadas para encaminhar.

Em 2005, Garrett (2005) em seu artigo propôs uma abordagem na construção de aplicações Web chamada Ajax. O nome é uma abreviação de *Asynchronous JavaScript + XML*, e apresentava uma mudança significativa no modelo tradicional de aplicações Web: sob o modelo tradicional, as interações do usuário disparavam uma requisição HTTP para um servidor, que processava os dados necessários e servia uma nova página HTML como resposta à requisição. Com Ajax, segundo Oliveira (2017), a comunicação entre cliente e servidor se torna assíncrona através de uma camada adicional chamada motor Ajax, que solicita dados ao servidor e que realiza no lado do cliente todo o processamento que possa ser feito sem a necessidade de enviar ou solicitar dados ao servidor.

Com o uso de Ajax, para aumentar a interatividade da aplicação para o utilizador e para reduzir pedidos feitos ao servidor desenvolveu-se o conceito de SPA (*Single Page Applications*): páginas que atualizam sua interface à medida que seus usuários interagem, sem a necessidade de recarregar todo o conteúdo, conceito que será retratado com melhor detalhe na seção 2.6 deste documento.

2.3 REST

É um estilo de arquitetura de *software* para sistemas hipermídia distribuídos, como, por exemplo, a Web do jeito que conhecemos atualmente, ou seja, onde utilizamos um navegador Web para acessar recursos que estamos interessados, geralmente uma página HTML, mediante a digitação a digitação de uma URL.(SANTOS, 2015)

Dados e funcionalidades são considerados recursos e estes são acessados via URIs geralmente via links. Por ser baseado em HTTP, a arquitetura geralmente é cliente-servidor e *stateless*. Aplicar os princípios da arquitetura REST faz com que as aplicações se tornem mais simples, leves e de alta performance.

Um dos princípios do REST é que todo recurso deve possuir uma identificação única, sendo que um recurso é uma abstração sobre determinado tipo de informação que aplicação gerencia. Sem essa identificação não seria possível para a aplicação distinguir qual recurso deve ou não ser manipulado de acordo com a solicitação (FERREIRA, 2017). Com o recurso devidamente identificado, as ações sobre ele são descritas com o uso de quatro verbos para criar, retornar, alterar e excluir recursos: *GET*, *PUT*, *POST* e *DELETE*.

Segundo Santos (2015) quando se tem Web *service* utilizando o estilo de arquitetura REST, tem-se um Web *service* RESTful. Em RESTful ha o mapeamento dos principais métodos HTTP com operações CRUD de um banco de dados, como pode-se observar na Figura 1.

Figura 1 – Associação dos métodos HTTP com operações CRUD.

Métodos HTTP	Operações CRUD
GET	SELECT
POST	INSERT, UPDATE, DELETE
PUT	CREATE, UPDATE
DELETE	DELETE

Fonte: (SANTOS, 2015)

2.4 Linguagem JavaScript e *Frameworks* Relacionados

JavaScript é uma linguagem de programação *client-side* da Web, segundo Prescott (2016), pode ser considerada a linguagem de programação mais onipresente da história pois a ampla maioria dos sites modernos usam JavaScript e os navegadores modernos incluem seus interpretadores.

JavaScript é uma linguagem de alto nível, dinâmica, interpretada e não tipada, conveniente para estilos de programação orientados a objetos e

funcionais. Sua sintaxe é derivada da linguagem Java, das funções de primeira classe de *Scheme* e da herança baseada em protótipos de *Self* (FLANAGAN, 2007).

Flanagan (2007) afirma em seu livro que existe uma tríade de tecnologias que todos os desenvolvedores Web devem conhecer: HTML, CSS e JavaScript.

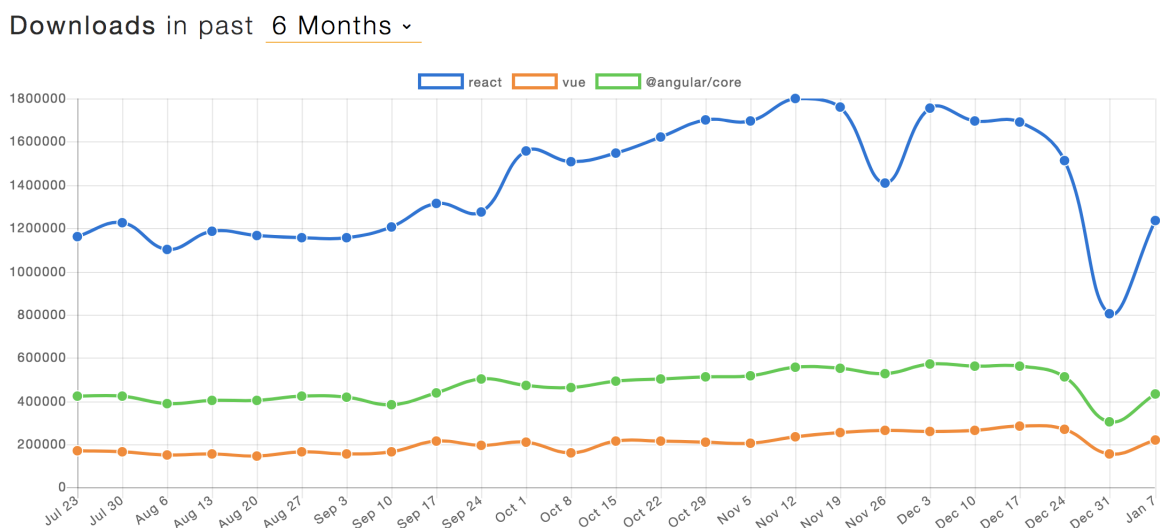
O acrônimo HTML vem do inglês e significa *Hypertext Markup Language* ou em português Linguagem de Marcação de Hipertexto. Segundo Bax (2001) é a linguagem base da Internet e possui um grupo de *tags* predefinidos, concebidos com a função de organizar e especificar a informação a ser transferida por meio de páginas Web.

CSS é a sigla para o termo em inglês *Cascading Style Sheets*, que traduzido para o português significa Folha de Estilo em Cascatas. É uma camada que se usa para controlar o estilo, especificar apresentação de páginas da Web (JOBSTRAIBIZER, 2009).

A comunidade de JavaScript se mostrou ativa o bastante para identificar as limitações que se apresentavam nas suas ferramentas e construir outras que corrigissem, amenizassem ou melhorassem tais aspectos. Nesse ambiente em que a criação de ferramentas pelos próprios usuários delas é tão comum.

Tomando como exemplo a tarefa de construção da interface de usuário, atualmente temos uma grande quantidade de opções de bibliotecas e *frameworks* para facilitar o desenvolvimento. As três estruturas que atualmente dominam em popularidade e uso são React, Angular e Vue. Cada um deles tem grandes comunidades e muitos recursos de treinamento disponíveis. A Figura 2 demonstra os downloads do NPM em um intervalo de seis meses e Figura 3 apresenta os valores e sua respectiva mudança percentual.

Figura 2 – Gráfico demonstrando os *downloads* do NPM de julho de 2017 a janeiro de 2018.



Fonte: (HANNAH, 2018).

Figura 3 – Número de *downloads* do NPM de janeiro de 2017 a janeiro de 2018 e sua respectiva mudança percentual.

Framework	Jan. 29, 2017	Jan. 28, 2018	Percent change
React	708,421	1,786,699	152.2%
Angular	229,375	609,087	165.5%
Vue	60,283	299,840	397.4% 🚀

Fonte: (HANNAH, 2018).

Analisando as Figura 2 e 3, percebe-se que o React está muito à frente do Angular e do Vue. Mas o que é menos aparente é que o Vue teve aproximadamente o dobro da taxa de crescimento no ano de 2017 em comparação com o Angular. Se as estrelas do GitHub são um indicador de interesse, então a Vue também parece forte, com 79.000 estrelas em comparação com as 32.000 da Angular e React que tem quase 86.000 estrelas. Segundo Hannah (2018), como o Vue possui a maior taxa de crescimento e, se as tendências atuais continuarem, o Vue ultrapassará o Angular como o segundo *framework* mais usado no final de 2018 ou início de 2019.

A comunidade JavaScript reconhece as vantagens de usar a mesma linguagem tanto para o *script* do lado do cliente quanto para o lado do servidor. Com isso, o NodeJs se tornou um dos ambiente mais populares para o desenvolvimento da Web do lado do servidor com o código JavaScript.

A comunidade também encontrou outra solução para o ainda existente problema de inconsistência entre navegadores: compilar o código-fonte de modo a simular certas funcionalidades da linguagem ainda não presente em todos os navegadores em função de APIs já suportadas por todos os navegadores (OLIVEIRA, 2017). Compiladores como o Babel, que será retratado com melhor detalhe na seção 2.4.5 deste documento, são utilizados em larga escala atualmente.

2.4.1 Vue

Vue (pronunciado como *View*), surgiu com a ideia de criar um *framework* que ajudaria na prototipagem rápida, oferecendo uma maneira fácil e flexível de ligação de dados reativos e componentes reutilizáveis.

Segundo Silva e Silva (2017) “é um *framework* progressivo do JavaScript de código aberto para a construção de interfaces de usuário”. A biblioteca central é focada apenas na camada de visualização e é fácil de ser coletada e integrada a outras bibliotecas ou projetos existentes. Por outro lado, o Vue também é perfeitamente capaz de fornecer aplicativos de página única sofisticados (VUEJS.ORG, 2016).

Para cada componente, o Vue utiliza um ficheiro de extensão *.vue* cuja estrutura se divide em três seções: *template*, *script* e *style*. No *template* é possível a utilização de qualquer linguagem como o HTML, no *script* é definido o comportamento do componente e no *style* é que se trata de estilizar o *template*.

Os componentes contém marcação, estilo e comportamento (HTML, CSS e JavaScript) e que juntos podem compor interfaces extremamente reaproveitáveis. Uma das características dos Web *components* é a possibilidade de usar eles como *tags* HTML customizadas, sendo fácil de usar, ler e entender como uma interface está sendo construída.

Infelizmente a palavra **reativo** ficou muito relacionada ao React, porém essa característica não é exclusiva do React. Há outras bibliotecas e *frameworks* que implementam essa técnica. Basicamente é observar um objeto JavaScript e refletir suas alterações no DOM do HTML. Quem já tentou isso com JavaScript puro ou até mesmo jQuery sabe que não é algo trivial, porém Vue torna isso extremamente simples

Criar componentes com Vue é muito simples e objetivo. Sua API é intuitiva e simples, seu sistema de *template* pega o que já estamos acostumados e torna muito simples, previsível e agradável. O grande destaque esta em como o código JavaScript é escrito, tudo possui seu lugar de maneira clara. Mesmo ele sendo extremamente flexível os caminhos que a informação e código percorrem são extremamente previsíveis, sendo muito fácil trabalhar em equipe e usar componentes de terceiros.

Vue possui uma performance excelente, em alguns testes, [Willmott \(2016\)](#), ele se saiu melhor do que o React e essa performance foi alcançada mesmo sem o VueJS fazer uso de Virtual-DOM como o React.

Uma característica do JavaScript, e por consequência também do Vue, é a capacidade de ser utilizado em qualquer tipo de projeto e tecnologia. Seu projeto pode estar sendo feito em Ruby, PHP, Python, NodeJs, Java, Go ou qualquer outra linguagem, você pode usar VueJS.

A escolha do Vue baseia-se nos fatos acima, além da curva de aprendizagem ser uma das mais curtas, segundo [Reis \(2016\)](#), e de acordo com uma pesquisa do [Galdino \(2017\)](#), o Vue possui uma média de aprovação dos desenvolvedores de 89%. Recebe cerca de 95 estrelas no GitHub por dia e é o 10º projeto mais votado no GitHub de todos os tempos.

2.4.1.1 Vuetify

O Vuetify é um *framework* responsivo em Vue baseado no *Material Design* do Google, possui uma boa gama de componentes e uma documentação sólida. [Vuetify \(2018\)](#) afirma que seu objetivo é fornecer componentes limpos, semânticos e reutilizáveis que facilitem a criação de seu aplicativo. Possui *design* responsivo e permite a visualização

das informações em diferentes tamanhos de telas, além de suportar todos os navegadores modernos (WEILER; OTTEQUIR; BECKER, 2017).

2.4.1.2 Vue Router

Vue Router é o roteador oficial para Vue. Ele se integra profundamente ao núcleo do Vue para facilitar a construção de SPA (VEDOVELLI, 2016). O Vue não impõe o uso desta biblioteca, mas usá-la faz com que se tenha uma integração mais consistente no *framework* e a garantia de que sempre será compatível no futuro.

O Vue Router que é responsável por mostrar ou esconder um ou mais elementos dependendo da URL que se acessa no *browser*. Esta é sua única responsabilidade, o que não significa que seja uma ferramenta simples.

Dentre as suas características, pode-se destacar: mapeamento de rota, visualização aninhada, configuração de roteador modular baseada em componente, controle de navegação refinado, comportamento de rolagem personalizável (VEDOVELLI, 2016).

2.4.1.3 Vue Navigator Online

Trata-se de um *plugin* para Vue.js para detectar o *status* atual da rede do usuário. A consulta deve ser feita diretamente no `navigator.onLine` e o retorno será um *boolean* de acordo com a situação.

2.4.1.4 ESLint Vue

ESLint busca e analisa padrões problemáticos ou certos estilos de codificação que não fazem parte de um estilo guia de escrita. Ele irá delatar o que está errado com a escrita do seu código e um arquivo de configuração que define como escrever métodos, declarar variáveis, etc, deverá ser utilizado, marcando então aquela linha que tem o problema e informar o mesmo (SILVA, 2016).

2.4.1.5 Vue Charts

Chart.js é uma biblioteca gráfica para representação de dados que utiliza o elemento `canvas` do HTML, CSS e JavaScript para renderizar os gráfico. É *open-source*, apresenta mais de oito diferentes tipos de gráficos e estes são responsivos. Os gráficos serem criados no navegador da Web proporciona uma página mais rápida e menos carga no servidor (CHART.JS, 2015).

O Vue Charts é um *wrapper* para Chart.js no Vue, que facilita a criação de componentes gráficos reutilizáveis.

2.4.2 NodeJs

Sistemas para Web desenvolvidos sobre plataforma .NET, Java, PHP ou Python paralisam um processamento enquanto utilizam um I/O no servidor, essa paralisação é conhecida como *Blocking-Thread* (PEREIRA, 2014a). Segundo Ihrig (2014), foi buscando contornar esse problema que o NodeJs foi criado em 2009, possuindo arquitetura totalmente *non-blocking thread* e utilizando apenas *single-thread*, única *thread* por processo, fazendo com que os sistemas que o utilizam não sofram de *dead-locks*. É escrita em C++ e em JavaScript; e é construída sobre o motor JavaScript do Google Chrome(V8).

Trata-se de uma plataforma altamente escalável, de baixo nível, orientado a eventos e segue a mesma filosofia de orientação de eventos do JavaScript *client-side* (PEREIRA, 2014a). Assim como o Gems do Ruby, ou o Maven do Java, o NodeJs também possui o seu próprio gerenciados de pacotes.

2.4.2.1 NPM

NPM é o nome reduzido de *Node Package Manager*, Gerenciador de Pacotes do Node, que trata-se de um gerenciador de pacotes para JavaScript e o maior registro de *software* do mundo, contendo mais de 600.00 pacotes (MORAES; OCUPACIONAIS, 2011). Ele é um utilitário de linha de comando que interage com este repositório *online*, que ajuda na instalação de pacotes, gerenciamento de versão e gerenciamento de dependências.

2.4.2.2 Express

O Express é um *framework* feito para Node.js, criado em 2009 por TJ Holowaychuck, lançado como *software* livre e de código aberto sob a licença MIT. Mínimo e flexível, fornece um conjunto robusto de recursos para aplicativos Web e móvel (ALMEIDA, 2015).

Segundo Brown (2014), uma das características mais atraentes do *framework* Express é sua natureza minimalista. Ao contrário da maioria dos *frameworks* em que são acoplados vários módulos que muitas vezes não são utilizados, o Express utiliza a ideia de “menos é mais”, oferecendo um ambiente enxuto, porém com a capacidade de adicionar módulos para necessidades específicas.

O *framework* possui implementações para a manipulação de rotas, *cache*, *cookies* e outros recursos que podem ser adicionados de acordo com a necessidade (HAHN, 2016).

2.4.2.3 Mongoose

Mongoose é uma biblioteca do Nodejs que proporciona uma solução baseada em esquemas para modelar os dados da sua aplicação. Ele possui sistema de conversão de tipos, validação, criação de consultas e *hooks* para lógica de negócios.

Segundo [NodeBr \(2016\)](#), o Mongoose fornece um mapeamento de objetos do MongoDB similar ao ORM (*Object Relational Mapping*), ou ODM (*Object Data Mapping*) no caso do Mongoose. Isso significa que o Mongoose traduz os dados do banco de dados para objetos JavaScript para que possam ser utilizados por sua aplicação.

2.4.2.4 Nodemon

"O Nodemon é um utilitário que monitora quaisquer alterações em sua origem e reinicia automaticamente seu servidor, perfeito para o desenvolvimento" ([NODEMON.IO, 2018](#)). Não requer alterações adicionais ao código e a sua instalação pode ser feita usando NPM.

2.4.2.5 Passport

É *middleware* de autenticação para NodeJs . Extremamente flexível e modular, o Passport pode ser acessado de forma não invasiva em qualquer aplicativo da Web baseado no Express.

O passport-jwt é estratégia do Passport para autenticação com um JWT.

2.4.2.5.1 Bcrypt

A biblioteca bcrypt no NPM facilita muito o *hash* e a comparação de senhas no NodeJs. Segundo [LaViska \(2007\)](#), o motivo pelo qual deve-se usar *hashes* unidirecionais em vez de criptografia é que *Hashing* é uma função irreversível, ao se aplicar o algoritmo não é possível recuperar a *string* original, o máximo possível seria encontrar uma outra *string* que forneça o mesmo *hash*, chamado de colisão. Já a criptografia é uma função de duas vias e conseqüentemente reversível, com a chave é possível obter a *string* original.

2.4.3 Webpack

Webpack um empacotador de código para projetos Web. "Quando ele processa seu aplicativo, ele cria internamente um gráfico de dependência que mapeia todos os módulos que seu projeto precisa e gera um ou mais pacotes configuráveis" ([WEBPACK.JS.ORG, 2015](#)). Modulariza partes reaproveitáveis do seu projeto, facilitando o desenvolvimento independente.

2.4.4 Axios

É uma biblioteca para comunicação HTTP para o navegador e NodeJs, fazendo solicitações AJAX. Ele usa promessas por padrão e dentre suas características se pode destacar: faz **XMLHttpRequests** a partir do navegador, requisições HTTP de NodeJs,

transforma dados de solicitação e resposta, é possível realizar transformações automáticas para dados JSON e possui suporte do lado do cliente para proteção contra CSRF¹ ([VUE.JS.ORG, 2017](#)). Pode combiná-lo com *async/wait* para obter uma API incrivelmente concisa e fácil de usar.

Existe uma biblioteca do próprio Vue chamada Vue-Resouce que possui praticamente as mesmas finalidades do Axios, mas segundo [Seo \(2018\)](#), ela não é bem usada devido ao seu ciclo lento de atualizações e à maior comunidade do Axios.

2.4.5 Babel

O Babel é um compilador JavaScript, *open-source* e gratuito usado no desenvolvimento Web. É uma ferramenta que é usada principalmente para converter o código ECMAScript 2015+ em uma versão compatível com versões anteriores do JavaScript em navegadores ou ambientes atuais e antigos ([PATEL, 2016](#)).

Segundo [Groner \(2018\)](#), por ser altamente configurável, permite usar *features* experimentais e extensões, compilando de volta para o JavaScript antigo que pode ser suportado em uma ampla quantidade de plataformas. Babel permite o *debug* do código-fonte incluindo *source maps* com o JavaScript compilado.

2.5 Bases de Dados

Os primeiros Sistemas de Gerenciadores de Bancos de Dados (SGDBs) surgiram por volta de 1960 e foram desenvolvidos com base nos primitivos sistemas de arquivos ([LÓSCIO; OLIVEIRA; PONTES, 2011](#)). Atualmente existem diversas ofertas, cada uma com suas particularidades. É necessário analisar as características do negócio e da aplicação para decidir qual deles melhor se adapta.

Durante anos os bancos de dados relacionais têm sido as principais tecnologias de armazenamento. Nesse tipo de SGBDs, os dados são armazenados de maneira fixa, estruturados e representados por meio de tabelas e relacionamentos entre essas tabelas. [Silva \(2017\)](#) afirma que armazenar as informações de maneira estruturada permite recuperar informações de forma eficiente. Dentre os motivos do grande sucesso do modelo relacional destacam-se a padronização de conceitos, sua base formal e a facilidade de uso da linguagem SQL (*Structured Query Language*), linguagem padrão para consultas e manipulação de dados relacionais ([LÓSCIO; OLIVEIRA; PONTES, 2011](#)). SGBDs do tipo relacional geralmente adotam uma arquitetura centralizada e garantem a manutenção das propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) entre suas

¹ Do inglês *Cross-site request forgery*- Falsificação de solicitação entre sites, também conhecida como ataque ou sessão com um clique. É um tipo de exploração maliciosa de um site em que comandos não autorizados são transmitidos de um usuário que o aplicativo da Web confia ([SOUZA, 2012](#)).

transações; porém têm como principal desvantagem a dificuldade de tornar os banco de dados escaláveis e baratos (FALCÃO, 2014).

Com a evolução das aplicações de banco de dados, foi surgindo a necessidade de manipulação de outros formatos de dados como som, vídeo e imagens. Assim, novas soluções foram propostas para atender esses novos requisitos, como os bancos de dados orientados a objetos e os banco de dados objeto-relacionais.

Posteriormente, com o advento das aplicações Web, novos requisitos de banco de dados foram surgindo. Sendo essencial que as aplicações que gerenciam esses dados tolerem falhas, disponibilidade e escalabilidade; além da necessidade de manipulação de dados não estruturados ou semi-estruturados (SCAVUZZO; NITTO; CERI, 2014). Com o objetivo de atender essas necessidades surgiram os SGBDs NoSQL.

Segundo Lóscio, OLIVEIRA e PONTES (2011), inicialmente as propostas de bancos de dados não relacionais foram desenvolvidas por pequenas empresas e por comunidades de *software* livre. Tais soluções foram então agrupadas em um termo, NoSQL (*Not Only SQL*), que significa “não apenas SQL”. Este termo faz referência a SGBDs que não adotam o modelo relacional e são mais flexíveis quanto às propriedades ACID.

SGBDs NoSQL são projetados para atender aos requisitos de desempenho e escalabilidade que não podem ser abordados por bases de dados relacionais tradicionais (NOGUERA et al., 2018). Por esse motivo, segundo Chandra (2015), são utilizados principalmente em aplicativos baseados na Web, apesar de não serem dedicados exclusivamente a esse domínio.

Os bancos de dados NoSQL normalmente atuam sem um esquema, permitindo que sejam adicionados, livremente, campos aos registros do banco de dados, sem ter de definir primeiro quaisquer mudanças na estrutura. Podem ser classificados em quatro modelos de dados principais: chave-valor, colunar, documentos e grafos.

Pokorny (2013) afirma que nos bancos de dados orientados a documentos os dados não são armazenados em tabelas, mas em documentos. Cada documento pode compreender valores escalares, listas ou mesmo documentos aninhados; pode não haver esquema para documentos, e cada documento pode ter seus próprios atributos, definidos em tempo de execução. Os documentos armazenados nestes sistemas são normalmente de tipos bastante comuns como XML, JSON, PDF, entre outras (NAYAK; PORIYA; POOJARY, 2013). Os documentos não precisam respeitar necessariamente um esquema. Normalmente, todos os documentos de uma coleção têm um propósito relacionado (STANESCU; BREZOVAN; BURDESCU, 2016).

Características como modelo de dados persistente, replicação de documentos, distribuição automática entre servidores e estrutura de índice podem ser aplicadas ao modelo de armazenamento de documentos (MATHEW; KUMAR, 2015).

Como NoSQL envolve tecnologias relativamente recentes, a falta de padrão é uma grande preocupação para organizações interessadas em adotar qualquer um destes sistemas (STONEBRAKER, 2011). A falta de um esquema predefinido pode também dificultar o desenvolvimento, o aspecto desnormalizado de bancos de dados NoSQL faz que o desempenho de consulta seja altamente dependente de caminhos de acesso. Por esse motivo a modelagem no modelo NoSQL deve ser feita buscando maximizar o desempenho das consultas (NOGUERA et al., 2018).

Noguera et al. (2018) em seu trabalho demonstra como a implementação das consultas e a eficiência dos resultados podem mudar conforme a estrutura adotada, exigindo estudos cuidadosos no projeto inicial.

2.5.1 MongoDB

MongoDB é um SGBD NoSQL de código aberto e orientado a documentos mais popular do mundo (RANKING, 2018). Sua primeira versão foi lançada em 2007 e fornece recursos sofisticados como alta disponibilidade, baixo custo de manutenção e operação além de flexibilidade para lidar com dados. É possível utilizá-lo em diferentes sistemas operacionais, como Windows, Linux, OS X e Solaris. Possui *drivers* para diversas linguagens de programação, entre elas: C, C++, Java, Perl, PHP, Python, Ruby e NodeJs (LÓSCIO; OLIVEIRA; PONTES, 2011). Por estes motivos, este modelo de dados NoSQL foi escolhido para fins deste trabalho.

Segundo Silva (2017), o MongoDB não impõe uma estrutura fixa a seus documentos, esta flexibilidade facilita o mapeamento de documentos a uma entidade ou objeto. Relacionamento entre os dados armazenados podem ser representados através de referência e/ou documentos embutidos. O uso de referências consiste no armazenamento da relação entre os dados e os documentos embutidos consiste em capturar as relações entre os dados e armazená-las em um único documento. Documentos embutidos são úteis quando você deseja retornar o documento inteiro nas buscas, sem refinar muito a *query*.

2.6 *Single-Page Applications*

Single-page applications (SPAs) é um conceito de desenvolvimento de um *website* com uma única página, ela atualiza sua interface à medida que seus usuários interagem, sem a necessidade de recarregar todo o conteúdo. Essas aplicações foram ganhando espaço por prover uma experiência mais próxima de aplicações nativas e por oferecer uma economia no uso da rede, visto que a comunicação cliente-servidor agora envolve majoritariamente dados, e não uma interface completa (OLIVEIRA, 2017). Todo o material necessário para a execução deste tipo de *website* é retirado do servidor no carregamento inicial da página

ou é dinamicamente retirado ao longo da execução da aplicação dependendo da interação do utilizador.

Quando os navegadores de internet foram concebidos, os seus desenvolvedores não tinham pensado no conceito de SPA, como tal este cenário causou inúmeras dificuldades. Segundo Duarte (2015), o HTML5 foi uma grande ajuda para consolidar este conceito nas páginas Web. Este tipo de cenários são normalmente utilizados quando é necessário desenvolver o lado do cliente de raiz com inúmeras funcionalidades sem grandes complicações. Em alguns casos será necessário alterar o lado do servidor para suportar uma interface em REST e JSON para conseguir integrar melhor com a plataforma.

Este conceito de SPA é bastante similar ao conceito *Single Document Interface* que é muito popular em aplicações nativas para computador.

Aplicações SPA que utilizam a pilha de tecnologia MEVN apresenta dois problemas: a API é *stateless*² e a lógica da aplicação é entregue completa ao navegador, portanto não é possível limitar exposição do código. Segundo Holmes (2016), a solução lógica para esse problema é manter no próprio navegador algum tipo de estado de sessão e deixar então que a aplicação decida o que se pode ou não mostrar ao usuário atual. O JWT, sessão 2.6.1, é objeto capaz de manter de maneira eficiente esses dados no navegador para gerir a sessão.

2.6.1 JWT

O JWT é um objeto JSON criptografado e compactado que pode ser entendido apenas pela aplicação e servidor.

Em sua forma compacta, JWT consiste em três partes: *Header*, *Payload* e *Signature*. O *Header* define informações sobre o tipo do *token* e o algoritmo de criptografia usado em sua assinatura. *Payload* contém informações da entidade tratada, normalmente o usuário autenticado. *Signature* é a concatenação dos *hashes* gerados a partir do *Header* e *Payload*, com uma chave secreta que apenas o servidor de origem conhece.

2.7 Dispositivo para Monitoramento da Temperatura

No ano de 2008, o número de dispositivos conectadas à Internet excedeu o número de pessoas existentes no planeta Terra (CISCO, 2011). Eles não são apenas *smartphones* e *tablets*, mas todo dispositivo capaz de transmitir dados pela rede, desde pequenos sensores para monitoramento de um ambiente a microcontroladores com Inteligência Artificial.

² Não orientada a estado, uma vez que o Express e o NodeJs não trabalham com conceito de sessões de usuário (SANDOVAL, 2017).

O vínculo da Internet com sistemas microprocessados possibilita o conceito de “Internet das coisas”, do termo original em inglês *Internet of Things* (SINGER, 2012). Esse conceito extrapola a ideia da Internet como uma rede de computadores global e interconectada, para descrever uma rede de coisas (dispositivos) interconectados, tais como objetos do dia a dia e produtos e objetos do ambiente.

Segundo Cantú (2013), esses sistemas são geralmente denominados de sistemas embarcados, que executam funções específicas para as quais eles foram programados. Eles estão presentes na maioria dos equipamentos eletrônicos e podem interagir com sensores e atuadores obtendo dados do ambiente e interagindo, ou seja, enviando instruções ou comandos para outros dispositivos.

Existem diversas plataformas de prototipagem eletrônica disponíveis no mercado para o desenvolvimento de projetos eletrônicos de baixo custo, dentre as principais estão o Arduino e o Raspberry Pi, ambas de *hardware* e *software* aberto.

Em seu trabalho Cantú (2013) utilizou uma placa Arduino e um sensor DHT11 para montar um dispositivo para ler a temperatura. Foi criado um aplicativo Python responsável por fazer a comunicação serial com o Arduino e obter os dados das leituras do sensor, salvando os registros em um banco de dados.

Sandoval (2014) realizou a montagem do dispositivo de monitoramento usando Arduino Uno, sensor de umidade e temperatura DHT11 e *bluetooth* HC-05. Para a comunicação com o Arduino, foi desenvolvido uma GUI em C++ utilizando o Qt Creator como IDE.

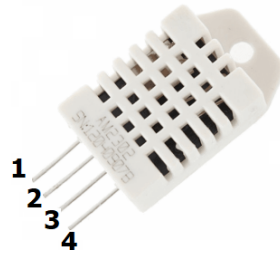
Silva (2018) utilizou um sensor DHT11 e um NodeMCU para fazer o monitoramento da temperatura e umidade do interior de veículos. A análise dos dados foi realizada pelo serviço *online* ThingSpeak, explicado melhor na seção 2.7.4 deste trabalho, com alerta para o usuário através da rede social Twitter.

O dispositivo montado por Queiroz et al. (2016) não utilizou Arduino e sim um microcontrolador Raspberry Pi e um sensor DHT11. O armazenamento dos valores obtidos pelos sensores e data atual do sistema foi feito através do banco de dados SQLite.

2.7.1 Sensor DHT22

O DHT22 ou AM2302, Figura 4, é um sensor de temperatura e umidade que permite fazer leituras de temperaturas entre -40 a +80 graus Celsius e umidade entre 0 a 100%, sendo muito fácil de usar com Arduino, Raspberry e outros microcontroladores pois possui apenas 1 pino com saída digital (CIA, 2015). Possui quatro pinos, onde o pino 1 deve ser ligado ao pino positivo da fonte de tensão, o pino 2 ligado a um pino de dados do microcontrolador, o pino 3 é não conectado e o pino 4 ligado ao pino negativo da fonte de alimentação(terra).

Figura 4 – Sensor DHT22.



Adaptado de: (CIA, 2015).

Esse sensor foi escolhido devido a suas características e comunicação: a faixa de medição de DHT22 é bem mais abrangente que a do DHT11, enquanto a menor temperatura medida pelo DHT11 é de 0°C o DHT22 pode medir até -40°C com uma resolução de 0.1°C comparada ao DHT11 que sua resolução é de 1°C.

2.7.2 Raspberry Pi

Raspberry Pi é um computador de baixo custo e que tem o tamanho de um cartão de crédito. Abriga processador, processador gráfico, slot para cartões de memória, interface USB, HDMI e seus respectivos controladores (RICHARDSON; WALLACE, 2013). Além disso, ele também apresenta memória RAM, entrada de energia e barramentos de expansão. Ainda que minúsculo, o Raspberry é um computador completo. Para usá-lo, basta conectar um teclado e um mouse padrão a ele e conectar tudo isso a um monitor ou a uma televisão.

A função básica é oferecer uma alternativa prática e acessível para explorar todas as capacidades da computação. Além disso, também visa facilitar a aprendizagem de programação em linguagens como Scratch e Python.

O sistema operacional deverá ser instalado em um cartão de memória SD, já que o computador não apresenta disco rígido próprio. Com o sistema operacional, você pode usar o Raspberry Pi para navegar na Internet, escrever textos, ver vídeos, ouvir música, criar planilhas e realizar praticamente qualquer tarefa possível num computador convencional.

A placa Raspberry Pi Permite, assim como o Arduino, que você ligue sensores, *displays* e outros componentes utilizando o conector GPIO. GPIO significa *General Purpose Input/Output*, ou Entrada e saída de uso geral, em tradução livre.

Como o Raspberry são computadores, torna-se necessária a existência de um sistema operacional, geralmente baseado em linux, para as mesmas que possam ter a capacidade de executar tarefas. Alguns sistemas operacionais compatíveis são: Raspbian, Windows 10 IoT, Ubuntu, OSMC, Recalbox e Pidora.

O Raspberry pode ser programado utilizando diversas linguagens como: Python, C++, Java, entre outros. Mas segundo Telecomunicações (2009), Python é linguagem de

programação preferida da equipe do projeto, o **Pi** do nome vem de Python. Sendo assim essa linguagem foi escolhida para se realizar a programação.

Apesar do Raspberry ser destinado a usos mais complexos do que de um Arduino, ele foi escolhido por a autora ainda não ter trabalhado com esse dispositivo além do orientador ter mais experiência para auxiliá-la.

2.7.3 Python

Python é uma linguagem de programação interpretada, de código-fonte aberto e disponível para vários sistemas operacionais. Diz-se que uma linguagem é interpretada se esta não precisa ser compilada, mas sim lida por um outro programa, chamado de interpretador, que traduzirá para a máquina o que seu programa quer dizer. O interpretador para Python é interativo, ou seja, é possível executá-lo sem fornecer um *script* (programa) para ele (TELECOMUNICAÇÕES, 2009).

Python suporta paradigma orientado a objetos, imperativo, funcional e procedural, por isso é dita multi paradigma. Possui tipagem dinâmica e uma de suas principais características é sua sintaxe da linguagem minimalista, isso é, mantém somente o necessário, consequentemente a manutenção do código se torna mais fácil e compreensível.

2.7.4 ThingSpeak

O ThingSpeak é um serviço de plataforma de analítica da IoT que permite agregar, visualizar e analisar fluxos de dados ao vivo na nuvem. Fornece visualizações instantâneas de dados postados por seus dispositivos, com a capacidade de executar o código MATLAB no ThingSpeak você pode realizar a análise e o processamento *online* dos dados conforme eles são recebidos.

Segundo ThingSpeak (2018), algumas das principais capacidades do ThingSpeak incluem a capacidade de configurar facilmente dispositivos para enviar dados para o ThingSpeak usando protocolos IoT populares, visualizar seus dados do sensor em tempo real, agregar dados sob demanda de fontes de terceiros, usar o poder do MATLAB para entender seus dados de IoT, execute sua analítica da IoT automaticamente com base em agendamentos ou eventos, prototipia e construir sistemas de IoT sem configurar servidores ou desenvolver *software* da Web e atuar automaticamente em seus dados e comunique-se usando serviços de terceiros, como o Twitter.

O ThingSpeak não é totalmente gratuito, sendo que para usufruir de alguns recursos é necessário adquirir um plano pago. Contas gratuitas oferecem uma experiência totalmente funcional, com limites em certas funcionalidades, pode-se enviar por dia 8.200 mensagens e o limite do intervalo de atualização de mensagens é a cada 15 segundos.

2.8 Ferramentas auxiliares

2.8.1 Sublime Text

De acordo com [Trindade \(2016\)](#), o Sublime Text é “um editor de texto leve, possui uma interface limpa e fácil de usar, mas é altamente flexível, podendo se adaptar a diferentes tipos de profissionais”.

Contempla diversos formatos de códigos como: C, C++, HTML, Haskell, Java, LaTeX, PHP, Ruby, SQL, JavaScript, Vue e várias outras. Por meio de *plugins* a IDE (*Integrated Drive Electronics*) disponibiliza vários recursos exclusivos tornando a programação produtiva ([PEREIRA, 2014b](#)).

2.8.2 Postman

O Postman é uma aplicação que está disponível na Web Store da Google e fornece uma interface gráfica para requisições HTML, além de ter outras características como a possibilidade de salvar requisições e compartilhá-las com uma equipe ([INC, 2017](#)).

O Postman contém uma versão *open source* e uma paga. A diferença entre a versão paga e a versão gratuita é que a segunda não permite executar um conjunto de testes unitários simultaneamente, conhecido como *Loading Test*, além de não permite o acesso a fontes de dados externas.

2.8.3 Lucidchart

Trata-se de um site de colaboração visual com base em HTML5 que torna a criação de diagramas mais rápida e fácil. Com ela você pode trabalhar junto a um número ilimitado de pessoas para criar e editar diagramas em tempo real, com alterações incorporadas e sincronizadas instantaneamente. Possui integração com o Google Drive, podendo assim ter os documentos criados também na nuvem. O Lucidchart não é totalmente gratuito, sendo que para usufruir de alguns recursos é necessário adquirir um plano pago.

2.8.4 Fritzing

Fritzing é um programa *open-source* que possibilita a montagem de circuitos em um ambiente amplo com vários componentes eletrônicos como Arduino, Raspberry Pi, ou mesmo somente a matriz de contatos e alguns componentes eletrônicos.

2.9 Sistemas Similares

2.9.1 MarketUP

O MarketUP é um sistema de gestão gratuito, online e sem limitação de uso; voltado para pequenas e micro empresas. Ele é voltado para atender diversos segmentos e não somente o de sorveterias, “com o propósito de ajudar o pequeno empreendedor a conseguir resultados mais eficazes a fim de garantir o melhor gerenciamento do seu negócio” (MARKETUP, 2017).

Nesse sistema se tem acesso à condições especiais para comprar certificado digital, equipamentos e mais. Possui uma equipe de suporte e todas as informações são criptografadas e com backups realizados diariamente. Dentre suas funcionalidades, pode-se destacar: loja virtual, PDV, controle financeiro, relatórios, emissão de notas fiscais e controle de estoque.

O MarketUP não apresenta nenhuma funcionalidade voltado para o gerenciamento de empresas com filiais e não conta com nenhum dispositivo para auxiliar no monitoramento da temperatura.

2.9.2 Consumer

O Consumer é um programa voltado para restaurantes, lanchonetes, pizzarias e similares. Segundo Consumer (2018), com ele é possível melhorar o fluxo de pedidos, controle de produção dos pratos e entrega, a gestão e relatórios são descomplicados podendo assim acompanhar o desempenho do negócio.

Algumas funcionalidades não se encaixam nas regras de negócio da Sorveteria Sol e Verão, como por exemplo a integração com o IFood, painel de senha, bina identificador de chamadas e comanda móbile para garçom; além de não apresentar uma gestão voltada para filiais. Para utilizar o programa Consumer é necessário adquirir um plano, que na versão completa tem um valor de R\$ 900 ,00.

2.9.3 CTP Sorveteria

O CTP Sorveteria é um *software* para o gerenciamento para sorveteria, com o objetivo de fornecer organização, controle, segurança, agilidade e modernização. Dentre suas principais funcionalidades pode-se destacar: mapa de mesas, gerenciamento financeiro e de estoque, gerenciamento de dados, emissão de relatórios e gerenciamento de compromissos (CPT, 2018).

Trata-se de um *software* pago, aproximadamente R\$840, sendo que para que o sistema seja completo é necessário adquirir outros módulos da empresa, como por exemplo

o módulo produção. Outra desvantagem é que o sistema não é *online*, o que ocasiona um problema quando o gerenciamento é feito a distância; além de não ter nenhuma funcionalidade voltado para filiais.

3 MATERIAIS E MÉTODOS

Para o elaboração do sistema utilizou-se um computador pessoal cuja especificações encontram-se na Tabela 1.

Tabela 1 – Configuração do computador pessoal.

Processador Intel Core i7 3630QM 2,4 GHz
Memória RAM 6GB
Armazenamento 1TB
Processador Gráfico Intel® HD Graphics.
Sistema operacional Windows 10

Fonte: Elaborado pela autora.

Os *softwares* utilizados no desenvolvimento do projeto foram:

- Sublime Text versão 3.1.1;
- Postman versão 5.5.3;
- Google Chrome versão 69.0.3497.100;
- Lucidchart;
- Fritzing;
- ThingSpeak.

A seguir, são apresentadas as tecnologias de desenvolvimento e suas respectivas versões utilizadas no protótipo do sistema e programação do Raspberry Pi.

- Vue versão 2.9.6;
- Vuetify versão 1.2.5;
- Vue Router versão 3.0.1;
- Vue Navigator Online versão 0.0.2;
- ESLint versão 4.15.0;
- NodeJs versão 8.11.3;
- Express versão 4.16.0

- Mongoose versão 5.2.13;
- Nodemon versão 1.18.4;
- Passport-jwt versão 4.0.0;
- Bcrypt versão 0.0.3;
- Axios versão 0.18.0;
- Babel versão 6.22.1;
- MongoDB versão 4.0.1;
- Vue Chart 0.3.24;
- Python versão 2.7.12.

Abaixo estão listados os materiais utilizados para desenvolvimento do dispositivo para monitoramento da temperatura.

- *Protoboard*;
- Cabos para conexão;
- Sensor de temperatura DHT22;
- Resistor de de 100k Ω ;
- Raspberry Pi 3 Model B, melhor abordado na seção [3.1](#).

3.1 Raspberry Pi 3 Model B

O Raspberry Pi 3 Modelo B, Figura 5, é o modelo mais antigo da terceira geração do Raspberry Pi. Substituiu o Raspberry Pi 2 Modelo B em fevereiro de 2016.

Figura 5 – Raspberry Pi 3 Model B.



Fonte: (RASPBERRYPI.ORG, 2014).

Segundo (RASPBERRYPI.ORG, 2014), dentre suas especificações pode-se destacar:

- CPU Quad Core 1.2GHz Broadcom BCM2837 64bit;
- 1 GB de RAM;
- Rede sem fio e Bluetooth;
- 4 portas USB 2;
- GPIO estendido de 40 pinos, Figura 6;
- Full size HDMI;
- Porta da câmera CSI para conectar uma câmera Raspberry Pi;
- Porta de exibição DSI para conectar uma tela sensível ao toque do Raspberry Pi;
- Porta Micro SD para carregar seu sistema operacional e armazenar dados;
- Fonte de energia Micro USB comutada atualizada até 2,5A.

Figura 6 – Raspberry Pi 3 GPIO Header.

Pin#	NAME		NAME	Pin#
01	3.3v DC Power	●	DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)	●	DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)	●	Ground	06
07	GPIO04 (GPIO_GCLK)	●	(TXD0) GPIO14	08
09	Ground	●	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	●	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	●	Ground	14
15	GPIO22 (GPIO_GEN3)	●	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	●	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	●	Ground	20
21	GPIO09 (SPI_MISO)	●	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	●	(SPI_CE0_N) GPIO08	24
25	Ground	●	(SPI_CE1_N) GPIO07	26
27	ID_SD (I ² C ID EEPROM)	●	(I ² C ID EEPROM) ID_SC	28
29	GPIO05	●	Ground	30
31	GPIO06	●	GPIO12	32
33	GPIO13	●	Ground	34
35	GPIO19	●	GPIO16	36
37	GPIO26	●	GPIO20	38
39	Ground	●	GPIO21	40

Fonte: (RICHARDSON; WALLACE, 2013).

3.1.1 Raspbian

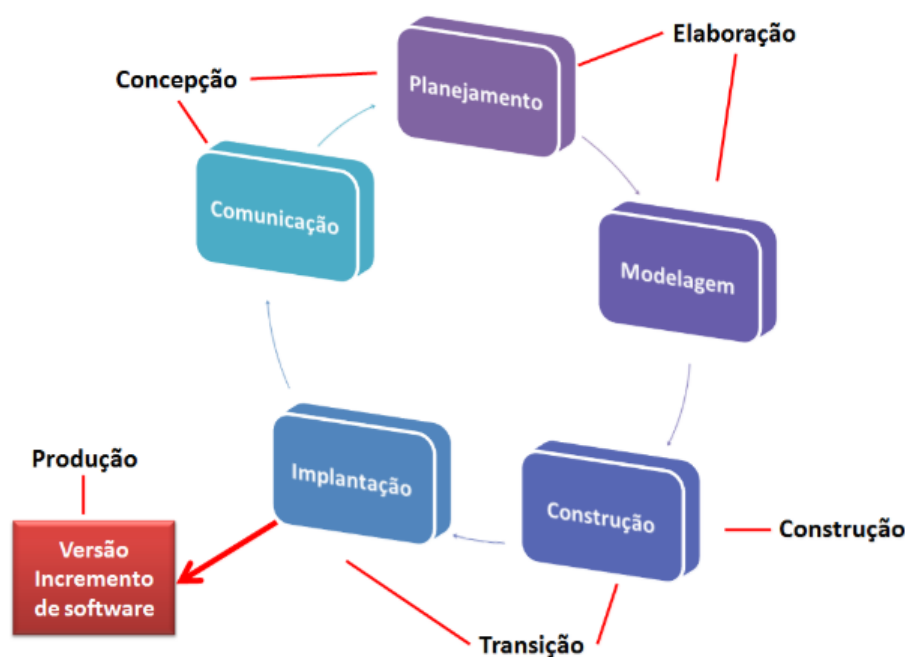
O Raspbian é a distribuição oficialmente recomendada da Fundação Raspberry, com base no Debian. É um *software* livre com mais de 35.000 pacotes pré-compilados que podem ser facilmente instalados (HEIN, 2013). O NOOBS foi utilizado para instalar de maneira fácil o sistema operacional Raspbian no cartão SD.

3.2 Metodologia

O presente trabalho foi inspirado na metodologia baseada no Processo Unificado (PU), trata-se de um modelo iterativo e adaptativo, desta forma consegue produzir um sistema de grande porte como se fosse vários pequenos sistemas, o que diminui o risco do projeto. Surgiu como um processo popular para o desenvolvimento de *software* visando à construção de sistemas orientados a objetos (PRESSMAN; MAXIM, 2016).

Consiste na repetição de uma série de ciclos durante o desenvolvimento de um sistema, por isso esse processo é dito como evolucionário. Cada ciclo é dividido em 4 Fases: Concepção, Elaboração, Construção e Transição, Figura 7. Cada ciclo resulta em um incremento, que é uma versão do sistema que contém funcionalidades adicionais ou melhoradas em comparação com a versão anterior (LARMAN, 2002). O PU usa a UML como linguagem de modelagem.

Figura 7 – Fases do PU.



Fonte: (PRESSMAN; MAXIM, 2016).

"Ao invés de combater a inevitável mudança que ocorre no desenvolvimento de *software* (principalmente nas fases iniciais), o PU prega uma atitude de aceitar a mudança e a adaptação como fatores inevitáveis e, de fato essenciais" (CARDOSO, 2018).

Breves detalhes sobre cada uma das fases são apresentados a seguir, nas subseções: 3.2.1, 3.2.2, 3.2.3 e 3.2.4. Desta forma o protótipo do sistema evoluiu seguindo esta metodologia baseada no PU, onde as fases foram repetidas desde o momento da concepção do protótipo até a sua finalização. A seção 3.2.5 descreve as macro entregas do projeto.

3.2.1 Concepção

Cada iteração iniciou-se com o entendimento dos requisitos de negócio para o *software*. O planejamento identifica recursos, avalia os principais riscos, define um cronograma e estabelece uma base para as fases que devem ser aplicadas à medida que o incremento de *software* é desenvolvido.

São exemplos de atividades desenvolvidas nesta fase as reuniões com o administrador das filiais e com o orientador, onde foi selecionado o escopo do projeto e um cronograma.

3.2.2 Elaboração

A elaboração refina e expande os casos de uso preliminares com a definição dos requisitos funcionais e modelagem do protótipo da aplicação em desenvolvimento. Além

de um cronograma mais realista do que o elaborado na fase anterior, essa fase permite identificar o real tamanho do sistema.

3.2.3 Construção

Após modelar o conteúdo da fase avança-se para a construção, onde o sistema é efetivamente desenvolvido e em geral, tem condições de ser operado, mesmo que em ambiente de teste.

3.2.4 Transição

Após a construção, o resultado da iteração é exposto ao orientador e ao administrador das filiais, para que este possa ser avaliado e um *feedback* é realizado sobre efeitos e modificações necessárias. Então incrementos do sistema são implantados e defeitos são corrigidos, se necessário.

3.2.5 Macro Entregas do Projeto

As fases da metodologia baseada no PU foram replicadas para cada iteração até completar o escopo do projeto. A primeira macro entrega foi modelagem e implementação das operações básicas relacionadas funcionários, lojas e produtos. Posteriormente foi desenvolvido o caixa e as operações de caixa, juntamente com a movimentação do estoque. A próxima macro entrega tratou-se da implementação do gerenciamento de autenticação, sessão e criptografia das senhas. Em seguida foi produzido o dispositivo para monitoramento de temperatura e a comunicação do mesmo com o sistema. A última macro entrega contemplou a geração de relatórios simples de fluxo de caixa, assim como a escrita da monografia deste Trabalho de Conclusão de Curso.

4 PROJETO E DESENVOLVIMENTO

4.1 Modelagem

A modelagem é uma ferramenta fundamental no desenvolvimento de *softwares*, pois assim se pode compreender melhor o sistema que estamos desenvolvendo. Os modelos ajudam a visualizar o sistema como ele é ou como desejamos que seja, permitem especificar a estrutura ou o comportamento de um sistema, proporcionam um guia para a construção do sistema, além de documentar as decisões tomadas.

4.1.1 Diagrama de Caso de Uso

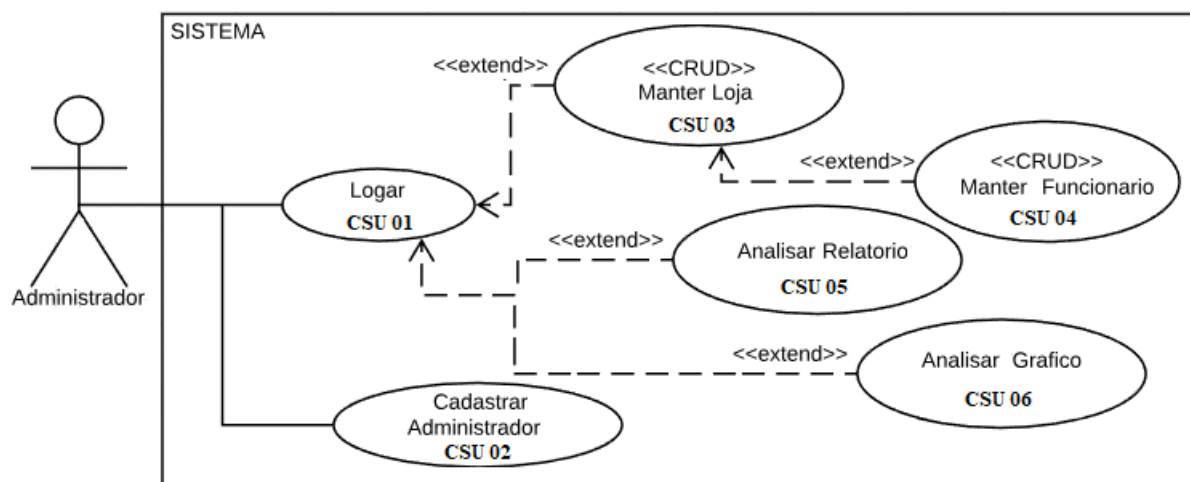
Um diagrama de caso de uso exhibe um conjunto de casos de uso, atores e seus relacionamentos. Segundo Larman (2002), o diagrama representa uma possível utilização do sistema por um ator, que pode ser uma pessoa, dispositivo físico, mecanismo ou subsistema que interage com o sistema alvo, utilizando algum de seus serviços.

A seguir são mostrados os casos de usos do sistema Web desenvolvido e no Apêndice A estão as expansões.

4.1.1.1 Administrador

A Figura 53 demonstra as funcionalidades que o administrador do sistema é capaz de realizar.

Figura 8 – Diagrama de Caso de Uso para administrador.

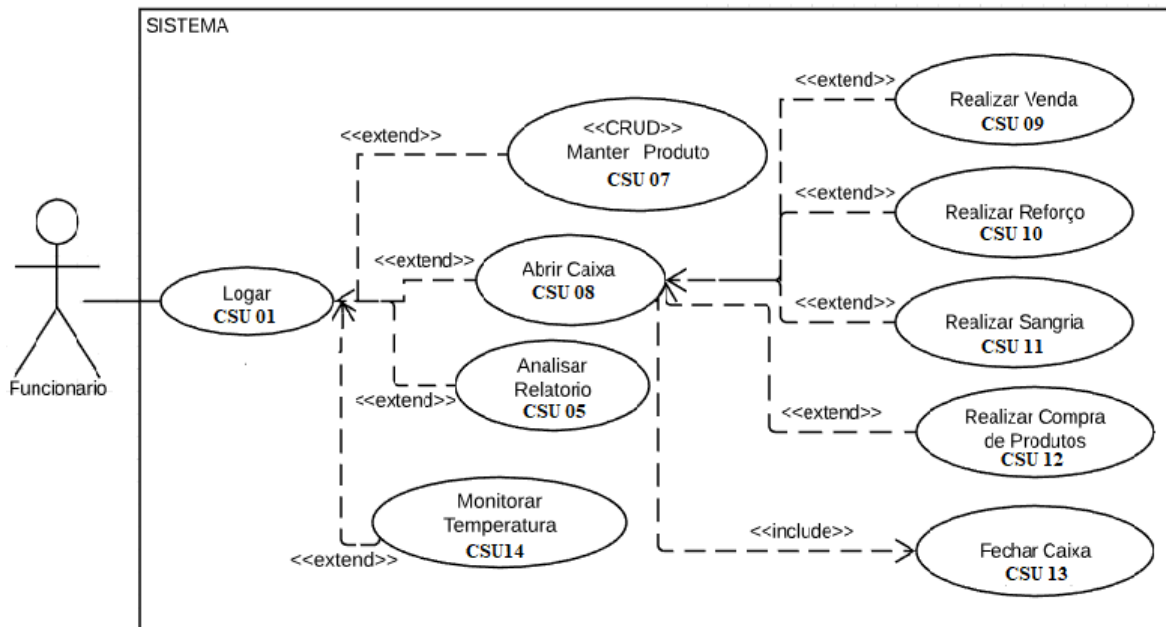


Fonte: Elaborado pela autora.

4.1.1.2 Funcionário

A Figura 54 demonstra as funcionalidades que o funcionário do sistema é capaz de realizar.

Figura 9 – Diagrama de Caso de Uso para funcionário.



Fonte: Elaborado pela autora.

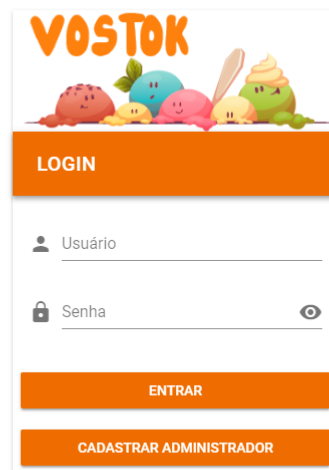
4.2 Interfaces

O sistema foi denominado **Vostok**, este nome foi escolhido por se tratar de uma região na Antártica e é considerada o ponto mais frio da Terra com a temperatura mais baixa registrada no planeta. O sistema é baseado na cor laranja pois ela tem caráter estimulante e relaciona-se com produtos comestíveis (PAULINO, 2015).

A seguir serão apresentadas as interfaces que interagem com o usuário.

4.2.1 GUI Login

O acesso ao sistema é iniciado a partir de um *login*, Figura 10, apresenta dois campos que devem ser preenchidos: usuário e senha. A senha pode ser revelada ao clicar no símbolo do olho na lateral do campo. Realizada a autenticação o usuário é redirecionado para sua página *Home*, mas se houver algum erro então mensagens de alerta, localizadas na parte superior da tela, informam ao usuário o motivo da falha. A Figura 11 ilustra um erro de autenticação de senha. Caso o usuário ainda não possua cadastro, ao clicar em **cadastrar administrador** abrirá a tela de cadastramento.

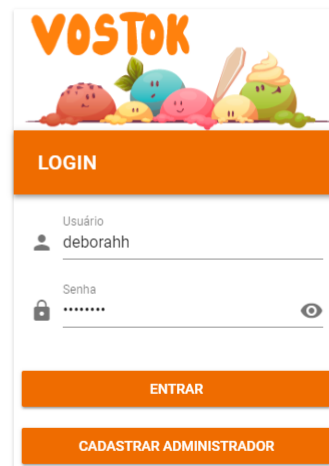
Figura 10 – GUI *Login* do sistema.

©2018 – Déborah Resende

Fonte: Elaborado pela autora.

Figura 11 – GUI *Login* do sistema com erro de autenticação.

▲ Falha na autenticação. Senha incorreta.



©2018 – Déborah Resende

Fonte: Elaborado pela autora.

4.2.2 GUI Cadastro Administrador

A GUI de cadastro de administrador, Figura 12, apresenta nove campos a serem preenchidos: nome completo, data de nascimento, sexo, CPF, endereço, telefone, *email*, usuário e senha. Enquanto todas as informações de todos os campos não forem consideradas

válidas o botão de cadastrar não será habilitado. Se o usuário optar por cancelar cadastro, voltará então para a tela de login.

Figura 12 – GUI Cadastro de Administrador.

VOSTOK

Cadastrar Administrador

Nome Completo _____ Data de Nascimento _____ Sexo _____ CPF _____

Endereço _____ Telefone _____ Email _____

Usuário _____ Senha _____

©2018 – Déborah Resende

Fonte: Elaborado pela autora.

Caso haja algum problema com a validação da informação, será informado avisos abaixo do campo para que o usuário a corrija, conforme pode-se observar na Figura 13. Isso acontece em todas as telas de cadastro e atualização de dados do sistema.

Figura 13 – GUI Cadastro de Administrador com dados inválidos nos campos.

VOSTOK

Cadastrar Administrador

Nome Completo _____
Informe o nome completo

Data de Nascimento _____
Informe a Data de Nascimento

Sexo _____
Selecione uma opção

CPF _____
Informe o CPF

Endereço _____
Informe o endereço completo

Telefone _____
Informe o telefone

Email _____
Informe o email

Usuário _____
Informe o usuário

Senha _____
Informe a senha

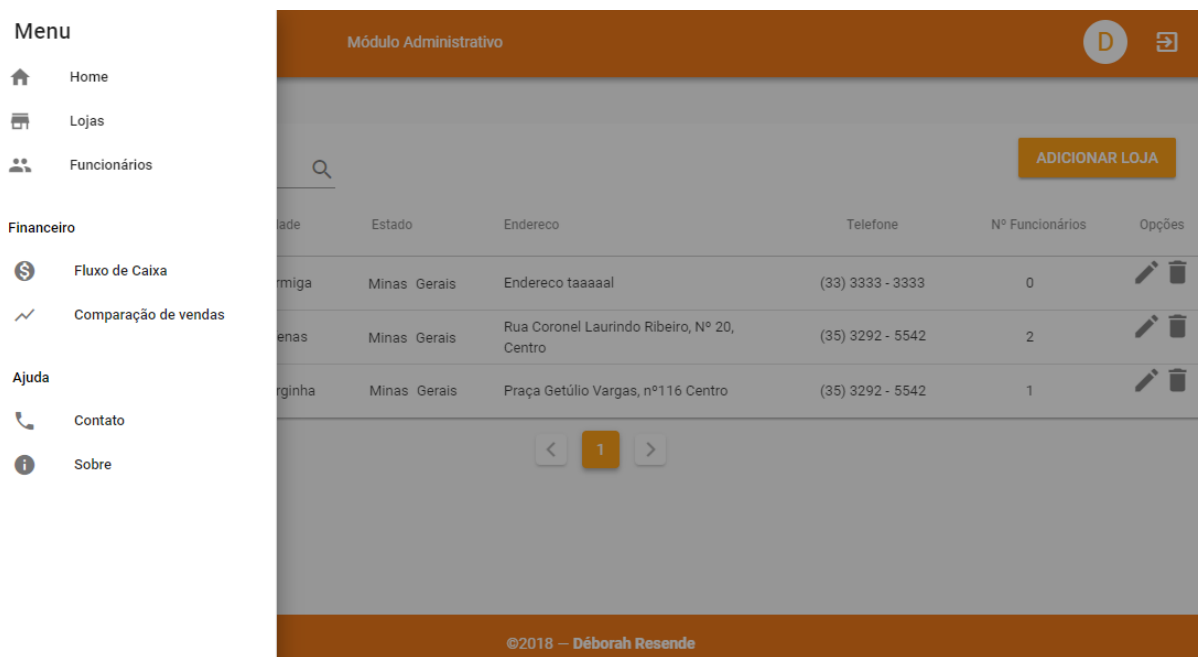
©2018 – Déborah Resende

Fonte: Elaborado pela autora.

4.2.3 Menu

Todas as funcionalidades que determinado usuário tem acesso estão localizadas no menu lateral esquerdo, ele também fica escondido na tela. O menu contém ícones intuitivos conforme a sua finalidade. O administrador pode acessar informações sobre lojas, funcionários, fluxo de caixa e comparação das vendas de suas lojas, *Home*, além das páginas de ajuda: Contato e Sobre, conforme pode-se observar na Figura 14.

Figura 14 – Menu Lateral para administradores.



Fonte: Elaborado pela autora.

Os funcionários podem acessar informações sobre produtos, estoque, monitoramento da temperatura dos *freezers*, *Home* e Ajuda, além do PDV (ponto de venda). As opções da categoria PDV mudam se o funcionário está ou não com o caixa aberto. Caso o funcionário esteja com o caixa fechado, o menu possui a estrutura ilustrada na Figura 15a. Se o caixa estiver aberto, então ele terá acesso a novas opções como realizar venda, operações de caixa e fechamento de caixa, conforme Figura 15b.

Figura 15 – Menu Lateral para funcionários.

(a) Menu Lateral com caixa fechado.



(b) Menu Lateral com caixa aberto.



Fonte: Elaborado pela autora.

4.2.4 Barra de Ferramentas

O componente Barra de Ferramentas, situado na parte superior da tela, é fundamental para a interface: apresenta um ativador para abrir e fechar o Menu Lateral, se o usuário for do tipo funcionário exibe notificação caso a temperatura de algum *freezer* esteja fora da temperatura ideal, exibe um *Avatar* com a letra inicial do usuário e o botão para sair do sistema. A Figura 16 ilustra a Barra de Ferramentas de um funcionário com notificação sobre a temperatura fora do ideal e um *Avatar* com a inicial S. A cor azul da notificação foi escolhida para contrastar com a cor laranja do fundo.

Figura 16 – Barra de Ferramentas.

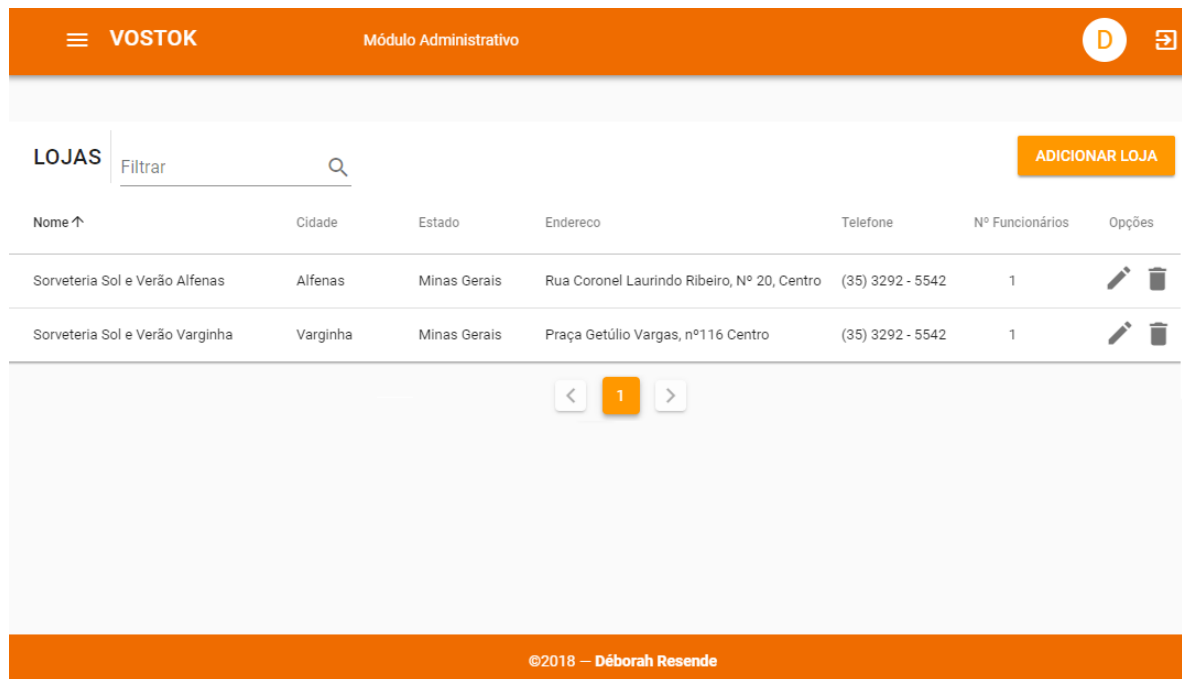


Fonte: Elaborado pela autora.

4.2.5 GUI Lojas

A GUI de gerenciamento de lojas, Figura 17, conta uma tabela mostrando todas as lojas daquele administrador e suas informações: nome, cidade, estado, endereço, telefone e funcionários.

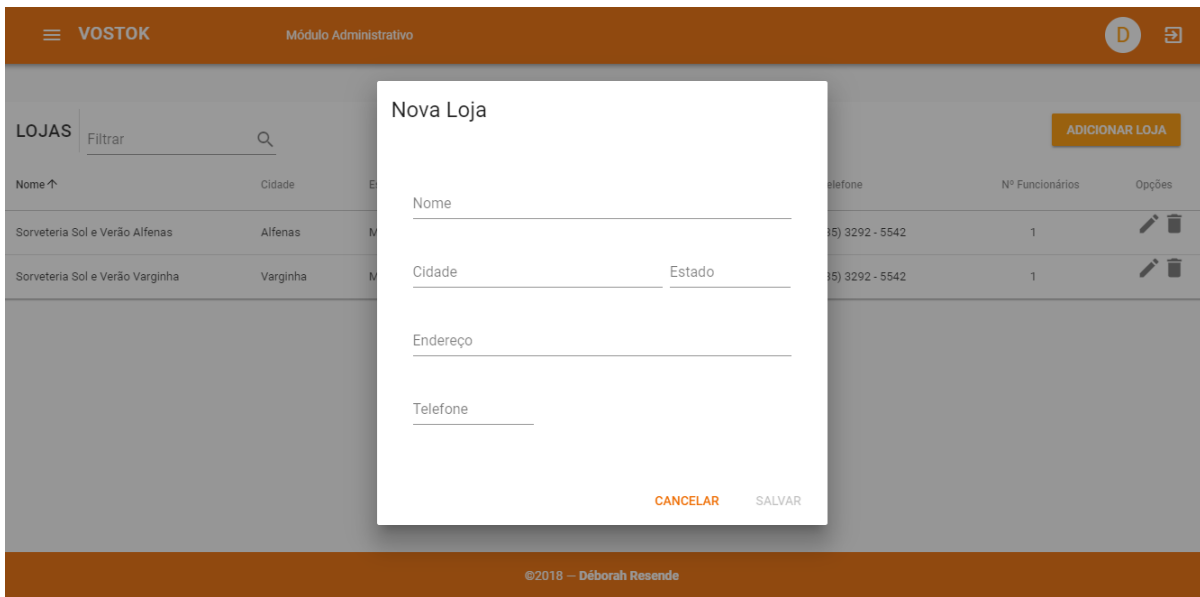
Figura 17 – GUI Lojas.



Fonte: Elaborado pela autora.

É possível adicionar uma nova loja clicando no botão **Adicionar Loja**, no canto superior direito da tabela, então é aberto um painel com os campos a serem preenchidos, Figura 18.

Figura 18 – Cadastrando Nova Loja.



Fonte: Elaborado pela autora.

Para se editar ou excluir determinada loja deve-se clicar no ícone referente à operação na coluna **Opções**, caso clique no ícone de editar é aberto um painel com os dados da loja selecionada, Figura 19 e caso clique no ícone de excluir é aberto um painel para confirmar exclusão, Figura 20.

Figura 19 – Editando Loja.

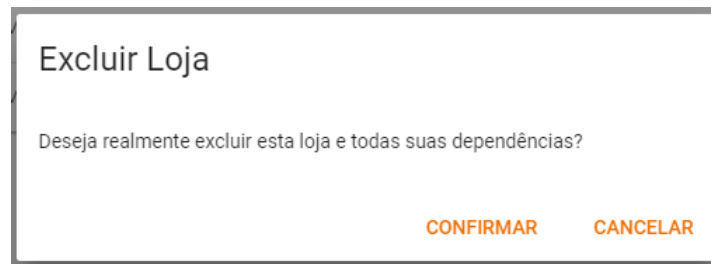
O formulário 'Editar Loja' apresenta os seguintes dados preenchidos:

- Nome: Sorveteria Sol e Verão Alfenas
- Cidade: Alfenas
- Estado: Minas Gerais
- Endereço: Rua Coronel Laurindo Ribeiro, Nº 20, Centro
- Telefone: (35) 3292-5542

Na base do formulário, há dois botões: 'CANCELAR' e 'SALVAR'.

Fonte: Elaborado pela autora.

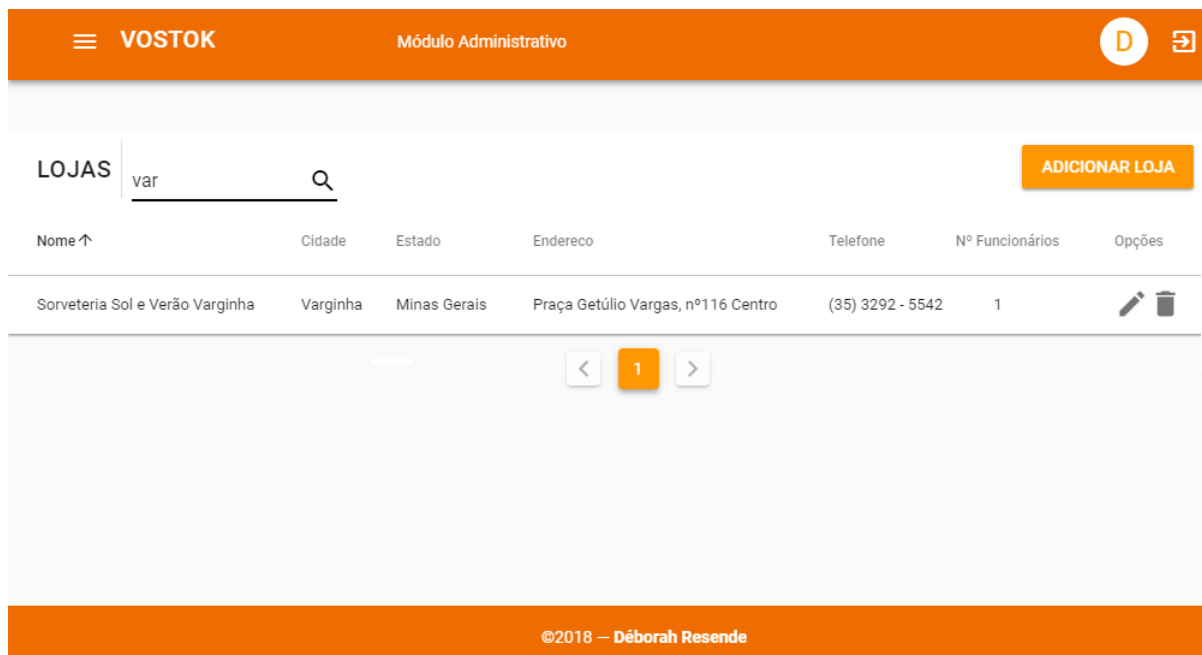
Figura 20 – Confirmação para excluir loja.



Fonte: Elaborado pela autora.

É possível utilizar filtros para selecionar as lojas que são mostradas, conforme pode-se observar na Figura 21.

Figura 21 – Filtrando lojas.



Fonte: Elaborado pela autora.

4.2.6 GUI Funcionários

A GUI de gerenciamento de funcionários, Figura 22, conta uma tabela mostrando os funcionários de todas as lojas daquele administrador e suas informações: nome, data de nascimento, data de admissão, sexo, CPF, *email*, telefone e em qual loja ele trabalha.

Figura 22 – GUI Funcionários.

Nome ↑	Data Nascimento	Data de Admissão	Sexo	CPF	Email	Telefone	Loja	Opções
Funcionario 1	01/01/1970	12/07/2018	Masculino	3333333333	funcionario@teste	3333333333	Sorveteria Sol e Verão Alfenas	
Funcionario 2	01/09/1970	10/07/2018	Feminino	2222222222	fun2@testando	2222222222	Sorveteria Sol e Verão Alfenas	
Funcionario 3	10/09/1990	12/07/2018	Não declarar	2222222222	fun3@teste	2222222222	Sorveteria Sol e Verão Alfenas	

Fonte: Elaborado pela autora.

Seguindo o mesmo modelo da Tela Lojas na seção 4.2.5, é possível adicionar novo funcionário, editar, excluir e filtrar por dados específicos.

4.2.7 GUI Produtos

A GUI de gerenciamento de produtos, Figura 23, conta uma tabela mostrando todas as informações sobre todos produtos cadastrado naquela loja: código do produto, nome, descrição, unidade, preço de compra e preço de vendas.

Seguindo o mesmo modelo da Tela Lojas na seção 4.2.5, é possível adicionar novo produto, editar, excluir e filtrar por dados específicos.

Figura 23 – GUI Produtos.

Código	Nome	Descrição	Unidade	Preço Compra	Preço Venda	Opções
1	Água c/ gás	Água mineral com gas	Unidade	R\$ 1.2	R\$ 3	 
2	coca lata	refrigerante coca	Lata	R\$ 2	R\$ 4.5	 
3	Sorvete	Sorvete cremoso diversos sabores	Kg	R\$ 12.9	R\$ 29.9	 
4	açaí	açaí cremoso	Kg	R\$ 9.9	R\$ 27.9	 

©2018 – Déborah Resende

Fonte: Elaborado pela autora.

4.2.8 GUI Estoque

A GUI de gerenciamento de estoque, Figura 24, conta uma tabela mostrando todos os produtos e seu estoque mínimo e o estoque atual naquela loja. Só é possível adicionar produtos ao estoque se o caixa estiver aberto, caso isso ocorra aparecerá um alerta. Outro alerta está relacionado a quantidades de produtos com estoque abaixo do mínimo.

Se o caixa estiver aberto, é habilitado um botão na coluna Adicionar Estoque da Figura 24. A Figura 25 demonstra como adicionar produtos ao estoque, o custo da compra é calculado de acordo com a quantidade de produtos e seu preço de custo. O custo da compra será descontado no caixa e essa movimentação estará no fluxo de caixa.

Figura 24 – GUI Estoque com caixa fechado.

Código	Nome ↑	Descrição	Unidade	Estoque Mínimo	Estoque Atual	Adicionar Estoque
4	açai	açai cremoso	Kg	15.00	0.00	
2	coca lata	refrigerante coca	Lata	10.00	12.00	
3	Sorvete	Sorvete cremoso diversos sabores	Kg	20.00	0.00	
1	Água c/ gás	Água mineral com gas	Unidade	20.00	12.00	

Fonte: Elaborado pela autora.

Figura 25 – Adicionando quantidade de determinado produto ao estoque.

Nome	Quantidade adicionada	Custo R\$
coca lata	10	20.00

CANCELAR ADICIONAR

Fonte: Elaborado pela autora.

4.2.9 GUI Abrir Caixa

A GUI Abrir Caixa, Figura 26, apresenta apenas um campo que deve ser preenchido: o valor disponível em caixa para iniciar as operações. Se o caixa estiver fechado não é possível realizar a venda, movimentação de estoque e nenhuma operação no caixa.

Figura 26 – GUI Abrir Caixa.

VOSTOK Módulo Funcionário

Abrir Caixa

Informe o valor disponível em caixa para iniciar suas operações

\$ _____

CONFIRMAR ABERTURA

©2018 – Déborah Resende

Fonte: Elaborado pela autora.

4.2.10 GUI Reforço

A GUI Reforço de caixa, como pode ser visto na Figura 27, apresenta o valor atual em caixa e apenas um campo que devem ser preenchido: o valor do reforço caixa.

Figura 27 – GUI Realizar Reforço de Caixa.

Reforço

Valor em caixa: R\$ 200
Informe o valor de reforço

\$ _____

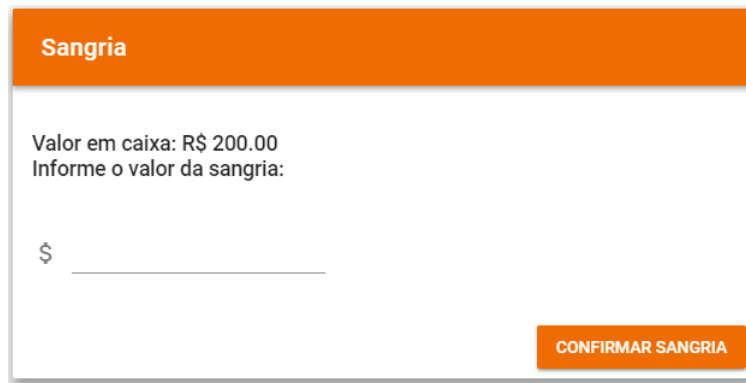
CONFIRMAR REFORÇO

Fonte: Elaborado pela autora.

4.2.11 GUI Sangria

A GUI Sangria, Figura 28, apresenta o valor atual em caixa e apenas um campo que devem ser preenchido: o valor da sangria.

Figura 28 – GUI Realizar Sangria.

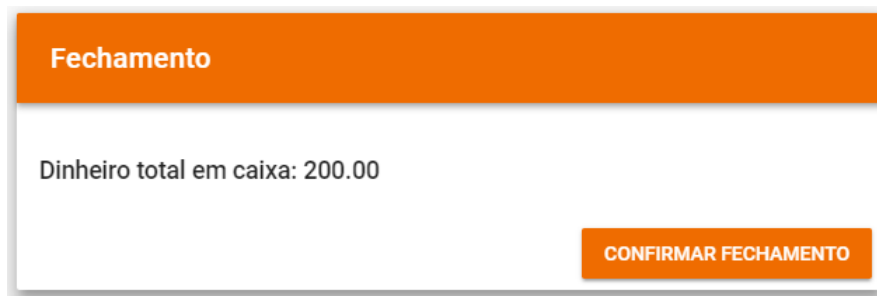


Fonte: Elaborado pela autora.

4.2.12 GUI Fechamento

A GUI Fechamento, Figura 29, apresenta o valor atual em caixa e apenas um botão para confirmar se deseja realmente fechar o caixa.

Figura 29 – GUI Fechamento de Caixa.

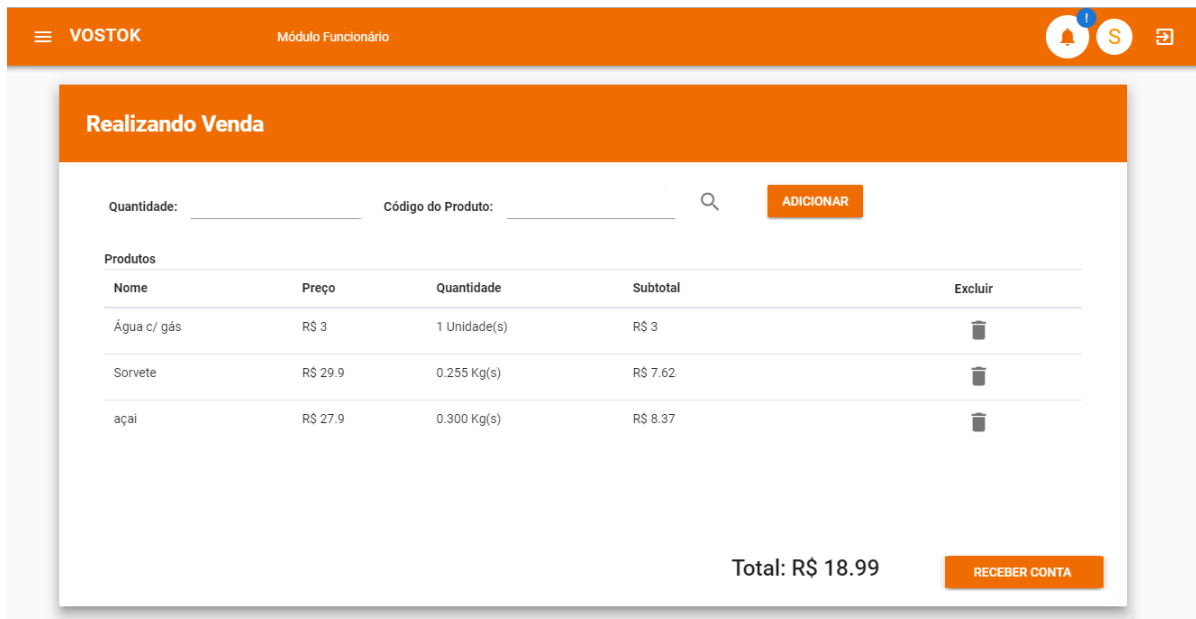


Fonte: Elaborado pela autora.

4.2.13 GUI Vendas

A GUI Vendas, Figura 30, contém dois campos que devem ser inseridos: quantidade e o código do produto, então ele será adicionado a lista de itens que vão ser comprados.

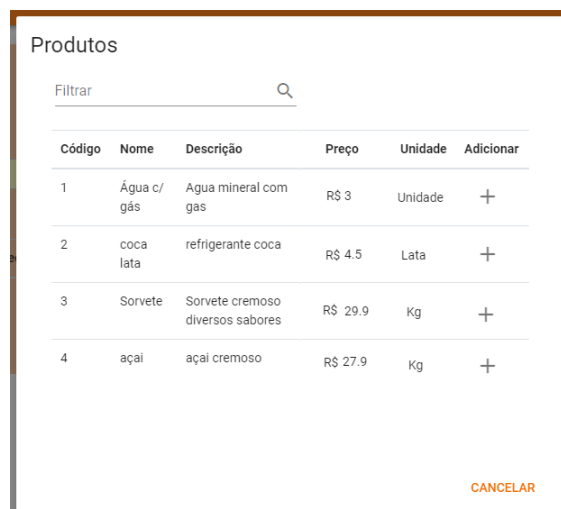
Figura 30 – GUI Vendas.



Fonte: Elaborado pela autora.

É possível pesquisar pelo produto clicando na lupa, caso o funcionário não saiba o código do produto, Figura 31. Para adicioná-lo deve-se clicar no símbolo "+".

Figura 31 – GUI Vendas pesquisando produtos.

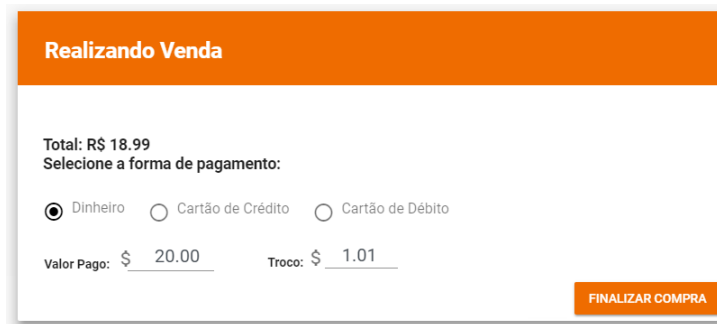


Fonte: Elaborado pela autora.

Caso o usuário entre com um código que não pertence a nenhum produto ele é notificado por um alerta na parte superior da tela. Se o usuário tentar adicionar uma quantidade de determinado produto superior à aquela encontrada no estoque, a quantidade daquele produto na lista de itens que vão ser comprados será no máximo a que está no estoque.

Após ter adicionado os produtos que serão comprados, Figura 30, então deve-se clicar em Receber Conta. Nessa parte é selecionado a forma de pagamento: dinheiro, cartão de crédito ou de débito. Se o pagamento for em dinheiro, aparecerá um campo para colocar o valor pago pelo cliente e então é calculado o troco. Ao clicar em Finalizar compra, a compra é registrada no banco de dados e o estoque atualizado.

Figura 32 – GUI Vendas selecionando tipo de pagamento.



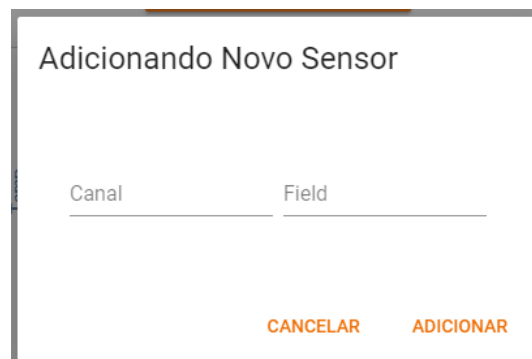
A interface 'Realizando Venda' apresenta o total de R\$ 18,99 e solicita a seleção da forma de pagamento. Há três opções de pagamento: Dinheiro (selecionada), Cartão de Crédito e Cartão de Débito. Abaixo, há campos para 'Valor Pago' (R\$ 20,00) e 'Troco' (R\$ 1,01). Um botão laranja 'FINALIZAR COMPRA' está no canto inferior direito.

Fonte: Elaborado pela autora.

4.2.14 GUI Monitoramento da Temperatura

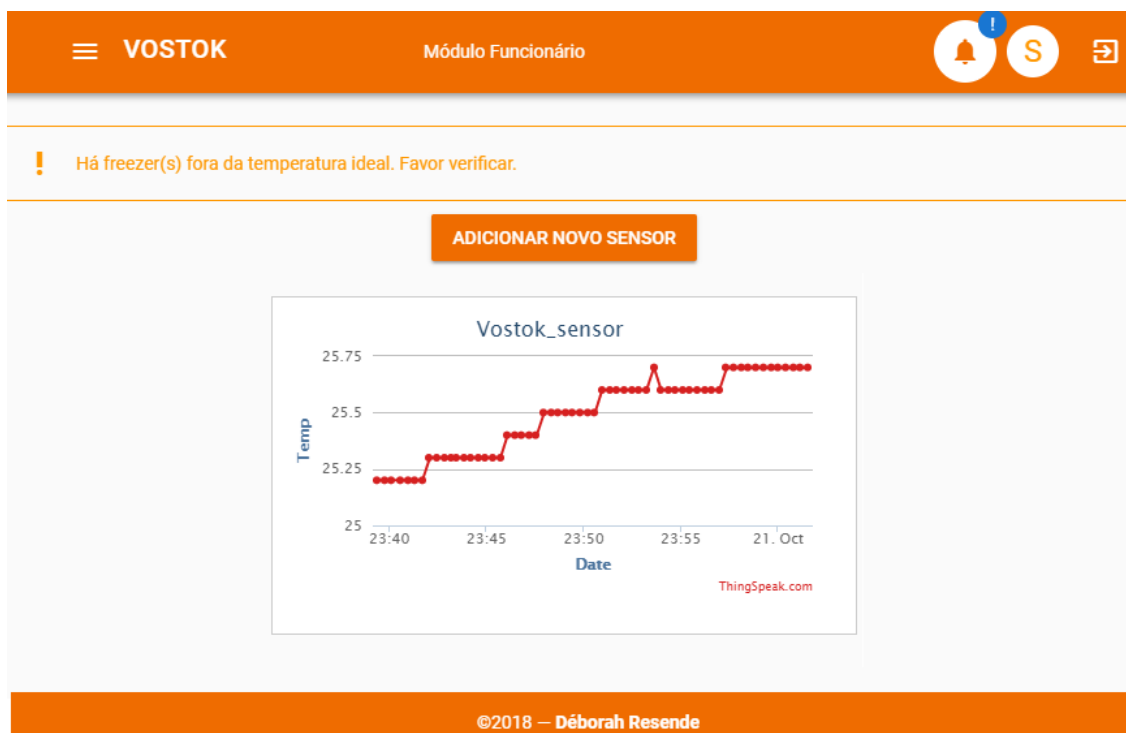
É possível adicionar um novo sensor de temperatura informando o nome do canal do ThingSpeak e nome do *Field*(campo), Figura 33. A tela monitoramento de temperatura, Figura 34, apresenta um alerta caso haja algum *freezer* que não se encontra na temperatura ideal. Apresenta os gráficos, cada um representando um sensor e suas 60 últimas leituras. Os gráficos são atualizados a cada 10 segundos.

Figura 33 – Adicionando novo sensor de temperatura.



A interface 'Adicionando Novo Sensor' possui dois campos de entrada: 'Canal' e 'Field'. Abaixo dos campos, há dois botões laranja: 'CANCELAR' e 'ADICIONAR'.

Fonte: Elaborado pela autora.

Figura 34 – GUI monitoramento da temperatura dos *freezers*.

Fonte: Elaborado pela autora.

4.2.15 GUI Fluxo de Caixa

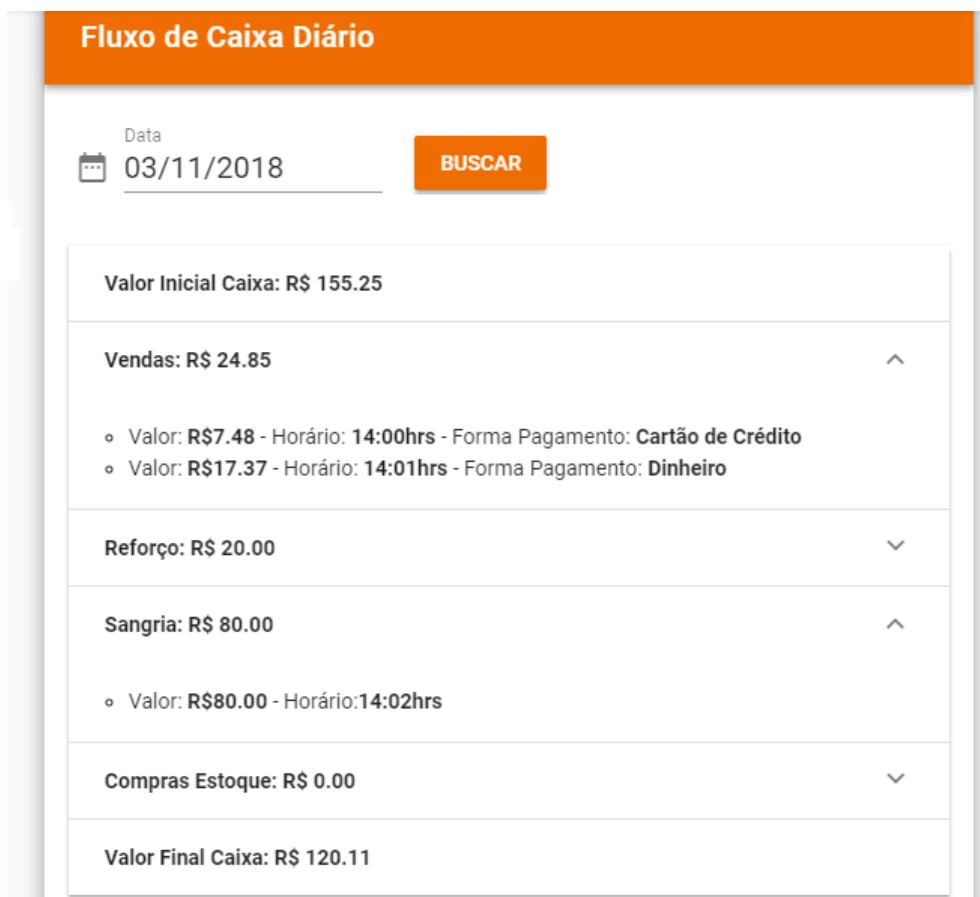
Na GUI Fluxo de Caixa, Figura 35, é possível pesquisar o fluxo de caixa diário por data. Se a data pesquisada não corresponder a nenhuma movimentação de caixa então aparece uma alerta informando ao usuário. Se a busca corresponder então é mostrado todo o fluxo de caixa: valor que o caixa iniciou suas operações, vendas realizadas, reforço, sangria, compras e o valor final do caixa. Ao se clicar na seta ao lado do campo é possível obter mais detalhes sobre ele, Figura 36.

Figura 35 – GUI Fluxo de Caixa Diário.

The screenshot shows the 'Fluxo de Caixa Diário' search interface. It features an orange header with the title 'Fluxo de Caixa Diário'. Below the header, there is a search form with a date input field labeled 'Data' and a placeholder 'DD/MM/AAAA'. A calendar icon is visible to the left of the input field. To the right of the input field, there is an orange button labeled 'BUSCAR'.

Fonte: Elaborado pela autora.

Figura 36 – GUI Fluxo de Caixa Diário da data 03/11/2018.



Fonte: Elaborado pela autora.

4.2.16 GUI Gráfico de Comparação de Vendas das Lojas

A GUI Gráfico de Comparação de Vendas das Lojas, Figura 37, permite gerar gráficos comparando a vendas das lojas de determinado administrador por um intervalo de tempo. Para isso deve-se informar o intervalo desejado e então clicar em gerar.

Figura 37 – GUI Gráfico de Comparação de Vendas das Lojas.

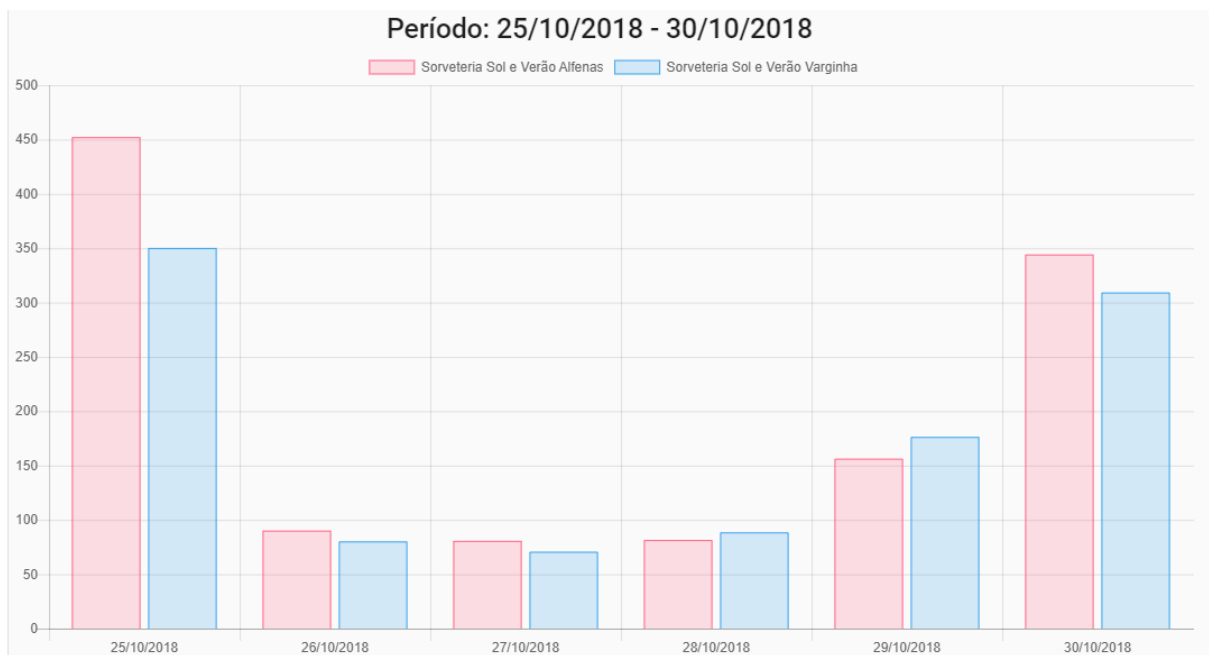
Gráfico de comparação de vendas das lojas por período

Data Início: DD/MM/AAAA Data Fim: DD/MM/AAAA **GERAR**

Fonte: Elaborado pela autora.

A Figura 38 mostra a GUI Gráfico de comparação das duas lojas que o administrador no intervalo de seis dias: 25/10 a 30/10. A barra mais a esquerda de cada dia refere-se aos dados da loja situada em Alfenas e a direita refere-se aos dados da loja de Varginha.

Figura 38 – GUI Gráfico de Comparação de Vendas das Lojas gerando gráfico do intervalo: 25/10 - 30/10.



Fonte: Elaborado pela autora.

4.3 Modelo de Dados

Inicialmente foi conectado a aplicação ao banco de dados, usando o Mongoose para definir *schemas* e modelos. O Mongoose oferece uma maneira prática de definir e manter estruturas de dados, facilitando o fluxo de desenvolvimento de aplicações; além de incluir a possibilidade de validar as definições de dados.

O MongoDB conversa apenas com o Mongoose, e o Mongoose repassa a comunicação para a dupla NodeJs/Express. O Vue não conversa diretamente com o MongoDB ou Mongoose, em vez disso obterá seus dados da aplicação em Express.

Não é necessário criar o banco de dados separadamente, pois o MongoDB o fará na primeira vez em que for usado.

A conexão Mongoose é algo simples, basta declarar a URI para o banco de dados e passá-la ao método *connect*, conforme Figura 39. A URI de banco de dados é uma linha de texto que contém o protocolo do MongoDB, endereço do servidor, porta e nome do banco de dados.

Figura 39 – Conexão Mongoose e método *connect*.

```
const mongoDB = 'mongodb://127.0.0.1/bdSysTCC';
mongoose.connect(mongoDB);
```

Fonte: Elaborado pela autora.

O monitoramento da conexão foi realizado através de métodos que tomam como base os estados de conexão, para se ter ideia do que está acontecendo. Eventos do processo NodeJs também foram monitorados para que se possa forçar o encerramento de conexões do Mongoose quando a aplicação for encerrada.

Como o MongoDB é SGBD NoSQL orientado a documentos e trata-se de uma tecnologia relativamente recente, ele não apresenta um esquema predefinido a se seguir, fazendo com que a modelagem deva ser feita levando em consideração a abstração do desenvolvedor e buscando ter um bom desempenho das consultas. Assim, optou-se por realizar a modelagem do sistema utilizando três *schemas*: Usuário, Loja e Movimentação.

4.3.1 Usuário

A Figura 40 demonstra a estrutura do *schema* usuário, ele possui onze campos e todos são exigidos para se ter sucesso ao adicionar ou atualizar. Os atributos são: nome, usuário, senha, data de nascimento, sexo, CPF, telefone, endereço, tipo (administrador ou funcionário), *email* e data de admissão. A Figura 41, representa um exemplo de documento usuário do tipo administrador.

Figura 40 – *Schema* usuário.

```
var UserSchema = new Schema({
  nome: { type: String, required: true },
  username: { type: String, unique: true, required: true },
  password: { type: String, required: true },
  dataNasc: { type: Date, required: true },
  sexo: { type: String, required: true },
  cpf: { type: String, required: true },
  tel: { type: String, required: true },
  endereco: { type: String, required: true },
  tipo: { type: String, required: true },
  email: { type: String, required: true },
  dataAdmissao: { type: Date, required: true }
});
```

Fonte: Elaborado pela autora.

Figura 41 – Exemplo de um documento usuário.

```
{
  "_id": "5bbe674064ff270a209df0d4",
  "nome": "deborah aparecida resende",
  "username": "deborah",
  "password": "$2a$10$g67wZg3nH1/Iw3lp19jloeIx60b3RIehgJM/1wbiGmBGYOk.CGyW2",
  "dataNasc": "1994-09-24T00:00:00.000Z",
  "sexo": "Feminino",
  "cpf": "11111111111",
  "tel": "5537980517",
  "endereco": "Rua Coronel Bragança 1271",
  "tipo": "administrador",
  "email": "deborah@gmail.com",
  "dataAdmissao": "2018-10-10T20:55:28.644Z",
  "__v": 0
},
```

Fonte: Elaborado pela autora.

4.3.2 Loja

O *schema* loja guarda informações acerca das lojas e suas dependências (produtos e sensores) cadastradas na aplicação. Possui nove campos: nome, cidade, estado, endereço, telefone, administrador, funcionários, produtos e sensores. Sendo que os campos produtos e sensores podem ser considerados estruturas aninhadas. O atributo **sensor** é formado apenas por dois campos: canal e *field*(campo), por isso não foi necessário criar um *schema* separado para ele. Mas como o atributo **produto** possui mais campos, colocá-lo em um *schema* diferente torna mais legível a estrutura.

Figura 42 – *Schema* loja.

```
const LojaSchema = new Schema({
  nome: {type: String, required: true },
  cidade: {type: String, required: true },
  estado: {type: String, required: true },
  endereco: {type: String, required: true },
  tel: {type: String, required: true },
  administrador: {type: String, required: true },
  funcionarios: [ String ],
  produtos: [ Produto ],
  sensores: [{
    canal: String,
    field: String
  }]
});
```

Fonte: Elaborado pela autora.

A Figura 43 demonstra a estrutura de produtos, contendo oito campos: código, nome, descrição, preço de venda, preço de compra, unidade de medida, estoque mínimo e o estoque atual. Todos os campos são exigidos para se ter sucesso ao adicionar ou atualizar.

Figura 43 – *Schema* produto.

```
const ProdutoSchema = new Schema({
  cod: {type: String, required: true },
  nome: {type: String, required: true },
  descricao: {type: String, required: true },
  precoCompra: {type: Number, required: true },
  precoVenda: {type: Number, required: true },
  unidade: {type: Number, required: true },
  estoqueMin: {type: Number, required: true },
  estoqueAtual: {type: Number, required: true }
});
```

Fonte: Elaborado pela autora.

A Figura 44 representa um exemplo de documento loja que possui dois funcionários e apenas um produto. Optou-se por usar como referência nos atributos funcionários e administrador o nome de usuário, por ele ser único.

Figura 44 – Exemplo de um documento loja.

```

{
  "funcionarios": [
    "simone",
    "funcionario3"
  ],
  "sensor": [],
  "_id": "5bc49b91366bdf2524708454",
  "nome": "Sorveteria Sol e Verão Alfenas",
  "cidade": "Alfenas ",
  "estado": "Minas Gerais",
  "endereco": "Rua Coronel Laurindo Ribeiro, Nº 20, Centro",
  "tel": "3532925542",
  "administrador": "deborah",
  "sensor": [],
  "produtos": [
    {
      "_id": "5bc49c41366bdf2524708458",
      "cod": "1",
      "nome": "Sorvete",
      "descricao": "sorvete cremoso diversos sabores",
      "precoCompra": 9.9,
      "precoVenda": 27.8,
      "unidade": "Kg",
      "estoqueMin": 15,
      "estoqueAtual": 4.655
    }
  ],
  "__v": 0
}

```

Fonte: Elaborado pela autora.

4.3.3 Movimentação

A Figura 45 demonstra a estrutura do *schema* movimentação, contendo sete campos sendo três deles com estrutura aninhada. O atributo **abertura** e **fechamento** são as datas e horários que o funcionário abriu e fechou o caixa. O atributo **valorAbertura** armazena o valor que o caixa iniciou suas operações e **user** armazena o nome do usuário do funcionário que realizou a abertura. Quando se realiza uma **venda** é necessário armazenar o horário, a relação de produtos, o total da venda e sua forma de pagamento. Pode-se realizar três tipos de operações: sangria, reforço ou estoque, sendo o tipo estoque a entrada de produtos e conseqüentemente a retirada do valor do caixa.

Figura 45 – Modelo documento movimentação.

```
const MovimentacaoVendaSchema = new Schema({
  abertura: {type: Date, required: true},
  valorAbertura: {type: Number, required: true},
  user: {type: String, required: true},
  vendas: [{
    horario: {type: String, required: true},
    relProdutos: [ProdQtd],
    total: {type: Number, required: true},
    formaPagamento: {type: String, required: true}
  ]},
  operacoes: [{
    tipo: {type: String, required: true},
    valor: {type: Number, required: true},
    horario: {type: Date, required: true}
  ]},
  fechamento: {type: Date, required: true},
  valorFinal: {type: Number, required: true}
});
```

Fonte: Elaborado pela autora.

Figura 46 – Modelo documento Produto e Quantidade.

```
const ProdQtdSchema = new Schema({
  cod: {type: Number, required: true},
  qtd: {type: Number, required: true},
  preco: {type: Number, required: true},
  subTotal: {type: Number, required: true}
});
```

Fonte: Elaborado pela autora.

A Figura 47 apresenta um exemplo de documento movimentação após realizar a abertura de caixa e Figura 48 apresenta um exemplo de documento movimentação após realizar vendas, operações e fechamento de caixa.

Figura 47 – Exemplo de um documento movimentação após abertura de caixa.

```
{
  "_id": "5bd85f24111d0f0184ad4586",
  "abertura": "2018-10-30T13:39:48.127Z",
  "valorAbertura": 155,
  "user": "suena123",
  "fechamento": "",
  "valorFinal": null,
  "vendas": [],
  "operacoes": [],
  "__v": 0
}
```

Fonte: Elaborado pela autora.

Figura 48 – Exemplo de um documento movimentação após vendas, operações e fechamento de caixa.

```

{
  "_id": "5bd85f24111d0f0184ad4586",
  "abertura": "2018-10-30T13:39:48.127Z",
  "valorAbertura": 155,
  "user": "suenal23",
  "fechamento": "2018-10-30T14:20:59.397Z",
  "valorFinal": 182.5,
  "vendas": [
    {
      "relProdutos": [
        {
          "_id": "5bd86830111d0f0184ad458b",
          "cod": 1,
          "nome": "Água c/ gás",
          "qtd": 1,
          "preco": 3,
          "subTotal": 3
        },
        {
          "_id": "5bd86830111d0f0184ad458a",
          "cod": 2,
          "nome": "coca lata",
          "qtd": 1,
          "preco": 4.5,
          "subTotal": 4.5
        }
      ]
    },
    {
      "_id": "5bd86830111d0f0184ad4589",
      "horario": "2018-10-30T14:18:24.236Z",
      "total": 7.5,
      "formaPagamento": "Cartão de Crédito"
    }
  ],
  "operacoes": [
    {
      "_id": "5bd86717111d0f0184ad4587",
      "tipo": "sangria",
      "valor": 30,
      "horario": "2018-10-30T14:13:43.770Z"
    },
    {
      "_id": "5bd86744111d0f0184ad4588",
      "tipo": "reforco",
      "valor": 50,
      "horario": "2018-10-30T14:14:28.297Z"
    }
  ],
  "__v": 0
}

```

Fonte: Elaborado pela autora.

4.4 Monitoramento da Temperatura

Inicialmente foi criado uma conta gratuita no ThingSpeak, com as configurações demonstradas na Figura 49a. O ID do canal é 606462 e serão enviados dados para dois *fields*(campos): Tem(temperatura) e Humidity(humidade).

Para enviar dados no canal é necessário ter uma chave de escrita, gerada pelo próprio site, Figura 49b .

Figura 49 – Configurando ThingSpeak.

(a) Configurações canal.

Channel Settings

Percentage complete 30%

ID do canal 606462

Nome

Descrição

Campo 1

Campo 2

(b) Chave de escrita do canal.

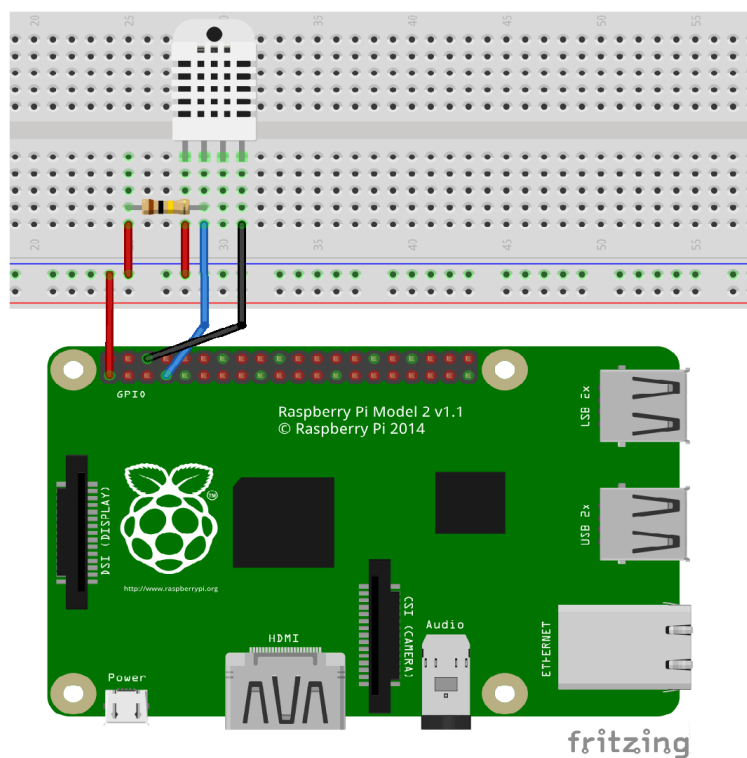
Chave de Escrita

Chave

Fonte: Elaborado pela autora.

A Figura 50 refere-se ao esquemático desse circuito responsável pelo monitoramento da temperatura dos *freezers*.

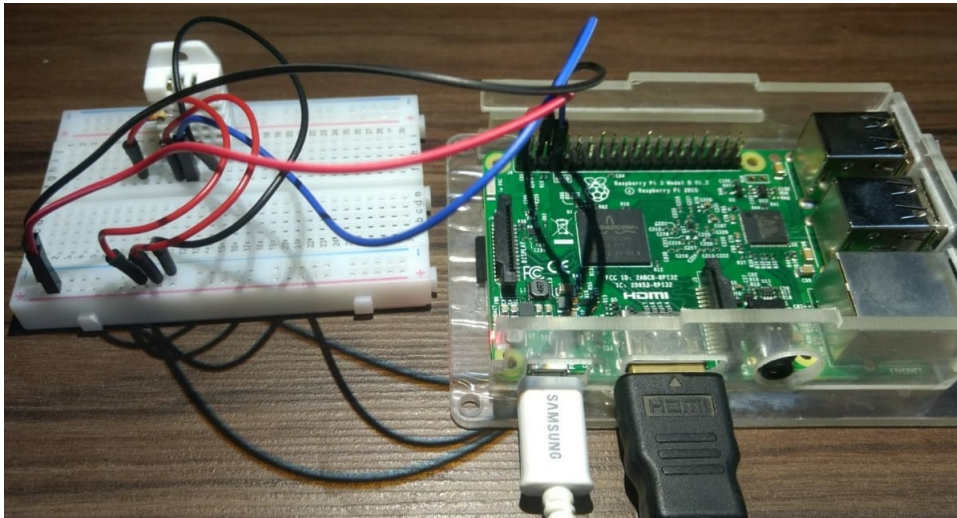
Figura 50 – Esquemático da montagem do circuito utilizando a ferramenta Fritzing.



Fonte: Elaborada pela autora.

Como observa-se no esquemático acima, o DHT22 foi alimentado com 3.3V. Isso é importante pois ao alimentar o DHT22 com os 5V da GPIO, se corre o risco de danificar a placa pois as portas do Raspberry trabalham com nível de sinal de 3.3V. A alimentação do sensor, pino 1 (o mais a esquerda no esquemático), foi ligada ao pino 1 do Raspberry pi, mas poderia ser ligada a qualquer pino de 3.3V, como por exemplo o 17. O pino 2 do sensor é o pino de dados e pode ser utilizado qualquer GPIO não utilizado da placa, então foi escolhido o pino 4. O pino 4 do sensor (o mais a direita) é o terra e poderia ter sido ligado a qualquer outro da mesma categoria mas foi escolhido o pino 6 do Raspberry. Informações sobre GPIO do Raspberry encontra-se na Figura 6. A Figura 51 refere-se a montagem do circuito do esquemático.

Figura 51 – Montagem do circuito.



Fonte: Elaborada pela autora.

Para que o programa em Python funcione corretamente é necessário uma biblioteca adicional do **Adafruit**, esta biblioteca serve para o DHT22 e também para o Sensor DHT1, uma versão menos precisa do sensor. Também foi feita a atualização o Raspbian e baixado o python-dev e setup.py para o correto funcionamento da biblioteca.

O código em Python abaixo é responsável por ler as informações do sensor e então envia-las ao Thingspeak em um certo intervalo de tempo.

```
1 # Carrega as bibliotecas
2 import sys
3 from time import sleep
4 import Adafruit_DHT
5 import RPi.GPIO as GPIO
6 import urllib2
7
8 baseURL = 'https://api.thingspeak.com/update?'
9 chaveVostok = 'BYYM1QXQ7Z9RJSJ5'
10
11 sensor_args = { '11': Adafruit_DHT.DHT11,
12                 '22': Adafruit_DHT.DHT22,
13                 '2302': Adafruit_DHT.AM2302 }
14 if len(sys.argv) == 3 and sys.argv[1] in sensor_args:
15     sensor = sensor_args[sys.argv[1]]
16     pin = sys.argv[2]
17 else:
18     print('Modelo: python SensorTempVostok.py [11|22|2302] <GPIO pin number>
19           ')
19     print('Uso correto: python SensorTempVostok.py 22 4')
20     sys.exit(1)
21
22 humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
23
24 while(1):
25     humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
26     if humidity is not None and temperature is not None:
27         print('Temperatura = {0:0.1f}*C. Humidade do Ar = {1:0.1f}%'.format(
28               temperature, humidity))
28         f = urllib2.urlopen(baseURL + 'api_key=' + chaveVostok + '&field1=' + str(
29               temperature) + '&field2=' + str(humidity))
29         print('Enviando ThingSpeak...')
30         f.read()
31         f.close()
32         sleep(11)
33     else:
34         print('Nao foi possivel ler. Tente novamente!')
35 sys.exit(1)
```

Como se pode observar acima, foram importadas cinco bibliotecas: `sys`, `time sleep`, `Adafruit_DHT`, `GPIO` e `urllib2`. O módulo `sys` fornece acesso a algumas variáveis usadas ou mantidas pelo interpretador e a funções que interagem fortemente com o interpretador. O `sleep` suspende a execução pelo número de segundos especificado. `Adafruit_DHT` é usado para ler a série DHT de sensores de umidade e temperatura em um Raspberry Pi.

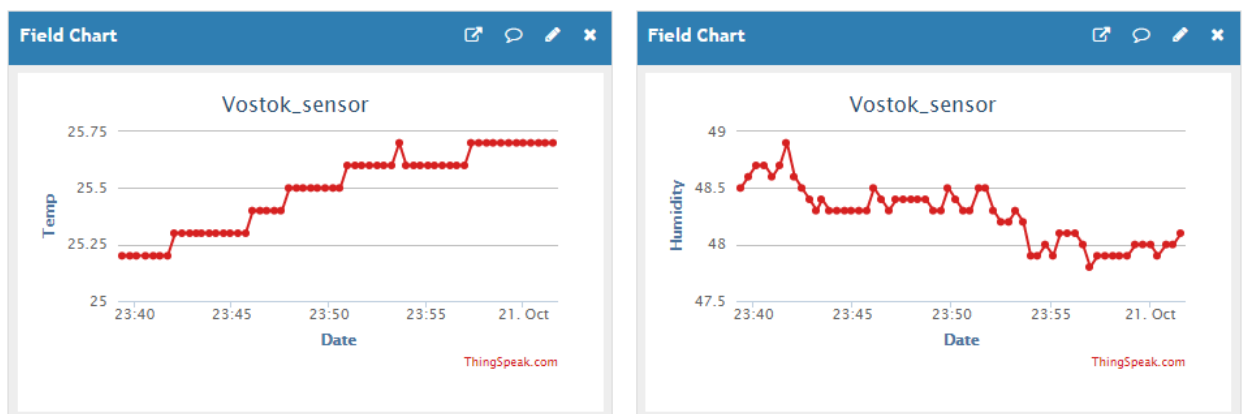
GPIO é usado para interagir com os pinos programáveis de entrada e saída da placa.

Depois de importar as bibliotecas é verificado se a quantidade dos parâmetros ao executar o programa estão corretos e então os valores referentes ao modelo do sensor e a porta do Raspberry são carregados. Na linha 23 do código é feita a leitura do sensor. O programa ficará sempre em execução e logo que os valores do sensor não forem nulos é realizado um *print* na tela com os valores e também é feito um *update* na conta do Thingspeak, levando em consideração a base da URL e a chave. Existe um *delay* de 10 segundos entre uma leitura e outra. A sintaxe de execução é dada por:

```
' python SensorTempVostok.py 22 4 '
```

A Figura 52 demonstra os *Fields Charts* do ThingSpeak do canal 606462 após receber alguns dados. O sensor no momento desta medição estava em uma sala e não dentro de um *freezer*, logo as temperaturas são mais altas do que as que serão trabalhadas. Mas o intuito da Figura 52 foi demonstrar seu correto funcionamento do sensor e envio de dados.

Figura 52 – *Field Charts* do Thingspeak do canal 606462 após receber alguns dados.



Fonte: Elaborada pela autora.

Para recuperar os dados do ThingSpeak é necessário fazer requisições HTTP na aplicação, identificando o canal e nome do *field*.

Por se tratar de um protótipo, este dispositivo não deve ser colocado dentro do *freezer*, pelo fato do Raspberry não suportar tal temperatura.

5 CONSIDERAÇÕES FINAIS

O sistema Vostok, produto deste trabalho de conclusão de curso atendeu aos requisitos que foram levantados durante a modelagem do sistema e atendeu aos objetivos específicos propostos.

Ele é capaz de atender as principais necessidades de empresas no ramo de sorvete que possuem filiais. O administrador é capaz de manter suas lojas e funcionários em um só sistema, ter acesso aos fluxos de caixa de suas lojas além de poder realizar uma comparação sobre as vendas das lojas. O funcionário é responsável por manter produtos, realizar compra de produtos, realizar operações de caixa, efetuar vendas e monitorar a temperatura dos *freezers*.

Foi realizada a validação por um administrador e um funcionário da empresa, que afirmaram que o sistema melhorará a gestão administrativa consequentemente uma redução considerável de tempo.

O sistema deve ser melhorado em trabalhos futuros para implementar mais funcionalidades como: mais relatórios financeiros e gráficos comparativos, curva ABC de estoque, contas a receber, contas a pagar e fornecedores.

Ao analisar toda a capacidade de processamento do Raspberry e a quantidade utilizada para realizar a medição da temperatura, observou-se que há um grande desperdício da capacidade do *hardware*, no maior ápice observado chegou a 11%. Logo, em trabalhos futuros deve ser desenvolvido um novo dispositivo, visando um melhor aproveitamento dos componentes, melhor custo benefício e que este possa ser inserido dentro dos *freezers*.

Referências

ALMEIDA, F. *Mean: Full stack JavaScript para aplicações web com MongoDB, Express, Angular e Node*. Editora Casa do Código, 2015. Acessado: 14 jul. 2018. Disponível em: <<https://goo.gl/VkNx1b>>. Citado na página 37.

AREND, R. C. *Implementação de aplicação web para controle de gastos da central telefônica da UTFPR*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2013. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/UgKYgz>>. Citado na página 31.

AZEVEDO, K. *21 franquias de sorvete para investir*. 2016. Acessado: 22 set. 2018. Disponível em: <<https://goo.gl/nJtrm8>>. Citado na página 25.

BAX, M. P. Introdução às linguagens de marcas. *Ciência da Informação*, SciELO Brasil, v. 30, n. 1, p. 32–38, 2001. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/sLx3YC>>. Citado na página 33.

BROWN, E. *Web Development with Node and Express: Leveraging the JavaScript Stack*. O'Reilly Media, 2014. Acessado: 12 ago. 2018. ISBN 9781491902301. Disponível em: <<https://books.google.com.br/books?id=1cHvAwAAQBAJ>>. Citado na página 37.

CANTÚ, D. *Sistema web para monitoramento de sensores de temperatura e umidade*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2013. Acessado: 02 out. 2018. Disponível em: <<https://goo.gl/Xbu1BG>>. Citado na página 43.

CARDOSO, B. E. E. *DESENVOLVIMENTO DE WEBSITES RESPONSIVOS PARA REDE DE HÓTEIS*. 2018. Acessado: 13 out. 2018. Disponível em: <<https://goo.gl/X8WiD4>>. Citado na página 53.

CHANDRA, D. G. Base analysis of nosql database. *Future Generation Computer Systems*, Elsevier, v. 52, p. 13–21, 2015. Acessado: 02 jul. 2018. Disponível em: <<https://goo.gl/WX4qFR>>. Citado na página 40.

CHART.JS. Chart.js. 2015. Acessado: 17 out. 2018. Disponível em: <<https://goo.gl/YyTRkx>>. Citado na página 36.

CIA, A. *Sensor de temperatura e umidade DHT22 (AM2302)*. 2015. Acessado: 02 out. 2018. Disponível em: <<https://goo.gl/LxB6L8>>. Citado 2 vezes nas páginas 43 e 44.

CISCO. *Cisco visualizations*. 2011. Acessado: 02 out. 2018. Disponível em: <<https://goo.gl/UpLWbt>>. Citado na página 42.

CONSUMER. *Cresça o seu negócio com o Consumer*. 2018. Acessado: 06 dez. 2018. Disponível em: <<https://www.programaconsumer.com.br/>>. Citado na página 47.

CPT, S. *CPT Sorveteria - Software para Gerenciamento de Sorveteria*. 2018. Acessado: 06 dez. 2018. Disponível em: <<https://goo.gl/KiRpLC/>>. Citado na página 47.

- DUARTE, D. *CONFIRA AS TENDÊNCIAS PARA O MERCADO DE SORVETES*. 2017. Acessado: 22 set. 2018. Disponível em: <<https://goo.gl/3vEcxk>>. Citado na página 25.
- DUARTE, N. F. B. *Frameworks e Bibliotecas Javascript*. Tese (Doutorado), 2015. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/mKxUVm>>. Citado 2 vezes nas páginas 31 e 42.
- DURSO, F. M. *Fatores que aferam a vida de prateleira de sorvetes*. 2012. Acessado: 07 jul. 2018. Disponível em: <<https://goo.gl/hz6eX9>>. Citado na página 29.
- ERBANO, B. L. et al. Fluxo de caixa. *Maiêutica-Estudos Contemporâneos em Gestão Organizacional*, v. 1, n. 1, 2014. Acessado: 13 out. 2018. Disponível em: <<https://goo.gl/BBgjpf>>. Citado na página 30.
- ESTADÃO. *Gestão de franquias: dicas fundamentais para o sucesso*. 2017. Acessado: 22 set. 2018. Disponível em: <<https://goo.gl/nAV6MU>>. Citado na página 25.
- FALCÃO, D. F. D. Uma arquitetura de cloud computing para análise de big data provenientes da internet of things. 2014. Acessado: 01 jul. 2018. Disponível em: <<https://goo.gl/PwLDA6>>. Citado na página 40.
- FERREIRA, R. *REST: Princípios e boas práticas*. 2017. Acessado: 16 out. 2018. Disponível em: <<https://goo.gl/6SrQ4p>>. Citado na página 32.
- FLANAGAN, D. *JavaScript: O guia definitivo*. Bookman Editora, 2007. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/EWVG7t>>. Citado na página 33.
- GALDINO, F. *VueJs Tutorial*. 2017. Acessado: 12 set. 2018. Disponível em: <<https://goo.gl/Y1Ppqz>>. Citado na página 35.
- GARRETT, J. J. Ajax: A new approach to web applications | adaptive path. 2005. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/Dnxy7y>>. Citado na página 31.
- GRONER, L. *Learning JavaScript Data Structures and Algorithms: Write complex and powerful JavaScript code using the latest ECMAScript, 3rd Edition*. Packt Publishing, 2018. Acessado: 12 set. 2018. ISBN 9781788624947. Disponível em: <<https://books.google.com.br/books?id=mVBZDwAAQBAJ>>. Citado na página 39.
- HAHN, E. *Express in Action: Writing, Building, and Testing Node.js Applications*. Manning Publications, 2016. Acessado: 12 ago. 2018. ISBN 9781617292422. Disponível em: <<https://books.google.com.br/books?id=hTcfrgEACAAJ>>. Citado na página 37.
- HANNAH, J. *The Ultimate Guide to JavaScript Frameworks*. 2018. Acessado: 12 set. 2018. Disponível em: <<https://goo.gl/vCVUhf>>. Citado 2 vezes nas páginas 33 e 34.
- HEIN, W. Raspberry pi aplicado a projetos do mundo real. *Linux Magazine*, 2013. Acessado: 05 out. 2018. Disponível em: <<https://goo.gl/3pJTH5>>. Citado na página 52.
- HOLMES, S. *MEAN Definitivo: com Mongo, Express, Angular e Node*. NOVATEC, 2016. ISBN 9788575224915. Disponível em: <<https://books.google.com.br/books?id=3G3iCwAAQBAJ>>. Citado na página 42.

- IHRIG, C. J. *Pro Node.js para Desenvolvedores*. [s.n.], 2014. Acessado: 12 ago. 2018. ISBN 978-85-399-0552-2. Disponível em: <<https://goo.gl/Xkazpw>>. Citado na página 37.
- INC, P. T. *Postman | Supercharge your API workflow*. Tese (Doutorado), 2017. Acessado: 23 jul. 2018. Disponível em: <<https://www.getpostman.com/>>. Citado na página 46.
- JOBSTRAIBIZER, F. *Criação de sites com o CSS*. Universo dos Livros Editora, 2009. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/E6ySa4>>. Citado na página 33.
- LARMAN, C. *Utilizando UML e padrões*. Bookman Editora, 2002. Acessado: 02 out. 2018. Disponível em: <<https://goo.gl/Atv1zL>>. Citado 2 vezes nas páginas 52 e 55.
- LAVISKA, C. *Hashing Passwords with Node.js and Bcrypt*. 2007. Acessado: 16 out. 2018. Disponível em: <<https://goo.gl/LZiUuR>>. Citado na página 38.
- LÓSCIO, B. F.; OLIVEIRA, H. R. d.; PONTES, J. C. d. S. Nosql no desenvolvimento de aplicações web colaborativas. *VIII Simpósio Brasileiro de Sistemas Colaborativos*, v. 10, n. 1, p. 11, 2011. Acessado: 01 jul. 2018. Disponível em: <<https://goo.gl/6nN6Fo>>. Citado 3 vezes nas páginas 39, 40 e 41.
- MARKETUP. *Recursos*. 2017. Acessado: 06 dez. 2018. Disponível em: <<https://marketup.com/recursos/>>. Citado na página 47.
- MAROMA. *VOCÊ SABE COMO MANTER A QUALIDADE DO SORVETE?* 2016. Acessado: 12 set. 2018. Disponível em: <<https://goo.gl/8gDzXH>>. Citado na página 29.
- MATHEW, A. B.; KUMAR, S. M. Analysis of data management and query handling in social networks using nosql databases. In: IEEE. *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*. 2015. p. 800–806. Acessado: 02 jul. 2018. Disponível em: <<https://goo.gl/1QoRrw>>. Citado na página 40.
- MIGUEL, J. *PROJETO DE IMPLANTAÇÃO DE INDÚSTRIA DE SORVETE*. 2010. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/6Gh4XC>>. Citado na página 29.
- MORAES, M.; OCUPACIONAIS, D. Construindo aplicações com nodejs. *São Paulo: Iátria*, 2011. Acessado: 11 jul. 2018. Disponível em: <<https://goo.gl/U5FpGu>>. Citado na página 37.
- NAYAK, A.; PORIYA, A.; POOJARY, D. Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems*, v. 5, n. 4, p. 16–19, 2013. Acessado: 02 jul. 2018. Disponível em: <<https://goo.gl/Y2YtAo>>. Citado na página 40.
- NODEBR. *Nodejs e MongoDB – Introdução ao Mongoose*. 2016. Acessado: 17 set. 2018. Disponível em: <<https://goo.gl/Wn6F6e>>. Citado na página 38.
- NODEMON.IO. *nodemon reload, automatically*. 2018. Acessado: 13 out. 2018. Disponível em: <<https://goo.gl/GvHMFY>>. Citado na página 38.
- NOGUERA, V. E. R. et al. Extensão de uma álgebra er para execução de consultas em bancos de dados nosql orientados a documentos. Universidade Federal de São Carlos, 2018. Acessado: 02 jul. 2018. Disponível em: <<https://goo.gl/tJb6tW>>. Citado 2 vezes nas páginas 40 e 41.

OLIVEIRA, D. de J. Uma proposta de arquitetura para single-page applications. 2017. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/nEKaaw>>. Citado 3 vezes nas páginas 31, 34 e 41.

PATEL, S. *Quick Es2015 Scripting Using Babel.js: Learn Es6 Important Features Quickly*. CreateSpace Independent Publishing Platform, 2016. Acessado: 12 set. 2018. ISBN 9781532783869. Disponível em: <<https://goo.gl/4BJUpu>>. Citado na página 39.

PAULINO, R. *Teoria da Cor para WebDesigners*. 2015. Acessado: 07 out. 2018. Disponível em: <<https://goo.gl/NyEQdv>>. Citado na página 56.

PEREIRA, C. *Aplicações web real-time com Node.js*. Casa do Código, 2014. ISBN 9788566250930. Disponível em: <<https://books.google.com.br/books?id=Wm-CCwAAQBAJ>>. Citado na página 37.

PEREIRA, M. *AngularJS: Uma abordagem prática e objetiva*. Novatec Editora, 2014. Acessado: 23 jul. 2018. ISBN 9788575224113. Disponível em: <<https://books.google.com.br/books?id=MUelBQAAQBAJ>>. Citado na página 46.

POKORNY, J. Nosql databases: a step to database scalability in web environment. *International Journal of Web Information Systems*, Emerald Group Publishing Limited, v. 9, n. 1, p. 69–82, 2013. Acessado: 02 jul. 2018. Disponível em: <<https://goo.gl/qaiJ6W>>. Citado na página 40.

PRESCOTT, P. *Programação em JavaScript*. Babelcube Inc., 2016. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/ffFdQK>>. Citado na página 32.

PRESSMAN, R.; MAXIM, B. *Engenharia de Software-8ª Edição*. [S.l.]: McGraw Hill Brasil, 2016. Citado 2 vezes nas páginas 52 e 53.

QUEIROZ, T. A. et al. *Sistema embarcado linux para análise de sensores de temperatura dht11 e lm35*. ERIPI, 2016. Acessado: 03 out. 2018. Disponível em: <<https://goo.gl/jhKh6U>>. Citado na página 43.

RANKING. Db-engines ranking. 2018. Acessado: 03 jul. 2018. Disponível em: <<https://goo.gl/NrQ1o8>>. Citado na página 41.

RASPBERRYPI.ORG. *Raspberry Pi 3 Model B*. 2014. Acessado: 03 out. 2018. Disponível em: <<https://goo.gl/dCrDST>>. Citado na página 51.

REIS, V. *Por que VueJS é uma boa opção?* 2016. Acessado: 12 set. 2018. Disponível em: <<https://goo.gl/ahua2F>>. Citado na página 35.

RICHARDSON, M.; WALLACE, S. Primeiros passos com o raspberry pi. *Primeira Edição. Novatec Editora Ltda*, p. 20, 2013. Acessado: 02 out. 2018. Disponível em: <<https://goo.gl/UPKez5>>. Citado 2 vezes nas páginas 44 e 52.

RUSCHEL, D. *Franchising*. 2017. Acessado: 22 set. 2018. Disponível em: <<https://goo.gl/W32766>>. Citado na página 25.

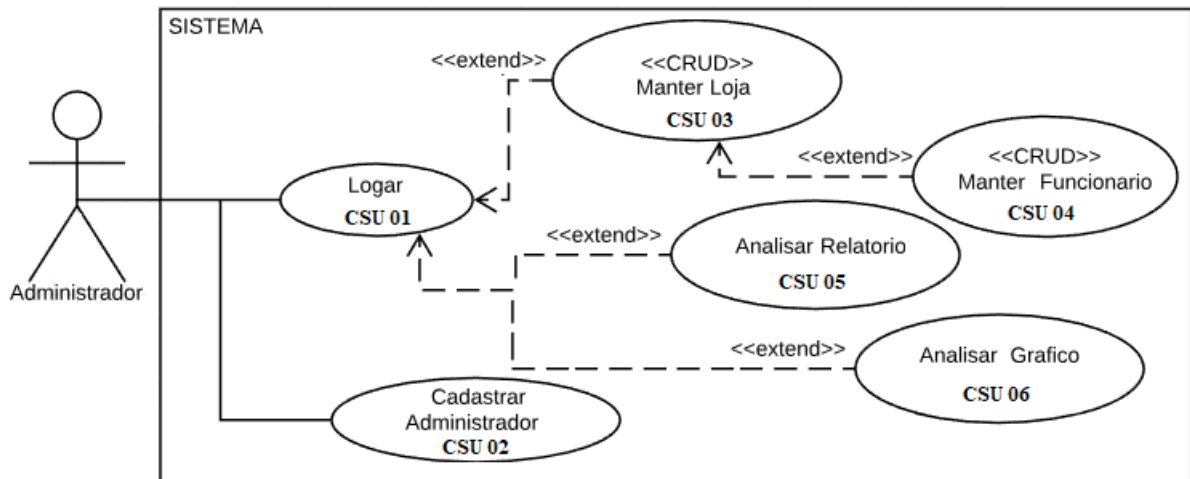
SANDOVAL, K. *Defining Stateful vs Stateless Web Services*. 2017. Acessado: 16 out. 2018. Disponível em: <<https://goo.gl/8pW3A3>>. Citado na página 42.

- SANDOVAL, P. E. P. Monitoramento remoto aplicado ao controle da umidade ambiental. 2014. Acessado: 02 out. 2018. Disponível em: <<https://goo.gl/F3vv9j>>. Citado na página 43.
- SANTOS, W. R. d. Restful web services e a api jax. 2015. Acessado: 15 out. 2018. Disponível em: <<https://goo.gl/28VVWk>>. Citado na página 32.
- SCAVUZZO, M.; NITTO, E. D.; CERI, S. Interoperable data migration between nosql columnar databases. In: IEEE. *Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International*. 2014. p. 154–162. Acessado: 01 jul. 2018. Disponível em: <<https://goo.gl/66mfNg>>. Citado na página 40.
- SEBRAE. Guia do empreendedor: Fluxo de caixa e custos na indústria. 2008. Acessado: 13 out. 2018. Disponível em: <<https://goo.gl/5fJZ35>>. Citado na página 30.
- SEBRAE. *Como se destacar no mercado de sorvetes*. 2017. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/tLUCBF>>. Citado 2 vezes nas páginas 25 e 29.
- SEO, H. *Usando o Axios no Vue.js*. 2018. Acessado: 13 out. 2018. Disponível em: <<https://goo.gl/41fU48>>. Citado na página 39.
- SILVA, G. N. d.; SILVA, T. N. d. Desenvolvimento de uma aplicação para disponibilização de aplicativos classificados conforme taxonomia de bloom. 2017. Acessado: 12 set. 2018. Disponível em: <<https://goo.gl/VYpvHf>>. Citado na página 34.
- SILVA, J. P. d. *EcoCIT: uma plataforma escalável para desenvolvimento de aplicações de IoT*. Dissertação (Mestrado) — Brasil, 2017. Acessado: 01 jul. 2018. Disponível em: <<https://goo.gl/wECLXo>>. Citado 2 vezes nas páginas 39 e 41.
- SILVA, M. C. d. Estação iot para monitoramento da temperatura e umidade do interior de veículos. 2018. Acessado: 03 out. 2018. Disponível em: <<https://goo.gl/XuJyMT>>. Citado na página 43.
- SILVA, M. R. R. d. Projeto e desenvolvimento de um sistema para gerenciamento de trabalhos de conclusão de curso. Universidade Federal de Uberlândia, 2017. Acessado: 17 jul. 2018. Disponível em: <<https://goo.gl/UgKYgz>>. Citado 2 vezes nas páginas 25 e 30.
- SILVA, P. H. P. *Qualidade de código com ESLint*. 2016. Acessado: 16 out. 2018. Disponível em: <<https://goo.gl/qCTmb7><https://goo.gl/41fU48>>. Citado na página 36.
- SINGER, T. Tudo conectado: conceitos e representações da internet das coisas. *Simpósio em tecnologias digitais e sociabilidade*, v. 2, p. 1–15, 2012. Acessado: 02 out. 2018. Disponível em: <<https://goo.gl/Rkdnn5>>. Citado na página 43.
- SOUZA, L. L. de. Desenvolvimento seguro de aplicações web seguindo a metodologia owasp. 2012. Acessado: 14 out. 2018. Disponível em: <<https://goo.gl/zfYzAD>>. Citado na página 39.
- STANESCU, L.; BREZOVAN, M.; BURDESCU, D. D. Automatic mapping of mysql databases to nosql mongodb. In: IEEE. *Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on*. 2016. p. 837–840. Acessado: 02 jul. 2018. Disponível em: <<https://goo.gl/3eKy2V>>. Citado na página 40.

- STONEBRAKER, M. Stonebraker on nosql and enterprises. *Communications of the ACM*, ACM, v. 54, n. 8, p. 10–11, 2011. Acessado: 02 jul. 2018. Disponível em: <<https://goo.gl/UqhGWP>>. Citado na página 41.
- TASSI, W. G. *Elaboração de um sistema de custos e preços para uma sorveteria*. 2014. Acessado: 07 jul. 2018. Disponível em: <<https://goo.gl/Q8vdX1>>. Citado na página 30.
- TELECOMUNICAÇÕES, C. d. E. de. *Tutorial de introdução ao python*. 2009. Acessado: 17 out. 2018. Disponível em: <<https://goo.gl/zUCtkc>>. Citado 2 vezes nas páginas 44 e 45.
- THINGSPEAK. *Learn More About ThingSpeak*. 2018. Acessado: 02 out. 2018. Disponível em: <<https://goo.gl/Td3Lyj>>. Citado na página 45.
- TRINDADE, M. *Jogo interativo para o ensino da leitura e escrita*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2016. Acessado: 23 jul. 2018. Disponível em: <<https://goo.gl/jovBte>>. Citado na página 46.
- VEDOVELLI, F. *Vue-router*. 2016. Acessado: 17 set. 2018. Disponível em: <<https://goo.gl/uNY6LY>>. Citado na página 36.
- ÁVILA, R. *Como fazer o fechamento de caixa*. 2015. Acessado: 17 out. 2018. Disponível em: <<https://goo.gl/Kn3KrF>>. Citado na página 30.
- VUEJS.ORG. *Guide: What is Vue.js?* 2016. Acessado: 12 set. 2018. Disponível em: <<https://goo.gl/Um1xLv>>. Citado na página 34.
- VUE.JS.ORG. *Usando Axios para Consumir APIs*. 2017. Acessado: 13 out. 2018. Disponível em: <<https://goo.gl/UuEY1N>>. Citado na página 39.
- VUETIFY. *Vuetify: Quick start*. 2018. Acessado: 12 set. 2018. Disponível em: <<https://goo.gl/sepuMd>>. Citado na página 35.
- WEBPACK.JS.ORG. *Conceitos*. 2015. Acessado: 11 out. 2018. Disponível em: <<https://goo.gl/AFbdh4>>. Citado na página 38.
- WEILER, D.; OTTEQUIR, D.; BECKER, F. W. *Protótipo para controle de nível e vazão de água utilizando conceito de internet das coisas em reservatórios hídricos de condomínios*. Faculdade Senac Blumenau, 2017. Acessado: 12 set. 2018. Disponível em: <<https://goo.gl/mLhKuL>>. Citado na página 36.
- WILLMOTT, M. *"From a React point of Vue...- comparing React.js to Vue.js for dynamic tabular data"*. 2016. Acessado: 12 set. 2018. Disponível em: <<https://goo.gl/3BWNRA>>. Citado na página 35.

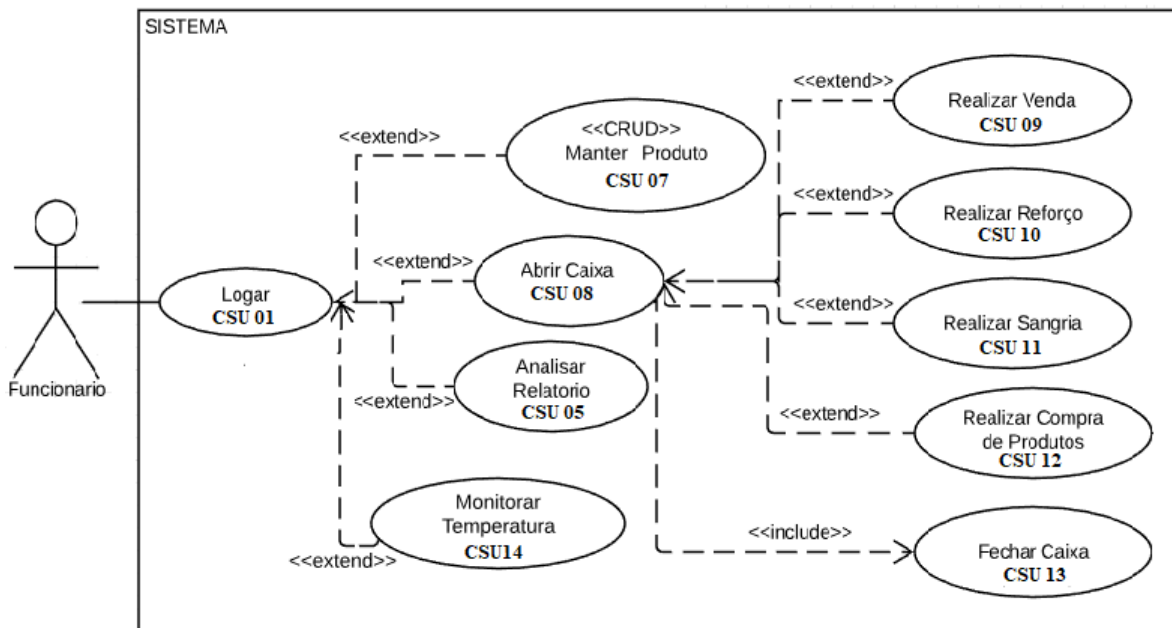
APÊNDICE A – Diagrama e Expansões de Casos de Uso

Figura 53 – Diagrama de Caso de Uso para administrador.



Fonte: Elaborado pela autora.

Figura 54 – Diagrama de Caso de Uso para funcionário.



Fonte: Elaborado pela autora.

ID: CSU01
Nome do CSU: Logar.
Sumário: Processo de identificação que permite que o sistema reconheça usuários cadastrados, isto é feito com usuário e senha.
Atores envolvidos: Administrador e Funcionário.
Pré-Condições: Os usuários devem estar previamente cadastrados.
Fluxo Principal: <ol style="list-style-type: none"> 1- O usuário informa o nome de usuário e a senha. 2- O sistema autentica o <i>login</i>. 3- O caso de uso é encerrado.
Variantes: Não aplicável.
Fluxo de Exceção: Usuário ou senha não são válidos , o sistema deve informar a falha na autenticação e permanecer na tela de <i>login</i> .
Pós-Condições: O sistema exibe a página inicial do sistema.

ID: CSU02
Nome do CSU: Cadastrar administrador.
Sumário: Processo de cadastro do administrador das filiais.
Atores envolvidos: Administrador.
Pré-Condições: -
Fluxo Principal: <ol style="list-style-type: none"> 1- O administrador entra com seus dados pessoais: nome completo, data de nascimento, sexo, CPF, endereço, telefone e <i>email</i>. 2- O administrador escolhe seu usuário e senha. 3- O administrador clica em "Cadastrar". 4- O caso de uso é encerrado.
Variantes: Não aplicável.
Fluxo de Exceção: Dados considerados inválidos ou nome de usuário já pertencente a outro administrador, o sistema deve colocar uma notificação abaixo do campo para orientar o administrador.
Pós-Condições: Cadastro do administrador mantido no sistema e então é exibido a página de <i>Login</i> .

ID: CSU03
Nome do CSU: Manter Loja.
Sumário: Este caso de uso tem por objetivo permitir consultar, incluir, alterar e excluir o registro da loja no sistema.
Atores envolvidos: Administrador.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de administrador.
Fluxo Principal: <ul style="list-style-type: none"> 1- O caso de uso é iniciado quando o administrador acessa o sistema e seleciona a opção "Lojas" no Menu. 2- O sistema apresenta a Tela Lojas. 3- O administrador consulta informações sobre todas as suas lojas. 4- O caso de uso é encerrado.
Variantes: <ul style="list-style-type: none"> V1- Cadastrar Loja. <ul style="list-style-type: none"> V1.1- O administrador clica em "Cadastrar Loja". V1.2- O sistema abre um formulário sobre a loja para ser preenchido. V1.3- O administrador informa os dados da loja: nome, cidade, estado, endereço e telefone. V1.4- O administrador clica em "Salvar". V1.5- O sistema apresenta a mensagem: "Operação realizada com sucesso". V2- Alterar Loja. <ul style="list-style-type: none"> V2.1- O administrador clica no ícone de editar loja. V2.2- O sistema abre um formulário com as informações da loja selecionada. V2.3- O administrador modifica os campos desejados. V2.4- O administrador clica em "Salvar". V2.5- O sistema apresenta a mensagem: "Operação realizada com sucesso". V3- Excluir Loja. <ul style="list-style-type: none"> V3.1- O administrador clica no ícone de excluir loja. V3.2- O sistema solicita a confirmação da exclusão. V3.3- O ator confirma a exclusão selecionando a opção "Confirmar".
Fluxo de Exceção: Se alguns dos dados forem considerados inválidos na hora do cadastro ou edição, o sistema deve colocar uma notificação abaixo do campo para orientar o administrador.
Pós-Condições: Lojas mantidas no sistema e é exibido a Tela Lojas.

ID: CSU04
Nome do CSU: Manter funcionário.
Sumário: Este caso de uso tem por objetivo permitir consultar, incluir, alterar e excluir o registro de funcionários no sistema.
Atores envolvidos: Administrador.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de administrador. O funcionário deverá ter entregado toda a documentação solicitada na contratação.
Fluxo Principal: <ol style="list-style-type: none"> 1- O caso de uso é iniciado quando o administrador acessa o sistema e seleciona a opção "Funcionários". 2- O sistema apresenta a Tela Funcionários. 3- O administrador consulta informações sobre todos os seus funcionários. 4- O caso de uso é encerrado.
Variantes: <ul style="list-style-type: none"> V1- Cadastrar Funcionário. <ul style="list-style-type: none"> V1.1- O administrador clica em "Cadastrar Funcionário". V1.2- O sistema abre um formulário sobre o funcionário para ser preenchido. V1.3- O administrador informa os dados do funcionário: nome, data de nascimento, data de admissão, sexo, CPF, <i>email</i>, telefone, usuário, senha e a qual loja ele trabalha. V1.4- O administrador clica em "Salvar". V1.5- O sistema apresenta a mensagem: "Operação realizada com sucesso". V2- Alterar Funcionário <ul style="list-style-type: none"> V2.1- O administrador clica no ícone de editar funcionário. V2.2- O sistema abre um formulário com as informações do funcionário selecionado. V2.3- O administrador modifica os campos desejados. V2.4- O administrador clica em "Salvar". V2.5- O sistema apresenta a mensagem: "Operação realizada com sucesso". V3- Excluir Funcionário. <ul style="list-style-type: none"> V3.1- O administrador clica no ícone de excluir funcionário. V3.2- O sistema solicita a confirmação da exclusão. V3.3- O ator confirma a exclusão selecionando a opção "Confirmar".
Fluxo de Exceção: Se alguns dos dados forem considerados inválidos na hora do cadastro ou edição, o sistema deve colocar uma notificação abaixo do campo para orientar o administrador.
Pós-Condições: Funcionários mantidos no sistema e é exibido a Tela Lojas.

ID: CSU05
Nome do CSU: Analisar Relatório.
Sumário: Este caso de uso tem por objetivo permitir ao administrador consultar relatório financeiro de fluxo de caixa de suas lojas.
Atores envolvidos: Administrador e Funcionário.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de administrador.
Fluxo Principal: <ol style="list-style-type: none">1- O caso de uso é iniciado quando o administrador acessa o sistema e seleciona a opção "Fluxo de Caixa" na seção "Financeiro".2- O sistema apresenta a Tela Fluxo de Caixa.3- O administrador seleciona a data para visualizar o fluxo de caixa.4- O administrador clica em "Buscar".5- O sistema apresenta o fluxo de caixa de todas as lojas referente a data de pesquisa.6- O caso de uso é encerrado.
Variantes: <p>V1 - Funcionário analisando relatório.</p> <p>V1.1- O caso de uso é iniciado quando o funcionário acessa o sistema e seleciona a opção "Fluxo de Caixa" na seção "Relatórios".</p> <p>V1.2- O sistema apresenta a Tela Fluxo de Caixa.</p> <p>V1.3- O funcionário seleciona a data para visualizar o fluxo de caixa.</p> <p>V1.4- O funcionário clica em "Buscar".</p> <p>V1.5- O sistema apresenta o fluxo de caixa daquele funcionário referente a data de pesquisa.</p> <p>V1.6- O caso de uso é encerrado.</p>
Fluxo de Exceção: Se a data não corresponder a nenhum fluxo de caixa das lojas do administrador, o sistema deve informar através de uma notificação ao administrador.
Pós-Condições: -

ID: CSU06
Nome do CSU: Analisar Gráfico.
Sumário: Este caso de uso tem por objetivo permitir ao administrador consultar o gráfico de comparação de vendas das suas lojas.
Atores envolvidos: Administrador.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de administrador.
Fluxo Principal: <ol style="list-style-type: none">1- O caso de uso é iniciado quando o administrador acessa o sistema e seleciona a opção "Comparação de Vendas" na seção "Financeiro".2- O sistema apresenta a Tela Gráfico de Comparação de Vendas das Lojas.3- O administrador seleciona o intervalo da data para visualizar o gráfico.4- O administrador clica em "Gerar".5- O sistema apresenta o gráfico de comparação da venda das lojas no intervalo desejado.6- O caso de uso é encerrado.
Variantes: -
Fluxo de Exceção: Data inexistente, o sistema deve limpar a data de início e fim do intervalo e informar através de uma notificação o erro ao administrador.
Pós-Condições: -

ID: CSU07
Nome do CSU: Manter produto.
Sumário: Este caso de uso tem por objetivo permitir consultar, incluir, alterar e excluir o registro de produtos no sistema.
Atores envolvidos: Funcionário.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de funcionário.
Fluxo Principal: <ul style="list-style-type: none"> 1- O caso de uso é iniciado quando o funcionário acessa o sistema e seleciona a opção "Produtos". 2- O sistema apresenta a Tela Produtos. 3- O funcionário consulta informações sobre todos os produtos da loja em que ele trabalha. 4- O caso de uso é encerrado.
Variantes: <ul style="list-style-type: none"> V1- Cadastrar Produto. <ul style="list-style-type: none"> V1.1- O funcionário clica em "Cadastrar Produto". V1.2- O sistema abre um formulário sobre o produto para ser preenchido. V1.3- O funcionário informa os dados do produto: nome, descrição, unidade, preço de compra e preço de venda. V1.4- O funcionário clica em "Salvar". V1.5- O sistema apresenta a mensagem: "Operação realizada com sucesso". V2- Alterar Produto. <ul style="list-style-type: none"> V2.1- O funcionário clica no ícone de editar produto. V2.2- O sistema abre um formulário com as informações do produto selecionado. V2.3- O funcionário modifica os campos desejados. V2.4- O funcionário clica em "Salvar". V2.5- O sistema apresenta a mensagem: "Operação realizada com sucesso". V3- Excluir Produto. <ul style="list-style-type: none"> V3.1- O funcionário clica no ícone de excluir produto. V3.2- O sistema solicita a confirmação da exclusão. V3.3- O funcionário confirma a exclusão selecionando a opção "Confirmar".
Fluxo de Exceção: Se alguns dos dados forem considerados inválidos na hora do cadastro ou edição, o sistema deve colocar uma notificação abaixo do campo para orientar o funcionário.
Pós-Condições: Produtos mantidos no sistema e é exibido a Tela Produtos.

ID: CSU08
Nome do CSU: Abrir Caixa.
Sumário: Este caso de uso tem por objetivo permitir o funcionário abrir o caixa.
Atores envolvidos: Funcionário.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de funcionário e não deve ter nenhum caixa em situação aberta com este funcionário.
Fluxo Principal: <ol style="list-style-type: none">1- O caso de uso é iniciado quando o funcionário acessa o sistema e seleciona a opção "Abrir Caixa".2- O sistema apresenta a Tela Abrir Caixa.3- O funcionário informa o valor disponível em caixa para iniciar as operações.4- O administrador clica em "Confirmar Abertura".5- O sistema apresenta a mensagem: "Operação realizada com sucesso".6- O caso de uso é encerrado.
Variantes: -
Fluxo de Exceção: Se o valor informado for inferior ou igual a zero, o sistema deve colocar uma notificação para orientar o funcionário.
Pós-Condições: O sistema armazena informações sobre abertura de caixa e é exibido a Tela Vendas.

ID: CSU09
Nome do CSU: Realizar Venda.
Sumário: Este caso de uso tem por objetivo permitir o funcionário efetuar vendas.
Atores envolvidos: Funcionário.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de funcionário e deve ter caixa em situação aberta com este funcionário.
Fluxo Principal: <ul style="list-style-type: none"> 1- O caso de uso é iniciado quando o funcionário acessa o sistema e seleciona a opção "Realizar Venda". 2- O sistema apresenta a Tela Venda. 3- O funcionário adiciona produtos no "carrinho"informando a quantidade e o código e clicando em "Adicionar", isso pode ser feito quantas vezes for necessário. 4- O funcionário clica em "Receber Conta". 5- O funcionário seleciona a forma de pagamento. 6- O funcionário clica em "Finalizar Compra". 7- O sistema apresenta a mensagem: "Operação realizada com sucesso". 8- O caso de uso é encerrado.
Variantes: V1- Pagamento em Dinheiro <ul style="list-style-type: none"> V1.1- O funcionário informa o valor pago. V1.2- O sistema retorna o valor troco do cliente. V2- Pagamento em Cartão de Crédito. <ul style="list-style-type: none"> V2.1- O funcionário informa a quantidade de parcelas. V2- Pagamento em Cartão de Débito.
Fluxo de Exceção: Se nenhuma forma de pagamento for selecionada ou o valor pago pelo cliente informado for inferior ao valor total da compra, o sistema deve colocar uma notificação para orientar o funcionário.
Pós-Condições: O sistema armazena informações sobre a venda, atualiza estoque e é exibido a Tela Vendas.

ID: CSU10
Nome do CSU: Realizar Reforço.
Sumário: Este caso de uso tem por objetivo permitir o funcionário realizar reforço de caixa.
Atores envolvidos: Funcionário.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de funcionário e deve ter caixa em situação aberta com este funcionário.
Fluxo Principal: <ol style="list-style-type: none">1- O caso de uso é iniciado quando o funcionário acessa o sistema e seleciona a opção "Realizar Reforço".2- O sistema apresenta a Tela Reforço com o valor disponível em caixa e um campo a ser preenchido: o valor do reforço de caixa.3- O funcionário informa o valor do reforço.4- O administrador clica em "Confirmar Reforço".5- O sistema apresenta a mensagem: "Operação realizada com sucesso".6- O caso de uso é encerrado.
Variantes: -
Fluxo de Exceção: Se o valor informado for inferior ou igual a zero, o sistema deve colocar uma notificação para orientar o funcionário.
Pós-Condições: O sistema armazena informações sobre reforço de caixa e é exibido a Tela Vendas.

ID: CSU11
Nome do CSU: Realizar Sangria
Sumário: Este caso de uso tem por objetivo permitir o funcionário realizar sangria de caixa.
Atores envolvidos: Funcionário.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de funcionário e deve ter caixa em situação aberta com este funcionário.
Fluxo Principal: <ol style="list-style-type: none">1- O caso de uso é iniciado quando o funcionário acessa o sistema e seleciona a opção "Realizar Sangria".2- O sistema apresenta a Tela Sangria com o valor disponível em caixa e um campo a ser preenchido: o valor da sangria.3- O funcionário informa o valor da sangria.4- O funcionário clica em "Confirmar Sangria".5- O sistema apresenta a mensagem: "Operação realizada com sucesso".6- O caso de uso é encerrado.
Variantes: -
Fluxo de Exceção: Se o valor informado for inferior a zero ou superior ao valor total disponível em caixa, o sistema deve colocar uma notificação para orientar o funcionário.
Pós-Condições: O sistema armazena informações sobre sangria de caixa e é exibido a Tela Vendas.

ID: CSU12
Nome do CSU: Realizar Compra de Produtos.
Sumário: Este caso de uso tem por objetivo permitir o funcionário adicionar produtos ao estoque.
Atores envolvidos: Funcionário.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de funcionário e deve ter caixa em situação aberta com este funcionário.
Fluxo Principal: <ol style="list-style-type: none">1- O caso de uso é iniciado quando o funcionário acessa o sistema e seleciona a opção "Estoque".2- O sistema apresenta a Tela Estoque com todos os produtos da loja em que ele trabalha com os valores disponíveis no estoque o qual seria seu estoque mínimo.3- O funcionário seleciona qual produto será comprado mais quantidade.4- O funcionário informa a quantidade e então é calculado o valor total desta compra.5- O funcionário clica em "Adicionar".6- O sistema apresenta a mensagem: "Operação realizada com sucesso".7- O caso de uso é encerrado.
Variantes: -
Fluxo de Exceção: Se o valor informado for inferior a zero ou o valor da compra for superior ao valor disponível em caixa, o sistema deve colocar uma notificação para orientar o funcionário.
Pós-Condições: O sistema atualiza a quantidade do produto comprado e é exibido a Tela Estoque.

ID: CSU13
Nome do CSU: Fechar Caixa.
Sumário: Este caso de uso tem por objetivo permitir o funcionário realizar o fechamento de caixa.
Atores envolvidos: Funcionário.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de funcionário e deve ter caixa em situação aberta com este funcionário.
Fluxo Principal: <ul style="list-style-type: none"> 1- O caso de uso é iniciado quando o funcionário acessa o sistema e seleciona a opção "Fechar Caixa". 2- O sistema apresenta a Tela Fechar Caixa com o valor disponível em caixa. 3- O funcionário clica em "Confirmar Fechamento". 4- O sistema apresenta a mensagem: "Operação realizada com sucesso". 5- O caso de uso é encerrado.
Variantes: -
Fluxo de Exceção: -
Pós-Condições: : O sistema armazena informações sobre fechamento de caixa e é exibido a Tela Abrir Caixa.

ID: CSU14
Nome do CSU: Monitorar Temperatura.
Sumário: Este caso de uso tem por objetivo permitir ao funcionário monitorar a temperatura dos <i>freezers</i> da loja em que trabalha.
Atores envolvidos: Funcionário.
Pré-Condições: O usuário deve estar logado no sistema com o perfil de funcionário.
Fluxo Principal: <ul style="list-style-type: none"> 1- O caso de uso é iniciado quando o funcionário acessa o sistema e seleciona a opção "Monitorar Temperatura". 2- O sistema apresenta a Tela Monitoramento da Temperatura com o gráficos das oscilações das temperaturas de todos os <i>freezers</i> daquela loja . 3- O caso de uso é encerrado.
Variantes: V1- O funcionário deseja cadastrar um novo sensor de monitoramento. <ul style="list-style-type: none"> V1.1- O funcionário clica em "Adicionar Novo Sensor". V1.2- O funcionário informa o canal e o <i>field</i>(campo). V1.3- O funcionário clica em "Adicionar". V1.4- O sistema apresenta a mensagem: "Operação realizada com sucesso".
Fluxo de Exceção: Se o canal e <i>field</i> informado não existirem ou ocorrer algum erro durante a primeira requisição de dados(feita antes de efetivar o cadastro para validar), o sistema deve informar através de uma notificação.
Pós-Condições: : -