

Universidade do Minho
Escola de Engenharia

Engenharia de Aplicações
1º/4º ano MEI/MIEI
Gym@Home
Relatório de Desenvolvimento

João Paulo Oliveira de Andrade Marques
(a81826@alunos.uminho.pt)

José André Martins Pereira
(a82880@alunos.uminho.pt)

Ricardo André Gomes Petronilho
(a81744@alunos.uminho.pt)

20 de Março de 2021

Conteúdo

1	Introdução	3
2	Contextualização	4
3	Glossário	5
4	Levantamento e Análise de Requisitos	6
4.1	Requisitos Comuns	6
4.2	Requisitos Cliente	6
4.3	Requisitos Personal Trainer	7
5	Modelo de domínio	8
6	Arquictetura	12
6.1	Arquitetura orientada a serviços	12
6.2	Client Service	13
6.2.1	Factory Method	15
6.2.2	Adpater	16
6.2.3	Hibernate	16
6.2.4	EJB	16
6.2.5	Multi-camada e Facade	17
6.2.6	Singleton	17
6.2.7	REST vs WSDL	17
6.2.8	Token Authentication	17
6.3	Personal Trainer Service	18
6.4	Core	19
6.4.1	Método finishWorkout	21
6.4.2	Builder	21
6.5	Requests Service	21
6.6	Notifications Service	22
6.7	GymAtHome Service	23
6.7.1	Propagação dos tokens	23
6.8	Frontend service	23
6.8.1	MVC - Model View Controller	23

7 Usabilidade	25
7.1 Conhecer os utilizadores	25
7.2 Princípios de usabilidade	25
8 Princípios de Usabilidade presentes e avaliação por Heurísticas de Normam das Interfaces	27
8.1 Princípios/heurísticas genéricos(as) a todas as views	27
8.2 Interfaces Comuns	28
8.2.1 Mensagens de sucesso, erro e aviso	28
8.2.2 Login	29
8.2.3 Notificações	30
8.3 Interfaces Clientes	32
8.3.1 Registar Cliente	32
8.3.2 Cliente consulta/edita o próprio perfil	33
8.3.3 Cliente consultar o perfil do PT	35
8.3.4 Procurar Personal Trainer	36
8.3.5 Preencher Pedido	38
8.3.6 Consultar Semana do plano	39
8.3.7 Consultar Workout	41
8.3.8 Pedidos enviados pelo Cliente	42
8.3.9 Avaliar PT	43
8.4 Interfaces PTs	45
8.4.1 Registar PT	45
8.4.2 PT consulta/edita o próprio perfil	46
8.4.3 Consulta do perfil do Cliente pelo PT	47
8.4.4 Pedidos recebidos pelo PT	49
8.4.5 Clientes do PT	51
8.4.6 Criar Semana	52
8.4.7 Semana do Cliente vista pelo PT	54
8.4.8 Workout do Cliente visto pelo PT	55
8.5 Alertas	57
9 Deployment	58
9.1 Ficheiros de configuração necessários para o deployment	58
9.2 Processo deployment	60
9.2.1 Requisitos	60
9.2.2 Passo a passo	60
9.3 Preparado para Docker Swarm	60
10 Conclusão	61

Capítulo 1

Introdução

No Mestrado integrado de Engenharia Informática, no perfil de Engenharia de Aplicações, nas unidades curriculares de Arquitecturas Aplicacionais e Sistemas Interactivos foi proposto o desenvolvimento de uma aplicação web seguindo todos os princípios lecionados ao longo do semestre.

Ao longo do relatório apresenta-se todas as etapas efectuadas para a concretização deste projecto. O projecto o qual intitulou-se de "**Gym@Home**" consiste no desenvolvimento de uma aplicação web de acompanhamentos de treino online criados por **personal trainers** e realizados por **clients**.

Inicialmente faz-se um contextualização do problema, apresentando o domínio do mesmo, bem como os seus objectivos.

Após o conhecimento do problema, apresenta-se a terminologia do meio e a lista de requisitos funcionais e não funcionais do projecto bem como a sua análise, sendo bastante importante visto que reflectem as funcionalidades do sistema.

De seguida, apresenta-se a modelação desenvolvida: modelo domínio, diagrama de classe. De forma, mais detalhada e pormenorizada em relação ao desenvolvimento, apresenta-se o diagrama de classes, com a estrutura global do sistema e as respectivas decisões e explicações necessárias sobre o mesmo, bem como identificação de problemas que de alguma forma foram prevenidos. Ainda relacionado com o diagrama de classes, também se explica a obtenção de código a partir do mesmo, bem como o código da framework **Hibernate** para a persistência de dados.

De seguida, apresenta-se a prototipagem das interfaces, bem como o resultado final. Para cada interface explica-se o que fazem, os princípios encontrados nas mesmas, bem como as heurísticas de norman por forma a avaliar o resultado.

Após a explicação dos vários componentes na modelação, explica-se como se faz o deployment automatizado da infraestrutura/arquitectura da aplicação utilizando docker-compose.

Por último será feita uma avaliação geral do projecto, com objectivos atingidos, pontos a melhorar e trabalho futuro.

Capítulo 2

Contextualização

Na sequência da pandemia COVID-19 (Sars-Cov-2) e na queixa de alguns Personal Trainers sobre a falta de plataformas onde pudessem ter uma ligação mais personalizada com clientes à distância, o grupo de trabalho propôs a criação do **Gym@Home**. A plataforma passará por permitir a Personal Trainers criarem planos de treino para os seus clientes segundo as suas especificações e a sua condição física.

Desta forma os Clientes terão a possibilidade de encontrar Personal Trainers do seu agrado e dizer o tipo de Plano de treino que pretende efetuar. Os Personal Trainers a quando da recepção dos pedidos poderão decidir sobre se querem ou não criar um Plano para esse Cliente.

Capítulo 3

Glossário

- **PT:** Personal Trainer.
- **Plano:** Conjunto de Semana.
- **Semana:** Conjunto de Workouts.
- **Workout:** Conjunto de Tarefas.
- **Tarefa:** Conjunto de Séries.
- **Série:** Elemento mais básico de um Plano, constituído por uma duração/repetições, um tempo de descanso e uma descrição.
- **UI - User Interface**
- **AJAX - Asynchronous JAVaScript and XML**
- **HTTP - HyperText Transfer Protocol**
- **API: Application Programming Interface**
- **DAO: Data Access Object**
- **CRUD: Create Read Update Delete**
- **EJB: Enterprise Java Beans**
- **Java EE: Java Enterprise Edition**
- **JNDI: Java Naming and Directory Interface**
- **WSDL: Web Services Description Language**
- **REST: REpresentational State Transfer**
- **BD: Base de Dados.**
- **VM: Virtual Machine.**

Capítulo 4

Levantamento e Análise de Requisitos

Na presente secção serão ilustrados os requisitos levantados e analisados para o problema em questão. Existindo dois tipos de utilizadores os requisitos estão divididos em três partes, a parte comum a ambos, a parte do Cliente e a parte do PT.

4.1 Requisitos Comuns

- **Login:** os utilizadores necessitam de se autenticar para usufruir do sistema.
- **Logout:** os utilizadores podem fazer logout para terminar a sessão na plataforma.
- **Registar:** os utilizadores necessitam de se registar no sistema.
- **Visualizar Perfil:** os utilizadores poderão visualizar o seu perfil e o perfil de outros utilizadores.
- **Editar Perfil:** os utilizadores poderão editar os seus dados para os manter actualizados.
- **Ver Notificações:** os utilizadores podem visualizar notificações para saber acções que outros utilizadores tomaram no sistema relacionadas com ele.
- **Visualizar Semana:** os utilizadores podem visualizar a semana, tendo a perspectiva de todos os workouts para essa semana e também a sua condição física através dos dados biométricos do Cliente.
- **Visualizar Workout:** os utilizadores podem visualizar o workout, vendo assim as tarefas desse workout.

4.2 Requisitos Cliente

- **Realizar Workout:** o Cliente realiza workouts.
- **Procurar Personal Trainer:** o Cliente procura por PTs podendo especificar alguns filtros para uma selecção mais refinada de um PT.
- **Fazer um Pedido:** o Cliente preenche um formulário com o tipo de plano que pretende realizar.
- **Avaliar Personal Trainer:** o Cliente no final do Plano pode avaliar o PT.
- **Visualizar Pedidos Feitos:** o Cliente visualiza todos os pedidos que realizou a PTs.

4.3 Requisitos Personal Trainer

- **Criar Semana:** o PT cria uma semana para ao plano do cliente.
- **Responder a Pedido:** o PT aceita ou rejeita pedidos efectuados pelos clientes.
- **Visualizar Clientes:** o PT vê os seus clientes onde poderá obter informações sobre os seus dados pessoais e acrescentar semanas ao seu plano.
- **Visualizar Pedidos Recebidos:** o PT visualiza os pedidos pendentes enviados pelos Clientes.

O diagrama de Use Cases não está presente, pois este é directamente análogo aos requisitos levantados. O diagrama de Use Cases estaria dividido em dois subsistemas, que seria o do Cliente e o do PT, sendo que no do PT estariam todos os Use Cases respectivos aos requisitos comuns e do PT, no subsistema do Cliente estariam os requisitos comuns e do Cliente.

Capítulo 5

Modelo de domínio

O modelo de domínio pode ser dividido em 2 grandes grupos que mais à frente, na arquitectura, são traduzidos em 2 sub-sistemas e mesmo serviços independentes. Existe o grupo que representa a informação dos utilizadores e, por outro lado, o grupo que contém as entidades principais referentes ao modelo de negócio.

De seguida apresentam-se as três entidades do grupo dos utilizadores: **Utilizador**, **Cliente** e **Personal Trainer**. Na seguinte figura encontra-se a informação referente ao Utilizador.

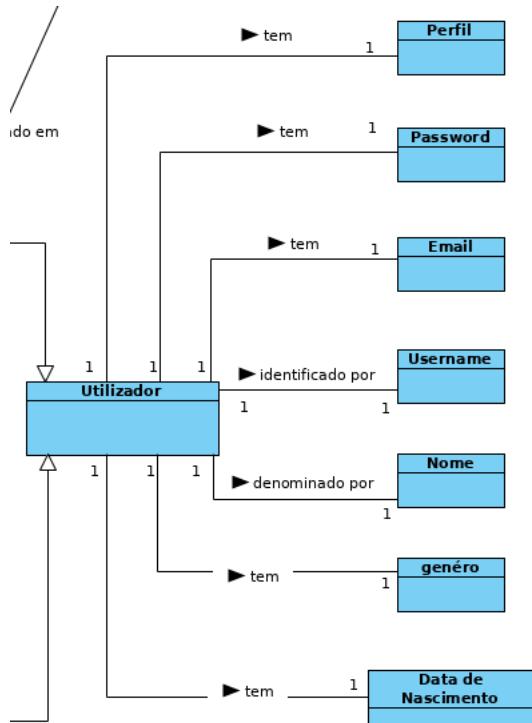


Figura 5.1: Entidade Utilizador.

Tal como se pode observar a entidade **Utilizador** representa a informação comum a ambos os utilizadores, essa informação é: o **username** pelo qual o mesmo é identificado, o **nome** pelo qual é denominado, o **email** para pode ser contactado, a **password** que assegura a sua conta, o **género** e por última a sua

data de nascimento (a partir da mesma pode ser derivada a sua idade). Juntamente com a informação referida e outras, específicas a cada utilizador (referidas adiante), o Utilizador contém um **perfil** que pode ser consultado por outros.

Existem dois tipos de utilizadores: o **Cliente** e o **Personal Trainer**. De seguida seguida encontra-se representado o Cliente.

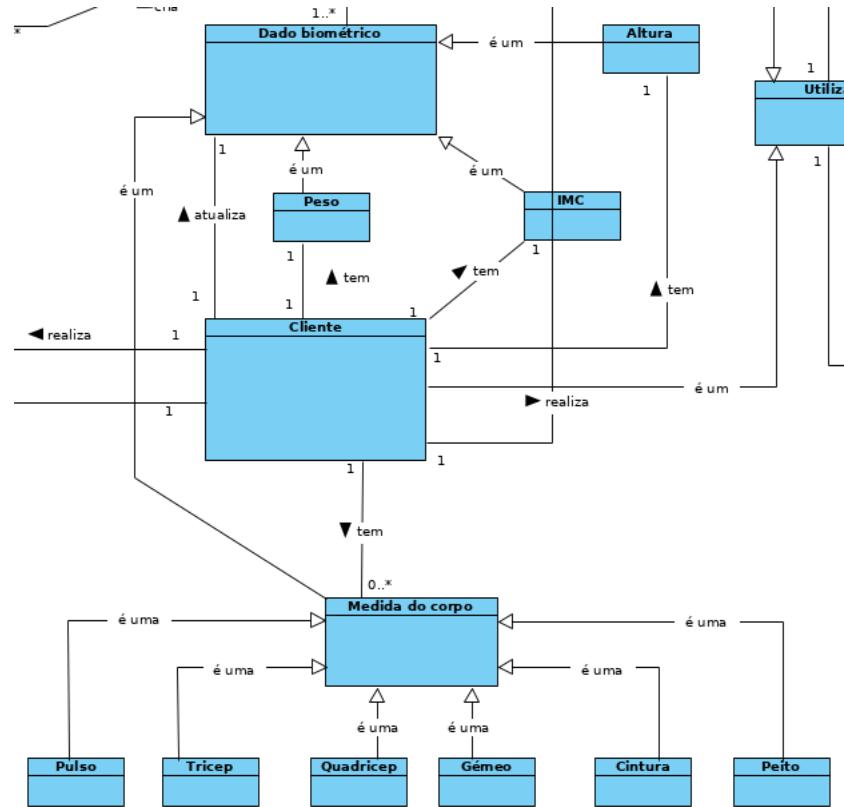


Figura 5.2: Entidade Cliente.

O Cliente, a acrescentar à informação já referida, pode conter oito **dados biométricos**, sendo que as medidas: do **pulso**, **tríceps**, **quadríceps**, **gémeos**, **cintura** e **peito** são dados opcionais, no entanto a introdução do **peso** e **altura** é obrigatória no momento de registo no sistema. O **IMC - Índice de Massa Corporal** - é derivado através de uma fórmula a partir do peso e altura sendo que as categorias (do IMC) que apresentamos no sistema seguem os dados fornecidos pela OMS - Organização Mundial de Saúde. Todos estes dados podem ser actualizados a qualquer momento e são sempre preservados os registos anteriores de modo a existir uma noção de histórico/ evolução. Note-se que apesar de opcionais, é recomendado a introdução de todos os dados uma vez que são métricas que permitem um acompanhamento mais personalizado e conclusivo sobre a evolução do plano de treino permitindo ajustar o plano consoante os resultados.

O Personal Trainer é representado de seguida.

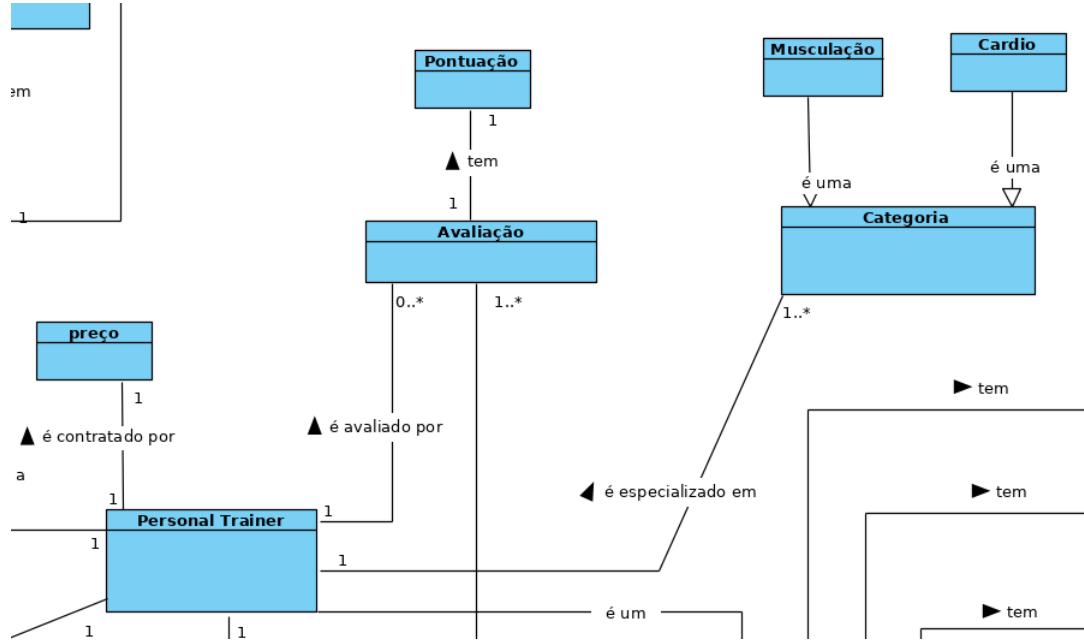


Figura 5.3: Entidade Personal Trainer.

Do mesmo modo, além da informação comum, contém também o **preço** que aplica ao Cliente por cada Workout (entidade explicada mais à frente), contém também a **categoría** em que é especializado, podendo ser em **musculação ou cardio**. Esta categoria pode ser útil quando o Cliente está a escolher o Personal Trainer que deseja ou mesmo pode ser útil ao próprio Personal Trainer para aceitar ou recusar um pedido de plano de treino uma vez que esse pedido pode não corresponder à sua categoria. A **avaliação** pode ser, também, decisiva no momento da escolha. Esta é realizada pelo Cliente uma única vez durante o plano de treino, numa escala de 1 a 5 estrelas (**pontuação**), sendo registada e associada às avaliações já existentes, calculando-se a média das mesmas.

De seguida apresentam-se as entidades principais do modelo de negócio.

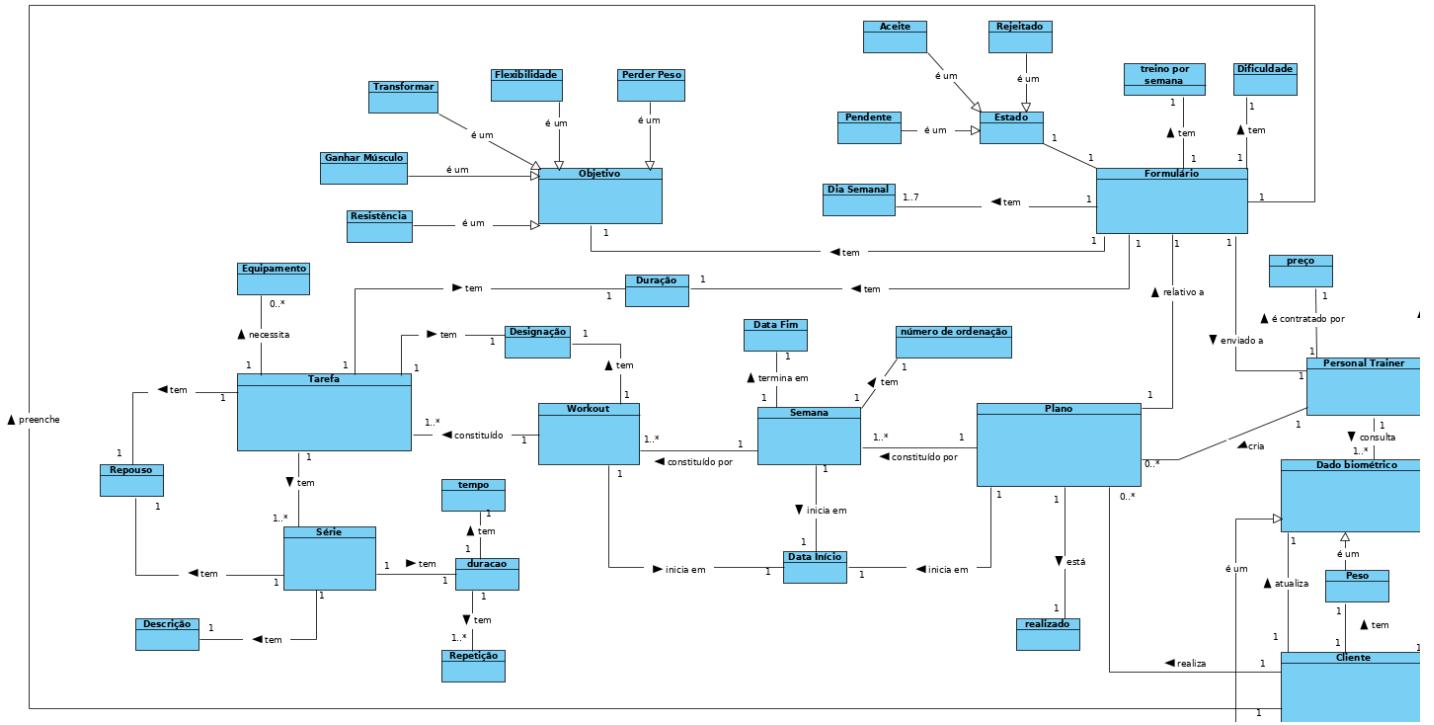


Figura 5.4: Entidades principais do modelo de negócio.

Para iniciar um **plano** de treino, previamente, o Cliente tem de preencher um **formulário** onde indica a **dificuldade** que pode ser: **fácil, normal, difícil ou extrema**; é indicado também o número de **treinos semanais** juntamente com os **dias semanais** que tem disponibilidade, o **objectivo** que pode ser: **perder peso, ganhar músculo, resistência, flexibilidade e transformar gordura em massa magra**; e por último a **duração** em semanas.

O formulário fica com o **estado pendente** até ao Personal Trainer o **rejeitar** ou **aceitar**. Caso o Personal Trainer aceite, imediatamente pode criar a primeira **semana do plano** de treino.

Uma semana ocorre entre duas datas: **data de início** e **data de fim**, contém também (internamente) o **número (de ordenação)** da semana no plano de treino.

A **semana** é constituída, também, por 1 ou mais **Workouts** que representa um dia de treino, por isso ocorre numa determinada **data** (num dia da semana), a **designação** que resume o treino nesse dia e a lista de **tarefas** a realizar.

Uma tarefa contém, também, uma **designação**, uma **duração** (em minutos), pode necessitar de **equipamento** e as sucessivas tarefas são intervaladas por um **tempo de repouso**, por último é constituída por várias **séries**.

A **série** é a unidade mais básica do plano de treino, contém uma **descrição** que indica a actividade a ser realizada. Tal como a tarefa, cada série é também intervalada com um **tempo de repouso**. Por último contém uma **duração** que pode ser em **tempo** (segundos ou minutos) ou em **número de repetições**. Para clarificar a noção de duração observe-se os seguintes exemplos: caso a série seja realizar 30 abdominais, a duração é 30 repetições, por outro lado, caso a série seja realizar a o movimento de prancha durante 1 minuto, a duração é em tempo (neste caso em minutos).

Capítulo 6

Arquitectura

Na elaboração da arquitectura o primeiro passo foi definir as entidades em termos de classes do sistema, maior parte já detectadas no modelo de domínio. Durante esta fase o grupo apercebeu-se que fazia sentido dividir as entidades em dois grupos que se traduzem em dois subsistemas: os **Recursos Humanos** e o **Core**. Note-se que estes sub-sistemas a nível de implementação traduzem-se em dois packages.

Os recursos humanos contém as classes que representam os utilizadores do sistema, ou seja, o cliente e personal trainer.

No entanto após alguma pesquisa e discussão de ideias entre o grupo, percebeu-se que na actualidade as aplicações webs ou outras são orientadas a serviços.

6.1 Arquitetura orientada a serviços

Desta forma, a ideia inicial foi separar os dois packages definidos anteriormente o **core** e os **recursos humanos**, bem como o frontend, sendo já uma boa arquitectura. No entanto, percebeu-se que poderia ser mais bem dividido, para se tirar melhor partido, dessa forma chegou-se aos sete serviços:

- Core: responsável pelas funcionalidades que interagem com planos.
- Requests: responsável pelas funcionalidades que interagem com os pedidos/formulários.
- Notifications: responsável pelas funcionalidades que interagem com as notificações.
- Clients: responsável pelas funcionalidades que interagem com os clientes.
- PersonalTrainers: responsável pelas funcionalidades que interagem com os PTs.
- GymAtHome: API do backend, ou seja, dispatcher dos pedidos para os serviços correctos.

A tomada desta decisão passou por uma verificação das vantagens e desvantagens do desenvolvimento deste tipo de arquitecturas.

As **vantagens** de desenvolver um arquitectura orientada a serviços são as seguintes:

- divisão de trabalho mais fácil, aumentando assim a produtividade.
- divisão de responsabilidades.
- são mais contidos, dessa forma, mais fácil encontrar erros, bem como fazer deployment.

- permitem escalabilidade da arquitectura para aumento de carga.
- a falha de serviços não provoca directamente a falha do sistema, isto é, resiliência.
- quando são feitas alterações ao código, apenas necessita-se de fazer deployment do serviço alterado, não precisando de esperar por outros serviços, ou seja, deployment contínuo.
- permitem mais flexibilidade da arquitectura.
- ...

As **desvantagens/faláncias** cometidas quando de desenvolve orientado a serviços:

- pensar que a banda larga é infinita.
- pensar que não existe latência.
- pensar que os dispositivos de rede são uniformes.
- pensar que não existem hackers.
- necessidade de mais recursos.
- ...

No entanto, apesar de conter-se algumas desvantagens, as vantagens sobrepõem-se, obtendo-se melhores resultados a nível de escalabilidade com uma arquitectura orientada a serviços.

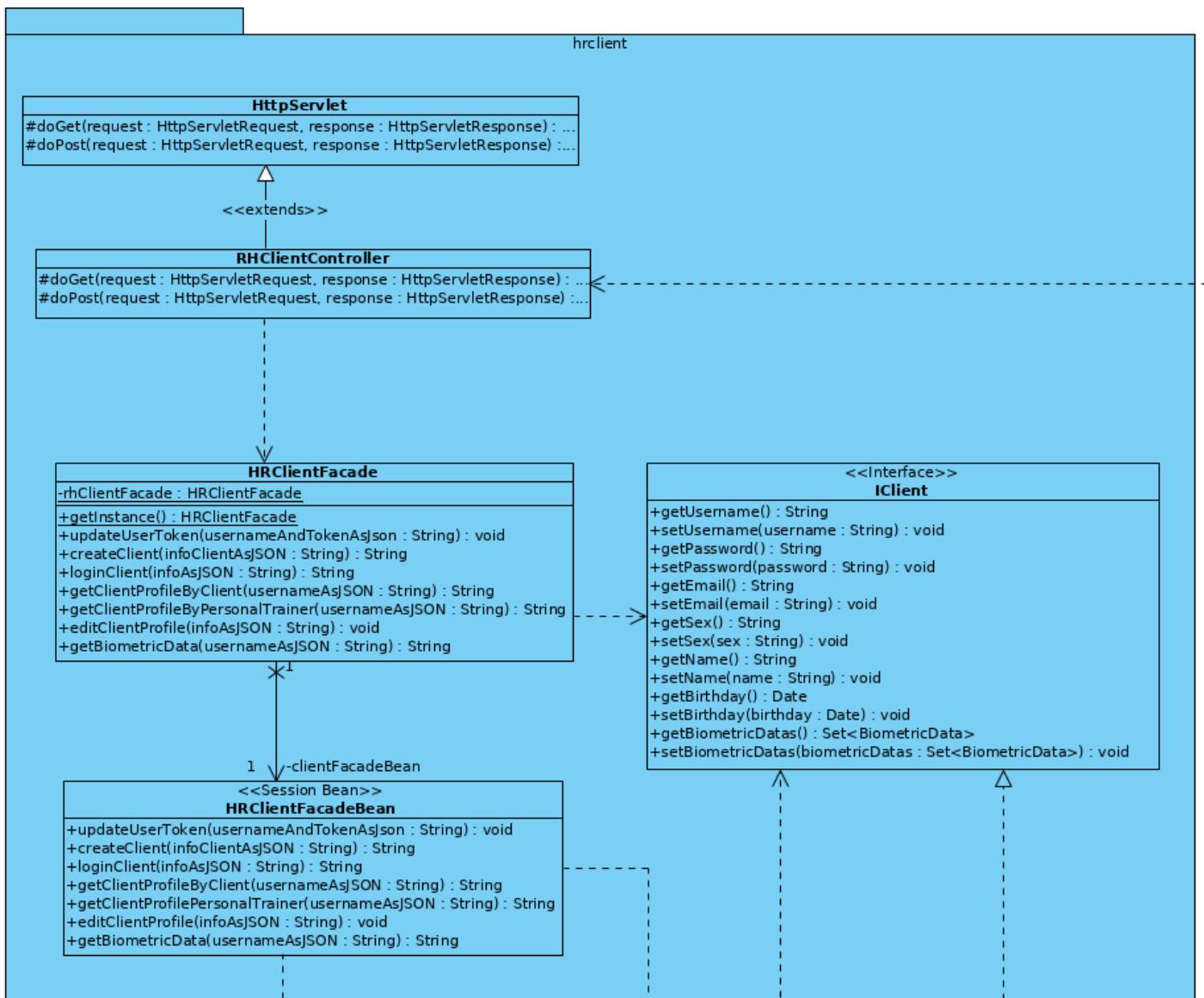
Na verdade, ”nem tudo é um mar de rosas”, ao desenvolver uma arquitectura orientada a serviços, acresce uma complexidade, que consiste em manter serviços consistentes, pois alguns depende dos dados de outros, como por exemplo os outros serviços dependem do clientes e personal trainers, pois quando existe o registo dos mesmos, tem-se que propagar essa **escrita** para todos os serviços. No entanto, mais uma vez, esta complexidade extra em alguns serviços, não sobrepõe as vantagens.

Outro problema encontrado nesta arquitectura foi a necessidade mais recursos de CPU e RAM do que o costume. No entanto em produção não seria problema, visto que seriam disponibilizados mais recursos.

De seguida, serão apresentadas as várias decisões, tomadas na arquitectura, bem como a apresentação dos serviços individualmente.

6.2 Client Service

O package HRClient (HR alusivo a Human Resource) encontra-se especificado nas seguintes figuras.



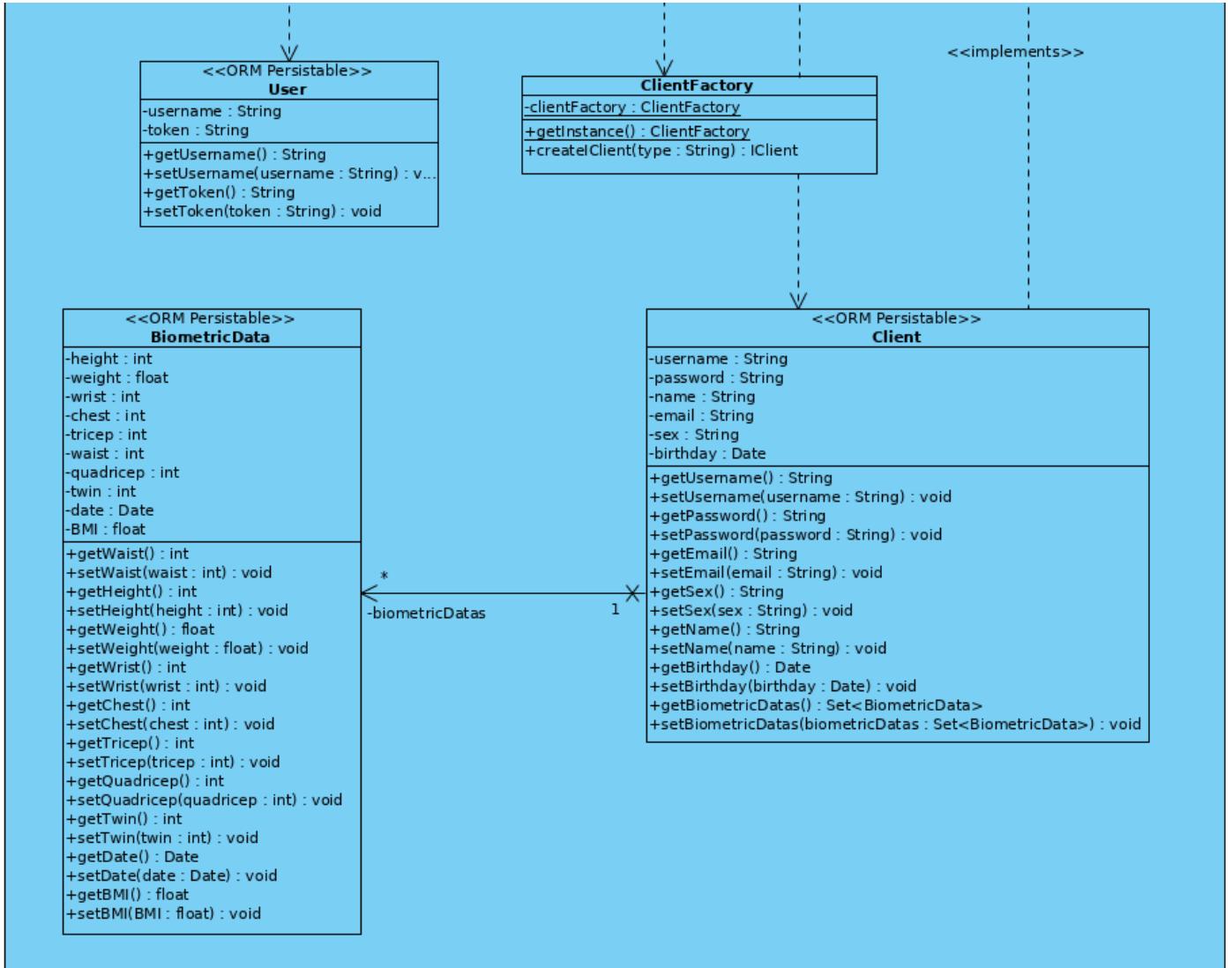


Figura 6.2: Diagrama de classes do package HRClient - parte 2.

Em relação ao cliente foi criada a classe **Client** com a informação específica do mesmo, tal como referido anteriormente no modelo de domínio, o cliente contém um histórico de dados biométricos, para isso foi criada uma associação entre a classe Client e a classe **BiometricData**. **Note-se que a relação é de um para muitos uma vez que existe a noção de histórico** apesar de não estar implementada a funcionalidade de visualização do mesmo a na UI. De seguida foi introduzida a interface **IClient** de forma a que as classes que utilizam a mesma dependam apenas da interface e não da classe concreta. Esta prática **facilita a evolução e manutenção do código** pois caso, no futuro, sejam introduzidas novas classes concretas que representam clientes (ex: ClientPremium, etc) desde que as mesmas implementem o tipo **IClient**, o código continuará a funcionar sem alterações.

6.2.1 Factory Method

Com o mesmo objectivo, para facilitar a evolução e manutenção do código, foi introduzido o design pattern **Factory Method** que é responsável pela criação dos clientes. Para tal foi criada a classe **Factory**

Method. A grande vantagem da utilização deste design pattern é que o processo de criação/ instanciação do objecto do tipo IClient passa a ser responsabilidade da classe ClientFactory, desta forma as classes que utilizam o IClient em vez de dependerem dos tipos concretos (uma vez que antes as próprias faziam a instânciação do objecto) passam a depender apenas da interface IClient e da classe ClientFactory. **Analisando a arquitectura actual, pode parecer que não houve vantagem na introdução do Factory Method** uma vez que antes existiam 2 dependências, com a interface IClient e a classe Client e após a introdução do mesmo continuam a existir o mesmo número de dependências, neste caso com a interface IClient e classe ClientFactory. No entanto, numa perspectiva evolutiva e pensando no futuro, caso sejam introduzidas N classes do tipo IClient, as dependências são reduzidas de $N+1$ (N classes + interface IClient) para 2 apenas (ClientFactory + IClient) por este motivo o grupo entendeu que se justifica a introdução do Factory Method.

Note-se que apesar do design pattern referido anteriormente ter sido introduzido na proposta inicial da arquitectura, após gerado o código, na prática acabou por não ter sido implementado pelo grupo uma vez que foi utilizada a API [gson](#), desenvolvida pela Google, responsável pela serialização e deserialização de objectos Java para o formato [JSON](#). Esta API internamente já implementa o Factory Method por isso não foi necessário uma implementação concreta pelo grupo. No entanto a classe ClientFactory permanece no código, apesar de não implementada, para que, caso, no futuro, seja necessária a sua utilização, a arquitectura já conte com a mesma. Para clarificar, uma possível hipótese de necessidade de utilizar a própria implementação por parte do grupo seria por exemplo, caso a API [gson](#) fosse descontinuada.

6.2.2 Adpater

Ainda devido à utilização da API referida anteriormente foi necessária a introdução do design pattern **Adapter** uma vez que a API apenas consegue serializar/deserializar certos tipos de objectos conhecidos (int, boolean, String, List, Array, Date, etc) por isso para serializar/deserializar um Client visto que tem uma relação com a classe BiometricData é necessário indicar como essa serialização/deserialização deve ser feita, sendo a mesma através de classes auxiliares como a **BiometricDataSerializer** que se comporta como um adaptador do tipo BiometricData.

6.2.3 Hibernate

Tanto as classes User (mais á frente explicada), Client como BiometricData foram identificadas como **ORM Persistable** uma vez que se pretende persistir a informação das mesmas. Para tal utilizou-se a framework [Hibernate](#) integrada como plugin (built-in) na ferramenta [Visual Paradigm](#) de modo a automatizar a criação das classes [DAO](#) que são responsáveis por realizar operações CRUD (entre outras interrogações) sobre os objectos persistidos. Para tal a framework hiberante fornece **mapeamento automático (ou de fácil configuração) das classes (neste caso Java) para tabelas numa base de dados relacional** (neste caso MySQL) através da sinalização no diagrama de classes ou anotações no código gerado. Ainda fornece **abstracção da tecnologia de conexão à base de dados, inclusive gestão de sessões**, podendo, por isso, ser configurada para diferentes base de dados, **sem alteração do código**. No entanto tem a desvantagem de existir um acréscimo computacional no uso da framework, ainda assim analisando as vantagens oferecidas compensa a utilização da mesma.

6.2.4 EJB

Foi criada também a classe **HRClientFacadeBean** que implementa uma **Session Bean Stateless** onde as operações a realizar são definidas. A utilização dos EJB fornece a gestão automática de **sincronização das interacções entre objectos em ambiente concorrente**, gestão da **partilha de recursos/**

objectos entre componentes ou intervenientes no sistema, por exemplo na partilha das sessões na conexão à base de dados, utiliza a **JNDI** que é uma API de lookup que permite a **procura de recursos (procura da referência do recurso)** através de um nome previamente atribuído a esse mesmo recurso, fornece também **interoperabilidade**, por exemplo através de protocolos como o **WSDL**, apesar do grupo não ter utilizado neste projecto, uma vez que foi utilizado o protocolo **REST** para comunicação entre serviços que será detalhado mais à frente, a utilização de EJBs disponibiliza outras funcionalidades não referidas. Em suma, o grupo entendeu que a utilização de EJBs é vantajosa uma vez se adequa ao contexto de utilização do sistema pois existem:

- muitos utilizadores, isto é, muitas instâncias de aplicações cliente;
- milhares de objectos criados e em utilização;
- muitas interacções entre os objectos;
- concorrência e operações com requisitos transaccionais.

6.2.5 Multi-camada e Facade

De forma a implementar o **padrão arquitectural** denominado por **multi camada** foi implementado o design pattern **Facade** que expõe a **API** do serviço contido nesse package de modo a que **as classes contidas nos packages exteriores apenas comunicuem com as classes internas a este package através de um único ponto**, neste caso a classe **HRClientFacade**, reduzindo assim o número de dependências entre os packages. Para se entender a vantagem deste design pattern, segue-se um exemplo: caso o package business apenas comunique com o package data através do DataFacade, se a implementação interna do package data muda-se drasticamente, **o package business não sofria qualquer alteração** desde que a classe DataFacade continuasse a existir e mantivesse os protótipos dos seus métodos inalterados, isto é, a sua API inalterada.

6.2.6 Singleton

Ainda neste package, visto que ambas as classes ClientFactory e HRClientFacade devem ter um única instância em todo o sistema foi implementado o design pattern **Singleton**.

6.2.7 REST vs WSDL

De forma a tornar a funcionalidade fornecida por este package **facilmente acessível por qualquer outro serviço ou sistema** foi implementado um servidor utilizando Java Servlets contidos na API Java EE, na qual a classe que implementa o Servlet, neste caso denominada por **HRClientController**, responde a pedidos recebidos através do protocolo **HTTP**. O grupo decidiu utilizar o protocolo REST ao invés de WSDL para a comunicação entre serviços uma vez que a complexidade de cada serviço é relativamente reduzida (consequência directa da divisão do sistema em serviços menos complexos), outra razão é o facto de existir total conhecimento no contexto de utilização e conteúdo transmitido nos pedidos HTTP, por último visto que na componente frontend, algumas funcionalidades são acedidas por **AJAX**, a utilização de REST torna-se mais adequada.

6.2.8 Token Authentication

Observando o diagrama de classes deste package existe uma classe ainda não abordada, a classe **User**. Esta classe foi criada unicamente para implementar um método de autenticação do utilizador, quer seja um

cliente ou personal trainer, na evocação externa de qualquer método. Desta forma no momento de **registo** ou **login** de qualquer utilizador no sistema é gerado um **token único e aleatório** armazenado no sistema numa instância da classe User e retornado ao utilizador para que seja também armazenado temporariamente na aplicação client-side, assim numa futura evocação de qualquer método o token é verificado e caso coincida é permitida a interacção do utilizador.

6.3 Personal Trainer Service

O package HRPersonalTrainer encontra-se especificado nas seguintes figuras.

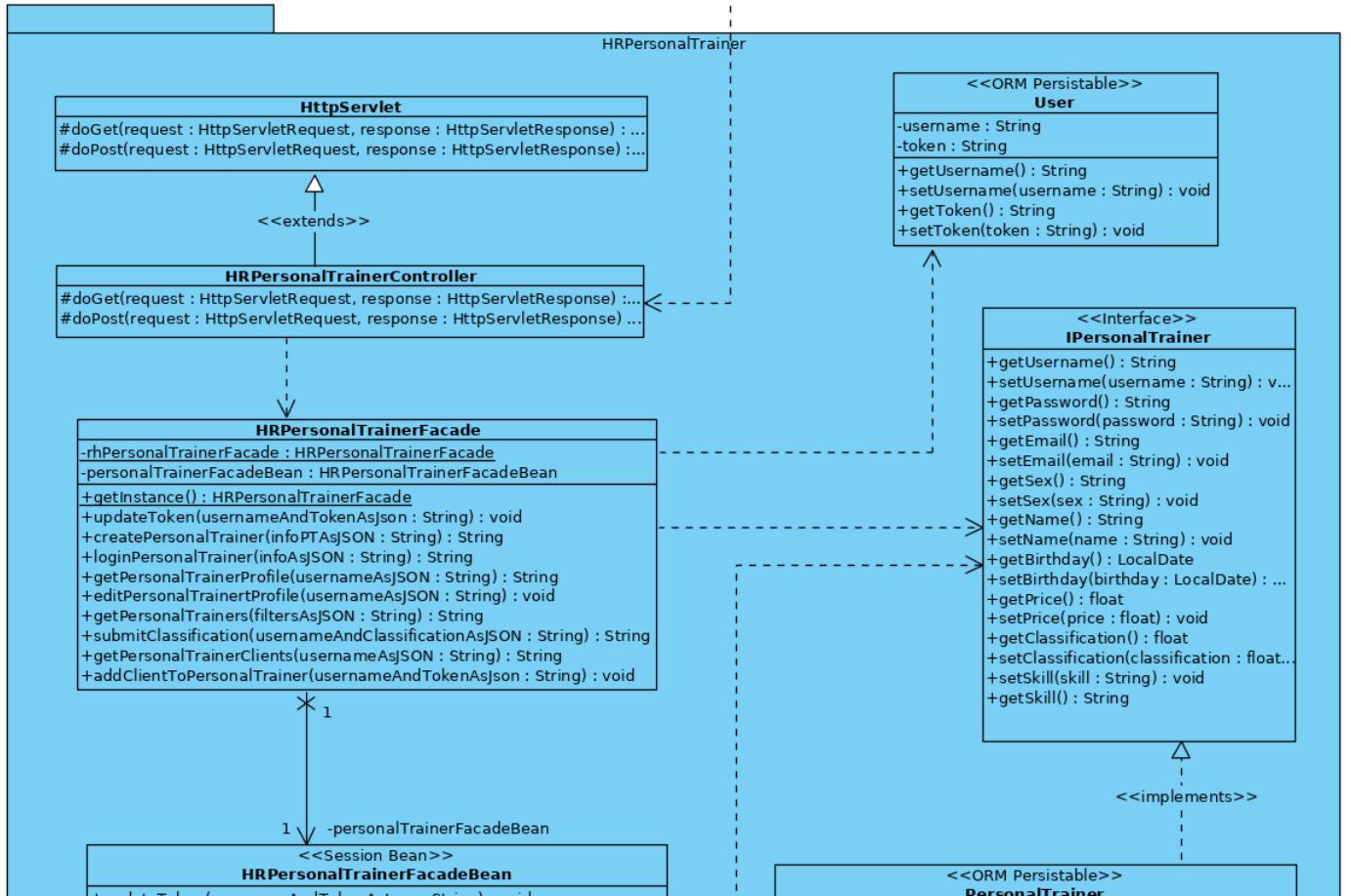


Figura 6.3: Diagrama de classes do package HRPersonalTrainer - parte 1.

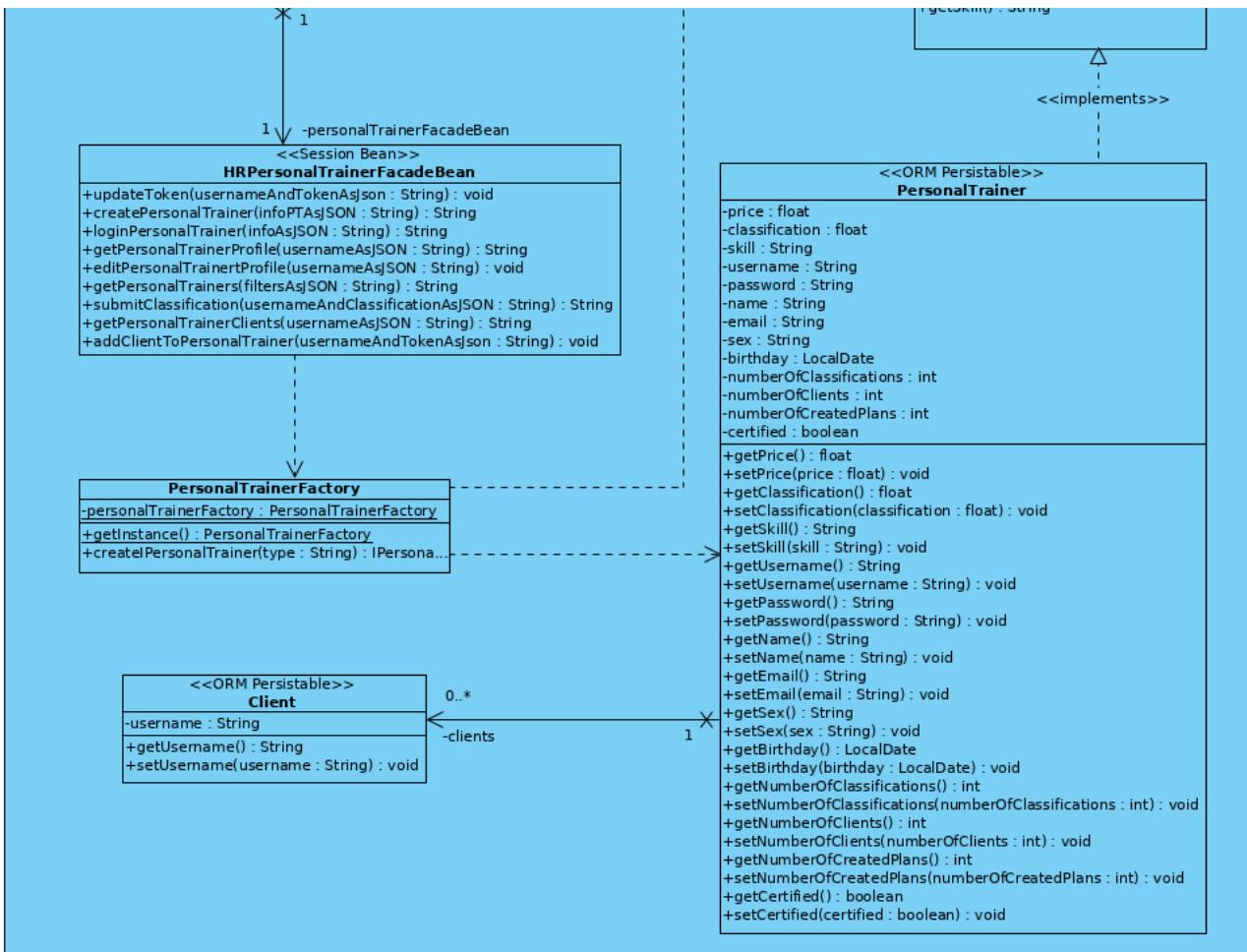


Figura 6.4: Diagrama de classes do package HRPersonalTrainer - parte 2.

Analogamente ao Client Service e pelos mesmos motivos no package HRPersonalTrainer foram implementados os design pattern: **FactoryMethod**, **Adapter**, **Singleton** e **Facade**. A lógica de negócios foi, também, implementada num **Session Bean** e o serviço exposto através do protocolo **REST** utilizando **autenticação por tokens**.

No entanto foi necessária a associação entre um cliente e um personal trainer para isso foi introduzida a classe **Client** também esta persistida.

6.4 Core

O package Core encontra-se especificado nas seguintes figuras.

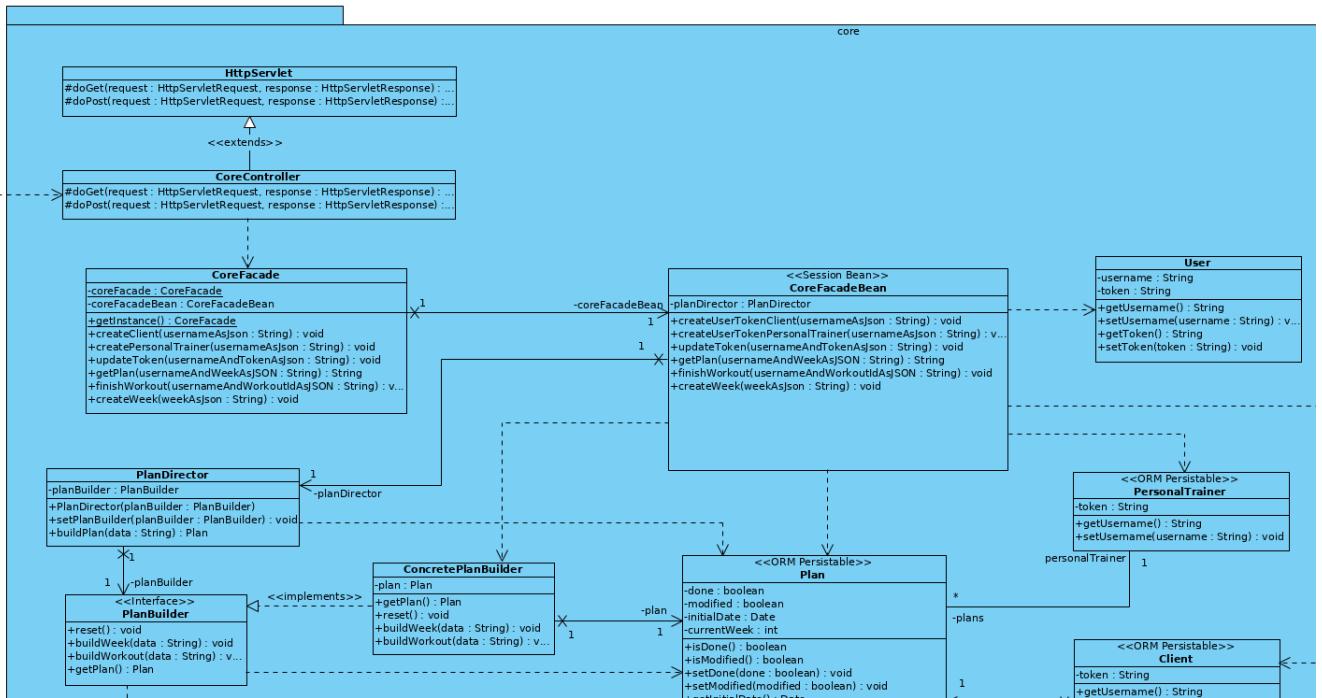


Figura 6.5: Diagrama de classes do package Core - parte 1.

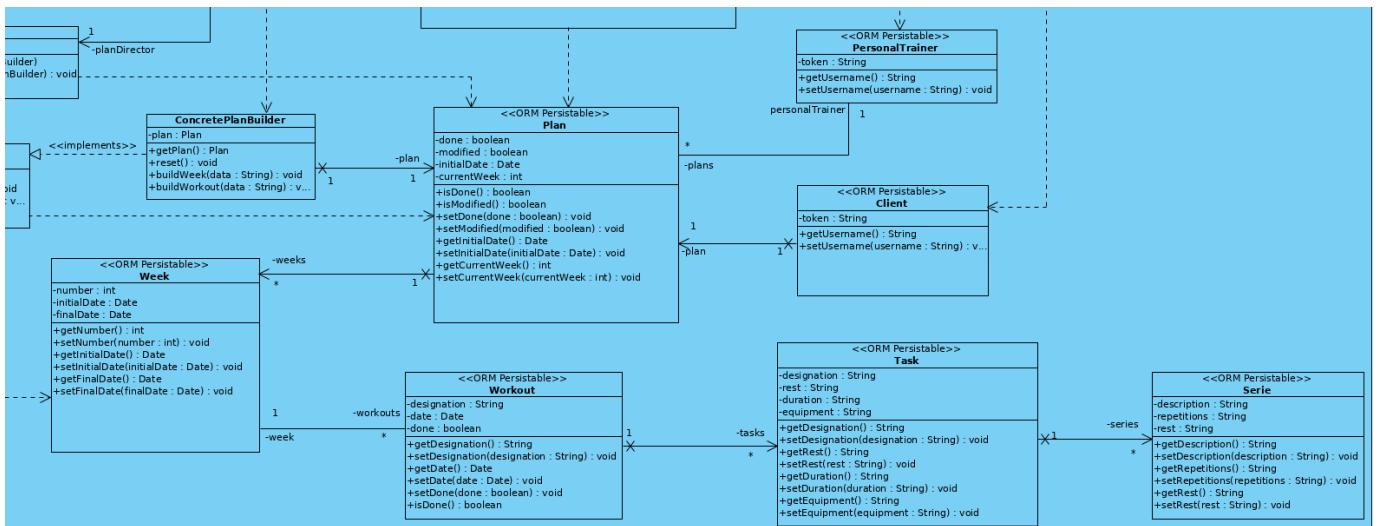


Figura 6.6: Diagrama de classes do package Core - parte 2.

O Core contém as entidades principais do modelo de negócio, no contexto deste projeto, contém as classes: **Plan**, **Week**, **Workout**, **Task** e **Serie**; já descritas anteriormente no modelo de domínio.

Analogamente aos serviços anteriores e pelos mesmos motivos no package Core foram implementados os design pattern: **Adapter**, **Singleton** e **Facade**. A lógica de negócios foi, também, implementada num **Session Bean** e o serviço exposto através do protocolo **REST** utilizando **autenticação por tokens**.

Note-se na relação entre certas classes foi vantajoso utilizar a dupla naveabilidade uma vez que o dupla direcção no acesso entre ambos os objectos é útil.

6.4.1 Método finishWorkout

O método **finishWorkout()** responsável por finalizar um determinado Workout tem elevada relevância no sistema. Não só realiza a operação evidente de Update (neste caso actualização do estado do workout para realizado), mas também tem uma funcionalidade adicional tal como, avançar automaticamente a semana do plano em que o cliente se encontra caso todos os workouts dessa semana tenham sido realizados. Para clarificar segue-se um exemplo, caso por exemplo o Plano de treino do Cliente se encontre na semana 3 (semana actual), mas se este realizar todos os workouts da semana 5 a sua semana actual passa a ser a 6, este sistema de actualização da semana actual foi baseado em outras aplicações, tal como a Netflix, em que se um utilizador estiver na temporada 3, mas depois for ver um episódio da temporada 5, quando voltar a entrar no site para voltar a ver a série será remetido automaticamente para a temporada 5.

6.4.2 Builder

No Core foi implementado o Design Pattern Builder, este é um Pattern criacional, no qual os principais objectivos é reduzir o número de dependências, pois todos as classes apenas ficam dependentes do PlanDirector, sendo este o único a ter as dependências necessárias. Outro objectivo é a criação do objecto por passos.

Tal como nos serviços Client e PT o este design pattern acabou por não ser implementado em código porque a biblioteca GSON consegue ela própria criar o objecto todo através de um json e das classes de Deserialização criadas.

6.5 Requests Service

Analogamente ao serviços anteriores e pelos mesmos motivos no package Requests foram implementados os design pattern: **Singleton** e **Facade**. A lógica de negócios foi, também, implementada num **Session Bean** e o serviço exposto através do protocolo **REST** utilizando **autenticação por tokens**.

O serviço Requests tal como o nome sugere é um serviço que guarda informações sobre pedidos, pedidos estes submetidos por Clientes para os PTs, nos quais consta informação sobre o tipo de Plano que desejam realizar. O Cliente tem acesso a todos os pedidos que realizou, por outro lado o PT só tem acesso aos pedidos que ainda tem pendentes.

Os Pts podem aceitar ou rejeitar pedidos sempre essa informação ilustrada aos Clientes, pois o aceitar ou rejeitar de um pedido altera o seu estado.

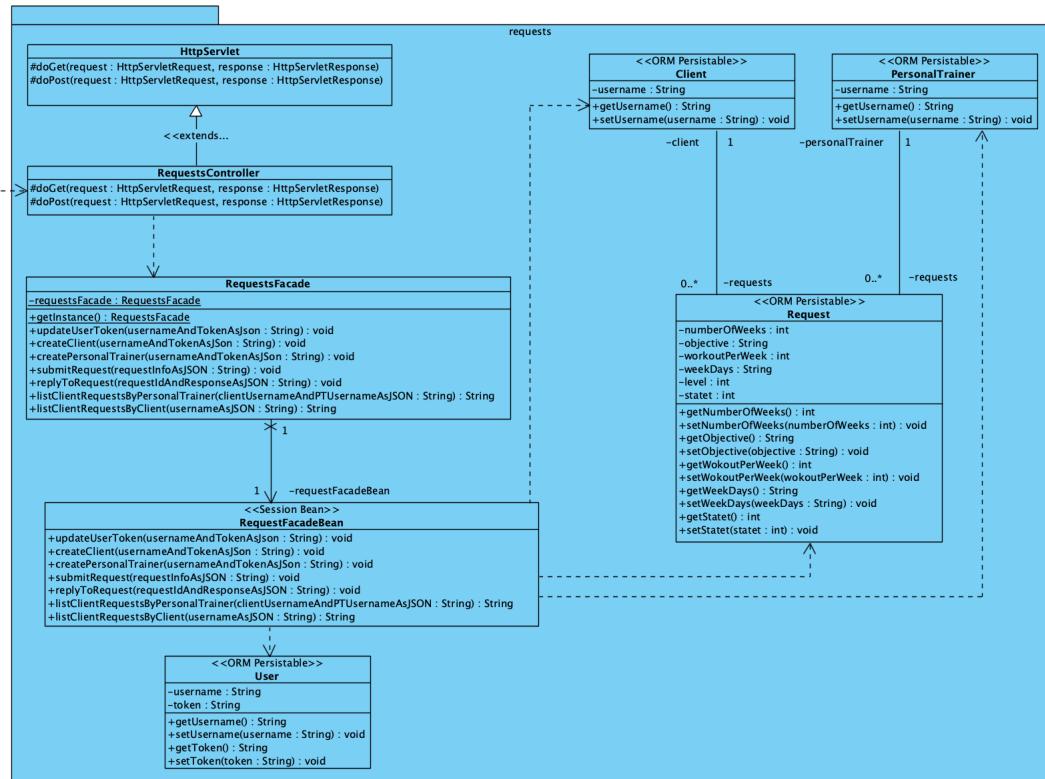


Figura 6.7: Diagrama de classes do package Notification

6.6 Notifications Service

Analogamente ao serviços anteriores e pelos mesmos motivos no package Notifications foram implementados os design pattern: **Singleton** e **Facade**. A lógica de negócios foi, também, implementada num **Session Bean** e o serviço exposto através do protocolo **REST** utilizando **autenticação por tokens**.

O serviço Notifications tal como o nome sugere é um serviço que guarda as notificações geradas pelo sistema para os utilizadores. O sistema apenas gera notificações consoante acções de utilizadores, estes quando realizam uma acção relacionada com outro o sistema irá criar uma notificação para o outro utilizador, de forma a que esta possa ser informado de alterações que existiram no sistema.

Tanto os Clientes como os PTs podem marcar as notificações como lidas, desta forma será mais fácil numa próxima vista das notificações distinguir daquelas que estão por ler e as que já estão lidas.

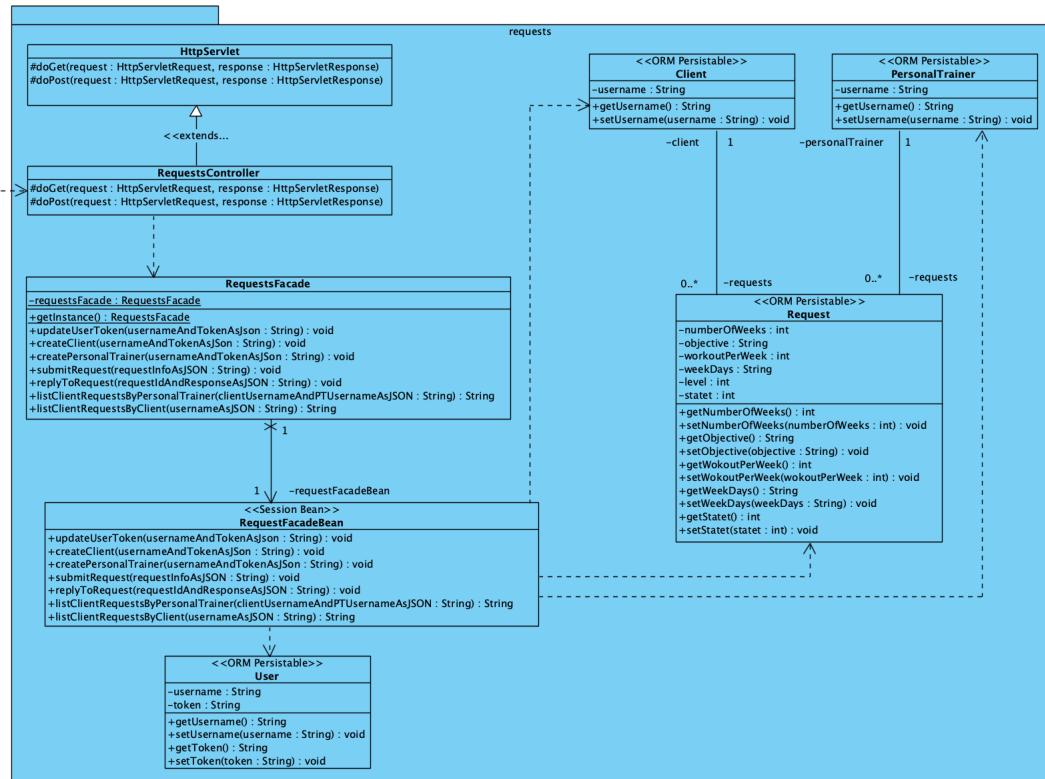


Figura 6.8: Diagrama de classes do package Request

6.7 GymAtHome Service

6.7.1 Propagação dos tokens

Tal como referido anteriormente todos os serviços contêm um método de autenticação da interacção do utilizador através de um token gerado no login ou registo. Desta forma para que todos os serviços tenham todos os tokens mais actuais de cada cliente, este serviço principal (GymAtHome) tem a responsabilidade de propagar os tokens para os respectivos serviços externos, para isso foi definido o método updateToken nos mesmos.

Este serviço tem ainda o objectivo de criar notificações para os utilizadores, à medida que os utilizadores realizam certas acções podem ser geradas algumas notificações.

A principal função deste serviço é servir de dispatcher, pois todos os pedidos enviados pelo **Frontend** são direcionados a este e este é que trata de encaminhar para o serviço correcto e de apenas reencaminhar a resposta do serviço que comunicou. Com isto os outros serviços estão mais protegidos uma vez que o único ponto de acesso destes é através deste serviço.

6.8 Frontend service

6.8.1 MVC - Model View Controller

Para a comunicação entre os **Controllers** e respectivas páginas de apresentação (**Views**) foi implementado o padrão arquitectural **MVC**. O acesso ao **Model** (que contem a lógica de negócio), por parte

do Controller é realizado através do protocolo REST. A utilização deste padrão arquitectural, novamente, torna a evolução e manutenção do código facilitada uma vez que, por exemplo, caso o pacakage presentation (que contém os Controllers e respectivas Views) seja alterado, por exemplo, ao implementar uma nova UI para o sistema operativo Android ou IOS, visto que toda a lógica de negócio está unicamente contida no Model, o sistema (na componente backend) continuará totalmente funcional e inalterado.

Capítulo 7

Usabilidade

Nesta secção discute-se a importância da usabilidade no desenvolvimento deste projecto, os princípios que se podem encontrar, e em que medida os mesmos trazem melhorias para a interacção com o sistema.

7.1 Conhecer os utilizadores

Um dos grandes sábios de sistemas interactivos, Donald A. Norman disse a seguinte afirmação, "We must design for the way people behave, not for how we would wish them to behave.", isto significa que, para desenvolver um bom sistema interactivo deve-se seguir a ideologia de que **o programa é feito para o utilizador**. Desta forma, deve-se desenvolver os programas a pensar nas necessidades do utilizador e como o mesmo pretende **usar** (usabilidade) o sistema de forma eficiente, eficaz e satisfatória, para atingir os seus objectivos. De forma muito resumida, deve-se desenvolver/desenhar focado na usabilidade. Por este motivo, neste projecto aplicou-se a mesma ideologia, ou seja, necessita-se de conhecer os utilizadores do programa.

O conhecimento/análise dos utilizadores contém diversas técnicas conhecidas tais como: entrevista, observação, personas, etc. No entanto, face aos recursos que a equipa tem, a forma mais eficaz de conhecer o utilizador foi através de pesquisa e estudo do domínio do problema (exercício físico/ ginásio). Na verdade, começou-se por perceber a linguagem utilizada no meio, isto é, termos como: planos de treino, workouts, series, tempos entre series, personal trainer, entre outros. Apesar de poder parecer pouco importante, isto permitiu desenvolver uma interface com uma linguagem adequada aos utilizadores. Do mesmo modo, a utilização de interfaces simplistas, como se verá mais adiante também foi pensada no utilizador, pois pessoas focadas no exercício tendem a ser simplistas. Outra razão deve-se ao facto de a tendência actual de desenvolvimento web seguir a simplicidade das interfaces (seguir as **tendências/padrão** torna-se muito importante para se ter uma interface **consistente** com a actualidade).

7.2 Princípios de usabilidade

Poder-se-á dizer que o desenvolvimento de interfaces torna-se uma processo consistente, desta forma, foram definidos/partilhados, princípios de usabilidade. Pode-se dizer que os princípios de usabilidade são regras/normas que guiam os desenvolvedores/designers a desenharem boas interfaces. Desta forma, a equipa desenhou a interface do projecto focada na usabilidade e nos princípios.

De forma sucinta, apresentar-se-á de seguida alguns dos princípios de usabilidade mais importantes presentes neste projecto, visto que os na secção 8 serão abordados com mais detalhe para cada **view**.

Um dos princípios presentes no projecto, e dos mais importantes no desenho de interfaces, como já foi dito anteriormente consiste na **Consistência**. A consistência a nível de linguagem, componentes de interface

ao longo das várias views, em relação á actualidade onde o desenho segue a tendência/consistência actual (simplicidade) entre outros casos.

A **Familiarização** um pouco semelhante ao anterior, consiste na utilização de padrões, por exemplo, a apresentação de uma semana do plano segue o estilo **comum** de um calendário/horário semanal, a inserção de datas faz-se com um calendário, etc.

Outro princípio bastante comum ao longo do projecto consiste na **Predictability**, isto é, quando se sabe que algo não pode ser feito, **impede-se** de fazer, por exemplo o botão "Guardar Semana" de uma plano, está desabilitado quando não existem workouts definidos para a mesma, pois não será possível guardar, tornando-se, desta forma, a interface mais eficiente/eficaz.

A **Synthesizability** consiste em promover ao utilizar o conhecimento do estado do sistema, ou seja, informar o sucesso ou insucesso de acções feitas pelo mesmo. Neste projecto, pode-se encontrar este princípio em todas as views, com a informação de erros, conclusão ou execução de ações (sucessos), também na criação plano, quando são adicionados workouts à semana, o personal trainer consegue perceber essa adição com a view que foi feita, tal como se verá mais adiante.

A nível de flexibilidade, está presente o princípio **Observability**, mas mais concretamente a característica de **Browsability**, para a navegação entre semanas no plano, permitindo ao cliente sentir-se ter controlo do sistema.

Mais adiante, como já foi dito anteriormente, na secção 8 (secção seguinte) , os princípios serão avaliados para cada view com mais detalhe.

Capítulo 8

Princípios de Usabilidade presentes e avaliação por Heurísticas de Normam das Interfaces

Neste capítulo serão ilustrados as mockups desenvolvidas para o projecto, bem como o resultado final/actual na aplicação web. Primeiro serão ilustradas as interfaces comuns, depois as interfaces dos Clientes e por último dos PTs. De seguida, serão também ilustrados alguns pormenores incorporados para uma melhor experiência dos utilizadores.

8.1 Princípios/heurísticas genéricos(as) a todas as views

Inicialmente, apresentam-se os princípios/heurísticas genéricos/comuns a toda a interface, aqueles que ocorrem em praticamente todas as views.

Princípios de usabilidade

- **Predictability:** de uma forma geral, todas as views cumprem este princípio, onde o utilizador comprehende o efeito das acções no sistema.
- **Synthesizability:** de uma forma geral, todas as views, informam ao utilizador o estado a aplicação após acções do mesmo através de mensagens de erro, aviso e sucesso, que serão explicadas adiante.
- **Consistency:** de forma geral, a interface segue este princípio em todas as suas views, bem como em relação a todo sistema.
- **Customizability:** tamanho da janela adapta-se em computadores e tablets, no entanto, em smartphones seria necessário mais desenvolvimento, pois certas páginas não se adaptam correctamente.

Heurísticas de Normam

- **Aesthetic and minimalist design:** tal como já foi referido anteriormente, toda a interface foi desenhada de forma simplista.
- **Visibility of system status:** em todas as interfaces existe uma nav bar que ajuda o utilizador a perceber onde se encontra.

8.2 Interfaces Comuns

Nesta secção apresenta-se as interfaces que são comuns aos dois tipos de utilizadores do sistema, isto é, cliente e PT.

8.2.1 Mensagens de sucesso, erro e aviso

Visto que todas as views contém mensagens com a informação de erro, sucesso ou aviso sobre as acções do utilizador, acha-se importante começar a falar sobre as mesmas, pois aparecem em todas as views.

Descrição da mensagem de sucesso

O alerta de Sucesso é utilizado para informar os utilizadores quando as suas acções são realizadas com sucesso mas não é óbvio para o utilizador que estas foram realizadas com sucesso.

Sucesso: O seu perfil foi criado com sucesso! Aqui pode ver o seu perfil onde pode alterar os seus dados a qualquer momento. 

Figura 8.1: Alerta de Sucesso.

Descrição da mensagem de aviso

O alerta de Aviso é utilizado quando ocorre algum problema, no entanto não impede a continuidade da acção.

Aviso: Neste momento não foi possível carregar os dados biométricos. Tente recarregar a página. 

Figura 8.2: Alerta de Aviso.

Descrição da mensagem de erro

O alerta de Erro é utilizado quando o utilizador introduz dados inválidos ou quando existem erros internos dos servidores.

Erro: Credenciais inválidas. 

Figura 8.3: Alerta de Erro.

Princípios de usabilidade

- **Synthesizability:** as mensagens de erro, sucesso ou aviso informam o utilizador do estado do sistema. Desta forma, visto que as mensagens ocorrem em todas as páginas, pode-se afirmar que este princípio também.

- **Generability:** visto que é genérico a todas as views, ou seja, igual em todas as views, mudando apenas o conteúdo da mensagem.
- **Consistency:** tal com foi referido, todas as views implementam as mensagens de erros, sucesso e avisos, logo, torna-se consistente.

Heurísticas de Normam

- **Visibility of system status:** as mensagens mantém o utilizador informado sobre o que está a acontecer, ou o estado da aplicação.
- **Consistency and standards e Aesthetic and minimalist design:** o design usado para as mensagens de erro são **minimalista** e **comum/padrão**, tal como se pode ver nas imagens 8.1, 8.2 e 8.3.
- **Help users recognize and recover from errors:** as mensagens de erro são bastante informativas na medida do necessário em informar o que aconteceu, bem como dar uma possível solução para o problema ou alternativa.

8.2.2 Login

Descrição

De forma a facilitar a autenticação para o utilizador o login é comum aos dois tipos de utilizadores da plataforma, sendo que estes apenas precisam de introduzir as suas credenciais. Podem ainda seleccionar "Registar Cliente" ou "Registar Personal Trainer" caso sejam novos no sistema.

Figura 8.4: Mockup Login.

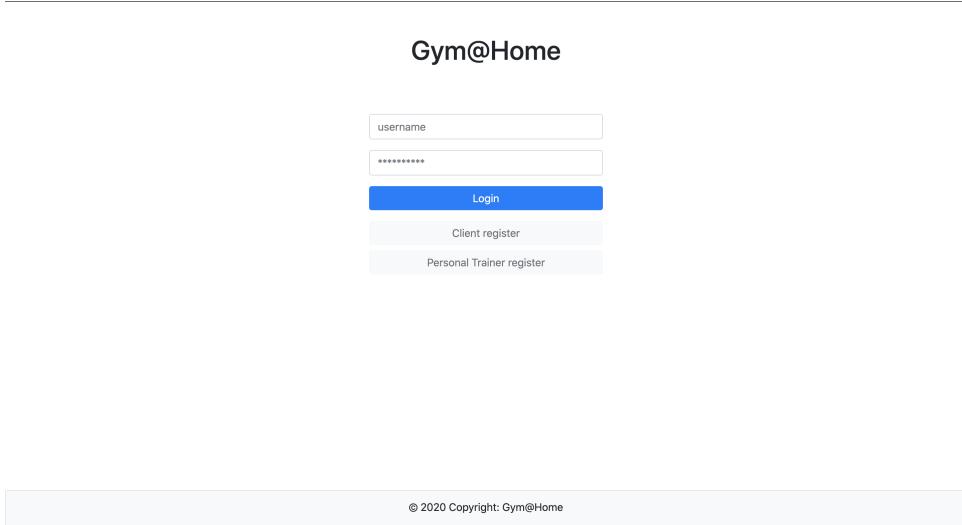


Figura 8.5: Interface Login.

Princípios de usabilidade

- **Predictability:** o utilizador percebe o efeito das acções, neste caso, fazer o login ou registar-se.
- **Familiarity:** seguiu-se o padrão de desenho de logins presente normalmente em maior parte das aplicações web.
- **Consistency:** bastante semelhante ao anterior, na medida em que o princípio da consistência torna-se mais abrangente.

Heurísticas de Normam

- **Visibility of system status:** o resultado do login é confirmado com um mensagem de erro ou sucesso dependendo da situação, informando do que está a acontecer.

8.2.3 Notificações

Descrição

Com esta mockup tanto o cliente, como o PT consegue saber todas as acções que aconteceram relacionadas com o mesmo. Aqui pode consultar as notificações, isto é, a data, o estado e a descrição bem como marcar como lidas.

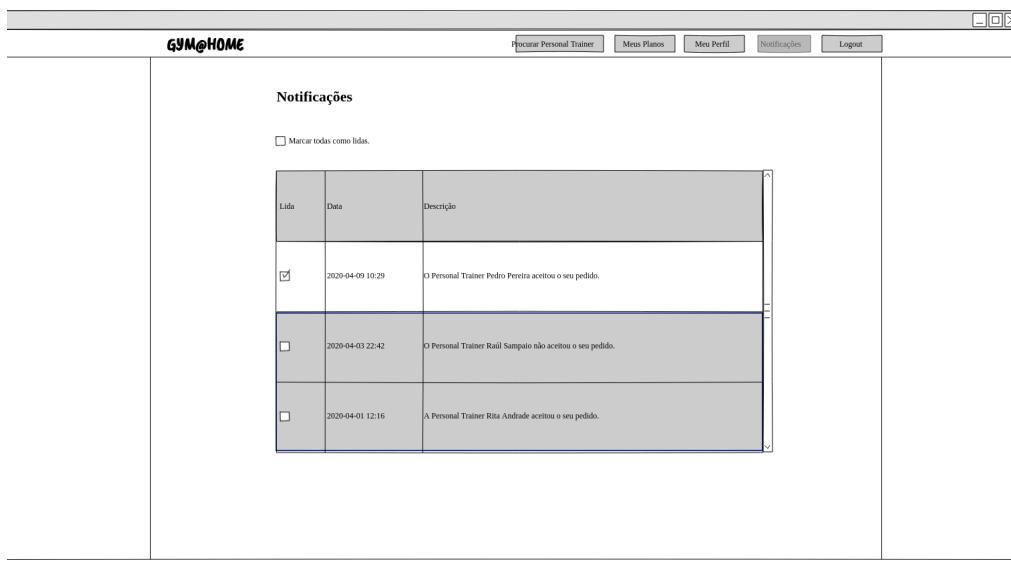


Figura 8.6: Mockup Notificações Cliente.

The actual client-side interface for 'Gym@Home'. It features a header with the 'Gym@Home' logo and navigation links for 'Procurar Personal Trainer', 'Meus Pedidos', 'Meu Plano', 'Meu Perfil', 'Notificações', and 'Logout'. Below the header is a blue button labeled 'Marcar como lidas'. The main content area displays a table with four columns: 'Data', 'Estado', and 'Descrição'. The first row is highlighted in grey and shows '19-07-2020' and 'lida' with the description 'O seu Personal Trainer adicionou uma semana ao seu plano.'. The second row shows '19-07-2020' and 'não lida' with the description 'Personal Trainer com o username ptJoao aceitou o teu pedido.' A copyright notice at the bottom reads '© 2020 Copyright: Gym@Home'.

Figura 8.7: Interface Notificações Cliente.

Princípios de usabilidade

- **Predictability:** quando se selecciona a checkbox para seleccionar todas as notificações para marcar como lidas, apenas selecciona as não lidas, para evitar que se envie o pedido de marcar como lida para o servidor desnecessariamente para notificações que já estão lidas
- **Synthesizability:** após marcar como lidas, as notificações aparecem com uma cor de background diferente, permitindo ao utilizador perceber o efeito da sua acção.
- **Familiarity:** As notificações por ler tem uma cor diferente das notificações lidas, um padrão **comum**.

Heurísticas de Normam

- **Visibility of System Status:** os utilizadores obtém informações sobre acções que acontecerem no sistema relacionadas com os mesmos, mas sem ter sido este a alterar o estado.
- **Error Prevention:** quando selecciona o marcar todas como lidas, apenas selecciona as não lidas, para que não sejam mandados pedidos para o servidor desnecessariamente, tal como já tinha sido dito no princípio de **Predictability**.

8.3 Interfaces Clientes

8.3.1 Registar Cliente

Descrição

De forma a que o Cliente se registe na aplicação é necessário este inserir alguns dados obrigatórios, tais como o nome, username, email, etc..., bem como dados opcionais relacionados com dados biométricos. No entanto, importante referir que dos dados obrigatórios excepcionalmente a altura e peso são dados obrigatórios.

A interface de usuário para registrar um cliente em GYM@HOME é dividida em duas secções principais: "Dados Obrigatórios" e "Dados Opcionais".

Dados Obrigatórios:

- Nome: Ex: João Costa
- Username: Ex: cliente-1
- Email: Ex: cliente-1@gym.pt
- Password: Senha oculta
- Repetir Password: Senha oculta
- Género: Masculino
- Data de Nascimento: 14-04-2020
- Altura (cm): Ex: 173
- Peso (Kg): Ex: 60

Dados Opcionais:

- Cintura (cm): Ex: 78
- Peito (cm): Ex: 87
- Gêmeo (cm): Ex: 36
- Quadricep (cm): Ex: 51
- Tricep (cm): Ex: 28
- Pulso (cm): Ex: 15

No lado direito da interface, há uma barra com ícones para maximizar, minimizar e fechar a janela. No topo, ao lado do logo "GYM@HOME", há um botão "Login". No lado esquerdo, uma barra vertical com ícones de usuário, configuração e ajuda. No lado direito, uma barra com ícones de busca, filtros e outras opções. No lado inferior, uma barra com links para "Home", "Sobre", "Contactos" e "Ajuda".

Figura 8.8: Mockup Registar Cliente.

Figura 8.9: Interface Registrar Cliente.

Princípios de Usabilidade

- **Familiarity:** Colocar a "Data de Nascimento" é feito através de um calendário, pelo que o Cliente como ser humano está mais familiarizado a utilizar.

Heurísticas de Normam

- **Error prevention:** Todos os inputs numéricos aceitam apenas números, o "Género" apenas permite os valores do dropdown, a "Data de Nascimento" apenas permite datas seleccionadas pelo calendário e o "Email" apenas aceita um email (tem de conter o @). **Flexibility and efficiency of use:** visto que após ser feito o registo, faz-se o login automático, sendo mais eficiente para o cliente.

8.3.2 Cliente consulta/edita o próprio perfil

Descrição

O mockup PERfil DO CLIENTE, permite ao Cliente visualizar, bem como, editar o seu próprio perfil para manter as suas informações mais actualizadas possíveis. De forma a ficar mais consistente com o Registo de um Cliente na interface foram mantidas as duas colunas de dados, sendo que no topo da página ficam os dados pessoais, de seguida os dados biométricos e a barra sobre o estado do IMC segundo os valores de referência da OMS.

A submissão das alterações efectuadas ao perfil, basta alterar os dados que ele necessitar e carregar no botão "Salvar Alterações", este botão de forma a ficar mais visível na interface final foi colocado no topo da página em vez de estar no fundo da mesma, pois em dispositivos mais pequenos o botão poderia ficar escondido.

GYM@HOME

Username

Nome: João Costa
Email: exemplo@email.com
Género: Masculino

Dados biométricos:

Dados obrigatórios:

Altura (cm): 175
Peso (Kg): 62

Dados opcionais:

Cintura (cm): 80
Peito (cm): 120
Gêmeo (cm): 35
Quadríceps (cm): 50
Tríceps (cm): 30
P脉so (cm): 18

Salvar alterações

Copyright gym@home

Figura 8.10: Mockup Perfil Cliente visto pelo próprio.

Gym@Home

@cJoao

Dados Pessoais

Nome: João Email: admin@tupi.pi
Género: Masculino Nova Password: *****
Data de Nascimento: 17/07/1998 Confimar Password: *****

Dados Biométricos

Peso (kg): 62,0	Cintura (cm): Ex: 78
Altura (cm): 173	Peito (cm): Ex: 87
Quadríceps (cm): Ex: 51	Gêmeo (cm): Ex: 36
Tríceps (cm): Ex: 28	P脉so (cm): Ex: 15

IMC: 20.715693

subnutrido saudável sobre peso obesidade grau I obesidade grau II (severa) obesidade grau III (mórbida)

18.5 25 30 35 40

© 2020 Copyright: Gym@Home

Figura 8.11: Interface Perfil Cliente visto pelo próprio.

Princípios de usabilidade

- Substitutivity:** Os inputs da interface onde o Cliente pode alterar os valores são também outputs. Sempre que o Cliente alterar dados estes são guardados na BD e novamente carregados para os outputs que o Cliente tinha utilizado como input.
- Observability:** O botão "Salvar Alterações" está no topo da página para o Cliente o conseguir observar mal carregue a página.
- Familiarity:** Colocar a "Data de Nascimento" é feito através de um calendário, pelo que o Cliente como ser humano está mais familiarizado a utilizar.

Heurísticas de Normam

- **Error prevention:** Todos os inputs numéricos aceitam apenas números, o "Género" apenas permite os valores do dropdown, a "Data de Nascimento" apenas permite datas seleccionadas pelo calendário e o "Email" apenas aceita um email (tem de conter o @).
- **Flexibility and efficiency of use:** tal como diz respeito ao princípio **Substitutivity**, usar o mesmo componente quer para visualizar, quer para editar o perfil, torna a interface mais eficiente para o PT.

8.3.3 Cliente consultar o perfil do PT

Descrição

De forma a que o Cliente tenha uma visão mais detalhada do PT antes de realizar um pedido de Plano são mostradas algumas estatísticas do mesmo. De forma a que o Cliente disponha da informação do PT sem mudar de página este é ilustrado num pop-up, desta forma os filtros aplicados no refinamento dos PTs a observar são mantidos porque não é feito reload à página de "Procurar Personal Trainer".

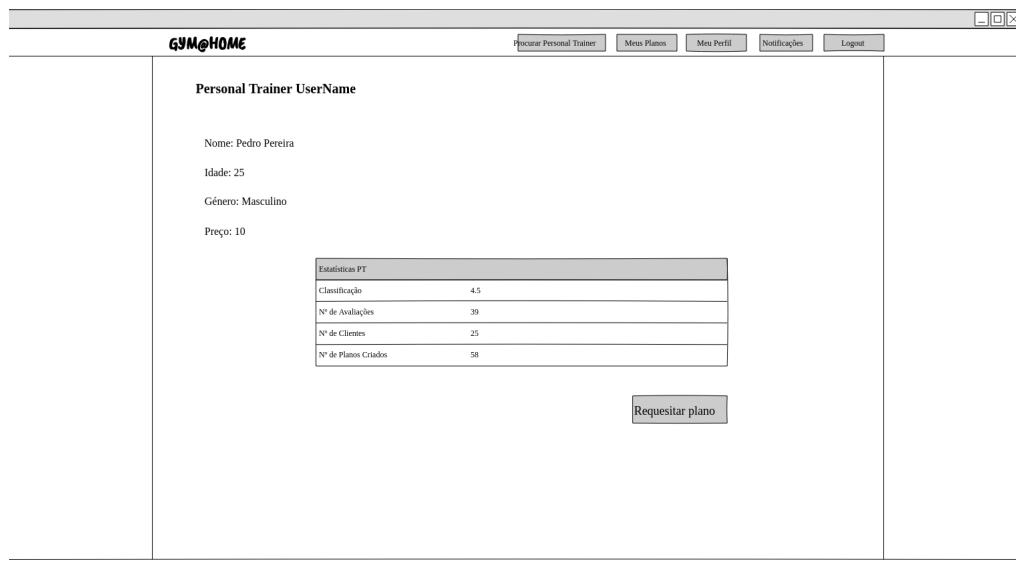


Figura 8.12: Mockup Perfil PT visto pelo Cliente.

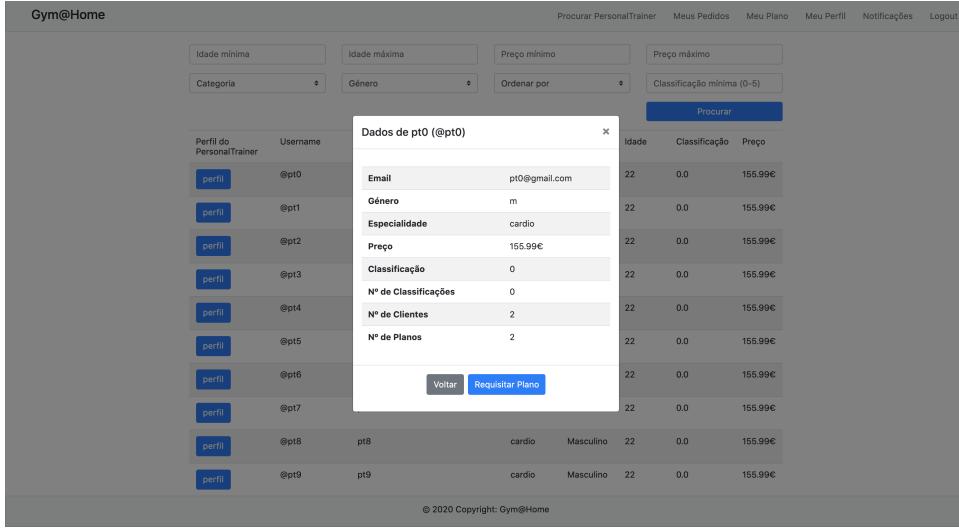


Figura 8.13: Interface Perfil PT visto pelo Cliente.

Princípios de usabilidade

- **Generability e Consistency:** na medida em que a estrutura de apresentação deste pop-up com a informação do perfil torna-se semelhante para ambos os utilizadores quando desejam consultar a informação da entidade oposta.

Heurísticas de Normam

- **Recognition rather than recall:** o cliente não necessita de lembrar-se/memorizar dos dados de perfil do PT, podendo consultá-los no momento em que precisa dos mesmos de forma eficiente.
- **Flexibility and efficiency of use:** o botão "perfil", que mostra as informações do PT torna esse processo de consulta mais eficiente, como uma espécie de atalho.

8.3.4 Procurar Personal Trainer

Descrição

Nesta mockup são ilustrados todos os PTs disponíveis, aqui o Cliente pode aplicar filtros para uma pesquisa mais refinada por PTs que possam ser do seu interesse. Devido ao tamanho dos filtros e respectivo conteúdo estes não cabiam numa única linha tal como na mockup pelo que na interface final estes aparecem em duas linhas, ficando assim mais legíveis para o Cliente.

Tal como foi mostrado no mockup anterior, cada linha, que corresponde a um PT contém um botão denominado "perfil" que apresenta um pop-up com as informações do perfil do PT, onde se encontram algumas informações adicionais.

Gym@Home

Procurar Personal Trainer | Meus Planos | Meu Perfil | Notificações | Logout

Id	Nome	Categoria	Género	idade	classificação	Preço
1	Pedro Pereira	Musculação	Masculino	25	4,5	10
2	Filipa Marques	Cardio	Feminina	30	4,8	10
3	Joana Martins	Musculação	Feminina	27	4,3	10
4	Luis Afonso	Cardio	Masculino	21	4,0	10
5	Pedro Fonseca	Musculação	Masculino	45	4,9	15

Copyright gym@home

Figura 8.14: Mockup Procurar Personal Trainer.

Gym@Home

Procurar Personal Trainer | Meus Pedidos | Meu Plano | Meu Perfil | Notificações | Logout

Perfil do Personal Trainer	Username	Nome	Categoria	Género	idade	Classificação	Preço	Requisitar Plano
Perfil	@pt0	pt0	cardio	Masculino	22	0.0	155.99€	<button>Requisitar Plano</button>
Perfil	@pt1	pt1	cardio	Masculino	22	0.0	155.99€	<button>Requisitar Plano</button>
Perfil	@pt2	pt2	cardio	Masculino	22	0.0	155.99€	<button>Requisitar Plano</button>
Perfil	@pt3	pt3	cardio	Masculino	22	0.0	155.99€	<button>Requisitar Plano</button>
Perfil	@pt4	pt4	cardio	Masculino	22	0.0	155.99€	<button>Requisitar Plano</button>
Perfil	@pt5	pt5	cardio	Masculino	22	0.0	155.99€	<button>Requisitar Plano</button>
Perfil	@pt6	pt6	cardio	Masculino	22	0.0	155.99€	<button>Requisitar Plano</button>
Perfil	@pt7	pt7	cardio	Masculino	22	0.0	155.99€	<button>Requisitar Plano</button>

© 2020 Copyright: Gym@Home

Figura 8.15: Interface Procurar Personal Trainer.

Princípios de usabilidade

- **Synthesizability:** nesta interface, existem duas situações deste princípio: inicialmente quando são listados todos os PTs, isto é, consegue perceber o efeito da sua acção, por outro lado, quando selecciona um PT e deseja requisitar um plano do mesmo, redirecciona-se para a página para esse efeito, e dessa forma, também consegue perceber o efeito da sua acção.
- **Familiarity:** a estrutura utilizada para a filtragem seguiu o padrão comum, que se pode encontrar em muitas aplicações web.

Heurísticas de Normam

- **Error prevention:** Todos os inputs numéricos aceitam apenas números e o ”Género”, ”Categoria” e ”Ordenar Por” apenas permite os valores dos dropdowns.
- **Recognition rather than recall:** utilização de dropdowns nos filtros permite que o utilizador não tenha que memorizar as opções existentes.
- **Flexibility and efficiency of use:** o botão ”perfil”, que mostra as informações do PT torna esse processo de consulta mais eficiente, como uma espécie de atalho.

8.3.5 Preencher Pedido

Descrição

A requisição dum Plano, necessita do preenchimento de um formulário com alguns dados pelo Cliente. A mockup/protótipo e a interface final foi removida a opção de adicionar os dados biométricos opcionais do Cliente, sendo que todos os dados biométricos são enviados automaticamente ao PT.

The mockup shows a web page titled 'GYM@HOME' with a navigation bar at the top. The main content area is titled 'Definir tipo de plano pretendido'. It contains several input fields and dropdown menus:

- Nº de semanas (disponibilidade):
- Objetivo:
- Nº de treinos semanais:
- Disponibilidade semanal: 2^a 3^a 4^a 5^a 6^a Sab Dom
- Dificuldade do plano:
- Adicionar dados biométricos opcionais: Sim

At the bottom right is a large grey button labeled 'Submeter*'. Below it, a small note states: '*Dados biométricos obtidos são enviados ao Personal Trainer.'

Figura 8.16: Mockup Preencher Pedido.

Gym@Home

Procurar PersonalTrainer Meus Pedidos Meu Plano Meu Perfil Notificações Logout

Definir tipo de plano pretendido

Nº de semanas (disponibilidade):

Objetivo:

Nº de treinos semanais:

Disponibilidade semanal:

2ª 3ª 4ª 5ª 6ª Sab Dom

Dificuldade:

Submeter

© 2020 Copyright: Gym@Home

Figura 8.17: Interface Preencher Pedido.

Princípios de usabilidade

- **Synthesizability:** após a submissão do formulário/pedido de plano, vai ser redirecionado para a página que tem todos os pedidos enviados pelo Cliente, permitindo assim que o mesmo perceba o efeito da sua acção.

Heurísticas de Normam

- **Error prevention:** Todos os inputs estão predefinidos, evitando assim erros por parte do Cliente.
- **Recognition rather than recall:** utilização de dropdowns permite que o utilizador não tenha que memorizar as opções existentes.

8.3.6 Consultar Semana do plano

Descrição

A mockup de visualização da Semana de um plano é comum a ambos os utilizadores, excepto que o Cliente contém um botão para avaliar o PT. Dessa forma, vamos apresentar a do Cliente, no entanto são praticamente semelhantes.

O Cliente consegue ver os workouts da semana, bem como verificar a sua condição física através dos seus dados biométricos. Na interface final foi acrescentada ainda uma barra do IMC com os valores de referência da OMS, informando assim o Cliente sobre o seu estado do IMC de forma mais intuitiva.

Na tabela o Cliente pode seleccionar um workout tanto por realizar como realizado, sendo que no caso de já estar realizado não lhe é permitido realizar outra vez. Isto é possível para o Cliente poder ver as Tarefas que realizou em Workouts anteriores.

Ainda nesta mockup contém a funcionalidade de avaliar o PT, para o cálculo da sua classificação.

Gym@HOME

Semana 8 (atual)

Dia 50 (3/10)	Dia 51 (4/10)	Dia 52 (5/10)	Dia 53 (6/10)	Dia 54 (7/10)	Dia 55 (8/10)	Dia 56 (9/10)
-	tronco	-	quadrícep	-	-	bícep
	workout concluído		consultar workout			consultar workout

1 ... 6 7 8

Dados biométricos:

Altura (cm): 175	Gêmeo (cm): 35
Peso (Kg): 62	Quadrícep (cm): 50
Cintura (cm): 80	Tríceps (cm): 30
Peito (cm): 120	Pulso (cm): 18

Avaliar PT

Copyright gym@home

Figura 8.18: Mockup Semana.

Gym@Home

Semana 1 (atual)

Segunda Dia 1 - 27/6	Terça Dia 2 - 28/6	Quarta Dia 3 - 29/6	Quinta Dia 4 - 30/6	Sexta Dia 5 - 31/6	Sábado Dia 6 - 1/7	Domingo Dia 7 - 2/7
	cardio				musculação	
	consultar workout				consultar workout	

« 1 2 3 4 5 6 »

Dados biométricos: (atualizados em 20/6/2020)

Altura (cm): 190	Gêmeo (cm): ---
Peso (Kg): 87,0	Quadrícep (cm): ---
Cintura (cm): ---	Tríceps (cm): ---
Peito (cm): ---	Pulso (cm): ---

Índice de Massa Corporal: 24,1

saudável 18,5 sobre peso 25 obesidade grau I 30 obesidade grau II (severa) 35 obesidade grau III (morbida) 40

Avaliar Personal Trainer

© 2020 Copyright: Gym@Home

Figura 8.19: Interface Semana.

Princípios de Usabilidade

- **Observability:** Mais especificamente **Browsability** pois o Cliente consegue navegar entre as várias semanas do seu plano.
- **Familiarity:** utilizou-se o formato semanal **comum/padrão** em horários e calendários, visto que é mais familiar aos utilizadores.
- **Generalizability e Consistency:** visto que utilizou-se a mesma estrutura tanto para a consulta pelo Cliente, bem como pelo PT, apesar de terem algumas diferenças, manteve-se a estrutura semanal, bem como os dados biométricos, excepto o botão de avaliação do PT, que não faz sentido no lado do PT.

Heurísticas de Normam

- **Visibility of system status:** o utilizador conhece o estado da aplicação, como por exemplo, os **workouts** que já foram feitos, aparecem com uma cor diferente.
- **Recognition rather than recall:** Os dados biométricos do Cliente são ilustrados durante a visualização da semana para o Cliente não ter de se lembrar de quais eram os valores dos dados biométricos.
- **Consistency and standards:** tal como foi dito em relação à **Familiarity**, sobre a estrutura semana usada, também se aplica a esta heurística relacionada com consistência e padrões.

8.3.7 Consultar Workout

Descrição

Da mesma forma que o anterior, este mockup também é bastante semelhante a ambos os utilizadores, pelo que apresentar-se-á apenas a view do Cliente. Nesta mockup o Cliente vê uma Tarefa de cada vez, com a listagem das séries e toda a informação necessário sobre a Tarefa. Aqui pode avançar e recuar nas tarefas, sendo que na última Tarefa do Workout pode terminar o Workout.

The mockup shows a desktop application window for 'GYM@HOME'. The top navigation bar includes icons for 'Iniciar Personal Trainer', 'Meus Planos', 'Meu Perfil', 'Notificações', and 'Logout'. The main content area displays a workout session titled 'workout - Dia 53 (6/10) - Tronco'. It shows a task 'Tarefa 1 - corrida' with a duration of '20 minutos', exercise type 'cardio', and equipment 'passadeira'. Below this is a table listing three exercises: 'correr lentamente para aquecer', 'correr mais depressa', and 'abrandar', each with its respective duration and rest information. A red text at the bottom indicates a 2-minute break until the next task. At the bottom right are buttons for 'Voltar para plano' and 'Próxima tarefa'.

Descrição	Repetições ou tempo	Reposo entre séries
correr lentamente para aquecer	5 minutos	sem repouso
correr mais depressa	10 minutos	sem repouso
abrandar	5 minutos	sem repouso

Figura 8.20: Mockup Workout.

The screenshot shows a web-based workout interface for 'Gym@Home'. At the top, there's a navigation bar with links: 'Procurar PersonalTrainer', 'Meus Pedidos', 'Meu Plano', 'Meu Perfil', 'Notificações', and 'Logout'. Below the navigation, the title 'Workout - Dia (25/6) - Musculação' is displayed. Underneath it, the section 'Tarefa 1 - Abdominais' is shown. A note indicates '15,4 minutos' and 'Equipamento necessário:'. A table lists the exercise details:

Descrição	Repetições ou tempo	Reposo entre séries
Abdominais	25 vezes	1 minuto(s)
Abdominais	25 vezes	1 minuto(s)
Abdominais	25 vezes	1 minuto(s)
Abdominais	25 vezes	1 minuto(s)
Abdominais	25 vezes	1 minuto(s)

A yellow bar at the bottom of the table area contains the text 'Reposo até à próxima tarefa: 0 segundos'. Below the table are two buttons: 'Voltar ao plano' (grey) and 'Terminar Workout' (green). At the very bottom of the page, a footer bar displays the copyright information: '© 2020 Copyright: Gym@Home'.

Figura 8.21: Interface Workout.

Princípios de usabilidade

- **Observability:** Mais especificamente **Browsability** pois o Cliente consegue navegar entre as várias tarefas de um workout.
- **Generalizability e Consistency:** visto que utilizou-se a mesma estrutura tanto para a consulta pelo Cliente, bem como pelo PT, apesar de terem algumas diferenças, manteve-se a estrutura tabelar das séries de uma tarefa, bem como as informações da tarefa, considerando-se assim que torna-se genérica para ambos os utilizadores.

Heurísticas de Normam

- **Recognition rather than recall:** as informações sobre o dia da semana e a definição do workout são passadas para esta view para que o cliente/PT não necessitem de memorizar.

8.3.8 Pedidos enviados pelo Cliente

Descrição

A interface Pedidos enviados pelo Cliente foi adicionada posteriormente ao desenho das mockups, pelo que só existe a interface final. Nesta interface o Cliente poderá ver todos os pedidos que já realizou, sendo apresentadas todas as informações do pedido, tais como o seu estado (aceite, rejeitado ou pendente), nome do PT, dados do formulário preenchido e ainda a possibilidade de ver o perfil do PT ao qual o pedido foi direcionado.

Gym@Home							Procurar PersonalTrainer	Meus Pedidos	Meu Plano	Meu Perfil	Notificações	Logout
Perfil do PersonalTrainer	Username	Semanas disponíveis	Objetivo	Workouts por semana	Disponibilidade (dias da semana)	Nível de treino	Estado					
perfil	@ptJoao	6	Ganhar Musculo	3	Ter, Qui e Sáb	Fácil	Pendente					

© 2020 Copyright: Gym@Home

Figura 8.22: Interface Pedidos.

Princípios de usabilidade

- **Synthesizability:** a acção possível nesta interface consiste em consultar o perfil do PT, pelo que após clicar no botão ”perfil”, aparece um pop-up com as informações do mesmo, ou seja, o cliente consegue perceber o efeito das suas acções.
- **Generalizability e Consistency:** o PT também contem uma interface para ver os pedidos enviados para o mesmo, que a nível de estrutura pode-se considerar bastante semelhante, difere apenas no conteúdos de algumas colunas, pelo que se pode considerar que se trata de uma interface genérica no sistema.
- **Observability:** esta interface pode aumentar bastante o número de linhas da tabela, no entanto cada linha tem os botões para as acções desse pedido, o que significa que se houvesse uma tabela gigante, as acções possíveis estariam lá, não necessitando de subir a página ou outra forma para encontrar os botões. No entanto o grupo reconhece que aqui também seria um bom candidato a paginação, não sendo possível implementar face à falta de tempo, mas será uma boa actualização no futuro.

Heurísticas de Normam

- **Recognition rather than recall:** o cliente não precisa de memorizar o que preencheu no formulário quando enviou o pedido, pois pode consultar na tabela esses mesmos campos.
- **Flexibility and efficiency of use:** o botão ”perfil”, que mostra as informações do PT torna esse processo de consulta mais **eficiente**, como uma espécie de atalho.

8.3.9 Avaliar PT

Descrição

A qualquer momento o Cliente pode avaliar o PT pelo seu trabalho, sendo pedido uma classificação de 1 a 5 estrelas, estas avaliações serão usadas posteriormente como parâmetro para escolher ou não um PT.

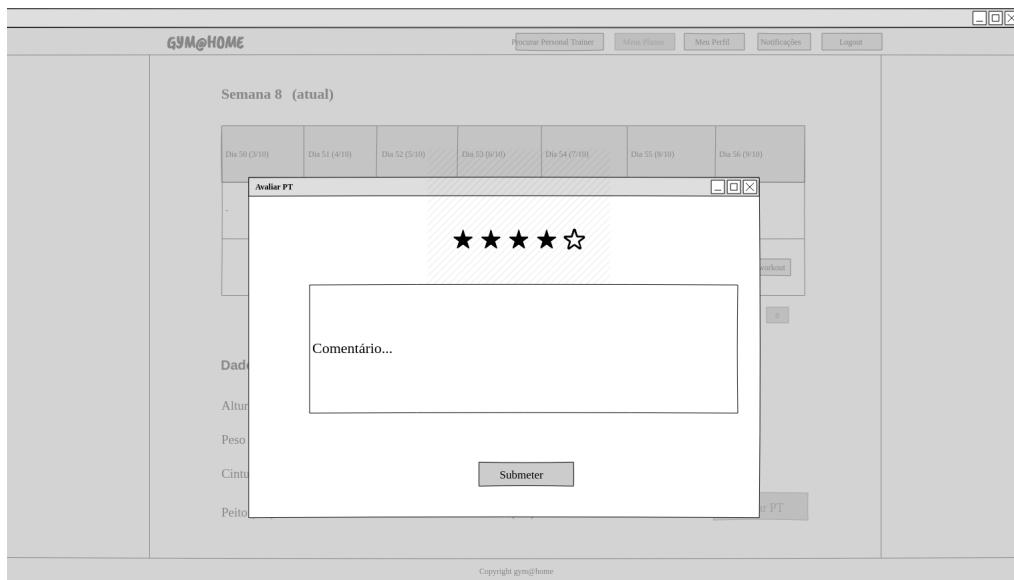


Figura 8.23: Mockup Avaliar PT.

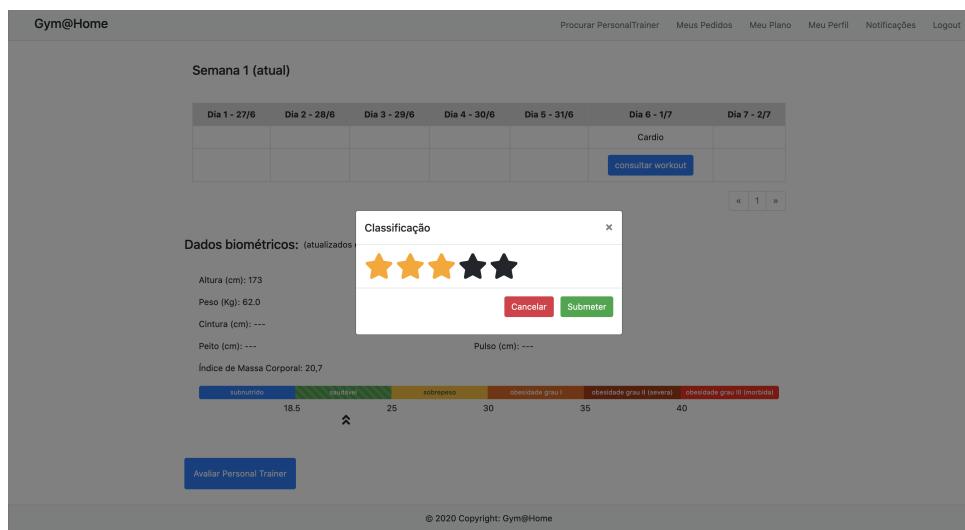


Figura 8.24: Interface Avaliar PT.

Princípios de usabilidade

- **Familiarity e Consistency:** usou-se uma forma padrão para as avaliações, através de cinco estrelas onde o utilizador selecciona.

Heurísticas de Normam

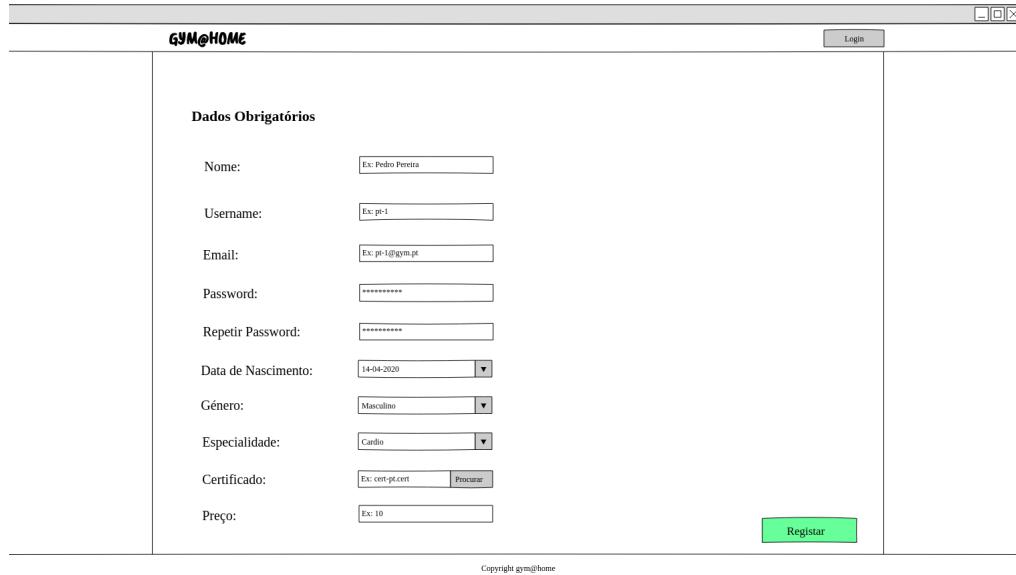
- **Error prevention:** a avaliação é inicializada com pelo menos uma estrela seleccionada, para evitar que o utilizador pudesse enviar zero estrelas, visto que a escala vai de um a cinco.
- **Consistency and standards:** tal como foi dito anteriormente, segue os princípios de **Familiarity e Consistency**, pelo que significa que também segue esta heurística relacionada com **padrões**.

8.4 Interfaces PTs

8.4.1 Registar PT

Descrição

Tal como no registo Cliente são pedidos alguns dados obrigatórios, sendo que o PT não tem dados opcionais.

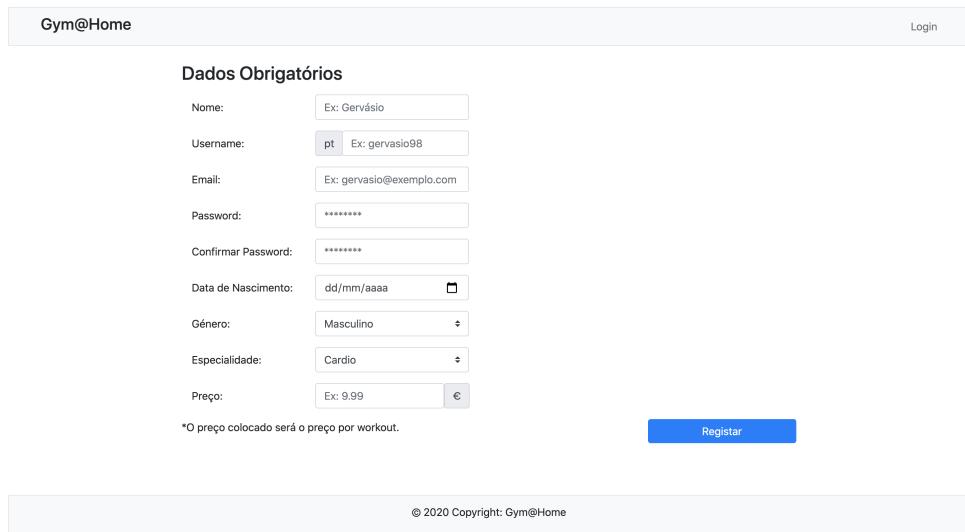


A interface de registo de PT é intitulada "GYM@HOME". No topo, há uma barra com ícones para maximizar, minimizar e fechar, e uma barra com o nome "GYM@HOME" e um botão "Login". O formulário principal é intitulado "Dados Obrigatórios". Ele contém os seguintes campos:

- Nome: Ex: Pedro Pereira
- Username: Ex: pt-1
- Email: Ex: pt-1@gym.pt
- Password: Senha
- Repetir Password: Senha
- Data de Nascimento: 14-04-2020
- Género: Masculino
- Especialidade: Cardio
- Certificado: Ex: cert-pt.cert [Procurar]
- Preço: Ex: 10

No lado direito do formulário, há um botão verde intitulado "Registrar". Abaixo do formulário, há uma barra com o copyright "Copyright gym@home".

Figura 8.25: Mockup Registar PT.



A interface de registo de PT é intitulada "Gym@Home". No topo, há uma barra com o nome "Gym@Home" e um botão "Login". O formulário principal é intitulado "Dados Obrigatórios". Ele contém os seguintes campos:

- Nome: Ex: Gervásio
- Username: pt Ex: gervasio98
- Email: Ex: gervasio@exemplo.com
- Password: Senha
- Confirmar Password: Senha
- Data de Nascimento: dd/mm/aaaa
- Género: Masculino
- Especialidade: Cardio
- Preço: Ex: 9.99 €

Abaixo do formulário, há uma nota: "*O preço colocado será o preço por workout." E no lado direito, há um botão azul intitulado "Registrar". Abaixo do formulário, há uma barra com o copyright "© 2020 Copyright: Gym@Home".

Figura 8.26: Interface Registar PT.

Princípios de Usabilidade

- **Familiarity e Consistency:** Colocar a "Data de Nascimento" é feito através de um calendário, pelo que o PT como ser humano está mais familiarizado a utilizar.

Heurísticas de Normam

- **Error prevention:** Todos os inputs numéricos aceitam apenas números, o "Género" e "Especialidade" apenas permitem os valores dos dropdowns, a "Data de Nascimento" apenas permite datas selecionadas pelo calendário e o "Email" apenas aceita um email (tem de conter o @).
- **Consistency and standards:** tal como foi dito anteriormente, segue os princípios de **Familiarity** e **Consistency**, pelo que significa que também segue esta heurística relacionada com **padrões**.
- **Flexibility and efficiency of use:** visto que após ser feito o registo, faz-se o login automático, sendo mais **eficiente** para o cliente.

8.4.2 PT consulta/edita o próprio perfil

Descrição

Do mesmo modo que o Cliente o PT pode visualizar e editar o seu perfil através desta mockup. Também nesta interface final, o botão de "Salvar Alterações" foi deslocado para o topo da página para facilitar a sua visualização em ecrãs mais pequenos.

The mockup shows a web page titled "GYM@HOME". At the top right are links for "Meus pedidos", "Meus Clientes", "Meu Perfil", and "Logout". The main content area has a heading "Username" and a form with the following fields:

- Nome: Pedro Pereira
- Email: exemplo@email.com
- Género: Masculino
- Especialidade: Cardio
- Preço: 10

A "Salvar alterações" button is located at the bottom of the form. The footer contains the text "Copyright gym@home".

Figura 8.27: Mockup Perfil PT visto pelo próprio.

Figura 8.28: Interface Perfil PT visto pelo próprio.

Princípios de usabilidade

- **Substitutivity:** Os inputs da interface onde o PT pode alterar os valores são também outputs. Sempre que o PT alterar dados estes são guardados na BD e novamente carregados para os outputs que o PT tinha utilizado como input.
- **Observability:** O botão "Salvar Alterações" está no topo da página para o PT o conseguir observar mal carregue a página.
- **Familiarity:** Colocar a "Data de Nascimento" é feito através de um calendário, pelo que o PT como ser humano está mais familiarizado a utilizar.

Heurísticas de Normam

- **Error prevention:** Todos os inputs numéricos aceitam apenas números, o "Género" e "Especialidade" apenas permitem os valores dos dropdowns, a "Data de Nascimento" apenas permite datas selecionadas pelo calendário e o "Email" apenas aceita um email (tem de conter o @).
- **Flexibility and efficiency of use:** tal como diz respeito ao princípio **Substitutivity**, usar o mesmo componente quer para visualizar, quer para editar o perfil, torna a interface mais eficiente para o PT.

8.4.3 Consulta do perfil do Cliente pelo PT

Descrição

Com o objectivo de facilitar o acesso rápido às informações do Cliente sem ter que ir para uma nova página e voltar, colocou-se um botão/atalho, que mostra os mesmos num pop-up, sendo isto possível e consistente em diversas interfaces, sendo que no lado do PT será no "Meus Clientes" e "Meus Pedidos de Clientes".

Assim reduziu-se a quantidade de cliques necessários para aceder aos dados, desta forma, acha-se que foi um boa melhoria, principalmente no ponto de vista do PT que fará esta acção repetitivamente, tornando assim a utilização da aplicação mais eficiente.

A alteração para pop-up, torna-se bastante diferente do protótipo, no entanto, as alterações na opinião da equipa foram para melhorar.

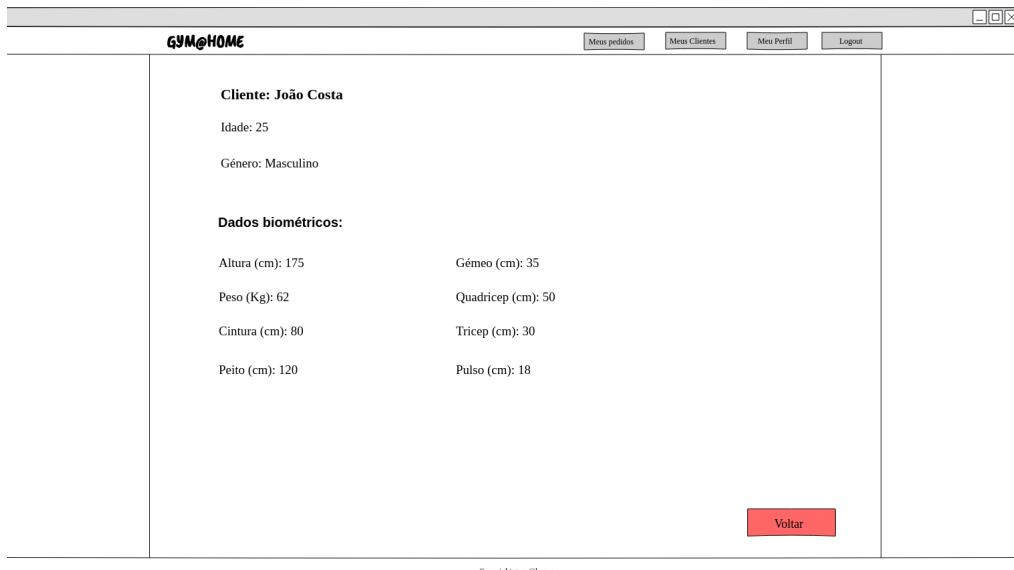


Figura 8.29: Mockup Perfil Cliente visto pelo PT.

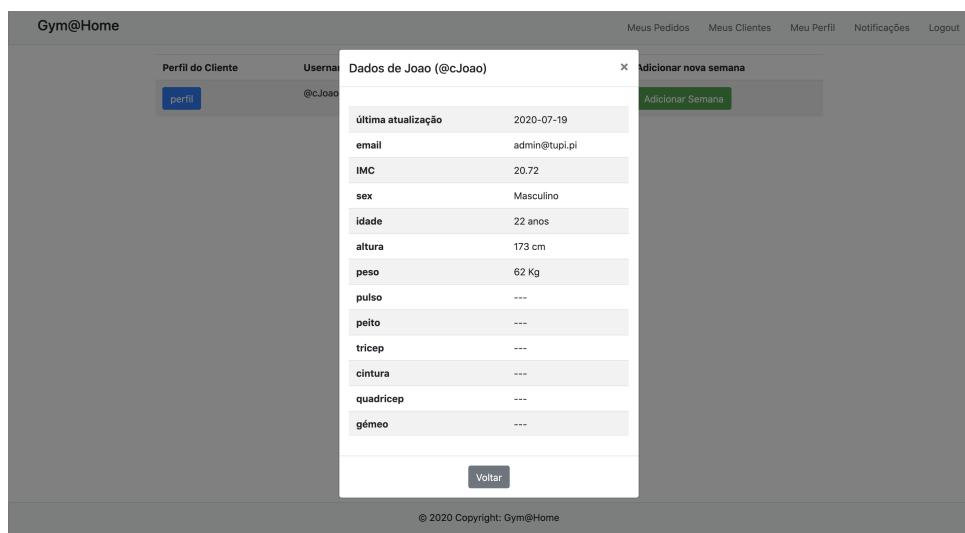


Figura 8.30: Interface Perfil Cliente visto pelo PT.

Princípios de usabilidade

- Consistency:** na medida em que, mantém-se sempre a mesma forma de apresentar o perfil (pop-up) nas várias views que necessitem dessa informação. Pode-se também considerar consistente devido ao facto de a apresentação do perfil do PT ter a mesma estrutura mas com campos diferentes.
- Flexibility and efficiency of use:** o botão "perfil", que mostra as informações do Cliente torna esse processo de consulta mais **eficiente**, como uma espécie de atalho.

Heurísticas de Normam

- **Consistency and Standards:** devido à consistência da mesma forma de apresentação pelas várias mockups que necessitam de aceder à informação do cliente.
- **Flexibility and efficiency of use:** o botão de "perfil" presente em algumas interfaces que necessitam de consultar os dados do cliente, torna esse processo mais eficiente para o utilizador, neste caso para o PT. Principalmente para este utilizador, que poderá lidar com muitos pedidos de plano, e para tal precisará de consultar os dados dos clientes, e com apenas um clique consegue fazê-lo.

8.4.4 Pedidos recebidos pelo PT

Descrição

Inicialmente importa referir que esta foi uma das interfaces que mudou drasticamente em relação ao protótipo, pois achou-se que não fazia sentido. Primeiramente a tabela lateral apenas tinha o nome do cliente, o que não é nada informativo para o PT. Depois, o PT tinha que clicar na tabela para ver os dados do formulário de cada cliente, o que não é eficiente para o mesmo. Do mesmo modo como já foi referido, o botão para consultar o perfil na página do cliente não é de todo a forma mais eficiente.

A solução para tornar esta interface mais eficiente, foi em listar os pedidos, bem como os dados mais importantes referentes ao pedido (campos do formulário), permitindo facilmente ao PT ver a informação de cada pedido sem necessitar de clicar em cada cliente. Isto permite por exemplo excluir alguns pedidos apenas olhando para estes dados da tabela. No entanto, consultar os dados biométricos é importante em certas situações para se decidir se aceita ou rejeita o pedido. Dessa forma, e como em outras views onde se necessita de consultar dados de perfil, colou-se um botão em cada pedido para aparecer um pop up com a informação do cliente. Como por vezes o PT pode decidir no momento em que consulta esses dados se vai aceitar ou rejeitar, logo, propaga-se os botões de aceitar ou rejeitar para esse pop-up.

Após aceitar um pedido, será redirecionado para a interface "Criar uma semana do plano", caso rejeite, fica na mesma interface e o pedido é removido da tabela.

Copyright gym@home

Figura 8.31: Mockup Pedidos Clientes.

The screenshot shows a web-based application for a personal trainer (PT). At the top, there is a navigation bar with links for 'Meus Pedidos', 'Meus Clientes', 'Meu Perfil', 'Notificações', and 'Logout'. Below the navigation bar is a table with the following data:

Perfil do Cliente	Username	Semanas disponíveis	Objetivo	Workouts por semana	Disponibilidade (dias da semana)	Nível de treino
perfil	@cJoao	6	Ganhar Musculo	3	Ter, Qui e Sáb	0

Below the table are two buttons: 'Aceitar' (Accept) in green and 'Rejeitar' (Reject) in red. At the bottom of the page, there is a footer with the text '© 2020 Copyright: Gym@Home'.

Figura 8.32: Interface Pedidos Clientes.

Princípios de usabilidade

- **Synthesizability:** uma das acções possíveis nesta interface consiste em consultar o perfil do Cliente, pelo que após clicar no botão "perfil", aparece um pop-up com as informações do mesmo, ou seja, o PT consegue perceber o efeito das suas acções. Por outro lado, o botão aceitar, redireciona para a página "Criar semana do plano", logo, da mesma forma, o PT, também percebe o efeito da sua ação. Por fim, o rejeitar, marca o pedido como rejeitado, e o mesmo não aparecerá na tabela no próximo refresh, mostrando ao PT o efeito de ter rejeitado.
- **Generalizability e Consistency:** tal como já foi dito, o Cliente também contém uma interface para ver os pedidos enviados pelo o mesmo, que a nível de estrutura pode-se considerar bastante semelhante a este, difere apenas no conteúdos de algumas colunas, pelo que se pode considerar que se trata de uma interface genérica/consistente no sistema.
- **Observability:** esta interface pode aumentar bastante o número de linhas da tabela, no entanto cada linha tem os botões para as acções desse pedido, o que significa que se houvesse uma tabela gigante, as acções possíveis estariam lá, não necessitando de subir a página ou outra forma para encontrar os botões. No entanto o grupo reconhece que aqui também seria um bom candidato a paginação, não sendo possível implementar face à falta de tempo, mas será uma boa actualização no futuro.

Heurísticas de Normam

- **Recognition rather than recall:** o PT não precisa de memorizar as informações dos formulários no momento em que aceita, pois esses dados são enviados para a interface "Criar Semana do Plano".
- **Flexibility and efficiency of use:** o botão "perfil", que mostra as informações do Cliente torna esse processo de consulta mais eficiente, como uma espécie de atalho.

8.4.5 Clientes do PT

Descrição

A interface permite ao PT ter acesso rápido a todos os seus Clientes, que tem actualmente, com a finalidade de ver o Plano de cada um, o perfil ou adicionar uma nova semana aos planos dos mesmos.

Em cada linha da tabela, aparecem os botões referentes às funcionalidades referidas acima associados a cada cliente.

ID	Name	Número de Semanas do Plano Atual	Ações
1	João Costa	8	Sim
2	Pedro Pereira	1	Não
3	Filipa Martins	1	Sim
4	João Martins	3	Sim
5	Pedro Costa	1	Sim

Figura 8.33: Mockup Clientes do PT.

The interface shows a summary of the user's profile:

Perfil do Cliente	Username	Avaliações	Ver plano atual	Adicionar nova semana
perfil	@c.Joao	não	Ver plano	Adicionar Semana

Figura 8.34: Interface Clientes do PT.

Princípios de usabilidade

- **Synthesizability:** uma das acções possíveis nesta interface consiste em consultar o perfil do Cliente, pelo que após clicar no botão "perfil", aparece um pop-up com as informações do mesmo, ou seja, o PT consegue perceber o efeito das suas acções. Por outro lado, o botão ver plano, redirecciona para a página "Consultar Plano", logo, da mesma forma, o PT, também percebe o efeito da sua acção. Por fim, o adicionar semana, redirecciona para a página "Adicionar Semana", pelo que o PT percebe o efeito da sua acção.
- **Observability:** esta interface pode aumentar bastante o número de linhas da tabela, no entanto cada linha tem os botões para as acções desse cliente, o que significa que se houvesse uma tabela gigante, as acções possíveis estariam lá, não necessitando de subir a página ou outra forma para encontrar os botões. No entanto, o grupo reconhece que aqui também seria um bom candidato a paginação, não sendo possível implementar face à falta de tempo, mas será uma boa actualização no futuro.

Heurísticas de Normam

- **Recognition rather than recall:** o PT não precisa de memorizar as informações do cliente no momento em que cria uma nova semana para o plano do mesmo, pois esses dados são enviados para a interface "Criar Semana do Plano".
- **Flexibility and efficiency of use:** o botão "perfil", que mostra as informações do Cliente torna esse processo de consulta mais **eficiente**, como uma espécie de atalho.

8.4.6 Criar Semana

O workout "Criar Semana" foi um dos mockups que teve que ser revisto face à primeira fase de apresentação do projecto, após o conselho da equipa docente de repensar o mesmo, pois estava demasiado complexo, confuso e pouco usável. A equipa decidiu que teria de ser tudo feito numa única página. Dessa forma, Chegou-se ao resultado que se pode ver nas figuras [8.35](#) e [8.36](#).

Primeiramente, importa realçar o preload dos dados do formulário do cliente, importantes para saber criar a semana. À esquerda, tem-se uma tabela, com os workouts criados, sendo que inicialmente estará vazia. Depois tem-se a escolha do nome para o workout, bem como o dia da semana, sendo que este dropDown dos dias, vai automaticamente buscar os dias do formulário e reduzindo-os cada vez que se faz um workout. Por fim, tem-se uma tabela, super dinâmica, que aumenta e diminui de linhas, permitindo ao PT adicionar/remover o número de tarefas que pretender, não sendo um valor fixo.

Importante realçar nesta interface que a semana só é guardada no momento em que clicar no botão guardar semana, caso contrário, tudo que foi criada é descartado. Como se trata de opções impossíveis de reverter, também são difíceis de realizar, pois aparece sempre um pop-up a perguntar se tem a certeza da acção.

Descrição

Gym@HOME

Semana 1: João Costa

Dia	Workout
3ª feira	Levantar Pesos

3º feira
5º feira
6º feira
Sábado

Nº de semanas (disponibilidade): 6 semanas
Objetivo: Ganhar Músculo
Nº de treinos semanais: 3 dias por semana
Disponibilidade semanal: Sáb, 3º, 5º e 6º
Dificuldade do plano: Normal

Workout

Nº	Tarefa	Pesos em Kg	Séries	Repetições ou Tempo (min, seg, vezes)	Tempo entre Séries (min, seg)
1	Corrida		2	5 min	30 seg
2	Levantar Pesos	10	3	14 vezes	1 min

Levantar Pesos
Abdominais
Corrida

Adicionar Linha

Cancelar Workout **Guardar Workout**

Copyright gym@home

Figura 8.35: Mockup Criar Semana.

Gym@Home

Semana 1: @cJoaoo

Workouts já definidos:

Dia	Workout	Tempo
X Quinta	Cardio	32,3 minuto(s)

Nº de semanas (disponibilidade): 6 semanas
Objetivo: Ganhar Musculo
Nº de treinos semanais: 3 dias por semana
Disponibilidade semanal: Ter, Qui e Sáb
Dificuldade do plano: Fácil

Designação do Workout: Dia da semana do Workout:
Musculação Sábado

Workout Tabela:

	Tarefa	Séries	Repetições ou Tempo	Descanso entre Séries	Descanso entre Tarefas	Equipamento
X	Abdominais	5	25 vezes	1 min	0 seg	Ex: passadeira

Adicionar Linha

Cancelar Workout Atual **Adicionar Workout**

© 2020 Copyright: Gym@Home

Figura 8.36: Interface Criar Semana.

Princípios de usabilidade

- **Substitutivity:** O PT pode definir um tempo em segundos ou em minutos.
- **Customizability:** pois o PT pode adicionar/remove mais linhas à tabela onde define as tarefas de um workout
- **Observability:** na medida em que, quando o PT guarda um workout, o mesmo faz **preload** na tabela que está no topo (tabela denominada, "workouts já definidos", podendo o mesmo observar os workouts que já definiu).

- **Recoverability:** a tabela onde estão os workouts já definidos permite apagar, ou seja, reverter. De modo semelhante, o botão "Cancelar workout" também reverte a tabela onde se define as tarefas.

Heurísticas de Normam

- **Recognition rather than recall:** Os dados do pedido do Cliente são ilustrados durante a criação da semana para o PT não ter de se lembrar de quais eram os requisitos do pedido.
- **Error prevention:** Todos os inputs numéricos aceitam apenas números. Não é permitido ao PT "Guardar Semana" se a Semana não tiver nenhum Workout criado para a mesma.

8.4.7 Semana do Cliente vista pelo PT

Descrição

A mockup é completamente análoga à "Semana" vista pelo Cliente, a única diferença é que no PT aparece o nome do Cliente ao qual aquela Semana pertence.

GYM@HOME

Cliente: João Costa
Semana 8 (atual)

Dia 50 (3/10)	Dia 51 (4/10)	Dia 52 (5/10)	Dia 53 (6/10)	Dia 54 (7/10)	Dia 55 (8/10)	Dia 56 (9/10)
-	tronco	-	quadrícep	-	-	bicep
	workout concluído		consultar workout			consultar workout

Dados biométricos:

Altura (cm): 175	Gêmeo (cm): 35
Peso (Kg): 62	Quadrícep (cm): 50
Cintura (cm): 80	Tríceps (cm): 30
Peito (cm): 120	Pulso (cm): 18

Voltar

Figura 8.37: Mockup Semana Cliente.

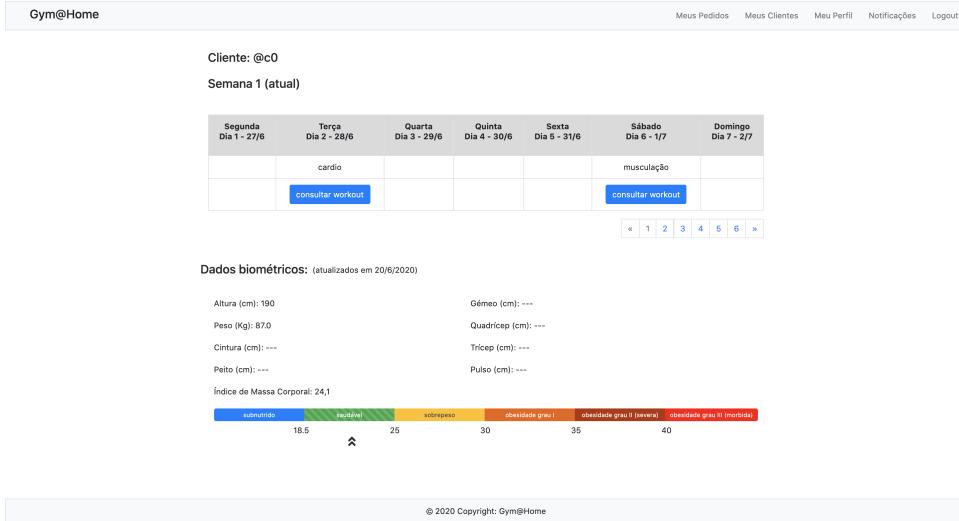


Figura 8.38: Interface Semana Cliente.

Princípios de Usabilidade

- **Observability:** Mais especificamente **Browsability** pois o PT consegue navegar entre as várias semanas do plano do Cliente.
- **Familiarity:** utilizou-se o formato semanal **comum/padrão** em horários e calendários, visto que é mais familiar aos utilizadores.
- **Generalizability e Consistency:** visto que utilizou-se a mesma estrutura tanto para a consulta pelo PT, bem como pelo Cliente, apesar de terem algumas diferenças, manteve-se a estrutura semanal, bem como os dados biométricos, excepto o botão de avaliação do PT, que não faz sentido no lado do PT.

Heurísticas de Normam

- **Recognition rather than recall:** Os dados biométricos do Cliente são ilustrados durante a visualização da semana para o PT não ter de se lembrar de quais eram os valores dos dados biométricos.

8.4.8 Workout do Cliente visto pelo PT

Descrição

Tal como na mockup anterior esta é totalmente análoga à do Cliente, sendo que nesta aparece o nome do Cliente e o **PT não pode finalizar Workout**.



Figura 8.39: Mockup Workout Cliente.

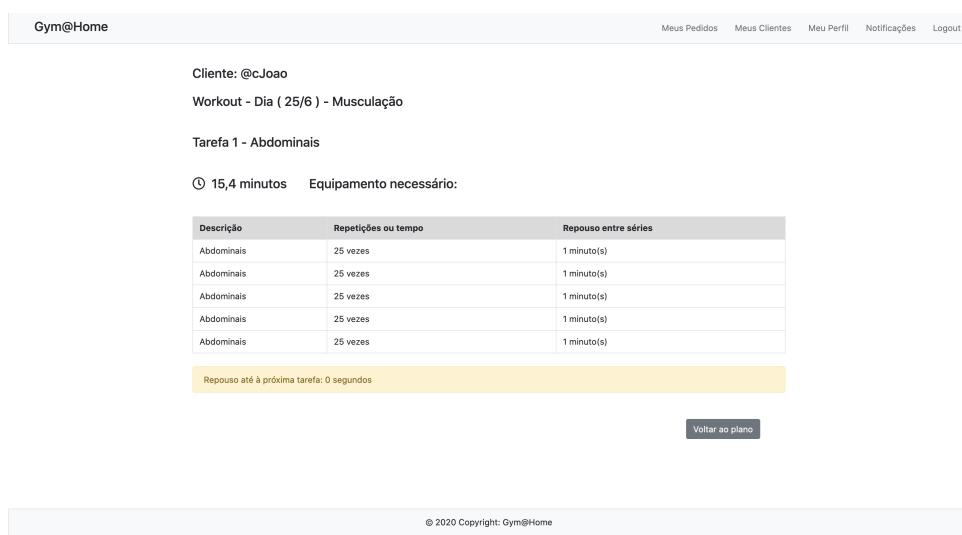


Figura 8.40: Interface Workout Cliente.

Princípios de usabilidade

- **Observability:** Mais especificamente **Browsability** pois o PT consegue navegar entre as várias tarefas de um workout.
- **Generalizability e Consistency:** visto que utilizou-se a mesma estrutura tanto para a consulta pelo Cliente, bem como pelo PT, apesar de terem algumas diferenças, manteve-se a estrutura tabelar das séries de uma tarefa, bem como as informações da tarefa, considerando-se assim que torna-se genérica para ambos os utilizadores.

Heurísticas de Normam

- **Recognition rather than recall:** as informações sobre o dia da semana e a definição do workout são passadas para esta view para que o cliente/PT não necessitem de memorizar.

8.5 Alertas

Algumas acções são impossíveis de reverter, dessa forma, deve-se dificultar as mesmas. Assim, foi criado o alerta de **confirmação** para ter a certeza que o utilizador pretende executar a acção.

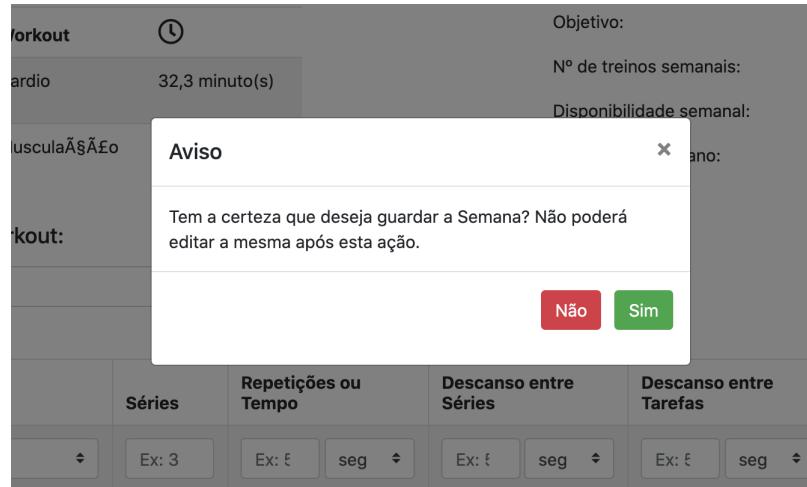


Figura 8.41: Alerta de Confirmação.

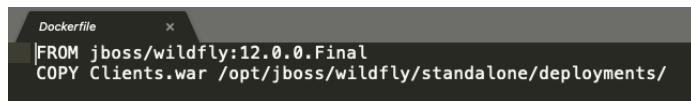
Capítulo 9

Deployment

9.1 Ficheiros de configuração necessários para o deployment

A colocação da arquitectura em funcionamento necessita de um deployment de todos os componentes, sendo de preferência um processo automatizado.

Inicialmente foi necessário criar as imagens de todos os componentes desenvolvidos pela equipa. Na verdade, consiste em criar uma ficheiro Dockerfile tal como se pode observar na figura 9.1. Nestas imagens necessita-se do servidor web, neste caso **wildfly**, versão 12, que é automaticamente descarregado da internet. De seguida, necessita-se do ficheiro **.war** do projecto **NetBeans** do serviço, que será copiada para a pasta deployments do servidor web. Após a criação do ficheiro Dockerfile, necessita-se garantir que os ficheiros necessários para a criação da imagem estão juntamente com o ficheiro.



```
Dockerfile
FROM jboss/wildfly:12.0.0.Final
COPY Clients.war /opt/jboss/wildfly/standalone/deployments/
```

Figura 9.1: Exemplo de um dockerfile para um serviço do backend.

De seguida, após concluir a criação de todos os dockerfiles, procede-se à criação do ficheiro de configuração denominado docker-compose.yml, que contém os serviços da arquitetura. Alguns serviços serão externos, tais como as BDs associadas a cada um dos sete serviço desenvolvido pelo grupo. A imagem dessas BDs será descarregada de um repositório, sendo que a mesma foi criada pela equipa de desenvolvimento do **MySQL**.

A ferramenta docker tem a grande vantagem na criação de containers, que de uma forma sucinta, isolam um serviço com as dependências necessárias para o mesmo, permitindo facilidade na instalação de serviços.

Uma grande vantagem da utilização do docker, consiste na criação de uma rede interna, onde todos os containers comunicam através da mesma, contendo até um servidor DNS, o que significa que para fazer a comunicação entre dois serviços basta utilizar o nome que lhes foi dado ao serviço, tal como se pode verificar na figura 9.2, onde se faz a conexão a uma base de dados usando o nome do serviço **clientdb**. Assim, esta característica facilita na criação do docker-compose.yml.

```

<Setting type="2">
  <Driver>MySQL (Connector/J Driver)</Driver>
  <DriverFiles>&lt;MySQL Connector/J 8.0.20&gt;&lt;/DriverFiles>
  <Dialect>org.hibernate.dialect.MySQLDialect</Dialect>
  <DriverClass>com.mysql.jdbc.Driver</DriverClass>
  <URL>jdbc:mysql://Clientdb:3306/clients?serverTimezone=UTC</URL>
  <UserName>root</UserName>
  <EncryptedPassword>86B5D09CD8912AF029972D274BBCBE9B</EncryptedPassword>
  <HostName>Clientdb</HostName>
  <PortNo>3306</PortNo>
  <DBName>clients?serverTimezone=UTF8</DBName>
  <ServiceName></ServiceName>
  <ServerName></ServerName>
  <SelectedForm>true</SelectedForm>
</Setting>

```

Figura 9.2: Utilização do nome do serviço docker para conectar a base de dados.

O ficheiro de configuração docker-compose.yml define os serviços a fazer deployment, onde se especifica o nome, a localização da imagem/dockerfile, portas, entre outras variáveis importantes. Na 9.3, o quadrado azul representa um serviço para administração das base de dados. O quadrado vermelho corresponde ao serviço frontend. O quadrado amarelo, corresponde ao serviço GymAtHome, que consiste no servidor **API** do backend, ou seja, o serviço responsável por fazer o dispatcher dos pedidos. Por fim, a verde tem-se um exemplo de um serviço do backend, responsável por guardar e gerir os planos. Todos os restantes serviços desenvolvidos pela equipa, são idênticos a este último, mudando apenas o nome do serviço, da imagem e base de dados.

Com a finalidade de facilitar/automatizar o processo de deployment, o grupo desenvolveu um makefile, que faz todos os comandos necessários para obter os .wars mais recentes, bem como executar o comando **docker-compose up -d --build**, para efetuar o deployment.

```

version: '3'

services:

  # mysql database management
  adminer:
    image: adminer
    restart: always
    ports:
      - "8000:8080"

  frontend:
    image: frontend
    build: frontend/
    ports:
      - "8080:8080"

  gymathome:
    image: gymathome
    build: gymathome/
    ports:
      - "8081:8080"

  coredb:
    image: mysql
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: core
    core:
      image: core
      build: core/

```

Figura 9.3: Ficheiro docker-compose.yml (o ficheiro tem os restantes serviços, no entanto, para explicar basta apenas esta parte do ficheiro).

9.2 Processo deployment

Nesta secção apresenta-se o processo necessário para fazer deployment da aplicação.

9.2.1 Requisitos

- docker
- diretoria docker/
- postman (ou outra alternativa que permita fazer HTTP POST)
- makefile

9.2.2 Passo a passo

- cd docker/
- make up
- no postman terá que ser feito um pedido post para **IP_HOST:8081/GymAtHome/api/createdbs**, com o body em /application/json com o token = admin (em json), tal como se pode verificar na figura 9.4, para inicializar as base de dados.

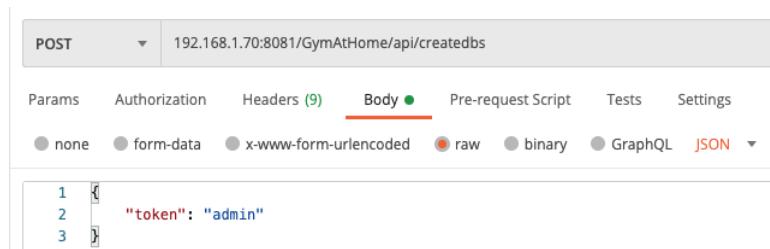


Figura 9.4: Pedido feito no postman para inicializar as base de dados.

Desta forma, verifica-se que o processo de deployment torna-se bastante automatizado, necessitando apenas de um comando e um pedido http no postname, sendo que este pedido só seria necessária na primeira vez para a criação dos schemas nas base de dados.

9.3 Preparado para Docker Swarm

Apesar de não ter sido feito, os ficheiros de configuração (docker-compose.yml) estão preparados para colocar a aplicação em produção, por exemplo num **Docker Swarm** com várias máquinas virtuais em **Swarm**, onde todas as máquinas poderão alojar serviços da arquitectura, no entanto não foi possível conceber o **Docker Swarm** nesta fase do projeto.

Outra nota importante consiste em colocar as imagens no **DockerHub**, um repositório de imagens, para que seja ainda mais fácil o deployment, necessitando apenas do ficheiro de configuração e não das imagens.

No entanto, como foi dito, neste momento o deployment não está a ser feito para um **Docker Swarm**, no entanto, está preparado para tal.

Capítulo 10

Conclusão

Em suma, conclui-se que grande parte dos objectivos inicialmente propostos foram atingidos, conseguindo-se chegar a uma aplicação web que na opinião da equipa com uma boa interface focada na usabilidade, bem como no utilizador. Do mesmo modo, a arquitectura orientada a serviços foi conseguida com sucesso, permitindo ao grupo perceber as vantagens e desvantagens das mesmas.

Inicialmente houveram algumas dificuldades no desenho do diagrama de classes, pois não estávamos a entender se era vantajoso aplicar determinados design patterns como o **Builder**, **Factory**, entre outros, que após algumas sessões de dúvidas foram se dissipando.

Depois a equipa teve complicações a usar as ferramentas **NetBeans** e **IntelliJ** para a criação de beans. No entanto, após um pouco de pesquisa percebeu-se que existia um problema nos ficheiros de configuração.

Após a conclusão dos serviços foi necessário fazer a infraestrutura, bem como o deployment, no entanto, foi nesse momento que o grupo percebeu que a arquitectura é bastante pesada, correndo apenas num computador de um elemento do grupo, limitando-nos para testar pois estávamos dependentes do mesmo. No entanto encontrou-se uma solução de executar os serviços em localhost, todos partilhando o mesmo servidor web, ficando muito mais leve, permitindo que já não houvesse tanta dependência.

A nível de frameworks apenas usou-se o Hibernate, de resto foi tudo o que se aprendeu nas aulas, pois são as ferramentas que o grupo se sentiu mais à vontade, tais como JSP, Servlets, Beans, ...

Uma das interfaces, Criar uma semana, na primeira apresentação estava demasiado complexa, onde o grupo foi avisado pela equipa docente do mesmo. Dessa forma, a equipa decidiu redesenhar o mockup com apenas uma página, conseguindo-se o resultado acima, que na opinião do grupo ficou "perfeito".

Após a utilização do Hibernate gerado pelo Visual Paradigm, o grupo ficou um pouco reticente à utilização do mesmo neste formato, pois "tira-nos" um bocado liberdade, se o projecto começasse hoje provavelmente o grupo optaria por fazer com Hibernate em anotações, ou outra framework ou até desenvolver a comunicação com a BD.

Como em todos os projectos existe sempre pontos a melhorar e/ou acrescentar, pelo que este projecto não foi excepção. De seguida são enumerados alguns desses pontos.

- Introduzir pagamentos na aplicação, para os Clientes poderem pagar pelos seus planos e consequentemente os PTs receberem pelo seu trabalho.
- Colocar passagem de tempo nas tarefas para o Cliente não necessitar de utilizar cronómetros ou outros dispositivos para controlar a passagem do tempo.
- Adicionar um histórico de planos no Cliente para este poder visualizar todos os planos que já realizou.
- Pedir o certificado de PT aos PTs para estes se poderem registar na aplicação, uma vez que é necessário

controlar se quem cria planos está devidamente certificado para o fazer.

- Adicionar novo tipo de utilizador (Administrador) para validar os certificados introduzidos pelos PTs.
- Acrescentar Workouts predefinidos para os PTs não terem de preencher a mesma informação várias vezes.
- Para uma melhor experiência do utilizador, acrescentar paginação nas notificações, nos pedidos realizados (Cliente), nos pedidos recebidos (PT) e nos clientes actuais (PT).
- De forma a melhorar ainda mais a experiência dos utilizadores, fazer o redimensionamento das interfaces para dispositivos mais pequenos.

Assim, conclui-se, que este projecto foi importante para se perceber a importância quer de desenhar um boa arquitectura e as consequências do mesmo, quer da importância do desenvolvimento da interface gráfica focada nos objectivos do utilizador para maximizar a usabilidade.