



Aplicação de Registo Nutricional

ISEP

2020 / 2021

1170500 Hugo Frias

Aplicação de Registo Nutricional

ISEP

2020 / 2021

1170500 Hugo Frias



Licenciatura em Engenharia Informática

Setembro 2021

Orientador ISEP: **Goreti Marreiros**

Supervisor Externo: **Diogo Martinho**

A minha família, amigos e parceira

Agradecimentos

Agradeço à minha família que me possibilitou a realização da licenciatura e por todo o apoio dado na sua realização.

Agradeço aos meus colegas e amigos que me acompanharam durante esta jornada académica.

Agradeço à professora Goreti Marreiros pela orientação dada no decorrer do projeto de estágio e pelo tempo dedicado à revisão do relatório

Por fim, agradeço ao professor Diogo Martinho pela disponibilidade e por todo o *feedback* dado durante a realização do projeto.

Resumo

A quantidade e qualidade dos alimentos que fazem parte dos nossos hábitos alimentares têm uma importância fulcral no que toca ao aparecimento de doenças relacionadas com a dieta alimentar como os diabetes, o excesso de peso ou até o cancro. De modo a prevenir o aparecimento de doenças ligadas ao comportamento alimentar é essencial o registo dos hábitos alimentares individuais bem como um acompanhamento médico por parte de um nutricionista. Em tempos pandémicos, revelou-se essencial o desenvolvimento de ferramentas para o acompanhamento nutricional de pacientes à distância.

Neste relatório é detalhado o processo de desenvolvimento de uma aplicação web orientada a pacientes e nutricionistas que possui funcionalidades como:

- A criação de planos alimentares customizados por parte de nutricionistas para os seus pacientes;
- O registo dos consumos alimentares diários de pacientes da aplicação;
- A realização de desafios nutricionais motivadores por parte de pacientes;
- Um sistema que possibilitasse a criação de refeições customizadas com através dos alimentos presentes na base de dados da aplicação *web*;
- Um sistema de *Chat* direto entre o paciente e o seu nutricionista, etc.

Foi desenvolvida uma aplicação *web* confiável que vai de encontro às funcionalidades propostas inicialmente, como a criação de planos detalhados por nutricionistas e o registo dos hábitos alimentares diários dos seus pacientes, conseguindo proporcionar soluções a utilizadores que desejam iniciar uma jornada nutricional.

Palavras-chave (Tema): Nutrição, Registo de Hábitos Alimentares, Acompanhamento Nutricional

Palavras-chave (Tecnologias): *JavaScript, React, mongoDB, Node.JS*

Índice

1.	<i>Introdução</i>	3
1.1.	Enquadramento/Contexto	3
1.2.	Descrição do Problema	4
1.2.1.	Objetivos	4
1.2.2.	Abordagem.....	5
1.2.3.	Contributos	6
1.2.4.	Planeamento do trabalho	6
1.3.	Estrutura do relatório	7
2.	<i>Estado da arte</i>	9
2.1.	Trabalhos relacionados	9
2.1.1.	Lifesum	9
2.1.2.	Yazio	11
2.1.3.	MyFitnessPal	13
2.1.4.	Comparações entre plataformas	15
2.2.	Tecnologias existentes	17
2.2.1.	JavaScript	18
2.2.2.	Bases de dados.....	19
2.2.3.	Node.js e Express.js.....	20
2.2.4.	React.....	22
2.2.5.	JavaScript vs C#	23
2.2.6.	React vs Angular.....	25
2.2.7.	Outras bibliotecas	26
3.	<i>Análise e desenho da solução</i>	28
3.1.	Domínio do problema	28
3.2.	Requisitos funcionais e não funcionais	30
3.2.1.	Requisitos funcionais	30
3.2.2.	Requisitos não funcionais	32
3.3.	Desenho	33
3.3.1.	Vista Física	34
3.3.2.	Vista Lógica.....	35

3.3.3.	Vista de Processos.....	37
3.3.4.	Vista de Implementação	49
4.	<i>Implementação da Solução</i>	51
4.1.	Descrição da implementação.....	51
4.1.1.	UC1 – Registo de utilizador na aplicação	52
4.1.2.	UC2 – Registo de utilizador na aplicação	54
4.1.3.	UC3 – Enviar pedido de acompanhamento	56
4.1.4.	UC4 – Criar plano alimentar.....	57
4.1.5.	UC5 – Adição de refeição planeada	58
4.1.6.	UC6 – Visualização de gráficos e tabelas relativas a planos	61
4.1.7.	UC7 – Decidir sobre o pedido de acompanhamento.....	63
4.1.8.	UC8 – Registo de peso diário	64
4.1.9.	UC9 – Criação de prato/alimento customizado.....	66
4.1.10.	UC10 – Registo de refeição consumida.....	68
4.1.11.	UC11 – Histórico de refeições do cliente	69
4.1.12.	UC12 – Seleção de desafio nutricional.....	70
4.1.13.	UC13 – Seleção de desafio nutricional.....	72
4.1.14.	Bibliotecas utilizadas	73
4.2.	Testes	76
4.2.1.	Testes de integração	76
4.2.2.	Testes unitários	77
4.3.	Avaliação da solução	79
5.	Conclusões	81
5.1.	Objetivos concretizados	81
5.2.	Limitações e trabalho futuro	82
5.3.	Apreciação final	83
<i>Referências.....</i>	85	
<i>Anexo A – Diagramas.....</i>	90	

Índice de Figuras

Figura 1 - Diagrama de Gantt (gerado em [3])	7
Figura 2 - Banner da Lifesum (retirada de [4])	9
Figura 3 - Algumas das funcionalidades da LifeSum (adaptado de [6])	11
Figura 4 - Banner da Yazio (retirado de [8])	12
Figura 5 - Algumas das diversas funcionalidades da Yazio (adaptada de [9])	13
Figura 6 - Banner do MyFitnessPal (retirado de [13])	14
Figura 7 - Exemplos de funcionalidades do MyFitnessPal (adaptado de [14])	15
Figura 8 - Exemplo de um Schema da MongoDB (retirado de [29])	20
Figura 9 - Logo oficial do Node.js (retirado de [31])	21
Figura 10 - Vista física (nível 2).....	34
Figura 11 - Vista lógica (nível 2).....	35
Figura 12 - Vista lógica BackEnd (nível 3).....	35
Figura 13 - Vista lógica UI (nível 3)	36
Figura 14 - SSD do caso de uso 1.....	37
Figura 15 - Secção do SD do caso de uso 1 relativa ao nutricionista	38
Figura 16 - SSD do caso de uso 2	38
Figura 17 - SD do caso de uso 2.....	39
Figura 18 - Secção do SD do caso de uso 1 relativa ao nutricionista	40
Figura 19 - SSD do caso de uso 3	40
Figura 20 - SSD do caso de uso 4	41
Figura 21 - SSD do caso de uso 5	42
Figura 22 - SSD do caso de uso 6	43
Figura 23 - SSD do caso de uso 7	44
Figura 24 - SSD do caso de uso 8	45
Figura 25 - SSD do caso de uso 9	46

Figura 26 - SSD do caso de uso 11.....	47
Figura 27 - SSD do caso de uso 12.....	48
Figura 28 - Vista de implementação (nível 2).....	49
Figura 29 - Vista de implementação da BackEnd (nível 3)	50
Figura 30 - Vista de implementação da UI (nível 3)	50
Figura 31 - Menus da aplicação para utilizadores não registados, nutricionistas e clientes respetivamente	51
Figura 32 - Página inicial do registo onde é necessário indicar o tipo de utilizador	52
Figura 33 - Página de registo de um cliente	53
Figura 34 - Página de registo de um nutricionista	53
Figura 35 - Alguns dos termos e condições mostrados no registo	54
Figura 36 - Mensagem de sucesso no registo	54
Figura 37 - Página de login	55
Figura 38 – Mensagem de sucesso no Login.....	55
Figura 39 - Tabela com os dados dos clientes sem acompanhamento.....	56
Figura 40 - Mensagem de sucesso no envio de um pedido de acompanhamento	57
Figura 41 - Página de criação de um plano	57
Figura 42 - Mensagem de sucesso na criação do plano.....	58
Figura 43 - Página de seleção do plano a editar	59
Figura 44 - Página para inserir a comida desejada.....	59
Figura 45 - Página com a seleção da comida desejada	60
Figura 46 - Página com a seleção da porção de comida	60
Figura 47 - Refeição criada e adicionada ao calendário.....	61
Figura 48 - Página com os dados de um determinado plano.....	61
Figura 49 - Notificação de recebimento de pedido de acompanhamento.....	63
Figura 50 - Página para a escolha de nutricionista.....	64
Figura 51 - Mensagem de sucesso na aceitação do pedido de acompanhamento	64

Figura 52 - Perfil pessoal de um utilizador	65
Figura 53 - Mensagem de sucesso no registo diário do peso	65
Figura 54 - Proibição de inserção do peso diário após registo no mesmo dia	66
Figura 55 - Página de criação de um prato personalizado	66
Figura 56 - Tabela com o alimento adicionado ao prato	67
Figura 57 - Mensagem de sucesso na adição do prato/alimento personalizado	67
Figura 58 - Calendário para registo de refeição com nota auxiliar relativa às refeições planeadas	68
Figura 59 - Calendário com a refeição registada.....	69
Figura 60 - Página com as informações do plano para o cliente	69
Figura 61 - Mensagens relativas ao desempenho do utilizador	70
Figura 62 - Desafios nutricionais disponíveis	70
Figura 63 - Seleção de um desafio nutricional	71
Figura 64 – Ativação do desafio escolhido.....	71
Figura 65 - Crachás possíveis após completamento de um desafio, sendo ele bem-sucedido (crachá da esquerda) ou malsucedido (crachá da direita).....	72
Figura 66 - Janela de Chat entre utilizadores	72
Figura 67 - Código relativo à implementação da biblioteca FullCalendar	73
Figura 68 - Dados para a geração de gráficos	74
Figura 69 - Lista de utilizadores inseridos na base de dados do ChatEngine.....	75
Figura 70 - Chamada à API do ChatEngine para a criação de um chat entre dois utilizadores	75
Figura 71 - Código relativo à implementação da janela de Chat	75
Figura 72 - Código de um dos testes de integração	76
Figura 73 - Código relativo a um dos testes unitários.....	78
Figura 74 - Modelo de domínio	91
Figura 75 - Diagrama de casos de uso	92
Figura 76 - SD do caso de uso 1.....	93

Figura 77 - SD do caso de uso 3.....	94
Figura 78 - SD do caso de uso 4.....	95
Figura 79 - SD do caso de uso 5.....	96
Figura 80 - SD do caso de uso 6.....	97
Figura 81 - SD do caso de uso 7.....	98
Figura 82 - SD do caso de uso 8.....	99
Figura 83 - SD do caso de uso 9.....	100
Figura 84 - SD do caso de uso 11.....	101
Figura 85 - SD do caso de uso 12.....	102

Índice de Tabelas

Tabela 1 - Comparação entre a web-app desenvolvida e as 3 apps mencionadas neste capítulo	16
Tabela 2 - Comparação entre as propriedades do JavaScript e as propriedades do C#.....	24
Tabela 3 - Comparação entre as componentes do JavaScript e as componentes do C#.....	24
Tabela 4 - Comparação entre as componentes do React e as componentes do Angular	26
Tabela 5 - Tabela com exemplos de alguns dos testes da classe Client	77
Tabela 6 - Tabela com exemplos de testes unitários feitos na classe Client	78

Índice de Gráficos

Gráfico 1 - Linguagens mais utilizadas dos utilizadores do StackOverflow (adaptado de [22])	19
Gráfico 2 - Web Frameworks mais utilizadas dos utilizadores do StackOverflow (adaptado de [44])	23
Gráfico 3 - Gráfico com a flutuação de calorias diárias de um cliente	62
Gráfico 4 - Gráfico circular com a quantidade de refeições registadas comparadas com as refeições planeadas, porções planeadas e dias das refeições	62

Notação e Glossário

AAA	<i>Arrange, Act and Assert</i>
Angular	Biblioteca de <i>JavaScript</i>
API	<i>Application Programming Interface</i>
BackEnd	Componente relativa ao processamento e registo dos dados
Browser	Navegador que permite a interação com documentos <i>HTML</i>
C#	Linguagem de programação
CSS	<i>Cascading Style Sheets</i>
DOM	<i>Document Object Model</i>
DTO	<i>Data Transfer Object</i>
FrontEnd	Componente que representa a interface de interação com o utilizador
GECAD	Grupo de Investigação em Engenharia e Computação Inteligente para a Inovação e o Desenvolvimento
HTTP	<i>Hypertext Transfer Protocol</i>
JavaScript	Linguagem de programação
JSON	<i>JavaScript Object Notation</i>
NPM	<i>Node Package Manager</i>
OOP	<i>Object-Oriented Programming</i>
React	Biblioteca de <i>JavaScript</i>
REST	<i>Representational State Transfer</i>
SD	<i>Sequence Diagram</i>
Sinon	Biblioteca de <i>JavaScript</i> para testes
SO	Sistema operativo
SQL	<i>Structured Query Language</i>
SSD	<i>System Sequence Diagram</i>

TypeScript Linguagem de programação

UC *Use Case*

UI *User Interface*

UML *Unified Modeling Language*

VSC *Visual Studio Code*

Web-App Aplicação *web*

1. Introdução

Esta secção está subdividida em 3 secções. Primeiramente, começa-se com um enquadramento do problema no âmbito deste projeto e é descrita a motivação para a aceitação do mesmo. Prossegue-se com a descrição do problema, o que está dividido nos objetivos do projeto, a abordagem considerada para o mesmo, os contributos que este projeto traz para a sociedade e o planeamento deste trabalho. Por fim, realiza-se uma apresentação sucinta dos capítulos incluídos no relatório.

1.1. Enquadramento/Contexto

O tipo e a quantidade de alimentos que ingerimos tem um grande impacto na nossa saúde. As doenças relacionadas com a dieta alimentar são uma das principais preocupações de saúde pública, continuando a pôr em perigo a saúde da população, assim como a própria sustentabilidade dos sistemas de saúde. Muitos destes problemas podem ser prevenidos e até evitados através de um acompanhamento e registo dos hábitos alimentares de um paciente, quando acompanhados pela supervisão de um profissional de saúde. Contudo, nos dias correntes, é cada vez mais difícil que uma pessoa comum tenha a disponibilidade para consultas regulares com profissionais de saúde, especialmente no contexto pandémico atual que implica ainda mais restrições a consultas hospitalares. Tendo em conta os avanços tecnológicos e o aumento da literacia digital, a criação de uma solução remota para a gestão de saúde e acompanhamento destes pacientes providênciam uma excelente oportunidade para o desenvolvimento individual nutricional.

A motivação pessoal do estudante para a aceitação deste projeto baseou-se no facto de que o mesmo sempre teve um interesse em nutrição e sempre realizou registos de consumos alimentares, estando já familiarizado com o processo e com as necessidades dos potenciais pacientes desta *web-app*. Para além disso, este projeto tem como principal objetivo ajudar a sociedade no ramo da saúde, que é algo que traz um grande sentido de autorrealização ao estudante.

1.2. Descrição do Problema

A ingestão desequilibrada de alimentos pode resultar em distúrbios metabólicos, desnutrição, excesso de peso, baixo desempenho mental e outros fatores médicos de risco, como doenças cardiovasculares, diabetes ou cancro. Pelos dados publicados pela *Health at a Glance* da Organização para a Cooperação e Desenvolvimento Económico (*OCDE*) em 2019, é possível visualizar que a taxa da população acima de 15 anos com sobre peso ou obesidade em Portugal é de 67,6% [1] estando em 4º lugar entre os países da *OCDE*. Portugal apresenta assim um grande problema no que toca à ingestão alimentar, algo que pode ser combatido através do acompanhamento profissional de um nutricionista, e da realização de registos alimentares. Através do uso de uma aplicação tecnológica, este combate pode ser feito de maneira mais conveniente quando comparada aos métodos tradicionais (acompanhamento através de consultas em meio hospitalar), permitindo a pacientes melhorarem a qualidade do seu estado nutricional e da sua ingestão alimentar a partir de sua própria casa. Uma aplicação nutricional possui diversas funcionalidades que permitem ao paciente atingir esses objetivos, permitindo o contacto direto entre o profissional de saúde e o paciente, bem como a criação de planos nutricionais customizados com base nas necessidades nutricionais do mesmo e o seu registo diário de consumos alimentares que posteriormente serão analisados pelo nutricionista que o segue.

1.2.1. Objetivos

O grande objetivo deste projeto é facilitar tanto o registo diário alimentar dos pacientes como o seu acompanhamento por parte de um profissional de saúde, medidas fundamentais para abordar os problemas descritos na secção anterior (1.2). É então necessário a criação de uma plataforma que permita a comunicação entre nutricionistas e pacientes e o seu acompanhamento através da criação de planos alimentares detalhados criados pelo profissional de saúde e de UI's *user-friendly* para o registo diário alimentar de pacientes que não possuam um grande conhecimento de tecnologias, visto serem estes os mais afetados pelos problemas referidos na secção anterior (1.2). A *web-app* pretende atingir isto através de funcionalidades como:

- Sistema de registo/login para pacientes e nutricionistas, funcionalidade fundamental para o uso da aplicação;
- Sistema de registo diário de refeições com base numa base de dados de alimentos de modo a permitir que o paciente possua um registo detalhado dos seus hábitos alimentares, para serem posteriormente analisados pelo nutricionista;
- Sistema para criação de refeições customizadas com o intuito de que o paciente consiga partilhar as suas receitas personalizadas com outros pacientes da aplicação ou possivelmente com o nutricionista que o acompanha;
- Sistema para criação de planos nutricionais personalizados e sua edição para serem posteriormente atribuídos aos pacientes que são seguidos pelo nutricionista em questão;
- Sistema de escolha de pacientes por parte de um nutricionista;
- Sistema de desafios nutricionais para um paciente, motivando-o durante a sua jornada nutricional e introduzindo aspetos de gamificação à aplicação;
- Perfil pessoal de um paciente criando assim uma área pessoal onde este pode registar o seu peso diário, ter acesso aos seus dados gerais, bem como aos seus desafios nutricionais;
- Secção para a visualização das estatísticas dos planos dos pacientes de um nutricionista de modo a permitir ao profissional de saúde fazer um melhor acompanhamento dos seus pacientes através da visualização de gráficos e tabelas;
- Secção para o paciente visualizar as estatísticas do seu plano de modo, onde este pode verificar o seu progresso e receber mensagens personalizadas que lhe deem *feedback* relativamente ao mesmo;
- Sistema de *chat* direto entre o paciente e o nutricionista permitindo a troca de *feedback* entre ambos, bem como o atendimento especializado do nutricionista ao paciente num regime remoto.

1.2.2. Abordagem

Após uma análise do problema em questão e do contexto do projeto, a abordagem tomada consistiu no desenvolvimento de uma aplicação *web* com recurso a tecnologias como o *Javascript* e o *React* que tem como atores principais nutricionistas e pacientes e que possui

funcionalidades orientadas para a nutrição, com enfâse na criação de planos alimentares e no registo de informação diária nutricional e consequente análise dos registos alimentares por parte dos nutricionistas.

A aplicação está dividida em duas componentes: A *frontend* encontra-se no formato *web* usando a biblioteca *React* da linguagem *Javascript* e trata das interações entre nutricionistas e pacientes e regista as informações dadas pelos mesmos. A *backend* encontra-se na linguagem *Javascript* e trata do tratamento dos dados provenientes da *frontend* e, posteriormente, guarda-os numa base de dados.

1.2.3. Contributos

O contributo principal desta aplicação é ajudar pacientes que pretendem melhorar o seu estilo de vida alimentar, fornecendo-lhes ferramentas para conseguirem ter uma ingestão equilibrada de alimentos, através de um registo remoto diário dos seus consumos alimentares. A *web-app* também ajuda nutricionistas a estarem mais envolvidos com os pacientes, quer diretamente (através de um *chat*) ou indiretamente (acompanhamento das refeições tomadas e alterações em planos). O trabalho aqui desenvolvido irá também dar um contributo ao *Food Friend*, um projeto a decorrer atualmente no *GECAD* que está relacionado com a área de gestão de doentes com diabetes. Este é um projeto focado em abordar dois dos aspetos das doenças relacionadas com a dieta alimentar: a prevenção de má nutrição para pacientes que necessitem de alimentação por sonda e o tratamento nutricional transmural para pacientes com doenças crónicas. De modo a combater estes dois aspetos o *Food Friend* pretende combinar *hardware* na forma de sensores e *software* na forma de aplicações ou portais *web*, permitindo uma oportunidade de crescimento num mercado avaliado em 4.2 triliões de dólares em 2017 e o melhoramento da vida de pacientes com doenças relacionadas com a sua dieta alimentar [2].

1.2.4. Planeamento do trabalho

Este projeto foi planeado imediatamente após a sua aceitação, tendo sido definidas datas para as seguintes categorias:

- Estudo do estado da arte: de 22 de Março a 5 de Abril;
- Planeamento e design da solução: de 06 de Abril a 30 de Abril;
- Implementação da solução: de 1 de Maio a 1 de Agosto;
- Testes e validação: de 1 de Maio a 1 de Agosto;
- Escrita do relatório: de 22 de Março a 12 de Setembro;
- Revisão: de 1 de Setembro a 12 de Setembro.

É possível visualizar o planeamento anteriormente descrito no diagrama de *Gantt* da figura 1

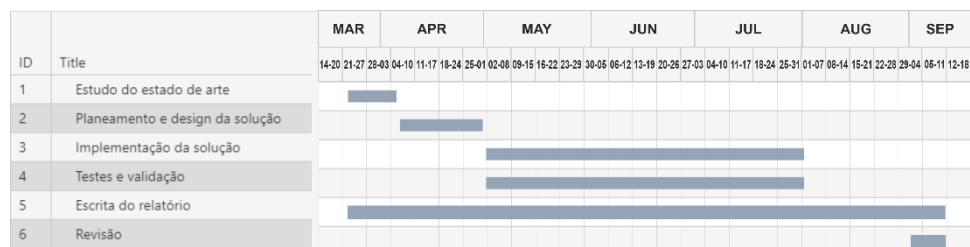


Figura 1 - Diagrama de Gantt (gerado em [3])

Este projeto foi acompanhado semanalmente através de reuniões com o supervisor assumindo um padrão de desenvolvimento ágil através da verificação do estado de tarefas e definição de posteriores tarefas de forma iterativa.

Adicionalmente, recorreu-se ao uso da plataforma *Trello* de modo a fazer a gestão diária do projeto.

1.3. Estrutura do relatório

Para além deste capítulo introdutório, este relatório possui mais 4 capítulos sendo eles:

- Capítulo 2 – Estado da arte: Neste capítulo são analisadas todas as tecnologias utilizadas neste projeto, sendo também feito um estudo quanto a outros trabalhos dentro do mesmo ramo;
- Capítulo 3 – Análise e desenho da solução: Neste capítulo faz-se uma análise do domínio do problema, fala-se dos requisitos funcionais e não funcionais e faz-se um desenho detalhado da solução;
- Capítulo 4 - Implementação da solução: Neste capítulo fala-se da implementação documentada no capítulo anterior em que se faz uma descrição da implementação e dos testes realizados e posterior avaliação dos mesmos;

- Capítulo 5 – Conclusões: Neste capítulo apresenta-se os resultados obtidos contrastados pelos objetivos definidos inicialmente e faz-se uma análise às limitações e possíveis extensões do mesmo, finalizando com uma apreciação final do trabalho.

2. Estado da arte

Neste capítulo é feito um resumo de alguns trabalhos relacionados com o projeto em questão, sendo abordadas as suas funcionalidades e a sua história. De seguida, faz-se uma revisão e documentação das tecnologias que foram necessárias ao desenvolvimento do projeto.

2.1. Trabalhos relacionados

A nutrição é uma área da saúde bastante importante nos dias correntes, acabando por criar uma grande necessidade da existência de plataformas simples e acessíveis para serem utilizadas por qualquer utilizador, independentemente do seu estatuto educacional ou conhecimento tecnológico. Isto levou à criação de inúmeras aplicações nutricionais gratuitas e de fácil acesso, com o intuito de preencher esse *gap* no mercado e posteriormente angariar subscrições monetárias mensais dos seus utilizadores. Neste relatório são abordadas três das aplicações nutricionais com melhor classificação e mais *downloads* na *Play Store* da *Google*.

2.1.1. Lifesum



Figura 2 - Banner da Lifesum (retirada de [4])

Uma das aplicações estudada é a *LifeSum*, sendo uma aplicação muito bem cotada na *Google Play Store* possuindo 4,5 estrelas em 5 possíveis e tendo mais de 45 milhões de instalações à data de escrita do relatório.

Esta é uma aplicação que possui 2 escritórios físicos em Estocolmo e Los Angeles, angariando mais de 20 milhões de dólares anuais e estando disponível em 11 linguagens, sendo acessível a maior parte do seu mercado principal sendo ele os Estados Unidos e a Europa [5].

Depois de instalar esta aplicação, o utilizador depara-se com um menu de login, permitindo que seja feito através de e-mail ou da ligação das contas *Google* ou *Facebook* do mesmo à aplicação. No registo da aplicação é primeiramente perguntado ao utilizador as suas metas, sendo elas o mantimento, perda ou ganho de peso. De seguida são registadas as informações básicas do utilizador como o gênero, idade, altura e peso. Dependendo dos valores introduzidos, é automaticamente calculado a quantidade de calorias e macronutrientes (carboidratos, gordura e proteínas) que o utilizador precisa de ingerir diariamente, sendo posteriormente remetido à página inicial da aplicação. Aqui, o utilizador começa a ter acesso às funcionalidades da aplicação, nomeadamente:

- Secção de registo do consumo de água, bem como do consumo de alimentos, estando ele dividido pelas 4 refeições diárias (pequeno-almoço, almoço, lanche e jantar);
- Secção de registo de atividade física;
- Um teste relativo aos hábitos de consumo atuais do utilizador, atribuindo uma nota ao mesmo;
- Uma área de perfil, possuindo algumas estatísticas individuais e as informações básicas do utilizador;
- Uma área de planos de nutrição, fazendo um inquérito ao utilizador e recomendando um ao mesmo com base nas suas respostas;
- Uma secção com diversas receitas de pratos para o utilizador.

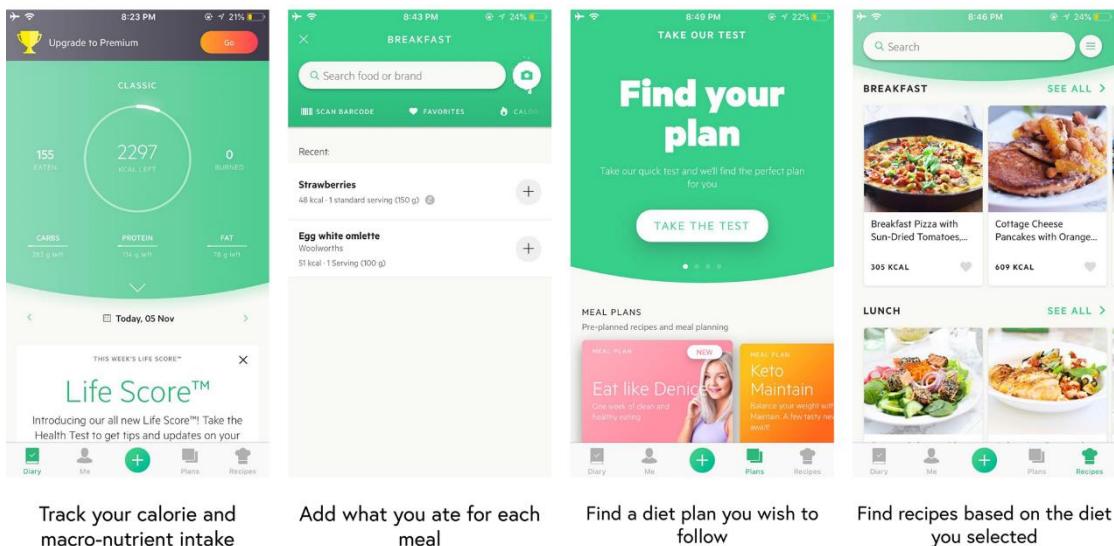


Figura 3 - Algumas das funcionalidades da LifeSum (adaptado de [6])

O nível de detalhe presente nestas funcionalidades irá depender diretamente do estatuto do utilizador, caso ele não seja pago não irá ter acesso a algumas destas funcionalidades (como à realização de planos nutricionais), apresentando incentivos para que o mesmo se torne num utilizador *premium*.

2.1.2. Yazio

Outra das aplicações com um grande renome no mundo das aplicações nutricionais é a *Yazio*, contendo um rating até à data de escrita do relatório de 4,6 estrelas em 5 possíveis e mais de 10 milhões de *downloads*.

Esta foi criada por Sebastian Weber e Florian Weißenstein em 2013, tendo uma base física em Erfurt, na Alemanha. À data de Junho de 2020 contavam com uma equipa de 35 trabalhadores e a aplicação estava disponível em 20 linguagens diferentes [7].



Figura 4 - Banner da Yazio (retirado de [8])

Após a instalação da aplicação é inicialmente pedido ao utilizador para definir o seu objetivo principal, seja ele a perda de peso, ganho de massa muscular ou retenção do peso atual. De seguida são sugeridas algumas funcionalidades para o uso do cliente, entre elas a contagem de calorias ou a apresentação de receitas saudáveis. Após isso, são feitas algumas perguntas ao utilizador em relação ao seu peso atual, ao objetivo de peso, gênero, altura e idade. Após a introdução destas informações básicas, o utilizador é redirecionado para a *homepage* da aplicação, onde é possível começar a aceder às funcionalidades da aplicação, destacando-se:

- Secção com um resumo das calorias diárias necessárias, e macronutrientes associados (carboidratos, proteína e gordura);
- Área para introdução de refeições, segmentada nas 4 refeições principais do dia (pequeno-almoço, almoço, lanche e jantar);
- Área que monitoriza atividade física, sendo possível registar passos ou exercícios especializados;
- Área de introdução do peso diário e monitores do consumo diário de água
- Secção onde é possível anotar o estado emocional do utilizador;
- Área relativa à prática do jejum intermitente, possuindo monitores, relógios e informação educacional sobre o ato em si;
- Área com diversas receitas de pratos recomendados;
- Área de criação de planos nutricionais;
- Área com um perfil do utilizador, contendo o seu progresso e desafios nutricionais.



Figura 5 - Algumas das diversas funcionalidades da Yazio (adaptada de [9])

O nível de pormenor destas funcionalidades vai estar, como em todas as aplicações, dependente do facto do mesmo ser um cliente pago ou não, havendo um maior detalhe em cada funcionalidade e até mesmo funcionalidades extra caso este faça parte do grupo de membros *premium* da aplicação.

2.1.3. MyFitnessPal

A aplicação *MyFitnessPal* acaba por ser a mais popular dentro do género de aplicações de nutrição possuindo, até à data de escrita deste relatório, mais de 80 milhões de downloads e uma classificação de 4,4 estrelas em 5 possíveis.

Esta é uma aplicação fundada em 2005 por Albert Lee e Mike Lee [10] sendo posteriormente vendida à empresa *Under Armor*, num negócio realizado em 2015 e com valores na ordem dos 475 milhões de dólares, valores esses atingidos quando a aplicação já contava com 80 milhões de utilizadores registados [11]. Esta plataforma mudou novamente de mãos em 2020, quando foi vendida à empresa *Francisco Partners* num negócio que rondou os 345 milhões de dólares [12].



Figura 6 - Banner do MyFitnessPal (retirado de [13])

Aquando da instalação da aplicação, é inicialmente feito um registo na mesma, sendo possível fazer o mesmo através de uma conta no *Facebook*. Antes mesmo de ser registado o email do utilizador, são perguntadas algumas das suas informações e metas na jornada nutricional do utilizador, começando por perguntar quais são os seus objetivos (perder, manter ou ganhar peso), os seus níveis de atividade, gênero, data de nascimento e localização. De seguida são questionados os valores do peso e da altura, sendo por fim registados os dados de acesso à aplicação (email, senha e nome de usuário) assim que aceites os termos e condições da *app*.

Assim que o utilizador completa o processo de registo na aplicação, este tem um pequeno tutorial de modo a ser introduzido nas funcionalidades principais da aplicação. Entre elas destacam-se:

- Uma área de registo diário da sua alimentação, subdividindo os alimentos pelos tipos de refeições diárias (pequeno-almoço, lanches, almoço e jantar). Nesta área é então possível adicionar os alimentos consumidos, procurando por eles numa base de dados dinâmica, sendo também possível adicionar informações relativas ao consumo de água e contadores de passos;
- Uma funcionalidade que contém o resumo do progresso do utilizador, indicando as flutuações de peso do mesmo;
- Uma área relativa à criação de metas pessoais relativamente ao peso, a níveis de nutrição ou até de atividade física;

- Uma funcionalidade que possuí os dados nutricionais registados, indicando os valores de nutrientes consumidos e metas previstas para os mesmos;
- Uma área para a adição de alimentos, refeições e receitas personalizadas pelo utilizador;
- Áreas de ligação a aparelhos de monitorização de passos e de recomendações de aplicações parceiras;
- Sistemas comunitários como fóruns em que utilizadores podem compartilhar experiências ou receitas, sistemas de amizade entre utilizadores e consequente sistema de mensagens entre os mesmos.

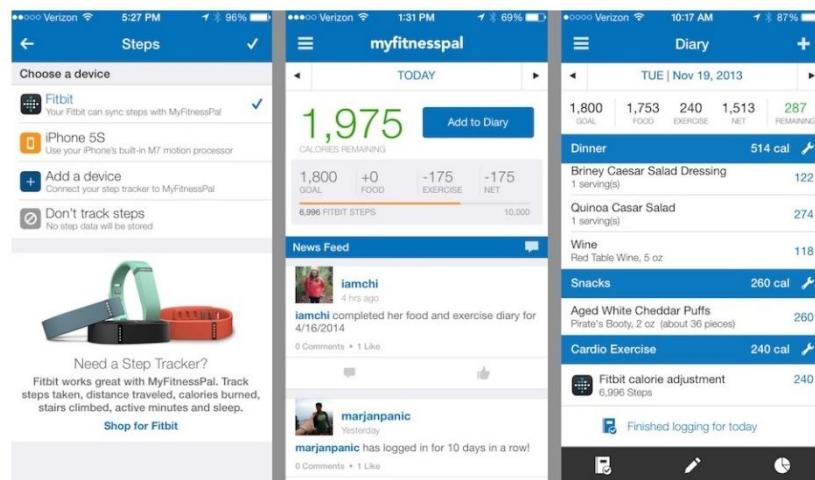


Figura 7 - Exemplos de funcionalidades do MyFitnessPal (adaptado de [14])

O nível de detalhe destas funcionalidades está dependente do estatuto do cliente, sendo que um utilizador subscrito ao serviço *premium* da aplicação possui mais benefícios que um utilizador comum, quer seja a nível do detalhe nutricional presente na análise dos alimentos consumidos, ou até à remoção dos anúncios presentes na aplicação.

2.1.4. Comparações entre plataformas

Todas as plataformas descritas apresentam diversas funcionalidades que permitem ao utilizador uma melhor gestão e acompanhamento da sua jornada nutricional, o que lhes permitiu estabelecerem-se fortemente neste mercado angariando milhões de *downloads*. Na tabela seguinte irá ser feita uma comparação entre as aplicações descritas e a aplicação *web* desenvolvida e detalhada neste relatório.

Tabela 1 - Comparação entre a web-app desenvolvida e as 3 apps mencionadas neste capítulo

<i>Funcionalidade</i>	<i>Healthy You (web-app)</i>	<i>MyFitnessPal</i>	<i>LifeSum</i>	<i>Yazio</i>
<i>Sistema de registo/login</i>	x	x	x	x
<i>Registo dos alimentos consumidos num determinado dia</i>	x	x	x	x
<i>Registo da alteração de peso do utilizador</i>	x	x	x	x
<i>Área de criação de objetivos pessoais (peso, calorias diárias, etc.)</i>		x	x	x
<i>Adição de alimentos / pratos customizados</i>	x	x	x	x
<i>Área de registo de refeições consumidas e dados das mesmas</i>	x	x	x	x
<i>Área de recomendações de aplicações parceiras</i>		x		x
<i>Fóruns comunitários e sistema de amizade</i>		x		
<i>Secção de adição de atividade física</i>		x	x	x
<i>Secção de registo do estado emocional</i>		x		x
<i>Sistema de planos nutricionais</i>	x	x	x	x

<i>Sistema de desafios nutricionais</i>	x	x
<i>Sistema de Chat entre nutricionista e utilizador</i>	x	
<i>Sistema de escolha de um utilizador por parte do nutricionista</i>	x	

A partir desta tabela é possível concluir que a *web-app* desenvolvida apresenta algumas funcionalidades que não estão presentes nas outras aplicações estudadas, sendo a principal diferença a inclusão de funcionalidades orientadas ao nutricionista, visto que as aplicações estudadas são mais orientadas ao utilizador comum. Porém, também é possível verificar que a *web-app* não contém algumas das funcionalidades inclusas nas aplicações descritas, sendo que maior parte dessas funcionalidades têm interesse e poderiam ser desenvolvidas no futuro (a existência de uma área de objetivos pessoais, um sistema de fórum entre utilizadores, entre outras).

2.2. Tecnologias existentes

De modo a ser possível a realização deste projeto, foi necessária a adoção de tecnologias orientadas ao desenvolvimento de aplicações *web*. Nesta secção irão ser descritas e analisadas as *frameworks*, bibliotecas e plataformas utilizadas, bem como a linguagem em que este projeto foi escrito.

Este projeto foi desenvolvido no editor de código-fonte *Visual Studio Code* utilizando a linguagem *JavaScript* para o desenvolvimento de ambas a *backend* e *frontend*, tendo sido utilizada também a linguagem *CSS* na *frontend*. A persistência de dados realizada a partir da *backend* foi realizada através do uso da biblioteca *mongoose*, dados esses estruturados através de *schemas*. Os dados são guardados numa base de dados *mongoDB*. A troca de dados entre a *backend* e a *frontend* é feita através de pedidos *HTTP* utilizando dados no formato *JSON*. É também utilizado o *node.js* como ambiente de tempo de execução e a *framework*

express.js para a construção da aplicação *web*, sendo posteriormente usadas diversas bibliotecas para a criação de interfaces na *frontend*, sendo a principal das bibliotecas o *React*.

2.2.1. JavaScript

O *JavaScript* é uma linguagem de programação de *script* em alto nível baseada no *ECMAScript* [15] uma das tecnologias principais da *World Wide Web* visto que mais de 97% dos sites a usam em *cliente-side*, tornando-a na linguagem de programação principal em *browsers* [16].

Esta linguagem de programação foi criada em 1995 pelo programador Brendan Eich quando este se encontrava pela *NetScape* [17]. Brendan Eich foi contratado pela *NetScape* com o objetivo de criar uma linguagem que possibilitasse a existência de uma versão mais dinâmica de *browsers*, algo que à data ainda era muito estático. Para a criação do *JavaScript*, Brendan Eich retirou inspiração de outras linguagens, nomeadamente à sintaxe e valores primitivos *versus* objetos do *Java*, às funções de primeira classe do *Scheme* e *AWK*, à herança prototípica do *Self* e finalmente às *strings*, *arrays* e expressões regulares do *Perl* e do *Python* [18].

O *JavaScript* apresenta diversas características importantes, sendo elas comuns a todas as implementações em conformidade com o *ECMAScript*. Entre elas destacam-se [18] [19]:

- Ser baseada em objetos;
- Ser uma linguagem funcional, apresentando funções de primeira classe, funções *nested* ou fechamentos;
- Ser uma linguagem dinâmica em que propriedades de objetos e variáveis podem guardar um valor de qualquer tipo, em que a variável *y* pode ser associada a um número, e mais tarde associada a uma *string*;
- Possui suporte universal, sendo então utilizada por todos os navegadores *web* modernos e populares.

O uso principal do *JavaScript* é na escrita de funções incluídas em páginas *HTML* que interagem com o Modelo de Objeto de Documentos (*DOM*) da mesma, sendo que todos os browsers de topo contêm um *JavaScript Engine* que executa o código na máquina do utilizador [20].

A quantidade e qualidade de bibliotecas e *frameworks* do *JavaScript* são um dos seus pontos fortes, sendo que 80% dos sites usam uma biblioteca/*framework* desenvolvida por terceiros. A mais popular destas é a *JQuery*, usada por 75% dos websites, existindo também o *React*

(criada e usada pelo *Facebook* e mais detalhada na secção 2.2.5) que é usada pelo *Twitter*, e o *Angular* que foi criada pela *Google* para ser usada nos seus *websites* [21].

Isto tudo levou a que o *JavaScript* se tornasse na linguagem de programação mais usada do mundo por desenvolvedores de *software*, dados esses presentes no inquérito de 2020 do *StackOverflow* [22].

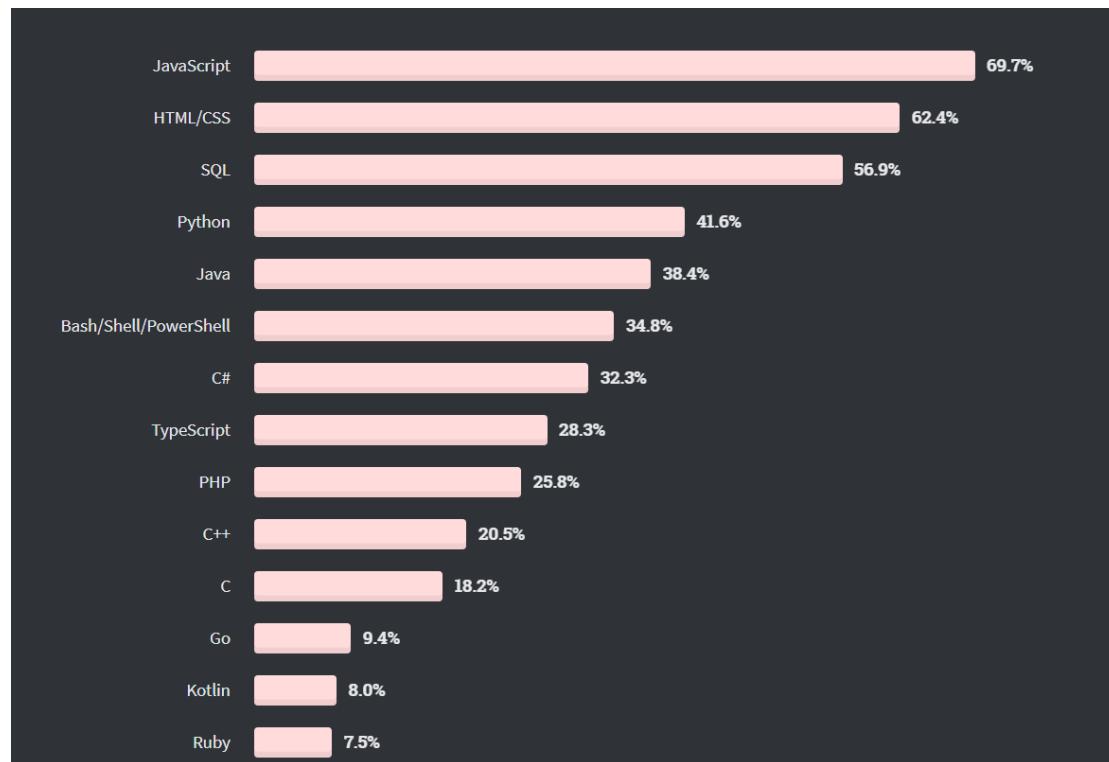


Gráfico 1 - Linguagens mais utilizadas dos utilizadores do StackOverflow (adaptado de [22])

2.2.2. Bases de dados

Uma base de dados é uma coleção organizada de informação estruturada ou dados sendo ela guardada eletronicamente num sistema computacional [23]. Estas coleções são muitas vezes classificadas com base na sua estrutura organizacional, havendo vários tipos de bases de dados como as bases de dado relacionais, as bases de dados distribuídas, as bases de dados orientadas a objetos ou até as bases de dados *NOSQL*, entre as quais se encontra a utilizada neste projeto, a *MongoDB* [24].

A *MongoDB* é uma base de dados orientada a documentos com código aberto, tendo sido desenvolvida na linguagem *C++* [25]. Esta base de dados apresenta diversas características, entre elas o suporte de consultas *ad hoc*, a indexação ou a replicação [26]. A *MongoDB* funciona através do uso de documentos semelhantes a *JSON's* com *Schemas* opcionais.

JavaScript Object Notation (mais conhecido por *JSON*) é um formato leve, compacto e intuitivo, sendo o ideal para fazer uma troca de dados entre sistemas. Devido às suas propriedades, ele é completamente independente de linguagem usando convenções familiares a linguagens como *C*, *Java*, *Python* entre outras [27].

Um *Schema* é um objeto *JSON* que permite a definição do formato e conteúdo dos documentos que irão ser inseridos na base de dados. Estes também são utilizados para a definição de campos e o seu conteúdo, mas também na validação caso haja mudanças em documentos [28].

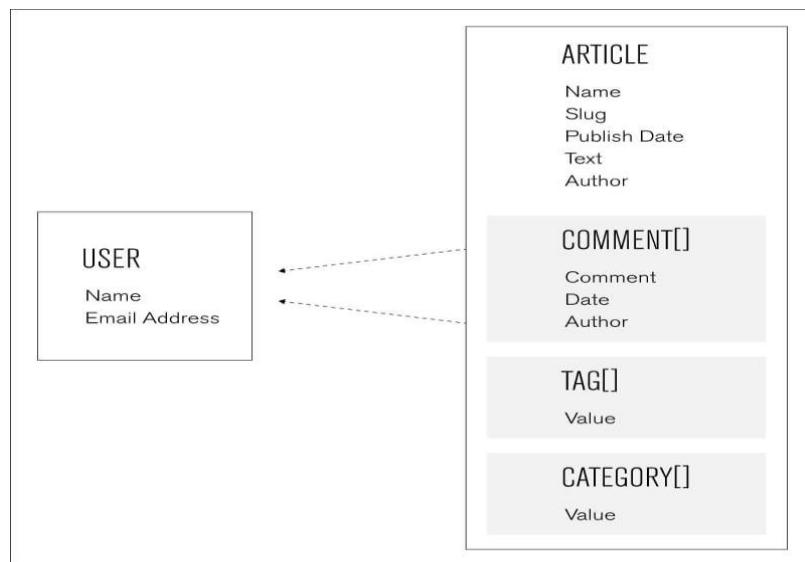


Figura 8 - Exemplo de um Schema da MongoDB (retirado de [29])

2.2.3. Node.js e Express.js

De modo a construir aplicações de rede facilmente escaláveis, foi necessário o desenvolvimento de um ambiente de execução assíncrono orientado a objetos, o *Node.js* [30].



Figura 9 - Logo oficial do Node.js (retirado de [31])

Geralmente, o modelo de concorrência mais utilizado é através de *threads* no sistema operativo. Aplicações de rede baseadas neste tipo de modelos acabam por se tornar ineficientes e complicadas de serem utilizadas, levando também *deadlocks* de processos. No caso do *Node.js* esses problemas são inexistentes devido à forma como o mesmo funciona. No *Node.js* praticamente não se usam funções que realizam diretamente operações de entrada/saída, levando a que os processos não bloqueiem, o que por sua vez facilita o desenvolvimento de sistemas escaláveis [30].

Este é um *software* de código aberto criado por Ryan Dahl em 2009 [32] tendo sido diretamente influenciado por sistemas como o *Event Machine* do *Ruby* ou pelo *Twisted* do *Python* cuja maior diferença relativamente a estes é a exposição do *Event Loop* como uma parte do ambiente de execução opondo a sua utilização como uma biblioteca [30] o que faz com que não haja a necessidade de criar novas *threads* que potencialmente consumam mais recursos computacionais [33].

O *Node.js* apresenta diversas vantagens como [33]

- A sua flexibilidade, que vem na forma do *NPM (Node Package Manager)* que é o maior repositório de softwares do mundo e um gerenciador de pacotes sendo um deles o *Express.js*;
- Ser bastante leve, devido a não exigir muitos recursos computacionais em comparação às tecnologias mais tradicionais, especialmente quando usado com ferramentas como o *Docker*;
- O seu uso leva a uma maior produtividade visto o fornecimento gratuito de pacotes reutilizáveis por parte do *NPM* e pelo uso da mesma linguagem nas componentes de *frontend* e *backend*.

O *Node.js* apresenta diversas vantagens o que faz com que bastantes empresas de topo mundiais o usem. Esta ferramenta é utilizada por nomes como a *Netflix*, o *Uber*, o *EBay*, o *Twitter* ou a própria *NASA* [34]. Através de um questionário realizado pelo *StackOverflow* foi também possível concluir que esta é a ferramenta mais utilizada no mundo, sendo usada por metade dos participantes do inquérito e estando à frente do *.NET* e *.NET Core* [35].

De modo a construir aplicações web e *APIs*, em conjunto com o *Node.js*, é também necessária uma framework de servidores, sendo a mais utilizada a *Express.js* [36]. Esta é uma framework lançada em 2010 por TJ Holowaychuk inspirada pelo software *Sinatra* [37]. Esta ferramenta é utilizada por grandes empresas entre elas a *Uber*, o *PayPal*, a *IBM* ou a *Fox Sports* [38].

2.2.4. React

Em 2013 uma das grandes empresas tecnológicas, o *Facebook*, decidiu produzir e lançar uma ferramenta de *JavaScript* de código aberto para a criação de *UI's* em páginas *web*, e essa solução era o *React*. [39] Mais tarde viria a ser adicionado um módulo complementar, o *React Native*, possibilitando o desenvolvimento de aplicações para *Android* e *iOs* sem necessitar de um *browser*, correndo numa instância embutida *JavaScriptCore* [40].

O *React* (também conhecido por *ReactJS*) funciona através da utilização de um *DOM* virtual, fazendo as mudanças pretendidas no mesmo que, posteriormente, encontra a maneira mais eficiente de atualizar o *DOM* do *browser* [41]. Esta viria também a ser a biblioteca a popularizar o conceito de *component-based architectures*, possuindo diversas vantagens como [42]:

- Componentes modulares e coesos, sendo altamente reusáveis o que contribui para um menor tempo de desenvolvimento;
- Compatibilidade com desenvolvimento *mobile* permitindo aos desenvolvedores o reuso da parte lógica, apenas necessitando do ajuste da visualização;
- Fácil manutenção devido aos componentes serem independentes.

Nos dias correntes o *React* e o *React Native* são das bibliotecas mais usadas em *JavaScript*, sendo usadas em sites como o próprio *Facebook*, o *Yahoo*, o *PayPal* ou o *Discord*. [43]

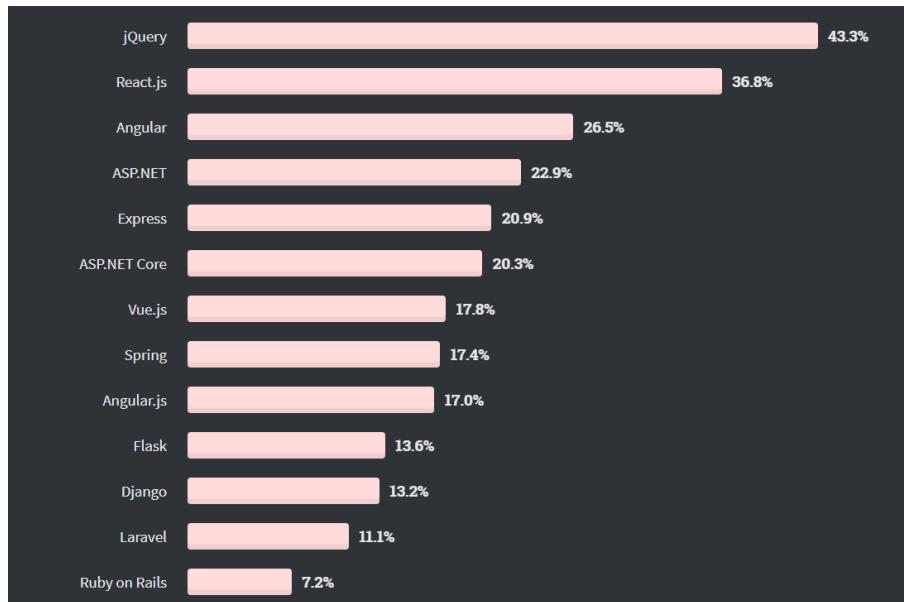


Gráfico 2 - *Web Frameworks mais utilizadas dos utilizadores do StackOverflow (adaptado de [44])*

Adicionalmente, foi realizado um questionário pelo *StackOverflow* que conclui que esta é a segunda *Web Framework* mais utilizada pelos seus utilizadores, estando apenas atrás da *jQuery*, sendo também uma das *frameworks* mais amadas pela sua base de utilizadores [44].

2.2.5. JavaScript vs C#

Para o desenvolvimento da componente da *BackEnd* da aplicação web foi considerado o uso de outra linguagem de programação, nomeadamente o *C#*.

Esta é uma linguagem desenvolvimento moderna e orientada a objetos, que foi desenvolvida pela *Microsoft* no ano de 2000 [45]. Esta linguagem permite o desenvolvimento de aplicações robustas e seguras que consigam rodar no *.NET* tendo sido concebida com base na família de linguagens de programação *C*, sendo familiar a linguagens como *C* ou *C++* [45].

O *C#* apresenta diversas características, entre elas o facto de [46]:

- Ser uma linguagem simplificada, que não usa alguns dos fatores mais complexos das linguagens da mesma família como os *pointers* ou os operadores “::” e “->”;
- Ser *Type Safe*, prevenindo conversões de variáveis inseguras e verificando *overflows*;

- Ser orientada a objetos, permitindo encapsulamento de dados, polimorfismo, interfaces entre outros.

Foi então feita uma comparação entre a linguagem previamente descrita na secção 2.2.2 *JavaScript* e a analisada nesta secção *C#* de modo a concluir a que iria ser utilizada no desenvolvimento da *backend*, comparação que está presente na tabela 2 [47] [48].

Tabela 2 - Comparação entre as propriedades do JavaScript e as propriedades do C#

<i>Aspetto em comparação</i>	<i>JavaScript</i>	<i>C#</i>
<i>Tipo de Linguagem</i>	Linguagem de script	Linguagem baseada em objetos
<i>Tipagem</i>	Tipagem dinâmica	Tipagem estática
<i>Sintaxe</i>	Baseada no <i>HTML (OBSL)</i>	Semelhante a sintaxes <i>OOP</i>
<i>Thread</i>	<i>Single-Thread</i>	<i>Multi-Thread</i>
<i>Compilação</i>	Não necessita de compilação sendo corrida em browser	O código necessita de compilação, resultando em ficheiros executáveis

Foram também analisadas outras componentes das linguagens referidas, verificando qual das linguagens é melhor para cada uma dessas componentes o que pode ser visto na tabela 3 [49].

Tabela 3 - Comparação entre as componentes do JavaScript e as componentes do C#

<i>Aspetto em comparação</i>	<i>JavaScript</i>	<i>C#</i>
<i>Estabilidade</i>		x
<i>Popularidade</i>	x	
<i>Sintaxe</i>		x
<i>Familiaridade com a linguagem</i>	x	

<i>Comunidade</i>	X
<i>Performance</i>	X

Após análise das vantagens e desvantagens de cada linguagem de programação, foi possível concluir que se tratava de 2 linguagens de topo de qualidade equiparável, tendo-se decidido optar pelo uso da linguagem *JavaScript*.

2.2.6. React vs Angular

Para o desenvolvimento da componente da *FrontEnd* da aplicação web foi considerado o uso de outra *framework*, nomeadamente o *Angular*.

O *Angular* é uma *framework* baseada na linguagem *TypeScript* utilizada para o desenvolvimento de aplicações *web*. Esta plataforma tem como características [50] [51]:

- Ser uma *framework* baseada em componentes o que permite a construção de aplicações *web* com um alto escalamento;
- Possuir uma vasta coleção de librarias integradas que providenciam diversas funções como as comunicações entre o cliente e o *server* ou o *routing*;
- Suporta a injeção de dependências de modo a aumentar a flexibilidade e modularidade das aplicações;
- Usa um *DOM* real que é atualizado após qualquer mudança na sua estrutura de dados.

Esta *framework* é das mais utilizadas no mundo para o desenvolvimento de aplicações *web*, sendo apenas ultrapassada pelo *React* e pelo *jQuery* num dos inquéritos anuais do *StackOverflow* [43].

Foi então feita uma comparação entre a *framework* previamente descrita na secção 2.2.5 *React*, e a que foi analisada nesta secção *Angular* de modo a concluir a que iria ser utilizada no desenvolvimento da *frontend*, comparação que está presente na tabela 4 [42] [52].

Tabela 4 - Comparação entre as componentes do React e as componentes do Angular

<i>Aspetto em comparação</i>	<i>React</i>	<i>Angular</i>
<i>Possui routing nativo</i>		x
<i>Popularidade</i>	x	
<i>Tratamento de erros</i>		x
<i>Familiaridade com a framework/biblioteca</i>	x	
<i>Performance</i>	x	

Após análise das vantagens e desvantagens de cada *framework*, foi possível concluir que se trata de 2 *frameworks* de grande qualidade. Contudo, as vantagens oferecidas pelo *React* quando comparado com o *Angular*, pesaram mais na escolha entre as duas tecnologias.

2.2.7. Outras bibliotecas

Na *frontend* foi necessário o uso de diversas bibliotecas de modo a apresentar uma melhor visualização dos dados. As bibliotecas utilizadas foram:

- *Material-UI* usada em diversas componentes de visualização como botões, caixas de texto ou tabelas;
- *React-Bootstrap* usada em componentes de separação de secções na visualização como *lines* e *rows*;
- *react-router-dom* usada para realizar o roteamento entre as diversas funcionalidades da aplicação;
- *sweetalert2* usado para o *display* de alertas, informando o utilizador do sucesso das suas chamadas à *backend*;
- *react-chat-engine* usado na funcionalidade do *chat*, fazendo o *display* do mesmo;
- *react-chartjs-2* usado para a construção de diversos gráficos de mostragem de dados;
- *fullcalendar* usado no registo das refeições num certo espaço temporal;
- *Sinon* usada para a realização de testes unitários e de integração.

3. Análise e desenho da solução

Neste capítulo, irá ser feita uma análise e o desenho da solução implementada. Este está subdividido em 3 secções. Primeiramente, é feita uma especificação dos conceitos do domínio do problema com recurso a artefactos adequados. De seguida são especificados os requisitos funcionais e não funcionais do sistema. Por fim, é feito o desenho da solução apresentando todos os diagramas necessários sobre o mesmo.

3.1. Domínio do problema

Após uma análise à proposta realizada inicialmente e às aplicações semelhantes ao projeto a ser desenvolvido, foi permitida a elaboração dos casos de uso que iriam ser criados e o respetivo modelo domínio necessário. O modelo de domínio está representado na figura 74 que pode ser vista no Anexo A.

Após feita a identificação dos conceitos necessários, foi necessário fazer uma especificação dos mesmos:

- *User* representa um utilizador da web-app, sendo que este poderá ser um *Nutritionist* ou um *Client*. Um utilizador irá possuir um *id*, um nome, um email, uma password e o seu tipo de utilizador sendo estes atributos comuns à classe *Client* e *Nutritionist*;
- Um *Client* é o paciente que irá usar a web-app com o fim de melhorar o seu estado nutricional, fazendo-o através dos casos de uso detalhados na secção 3.2. Para além dos atributos comuns com a classe *User*, um cliente irá ter adicionalmente uma altura associada, uma idade, um género e uma lista com o seu registo de pesos diários. Este terá também um atributo booleano que define se já tem um nutricionista associado;
- Um *Nutritionist* é o utilizador do sistema que irá usar a web-app para ajudar os clientes que se registam nela, fazendo-o através da conceção de planos ou do acompanhamento dos clientes. Para além dos atributos comuns com a classe *User*, este irá ter também uma lista de clientes;
- *Weight* representa uma instância de peso registada por um cliente, tendo associada a ele um *id*, uma data e o valor do peso registado;

- Um *ClientRequest* é um pedido feito por parte do nutricionista para o acompanhamento de um cliente, tendo como atributos o seu *id*, o email do cliente, o email do nutricionista e um atributo booleano que define se o pedido já foi tratado;
- Um *Challenge* é um desafio nutricional que pode ser realizado por um cliente (como por exemplo, não consumir chocolate durante uma semana). Um desafio nutricional tem como atributo um *id*, um nome, uma descrição, o *id* da comida presente no desafio, a quantidade dessa comida que pode ser consumida e a duração em dias do desafio;
- O *ChallengeRegistry* é o registo de um desafio nutricional ativo ou completo por parte do cliente, tendo como atributos um *id*, o *id* do desafio nutricional em questão, o *id* do cliente que está a realizar o desafio, as datas de início e de fim do desafio e atributos booleanos que indicam se o desafio foi falhado e se a duração do mesmo já expirou (estando assim completo);
- Um *Plan* é um plano nutricional criado por um nutricionista e que será seguido pelo cliente. Este plano irá ter como atributos um *id*, um nome, o email do cliente que o está a seguir, as datas de início e de fim, o email do nutricionista que o criou e duas listas em que uma irá conter as refeições planeadas pelo nutricionista e a outra irá conter as refeições realizadas/consumidas pelo cliente;
- Uma *Meal* irá ser uma refeição registada que faz parte de um plano, tendo sido ela ou planeada por um nutricionista ou consumida por um cliente. Uma refeição irá ter um *id*, o *id* do plano a que diz respeito, o *id* do alimento/prato a que a refeição consiste, a porção consumida e a data da refeição;
- Uma *Food* irá ser um alimento presente na base de dados ou um prato customizado criado por um cliente, que irá fazer parte de uma refeição. A *Food* irá ter como atributos o *id*, o nome, os carboidratos, a gordura, a proteína, o açúcar e as calorias do alimento/comida. Também tem um atributo booleano que irá indicar se a comida foi criada manualmente, e caso afirmativo irá possuir uma lista com os ids dos alimentos que a compõem.

3.2. Requisitos funcionais e não funcionais

3.2.1. Requisitos funcionais

Nesta secção são descritos os requisitos funcionais propostos, em forma de casos de uso, e os atores identificados após uma análise ao problema proposto.

Os atores identificados são:

- O *nutricionista* que faz o acompanhamento dos clientes;
- O *cliente* que usa as funcionalidades da aplicação para melhorar o seu estado nutricional;
- O *utilizador não registado* que tem que fazer o registo na aplicação, tornando-se posteriormente num *cliente* ou num *nutricionista*.

Tendo em conta estes atores e o problema proposto, foram criados os casos de uso presentes na figura 75, que pode ser consultada no Anexo A.

Todos os atores têm acesso a funcionalidades diferentes, à exceção do *chat* e do login que são partilhadas entre nutricionistas e clientes.

De seguida são especificados os casos de uso presentes na figura 75, assumindo um cenário de sucesso:

- UC1 – O utilizador não registado pretende registar-se na aplicação. É-lhe perguntado se pretende registar-se como nutricionista ou como cliente. Caso pretenda ser registado como nutricionista, o utilizador não registado introduz o seu nome, email e password e, após leitura, aceita os termos e condições da aplicação *web*, submetendo os seus dados. Caso pretenda ser registado como cliente, o utilizador não registado introduz o seu email, nome, password, altura, peso inicial, idade e género e, após leitura, aceita os termos e condições da aplicação *web*, submetendo os seus dados. Caso os dados submetidos sejam aceites, o utilizador não registado é registado na aplicação;
- UC2 – O nutricionista e o cliente pretendem fazer o login na aplicação. É-lhes pedido para introduzirem o email, a password, e para indicarem se são nutricionistas ou clientes. Caso os dados inseridos sejam válidos aquando da submissão, é feito o login;

- UC3 – O nutricionista pretende enviar um pedido de acompanhamento a um cliente. É-lhe mostrada uma lista com todos os clientes disponíveis e é-lhe pedido para selecionar um. Após seleção, é enviado o pedido ao cliente;
- UC4 – O nutricionista pretende criar um plano para um dos seus clientes. É-lhe pedido para introduzir o nome do plano, as datas de início e fim do mesmo, e para selecionar o cliente ao qual será atribuído. Caso os dados submetidos sejam aceites, é redirecionado para uma página com um calendário em que pode adicionar refeições ao plano;
- UC5 – O nutricionista pretende adicionar refeições planeadas a um plano. É-lhe pedido para selecionar o plano em questão. De seguida é-lhe mostrado um calendário onde ele pode introduzir as refeições planeadas, clicando na hora e data pretendida e selecionando o alimento/prato e porções pretendidas. As refeições são guardadas no plano;
- UC6 – O nutricionista pretende visualizar o estado em que se encontra o plano de um dos seus clientes. É-lhe pedido para selecionar qual plano ele pretende visualizar e, após seleção, são-lhe mostradas várias informações quer a nível de tabelas, como em gráficos;
- UC7 – O cliente pretende aceitar o pedido de acompanhamento enviado pelo nutricionista. É-lhe enviada uma notificação após o login que recebeu um pedido de acompanhamento por parte do nutricionista e, após análise do pedido, é-lhe perguntado se deseja aceitar ou não. Em caso de aceitação, o cliente fica vinculado ao nutricionista, sendo-lhe possível falar por *chat* com o mesmo e permitindo a criação do plano por parte do nutricionista;
- UC8 – O cliente pretende registar o seu peso diário. Na aba do perfil, tem uma caixa de texto para introduzir o seu peso e, após confirmação, o mesmo é submetido e atualizado. Caso já tenha introduzido um peso naquele dia, não lhe será permitido adicionar um novo peso;
- UC9 – O cliente pretende criar uma comida customizada. É-lhe pedido para introduzir o nome da comida. De seguida é-lhe pedido para introduzir, um a um, os alimentos que irão constituir a sua comida customizada e as suas porções. Quando termina de introduzir os alimentos, faz a submissão da comida customizada e ela é guardada na base de dados;
- UC10 – O cliente pretende registrar as refeições que consumiu num determinado dia. É-lhe mostrado um calendário no qual ele pode introduzir as refeições consumidas,

clicando na hora e data pretendida e selecionando o alimento consumido. As refeições consumidas são guardadas no seu plano;

- UC11– O cliente pretende ter acesso ao histórico das suas refeições consumidas e visualizar mensagens de apoio. É-lhe mostrada uma tabela que possui as refeições consumidas e as planeadas e, com base na sua performance, é-lhe mostrada uma mensagem indicatória do sucesso do seu plano;
- UC12 – O cliente pretende realizar desafios nutricionais. No seu perfil, são-lhe mostrados os desafios que ele já conclui, os disponíveis, e o que está ativo de momento. Caso não tenha nenhum ativo, ele pode selecionar um dos desafios disponíveis. Ao selecionar o desafio, ele tornar-se-á ativo;
- UC13 – O cliente e o nutricionista pretendem comunicar um com o outro. Após a aceitação do pedido de acompanhamento do nutricionista por parte do cliente, irá ser disponibilizada uma janela de *chat* entre os dois, que poderá ser consultada a qualquer momento. É-lhe permitido enviar mensagens, ficheiros, entre outros.

3.2.2. Requisitos não funcionais

Nesta secção são descritos os requisitos não funcionais propostos, sendo eles divididos em diversas categorias. A identificação dos requisitos não funcionais segue o modelo *FURPS+*. Este é um modelo que distingue os requisitos não funcionais em várias categorias, como a funcionalidade, usabilidade, confiabilidade (*reliability*), desempenho (*performance*) e suportabilidade. O elemento “+” engloba todas as outras categorias existentes como o *packaging* ou as questões legais [53].

Primeiramente, os requisitos não funcionais a nível da usabilidade são:

- A *UI* da aplicação precisa de ser *user-friendly*, possuindo uma estrutura e terminologias simples de modo que a web-app seja acessível a qualquer tipo de utilizador.

Foi também identificado um requisito não funcional na categoria da segurança:

- Os atores identificados na secção 3.2.1 apenas podem executar as funcionalidades a que eles estão atribuídos.

Quanto à categoria da compatibilidade, tem-se que:

- Esta aplicação necessita de ser compatível com *browsers* de internet.

No que toca à suportabilidade, foi identificado que:

- A arquitetura implementada no projeto deve permitir uma possível extensão para uma aplicação *mobile* no futuro.

Por fim, no que toca ao âmbito legal:

- A *web-app* precisa de adotar os processos que garantem a segurança e proteção de dados sendo que estes não poderão ser disponibilizados a terceiros para qualquer outro fim para além dos mencionados. Os seus termos e condições incluem:
 - A permissão da partilha das receitas personalizadas criadas pelos clientes com o resto dos utilizadores da aplicação;
 - O consentimento da partilha dos dados pessoais do cliente (como o email, altura, idade etc.) com o seu nutricionista de modo a permitir uma melhor avaliação nutricional do mesmo.

3.3. Desenho

Na realização do desenho da aplicação é utilizado o modelo de vidas de arquitetura 4+1 e o modelo C4.

O modelo de vidas 4+1 é um modelo usado para descrever o software com base em 4 vidas: vista lógica, vista de processos, vista de implementação e uma vista física. As vidas são definidas do seguinte modo [54] [55] [56]:

- Vista lógica – Vista relativa à funcionalidade que o sistema traz ao usuário final.
- Vista processos – Vista relativa às interações no sistema.
- Vista de implementação – Vista relativa à organização do sistema de um ponto de vista do programador.
- Vista física – Vista relativa à organização do sistema do ponto de vista de um engenheiro de sistemas.

O modelo C4 é uma técnica de modelagem da arquitetura de sistemas de software através da decomposição do seu sistema em componentes e contentores. Os níveis deste modelo são [56] [57]:

- Nível 1 – Descrição do contexto do sistema em si
- Nível 2 – Descrição dos contentores do sistema
- Nível 3 – Descrição dos componentes presentes em cada contentor
- Nível 4 – Descrição do código presente nos componentes (algo que não será abordado nesta secção)

De modo a fazer a representação do sistema de uma forma visual, é utilizada a linguagem de modelagem *UML*.

3.3.1. Vista Física

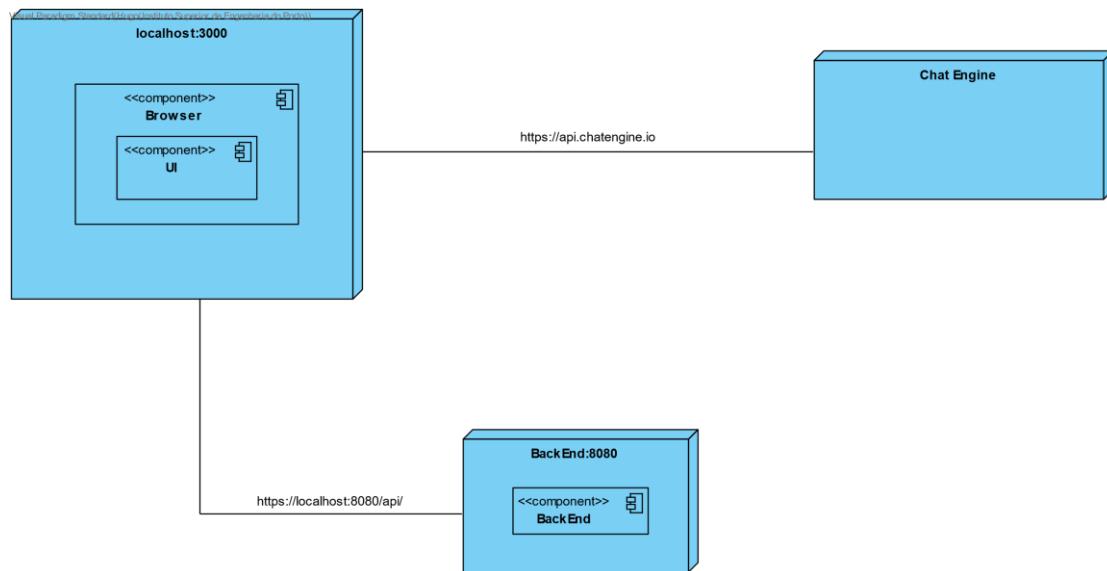


Figura 10 - Vista física (nível 2)

A aplicação desenvolvida possui um servidor correspondente à *backend* que comunica diretamente com um servidor que corresponde à *frontend* que, por sua vez, irá comunicar com a API *Chat Engine* que está responsável pela gestão dos *chats* e respetivas mensagens entre clientes e nutricionistas, o que pode ser visto na figura 10.

3.3.2. Vista Lógica

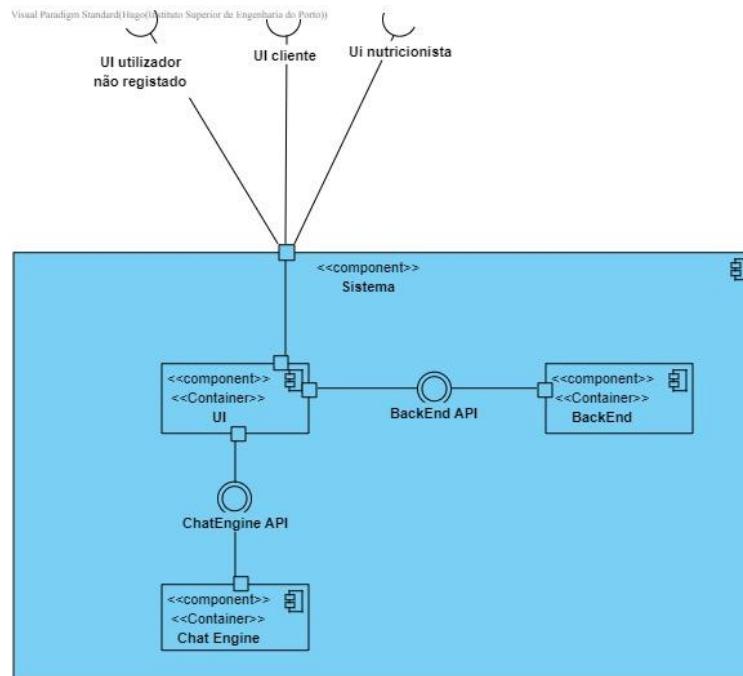


Figura 11 - Vista lógica (nível 2)

A aplicação web é um sistema com 3 componentes: uma *UI* que atua como *frontend*, uma *BackEnd* e o *Chat Engine*. A *UI* comunica com a *BackEnd* através da *BackEnd API*, e com o *Chat Engine* através da *ChatEngine API*. O sistema irá possuir *UI's* com opções diferentes dependendo do Ator que está a utilizar a *web-app*, o que pode ser visto na figura 11.

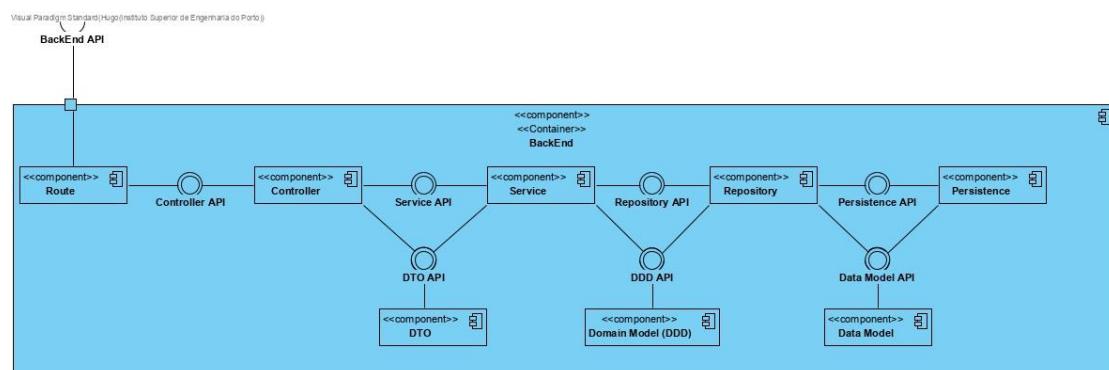


Figura 12 - Vista lógica BackEnd (nível 3)

O container *BackEnd* possui diversos componentes. A *BackEnd API* comunica diretamente com a componente dos *Routes*, componente essa que se liga ao *Controller* através da *Controller API*. O *Controller* comunica-se com o *Service* através da *Service API*, sendo que o *Service* comunica-se por sua vez com o *Repository* através da *Repository API*. Finalmente, o *Repository* liga-se à *Persistence* através da *Persistence API*. Existem também componentes de *DTO's* (que se ligam aos *Controllers* e *Services*), *DDD's* (que se ligam a *Services* e *Repositories*) e *Data Models* (que se ligam a *Repositories* e à *Persistence*). Esta vista lógica pode ser vista na figura 12.

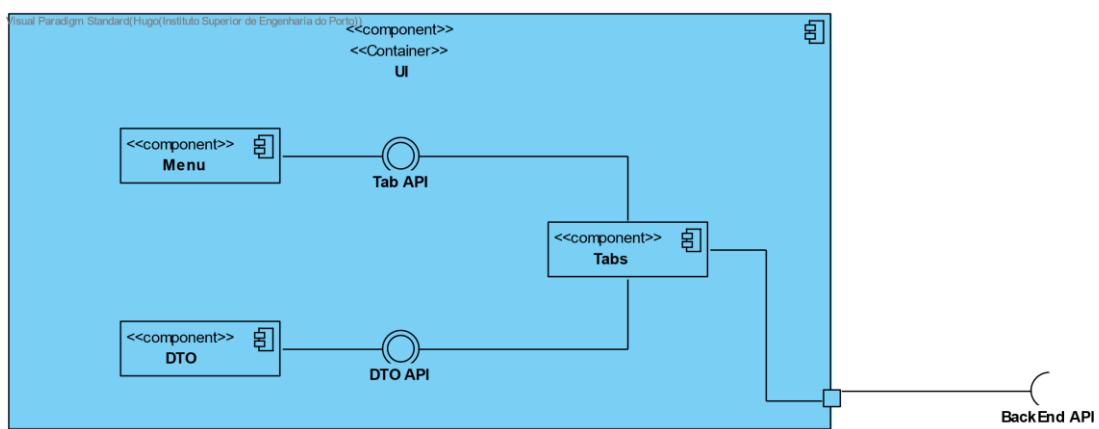


Figura 13 - Vista lógica UI (nível 3)

O container da *UI* possui 3 componentes, incluindo um *Menu* que irá comunicar com as *Tabs* através da *Tab API*. Essas *Tabs* por sua vez geram *DTO's* através da *DTO API*, e por fim comunicam com a *BackEnd API*, o que pode ser visto na figura 13.

3.3.3. Vista de Processos

3.3.3.1. UC1 – Registro na aplicação

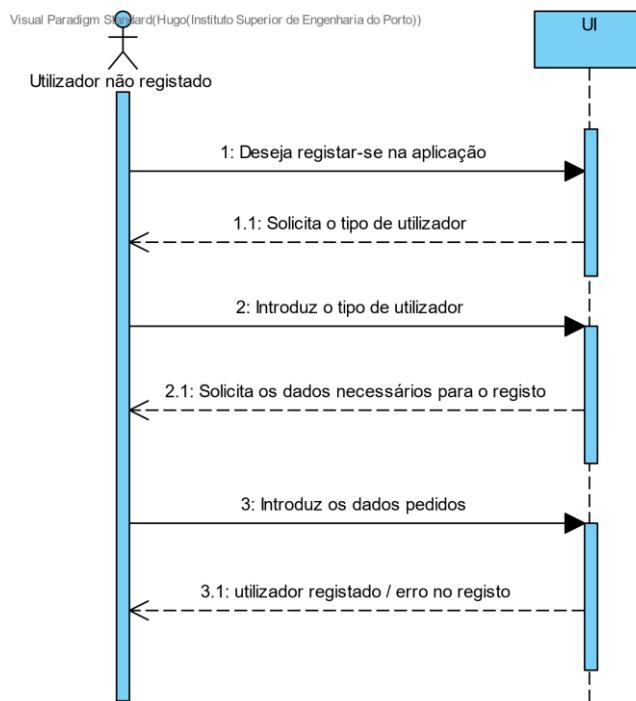


Figura 14 - SSD do caso de uso 1

Neste caso de uso, um utilizador não registrado deseja registar-se na aplicação. Este começa por selecionar o seu tipo de utilizador (Cliente ou Nutricionista) e introduzir os dados solicitados. Posteriormente são-lhe mostrados os termos e condições da aplicação e é-lhe pedida uma confirmação. Após confirmação, os dados registados do cliente são enviados através de um *POST* para a *UserRoute*. Este comunica com o *UserController*, enviando-lhe os dados recebidos para tratamento. Dependendo do tipo de utilizador a registar, o caso de uso pode seguir dois fluxos no ponto 4.1.1 do *SD* representado na figura 76, que pode ser visto no Anexo A.

No caso de o tipo de utilizador ser um nutricionista, é criado um *DTO* a partir dos dados iniciais, através do *NutritionistMap*. Este *DTO* é enviado ao *NutritionistService* que, a partir dele, cria um objeto de domínio *Nutritionist*, sendo ele por sua vez enviado ao *NutritionistRepo* para ser persistido na base de dados. O caso de uso retorna à *UI* um *DTO* do nutricionista criado ou erro.

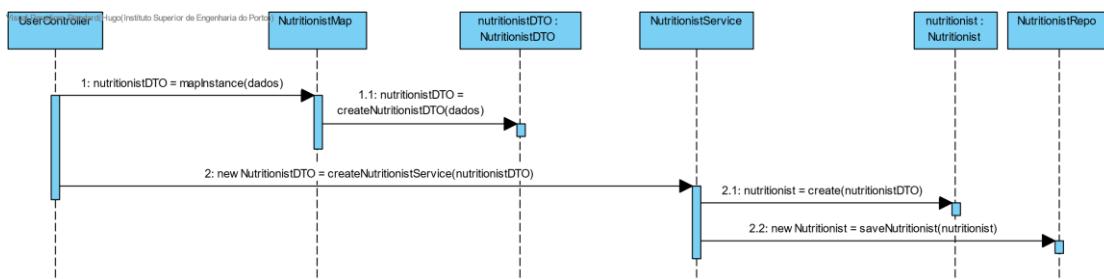


Figura 15 - Secção do SD do caso de uso 1 relativa ao nutricionista

No caso de o tipo de utilizador ser um cliente, o processo é semelhante ao do nutricionista, sendo que as diferenças são nas classes utilizadas e na adição da criação de um *DTO* relativo ao *Weight*, que é incorporado no objeto de domínio do *Client*.

3.3.3.2. UC2 – Login na aplicação

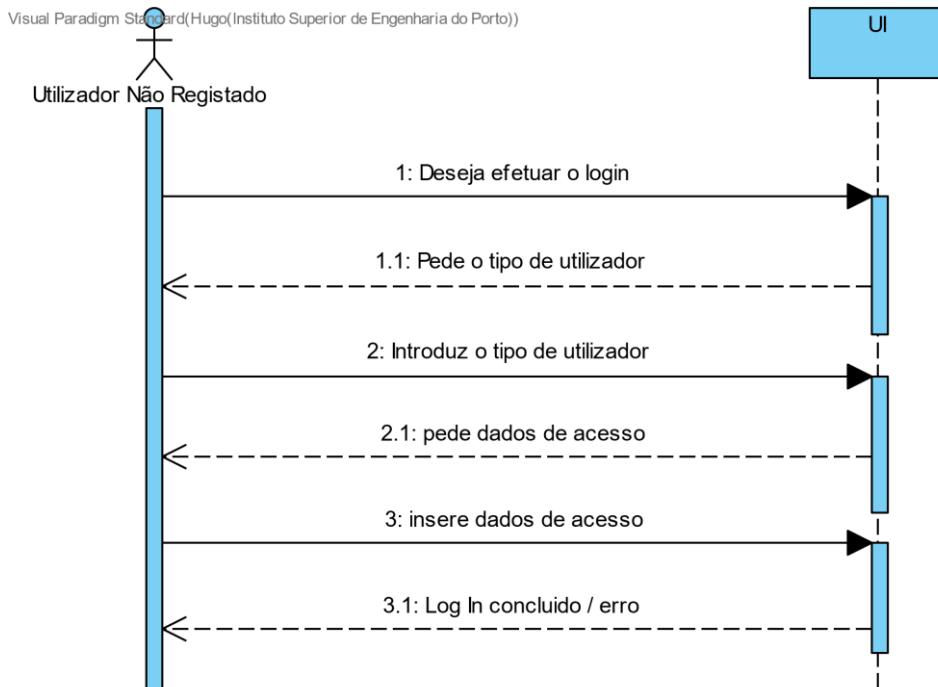


Figura 16 - SSD do caso de uso 2

Neste caso de uso, o utilizador não registrado deseja realizar o *login* na aplicação. Primeiro, é-lhe pedido o seu tipo de utilizador e após inserir o mesmo são-lhe pedidos os dados de acesso. Os dados inseridos são então enviados para a *UserRoute* através de um pedido *REST*, sendo

enviados posteriormente para o *UserController*, como pode ser visto no *SD* da figura 17. Aqui, dependendo do tipo de utilizador, o caso de uso pode seguir dois fluxos.

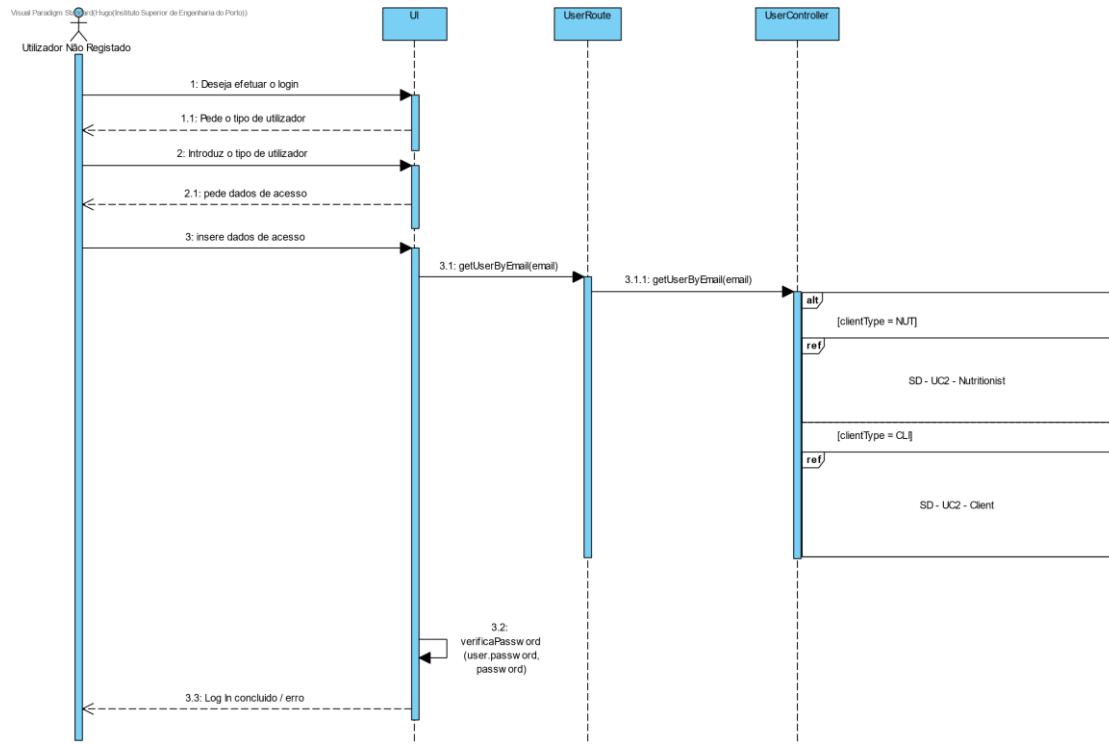


Figura 17 - SD do caso de uso 2

Caso o user seja um nutricionista, o email do mesmo será enviado para o *NutritionistService* que posteriormente irá comunicar com o *NutritionistRepo* para este recolher o objeto do nutricionista da base de dados. Este objeto é retornado à *UI* e é então verificado se a palavra-passe introduzida pelo utilizador não registado coincide com a palavra-passe do objeto retornado. Caso a palavra-passe seja a correta, o utilizador não registado efetua o *login*, mudando de *role*, caso contrário é apresentado um erro. O caso do nutricionista pode ser visto na figura 18.

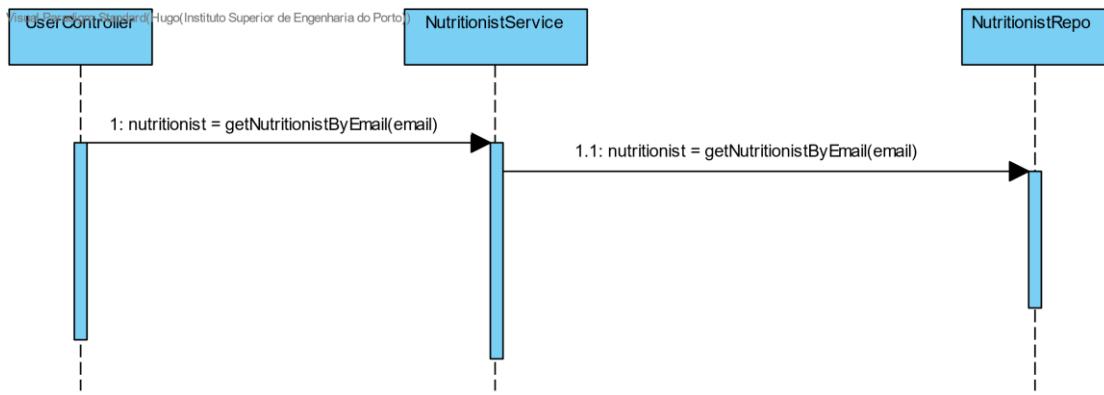


Figura 18 - Secção do SD do caso de uso 1 relativa ao nutricionista

No caso de o tipo de utilizador ser um cliente, o processo é semelhante ao do nutricionista, sendo a única diferença as classes utilizadas.

3.3.3.3. UC3 – Enviar pedido de acompanhamento

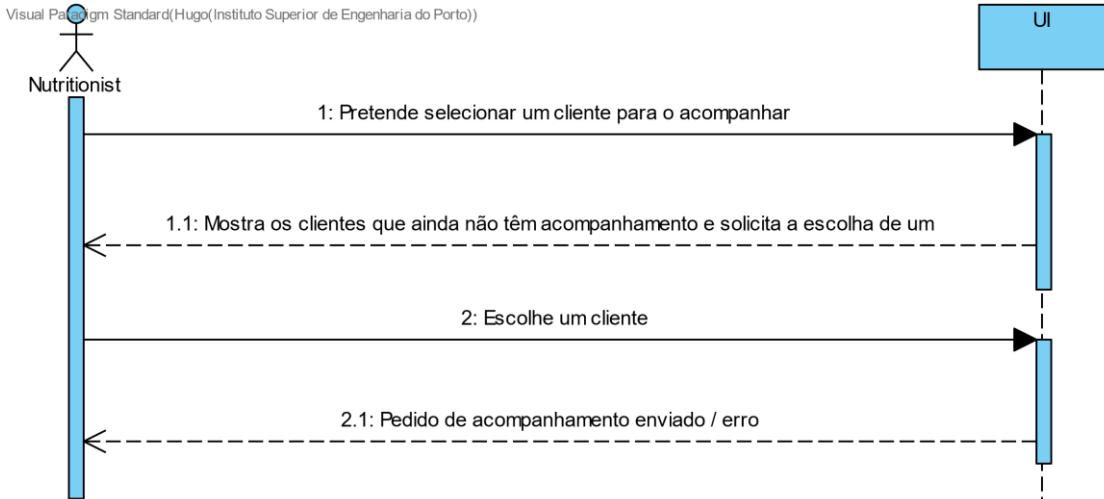


Figura 19 - SSD do caso de uso 3

Neste caso de uso o nutricionista pretende selecionar um cliente para realizar o seu acompanhamento. O caso de uso começa por um pedido *GET* à *UserRoute*, no qual se obtém uma lista dos clientes que ainda não possuem um nutricionista associado. Estes clientes são posteriormente mostrados na *UI*, onde é solicitado ao nutricionista para escolher um. Após a sua escolha, é criado um *DTO* relativo ao pedido que é enviado para a *ClientRequestRoute*. De seguida este é enviado ao *ClientRequestController*, que comunica com o *ClientRequestService* onde é criado o objeto de domínio *ClientRequest*. Este objeto é persistido na base de dados

no *ClientRequestRepo*. A *BackEnd* retorna à *UI* o pedido criado ou erro. O *SD* deste caso uso está presente na figura 77 que pode ser consultada no Anexo A.

3.3.3.4. UC4 – Criar plano alimentar

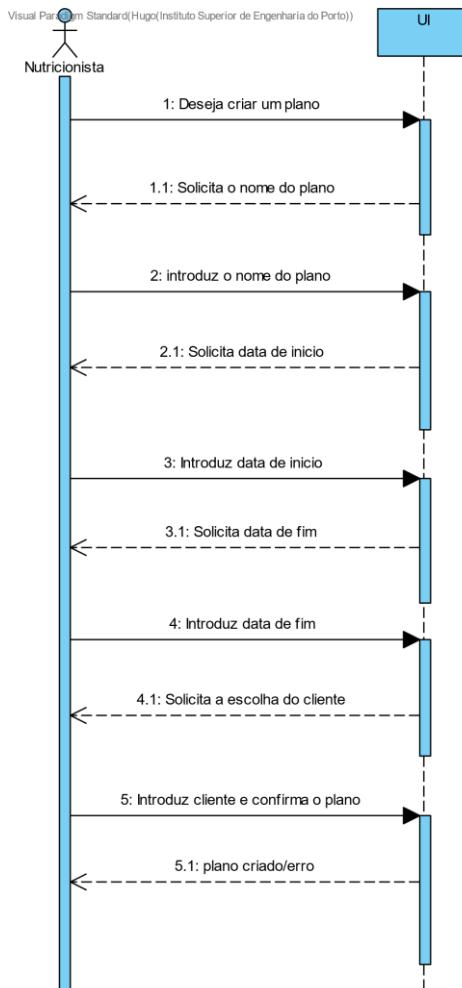


Figura 20 - SSD do caso de uso 4

Neste caso de uso o nutricionista deseja criar um plano alimentar. Este começa por introduzir informações relativas ao plano, como o nome do mesmo, a data de início e a data de fim. Depois é feito um pedido *GET* à *UserRoute* de modo a obter o objeto do nutricionista, que possui a sua lista de clientes. De seguida, na *UI*, é mostrada a lista de clientes ao nutricionista para que este escolha um deles e confirme o plano. Após confirmação do plano, é criado um *DTO* com os dados do mesmo e enviado para a *PlanRoute* através de um pedido *POST*. Estes dados serão passados para o *PlanController* que por sua vez os vai transmitir para o *PlanService* onde vai ser criado o objeto de domínio *Plan*. Este objeto é enviado para o

PlanRepo e persistido na base de dados. Por fim é retornada à *UI* o plano criado, ou erro. O *SD* deste caso de uso está presente na figura 78, que pode ser vista no Anexo A.

3.3.3.5. UC5 – Adição de refeição planeada

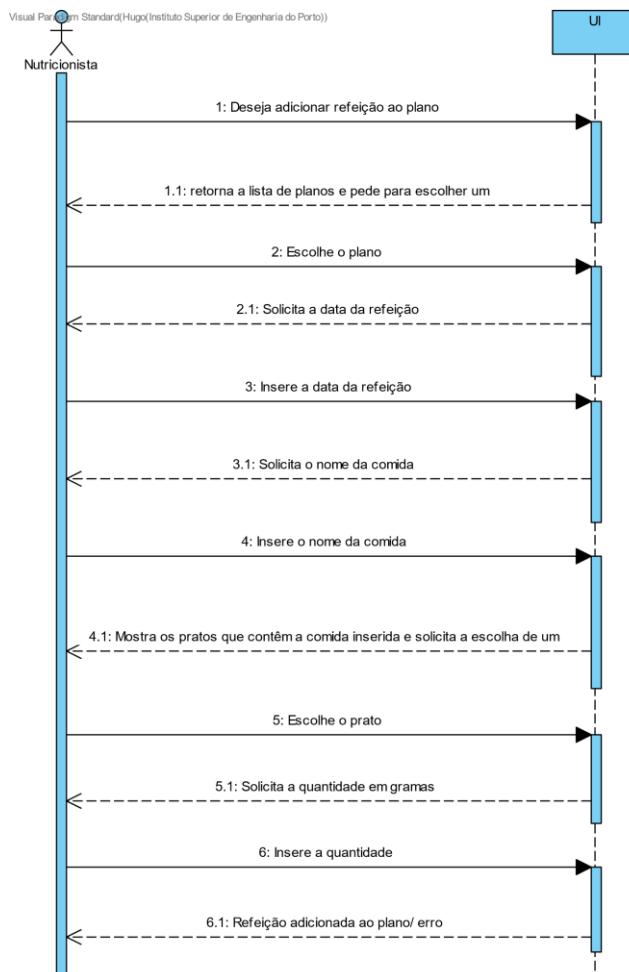


Figura 21 - SSD do caso de uso 5

Neste caso de uso o nutricionista deseja adicionar as refeições planeadas ao plano de um cliente. Este começa por um pedido *GET* à *PlanRoute* de modo a obter a lista dos planos criados pelo nutricionista, retornando-os à *UI*. Este começa com a seleção de uma data e hora por parte do nutricionista, que de seguida insere o nome de um prato/alimento. É feito um pedido *GET* à *FoodRoute* de modo a recolher todos os pratos/alimentos relacionados com o nome introduzido, sendo eles retornados à *UI*. De seguida o nutricionista escolhe um dos pratos/alimentos, insere a sua porção e confirma a refeição a adicionar, fazendo com que a

UI envie um pedido *POST* à *PlanRoute* que por sua vez comunica com o *PlanController*. Este vai aceder a dois *Services* criando a *Meal* através do *MealService* e fazendo o *update* da lista de refeições planeadas através do *PlanService*. É retornada à *UI* a refeição planeada ou erro. O *SD* deste caso de uso está presente na figura 79, que pode ser vista no Anexo A.

3.3.3.6. UC6 – Visualização de gráficos e tabelas relativos a planos

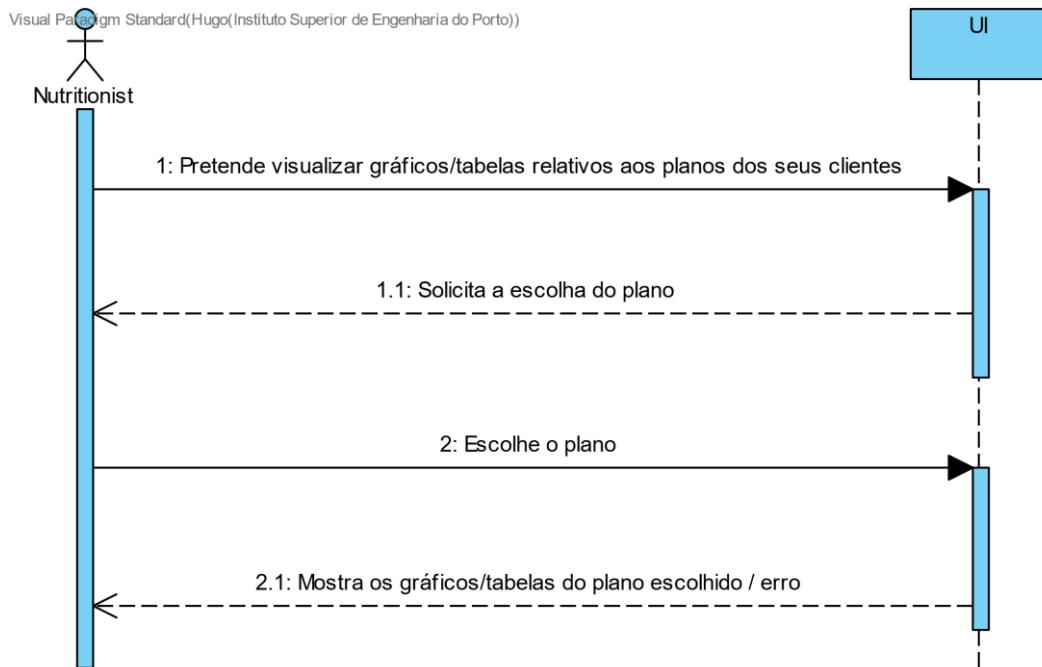


Figura 22 - SSD do caso de uso 6

Neste caso de uso o nutricionista pretende visualizar gráficos e tabelas auxiliares de modo a analisar o progresso de um dos seus clientes. Primeiramente, é feito um pedido *GET* à *PlanRoute* de modo a obter a lista dos planos criados pelo nutricionista. De seguida, é pedido ao nutricionista para selecionar um deles. Após a seleção do mesmo, são geradas na *UI* as tabelas e gráficos a partir dos dados do plano. O *SD* deste caso de uso está presente na figura 80, que pode ser vista no Anexo A.

3.3.3.7. UC7 – Decidir sobre pedido de acompanhamento

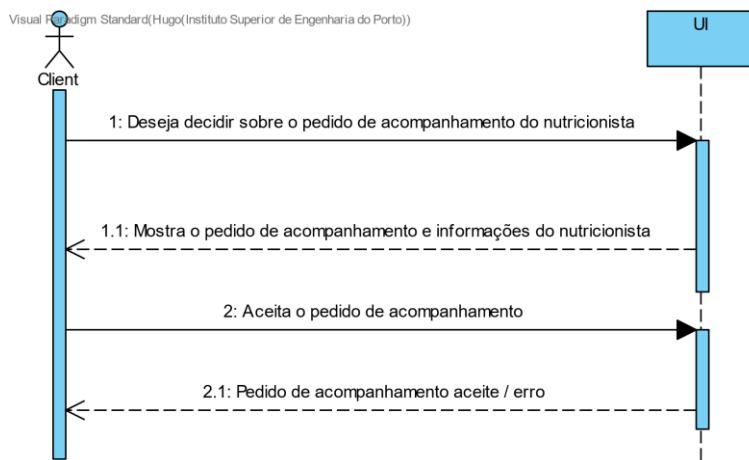


Figura 23 - SSD do caso de uso 7

Neste caso de uso o cliente deseja decidir sobre o pedido de acompanhamento do nutricionista. Este caso de uso começa por um pedido *GET* à *UserRoute* de modo a recolher o objeto do cliente atual, bem como outro pedido à *ClientRequestRoute* para pedir o pedido de acompanhamento associado a este cliente. De seguida, o cliente decide sobre o pedido de acompanhamento, pelo qual o cenário de sucesso é a aceitação do mesmo. Em seguimento a esta decisão, são efetuados dois pedidos *REST* à *UserRoute* em que no primeiro é feito um update ao atributo *hasNutritionist* do cliente, passando a ser *true*. O outro pedido de update é feito no nutricionista que enviou o pedido de acompanhamento, onde é adicionado o cliente ao atributo relativo à sua lista de clientes. De seguida, é feito um pedido à *ClientRequestRoute* onde se dá update ao atributo *isHandled* passando a ser *true*. Por fim, é feito um pedido de criação de *chat* entre os dois *users*, enviando um pedido final à componente *ChatEngine*. Caso o cliente rejeite o pedido de acompanhamento, os pedidos *REST* 2.1, 2.2 e 2.4 da figura 81 não são efetuados, diagrama esse que pode ser visto no Anexo A

3.3.3.8. UC8 – Registo de peso diário

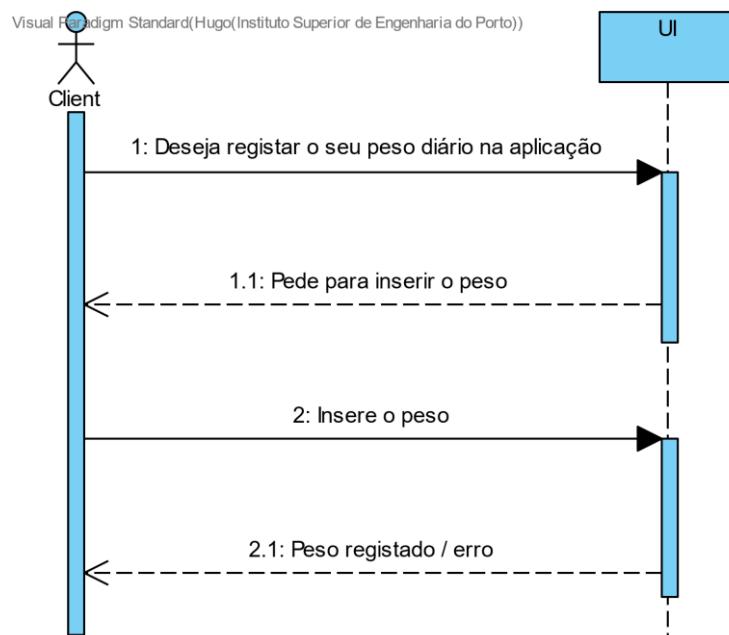


Figura 24 - SSD do caso de uso 8

Neste caso de uso o cliente deseja registrar o seu peso diário na aplicação. Este caso de uso começa com um pedido *GET* à *UserRoute* de modo a recolher o objeto do cliente atual. De seguida, é feito outro pedido *GET* relativo ao último peso registado do cliente, sendo este feito à *WeightRoute*. Caso não haja nenhum peso registado na data corrente, é pedido ao cliente para introduzir o seu peso. Após a inserção, é feito um pedido *REST* à *UserRoute* de modo a criar um novo objeto *Weight* e inseri-lo na base de dados, sendo de seguida adicionado o *id* do mesmo num *update* à lista de pesos do cliente. O *SD* deste caso uso está presente na figura 82 que pode ser consultada no Anexo A.

3.3.3.9. UC9 – Criação de prato/alimento customizado

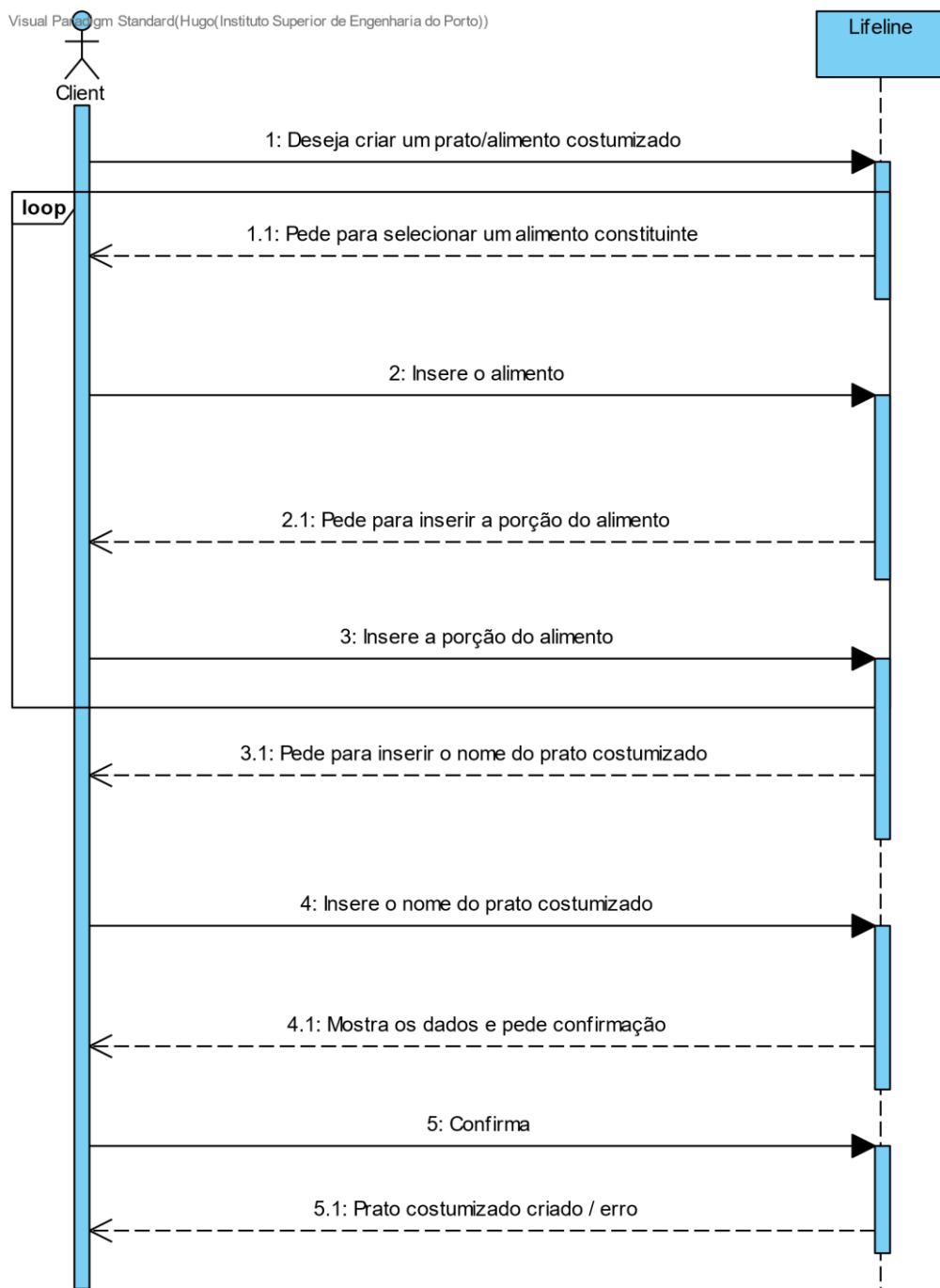


Figura 25 - SSD do caso de uso 9

Neste caso de uso o cliente deseja criar um prato/alimento customizado. Primeiramente, é feito um pedido *GET* à *FoodRoute* de modo a retornar todos os alimentos da base de dados. De seguida, na *UI*, é pedido ao cliente para escolher os alimentos e porções desejadas em *loop*. Quando o cliente tiver terminado, este insere um nome para o seu prato customizado e

submete a informação, mandando um pedido *POST* à *FoodRoute* com o *DTO* do prato criado, sendo guardado na base de dados. O *SD* deste caso uso está presente na figura 83 que pode ser consultada no Anexo A.

3.3.3.10.UC10 – Registo de refeição consumida

Neste caso de uso o cliente deseja adicionar uma refeição consumida ao registo de refeições do seu plano. Este caso de uso é bastante semelhante ao caso de uso 5, sendo a única diferença o atributo ao qual se atribui a refeição registada (sendo ela aqui adicionada às *registeredMeals* em vez das *plannedMeals*) pelo qual não se sentiu necessidade de desenhar uma vista de processos.

3.3.3.11.UC11 – Histórico de refeições consumidas e mensagens de apoio

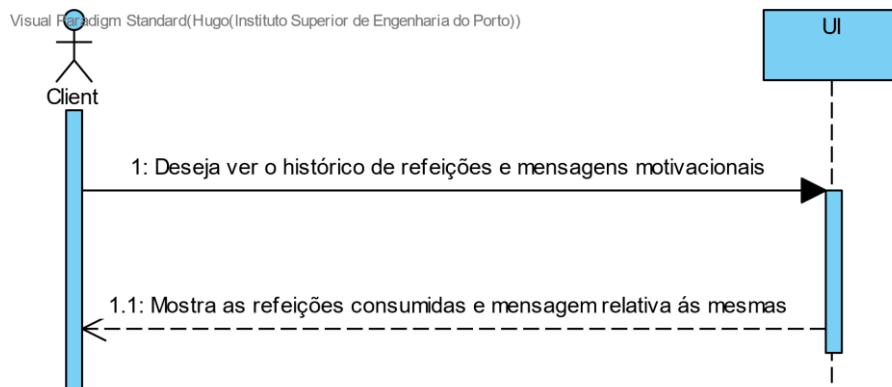


Figura 26 - SSD do caso de uso 11

Neste caso de uso o cliente deseja visualizar o seu histórico de refeições planeadas e refeições registadas bem como receber o *feedback* da aplicação no que diz respeito à avaliação do seu progresso.

O caso de uso começa pelo envio de um pedido *GET* por parte da *UI* à *backend* de modo a obter o objeto do plano do cliente em questão. De seguida, para todas as refeições planeadas e registadas do cliente irão ser feitos novos pedidos *GET* de modo a obter a informação sobre os pratos/alimentos que compõem as refeições presentes no plano. Por fim, é feito novo

pedido *GET* à *backend* de modo a obter a mensagem personalizada, sendo que para a avaliação do sucesso do cliente o *PlanController* irá comunicar com os *Services* do *Plan* (de modo a obter o objeto do plano) e das *Meals* (de modo a obter os *ids* dos alimentos consumidos) para finalmente calcular a percentagem de *registered meals* com os mesmos pratos/alimentos das *planned meals*, atribuindo assim uma avaliação ao cliente (bom, médio ou mau) e retornando a avaliação para a *UI* que, de acordo com o código *status* enviado, faz display da mensagem mais acertada. O *SD* deste caso de uso pode ser visualizado na figura 84 que se encontra no Anexo A.

3.3.3.12.UC12 – Seleção de desafio nutricional

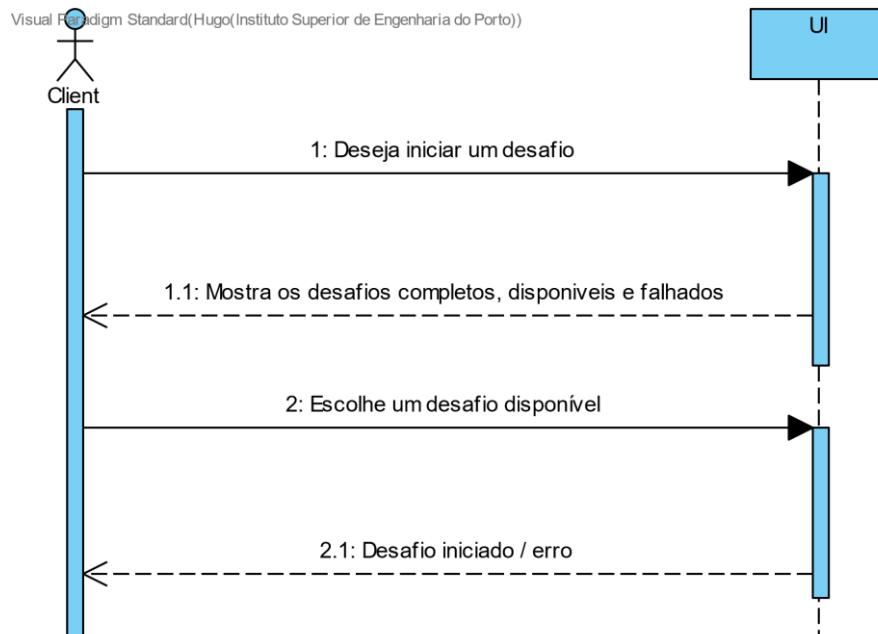


Figura 27 - SSD do caso de uso 12

Neste caso de uso o cliente deseja visualizar os desafios nutricionais da aplicação (disponíveis, completos e falhados) e deseja começar um desafio. Este caso de uso começa por fazer um pedido *GET* à *ChallengeRegistryRoute* de modo a obter o desafio ativo do cliente. Caso o cliente esteja a realizar um desafio, irá ser verificado se este desafio já pode estar completo ou se foi falhado, através da verificação das refeições consumidas do plano do cliente (caso tenha consumido um “alimento proibido”, o desafio irá falhar) e através da verificação da data do plano. Caso algum destes fluxos aconteça, o plano irá ser atualizado e adicionado ao

histórico de desafios do cliente e irá ser enviada uma mensagem a comunicar tal facto. De seguida a *UI* faz dois pedidos *GET* à *backend* de modo a obter o histórico de desafios e os desafios disponíveis, sendo todos eles mostrados na *UI*. Caso o cliente deseje escolher um dos desafios disponíveis irá ser criado um *DTO* de um novo *ChallengeRegistry* e este irá ser enviado num pedido *POST* à base de dados de modo a ativar o desafio e a ser adicionado aos desafios ativos do cliente. O *SD* deste caso de uso pode ser visualizado na figura 85 que se encontra no Anexo A.

3.3.3.13.UC13 – Chat entre utilizadores

Neste caso de uso um utilizador da aplicação deseja abrir o seu *chat* e comunicar com outro utilizador. Ao entrar na aba do *chat*, a *UI* usa a biblioteca da componente *Chat Engine* fornecendo-lhe o *Project ID*, o nome do utilizador e a sua password, mostrando posteriormente o seu *chat*. Devido à simplicidade do caso de uso, não se sentiu necessidade de desenhar uma vista de processos.

3.3.4. Vista de Implementação

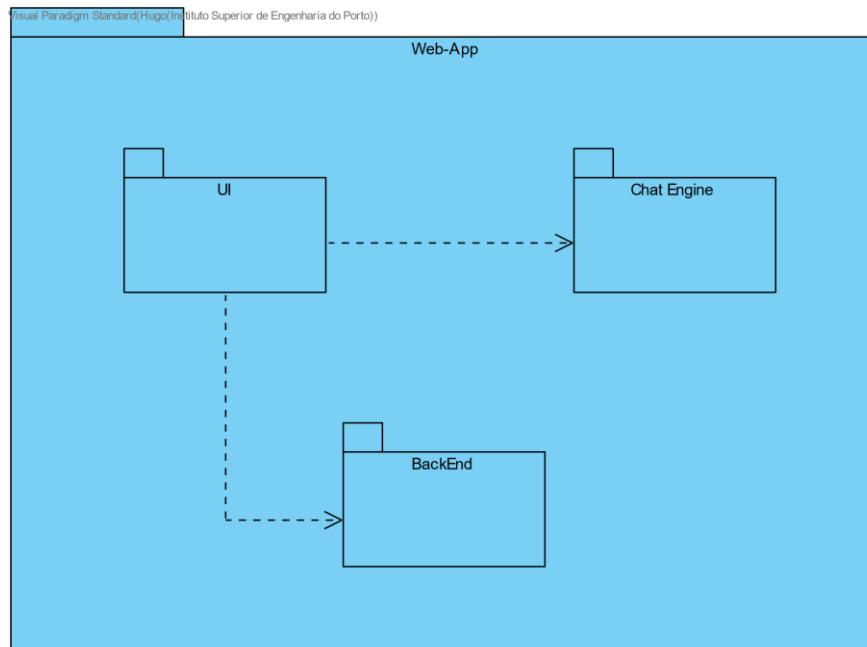


Figura 28 - Vista de implementação (nível 2)

A aplicação desenvolvida é composta por 3 componentes, contendo uma *BackEnd* que comunica diretamente com a componente relativa à *frontend*, a *UI*. Por sua vez, a *UI* comunica com uma componente externa, o *Chat Engine*.

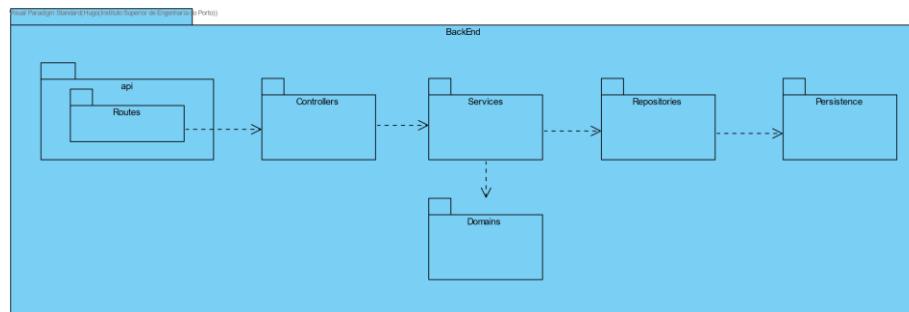


Figura 29 - Vista de implementação da BackEnd (nível 3)

A *BackEnd* irá receber os pedidos *REST* da *FrontEnd*, sendo que eles irão ser recebidos primeiramente nas *Routes*, que de seguida são tratados nos *Controllers*. A camada seguinte, *Services*, constrói os objetos de domínio e comunica posteriormente com os *Repositories* de modo a que este consiga persistir os dados na base de dados.

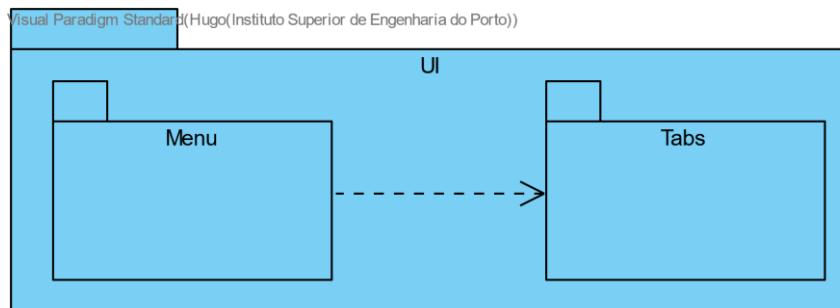


Figura 30 - Vista de implementação da UI (nível 3)

A *FrontEnd* da aplicação é uma *UI* que irá ser composta por um *Menu*, menu esse que irá ter as funcionalidades propostas dispersas por *Tabs* (*abas*).

4. Implementação da Solução

Neste capítulo é feita uma síntese da implementação da solução. Primeiramente vai ser descrita a implementação feita, sendo dadas explicações e evidências do mesmo. De seguida é feita uma descrição dos testes efetuados e por fim descreve-se a avaliação da solução final.

4.1. Descrição da implementação

Nesta secção são descritas as funcionalidades implementadas, apresentando os seus fluxos de execução e são detalhadas as bibliotecas e ferramentas mais importantes utilizadas nos casos de uso. De modo a navegar entre as funcionalidades disponíveis na *web-app*, é necessário o uso de um menu presente na *UI*, menu esse que irá ser diferente dependendo do tipo de utilizador que esteja a utilizar o sistema, havendo um menu para o utilizador não registado, o nutricionista e o cliente, estando eles presentes na figura 31.

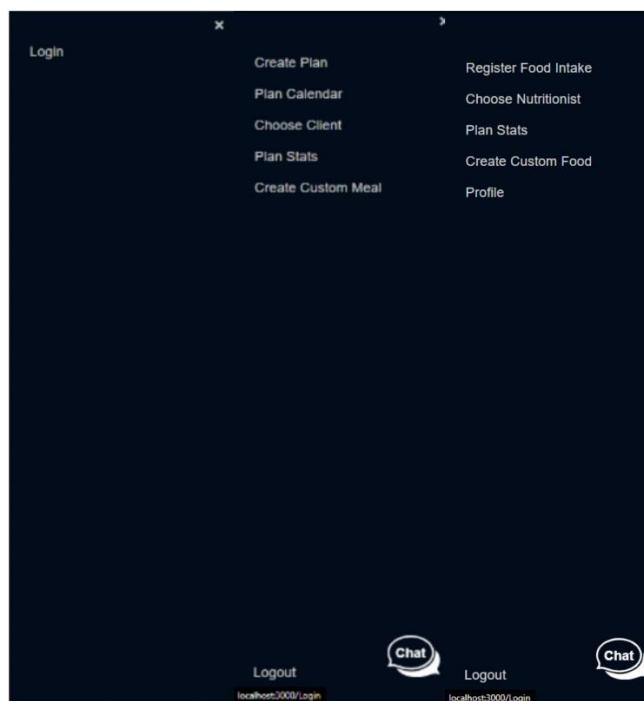


Figura 31 - Menus da aplicação para utilizadores não registados, nutricionistas e clientes respetivamente

4.1.1. UC1 – Registo de utilizador na aplicação

Neste caso de uso, um utilizador não registado pretende fazer o seu registo na aplicação. Foram então implementados os métodos de acordo com o desenho presente na secção 3.3.3.1.

O utilizador não registado começa este caso de uso através do uso do menu mencionado no ponto 4.1, que após a seleção do item “*Login*” irá ser redirecionado para o mesmo. Esta página possui um botão que lhe permite fazer o registo na aplicação, começando por questioná-lo quanto ao seu tipo de utilizador, como pode ser visto na figura 32.

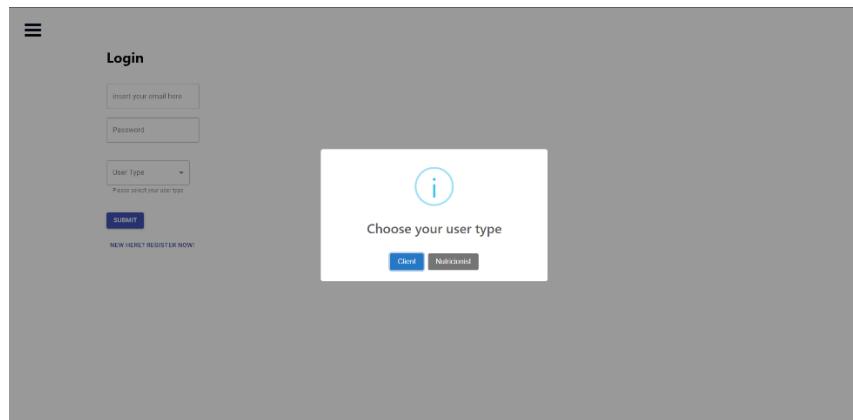


Figura 32 - Página inicial do registo onde é necessário indicar o tipo de utilizador

Caso este deseje se inscrever como cliente, então ser-lhe-á pedido para introduzir o seu email, um nome, uma password, a sua altura, peso, idade e por fim o seu género, o que pode ser visto na figura 33.

Insert your email here
hugoClient@gmail.com

Insert your name here
Hugo

Password
xxxx

Confirm your Password
xxxx

Height
171

Weight
65

Age
22

Gender
Male

Please select your gender

ACCEPT TERMS AND CONDITIONS

Figura 33 - Página de registo de um cliente

No caso de o tipo de utilizador pretendido ser o de nutricionista, ser-lhe-á pedido para introduzir apenas o seu email, nome e password, como se pode ver na figura 34.

Insert your email here
matildeNutricionista@gmail.com

Insert your name here
Matilde

Password
xxxx

Confirm your Password
xxxx

ACCEPT TERMS AND CONDITIONS

Figura 34 - Página de registo de um nutricionista

Em ambos os casos, antes de fazer o registo, é solicitado ao utilizador não registado a aceitação dos termos e condições da aplicação presentes na figura 35.

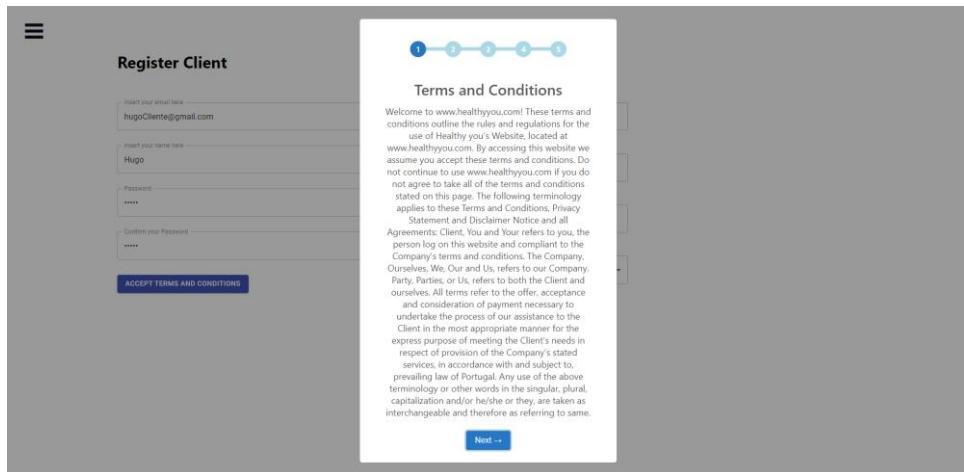


Figura 35 - Alguns dos termos e condições mostrados no registo

Após aceitação dos termos e condições é possível finalizar o registo na aplicação, introduzindo o novo utilizador na base de dados.

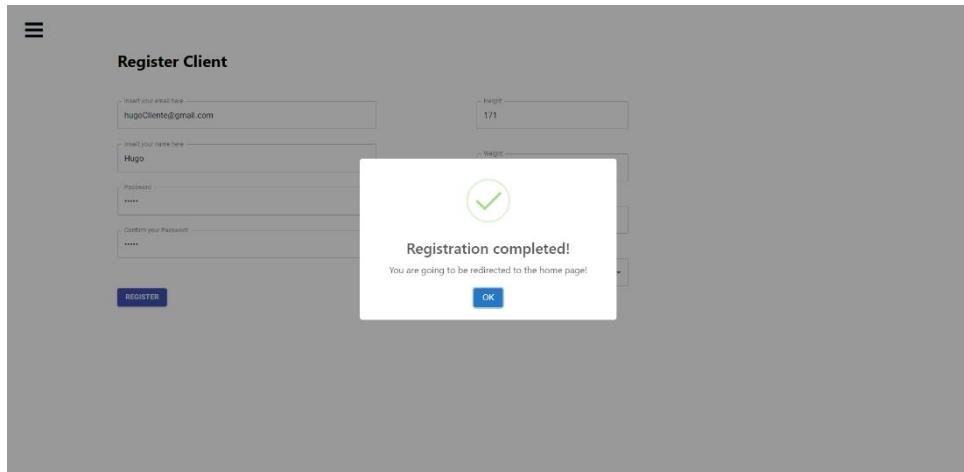


Figura 36 - Mensagem de sucesso no registo

4.1.2. UC2 – Login na aplicação

Neste caso de uso o cliente ou nutricionista deseja efetuar o login na aplicação, começando por, através do menu, selecionar o item relativo ao Login, o que o irá levar à página do mesmo, página presente na figura 37.

The screenshot shows a login form titled 'Login'. It contains three input fields: 'Insert your email here' with the placeholder 'matildeNutricionista@gr', 'Password' with the placeholder '*****', and 'User Type' with the dropdown value 'Nutricionist'. Below the User Type field is a message: 'Please select your user type'. At the bottom are a 'SUBMIT' button and a link 'NEW HERE? REGISTER NOW'.

Figura 37 - Página de login

Aqui irá ser necessária a introdução dos dados relativos ao login, sendo solicitado o email do utilizador, a sua palavra-passe e o seu tipo de utilizador. Após a inserção dos mesmos, a *web-app* comunica com a base de dados verificando a validade das informações introduzidas. Caso o cliente ou nutricionista tenha introduzido uma combinação correta ele irá efetuar o login com sucesso, o que pode ser visto na figura 38. Pelo outro lado, caso a combinação introduzida pelo utilizador não registado seja inválida, o sistema irá notificá-lo do mesmo e o login não é concretizado.

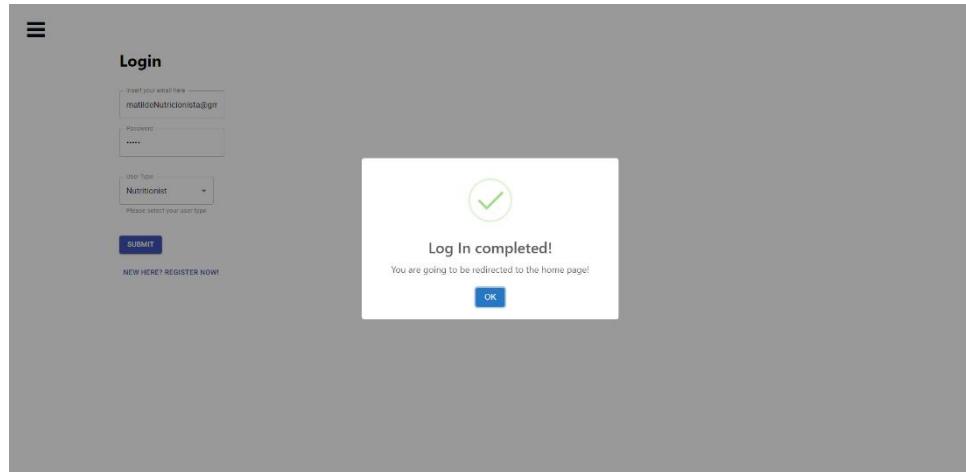


Figura 38 – Mensagem de sucesso no Login

4.1.3. UC3 – Enviar pedido de acompanhamento

Neste caso de uso o nutricionista pretende selecionar um cliente de modo a realizar o seu acompanhamento. Este caso de uso começa com a seleção do item “Choose Client” do menu da *UI*, redirecionando o nutricionista à página onde pode escolher o cliente a acompanhar. Nesta página, é possível ao nutricionista visualizar todos os clientes da aplicação que ainda não possuem um nutricionista associado, podendo ver fatores como o seu nome, a sua altura, peso, idade e gênero, como pode se visto na figura 39.

The screenshot shows a user interface titled "Choose Client". At the top, there is a table with columns: name, height, weight, age, and gender. The data in the table is as follows:

name	height	weight	age	gender
Stereim	170	75	23	MALE
Joana	170	80	22	FEMALE
Hugo	171	65	22	MALE

Below the table, there is a dropdown menu labeled "Client Name" with the placeholder text "Please select the client's name". A blue "CONFIRM" button is located at the bottom of the dropdown menu.

Figura 39 - Tabela com os dados dos clientes sem acompanhamento

Após ponderação por parte do nutricionista, este irá utilizar uma caixa de seleção para escolher o cliente a quem deseja enviar pedido, pedido esse que irá ser enviado após confirmação, pelo que se pode ver na figura 40.

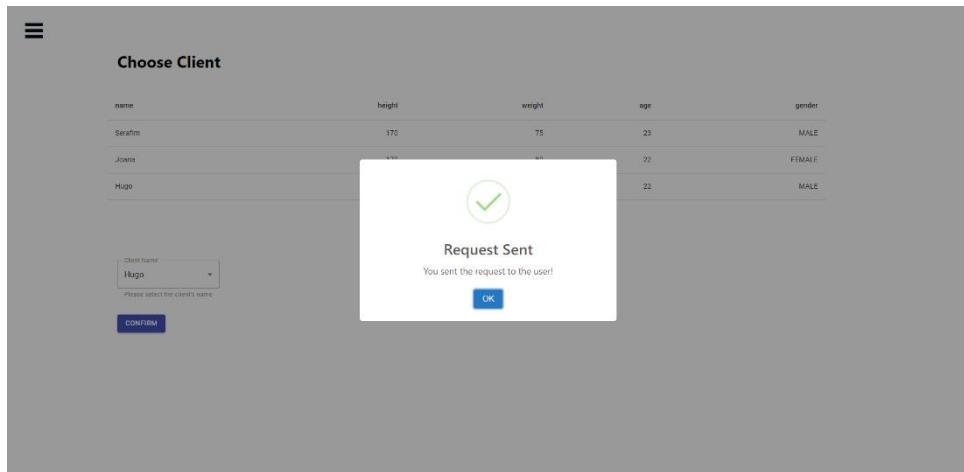


Figura 40 - Mensagem de sucesso no envio de um pedido de acompanhamento

4.1.4. UC4 – Criar plano alimentar

Neste caso de uso o nutricionista deseja criar um plano alimentar e atribuí-lo a um dos seus clientes. O caso de uso começa pela seleção do item “*Create Plan*” do menu, redirecionando o nutricionista à página de criação do plano. Nesta página, é solicitado ao nutricionista que introduza o nome do plano, uma data de início, uma data de fim e que por fim selecione o utilizador ao qual este deseja atribuir o plano, como pode ser visto na figura 41.

The screenshot shows a 'Plan' creation form. It has fields for 'Plan Name' (with placeholder 'Insert the name of the plan here'), 'Start Date' (dd/mm/yyyy format), 'End Date' (dd/mm/yyyy format), and 'Client Email' (dropdown menu with placeholder 'Please select the client's email'). At the bottom is a 'CREATE PLAN' button.

Figura 41 - Página de criação de um plano

Após a introdução dos dados solicitados, o nutricionista submete os mesmos e, caso os dados sejam válidos, é criado o plano nutricional e atribuído ao cliente escolhido, redirecionando o nutricionista para a página de edição do plano.

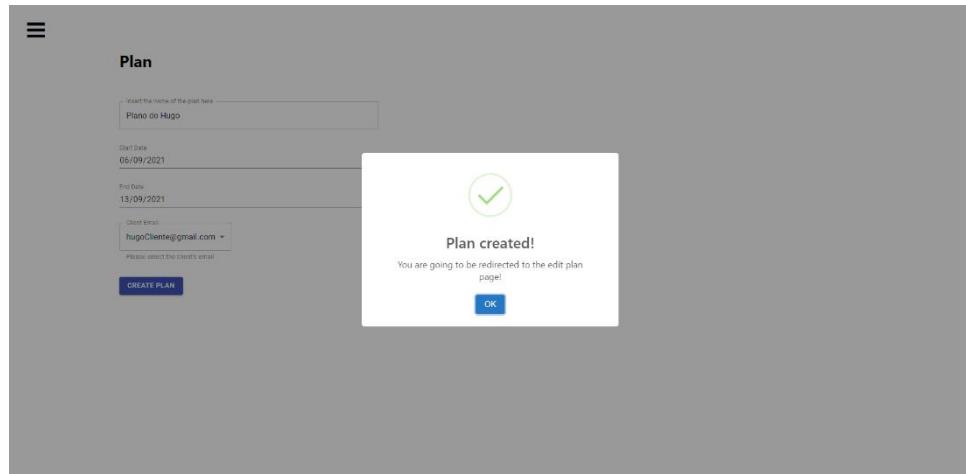


Figura 42 - Mensagem de sucesso na criação do plano

4.1.5. UC5 – Adição de refeição planeada

Neste caso de uso o nutricionista deseja adicionar uma refeição planeada a um plano existente. Ele começa pela seleção do item “Edit Plan” do menu, redirecionando-o para a página de adição de refeições planeadas. Neste caso de uso realça-se o uso da biblioteca *FullCalendar* que permitiu o uso de um calendário de modo a dinamizar a adição de refeições ao plano. O nutricionista começa por selecionar o plano que este deseja editar, como se pode ver na figura 43.

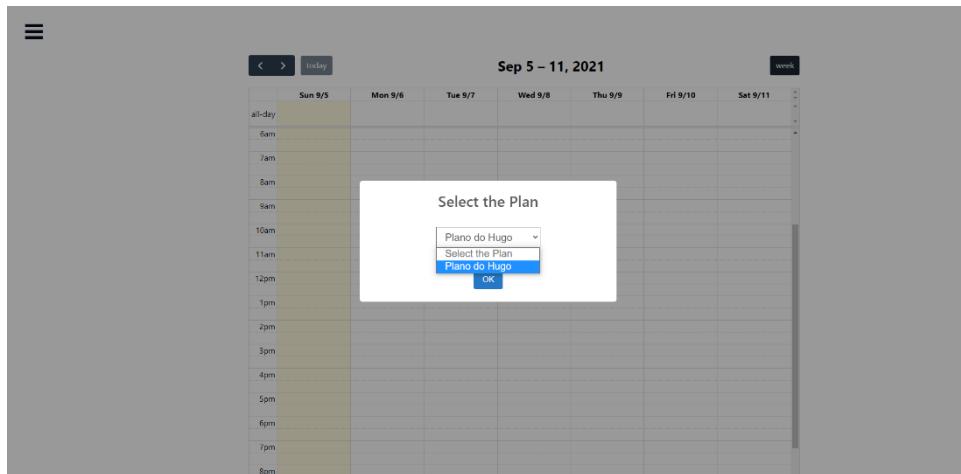


Figura 43 - Página de seleção do plano a editar

Após a seleção do plano, o nutricionista seleciona através de um clique o dia e hora ao qual deseja atribuir uma refeição, o que irá provocar o aparecimento de um *alert* no qual é possível a introdução do tipo de comida que este deseja adicionar pelo que se pode ver pela figura 44.

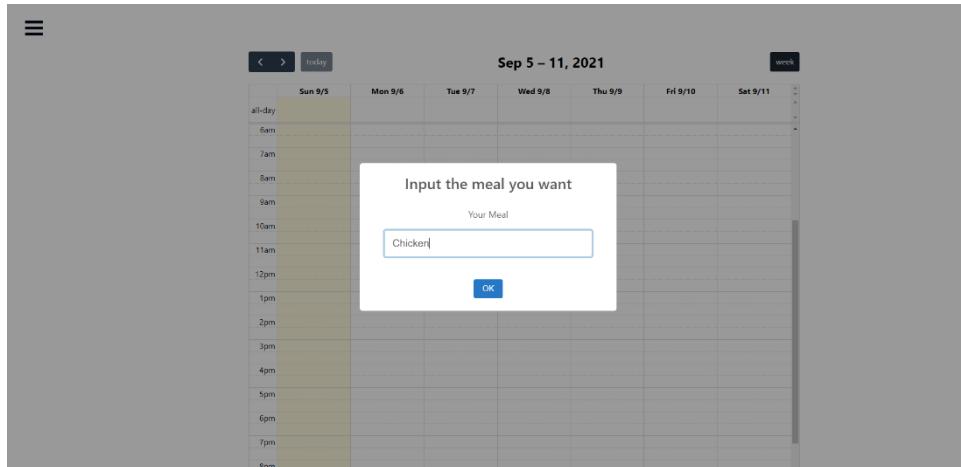


Figura 44 - Página para inserir a comida desejada

De seguida, a *backend* procura então todas as comidas relacionadas com o termo introduzido pelo nutricionista, mostrando ao mesmo uma lista das mesmas para a seleção do nutricionista, algo que pode ser visto na figura 45.

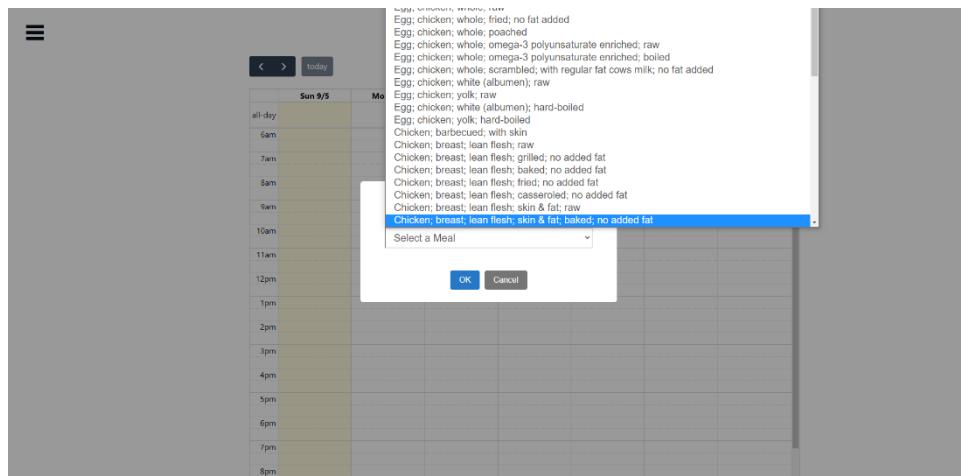


Figura 45 - Página com a seleção da comida desejada

Após seleção, é solicitado ao nutricionista a introdução da porção em gramas pretendida, como se pode ver na figura 46.

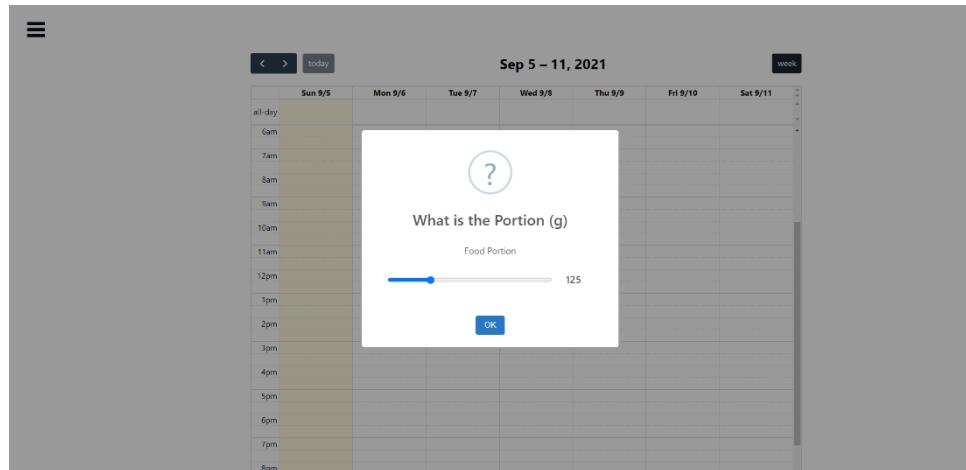


Figura 46 - Página com a seleção da porção de comida

Ao confirmar a porção desejada a refeição é guardada na base de dados, adicionada ao plano, e pode ser vista no calendário do nutricionista presente na figura 47.

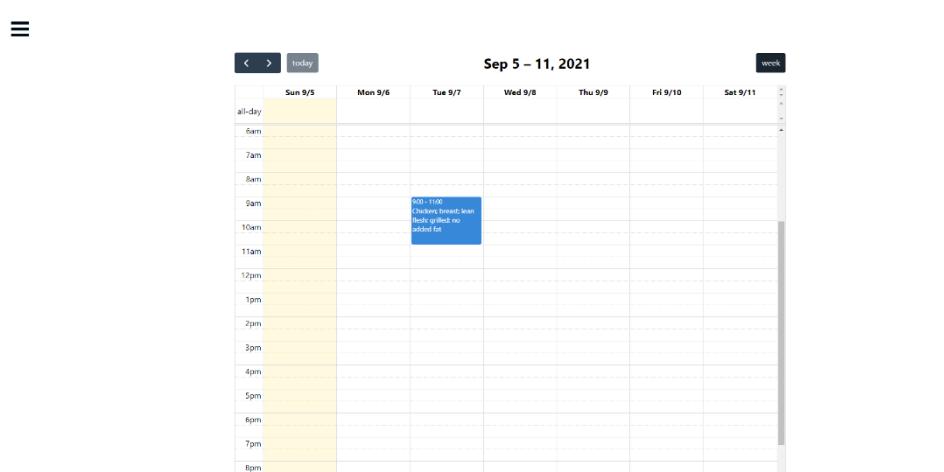


Figura 47 - Refeição criada e adicionada ao calendário

4.1.6. UC6 – Visualização de gráficos e tabelas relativas a planos

Neste caso de uso o nutricionista deseja visualizar gráficos e tabelas relativas aos hábitos alimentares dos seus clientes após estarem atribuídos a um plano. Ao selecionar o item do menu “*Plan Stats*” o nutricionista é redirecionado a uma página onde, após escolha do plano a analisar, é possível visualizar diversas estatísticas relativas ao desempenho de um cliente. É possível primeiramente ver o nome do cliente, o nome do seu plano e a duração do mesmo. De seguida são mostradas tabelas que possuem todas as refeições planeadas e registadas guardadas no plano, como pode ser visto pela figura 48.

idMeal	date	Meal Name	portion	carbohydrates	fat	protein	sugar	calories
FOOD-qUzP7xk0sy	2021-09-07T09:00:00+01:00	Chickens; breast; lean flesh; grilled; no added fat	125	0	2.5	29.8	0	143
FOOD-iCgkg60sa	2021-09-09T09:00:00+01:00	Turkey; hindquarter; lean flesh; raw	348	0	5.9	18.4	0	122
FOOD-9WzyV2c3M	2021-09-06T10:00:00+01:00	Muffin; cake-style; chocolate chip; commercial	100	48.1	17.7	6.4	29.5	374
FOOD-TZ0qk15Wc	2021-09-08T08:30:00+01:00	Salmon; Pacific King; fillet; steamed; no added fat	167	0	28.8	21.3	0	323

Figura 48 - Página com os dados de um determinado plano

Por fim, de modo a facilitar a análise desta informação, são apresentados diversos gráficos relativos às refeições monitorizadas, que dizem respeito aos gráficos 3 e 4.

Daily Calories Graph

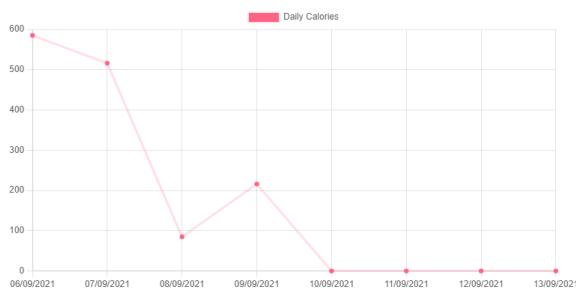


Gráfico 3 - Gráfico com a flutuação de calorias diárias de um cliente

Meals Graph

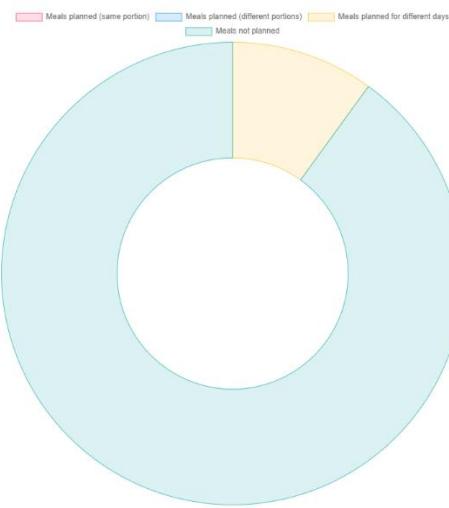


Gráfico 4 - Gráfico circular com a quantidade de refeições registadas comparadas com as refeições planeadas, porções planeadas e dias das refeições

É de realçar o uso da biblioteca “react-chartjs-2” para a demonstração dos gráficos presentes nos gráficos 3 e 4

4.1.7. UC7 – Decidir sobre o pedido de acompanhamento

Neste caso de uso o cliente deseja decidir se deseja aceitar um pedido de acompanhamento de um nutricionista. Ao fazer login na aplicação, irá aparecer um alerta no canto inferior direito do ecrã a informar o cliente que recebeu um pedido de acompanhamento como pode ser visto pela figura 49.

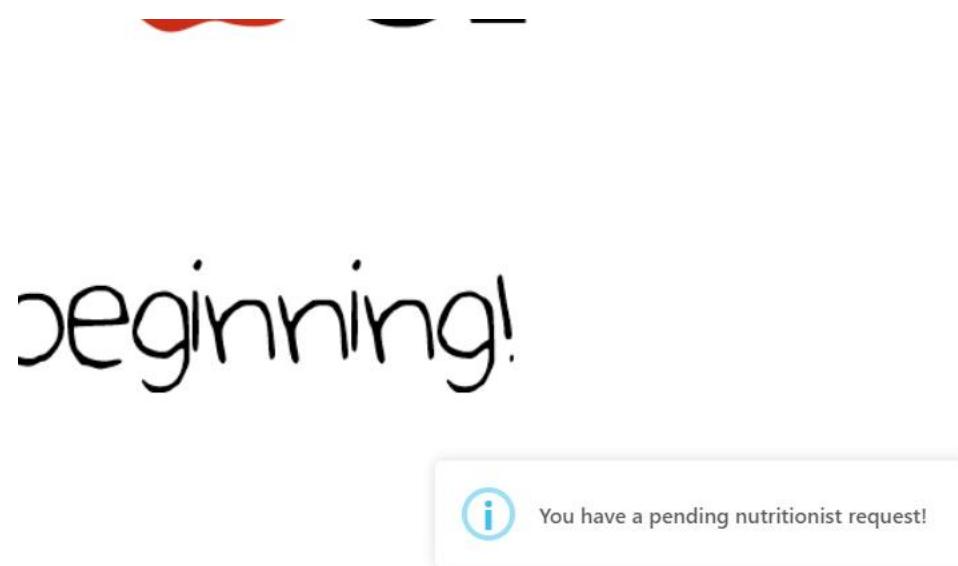


Figura 49 - Notificação de recebimento de pedido de acompanhamento

Para o visualizar, este deve ir à secção “Choose Nutritionist” do menu. Após redireccionamento para a página é possível a visualização do pedido de acompanhamento do nutricionista, mostrando o *id* do pedido e o nome do nutricionista como se pode ver na figura 50.

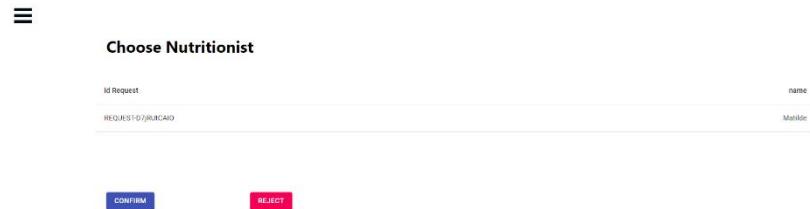


Figura 50 - Página para a escolha de nutricionista

Após visualização do pedido cabe ao cliente decidir sobre o pedido de acompanhamento o que, em caso afirmativo, irá fazer com que o mesmo seja seguido pelo nutricionista.

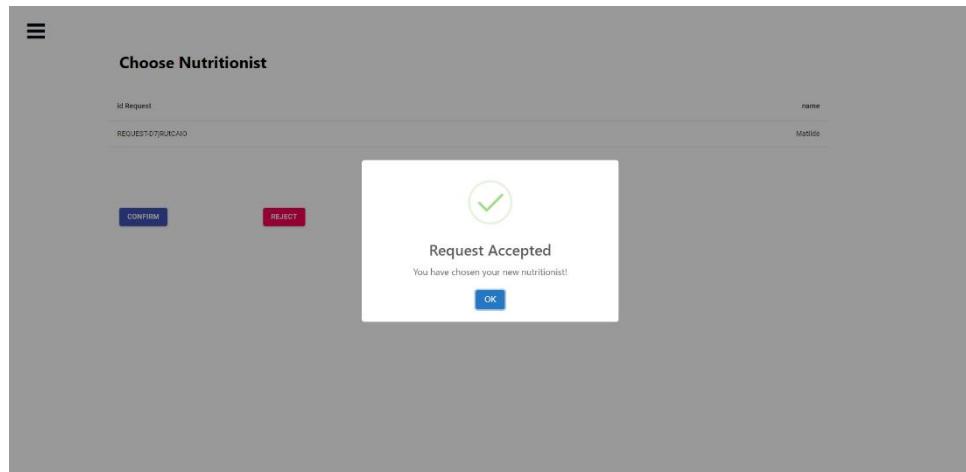


Figura 51 - Mensagem de sucesso na aceitação do pedido de acompanhamento

4.1.8. UC8 – Registo de peso diário

Neste caso de uso o cliente deseja efetuar o registo do seu peso diário. Para o fazer, este necessita de ir ao seu perfil, selecionando o item “Perfil” do menu.

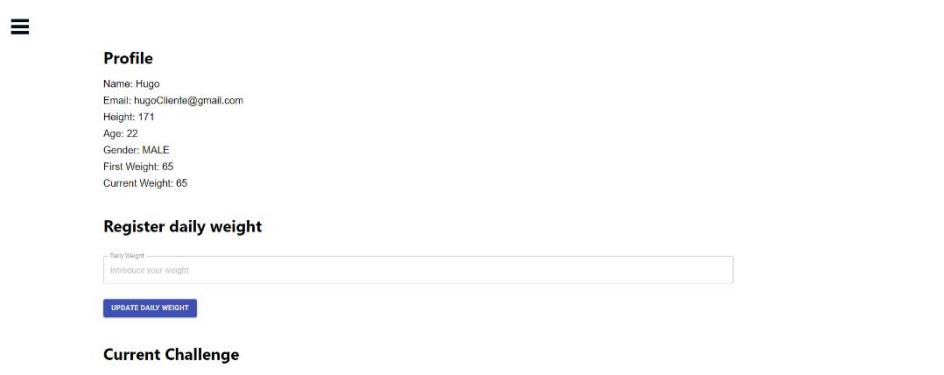


Figura 52 - Perfil pessoal de um utilizador

Aqui é possível ao cliente visualizar os seus dados pessoais como o nome, email, altura, idade, gênero, o peso inicial introduzido no registo na aplicação e o peso atual, algo presente na figura 52. É então possível ao cliente introduzir o peso diário medido e guardá-lo na base de dados.

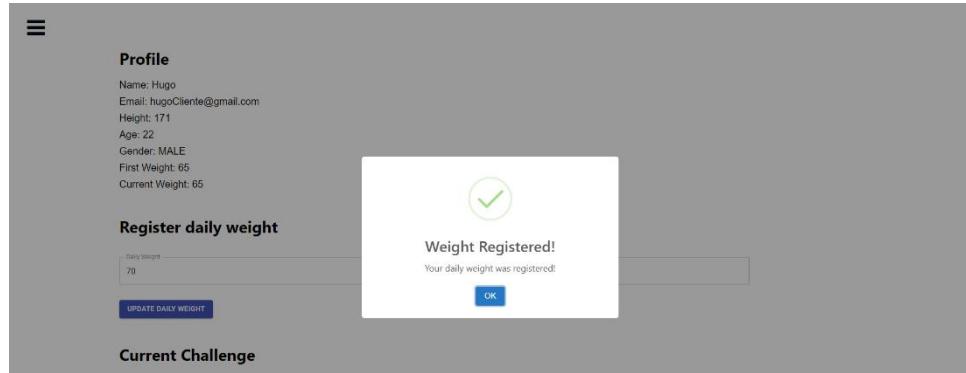


Figura 53 - Mensagem de sucesso no registo diário do peso

Após guardar o peso diário, o perfil irá ser atualizado de modo a mostrar o peso registado como peso atual, bem como irá ser bloqueada a introdução de uma nova medição de peso até ao dia seguinte, o que pode ser visto na figura 54.

The screenshot shows a mobile application interface. At the top, there is a navigation bar with three horizontal lines. Below it, a section titled "Profile" displays personal information: Name: Hugo, Email: hugoCliente@gmail.com, Height: 171, Age: 22, Gender: MALE, First Weight: 65, and Current Weight: 70. Below the profile is a section titled "Register daily weight" with a note indicating that the user has already entered their weight for the day. Further down, there is a "Current Challenge" section, a "Challenge History" link, and a "Available Challenges" link.

Figura 54 - Proibição de inserção do peso diário após registo no mesmo dia

4.1.9. UC9 – Criação de prato/alimento customizado

Neste caso de uso o cliente pretende realizar a criação de um prato ou alimento personalizado. Após seleção da opção “Create Custom Food” do menu, o cliente é redirecionado para uma página segmentada em 3 secções. Na primeira secção é possível fazer a seleção de um dos alimentos da base de dados e da porção pretendida como pode ser visto na figura 55.

The screenshot shows a mobile application interface for creating a custom meal. At the top, there is a navigation bar with three horizontal lines. Below it, a section titled "Custom Food" allows the user to enter a food name ("Wine: rose") and its portion size ("14"). A "ADD FOOD TO MEAL" button is present. The next section, "Foods Included", displays nutritional information for the selected food: Total, portion (g), carbohydrates, fat, protein, sugar, and calories. The values shown are 0 for all categories. Finally, a "Save Custom Food" section asks the user to input the name of the custom meal and provides a "SUBMIT CUSTOM FOOD" button.

Figura 55 - Página de criação de um prato personalizado

Após a adição desse alimento ao prato personalizado, o mesmo irá aparecer na secção *Foods Included* sendo possível ver os seus valores nutricionais com base na porção introduzida presentes na figura 56.

The screenshot shows the 'Custom Food' creation interface. At the top, there's a search bar with 'Food' and 'Chicken; breast; lean flesh; cas...' entered. Below it is a portion input field set to '100'. A blue button labeled 'ADD FOOD TO MEAL' is visible. The main area is titled 'Foods Included' and contains a table:

Food Name	portion (g)	carbohydrates	fat	protein	sugar	calories
Wine, rose	10	0.07	0	0.02	0.07	7
Total	10	0.07	0	0.02	0.07	7

Below the table is a section titled 'Save Custom Food' with a text input field for the meal name and a blue 'SUBMIT CUSTOM FOOD' button.

Figura 56 - Tabela com o alimento adicionado ao prato

O cliente então insere todos os alimentos pretendidos da mesma forma, finalizando a criação do prato personalizado com a introdução do seu nome e guardando-o na base de dados.

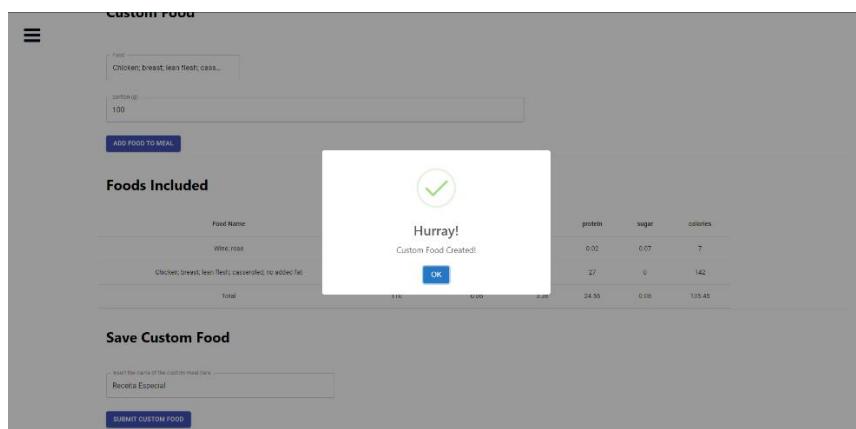


Figura 57 - Mensagem de sucesso na adição do prato/alimento personalizado

4.1.10. UC10 – Registo de refeição consumida

Neste caso de uso o cliente pretende registar uma refeição que o mesmo consumiu. Após seleção do item “*Register Food Intake*” do menu, este é redirecionado para um calendário igual ao presente no caso de uso 5 como pode ser visto pela figura 58.

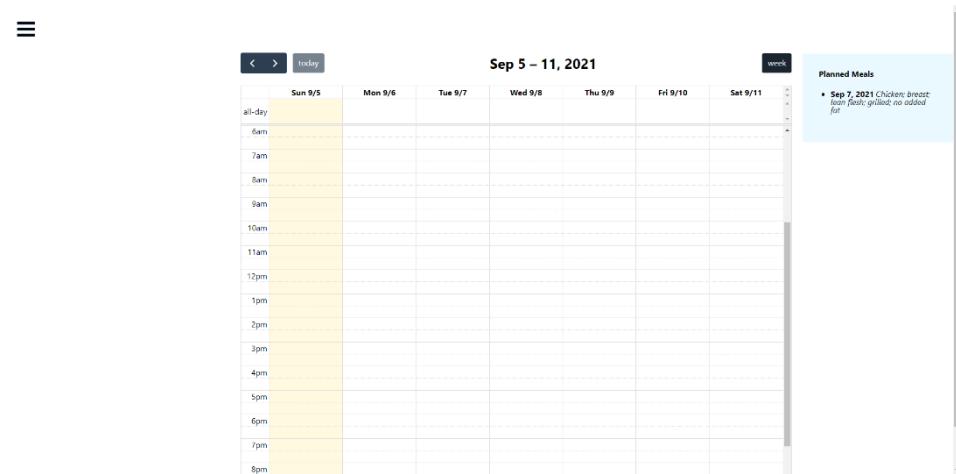


Figura 58 - Calendário para registo de refeição com nota auxiliar relativa às refeições planeadas

É possível também a visualização das refeições planeadas pelo nutricionista, de modo a auxiliar as escolhas do cliente. O processo para o registo da refeição consumida é igual ao do caso de uso 5, onde após a seleção de uma data e hora é solicitada a comida e porção consumida e registada na base de dados, aparecendo de seguida no calendário do utilizador presente na figura 59.

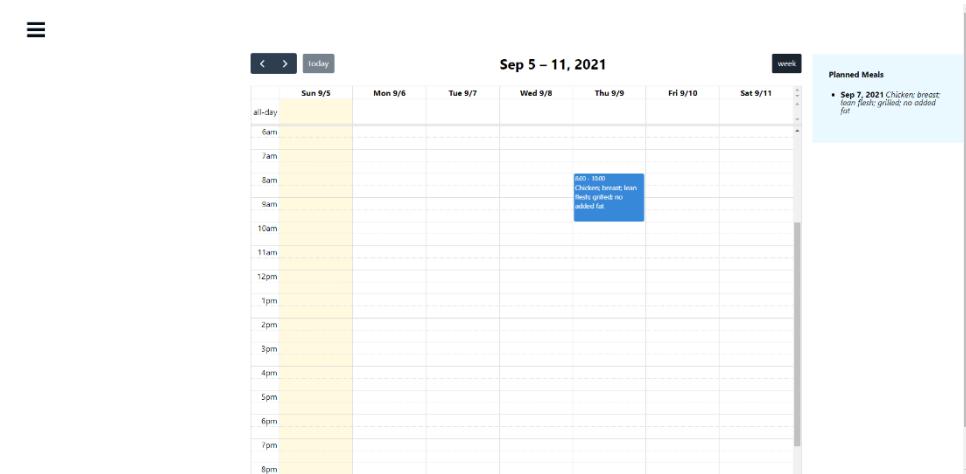


Figura 59 - Calendário com a refeição registada

4.1.11. UC11 – Histórico de refeições do cliente

Neste caso de uso o cliente deseja visualizar as estatísticas relativas ao seu plano. Ao selecionar o item “*Plan Stats*” do menu, este é redirecionado para uma página que possui as refeições planeadas e registadas do seu plano, bem como o nome do mesmo e a sua duração, dados presentes na figura 60.

A screenshot of a client's meal history page. At the top, it displays "Plan Info" with the plan name "Piano do Hugo" and duration "2021-09-06 - 2021-09-13". Below this, there are two sections: "Planned Meals" and "Registered Meals", each listing meals with their details and nutritional values.

idMeal	date	Meal Name	portion	carbohydrates	fat	protein	sugar	calories
FOOD-02f7X99y	2021-09-07T09:00:00+01:00	Chicken; breast; lean flesh; grilled; no added fat	125	0	2.5	29.8	0	145
FOOD-1Cgkgsd0sa	2021-09-06T09:00:00+01:00	Turkey; hindquarter; lean flesh; raw	348	0	5.3	18.4	0	122
FOOD-1Wzyf2c0M	2021-09-06T10:00:00+01:00	Muffin; cake style; chocolate chip; commercial	100	48.1	17.7	6.4	29.3	374
FOOD-1ZDqai1K9w	2021-09-07T08:00:00+01:00	Salmon; Pacific King; fillet; steamed; no added fat	167	0	26.0	21.3	0	323

idMeal	date	Meal Name	portion	carbohydrates	fat	protein	sugar	calories
FOOD-02f7X99y	2021-09-07T09:00:00+01:00	Chicken; breast; lean flesh; grilled; no added fat	151	0	2.5	29.8	0	145
FOOD-1ZDqai1K9w	2021-09-08T08:00:00+01:00	Salmon; Pacific King; fillet; steamed; no added fat	181	0	26.8	21.3	0	323

Figura 60 - Página com as informações do plano para o cliente

Com base nestas refeições, é também mostrada uma mensagem ao cliente relativa ao seu progresso, sendo a mensagem obtida através do cálculo da percentagem de refeições

consumidas iguais ou com uma porção semelhante às refeições planeadas. Os exemplos das mensagens mostradas podem ser vistos na figura 61.



Figura 61 - Mensagens relativas ao desempenho do utilizador

4.1.12. UC12 – Seleção de desafio nutricional

Neste caso de uso o cliente pretende selecionar um desafio nutricional para realizar. Na sua página de perfil, é possível a visualização dos desafios nutricionais presentes na web-app, onde é possível efetuar a escolha de um dos desafios disponíveis, estando eles presentes na figura 62.

Available Challenges



Figura 62 - Desafios nutricionais disponíveis

Após a escolha de um desafio são mostradas as informações do mesmo e é possível o começo do mesmo, como se pode ver pela figura 63.

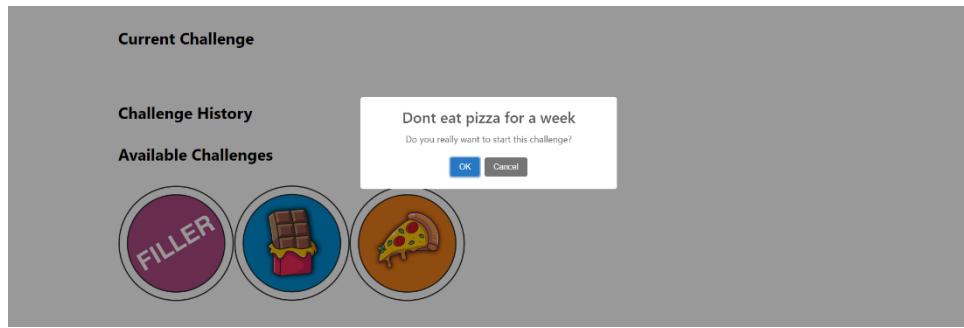


Figura 63 - Seleção de um desafio nutricional

Após começo, o desafio irá ser colocado na secção do *Current Challenge*, como pode ser visto na figura 64.

Current Challenge



Challenge History

Available Challenges



Figura 64 – Ativação do desafio escolhido

Caso os hábitos alimentares do cliente estejam de acordo com o que foi estabelecido pelo desafio (p. Ex. O cliente não consome pizza durante uma semana) o desafio irá ser marcado como concluído e guardado no *Challenge History*. Caso o desafio seja concluído com sucesso, irá apresentar um contorno verde. Caso o cliente falhe na realização do desafio, o “crachá” do mesmo irá ter um contorno vermelho.

Challenge History



Challenge History



Figura 65 - Crachás possíveis após completamento de um desafio, sendo ele bem-sucedido (crachá da esquerda) ou malsucedido (crachá da direita)

4.1.13. UC13 – Chat entre utilizadores

Neste caso de uso o nutricionista e/ou o cliente desejam utilizar um *chat* para comunicarem um com o outro. Após a seleção da bolha “*Chat*” do menu irão ser redirecionados para a funcionalidade de *chat* onde podem trocar mensagens ou imagens um com o outro. O *chat* pode ser visto na figura 66.

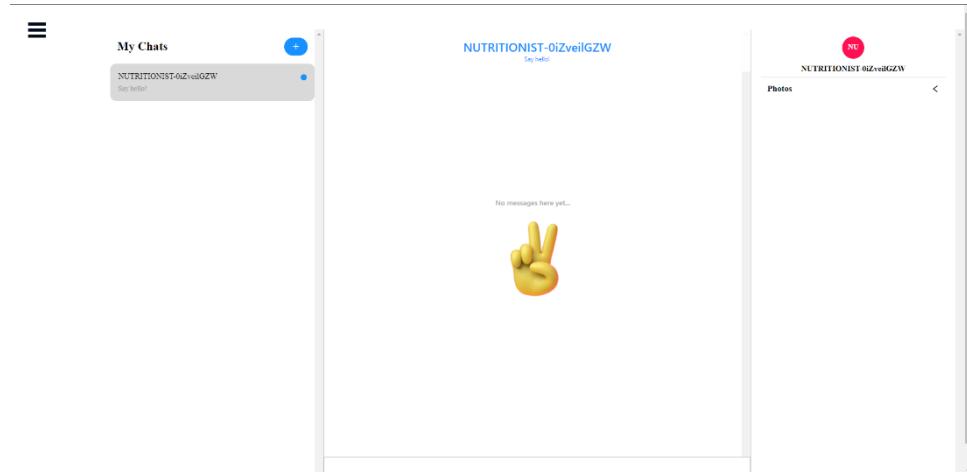


Figura 66 - Janela de Chat entre utilizadores

De realçar o uso da API *ChatEngine.io* a qual permitiu a implementação desta funcionalidade através de chamadas *REST* à mesma.

4.1.14. Bibliotecas utilizadas

De modo a implementar determinados casos de uso e construir uma *UI* de melhor qualidade foi necessário o uso de determinadas bibliotecas e ferramentas, das quais 3 se destacaram e são detalhadas nas próximas secções. As restantes bibliotecas utilizadas na implementação da solução estão descritas na secção 2.2.7. do capítulo 2.

4.1.14.1.FullCalendar

De modo a dinamizar a introdução das refeições consumidas do cliente e das refeições planeadas do nutricionista era necessária a implementação de um calendário dinâmico que permitisse a adição de refeições a certas datas e horas. Isto foi atingido com o uso da biblioteca *FullCalendar*.

Na figura 67 é possível ver a utilização desta biblioteca onde foi possível configurar o *display* das refeições sobre a forma de *events*, e a adição das mesmas através de um *select*, sendo possível a remoção das refeições tanto do plano como do *display* no calendário.

```
<FullCalendar
    disabled={true}
    plugins={[dayGridPlugin, timeGridPlugin, interactionPlugin]}
    headerToolbar={{
        left: 'prev,next today',
        center: 'title',
        right: 'timeGridWeek'
    }}
    initialView="timeGridWeek"
    editable={true}
    selectable={true}
    selectMirror={true}
    dayMaxEvents={true}
    weekends={[true]}
    events={
        | this.state.currentEvents
    }
    select={this.handleDateSelect}
    eventClick={this.handleEventClick}
/>
```

Figura 67 - Código relativo à implementação da biblioteca FullCalendar

4.1.14.2.React-chartjs-2

De modo a facilitar e auxiliar a análise dos planos dos seus clientes, os nutricionistas necessitam de formas de visualizar os dados relativos aos hábitos alimentares de quem seguem. Isto é feito através do uso de tabelas e gráficos, o que levou à necessidade do uso de uma biblioteca para conseguir fazer o *display* desses mesmos gráficos, a *React-chartjs-2*.

Na aplicação foram utilizados diversos tipos de gráficos, entre eles gráficos de barras ou gráficos circulares. Para a construção dos mesmos, é necessária a definição de *DTO's* com uma estrutura específica ao gráfico em questão, como se pode ver na figura 68.

```
data = {
  labels: ['Meals planned (same portion)', 'Meals planned (different portions)', 'Meals planned for different days', 'Meals not planned'],
  datasets: [
    {
      label: 'Registered Meals',
      data: arr,
      backgroundColor: [
        'rgba(255, 99, 132, 0.2)',
        'rgba(54, 162, 235, 0.2)',
        'rgba(255, 206, 86, 0.2)',
        'rgba(75, 192, 192, 0.2)'
      ],
      borderColor: [
        'rgba(255, 99, 132, 1)',
        'rgba(54, 162, 235, 1)',
        'rgba(255, 206, 86, 1)',
        'rgba(75, 192, 192, 1)'
      ],
      borderWidth: 1,
    },
  ],
};
```

Figura 68 - Dados para a geração de gráficos

4.1.14.3.ChatEngine.io

De modo a permitir a comunicação direta entre um cliente e o nutricionista que o acompanha, era necessário que a *web-app* contasse com uma ferramenta de *chat*. Para a implementação do *chat* foi necessário o uso da *API ChatEngine.io*.

Ao registar-se na aplicação, um utilizador também é guardado numa base de dados da *ChatEngine.io* através de uma chamada à *API*.

Username	Created	Action
CLIENT-iuBK4FECke	2021-07-27T15:38:16.683962Z	Edit Delete
NUTRITIONIST-CiwhxHBF9R	2021-07-27T15:41:18.143211Z	Edit Delete
CLIENT-nT6PgJ4wKX	2021-07-28T18:23:50.792156Z	Edit Delete

Figura 69 - Lista de utilizadores inseridos na base de dados do ChatEngine

A criação de um *chat* irá ocorrer após uma decisão afirmativa de um cliente a um pedido de acompanhamento, onde é feita nova chamada à API de modo a criar um *chat* entre esses dois utilizadores.

```
let resp4 = await axiosChat({
  baseURL: 'https://api.chatengine.io/',
  method: 'post',
  url: 'chats/',
  headers: {
    'Project-ID': '6950a850-d4a4-4450-bf56-d2205e207775',
    'User-Name': this.state.nutritionistObject.idUser,
    'User-Secret': this.state.nutritionistObject.password
  },
  data: {
    "title": this.state.nutritionistObject.idUser+"/"+this.state.clientObject.idUser,
    "is_direct_chat": true
  }
})
```

Figura 70 - Chamada à API do ChatEngine para a criação de um chat entre dois utilizadores

De modo a fazer *display* do *chat* é então implementada a componente *ChatEngine*, como pode ser visto na figura 71.

```
<ChatEngine
  height={'100vh'}
  projectID={'6950a850-d4a4-4450-bf56-d2205e207775'}
  userName={this.state.userName}
  userSecret={this.state.userSecret}
/>
```

Figura 71 - Código relativo à implementação da janela de Chat

4.2. Testes

De modo a assegurar um funcionamento correto da aplicação web, são efetuados diversos testes para cada caso de uso da aplicação. Os testes implementados são testes unitários e de integração, tendo sido necessário o uso de uma biblioteca para a sua implementação, a biblioteca *Sinon* e seguiram a abordagem AAA (*Arrange, Act and Assert*).

4.2.1. Testes de integração

De modo a validar as diversas funcionalidades da aplicação e assegurar a comunicação entre as componentes da mesma, foram realizados diversos testes de integração tendo sido eles feitos entre as componentes *Service* e *Repo*. Na figura 72 é possível visualizar o código relativo ao teste da comunicação entre o *ClientService* e o *ClientRepo* na funcionalidade *CreateClient*. Primeiramente são definidos os *DTO*'s relativos ao peso, ao cliente a criar e o *DTO* retornado após a criação do cliente. De seguida é definido o retorno da função *findOne* do *ClientSchema*, sendo ele *false* (o que indica que ainda não existe um cliente com o mesmo *ID* ou email na base de dados) através da função *stub*. Após execução da função *createClientService* irá ser obtido o *DTO* retornado, que é igual ao *finalDTO* definido previamente.

```
it('createClient: returns dto', async function () {
  let dtoWeight = {
    idWeight: "WEIGHT-teste",
    weightValue: 80,
    date: "date-test"
  }
  let dto = {
    idUser: "CLIENT-teste", name: "nome", email: "email@email.com", password: "12345",
    userType: "CLI", height: 170, weight: dtoWeight, age: 23, gender: "MALE"
  };
  let finalDto = {
    idUser: "CLIENT-teste", name: "nome", email: "email@email.com", password: "12345",
    userType: "CLI", height: 170, weight: ["WEIGHT-teste"], age: 23, gender: "MALE"
  }

  sinon.stub(clientSchema, "findOne").returns(false);
  const clientRepoInstance = new clientRepoClass(clientSchema)
  const clientServiceInstance = new clientServiceClass(clientRepoInstance)

  let newDto = await clientServiceInstance.createClientService(dto)
  sinon.assert.match(newDto, finalDto)
})
```

Figura 72 - Código de um dos testes de integração

Na tabela 5 é possível visualizar alguns dos testes de integração relativos à função *CreateClient* da classe *Client*.

Tabela 5 - Tabela com exemplos de alguns dos testes da classe Client

Teste	Descrição
<i>CreateClient: returns dto</i>	Teste do cenário de sucesso na criação de um cliente, em que retorna um <i>DTO</i> com os dados do cliente criado
<i>CreateClient: returns 401</i>	Teste no cenário de insucesso na criação de um cliente devido à sua prévia existência na base de dados, em que retorna o código 401
<i>CreateClient: returns 400</i>	Teste no cenário de insucesso na criação de um cliente devido à invalidade dos dados do cliente, retornando o código 400

Foram também feitos testes às restantes classes do sistema (*Nutritionist*, *Challenge*, *ChallengeRegistry*, *ClientRequest*, *Food*, *Meal*, *Plan* e *Weight*) seguindo a mesma lógica.

4.2.2. Testes unitários

De modo a verificar o comportamento das componentes da aplicação de forma isolada, foram realizados diversos testes unitários. Na figura 73 é possível visualizar o exemplo do código do teste unitário *CreateClient* realizado à componente *ClientService*. Primeiramente são definidos os *DTO's* relativos ao peso, ao cliente a criar e o *DTO* retornado após a criação do cliente. De seguida é feita a *stub* relativamente ao retorno da função *saveClient* do *ClientRepo*, sendo ele um objeto da classe *Client*. Após execução da função *createClientService* irá ser obtido o *DTO* retornado, que é igual ao *expectedDTO* definido previamente.

```

it('CreateClient: returns domain object with values Service', async function() {
    let dtoWeight = {
        idWeight: "WEIGHT-teste",
        weightValue: 80,
        date: "date-test"
    }
    let dto = {
        idUser: "CLIENT-teste", name: "nome", email: "email@email.com", password: "12345",
        userType: "CLI", height: 167, weight: dtoWeight, age: 23, gender: "MALE"
    };
    let expectedDto = {
        idUser: "CLIENT-teste", name: "nome", email: "email@email.com", password: "12345",
        userType: "CLI", height: 167, weight: [dtoWeight.idWeight], age: 23, gender: "MALE", hasNutritionist: false
    };

    let object = clientInstance.create(dto);
    sinon.stub(clientRepoInstance, "saveClient").returns(object);
    const serviceInstance = new clientServiceClass(clientRepoInstance);

    let newDto = await serviceInstance.createClientService(dto);

    sinon.assert.match(newDto, expectedDto);
})

```

Figura 73 - Código relativo a um dos testes unitários

Na tabela 6 é possível visualizar os testes unitários realizados à funcionalidade de criação de um cliente nas componentes *Controller*, *Service* e *Repo* dos *Clients*.

Tabela 6 - Tabela com exemplos de testes unitários feitos na classe Client

Teste	Descrição
<i>CreateClient: returns response with status 201 (controller)</i>	Teste do cenário de sucesso na criação de um cliente, em que retorna uma resposta com status 201
<i>CreateClient: returns response with status 401 (controller)</i>	Teste no cenário de insucesso na criação de um cliente devido à sua prévia existência na base de dados, retornando uma resposta com status 401
<i>CreateClient: returns response with status 400 (controller)</i>	Teste no cenário de insucesso na criação de um cliente devido à invalidade dos dados do cliente, retornando uma resposta com status 400
<i>CreateClient: returns dto (service)</i>	Teste do cenário de sucesso na criação de um cliente, em que retorna um <i>DTO</i> com os dados do cliente criado

<i>CreateClient: returns 401 (service)</i>	Teste no cenário de insucesso na criação de um cliente devido à sua prévia existência na base de dados, em que retorna o código 401
<i>CreateClient: returns 400 (service)</i>	Teste no cenário de insucesso na criação de um cliente devido à invalidade dos dados do cliente, retornando o código 400
<i>SaveClient: returns dto with values (repo)</i>	Teste do cenário de sucesso na criação de um cliente, em que retorna um objeto com os dados do cliente criado
<i>SaveClient: returns null (repo)</i>	Teste no cenário de insucesso na criação de um cliente devido à sua prévia existência na base de dados, retornando nulo

Foram também feitos testes às restantes classes do sistema (*Nutritionist*, *Challenge*, *ChallengeRegistry*, *ClientRequest*, *Food*, *Meal*, *Plan* e *Weight*) seguindo a mesma lógica.

4.3. Avaliação da solução

Para fazer uma avaliação ao projeto foi necessária a realização de reuniões bissemanais com o supervisor de modo a verificar o desenvolvimento das funcionalidades. Em cada reunião eram feitos testes de aceitação através da demonstração das diversas funcionalidades estabelecidas nos requisitos iniciais e, com base no *feedback* do supervisor (que atuava como um cliente da aplicação), eram feitas correções nas funcionalidades, melhorando assim o projeto e verificava-se o cumprimento das mesmas.

A solução desenvolvida entregue encontra-se num bom estado, sendo já possível a sua utilização por parte de nutricionistas e pacientes, em que todos os casos de uso propostos inicialmente foram implementados com sucesso.

5. Conclusões

Neste capítulo é feito um resumo do trabalho desenvolvido, havendo um enquadramento dos objetivos concretizados comparativamente com os enunciados, uma identificação às limitações do trabalho desenvolvido e de possíveis desenvolvimentos futuros, bem como uma apreciação final.

5.1. Objetivos concretizados

Na secção 1.2.1 do primeiro capítulo foram identificadas as diversas funcionalidades a ser implementadas neste projeto, sendo elas:

- Sistema de registo/login para pacientes e nutricionistas;
- Sistema de registo diário de refeições com base numa base de dados de alimentos;
- Sistema para criação de refeições customizadas;
- Sistema para criação de planos nutricionais e sua edição;
- Sistema de escolha de pacientes por parte de um nutricionista;
- Permitir a um nutricionista acompanhar os planos dos seus pacientes através de tabelas e gráficos;
- Mostrar mensagens de apoio ao paciente com base no seu desempenho;
- Permitir a um paciente o registo do seu peso diário;
- Sistema de desafios nutricionais para um paciente;
- Sistema de *chat* entre o paciente e o nutricionista.

Todas estas funcionalidades foram desenvolvidas e implementadas, sendo elas relativas aos diversos casos de uso identificados na secção 3.2.1 do capítulo 3 em que:

- O sistema de registo/login para pacientes e nutricionistas diz respeito aos casos de uso 1 e 2;
- O sistema de registo diário de refeições com base na base de alimentos diz respeito ao caso de uso 10;
- O sistema de criação de refeições personalizadas diz respeito ao caso de uso 9;

- O sistema para a criação de planos nutricionais e sua edição diz respeito aos casos de uso 4 e 5;
- O sistema de escolha de pacientes por parte do nutricionista diz respeito aos casos de uso 3 e 7 (sendo que um diz respeito à aceitação do paciente);
- A aba que possui as estatísticas de um plano de um paciente diz respeito ao caso de uso 6;
- A mostragem de mensagens personalizadas é feita no caso de uso 11;
- A secção que possibilita a adição de peso diário diz respeito ao caso de uso 8;
- O sistema de desafios nutricionais diz respeito ao caso de uso 12;
- O sistema de *chat* diz respeito ao caso de uso 13.

Para além das funcionalidades idealizadas inicialmente, foram ainda implementadas mais funcionalidades que após uma análise posterior foram identificadas como importantes, sendo elas a adição de um perfil pessoal (onde irá estar localizada a informação do plano do paciente, a secção para a adição do peso e os desafios nutricionais) bem como a adição de uma página com as estatísticas do plano para a visualização do paciente (que irá conter o *display* da mensagem personalizada que diz respeito ao caso de uso 11).

5.2. Limitações e trabalho futuro

O projeto desenvolvido apresenta soluções a todas as funcionalidades propostas inicialmente, porém é sempre possível adicionar outras funcionalidades das quais os atores da aplicação possam tirar uso. No ramo da nutrição existem sempre muitas variáveis a serem consideradas, o que torna importante o registo de dados em áreas que não foram abordadas nesta aplicação. Seria então possível o desenvolvimento de outras funcionalidades relativas a exercício físico, como a adição de planos de *workouts* recomendados pelos nutricionistas, o registo de exercícios físicos realizados por pacientes e consequentes calorias perdidas pelos mesmos, etc. Podiam ser também adicionadas funcionalidades ainda no âmbito alimentar como o registo do consumo de água diário e aconselhamento sobre o mesmo, ou uma secção com receitas completas criadas por membros da comunidade ou pelos profissionais de nutrição. Relativamente ao fator comunitário, outra das funcionalidades que podia ser implementada seria um sistema de amizade entre utilizadores, o que seria útil para a partilha das receitas mencionadas anteriormente ou para partilhar os progressos individuais, quer a

nível dos desafios nutricionais ou de flutuações no peso. Muitas destas ideias de funcionalidades foram retiradas das aplicações semelhantes, tendo sido mencionadas também na secção 2.1 do capítulo 2.

5.3. Apreciação final

Após a realização deste projeto, a apreciação final foi positiva. Foi possível aplicar diversos dos conhecimentos adquiridos durante a Licenciatura em Engenharia Informática, conhecimentos esses que permitiram a implementação de uma solução viável ao problema proposto inicialmente.

De um ponto de vista pessoal foi uma experiência enriquecedora, pois permitiu uma melhor consolidação dos conhecimentos apreendidos durante a LEI. Possibilitou ainda, adquirir novos conhecimentos ao usar ferramentas e tecnologias não usadas até então bem como um melhor entendimento pessoal relativamente a um contexto mais profissional.

O projeto desenvolvido trata-se de uma solução interessante e funcional face ao problema inicial e às funcionalidades que foram propostas, sendo um projeto pronto a ser utilizado por qualquer paciente que deseje dar início à sua jornada nutricional, ou a qualquer nutricionista que deseja ajudar utilizadores a fazê-lo.

De modo a desenvolver a melhor solução possível e a ultrapassar todos os obstáculos encontrados, foi fundamental a ajuda e apoio do supervisor Diogo Martinho e da orientadora Goreti Marreiros, os quais ajudaram a desenvolver e a aperfeiçoar as capacidades técnicas e profissionais.

Referências

- [1] ANILACT (2019, Novembro): *OCDE indica que 67,6% da população portuguesa é obesa* retirado a 10 de Setembro de 2021 de <https://www.anilact.pt/info/actual/nutricao/item/4269-ocde-indica-que-67-6-da-populacao-portuguesa-e-obesa>
- [2] ITEA3. (N.D.): *Food Friend Project Profile* retirado a 10 de Setembro de 2021 de <https://itea4.org/project/food-friend.html>
- [3] Visual Paradigm Online (N.D.): <https://online.visual-paradigm.com/pt/>
- [4] LifeSum (N.D.): *lifesum app banner*, retirado a 10 de Setembro de 2021 de <https://www.linkedin.com/company/lifesum-app/>
- [5] Lifesum Health App (N.D.): *About Lifesum* retirado a 10 de Setembro de 2021 de <https://lifesum.com/about/>
- [6] Zoshua Colah (2018): *Zoshua Colah - Making it easier to add meals in Lifesum* retirado a 10 de Setembro de 2021 de <https://zoshuacolah.com/making-it-easier-to-add-meals-in-lifesum>
- [7] Yazio (2020, Junho): *Factsheet June 2020* retirado a 10 de Setembro de 2021 de <https://filecontent.yazio.com/press/factsheet-yazio-en.pdf>
- [8] Yazio (N.D.): *Emagrecimento e Alimentação Saudáveis: Perca Peso Rápido com o YAZIO* retirado a 10 de Setembro de 2021 de <https://www.yazio.com/pt>
- [9] Yazio (2021, Março) Foto retirada a 10 de Setembro de 2021 de <https://www.facebook.com/yazio/photos/a.676779235714626/4037412669651249>
- [10] Crunchbase (N.D.): *MyFitnessPal - Crunchbase Company Profile & Funding* retirado a 10 de Setembro de 2021 de <https://www.crunchbase.com/organization/myfitnesspal>
- [11] Perez, S. (2015, Fevereiro): *Under Armour Snatches Up Health And Fitness Trackers Endomondo And MyFitnessPal* retirado a 10 de Setembro de 2021 de <https://techcrunch.com/2015/02/04/athletic-apparel-company-under-armour-snatches-up-health-and-fitness-trackers-endomondo-and-myfitnesspal/>
- [12] Bloomberg (2020, Dezembro): *Under Armour completes sale of the MyFitnessPal platform to Francisco Partners* retirado a 10 de Setembro de 2021 de <https://www.bloomberg.com/press-releases/2020-12-18/under-armour-completes-sale-of-the-myfitnesspal-platform-to-francisco-partners>

- [13] Davis, H. (2020, Fevereiro): *Why MyFitnessPal is the best free fitness app* retirado a 10 de Setembro de 2021 de <https://screenrant.com/best-fitness-health-app-myfitnesspal/>
- [14] Guerreiro, A. (2017, Abril): *6 aplicações gratuitas para não estragar a dieta* retirado a 10 de Setembro de 2021 de <https://www.nit.pt/fit/alimentacao-saudavel/6-aplicacoes-gratuitas-para-nao-estragar-a-dieta/attachment/116625>
- [15] ECMAScript (2021, Setembro): *Draft ECMA-262 / September 9, 2021*, retirado a 10 de Setembro de 2021 de <https://tc39.es/ecma262/#sec-overview>
- [16] W3Techs (N.D.): *Usage statistics of JavaScript as client-side programming language on websites* retirado a 10 de Setembro de 2021 de <https://w3techs.com/technologies/details/cp-javascript/>
- [17] Luiz, A. (2016, Agosto): *JavaScript #1 - Uma breve história da linguagem* retirado a 10 de Setembro de 2021 de <http://shipit.resultadosdigitais.com.br/blog/javascript-1-uma-breve-historia-da-linguagem/>
- [18] Rauschmayer, A. (2014, Janeiro): *Speaking JavaScript: An In-Depth Guide for Programmers. Chapter 1*, Publicado por O'Reilly Media, retirado a 10 de Setembro de 2021 de <http://speakingjs.com/es5/ch01.html# influences and nature of the language>
- [19] Nicácio, R. (2019, Janeiro): *Conheça as principais características do JavaScript*, retirado a 10 de Setembro de 2021 de <https://oportalin10.com.br/conhecas-as-principais-caracteristicas-do-javascript-86379/>
- [20] Brewster, C. (n.d.): *6 Examples of JavaScript: Where and When to Use*, retirado a 10 de Setembro de 2021 de <https://trio.dev/blog/examples-javascript>
- [21] W3Techs (N.D.): *Usage statistics of JavaScript libraries for websites* retirado a 10 de Setembro de 2021 de https://w3techs.com/technologies/overview/javascript_library
- [22] StackOverflow (2020, Fevereiro): *Stack Overflow Developer Survey 2020*, retirado a 10 de Setembro de 2021 de <https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-professional-developers>
- [23] Oracle (N.D.): *What is a database?* retirado a 10 de Setembro de 2021 de <https://www.oracle.com/pt/database/what-is-database/>

- [24] Hughes, A. (2019, Julho): *What is a database? - Definition*, retirado a 10 de Setembro de 2021 de <https://searchdatamanagement.techtarget.com/definition/database>
- [25] MongoDB (N.D.): *The most popular database for modern apps*, retirado a 10 de Setembro de 2021 de <https://www.mongodb.com/>
- [26] Javatpoint (N.D.): *MongoDB Features*, retirado a 10 de Setembro de 2021 de <https://www.javatpoint.com/mongodb-features>
- [27] JSON (N.D.): *Introdução ao JSON*, retirado a 10 de Setembro de 2021 de <http://json.org/json-pt.html>
- [28] MongoDB (N.D.): *Document Schemas*, retirado a 10 de Setembro de 2021 de <https://docs.mongodb.com/realm/mongodb/document-schemas/>
- [29] MongoDB (N.D.): *Database Schema Example*, retirado a 10 de Setembro de 2021 de <https://www.mongodb.com/scale/database-schema-example>
- [30] Node.js (N.D.): *About Node.js*, retirado a 10 de Setembro de 2021 de <https://nodejs.org/en/about/>
- [31] Node.js (N.D.): *Logos and Graphics*, retirado a 10 de Setembro de 2021 de <https://nodejs.org/en/about/resources/>
- [32] training.com (2016, Setembro): *About Node.js, and why you should add Node.js to your skill set?* retirado a 10 de Setembro de 2021 de <http://blog.training.com/2016/09/about-nodejs-and-why-you-should-add.html>
- [33] lenon (2018, Setembro): *Node.js - O que é, como funciona e quais as vantagens*, retirado a 10 de Setembro de 2021 de <https://www.opus-software.com.br/node-js/>
- [34] Butusov, M. (2020, Outubro): *Companies that use Node.js for backend: how do big players benefit from it?* retirado a 10 de Setembro de 2021 de <https://www.techmagic.co/blog/companies-that-use-node-js-for-backend-how-do-big-players-benefit-from-it/>
- [35] StackOverflow (2020, Fevereiro): *Stack Overflow Developer Survey 2020*, retirado a 10 de Setembro de 2021 de <https://insights.stackoverflow.com/survey/2020#technology-other-frameworks-libraries-and-tools-all-respondents3>
- [36] Express (N.D.): *Node.js web application framework*, retirado a 10 de Setembro de 2021 de <https://expressjs.com/>

- [37] Holowaychuk, TJ (2010, Julho): *Express1.0beta*, retirado a 10 de Setembro de 2021 de <https://web.archive.org/web/20150706050636/https://tjholowaychuk.tumblr.com/post/820103177/express-1-0beta>
- [38] Express (N.D.): *Companies using Express in production*, retirado a 10 de Setembro de 2021 de <https://expressjs.com/en/resources/companies-using-express.html>
- [39] JSConf (2013, Agosto): *[JSConfUS 2013] Tom Occhino and Jordan Walke: JS Apps at Facebook*, retirado a 10 de Setembro de 2021 de [\[JSConfUS 2013\] Tom Occhino and Jordan Walke: JS Apps at Facebook](#)
- [40] React Native (2021, Agosto): *JavaScript Environment – React Native*, retirado a 10 de Setembro de 2021 de <https://reactnative.dev/docs/javascript-environment>
- [41] GeeksForGeeks (2021, Julho): *React.js (Introduction and Working)*, retirado a 10 de Setembro de 2021 de <https://www.geeksforgeeks.org/react-js-introduction-working/>
- [42] Reis, J. (2020, Fevereiro): *Angular vs React: A comparison of both frameworks*, retirado a 10 de Setembro de 2021 de <https://www.imaginarycloud.com/blog/angular-vs-react/#React>
- [43] AnyforSoft (2021, Maio): *10 Famous websites built with React JS*, retirado a 10 de Setembro de 2021 de <https://anyforsoft.com/blog/10-famous-websites-built-react-js/>
- [44] StackOverflow (2020, Fevereiro): *Stack Overflow Developer Survey 2020*, retirado a 10 de Setembro de 2021 de <https://insights.stackoverflow.com/survey/2020#technology-web-frameworks>
- [45] Wagner, B. et al. (2021, Agosto): *A tour of the C# language*, retirado a 10 de Setembro de 2021 de <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>
- [46] Ganesh, G. (2017, Maio): *C# and its features*, retirado a 10 de Setembro de 2021 de <https://www.c-sharpcorner.com/article/C-Sharp-and-its-features/>
- [47] Pedamkar, P (2021, Março): *C# vs JavaScript: Top 8 useful comparisons to learn*, retirado a 10 de Setembro de 2021 de <https://www.educba.com/c-sharp-vs-javascript/>
- [48] GeeksForGeeks (2020, Novembro): *Difference between JavaScript and C Sharp*, retirado a 10 de Setembro de 2021 de <https://www.geeksforgeeks.org/difference-between-javascript-and-c-sharp/>
- [49] Krajewski, R. (2021, Julho): *C# vs JavaScript: Which programming language is better for your needs?* retirado a 10 de Setembro de 2021 de <https://www.ideamotive.co/blog/c-sharp-vs-javascript>

- [50] Angular (2021, Março): *What is Angular?* retirado a 10 de Setembro de 2021 de <https://angular.io/guide/what-is-angular>
- [51] Angular (N.D): *Dependency injection in Angular*, retirado a 10 de Setembro de 2021 de <https://angular.io/guide/dependency-injection>
- [52] Srivastava, S. (2021, Agosto): *React vs Angular – the best choice for mobile app development*, retirado a 10 de Setembro de 2021 de <https://appinventiv.com/blog/react-vs-angular/>
- [53] San José State University (N.D.): *Identifying non-functional requirements*, retirado a 10 de Setembro de 2021 de <http://www.cs.sjsu.edu/faculty/pearce/modules/lectures/ooa/requirements/IdentifyingURPS.htm>
- [54] Kruchten, P. (1995, Novembro): *Architectural Blueprints – the “4+1” view model of software architecture*, retirado a 10 de Setembro de 2021 de <https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf>
- [55] Kontio, M. (2005, Fevereiro): *Architectural manifesto: Designing software architectures, Part 5*, retirado a 10 de Setembro de 2021 de <https://pt.scribd.com/document/84656947/Architectural-manifesto-Designing-software-architectures-Part-5>
- [56] Silva, N. (2020, Dezembro): *ARQSI: bulletproof-nodejs+ddd*, retirado a 10 de Setembro de 2021 de <https://bitbucket.org/nunopsilva/bulletproof-nodejs-ddd/wiki/Views>
- [57] C4Model (N.D.): *The C4 model for visualizing software architecture*, retirado a 10 de Setembro de 2021 de <https://c4model.com/>

Anexo A – Diagramas

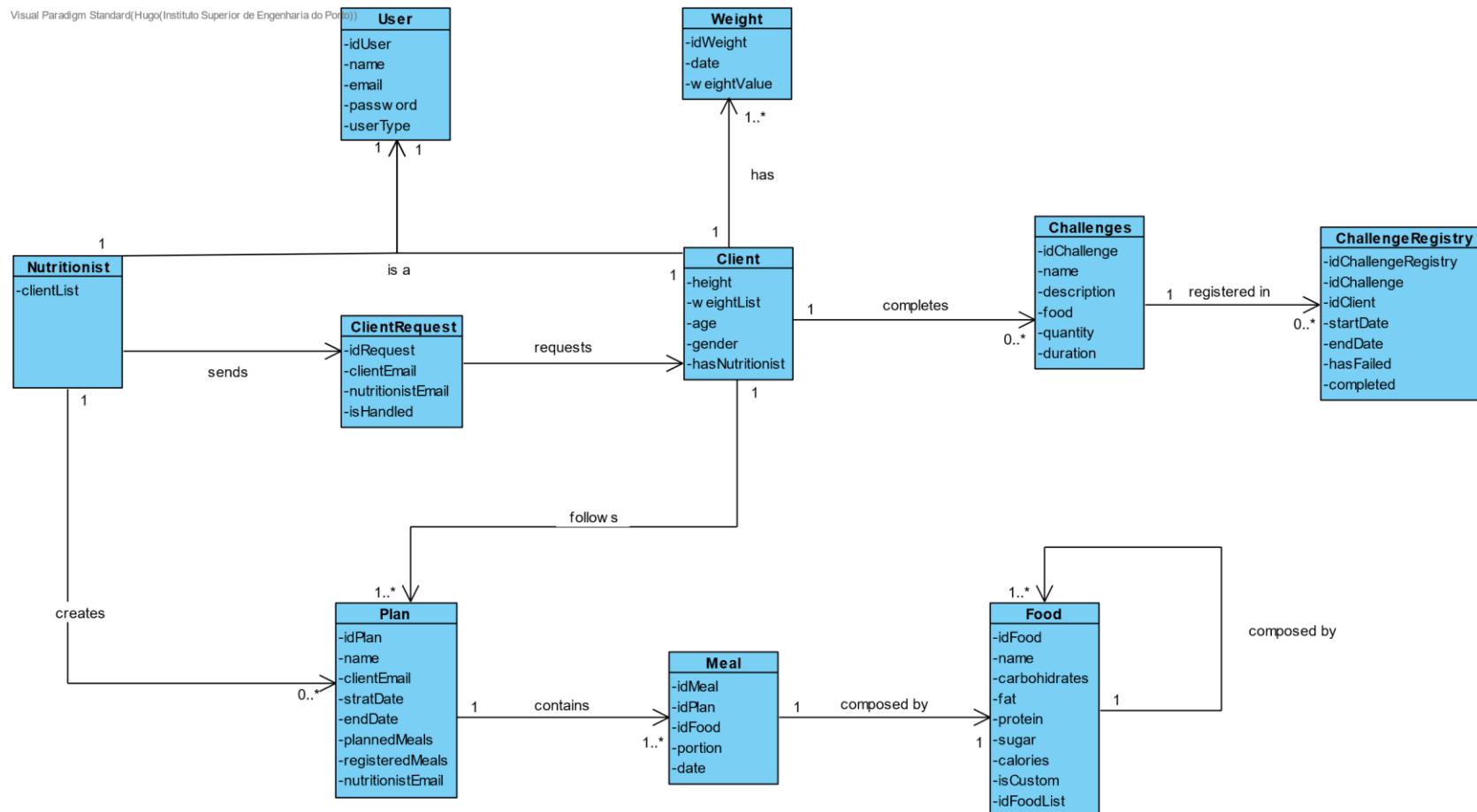


Figura 74 - Modelo de domínio

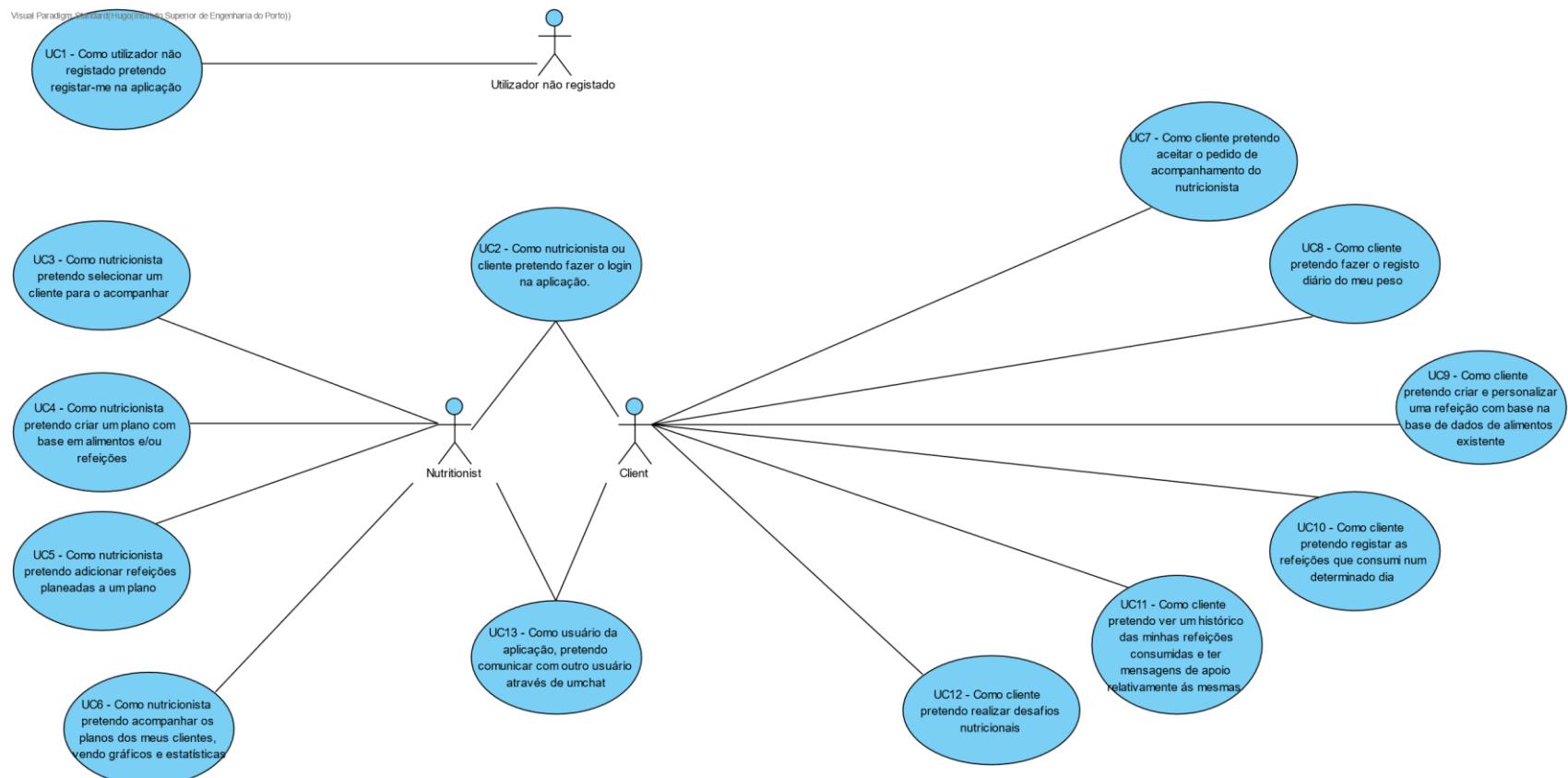


Figura 75 - Diagrama de casos de uso

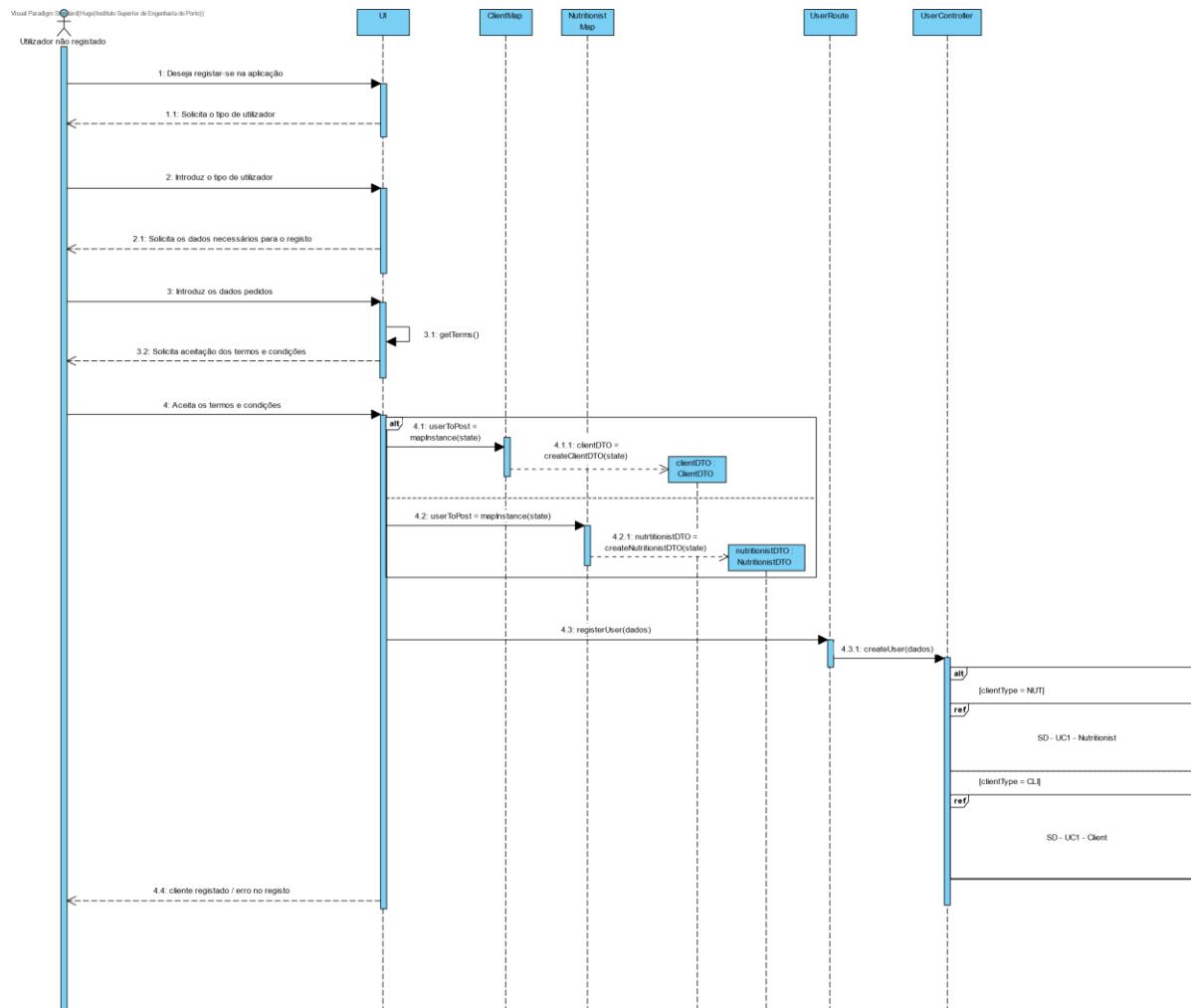


Figura 76 - SD do caso de uso 1

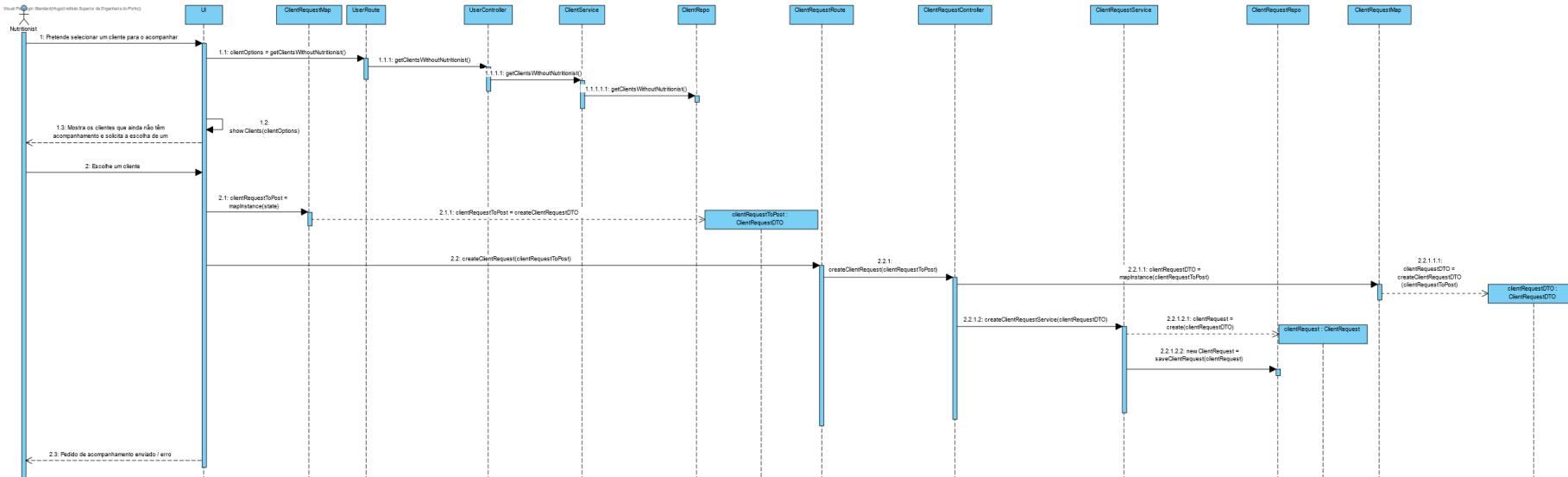


Figura 77 - SD do caso de uso 3

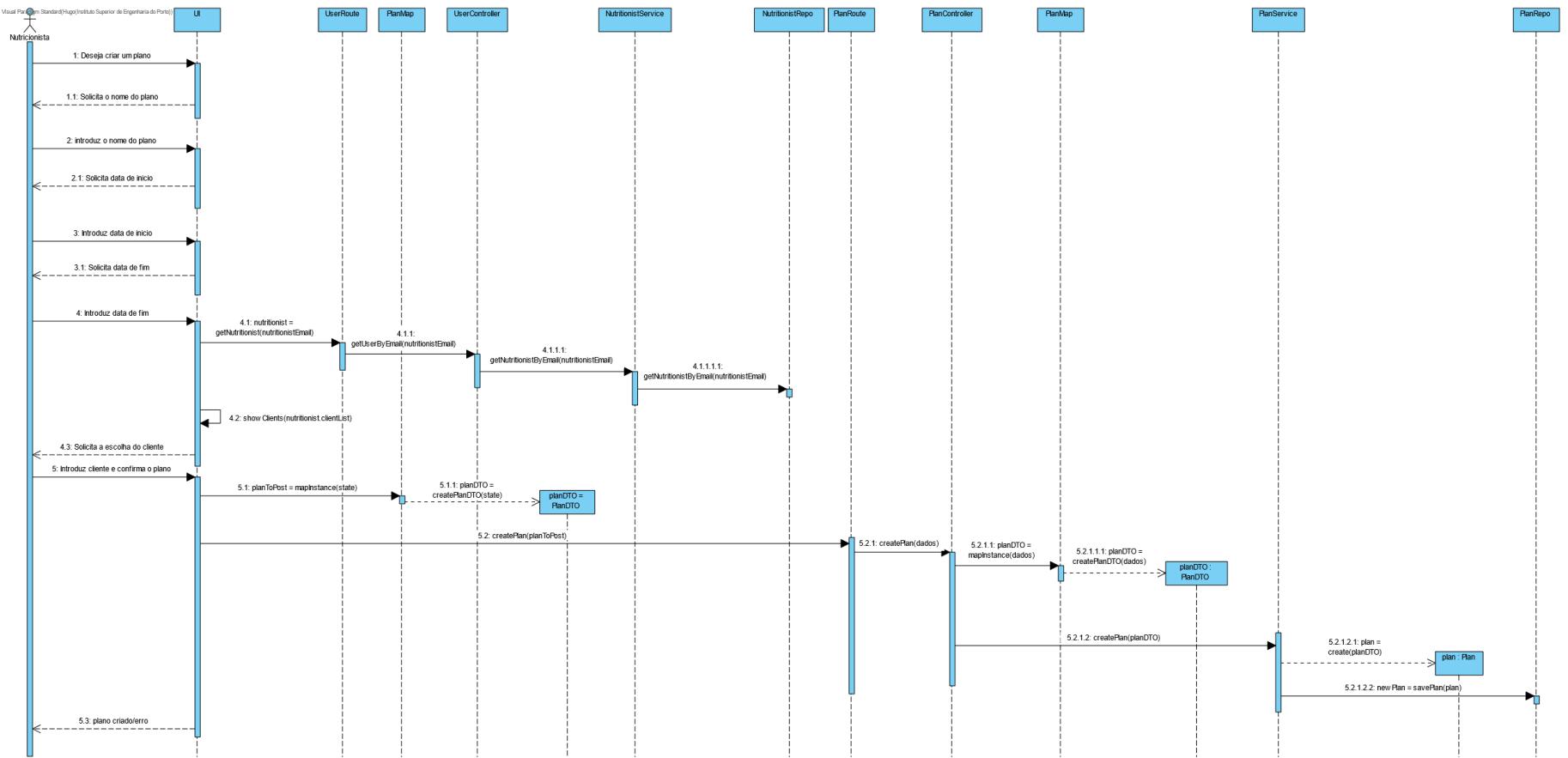


Figura 78 - SD do caso de uso 4

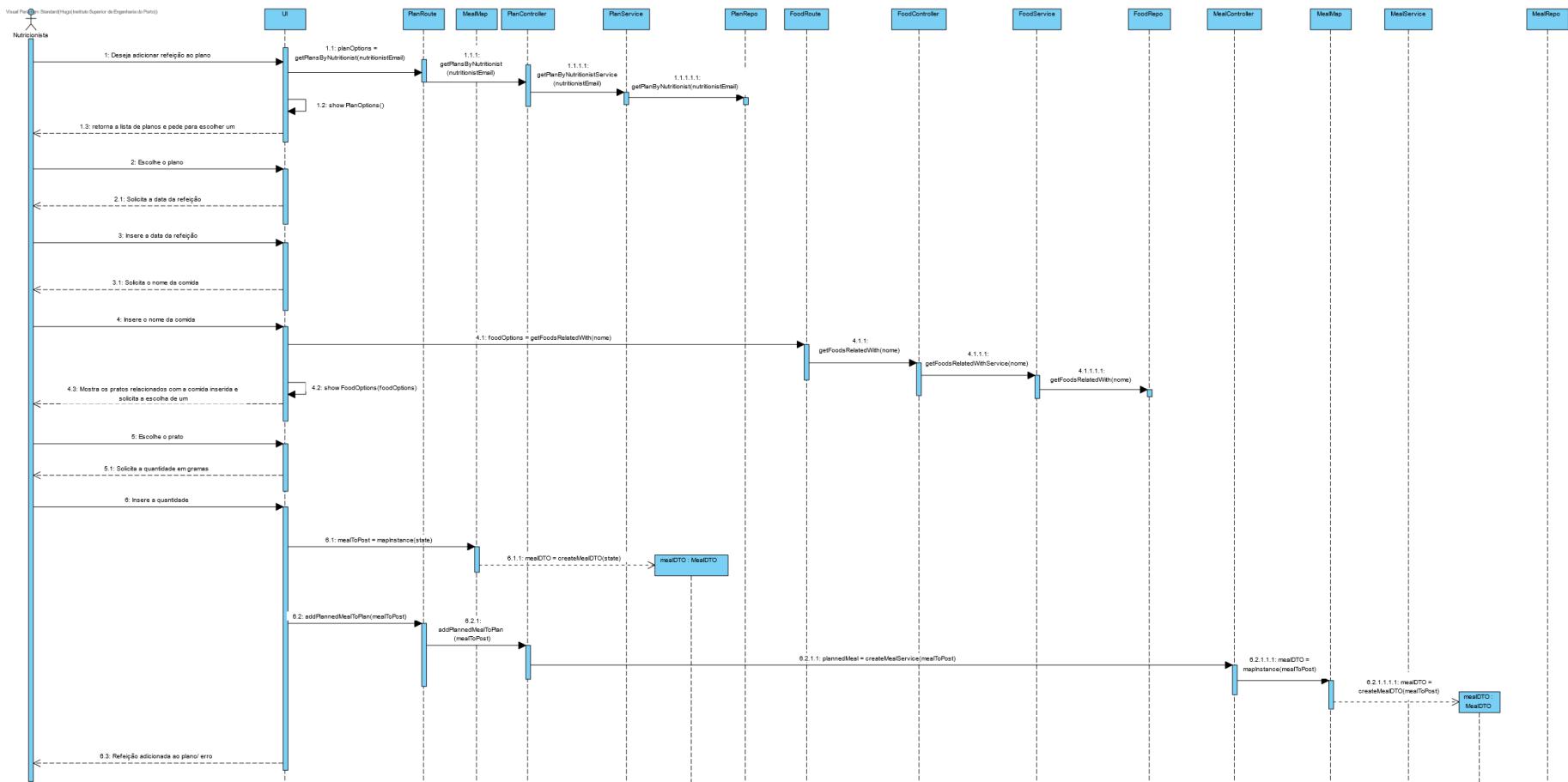


Figura 79 - SD do caso de uso 5

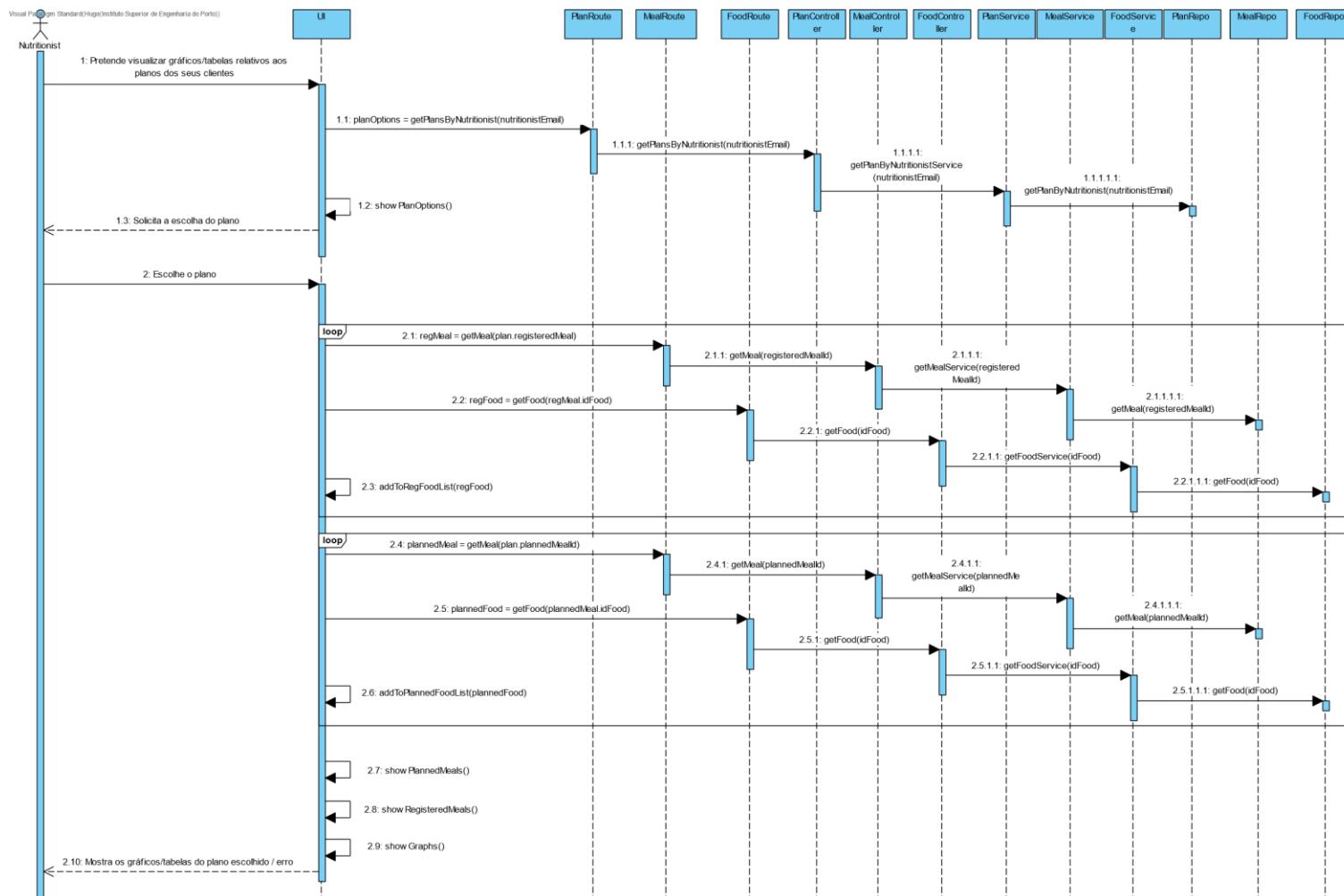


Figura 80 - SD do caso de uso 6

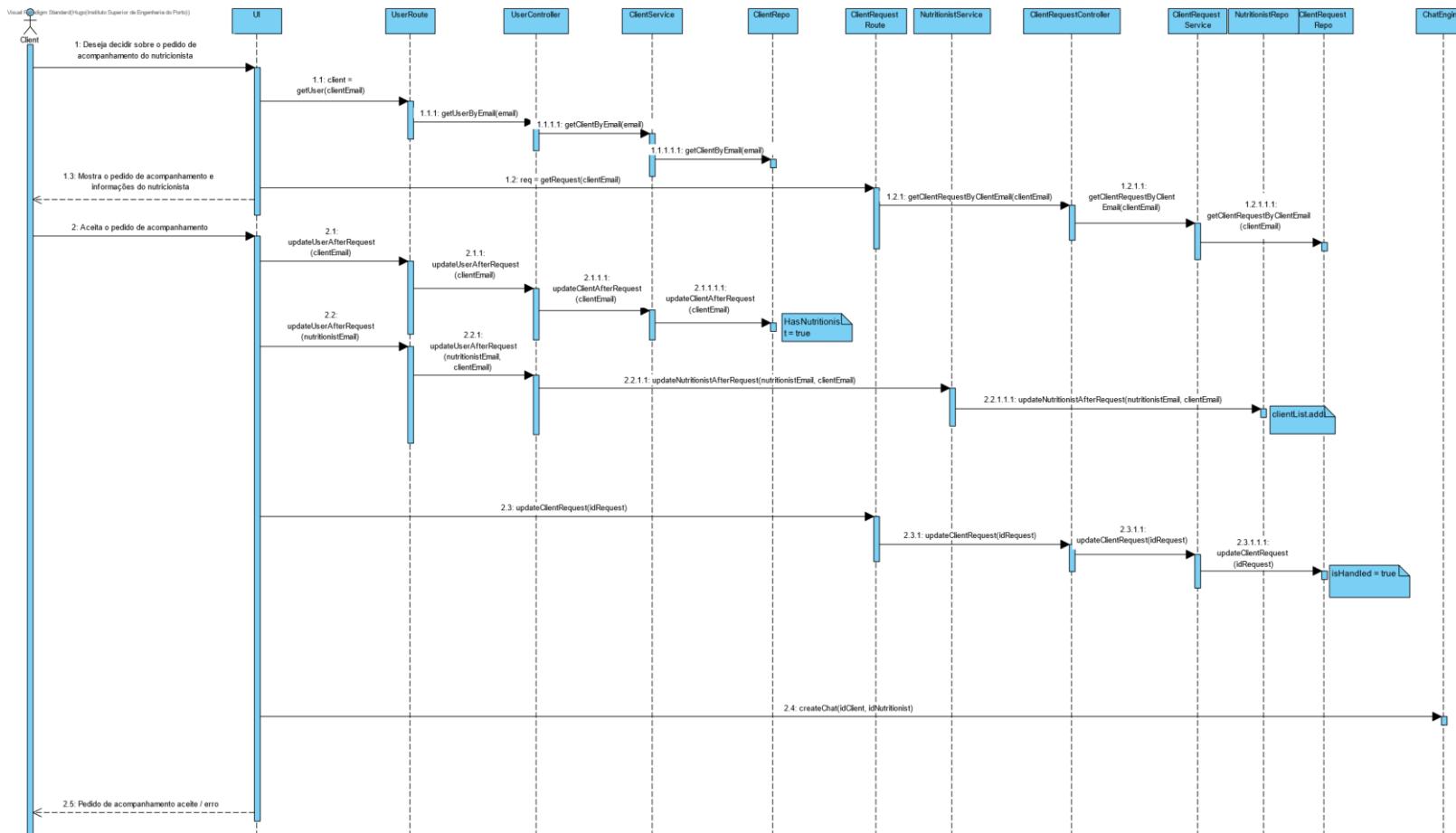


Figura 81 - SD do caso de uso 7

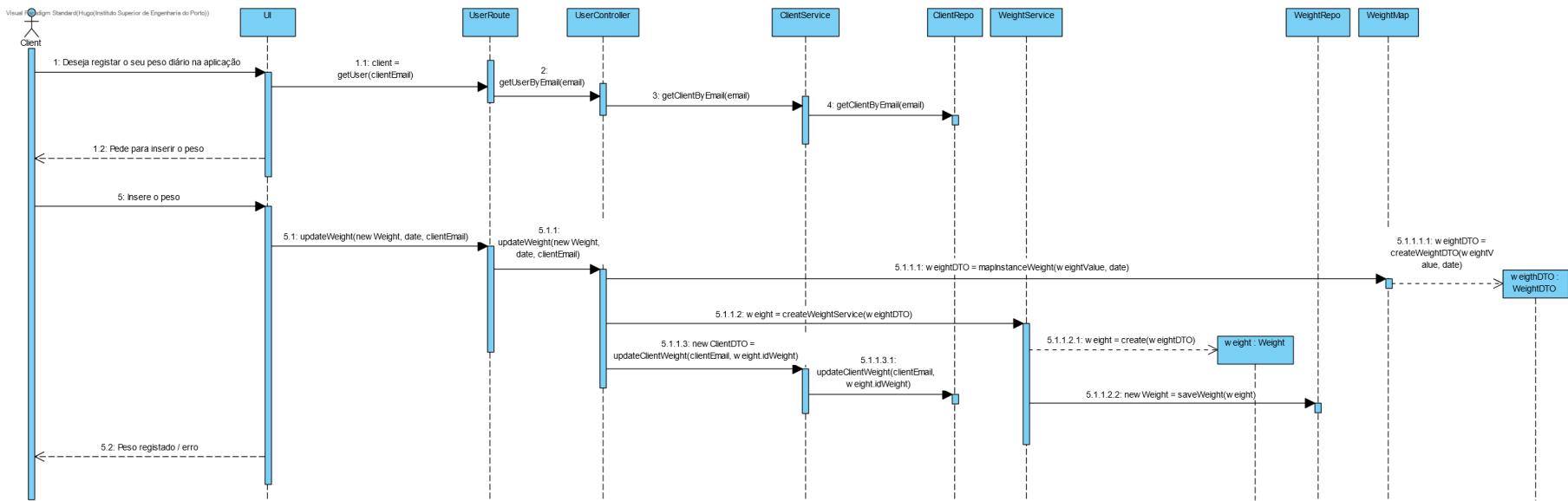


Figura 82 - SD do caso de uso 8

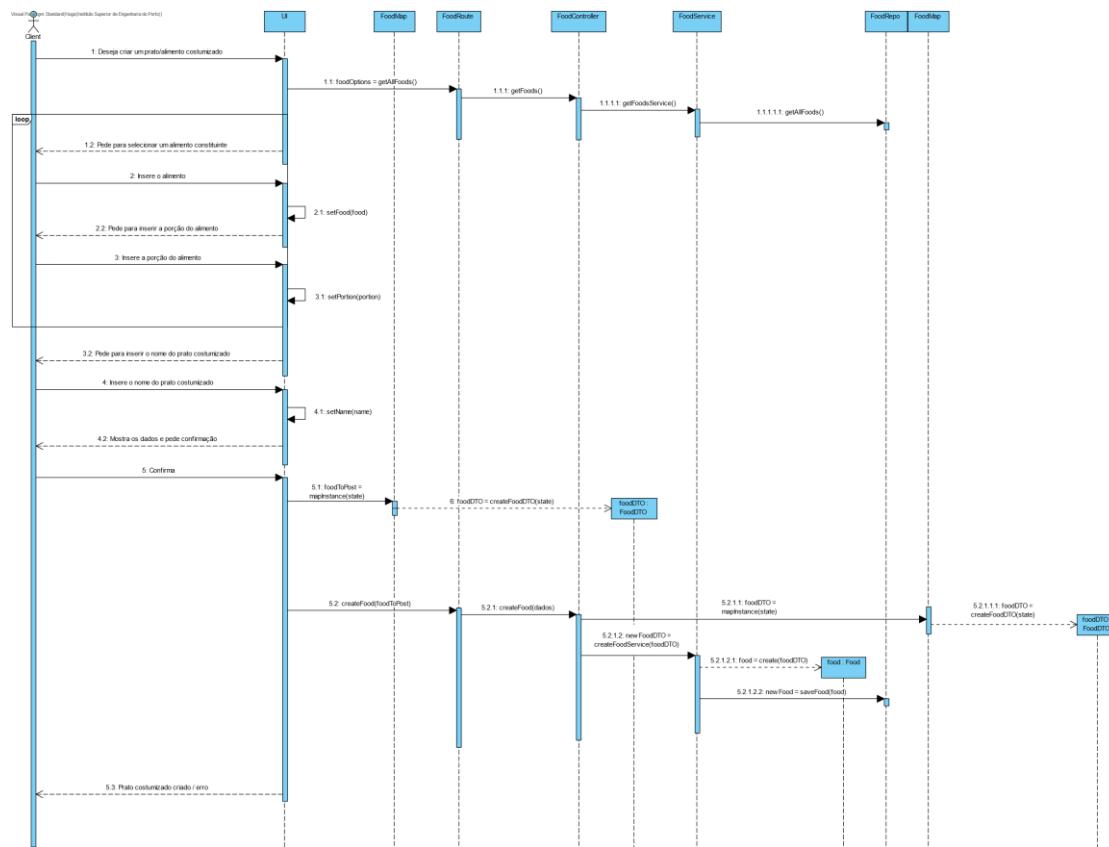


Figura 83 - SD do caso de uso 9

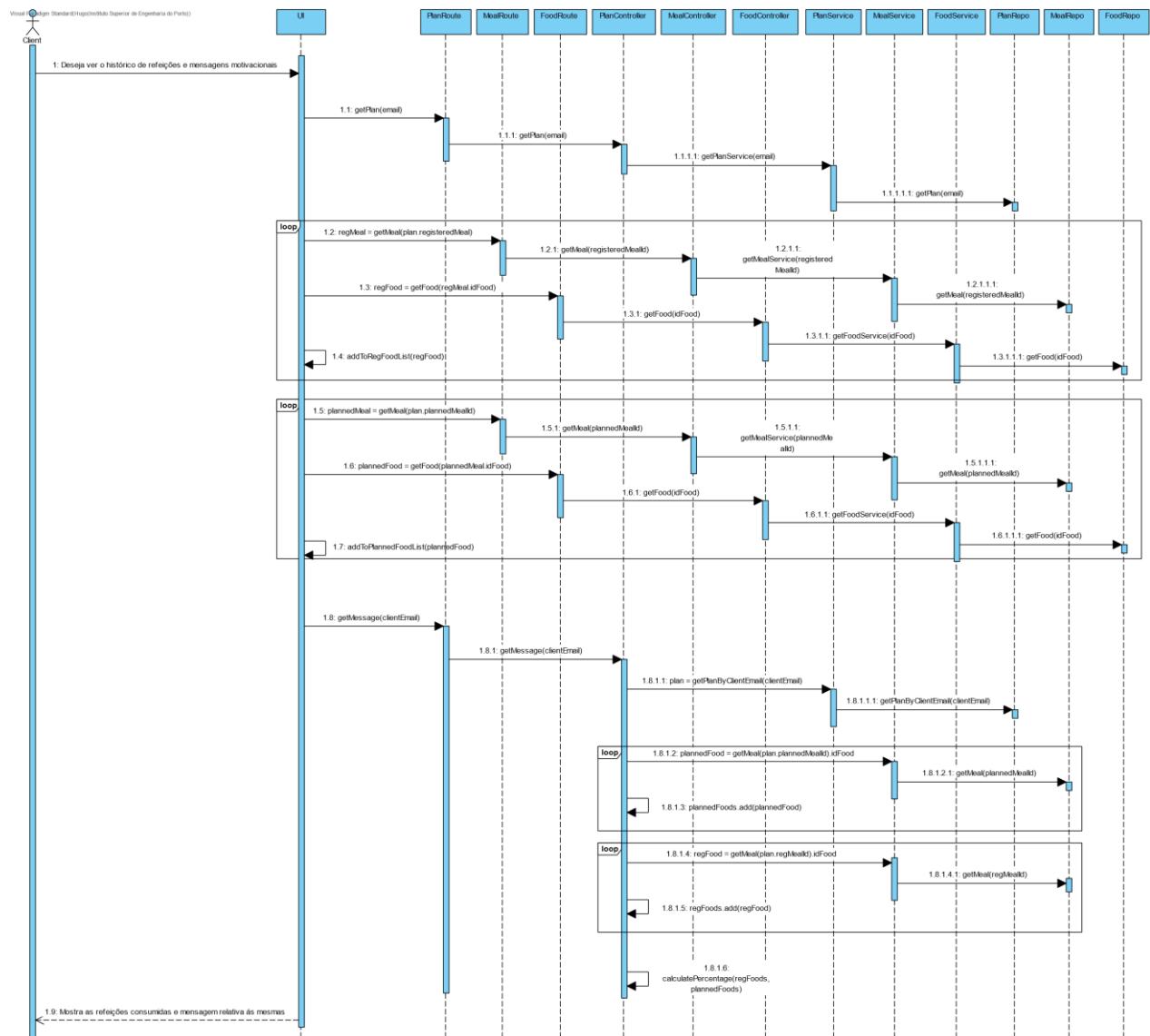


Figura 84 - SD do caso de uso 11

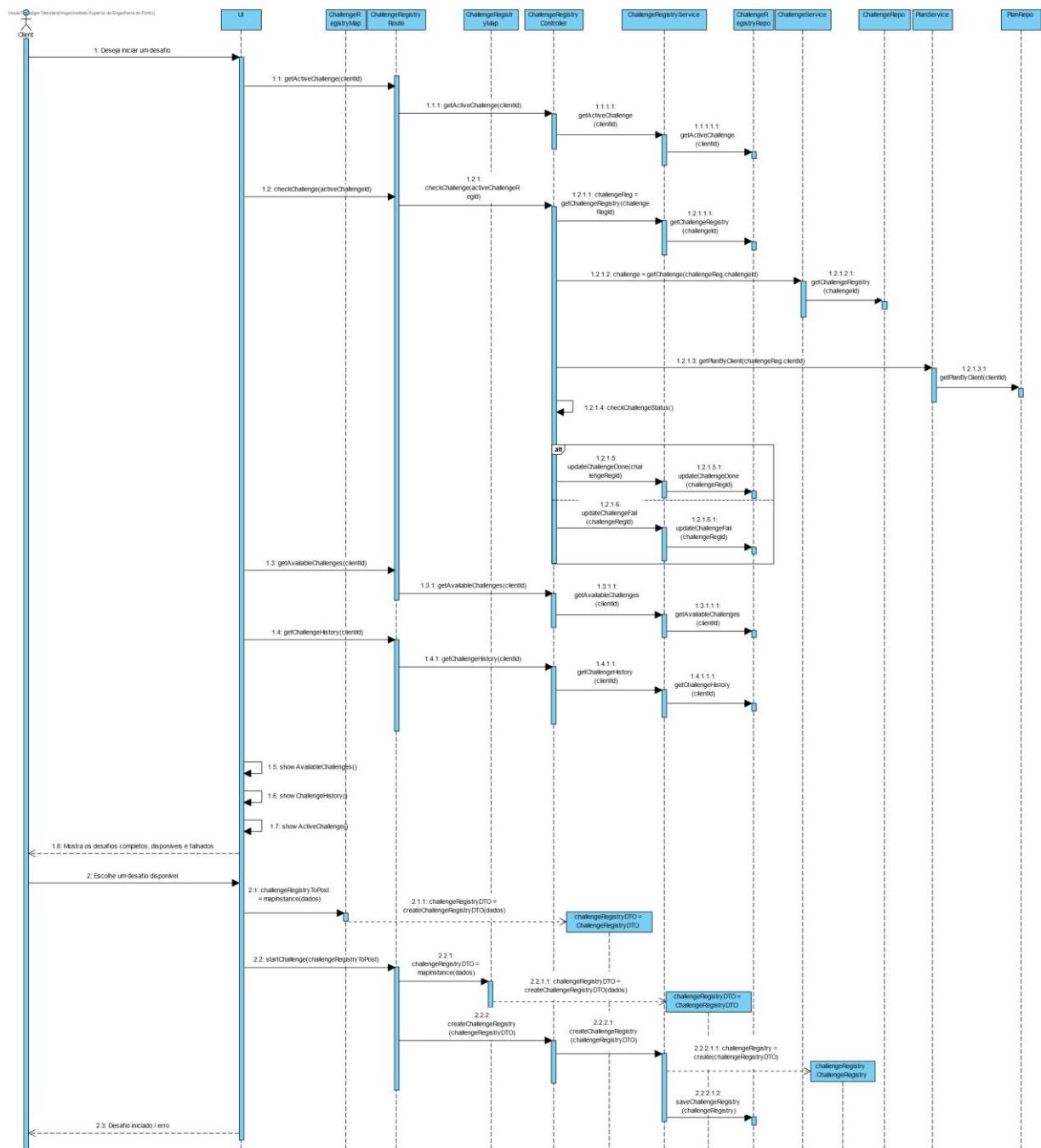


Figura 85 - SD do caso de uso 12