UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

LUCAS RODRIGUES DE MORAES

SISTEMA DE GERENCIAMENTO NUTRICIONAL

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2022

LUCAS RODRIGUES DE MORAES

SISTEMA DE GERENCIAMENTO NUTRICIONAL

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso 2, do curso de Engenharia da Computação da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. Alessandro Silveira Duarte.

CORNÉLIO PROCÓPIO
2022

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Diretoria de Graduação
Departamento Acadêmico de Computação
Engenharia de Computação

TERMO DE APROVAÇÃO

Sistema de Gerenciamento Nutricional

por

Lucas Rodrigues de Moraes

Este Trabalho de Conclusão de Curso de graduação foi julgado adequado para obtenção do Título de Bacharel em Engenharia de Computação e aprovado em sua forma final pelo Programa de Graduação em 2022 da Universidade Tecnológica Federal do Paraná.

Cornélio Procópio, 06/12/2022

_____

Prof. Me. Alessandro Silveira Duarte

_____

Prof. Me. José Antônio Gonçalves

_____

Prof. Dr. Alexandre L'Erário

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso"

Dedico este trabalho à minha família.

RESUMO

MORAES, Lucas Rodrigues. **Sistema de Gerenciamento Nutricional**. 2022. 53 f. Trabalho de Conclusão de Curso – Engenharia da Computação. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2022.

Considerando o crescente número de doenças relacionadas a má alimentação, é perceptível a necessidade de oferecer softwares que possam auxiliar a interação entre pacientes e nutricionistas. Este trabalho teve como objetivo construir um sistema que permitisse aos nutricionistas criar dietas alimentares, definindo refeições e alimentos para os pacientes, e utilizar a classificação de alimentos denominada NOVA. O software desenvolvido trata-se de um Website, implementado utilizando principalmente a linguagem de programação Java e TypeScript, aliados aos *frameworks* de desenvolvimento de código *Springboot* e *React*. A metodologia de desenvolvimento utilizada foi o Scrum Solo, pois considera-se que esta oferece um bom suporte a equipes reduzidar ou de um único desenvolvedor.

Palavras-chave: Website, *Springboot, React*, NOVA, Dieta.

ABSTRACT

MORAES, Lucas Rodrigues. **Nutrition Management System**. 2022. 53 f. Trabalho de Conclusão de Curso – Engenharia de Software. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2022.

Considering the growing number of diseases related to poor diet, it is noticeable the need to offer software that can help the interactions between patient and nutritionists. This work aimed to build a system that would allow nutritionists to create diets, defining meals and foods for patients, and using the NOVA food classification. The developed software is a Website, implemented mainly using the Hava and TypeScript programming languages, combined with Springboot and React code development frameworks. The development methodology used was Scrum Solo, as it offered great support to teams of a single developer.

Keywords: Diet. NOVA classification. *Springboot*. *React*.

LISTA DE FIGURAS

FIGURA 1 – EXEMPLO DE COMUNICAÇÃO REST..... 14
FIGURA 2 – REPRESENTAÇÃO VISUAL SPRINGBOOT..... 18
FIGURA 3 – EXEMPLO DE ANOTAÇÕES SPRING..... 18
FIGURA 4 – EXEMPLO DE OBJETO UTILIZANDO LOMBOK..... 19
FIGURA 5 – EXEMPLO DE CÓDIGO EM JAVASCRIPT..... 20
FIGURA 6 – EXEMPLO DE CÓDIGO EM TYPESCRIPT..... 20
FIGURA 7 – DISPOSIÇÃO DOS COMPONENTES NO FRONTEND..... 21
FIGURA 8 – TABELAS EXISTENTES NO SISTEMA..... 22
FIGURA 9 – CONTEÚDO DA TABELA DE ALIMENTOS..... 22
FIGURA 10 – PADRÃO MVC..... 24
FIGURA 11 – DISPOSIÇÃO DOS PACOTES NO BACKEND..... 25
FIGURA 12 – EXEMPLO DE UTILIZAÇÃO DE DTO..... 26
FIGURA 13 – EXEMPLO DE CABEÇALHO JTW..... 27
FIGURA 14 – EXEMPLO DE CONTEÚDO JWT..... 27
FIGURA 15 – EXEMPLO DE ASSINATURA JWT..... 28
FIGURA 16 – EXEMPLO DE TOKEN JWT..... 28
FIGURA 17 – DIAGRAMA DE CASOS DE USO CRUD..... 31
FIGURA 18 – DIAGRAMA DE CASOS DE USO DO CLIENTE..... 32
FIGURA 19 – DIAGRAMA DE ENTIDADE-RELACIONAMENTO..... 33
FIGURA 20 – EXEMPLO DE TESTE UNITÁRIO..... 34
FIGURA 21 – CARACTERÍSTICAS DO SCRUM SOLO..... 35
FIGURA 22 – LANDING PAGE E MENU DE NAVEGAÇÃO DESKTOP..... 39
FIGURA 23 – TELA DE LOGIN..... 40
FIGURA 24 – TELA DE LISTAGEM DOS ALIMENTOS..... 41
FIGURA 25 – TELA DE CRUD DE ALIMENTOS..... 42
FIGURA 26 – TELA DE CADASTRO E EDIÇÃO DE ALIMENTOS..... 43
FIGURA 27 – CONTROLADOR DE INFORMAÇÕES CORPORAIS..... 44
FIGURA 28 – TELA DE DIETA..... 44

LISTA DE QUADROS

QUADRO 1 – GRUPOS DA CLASSIFICAÇÃO NOVA..... 12
QUADRO 2 – MÉTODOS HTTP..... 14
QUADRO 3 – SISTEMAS SIMILARES..... 15
QUADRO 4 – FERRAMENTAS UTILIZADAS..... 16
QUADRO 5 – REQUISITOS NÃO FUNCIONAIS..... 29
QUADRO 6 – REQUISITOS FUNCIONAIS..... 30
QUADRO 7 – PRIMEIRAS ATIVIDADES PRODUCT BACKLOG..... 36
QUADRO 8 – SPRINT’S REALIZADAS..... 38

LISTA DE ACRÔNIMOS

API	Application Programming Interface
CSS	Cascading Styles Sheets
CRUD	Create Register Update Delete
DCNT	Doenças Crônicas Não Transmissíveis
DTO	Data Transfer Object
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Enviroment
JPA	Java Persistence API
JSON	Javascript Object Notation
JVM	Java Virtual Machine
JWT	JSON Web Token
MVC	Model View Controller
NUPENS	Núcleo de Pesquisas Epidemiológicas em Nutrição e Saúde
PSP	Personal Software Process
REST	Representational State Transfer
RSS	Rich Site Summary
SOAP	Simple Object Access Protocol
URL	Uniform Resource Locator
WSDL	Web Services Description Language
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO..... 10
2 FUNDAMENTAÇÃO TEÓRICA..... 12
2.1 A classificação NOVA de alimentos..... 12
2.2 Serviços Web, Rest e HTTP..... 13
2.5 Sistemas similares..... 15
3 FERRAMENTAS..... 16
3.1 Java..... 16
3.2 Springboot..... 17
3.3 Lombok..... 18
3.4 Typescript..... 19
3.5 React..... 20
3.6 H2 DATABASE..... 22
4 DESENVOLVIMENTO, TESTES E METODOLOGIA..... 23
4.1 Padrões de projeto..... 23
4.1.1 Desenvolvimento em camadas..... 23
4.1.2 Data Transfer Objects..... 26
4.1.3 JSON Web Token..... 27
4.2 Análise de requisitos..... 29
4.2.1 Requisitos não funcionais e funcionais..... 29
4.2.3 Diagrama de casos de uso e entidade-relacionamento..... 31
4.3 Testes..... 34
4.4 Metodologia..... 35
4.4.1 Scrum Solo..... 35
4.4.2 Cronograma..... 37
5 SISTEMA DESENVOLVIDO..... 39
5.1 Navegação..... 39
5.2 Funcionalidades..... 41
5.2.1 Tela de Listagem dos Alimentos..... 41
5.2.2 Telas de Crud..... 42
5.2.3 Tela de Dieta e informação corporal próprias..... 44
6 CONSIDERAÇÕES FINAIS..... 45
APÊNDICE A – Product backlog..... 50
APÊNDICE B – Telas do sistema..... 53

1 INTRODUÇÃO

Uma dieta inadequada pode ser descrita como aquela onde o indivíduo consome uma quantidade irregular de macronutrientes, micronutrientes ou calorias. Estes excessos ou carências de nutrientes podem afetar o sistema imunológico ao passo que as calorias afetam diretamente a disposição, ganho e perda de peso (MALTA *et al*., 2014). O desiquilíbrio presente na dieta resulta em maior risco a suscetíveis doenças crônicas não transmissíveis, conhecidas por DCNT's, sendo exemplo destas: o diabetes, a hipertensão e a obesidade.

É possível observar que no Brasil os índices de obesidade tornaram-se mais preocupantes a cada ano, e isto é resultado de não apenas a falta da prática de exercícios físicos, mas também devido à má alimentação (LOUZADA *et al*., 2017). Para combater este problema, o governo criou os Guias Alimentares, lançados anualmente no Brasil, que buscam informar a população sobre os riscos de consumir certos alimentos, considerados prejudiciais, e quais outros devem compor uma dieta com intuito de alcançar uma alimentação saudável.

Os primeiros guias alimentares, lançados em meados de 2006, utilizavam as pesquisas e estudos de sua época e de certa forma podem ser considerados essenciais naquele momento, porém, com os debates e os novos estudos acerca das classificações alimentares, novos padrões surgiram (MONTEIRO et al., 2017). Devido a estas renovações, nestes guias de classificação, os softwares produzidos ao decorrer do tempo podem tornar-se obsoletos ou até mesmo pouco recomendados, visto que estes utilizam os estudos disponíveis durante seu tempo de concepção.

Em 2020, a pandemia de Covid 19 retomou a importância de se manter saudável, pois, foi notado que esta era mais mortal entre indivíduos que possuíam alguma enfermidade. Para uma dieta ser recomendada, é necessário que existam quantidades específicas de alimentos e refeições, levando em consideração a necessidade individual de cada pessoa. Uma dieta recomendada à um atleta de fisiculturismo, por exemplo, dificilmente será igualmente aplicada a um adulto não atleta, pois, existem diferenças nas necessidades destes indivíduos. Estas diferenças se aplicam nos nutrientes, calorias, e até mesmo na quantidade de refeições por dia, ou o modo de preparo dos alimentos contidos na dieta. Esta necessidade de uma dieta individual faz que a existência de profissionais da área de nutrição seja essencial, pois,

uma dieta feita por um amador dificilmente possuirá um balanço ideal e estará mais propensa a erros na seleção dos alimentos.

Um erro muito comum presente na dieta da população em geral é o consumo de alimentos ultraprocessados (MONTEIRO et al., 2017). Estes alimentos são grandes riscos para a saúde, pois, possuem um grau de saciedade baixo e em contrapartida são altamente calóricos. A classificação NOVA busca classificar os alimentos por meio do grau de processamento, contendo alimentos naturais, processados, ultra processados e ingredientes culinários. Esta classificação é eficiente, pois, foi comprovado por meio de estudos que os excessos normalmente ocorrem em alimentos com grau de processamento maior.

Atualmente em 2022, ainda é comum que os aplicativos de dieta existentes façam uma menção detalhada dos nutrientes e não indique nada perante o grau de processamento destes. O problema disto é que alimentos ultra processados poderiam estar próximos a alimentos naturais, o que gera uma certa dúvida sobre o que é recomendado ou não. Um exemplo desta ocorrência é a proximidade entre os cereais matinais industrializados, contendo doses altas de açúcar, próximos a grãos naturais como a aveia.

Diante da presente necessidade de facilitar a interação entre paciente e nutricionista, este trabalho apresenta o desenvolvimento de um software capaz de registrar e alterar diversas entidades relacionadas ao domínio da nutrição, fazendo o uso da classificação NOVA de alimentos. O objetivo do sistema desenvolvido pode ser descrito por:

a) Resgatar informações de dieta individual;
b) Cálculo de calorias e massa corpórea;
c) Gerenciar alimentos, refeições, dietas e usuários, assim como a interação entre estes.

Para a melhor compreensão dos conteúdos abordados neste trabalho, ele foi dividido em seis capítulos. O capítulo 2, Fundamentação Teórica, apresenta a bibliografia utilizada e os conceitos técnicos estudados. As ferramentas e os métodos utilizados no desenvolvimento estão expostas no capítulo 3. Questões sobre o desenvolvimento do sistema em si, como padrões de projeto e diagramas, podem ser encontradas no capítulo 4. O capítulo 5 contém uma visão geral do sistema desenvolvido, apresentando as telas existentes e suas funcionalidades. Por fim, os resultados obtidos e a conclusão estão no capítulo 6.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo contém o referencial teórica necessário para que o sistema proposto fosse desenvolvido. Primeiramente será explicitado o problema existente perante as classificações alimentares e posteriormente conceitos fundamentais para a compreensão de uma aplicação web moderna.

2.1 A classificação NOVA de alimentos

A classificação NOVA, trata-se da mais recente classificação de alimentos no momento, foi desenvolvida pelo Núcleo de Pesquisas Epidemiológicas em Nutrição e Saúde (NUPENS) e é utilizada em diferentes estudos que demonstraram a nocividade dos alimentos ultraprocessados a saúde humana, onde seus desenvolvedores apontam uma grande necessidade de que a população em geral reduza drasticamente o consumo deste tipo alimento (MOUBARAC *et al*., 2017). Esta classificação impõe uma grande ênfase ao propósito e a natureza do grau de processamento utilizado pela indústria alimentar para produzir determinado alimento.

Segundo a NOVA, os alimentos podem ser divididos em quatro grupos distintos, sendo estes:

Quadro 1 – Grupos da classificação NOVA

Grupo	Descrição
1)In natura	Podem possuir apenas o processamento com o intuito de separar partes do alimento comestíveis das não comestíveis ou que facilitem o transporte e refrigeração deste. Temos como exemplo as frutas, verduras e carnes.
2)Ingredientes culinários	Estes alimentos são utilizados para aumentar a palatabilidade dos alimentos do grupo 1 e devem ser utilizados com cautela para evitar excessos. Sais, óleos e açucares são exemplos de alimentos deste grupo.
3)Processados	Os alimentos deste grupo são obtidos por meio do preparo industrial de alimentos do grupo 1 e 2. Esta junção não pode ser reproduzida de maneira caseira e por este motivo é considerada perigosa, visto que o consumidor final não possui controle sobre as quantidades de alimentos utilizadas de cada grupo. Temos como exemplo as frutas em conserva e peixes enlatados.
4)Ultraprocessados	Este grupo possui os alimentos com o menor valor nutricional, pois a composição dos alimentos ultraprocessados utilizam uma quantidade muito baixa de alimentos do grupo 1(MOUBARAC et al., 2017). Vale ressaltar que estes alimentos possuem um grau de saciedade baixo e uma densidade calórica alta, o que resulta numa maior facilidade em cometer excessos na dieta.

Fonte: Autoria própria (2022).

Utilizando a classificação NOVA, detalhada anteriormente, o sistema de gerenciamento nutricional foi desenvolvido levando também em consideração diversos conceitos ligados a desenvolvimento de software. Estes conceitos serão descritos na subseção a seguir.

2.2 Serviços WEB, Rest e HTTP

Um serviço web pode ser considerado todo recurso obtido por meio da navegação pela internet. Para que este recurso esteja disponível, é necessário que haja um fornecedor, também descrito como servidor, em algum local do mundo para que um cliente possa interagir com este (TANENBAUM; MAARTEN VAN STEEN, 2016).

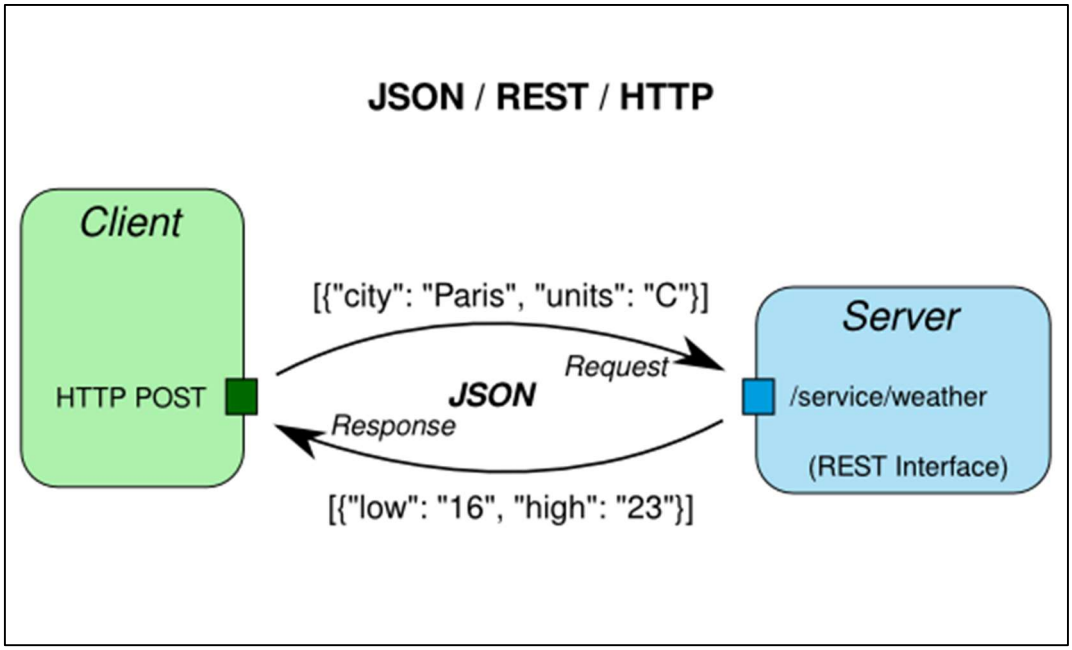
Estes serviços são desenvolvidos seguindo determinados padrões, para que assim, diferentes clientes consigam consumir e acessar estes recursos. Os principais protocolos existentes atualmente utilizados para exercer a comunicação entre cliente e servidor são Simple Object Acess Protocol (SOAP) e Representational State Transfer (REST).

O protocolo SOAP, por exemplo, utilizará arquivos Extensible Markup Language, ou XML, para comunicar-se, usando o protocolo Hypertext Transfer Protocol, ou HTTP, como a maneira de transportar os dados. Este protocolo possui também um documento que contém as informações da localidade deste serviço a ser fornecido e suas funcionalidades, sendo que este documento tem o nome Web Services Description Language, conhecido por WSDL.

Diferente do protocolo SOAP, o REST é capaz de receber e responder mensagens em diferentes formatos, como XML, Javascript Object Notation (JSON), Really Simple Syndication (RSS) e outros. Para transmitir seus dados, o protocolo REST também utilizará o protocolo de transferência HTTP. O REST é o padrão mais popular utilizado atualmente em serviços web (BARRY, 2021).

A Figura 1, exemplifica uma comunicação entre servidor e cliente onde o consumir envia uma requisição REST, por meio de uma URL e parâmetros HTTP, e o provedor responde à esta requisição por meio de um arquivo JSON.

Figura 1 – Exemplo de comunicação REST.



Fonte: https://miro.medium.com/max/1124/1*YFHUPQPqWLycmi7h17xr3A.png(2022).

Este protocolo é completamente suportado em uma aplicação utilizando o *framework Spring*, utilizado no desenvolvimento do *backend* do sistema. Isto ocorre, pois, o *Spring* fornece diversas anotações que podem ser utilizadas nas classes ou métodos, onde estas facilitam o desenvolvimento e resolvem diversos problemas relacionados a requisições e parâmetros.

O protocolo de transporte de dados HTTP, foi projetado para permitir a transferência em ambas as direções sobre uma arquitetura cliente-servidor. Este protocolo pode ser considerado flexível pois com ele é possível fazer a transferência de qualquer tipo de dado. O HTTP possui operações que são utilizadas para por meio do envio de mensagem do cliente ao servidor, que irá tomar diferentes ações sobre os dados existentes a partir destas operações (YU; WU; YIN, 2003).

O quadro 2 apresenta os métodos HTTP utilizados na aplicação desenvolvida, seus significados e um exemplo no sistema desenvolvido.

Quadro 2 – Métodos HTTP

Método	Função resultando no servidor	Exemplo no trabalho
GET	Buscar dados	Buscar todos os alimentos ou um alimento específico.
PUT	Atualizar dados	Atualizar informações de um alimento
POST	Cadastrar novos dados	Inserir um novo alimento no sistema
DELETE	Apagar dados	Apagar determinado alimento

Fonte: Autoria própria.

A próxima seção contém uma relação entre os sistemas existentes e o desenvolvido, assim como as suas funcionalidades.

2.3 Sistemas Similares

Atualmente, em 2022, existem sistemas que são capazes de criar dietas e consultar alimentos, porém foi observado que ainda não é comum que estes façam alguma menção ao grau de processamento dos alimentos. O quadro 3 contém as funcionalidades dos softwares existentes em comparação com o sistema desenvolvido.

Quadro 3 – Sistemas Similares

Nome do Software	Contagem de Calorias	Quantidade de Macronutrientes	Quantidade de Micronutrientes	Grau de Processamento
MyFitnessPal	Sim	Sim	Sim	Não
Webmd	Sim	Sim	Sim	Não
Calorie-control	Sim	Sim	Não	Não
Fatsecret	Sim	Sim	Sim	Não
Sistema Proposto	Sim	Sim	Sim	Sim

Fonte: Autoria própria (2022).

No próximo capítulo serão detalhadas as ferramentas utilizadas para o desenvolvimento do sistema.

3 FERRAMENTAS

Para o desenvolvimento do sistema, foi necessário que várias ferramentas fossem utilizadas para a organização e eficiência do código escrito. Os tópicos a seguir contém informações dos *frameworks* de desenvolvimento de código e linguagens de programação utilizadas.

As ferramentas e tecnologias utilizadas para desenvolver o sistema proposta estão descritas no Quadro:

Quadro 4 – Ferramentas utilizadas

Software	Licença	Versão	Propriedade
Java	Gratuito	11.0.16	Azul Zulu
Spring Boot	Gratuito	2.6.2	Pivotal Software
Lombok	Gratuito	1.18.6	Project Lombok
Typescript	Gratuito	4.8.4	Microsoft
React	Gratuito	17.0.2	Meta
H2	Gratuito	2.1.210	HSQLDB

Fonte: Autoria própria (2022).

3.1 JAVA

A linguagem Java continua atualmente, em 2022, sendo uma das linguagens de computação mais populares do mercado (SCHILDT, 2022). Mesmo sendo concebida em 1995, esta continuou sendo relevante pois recebeu diversas atualizações, ou versões, conforme o tempo.

A plataforma Java se destacou das demais, pois, conseguia entregar ao desenvolvedor todo o ambiente necessário para a codificação de aplicações. Vale ressaltar que também possui sua própria máquina virtual, conhecida como *Java Virtual Machine* ou JVM. A JVM permite que a mesma aplicação Java seja executada de maneira similar em diferentes sistemas operacionais e a existência deste mecanismo permitiu uma maior facilidade no desenvolvimento de software e tempos passados onde o gerenciamento de memória, por exemplo, era específico em cada sistema e precisa ser configurado pelo desenvolvedor da aplicação (SCHILDT, 2022).

O Java é gratuito para uso pessoal e o *donwload* de suas versões pode ser encontrado em sites de diferentes distribuidores.

O framework de desenvolvimento utilizado em conjunto com a linguagem Java na codificação do *backend* será detalhada na seção a seguir.

3.2 SPRINGBOOT

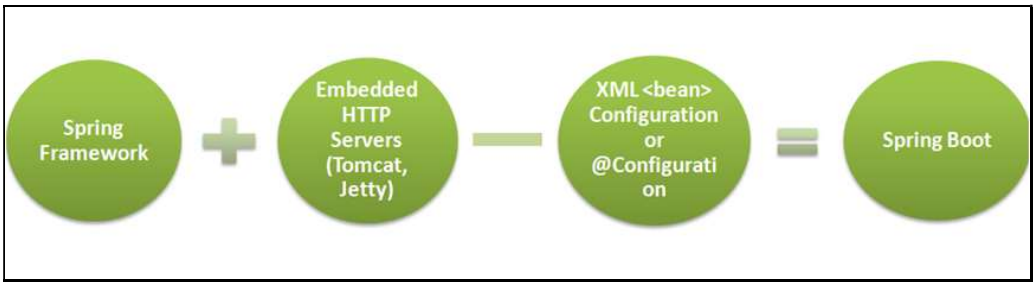
Ao tratar-se de desenvolvimento de software utilizando a linguagem Java, o *Spring Framework* se destaca pela quantidade de ferramentas e conteúdos disponíveis para aprendizado e consulta. Este Framework pode atender projetos menos complexos, como interfaces de programação (API) *Rest* básicas até *backend's* completos utilizando arquitetura de microserviços em nuvem (MYTHILY; SAMSON ARUN RAJ; THANAKUMAR JOSEPH, 2022).

Embora existam diversos módulos com diferentes funcionalidades no *Framework Spring*, como *Cloud*, *Serverless*, *Event Driven*, neste trabalho foram empregados os conceitos do *Spring Boot, Spring Security* e *Spring Data JPA*.

Sobre estes, o primeiro é descrito como uma aplicação básica previamente configurada para ser utilizada como base de qualquer projeto *Spring*. Para gerar uma aplicação *Spring Boot*, é necessário apenas informar na ferramenta online *Spring Initiliazr* qual será o tipo de projeto, entre *Maven* e *Gradle*, a linguagem que será utilizada no desenvolvimento e sua versão, a versão do Spring e uma lista básica de dependências a serem declaradas, que podem sofrer alterações posteriormente.

O *Spring Security* foi empregado para resolver questões de autenticação a autorizações, pois era necessário que o usuário pudesse entrar no sistema de alguma maneira e interagir com as funcionalidades conforme seu cargo, e para tanto, este módulo permite a criptografia de senhas e gestão de *tokens*. Para configurar os dados a serem persistidos, quais objetos e seus atributos, a busca, deleção, criação e atualização destes, fez-se necessário a utilização do módulo *Spring Data JPA*. A Figura 2 mostra que o *Spring Boot* nada mais é do que um resultado do *Spring Framework* com adições de configurações adicionais.

Figura 2 – Representação Visual Spring Boot



Fonte: https://gitagupta1611.medium.com/spring-boot-44cb004ae8c (2022).

Conforme citado anteriormente, o *Spring Framework* faz a utilização de diversas anotações para facilitar o desenvolvimento de aplicações web. Estas anotações são utilizadas ao inserir no código o símbolo @ seguido de alguma palavra. Um exemplo da utilização desta facilidade no projeto está presenta na Figura 3, onde as anotações @RestController e @RequestMapping são utilizadas.

Figura 3 – Exemplo de anotações Spring.

```
1   @RestController
2   @RequestMapping("/food")
3   public class FoodResource {
4
5       @Autowired
6       private FoodService foodService;
7
8       @GetMapping("/{id}")
9       public ResponseEntity<FoodDTO> findBy
10          return ResponseEntity.ok().body(f
```

Fonte: Autoria própria (2022).

A subseção a seguir contém uma biblioteca utilizada para facilitar o desenvolvimento que também utiliza estas anotações.

3.3 LOMBOK

Durante o desenvolvimento de uma aplicação utilizando os conceitos da orientação a objetos, certos trechos de código podem comumente se repetir. Um bom exemplo disto na linguagem Java são os métodos Construtores, *Getters*, *Setters* *Hashcode* e *Equals*. Para facilitar o desenvolvimento, Integrated Development Enviroment's (IDE) modernas já conseguem implementar estes métodos automaticamente com alguns cliques. A proposta do Lombok é facilitar ainda mais este processo.

Ao utilizar esta ferramenta, é possível que os métodos citados acima sejam automaticamente implementados, levando em consideração o nome dos atributos e seus tipos sobre a classe ao qual o Lombok foi adicionado. Para que isto seja possível,

é necessário apenas que o programador digite, acima da declaração da classe na maioria das vezes, uma anotação especial, denotada pelo símbolo @ seguido do nome do método a ser implementado, para cada tipo de método citado anteriormente. A Figura 4 apresenta a classe alimentos contida no *backend* desenvolvido e que faz uso das anotações do Lombok encontradas na linha 1 a 5.

Figura 4 – Exemplo de objeto utilizando Lombok

```
1  @AllArgsConstructor
2  @NoArgsConstructor
3  @Getter
4  @Setter
5  @EqualsAndHashCode
6  @Entity
7  @Table(name = "tb_food")
8  @Builder
9  public class Food implements Serializable {
10
11     private static final long serialVersionUID = 9178661439383356177L;
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15     private String name;
16     @Enumerated(EnumType.STRING)
17     private FoodGroup foodGroup;
18     private String imgUrl;
19     private Integer quantity;
20     private Double calorie;
21     private Double protein;
22     private Double carbohydrate;
23     private Double fat;
24     private Double sodium;
25     private Double sugar;
26     private Double vitaminA;
27     private Double vitaminC;
28     private Double iron;
29 }
```

Fonte: Autoria própria (2022).

A subseção a seguir contém detalhes da linguagem de programação utilizada no *frontend* do projeto.

3.4 TYPESCRIPT

A linguagem de programação Javascript é a mais popular escolha para o desenvolvimento de aplicativos *web*, *mobile*, *dekstop* e até mesmo em soluções *backend* (BOGNER; MERKEL, 2022). Isto ocorre devido a esta ser considerada uma linguagem dinâmica e flexível, contudo estas mesmas facilidades também resultam em um software com baixa qualidade (BOGNER; MERKEL, 2022). Uma das soluções para este problema é a utilização do TypeScript, que nada mais é do que um *superset*, uma extensão da linguagem, que possui tipagem segura.

Ao desenvolver código utilizando TypeScript, o programador pode fazer o uso de anotações e do compilador TypeScript, que transpõe o código para a

linguagem Javascript, para que este seja interpretado. A grande diferença nessa operação é que o TypeScript realiza a checagem de tipos durante o momento de transpilação, ao passo que o Javascript faz esta durante a própria execução. Este também é o motivo da linguagem JavaScript apresentar altos índices de código com baixa qualidade.

Ao declarar uma variável em JavaScript, esta pode ter seu tipo alterado conforme as próximas linhas de código a serem executadas. A Figura 5 apresenta exemplos onde uma variável é utilizada em situações adversas a seu tipo utilizando a linguagem JavaScript e mesmo assim, o código contido é executado.

Figura 5 – Exemplo de Código em Javascript

```
let umaFrase = "Isto é uma frase"; // Inicia uma variável STRING com o valor entre aspas duplas
umaFrase = 2236;                   // A variável umaFrase torna-se do tipo NUMBER

square(true) // Função utilizada para calcular o quadrado do número entre parentêsis
             // O resultado desta operação em JavaScript é 1, o que pode ser difícil de apurar
```

Fonte: Autoria própria (2022).

A linguagem TypeScript não permitiria as operações anteriores e lançaria erros, conforme observado na Figura 6.

Figura 6 – Exemplo de Código em TypeScript

```
let umaFrase: string = "Isto é uma frase"; // Inicia uma variável STRING com o valor entre aspas duplas
umaFrase = 2236;                           // O resultado desta operação é um erro

square(true); // Função utilizada para calcular o quadrado do número entre parentêsis
              // O resultado desta operação é um erro
```

Fonte: Autoria própria (2022).

O *framework* de desenvolvimento utilizado em conjunto a linguagem *Typescript* é apresentado na subseção a seguir.

3.5 REACT

O *framework* de desenvolvimento *React* é utilizado para implementar interfaces de usuário e aplicações web em geral de maneira ágil e eficiente, com uma quantidade de código pequena em comparação à utilização de *JavaScript* puro.

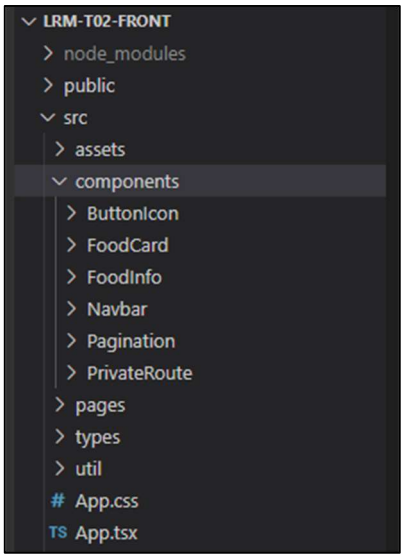
Para desenvolver em *React*, é necessário entender o seu mecanismo mais importante, os componentes. Os componentes podem ser descritos como trechos independentes de código que podem ser reutilizados em diversos locais. Ao utilizá-los, o programador garante um baixo acoplamento do conteúdo renderizado no

navegador ou aplicativo e reduz significativamente a complexidade de manutenção do código.

A principal responsabilidade do *React* é tratar de tudo aquilo que será visualizado pelo usuário final na aplicação, ou seja, o *frontend*. Para isto, este framework permite a implementação de aplicações de página única, onde um único arquivo HTML será apresentado incialmente e seu próprio conteúdo será alterado conforme a navegação posterior do usuário.

A Figura 7 apresenta a disposição das pastas utilizadas no *frontend* do sistema desenvolvido no trabalho. Os componentes criados, contidos na pasta componentes, podem ser declarados em diferentes locais da aplicação e serem reaproveitados. Um exemplo desta característica é o componente NavBar, pois este único componente está presente em todas as outras telas do sistema, localizado na parte superior do site.

Figura 7 – Disposição dos componentes no *frontend*



Fonte: Autoria própria.

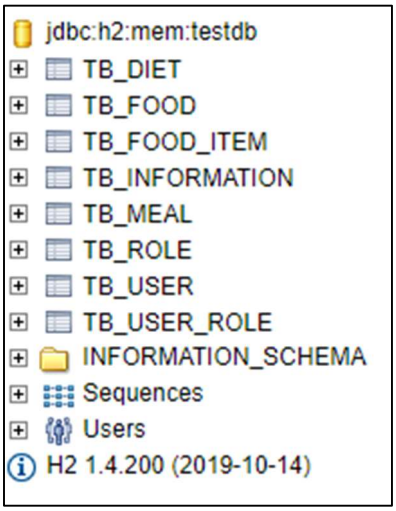
A subseção a seguir apresenta a funcionalidade utilizada para observar os dados a serem persistidos.

3.6 H2 DATABASE

Durante a implementação das entidades previstas no sistema, foi necessário verificar como os dados eram de fato persistidos no banco de dados. Uma maneira ágil de visualizar estes dados com poucas configurações é por meio da utilização do banco de dados em memória H2.

Ao desenvolver um *backend* utilizando *Springboot*, é comum que o projeto contenha a dependência com o banco de dados H2 de maneira nativa. Com este, podemos visualizar as tabelas, colunas e linhas atuais no sistema, assim como o conteúdo de cada um destes elementos. A Figura 8 contém o nome de todas as tabelas presentes no sistema.

Figura 8 – Tabelas existentes no sistema



Fonte: Autoria própria.

A Figura 9 apresenta a visualização dos dados persistidos da tabela alimentos durante a utilização do H2 no desenvolvimento da aplicação.

Figura 9 – Conteúdo da tabela de alimentos

ID	CALORIE	CARBOHYDRATE	FAT	FOOD_GROUP	IMG_URL
1	98.3	26.0	0.0	INNATURA	https://img.freepik.com/fotos-gratis/bando-de-banana-isolado_88281
2	53.0	15.2	0.0	INNATURA	https://media.soujusto.com.br/__sized__/products/207263-thumbnail
3	108.0	0.0	12.0	INGREDIENTS	https://http2.mlstatic.com/D_NQ_NP_731025-MLU47590779855_09
4	0.0	0.0	0.0	INGREDIENTS	https://www.ingredientesonline.com.br/media/catalog/product/cache/
5	191.0	0.0	10.53	PROCESSED	https://static.paodeacucar.com/img/uploads/1/289/19919289.jpg
6	53.0	0.0	3.5	PROCESSED	https://media.soujusto.com.br/products/Creme_De_Queijo_Ricota_L
7	201.0	24.4	10.72	ULTRAPROCESSED	https://a-static.mlcdn.com.br/800x560/pote-de-sorvete-eskimo-grand sorvetes/eskimosorvetes/6504903cb42d11eb82584201ac18500e/e9
8	500.0	55.0	25.0	ULTRAPROCESSED	https://m.media-amazon.com/images/I/610trEtCQuS._AC_SX425_.jp

Fonte: Autoria própria.

Descritas as ferramentas utilizadas para desenvolver todo o código do trabalho, a subseção a detalha o desenvolvimento da aplicação em si e os testes realizados para a validação.

4 DESENVOLVIMENTO, TESTES E METODOLOGIA.

Neste capítulo serão descritos os padrões e as práticas utilizadas na codificação do sistema e a análise de requisitos do software desenvolvido, contendo a descrição de cada requisito levantado e os diagramas desenvolvidos a partir destes, assim como a apresentação dos testes realizados no sistema e a metodologia de desenvolvimento utilizada.

4.1 Padrões de Projeto

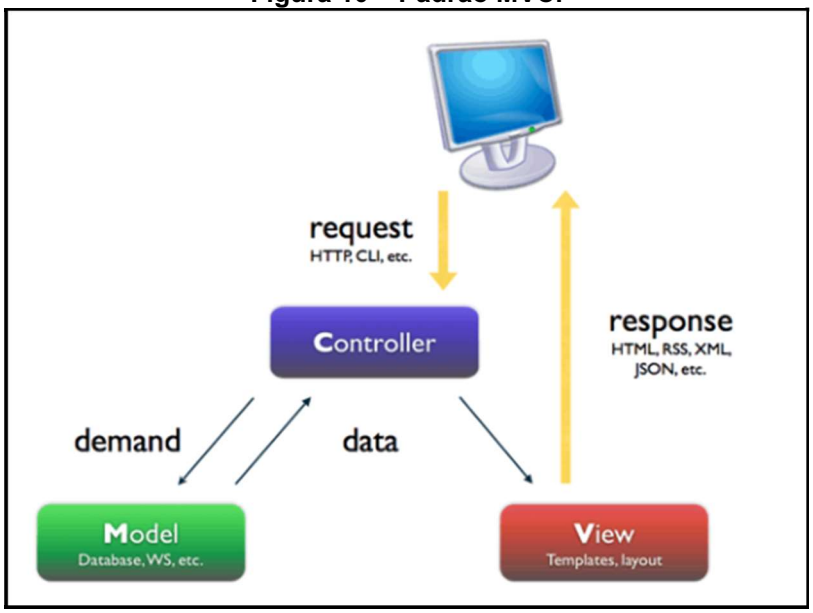
A escrita de código pode ser realizada de diversas maneiras. Para determinadas situações, um padrão pode ser mais ou menos aconselhável do que outro, levando em consideração o tamanho do projeto, da equipe, assim como outros fatores relevantes. Os padrões de projeto existem para facilitar o desenvolvimento, pois por meio destes, é possível ler e prever comportamentos específicos para cada trecho de código desenvolvido. Vale ressaltar que a leitura e manutenção de código por terceiros, aquele que não propriamente escreveu-o, torna-se mais eficiente a mérito da fidelidade utilizada em seguir tais padrões.

As subseções a seguir irão explicar e exemplificar os padrões utilizados na escrita do código da aplicação desenvolvida.

4.1.1 Desenvolvimento em Camadas

Um dos padrões mais utilizados para desenvolvimento *web* é o *Model View Controller*, conhecido como MVC. Este padrão de arquitetura de software permite tratar o software em camadas e a partir disso, alcançar uma melhor segurança, organização e eficiência do código. O padrão MVC baseia-se em três camadas iniciais, Modelo, Visão e Controlador, porém existe a possibilidade da inserção de novas camadas conforme a necessidade de cada sistema (POP; ALTAR, 2014). A Figura 11, vide próxima página, apresenta a interação básica entre as três camadas.

Figura 10 – Padrão MVC.



Fonte: http://stackoverflow.com/questions/5966905/which-mvc-diagram-is-correct-web-app (2022).

As camadas básicas possuem as seguintes responsabilidades:

a) Modelo: Conter como e quais dados serão armazenados no sistema. Deve receber dados dos Controladores e verificar se estes estão corretos e/ou atendem as regras os requisitos do sistema. A camada de modelo é desenvolvida levando em consideração as regras de negócio, assim como validação e autenticação.

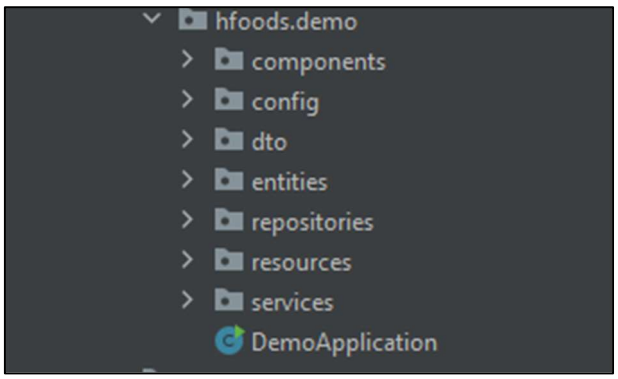
b) Visão: Definir como a aparência do sistema será apresentada ao usuário, onde isto compreende telas, botões, formulários, assim como qualquer elemento que caracterize uma interface gráfica que interaja com o usuário do sistema. É responsável por enviar requisições aos Controladores e possivelmente receber respostas. Esta camada é fortemente relacionada as linguagens de computação utilizadas para desenvolver um *frontend*, sendo estas Javascript, HTML e CSS.

c) Controlador: Intermediar as interações entre a View e a Model. Este intermédio permite que as requisições do usuário sejam convertidas e interpretadas pelo Modelo, assim como a resposta seja resgatada em determinadas situações. Esta camada é acionada por meio dos parâmetros HTTP e URL'S específicas configuradas previamente.

O *framework Spring*, utilizado na codificação do *backend* da aplicação desenvolvida, implementa este padrão por meio do *Spring* MVC. Essa *implementação*

sugere a inclusão de novas camada para melhor atender as necessidades de uma API escrita na linguagem Java. Portanto, ao invés de 3 camadas, o sistema possui 4, sendo estas: Controlador, Serviço, Entidades e Repositórios. Estas camadas podem ser visualizadas por meio da disposição dos pacotes no *backend*, conforme a Figura 12, onde a camada Controlador possui o nome de Resources.

Figura 11 – Disposição dos pacotes no *backend*.



Fonte: Autoria própria (2022).

A camada de Controlador, no inglês *Controller* ou *Resources*, é responsável por receber as requisições enviadas do *frontend* e responder estas. Estas requisições podem conter diferentes endereços, também conhecidos como Uniform Resource Locator ou URL's, parâmetros, corpos e cabeçalhos HTTP.

Na camada de Serviço, no inglês Service, encontramos as regras de negócio do sistema, o lançamento de exceções por meio da validação dos dados, além de intermediar a camada de Controlador e Repositório.

A camada de Entidade define os objetos que serão persistidos no sistema e posteriormente convertidos em tabelas. Estes objetos, por sua vez, possuem atributos e cada um destes possui um tipo específico.

Por fim, a camada de Repositório irá tratar da conversão de objetos Java e a busca destes no domínio SQL, utilizando diferentes *Statements*.

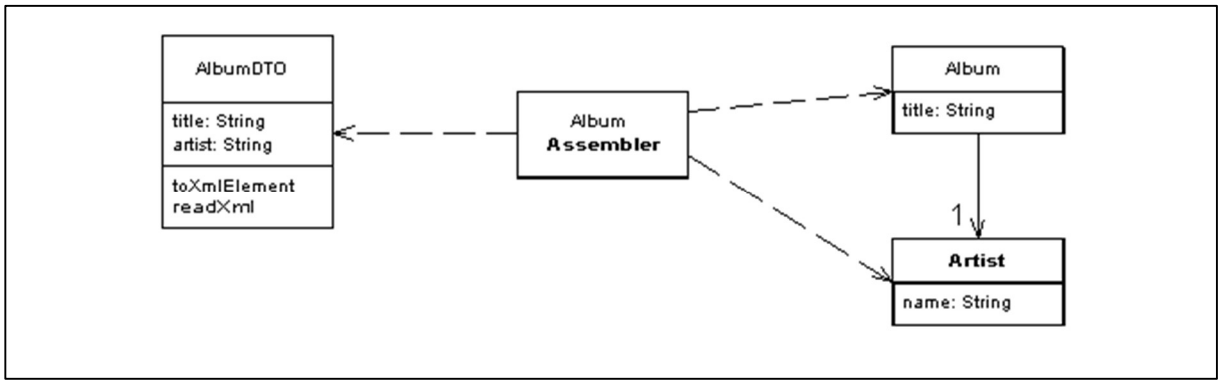
A subseção a seguir contém detalhes da funcionalidade utilizada para otimizar a transferência de informações entre o *frontend* e *backend* do sistema desenvolvido.

4.1.2 *Data Transfer Objects*

Para que a interação entre o *backend* e o *frontend* seja possível, é necessário que se utilize os métodos HTTP, URL's específicas, parâmetro e possíveis

corpos com dados referentes à requisição ou resposta, este último normalmente no formato JSON. Os data transfer objects, comumente chamados de DTO'S, são utilizados para otimizar esta comunicação, pois, por meio deste tipo de dados pode-se decidir quais informações são esperadas para uma requisição e filtrar dados em uma resposta (FOWLER, 2015). Dados sensíveis, como o Cadastro de Pessoa Física (CPF) ou endereço, por exemplo, podem ser removidos dos objetos alvo para que a resposta não os contenha, aumentando assim a segurança do software. Outro problema que pode ser resolvido com a utilização dos DTO'S é a junção de informação contida em entidades distintas. A Figura 12 apresenta a utilização de um DTO que une as informações de um Álbum e Artista, assim evitando que duas requisições sejam feitas no sistema para obter os dados desejados (FOWLER, 2015). As requisições e respostas realizadas no *backend* do sistema desenvolvido foram implementadas utilizandos os DTO'S.

Figura 12 – Exemplo de utilização de DTO.



Fonte: https://martinfowler.com/eaaCatalog/dataTransferObject.html (2022).

A subseção a seguir contém um conceito importante utilizado para a autenticação dos usuários no sistema desenvolvido.

4.1.3 JSON Web Token

O processo autenticação em sistemas que utilizam o conceito de *token* fazem o uso de artefatos HTTP específicos, como os cabeçalhos de autorização e parâmetros de uma requisição. Cabe ao JSON Web Token, conhecido por JWT, realizar o processo de decodificação de um *token* existente para assim conseguir autenticar uma requisição (ADAM; MOEDJAHEDY; MARAMIS, 2020). Em um serviço web, é comum que os *Token's* sejam gerados no *backend* da aplicação e sejam enviados ao *frontend* após o usuário realizar o login com as credenciais corretas. Por meio da decodificação do *Token*, é possível verificar qual é seu usuário pertencente e as permissões que este usuário possui no sistema.

Um *Token* JWT, token utilizado no sistema desenvolvido, possui a seguinte estrutura:

a) *Header*: O cabeçalho do token contém o seu tipo e o algoritmo de hash utilizado para a criptografia deste. A Figura 13 contém um exemplo de header utilizado para configurar um token.

Figura 13– Exemplo cabeçalho JWT.

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Fonte: https://jwt.io/introduction (2022).

b) *Payload*: O conteúdo do token, como informações sobre o usuário pertencente e permissões é encontrado no *payload*. A Figura 14 apresenta um exemplo de informações contidas na carga de um token.

Figura 14 – Exemplo conteúdo JWT.

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Fonte: https://jwt.io/introduction (2022).

c) *Signature*: A assinatura digital permite que o *token* não possa ser de criptografado por usuários que não tenham conhecimento das regras utilizadas para sua geração, assim como validar se o conteúdo do *token* não foi violado durante seu

transporte. A Figura 15, expõe um exemplo de assinatura de *token* utilizando um segredo específico.

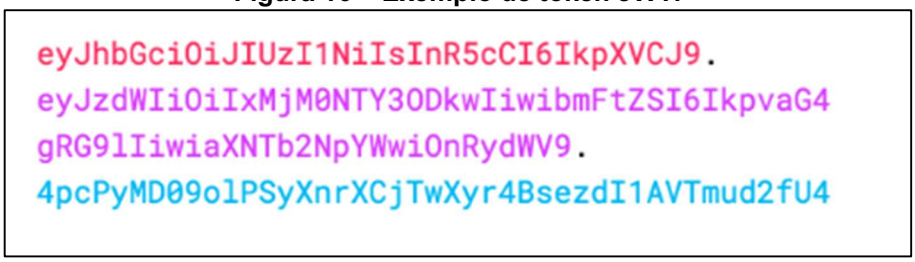
Figura 15 – Exemplo assinatura JWT.

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

Fonte: https://jwt.io/introduction (2022).

O *token JWT* gerado ao utilizar como estrutura o conteúdo das Figuras 14, 15 e 16, pode ser encontrado na Figura 16. O trecho em vermelho corresponde ao cabeçalho, o lilás ao conteúdo e a assinatura possui a cor azul.

Figura 16 – Exemplo de token JWT.



Fonte: https://jwt.io/introduction (2022).

4.2 Análise de Requisitos

Esta seção apresenta os requisitos levantados durante a etapa inicial do desenvolvimento do software, e os diagramas criados para facilitar o desenvolvimento da aplicação.

4.2.1 Requisitos Não Funcionais e Funcionais

Os requisitos não funcionais do sistema, obtidos por meio da análise dos objetivos do projeto estão presentes no quadro 5.

Quadro 5 – Requisitos não funcionais

ID	DESCRIÇÃO
RFN01	O *backend* deverá ser desenvolvido utilizando a linguagem Java em conjunto do framework Spring.
RFN02	O *frontend* deverá ser desenvolvido utilizando as linguagens *Typescript*, HTML e CSS em conjunto do *framework React*.
RFN03	O site desenvolvido deve possuir responsividade às telas de celulares, de tablets e de computadores.
RFN04	Utilizar arquivos JSON para a comunicação entre o *backend* e *frontend*.
RFN05	Utilizar como variável de caminho o nome da entidade e os verbos HTTP para o acesso aos recursos do *backend*.

Fonte: Autoria própria (2022).

Para melhor descrever os objetivos funcionais, descritos no quadro 6, foram utilizados os métodos HTTP e os tipos de usuário que podem acessar o sistema, visto que estes possuem permissões diferentes para cada funcionalidade. Vale ressaltar que várias destas são resultado das ações de Create, Register, Update e Delete das entidades, e estas ações podem ser descritas pelo termo CRUD.

Quadro 6 – Requisitos Funcionais

ID	DESCRIÇÃO	MÉTODOS HTTP	USUÁRIOS
RF01	Deverá possuir recursos para o CRUD de alimentos.	GET, POST, PUT DELETE.	Administrador e nutricionista.
RF02	Deverá possuir um recurso que retornar todos os alimentos sem necessidade de autenticação.	GET.	Todos.
RF03	Deverá possuir recursos para o CRUD de refeições.	GET, POST, PUT E DELETE.	Administrador e nutricionista.
RF04	Deverá possuir recursos para o CRUD de dietas.	GET, POST, PUT E DELETE.	Administrador e nutricionista.
RF05	Deverá possuir um recurso que retorne a dieta do usuário autenticado.	GET.	Cliente, nutricionista e administrador.
RF06	Deverá possuir recursos para o CRUD de informações corporais.	GET, POST, PUT E DELETE.	Administrador e nutricionista.
RF07	Deverá possuir um recurso que retorne as informações corporais do usuário autenticado.	GET.	Cliente, nutricionista e administrador.
RF08	Deverá possuir recursos para o CRUD de usuários.	GET, POST, PUT E DELETE.	Administrador.
RF09	Deverá possuir um recurso para realizar login com a utilização de *token*.	POST.	Cliente, nutricionista e administrador.

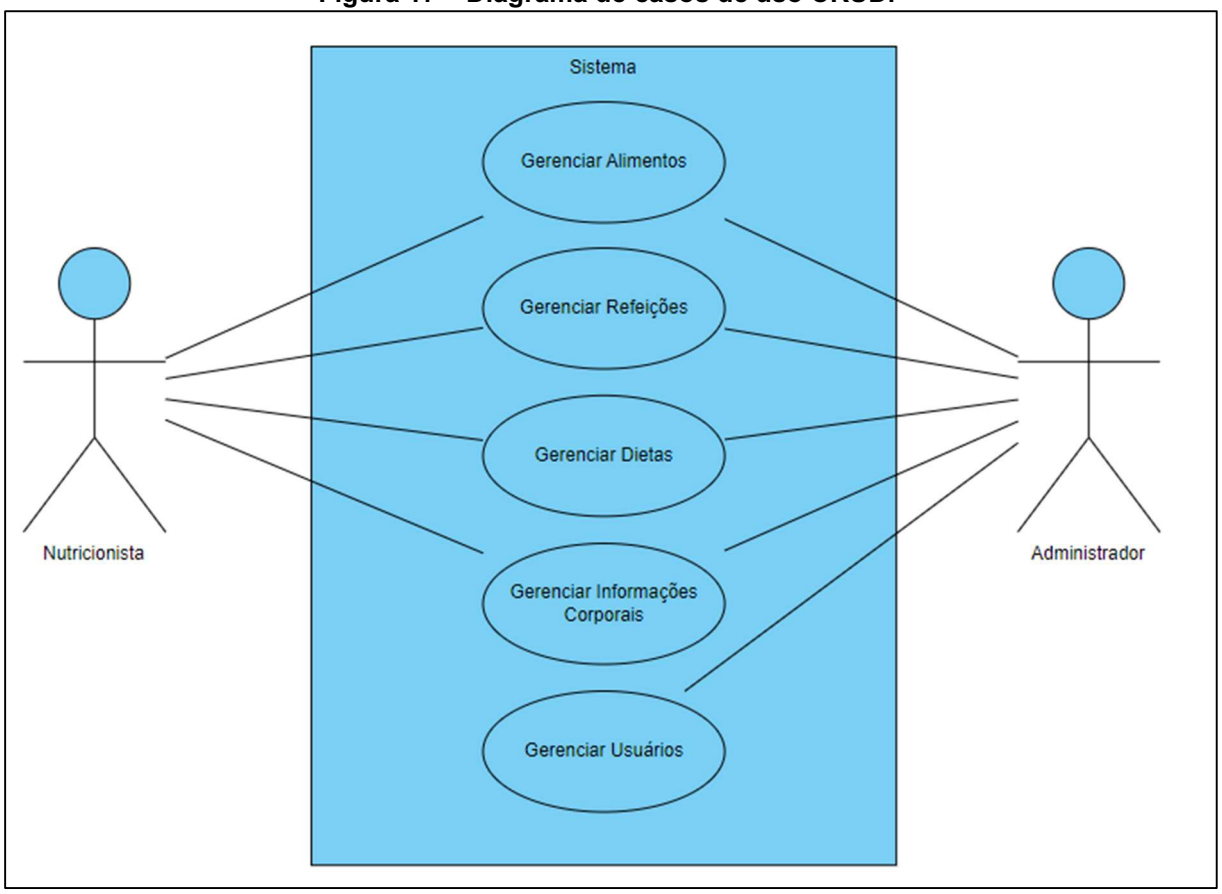
Fonte: Autoria própria (2022).

4.2.3 DIAGRAMA DE CASOS DE USO E ENTIDADE-RELACIONAMENTO

Devido a presença de diferentes tipos de usuário no sistema desenvolvido, o diagrama de casos de uso foi divido em diferentes atores.

O diagrama de casos de uso dos Usuários que ocupam os cargos de Nutricionista e Administrador é apresentado por meio da Figura 17. O nutricionista será o profissional responsável por manipular todos os dados referentes à dieta do usuário e os dados corporais deste usuário, enquanto o administrador poderá exercer todas as funcionalidades presentes no sistema. Pode se observar que ao gerenciar os alimentos, os requisitos funcionais RF01, RF02, RF03, RF04 e RF05 serão atendidos mediante o login do ator no sistema.

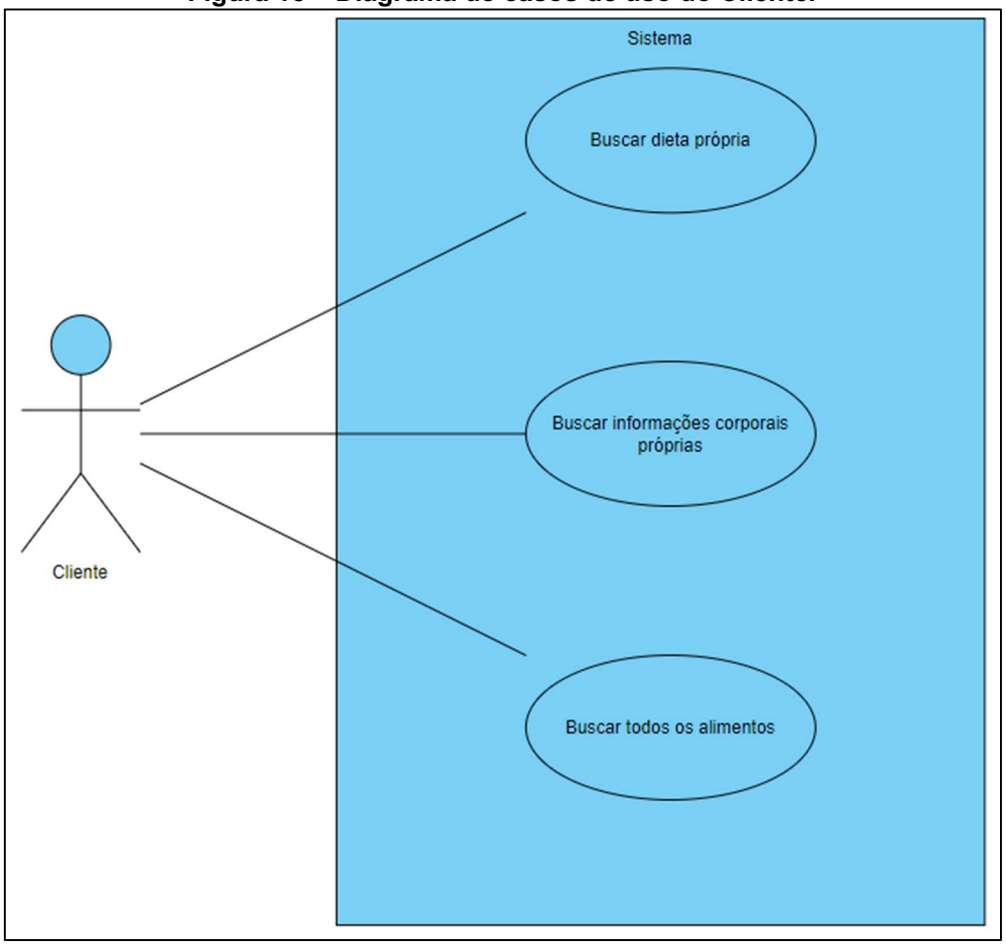
Figura 17 – Diagrama de casos de uso CRUD.



Fonte: Autoria própria (2022).

O diagrama de casos de uso do ator Cliente, onde este pode ser descrito como o paciente do nutricionista, está contido na Figura 18. Este possui funcionalidades limitadas em comparação aos outros tipos de usuário.

Figura 18 – Diagrama de casos de uso do Cliente.

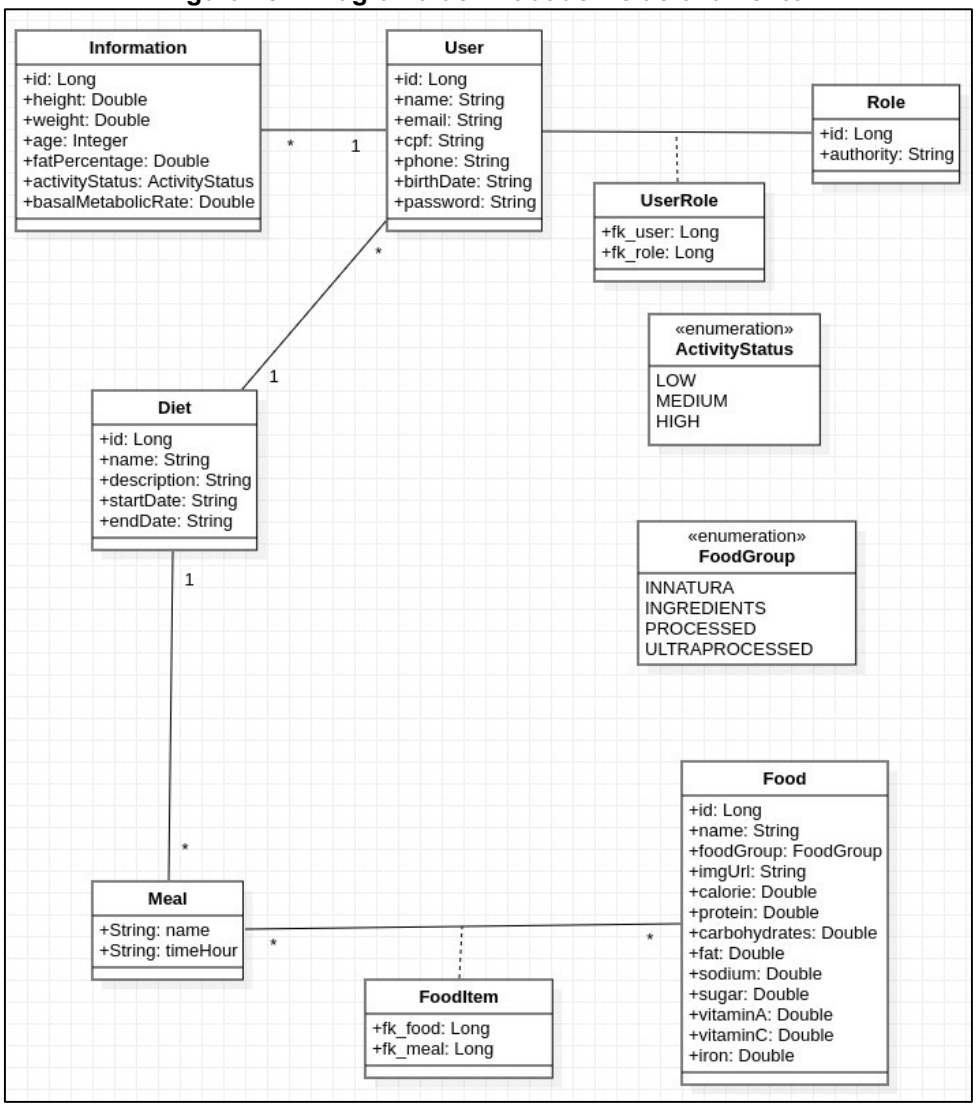


Fonte: Autoria própria (2022).

Cada caso de uso faz referência a algum dos requisitos funcionais descritos anteriormente, onde estes serão responsáveis por cadastrar, editar e remover dados pertinentes à alguma das entidades contidas no sistema.

A Figura 19 apresenta o diagrama de Entidade-Relacionamento o qual contém os relacionamentos entre as entidades usuário, cargos, informações corporais, dietas, refeições e alimentos. Vale ressaltar que as tabelas no sistema possuem os mesmos nomes do modelo conceitual abaixo, porém com a adição do termo “tb_” antes no nome de cada uma delas, para assim evitar possíveis conflitos com as palavras reservadas da linguagem SQL.

Figura 19 – Diagrama de Entidade-Relacionamento.



Fonte: Autoria própria (2022).

Na próxima seção será apresentados os testes realizados no sistema.

4.3 Testes

Para validar as funcionalidades e regras de negócio implementadas no sistema, foram utilizados testes unitários. Os testes unitários garantem a integridade de métodos específicos sem levar em consideração o comportamento daquilo que está fora do escopo do método a ser testado. Como o sistema utilizou um padrão de camadas para ser desenvolvido, os testes unitários por sua vez foram implementados tendo como referência os métodos de cada camada do sistema.

Tratando-se de programação na linguagem Java, a biblioteca de testes mais comum a ser utilizada é a JUNIT. Esta permite que os métodos Java sejam testados utilizando testes de integração ou unitários. Os testes de integração simulam a execução real de uma funcionalidade do sistema, porém é mais lento, enquanto o teste unitário faz uso da simulação de certas funcionalidades e é expressamente mais rápido. Como a aplicação em camadas possui um baixo acoplamento das classes e métodos, foi preferível que os testes implementados fossem unitários.

O nome de cada teste implementado é significativo, pois contém o método a ser testado, o resultado esperado e a clausula que denota quando este irá ocorrer. Temos um exemplo de testa na Figura 20, um teste da camada de Controller da entidade Food. O método a ser testado é o findById, e este deve retornar o código HTTP 404 quando não encontrar determinado alimento ao receber como parâmetro um identificador. Vale ressaltar que é necessário simular o comportamento da camada de Service, assim como fornecer um email e senha para a autenticação.

Figura 20 – Exemplo de teste unitário

```
1   @Test
2       @SneakyThrows
3       public void findById_deveRetornarResourceNotFound_quandoNaoEncontrado() {
4           when(service.findById(1L)).thenThrow(ResourceNotFoundException.class);
5
6           var accessToken = tokenUtil.obtainAccessToken(mockMvc, "john@gmail.com", "123456");
7
8           mockMvc.perform(get("/food/{id}", 1L)
9                           .header("Authorization", "Bearer " + accessToken)
10                          .accept(MediaType.APPLICATION_JSON))
11                  .andExpect(status().isNotFound());
12      }
```

Fonte: Autoria própria (2022).

Descritos os detalhes do desenvolvimento do sistema e seus testes, a subseção a seguir apresenta a metodologia utilizada e o cronograma das atividades.

4.4 Metodologia

Para o desenvolvimento de um software com qualidade, é necessário que alguma metodologia seja seguida. Isto ocorre devido a diversos fatores, que podem ser inclusive alheios a própria codificação em si, e estes podem acarretar atrasos ou significativos avanços na velocidade de execução de um projeto de software.

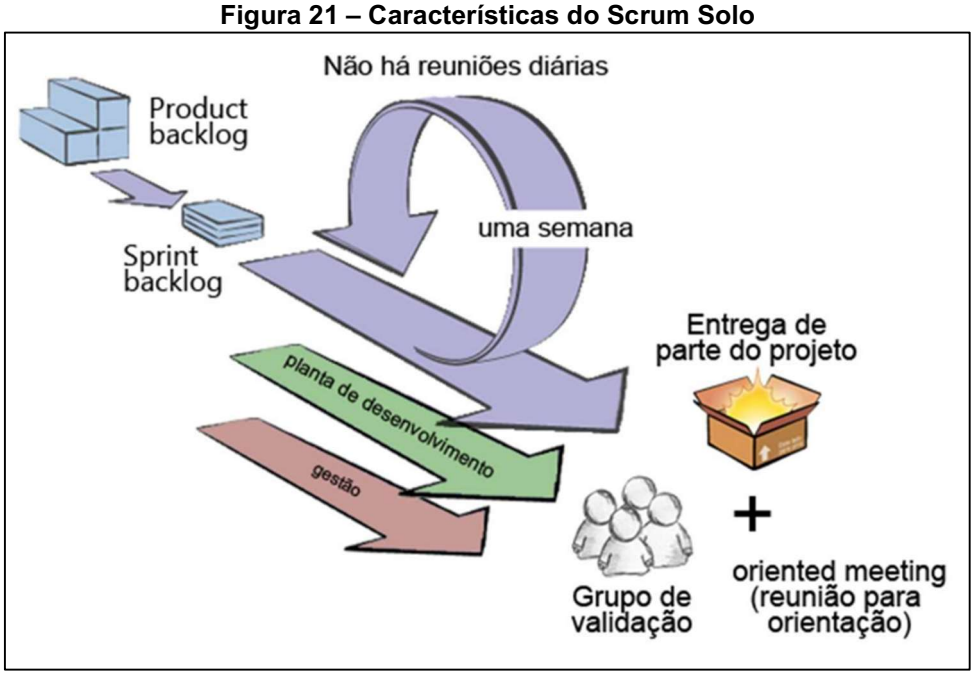
As subseções a seguir apresentam a metodologia de desenvolvimento utilizada e o cronograma final do projeto.

4.4.1 Scrum Solo

A metodologia de desenvolvimento utilizada no projeto possui o nome de Scrum Solo. Esta metodologia utiliza os conceitos do Scrum convencional e do Personal Software Process (PSP), podendo ser descrita como uma adaptação da união destes para uma equipe de apenas um desenvolvedor (PAGOTTO et al., 2016).

A escolha desta metodologia levou em consideração que o Scrum Convencional possui artefatos que não conseguem atender com eficiência equipes que possuem poucos indivíduos. Um grande exemplo disto é a presença de reuniões diárias, onde caso fossem empregadas no desenvolvimento deste trabalho, seriam certamente pouco produtivas. A Figura 21 mostra as principais características do Scrum Solo.

Figura 21 – Características do Scrum Solo



Fonte: Pagotto et al (2016).

Por meio das adaptações presentes no Scrum Solo, esta metodologia consegue reduzir o tempo gasto no processo de desenvolvimento de software e manter a qualidade daquilo que é entregue (PAGOTTO *et al*., 2016).

Esta metodologia prevê a geração de diferentes documentos. Parte destes foram desenvolvidos durante a disciplina de Trabalho de Conclusão de Curso 01 e outras foram adaptadas durante o desenvolvimento do projeto em si. O *product backlog* previsto incialmente, por exemplo, precisou ser alterado devido a alterações nas funcionalidades do sistema assim como a percepção de possíveis melhorias das divisões das *sprint's* e atividades a serem realizadas pelo autor durante o desenvolvimento do projeto.

O *Product Backlog* do trabalho desenvolvido pode ser visto em duas partes, uma para o *backend*, compreendo 16 atividades, e uma para o *frontend*, contendo outras 24. O Quadro 7 apresenta as primeiras 10 atividades do projeto e o *Product Backlog* completo por ser encontrado no apêndice A deste trabalho.

Quadro 7 – Primeiras atividades Product Backlog

ID	DESCRIÇÃO	DATA DA INSERÇÃO	DATA DE SELEÇÃO PARA A SPRINT
1	Entidade User e Role	23/01/2022	31/01/2022
2	Entidade Information	23/01/2022	31/01/2022
3	Entidade Diet, Meal e Food	23/01/2022	07/02/2022
4	Refatoração de código	13/02/2022	14/02/2022
5	Seed de dados	13/02/2022	07/03/2022
6	Camada de Repository	13/02/2022	07/03/2022
7	Camada de Serviço da entidade User	13/02/2022	14/03/2022
8	CRUD da entidade Food	13/03/2022	28/03/2022
9	Autenticação das rotas da entidade Food	13/03/2022	28/03/2022
10	CRUD das entidades Meal e Diet	13/03/2022	04/04/2022

Fonte: Autoria própria (2022).

A subseção a seguir contém o cronograma utilizado para o desenvolvimento das *sprint's* do projeto.

4.4.2 Cronograma

Por meio da utilização das atividades descritas no *Product Backlog*, foi possível que o cronograma das *Sprint's* do projeto fosse fracionado utilizando uma quantidade específica de tarefas a cada semana.

Vale ressaltar que o cronograma inicial proposto durante e fase de projeto do sistema foi fortemente modificado devido a percepção do autor da dificuldade de realizar certas implementações no sistema. Algumas atividades, por exemplo, que foram propostas a serem realizadas em uma semana acabaram tomando mais tempo, devido a necessidade de estudo ou dificuldades na programação daquele componente em questão. Por fim, o planejamento de um software se torna cada vez mais assertivo conforme a experiência de quem irá descrever as atividades a serem realizadas e do programador em questão.

O Quadro 5, vide próxima página, apresenta a quantidade de Sprint's necessárias para o desenvolvimento e quais atividades foram realizadas em cada uma destas.

Quadro 8 – Sprint's realizadas.

Sprint	Descrição	Tarefas
1	Início do projeto e criação das entidades User, Role e Information	1-2
2	Criação das entidades Diet, Meal e Food	3
3	Refatoração de código e solução de erros	4
4	Seed de dados e implementação da camada de repository	5-6
5	Validação e autenticação de usuário	7
6	Funcionalidades da entidade Food	8-9
7	Funcionalidades das entidades Meal e Diet	10-11
8	Funcionalidades das entidades Information e User	12-13
9	Refatoração de código e solução de erros	14
10	Funcionalidades para usuário logado	15-16
11	Criação do projeto e Navbar	17-18
12	Implementação da página inicial	19
13	Component Food e botões	20-21
14	Navbar da página de administração	22-23
15	Utilização de loader nos components	24
16	Refatoração de código e solução de erros	25
17	Tela de login	26
18	Estilização da tela de login	27
19	Validações de Login	28
20	Requisições com usuário logado	29
21	Utilização de token nas requisições	30
22-23	Página CRUD de alimentos	31-32
24-25	Página CRUD de refeições	33-34
26-27	Página CRUD de dietas e informações corporais	35-36
28	Página CRUD de usuários	37-38
29-30	Página de informação corporal e dieta própria	39-40

Fonte: Autoria própria.

Utilizando o *framework* de desenvolvimento descrito anteriormente, o sistema implementado através deste será apresentado no capítulo a seguir.

5 SISTEMA DESENVOLVIDO

Neste capítulo será apresentado o sistema desenvolvido e suas funcionalidades.

5.1 Navegação

O usuário poderá interagir com o sistema utilizando um navegador no computador ou celular, visto que as telas produzidas possuem responsividade que atendem *displays* de diferentes tamanhos. A maneira selecionada para que este usuário seja direcionado para outras telas no sistema foi o uso de um menu de navegação. O menu de navegação está presente na parte superior de todas as telas desenvolvidas e ao clicar em alguma de suas opções, o usuário será redirecionado para a tela desejada. As figuras 22 e 23, possuem respectivamente a página inicial e o menu de navegação no formato para computadores e aparelhos móveis.

Para melhor entendimento, a lista abaixo possui a descrição de cada tela do sistema.

Landing Page: Sendo a página inicial do sistema, esta pode ser requisitado por qualquer usuário estando este autenticado no sistema ou não. Possui uma mensagem motivacional e um Link para a tela de listagem de alimentos.

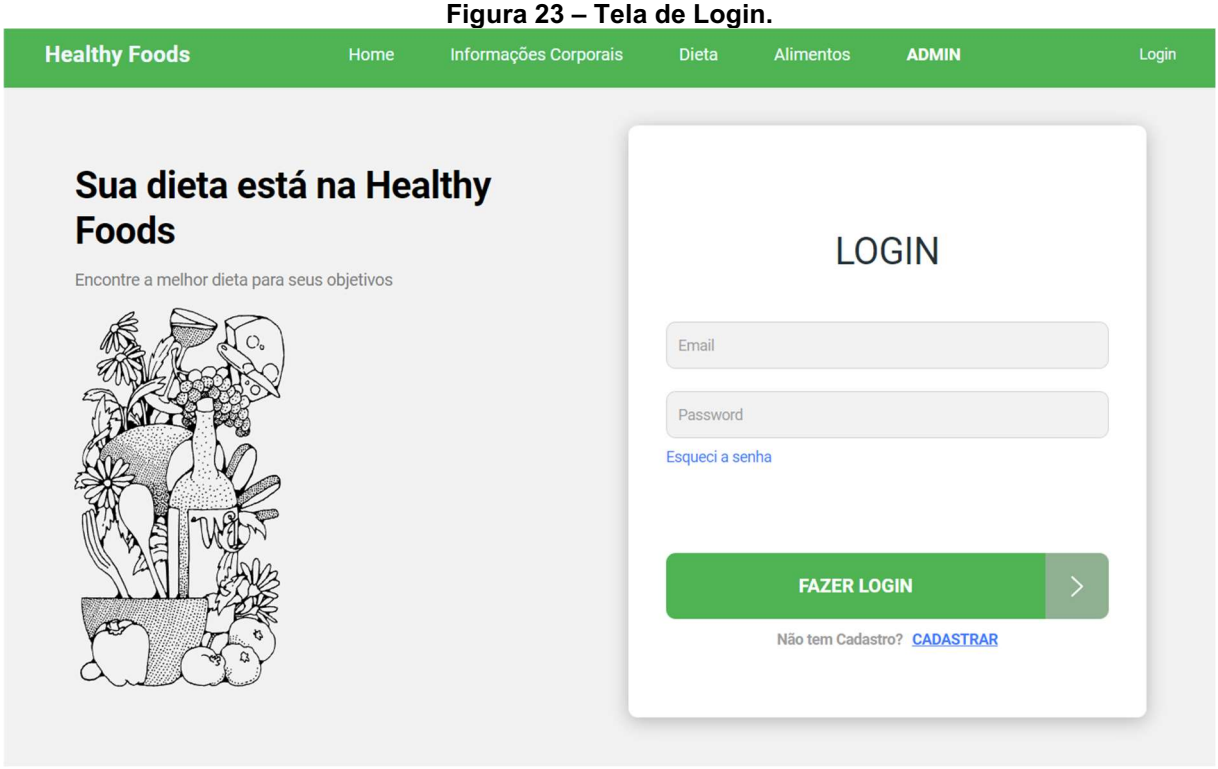
Figura 22 – Landing Page e menu de navegação desktop.

Fonte: Autoria própria (2022).

Listagem de Alimentos: a página de listagem de alimentos também não exige autenticação e pode ser acessada por qualquer usuário. Esta trará informações detalhadas sobre os nutrientes de cada alimento e o grau de processamento deste.

Login: no caso de o usuário tentar acessar um recurso protegido ou clicar no menu de login, este será redirecionado para a página de login. Nesta é possível que o usuário realiza a autenticação no sistema e receba em seu *local storage* do navegador suas credenciais.

Figura 23 – Tela de Login.



Fonte: Autoria própria (2022).

CRUD'S: As telas de CRUD são protegidas e necessitam de autenticação para que sejam acessadas. Além da autenticação, é necessário que o usuário possua o cargo de nutricionista ou administrador para que este veja estas telas. Caso estas especificações não sejam atendidas, o usuário é redirecionado à *Landing Page* ou login ao tentar acessar estes recursos.

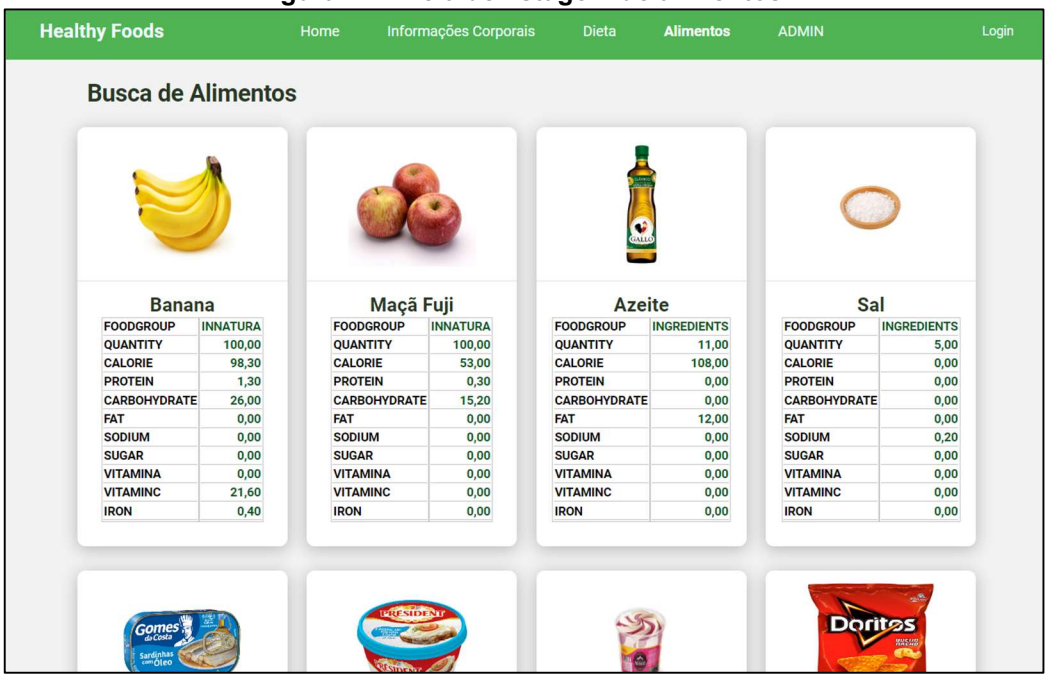
5.2 FUNCIONALIDADES

Este tópico detalhará as especificações utilizadas para atender cada uma das funcionalidades do sistema.

5.2.1 Tela De Listagem Dos Alimentos

A Tela de Listagem dos alimentos pode ser acessada por qualquer usuário, mesmo aqueles que não estão autenticados no sistema. Durante a renderização desta tela, contida na Figura 24, o *frontend* da aplicação faz uma requisição ao *backend* para que este retorne um dado do tipo Page que contém todos os alimentos cadastrados no sistema. Esta requisição não contém parâmetros ou conteúdos de corpo JSON, sendo necessário para sua utilização apenas a URL e a utilização do método HTTP get.

Figura 24 – Tela de listagem de alimentos.



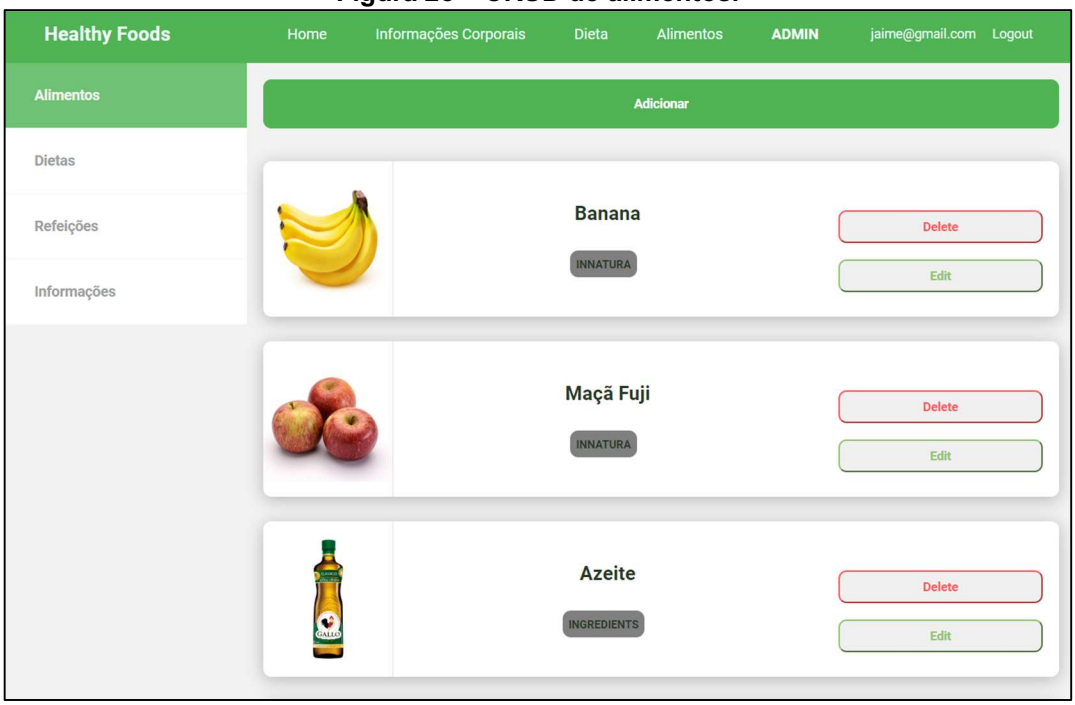
Fonte: Autoria própria (2022).

5.2.2 Telas De CRUD

As telas de CRUD são responsáveis pelo cadastro, edição e deleção das entidades do sistema. O funcionamento de cada uma destas é semelhante, possuindo apenas diferenças quanto as informações informadas nas requisições e seus retornos. Os dados necessários para cadastrar cada uma das entidades difere entre si, porém os métodos de validação de usuário autenticado e seu cargo são os mesmos.

Para renderizar os componentes contendo as entidades existentes no sistema, o mesmo mecanismo utilizado para listar todos os alimentos é utilizado. A diferença desta tela, porém, é que o *frontend* é capaz de, por meio do elemento identificador da entidade, interagir com o *backend* para que as informações da entidade que contém este identificador sejam alteradas ou deletadas. Vale ressaltar que o método de adicionar uma nova entidade não recebe um identificador do *frontend*, pois o *backend* é capaz de gerar este utilizando uma sequência pré-programada. A tela que permite a implementação do CRUD de alimentos está contida na Figura 25.

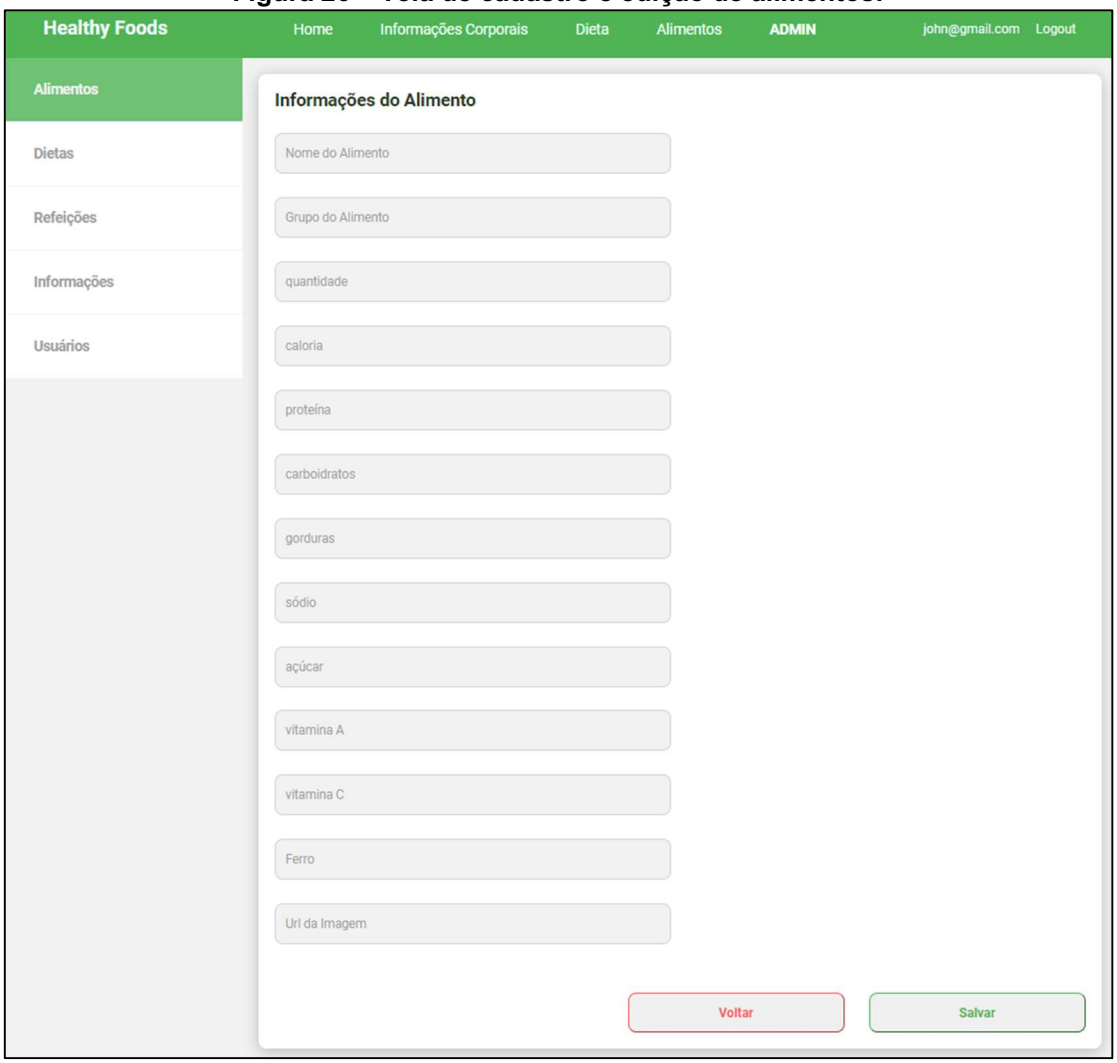
Figura 25 – CRUD de alimentos.



Fonte: Autoria própria (2022).

Para que a edição e criação de um alimento seja realizada, o sistema utiliza a mesma tela para interagir com o *backend*. Para editar um alimento, basta que o usuário clique no botão Edit no cartão com o alimento alvo e para criar, o usuário deve criar no botão adicionar localizado na parte superior da tela. Estes botões podem ser encontrados na Figura 25. Ao clicar em algum destes, o usuário é redirecionado para a tela contida na Figura 26 e nesta o usuário poderá preencher ou alterar os dados do alimento conforme a necessidade. Para salvar as modificações, o usuário deve clicar no botão salvar ou retornar a tela anterior clicando no botão voltar.

Figura 26 – Tela de cadastro e edição de alimentos.



Fonte: Autoria própria (2022).

Devido às outras telas do sistema serem semelhantes, as demais referentes as outras entidades estarão contidas no anexo B do trabalho.

5.2.3 Telas De dieta e Informações Próprias

Para que o paciente tenha acesso a sua dieta própria e suas informações corporais, foram desenvolvidas consecutivamente as telas de dieta e informações corporais. Para que estas sejam renderizadas o *frontend* realiza uma requisição ao *backend* utilizando apenas as credenciais do usuário, um parâmetro HTTP get e uma URL específica. Vale ressaltar que esta requisição não possui outros parâmetros ou variáveis, como apresentado na Figura 27 no Controlador das Informações Corporais Próprias, sendo responsabilidade do *backend* validar as credenciais enviadas e encontrar à qual usuário cadastrado estas pertencem.

Caso o usuário acesse a esta página sem estar autenticado, é apresentado uma mensagem lembrando-o que para acessar este recurso é necessário fazer o login, e caso este não possua nenhuma dieta cadastradas para si, a tela não apresentará nenhuma dieta.

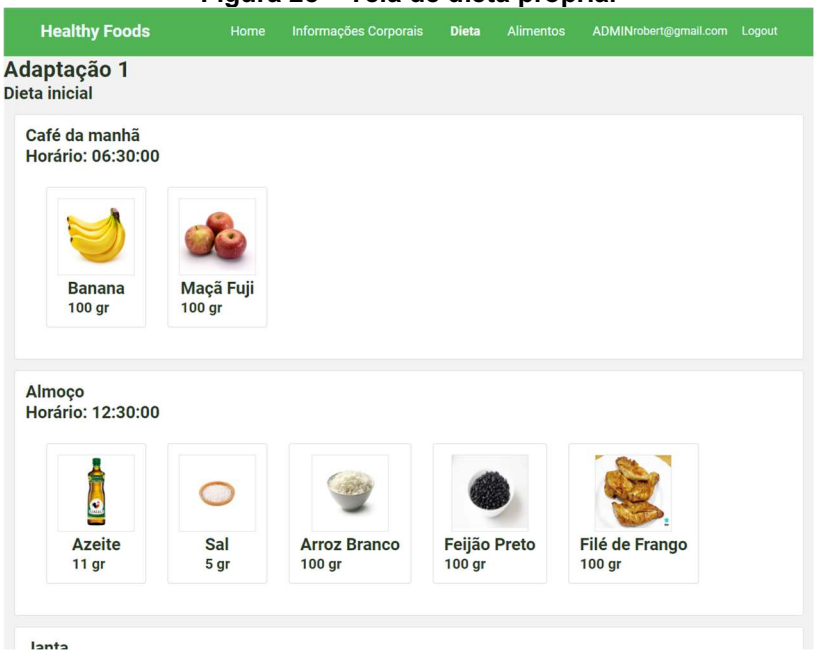
Figura 27 – Controlador de informações corporais.

```
@GetMapping
public ResponseEntity<Page<InformationDTO>> informationForCurrentUser(Pageable pageable) {
    var page :Page<InformationDTO> = informationService.informationForCurrentyUser(pageable);
    return ResponseEntity.ok().body(page);
}
```

Fonte: Autoria própria (2022).

A Figura 28 contém a dieta cadastrada para um usuário paciente como exemplo.

Figura 28 – Tela de dieta própria.



Fonte: Autoria própria (2022).

6 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo a implementação de um sistema de gerenciamento nutricional que utilizasse a classificação NOVA de alimentos, utilizando os *frameworks* Spring e React. Para que estes *frameworks* fossem utilizados, foi imprescindível o conhecimento dos conceitos aprendidos durante o curso de graduação nas linguagens de programação e métodos de desenvolvimento.

Por meio da execução do trabalho, foi possível entender de maneira aprofundada a complexidade dos diferentes processos utilizados para o desenvolvimento de um software. Após o fim de todas Sprint's planejadas, foi obtido um software que atende as especificações iniciais. O software desenvolvido pode, com maior clareza, prestar informações quanto ao consumo de alimentos processados e ultraprocessados.

Dentre as dificuldades encontradas durante a execução do projeto, pode se destacar a necessidade de diversos ajustes durante as Sprint's devido a pouca experiência do autor no processo de planejamento de software, assim como a necessidade de estudo sobre diferentes assuntos relacionados aos *frameworks Spring* e *React*, visto que não havia conhecimento pleno sobre os conceitos referentes a integração entre projetos *backend* e *frontend*.

7.1 Trabalhos Futuros

Cabe dizer que o sistema desenvolvido ainda não é uma aplicação comercial, pois customizações ainda podem ser aplicadas. Estas, poderiam ser desenvolvidas por meio de testes de usabilidade e/ou avaliações de profissionais de nutrição, a fim de encontrar possíveis pontos de melhoria.

REFERÊNCIAS

MALTA, D. C. et al. Mortalidade por doenças crônicas não transmissíveis no Brasil e suas regiões, 2000 a 2011. **Epidemiologia e Serviços de Saúde**, v. 23, n. 4, p. 599–608, dez. 2014.

LOUZADA, M. L. DA C. et al. The share of ultra-processed foods determines the overall nutritional quality of diets in Brazil. **Public Health Nutrition**, v. 21, n. 1, p. 94–102, 17 jul. 2017.

MONTEIRO, C. A. et al. The UN Decade of Nutrition, the NOVA food classification and the trouble with ultra-processing. **Public Health Nutrition**, v. 21, n. 1, p. 5–17, 21 mar. 2017.

MOUBARAC, J.-C. et al. Consumption of ultra-processed foods predicts diet quality in Canada. **Appetite**, v. 108, p. 512–520, jan. 2017.

TANENBAUM, A. S.; MAARTEN VAN STEEN. **Distributed systems : principles and paradigms**. [s.l.] Niederlande] Maarten Van Steen, 2016.

BARRY, D. K. **Representational State Transfer (REST)**. Barry & Associates. Disponível em: <https://www.service-architecture.com/articles/web-services/representational-state-transfer-rest.html>. Acesso em 20 out. 2022.

YU, X.; WU, J.; YIN, X. **Study on conformance testing of hypertext transfer protocol**. Disponível em: <https://ieeexplore.ieee.org/document/1209063>. Acesso em: 22 out. 2022.

SCHILDT, H. **Java : a beginner's guide**. New York Ny: Mcgraw Hill, 2022.

MYTHILY, M.; SAMSON ARUN RAJ, A.; THANAKUMAR JOSEPH, I. **An Analysis of the Significance of Spring Boot in The Market**. Disponível em: <https://ieeexplore.ieee.org/document/9850910>. Acesso em: 23 out. 2022.

BOGNER, J.; MERKEL, M. **To Type or Not to Type? A Systematic Comparison of the Software Quality of JavaScript and TypeScript Applications on GitHub**. Disponível em: <https://ieeexplore.ieee.org/document/9796341>. Acesso em: 25 out. 2022.

PAGOTTO, T. et al. **Scrum solo: Software process for individual development**. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7521555>.

POP, D.-P.; ALTAR, A. Designing an MVC Model for Rapid Web Application Development. **Procedia Engineering**, v. 69, p. 1172–1179, 2014.

FOWLER, M. **Patterns of enterprise application architecture**. Boston, Mass. ; Munich: Addison-Wesley, 2015.

ADAM, S. I.; MOEDJAHEDY, J. H.; MARAMIS, J. **RESTful Web Service Implementation on Unklab Information System Using JSON Web Token (JWT)**. Disponível em: <https://ieeexplore.ieee.org/document/9320801>. Acesso em: 22 out. 2022.

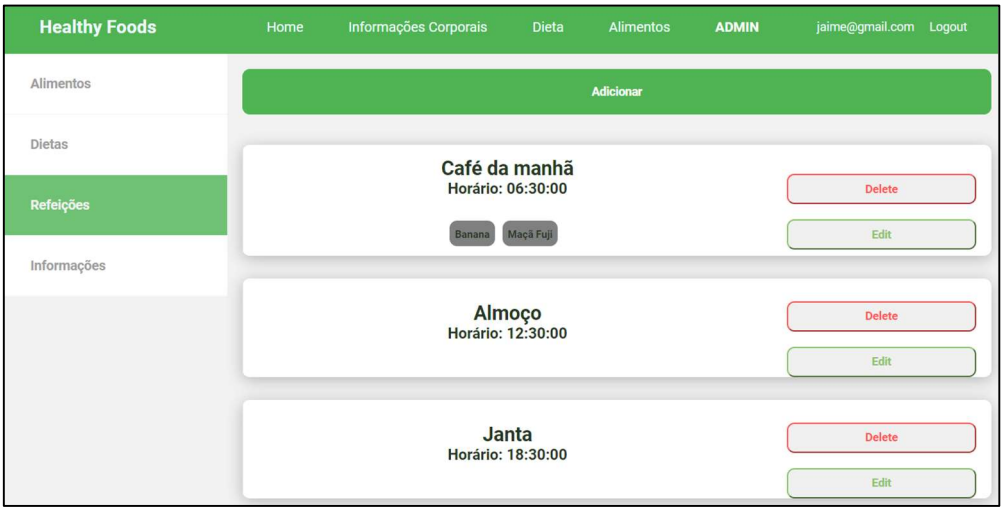
APÊNDICE A – *PRODUCT BACKLOG*

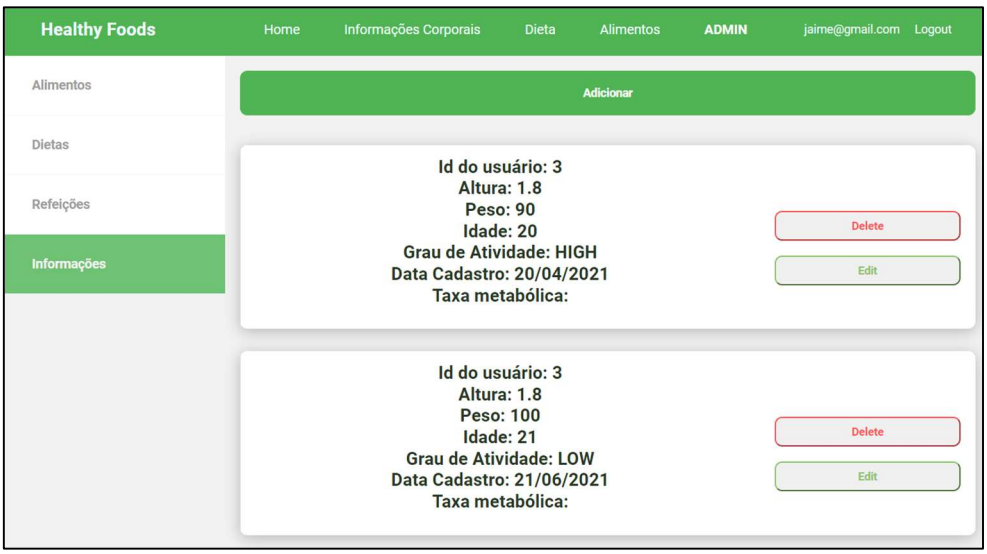
ID	DESCRIÇÃO	DATA DA INSERÇÃO	DATA DE SELEÇÃO PARA A SPRINT
1	Entidade User e Role	23/01/2022	31/01/2022
2	Entidade Information	23/01/2022	31/01/2022
3	Entidade Diet, Meal e Food	23/01/2022	07/02/2022
4	Refatoração de código	13/02/2022	14/02/2022
5	Seed de dados	13/02/2022	07/03/2022
6	Camada de Repository	13/02/2022	07/03/2022
7	Camada de Serviço da entidade User	13/02/2022	14/03/2022
8	CRUD da entidade Food	13/03/2022	28/03/2022
9	Autenticação das rotas da entidade Food	13/03/2022	28/03/2022
10	CRUD das entidades Meal e Diet	13/03/2022	04/04/2022
11	Autenticação das rotas das entidades Meal e Diet	13/03/2022	04/04/2022
12	CRUD das entidades Information e User	13/03/2022	11/04/2022
13	Autenticação das rotas Information e User	13/03/2022	11/04/2022
14	Refatoração de código	17/03/2022	18/04/2022
15	Implementação das rotas de informação corporal e dietas próprias	05/06/2022	06/06/2022
16	Configurações adicionais de integração	05/06/2022	13/06/2022

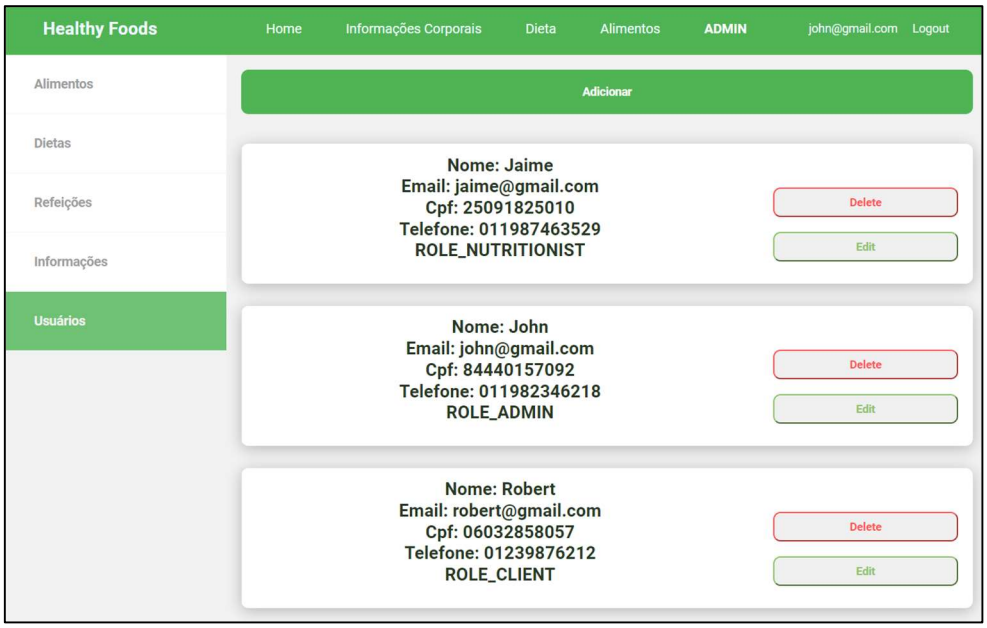
ID	DESCRIÇÃO	DATA DE INSERÇÃO	DATA DE SELEÇÃO PARA A SPRINT
17	Criação do projeto e ajuste de dependências	08/05/2022	23/05/2022
18	Component Navbar	08/05/2022	23/05/2022
19	Página Landing Page	08/05/2022	30/05/2022
20	Component FoodCard	08/05/2022	06/06/2022
21	Botão Home e ajustes de rotas	08/05/2022	06/06/2022
22	Navbar Admin	08/05/2022	13/06/2022
23	Rotas Admin Navbar	08/05/2022	13/06/2022
24	Ajustes Loader Components	08/05/2022	20/06/2022
25	Refatoração de código	26/06/2022	27/06/2022
26	Tela de Login	26/06/2022	11/07/2022
27	Estilos Telo de Login	26/06/2022	18/07/2022
28	Validações Formulário Login	26/06/2022	01/08/2022
29	Resquisições com Usuário logado	26/06/2022	08/08/2022
30	Decodificando Token e finalizando ajustes de autenticação	17/07/2022	15/08/2022
31	Página CRUD de alimentos	17/07/2022	05/09/2022
32	Estilização CRUD de alimentos	17/07/2022	12/09/2022
33	Página CRUD de refeições	17/07/2022	19/09/2022
34	Estilização CRUD de refeições	17/07/2022	26/09/2022
35	Página CRUD de dietas e informações corporais	18/09/2022	03/10/2022
36	Estilização CRUD de dietas e informações corporais	18/09/2022	10/10/2022
37	Página CRUD de usuários	18/09/2022	17/10/2022
38	Estilização CRUD de usuários	18/09/2022	17/10/2022
39	Páginas de dieta e informações corporais próprias	16/10/2022	24/10/2022
40	Estilização dietas e informações corporais próprias	16/10/2022	31/10/2022

APÊNDICE B – OUTRAS TELAS DO SISTEMA


Healthy Foods
Home
Informações Corporais
Dieta
Alimentos
ADMIN
jaime@gmail.com
Logout
Alimentos
Dietas
Refeições
Informações
Adicionar
Cetogenica
Dieta baseada na baixa ingestao de carboidratos
Início: Fri, 04 Nov 2022 09:35:19 GMT
Fim: Thu, 04 May 2023 09:35:19 GMT
Refeições: Café da manhã Almoço Janta
Usuários: robert@gmail.com
Delete
Edit


Healthy Foods
Home
Informações Corporais
Dieta
Alimentos
ADMIN
jaime@gmail.com
Logout
Alimentos
Dietas
Refeições
Informações
Adicionar
Café da manhã
Horário: 06:30:00
Banana
Maçã Fuji
Delete
Edit
Almoço
Horário: 12:30:00
Delete
Edit
Janta
Horário: 18:30:00
Delete
Edit


Healthy Foods
Home
Informações Corporais
Dieta
Alimentos
ADMIN
jaime@gmail.com
Logout
Alimentos
Dietas
Refeições
Informações
Adicionar
Id do usuário: 3
Altura: 1.8
Peso: 90
Idade: 20
Grau de Atividade: HIGH
Data Cadastro: 20/04/2021
Taxa metabólica:
Delete
Edit
Id do usuário: 3
Altura: 1.8
Peso: 100
Idade: 21
Grau de Atividade: LOW
Data Cadastro: 21/06/2021
Taxa metabólica:
Delete
Edit


Healthy Foods
Home
Informações Corporais
Dieta
Alimentos
ADMIN
john@gmail.com
Logout
Alimentos
Dietas
Refeições
Informações
Usuários
Adicionar
Nome: Jaime
Email: jaime@gmail.com
Cpf: 25091825010
Telefone: 011987463529
ROLE_NUTRITIONIST
Delete
Edit
Nome: John
Email: john@gmail.com
Cpf: 84440157092
Telefone: 011982346218
ROLE_ADMIN
Delete
Edit
Nome: Robert
Email: robert@gmail.com
Cpf: 06032858057
Telefone: 01239876212
ROLE_CLIENT
Delete
Edit