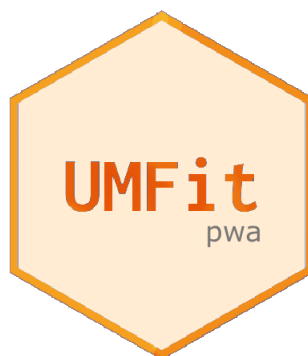




DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA

LABORATÓRIOS DE INFORMÁTICA 4



Autores:

Paulo LIMA, A89983

Pedro MACHADO, A83719

João AZEVEDO, A85227

Paulo ARAÚJO, A85729

Hugo CUNHA, A84656

ANO LETIVO 2019-2020

RESUMO

Este relatório foi elaborado no âmbito do desenvolvimento de um assistente de *Fitness*, uma aplicação multi plataforma para um ginásio que ajuda e guia os utilizadores na sua rotina diária de exercícios e no seu plano alimentar, estabelecidos pelo instrutor.

A primeira etapa compreendeu a apresentação do domínio da aplicação, bem como a apresentação do sistema que nos propusemos a desenvolver. De seguida, foi desenvolvido o Modelo de Domínio do sistema, os seus Use Cases e por último um plano de futuro desenvolvimento nas suas próximas fases.

Os objectivos que se seguiram marcaram o início do processo de materialização das ideias e objetivos apresentados. Este início foi marcado com o desenvolvimento do modelo Conceptual e Lógico da Base de Dados, especificação de Use Cases, criação de *Views* para a ferramenta a utilizar e o Diagrama de Classes do sistema.

Para além destas tarefas indicadas, o grupo explorou também várias tecnologias para a implementação da aplicação.

Área de Aplicação: Serviços de assistência de *Fitness*.

Palavras-Chave: Engenharia de Software, Ginásio, Microsoft, Base de Dados.

CONTEÚDO

1	Resumo	2
2	Introdução	8
2.1	Contextualização	8
2.2	Apresentação do Caso de Estudo	8
2.3	Motivação e Objetivos	9
2.4	Plano de Desenvolvimento	10
2.5	Estrutura do Relatório	11
3	Levantamento de Requisitos	13
3.1	Requisitos Funcionais	13
3.1.1	Autenticação de utilizadores	13
3.1.2	Criação planos de treino e alimentares	14
3.1.3	Consultar planos de treino e alimentares	14
3.1.4	Criação de metas [EXTRA]	14
3.1.5	Marcar presença num espaço [EXTRA]	15

3.1.6	Apresentação de lotação dos diferentes espaços do ginásio [EXTRA]	15
3.1.7	Consultar estatísticas de lotação por horário [EXTRA]	15
3.2	Requisitos Não Funcionais	16
4	Modelo de Domínio	17
5	Diagrama de Use Cases	19
6	Especificação de Use Cases	21
6.1	Consultar Aulas de Grupo	21
6.2	Consultar Evolução	21
6.3	Consultar Plano Alimentar	22
6.4	Consultar Plano de Treino	22
6.5	Consultar resultado da última avaliação de um dado utilizador	23
6.6	Consultar Resultado da última avaliação	24
6.7	Consultar Última Avaliação	24
6.8	Criar Plano alimentar para Cliente	25
6.9	Criar Plano de treino para Cliente	25
6.10	Criar resultado de avaliação para Cliente	26
6.11	Criar utilizador	27

6.12	Editar planos de aulas semanal	28
6.13	Log In	29
6.14	Log Out	30
6.15	Marcar presença na aula	30
6.16	Remover Utilizador	31
7	Arquitetura da Camada de Negócios	32
7.1	Camada de Negócios	32
7.2	Dicionário das principais Classes	32
7.3	Descrição da arquitetura	34
7.4	Diagrama de Classes	35
8	Camada de Dados	36
8.1	Modelo Conceptual	36
8.2	Diagrama do Modelo Conceptual	38
8.3	Modelo Lógico	38
8.4	Diagrama do Modelo Lógico	39
9	Propostas de Interface	40

10 Metodologia de Implementação	43
11 Ferramentas utilizadas na implementação	44
11.1 Desenho e gestão da Base de dados	44
11.2 <i>Back-end</i>	45
11.3 <i>Front-end</i>	45
12 Desenvolvimento do Projeto	47
12.1 Mecanismo de Sessão	47
12.2 API	49
12.3 Conexão da base de dados	51
12.4 Interfaces	52
13 Produto Final	53
13.1 Página Inicial (pré autenticação)	54
13.2 Página Cliente Standard	56
13.3 Página Cliente Premium	61
13.4 Página Instrutor Autenticado	63
13.5 Página Recepcionista Autenticado	66
13.6 Deploy da API	72

13.7 Deploy da Interface	72
13.8 Progressive Web App (pwa)	73
14 Conclusão	76

INTRODUÇÃO

2.1 CONTEXTUALIZAÇÃO

O número de ginásios está a aumentar e estima-se que em 2017 a indústria tenha arrecadado cerca de 73 mil milhões de euros a nível mundial, assumindo, assim, um factor de relevo na economia mundial.

Além do exercício físico, o seguimento nutricional dos seus utilizadores juntamente com um aumento no número de planos de baixo custo faz com que o mercado proveniente de ginásios aumente.

2.2 APRESENTAÇÃO DO CASO DE ESTUDO

A **UMFit** é um software de apoio e orientação *Fitness* desenvolvido no âmbito da unidade curricular de Laboratórios de Informática IV. O software arquitectado é ambicioso e, por esta razão, todo o processo que constitui a sua criação foi extremamente rigoroso. Para o desenvolvimento do **UMFit** foram exigidos os melhores padrões e metodologias de trabalho com vista à rotina dos clientes do ginásio ser facilitada, aumentando assim, a satisfação dos mesmos.

Este software tem alguns objectivos muito bem definidos, nomeadamente o de proporcionar aos seus utilizadores experiências de utilização do ginásio mais simples e cómodas, para além de unificar o exercício físico à alimentação mais adequada, consoante os objectivos definidos por cada cliente.

2.3 MOTIVAÇÃO E OBJETIVOS

Os ginásios disponibilizam aos seus clientes uma grande variedade de serviços. Tais como, aulas de grupo, serviços de nutrição e planeamento de treinos.

A aplicação permite auxiliar os clientes do ginásio para que estes alcancem os seus objectivos, por exemplo, chegar à sua forma ideal de forma mais rápida e saudável. Para além disso, permite que, de uma forma centralizada, os utilizadores possam ver receitas dos seus planos alimentares. Deste modo, todos os diversos exercícios e planos alimentares do cliente estarão disponíveis através dos seus dispositivos moveis e também em qualquer outra plataforma, por um navegador Internet.

A apresentação ao Utilizador das aulas de grupo pretendidas, com o respectivo número de participantes, é apenas umas das muitas funcionalidades que a aplicação apresenta. Depois de entrar pela primeira vez na aplicação, o utilizador tem a oportunidade de definir uma configuração inicial para o sistema, escolhendo, por exemplo, dentro de uma das aulas de grupo disponíveis, uma que pretenda frequentar. Este poderia também ao longo do seu percurso agendar avaliações consoante a disponibilidade dos instrutores[Funcionalidade Extra].

2.4 PLANO DE DESENVOLVIMENTO

O plano de desenvolvimento do projeto seguiu um trajeto muito semelhante ao apresentado na figura 1.

Como é possível verificar, a fase inicial de requisitos da aplicação teve uma breve duração em comparação com a implementação.

Assim, é preciso notar que o desenvolvimento de interfaces teve uma longa duração devido à falta de experiência com ferramentas e linguagens necessárias para o desenvolvimento desta aplicação *web*.

Por último, realçar que o modelo lógico foi sendo atualizado consoante as necessidades da API e, por isso mesmo, também acabou por ter uma longa duração.

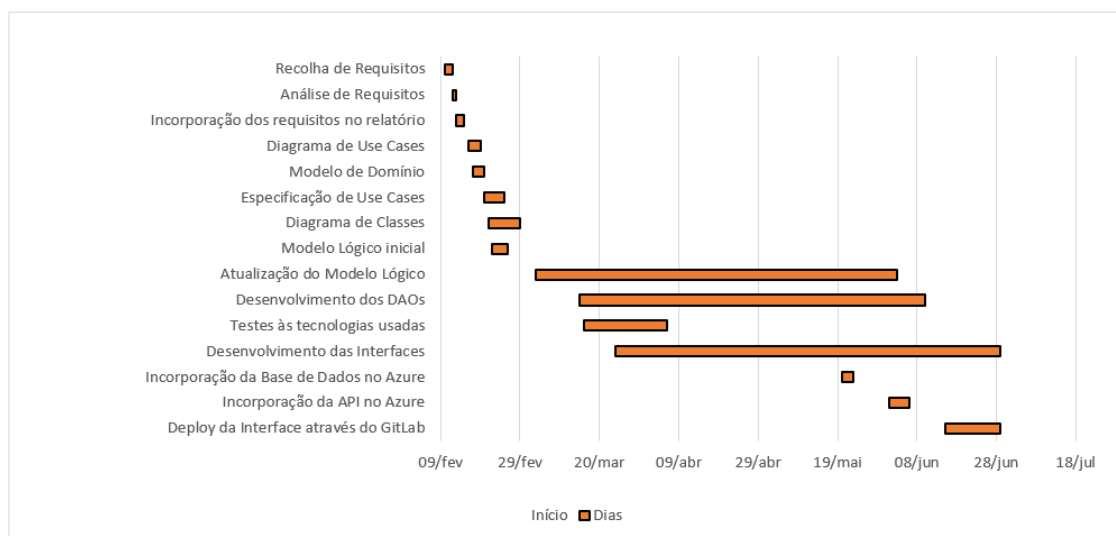


Figura 1: Diagrama de Gantt

2.5 ESTRUTURA DO RELATÓRIO

O presente relatório encontra-se dividido em 14 capítulos nos quais são apresentadas informações sucinta e pertinentes referentes a cada um.

O primeiro capítulo representa um pouco do resumo inicial do projeto e o que foi realizado em termos de estudo aplicativo.

No segundo surge tudo o que se relaciona com a elaboração inicial do projeto desde a sua contextualização, apresentação do caso de estudo, as motivações e objetivos e ainda todo o plano de desenvolvimento. Etapas cruciais na fase inicial para uma boa definição do que a aplicação iria representar e suportar por onde o grupo tentou se guiar o mais fielmente possível.

O terceiro capítulo contém a definição de todos os requisitos funcionais e não funcionais que a aplicação irá suportar. Friso a existência de requisitos extras, os quais o grupo definiu sem se comprometer a realizar obrigatoriamente, ou seja, apenas seriam realizados se o grupo se encontrasse com algum tempo extra.

Ainda no quarto, quinto e sexto capítulos apresentámos informações relativas à conceção lógica do nosso software. Neste capítulos foram apresentadas as definições do modelo de domínio, os diagramas dos use cases e ainda a especificação dos mesmos. Capítulos esses que nos conferiram uma vantagem enorme quanto à definição da estrutura do trabalho e que nos obrigou a olhar para a *big picture* do projeto.

Agora no sétimo capítulo surge o modelo arquitetural propriamente dito, ou seja, a arquitetura da camada de negócios. Que nos permitiu perceber qual o estilo de arquitetura a adotar, quais os componentes do sistema e quais as suas interfaces e ainda a composição de elementos estruturais e comportamentais. Para tal foram definidas a camada de negócios, o dicionário das principais classes, a descrição da arquitetura e o diagrama de classes.

No capítulo oito surge toda a estrutura de implementação relacionado com a base de dados desde a definição do modelo conceptual à definição do modelo lógico, e consequente apresentação dos diagramas dos mesmos.

No nono capítulo surge as propostas de interface e tudo o que o grupo pensou ser fulcral para o desenho das mesmas para um ambiente de interação mais saudável para o utilizador.

Agora no capítulo dez surge a metodologia de implementação que se refere basicamente à estruturação em camadas do nosso software e como se iria realizar a comunicação entre as mesmas.

O capítulo onze vem completar o anterior e refere que ferramentas foram utilizadas para cada camada da aplicação e o porquê dessas mesmas escolhas.

Já no capítulo doze é explicado alguns processos que achamos pertinentes e importantes no desenvolvimento do projeto. Foram feitos inúmeros processos mas enumerá-los todos um a um neste capítulo só o tornaria extenso e maçador. Sendo assim podemos encontrar explicações referentes ao mecanismo de sessão, API, conexão da base de dados e ainda informações referentes às interfaces.

No capítulo treze é apresentado o resultado final da aplicação já com todas as funcionalidades operacionais e imagens ilustrativas de um pouco da execução das mesmas. Também cada uma é acompanhada de um breve texto com uma pequena explicação de como utilizar e para que serve a funcionalidade.

Por fim, no último capítulo apresentámos a conclusão do projeto. Aqui falamos um pouco sobre o que representou para nós este projeto, quer a nível de contra-tempos e obstáculos quer a nível de preparação para projetos futuros.

LEVANTAMENTO DE REQUISITOS

A fase de especificação, consistiu no levantamento de requisitos para o desenvolvimento do sistema da **UMFit**. Este processo teve por base a experiência própria de elementos do grupo em ginásios e na utilização de aplicações semelhantes. Foi também feita uma análise a várias aplicações de ginásios e de *Fitness*, para conseguir distinguir as funcionalidades essenciais que distinguem a **UMFit** das plataformas da mesma área de negócio. Algumas aplicações analisadas foram “MyFitnessPal”, “Academia Fitness BVFamalicão”, “Gym WP”, entre outras.

3.1 REQUISITOS FUNCIONAIS

3.1.1 AUTENTICAÇÃO DE UTILIZADORES

Definição de requisitos de utilizador

1. O utilizador deverá introduzir o seu e-mail e a respetiva password para iniciar sessão na aplicação.

Especificação de requisitos de sistema

- 1.1. O sistema deve verificar a validade da tentativa de autenticação, verificando se o par e-mail e password inseridos correspondem a algum utilizador existente no sistema. Caso não correspondam, o sistema não deve permitir a autenticação.

3.1.2 CRIAÇÃO PLANOS DE TREINO E ALIMENTARES

Definição de requisitos de utilizador

1. O instrutor deve inserir as opções "Criar Plano Alimentar" ou "Criar Plano de Treino" e inserir o e-mail do respetivo cliente para atribuir o respetivo plano.

Especificação de requisitos de sistema

- 1.1. O sistema deve verificar a validade do e-mail de cliente escolhido e adicionar o plano a esse utilizador. Caso o e-mail de cliente será impossível adicionar o plano ao mesmo.

3.1.3 CONSULTAR PLANOS DE TREINO E ALIMENTARES

Definição de requisitos de utilizador

1. O cliente deve inserir as opções "Consultar Plano alimentar" ou "Consultar Plano de Treino" de modo a visualizar o respetivo plano.

Especificação de requisitos de sistema

- 1.1. O Sistema apresenta os dados do plano alimentar ou de treino. Caso o cliente não tenha um plano este não é mostrado.

3.1.4 CRIAÇÃO DE METAS [EXTRA]

Definição de requisitos de utilizador

1. O cliente deve inserir a opção "Criar Meta" e escrever a meta pretendida.

Especificação de requisitos de sistema

- 1.1. O Sistema guarda a meta do cliente.

3.1.5 MARCAR PRESENÇA NUM ESPAÇO [EXTRA]

Definição de requisitos de utilizador

1. Um cliente deve declarar a presença numa certa área do ginásio.

Especificação de requisitos de sistema

- 1.1. O Sistema atualiza a lotação na área selecionada.

3.1.6 APRESENTAÇÃO DE LOTAÇÃO DOS DIFERENTES ESPAÇOS DO GINÁSIO [EXTRA]

Definição de requisitos de utilizador

1. O cliente deve inserir a opção "Consultar lotação" e escolher uma área de modo a visualizar as estatísticas.

Especificação de requisitos de sistema

- 1.1. O sistema disponibiliza a lotação atual ao cliente-

3.1.7 CONSULTAR ESTATÍSTICAS DE LOTAÇÃO POR HORÁRIO [EXTRA]

Definição de requisitos de utilizador

1. O cliente deve inserir a opção "Consultar Lotação" e escolher um horário de modo a visualizar as estatísticas.

Especificação de requisitos de sistema

- 1.1. O Sistema apresenta os dados da consulta.

3.2 REQUISITOS NÃO FUNCIONAIS

1. Um utilizador pode ser Premium ou Standard.
2. Os Premium têm acesso a todas as funcionalidades do ginásio referidas acima.
3. Os utilizadores Standard podem consultar e participar nas aulas de grupo, bem como usufruir das zonas de "cardio", musculação e piscina do ginásio. Contudo, estes últimos não têm acesso a planos de treino e/ou alimentares mas podem, no entanto, visualizar a sua última avaliação e evolução.
4. A última avaliação de um cliente deve apresentar a data e todas as medidas de perímetro e composição corporal, incluindo um gráfico da evolução, adequando a interface à medida de risco associada à saúde do cliente;
5. Os planos de treino e as aulas de grupo devem apresentar a data/hora e um conjunto de imagens ilustrativas;
6. A aplicação deverá ser de fácil uso, com um layout o mais simples possível.
7. A aplicação deverá estar disponível durante os 7 dias da semana, 24 horas por dia.
8. O tempo de resposta da aplicação deve ser o mais curto possível para não influenciar de forma negativa a experiência do utilizador na aplicação.
9. O sistema deve ser suportado por todos os browsers e telemóveis.

MODELO DE DOMÍNIO

Tendo por base os requisitos previamente recolhidos e analisados, foi desenvolvido o modelo de domínio que visa recolher e representar de maneira geral os conceitos, comportamentos e algumas interações que estarão incluídas no sistema a desenvolver.

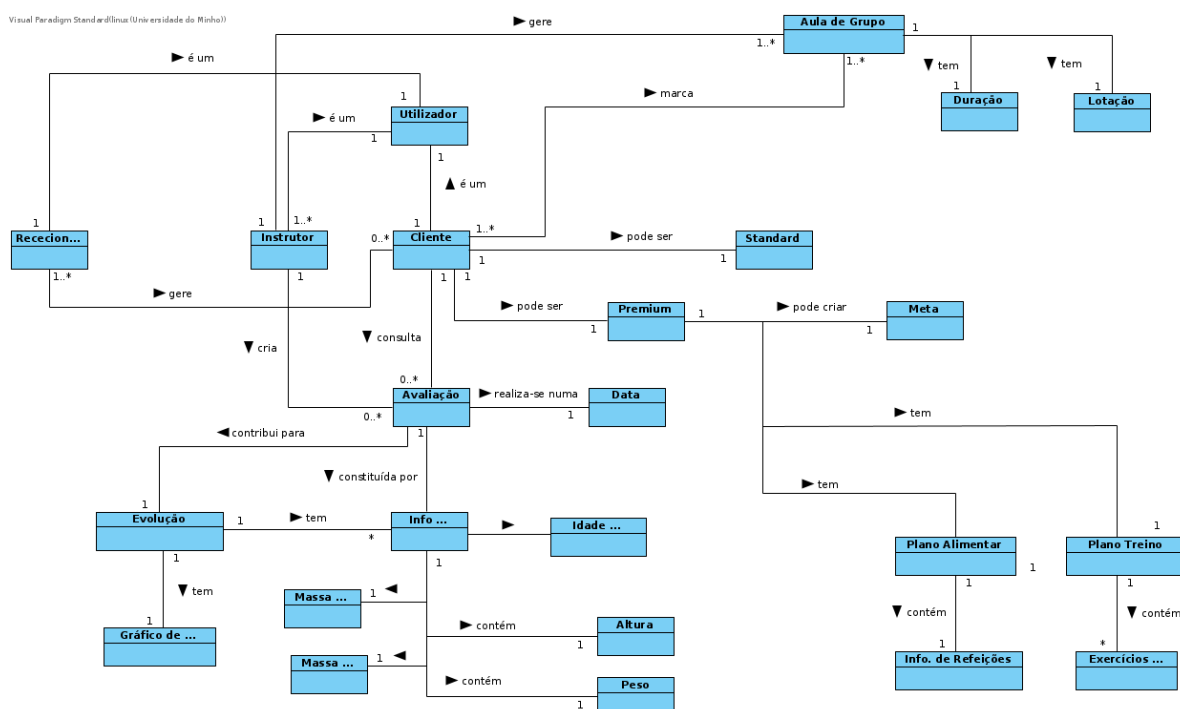


Figura 2: Modelo de Domínio

Numa rápida análise é bastante fácil e intuitivo de perceber que já estão modelados todos os principais objetivos da aplicação nomeadamente a interação Utilizador - Aula com destaque obviamente para as aulas e ainda, também a título de exemplo, alguns dos aspetos que constituem um plano de treino, sejam eles a sua área de treino, nível de dificuldade, as zonas

musculares que a compõe e até as instruções para a sua execução.

DIAGRAMA DE USE CASES

O Diagrama de Use Cases tem o objetivo de auxiliar a comunicação entre o programador e o cliente, ou seja, um diagrama de Use Cases descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário.

Aqui apresentámos todos as possíveis funcionalidades que cada actor conseguirá executar dentro da aplicação. Na imagem abaixo conseguimos perceber que um cliente premium tem todas as funcionalidades de um cliente standard e ainda mais três que são criar meta, consultar plano de treino e consultar plano alimentar.

Vemos ainda a cinza funcionalidades que apresentámos como extra, ou seja, funcionalidades que o grupo não se compromete a realizar mas se surgir uma vaga temporal iremos tentar implementar.

Este diagrama é assim importante para apresentar de uma forma mais acessível ao cliente todas as funcionalidades que o programa poderá vir a implementar. No nosso caso em específico serviu de guia para o nosso projeto para saber que funcionalidades seriam necessárias implementar.

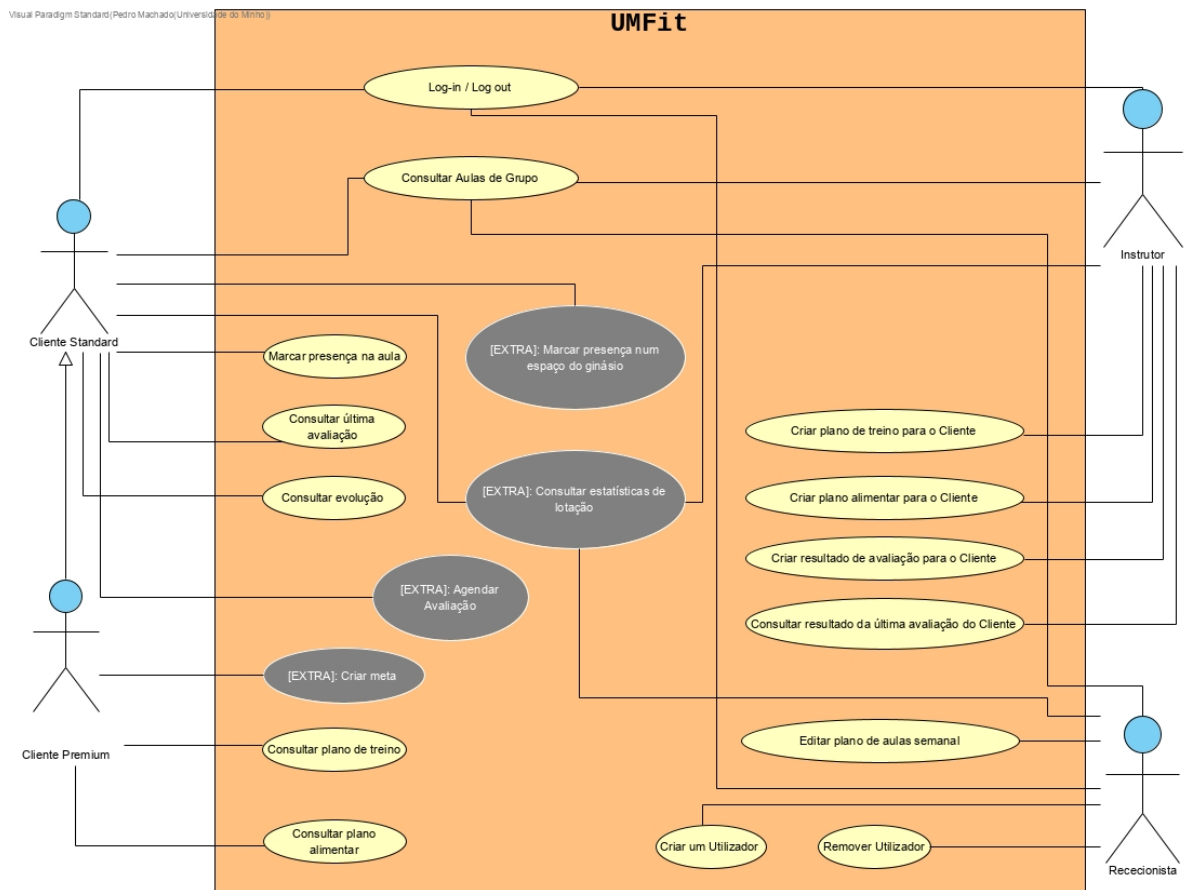


Figura 3: Diagrama de Use Case

ESPECIFICAÇÃO DE USE CASES

6.1 CONSULTAR AULAS DE GRUPO

Descrição: Utilizador verifica a informação referente a uma aula de grupo.

Pré-Condição: O Utilizador tem de estar autenticado no sistema e existirem aulas de grupo no sistema.

Pós-Condição: O Utilizador vê a informação referente à aula de grupo.

Fluxo Normal:

1. O Utilizador seleciona a opção "consultar aulas de grupo".
2. O sistema apresenta a lista com as aulas de grupo e as suas informações respetivas.

6.2 CONSULTAR EVOLUÇÃO

Descrição: Utilizador verifica dados da sua evolução.

Pré-Condição: O utilizador ter pelo menos uma avaliação.

Pós-Condição: O utilizador vê os dados da sua evolução.

Fluxo Normal:

1. O Utilizador seleciona a opção "Consultar Evolução".
2. O Sistema apresenta os dados da sua evolução.

6.3 CONSULTAR PLANO ALIMENTAR

Descrição: Utilizador verifica o seu plano alimentar.

Pré-Condição: O utilizador ter-lhe atribuído um plano alimentar.

Pós-Condição: O utilizador vê os dados do seu plano alimentar.

Fluxo Normal:

1. O Utilizador seleciona a opção "Consultar Plano alimentar".
2. O Sistema apresenta os dados do plano alimentar.

6.4 CONSULTAR PLANO DE TREINO

Descrição: Utilizador verifica o seu plano de treino.

Pré-Condição: O utilizador ter-lhe atribuído um plano de treino.

Pós-Condição: O utilizador vê os dados do seu plano de treino.

Fluxo Normal:

1. O Utilizador seleciona a opção "Consultar Plano de treino".
2. O Sistema apresenta os dados do plano de treino.

6.5 CONSULTAR RESULTADO DA ÚLTIMA AVALIAÇÃO DE UM DADO UTILIZADOR

Descrição: O instrutor consulta a última avaliação.

Pré-Condição: O instrutor está autenticado no sistema.

Pós-Condição: O instrutor obtém os dados da ultima avaliação.

Fluxo Normal:

1. O instrutor seleciona a opção "Consultar a última avaliação".
2. O instrutor insere o email do cliente do qual pretende ver a última avaliação.
3. O sistema valida o cliente escolhido.
4. O sistema apresenta os dados da última avaliação.

Fluxo Exceção 1: [O email inserido não é válido] (passo 3)

- 3.1 O Sistema invalida o email inserido.
- 3.2 O Sistema termina o processo.

Fluxo Exceção 2: [O cliente não tem avaliações] (passo 4)

- 4.1 O Sistema termina o processo.

6.6 CONSULTAR RESULTADO DA ÚLTIMA AVALIAÇÃO

Descrição: O Rececionista verifica dados da sua última avaliação do utilizador escolhido.

Pré-Condição: O Utilizador ter pelo menos uma avaliação e o Rececionista estar autenticado.

Pós-Condição: O Rececionista vê os dados da avaliação.

Fluxo Normal:

1. O Rececionista seleciona a opção "Consultar última Avaliação".
2. O Rececionista insere o e-mail do Utilizador pretendido;
3. O Sistema apresenta a última avaliação do utilizador ao Rececionista.

6.7 CONSULTAR ÚLTIMA AVALIAÇÃO

Descrição: Utilizador verifica dados da sua última avaliação.

Pré-Condição: O utilizador ter pelo menos uma avaliação.

Pós-Condição: O utilizador vê os dados da avaliação.

Fluxo Normal:

1. O Utilizador seleciona a opção "Consultar última Avaliação".
2. O Sistema apresenta a última avaliação do utilizador.

6.8 CRIAR PLANO ALIMENTAR PARA CLIENTE

Descrição: O Instrutor atribui a um utilizador um plano alimentar.

Pré-Condição: O instrutor está autenticado no sistema.

Pós-Condição: O sistema insere um novo plano alimentar no cliente.

Fluxo Normal:

1. O instrutor seleciona a opção "Criar Plano Alimentar".
2. O instrutor insere o email do cliente a quem deseja atribuir o plano alimentar.
3. O sistema valida o cliente escolhido.
4. O instrutor insere os dados do plano alimentar.
5. O sistema adiciona o plano alimentar ao utilizador.

Fluxo Exceção 1: [O email inserido não é válido] (passo 3)

- 3.1 O Sistema invalida o email inserido.
- 3.2 O Sistema termina o processo.

6.9 CRIAR PLANO DE TREINO PARA CLIENTE

Descrição: O Instrutor atribui a um utilizador um plano de treino.

Pré-Condição: O instrutor está autenticado no sistema.

Pós-Condição: O sistema insere um novo plano de treino no cliente.

Fluxo Normal:

1. O instrutor seleciona a opção "Criar Plano de treino".
2. O instrutor insere o email do cliente a quem deseja atribuir o plano de treino.
3. O sistema valida o cliente escolhido.
4. O instrutor insere os dados do plano de treino.
5. O sistema adiciona o plano de treino ao utilizador.

Fluxo Exceção 1: [O email inserido não é válido] (passo 3)

- 3.1 O Sistema invalida o email inserido.
- 3.2 O Sistema termina o processo.

6.10 CRIAR RESULTADO DE AVALIAÇÃO PARA CLIENTE

Descrição: O Instrutor atribui a um utilizador uma avaliação.

Pré-Condição: O instrutor está autenticado no sistema.

Pós-Condição: O sistema insere uma nova avaliação no cliente.

Fluxo Normal:

1. O instrutor seleciona a opção "Criar Avaliação".
2. O instrutor insere o email do cliente a quem deseja atribuir a avaliação.
3. O sistema valida o cliente escolhido.
4. O instrutor insere os dados da avaliação;

5. O sistema adiciona a avaliação ao utilizador.

Fluxo Exceção 1: [O email inserido não é válido] (passo 3)

3.1 O Sistema invalida o email inserido.

3.2 O Sistema termina o processo.

6.11 CRIAR UTILIZADOR

Descrição: Um novo utilizador é criado no sistema pelo rececionista.

Cenários: Um Emanuel quer entrar no ginásio e a Maria rececionista cria a sua conta.

Pré-Condição: O rececionista tem de estar autenticado no sistema.

Pós-Condição: Um novo utilizador é adicionado ao sistema.

Fluxo Normal:

1. O rececionista insere o nome, e-mail e morada do novo utilizador, juntamente com o seu género e outras informações pertinentes ao registo.
2. O Sistema valida os dados e regista um novo utilizador com credenciais aleatórias.

Fluxo Excepção:

- 2.1 Os dados fornecidos ao Administrador são inválidos;
- 2.2 O Administrador não cria a nova conta;

6.12 EDITAR PLANOS DE AULAS SEMANAL

Descrição: A rececionista altera as aulas de grupo.

Pré-Condição: O rececionista está autenticado no sistema.

Pós-Condição: No sistema são alteradas as informações das aulas de grupo.

Fluxo Normal:

1. O rececionista seleciona a opção "Editar plano de aulas".
2. O sistema apresenta as opções de "edição de aulas" ou "adição de aulas".
3. O rececionista seleciona a opção edição de aulas.
4. O sistema apresenta uma lista das aulas de grupo existentes.
5. O rececionista escolhe uma aula de grupo para ser editada.
6. O sistema apresenta os dados referentes a essa aula.
7. O rececionista altera os dados da aula.
8. O sistema altera a informação da aula de grupo com os novos dados.

Fluxo Alternativo 1: [O rececionista seleciona adição de aulas] (passo 3)

- 3.1 O sistema pergunta ao rececionista as informações sobre a nova aula.
- 3.2 O rececionista insere as informações referentes a essa aula.
- 3.3 A nova aula de grupo é adicionada ao sistema.

Fluxo Exceção 1: [Não existem aulas disponíveis] (passo 4)

- 4.1 O sistema informa que não existem aulas de grupo no sistema.
- 4.2 O sistema termina o processo.

6.13 LOG IN

Descrição: Autenticar um cliente, instrutor ou rececionista.

Pré-Condição: O Utilizador tem de estar registado no Sistema e não pode haver outro Utilizador no Sistema.

Pós-Condição: O Utilizador fica autenticado no Sistema como cliente, instrutor ou rececionista.

Fluxo Normal:

1. O Sistema apresenta as opções de Autenticação.
2. O Utilizador escolhe opções de entrar como cliente, instrutor ou rececionista e insere as credenciais.
3. O Sistema valida as credenciais inseridas.
4. O Utilizador autentica-se no Sistema.

Fluxo de excepção 1: [As credenciais são inválidas] (passo 3)

- 3.1 O Sistema avisa o Utilizador que as credenciais são inválidas.
- 3.2 O Utilizador sai do Sistema.

6.14 LOG OUT

Descrição: O utilizador faz log out da aplicação.

Pré-Condição: O Utilizador tem de estar autenticado no Sistema.

Pós-Condição: O utilizador saiu do sistema.

Fluxo Normal:

1. O utente selecciona a opção "Log out".
2. O sistema faz log-out do utilizador.

6.15 MARCAR PRESENÇA NA AULA

Descrição: Utilizador escolhe uma aula na qual pretende marcar presença.

Pré-Condição: Utilizador estar autenticado no sistema e existirem vagas para a aula seleccionada.

Pós-Condição: O utilizador fica inscrito na aula escolhida.

Fluxo Normal:

1. O Utilizador selecciona a opção "Consultar Aulas";
2. O Sistema apresenta a lista aulas.
3. O Utilizador selecciona a aula.
4. O Sistema inscreve o utilizador na aula.

6.16 REMOVER UTILIZADOR

Descrição: Um Utilizador é removido do sistema pelo Rececionista.

Cenários: Para facilitar a gestão de Utilizador no Sistema, o Rececionista tem a possibilidade de remover Utilizadores;

Pré-Condição: O Rececionista tem de estar autenticado no Sistema.

Pós-Condição: Um Utilizador é removido do Sistema.

Fluxo Normal:

1. O Rececionista escolhe a opção de "Remover Utilizador".
2. O Rececionista escolhe o e-mail do Utilizador que pretende remover.
3. O Rececionista escolhe a opção de "Remover do Sistema".
4. O Rececionista seleciona o Utilizador que quer remover.
5. O Sistema remove o Utilizador e todo o seu conteúdo.

ARQUITETURA DA CAMADA DE NEGÓCIOS

7.1 CAMADA DE NEGÓCIOS

Como consequência lógica da análise conjunta dos requisitos e do modelo de domínio apresentado, identificamos e caracterizamos as classes e desenvolvemos uma pequena descrição da arquitetura da solução a desenvolver. Esta descrição foi essencial na construção do diagrama de classes representativo da arquitetura da camada de negócios.

7.2 DICIONÁRIO DAS PRINCIPAIS CLASSES

Utilizador (Utilizador)- Corresponde à representação no sistema dos utilizadores da aplicação e contém a informação pessoal, informação de acesso e nível de dificuldade.

Avaliação (Avaliação) - É uma componente atribuída a um cliente por um instrutor, que contém todos os dados importantes para o cliente perceber a sua evolução e forma física atual. Esta classe guarda a informação referente à data, massa gorda, massa marga, imc, id, peso, idade metabólica, altura, se foi realizada ou nao(foi_realizada), nif do cliente, e nif do instrutor.

Aula Grupo (Aula_Grupo) - É a classe que representa a aula de grupo a ser escolhida pelo cliente. Esta classe contém um id, uma dificuldade, uma data, uma lotação actual, um nome, uma lotação máxima, e o nif do instrutor.

Plano Treino (Plano_Treino) - Classe que armazena um conjunto de exercícios para serem atribuídos a um cliente final. Quem atribui e constrói este plano de treino é o instrutor, sendo que o plano de treino contém um conjunto de exercícios, uma data de inicio, uma data de fim, e um id.

Plano Alimentar (Plano_Alimentar) - Classe que representa um plano idêntico ao mencionado anteriormente, mas este contém um conjunto de refeições. Tal como o anterior este é atribuído e construído também pelo instrutor contendo assim um conjunto de refeições, um nome, um id, uma data de inicio, uma data de fim, uma frequência e um numero de refeições livres.

Espaço Ginásio (Espaço_Ginasio) - Classe que representa uma divisão do ginásio, ou seja, um ginásio pode se encontrar organizado por zonas como cardio, musculação, aulas em grupo... Por isso a importância desta classe em armazenar o nome da zona e a lotação associada à mesma.

7.3 DESCRIÇÃO DA ARQUITETURA

Cada utilizador pode ou não ter associado a si planos de treino, planos alimentares e avaliações. A existência destas associações vai mudar a perspectiva do cliente quanto à aplicação, visto que se o mesmo não tiver nenhum plano de treino associado ao tentar visualizar os seus planos de treino não vai encontrar nada.

A existência das aulas é algo fulcral na nossa implementação e que cria uma maior dinâmica entre cliente, instrutor e recepcionista. Quer seja através do cliente se quiser inserir numa dada aula, na visualização da lotação da mesma ou da alteração dessas mesmas aulas por parte da recepcionista.

Todas estas classes instrutor, cliente, recepcionista vão ter de ter como atributo um identificador único, para se poderem atribuir e relacionar com outras classes.

A existência de classes como por exemplo as avaliações irá nos permitir no futuro apresentar ao cliente gráficos e estatísticas referentes à sua evolução e desempenho desde que chegou ao ginásio

7.4 DIAGRAMA DE CLASSES

O modelo de classes derivado através da descrição encontra-se definido mais a abaixo. Adicionalmente, às entidades mencionadas foi também criado um facade, a criação deste teve como objectivo mediar a interacção com toda a camada de negócios. Isto para que a solução apresentada fosse modular e composicional com um sistema congruente com uma arquitectura de três camadas. Apresentamos portanto em seguida todas as classes que achamos essenciais à aplicação já com alguns atributos que achamos relevantes mas não definitivos.

Pode-se observar também neste diagrama a presença de DAO's (Data Access Object), ou seja interfaces que estabelecem o contacto entre as classes já definidas e a base de dados.

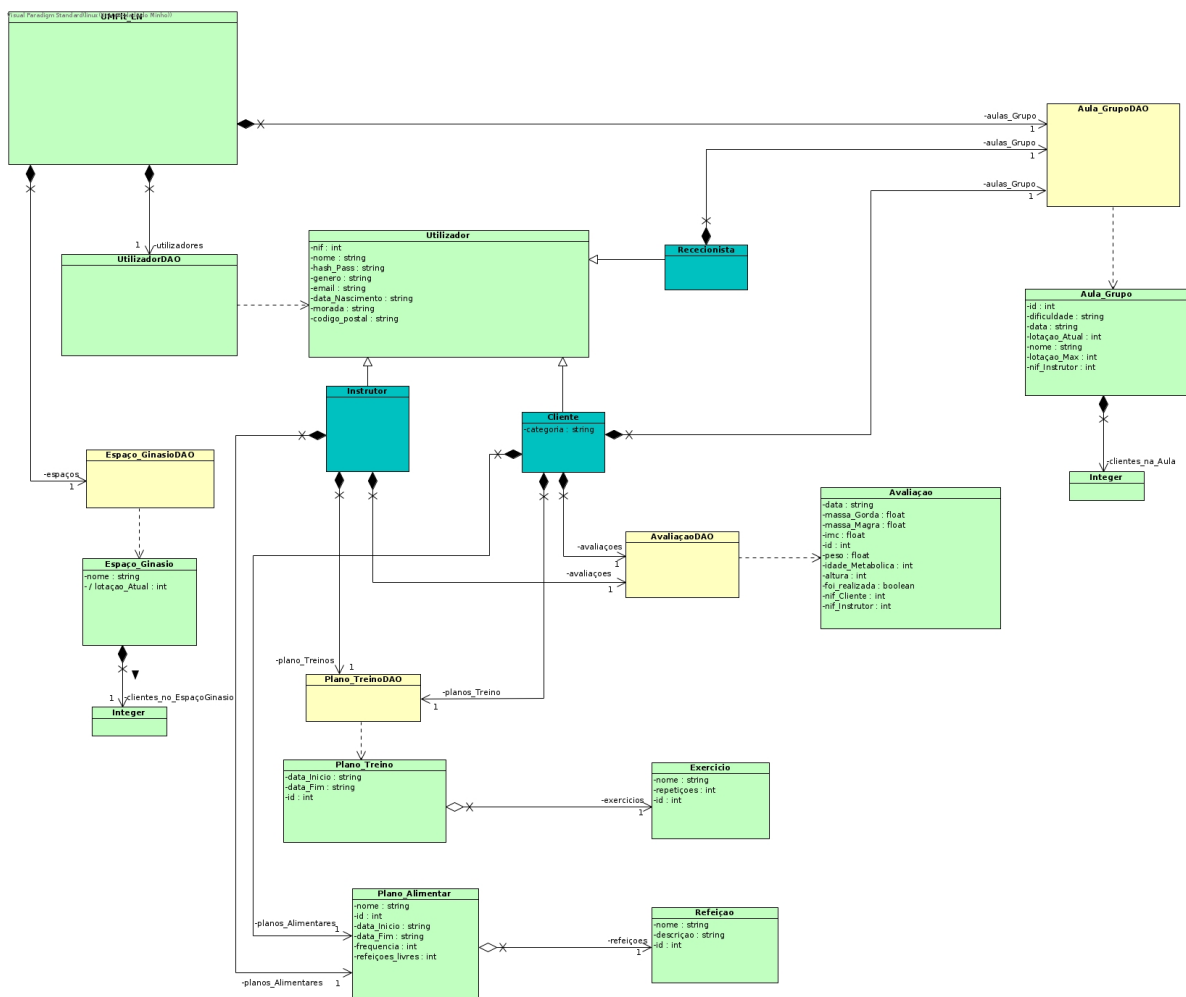


Figura 4: Diagrama de Classes

CAMADA DE DADOS

Esta camada, como na maioria das aplicações, surge com o intuito de aplicar metodologias que permitem garantir persistência, integridade, segurança e acessibilidade de dados utilizando ferramentas de gestão de bases de dados.

Nesta aplicação, o modelo de gestão de base de dados adotado segue uma ótica relacional, isto é, para cada entidade, em que se pretende persistir dados, estabelecem-se relações com outras entidades e com isso existem uma série de regras que tivemos de adotar para administrar corretamente um sistema como o presente na nossa aplicação.

8.1 MODELO CONCEPTUAL

O modelo conceptual da aplicação demonstra, de maneira simples, a estrutura que a base de dados relacional irá ter.

Desta forma, é preciso destacar algumas entidades fulcrais para o funcionamento e estruturação dos dados:

- **Utilizador**

Visto que a aplicação possui três tipos de utilizadores, foram desenvolvidas três entidades diferentes¹ para a separação de utilizadores.

¹Cliente, Instrutor e Rececionista. Este último comporta-se como o administrador da Base de Dados.

Esta separação foi efetuada para possibilitar uma consulta mais controlada e eficiente a nível das funcionalidades pretendidas, ou seja, qualquer funcionalidade específica para um rececionista apenas consulta os dados da tabela respetiva.

- **Aula de Grupo**

Através da entidade **Aula_Grupo** é possível identificar todas os atributos que caracterizam uma aula.

- **Plano de Treino**

A entidade **Plano_Treino** caracteriza todos os valores relacionados com um plano de treino, pelo que este pode estar associado a um cliente contendo os seus exercícios.

- **Plano Alimentar**

Tal como o Plano de Treino, o Plano Alimentar através da sua entidade, **Plano_Alimentar**, caracteriza todas as refeições de um plano que pode estar associado a um cliente.

- **Avaliação**

Uma avaliação de um cliente está dividida em duas entidades separadas na Base de Dados, de forma a podermos distinguir uma avaliação realizada de outra apenas agendada, através das tabelas **Avaliacao_Agendada** e **Avaliacao_Realizada**.

- **Espaço do Ginásio**

Esta entidade foi desenvolvida no início do projeto com a intenção de servir como identificador do local onde o cliente se encontrava dentro do ginásio.

Contudo, acabou por não ter efeito na API da aplicação por questões de opção da colocação dos recursos do grupo. Mesmo assim, há que realçar que a base de dados estaria pronta para a implementação dessa funcionalidade.

- **Utilizadores Online**

Por último, esta entidade tem como função armazenar quais os utilizadores se encontram *online* tendo em conta o seu respetivo *token* e data de expiração.

Assim, construímos o modelo conceptual, como podemos observar na secção seguinte.

8.2 DIAGRAMA DO MODELO CONCEPTUAL

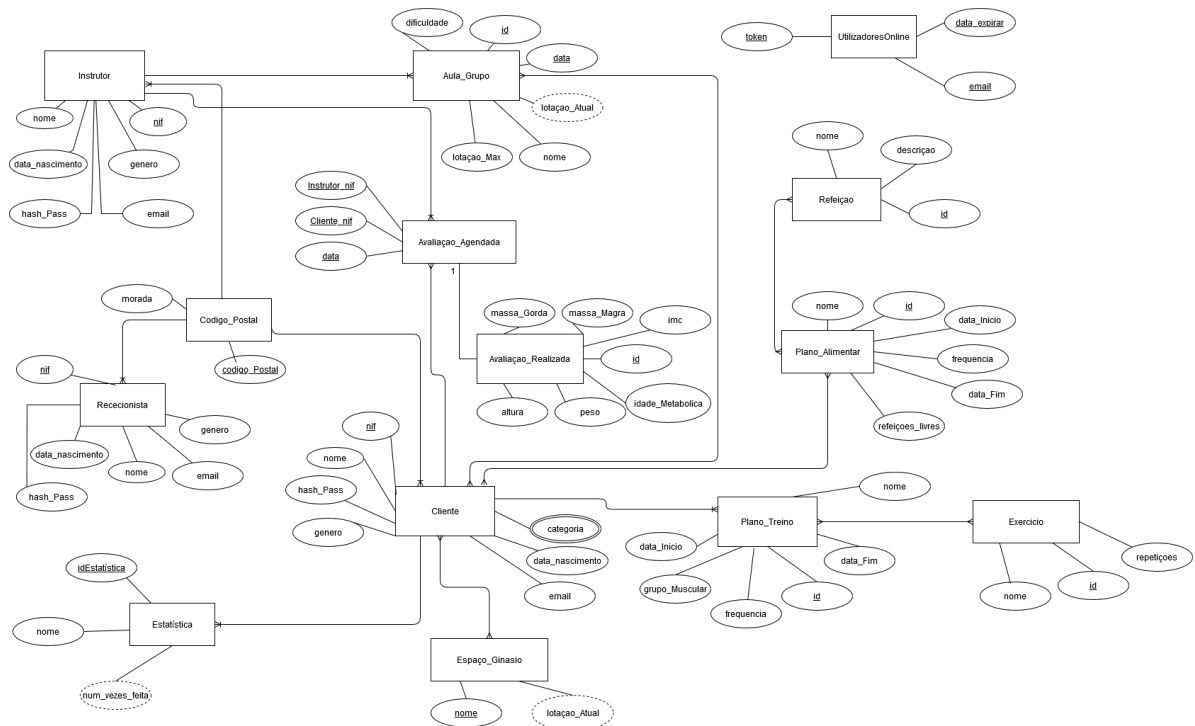


Figura 5: Diagrama do Modelo Conceptual

8.3 MODELO LÓGICO

O modelo lógico apresentado, resulta essencialmente, da estruturação da informação importante a guardar da modelação da camada de negócios apresentada.

Assim, este modelo foi obtido a partir do conceptual, usando para isso a ferramenta *MySQL Workbench*, como podemos ver na secção seguinte.

8.4 DIAGRAMA DO MODELO LÓGICO

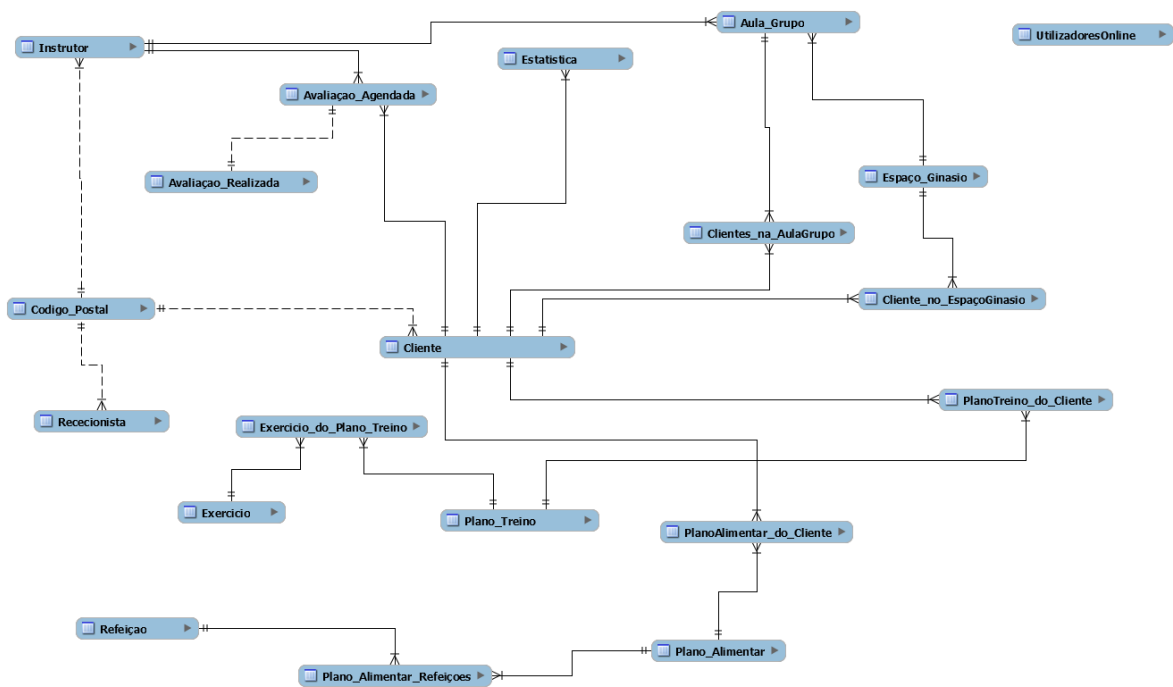


Figura 6: Diagrama do Modelo Lógico

PROPOSTAS DE INTERFACE

Uma das camadas mais importantes de um sistema de software, e que muitas das vezes dita o sucesso ou insucesso do sistema é a sua interface de comunicação com o utilizador. Por vezes a aplicação por muito boa que seja e por mais funcionalidades que apresente se não for apresentável e user "friendly", o utilizador pode acabar por não usufruir ao máximo de todas essas funcionalidades e pode mesmo desistir da utilização do software. Sendo assim a "UMFit" focou-se no público alvo do seu software e desenhou uma aplicação que vá de encontro às suas necessidades.

Para poder alcançar o sucesso esperado é necessário garantir que esta aplicação se vai apresentar com uma interface simples, elegante, fácil, intuitiva e responsiva. Um dos objectivos é garantir que a "UMFit", com uma curva de aprendizagem inicial baixa, garanta logo nas primeiras utilizações do utilizador uma experiência interactiva de qualidade. Permitindo assim que quem utiliza o software tenha vontade de o voltar a usar e vontade de o recomendar a outros.

Deste modo, desde uma fase inicial do projecto mantivemos uma grande preocupação no planeamento e desenho da interface gráfica para ela ir acompanhando sequencialmente a evolução da especificação do sistema em geral e para ser possível um grande número de etapas de validação com o utilizador.

Um dos objectivos passou por manter uma estrutura e uma dinâmica ao longo do desenho das interfaces que permitissem ao utilizador seguir a mesma linha de procura de informação para diferentes menus e funcionalidades. Ou seja seria necessário definir uma "paleta" de cores a serem utilizadas para que a mudança de funcionalidades não fosse demasiado drástica visualmente. Seria necessário seguir uma estrutura de inserção de dados e de botões que per-


mitisse ao utilizador usar consoante a experiência passada na nossa aplicação, e que não fosse necessário aprender novas "skills" sempre que mudava de funcionalidade.

Apesar de serem interfaces bastantes acessíveis ao utilizador comum tentamos ainda garantir, que sempre que fosse inserida uma funcionalidade um pouco mais "rebuscada" fossem dadas as ferramentas e as indicações necessárias para o utilizador não se perder nas mesmas.

The image shows a web application interface for creating a new user. On the left is a sidebar for 'UMFit Ginásio da Universidade do Minho' with a menu including 'O meu perfil', 'Plano de Aulas', 'Atualizar perfil', 'Criar Utilizador' (selected), 'Editar Utilizador', 'Remover Utilizador', and 'Log-Out'. The main area is titled 'Novo utilizador' and contains a form with fields for 'Nome Completo*', 'E-Mail *', 'Definir a password *', 'Localidade', and 'Nif'. Below these are dropdowns for 'Tipo de sócio:', 'Género:', and a date field for 'Data de Nasc.: Feb 20, 1999'. At the bottom are buttons for 'CRIAR NOVO UTILIZADOR' and 'LIMPAR FORMULÁRIO'. The footer of the sidebar indicates '© 2020 UMFit'.

Figura 7: Inserção de novo utilizador

Atualizar perfil




X


Nome Completo *

X

E-Mail *



Definir nova password




Nova Localidade


X

Nif


X

 Tipo de sócio:

X

 Género:

X

 Data de Nascimento:

ATUALIZAR INFORMAÇÕES

LIMPAR FORMULÁRIO

Figura 8: Atualização de um perfil

Como pode ser observado nas duas interfaces apresentadas anteriormente, as mesmas tentam manter um padrão similar de cores, neste caso predominância do laranja, branco e verde. E tentam ainda manter a mesma estrutura e sequência dos dados apresentados quer seja na inserção do utilizador quer seja na actualização do mesmo.

Este padrão é seguido à risca para a criação de todas as interfaces do projecto direccionado assim o foco da atenção do utilizador e evitando que este se disperse e perca ao longo da funcionalidade sem perceber ao certo o objectivo da mesma.

METODOLOGIA DE IMPLEMENTAÇÃO

Esta aplicação exige uma divisão coordenada em várias camadas que interatuam entre si de tal forma que podem ser desenvolvidas de forma independente mas prestando serviços umas às outras através de práticas usuais em sistemas comuns de desenvolvimento de *software*.

Deste modo, como modelo de engenharia de *software*, para um sistema desta natureza, implementou-se uma arquitetura com um desenvolvimento gradual (iterativo) e incremental onde, inicialmente, estabelecemos um conjunto de tarefas de planeamento inicial, incluindo casos de uso base que adviram duma análise extensiva de requisitos e posteriormente fomos, em ciclos bem definidos, desenvolvendo protótipos de interfaces, suporte funcional da nossa API remota e adaptação da base de dados a novos casos de uso, testando, desta forma, o que havia sido desenvolvido para que em ciclos subsequentes, tudo o que já foi desenvolvido também já estaria devidamente testado.

Na prática, no que toca às componentes da API e algum do desenvolvimento do *front-end*, adotou-se a tradicional organização MVC (*Model-View-Controller*) que garante independência de camadas dentro de uma aplicação, separando aquilo que é o desenvolvimento de classes (objetos), do código da *business-layer* e posteriormente daquilo que é apresentado ao utilizador final.

Muitas destas práticas foram facilitadas pelas ferramentas utilizadas na aplicação, que serão descritas, com detalhe, no próximo capítulo, mas surge obviamente como necessidade de qualquer sistema real para aplicações que exigem interação responsiva e interativa.

FERRAMENTAS UTILIZADAS NA IMPLEMENTAÇÃO

Neste capítulo vamos dar relevo às ferramentas que foram utilizadas na implementação da aplicação *UMFit*, desde o desenvolvimento de toda a base de dados, API de suporte à aplicação (*back-end*) e as interfaces *web* (*front-end*).

11.1 DESENHO E GESTÃO DA BASE DE DADOS

Na gestão de uma base de dados relacional é necessário que exista uma boa análise prévia dos dados que serão necessários persistir para dar suporte a todas as entidades do nosso sistema, portanto uma boa análise e desenho conceptual do sistema é necessário para que sejam evitadas grandes alterações num sistema relacional.

Para isso, o desenho do modelo conceptual (entidades) foi feito utilizando ferramentas como o *draw.io* que disponibilizaram uma série de funcionalidades associadas ao *UML*, usualmente ligado a engenharia de *software*.

Já o modelo lógico e físico foram traduzidos a partir do modelo conceptual com a ajuda do *MySQL Workbench*, ferramenta essa disponibilizada pelo sistema de gestão de bases de dados *MySQL*.

11.2 *Back-end*

No caso do *back-end* incluímos todo o desenvolvimento da API que serviria tanto de controlador entre a interface e os dados requisitados pelos utilizadores, como de interface pública de acesso a toda a lógica de negócio da aplicação.

A ferramenta utilizada nesta camada é uma junção de várias fornecidas pela *Microsoft* com uma base no *.NET core*, um sistema multiplataforma de desenvolvimento de aplicações, documentado de forma muito completa e com muito suporte *online*.

De forma geral, este sistema faz a gestão de todos os *end-points* do sistema, tratando de todas as burocracias de gestão de *input* e *output* de *sockets*, deixando o desenvolvimento mais *high-level* para os programadores poderem concentrar o desenvolvimento nas necessidades reais da aplicação.

Por outro lado, todo o desenvolvimento foi feito recorrendo a uma interface de desenvolvimento que dá suporte a esta tecnologia, o *IDE JetBrains Rider*, muito semelhante aos *IDEs* usuais disponibilizados pela *Microsoft*, mas surge numa necessidade de ter um programa adaptado a um sistema *Linux*, preferência dos elementos deste projeto.

11.3 *Front-end*

Chegando agora àquilo que é primeiramente mostrado ao utilizador final, ou seja, a página *web* que lhe permite a interação com o sistema, temos algumas ferramentas que foram importantes no *design*, adaptabilidade e construção da aplicação como algo progressivo e multi-plataforma.

No que toca a todo o desenho da estrutura das interfaces foi utilizada a *framework Ionic* para desenvolvimento de aplicações híbridas, isto é, sistemas escritos com base em tecnologias *web* mas que correm em sistemas nativos como outras aplicações com base em pacotes *Android* (*apk*).

Esta decisão permite-nos introduzir um conceito muito interessante, introduzido em 2015 por *Frances Berriman* e *Alex Russell* para aplicações progressivas, ou seja, PWAs (*pro-*

gressive web apps), o que facilitaria a criação de um sistema multi-plataforma necessário para a *UMFit*, visto que os utilizadores podem ser Rececionistas, Clientes ou Instrutores, certamente usarão dispositivos diferentes e com sistemas operativos móveis também diferentes, garantindo assim essa adaptabilidade e poupando o desenvolvimento independente de aplicações *Android* e *Desktop*.

Por outro lado, para a criação dos diferentes componentes que são mostrados na página *web* utilizamos uma junção de *Typescript* com *React*, aproveitando também alguns componentes visualmente agradáveis fornecidos pela *framework Ionic*. O *Typescript*, assim como o *React* e outras tecnologias utilizadas, eram algo que os elementos do trabalho nunca tinham utilizado revelando-se muito importante o seu processo de aprendizagem visto ser uma linguagem muito utilizada nos dias de hoje.

DESENVOLVIMENTO DO PROJETO

12.1 MECANISMO DE SESSÃO

Neste projeto, assim como em muitos sistemas que necessitam da gestão de autenticidade de um utilizador, é necessário estabelecer um bom mecanismo de sessão produzindo referências ou marcas de autenticação de um utilizador que, nos dias de hoje, se designa pela criação de *Tokens* com uma determinada validade.

A sessão do utilizador com introdução de *tokens* permite a produção de um identificador único de sessão associado a um determinado utilizador com vista a servir de ponte quando informação relativa ao mesmo é requisitada, enviando para a API esse mesmo *token* ficando a mesma com a função de verificar, estender a validade e fornecer de volta os dados requisitados ao cliente.

Para adotar este mecanismo de sessão poderiam ser utilizadas tecnologias como os JWT (*JSON Web Tokens*) que não necessitam que informação da sessão do utilizador fique guardada numa base de dados fazendo automaticamente a validação dos *tokens* e a sua renovação.

No entanto, devido à pouca familiaridade dos elementos com a maioria das ferramentas e visto que a sessão é algo que deve ser implementado, desde cedo, por ser aquilo que dá suporte a estabelecer a privacidade de um perfil do utilizador ou estabelecimento de rotas privadas na aplicação, o grupo decidiu implementar um sistema parecido ao *JWT*.

Neste sistema consideramos que a sessão de um utilizador deve ser persistida na base de dados, guardando aquando o *log-in* de um utilizador, as seguintes informações:

Tabela UtilizadoresOnline:

email VARCHAR; data_expirar DATETIME; token VARCHAR

O *email* corresponde ao do Utilizador, a *data_expirar* é a data a partir da data de sessão mais 5 dias, que estabelece a validade da sessão, sendo esta descartada se o Utilizador estiver *offline* por mais de esse tempo. Por fim o *token* é obtido através da geração de uma assinatura a partir de uma função criptográfica de *hashing*, mais propriamente *SHA-256*, entre a data atual de sessão e o *email* do Utilizador.

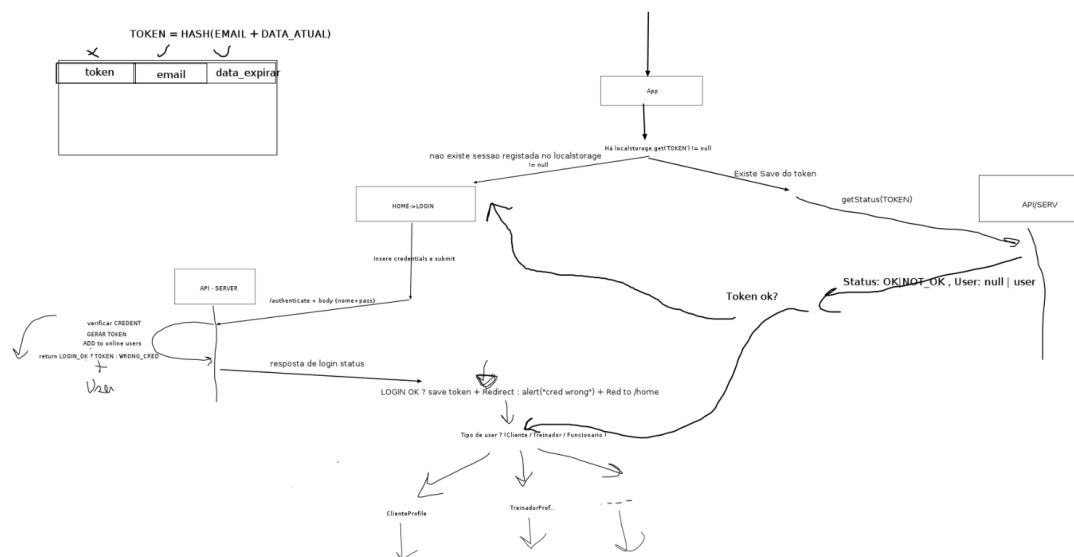


Figura 9: Fluxo de Sessão

Achamos por bem incluir um diagrama, como o da figura anterior, para descrever o mecanismo de sessão que estabelecemos neste aplicação e que havia sido feito numa fase inicial do desenvolvimento deste *software*.

12.2 API

API quer dizer *Application Programming Interface*, em outras palavras, é uma interface de comunicação entre desenvolvedores. Ou seja um utilizador nunca comunica directamente com a base dados, utiliza sempre a API como intermediário dessa mesma comunicação.

Quando falamos de *APIs*, toda a comunicação dessa interface é feita via web. Ou seja, tudo é feito através da requisição de um URL, que por sua vez, traz uma resposta.

Para tal usamos uma requisição *endpoint*, em que o URL nada mais é que o caminho para fazer a requisição e onde os argumentos do pedido, se existirem, são colocados no *endpoint* (corpo) da requisição. Uma API *rest* tem vários tipos de métodos que os pedidos podem ter, tais como *get*, *post*, *put* e *delete*. Como sempre que fazíamos algum pedido era necessário enviar um *token* de sessão para verificar se este era válido ou não, a maioria dos métodos utilizados pelo grupo foram *posts*. Em seguida temos um exemplo de um pedido:



Figura 10: Pedido de Autenticação

Como podemos observar na imagem anterior, o pedido de autenticação é feito ao controlador *User* e os argumentos deste pedido encontram-se no corpo do pedido que são o email e a password para serem posteriormente autenticados.

Para termos uma API mais organizada e estruturada foram criados vários *controllers* para as entidades mais importantes da nossa aplicação, ou seja, entidades como *User*, *Aulas* de

Grupo, Avaliação entre outras têm um *controller* próprio. O que o *controller* faz basicamente é receber o pedido, fazer *parse* dos argumentos que se encontram no corpo do pedido, se estes existirem, e comunica com a lógica negocial do nosso projeto, "UMFit_LN"

Posto isto a lógica negocial comunica com o DAO executando a função já com os argumentos correctos. Recebe uma resposta da base de dados e passa essa informação à função do *controller* que continha o pedido inicial.

Com esta informação o *controller* pode enviar assim uma resposta ao pedido feito inicialmente. Para facilitar o entendimento das respostas das APIs usamos padrões de códigos de estado. Os códigos por nós utilizados foram: 200 (OK), o 404 (not found), o 400 (bad request), e 500 (internal server error). No corpo desta resposta podem ainda ser enviados dados que contém a resposta referente ao pedido inicial. A resposta ao pedido exemplo dado anteriormente é a seguinte:

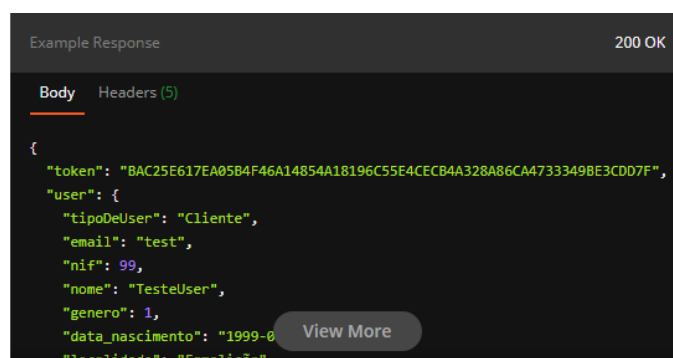


Figura 11: Resposta ao pedido de Autenticação

Constatamos assim que neste caso a resposta a ao pedido de autenticação retornou o estado 200 (OK), e portanto não ocorreram erros. Vemos assim que o corpo da resposta contém um *token* com a sessão do utilizador e classe *User* com toda a informação referente ao mesmo.

De referir ainda que surgiram ainda alguns contratemplos, como sempre que existia um novo pedido de acesso à base de dados era criada uma nova conexão à mesma, o grupo enfrentou alguns problemas no que toca à concorrência no acesso à base de dados. Já que o *mysql* não permite a existência de várias conexões simultâneas o grupo solucionou este pro-

blema utilizando *lock's* nos pedidos feitos pela API a determinadas variáveis. Feito isto foram solucionados todos os problemas de concorrência encontrados até então.

De salientar que em anexo se encontra toda a nossa API devidamente documentada para o caso de existirem de dúvidas adicionais. Esta documentação pode ainda ser consultada em [documentação API postman](#).

12.3 CONEXÃO DA BASE DE DADOS

Inicialmente de forma a criar a bases de dados pretendida para a “UMFit” foi utilizado o *MySql* com uso do *localhost*.

Assim, quase todo o processo de conexão à base de dados foi gerido pela API do *Asp.Net Core* que, se devidamente autenticada, conseguia aceder à base de dados.

De forma a garantir uma maior integridade nas conexões, sempre que um cliente se pretende conectar à base de dados, é feito um pedido à API, que através dos **DAOs**² existentes consegue aceder à informação requisitada.

Primeiramente, foi introduzido um povoamento inicial na base de dados de forma a simular um prévio conhecimento de dados de todas as tabelas desenvolvidas.

De seguida, numa fase mais tardia do desenvolvimento do projeto, foi necessário a introdução da base de dados *localhost* para um servidor de forma a possibilitar o acesso a esta por utilizadores através de vários aparelhos. Assim, foi feito um *deploy* da base de dados para o **Microsoft Azure** e a criação de uma conta para a API ter acesso.

²Data Access Object

12.4 INTERFACES

Para o desenvolvimento de todas as interfaces apresentadas ao longo da aplicação a framework ionic foi essencial. Aliado a esta ferramenta foi gerado código em typescript react capaz de manter uma estrutura e facilitar a portabilidade do código para as diferentes interfaces.

De salientar ainda que como o intuito do grupo passou desde cedo pela criação de uma pwa (progressive web app), ou seja uma aplicação multi plataforma capaz de suportar diferentes dispositivos. Foi necessário por isso implementar diferentes formas de mostrar a mesma interface consoante o sistema em que a interface estivesse a ser apresentada. Deste modo foi possível criar uma aplicação que dependendo do dispositivo utilizado altera a funcionalidade de modo a melhor se adaptar ao utilizador e ao seu sistema.

PRODUTO FINAL

Ao longo de várias semanas desenvolvemos o UMFit até conseguir obter o produto de software previamente idealizado. Em seguida, iremos demonstrar o workflow da aplicação, apresentando assim como é se pode encontrar cada uma das funcionalidades implementadas através de screenshots da interface final do produto e de pequenas explicações.

13.1 PÁGINA INICIAL (PRÉ AUTENTICAÇÃO)

Quando um utilizador autenticado acede ao website do UMFit, este é confrontado com a página inicial exposta em seguida, que permite aceder aos contactos e a alguma informação relativa à equipa de desenvolvimento.

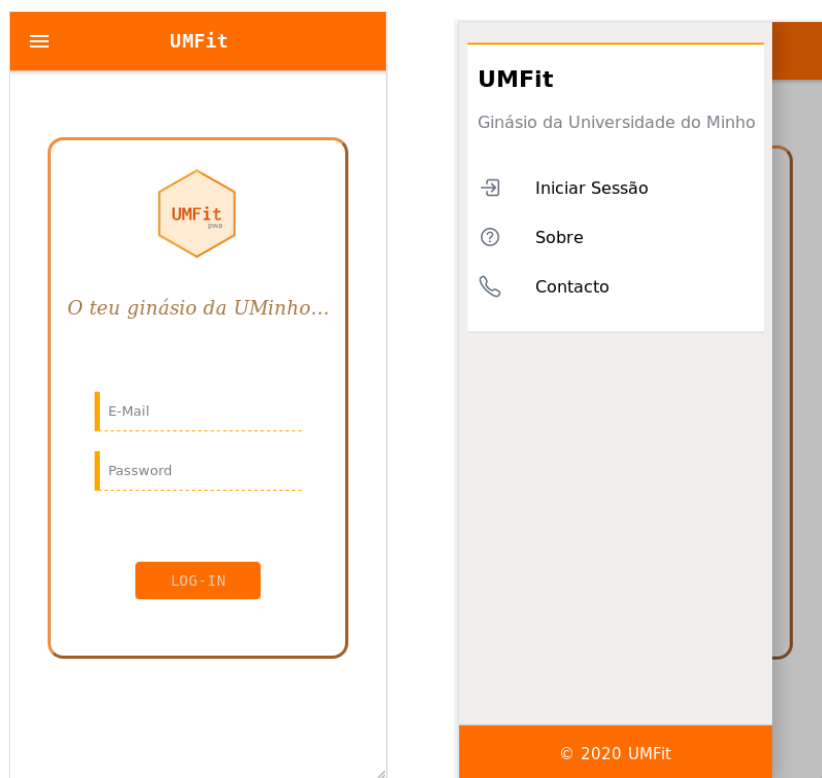


Figura 12: Página Inicial Mobile e Dashboard

Assim, através desta página é possível autenticar-se como cliente, instrutor ou rececionista. Além disso é também possível saber um pouco mais sobre a equipa ou contactar a mesma. Nas páginas autenticadas existe sempre

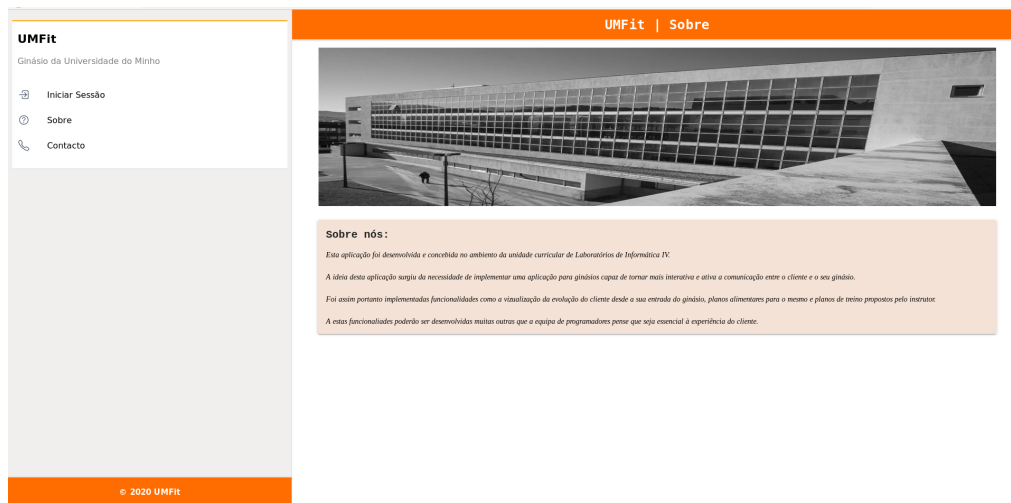


Figura 13: Página Sobre

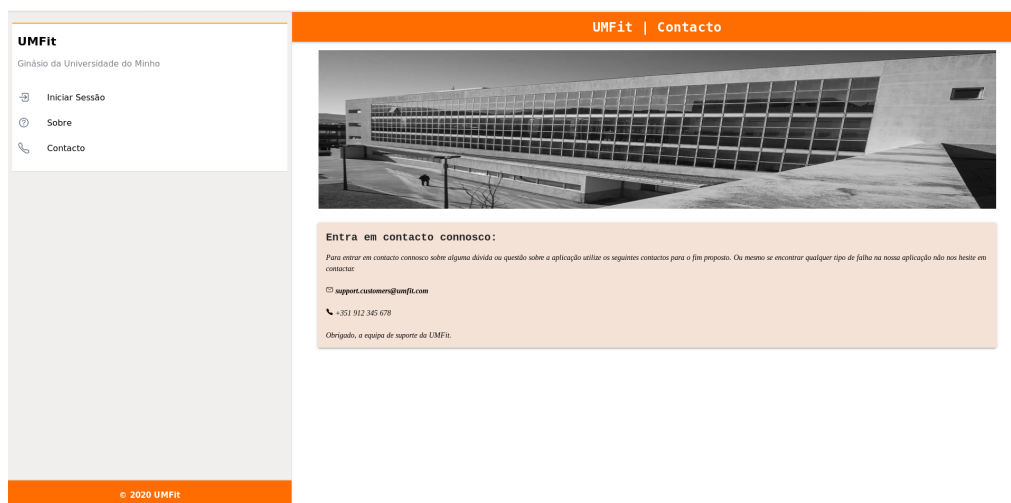


Figura 14: Página Contacto

13.2 PÁGINA CLIENTE STANDARD

Depois de autenticado, é apresentado ao cliente na sua dashboard, e páginas novas que podem ser acedidas pelo mesmo, esta versão do cliente permite aceder as funcionalidades básicas que qualquer cliente poderá aceder ao fazer login na aplicação.

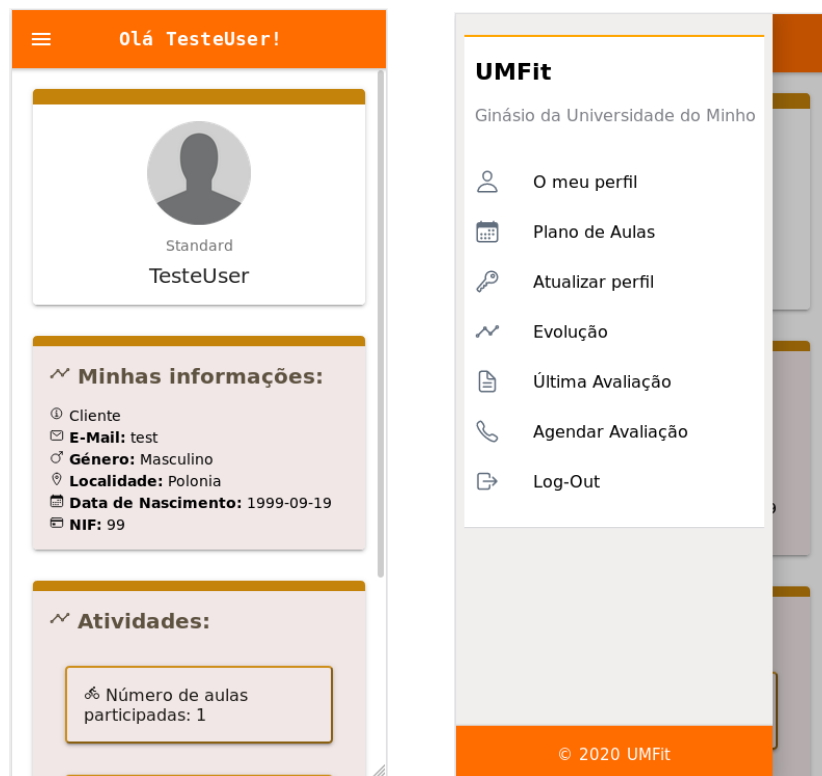


Figura 15: Perfil de cliente standard em Mobile e Dashboard

Agendar

Agendar Avaliação

SUBMITER

Data:

07-04-2020 18:48

Instrutor:

Nome do instrutor...

João Azevedo

Paulo Lima

Ana Silva

Data: 2020-08-02 15:31:18

Instrutor: Paulo Lima

Última avaliação

Detalhes:

Data realização: s/ seleção

Nome Cliente: s/ seleção

Mail Cliente: s/ seleção

Composição Corporal:

Altura: 0 cm

Peso: 0 kg

Idade Metabólica: 0 anos

IMC: 0

Massa Gorda (kg)

Figura 16: Informação de avaliação e agendamento

Tal como representado na figura acima (**Fig16**) o cliente tem a possibilidade de agendar avaliações, e de visualizar os dados da última avaliação feita pelo mesmo.

57

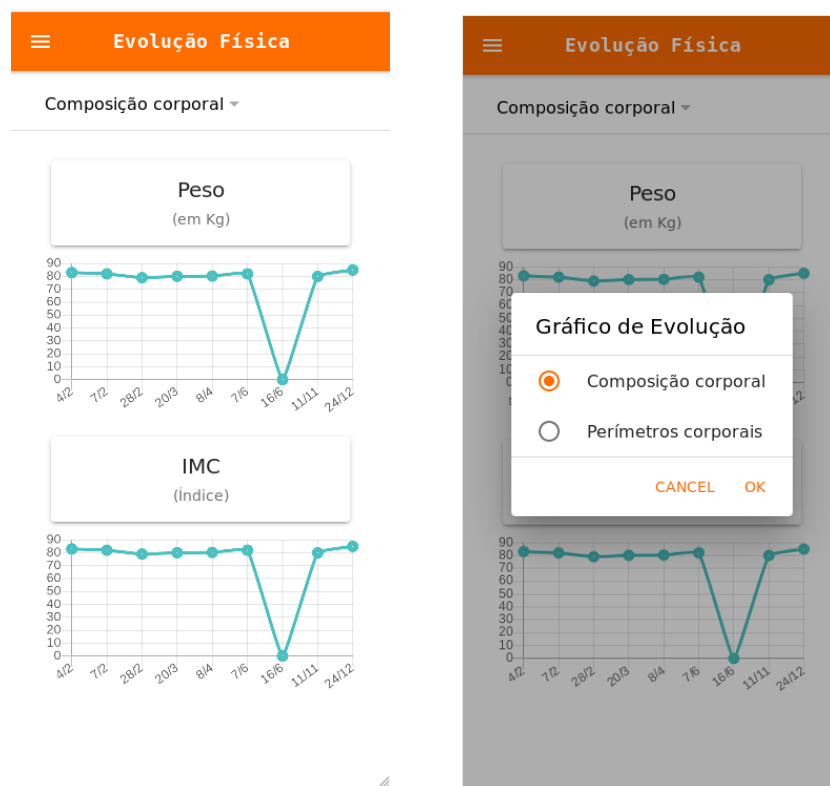


Figura 17: Página de evolução do cliente

Na página da evolução o cliente tem acesso aos dados adquiridos nas diversas avaliações, o que permite perceber qual o progresso feito no ginásio.



Figura 18: Plano de Aulas

Os clientes podem aceder ao planeamento de aulas para um dado dia da semana. Nesta funcionalidade o cliente pode aceder a bastante informação sobre a aula que pretende frequentar, é também possível visualizar a lotação da aula e fazer inscrição na mesma.

The image displays two mobile application screens for editing a user profile. Both screens have an orange header with a hamburger menu icon and the title 'Atualizar perfil'.

Left Screen:

- Fields to be updated (indicated by a key icon):
 - Definir nova password
 - Nova Localidade
- Fields to be updated (indicated by an 'X' icon):
 - Nif
- Fields to be updated (indicated by an 'X' icon and a dropdown arrow):
 - Tipo de s...
- Fields to be updated (indicated by an 'X' icon and a dropdown arrow):
 - Género:
- Fields to be updated (indicated by an 'X' icon and a calendar icon):
 - Data de Nasci...
- Buttons at the bottom:
 - ATUALIZAR INFORMAÇÕES (green button)
 - LIMPAR FORMULÁRIO (grey button)

Right Screen:

- Fields to be updated (indicated by an 'X' icon):
 - Nome Completo *
 - E-Mail *
- Fields to be updated (indicated by a key icon):
 - Definir nova password
 - Nova Localidade
- Fields to be updated (indicated by an 'X' icon):
 - Nif
- Fields to be updated (indicated by an 'X' icon and a dropdown arrow):
 - Tipo de s...
- Fields to be updated (indicated by an 'X' icon and a dropdown arrow):
 - Género:

Figura 19: Página de edição de perfil

Qualquer utilizador pode atualizar o seu perfil, modificando a sua password e/ou a sua localidade, conforme o desejado.

13.3 PÁGINA CLIENTE PREMIUM

Certas funcionalidades do cliente estão apenas disponíveis nesta versão premium, como tal vamos demonstrar as mesmas e as principais diferenças entre um cliente regular e um cliente premium.

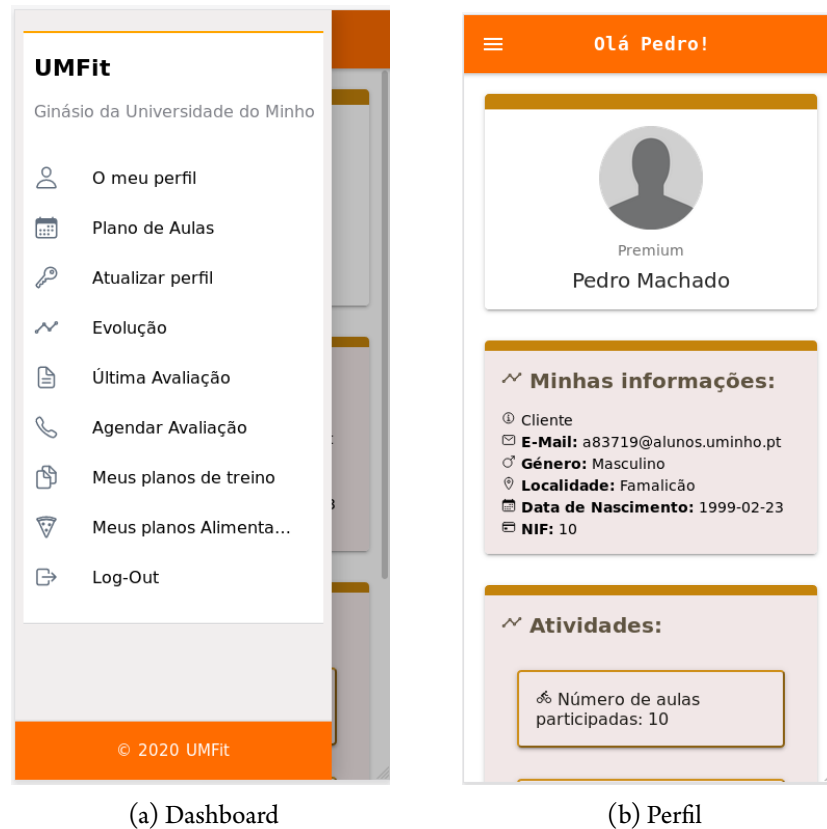


Figura 20: Funcionalidades de Cliente premium

O dashboard do cliente premium difere do cliente standard apenas nos planos de treino e planos alimentar, funcionalidades estas que apenas estão disponíveis a um utilizador com conta premium.

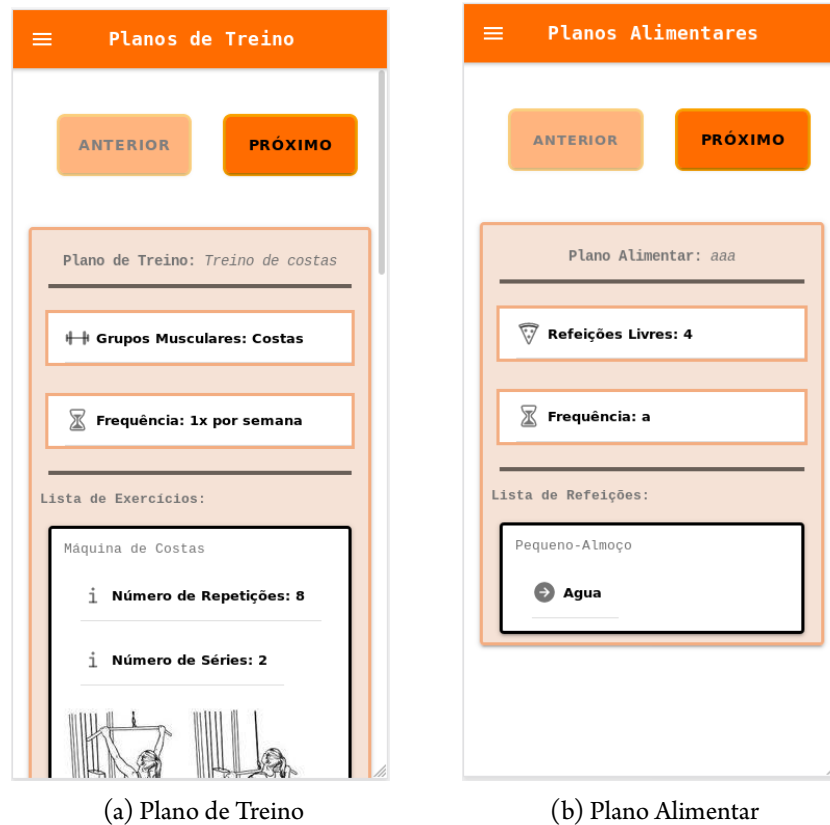


Figura 21: Visualização dos planos do cliente

Como já referido o cliente premium tem acesso a funcionalidades extra que são as mostradas (**Fig21**). A página plano de treino e plano alimentar permitem aceder a planos específicos ao cliente que ajudem no desenvolvimento do seu objetivo.

13.4 PÁGINA INSTRUTOR AUTENTICADO

É também possível que um utilizador seja um instrutor. Este tipo de utilizador terá funcionalidades específicas que permitem verificar a performance de todos os clientes do ginásio e ajudar os mesmos criando planos alimentares ou de treino.

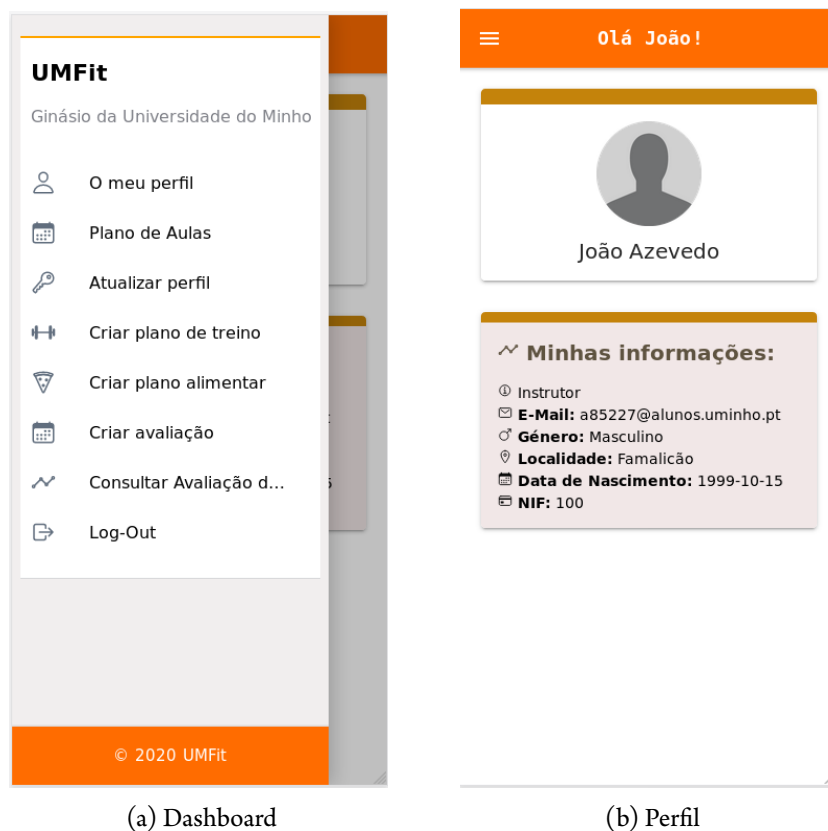


Figura 22: Funcionalidades do instrutor

Tal como nos tipos de utilizadores falados previamente, o dashboard do instrutor apresenta as funcionalidades que lhe são permitidas executar, este é demonstrado na imagem anterior.

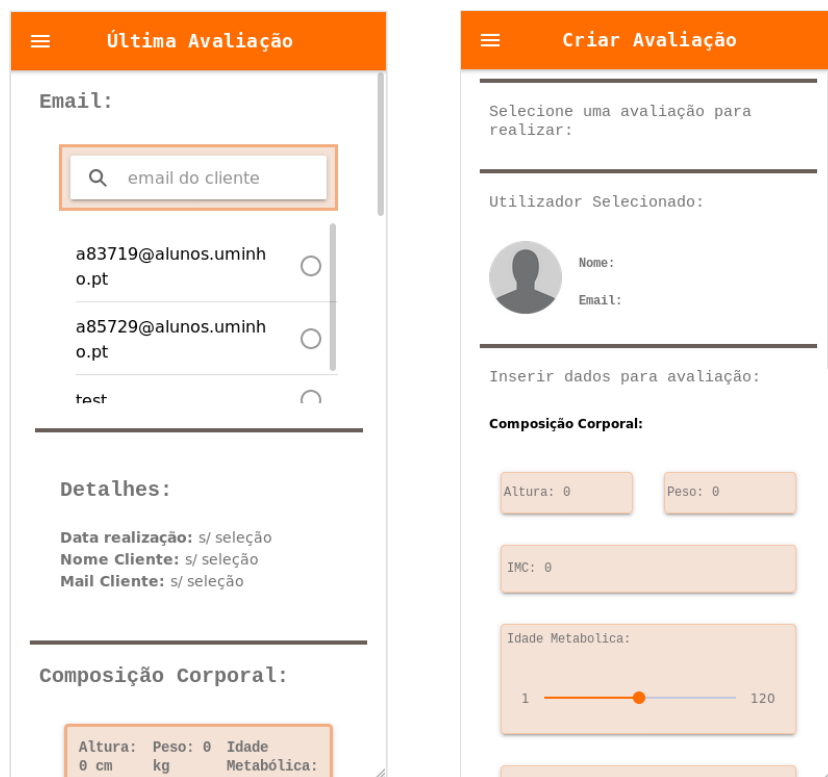


Figura 23: Visualização e criação de avaliações de clientes


Como já falado anteriormente é suposto que o perfil de um instrutor seja capaz de criar e visualizar novas atividades dos clientes, é por isso mostrado nas imagens seguintes algumas funcionalidades tal como a criação ou visualização de uma avaliação (**Fig23**), e a criação de planos alimentares ou de treino para um certo cliente (**Fig24**).

Plano Alimentar

Email (para clientes Premium):

☐

Utilizador Selecionado:



Nome:
Email:

Especificações do Plano Alimentar:


Nome:

Plano Treino

Email (para clientes Premium):

☐

Utilizador Selecionado:



Nome :
Email :

Especificações do Plano de treino:

Nome :

Tipo :

(a) Plano alimentar

(b) Plano de treino

Figura 24: Criação de planos para clientes

13.5 PÁGINA RECEPCIONISTA AUTENTICADO

Um utilizador pode ainda ser um rececionista, este tipo de utilizador funciona com um tipo de Administrador da aplicação. Ou seja, é ele que cria outros utilizador e os elimina se achar necessário. Tem ainda a si associadas funcionalidades como edição do plano de aulas semanal e entre outros.

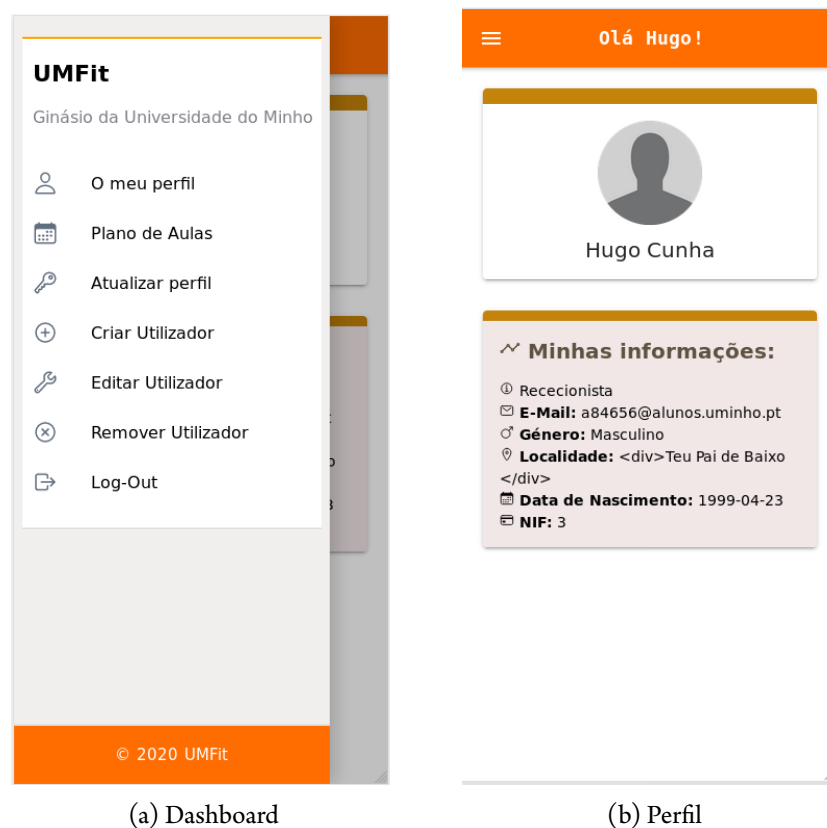


Figura 25: Funcionalidades da Rececionista

Sendo este também um utilizar a página inicial não se diferencia muito dos demais mantendo portanto a mesma estrutura. A *dashboard* apresenta no entanto funcionalidades diferentes de todos os outros, e que este pode executar, como ilustrado na imagem anterior.

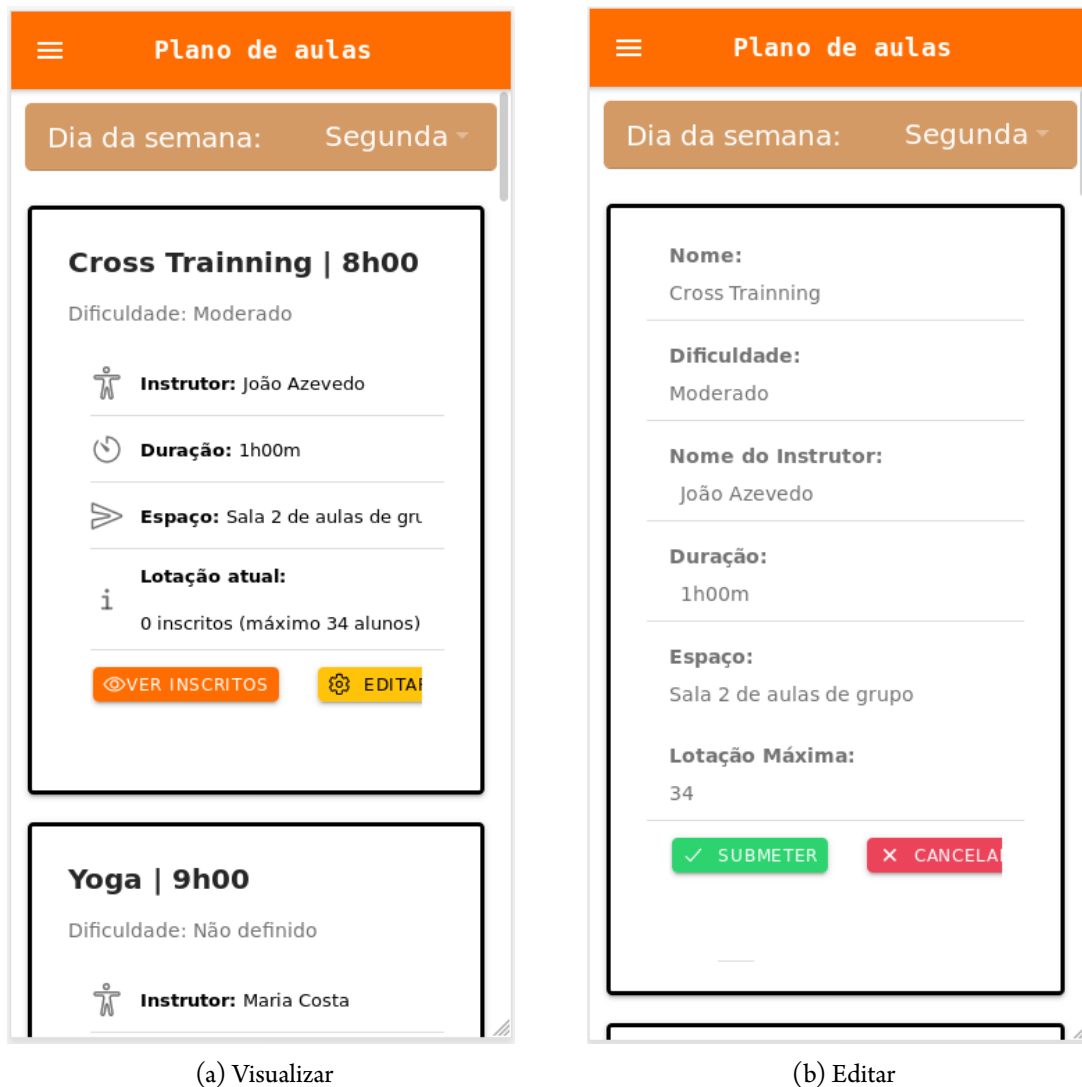


Figura 26: Plano semanal de Aulas

Aqui o recepcionista pode visualizar, tal como os outros clientes o plano semanal de aulas, com uma pequena diferença de que este pode alterar todas as informações referentes a uma aula, através do botão de editar. Pode portanto alterar desde o nome da aula até à sua lotação máxima, bastando para tal clicar no botão de submeter, ou descartar todas as alterações através do botão cancelar.

Two side-by-side screenshots of a mobile application interface for updating a user profile. Both screens have an orange header with a menu icon (three horizontal lines) and the title "Atualizar perfil".

The left screenshot shows a form with the following fields and options:

- Definir nova password** (with a key icon)
- Nova Localidade** (with a key icon)
- Nif** (with an 'X' icon)
- Tipo de s...** (with a person icon and a dropdown arrow)
- Género:** (with a person icon and a dropdown arrow)
- Data de Nasci...** (with a calendar icon)

At the bottom of the left screen are two buttons: a green button labeled "ATUALIZAR INFORMAÇÕES" and a grey button labeled "LIMPAR FORMULÁRIO".

The right screenshot shows the same form but with a profile picture placeholder (a grey circle with a person silhouette) at the top. The fields below are:

- Nome Completo *** (with an 'X' icon)
- E-Mail *** (with an 'X' icon)
- Definir nova password** (with a key icon)
- Nova Localidade** (with a key icon)
- Nif** (with an 'X' icon)
- Tipo de s...** (with a person icon and a dropdown arrow)
- Género:** (with a person icon and a dropdown arrow)

At the bottom of the right screen are the same two buttons: a green button labeled "ATUALIZAR INFORMAÇÕES" and a grey button labeled "LIMPAR FORMULÁRIO".

Figura 27: Atualizar dados do perfil

Pode ainda atualizar o seu perfil tal como um outro utilizador qualquer permitindo-lhe assim alterar a palavra-passe e a sua localidade.

The image displays two mobile application screens for creating a new user, titled "Novo utilizador".

Left Screen:

- Header: Novo utilizador
- Profile icon placeholder.
- Form fields:
 - Nome Completo *
 - E-Mail *
 - Definir a password *
 - Localidade
 - Nif
 - Tipo de sócio:

Right Screen:

- Header: Novo utilizador
- Form fields:
 - Definir a password *
 - Localidade
 - Nif
 - Tipo de sócio:
 - Género:
 - Data de Nas: Feb 20, 1999
- Buttons:
 - CRIAR NOVO UTILIZADOR (Green)
 - LIMPAR FORMULÁRIO (Grey)

Figura 28: Criar Utilizador

Nesta interface o rececionista controla a existência de novos utilizadores com a criação dos mesmos. Ou seja sempre que um cliente do ginásio deseja aceder à aplicação do ginásio, terá para tal de entrar em contacto com o rececionista para que este lhe crie uma nova conta e lhe forneça os dados.

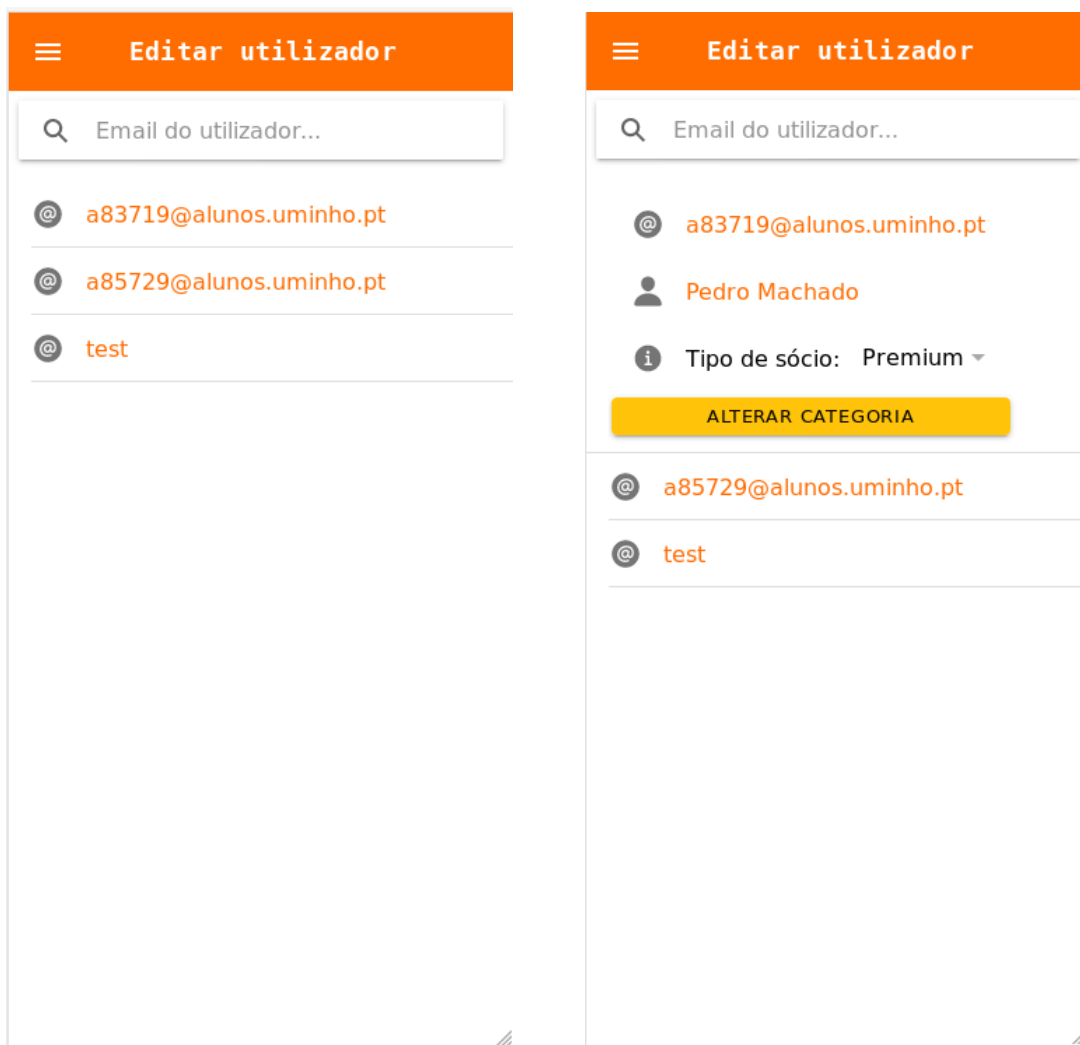


Figura 29: Editar um tipo de Utilizador

Esta nova funcionalidade permite ao rececionista conseguir alterar o tipo de sócio de um cliente. Para tal existe uma barra de pesquisa onde deve ser inserido o nome do cliente pretendido para alterar. Depois de selecionado o cliente correto pode ser alterado o seu tipo quer seja para *premium* ou para *standard*.

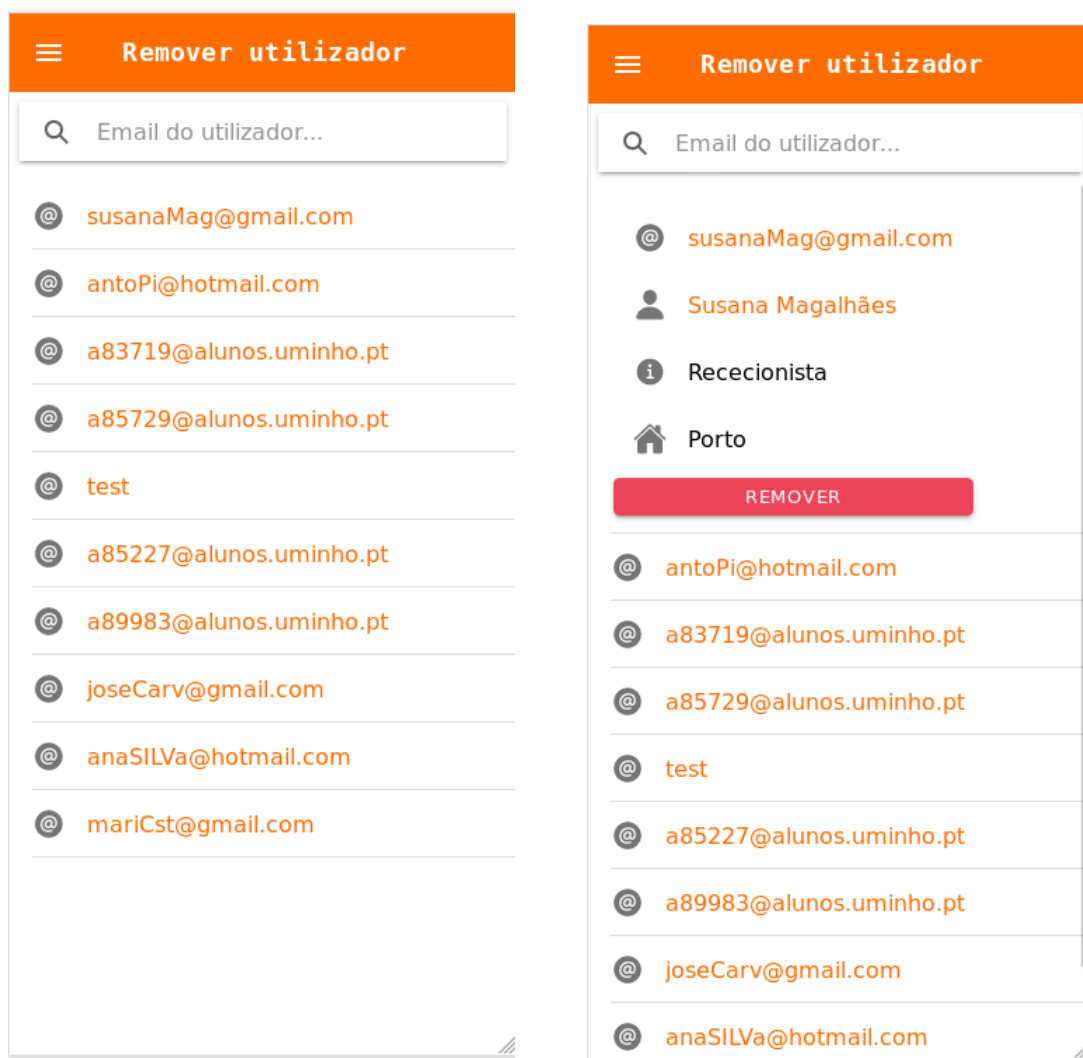


Figura 30: Remover um Utilizador

Para controlar o fluxo de clientes dentro da aplicação o rececionista pode ainda remover clientes que já não se encontrem nos quadros do ginásio. Para evitar manter dados "não necessários" na base de dados. Sendo assim o rececionista pode procurar por um cliente pretendido e em seguida removê-lo da aplicação.

Tal como um outro utilizador qualquer tem ainda a opção de *log-out* da aplicação.

13.6 DEPLOY DA API

Para o *deployment* da API utilizamos a plataforma na *Cloud* que fornece serviços computacionais e de execução de aplicações, o *Microsoft Azure*, que aliado a uma licença para estudantes nos permitiria, para fins de demonstração prática dos serviços da nossa aplicação, obter um crédito finito (totalizado em *100 euros*) com vista a alojar a API da UMFit obtendo assim um domínio próprio para ser utilizado como ponte remota no acesso aos mesmos recursos.

O domínio obtido é o que se apresenta no seguinte excerto, assim como alguns caminhos possíveis:

```
URL      : https://umfit-api.azurewebsites.net/
API      : https://umfit-api.azurewebsites.net/api/...
Caminhos : /api/aulagrupo
           /api/user
           /api/...
```

Tendo isto pronto, foi apenas preciso adicionar este domínio aos acessos efetuados na aplicação cliente *web*.

Por fim, é importante referir que grande parte da *performance* obtida na aplicação *web* é diminuída pela máquina que aloja a nossa API, correspondendo ao serviço mais barato da *Microsoft* onde é utilizada pouca memória *RAM* e apenas uma máquina com um processador.

13.7 DEPLOY DA INTERFACE

No que toca à interface e a disponibilizá-la de forma pública, todas as decisões visavam obter uma comunicação *HTTPSecure* com o servidor onde esta aplicação seria alojada, deste modo, a principal razão por detrás disto estava numa exigência imposta por uma aplicação seguindo um padrão *PWA*, que para ser *installable* teria de ser servida sobre uma ligação segura, para além de outros pormenores.

Existiram diversas opções que funcionariam para cumprir este nosso requisito, sendo

que alojar a aplicação no *Microsoft Azure*, referido anteriormente, seria uma boa opção, mas não ótima, no sentido em que teríamos de ser mais cuidadosos na utilização do crédito disponibilizado. A *Google* disponibiliza uma plataforma *Firebase* que também serve para a nossa utilização, no entanto implicava fazer uma série de registos e disponibilização de informações, trabalho esse que não será necessário na opção tomada no final.

Deste modo, decidimos utilizar um sistema, presente também no *GitHub*, que se trata do *Gitlab Pages* (semelhante ao *github.io*), mais simples, gratuito, sendo apenas preciso elaborar um ficheiro de configuração *.gitlab-ci.yml* contendo indicação dos *sources* da aplicação e os *scripts* necessários para gerar um *build* da mesma e hospedar esse *build* como uma *web app*. A sua configuração não vai ser demonstrada neste relatório, estando ela presente na *root* do nosso repositório no *Gitlab*, mas tem por base uma série de tutoriais na *internet* sobre este tema.

13.8 PROGRESSIVE WEB APP (PWA)

Esta secção é o *core* da nossa aplicação final e trata-se do objetivo principal, em termos visuais e práticos, que pretendíamos atingir com a nossa *app*.

Dentro das *Progressive Web Apps* existem um conjunto de requisitos a cumprir para que o *browser* reconheça a nossa página como elegível a ser transferida e utilizada em *Desktop* ou *Mobile* como uma aplicação como outra qualquer, sempre de forma segura (o tal *https* exigido).

Dos requisitos necessários para ser *installable*, esta deve ter um ficheiro *manifest.json*, que contém uma série de meta-dados necessários na altura de instalação, execução e visualização da aplicação, e um *service worker* que corresponde a um *script* que corre em segundo plano e que trata de todo o *caching* para as funcionalidades da aplicação que não necessitam de conexão a uma rede *internet*.

Após vários testes e todo o *setup* destes ficheiros nas diretorias corretas da nossa aplicação, acertando as referências lá mencionadas e meta-dados necessários, utilizamos sempre uma aplicação muito prática para testar a *performance*, *robustez*, verificar se estamos a optar por boas práticas na escrita de código e analisar os requisitos de uma *PWA*, aplicação essa da *Google*, o *Lighthouse*.

O *Google Lighthouse* recebe como *input* o nosso *url* e faz a geração de um relatório final da nossa aplicação sob a forma de uma página *html*, presente da diretoria *li4_umfit/other/pwa*. Seguem-se os aspetos relativos à *pwa* reportados por essa aplicação:

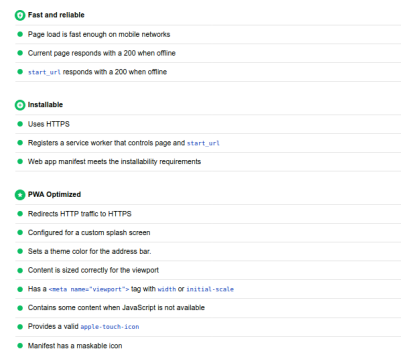


Figura 31: Requisitos de uma *pwa* reportados.

Desde a utilização de HTTPS com os serviços do GitLab até à implementação de ícons perfeitamente adaptáveis a qualquer equipamento, a nossa aplicação cumpre todos os requisitos para ser uma boa PWA, concluindo que a está devidamente configurada para o efeito.

Para comprovar que a aplicação pode ser instalada a partir de um *browser* comum, podemos ver o exemplo seguinte obtido a partir do *Google Chrome*:

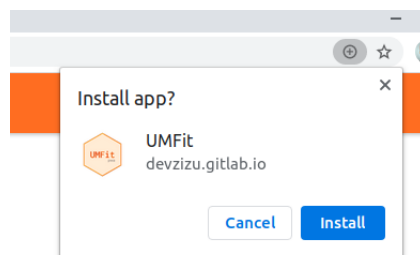


Figura 32: *PWA installation prompt*.

Por fim, o *score* global obtido pelo *Lighthouse* no que toca à aplicação como um todo, contando com aquele *bottleneck* causado pela API, estimando que sofremos cerca de 30 *por cento* de penalização na *performance* obtida, é o seguinte:

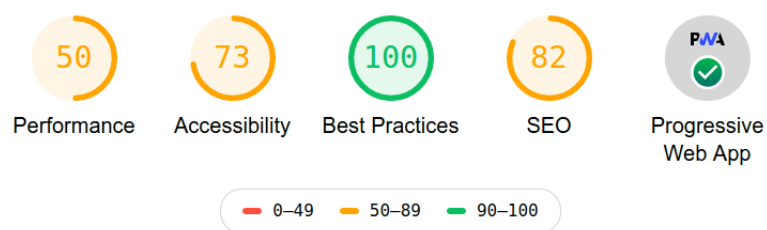


Figura 33: *Score* global reportado pelo *Lighthouse*.

CONCLUSÃO

A ideia de criar uma aplicação de apoio á atividade de um ginásio foi inicialmente difícil de organizar. No entanto o processo de idealização da UMFit permitiu alcançar novos limites que permitem que a utilização deste serviço ajude a resolver problemas até agora não muito abordados.

Durante o desenvolvimento do projeto procuramos encontrar as melhores soluções para agilizar os principais entraves num ginásio moderno. Procuramos também orientar o projeto, de forma a este incidir em funcionalidades com mais potencial para se diferenciar dos seus concorrentes. Esta conceptualização foi essencial para delinear um trajeto orientado aos nossos objetivos e facilitando a modelação do sistema de software.

Inicialmente foi necessário modelar qual o modelo necessário para a implementação do sistema, este passo foi fundamental para percebermos qual o formato mais indicado às necessidades do utilizador final.

O plano de desenvolvimento, que foi apresentado na forma de um Diagrama de Gantt, permitiu organizar e quantificar o número de horas de trabalho necessárias para o desenvolvimento da sistema proposto.

Desta forma, no final da primeira fase, sentimos que a fundamentação do nosso projeto ficou ilustrada de uma forma bastante clara e objetiva, consolidando assim os motivos associados ao desenvolvimento do mesmo

Na segunda fase do projeto, a Especificação, com o intuito de encadear o seu desenvolvimento com base nos componentes, foi concebido, através da análise detalhada dos requisitos

recolhidos, um modelo de domínio, de casos de uso, de classes, de base de dados e de comportamento da interface do utilizador.

A detalhada especificação dos casos de uso antecipou problemas não só de insatisfação dos requisitos, como também desmascarou incoerências estruturais nos desenhos iniciais que foram propostos nas várias interações.

Após o longo processo de implementação, o produto final é funcional, satisfaz os casos de uso e, acima de tudo, é flexível e utilizável o que sugere que trará bastante entusiasmo aos futuros utilizadores.