INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema de Fitness Inteligente: Exercícios

RUI FILIPE QUINTAS BARBOSA
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de fitness inteligente

Exercícios

Rui Filipe Quintas Barbosa

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: António Constantino Lopes Martins

Co-orientador: Luiz Felipe Rocha de Faria

Júri:

Presidente:

Vogais:

Porto, Outubro 2019

Dedicatória

Dedico esta dissertação ao meu avô.

Resumo

Os sistemas de recomendação têm valorizado e aumentado em popularidade nos últimos anos, com os utilizadores a procurarem a facilidade e simplicidade na obtenção de informações e sugestões, tendo por base as suas preferências e expectativas.

Adicionalmente, o estilo de vida fitness está na moda com cada vez mais pessoas a adotá-lo. Dessa forma, os praticantes deste tipo de estilo de vida utilizam aplicações para registar as suas atividades, definir objetivos, monitorizar o progresso e evolução, entre outros. Contudo, a maioria das aplicações são específicas a um tipo de utilizador, sendo difícil encontrar uma que seja direcionada a praticantes iniciantes e, ao mesmo tempo, a experientes. Além disso, existem várias lacunas nas aplicações existentes, como falta de funcionalidades importantes ou aspetos de usabilidade.

Posto isto, foi desenvolvido um sistema com uma componente móvel que permite a qualquer tipo de praticante de *fitness* realizar as funcionalidades fundamentais que a sua atividade requer, tais como registo das suas atividades, consulta de treinos ou exercícios fitness já existentes ou criados, personalização de planos e esclarecimento de dúvidas com um assistente pessoal. Além disso, possui um módulo de recomendação com o objetivo de recomendar treinos com base na natureza destes (p.e. nível ou músculos exercitados), nos objetivos do mesmo, e nas características do utilizador.

Para validar as funcionalidades gerais, a utilidade e a usabilidade dos componentes desenvolvidos, foram elaborados inquéritos de satisfação. O resultado foi bastante positivo, onde uma elevada percentagem dos utilizadores inquiridos indicou que os módulos desenvolvidos eram bastante úteis, intuitivos e simples. Adicionalmente, as técnicas de recomendação também foram alvo de teste de forma a analisar o erro de previsão de cada uma delas. De forma geral, os resultados também foram positivos, com erros de previsão consideravelmente baixos.

Palavras-chave: Fitness, Exercícios, Monitorização, Sistemas de Recomendação, Assistente Pessoal

Abstract

Recommender Systems have been increasing in popularity and appreciation over the past few years, with the users seeking ease and simplicity when obtaining information and suggestions, based on their preferences and expectations.

Additionally, the fitness lifestyle is being adopted by increasingly more people. In this way, practitioners of this type of lifestyle use applications to record their activities, set goals, monitor progress and evolution, among others. However, most applications are specific to one type of user, and it is difficult to find one that is targeted at both beginner and experienced practitioners. Moreover, there are several gaps in existing applications, such as lack of important functionalities and usability aspects.

With this, a mobile component system has been developed, which allows any type of fitness practitioner to perform the fundamental features that their activity requires, such as recording them, consulting existing or created fitness workouts, customizing plans and questions with a personal assistant. In addition, it has a recommendation module with the purpose of recommending workouts based on their nature (i.e. experience level or muscles exercised), and the user's characteristics and objectives.

To validate the general functionality, usefulness and usability of the developed components, satisfaction inquiries were developed. The result was very positive, where a high percentage of users stated that the modules developed were very useful, intuitive and simple. Additionally, the recommendation techniques were also tested to analyze the prediction error of each one. Overall, the results were also positive, with considerably low errors.

Keywords: Fitness, Exercises, Monitoring, Recommender Systems, Personal Virtual Assistant

Agradecimentos

Agradeço a todos os que tornaram possível a realização deste projeto, por muito mínima que possam considerar a sua ajuda.

Agradeço, em particular, aos Professores Constantino Martins e Luiz Faria pelo tempo despendido durante a realização do projeto em acompanhamento e orientação, enaltecendo a disponibilidade para reunir e discutir todas as dúvidas.

Agradeço ao Jorge, especialista na área de fitness, por todos os esclarecimentos que se tornaram fundamentais no bom decorrer do projeto.

Agradeço aos meus amigos David Abreu e Pedro Gonçalves pelos conselhos e amizade.

Por fim, agradeço à minha família e à minha namorada, pelo apoio incondicional ao longo de todo o projeto e pela força que me deram para o desenvolvimento do mesmo.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

AGAP	Associação de Empresas de Ginásios e Academias de Portugal
API	Application Program Interface
FC	Filtragem colaborativa
FCBI	Filtragem colaborativa baseada em itens
FCBU	Filtragem colaborativa baseada em utilizadores
FFE	Fuzzy Front-End
IA	Inteligência Artificial
LSI	Latent Semantic Indexing
MF	Matrix Factorization
NCD	New Concept Development
NPD	New Product Development
PT	Personal trainer
PVA	Personal Virtual Assistant
SO	Sistema Operativo
SR	Sistemas de Recomendação
SVD	Singular Value Decomposition
SWEBOK	Software Engineering Body of Knowledge

Lista de Símbolos

1 Introdução

Neste capítulo, é apresentado o contexto relativamente ao tema da dissertação, mencionados quais os problemas, bem como os objetivos a atingir tendo em conta esses problemas identificados. De seguida, são indicados os fatores motivacionais que levaram à realização deste projeto e a sua metodologia. Por fim, identifica-se a estrutura do documento, de forma a descrever o conteúdo de cada capítulo.

1.1 Contexto

Atualmente, o mercado de *fitness* encontra-se em contante crescimento a nível mundial, sendo o estilo de vida da moda. Diversos artigos e relatórios estudaram este mercado e concluíram que os Estados Unidos são o país com maior número de praticantes e ginásios [1]. Comparativamente à Europa, a Alemanha é o país com maior adesão por parte das pessoas [2]. No caso português, também é notório um crescimento significativo neste setor nas vertentes mencionadas - membros e ginásios [3].

A condição física e a prática de exercício são imprescindíveis para manter uma boa saúde. "A inatividade física é um fator de risco associado a doenças cardiovasculares e a uma variedade cada vez maior de outras doenças crónicas, incluindo diabetes, cancro, obesidade, hipertensão, doenças ósseas e articulares, e depressão." [4], sendo a prática de exercício físico fundamental para evitar essas doenças.

O acentuado crescimento pela procura de um estilo de vida fitness faz com que este mercado possua oportunidades de negócio a diversos níveis. Um deles, associado ao facto de que as pessoas, nos dias de hoje, usam com frequência um *smartphone*, a utilização de aplicações móveis com o intuito de registar e automatizar todas as atividades de *fitness* realizadas é cada vez mais usual [5].

De facto, a utilização deste tipo de aplicações é cada vez mais frequente, sendo que entre 2014 e 2016 o seu uso aumentou em mais de 330% [6]. Normalmente, os seus utilizadores

procuram uma app (aplicação) que disponibilize as funcionalidades desejáveis para o acompanhamento das suas atividades de *fitness*, como uma boa usabilidade, personalização, registo dos treinos, visualização da sua evolução, entre outros [7].

Determinados estudos foram realizados com o propósito de averiguar se as referidas aplicações providenciam algum tipo de comportamento diferente nas pessoas que as usam, sendo o resultado positivo. Por exemplo, na revisão da literatura do artigo de Molina e Sundar [8], é referido que os participantes que usaram uma app de *fitness* realizaram mais exercícios por semana comparativamente àqueles que não usaram.

1.2 Problema

Como mencionado anteriormente, o *fitness* encontra-se em contínuo crescimento, sendo que há cada vez mais pessoas a aderir a este tipo de estilo de vida.

A existência de um plano estruturado de treino, onde são definidos vários exercícios tendo em conta os objetivos de cada pessoa, é imprescindível para que haja uma evolução e mudanças positivas. Geralmente, este é elaborado por *personal trainers*, sendo também, fundamental, que haja uma monitorização do plano por parte do utilizador.

Em muitas situações, em indivíduos menos experientes no mundo do *fitness*, esta monitorização é deficiente ou não existente, afetando diretamente os seus resultados [9]. Este problema pode levar à desmotivação ou até mesmo à desistência em certo ponto. Por outro lado, em utilizadores mais experientes que possuem planos de treinos bem estruturados e organizados, existem outros tipos de dificuldades, nomeadamente, a complexidade da monitorização de um plano de treino avançado e a necessidade de retirar informação personalizada dos resultados obtidos.

Em ambos os casos descritos, a indisponibilidade de *personal trainers* ou a impossibilidade em ter um não permitem um acompanhamento eficaz a todos os praticantes, conduzindo a práticas pouco eficientes e desadequadas às necessidades e perfil de cada um.

Tendo em conta o referido, diversos praticantes de *fitness* recorrem a aplicações móveis para terem ajuda nessas vertentes. Contudo, as soluções existentes são específicas a um tipo de utilizador, sendo difícil encontrar uma que seja versátil a esse nível, adequada a praticantes iniciantes e, ao mesmo tempo, a experientes. Além disso, existem certas lacunas nas aplicações existentes: falta de funcionalidades importantes tendo em conta o tipo de utilizador; usabilidade; inexistência de funcionalidades que providenciem um acompanhamento próximo de um *personal trainer*; entre outros.

1.3 Objetivos

Tendo em conta os problemas expostos, verifica-se uma necessidade em encontrar uma forma de ajudar qualquer tipo de praticante de *fitness* de forma a monitorizar e melhorar as suas atividades. Com vista a resolver os problemas identificados, propõem-se o desenvolvimento de um sistema de monitorização aliado a um sistema de recomendação de treinos adaptado às necessidades e perfil do utilizador.

O sistema a desenvolver deverá oferecer um conjunto de funcionalidades relacionadas com o domínio de treinos ou exercícios de treino, como feedback personalizado de resultados (desempenho), ajudas, bem como recomendações de treinos com base na natureza do treino (p.e. nível, equipamentos utilizados ou músculos exercitados), nos objetivos do mesmo, e nas características do utilizador. Essa monitorização, seria, então, uma espécie de gestão e controlo sob os dados deste, tendo como objetivo capturar a sua evolução.

O registo de informação relativa a treinos ou exercícios deve ser feita de forma intuitiva e amigável, com o principal objetivo de fornecer uma gestão personalizada das rotinas de treino para todos os tipos de indivíduos, permitindo a consulta de informação acerca destes. Além disso, seria possível o acesso à informação por parte de um possível, mas não necessário, *personal trainer*, potencializando os resultados do utilizador. O sistema a desenvolver deverá ainda possuir um módulo de recomendação, melhorando o acompanhamento de utilizadores, através do uso de técnicas de Inteligência Artificial. Adicionalmente, um outro módulo a ser desenvolvido relacionar-se-á com um assistente virtual inteligente que terá como responsabilidade esclarecer dúvidas relativas a exercícios. O objetivo global descrito será alcançado através da persecução dos seguintes objetivos específicos:

- Investigação e análise de diversas aplicações/sistemas fitness existentes no mercado, bem como uma análise critica e comparação destas;
- Investigação e modelação do domínio do projeto - exercícios e treinos de *fitness*;
- Conceção e implementação de uma aplicação móvel que permita a consulta e criação de exercícios e treinos *fitness*, assim como outras funcionalidades fundamentais para a prática de exercício físico (ginásio);
- Conceção de um módulo de monitorização e controlo dos dados do utilizador, tendo como objetivo capturar a sua evolução;
- Desenvolvimento de um módulo de esclarecimento de dúvidas inteligente (*personal virtual assistant*) para interação com o utilizador;
- Realização de um estudo que demonstre a utilidade da ferramenta desenvolvida, seguida pela demonstração e análise dos resultados.
- Cumprimento de todas as fases do ciclo de desenvolvimento de *software* – especificação de requisitos, análise de diferentes propostas de arquitetura, implementação, testes e *deploy*.

1.4 Motivação

O tema apresentado identifica-se como uma melhoria no que toca a aplicações móveis *fitness*. A escolha deste tema deu-se à utilização pessoal deste tipo de apps por um dos intervenientes deste projeto, sendo que por mais aplicações utilizadas, não era encontrada uma que satisfazesse todos os critérios e funcionalidades pretendidas.

Ao nível motivacional, apresentar uma solução com todas as funcionalidades importantes, fundamentais e algumas inovadoras que consigam ajudar os praticantes de *fitness* através de: recomendações tendo em conta, por exemplo, os seus objetivos; consultas eficazes da sua evolução; e monitorização de todas as suas atividades, constitui um grande desafio e é extremamente aliciante. Desta forma, a solução pode auxiliar os treinos dos praticantes de *fitness*, motivando-os à prática de exercício físico melhorando a sua saúde.

O gosto e interesse pela área de saúde e fitness, a possibilidade de: realizar a análise de um problema de raiz; examinar diferentes concorrentes do mercado analisando os seus pontos fortes e fracos; identificar as melhores tecnologias para o problema em questão; desenvolver uma arquitetura para o sistema, assim como a sua implementação, foram, também, grandes fatores de decisão e motivação.

1.5 Metodologia

O presente projeto é realizado em paralelo com outro projeto de tese. Dessa forma, existe uma clara separação ao nível do domínio. De facto, a presente dissertação foca-se no domínio dos exercícios e treinos de fitness, sendo a restante relacionada com os planos e programas de treino. Assim, para garantir um bom resultado no desenvolvimento deste projeto são utilizadas ferramentas colaborativas como o *Trello* (ver Figura 1) e *Bitbucket*. Cada um dos domínios mencionados possui as suas próprias funcionalidades, sendo cada um dos seus intervenientes responsável por estas.

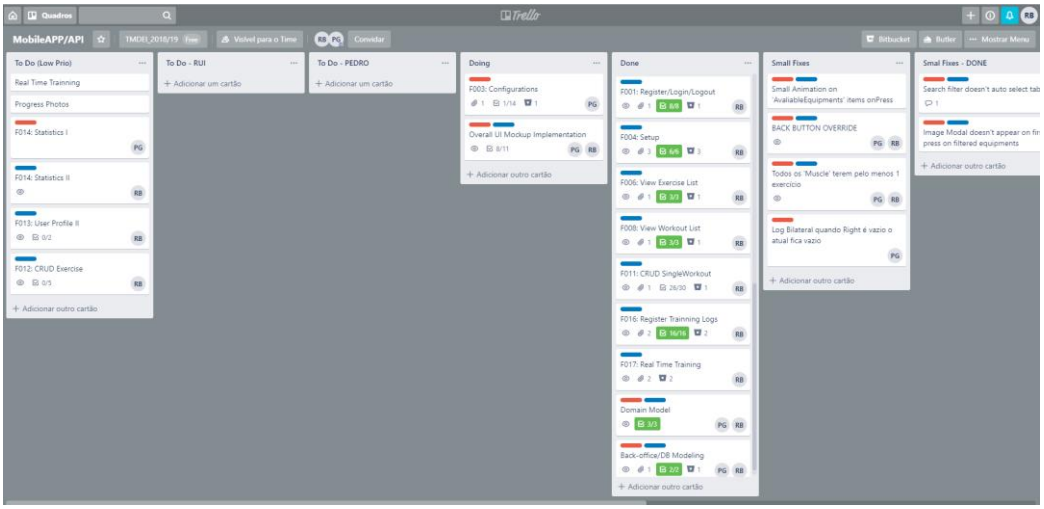


Figura 1 – Utilização da ferramenta *Trello*

1.6 Organização do documento

O presente relatório está estruturado da seguinte forma:

- **Primeiro capítulo – Introdução:** apresenta uma breve descrição desta dissertação, onde é explicado o seu contexto, os problemas, os objetivos, a motivação pela qual é feito este projeto e a estrutura deste documento;
- **Segundo capítulo – Análise de Valor:** é apresentada a descrição detalhada do contexto nas vertentes *fitness*, aplicações e exercícios de *fitness*, bem como uma descrição da análise de valor para o produto;
- **Terceiro capítulo – Estado de Arte:** o domínio de sistemas de recomendação será contextualizado, incluindo os tipos mais comuns, problemas existentes e exemplos destes tipos de sistemas relativos ao *fitness*. Por fim, serão apresentadas diversas aplicações existentes, fazendo uma análise crítica sobre cada uma destas, descrevendo as suas funcionalidades e aspetos positivos e negativos;
- **Quarto capítulo – Design:** neste capítulo são apresentados os requisitos funcionais e não funcionais, bem como os atores e partes interessadas do sistema. Também são apresentas as arquiteturas estudadas e uma análise a cada uma destas;
- **Quinto capítulo – Implementação:** é descrito todo o processo de desenvolvimento do sistema tendo em conta os objetivos e os requisitos funcionais e não funcionais;
- **Sexto capítulo – Avaliação:** aqui é apresentado como foi efetuada a análise da ferramenta desenvolvida, incluindo a definição das métricas, a formulação das hipóteses e as metodologias empregues, bem como a análise dos resultados obtidos;
- **Sétimo capítulo – Conclusão:** são apresentadas as conclusões relativas ao projeto desenvolvido, bem como os objetivos atingidos. São referidas e justificadas as limitações, e são sugeridas abordagens e soluções para possível trabalho futuro a desenvolver no sistema.

2 Análise de Valor

No presente capítulo, o contexto para o projeto será aprofundando e o módulo de análise de valor será definido, incluindo os cinco elementos *Front-End* do modelo *New Concept Development* (NCD), diferentes conceitos de valor, proposta de valor e o modelo CANVAS.

2.1 Contexto

De forma a contextualizar e apresentar com mais detalhe o domínio deste projeto, os conceitos de *fitness*, aplicações e exercícios de *fitness* serão aprofundados.

2.1.1 Fitness

Fitness pode ser definido como o "estado de ser fisicamente apto e saudável através de exercícios, dieta e hábitos de sono" [5]. O conceito de *fitness* e o de atividade física estão relacionados, sendo este último, segundo a Organização Mundial de Saúde, qualquer movimento corporal produzido pelos músculos, resultando em gasto energético [10].

A indústria de *fitness* tem sofrido um crescimento enorme nas últimas décadas e não existem sinais que irá parar [11][12]. *Les Mills*, uma empresa criadora de 13 grupos globais de fitness e planos de treinos para equipas, cita um estudo que afirma que o fitness é o novo grande desporto e ainda possui alto potencial de crescimento [13]. Um questionário realizado por esta a 4.600 pessoas de 13 países diferentes, revelou que 27% da população adulta frequenta um ginásio, academia de fitness ou clube de saúde, e que 61% dos exercícios são feitos em atividades de ginásio [13].

Os Estados Unidos da América lideram este mercado, com maior número de ginásios e membros, 60,9 milhões e 38 milhões, respetivamente [1].

Já na Europa, em termos de adesão, a Alemanha é o líder do mercado com 10,6 milhões de sócios de ginásios (+ 5,3%), seguida do Reino Unido (9,7 milhões), França (5,7 milhões), Itália

(5,3 milhões) e Espanha (5,2 milhões) [2]. A Figura 2 mostra que os três países mencionados apresentaram taxas de aumento de sócios acima da média no ano de 2017.

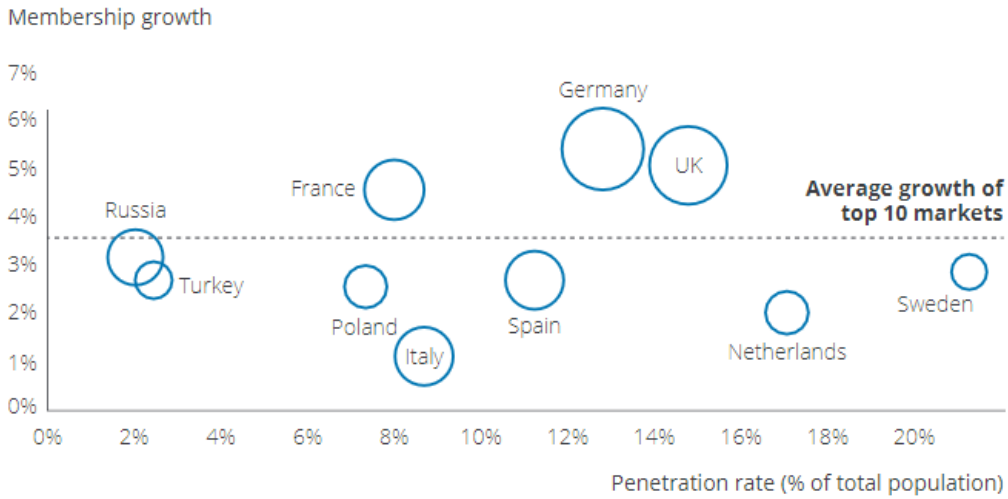


Figura 2 - Total de sócios nos ginásios, crescimento de sócios e taxa de penetração dos 10 principais mercados europeus de fitness (2017) [2]

Comparativamente ao mercado português de fitness, segundo o barómetro da AGAP, Associação de Empresas de Ginásios e Academias de Portugal, com dados referentes ao ano de 2015, o mercado português de fitness cresceu 13% nesse ano, somando 730 mil pessoas, o que equivale a 7,1% da população total [14]. Já em 2016, segundo dados do jornal Expresso, o número de sócios dos ginásios cresceu 30%, enquanto o número de novos espaços aumentou 14% [3]. Assim, pode-se admitir que os portugueses demonstram ser cada vez mais adeptos do fitness.

Com todos estes dados levantados de determinados estudos e pesquisas a afirmar que o fitness está na moda e em forte crescimento, uma questão torna-se muito pertinente:

Como é que o fitness se tornou tão popular mundialmente?

Um dos aspetos que contribuiu para tal encontra-se relacionado com as redes sociais [15]. Plataformas como o Instagram, Facebook e Twitter têm servido para que profissionais de saúde e fitness e, também, não profissionais, partilhem os seus conhecimentos e atividades com o público e seus seguidores [15]. É o exemplo de Kayla Itsines, uma entusiasta do fitness que através do Instagram conseguiu partilhar os seus conhecimentos com o mundo, possuindo, atualmente, 10,9 milhões de seguidores, uma aplicação e diversos blogs acerca do fitness [15]. Além disso, a evolução no que toca à prática de fitness também ajudou neste crescimento. É o exemplo de novos formatos específicos de treino, como o treino intervalado de alta intensidade (HIIT), treino de grupo, treino de força e treino realizando vários exercícios seguidos sem intervalos de descanso (*superset*), novos equipamentos nos ginásios e utilização de dispositivos *wearables* (p.e. Apple Watch) [11][12][16][17].

Outros fatores são reconhecidos por contribuírem para esta popularidade do fitness, tais como, o aparecimento de ginásios *low-cost* que disponibilizam preços acessíveis aos seus membros e o facto da idade não constituir uma barreira. Contudo, razões relacionadas com a saúde são as mais mencionadas [11][17].

Na verdade, esta indústria beneficiou de iniciativas de saúde que promoveram a importância da condição física no combate a doenças e a fatores que as originam, como diabetes e obesidade, e o sedentarismo, respetivamente [15][18]. Daqui, resultou uma mudança de atitude por parte das pessoas em relação aos cuidados de saúde, que inclui a procura por alimentos mais saudáveis e uma maior preocupação pelo corpo [11][12][15].

Para o futuro, está previsto que o valor desta indústria continue a crescer possuindo amplas oportunidades de negócio [15][17]. A personalização, perceção genética através de testes de DNA e aplicações móveis terão um impacto no que toca às novas tendências nesta área [19][20]. Personalização para que dispositivos de fitness forneçam um feedback útil, incluindo, por exemplo, o estado do funcionamento interno pessoal [19]. Testes de DNA à saliva, para a geração de um perfil genético e posterior geração de recomendações acerca da falta de nutrientes, vitaminas vitais para prevenir lesões, exercícios e planos de dieta adaptados geneticamente, entre outras [19]. Aplicações móveis continuarão a proporcionar uma experiência de fitness totalmente personalizada e integrada, que funde a vida dos membros fora do ginásio com suas ambições neste [20]. Na secção 2.1.2 este último tópico será aprofundado, visto ser fundamental na presente tese.

2.1.2 Aplicações de Fitness

As aplicações móveis de fitness podem ser definidas, tal como referido no estudo de Yul acerca dos principais determinantes no que toca à aceitação dos utilizadores neste tipo de apps, como um "[…] *software* que funciona em *smartphones* e dispositivos móveis sendo projetado para educar, entreter ou ajudar pessoas interessadas na condição física" [21].

Como já referido, nos últimos anos o número de uso de *smartphones* cresceu em enorme quantidade, e daí, o número de aplicações sofreu o mesmo efeito [5][6]. É o caso das aplicações direcionadas ao fitness, tornando-se uma ferramenta fundamental para tornar a vida das pessoas mais saudável [5][6][22][23].

Flurry, uma empresa que monitoriza mais de 1 milhão de aplicações de todas as categorias, lançou nos finais de 2017 um relatório de análise sobre o uso de aplicações de fitness resultante de dados combinados dos sistemas operativos *Android* e *iOS*, sendo que algumas das suas conclusões serão aqui referidas [6].

Entre 2014 e 2016 assistiu-se a um crescimento formidável no uso deste tipo de aplicações. De facto, nestes três anos, o seu uso aumentou em mais de 330%. Contudo, entre 2016 e 2017, este crescimento já não foi tão acentuado, existindo uma percentagem de apenas 9%

[6]. Para explicar este pequeno valor, os autores do estudo segmentarem o mercado de fitness em quatro subcategorias, como ilustrado na Figura 3.

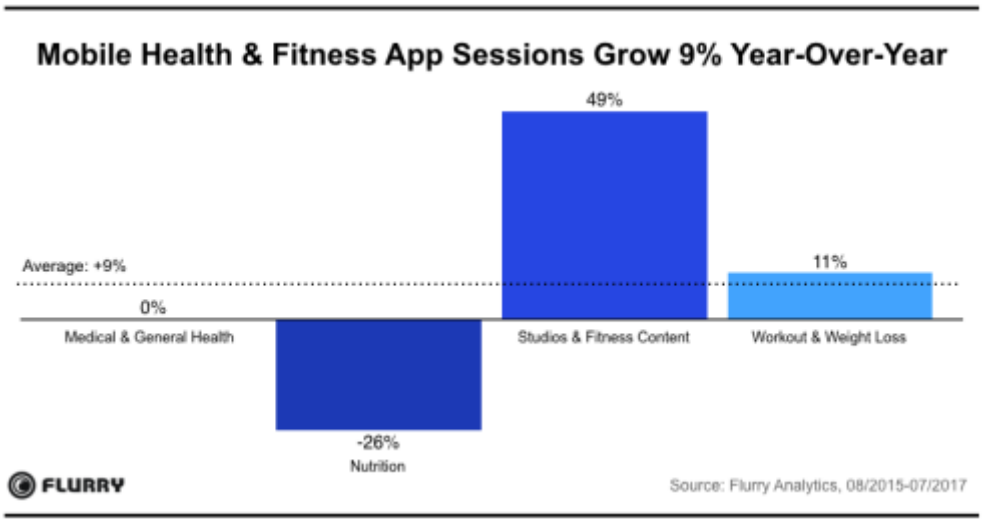


Figura 3 – Segmentos da categoria fitness e suas percentagens de crescimento entre 2016 e 2017 [6]

Através da análise da Figura 3 é possível concluir que o valor referido acima é influenciado devido à tendência negativa das aplicações de nutrição. Esta tendência é resultado de, possivelmente, à inclusão de planos de nutrição nas aplicações direcionadas aos ginásios (*Studios & Fitness Content*), não sendo o seu foco principal [6].

Não analisando em termos de crescimento, mas sim, em popularidade geral no mercado, apesar do vasto aumento das aplicações de ginásios (49%), esta não é a subcategoria que apresenta o maior valor neste critério.

Tal como ilustrado na Figura 4, as aplicações de treino e perda de peso são as mais populares, sendo responsáveis por quase três quartos de todas as apps de saúde e fitness (73%). Essa percentagem pode ser atribuída à utilização de dispositivos *wearables*, que incentivam os utilizadores a monitorizarem o seu peso ou exercícios diariamente [6].

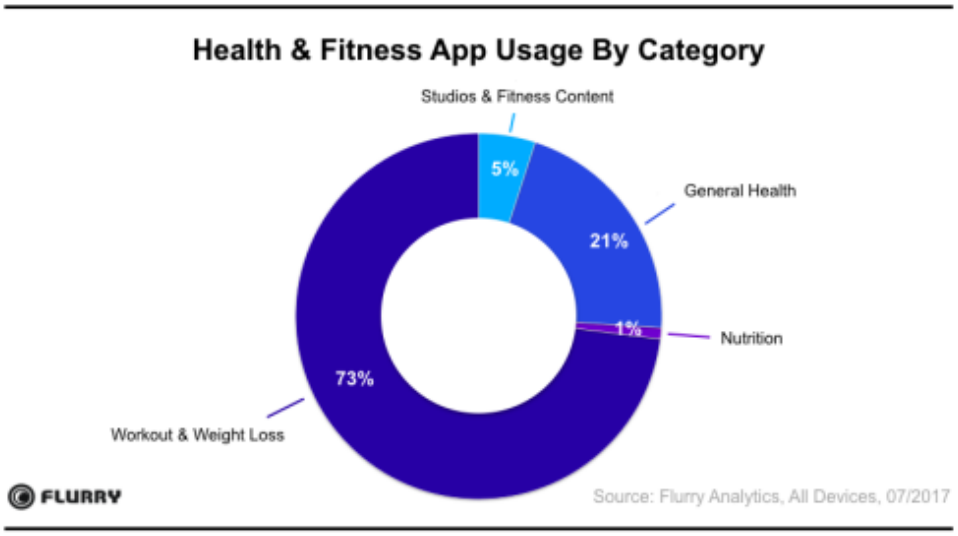


Figura 4 - Segmentos da categoria fitness em percentagem (2017) [6]

Com o crescimento já referido de 49% e evolução na popularidade de 2% em 2016 para 5% em 2017, as aplicações de *Studios & Fitness Content* fazem com que os ginásios sejam cada vez mais obrigados a disponibilizarem aos seus clientes uma app para organização dos treinos e aulas [6].

Analisando o período de utilização das aplicações de fitness, conclui-se que o seu uso está relacionado com os hábitos de treino. Isto é, são mais utilizadas durante a manhã e a noite, e durante segunda a quarta-feira, sendo o fim de semana a altura em que as apps não são tão utilizadas [6]. Cerca de 75% dos utilizadores abre a sua aplicação pelo menos 2 vezes durante a semana, sendo que 26% dessa percentagem, abre mais do que 10 vezes [6]. Os dados também mostraram que 96% dos utilizadores ativos usam apenas uma aplicação desta categoria [6].

Uma procura de estudos acerca da possível efetividade de aplicações fitness na mudança de comportamento e na saúde dos utilizadores foi realizada. Foi utilizada a biblioteca *PubMed* utilizando as seguintes *keywords*: "*Fitness mobile apps*" e com data a partir de 2014. Foram escolhidos quatro artigos e a sua escolha foi feita através da leitura do *abstract*, no sentido que permitisse, através dessa leitura, entender que o artigo retratava uma avaliação da efetividade já mencionada no início do presente parágrafo.

No estudo de A. Gabbiadini and T. Greitemeyer [22] foi avaliado se as apps de fitness afetam positivamente as atitudes, o controlo comportamental e as atividades físicas. Concluiu-se, que, os participantes que usaram uma app mostraram uma maior probabilidade de realizar uma caminhada e exercício físico. Assumiu-se, também, que o uso diário de um destas aplicações torna os participantes mais conscientes dos seus desempenhos, monitorizando o seu comportamento real, o que por sua vez leva a atitudes mais positivas relativamente a atividades saudáveis.

Na revisão da literatura do artigo de Molina e Sundar [8] vários artigos foram analisados. Um destes conclui que os participantes que usaram uma aplicação de fitness realizaram mais exercícios por semana comparativamente àqueles que não usaram. Da mesma forma, noutro artigo, aproximadamente três quartos dos que estavam a utilizar, mencionaram serem mais ativos.

No estudo de Sama, Eapen, Weinfurt, Shah e Schulman relativo a uma avaliação das ferramentas de aplicações de fitness e saúde, é referido que este tipo de aplicações "pode ter uma aplicação profunda na prevenção de doenças cardiovasculares ou no tratamento de pacientes com doenças crónicas, como diabetes e insuficiência cardíaca congestiva" [24].

Na revisão de literatura de Payne e de seus colegas acerca da influência das aplicações móveis na saúde demonstrou-se que o uso de uma app ficou associado a: melhorias na atividade física e tendências sociais mais amplas, mostrando o potencial de usar este tipo de aplicações para aspetos de comportamento de saúde [25].

Através de todos os dados acima, pode-se concluir que as aplicações de fitness tiveram um enorme crescimento, tanto a nível de número como de utilizadores, sendo expectável que continue a subir nessas vertentes.

2.1.3 Exercícios Fitness

Como já referido, um domínio fundamental da presente tese é o exercício fitness. "Exercício é o comprimido mágico" afirma Michael R. Bracko [26], presidente do Comité de Informação ao Consumidor do Colégio Americano de Medicina Desportiva. Dois artigos, um destes pertencente à Biblioteca Nacional de Medicina dos Estados Unidos [27] e outro escrito por Yasmine Ali e revisto por um médico certificado [28], afirmam que a prática de exercício possui inúmeros benefícios para a saúde, incluindo: prevenir a obesidade; melhorar a mobilidade; melhorar o humor; melhorar a qualidade de vida; prevenir diabetes; melhorar a longevidade saudável; reduzir o risco de muitas doenças crónicas; e prevenir doenças cardiovasculares.

Segundo um *personal trainer* afiliado à Universidade de Harvard, devem-se fazer exercícios de vários tipos [29]. São listados os 4 tipos mais importantes [29], contudo, para a presente dissertação, serão abordados mais dois, relacionados, nomeadamente, com a prática de ginásio:

- **Aeróbico:** caracterizado por acelerar o ritmo cardíaco e a respiração, com o objetivo de aumentar a resistência. Exemplo de: corrida, ciclismo, caminhada, natação e dança [29];

- **Força:** visa melhorar a força, os músculos e o ganho de massa muscular. Como exemplo: agachamentos e flexões [29];

- **Alongamento:** aumenta a amplitude de movimento das articulações e ajuda a manter a flexibilidade. O envelhecimento e a inatividade fazem com que os músculos, tendões e ligamentos percam a robustez. Da mesma forma, alongar os músculos torna-os mais longos e flexíveis [29];

- **Equilíbrio:** ajuda a manter o equilíbrio e a confiança em qualquer idade, prevenindo quedas e mantendo a independência. Exemplo: ficar suspenso sobre um pé [29];

- **Powerlifting:** tem como objetivo "levantar" o maior peso possível. É, também, considerado um deporto, tendo o atleta, numa plataforma dedicada, tentar levantar o maior peso que é capaz para uma única repetição. Os exercícios que o compõem são: aganhachamento (*squat*), dedicado para fortalecer a parte inferior do corpo; supino (*bench press*), fortalece a parte superior do corpo; e peso morto (*deadlift*), exercício composto que treina várias áreas musculares, como tronco, pernas, braços, ancas, entre outros [30];

- **Olympic Weightlifting**: idêntico ao explicado no ponto anterior, e também, considerado um desporto. *Olympic Weight Lifting* exige que o atleta levante uma barra utilizando uma combinação de força e velocidade, com o máximo de peso conseguido, do chão até uma posição suspensa de forma explosiva. Os exercícios que os constituem são: *snatch*, "elevador" de grande alcance e movimento único; e *clean and jerk*, um "elevador" de dois movimentos [31].

Na Figura 5 estão categorizados os principais grupos de músculos considerados pela Jefit, aplicação de fitness disponível para *iOS* ou *Android*. Cada exercício possui uma ou várias áreas de intervenção. Por exemplo, para o exercício *Chest Press* os braços (*arms*), peito (*chest*) e ombros (*shoulders*) são as áreas exercitadas [32].

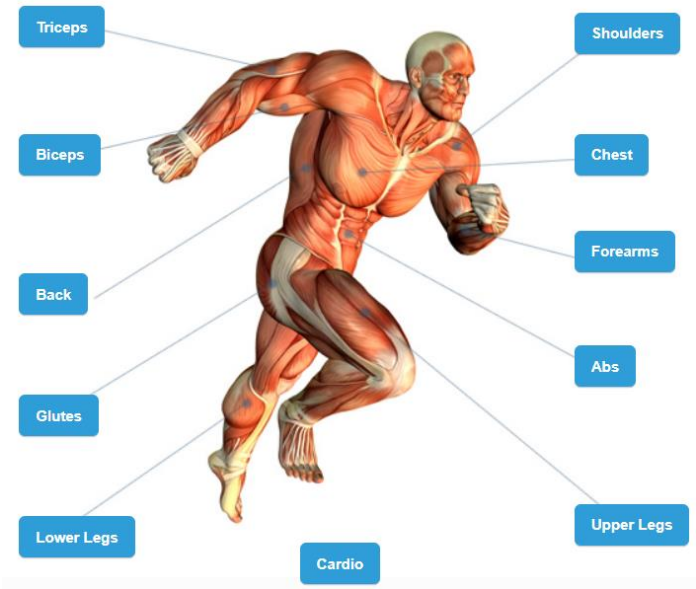


Figura 5 - Principais grupos de músculos [33]

Na Tabela 1 estão presentes os equipamentos mais gerais e populares onde milhares de exercícios existentes podem ser realizados [33]. Contudo, existem certos exercícios que não necessitam de um equipamento específico. É o exemplo das elevações (*pull ups*), sendo o peso do corpo o "equipamento".

Tabela 1 – Equipamentos de exercícios fitness mais populares [33][34]

Nome equipamento	**Descrição**
Banco (Bench)	Utilizado para fazer múltiplos exercícios, usando, por exemplo, halteres.
Barra com peso (Barbell)	Usado no levantamento de peso, musculação e outros.
Barra EZ (EZ Bar)	Variação da barra com peso, que ajuda a prevenir lesões e a isolar os bíceps.
Bola de estabilidade (Exercise Ball)	Usada, nomeadamente, para reabilitação de lesões, alongamento e equilíbrio.
Bola de fitness (Medicine Ball)	Bola com um determinado peso (p.e. 5kg) usada para melhorar a aptidão, força e coordenação.
Cabos (Cables)	Diversidade de exercícios que podem ser realizados exercitando todo o corpo.
Elásticos (Bands)	Ajudam na aprendizagem de novos exercícios e na fase de alongamento.
Halteres (Dumbbells)	Essencial para a musculação constituído por uma barra e pesos anexados em cada extremidade.
Kettlebell	Usado, essencialmente, para treino de força e cardio.
Máquinas – Cardio (Machine – Cardio)	Utilizadas para aumentar a resistência, condição física, perder gordura, entre outros.
Máquinas – Força (Machine – Strength)	Diversidade de máquinas em que as partes superiores e inferiores do corpo podem ser exercitadas.
Rolo (Foam Roll)	Ajudam a tratar a dor muscular, promovendo flexibilidade e relaxamento dos músculos.

Os exercícios podem ser classificados de diversas maneiras. Uma maneira comum de o fazer é em termos de como o exercício treina o corpo e quantos grupos musculares são exercitados. Neste caso, existem dois grupos: exercícios compostos - concentram-se na aptidão funcional para simular atividades da vida real e usar uma variedade de movimentos corporais, envolvendo o uso de mais do que um grupo muscular; e exercícios de isolamento - qualquer exercício em que apenas um grande grupo muscular é treinado [35]. Por outro lado, estes também podem ser classificados tendo em conta a sua dificuldade de execução, normalmente, divididos em principiante, intermédio e experiente [33].

Em suma, na Figura 6 estão presentes três exercícios diferentes, sendo que as vertentes descritas ao longo desta secção são classificadas para cada um desses segundo a aplicação *Jefit* e a base de dados de exercícios da *American Council on Exercise*.

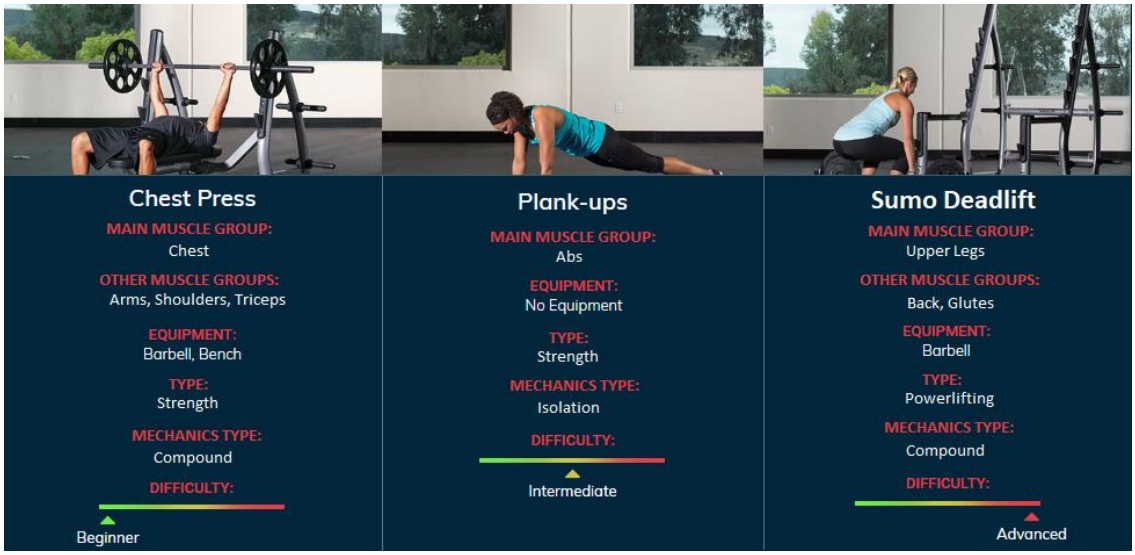


Figura 6 - Caracterização de três exercícios (adaptado de [32][36][37][38][39][40])

2.2 Modelo NCD (New concept development)

O processo de inovação pode ser definido como uma mudança que promove valor, podendo ser dividido em três fases: *Fuzzy Front-End* (FFE), *New Product Development* (NPD) e Comercialização. A primeira parte, FFE, é considerada uma das maiores oportunidades de melhoria do processo geral de inovação [41]. A falta de uma linguagem e vocabulário comuns, que afetavam a capacidade de criar novos conhecimentos e fazer distinções entre diferentes partes do processo, fizeram com que Koen e outros colaboradores desenvolvessem o modelo NCD [41]. Na Figura 7 está presente uma esquematização desse modelo.

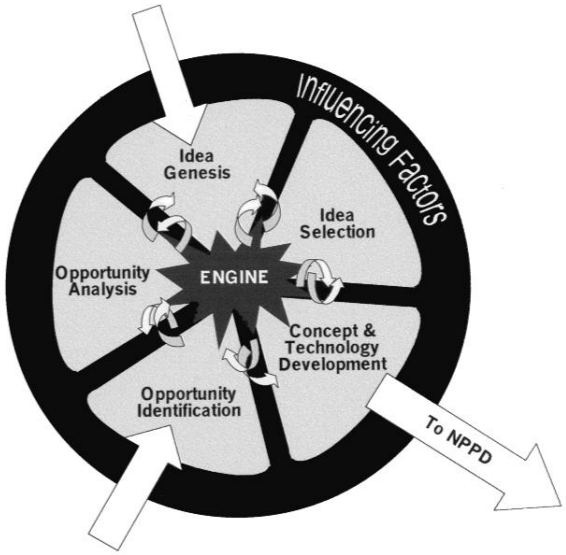


Figura 7 - Modelo NCD (New concept development) [42]

Como ilustrado na Figura 7, o modelo divide-se em três partes: o motor, representa o controlo através de um grupo executivo; os cinco elementos Front-End - Identificação de oportunidade, Análise de oportunidade, Geração e enriquecimento de ideias, Seleção de ideias e Definição do conceito; e os fatores externos que podem influenciar as fases anteriores [42].

2.2.1 Identificação de oportunidade

É nesta fase que a organização identifica as oportunidades que podem ser do seu interesse. Podem ser oportunidades relacionadas com um crescimento específico do mercado, uma vantagem competitiva, uma falha nas tecnologias existentes ou um novo processo de forma a simplificar, acelerar ou reduzir o custo das operações. Normalmente são identificadas através da análise dos mercados, das tecnologias existentes e da opinião/interesse dos clientes, resultando num produto novo ou numa atualização de um produto existente [42].

A oportunidade para a realização deste projeto surge com a perceção de um crescimento no mercado fitness. Aliado a este crescimento, e também ao facto de que, hoje em dia, as pessoas têm em sua posse um *smartphone*, existe uma maior necessidade de informatizar e automatizar a atividade com recurso a aplicações móveis.

Existe, também, uma dificuldade em encontrar apps móveis relativas ao fitness que consigam satisfazer todas as necessidades dos seus utilizadores, isto é, uma aplicação móvel destinada tanto a utilizadores iniciantes como experientes, com funcionalidades desejadas e importantes para cada um desses tipos. Este tópico é analisado ao longo do documento.

Outra oportunidade passa pela análise constante de competidores como Jefit, Keelo, Seven, entre outros, com vista a identificar os seus pontos fracos e efetuar melhorias.

2.2.2 Análise de oportunidade

Nesta fase é importante determinar se as oportunidades enunciadas são viáveis e dignas de continuar. Para isso, são necessárias informações adicionais incluído análise de concorrentes, estudos de mercado, necessidades dos utilizadores que não sejam satisfeitas pelos produtos existentes ou experiências tecnológicas [42].

Nas secções 2.1.1 e 2.1.2 relativas ao Contexto, certos artigos e relatórios foram analisados acerca do mercado e das aplicações fitness. Foi possível concluir que este setor sofreu um crescimento enorme nas últimas décadas, sendo os Estados Unidos da América os líderes deste mercado com 60,9 milhões de ginásios e 38 milhões de membros. Na Europa, a Alemanha é o líder do mercado com 10,6 milhões de membros. Já em Portugal, o fitness cresceu 13% em 2015, somando 730 mil pessoas, o que equivale a 7,1% da população total. Em 2016, o número de sócios dos ginásios cresceu 30%, enquanto o número de novos espaços aumentou 14%.

Relativamente às aplicações de fitness, o seu uso e o número destas nas plataformas *Apple Store* e *Google Play*, cresceu em enorme quantidade. Entre 2014 e 2016 o seu uso aumentou em mais de 330%, e entre 2016 e 2017, cresceu 9%.

No futuro, é previsto que o valor desta indústria continue a crescer proporcionando amplas oportunidades de negócio.

Como dito em cima, existem múltiplas aplicações móveis, contudo, estas apresentam determinadas lacunas. Podem estar relacionadas com a usabilidade, falta de funcionalidades ou uniformização para que utilizadores iniciantes ou experientes as consigam usar.

Depois da análise efetuada, é possível concluir que uma aplicação fitness com um grupo das funcionalidades mais importantes, algumas inovadoras e melhorias gerais tendo em conta os concorrentes, num mercado em contínuo crescimento, poderia ser uma mais valia, tratando-se, por isso, de uma oportunidade a explorar.

2.2.3 Geração e enriquecimento de ideias

Este elemento consiste na criação de ideias através das fases de construção, evolução, combinação, destruição, modificação e atualização. Geralmente este processo é iterativo, podendo passar por diversas alterações à medida que são obtidas novas informações durante a execução de todos os outros elementos do modelo NCD. Ideias provenientes de qualquer pessoa da organização, contacto direto com os clientes, identificação de novas tecnologias e troca de ideias na rede de trabalho podem ser fatores predominantes na geração e enriquecimento de ideias [42].

A ideia do projeto é uma aplicação móvel destinada aos praticantes de fitness e surgiu devido à utilização pessoal de um dos intervenientes deste projeto. De facto, entre as aplicações utilizadas, não foi possível encontrar uma que satisfizesse todos os critérios e funcionalidades requeridas.

Para isso, foi realizada uma sessão de brainstorming com todos os intervenientes do projeto para se perceber qual o sistema que melhor se adequa para colocar em prática:

- Desenvolvimento de uma aplicação móvel suportável para *iOS e Android* e uma aplicação *web*;
- Tornar o uso da aplicação possível para cada tipo de utilizador;
- Desenvolvimento das funcionalidades mais importantes para cada tipo de utilizador;
- Sistema capaz de acompanhar a evolução do utilizador;
- Utilização de técnicas de *machine learning*;
- Utilização de técnicas inteligentes para a geração de recomendações;
- Desenvolvimento de um assistente virtual para esclarecimento de dúvidas;
- Integração com dispositivos *wearables*;

- Desenvolvimento de um sistema capaz de detetar movimentos e aconselhar melhorias.

2.2.4 Seleção de ideias

A seleção de quais as ideias a escolher de forma a haver sucesso comercial é um processo crítico, não existindo um procedimento único para garantir uma boa seleção [42].

Das ideias identificadas anteriormente, foram escolhidas as que mais faziam sentido para a solução pretendida tendo em conta o tempo e, também, as limitações existentes. Como por exemplo a inexistência de numerosos dados para técnicas de machine learning, ou, então, falta de tempo para a realização e análise de um sistema capaz de detetar movimentos. Desta forma, foram selecionadas as seguintes ideias:

- Desenvolvimento de uma aplicação móvel suportável para iOS e Android e uma aplicação *web*;
- Tornar o uso da aplicação possível para cada tipo de utilizador;
- Desenvolvimento das funcionalidades mais importantes para cada tipo de utilizador;
- Sistema capaz de acompanhar a evolução do utilizador;
- Utilização de técnicas inteligentes para a geração de recomendações;
- Desenvolvimento de um assistente virtual para esclarecimento de dúvidas.

2.2.5 Definição do conceito

"O último elemento do modelo envolve o desenvolvimento de um caso de negócio baseado em estimativas de potencial do mercado, necessidades do cliente, requisitos de investimento, avaliações de concorrentes, incógnitas da tecnologia e projeto geral" [42].

Como já constatado, o mercado fitness encontra-se na moda e em constante crescimento, o que advém uma maior procura pelas aplicações móveis fitness.

Pretende-se que, com a construção de um sistema informático, se consiga auxiliar os praticantes deste setor através da monitorização e registo de todas as suas atividades.

O objetivo deste projeto é o desenvolvimento de uma aplicação de software, que tenha como principal propósito providenciar uma experiência personalizada e adaptada a cada tipo de utilizador, com automatização/monitorização da atividade fitness. Esta aplicação, permitirá ao utilizador controlar a sua atividade de forma mais eficaz e eficiente.

2.3 Valor, valor percecionado e valor para o cliente

De seguida, os conceitos de valor, valor percecionado e valor para o cliente serão definidos e aplicados tendo em conta o presente projeto.

2.3.1 Valor

"A criação de valor ocorre no domínio do cliente, enquanto que as empresas no domínio de fornecedoras facilitam a criação de valor, produzindo recursos e processos que representam valor potencial ou valor esperado para os seus clientes." [43]. O valor é visto como um conceito subjetivo que relaciona os benefícios do cliente com os custos para obter esses benefícios.

Este projeto, para o utilizador final (praticante de fitness), pretende dar resposta à necessidade de informatização de toda a sua atividade, fornecendo as funcionalidades mais importantes e algumas inovadoras. Para os ginásios, o valor acrescido ao negócio traduz-se no aumento da retenção de clientes, através de uma ligação mais direta entre o cliente e o *staff* e da disponibilização de um serviço superior de registo e monitorização. Por fim, para os *personal trainers*, será uma plataforma onde poderão orientar e acompanhar a evolução dos seus orientandos.

2.3.2 Valor percecionado

Doriana Morar na sua revisão da literatura acerca do valor para o cliente relativo ao valor percecionado e valor desejado, deparou com diversas definições sobre o valor percecionado, sendo que estas "[…] dependem de termos como a utilidade, valor, benefícios e qualidade" [44]. Por conseguinte, o valor percecionado varia de pessoa para pessoa, sendo comum vários indivíduos atribuírem diferente valores a um mesmo produto ou serviço.

Pretende-se que os utilizadores do sistema a ser desenvolvido o vejam como uma inovação pelos seus métodos de recomendação, integração com o *personal trainer* e acessível a todo o tipo de utilizadores com as funcionalidades fundamentais. No que diz respeito aos ginásios, o valor percebido relaciona-se com a retenção de membros.

2.3.3 Valor para o cliente

O valor para o cliente pode ser entendido como a assimilação da vantagem que se obtém entre os benefícios e os sacrifícios de uma determinada oferta de valor [45].

Tendo em conta a definição, os benefícios e sacrifícios para a presente proposta encontram-se identificados na Tabela 2. Através desta é possível constatar um maior número de benefícios comparativamente a sacrifícios constituindo valor para o cliente.

Tabela 2 – Benefícios e sacrifícios

Benefícios	Sacrifícios
• Facilidade de utilização: interface intuitiva • Inovação: recomendações inteligentes e assistente pessoal • Monitorização da evolução • Personalização e Adaptável ao utilizador • Suporte	• Publicidade • Subscrição

2.4 Proposta de valor

A proposta de valor é uma afirmação através da qual se pretende demonstrar a correspondência entre as necessidades dos consumidores e os benefícios que resultam da utilização do produto. Deverá de ser simples, sucinta, clara e eficaz transmitindo a informação necessária. Posto isto, é definida a proposta de valor para o produto:

A presente solução é uma aplicação móvel/web pertencente ao mercado de fitness que permite a qualquer tipo de praticante deste setor realizar as funcionalidades fundamentais que a sua atividade requer. Disponibiliza meios que facilitam todo o registo das suas atividades, bem como uma análise rápida e eficiente da sua evolução. Além disto, possui técnicas de recomendação com diferentes propósitos, a existência de um *personal virtual assistant* para esclarecimento de dúvidas e a possibilidade de um acompanhamento próximo, caso seja o caso, do *personal trainer*. Relativamente aos ginásios, o valor fornecido refere-se à retenção de membros, devido à oferta das referidas funcionalidades.

Para definir a proposta de valor apresentada, foram respondidas quatro perguntas [46]:

- Para (quais clientes-alvo)?
    - Praticantes de fitness e ginásios.
- O nosso produto é uma?
    - Aplicação móvel/web pertencente ao mercado de fitness.
- Que oferece?
    - Meios ao utilizador para registo das suas atividades, análise rápida e eficiente da sua evolução, técnicas de recomendação com diferentes propósitos, possibilidade de um acompanhamento próximo por parte do *personal trainer* e retenção de membros no que toca aos ginásios.
- O que difere do já existente?
    - Permite a **qualquer** tipo de praticante deste setor realizar as funcionalidades fundamentais que a sua atividade requer. Possui técnicas de IA, um *personal virtual assistant* para esclarecimento de dúvidas e funcionalidades que proporcionam um acompanhamento próximo com o PT.

2.5 Modelo de negócio CANVAS

O modelo de negócio CANVAS, uma ferramenta que foi desenvolvida por Alexander Osterwalder, tendo como objetivo descrever a lógica de como uma organização cria, entrega e captura valor [47]. Está dividido em 9 blocos: Segmentos de clientes, Proposta de valor, Canais, Relacionamento com clientes, Receitas, Recursos-chave, Atividades-chave, Parcerias-chave e Custos.

A Figura 8 apresenta o modelo com a identificação dos vários blocos para a solução a ser desenvolvida.

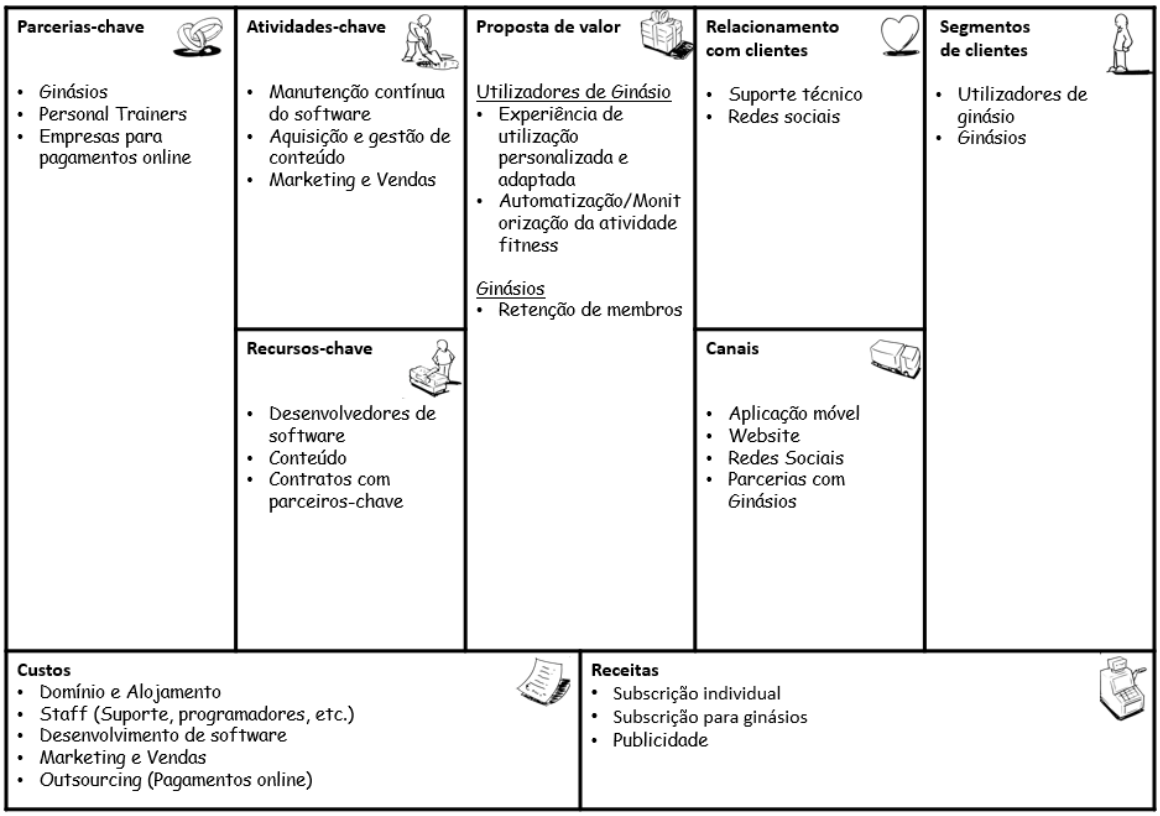


Figura 8 - Modelo de negócio CANVAS

3 Estado de Arte

Neste capítulo os sistemas de recomendação (SR) são contextualizados, descrevendo os vários tipos, os problemas e desafios destes, bem como exemplos de SR relacionados com o fitness. De seguida, são apresentadas aplicações existentes no mercado fitness, juntamente com a sua análise e comparação.

3.1 Sistemas de Recomendação

Os Sistemas de Recomendação consistem em técnicas inteligentes que lidam com diversos e complexos dados tendo como objetivo fornecerem sugestões ao utilizador, auxiliando no processo de tomada de decisão [48].

Existem dois elementos fundamentais em qualquer SR – utilizador e item. Com base nisso, uma definição mais geral destes sistemas foi definida: "Seja C o conjunto de todos os utilizadores; seja S o conjunto de todos os itens possíveis que podem ser recomendados; seja u uma função que mede a adequação do item s para o utilizador c, ou seja, $u: CxS \Rightarrow R$, onde R é um conjunto totalmente ordenado (p.e. números inteiros não negativos ou números reais dentro de um determinado intervalo). Então, para cada utilizador $c \in C$, queremos escolher tal itens $s' \in S$ que maximizem a preferência do utilizador." [49].

De forma a obter essa preferência e garantir a qualidade nas recomendações geradas, é fundamental adquirir informação do utilizador, quer seja direta ou indiretamente. Nos sistemas de recomendação existem duas formas essenciais de recolha de *feedback*: classificações explícitas - consistem na atribuição de um valor numérico, por exemplo, classificação de um item numa escala entre 1 a 5; e classificações implícitas – obtenção de dados adicionais do utilizador através da monitorização da sua interação com o sistema analisando histórico de visualizações, tempo gasto a ver um item, padrões de navegação, entre outros [50].

A utilização deste tipo de sistemas é cada vez mais frequente, sendo utilizado em vários domínios, como *e-commerce*, *e-learning* e serviços de *e-business*, tais como aplicações de música, filmes, livros, documentos, entre outros, evidenciando uma diversidade nos tipos de itens recomendados [48]. A sua adoção encontra-se relacionada, nomeadamente, com o aumento das vendas de produtos, isto é, os sistemas de recomendação geram com critério itens relevantes para os utilizadores, aumentando o volume de vendas e lucro para os comerciantes [51]. Posto isto, de seguida serão descritos vários sistemas de recomendação utilizados por empresas influentes.

- ***Amazon*:** recomenda produtos (p.e. livros, jogos, entre outros) no seu próprio *website* tendo em conta os *ratings* (escala de 1 a 5), o histórico de compras e o comportamento de navegação dos utilizadores, de forma a personalizar a experiência de compra destes. Estas informações são guardadas numa base de dados específica relacionando-as com o utilizador que se encontra autenticado [51];
- ***Netflix*:** recomenda filmes através das classificações atribuídas a estes, das observações e das pesquisas. O perfil do utilizador também é tido em conta priorizando, por exemplo, as categorias favoritas. Além disso, a Netflix fornece explicitamente exemplos de recomendações com base nos filmes que foram vistos. Por exemplo, se o filme *Avatar* tiver sido visto, aparecerá na interface uma secção com o título "Porquê você assistiu *Avatar*" com filmes recomendados tendo como base este. Apresentar explicações das recomendações efetuadas é importante para fornecer ao utilizador uma compreensão de como os itens foram sugeridos, podendo originar, de uma forma mais fácil, um possível interesse nesses [51];
- ***Google News***: recomenda notícias ao utilizador com base no seu histórico de notícias vistas. Essas visualizações ficam associadas ao utilizador através de mecanismos de identificação das contas Gmail. Neste caso, os artigos de notícias são tratados como itens, em que uma visualização destes pode ser vista como uma classificação positiva. Neste cenário, as classificações são obtidas através de ações do utilizador, invés de serem explicitamente atribuídas por este, como visto nos dois casos anteriores [51];
- ***Facebook:*** recomenda potenciais amigos aos utilizadores com o intuito de aumentar o número de ligações sociais, baseando-se em relacionamentos estruturais e não em dados de *ratings*. Por conseguinte, a natureza dos algoritmos subjacentes é diferente comparativamente aos três sistemas descritos anteriormente. Da mesma forma, os objetivos também são distintos. Embora uma recomendação de produto aumente diretamente o lucro do comerciante, facilitando as vendas de produtos, um aumento no número de conexões sociais melhora a experiência de um utilizador numa rede social [51].

Um sistema de recomendação deve utilizar técnicas de filtragem apropriadas para que determinados itens sejam classificados e devidamente recomendados. Na secção 3.1.1 serão descritas as técnicas de SR mais comuns.

3.1.1 Tipos de Sistemas de Recomendação

As técnicas de sistemas de recomendação mais comuns podem ser divididas em seis grupos distintos, descritos nas seguintes secções.

3.1.1.1 Filtragem Colaborativa

A filtragem colaborativa recomenda itens combinando utilizadores que possuam interesses semelhantes. Esta filtragem agrupa o *feedback* do utilizador na forma de classificações para um item específico e encontra correspondência dessas classificações entre os vários utilizadores, de forma a encontrar um grupo destes com preferências semelhantes [52]. Por exemplo, se os utilizadores *X* e *Y* "[…] tiverem a mesma preferência para os itens de *1 a k*, então as probabilidades são boas de terem a mesma preferência para os itens *k+1*" [49]. Desta forma, é definido um perfil de utilizador constituído pelas suas preferências obtidas de forma explícita ou implícita [52].

Existem dois tipos de métodos habitualmente utilizados na filtragem colaborativa:

- **Filtragem colaborativa baseada no utilizador:** dado um novo utilizador *A*, o algoritmo calcula a similaridade deste com todos os restantes utilizadores, encontrando os utilizadores mais semelhantes *B.* Com isto, os utilizadores *B* podem ser usados para fazer previsões de rating para *A* [51]. Ao escolher os utilizadores *B* para o *A*, o algoritmo pode ter uma de duas abordagens - encontrar os *k* utilizadores mais semelhantes (p.e. com o algoritmo *k-nearest neighbours*), sendo *k* um valor inteiro positivo; ou escolher todos os utilizadores com uma classificação de similaridade acima de um limite específico [51];
- **Filtragem colaborativa baseada em itens:** dada uma matriz de classificação na qual as linhas correspondem a um utilizador e cada coluna a um item, o algoritmo determina um conjunto de itens semelhantes ao item de destino. Habitualmente, um conjunto de classificações fornecidas por um utilizador, são usadas para prever se este gostará de um item específico [51].

Este tipo de técnica possui vantagens, nomeadamente, no facto de ser simples e as suas recomendações resultantes, geralmente, são fáceis de explicar. Por outro lado, também apresentam várias desvantagens. A dispersão de dados, mais conhecido por *data sparsity*, é um problema que ocorre quando apenas uma pequena parte dos itens disponíveis é classificada pelos utilizadores, o que faz com que a quantidade de avaliações na matriz de classificação seja insuficiente para o sistema fazer previsões precisas [49]. Por outro lado, *cold start,* em que não existem informações necessárias acerca de um novo utilizador ou item, e *grey sheep*, em que o utilizador não é semelhante a nenhum dos grupos de utilizadores existentes, são outros dois desafios a superar para este tipo de técnica [49].

3.1.1.2 Baseados em Conteúdo

Um sistema de recomendação baseado em conteúdo sugere itens baseando-se na semelhança entre estes e os perfis de utilizadores. O perfil do utilizador é definido tendo em conta o *feedback* e avaliações fornecidas acerca de itens. Perfis de itens também são definidos de forma a caraterizá-los no sistema. Por exemplo, um filme pode possuir determinadas *tags,* sendo estas utilizadas como um atributo (características) do objeto filme [52]. Assim, com base no perfil do utilizador e nos perfis associados aos itens, o sistema de recomendação utiliza mecanismos de correspondência estatísticos ou de aprendizagem computacional para selecionar os itens a recomendar [51].

Por exemplo, considerando uma situação em que o utilizador *U* classificou o filme *F1* considerando-o relevante. Baseando no perfil do item, neste caso, no filme, um filme *F2* que possua *tags* semelhantes ao filme *F1* (p.e. comédia) poderia ser recomendado ao utilizador [51].

Esta técnica de recomendação não contabiliza as preferências dos utilizadores "semelhantes" (filtragem colaborativa), não exigindo que as preferências de um grupo de utilizadores aumentem a precisão das recomendações [51].

Os métodos baseados em conteúdo têm algumas vantagens em realizar recomendações para novos itens, principalmente quando os dados de classificação sobre estes não são suficientes. No entanto, este tipo de sistema também possui limitações. Uma delas reside no fato da possibilidade de geração de recomendações *overspecialized*, isto é, gerar itens muito semelhantes àqueles que o utilizador já possuí conhecimento ou gerar recomendações sempre iguais. Outra limitação encontra-se relacionada com o *cold start*, ou seja, "mesmo que os métodos baseados em conteúdo sejam eficazes em fornecer recomendações para novos itens, eles não são eficazes em fornecer recomendações para novos utilizadores" [51].

3.1.1.3 Baseados em Conhecimento

"O processo de recomendação é realizado com base nas semelhanças entre os requisitos do utilizador e os perfis dos itens […]" [51]. É perguntado, então, explicitamente a este as preferências pretendidas, de forma a obter os requisitos mencionados, resultando num maior controlo dos utilizadores no processo de recomendação.

Assim, terão de existir regras específicas de domínio para relacionar os requisitos do utilizador com os itens. Essas regras podem assumir a forma de restrições, por exemplo: Filmes da Marvel em que o ator Robert Downey Jr participa. Desta forma, as recomendações baseadas em conhecimento pressupõem uma estrutura capaz de guardar estas relações e que permita a consulta e alteração de modo a possibilitar novas recomendações [51].

Tanto os sistemas de recomendação baseados em conteúdo, como os baseados em conhecimento dependem significativamente do perfil dos itens. Portanto, algumas desvantagens existentes no sistema de recomendação da secção 3.1.1.2, também se encontram presentes neste, nomeadamente, a possibilidade de geração de recomendações *overspecialized*. Estes dois SR, por vezes, são considerados "primos" relativamente à

abordagem tendo em conta o conteúdo, sendo que a "[…] principal diferença é que os sistemas baseados em conteúdo aprendem com o histórico do utilizador, enquanto que os sistemas de recomendação baseados em conhecimento recomendam com base na especificação dos utilizadores tendo em conta as suas necessidades e interesses" [51].

3.1.1.4 Demográficos

Os sistemas de recomendação demográficos são caracterizados por mapearem os utilizadores com base nos seus atributos pessoais. De facto, a partir de um diálogo interativo entre o sistema e os utilizadores, estes são agrupados em modelos de utilizador. De seguida, as recomendações são sugeridas utilizando técnicas baseadas num respetivo perfil demográfico criado [51].

A utilização deste tipo de sistemas aumenta significativamente o poder de outros sistemas de recomendação, como os híbridos, combinando este tipo de SR com o de conhecimento, aumentando a sua performance [51].

3.1.1.5 Matrix Factorization

Os sistemas de recomendação baseados em factorização de matrizes têm como principal característica reduzir os problemas de falta de dados e escalabilidade [53]. Uma das formas de solucionar esse problema de dados é a incorporação de *feedback* implícito obtido através da análise do comportamento do utilizador [53]. Assim, o presente tipo de SR assume-se diferente dos outros tipos mencionados anteriormente. De facto, invés de fazer sugestões tentando encontrar utilizadores ou itens semelhantes, utiliza um modelo matemático para fazer previsões [53].

Existem múltiplos algoritmos baseados em modelo para fazer tais previsões. Uma das abordagens mais comuns é o *Singular Value Decomposition* (SVD). Resumidamente, este algoritmo fatoriza uma matriz de itens do utilizador em matrizes mais pequenas, que podem ser usadas para preencher as classificações em falta.

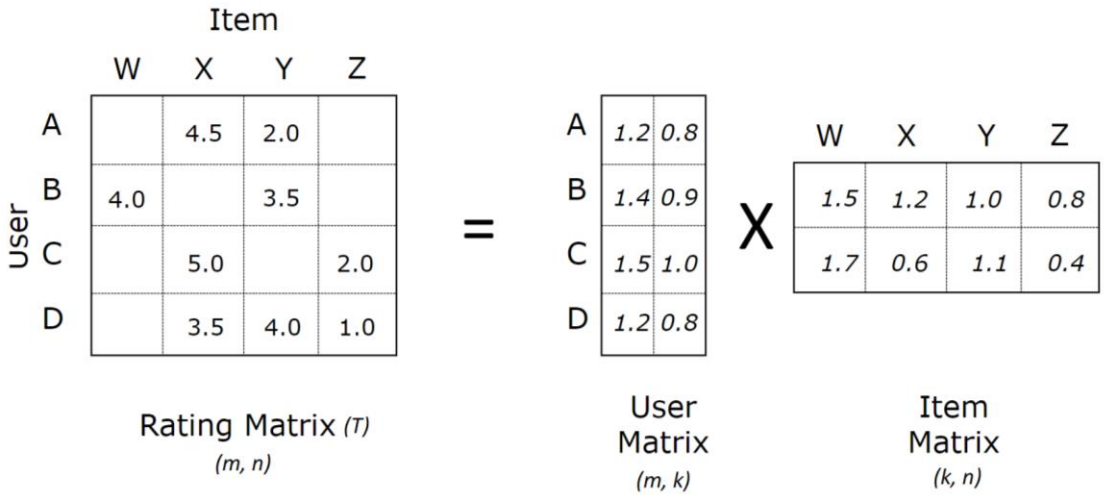


Figura 9 - Exemplo da factorização de matriz (adaptado de [54])

Na Figura 9 encontra-se ilustrado um exemplo da factorização de matriz. De facto, a matriz T com dimensionalidade de (m,n) pode ser vista como um produto escalar entre as outras duas matrizes, com dimensões de (m,k) e (k,n) [53]. Portanto, a factorização de matriz pode ser matematicamente expressa da seguinte forma:

$$\hat{r}_{ui} = q_i^T p_u \quad (1)$$

Na expressão temos que cada item i é associado a um vetor $q_i \in \mathbb{R}^f$, sendo f a dimensão do espaço de fator latente (*latent features*) mapeado pelo modelo (utilizando os utilizadores e itens), e cada u associado a um vetor $p_u \in \mathbb{R}^f$ [53].

De uma forma prática, utilizando o exemplo da Figura 9, é possível prever a classificação que o utilizador A daria ao item W através do cálculo do produto escalar entre os vetores (1,2; 0,8) e (1,5; 1,7), resultado o valor de 3,16.

3.1.1.6 Híbridos

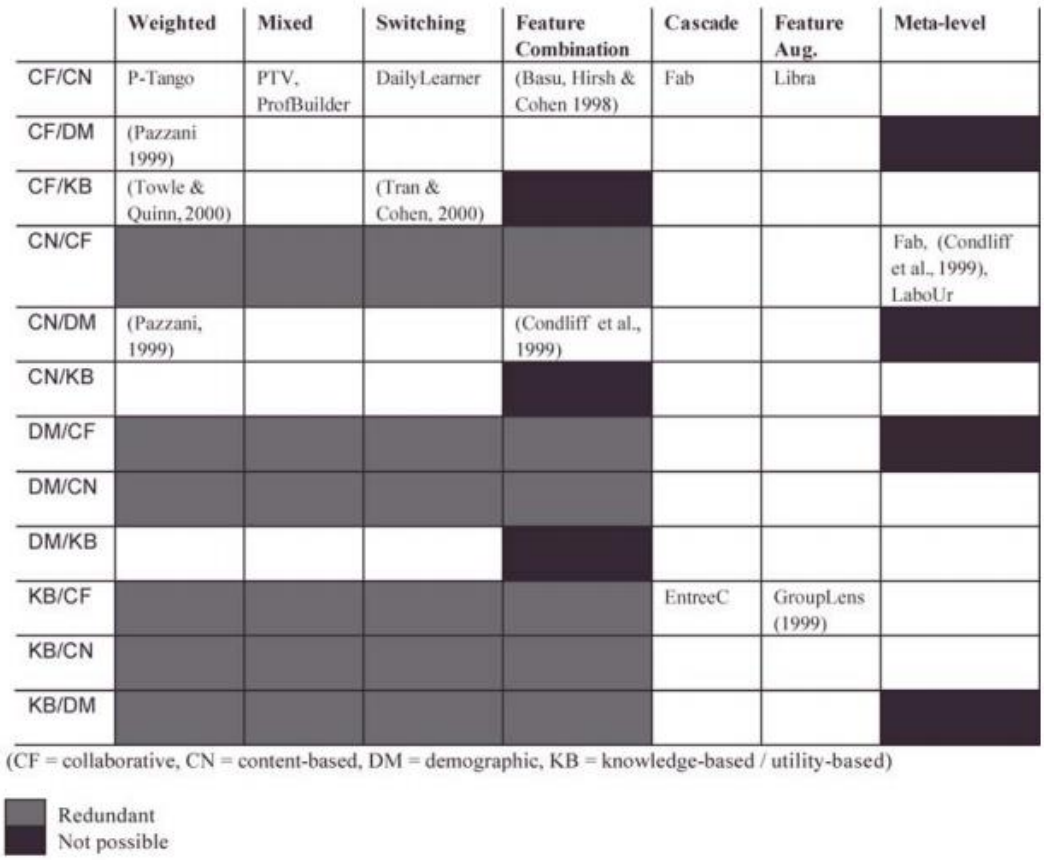
Os sistemas híbridos são combinações de várias características de diferentes sistemas de recomendação, com o objetivo de construir um sistema mais robusto [51]. Na verdade, combinando vários sistemas, é possível remover diversas desvantagens que um SR usado separadamente possui, aumentando o desempenho e a *performance* [52].

Na Tabela 3 são ilustrados alguns dos métodos de combinação de sistemas de recomendação existentes.

Tabela 3 - Métodos híbridos (adaptado de [55])

Método híbrido	**Descrição**
Weighted	As classificações de várias técnicas de recomendação são combinadas para produzir uma única recomendação.
Switching	O sistema alterna entre as técnicas de recomendação, dependendo da situação atual.
Mixed	Recomendações de várias técnicas de recomendação são apresentadas ao mesmo tempo.
Feature combination	Características de diferentes bases de dados são agrupadas para um único algoritmo de recomendação.
Cascade	Uma técnica de recomendação corrige as recomendações obtidas por outra.
Feature augmentation	O *output* de uma técnica é usado como *input* de outra.
Meta-level	O modelo aprendido por uma técnica é usado como *input* para outro.

No estudo de Robin Burke [55] acerca dos sistemas de recomendação foi definida uma tabela onde é possível visualizar a possibilidade de combinação entre os sistemas de recomendação mencionados anteriormente, tendo em conta os métodos apresentados na Tabela 3. Essa tabela encontra-se ilustrada na Figura 10.



	Weighted	Mixed	Switching	Feature Combination	Cascade	Feature Aug.	Meta-level
CF/CN	P-Tango	PTV, ProfBuilder	DailyLearner	(Basu, Hirsh & Cohen 1998)	Fab	Libra	
CF/DM	(Pazzani 1999)						
CF/KB	(Towle & Quinn, 2000)		(Tran & Cohen, 2000)				
CN/CF							Fab, (Condliff et al., 1999), LaboUr
CN/DM	(Pazzani, 1999)			(Condliff et al., 1999)			
CN/KB							
DM/CF							
DM/CN							
DM/KB							
KB/CF					EntreeC	GroupLens (1999)	
KB/CN							
KB/DM							

(CF = collaborative, CN = content-based, DM = demographic, KB = knowledge-based / utility-based)

Redundant
Not possible

Figura 10 - Possíveis sistemas de recomendação híbridos [55]

Como é possível visualizar na Figura 10 existem 60 espaços não redundantes, sendo algumas combinações impossíveis de serem realizadas. Existe, tendo em conta as áreas brancas, 53 sistemas de recomendação híbridos possíveis de combinar. Desses 53, apenas 14 foram explorados. Contudo, o estudo de Robin Burke foi publicado em 2002, sendo que, muito possivelmente, outras combinações já foram testadas, senão todas.

3.1.1.7 Comparação das Técnicas de Recomendação

Os sistemas acima mencionados diferem em várias vertentes, como o objetivo, os dados de *input*, o processo geral, as vantagens, as desvantagens, entre outros.

Na Tabela 4 são comparadas as diferentes técnicas de recomendação, comparando o seu objetivo principal e as diferentes formas de *input*.

Tabela 4 – Objetivos dos diferentes tipos de SR (adaptado de [51])

Tipo de SR	**Objetivo**	**Input**
Filtragem Colaborativa	Sugere recomendações combinando utilizadores com interesses semelhantes.	Classificações do utilizador + classificações da comunidade
Baseados em Conteúdo	Sugere recomendações com base no conteúdo (atributos dos itens) tendo em conta as classificações e comportamentos providenciados.	Classificações do utilizador + atributos do item
Baseados em Conhecimento	Sugere recomendações com base nos requisitos ou restrições definidas pelo utilizador.	Requisitos do utilizador + atributos do item + conhecimento do domínio
Demográficos	Sugere recomendações com base no perfil demográfico do utilizador.	Atributos pessoais do utilizador + atributos do item
Matrix Factorization	Sugere recomendações utilizando um modelo matemático.	Classificações dos utilizadores

Na Tabela 5 são apresentadas as vantagens e desvantagens para cada técnica de recomendação.

Tabela 5 - Vantagens e desvantagens dos SR (adaptado de [53] e [55])

Tipo de SR	**Vantagens**	**Desvantagens**
Filtragem Colaborativa	• Conhecimento de domínio desnecessário; • A qualidade da recomendação melhora ao longo do tempo; • Apenas necessário *feedback* implícito.	• Problema *cold start;* • Problema *gray sheep;* • Problema *sparsity;* • A qualidade está dependente de inúmeros dados.
Baseados em Conteúdo	• Recomendação para um novo item; • Conhecimento de domínio desnecessário; • A qualidade da recomendação melhora ao longo do tempo; • Apenas necessário *feedback* implícito.	• Problema *cold start*; • Problema *over specialization;* • A qualidade está dependente de inúmeros dados.
Baseados em Conhecimento	• Remoção do problema *cold start*; • Remoção do problema *over specialization;* • Adaptável a alterações de preferências.	• Necessário conhecimento do domínio; • A qualidade da recomendação não melhora ao longo do tempo.
Demográficos	• A qualidade da recomendação melhora ao longo do tempo; • Apenas necessário *feedback* implícito.	• A qualidade da recomendação não melhora ao longo do tempo.

Matrix Factorization	• Remoção do problema *sparsity;* • Funciona com *datasets* de tamanho considerável; • Fornece abordagens escalonáveis.	• Problema *cold start.*

3.1.2 Problemas e Desafios

Na secção 3.1.1 foram apresentados os diferentes tipos de sistema de recomendação, bem como os seus vários problemas ou desvantagens. Na presente secção, esses problemas e outros considerados comuns, serão detalhados.

3.1.2.1 Cold Start

Este problema ocorre quando novos utilizadores ou itens são adicionados à base de dados. Devido à falta de dados, nomeadamente, as poucas preferências dos utilizadores e as poucas classificações dos itens, leva a que os gostos dos novos utilizadores não consigam ser previstos, nem os novos itens recomendados, originando recomendações menos precisas. Os SR mais afetados com este problema são os de filtragem colaborativa e os baseados em conteúdo [51][52].

Para resolver este problema, destacam-se alguns procedimentos: solicitar ao utilizador a classificação de alguns itens; solicitar ao utilizador que escolha explicitamente os seus gostos; e solicitar itens ao utilizador com base nas suas informações demográficas [52].

3.1.2.2 Privacy

"Os sistemas de recomendação baseiam-se no feedback dos utilizadores, o que pode ser implícito ou explícito. Esse feedback contém informações importantes sobre os interesses do utilizador e pode revelar informações sobre as suas opiniões políticas, orientações sexuais e preferências pessoais. Em muitos casos, essas informações podem ser altamente confidenciais, o que leva a preocupações com a privacidade. [51]". Este problema impede o avanço dos algoritmos de recomendação, visto que a disponibilidade dos dados é crucial [51].

Alguns procedimentos foram desenvolvidos para resolver este problema, como: mecanismos de criptografia de forma a fornecerem recomendações personalizadas sem envolverem utilizadores e outras entidades; e técnicas de desordem aleatória que permitem que os utilizadores divulgam os seus dados privados sem expor a sua identidade [52].

3.1.2.3 Overspecialization

Este problema pode ser originado tendo em conta os poucos dados presentes. Os SR podem recomendar itens que embora estejam relacionados com o utilizador, não sejam novos para este [52].

De forma a resolver este problema, a adição de aleatoriedade através da utilização de algoritmos genéricos que trazem diversidade às recomendações, podem gerar mais facilmente itens novos e aleatórios [52].

3.1.2.4 Grey Sheep

Este problema ocorre no sistema de recomendação de filtragem colaborativa quando um utilizador não é semelhante a nenhum dos outros existentes, não sendo possível incluí-lo num grupo [52].

Os SR baseados em conteúdo podem resolver esse problema aquando da sugestão dos itens, explorando o perfil pessoal do utilizador e o conteúdo dos itens recomendados. Para a identificação dos utilizadores que não apresentam semelhanças com outros, podem ser usadas técnicas de *clustering*, incluindo *k-mean clustering*, de maneira a serem identificados e separados dos restantes utilizadores [52].

3.1.2.5 Sparsity

A enorme quantidade de itens existentes leva a que apenas uma pequena parte destes seja classificada pelos utilizadores, resultando numa matriz dispersa que origina recomendações menos precisas e dificulta a realização de previsões sobre os itens [52].

Para lidar com esta situação, várias abordagens podem ser usadas: utilização de um modelo de recomendação multidimensional; utilização de técnicas de *Singular Value Decomposition* (SVD), em que uma matriz é decomposta em duas matrizes; e utilização de um SR demográfico ou baseado em conteúdo [52].

3.1.2.6 Scalability

Os sistemas de recomendação necessitam de uma grande quantidade de dados dos utilizadores e de itens para se tornarem eficientes. Contudo, quando essa quantidade é muito extensa, torna-se difícil processar os dados em grande escala. Um exemplo é a Amazon que recomenda mais de 18 milhões de itens para mais de 20 milhões de clientes [52].

Este problema pode ser resolvido usando algoritmos de *clustering* que pesquisam utilizadores em pequenos *clusters*, invés de pesquisar em toda a base de dados ou utilizar técnicas de SVD reduzindo a dimensionalidade [52].

3.1.2.7 Synonymy

Este problema surge quando um item é representado através de dois ou mais nomes diferentes ou quando os itens apresentam significados semelhantes. Nestes casos, o SR não consegue identificar se os termos representam itens diferentes ou o mesmo item. Por exemplo, um sistema de recomendação de filtragem colaborativa irá interpretar “carro” e “automóvel” de forma diferente [52].

Podem ser utilizadas técnicas de ontologia – representação através de um grafo constituído pelos diversos conceitos e suas relações; técnicas de SVD; e técnicas de *Latent Semantic Indexing* (LSI) - encontra os relacionamentos ocultos (latentes) entre as palavras (semântica) para melhorar a compreensão da informação (indexação) -, como forma de remover o problema identificado no parágrafo anterior [52].

3.1.3 Sistemas de recomendação de Fitness

Os sistemas de recomendação relacionados com o domínio da saúde, particularmente na área do fitness e bem-estar geral, ainda denotam ser relativamente escassos [56]. Para ser possível caracterizar alguns exemplos destes sistemas, foram procurados artigos na biblioteca ACM *Digital Libray*, *B-on* e *Google* com as *keywords* "*Fitness recommender system*". O processo de escolha foi relativamente simples. De facto, foram lidos os *abstracts* e escolhidos os artigos mais relevantes.

Nas próximas secções serão apresentados os SR escolhidos.

3.1.3.1 PRO-Fit

PRO-Fit [57] é uma *framework* de assistência de *fitness* capaz de monitorizar de forma inteligente as atividades do utilizador, agrupando dados dos dispositivos *wearable*, sincronizando o calendário do utilizador e recomendando treinos personalizados baseados nas suas atividades passadas e de outros utilizadores semelhantes, nas suas preferências, no seu estado físico e na disponibilidade.

A *framework* é constituída por 2 módulos: um classificador de atividades em que algoritmos de *machine learning* são utilizados para construir modelos que classificam as atividades dos utilizadores em tipos específicos; e um mecanismo de *ranking* e recomendação que utiliza um SR do tipo híbrido para realizar recomendações tendo em conta dados do perfil do utilizador, tais como o estilo de vida (p.e. sedentário ou ativo), a idade, o peso, os objetivos (p.e. emagrecer) e as preferências (p.e. atividades físicas favoritas ou nível de intensidade)[57] .

"Por exemplo, a PRO-Fit pode recomendar uma aula de ioga de 1 hora no centro de *fitness* da Universidade durante o horário de almoço para o utilizador A, que é funcionário desta, e 20 minutos de corrida no parque mais próximo para o utilizador B, que é aluno e tem 30 minutos de intervalo entre as aulas [57]".

A arquitetura implementada desta *framework* encontra-se ilustrada na Figura 11.

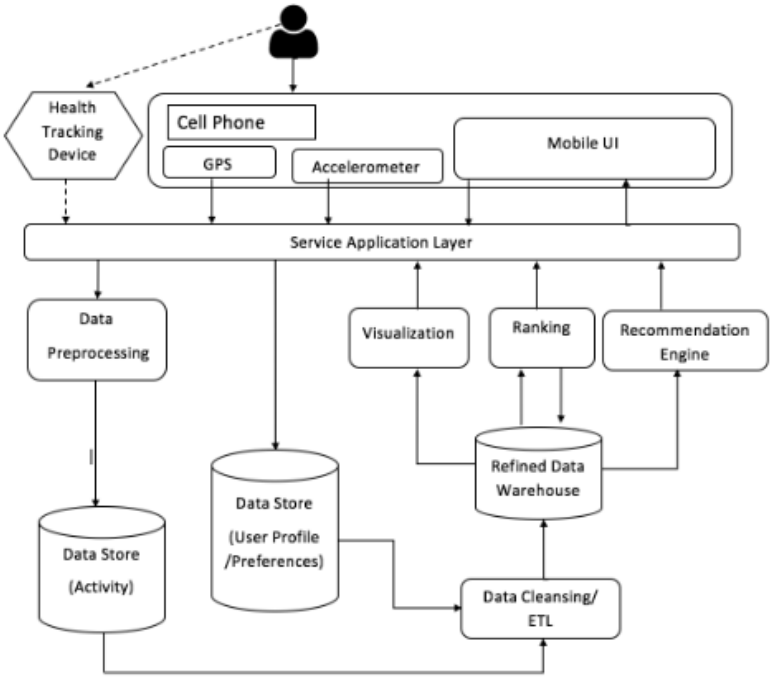


Figura 11 – Arquitetura PRO-Fit [57]

De seguida, será descrito resumidamente as responsabilidades de cada componente tendo em conta o artigo *PRO-Fit: A personalized fitness assistant framework* [57]:

- ***Health Tracking Device:*** aplicação móvel disponível nos sistemas operativos *Android* e *iOS* para que os utilizadores forneçam determinados dados e recebam recomendações;
- ***Service Application Layer:*** disponibiliza uma interface para interação entre a aplicação e o exterior. O componente é dividido numa API Rest responsável pela lógica de negócio e numa interface que lida com a gestão de notificações;
- ***Data pre-processing and classification:*** responsável por quatro tarefas principais - obter os dados vindos da aplicação móvel; pré-processamento dos dados que serão usados como *input* para o processo de classificação; armazenamento dos dados; e classificação dos dados, principalmente das atividades dos utilizadores em diversas categorias;
- ***Data Stores:*** são armazenados dois tipos principais de dados - os dados da atividade física do utilizador, que são usados na fase de aprendizagem do modelo em *machine learning*; e os dados do perfil do utilizador. O componente *Data Warehouse* extrai dados do módulo *Data Cleansing* com o intuito de convertê-los para um esquema que será útil para a análise destes;
- ***Visualization:*** "[…] responsável por gerar gráficos ou tendências que fornecem informações valiosas aos utilizadores.";
- ***Fitness Session Recommendation Engine:*** responsável por recomendar sessões de *fitness* aos utilizadores com base no perfil destes e nas suas atividades;

- ***Social Ranking Recommendation Engine:*** recomenda amigos aos utilizadores integrando com as suas redes sociais.

3.1.3.2 The Runner

The Runner [58] é um sistema de recomendação que tem como principal objetivo fornecer informações relacionadas com treinos e planos de nutrição a atletas amadores e profissionais de corrida, baseando-se no seu perfil, preferências e objetivos. Por exemplo, para correr 5 km em 25 minutos, a aplicação recomenda um programa de treino a ser realizado durante duas semanas. Um dos treinos propostos incluí correr 1 km durante 4 minutos e 20 segundos no asfalto.

Esta solução disponibiliza uma aplicação web desenvolvida numa arquitetura em três camadas, com interações bidirecionais entre elas, como ilustrado na Figura 12.

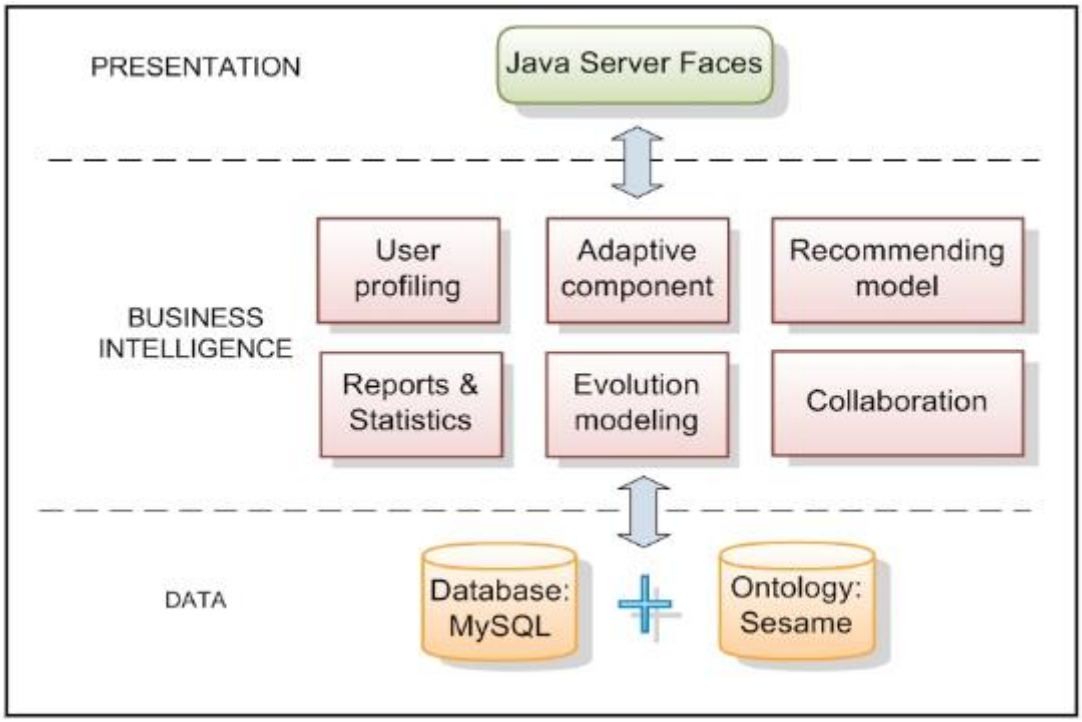


Figura 12 - Arquitetura The Runner [58]

A camada de apresentação é representada pela aplicação *web* e consiste em todas as interfaces gráficas que ajudam o utilizador a aceder às informações desejadas. Foi implementada usando o *Java Server Faces* [59], devido à sua escalabilidade e adaptação para sistemas web [58].

A camada de lógica de negócio é composta por vários componentes: componente de recomendação; componente adaptável responsável por personalizar a dieta e os exercícios de cada utilizador; componente para definição do perfil do utilizador; componente de colaboração entre os utilizadores da comunidade; componente responsável por fornecer

estatísticas e relatórios sobre o desempenho atual; e componente responsável por modelar a evolução em questões de tempo de uso [58].

Por fim, a camada de dados é constituída por dois componentes principais: uma base de dados em MySQL responsável por guardar os dados relativamente aos perfis dos utilizadores, dietas recomendadas e exercícios de treino; e uma ontologia geral (explicada na secção 3.1.2.7) que inclui três ontologias básicas. Uma destas encontra-se relacionada com o domínio da nutrição que inclui conceitos para todos os tipos de alimentos e nutrientes, possibilitando a visualização do menu e seus valores nutricionais aquando da recomendação da dieta. Outra ontologia relaciona-se com a medicina desportiva usada para recomendar possíveis tratamentos de lesões. A última ontologia refere-se aos atletas para que seja possível desenvolver uma rede social [58].

3.1.3.3 Fitness that Fits

Fitness that Fits [56] recomenda vídeos de exercícios baseando-se: no *dataset* Youtube-8M, que contém milhares de informações acerca dos milhões de vídeos presentes no Youtube; nas avaliações dos treinos de condição física fornecidas pelos utilizadores; e nas preferências e histórico de visualizações recentes do utilizador.

Um esquema do processo de recomendação da presente plataforma encontra-se na Figura 13.

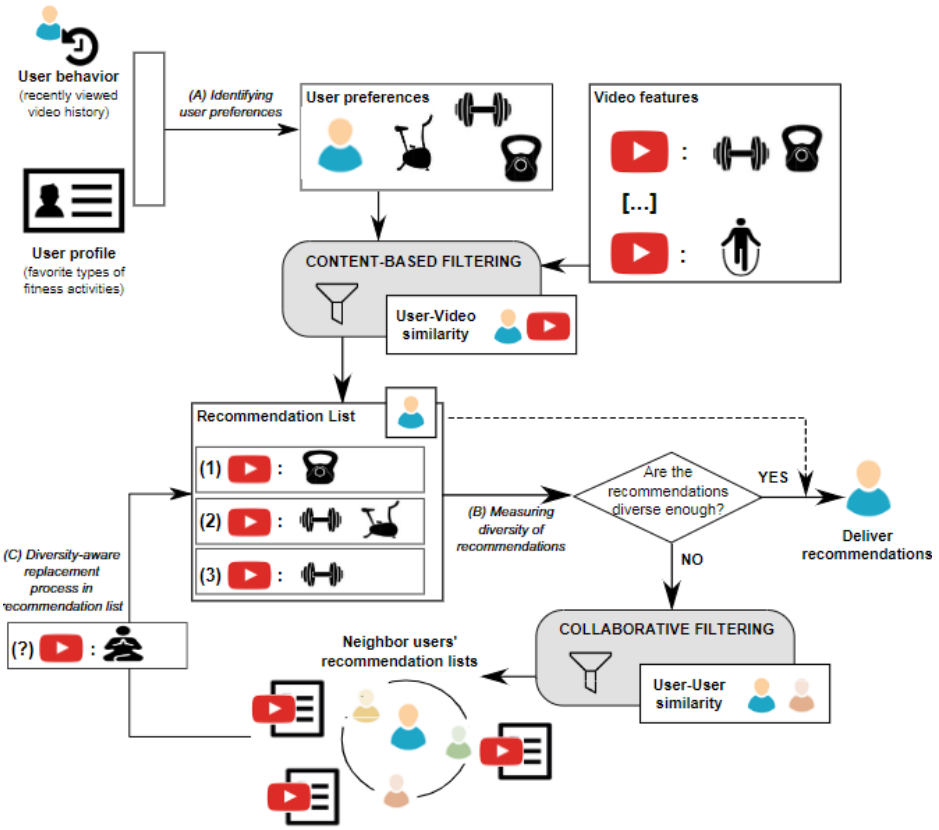


Figura 13 - Esquema do sistema de recomendação [56]

O SR ilustrado na Figura 13 consiste num tipo de sistema híbrido constituído por um sistema baseado em conteúdo e outro de filtragem colaborativa assente no algoritmo *k-nearest neighbours.* Existem 3 fases fundamentais neste processo:

1. **(A) Identificar as preferências do utilizador:** aqui são tomadas em conta duas informações fundamentais – o perfil do utilizador obtido através da especificação das atividades de *fitness* preferidas (p.e. ciclismo) e o comportamento deste capturado pela análise do histórico de visualizações recentes dos vídeos. Depois de obtidos estes dados, são aplicadas técnicas de SR baseadas em conteúdo para gerar uma lista de recomendações de vídeos. No entanto, esta pode não ser a lista final [56];
2. **(B) Avaliar a diversidade de recomendações:** nesta fase, a lista de recomendações gerada é avaliada quanto à sua diversidade. Caso seja diversa, o resultado é sugerido de imediato ao utilizador, caso contrário, são aplicadas técnicas de filtragem colaborativa baseadas nos utilizadores "vizinhos" para diversificar ainda mais a informação extraída da lista de recomendação [56];
3. **(C) Processo de substituição da lista de recomendações:** um procedimento iterativo é aplicado para diversificar a lista de recomendações. Em cada iteração, um dos vídeos recomendados ao utilizador é substituído por outro proveniente de uma lista de recomendações de utilizadores "vizinhos", com base em diferentes etapas. O procedimento termina quando as recomendações sejam diversas o suficiente, ou um número máximo de iterações previamente definido seja excedido [56].

3.1.3.4 Conclusão

Apesar dos sistemas de recomendação ligados ao fitness serem escassos, tal como já referido, os existentes, pelo menos os analisados, utilizam várias técnicas de inteligência artificial. Relativamente à arquitetura, foi possível constatar uma divisão de vários componentes, cada um com uma responsabilidade (p.e. recomendações, estatísticas, entre outros). Também apresentam mais do que uma base de dados, sendo, normalmente, uma responsável pelos dados do utilizador e outra pela lógica de negócio.

Sistemas de recomendação de filtragem colaborativa e baseados em conteúdo são geralmente utilizados para sugerir as respetivas recomendações, aliados a técnicas de ontologias e *k-nearest neighbours.*

Esta pesquisa é fundamental para a análise, decisão e posterior implementação de um sistema de recomendação com os requisitos desejados.

3.2 Soluções existentes

A forma como as pessoas têm usado os dispositivos móveis tem mudado ao longo do tempo, devido, essencialmente, ao rápido desenvolvimento da tecnologia e da internet sem fios [21]. Por conseguinte, o uso de aplicações móveis tem sido cada vez mais constante, tal pode ser comprovado pela celebração da Apple no que toca ao alcance de 50 mil milhões de aplicações transferidas em 2013, tendo a Google, na mesma altura, 48 mil milhões [60]. Este progresso fez com que o mercado de aplicações móveis fitness sofresse um elevado crescimento [21][25]. De 3 312 283 aplicações disponíveis na Apple Store [61], 100 315 dizem respeito à categoria *Health & Fitness* (3,03% do total) [62], já na Google Play, de 2 579 894 aplicações [63], a categoria *Health & Fitness* conta com 85 300 (3,31% do total) [64].

Estas aplicações possuem diversos benefícios, nomeadamente, podem representar uma ferramenta importante para os profissionais de saúde ou fitness no controlo de múltiplos parâmetros de saúde e condição física, estabelecimento de metas e ajuda no alcance de objetivos nos seus pacientes [5][24]. Por outro lado, a nível pessoal, podem: ajudar o utilizador a monitorizar e gerir os exercícios, a dieta, o peso ou os objetivos; disponibilizar um feedback rápido e confiável, incentivando as pessoas a alcançar metas definidas, lembrando, sempre que necessário, quando algo estiver em falta; e partilha nas redes sociais [5].

Nas duas próximas secções estão presentes os critérios de seleção das soluções, bem como os critérios para avaliá-las. De seguida, as várias aplicações serão descritas. Por fim, será feito uma descrição pormenorizada de diversos aspetos das diferentes apps e apresentada uma tabela comparando as principais funcionalidades das aplicações analisadas.

3.2.1 Critérios de seleção das soluções

Esta análise baseou-se na recolha de informação de nove aplicações fitness, e a sua escolha incidiu-se nos seguintes critérios:

i. Popularidade de pesquisa destas aplicações na Apple Store e Google Play;

ii. Apenas aplicações grátis que podem possuir planos de subscrição;

iii. Disponível para download tanto na Apple Store como na Google Play;

iv. Classificação superior a 4,4, numa escala entre 0 a 5, tanto na Apple Store como na Google Play;

v. Número de classificações superior a 2700 tanto na Apple Store como na Google Play;

vi. Diversidade de funcionalidades disponibilizadas através da leitura das descrições das aplicações;

vii. Popularidade destas aplicações em alguns blogs de análise [65][66][67][68].

Em primeiro lugar foram encontradas múltiplas aplicações usando a Apple Store e a Google Play utilizando filtros de ordenamento por popularidade, classificação e número de classificações. De seguida, foram utilizados critérios mais específicos, como: apenas aplicações grátis, disponíveis para download nos dois sistemas operativos, classificação (rating) superior a 4,4 e número de classificações superior a 2700 tanto na Apple como na Google. Na última fase foram lidas as descrições das aplicações encontradas por fim a detetar apps com diversidade de funcionalidades, e pesquisados alguns blogs de análise no Google com as *keywords* "The best fitness apps", com o objetivo de ler *reviews* de aplicações usadas ou analisadas.

Contudo, existem duas aplicações que não preenchem na totalidade os critérios apresentados. Intensity possui uma classificação de 4.1 na Apple Store e 4.3 na Google Play, com 27 e 244 números de classificações na Apple e Google, respetivamente. Assim, a escolha desta aplicação deu-se ao facto de providenciar um aspeto que nenhuma das outras o faz: é uma app dedicada a *Powerbuilding*.

A outra exceção diz respeito à app AmazinGym. Como será explicado na secção dedicada a esta, trata-se de uma aplicação disponibilizada num ginásio físico, e para que fosse possível a sua análise, visto que todas as apps deste género apenas são acessíveis para os próprios membros do ginásio, uma exceção foi feita.

3.2.2 Critérios de avaliação das soluções

Foi feito o *download* de todas as aplicações para tornar a análise destas mais fácil, sendo descrito, para cada uma delas, uma apresentação geral, funcionalidades disponibilizadas, pontos positivos ou inovadores, e pontos considerados como negativos. O *download* foi feito através da Apple Store sendo que a análise da usabilidade e funcionalidades foi realizada através do sistema operativo iOS, não sendo garantido que as funcionalidades descritas estejam também disponíveis para Android, ou o inverso, isto é, algumas podem estar disponibilizadas para Android e para iOS não.

Foram analisados determinados artigos com o intuito de criar um conjunto de critérios de avaliação para comentar sobre os aspetos positivos e negativos: Mary, Gwin, Cheney, e Wann conduziram um estudo que identificaram quais as funcionalidades mais valorizadas por estudantes pertencentes a um colégio [7]; Jang Yul realizou uma pesquisa para determinar quais as intenções dos utilizadores em adotar aplicações fitness [21]; Preethi Sama e outros colaboradores [24], John Higgins [5], Anna Tubek e Mariusz Duplaga [69] analisaram e avaliaram um conjunto de aplicações fitness nos seus estudos.

Da análise dos artigos foi possível apurar as funcionalidades mais valorizadas pelos utilizadores e as consideradas essenciais em apps de fitness que serão usadas como critérios de avaliação: definição de objetivos (*setup*); visualização de planos ou exercícios já pré-definidos; criação dos próprios treinos e exercícios; registo da atividade; treino em tempo real; consulta da atividade feita (p.e. no calendário); monitorização do progresso e histórico;

visualização de estatísticas; geração de planos de treino (recomendações); existência de um perfil para visualizar os seus dados; possibilidade de personalização da aplicação (configurações); integração com personal trainer; feed de atividade; consulta de planos alimentares; integrações (p.e. aplicação de saúde); partilha em redes sociais; e suporte a várias idiomas.

A par destes critérios, a possível falta de funcionalidades, a facilidade em usar, pormenores da interface, aspetos que não são comuns na maioria das aplicações, a existência de funcionalidades para motivação do utilizador e pormenores de desempenho, também serão servidos como critérios, visto serem, do mesmo modo, particularidades essenciais em aplicações deste tipo.

De seguida, são apresentadas as diversas aplicações de fitness analisadas.

3.2.3 Seven

Seven [70] foi considerada uma das melhores aplicações de fitness na Apple Store no ano de 2017 [71]. Além de estar disponível para smartphones Android e iOS, também é possível usá-la no iPad e Apple Watch, bem como visualizar informações sobre esta no website dedicado.

A aplicação é maioritariamente conhecida pelos seus treinos de 7 minutos baseados em estudos científicos com o intuito de proporcionar o máximo de benefício no menor tempo possível. Também apresenta outras funcionalidades, tais como: visualização de planos de treino pré-definidos, criar próprio plano de treino, existência de um feed para os membros da aplicação, receção de notificações (hora do treino e novos seguidores de um plano criado), várias integrações (p.e. siri [72] para processamento de alguns pedidos e aplicação Saúde da Apple [73] para incorporar determinadas métricas do corpo, tais como peso e altura), e efetuar um treino em tempo real, isto é, realização de diversos exercícios com o auxilio de um contador e detalhes específicos para o exercício corrente (explicação e GIF de demonstração).

Relativamente a pontos positivos menciona-se o conceito inovador “desafio de 7 meses”. Neste desafio o utilizar terá de realizar todos os treinos programados até ao final de um tempo (7 meses), caso falhe um destes treinos uma “vida” é perdida, num máximo de três. Todos os meses as vidas são repostas e na condição de perder as três vidas, o desafio é reiniciado e o respetivo progresso começara a 0%. O presente desafio atua como um método de motivação para a realização da atividade fitness.

Na Figura 14 está ilustrada a interface do desafio anteriormente mencionado.

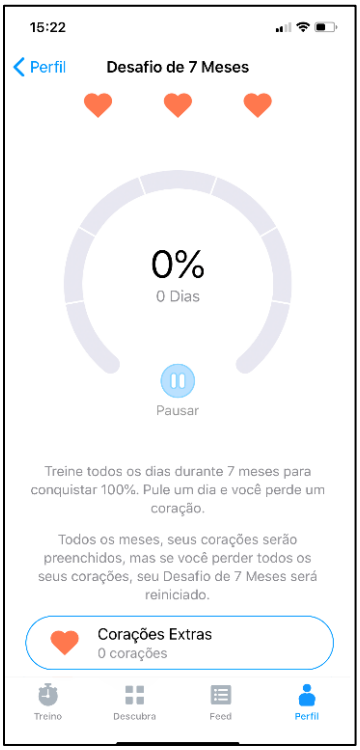


Figura 14 - Desafio de 7 meses

Além disso, a variedade de configurações, a interface para criar um plano de treino e as conquistas (à medida que a atividade na aplicação aumenta, são recebidas determinadas conquistas, p.e. “Um novo começo”, “Criando um hábito”, entre outras), também são pontos positivos a realçar.

Como pontos negativos enumeram-se os seguintes: necessária subscrição para ter acesso a grande parte das funcionalidades, não permite a adição de qualquer tipo de exercício existente na base de dados ao plano de treino, apenas os já visualizados até ao momento, poucas estatísticas ou quase nulas, tal como monitorização e recomendações, não permite a visualização de exercícios e o website disponível tem o único intuito de comercializar o produto.

3.2.4 Desafio Fitness de 30 Dias

Desafio Fitness de 30 Dias foi considerada pela Google Play [74] a melhor aplicação fitness do ano de 2016 e a melhor app de autoaperfeiçoamento. Providencia, essencialmente, a realização de treinos em casa, sendo estes elaborados por profissionais da área.

Na presente aplicação é possível visualizar diversos planos de treinos, realizar os treinos em tempo real, visualizar logs da atividade com indicação da duração do treino, integrações (p.e. aplicação de Saúde da Apple [73]), lembrete para a realização do treino e estatísticas, tais

como, duração dos treinos nos últimos 7 dias e visualização de um gráfico com evolução do peso.

Como pontos inovadores enumeram-se os múltiplos planos alimentares e os desafios de 30 dias de abdominais, corpo interior, entre outros, que ajudam a melhorar a forma física e a saúde [74].

Contudo, a presente aplicação é escassa em algumas funcionalidades. Na verdade, não possui um website para comercialização do produto ou então para uso por parte dos utilizadores, não permite a criação de planos de treino ou de exercícios, não tem monitorização de dados nem estatísticas, não permite a visualização de exercícios e não possui acompanhamento dos dados métricos do corpo (peso, altura, percentagem de gordura, entre outros).

3.2.5 Intensity

Intensity [75] é uma aplicação de gestão de exercícios baseada em progressão e projetada para tipos de treino de força. Utiliza a cloud como principal serviço para guardar os dados, sendo possível aceder a estes via dispositivos Android, iOS, Windows e também via web browser.

A presente aplicação é direcionada para utilizadores mais experientes no que toca à atividade de ginásio, ajudando a:

- **Melhorar a força:** acesso aos planos de treino mais populares, tais como: 5/3/1, Starting Strength, Stronglifts 5x5, The Texas Method, Smolov, entre outros;
- **Ser flexível:** tanto em bons como maus dias, a aplicação ajuda a melhorar o nível de performance dos treinos;
- **Ter uma visão:** possível visualizar progresso, identificar tendências e melhorar as áreas consideradas mais fracas;
- **Conectar:** adicionar amigos, partilhar treinos, visualizar recordes e disputar pelo primeiro lugar nas classificações relacionadas com quem alcançou um maior número de peso.

Adicionalmente, é possível customizar planos pré-definidos, adicionar novos exercícios e visualizar múltiplas estatísticas, por exemplo, visualização em gráfico do máximo/mínimo de volume alcançado, crescimento do peso e média, para um exercício.

A variedade de cálculos feitos pela aplicação , a importação e exportação de dados, o método *calibrating*, isto é, na adição de um exercício ao plano, é calibrado automaticamente o peso e o número de repetições tendo em conta os objetivos especificados, e o facto do website dedicado à aplicação não ter o único intuito de comercializar o produto, mas sim, um espaço onde é possível visualizar artigos de fitness e realizar determinadas funcionalidades (visualizar logs diários, estatísticas, entre outros), são pontos a realçar.

Por outro lado, a inexistência das funcionalidades treino em tempo real, geração de recomendações e suporte a diferentes linguagens, são alguns dos pontos menos positivos.

Além disso, o facto do público alvo da aplicação serem os utilizadores "experientes" que praticam ginásio, carece, também, por ser um ponto negativo, visto que utilizadores menos experientes não terão tanto interesse em usar a presente app, por ser demasiada especifica.

3.2.6 Freeletics Bodyweight

"Freeletics [Bodyweight] é uma aplicação de treinos de alta intensidade" [76][77] tendo como objetivo o alcance dos resultados esperados num curto espaço de tempo por parte dos seus utilizadores. A aplicação e todo o seu domínio envolvente (website e comercialização Apple Store e Google Play) destacam a funcionalidade *Coach*, um personal trainer digital implementado com o uso de Inteligência Artificial.

Além dessa funcionalidade e como primeira experiência na aplicação, é solicitado por esta, a realização de um setup contendo campos como sexo, objetivos, nível de condição e detalhes pessoais (nascimento, altura e peso). Depois desta fase é possível, então, executar as funcionalidades mais comuns, como: visualização de planos de treino e exercícios, integração com aplicação Apple Saúde, Facebook, Google e Spotify, visualização de um feed com toda a atividade realizada e publicações dos administradores da aplicação, entre outras.

Apresenta alguns aspetos inovadores:

- **Locais de treino:** a aplicação permite a visualização de diversos locais nas proximidades com indicação das pessoas que lá treinam e da atividade que estão a realizar;
- **Pontos:** cada plano de treino ou exercício possuem pontos e à medida que estes são feitos, os pontos referidos são acumulados no perfil. Atua como um fator motivacional;
- **Coach**[1]**:** como referido em cima, trata-se de um personal trainer digital que tira partido de técnicas de inteligência artificial.
    - **Adquirir conhecimento:** é fundamental colecionar dados acerca do utilizador e da sua atividade para que a informação a ser gerada seja a mais real possível. Para isso, são utilizadas várias formas para a obtenção destes:
        - **Setup:** todos os dados inseridos (altura, peso, objetivos, entre outros) são tomados em conta;
        - **Final do treino:** sempre que o utilizador acaba uma atividade, o coach aprende com o feedback dado por este (Figura 15) e, também, com o tempo decorrido na realização do exercício;
        - **Preferências do utilizador:** os exercícios favoritos e também equipamentos (barra, bench, entre outros) são usados para a AI.

[1] não foi possível uma análise detalhada a esta funcionalidade pois é necessária a realização de uma subscrição para ter acesso a esta. Contudo, foram utilizadas informações presentes no website da Freeletics Bodyweight [9] e da aplicação.

- **Funcionalidades:** orientação para vários tipos de atletas (iniciantes e avançados) e semana coach – "Uma semana do Coach é o seu plano de treino personalizado com instruções para a sua semana. Quando a semana acabar, o Coach reconhece a atividade de treino, a performance e o feedback para otimizar e personalizar a próxima semana" [78].

Figura 15 - Feedback fornecido pelo utilizador [77]

No entanto, existem alguns aspetos considerados não tão positivos. Nomeadamente, no setup - poderiam existir mais campos, ou então, se por uma decisão de não tornar essa fase tão extensa, alterar alguns dos atuais para outros mais importantes (p.e. número de dias de treino por semana ou grupo de músculos a serem exercitados) para que estes possam ser contabilizados na lógica subjacente às recomendações com base em técnicas de Inteligência Artificial – e ausência de funcionalidades importantes: calendário com logs das atividades realizadas; estatísticas; indicação de qual o músculo ou grupo de músculos estão a ser exercitados; e registar plano de treino/exercício e posterior partilha.

Um outro ponto negativo prende-se ao facto da necessidade da realização do download dos vídeos dos exercícios, mencionado, também, por Roger Moraes [79], autor de um dos comentários mais relevantes na página da aplicação no Google Play, "download dos vídeos da aula do dia seguinte poderiam ser feitos automaticamente no final de cada treino, evitando percas de tempo com download manual um a um".

3.2.7 AmazinGym

AmazinGym [80] é um ginásio localizado na zona de Matosinhos que disponibiliza aos seus sócios uma plataforma móvel onde estes podem realizar diversas funcionalidades. A respetiva aplicação foi desenvolvida pela empresa Virtuagym [81] que conta com mais de 4500 clientes espalhados por todo o mundo, tais como empresas direcionadas ao fitness, universidades ou clientes singulares.

Na presente aplicação é possível gerir a marcação de aulas, avaliações com o personal trainer e consultas de nutrição. Além disso, também é permitido realizar as funcionalidades mais comuns que um praticante de ginásio está acostumado a fazer, nomeadamente, visualização do seu perfil com informações estatísticas (p.e. "calorias gastas"), visualização gráfica do seu progresso ao nível da percentagem de gordura e outros indicadores, visualização de planos de treino e respetivos exercícios, e existência de um feed onde é possível partilhar a sua atividade.

Há que realçar determinados pontos ao nível da interface, tais como: possibilidade de filtrar exercícios de treino através da seleção de uma parte do corpo (ver Figura 16), visualizar progresso usando vários indicadores (percentagem de massa gorda, índice de massa corporal, percentagem de água no corpo, entre outros) e facilidade em adicionar ou visualizar plano de treino.

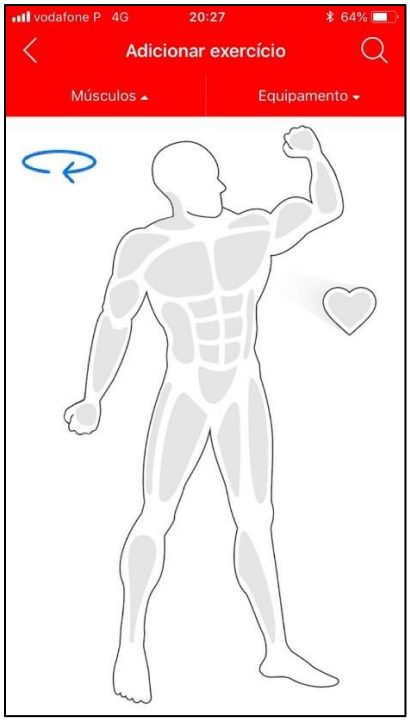


Figura 16 - Selecionar parte do corpo

Além disso, a aplicação suporta várias integrações. De facto, é possível integrar com os dispositivos *NEO Health Go*, *NEO Health Pulse* e *NEO Health One* com o objetivo de

acompanhar em tempo real o batimento cardíaco, e com a balança *NEO Health Onyx* para que dados como o peso, altura, entre outros, sejam automaticamente sincronizados com a aplicação, não sendo necessária a adição manual.

Porém, determinados pormenores ao nível da interface e a inexistência de recomendações, são pontos a realçar negativamente. Como aspetos ao nível da interface, enumeram-se: dificuldade em encontrar certas funcionalidades, como o registo de um plano de treino e a marcação de aulas, filtragem de planos de treino muito extensa não possuindo parâmetros importantes (p.e. grupo de músculos exercitados) e ausência do plano de treino aquando da visualização de um determinado desafio proposto.

3.2.8 J&J Official 7 Minute Workout

J&J Official 7 Minute Workout [82] é uma aplicação que se baseia na ciência de trabalhar em qualquer lugar e a qualquer hora, de forma simples e rápida, não necessitando do deslocamento ao ginásio nem, excetuando raros casos, de qualquer tipo de equipamento. Possui o conceito de 7 minutos, isto é, a app disponibiliza workouts com essa duração, caracterizados pela sua intensidade e intervalos, com exercícios de força, cardio e flexibilidade.

Dispõe de múltiplas funcionalidades, particularmente: visualização de exercícios de treino categorizados de 4 maneiras distintas – *total body, lower body, upper body* e *core* -, criação de um plano de treino em que é possível adicionar três exercícios para cada uma das categorias mencionadas, realização de um plano de treino pré-definido, personalizado ou smart workout em tempo real e notificação de inatividade ou lembrança.

De realçar toda a interface, pois demonstra ser intuitiva e *user friendly*. A possibilidade de escolher uma música do iTunes durante a realização do treino em tempo real também é um aspeto inovador. Por último, é importante o realce da recomendação de um plano de treino por parte da aplicação (*smart workout*). Aqui, é tomado em conta todos os dados fornecidos pelo utilizador: tipo de nível, feedback dado aquando da finalização dos *workouts* e registo de *likes* ou *dislikes* nos exercícios.

No entanto, o escasso número de exercícios e *workouts* comparativamente com outras aplicações do género, 72 e 22, respetivamente, é um aspeto desfavorável. Além disso, a pouca liberdade de criar um plano, ou seja, apenas são permitidos adicionar 3 exercícios em categorias especificas e a ausência de funcionalidades, como é o exemplo dos logs de treino e estatísticas, fazem parte desta análise não tão positiva.

3.2.9 Keelo

Keelo [83] é um programa direcionado para treinos de alta intensidade e condição física que promove resultados mesuráveis e reais. Segundo os websites dedicados à comercialização da aplicação, Apple Store [84] e Google Play [85], várias entidades de renome caracterizaram a app como: “Best Personal Training Apps” (Men's Health), “Top 12 Health & Fitness Apps”

(Forbes), “17 Apps for Resolutions” (Elite Daily), “Top New Health & Fitness Apps” (Google) e “Stronger Abs, Best New Updates” (Apple), mostrando, desta forma, a sua reputação e aplicabilidade.

Relativamente a funcionalidades, a app dispõe de diversos planos de treino especializados em força, condição física e cardio, sendo possível, para cada um, adicionar como favorito, partilhar nas redes sociais, visualizar gráfico com músculos que serão exercitados, dicas de treinador, adicionar música e visualizar leaderboard (utilizadores com mais pontuação). Essa pontuação, correspondente ao fitness score, mede a capacidade do utilizador de realizar atividades em períodos diferentes. Para isso, é importante registar a atividade para que a pontuação seja calculada de forma mais precisa possível. Também é permitido a realização de treinos em tempo real com a ajuda de um contador e um vídeo demonstrativo, bem como integração com a aplicação de Saúde da Apple e aplicação Google Fit pertencente à Android, para sincronização das colorias e atividades feitas.

De destacar certas funcionalidades: possibilidade de filtrar planos de treino através de inúmeros critérios – equipamento, modalidade, área funcional e excluindo certos exercícios -, visualização de um gráfico com as áreas funcionais exercitadas (ver Figura 17), integração com o *Coach*, isto é, possibilidade de enviar email a este com determinadas dúvidas ou conversar através de um chat (funcionalidade premium), e, por último, recomendação de um plano de treino usando inteligência artificial. Para a geração desse plano o algoritmo contabiliza o grupo de músculos exercitados, os exercícios e os planos de treino feitos, e o tempo dedicado a estes. Num futuro, é calculado que o algoritmo também tome em conta mais especificidades do utilizador, como a modalidade [86].

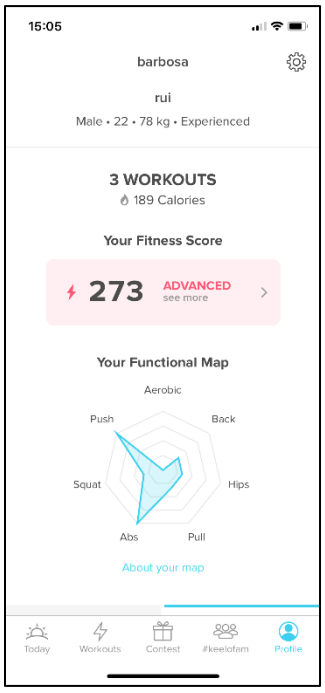


Figura 17 - Áreas funcionais exercitadas

Por outro lado, a falta de funcionalidades mais comuns como visualização de um calendário com logs dos treinos realizados, possibilidade de criar um plano de treino, visualização de exercícios pré-definidos ou estatísticas, e o facto da foto tirada no fim de um *workout* servir única e exclusivamente para partilha nas redes sociais, não permitindo guardá-la para posterior visualização do progresso, constituem pontos negativos tendo em conta a presente análise.

3.2.10 Strong

Strong [87] é uma aplicação orientada para qualquer tipo de praticante de ginásio, desde iniciantes a experientes, disponibilizando o necessário para o registo diário dos *workouts* realizados. Possui alguns planos de treino pré-definidos, mas assume-se como uma app completamente adaptável a qualquer tipo de *workout* criado.

Possui funcionalidades, como é o exemplo de: visualização de múltiplos exercícios de cardio e força, bem como detalhes destes acerca da sua correta execução, possibilidade de criar os próprios exercícios e planos, podendo adicionar notas, fotos ou partilhar em diferentes locais (p.e. Facebook ou mensagens), utilizar um temporizador para registar e contar o tempo de descanso, e cálculo de *warm up* e *plate* que informa qual a quantidade de peso para o aquecimento e qual a quantidade de peso para colocar na barra, respetivamente.

A aplicação possui vários aspetos positivos e alguns inovadores. Como é o exemplo da possibilidade de adição de shortcuts para reconhecimento por parte da Siri. Aqui, fornecendo um dado atalho (p.e. *Start Rui routine workout*), é feita a integração com a Siri e quando esse conjunto de palavras é pronunciado, a app realiza determinada lógica já configurada subjacente ao atalho escolhido, que, para o exemplo fornecido, a aplicação seria automaticamente aberta e o *workout* "Rui" seria iniciado. A visualização de estatísticas também demonstra ser muito completa, pois é permitido acompanhar o progresso de inúmeras métricas através do auxílio de um gráfico. Métricas como: melhor série, volume total, máximo de repetições consecutivas e total de repetições relativamente a um exercício, calorias por semana, peso, gordura corporal e circunferências de partes do corpo (pescoço, ombros, peito, entre outros). Por fim, existe suporte para vários tipos de exercícios, incluindo exercícios de aquecimento, supersets e exercícios agrupados, o que não é uma funcionalidade muito comum na maioria das aplicações fitness.

Contudo, existem certos pormenores a apontar negativamente. A inexistência da possibilidade de realização de um treino em tempo real é uma delas. De facto, a funcionalidade que se assemelha é a existência de um temporizador para acompanhar o tempo de descanso entre exercícios, o que levanta várias dúvidas acerca da sua utilidade e benefício, visto que tal finalidade já é permitida pelo iOS usando a aplicação relógio. O preço de 3€/mês ou 27€/ano para usufruir da opção PRO também pode ser considerado um ponto negativo, uma vez que as funcionalidades disponibilizadas para essa opção não justificam o preço. Na verdade, são funcionalidades como visualização de múltiplas estatísticas associadas a métricas que a opção normal não possui e possibilidade de criar mais do que 3 rotinas de

treino. Porventura, com a adição de recomendações, o preço poderia fazer mais sentido. Por último, tanto na versão normal como na PRO a aplicação disponibiliza um número pequeno de planos de treino pré-definidos, no total 5.

3.2.11 Jefit

JEFIT [88] é uma aplicação de fitness que para além de estar disponível para os sistemas operativos Android e iOS, também se encontra acessível através de um Android Smartwatch, Apple Watch ou via website. Com mais de 7 milhões de downloads, JEFIT atua com o objetivo de ser a melhor app para treino pessoal e registo diário da atividade, facilitando a maneira como os seus utilizadores acompanham o seu progresso e interajam com outros entusiastas do fitness.

A página principal da app disponibiliza cinco pontos principias:

- **Discover:** existência de um feed onde é possível visualizar a atividade feita, estados publicados, amigos adicionados e alterações nas métricas do corpo. Este pode ser filtrado para que apareçam apenas as publicações do próprio utilizador, as dos seus amigos, as da comunidade e as populares;
- **Workout:** visualização de um plano ativo escolhido pelo utilizador organizado por dias da semana ou então por números de dias, por exemplo, a segunda-feira é composta por um grupo de exercícios para treino de pernas ou o dia 3 é composto por um grupo de exercícios para treino de peito, respetivamente. Esse plano de treino pode ser escolhido entre os pré-definidos da aplicação, os partilhados por outros utilizadores ou os criados. Para criar um plano é necessário introduzir dados como o nome, nº de dias por semana, nível de dificuldade, tipo de plano, tipo de dia (numérico ou dia da semana), adicionar exercícios tendo em conta o nº de dias escolhido, reordenar e agrupar exercícios. Além da funcionalidade de visualização do plano ativo, também é possível ao utilizador visualizar múltiplos planos de treino. Estes podem ser filtrados pelo tipo (geral, força, ganhar massa muscular e perder gordura), dificuldade e organizados por popularidade, mais visto e mais transferido;
- **Exercises:** o primeiro passo é selecionar qual a parte do corpo pretendida, e, de seguida, os exercícios respetivos aparecerão. Também é possível filtrar por equipamento, favoritos e criados. Na adição de um exercício é preciso facultar um nome, músculo principal, tipo de registo (repetições, peso e repetições, cardio, entre outros) e segundo e terceiro músculo, se for o caso. Já nos detalhes de um exercício é disponibilizado um GIF com a demonstração da sua realização, tags associadas, orientações e histórico da sua atividade;
- **Logs:** toda a atividade registada encontra-se visível num calendário, onde é possível ver os dias em que certa atividade foi realizada. Escolhido um dia, pormenores dessa atividade aparecerão, bem como dados do corpo e fotos do progresso;
- **Profile:** encontra-se divido em duas partes: as várias opções a selecionar e a informação/atividade pessoal. Na primeira, é possível visualizar notificações

existentes, ver e adicionar amigos, acompanhar o relatório de treinos composto por gráficos relativos aos exercícios e atividade feita, e configurações – adicionar lembrete associado a um dia e hora, valores default de descanso, repetições e sets, e integração com a app Saúde da Apple. Por outro, na segunda parte, fazem parte os dados estatísticos do corpo e o progresso em forma das fotos adicionadas.

Adicionalmente, a aplicação suporta a realização de uma funcionalidade semelhante ao treino em tempo real. Isto é, depois da realização de um set de um exercício, um contador é iniciado, e no fim da realização deste, é passado, automaticamente, para outro.

A aplicação disponibiliza um número extenso de planos de treino e exercícios, 3855 e mais de 1300, respetivamente, o que é um aspeto positivo. Além disso, no início, são disponibilizados pequenos tutoriais relacionados com certas funcionalidades para que o utilizador consiga maximizar o uso dessas. Por fim, salientar a possibilidade de partilha nas redes sociais, e noutros locais, nomeadamente, na aplicação, para que outros utilizadores consigam ver e, eventualmente, tirar partido deles, bem como a vasta personalização na criação de exercícios ou planos de treino. É o exemplo de, na criação de um exercício, é pedido o tipo de registo, sendo que esta escolha terá um impacto na interface deste, pois se for escolhido cardio, na altura de registo será pedido uma duração (hh-mm-ss), por outro lado, se invés de cardio for escolhido repetições, será unicamente pedido o número respetivo.

Apesar dos pontos realçados, a aplicação denota determinados aspetos inconvenientes. A possibilidade de início de um workout composto por zero exercícios, o facto de ao selecionar a opção “more options” ao criar um plano apagar o formulário construído e o tempo de espera para conexão com o servidor, por vezes, não demonstrar ser o mais notável, demorando alguns segundos, constituem problemas ao nível da performance. A funcionalidade semelhante ao treino em tempo real referida em cima, poderia ser mais trabalhada. Na verdade, é necessária muita interação com o utilizador para a introdução de dados, fazendo com que o treino não seja fluido. Relativamente à interface, se algumas já foram aqui elogiadas, esta não é o caso, de facto, demonstra não ser muito intuitiva. Por último, não foi percetível em qualquer finalidade da app a utilidade do *Fly Mode*, este pode ser selecionado, mas não é entendida a usa utilidade.

3.2.12 Conclusão

Através da análise das aplicações anteriores é possível concluir que estas apresentam diferentes públicos alvos: (i) praticantes do ginásio ou fora deste (casa, locais públicos, entre outras) que pretendem realizar exercícios relacionados com condição física, musculação e levantamento de peso, (ii) praticantes do ginásio mais experientes caracterizados por necessitarem de funcionalidades mais especificas e não tão detalhadas, focados, essencialmente, no ganho de força e alcance de um corpo maior e mais forte, e (iii) praticantes de um ginásio especifico, sendo estes sócios.

Muitas das funcionalidades são comuns entre as várias aplicações analisadas, tais como: visualização de planos de treino e exercícios pré-definidos, registo da atividade, treino em tempo real, estatísticas, calendário com logs da atividade realizada, suporte a medidas de unidades diferentes (lib/inch ou kg/cm), perfil com informação pessoal e notificações. Contudo, algumas destas encontram-se implementadas de forma diferente, nomeadamente, ao nível da interface. Como é o exemplo do treino em tempo real, sendo que em certas aplicações é necessário um comportamento mais ativo no que toca ao registo de dados, principalmente, relacionado com o nº de repetições e peso, ao invés de outras, que tendo em conta a informação fornecida acerca do treino a realizar, não solicitam qualquer tipo de registo, sendo o treino mais fluido. Contudo, na generalidade, essas funcionalidades comuns acabam por fornecer o essencial.

Em todas as aplicações existiram aspetos positivos e negativos, identificados através dos critérios definidos na secção 3.2.2, que serão, tanto uns como os outros, tomados em conta para a implementação da presente solução. Dos referidos positivamente, destacam-se: conceitos de desafios, pontos e conquitas, criados com o intuito de motivar o utilizador para a prática de fitness; pormenores da interface ao nível de planos de treinos, exercícios e treino em tempo real; riqueza de configurações e de filtros para localizar os exercícios ou planos de treinos pretendidos; feed contento atividade pessoal e dos restantes utilizadores; e cálculos específicos. Por outro lado, os aspetos mencionados negativamente encontram-se relacionados com determinadas lacunas ao nível do desempenho, como o tempo de espera de conexão ao servidor, falta de intuição e usabilidade em algumas funcionalidades e, sobretudo, falta de funcionalidades, tais como: diversidade de estatísticas, calendário com registo da atividade e criação dos próprios exercícios ou planos.

Ao nível de integrações, pode-se concluir, que, todas as aplicações analisadas possuem pelo menos uma. Esta prende-se ao registo ou então à partilha da atividade, de planos ou de exercícios criados. Tanto uma como a outra utilizam, predominantemente, o Facebook ou então a Google para esses efeitos. Para o registo, possuindo uma conta numa destas plataformas, esse processo torna-se mais ágil sendo apenas necessário o início de sessão e os dados necessários relativos a esta são integrados na aplicação. Por outro lado, contendo os dados da sessão, a partilha em redes sociais ou Google, torna-se mais simples.

Duas das integrações mais usadas estão relacionadas com a aplicação de Saúde da Apple e com a Google Fit da Android, com o intuito da realização da sincronização com os dados pessoais, como a data de nascimento ou sexo, e, sobretudo, informação das calorias gastas e da atividade realizada, podendo ver o próprio progresso num único local. Embora não estejam integrados em todas as aplicações, os dispostos de *Heart Rate Monitor* são cada vez mais utilizados, existindo no mercado uma grande diversidade. São utilizados com a intenção de medição da frequência cardíaca para orientação do treino. Por vezes pode ser indicado manter o ritmo cardíaco relativamente baixo para perder gordura, enquanto, noutras ocasiões relativas com o beneficio de saúde ou criação de resistência, aumentar o ritmo cardíaco poderá ser o ideal [89]. Os dispositivos mais comuns presentes nas aplicações para a

realização desta integração são o Apple Watch, NEO Health Pulse e Garmin. Outras integrações existentes, mas não tão comuns, dizem respeito à Siri para a adição de shortcuts e à plataforma Spotify para a adição de música na realização do treino.

Um aspeto curioso é que de entre as nove aplicações analisadas, três destas já fazem uso de recomendações utilizando técnicas de Inteligência Artificial, sendo que, para a utilização desta funcionalidade, é necessária a realização de uma subscrição possuindo um determinado valor monetário. Embora esta funcionalidade já exista em determinadas aplicações, a sua utilidade é comum – gerar um plano de treino -, não havendo, pelo menos nas apps estudadas, uma recomendação diferente. Comparativamente aos dados que são contabilizados para a execução do algoritmo, são denotas semelhanças, mas também algumas diferenças. De facto, as aplicações Freeletics Bodyweight e J&J Official 7 Minute Workout tiram partido dos dados do setup, feedback, preferências e likes/dislikes providenciados pelo utilizador. Já a app Keelo conta com o grupo de músculos exercitados, exercícios e planos de treino realizados, e tempo dedicado a estes.

Tabela 6 - Comparação das aplicações com base nos critérios definidos

	Seven	**Desafio Fitness**	**Intensity**	**Freeletics**	**Amazin**	**J&J**	**Keelo**	**Strong**	**Jefit**
Registo/ Login	Sim	Não	Sim	Sim	Sim	Não	Sim	Sim	Sim
Setup	Sim	Não	Não	Sim	Não	Sim	Sim	Não	Não
Visualização de planos de treino pré-definidos	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Visualização de exercícios pré-definidos	Não	Não	Não	Sim	Não	Sim	Não	Sim	Sim
Criar próprio plano	Sim	Não	Sim	Não	Sim	Sim	Não	Sim	Sim
Criar próprio exercício	Não	Não	Não	Não	Não	Não	Não	Sim	Sim
Possível registar treinos e atividade	Não	Não	Sim	Sim	Sim	Não	Não	Sim	Sim
Treino em tempo real	Sim	Sim	Não	Sim	Sim	Sim	Sim	Não	Sim
Logs com calendário	Sim	Sim	Sim	Não	Sim	Não	Não	Sim	Sim
Dados do corpo e seu histórico	Não	Sim	Sim	Não	Sim	Não	Não	Sim	Sim
Adicionar Fotos para ver progresso	Não	Não	Não	Não	Não	Não	Sim	Não	Sim
Estatísticas	Sim	Sim	Sim	Sim	Sim	Não	Não	Sim	Sim
Recomendações	Não	Não	Não	Sim	Não	Sim	Sim	Não	Não

Perfil	Sim	Não	Sim	Sim	Sim	Não	Sim	Sim	Sim
Configurações	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Notificações	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Integração com Personal Trainer	Não	Não	Não	Não	Não	Não	Sim	Não	Não
Feed	Sim	Não	Sim	Sim	Sim	Não	Sim	Não	Sim
Planos alimentares	Não	Sim	Não	Não	Não	Não	Não	Não	Não
Importar | Exportar dados	Não	Não	Sim	Não	Não	Não	Não	Sim	Não
Integrações	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Partilha nas redes sociais	Sim	Sim	Sim	Não	Sim	Sim	Sim	Sim	Sim
Várias linguagens	Sim	Sim	Não	Sim	Não	Não	Não	Sim	Sim
Possui Website	Sim	Não	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Rating (Apple Store |Google Pay)	4.7|4.5	4.9|4.8	4.1|4.3	4.7|4.5	4.9|4.8	4.7|4.4	4.8|4.9	4.8|4.8	4.8|4.5
Preço (€/mês)	10	3.49	3.99	24.99	N/A	Grátis	11.99	4.99	6.99

Na Tabela 6 encontram-se presentes as funcionalidades mencionadas na secção 3.2.2, indicando, para cada aplicação, se possui ou não a respetiva funcionalidade, com a exceção dos dois últimos atributos, rating e preço. Para o atributo rating é indicado numa escala entre 0 a 5, sendo o valor 5 correspondente ao melhor valor, a classificação presente no website da Apple Store e Google Play no dia 21 de janeiro de 2019. Já para o preço foi escolhido indicar apenas o valor mensal. Este valor, em todas as aplicações analisadas que o empregam, significa acesso a mais funcionalidades ou remoção de anúncios.

Da análise à Tabela 6 é possível concluir vários pormenores:

- Não existe uma aplicação que contenha todas as funcionalidades. Na verdade, a que mais se aproxima é a Jefit com 19 das 24. Por outro lado, Seven, Desafio Fitness de 30 Dias e J&J Official 7 Minute Workout são das que possuem menos com um total de 11;

- Existem quatro funcionalidades que estão presentes em todas as aplicações: visualização de planos de treino ou treinos pré-definidos, realização de configurações, notificações e integrações;

- Apesar de não estarem presentes na totalidade das aplicações, existindo em pelo menos 6 das 9, as seguintes funcionalidades também demonstram ser populares: registo ou login através de email/password ou com outras plataformas (p.e. Facebook), criação de um plano ou de um treino adicionando, sobretudo, exercícios, realização de um treino em tempo real, visualização num calendário toda a atividade feita, diversidade de estatísticas, acompanhamento do progresso num perfil, partilha nas redes sociais e existência de um website para comercialização do produto ou disponibilização de diversas funcionalidades;

- As funcionalidades menos implementadas são: recomendações (3 aplicações), criação de um exercício, adição de fotos para visualizar progresso, importar ou exportar dados (2 aplicações), integração com personal trainer e visualização de planos alimentares (1 aplicação);

- Todas as aplicações escolhidas possuem um rating superior a 4.0. O menor número (4.1) é pertencente à Intensity na Apple Store e o maior (4.9) pertencente ao Desafio Fitness de 30 Dias e AmazinGym na Apple Store, e Keelo no Google Play;

- À exceção das aplicações AmazinGym e J&J Official 7 Minute Workout, sendo a primeira disponível apenas para os sócios do ginásio e a segunda grátis para todos os utilizadores, as restantes apresentam planos de subscrição. Esses planos dependem de app para app, podendo variar entre mensal, trimestral, anual, ou outros, sendo o escolhido para análise o mensal, visto que era comum a todas. Tal como já referido, a subscrição permite o acesso a múltiplas funcionalidades *premium* ou, no caso do Desafio Fitness de 30 Dias, os anúncios deixam de existir. Os preços são semelhantes,

com a exclusão do Freeletics Bodyweight, possuindo um preço maior devido à funcionalidade Coach, já explicada, que utiliza IA.

Todo este processo de análise às aplicações revelou-se imprescindível para o desenvolvimento de tarefas futuras. De facto, foi possível identificar algumas das aplicações mais populares na área de fitness reconhecendo os seus pontos fortes e fracos, especificar as funcionalidades indispensáveis que uma app deste ramo deve possuir analisando os seus comportamentos ao nível da usabilidade e assimilar as diversas opiniões dos milhares de utilizadores.

Além disso, a presente análise enriqueceu a nível motivacional. Na verdade, a existência de múltiplas avaliações de utilizadores diferentes salienta a aderência das pessoas pelo uso deste tipo de aplicações, e o facto da inexistência de funcionalidades em aplicações com rating superior a 4.4, consideradas, por esta análise, muito pertinentes, são fatores que podem ser relevantes no sucesso da solução a desenvolver.

4 Design

Este capítulo é composto por três partes: o levantamento de requisitos, o modelo de domínio e a arquitetura. O levantamento de requisitos tem como objetivo capturar não só os requisitos funcionais, como também, os requisitos não funcionais do sistema. Desta análise, resultarão as funcionalidades fundamentais para a aplicação. Uma vez analisados os requisitos, são apresentadas as alternativas para o desenvolvimento de uma arquitetura, bem como a apresentação de um modelo de domínio.

4.1 Requisitos

O levantamento de requisitos do presente projeto foi efetuado baseado em: funcionalidades imprescindíveis na prática de ginásio resultantes da análise de aplicações *fitness* (capítulo Estado de Arte) e resultados de um inquérito de análise de aplicações de *fitness* efetuado a praticantes de ginásio (Anexo 1).

Esta secção descreve os elementos fundamentais relativamente aos requisitos. Assim sendo, é composto pelos requisitos funcionais e não funcionais, mas também pela identificação dos *stakeholders* e atores do sistema. Nos requisitos funcionais são identificados e descritos os casos de uso do sistema, já nos requisitos não funcionais, foram analisados pontos fundamentais como a segurança, a usabilidade e outros, que influenciam fortemente o sucesso desta aplicação.

4.1.1 Stakeholders

"Um *stakeholder* na arquitetura do sistema é uma pessoa, conjunto de pessoas, organização, ou outra entidade, que tem interesse na realização do sistema." [90] Assim, existe a importância de definir quem são os *stakeholders*, dado que estes, influenciam ou influenciarão o processo de desenvolvimento do sistema. Por conseguinte, foram identificadas as seguintes partes interessadas:

- **Ginásio:** interesse em oferecer uma plataforma aos seus membros;
- **Utilizador não autenticado:** interesse em registar no sistema para usufruir das funcionalidades;
- **Praticante de fitness:** interesse em usar a aplicação e realizar um conjunto de funcionalidades para acompanhar as suas atividades de fitness;
- **Personal trainer/ treinador:** interesse em acompanhar a evolução dos seus orientandos;
- **Administrador do sistema:** interesse pela gestão do sistema com permissões especiais para gerir a aplicação;
- **Nutricionista:** interesse pela gestão do sistema em termos de informação.

4.1.2 Atores do sistema

Os atores podem representar papeis desempenhados por utilizadores, máquina/sistema externo ou outros, que interagem com o sistema e fazem uso das suas funcionalidades [91]. De referir, que os atores são sempre partes interessadas, mas nem todas as partes interessadas são atores, "já que nunca interagem diretamente com o sistema, mesmo que tenham o direito de se preocuparem com o comportamento deste" [92].

Pela análise anterior e considerando os requisitos funcionais (cf. Secção 4.1.3) definiram-se cinco atores no sistema:

- **Utilizador não autenticado:** não se encontra registado/autenticado no sistema em que apenas pode realizar o processo de registo/login para utilizar as funções disponibilizadas;
- **Utilizador (praticante de fitness):** utiliza a aplicação e usufrui das suas funcionalidades;
- **Personal trainer:** responsável pelas funcionalidades relacionadas com monitorização e acompanhamento dos seus orientandos;
- **Administrador:** responsável pedo sistema, validando sugestões de informações vinda dos utilizadores (p.e. exercícios ou planos de treino) e adição de novos dados;
- **Nutricionista:** responsável pela gestão de planos de nutrição.

4.1.3 Requisitos funcionais

Nesta secção são identificados os casos de uso e as suas ligações com os atores do sistema.

Assim, na Figura 18 e Figura 19 apresenta-se o diagrama de casos de uso com uma visão generalista dos atores e a relação com os casos de uso na aplicação.

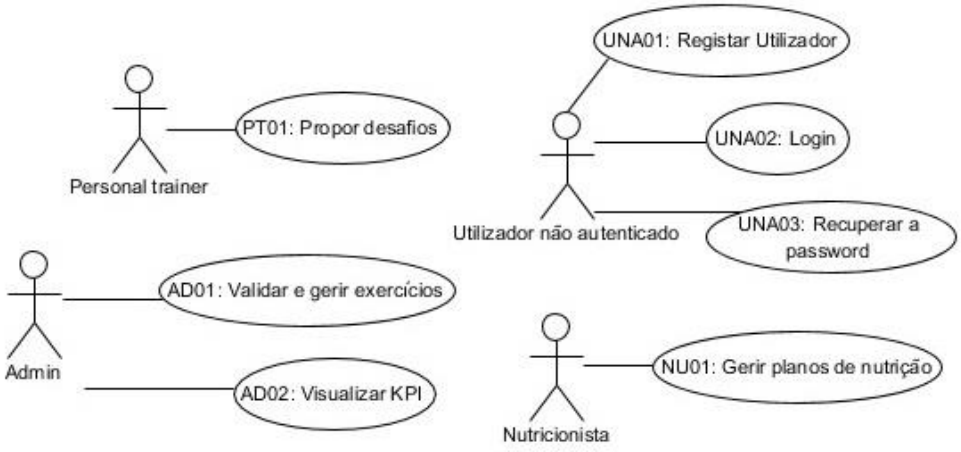


Figura 18 - Diagrama de casos de uso (parte 1)

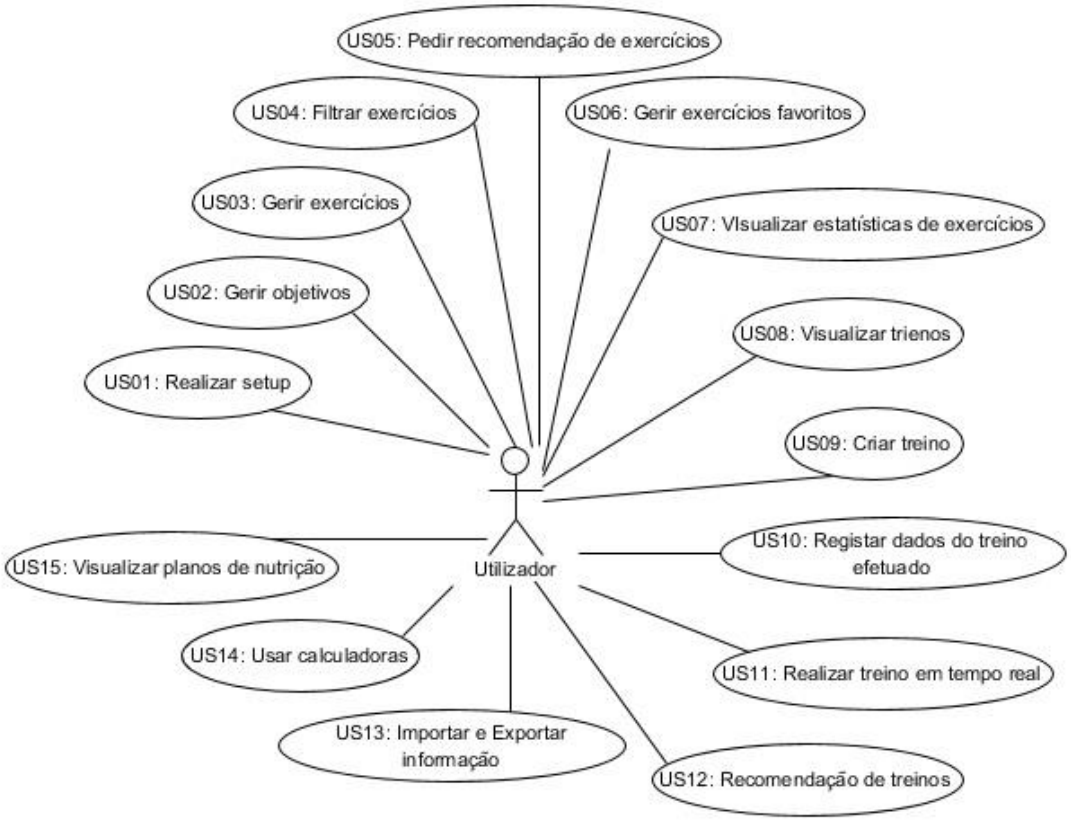


Figura 19 - Diagrama de casos de uso (parte 2)

Na Tabela 7 encontra-se ilustrada a prioridade de cada caso de uso tendo em conta a importância para o presente projeto.

Tabela 7 – Prioridade dos casos de uso

Nome	Prioridade
UA01: Registar Utilizador	5
UA02: Login	5
UA03: Recuperar password	5
US01: Realizar *setup*	5
US03: Gerir exercícios	5
US08: Visualizar treinos	5
US09: Criar treino	5
US10: Registar dados do treino efetuado	5
US11: Realizar treino em tempo real	5
US12: Recomendação de treinos	5
US04: Filtrar exercícios	4
US05: Pedir recomendação de exercícios	3
US02: Gerir objetivos	2
US06: Gerir exercícios favoritos	2
US07: Visualizar estatísticas de exercícios	2
US13: Importar e Exportar informação	2
US14: Usar calculadoras	2
AD01: Validar e gerir exercícios	2
US15: Visualizar planos de nutrição	1
PT01: Propor desafios	1
AD02: Visualizar KPI	1
NU01: Gerir planos de nutrição	1

4.1.3.1 UNA01: Registar Utilizador

O utilizador pode registar-se na plataforma e para isso deve introduzir alguns dados, como o email, *password* e confirmação da *password*.

4.1.3.2 UNA02: Login

Depois de introduzidos os dados da conta (e-mail e *password*), o sistema verifica se os dados estão corretos e se pertencem a um utilizador existente, permitindo ou não, o acesso às funcionalidades.

4.1.3.3 UNA03: Recuperar *password*

Em primeiro lugar, o utilizador introduz o seu e-mail. De seguida, como forma de validar que o utilizador tem acesso ou é o proprietário do e-mail, o sistema envia uma mensagem de correio eletrónico tendo como destinatário o e-mail introduzido. Caso o utilizador carregue no link enviado na mensagem, o sistema gera uma nova *password*.

4.1.3.4 US01: Realizar setup

Aquando do primeiro *login* feito pelo utilizador são realizadas certas perguntas a este com o intuito de definir um perfil inicial e estabelecer os seus objetivos: sexo; idade; peso; quantos dias por semana treina; e objetivos (p.e. entrar em forma, ficar forte e emagrecer).

4.1.3.5 US02: Gerir objetivos

O utilizador, a qualquer momento, pode alterar os objetivos definidos, adicionando ou removendo objetivos, tendo impacto no resultado dos algoritmos de recomendações.

4.1.3.6 US03: Gerir exercícios

É possível realizar uma gestão sobre os exercícios, podendo adicionar os seus próprios, visualizá-los, editá-los, removê-los, partilhá-los (sujeito a processo de validação pelo administrador) e visualizar os detalhes destes, como o nome, explicação e imagem ilustrativa, entre outros.

4.1.3.7 US04: Filtrar exercícios

Possibilidade de filtrar exercícios tendo em conta a sua dificuldade de execução (iniciante, intermédia e experiente), tipo de mecanismo, equipamento utilizado, áreas do corpo exercitadas ou favoritos.

4.1.3.8 US05: Pedir recomendação de exercícios

O utilizador pode requisitar a recomendação de exercícios. Aqui, o algoritmo de IA irá recomendar certos exercícios tendo em conta os seus objetivos, exercícios favoritos, nível de dificuldade, entre outros.

4.1.3.9 US06: Gerir exercícios favoritos

No presente caso de uso o utilizador pode fazer uma gestão sobre os seus exercícios favoritos, adicionando ou removendo exercícios. Esta funcionalidade é importante para que o utilizador possa visualizar mais facilmente os exercícios pretendidos.

4.1.3.10 US07: Visualizar estatísticas de exercícios

Possibilidade de visualização num gráfico as informações sobre o peso levantado (volume) para um determinado exercício, como o máximo, mínimo, médio ou crescimento, num determinado período.

4.1.3.11 US08: Visualizar treinos

O utilizador pode visualizar os múltiplos treinos existentes na aplicação, bem como os criados por si.

4.1.3.12 US09: Criar treino

O utilizador pode criar os seus próprios treinos. Aqui é necessária a introdução de dados, tais como: nome do treino, exercícios, número de *sets*[2], repetições e quantidade de peso. Existem, também, dados opcionais: tempo de descanso entre cada *set* e o tempo de duração.

4.1.3.13 US10: Registar dados do treino efetuado

O utilizador pode registar os dados sobre as suas atividades efetuadas, nomeadamente, relacionados com exercícios, treinos ou planos de treinos. Aqui serão pedidas informações, como o número de *sets*, peso, distância percorrida, entre outros.

4.1.3.14 US11: Realizar treino em tempo real

Em simultâneo, o utilizador pode estar a realizar o seu próprio treino e a usar a presente funcionalidade. Aqui é possível consultar em tempo real o exercício que tem de fazer com a indicação da sua explicação de execução, bem como o registo do peso levantado para cada repetição. Este processo é repetido até à execução de todos os exercícios do treino escolhido. Entre cada *set*, se tal tiver sido escolhido, é mostrado um temporizador com o tempo de descanso introduzido.

4.1.3.15 US12: Recomendação de treinos

O utilizador pode requisitar a recomendação de treinos. O algoritmo de IA irá recomendar certos treinos tendo em conta os seus objetivos, treinos favoritos, nível de dificuldade, feedback fornecido, entre outros.

4.1.3.16 US13: Importar e Exportar informação

Assim que o utilizador entender e que hajam dados para tal, é possível realizar a exportação da informação. Serão exportados os planos de treinos, com a indicação dos dias e exercícios. O número de *sets*, repetições e peso alcançado, também farão parte deste processo. Por outro lado, através de um *template* fornecido, é possível importar dados, os mesmos que a exportação.

4.1.3.17 US14: Usar calculadoras

O utilizador pode usar as calculadoras existentes com o intuito de auxílio na prática das suas atividades, entre elas podem estar:

- **Calculadora de aquecimento:** informa qual o número de *sets*, repetições e quantidade de peso para o aquecimento;
- **Calculadora de peso:** informa qual a quantidade de peso para colocar em cada lado da barra.

[2] Um *set* ou uma série é o número de vezes que um exercício é repetido. É composto, predominantemente, pela quantidade de peso levantada e pelo número de repetições.

4.1.3.18 US15: Visualizar planos de nutrição

O utilizador pode consultar e seguir um certo plano de nutrição definindo por um nutricionista.

4.1.3.19 PT01: Propor desafios

De maneira a motivar os seus orientados, o *personal trainer* pode propor vários desafios a estes. Podem estar relacionados com treinos específicos de forma a atingir certos objetivos.

4.1.3.20 AD01: Validar e gerir exercícios

O administrador, depois de um utilizador sugerir um exercício para ser disponibilizado para todos os utilizadores da aplicação, necessita de validá-lo. Além disso, pode fazer a gestão dos exercícios pré-definidos, podendo adicionar, alterar ou remover exercícios.

4.1.3.21 AD02: Visualizar KPI

O administrador pode visualizar diversos dados sobre o sistema, tais como: número de utilizadores, tempo médio despendido de cada utilizador na aplicação, funcionalidades mais usadas, entre outras.

4.1.3.22 NU01: Gerir planos de nutrição

O nutricionista pode fazer a gestão de planos de nutrição. Um plano de nutrição estará associado a determinados dias, sendo que para cada dia serão definidas as refeições a seguir para os mais importantes tipos de refeição (pequeno-almoço, almoço e jantar).

4.1.4 Requisitos não funcionais

Apresentam-se, de seguida, os requisitos não-funcionais recolhidos para o sistema segundo a classificação FURPS+. Estes encontram-se agrupados por atributos de qualidade de *software*.

4.1.4.1 Funcionalidades

Refere-se às funcionalidades tipicamente não capturadas nos casos de uso. O sistema deve assegurar o seguinte:

- **Autenticação:** a utilização do sistema requer uma autenticação do utilizador de maneira a disponibilizar os seus dados;
- **Autorização:** todas as funcionalidades devem ser protegidas por método de autorização conforme os papéis de cada ator do sistema;
- **Segurança:** a segurança dos dados é um aspeto crítico e essencial em qualquer sistema de informação. Para garantir um nível de segurança desejado é necessário apresentar algumas características, tais como:
    - **Autenticidade:** garantir que as entidades intervenientes são quem afirmam ser, conforme o atributo Autenticação mencionado anteriormente;
    - **Confidencialidade:** garantir a confidencialidade dos dados do utilizador utilizando métodos de cifragem, garantindo o RGPD.

4.1.4.2 Usabilidade

Refere-se à interação com o utilizador. Como já referido, é um atributo importante na adoção deste tipo de aplicações, pelo que é fundamental desenvolver uma interface com os seguintes aspetos:

- **Fácil utilização:** possuir uma interface intuitiva;
- **Adaptável:** adaptada a diferentes tipo de utilizadores de fitness;
- **Prevenção de erros:** interface cuidadosa que previne erros, como pedidos de eliminação que vêm acompanhadas de mensagens de confirmação.

4.1.4.3 Confiabilidade

Refere-se à capacidade do sistema em manter o seu funcionamento em circunstâncias inesperadas.

- **Previsibilidade:** o sistema deve ser confiável, isto é, deve estar livre de erros técnicos;
- **Tolerância a Falhas:** o sistema deve ser tolerante a erros para proteger o utilizador de erros não intencionais.

4.1.4.4 Desempenho

Refere-se aos requisitos de desempenho, nomeadamente:

- **Tempo de resposta:** acesso rápido aos dados desejados;
- **Disponibilidade dos dados:** deve garantir-se que a informação esteja disponível para o utilizador e livre de qualquer erro.

4.1.4.5 Suportabilidade

Refere-se a diversas características, tais como:

- **Portabilidade:** pretende-se que a aplicação esteja disponível nos sistemas operativos Android e iOS;
- **Testabilidade:** o sistema deve ser facilmente testado de maneira a proporcionar simples correções e estar isento de erros;
- **Manutenção:** o sistema deve possuir uma alta capacidade de manutenção, a fim de permitir novos requisitos ou reparos futuros;
- **Multilingue:** o sistema deve permitir suporte a vários idiomas.

4.2 Modelo de Domínio

O modelo de domínio é uma representação formal de um domínio que incorpora conceitos, papéis, tipos de dados, indivíduos, comportamentos e regras. Os conceitos incluem os dados envolvidos no negócio e as regras que o negócio usa em relação a esses dados [93]. Dessa forma, o modelo de domínio apresentado na Figura 20 resulta da análise do problema em questão e captura os principais conceitos de negócio e relações existentes entre si.

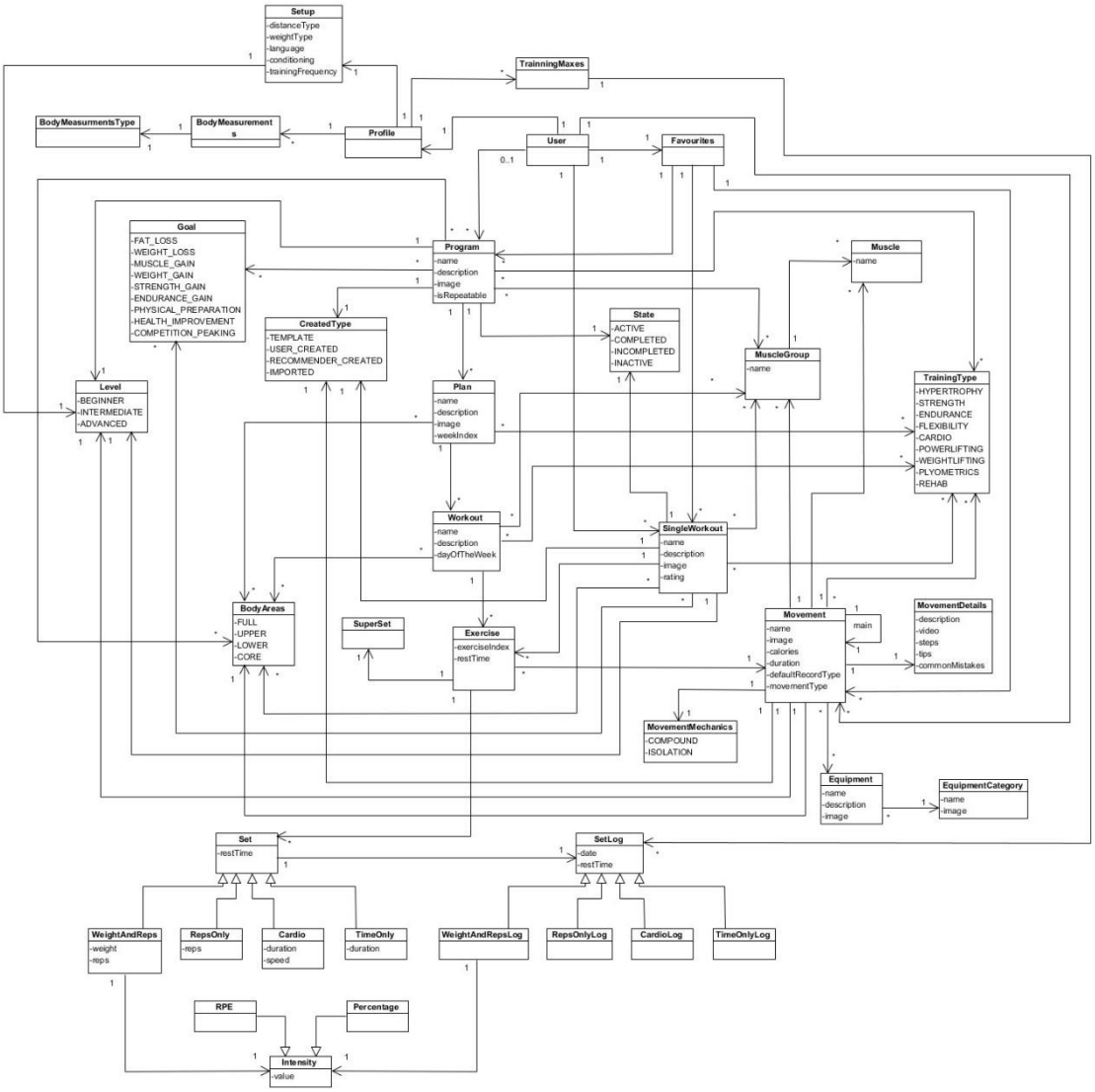


Figura 20 - Modelo de Domínio

De forma a simplificar a explicação das múltiplas entidades presentes no diagrama da Figura 20 e as suas relações, foram identificados os seguintes sete principais domínios:

- **Perfil** *(Profile):* o utilizador possui um perfil e este relaciona-se com vários registos de medidas corporais (p.e. altura ou comprimento do peito) e com recordes de levantamento de peso. Além disso, associa-se a um *Setup* em que são configuradas

informações relativas à língua, às preferências métricas, à frequência de atividade física, à experiência e ao nível de condição;

- **Programa** *(Program):* um utilizador pode possuir vários programas. Este é caracterizado por possuir um nome, uma descrição, uma imagem, um nível de experiência (p.e. iniciante), um tipo de criação (p.e. *template*), a área do corpo exercitada (p.e. parte superior), vários objetivos (p.e. perder peso), vários tipos de treino (p.e. força), ser ou não repetível e um conjunto de planos de treino;
- **Plano de treino** *(Plan):* pode ou não estar relacionado com um programa, possuindo os mesmos atributos deste, com a adição do índice da semana (preenchido caso seja de um programa) e um conjunto de treinos;
- **Treino** *(Workout):* pertence a um plano de treino, com os mesmos atributos deste, com a exceção do índice da semana, mas com a adição do dia da semana em que é realizado, e um conjunto de exercícios;
- **Treino único** *(SingleWorkout):* igual ao treino, com a exceção de pertencer a um plano de treino;
- **Exercícios** *(Exercise):* a modelagem deste domínio foi realizada tendo em conta a secção Exercícios Fitness. Neste caso, um exercício possui um movimento e este possui um nome, uma duração, um tipo de criação, uma dificuldade, um tipo de registo (p.e. tempo), um tipo de mecanismo (p.e isolado), uma variação, isto é, um exercício semelhante, mas realizado de forma diferente, vários equipamentos, vários grupos de músculos, vários músculos, vários tipos de treino e um conjunto de *sets*;
- **Sets** *(Set):* representa o número de vezes da realização de um exercício. Pode ser representado de quatro formas distintas: (i) peso e número de repetições, (ii) apenas número de repetições, (iii) duração e velocidade média e (iv) apenas duração. Somente a forma identificada como (i) possui o atributo intensidade que pode ser medido através do RPE (escala numérica para identificar o esforço exercido) ou percentagem. Todas elas apresentam um tempo de descanso a ser aplicado entre *sets*.

4.3 Arquitetura

Segundo Krutchen [94], "Uma arquitetura é o conjunto de decisões significativas sobre a organização de um sistema de *software*, a seleção dos elementos estruturais e suas interfaces por meio das quais o sistema é composto, juntamente com o seu comportamento conforme as colaborações entre esses elementos […]". De forma a identificar todos os elementos do sistema e suas interligações, de seguida, serão apresentas as várias alternativas para a arquitetura, sendo, posteriormente, identificada a escolhida.

4.3.1 Arquiteturas propostas

A arquitetura lógica suporta, principalmente, os requisitos funcionais - o que o sistema deve fornecer em termos de serviços aos seus utilizadores. Assim, o sistema é decomposto num conjunto de tomadas de decisões do domínio do problema [95]. A vista lógica tem o intuito de fornecer uma base para a compreensão da estrutura e organização do design do sistema, ilustrar os principais componentes e suas relações, que englobam comportamentos arquiteturalmente significativos.

Nas próximas secções encontram-se ilustradas as arquiteturas através do diagrama de componentes de alto nível, bem como a sua explicação.

4.3.1.1 Alternativa 1 – solução com API

Na Figura 21 encontram-se ilustrada a primeira alternativa.

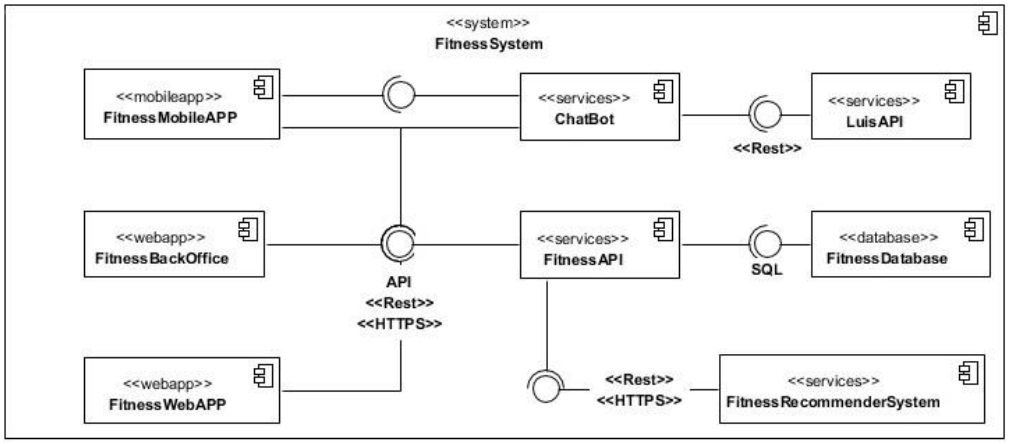


Figura 21 - Alternativa 1: diagrama de componentes

Antes da explicação de cada componente presente no diagrama anterior, é importante definir os estereótipos presentes na figura:

- **<<system>>:** representa o sistema;
- **<<mobileapp>>:** representa uma aplicação móvel;
- **<<webapp>>:** representa uma aplicação que pode ser acedida via *web browser*;
- **<<services>>:** representa um serviço, podendo ser interno (*FitnessAPI*) ou externo (*LuisAPI*);
- **<<database>>:** representa uma base de dados.

De seguida, os componentes da figura anterior serão descritos:

- **FitnessMobileAPP:** aplicação móvel que corre num dispositivo Android ou iOS que pode ser acedida por utilizadores registados;
- **FitnessWebAPP:** aplicação cliente que corre num *web browser* e está disponível para ser acedida a qualquer momento por qualquer pessoa (autenticada e autorizada);

- **FitnessRecommenderSystem:** responsável por toda a lógica inerente às recomendações;
- **FitnessBackOffice:** aplicação que corre num *web browser* destinada aos administradores;
- **FitnessAPI:** responsável por disponibilizar serviços que contêm a lógica de negócio da aplicação. Este componente tem a responsabilidade de comunicar com a base de dados através de SQL e de disponibilizar uma interface HTTPs e Rest facultando serviços;
- **FitnessDatabase:** componente responsável por guardar os dados referentes aos utilizadores, exercícios, planos de treinos, entre outros;
- **ChatBot** *(ChatBotAPI)***:** responsável pela lógica de interação entre o utilizador e um *bot* (*personal virtual assistant*);
- **LuisAPI:** serviço baseado em *machine learning* responsável por criar linguagem natural para utilização com o *bot*.

4.3.1.2 Alternativa 2 – solução baseada em micro serviços

Na Figura 22 encontra-se a alternativa número dois. Esta abordagem é baseada em micro serviços. Invés da utilização de uma única API, *FitnessAPI*, como ilustrado na figura anterior, esta divide-se em cinco serviços (*ExercisesService, ProgramsService, RecommendationsService, StatisticsService e UsersService*) tendo cada um a sua própria base de dados, segundo o padrão *Database per Service* [96], e uma determinada responsabilidade de negócio. Por exemplo, o serviço *UsersService* tem a própria base de dados, *UsersDB*, e é responsável pela lógica subjacente aos utilizadores, como o processo de registo, login, entre outros.

É utilizado, também, o componente *FitnessGateway* segundo o padrão *API Gateway* [97], que é responsável por ser o único ponto de entrada para todos os clientes, coordenando os pedidos dos diversos serviços.

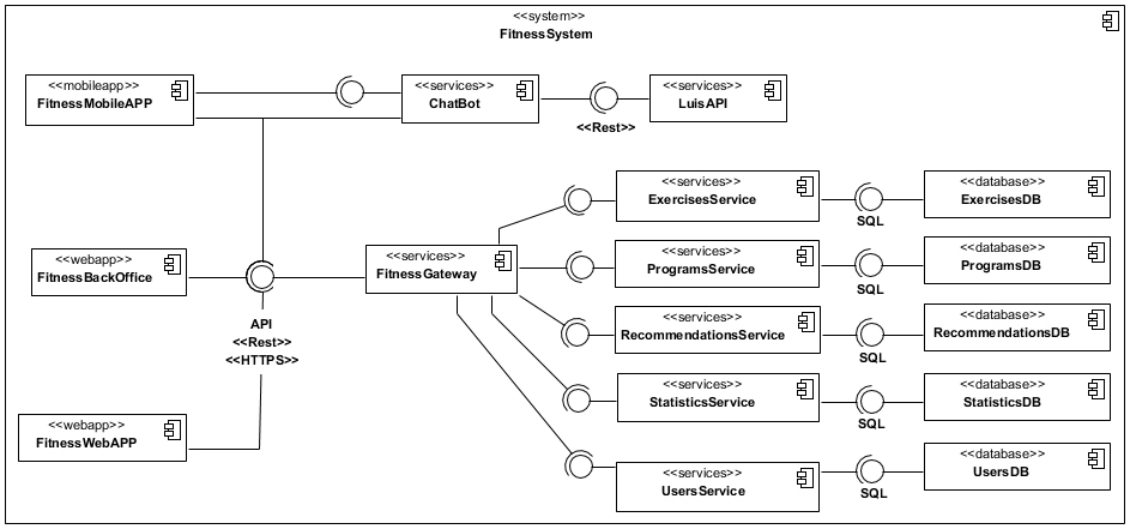


Figura 22 - Alternativa 2: diagrama de componentes

4.3.1.3 Alternativa 3 – solução sem API

Na Figura 23 apresenta-se uma arquitetura mais simples. Neste caso, cada aplicação, *mobile app* e *web app,* contém a sua própria lógica de negócio, conectando-se diretamente com a base de dados.

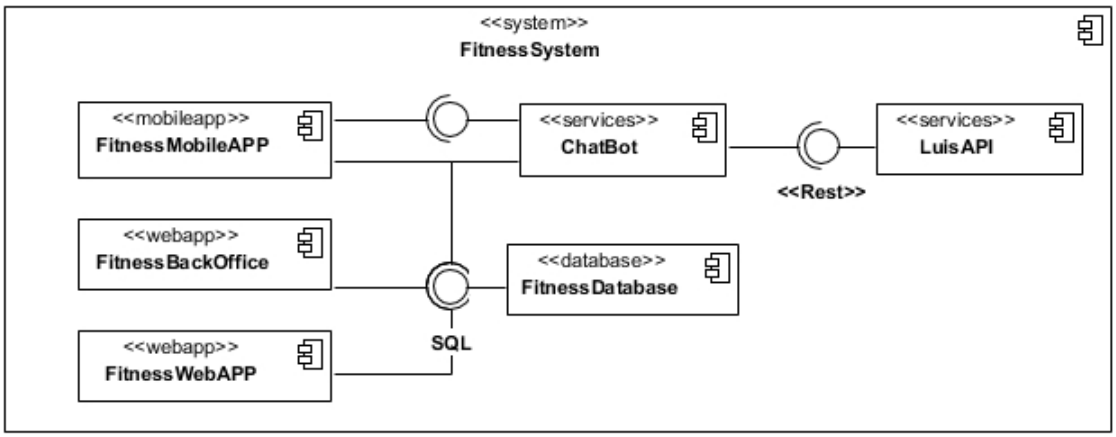


Figura 23 - Alternativa 3: diagrama de componentes

4.3.2 Arquitetura escolhida

Em primeiro lugar, a alternativa número 3 foi analisada. Apresenta-se como uma arquitetura monolítica em que apresenta vantagens em termos do tempo de desenvolvimento e facilidade de implantação. Contudo, o elevado número de desvantagens levou a que esta abordagem fosse rejeitada - difícil reutilização; a principal lógica de negócio estaria em cada uma das aplicações, dificultando a manutenção de código; e possibilidade de código repetido na aplicação móvel e aplicações web, nomeadamente, no acesso à base de dados e lógica de negócio.

Desta forma, a escolha de uma arquitetura passaria pelas alternativas 1 ou 2. A primeira solução com a inclusão de uma API apresenta vantagens, tais como: adequado para uma equipa de programadores pequena (entre dois a cinco elementos); existência de uma API que elimina a necessidade de lógica de negócio nas principais aplicações e a probabilidade de existência de redundância no código; simples de programar; vários ambientes de desenvolvimento (IDEs) apresentam funcionalidades para suportar este tipo de arquitetura; e suporte à escalabilidade. Contudo, também apresenta certas desvantagens, nomeadamente, dificuldade na manutenção quando o serviço obtém um tamanho considerável de código e lógica de negócio, dificultando a sua interpretação e modificação, possível sobrecarga do IDE devido ao número extenso de código e dificuldade no desenvolvimento em escala, isto é, no caso do serviço alcançar um tamanho consideravelmente extenso e existirem várias equipas com responsabilidades distintas, poderá dificultar a organização e independência de cada uma delas [98].

O desenvolvimento de uma arquitetura baseada em micro serviços remove algumas destas desvantagens. Cada um destes micro serviços tem um tamanho pequeno facilitando a rapidez do IDE e a manutenção, podendo adicionar ou alterar funcionalidades a cada serviço mais facilmente. Permitem o desenvolvimento contínuo de projetos grandes e complexos, facilitando a testabilidade e o processo de implantação no sentido em que podem ser implantados de forma independente, e possibilitam uma melhor organização na divisão do trabalho por equipas, podendo, cada uma, trabalhar num serviço específico. Por fim, removem a existência de um ponto de falha. No entanto, existe uma complexidade adicional na criação de um sistema distribuído - necessária a implementação de um mecanismo de comunicação entre serviços; em casos de uso que abrangem vários serviços serão necessárias transações distribuídas e uma coordenação cuidadosa entre as várias equipas. Além disso, existe uma determinada complexidade no processo de implantação, na perspetiva de que existem vários serviços [99].

Não existe uma arquitetura perfeita que possa ser solucionada para todos os projetos, sendo que é importante decidir tendo em conta as necessidades e limitações existentes. Em cima, foram analisadas as três alternativas identificadas tendo em conta as suas vantagens e desvantagens, com a alternativa número 3 a ser já rejeitada. O facto do desenvolvimento de uma arquitetura distribuída retardar a fase de desenvolvimento, ser indicada para equipas grandes, normalmente, com mais de 5 membros e existir uma sobrecarga operacional que requer padronização, automação de implementações e integração, fizeram com que a arquitetura escolhida fosse a Alternativa 1 – solução com API.

Assim, a arquitetura passa por ser uma junção das três alternativas. Não se apresenta tão simples como a alternativa 3, mas apresenta uma API com diversas funcionalidades, podendo evoluir lentamente para micro serviços.

4.3.3 Arquitetura escolhida detalhada

Na secção 4.3.1.1 foi apresentado o diagrama de componentes da solução de alto nível, desse modo, é importante detalhar a arquitetura a nível interno e físico. Na presente secção todos os componentes identificados na arquitetura escolhida serão aprofundados e descritos, bem como apresentado um diagrama de implantação com o intuito de fornecer uma visão do sistema de *hardware.*

4.3.3.1 Diagrama de componentes: FitnessAPI

A estruturação interna do presente componente é apresentada na Figura 24.

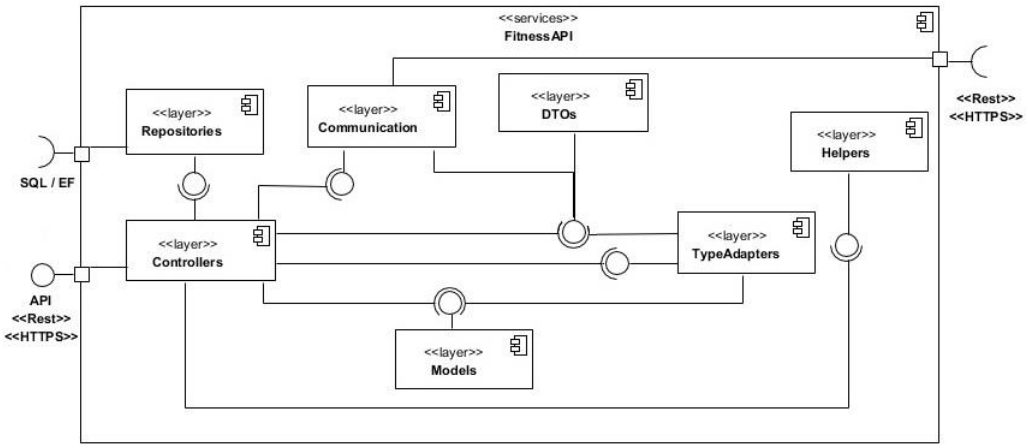


Figura 24 - Diagrama de componentes: FitnessAPI

Estereótipo presente na Figura 24:

- **<<layer>>:** representa uma camada na API, por exemplo, a camada *Controllers* será um *package* onde residirão todos os *controllers*.

Os componentes presentes na Figura 24 são:

- **DTOs:** usado para transferir dados entre a API e as aplicações agrupando um conjunto de valores (atributos) numa classe simples de forma a otimizar a comunicação. Não apresenta lógica de negócio;
- **Controllers:** responsável por tratar dos pedidos à API. Usa *DTOs* para a receção e envio de dados;
- **Repositories:** responsável por permitir que todo o código use objetos sem ter o conhecimento como estes são persistidos, permitindo uma abstração no código. Possui todo o conhecimento da persistência, incluindo o mapeamento de tabelas para objetos;
- **Models:** responsável por representar as classes de domínio e lógica de negócio sempre que necessário;
- **Communication:** responsável por tratar da comunicação com clientes, permitindo o desacoplamento da camada de tratamento de eventos da camada de comunicação com serviços externos;
- **Helpers:** responsável por possuir classes que sustentam o bom funcionamento da aplicação;
- **TypeAdapters:** responsável por possuir a lógica necessária à conversão entre *DTOs* e *Models*, e vice-versa.

4.3.3.2 Diagrama de componentes: FitnessRecommenderSystem

A estruturação interna do componente *FitnessRecommenderSystem* encontra-se presente na Figura 25.

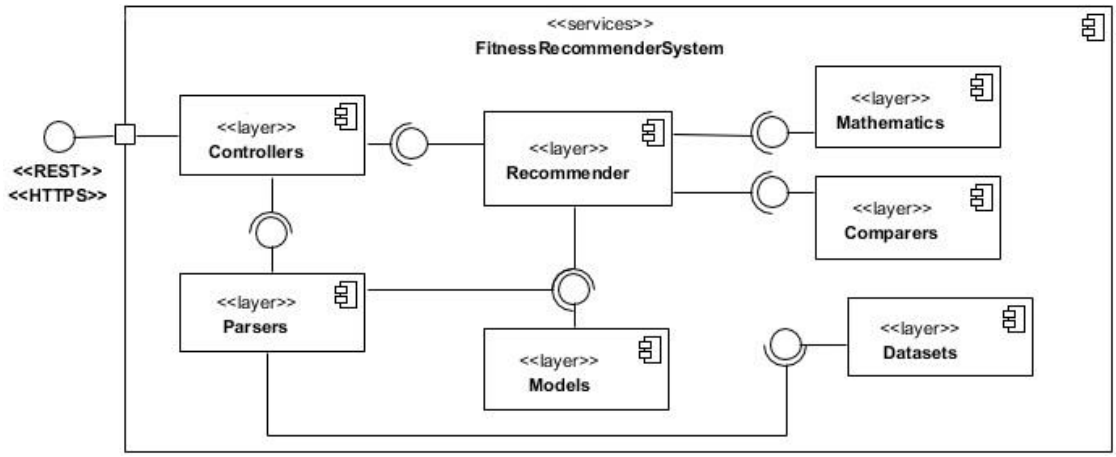


Figura 25 - Diagrama de componentes: FitnessRecommenderSystem

Os componentes presentes na Figura 25 são:

- **Controllers:** responsável por tratar dos pedidos à API;
- **Recommenders:** responsável por possuir a principal lógica inerente à recomendação;
- **Mathematics:** responsável por disponibilizar cálculos matemáticos como multiplicação e factorização de matrizes;
- **Comparers:** responsável por possuir os diferentes métodos para a obtenção do grau de semelhança entre utilizadores;
- **Models:** responsável por representar as classes de domínio;
- **Parsers:** responsável por importar os dados provenientes de ficheiros e mapear esses dados para a geração de recomendações;
- **Datasets:** conjunto de dados.

4.3.3.3 Diagrama de componentes: FitnessMobileApp

A estruturação interna do componente *FitnessMobileApp* encontra-se presente na Figura 26.

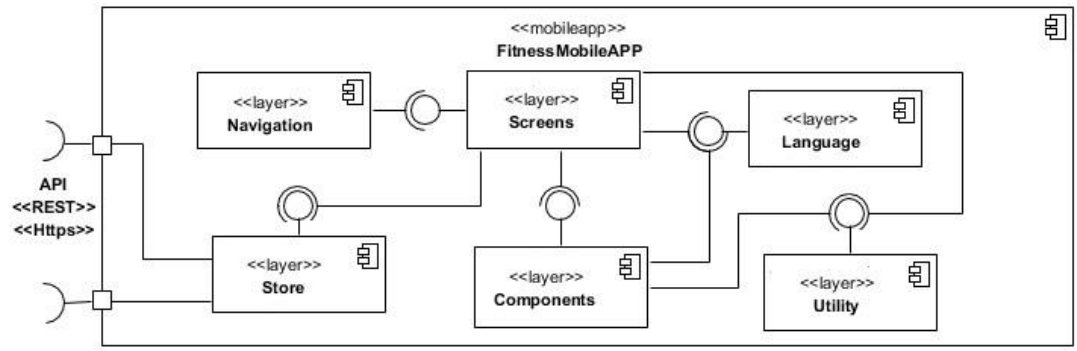


Figura 26 - Diagrama de componentes: FitnessMobileApp

Os componentes presentes na Figura 26 são:

- **Navigation:** responsável por possuir o registo de todos os ecrãs da aplicação;
- **Components:** recebe, opcionalmente, atributos e retorna um elemento que possui código relativo à interface gráfica. Um componente pode ser reutilizável podendo ser usado por diversos ecrãs;
- **Screens:** responsável, principalmente, pela organização da interface gráfica. Utiliza múltiplos componentes e comunica com a camada de comunicação para o retorno dos dados;
- **Store:** responsável por duas tarefas imprescindíveis – gere o estado da aplicação, isto é, proporciona a gestão dos dados apresentados ao utilizador e realiza a comunicação com o servidor;
- **Language:** responsável por disponibilizar os vários ficheiros de configuração das línguas disponíveis;
- **Utility:** responsável por possuir funções e constantes que sustentam o bom funcionamento da aplicação;

4.3.3.4 Diagrama de componentes: FitnessBackOffice & FitnessWebApp

A estruturação interna dos componentes *FitnessBackOffice* e *FitnessWebApp* encontra-se presente na Figura 27.

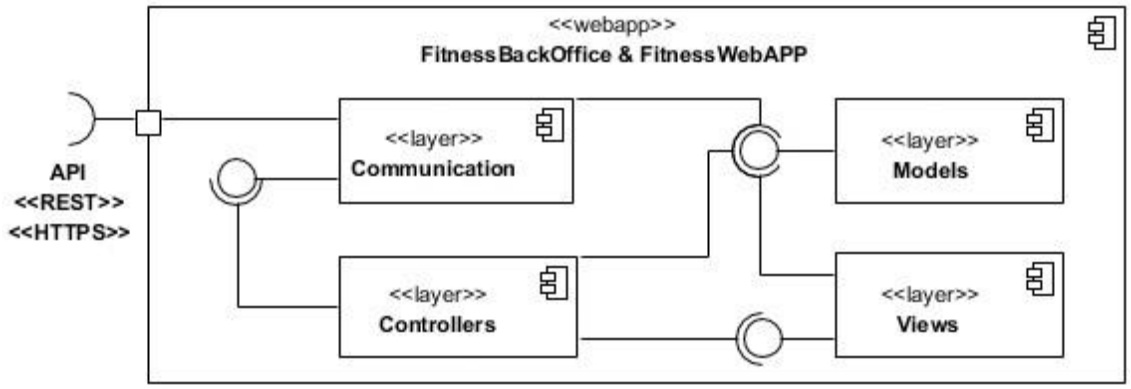


Figura 27 - Diagrama de componentes: FitnessBackOffice & FitnessWebApp

Os componentes presentes na Figura 27 são:

- **Controllers:** responsável por possuir métodos de ação que lidam com solicitações do *browser*, recuperar dados necessários do *model* e retornar as respostas apropriadas;
- **Models:** responsável por tratar da lógica de negócio, responder a pedidos para devolver informação sobre o estado da informação (usualmente vindos da *view*) e responder a instruções para mudar o estado da informação (usualmente vindos do *controller*);
- **Views:** responsável por gerir a apresentação de dados da aplicação e a interação com o utilizador;

- **Communication:** responsável por tratar da comunicação com a API, permitindo o desacoplamento da camada de tratamento de eventos da camada de comunicação com serviços externos.

4.3.3.5 Diagrama de implantação

A vista de implantação tem como objetivo entender as conexões físicas entre os nós, implantação e escalabilidade do sistema.

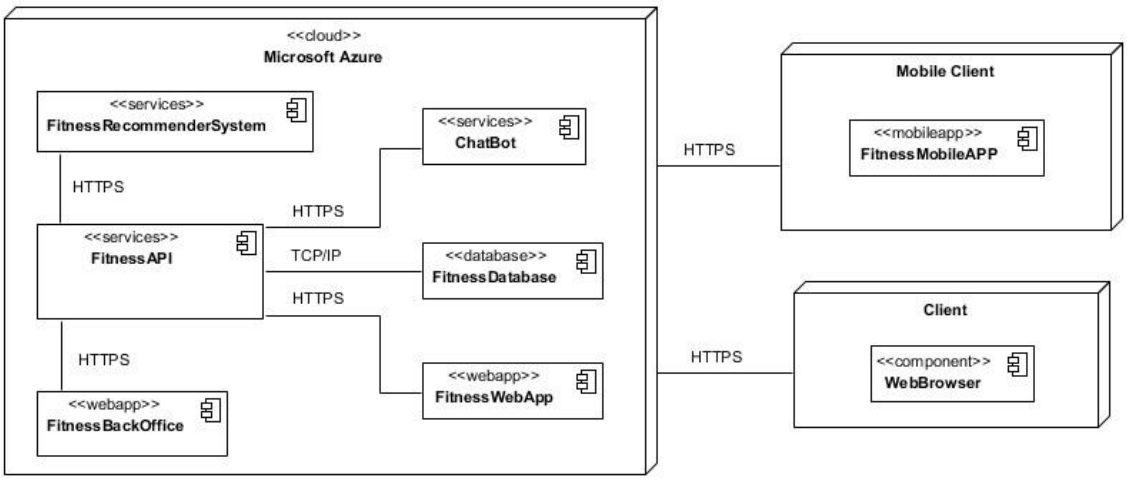


Figura 28 - Diagrama de implantação do sistema

A Figura 28 ilustra o diagrama de implantação do sistema. Tal como se pode observar, a generalidades dos componentes encontram-se implantados no *Azure* [100]. Foi usada esta abordagem devido à simplicidade do processo de implantação, à adoção da *cloud*, à gestão das aplicações e à acessível conexão com a bases de dados SQL. Além dos componentes instalados no *Azure*, a solução é composta por uma aplicação móvel que estará instalada no dispositivo móvel do cliente, que comunicará com a aplicação servidora. Por outro lado, o cliente, podendo ser, neste caso, um utilizador da aplicação ou administrador do sistema, através do *browser* pode aceder às funcionalidades da *FitnessWebApp* e *FitnessBackOffice*, respetivamente. De anotar que o componente *LuisAPI*, representado no diagrama de componentes da Figura 21, não se encontra ilustrado visto ser um serviço da Microsoft, não havendo a necessidade de implantá-lo.

5 Implementação

Neste capítulo será apresentado e discutido o processo de desenvolvimento levado a cabo para a implementação dos diferentes componentes do sistema. Para isso, será apresentado o processo de desenvolvimento para cada um desses componentes com recurso aos diagramas considerados pertinentes.

5.1 Sistema de Recomendação

Para a implementação dos sistemas de recomendação foi utiliza a linguagem *C#* e usado o *Visual Studio 2017* como ambiente de desenvolvimento.

De seguida, serão descritos os detalhes da implementação dos sistemas de recomendação.

5.1.1 Recomendação condicional

Um sistema de recomendação condicional foi criado devido à necessidade de possuir um sistema menos complexo e mais rápido, visto que grande parte deste projeto é relacionada com uma recomendação inteligente baseada em aprendizagem de máquina, processos iterativos, vasta quantidade de dados e criação de modelos.

Desta forma, a presente recomendação tem em conta múltiplos *inputs* para, através de filtros e técnicas matemáticas, sugerir treinos ao utilizador. Os inputs são:

- ***Dataset* de treinos:** representa a lista de treinos que será filtrada;
- **Histórico do utilizador:** representa os treinos já efetuados pelo utilizador, de forma a encontrar áreas ausentes, tendências, entre outros;
- **Preferências do utilizador:** dados preenchidos pelo utilizador durante a sua configuração inicial (*setup*).

O fluxo de funcionamento da recomendação condicional encontra-se apresentado no diagrama de sequência de alto nível da Figura 29.

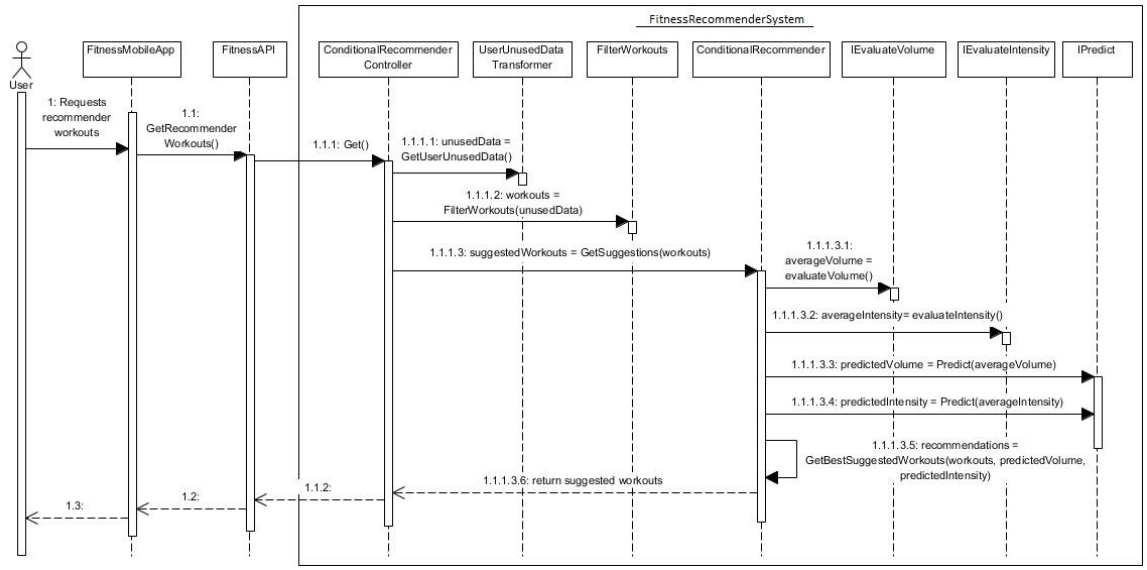


Figura 29 - Diagrama de sequência: recomendação condicional

Neste processo estão presentes três componentes da solução: *FitnessMobileApp*, *FitnessAPI* e *FitnessRecommenderSystem*. O fluxo inicia-se com o requisito de treinos feito pelo utilizador na aplicação móvel. Por sua vez, esta comunica com o servidor e este, com o serviço de recomendações enviando-lhe dados do utilizador e de treinos para a obtenção das sugestões. Esse processo de obtenção de treinos, como percetível através da Figura 30, pode-se resumir na sequência das seguintes três tarefas:

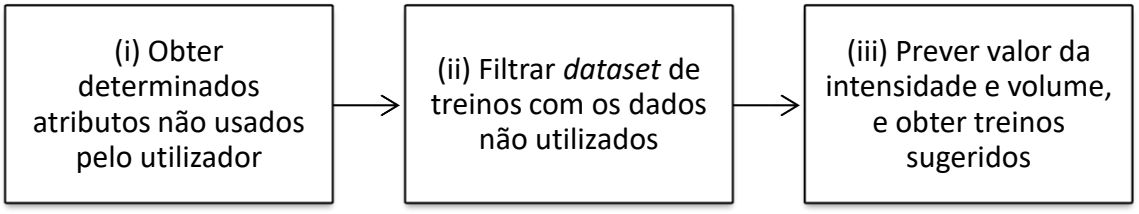


Figura 30 - Tarefas do componente de recomendação condicional

Na tarefa (i) o histórico de treinos do utilizador e os dados configurados por este no processo de *setup* (abordado na secção dos Casos de Uso) são tomados em conta. De facto, é feito um *loop* a todos os treinos realizados com o objetivo de encontrar os grupos de músculos, as partes do corpo, os tipos de treinos e os objetivos ainda "não treinados" pelo utilizador.

Para tornar o que foi dito anteriormente mais percetível, na Figura 31 encontra-se um excerto de código. Neste caso específico o propósito é retornar os objetivos ainda não utilizados pelo utilizador. Para isso, são necessários dois parâmetros, como dito no parágrafo anterior: treinos já realizados e os objetivos selecionados no processo de *setup*. Em primeiro lugar, são obtidos todos os objetivos dos treinos já realizados e, de seguida, estes são comparados com os objetivos escolhidos pelo utilizador, obtendo os objetivos ainda não utilizados.

```
public List<General> getUserUnusedGoals(List<SingleWorkout> userHistory, List<General> goals)
{
    List<General> goalsHistory = userHistory.Select(x => x.Goals).SelectMany(i => i).Distinct().ToList();

    return goals.Where(x => !goalsHistory.Any(y => y.ID == x.ID)).ToList();
}
```

Figura 31 – Excerto de Código: Obter objetivos ainda não treinados

Na tarefa (ii) o *dataset* de treinos existentes diminui à medida que determinados filtros são aplicados. Um exemplo de um desses filtros é a obtenção de treinos que tenham os mesmos objetivos que os objetivos não usados (resultante da tarefa (i)).

Na última tarefa (iii) é onde se encontra a parte mais complexa desta funcionalidade. Em primeiro lugar, para cada treino já realizado são calculados dois parâmetros – volume e intensidade, para decidir, posteriormente, qual o volume e a intensidade mais adequados para o próximo treino. Foi aplicado o padrão *Strategy* [101] para tornar o código reutilizável, flexível e permitir a existência de diversas formas de cálculo, como ilustrado na Figura 32.

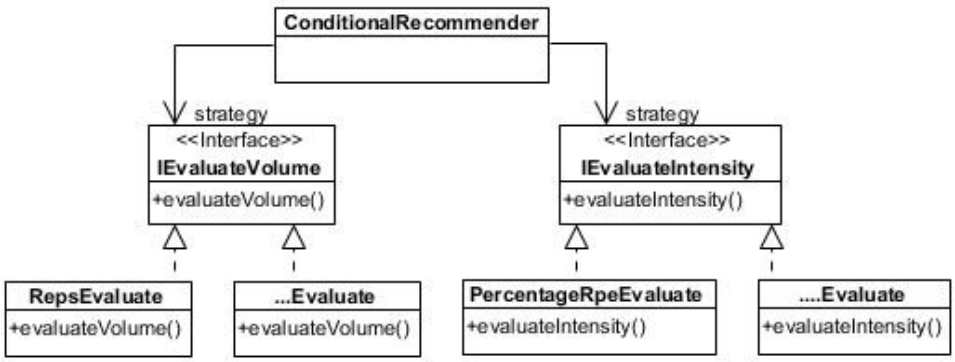


Figura 32 - Padrão Strategy: IEvaluate

Duas classes foram criadas para o cálculo do volume e da intensidade, *RepsEvaluate* e *PercentageRpeEvaluate*, respetivamente. A primeira, implementa o método *evaluateVolume()* que calcula, para cada treino realizado, o somatório do número de repetições, retornando uma lista de inteiros. Por outro lado, a segunda, *PercentageRpeEvaluate*, implementa o método *evaluateIntensity()* que, também para cada treino, realiza as seguintes médias aritméticas:

$$Iexercício\ (Ie) = \frac{Iset_1 + Iset_2 + \cdots + Iset_n}{n} \quad (2) \quad , \quad Itreino = \frac{Ie_1 + Ie_2 + \cdots + Ie_n}{n} \quad (3)$$

, onde Ie é a intensidade de um exercício, $Iset$ a intensidade de um *set* e n é o número de *sets* (equação (2)) ou de exercícios (equação (3)).

A equação (2) calcula a intensidade para um exercício. Como já referido no Modelo de Domínio, um exercício possui uma lista de *sets* e cada um destes possui um atributo "Intensity" que pode ser medido em percentagem ou RPE. Devido a esse facto, foi definida a seguinte escala numérica para atribuir uma intensidade a um *set*:

$$i = \begin{cases} 0.5, & Percentagem \leq 50 \\ 1, & (Percentagem > 50 \wedge Percentagem \leq 60) \vee (RPE \geq 6 \wedge RPE < 7) \\ 2, & (Percentagem > 60 \wedge Percentagem < 75) \vee (RPE \geq 7 \wedge RPE < 8) \\ 3, & (Percentagem \geq 75) \vee (RPE \geq 8 \wedge RPE < 10) \end{cases}$$

Depois de calculadas todas as intensidades dos exercícios, é aplicada a equação (3) para obter a intensidade de um treino.

De seguida, é calculado o valor previsto para os dois atributos, volume e intensidade. Mais uma vez, foi utilizado o padrão *Strategy* para que possam existir diversas formas de cálculo para esse valor previsto. Uma dessas formas é a utilização da técnica *Line Of Best Fit* [102], um procedimento matemático de regressão linear para encontrar a reta com o melhor ajuste para um determinado conjunto de pontos. É uma ferramenta de previsão, fornecendo um mecanismo para projetar valores futuros.

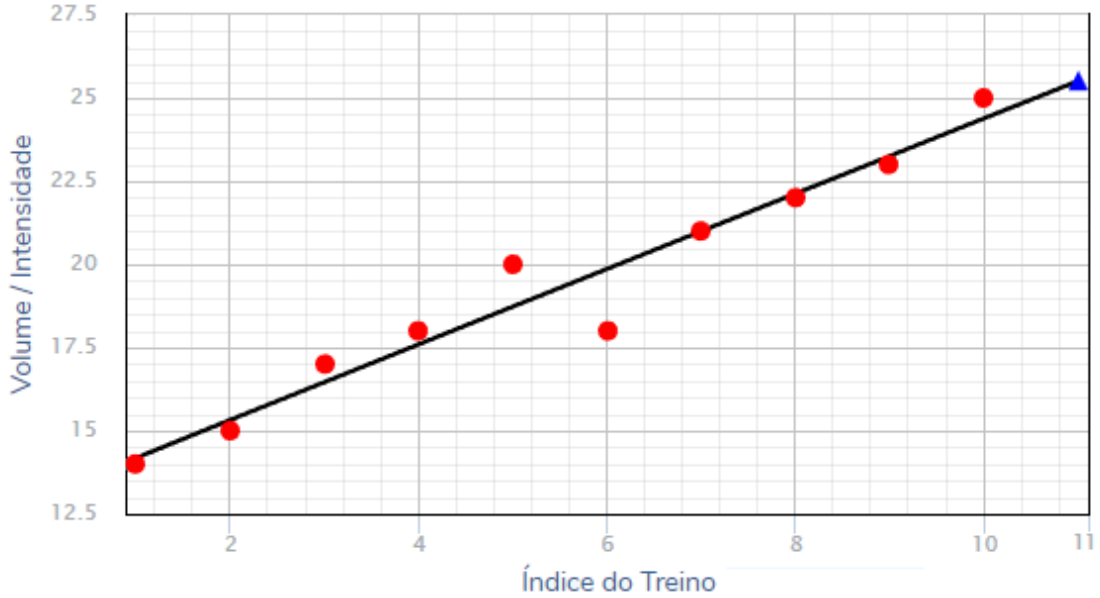


Figura 33 - Exemplo da técnica *Line Of Best Fit* [103]

Um exemplo prático desta técnica encontra-se presente na Figura 33. Considerando um conjunto de valores de volume ou intensidade para diferentes treinos, têm-se como objetivo prever qual será o próximo valor (presente no gráfico com o símbolo de triângulo).

São realizadas três etapas para a obtenção dos valores da equação que traduz a reta:

$$y = mx + b \tag{4}$$

, onde y é o valor do volume/intensidade, m é o declive, x o índice de treino e b é o valor da interseção com o eixo das ordenadas

1. Cálculo da média dos valores das variáveis X (índice do treino) e Y (volume/intensidade), $\bar{X}$ e $\bar{Y}$, respetivamente;
2. Cálculo do valor do declive através da fórmula:

$$m = \frac{\frac{\sum_{i=1}^{n} x_i y_i}{n - \bar{X}\,\bar{Y}}}{\frac{\sum_{i=1}^{n} x_i^2}{n - \bar{X}\,\bar{X}}} \tag{5}$$

3. Cálculo do valor da interseção com o eixo das ordenadas (eixo dos y):

$$b = m\bar{X} - \bar{Y} \tag{6}$$

Tendo os valores da equação, resta substituir o valor de x pelo índice de treino pretendido (índice de treino + 1) e o valor de y referente ao volume/intensidade previsto é retornado.

A realização dos cálculos anteriormente apresentados encontram-se ilustrados no excerto de código da Figura 34.

```
public static List<XYPoint> GenerateLinearBestFit(List<XYPoint> points)
{
    int numPoints = points.Count;
    double meanX = points.Average(point => point.X);
    double meanY = points.Average(point => point.Y);

    double sumXSquared = points.Sum(point => point.X * point.X);
    double sumXY = points.Sum(point => point.X * point.Y);

    double a = (sumXY / numPoints - meanX * meanY) / (sumXSquared / numPoints - meanX * meanX);
    double b = (a * meanX - meanY);
    [...]
}
```

Figura 34 – Excerto de Código: Cálculos da técnica *Line Of Best Fit*

Por fim, os valores previstos do volume e intensidade são multiplicados e retornados os N treinos (resultados da tarefa (ii)) que possuem os valores mais próximos do resultado dessa multiplicação.

5.1.2 Recomendação inteligente

De forma a facilitar a escolha de treinos e a realização de atividade física no que toca ao *fitness*, foi decidido utilizar um sistema de recomendação inteligente que auxiliasse no processo de decisão do utilizador. Um sistema é considerado inteligente caso possua a capacidade de sugerir recomendações baseadas num modelo criado e ter a competência de se adaptar tendo em conta os dados existentes [104].

Como primeira versão deste SR apenas as classificações dos utilizadores, as características dos itens e a similaridade entre utilizadores ou itens serão tomadas em conta, podendo, no futuro, ser incluído o perfil e atributos dos utilizadores, regras ou requisitos impostos, histórico de atividade, entre outros.

Desta forma, as melhores técnicas que contemplam o que foi dito anteriormente são: *Matrix Factorization*- técnica inteligente baseada em modelos, aprendizagem e otimizações; Filtragem Colaborativa baseada em itens e utilizadores- sugere tendo em conta a similaridade de utilizadores e itens; e Híbrida- combinação entre as várias técnicas, preferencialmente, usando a *Matrix Factorization*.

A implementação deste sistema foi baseada no artigo de Scott Clayton "*Building a Recommendation Engine in C#"* [105]. Possui uma classificação de 4.95/5 baseada em classificações dos leitores do *Code Project* [106], uma das maiores comunidades de programadores de *software* de todo o mundo, e recebeu o prémio de melhor projeto e artigo no concurso *Machine Learning* e Inteligência Artificial de 2018.

Ao longo desta secção os principais componentes que constituem este sistema serão explicados, tais como: o *dataset* usado; as classes utilizadas; as diversas técnicas empregues, sendo que, cada uma estará dividida nas três funcionalidades principais – treino, sugestões e previsão para um item; os diferentes cálculos de similaridade usados na técnica colaborativa; e um fluxo demonstrando a relação entre as várias funcionalidades do sistema.

5.1.2.1 Dataset

Este é um dos aspetos críticos e fundamentais num sistema de recomendação. De facto, possuir um *dataset* com grandes quantidades de dados é imprescindível para o bom funcionamento de qualquer SR. Na verdade, a geração de recomendações, o treino e a criação de modelos necessitam de muitos dados para funcionarem de forma precisa e com qualidade [107].

Este ponto tornou-se numa considerável limitação para o presente sistema, visto que a obtenção de dados relativos a classificações de treinos por parte de diferentes utilizadores, ou dados relacionados, num curto espaço de tempo, era uma tarefa inconcebível. Desta forma, foi feito uma pesquisa com o intuito de encontrar serviços relacionados com o fitness (APIs) que disponibilizassem certos dados que permitissem a criação de um *dataset* e, ao mesmo tempo, tentar comunicar com empresas de *fitness* (p.e. Jefit) para a disponibilização de dados. Contudo, ambas as tentativas não tiveram sucesso.

Assim, foram discutidas duas possíveis soluções para o problema:

I. Criação de um *dataset* fictício relacionado com o fitness;
II. Utilização de um *dataset* real não relacionado com o fitness.

A primeira abordagem tem a única vantagem de estar relacionada com o tema do projeto. Na verdade, como o *dataset* seria criado de forma subjetiva e um pouco aleatória, faria com que a geração de recomendações fosse elaborada baseada em dados "não reais", resultando em sugestões não precisas e com pouco fundamento.

Por outro lado, a segunda abordagem possui a desvantagem de não estar relacionada com o tema do projeto, contudo, proporcionava que a geração de recomendações fosse executada baseada em dados de pessoas reais e com fundamento.

BPU, uma empresa de tecnologia dedicada a aperfeiçoar a vida com a utilização de Inteligência Artificial, publicou um artigo sobre a importância de ter um bom dataset [108]. Nele, são feitas as seguintes afirmações: "[…] trabalhando como programador de IA, um dos maiores problemas que encontrei foi - além de criar um modelo para um problema específico - ter um bom *dataset* que se relaciona adequadamente com o problema em questão. Além disso, o conjunto de dados deve ser processado de forma a que o nosso modelo possa entender as informações. Dessa forma, o modelo pode aprender com êxito desse conjunto de dados."; "É de grande importância que os dados sejam adequados para o problema que queremos resolver. Não importa se temos *terabytes* de dados se os dados não estiverem alinhados com o problema." [108].

Tendo em conta a análise efetuada, foi escolhida a segunda opção. De facto, deu-se prioridade à testabilidade do algoritmo com um *dataset* real funcionando como uma prova de conceito, isto é, caso sejam obtidos resultados positivos nos testes efetuados com o *dataset* escolhido, é de esperar que estes resultados também sejam positivos com um conjunto de dados relativo ao fitness.

Sendo assim, procedeu-se à escolha de um *dataset*. Foi escolhido o MovieLens Dataset [109] devido à sua "vasta utilização na educação, pesquisa e indústria. [É de tal forma popular que] o seu *download* é feito centenas de milhares de vezes por ano, refletindo o seu uso em livros populares de programação, em cursos tradicionais e online, e no *software*." [110]. O *dataset* é caracterizado por conter mais de 27 milhões de classificações de 58 mil filmes diferentes por 280 mil utilizadores. Apresenta, principalmente, dois ficheiros em .csv: classificações dos utilizadores no formato <userID, itemID, rating, timestamp> e filmes no formato <itemID, title, tags>.

Para realizar a importação dos dados para o sistema, os ficheiros foram convertidos para .txt com o intuito de diminuir o seu tamanho (741Mb para 2MB). De seguida, foi criada uma determinada lógica de importação para cada ficheiro. Na Figura 35 encontra-se ilustrado um excerto de código para a importação do ficheiro dos *ratings* dos utilizadores.

```
private static UsersRatings LoadUsersRatings()
{
    List<string> lines = GetFileData(RatingsFilePath);

    List<Rating> ratings = new List<Rating>();
    List<User> users = new List<User>();

    int length = lines.Count;
    for (int i = 1; i < length; i++)
    {
        if (lines[i].Trim().Length > 0)
        {
            string[] cols = lines[i].Split(',');

            Rating rating = new Rating(Convert.ToInt32(cols[0].Trim()),
                Convert.ToInt32(cols[1].Trim()),
                Convert.ToDouble(cols[2].Trim()),
                Convert.ToInt32(cols[3].Trim()));
            User user = new User(Convert.ToInt32(cols[0].Trim()));

            ratings.Add(rating);
            users.Add(user);
        }
    }

    return new UsersRatings(users, ratings);
}
```

Figura 35 - Excerto de Código: Importação dos ratings dos utilizadores

5.1.2.2 Diagrama de Classes

Antes da explicação dos importantes componentes que constituem este sistema, é importante apresentá-los e mostrar as suas ligações. Assim, na Figura 36 está presente um diagrama de classes para o presente sistema.

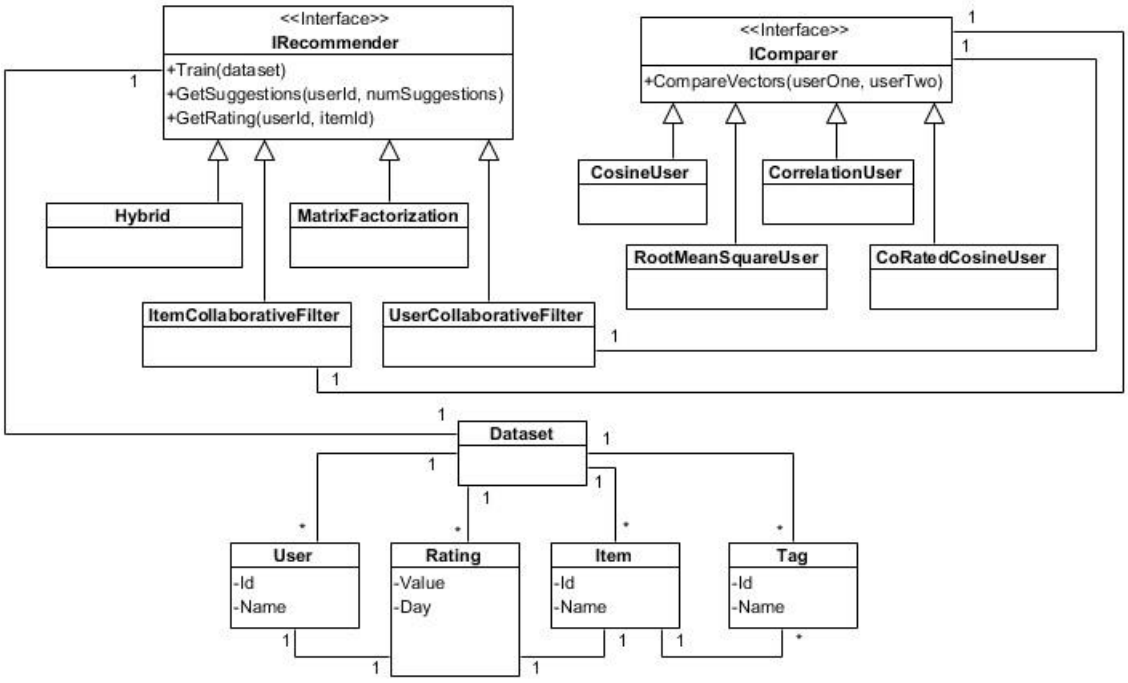


Figura 36 - Diagrama de Classes: Sistema de Recomendação

É possível observar que o padrão *Strategy* encontra-se aplicado em dois contextos, permitindo a utilização de diversas técnicas de programação e de diferentes cálculos para a obtenção da similaridade entre dois utilizadores ou itens.

Cada técnica de recomendação implementa, pelo menos, três algoritmos:

- ***Train*** (treino)**:** para o presente projeto o treino possui duas funções: construção da tabela constituída pelos utilizadores e itens; e realização do treino propriamente dito - treinar de forma iterativa o modelo com o intuito de aperfeiçoar as previsões e melhorar a taxa de sucesso. Visto que apenas uma das técnicas é considerada inteligente (*Matrix Factorization*), somente esta executa a segunda tarefa;
- ***GetSuggestions*** (obter sugestões)**:** tem como objetivo retornar os itens que obtiveram o melhor *score* para um utilizador;
- ***GetRating*** (prever classificação de um item para um utilizador)**:** responsável por prever o grau de apreciação de um determinado item para um utilizador específico.

5.1.2.3 Matrix Factorization

Matrix Factorization é considerada uma técnica inteligente que funciona de maneira diferente das técnicas de filtragem colaborativa. Na verdade, invés de procurar utilizadores ou itens semelhantes para realizar sugestões, cria um modelo matemático para realizar previsões.

Para esta técnica, o método de **treino**, tal como descrito no Diagrama de Classes, é responsável por treinar de forma iterativa o modelo. O treino é uma tarefa computacionalmente complexa envolvendo um fluxo longo e diversas aplicações matemáticas.

Na Figura 37 encontra-se presente um diagrama de atividades contemplando todas as etapas deste método.

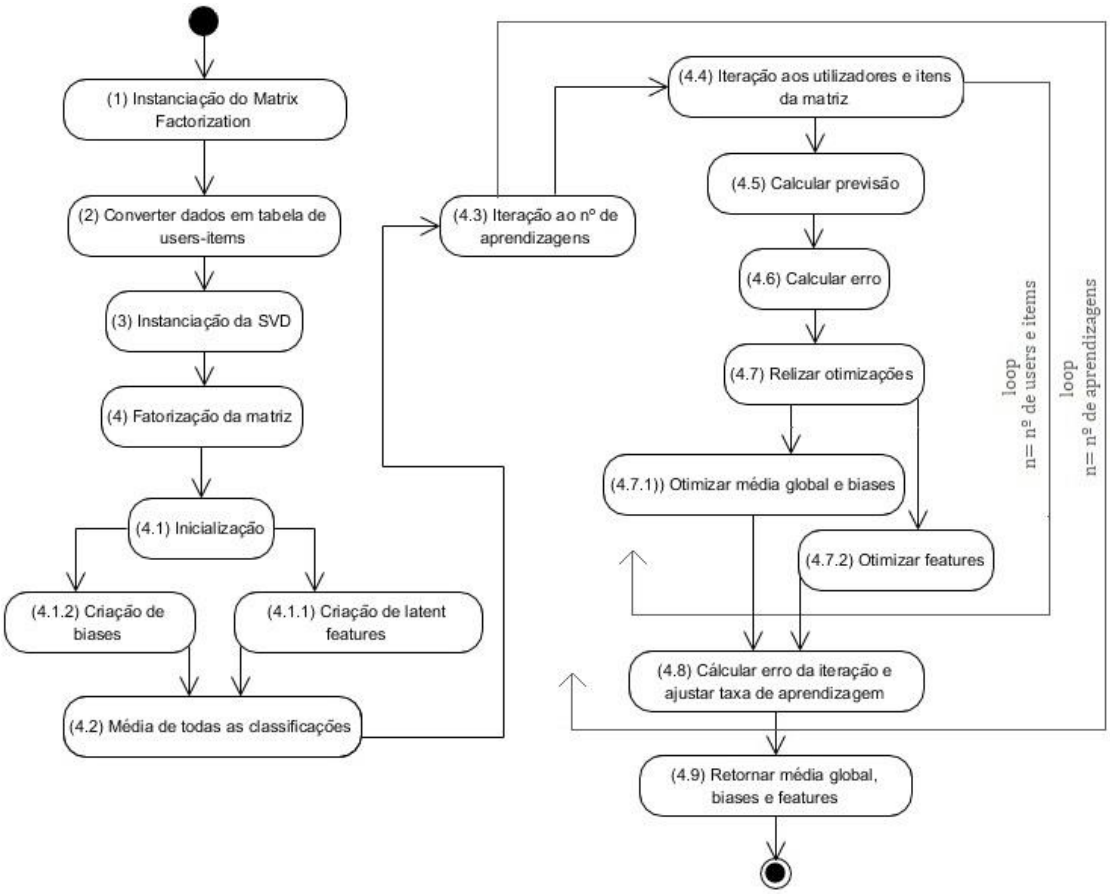


Figura 37 - Diagrama de atividades do treino da técnica Matrix Factorization

Em primeiro lugar, a classe *MatrixFactorizationRecommender* é instanciada com o número de *latent features* a ser usado no processo de aprendizagem. Esse número de *latent features* é uma das duas dimensões das duas matrizes a serem criadas - utilizadores e itens.

Não existe um número certo de *latent features*, estando este dependente de cada técnica de recomendação. Contudo, não pode ser um número baixo, porque origina a não aprendizagem dos dados por parte do modelo, no entanto, também não pode ser muito alto para não causar o *overfit* dos dados.

O próximo passo é a construção da tabela utilizadores-itens ou matriz de ratings/classificações, constituída por todos os utilizadores, itens e classificações. O método responsável por esta transformação encontra-se na Figura 38 e as classes usadas na Figura 39.

```
public UserItemRatingsTable GetUserItemRatingsTable()
{
    UserItemRatingsTable table = new UserItemRatingsTable();

    table.UserIndexToID = db.Users.OrderBy(x => x.ID).Select(x => x.ID).Distinct().ToList();
    table.ItemIndexToID = db.Items.OrderBy(x => x.ID).Select(x => x.ID).Distinct().ToList();

    foreach (int userId in table.UserIndexToID)
    {
        table.Users.Add(new UserItemRatings(userId, table.ItemIndexToID.Count));
    }

    var userItemRatingGroup = db.Ratings
        .Select(g => new { g.UserID, g.ItemID, Rating = g.RatingValue })
        .ToList();

    foreach (var userAction in userItemRatingGroup)
    {
        int userIndex = table.UserIndexToID.IndexOf(userAction.UserID);
        int itemIndex = table.ItemIndexToID.IndexOf(userAction.ItemID);

        table.Users[userIndex].ItemRatings[itemIndex] = userAction.Rating;
    }

    return table;
}
```

Figura 38 - Excerto de código: Transformação da tabela de classificações

```
public class UserItemRatingsTable
{
    public List<UserItemRatings> Users { get; set; }

    public List<int> UserIndexToID { get; set; }

    public List<int> ItemIndexToID { get; set; }

    […]
}
```

```
public class UserItemRatings
{
    public int UserID { get; set; }

    public double[] ItemRatings { get; set; }

    public double Score { get; set; }

    […]
}
```

Figura 39 - Excerto de Código: Classes usadas na importação

Em primeiro lugar, uma instância da classe *UserItemRatingsTable* é instanciada para que, de seguida, cada utilizador seja instanciado (*UserItemRatings*) contendo um vetor com as suas classificações.

A Figura 40 ilustra um exemplo de uma matriz de classificações com a existência das duas matrizes *latent features*. Ao longo desta secção as *latent features* serão exploradas.

		Items							
		I1	I2	I3	I4	I5	X	Y	Z
Users	*U1*	-	4	-	-	3	-1	2	3
	U2	3	-	-	2	-	4	3	2
	U3	-	3.5	-	3	-	3	-1	4
	U4	-	1	-	-	4	1	1	3
	U5	2	-	4	3	-	1	2	4
	A	3	2	2	1	4			
	B	-1	3	1	4	2			
	C	1	2	-1	3	3			

Figura 40 – Exemplo de matriz constituída pelos utilizadores, itens e *latent features*

Tendo a matriz de *ratings* criada, é altura para aplicar a técnica SVD. Como já descrito no Estado de Arte, esta técnica fatoriza uma matriz de itens do utilizador em matrizes mais pequenas, que podem ser usadas para preencher as classificações em falta. A técnica é então instanciada com o número de *latent features* e o número de iterações ou fases de aprendizagem. Este último parâmetro é o critério de paragem pelo qual o algoritmo passará para melhorar seu modelo de previsão.

Depois da instanciação da classe *SingularValueDecomposition*, o método de fatorizar a matriz é invocado passando por parâmetro a matriz de *ratings*. É neste método que o modelo responsável por realizar as previsões é construído.

Inicialmente, todas os dados necessários para a construção do modelo são inicializados: as duas matrizes de *latent features* (utilizadores e itens) são inicializadas com valores aleatórios para que depois, com determinados ajustes e otimizações, esses valores aproximarem-se dos valores reais; e as *biases* referentes aos utilizadores e itens.

Os utilizadores possuem comportamentos diferentes no que toca à classificação de itens. Isto é, certos utilizadores podem possuir a mesma apreciação por um item, contudo, uns podem classificá-lo com alta pontuação e outros o oposto (*biase* do utilizador). Da mesma forma, itens diferentes podem ter certas características que levam os utilizadores a classificá-los mais alto ou mais baixo do que outros (*biase* do item) [111]. Portanto, é importante entender que uma classificação pode conter vários significados, o que leva à inclusão de *biases* na presente técnica.

O método de inicialização dos dados encontra-se presente na Figura 41.

```
private void Initialize(UserItemRatingsTable ratings)
{
    numUsers = ratings.Users.Count;
    numItems = ratings.Users[0].ItemRatings.Length;

    Random rand = new Random();

    userFeatures = new double[numUsers][];
    for (int userIndex = 0; userIndex < numUsers; userIndex++)
    {
        userFeatures[userIndex] = new double[numFeatures];

        for (int featureIndex = 0; featureIndex < numFeatures; featureIndex++)
        {
            userFeatures[userIndex][featureIndex] = rand.NextDouble();
        }
    }

    itemFeatures = new double[numItems][];
    for (int itemIndex = 0; itemIndex < numItems; itemIndex++)
    {
        itemFeatures[itemIndex] = new double[numFeatures];

        for (int featureIndex = 0; featureIndex < numFeatures; featureIndex++)
        {
            itemFeatures[itemIndex][featureIndex] = rand.NextDouble();
        }
    }

    userBiases = new double[numUsers];
    itemBiases = new double[numItems];
}
```

Figura 41 - Excerto de Código: Inicialização de dados

Depois da inicialização dos dados estar concluída, é feita a média de todas as classificações presentes na matriz, que será usada em futuros cálculos matemáticos, e, posteriormente, o processo de aprendizagem iterativo é iniciado, como ilustrado na Figura 42.

```
public SvdResult FactorizeMatrix(UserItemRatingsTable ratings)
{
    Initialize(ratings);

    double squaredError;
    int count;
    List<double> rmseAll = new List<double>();

    averageGlobalRating = GetAverageRating(ratings);

    for (int i = 0; i < learningIterations; i++)
    {
        squaredError = 0.0;
        count = 0;

        for (int userIndex = 0; userIndex < numUsers; userIndex++)
        {
            for (int itemIndex = 0; itemIndex < numItems; itemIndex++)
            {
                if (ratings.Users[userIndex].ItemRatings[itemIndex] != 0)
                {
                    double predictedRating = averageGlobalRating + userBiases[userIndex] +
                        itemBiases[itemIndex] + Matrix.GetDotProduct(userFeatures[userIndex], itemFeatures[itemIndex]);

                    double error = ratings.Users[userIndex].ItemRatings[itemIndex] - predictedRating;

                    squaredError += Math.Pow(error, 2);
                    count++;

                    averageGlobalRating += learningRate * (error - regularizationTerm * averageGlobalRating);
                    userBiases[userIndex] += learningRate * (error - regularizationTerm * userBiases[userIndex]);
                    itemBiases[itemIndex] += learningRate * (error - regularizationTerm * itemBiases[itemIndex]);

                    for (int featureIndex = 0; featureIndex < numFeatures; featureIndex++)
                    {
                        userFeatures[userIndex][featureIndex] += learningRate * (error * itemFeatures[itemIndex][featureIndex]
                            - regularizationTerm * userFeatures[userIndex][featureIndex]);

                        itemFeatures[itemIndex][featureIndex] += learningRate * (error * userFeatures[userIndex][featureIndex]
                            - regularizationTerm * itemFeatures[itemIndex][featureIndex]);
                    }
                }
            }
        }

        squaredError = squaredError / count;

        rmseAll.Add(squaredError);

        learningRate *= learningDescent;
    }

    return new SvdResult(averageGlobalRating, userBiases, itemBiases, userFeatures, itemFeatures);
}
```

Figura 42 - Excerto de Código: Factorização da matriz

Cada iteração deste processo tem associado um erro e uma taxa de aprendizagem, sendo esta ajustada no fim de cada iteração. O principal objetivo é minimizar o erro de forma a aprender e construir um modelo melhor. Foi utilizada à fórmula do *Mean Squared Error* para obtenção do erro devido ao seu cálculo simples e à sua utilização comum [112][113]. A equação é a seguinte:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{7}$$

, onde Y_i é o valor previsto, $\hat{Y}_i$ o valor real e n o número de classificações.

Relativamente à taxa de aprendizagem, esta é usada para ajustar os valores nas matrizes de *latent features*, aproximando os seus valores dos reais. É, também, usada para regularizar as etapas entre cada iteração, como ilustrado no algoritmo desenvolvido, onde a taxa é reduzida em 1% em cada iteração, sendo um valor ajustável.

Sobre cada iteração do ciclo da aprendizagem, são realizadas mais duas - uma aos utilizadores (linhas da matriz) e outro aos itens (colunas da matriz). Em primeiro lugar, é verificado se o valor da célula corrente, dado pela linha e coluna corrente, é diferente de zero, pois, caso seja, não é possível aprender com ela. Por outro lado, sendo um valor diferente de zero, é calculada a classificação prevista com a seguinte equação:

$$CV = \bar{a} + b + c + \vec{u} * \vec{\imath} \quad (8)$$

, onde $\bar{a}$ é a media das classificações, b é a biase do utilizador, c é a biase do item e $\vec{u} * \vec{\imath}$ é o produto escalar dos vetores de *latent features*.

O valor previsto podia ser calculado unicamente através do produto escalar entre os vetores, contudo, as adições dos outros atributos tornam o modelo de previsão mais flexível e regularizado, porque leva em consideração diferentes variáveis.

Tendo calculado o valor previsto, é realizado o cálculo do erro através da subtração entre o valor previsto e o valor real (técnica do MSE).

Calculado o erro, a próxima etapa é a otimização e ajuste de todas as variáveis: *latent features*, média de classificações e *biases*. Esse procedimento é realizado através do algoritmo *Gradient Descent.* Existem diversas variações para esse algoritmo, sendo o *Stochastic Gradient Descent* [113] a escolhida para este projeto. Sucintamente, a otimização usando este algoritmo é feita individualmente, um valor de cada vez, invés de todos os valores simultaneamente.

De forma a atualizar os valores das matrizes de *latent features*, foi utilizada a seguinte fórmula de regularização do *Gradient Descent* [113]:

$$\Delta q_{if} = \lambda(\in p_{uf} - \gamma q_{if}), \qquad \Delta p_{uf} = \lambda(\in q_{if} - \gamma p_{uf}) \quad (9)$$

, onde Δq_{if} *e* Δp_{uf} são as alterações dos valores para o utilizador corrente e item das matrizes de *latent features*, λ é a taxa de aprendizagem, $\in$ é o erro associado à previsão do valor atual e γ é o termo de regularização. De forma a que não existam valores altos de *latent features*, não originando o *overfit* dos dados, o termo de regularização multiplica pelo valor da *feature* e subtrai pelo outro valor do termo.

Além de atualizar os valores das matrizes de *features*, também existe a necessidade de otimizar outras variáveis: média de classificações e *biases*. A equação é semelhante à apresentada anteriormente, com a exceção da remoção do multiplicador do erro (p_{uf} *e* q_{if}).

$$\Delta v = \lambda(\in -\gamma v) \quad (10)$$

, onde Δv são as alterações nos valores da média de classificações e nas *biases* dos utilizadores e itens, λ é a taxa de aprendizagem, $\in$ é o erro associado à previsão do valor atual e γ é o termo de regularização.

O processo descrito é repetido para todos os valores na matriz de classificações, sendo associado, como referido anteriormente, um erro a cada iteração (calculado utilizando a técnica MSE).

Finalmente, um objeto SVD contendo a média de classificações, *latent features* e *biases* dos utilizadores e itens já otimizadas é retornado, possibilitando a realização de previsões.

Depois de criado e treinado o modelo, a **obtenção de sugestões** torna-se simples. Para todos os itens que o utilizador ainda não atribuiu qualquer *rating*, é previsto o seu valor através da mesma equação presente no processo de factorização da matriz (equação (11)).

Na Figura 43 encontra-se um excerto de código para a obtenção de sugestões.

```
private double GetRatingForIndex(int userIndex, int articleIndex)
{
    return svd.AverageGlobalRating + svd.UserBiases[userIndex] + svd.ItemBiases[articleIndex]
        + Matrix.GetDotProduct(svd.UserFeatures[userIndex], svd.ItemFeatures[articleIndex]);
}

public List<Suggestion> GetSuggestions(int userId, int numSuggestions)
{
    int userIndex = ratings.UserIndexToID.IndexOf(userId);
    UserItemRatings user = ratings.Users[userIndex];
    List<Suggestion> suggestions = new List<Suggestion>();

    for (int articleIndex = 0; articleIndex < ratings.ItemIndexToID.Count; articleIndex++)
    {
        // If the user in question hasn't rated the given article yet
        if (user.ItemRatings[articleIndex] == 0)
        {
            double rating = GetRatingForIndex(userIndex, articleIndex);

            suggestions.Add(new Suggestion(userId, ratings.ItemIndexToID[articleIndex], rating));
        }
    }

    suggestions.Sort((c, n) => n.Rating.CompareTo(c.Rating));

    return suggestions.Take(numSuggestions).ToList();
}
```

Figura 43 – Excerto de Código: *GetSuggestions()* no Matrix Factorization

De seguida, são retornados os itens com as melhores classificações.

A **previsão da classificação de um item para um utilizador** utiliza, também, a equação referida anterior, retornando o resultado obtido.

5.1.2.4 Medidas de cálculo de semelhança

Um dos aspetos fundamentais no algoritmo de filtragem colaborativa é o cálculo da similaridade entre utilizadores/itens. Existem diversas formas para calcular a semelhança entre os utilizadores/itens, sendo que quanto maior o valor, maior o grau de similaridade. Para este projeto foram utilizados os seguintes cálculos:

- ***Pearson Correlation Similarity*** [114][115]***:*** mede o grau de correlação entre duas variáveis. Assume valores entre -1 e 1, em que -1 representa uma correlação negativa, 1 uma correlação perfeita e 0 as variáveis não dependem uma da outra. A fórmula de cálculo é apresentada na equação (12) e o excerto de código encontra-se na Figura 44:

$$PCS(a,u) = \frac{\sum_{i=1}^{n}(r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{n}(r_{a,i} - \bar{r}_a)^2 * (r_{u,i} - \bar{r}_u)^2}} \quad (13)$$

, onde $r_{a,i}$ é a classificação do item corrente correspondente ao utilizador 1, $\bar{r}_a$ é a média das classificações do utilizador 1, $r_{u,i}$ é a classificação do item corrente correspondente ao utilizador 2 e $\bar{r}_u$ é a média das classificações do utilizador 2.

```
public double CompareVectors(double[] userFeaturesOne, double[] userFeaturesTwo)
{
    double average1 = 0.0;
    double average2 = 0.0;
    int count = 0;

    for (int i = 0; i < userFeaturesOne.Length; i++)
    {
        if (userFeaturesOne[i] != 0 && userFeaturesTwo[i] != 0)
        {
            average1 += userFeaturesOne[i];
            average2 += userFeaturesTwo[i];
            count++;
        }
    }

    average1 /= count;
    average2 /= count;

    double sum = 0.0;
    double squares1 = 0.0;
    double squares2 = 0.0;

    for (int i = 0; i < userFeaturesOne.Length; i++)
    {
        if (userFeaturesOne[i] != 0 && userFeaturesTwo[i] != 0)
        {
            sum += (userFeaturesOne[i] - average1) * (userFeaturesTwo[i] - average2);
            squares1 += Math.Pow(userFeaturesOne[i] - average1, 2);
            squares2 += Math.Pow(userFeaturesTwo[i] - average2, 2);
        }
    }

    return sum / Math.Sqrt(squares1 * squares2);
}
```

Figura 44 - Excerto de Código: *Pearson Correlation Similarity*

- ***Cosine Similarity*** [114]***:*** são considerados dois vetores relativos a itens no espaço multidimensional do utilizador. A semelhança entre eles é medida através do cálculo do cosseno do ângulo entre esses dois vetores. A fórmula de cálculo é apresentada na equação (14) e o excerto de código encontra-se na Figura 45:

$$CS(a,u) = \frac{\sum_{i=1}^{n} a_i u_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} u_i^2}} \quad (15)$$

, onde a_i é a classificação do item corrente correspondente ao utilizador 1 e u_i é a classificação do item corrente correspondente ao utilizador 2.

```
public double CompareVectors(double[] userFeaturesOne, double[] userFeaturesTwo)
{
    double sumProduct = 0.0;
    double sumOneSquared = 0.0;
    double sumTwoSquared = 0.0;

    for (int i = 0; i < userFeaturesOne.Length; i++)
    {
        sumProduct += userFeaturesOne[i] * userFeaturesTwo[i];
        sumOneSquared += Math.Pow(userFeaturesOne[i], 2);
        sumTwoSquared += Math.Pow(userFeaturesTwo[i], 2);
    }

    return sumProduct / (Math.Sqrt(sumOneSquared) * Math.Sqrt(sumTwoSquared));
}
```

Figura 45 - Excerto de Código: *Cosine Similarity*

- ***Co-Rated Cosine Similarity*****:** a lógica para o presente cálculo é semelhante à similaridade do cosseno com as seguintes exceções: apenas são tomados em conta itens em que ambos os utilizadores classificaram; e a divisão só é aplicada caso os quadrados das somas dos dois vetores seja maior que 0. O excerto de código está presente na Figura 46.

```
public double CompareVectors(double[] userFeaturesOne, double[] userFeaturesTwo)
{
    double sumProduct = 0.0;
    double sumOneSquared = 0.0;
    double sumTwoSquared = 0.0;

    for (int i = 0; i < userFeaturesOne.Length; i++)
    {
        if (userFeaturesOne[i] != 0 && userFeaturesTwo[i] != 0)
        {
            sumProduct += userFeaturesOne[i] * userFeaturesTwo[i];
            sumOneSquared += Math.Pow(userFeaturesOne[i], 2);
            sumTwoSquared += Math.Pow(userFeaturesTwo[i], 2);
        }
    }

    if (sumOneSquared > 0 && sumTwoSquared > 0)
    {
        return sumProduct / (Math.Sqrt(sumOneSquared) * Math.Sqrt(sumTwoSquared));
    }
    else
    {
        return double.NegativeInfinity;
    }
}
```

Figura 46 - Excerto de Código: *Co-Rated Cosine Similarity*

- ***Root Mean Square Similarity*** [116]**:** calcula a distância média entre os itens. A fórmula de cálculo é apresentada na equação (16) e o excerto de código encontra-se na Figura 47:

$$RMS(a,u) = \sqrt{\frac{1}{N}\sum(a-u)^2} \tag{17}$$

, onde a é a classificação do item corrente correspondente ao utilizador 1, u a classificação do item corrente correspondente ao utilizador 2 e N é o número de classificações.

```
public double CompareVectors(double[] userFeaturesOne, double[] userFeaturesTwo)
{
    double score = 0.0;

    for (int i = 0; i < userFeaturesOne.Length; i++)
    {
        score += Math.Pow(userFeaturesOne[i] - userFeaturesTwo[i], 2);
    }

    return Math.Sqrt(score / userFeaturesOne.Length);
}
```

Figura 47 - Excerto de Código: *Root Mean Square Similarity*

5.1.2.5 Filtragem colaborativa baseada em utilizadores

A filtragem colaborativa baseada em utilizadores realiza recomendações com base no que os utilizadores considerados semelhantes gostaram.

No método de **treino** desta técnica é gerada uma tabela constituída por utilizadores, itens e os seus ratings, como ilustrado na Figura 48.

Users \ Items	I1	I2	I3	I4	I5
U1	-	4	-	-	3
U2	3	-	-	2	-
U3	-	3.5	-	3	-
U4	-	1	-	-	4
U5	2	-	4	3	-

Figura 48 - Exemplo de tabela para a filtragem colaborativa baseada nos utilizadores

O método de **obtenção de sugestões**, em primeiro lugar, invoca o método *GetNearestNeighbors()* que percorre todos os utilizadores da tabela, calcula, para cada um deles, a semelhança com o utilizador alvo (para quem estão a ser geradas as recomendações) usando uma das medidas de cálculo de semelhança descritas na secção 5.1.2.4, e retorna os utilizadores que possuem o maior valor de semelhança (ver Figura 49).

```
private List<UserItemRatings> GetNearestNeighbors(UserItemRatings user, int numUsers)
{
    List<UserItemRatings> neighbors = new List<UserItemRatings>();

    for (int i = 0; i < ratings.Users.Count; i++)
    {
        if (ratings.Users[i].UserID == user.UserID)
        {
            ratings.Users[i].Score = double.NegativeInfinity;
        }
        else
        {
            ratings.Users[i].Score = comparer.CompareVectors(ratings.Users[i].ItemRatings, user.ItemRatings);
        }
    }

    var similarUsers = ratings.Users.OrderByDescending(x => x.Score);

    return similarUsers.Take(numUsers).ToList();
}
```

Figura 49 – Excerto de Código: Método *GetNearestNeighbors()*

Depois de retornada a lista de utilizadores mais semelhantes, a lista de itens é iterada e, para cada um, é calculada a classificação ponderada baseada no *rating* que cada utilizador semelhante atribuiu ao item corrente. No final é feita a média de todas essas classificações. A equação (18) ilustra o cálculo da classificação para um item:

$$Classificação = \frac{\sum_{i=1}^{n}\left(r_i - \frac{i+1}{100}\right)}{n} \quad (19)$$

, onde r_i é a classificação que o utilizador semelhante corrente atribuiu ao item corrente, $\frac{i+1}{100}$ é o termo de regularização e n é o número de utilizadores semelhantes que classificaram o item corrente.

O termo de regularização funciona de forma a atribuir aos utilizadores mais semelhantes do utilizador alvo (encontram-se numa lista ordenada de forma decrescente pelo grau de semelhança) um maior peso à sua classificação.

Depois de realizados todos os cálculos, são retornados os N itens com a melhor classificação. Na Figura 50 encontra-se um excerto de código que ilustra o que foi dito.

```
public List<Suggestion> GetSuggestions(int userId, int numSuggestions)
{
    int userIndex = ratings.UserIndexToID.IndexOf(userId);
    UserItemRatings user = ratings.Users[userIndex];
    List<Suggestion> suggestions = new List<Suggestion>();

    var neighbors = GetNearestNeighbors(user, neighborCount);

    for (int itemIndex = 0; itemIndex < ratings.ItemIndexToID.Count; itemIndex++)
    {
        if (user.ItemRatings[itemIndex] == 0)
        {
            double score = 0.0;
            int count = 0;
            for (int u = 0; u < neighbors.Count; u++)
            {
                if (neighbors[u].ItemRatings[itemIndex] != 0)
                {
                    score += neighbors[u].ItemRatings[itemIndex] - ((u + 1.0) / 100.0);
                    count++;
                }
            }
            if (count > 0)
            {
                score /= count;
            }

            suggestions.Add(new Suggestion(userId, ratings.ItemIndexToID[itemIndex], score));
        }
    }

    suggestions.Sort((c, n) => n.Rating.CompareTo(c.Rating));

    return suggestions.Take(numSuggestions).ToList();
}
```

Figura 50 – Excerto de Código: Método *GetSuggestions()*

Na **previsão da classificação de um item para um utilizador** o método *GetNearestNeighbors()* é também invocado. De seguida, é realizada a média de todos os ratings fornecidos pelos utilizadores semelhantes ao item selecionado. Devido ao facto de que cada utilizador possui um comportamento diferente e classifica os itens consoante os seus gostos, a média final é normalizada subtraindo o valor de cada classificação do utilizador semelhante corrente pela média de todas as suas classificações, e adicionada a média de todas as classificações do utilizador alvo ao resultado da média final, como ilustrado na Figura 51 e na seguinte equação.

$$Classificação = \frac{\sum_{i=1}^{n} (r_i - \bar{r})}{n} + \bar{u} \quad (20)$$

, onde r_i é a classificação que o utilizador semelhante corrente atribuiu ao item corrente, $\bar{r}$ é a média de todas as classificações do utilizador semelhante corrente, n é o número de utilizadores semelhantes que classificaram o item corrente e $\bar{u}$ é a média de todas as classificações do utilizador alvo.

```
public double GetRating(int userId, int itemId)
{
    UserItemRatings user = ratings.Users.FirstOrDefault(x => x.UserID == userId);
    List<UserItemRatings> neighbors = GetNearestNeighbors(user, neighborCount);

    return GetRating(user, neighbors, itemId);
}


private double GetRating(UserItemRatings user, List<UserItemRatings> neighbors, int itemId)
{
    int articleIndex = ratings.ItemIndexToID.IndexOf(itemId);

    var nonZero = user.ItemRatings.Where(x => x != 0);
    double avgUserRating = nonZero.Count() > 0 ? nonZero.Average() : 0.0;

    double score = 0.0;
    int count = 0;
    for (int u = 0; u < neighbors.Count; u++)
    {
        var nonZeroRatings = neighbors[u].ItemRatings.Where(x => x != 0);
        double avgRating = nonZeroRatings.Count() > 0 ? nonZeroRatings.Average() : 0.0;

        if (neighbors[u].ItemRatings[articleIndex] != 0)
        {
            score += neighbors[u].ItemRatings[articleIndex] - avgRating;
            count++;
        }
    }
    if (count > 0)
    {
        score /= count;
        score += avgUserRating;
    }

    return score;
}
```

Figura 51 - Excerto de Código: Método *GetRating()*

5.1.2.6 Filtragem colaborativa baseada em itens

A filtragem colaborativa (FC) baseada em itens funciona de maneira semelhante à baseada em utilizadores. A grande diferença reside na procura de sugestões. Na verdade, ao encontrar sugestões para um utilizador específico, a técnica encontra itens semelhantes aos que o utilizador já viu. Por exemplo, se o utilizador A gostou do item 1, são encontrados itens semelhantes ao 1 e sugeridos ao utilizador.

O método de **treino** apresenta-se semelhante ao da FC baseada em utilizadores, mas com a adição de novas funcionalidades. Uma delas é a inclusão de *tags*. Assim, cada item possuirá

um conjunto de tags (p.e. o filme *Deadpool* terá ação e comédia como *tags*) que ajudará na procura de itens semelhantes. Em termos de implementação, depois de obtida a tabela de utilizadores-itens, tal como na FC baseada em utilizadores, são anexadas novas linhas à tabela, representando os *tags*, com os valores de 0 ou 1, possuindo ou não o *tag*, respetivamente.

Além disso, a matriz/tabela referida é transposta para que as linhas do utilizador e dos *tags* se tornem colunas e todas as colunas referentes aos itens se tornem linhas para tornar mais simples e rápido o cálculo da semelhança entre itens, pois cada linha é vista como um vetor de elementos. Na Figura 52 encontra-se um exemplo de uma matriz transposta e na Figura 53 um excerto de código da função transpor uma matriz.

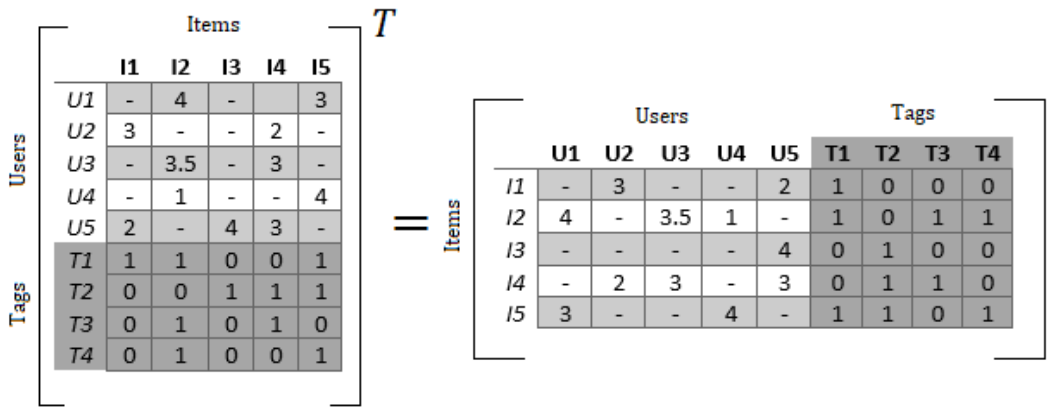


Figura 52 - Exemplo de uma matriz transposta

```
private void FillTransposedRatings()
{
    int features = ratings.Users.Count;
    transposedRatings = new double[ratings.ItemIndexToID.Count][];

    // Precompute a transposed ratings matrix where each row becomes
    //an item and each column a user or tag
    for (int a = 0; a < ratings.ItemIndexToID.Count; a++)
    {
        transposedRatings[a] = new double[features];

        for (int f = 0; f < features; f++)
        {
            transposedRatings[a][f] = ratings.Users[f].ItemRatings[a];
        }
    }
}
```

Figura 53 - Excerto de Código: Transpor a Matriz

O fluxo do método de **obtenção de sugestões** é o seguinte:

1. Obter os itens com maior *rating* classificados pelo utilizador;
2. Invocar o método *GetNearestNeighbors()* com o intuito de obter itens semelhantes para cada item resultante do passo 1;
3. Iterar todos os itens resultantes do passo 2 e realizar a média de todas as classificações feitas pelos restantes utilizadores ao item corrente;
4. Retornar os itens com a maior média.

A **previsão da classificação de um item para um utilizador** é calculada de forma simples. Em primeiro lugar, são calculadas duas médias: sobre todas as classificações atribuídas pelo utilizador selecionado; e sobre todas as classificações do item selecionado. No fim, é retornada a média sobre esses dois valores.

5.1.2.7 Híbrido

Como já referido, os sistemas híbridos são combinações de várias características de diferentes sistemas de recomendação. Dessa forma, para as funcionalidades de previsão de classificação de um item e obtenção de sugestões, os resultados obtidos das diferentes técnicas selecionadas são combinados, com determinadas particularidades relacionadas com a funcionalidade em questão, num único resultado.

A **previsão da classificação de um item para um utilizador** é obtida realizando uma média aritmética sobre os valores resultantes das várias técnicas do sistema (ver Figura 54).

```
public double GetRating(int userId, int itemId)
{
    return classifiers.Select(classifier => classifier.GetRating(userId, itemId)).Average();
}
```

Figura 54 - Excerto de Código: Previsão da Classificação de um Item

A **obtenção de sugestões** é feita de duas formas. Uma delas retribui equitativamente o número de sugestões a realizar por cada técnica e combina todos os resultados num só, isto é, sendo N o número de sugestões pedidas, N será dividido pelo número de técnicas existentes, ficando cada técnica com um valor igual para sugerir (ver Figura 55).

```
public List<Suggestion> GetSuggestions(int userId, int numSuggestions)
{
    List<Suggestion> suggestions = new List<Suggestion>();
    int numSuggestionsEach = (int)Math.Ceiling((double)numSuggestions / classifiers.Count);

    foreach (IRecommender classifier in classifiers)
    {
        suggestions.AddRange(classifier.GetSuggestions(userId, numSuggestionsEach));
    }

    suggestions.Sort((c, n) => n.Rating.CompareTo(c.Rating));

    return suggestions.Take(numSuggestions).ToList();
}
```

Figura 55 - Excerto de Código: Obtenção de Sugestões 1

Em contrapartida, no outro algoritmo a divisão mencionada não é aplicada, sendo que cada técnica gera um certo número de sugestões. De seguida, os itens comuns sugeridos pelas diferentes técnicas são tomados em conta. Na verdade, é retornada uma lista com os N itens ordenados de forma decrescente (em primeiro lugar pelos itens que apareceram mais vezes e, só depois, pelo *score* obtido), sendo N um número inteiro positivo. Na Figura 56 encontra-se ilustrado o que foi dito.

```
public List<Suggestion> GetCommonSuggestions(int userId, int numSuggestions)
{
    int internalSuggestions = 100;

    List<List<Suggestion>> suggestions = new List<List<Suggestion>>();
    foreach (IRecommender classifier in classifiers)
    {
        suggestions.Add(classifier.GetSuggestions(userId, internalSuggestions));
    }

    List<Tuple<Suggestion, int>> final = new List<Tuple<Suggestion, int>>();

    foreach (List<Suggestion> list in suggestions)
    {
        foreach (Suggestion suggestion in list)
        {
            int existingIndex = final.FindIndex(x => x.Item1.ItemID == suggestion.ItemID);

            if (existingIndex >= 0)
            {
                Suggestion highestRated = final[existingIndex].Item1.Rating > suggestion.Rating
                    ? final[existingIndex].Item1 : suggestion;
                final[existingIndex] = new Tuple<Suggestion, int>(highestRated, final[existingIndex].Item2 + 1);
            }
            else
            {
                final.Add(new Tuple<Suggestion, int>(suggestion, 1));
            }
        }
    }

    final = final.OrderByDescending(x => x.Item2).ThenByDescending(x => x.Item1.Rating).ToList();

    return final.Select(x => x.Item1).Take(numSuggestions).ToList();
}
```

Figura 56 - Excerto de Código: Obtenção de Sugestões 2

O **treino** é uma exceção ao que foi dito no primeiro parágrafo. De facto, cada uma das técnicas pertencentes ao sistema híbrido treina o *dataset* de forma independente, sem combinar resultados.

5.1.2.8 Fluxo

Explicadas todas as técnicas e as suas funcionalidades, é importante descrever o fluxo e como funciona o sistema de recomendação. Na Figura 57 encontra-se um diagrama de atividades relacionando todas as funcionalidades deste sistema.

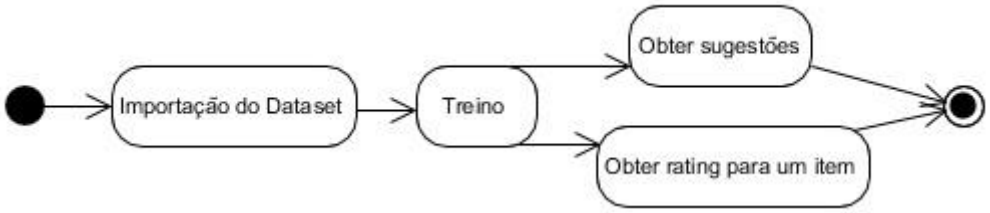


Figura 57 - Diagrama de atividade do Sistema de Recomendação

Na Figura 57 é possível observar que existe um determinado fluxo, isto é, só depois de importado o *dataset* e realizado o seu treino, é que é possível obter sugestões ou prever a classificação de um item para um utilizador.

Dentro do sistema, em primeiro lugar, é escolhida a técnica de recomendação a usar. Usualmente, opta-se pela *Matrix Factorization* ou a Híbrida utilizando a *Matrix Factorization* juntamente com outra(s), devido a ser a técnica com melhores resultados (explorado no capítulo de Avaliação). De seguida, é verificado se um determinado modelo já foi criado. Caso já tenha sido criado, é importado para o sistema permitindo a geração de recomendações para um utilizador. Por outro lado, caso ainda não exista, é feita a importação dos dados do *dataset* e, posteriormente, um modelo é criado e guardado tendo em conta os dados importados. Este fluxo encontra-se ilustrado na Figura 58.

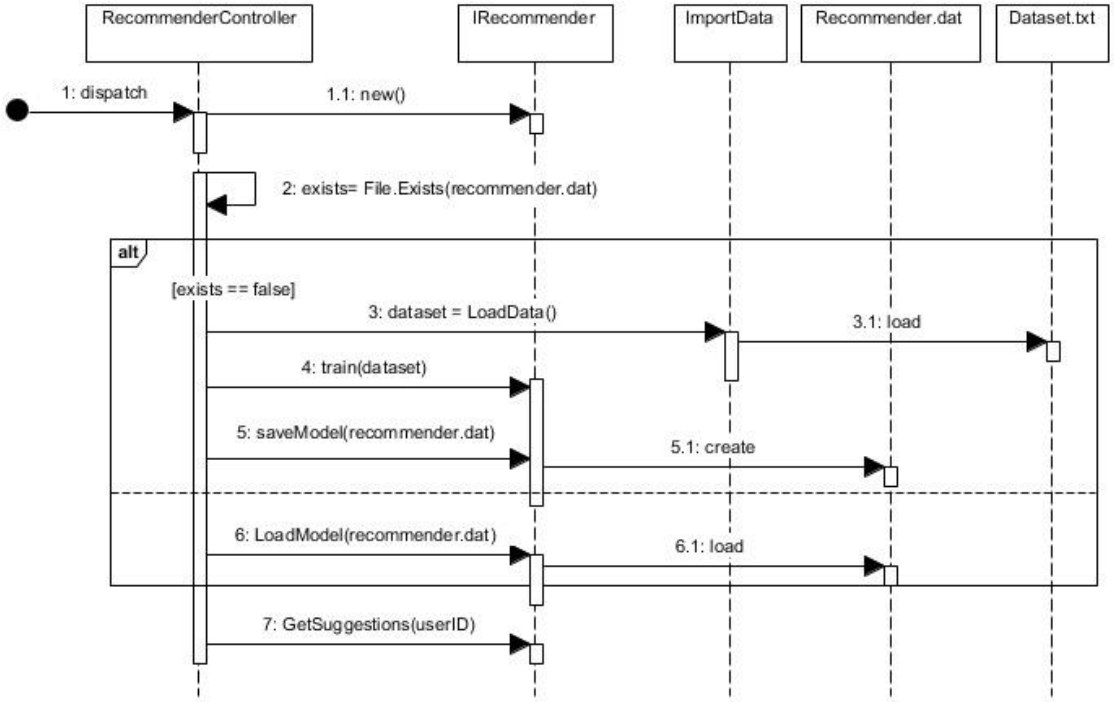


Figura 58 - Diagrama de sequência do sistema de recomendação

5.2 Aplicação Móvel

Como referido ao longo do documento, é importante qua a aplicação móvel possua uma interface gráfica que seja intuitiva e adaptável a qualquer tipo de utilizador, permitindo o acesso de forma ágil às mais variadas funcionalidades do sistema.

Foram utilizadas tecnologias que se encontram num estado consideravelmente avançado e que sejam uma tendência recente, de forma a criar uma aplicação baseada em boas práticas de desenvolvimento de *software*.

Assim, foi usado *React Native* [117], uma *framework* baseada em *React* que possibilita o desenvolvimento de aplicações móveis para os sistemas operativos (SOs) *Android* e *iOS*, utilizando JavaScript.

Adicionalmente, foi utilizado o Expo [118] juntamente com o *React Native*. É uma ferramenta que permite uma simples e rápida criação de aplicações *React* utilizando os dois SOs, concebendo um fácil acesso às APIs nativas do dispositivo (p.e câmara ou microfone) sem necessitar de qualquer configuração ou alteração no código nativo. Com o Expo, o processo de implantação e teste torna-se, também, mais simples.

O ambiente de desenvolvimento escolhido para a implementação da aplicação móvel foi o *Visual Studio Code* [119].

Nesta secção serão descritos os componentes mais importantes da aplicação móvel que permitiram a criação da app. Assim, em primeiro lugar, serão descritas as principais características do sistema desenvolvido. De seguida, serão referidas as bibliotecas usadas, e finalmente, um fluxo da interface gráfica de maneira a tornar percetível o funcionamento da aplicação.

5.2.1 Características do sistema

De forma a criar um sistema baseado em boas práticas do *software* promovendo a fácil manutenção e reutilização, foram tomadas determinadas decisões. Nesta secção serão referidas as características mais importantes do sistema e a forma como foram implementadas.

5.2.1.1 Componentes

O React permite definir componentes como classes possuindo, obrigatoriamente, o método *render()* que é responsável por definir diversos elementos da interface gráfica.

Assim, os componentes podem receber um conjunto de dados acessíveis através da variável *props* (propriedades) e retornar um elemento *React*. Este processo privilegia a reutilização, visto que um componente pode ser invocado em diferentes elementos, não sendo necessária a repetição de código.

Para este projeto, definiu-se a utilização de uma boa prática tendo em vista o aproveitamento das funcionalidades que os componentes disponibilizam, principalmente, a reutilização, como referido anteriormente. Assim, cada funcionalidade terá um ecrã principal (componente pai) que invocará outros componentes (componentes filhos), podendo estes, serem invocados por outros ecrãs.

5.2.1.2 Gestão de estado

Cada componente do *React Native* contém um estado constituído pelos dados a serem mostrados ao utilizador: dados em *cache*, respostas do servidor, dados locais, entre outros. O fluxo de dados no React pode ser feito unidireccionalmente, isto é, um componente pai envia os dados para o componente filho. Contudo, existem situações em que o fluxo não é assim tão simples. Por exemplo, quando um componente pai envia dados para vários componentes filhos e estes para outros componentes. Neste caso, a manutenção e atualização do estado torna-se numa tarefa complexa.

Desta maneira, foi utilizada a biblioteca ou padrão *Redux* [120]. É um gestor de estados para aplicações *JavaScript* que segue determinados padrões de design: *observer, flux, CQRS, event sourcing e singleton* [121].

O *Redux* é composto por uma *store* centralizada onde são guardados todos os estados da aplicação. Esta não é diretamente acessível ou mutável. Para ser acedida é necessário a emissão de uma ação (*action*), ou seja, um objeto que descreve qual o comportamento a

realizar. Com isto, o estado não é alterado diretamente, as mutações ou alterações pretendidas encontram-se explicitas em ações (ver Figura 59).

```
export const setPlans = plans => {
    return {
        type: SET_PLANS,
        plans: plans
    };
};
```

Figura 59 - Excerto de Código: Exemplo de uma ação

Cada ação tem um redutor (*reducer*) que tem a responsabilidade de comunicar com a *store* e alterar o estado pretendido (ver Figura 60). Por fim, todos os componentes que usam o estado atualizado, são notificados. Este fluxo encontra-se ilustrado na Figura 61.

```
const initialState = {
    plans: []
};

const reducer = (state = initialState, action) => {
    switch (action.type) {
        case SET_PLANS:
            return {
                ...state,
                plans: action.plans
            };
        default:
            return state;
    }
};
```

Figura 60 - Excerto de Código: Exemplo de um *reducer*

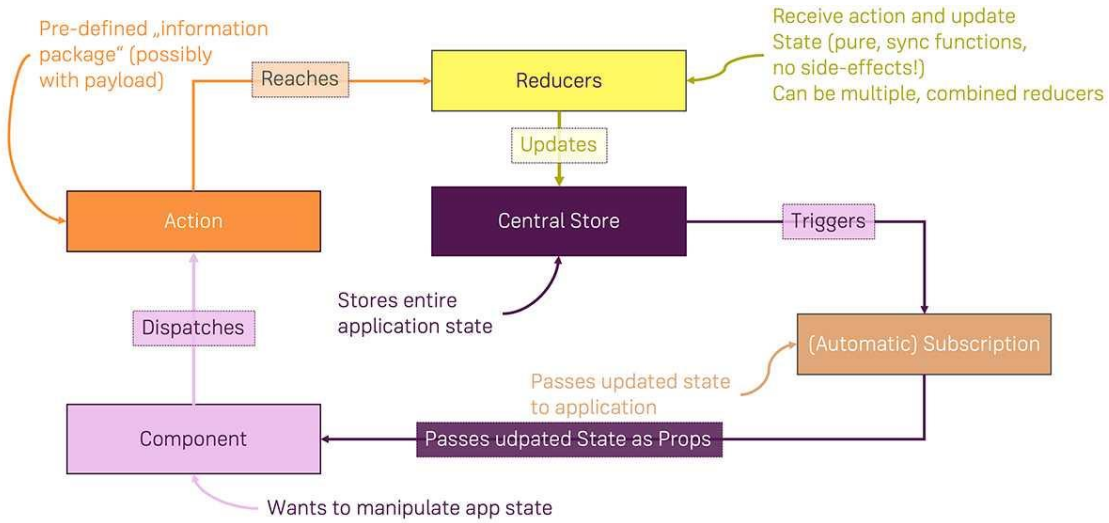


Figura 61 - Funcionamento do Redux [122]

5.2.1.3 AsyncStorage

AsyncStorage é um sistema de armazenamento simples, assíncrono e persistente que utiliza chaves globais para identificação dos dados a serem guardados [123]. A sua utilização prende-se, sobretudo, quando é necessário guardar dados que a aplicação precisa de usar, mesmo que esta seja fechada.

Para o presente projeto a sua utilização foi fundamental para armazenar dados relacionados com o utilizador - *token* e email, e com configurações - métricas preferidas, idioma e estado do *setup*.

5.2.1.4 Navegação

React Navigation tem a principal finalidade de criar o fluxo de navegação da aplicação através dos seus diversos navegadores [124]. A Figura 62 ilustra os principais tipos de navegadores.

Figura 62 - Diferentes tipos de navegadores no *React Navigation* (adaptado de [125])

Cada um dos navegadores tem uma determinada responsabilidade:

- ***Tab Navigator*:** barra na parte inferior do ecrã que permite alternar entre ecrãs;
- ***Drawer Navigator***: menu lateral com várias opções;
- ***Stack Navigator*:** disponibiliza uma maneira da aplicação executar transições entre ecrãs, onde cada "novo ecrã" é colocado no topo da *stack*;
- ***Switch Navigator*:** não se encontra presente na Figura 62, mas tem o objetivo de mostrar apenas um ecrã de cada vez.

Na aplicação móvel todos os navegadores apresentados foram utilizados. Na Figura 63 encontra-se ilustrado a relação entre os navegadores e todos os ecrãs criados.

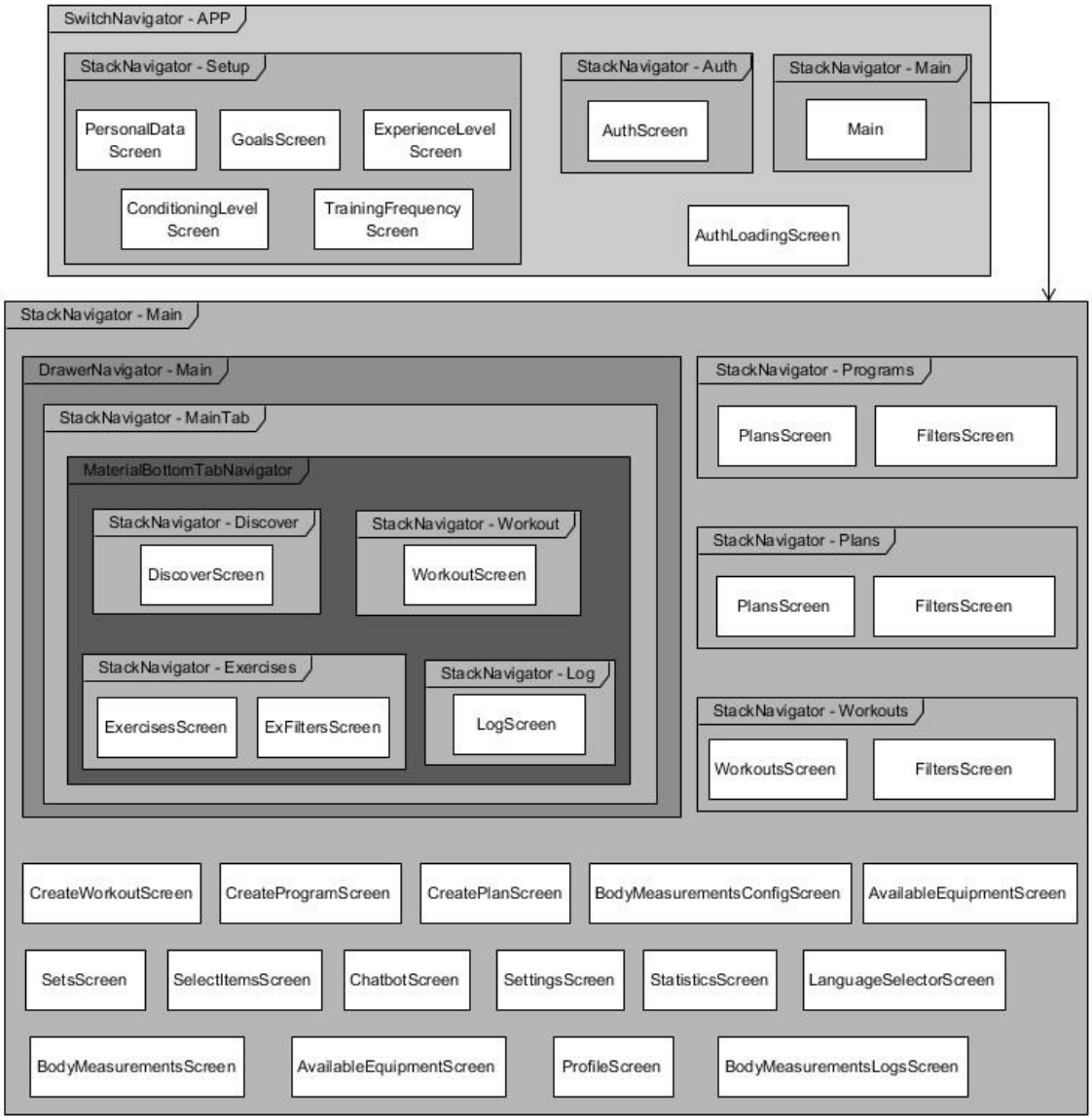


Figura 63 – Relação entre navegadores e ecrãs na aplicação móvel

5.2.1.5 Autenticação

Uma das finalidades presente nos Requisitos não funcionais é a existência da funcionalidade de autenticação. O sistema de autenticação presente neste projeto baseia-se no par de dados email e *password* ou através da rede social *Facebook*. Desta forma, o utilizador pode, facilmente, criar uma conta.

Funcionalidades como o login e recuperação da *password* também foram implementadas. Na secção dos Casos de Uso este tema é apresentado.

5.2.1.6 Multilingue

Outro dos requisitos não funcionais definido consiste na existência de um sistema que promova o uso de vários idiomas. Para dar resposta a este requisito, a aplicação móvel utiliza a biblioteca i18n[3]. É um módulo de tradução simples que funciona, predominantemente, com ficheiros de configuração JSON.

Para utilizar esta biblioteca foi necessário, para cada idioma, criar um ficheiro de configuração JSON composto por chaves (identificador da tradução) e as traduções associadas. Na Figura 64 encontra-se um exemplo para ficheiros na língua portuguesa e inglesa.

language > {} pt_PT.json

```
"discover": {
  "headers": {
    "workouts": "Categorias de treinos",
    "programs": "Programas para si",
    "plans": "Planos para si"
  },
```

language > {} en_US.json

```
"discover": {
  "headers": {
    "workouts": "Workout categories",
    "programs": "Programs for you",
    "plans": "Plans for you"
  },
```

Figura 64 – Excertos dos ficheiros de tradução pt-PT e en-US

Quando a aplicação é iniciada pela primeira vez, o sistema verifica qual é o idioma do dispositivo através da *Localization* API do Expo [126]. Caso o idioma não exista no sistema, a língua inglesa será definida como *default*.

5.2.2 Bibliotecas

Nos dias de hoje a utilização de React Native encontra-se cada vez mais presente no desenvolvimento de aplicações móveis, sendo a sua comunidade cada vez maior. Por conseguinte, existem múltiplas bibliotecas de boa qualidade criadas por empresas ou por pessoas dentro dessa comunidade, que ajudam a economizar tempo e a criar interfaces gráficas de forma mais rápida [127].

Na Tabela 8 estão presentes todas as bibliotecas usadas. Para cada uma delas é apresentado o nome, a descrição e a reputação, isto é, o número de estrelas do seu repositório *GitHub*.

[3] https://github.com/mashpie/i18n-node

Tabela 8 – Bibliotecas usadas

Nome	Descrição	Reputação[4]
react-native-animatable **[128]**	Transições declarativas e animações	578
react-native-app-intro-slider **[129]**	Slider com vários ecrãs	800
react-native-calendars **[130]**	Diversos tipos de calendário	4710
react-native-collapsible **[131]**	Componente dobrável	1572
react-native-dialog **[132]**	Componente de diálogo	209
react-native-elements **[133]**	Diversos componentes de UI	17133
react-native-gifted-chat **[134]**	Chat UI	7979
react-native-loading-spinner-overlay **[135]**	Componente *loading*	1416
react-native-material-menu **[136]**	Menu com opções	223
react-native-material-textfield **[137]**	Componente de *input* de dados	630
react-native-modal **[138]**	Componente modal personalizável	2845
react-native-paper **[139]**	Diversos componentes de UI	4206
react-native-parallax-scroll-view **[140]**	Componente com animação	1814
react-native-picker **[141]**	Componente para escolha de dados	1519
react-native-picker-select **[142]**	Componente para escolha de dados	574
react-native-progress **[143]**	Indicador de progresso	2514
react-native-pure-chart **[144]**	Gráficos	198
react-native-really-awesome-button **[145]**	Múltiplos botões	791
react-native-search-box **[146]**	Caixa de pesquisa	370
react-native-sectioned-multi-select **[147]**	Componente para escolha de dados	347
react-native-snap-carousel **[148]**	Componente de pré-visualização de dados	5954
react-native-sortable-listview **[149]**	Lista ordenável	839
react-native-stopwatch-timer **[150]**	Contador e temporizador	45
react-native-swipeable **[151]**	Componente de *swipe*	842
react-native-ui-kitten **[152]**	Diversos componentes de UI	4922

[4] Acedido a 25 de setembro de 2019

5.2.3 Interface gráfica

De maneira a simplificar a explicação da interface gráfica, foram criados cinco fluxos. Cada um deles será apresentado com uso a *screenshots*.

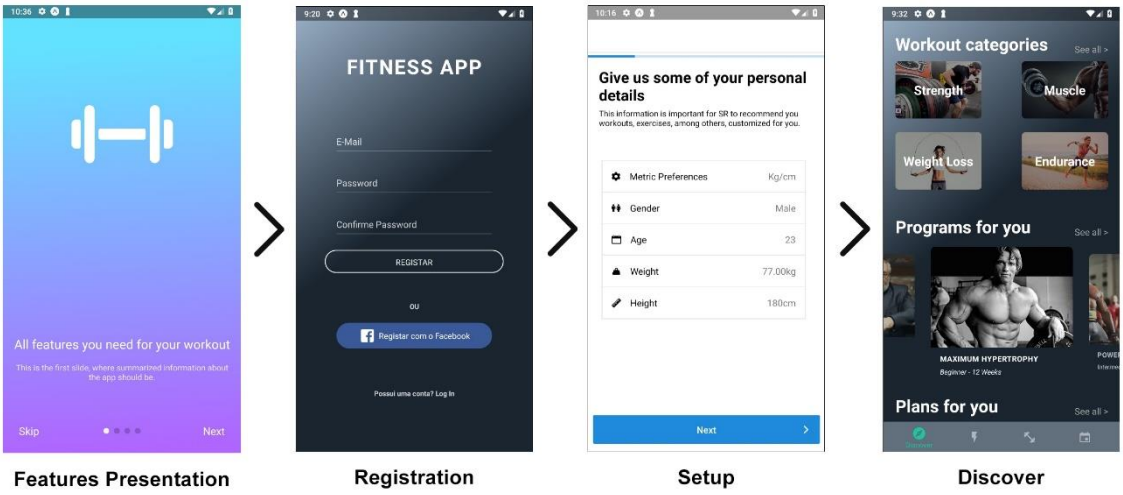


Figura 65 - Fluxo 1: Registo na aplicação

Na Figura 65 está presente o fluxo inicial de quando um utilizador se regista na aplicação móvel. Em primeiro lugar, são descritas as características e funcionalidades principais da app através do uso de *sliders*. De seguida, são pedidos determinados dados para o registo e *setup*, como email, password, dados pessoais, entre outros. Por fim, o utilizador é encaminhado para o ecrã *Discover* onde tem à disposição diversos dados *default* do sistema.

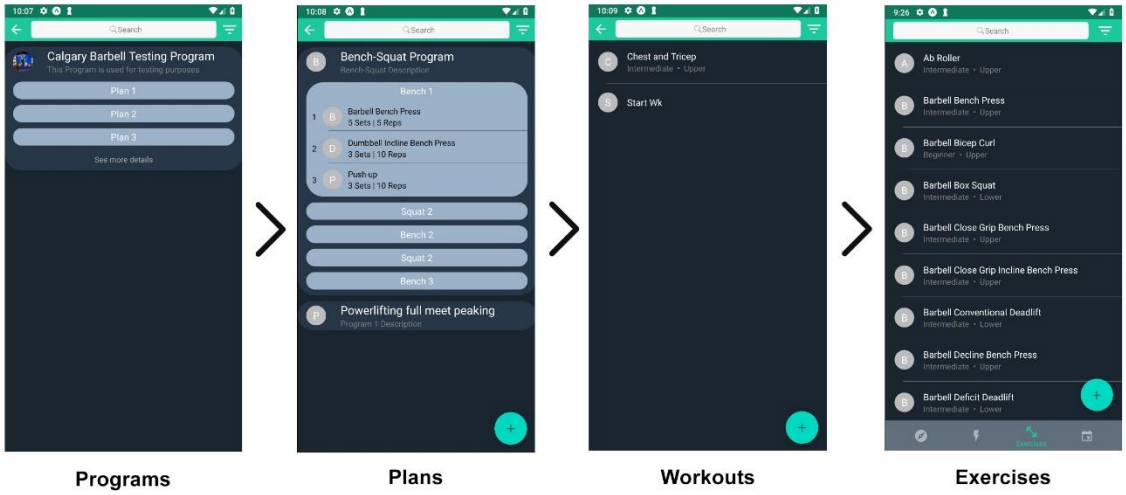


Figura 66 - Fluxo 2: Visualização de dados

Na Figura 66 está presente o fluxo de visualização de dados (programas, planos, treinos e exercícios). Os dados são relativos aos criados pelo utilizador, bem como aos já existentes no sistema.

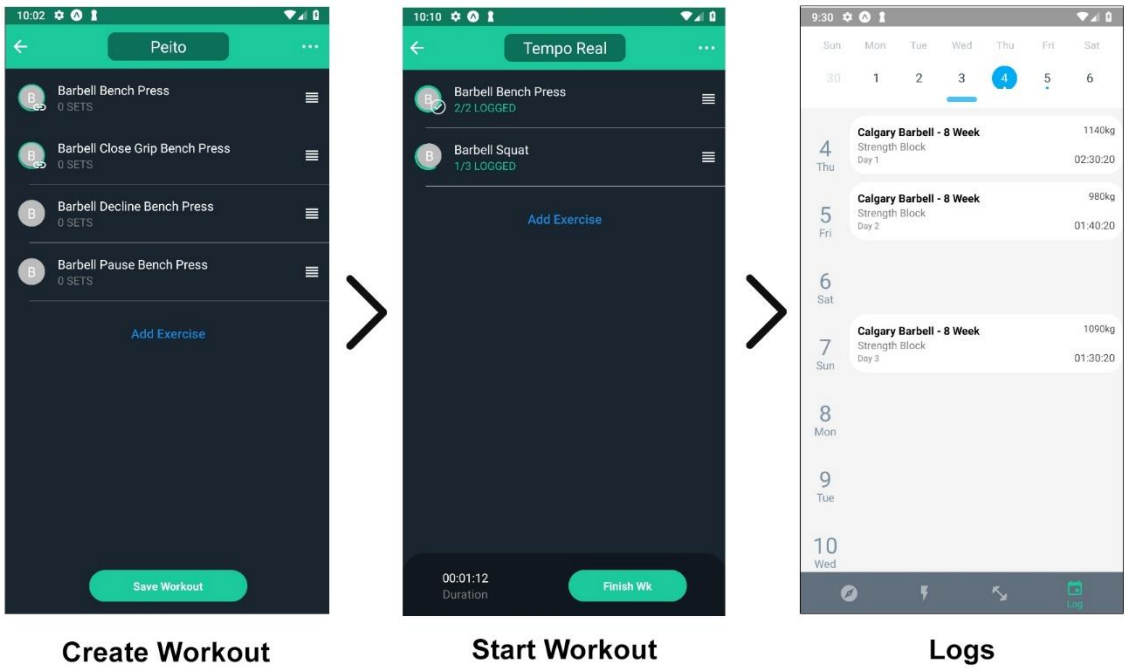


Figura 67 - Fluxo 3: Realização de um treino

Na Figura 67 está presente o fluxo de realização de um treino, com *screenshots* referentes à criação e realização em tempo real de um treino, e *logs* de treinos num calendário.

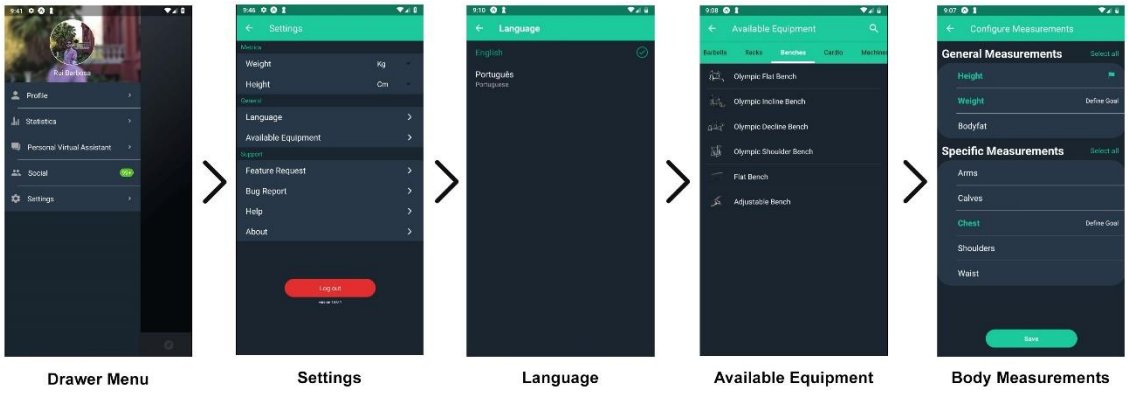


Figura 68 - Fluxo 4: Configurações

Na Figura 68 estão presentes as diversas configurações possíveis de realizar na app, como alteração do idioma, equipamento disponível e medidas corporais.

Ao longo desta secção foram descritos os fluxos principais da aplicação, sendo que não foi possível a inclusão de todos os ecrãs e funcionalidades desta. Consequentemente, na secção dos Casos de Uso este tema é detalhadamente explorado para cada funcionalidade.

5.3 Aplicação Servidora

A aplicação servidora tem como responsabilidade realizar determinadas operações computacionalmente complexas que não são aconselhadas a serem feitas nos seus clientes (p.e. aplicações móveis). Além disso, apenas esta comunica com a base dados, sendo encarregue de persistir e armazenar os dados enviados pelos clientes. Adicionalmente, se estes necessitarem de determinados dados, a API é designada por os obter (da base de dados) e enviar.

Tal como o Sistema de Recomendação, a API foi desenvolvida utilizando a linguagem *C#* e usado o *Visual Studio 2017* como ambiente de desenvolvimento.

De seguida, serão descritos os componentes mais importantes da aplicação servidora que permitiram a sua criação e o bom funcionamento.

5.3.1 Autenticação e Autorização

Para garantir a autenticação no servidor foi utilizado o *ASP.NET Identity* [153]. Esta ferramenta fornece um mecanismo para realizar a gestão de contas de utilizadores, providenciando classes que permitem a criação de utilizadores e *roles*, como ilustrado na Figura 69.

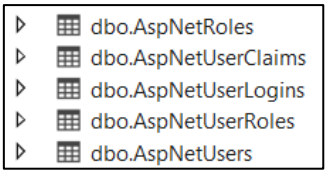


Figura 69 – Classes geradas pelo *ASP.NET Identity*

Além disso, são providenciados métodos que permitem o registo de utilizadores, login, alteração da *password* e geração de *tokens*.

Relativamente à autorização, foi utilizado o atributo *Authorize* que verifica se o utilizador se encontra autenticado (ver Figura 70). Este atributo foi utilizado em todos os *Controllers* da API de maneira a restringir todos os pedidos enviados ao servidor.

```
[Authorize]
[RoutePrefix("api/Movements")]
public class MovementsController : ApiController
{
```

Figura 70 – Autorização nos métodos da API

5.3.2 Encriptação

De forma a responder ao requisito de confidencialidade presente nos requisitos não funcionais, garantir a confidencialidade dos dados do utilizador, foi criada uma classe com métodos de encriptação e desencriptação de um conjunto de caracteres (*string*) baseada no *Stack Overflow*[5]. Os métodos são baseados na classe de encriptação *RijndaelManaged* [154]. *Rijndael* foi o algoritmo escolhido pela *National Institute of Standards and Technology* (NIST) "para proteger informações federais devido às suas responsabilidades estatutárias" [155] ficando com o nome *Advanced Encryption Standard* (AES). Importante referir que as especificações da seleção do algoritmo eram imensas, tais como: possuir uma cifra simétrica, desenvolvido para que o tamanho da chave pudesse aumentar e implementável tanto em *software* como *hardware*. O algoritmo também foi analisado tendo em conta a segurança, eficiência computacional, requisitos de memória, simplicidade e adequação *hardware* e *software* [156]. Devido a todas estas características, é coerente assumir que o algoritmo utilizado nos métodos é considerado uma boa prática.

Utilizando a classe *RijndaelManaged* juntamente com a função *Rfc2898DeriveBytes* [157] é possível gerar uma chave criptográfica aliada à *password* da conta de serviço. Também é utilizado o algoritmo *PBKDF2* que funciona como um algoritmo de reforço de senha que dificulta a verificação de qualquer senha como sendo a senha principal no decorrer de um ataque informático [158]. Assim, o algoritmo utiliza uma chave de encriptação, que é uma *string* arbitrária usada para encriptar e desencriptar.

5.3.3 Migrations

Com a evolução do projeto, o modelo de dados tem a tendência de ser modificado. Novas classes, novas relações, novos atributos, remoções, entre outras operações podem ser executadas no modelo, originando que este não fique sincronizado com a base de dados.

Desta forma, foram utilizadas m*igrations* para que a base de dados não tivesse de ser recriada sempre que uma alteração fosse realizada. Invés disso, o esquema da base de dados é atualizado. Na Figura 71 encontram-se presentes as *migrations* usadas neste projeto.

[5] https://stackoverflow.com/questions/10168240/encrypting-decrypting-a-string-in-c-sharp

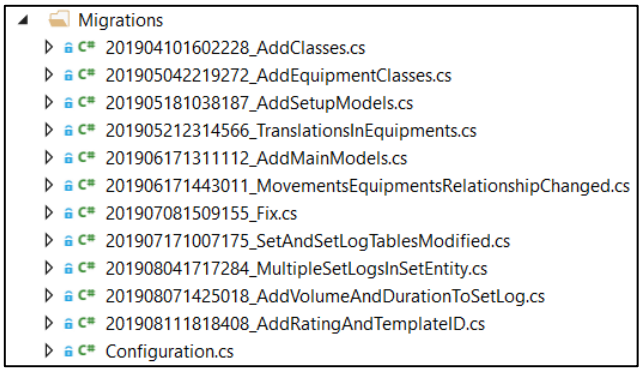


Figura 71 - Migrations criadas no projeto

5.3.4 Multilingue

Como requisito não funcional já mencionado e implementado na aplicação móvel, suporte a vários idiomas, também é necessária a sua implementação na aplicação servidora, mais especificamente, na base de dados.

A utilização da forma mais comum, ou seja, a classe de domínio possuir os seus próprios atributos como o nome, descrição, entre outros, não foi empregue, visto que não garante a suportabilidade de vários idiomas.

Dessa forma, foi analisada a solução que garantisse um modelo eficiente e que não originasse redundância nos dados. Portanto, o mecanismo implementado encontra-se ilustrado na Figura 72 identificado como (ii).

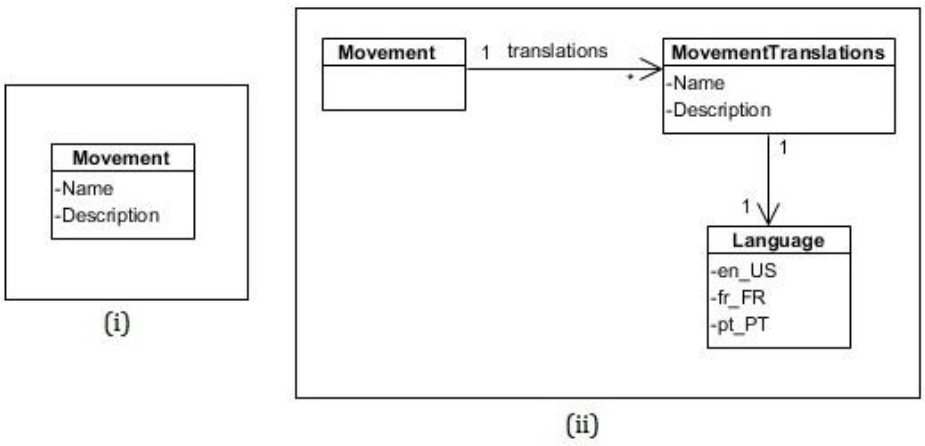


Figura 72 – Multi-idioma na base de dados

A implementação de todas as classes que possuem atributos que podem ser traduzidos para vários idiomas seguem o procedimento ilustrado no exemplo da Figura 72. Nesse exemplo, a classe *Movement* invés de declarar diretamente os atributos nome e descrição (i), relaciona-se com a classe *MovementTranslantion* numa relação de 1 para n (ii). Desta forma, uma instância de *Movement* pode conter uma lista de traduções, em que cada tradução diz respeito a um

idioma (p.e. uma instância de *MovementTranslation* ter o idioma português com nome Supino e descrição exercício intermédio).

5.3.5 Fluxo

Explicadas as características mais importantes da aplicação servidora, é importante descrever o fluxo e como funciona o tratamento da informação de um pedido. Na Figura 73 encontra-se ilustrado um diagrama de sequência que mostra como a API funciona internamente para responder a um pedido *GET*.

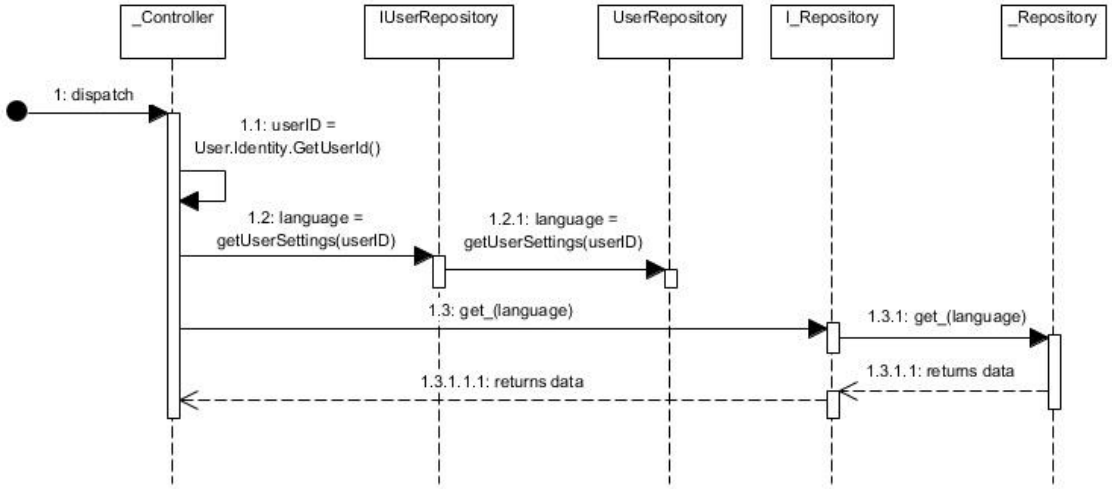


Figura 73 -Tratamento de pedidos GET na API

Em primeiro lugar, o *Controller* obtém o id do utilizador corrente através da variável User do *ASP.NET Identity* (possui informação do utilizador autenticado).

Possuindo o id do utilizador, o sistema obtém a língua deste através do método *getUserSettings(id)* presente no *UserRepository*. Desta maneira, a API pode retornar os dados pedidos no idioma predefinido do utilizador, garantindo o multi-idioma.

5.4 Personal Virtual Assistant

De forma a criar um aspeto inovador numa aplicação móvel de fitness, um *Personal Virtual Assistant* inteligente foi implementado. Para construir este elemento foram utilizados quatro componentes do sistema: aplicação móvel, onde o utilizador interage com o assistente pessoal; API do *ChatBot*, para onde são reencaminhadas as mensagens e, adicionalmente, produzida a mensagem que será enviado ao utilizador; API do *LUIS*[6], serviço da *Microsoft* baseado em *machine learning* responsável por criar linguagem natural para utilização com o *bot*; e aplicação servidora, que retorna dados da base de dados relacionados com exercícios. A comunicação entre os componentes referidos encontra-se na Figura 74.

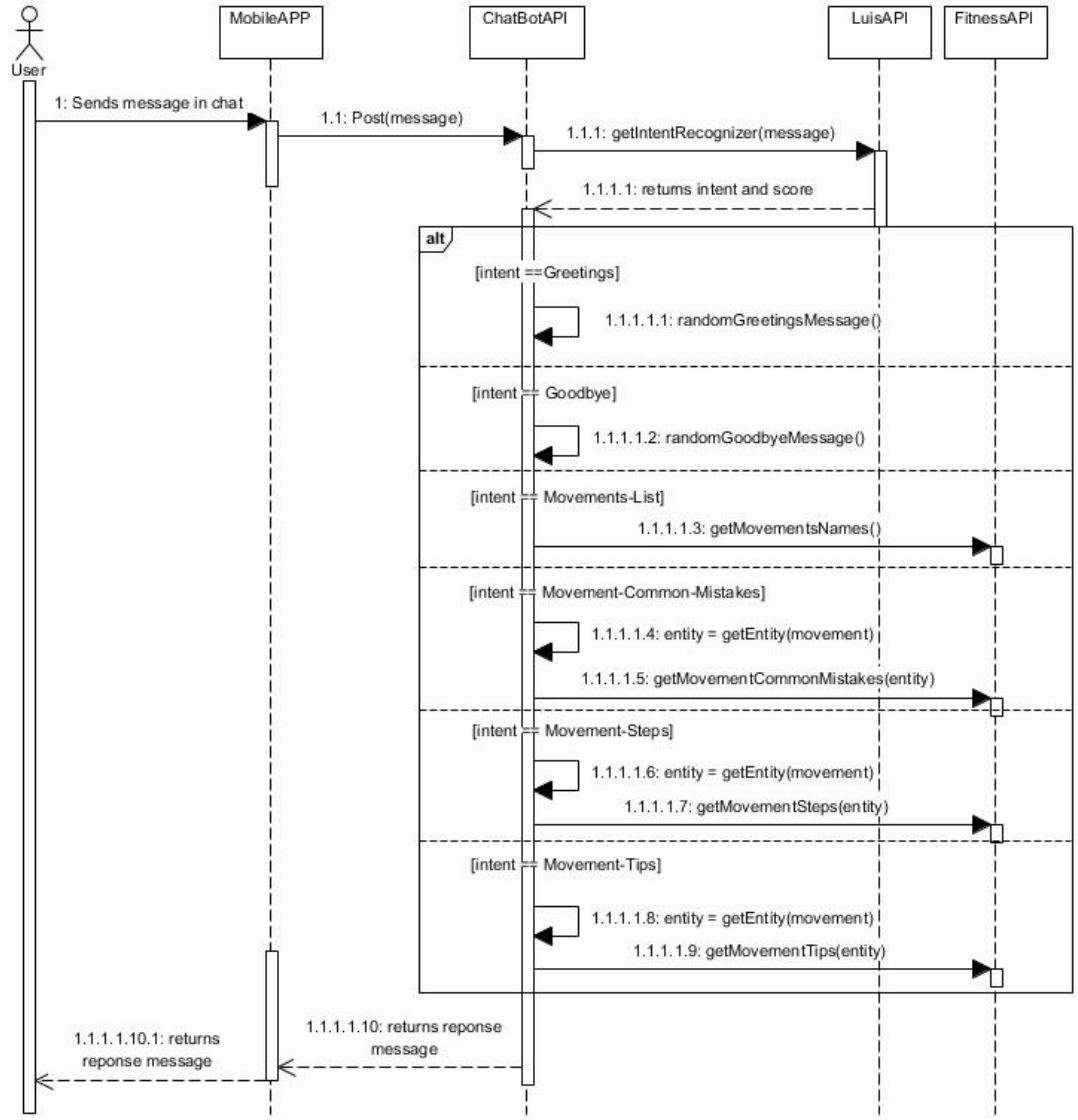


Figura 74 – Comunicação entre os componentes do PVA

[6] https://www.luis.ai/home

Como referido em cima, a *LuisAPI* foi utilizada para interpretar as mensagens do utilizador, adicionando inteligência ao assistente pessoal. Assim, uma mensagem de boas vindas pode ser escrita de formas diferentes (p.e. olá, bom dia, oi, hello, boa noite, entre outras), que o PVA responderá com sentido.

O *LUIS* fornece uma interface gráfica onde é possível criar as intenções (*intents*), as entidades (*entities*), treinar o modelo de *machine learning* criado e rever expressões do utilizador, isto é, atribuir as mensagens que a *LuisAPI* interpretou com dúvidas aos *intents* corretos.

Na Figura 75 estão presentes as fases que permitem a criação do modelo para a posteriori interpretação da mensagem do utilizador.

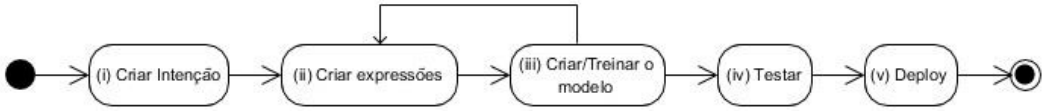


Figura 75 – Fases para a criação do modelo (*Luis*)

Na primeira fase é criada a intenção. Como se trata da primeira versão deste projeto, ficou decidido que apenas seis intenções seriam criadas: saudação; despedida; lista de exercícios; e erros comuns, dicas, e instruções para a realização de um exercício.

De seguida, são criadas as expressões de cada intenção. Por exemplo, para a intenção "dicas de um exercício", foram criadas as seguintes expressões: dicas para o exercício x, dicas do exercício x, quero as dicas do exercício x, quero ver dicas, pode-me facultar dicas para o exercício x, quero ver recomendações do exercício x, entre outras, como ilustrado na Figura 76. O x diz respeito a uma entidade, neste caso, é o nome do exercício ou o identificador deste.

Figura 76 - Expressões da intenção "dicas de um exercício"

As expressões definem-se como sendo as múltiplas e variadas formas de dizer algo que se relacione com a intenção. No exemplo anterior, todas as expressões descritas querem dizer exatamente o mesmo "dicas para o exercício x".

Tendo a intenção e as suas expressões criadas, o modelo pode ser criado e treinado selecionando a opção apropriada no LUIS.

Depois de finalizada a fase de treino, o modelo é testado verificando se através da introdução de uma expressão, p.e. "quero as dicas do exercício supino", o serviço retorna a intenção apropriada (ver Figura 77).

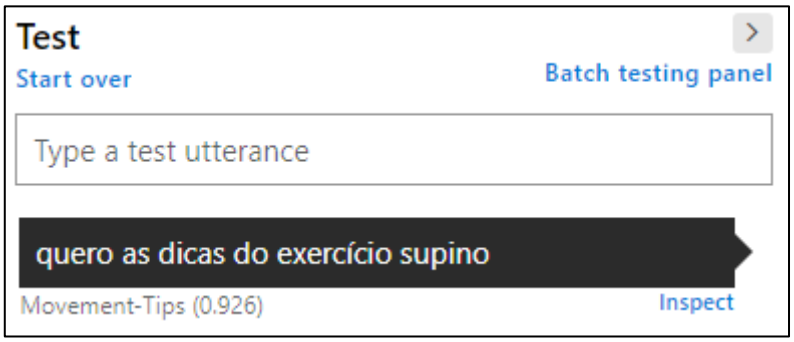


Figura 77 - Testar o modelo criado

Caso não seja obtida, os passos (ii) e (iii) presentes na Figura 75 podem ser repetidos até que seja garantida uma obtenção correta da intenção pretendida.

Numa última fase, depois de realizados os testes adequados, é feito o *deploy* do modelo criado, sendo possível à *ChatBotAPI* comunicar com a *LuisAPI* para interpretar as mensagens do utilizador.

Na verdade, quando uma mensagem é enviada para a API do *ChatBot*, a sua primeira tarefa é invocar a *LuisAPI* para reconhecer qual a intenção que obteve melhor classificação. A resposta desse pedido é uma lista com todas as intenções existentes ordenada por ordem decrescente pelo atributo *score* (classificação atribuída pelo serviço *Luis*).

Consoante cada tipo de intenção, o sistema atua de forma diferente. Portanto, foi criada uma interface *IIntentHandler* com o método *HandleIntent(…)*. Para todas as intenções foi criada uma classe que implementa a referida interface, e por sua vez, o método. Resumidamente, o comportamento de cada intenção é o seguinte:

- **Saudação e Despedida:** escolhe aleatoriamente uma *string* da lista de *strings* existente;
- **Lista de Exercícios:** invoca a *FitnessAPI* para obter os nomes de todos os exercícios;
- **Dicas, Erros Comuns e Instruções de um exercício:** extrai a entidade presente na mensagem do utilizador (id ou nome do exercício) e invoca a *FitnessAPI* para retornar as dicas, os erros comuns ou as instruções do exercício.

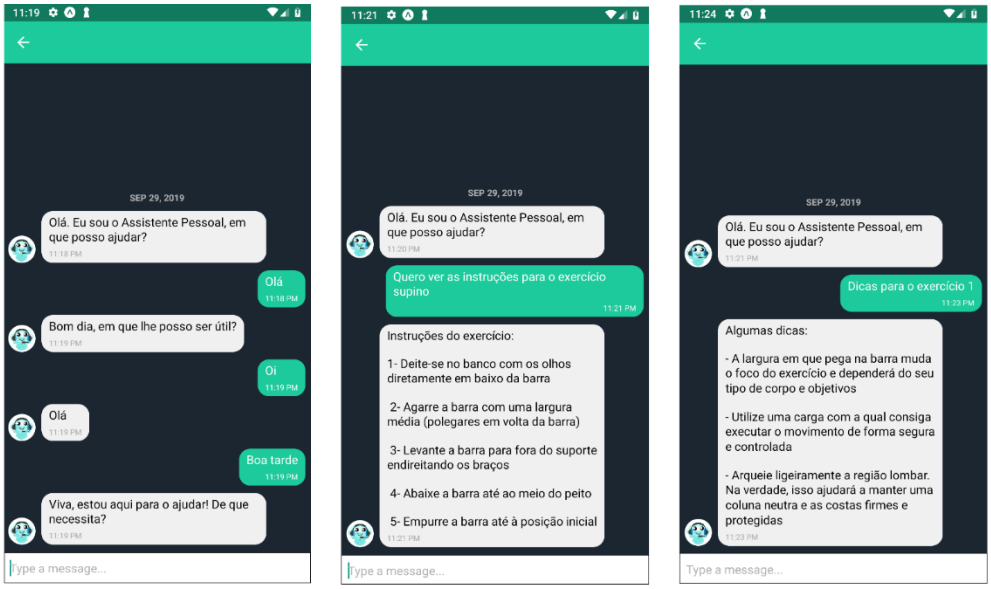
Depois de obtida a mensagem de resposta, esta é retornada.

Na aplicação móvel foi utilizada a biblioteca *gifted-chat* [134] para a implementação de um *chat*. Adicionalmente, foi utilizado o *Direct Line* [159], um protocolo de comunicação entre apps móveis e a *framework Microsoft Bot*. Na Figura 78 encontra-se ilustrado a utilização deste canal.

```
onSend = messages => {
  this.setState({ messages: [...messages, ...this.state.messages] });
  messages.forEach(message => {
    directLine
      .postActivity(giftedMessageToBotMessage(message))
      .subscribe(() => console.log("success"), () => console.log("failed"));
  });
};
```

Figura 78 - Utilização do canal *Direct Line* na aplicação móvel

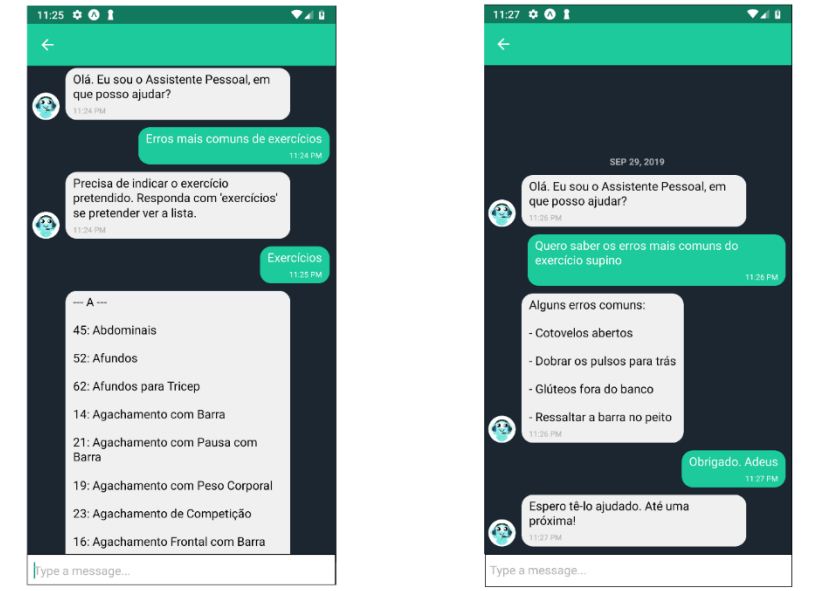
Na Figura 79 encontra-se ilustrado a interação com o assistente pessoal. É possível observar as diferentes respostas tendo em conta cada tipo de intenção. Além disso, para uma mesma intenção (neste exemplo, "*Greetings Intent*"), diferentes mensagens produzem o mesmo tipo de resposta.



Greetings Intent

Steps Intent

Tips Intent



Common Mistakes and Exercises Intents

Common Mistakes and Goodbye Intents

Figura 79 - Interação com o assistente pessoal

5.5 Casos de Uso

Nesta secção serão usados artefactos, tais como excertos de código, diagramas de sequência, e imagens da aplicação móvel, de maneira a descrever os detalhes da implementação de cada caso de uso, as opções tomadas e o modo como foram implementados.

5.5.1 UNA01: Registar Utilizador e UNA02: Login

Para garantir a autenticação no sistema foi implementado um processo que inclui registo, login e recuperação da *password*. Nesta secção o registo e o login serão abordados.

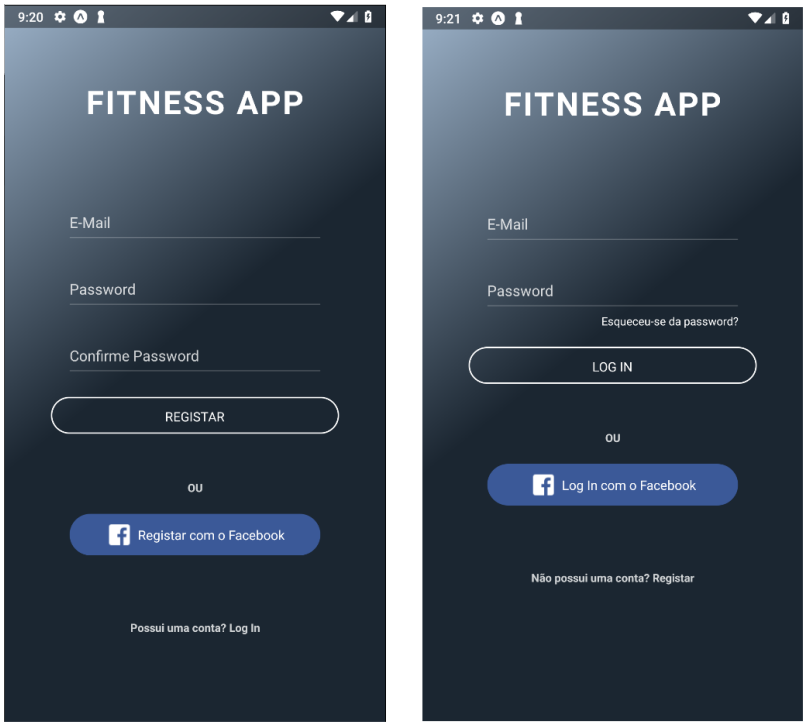


Figura 80 - Ecrã registo e login na aplicação móvel

Na Figura 80 encontram-se ilustrados os ecrãs de registo e login na aplicação móvel. É possível realizar o registo e o *login* de duas formas: introduzir o email e *password* ou realizar o *logjn* numa conta do Facebook, ficando o nome e o e-mail associados à conta criada.

Para implementar estas funcionalidades foram precisos três componentes: aplicação móvel (*FitnessMobileAPP*), aplicação servidora (*FitnessAPI*) e *FacebookAPI* [7]. A comunicação entre estes três componentes encontra-se presente na Figura 81.

[7] https://developers.facebook.com/docs/apis-and-sdks/

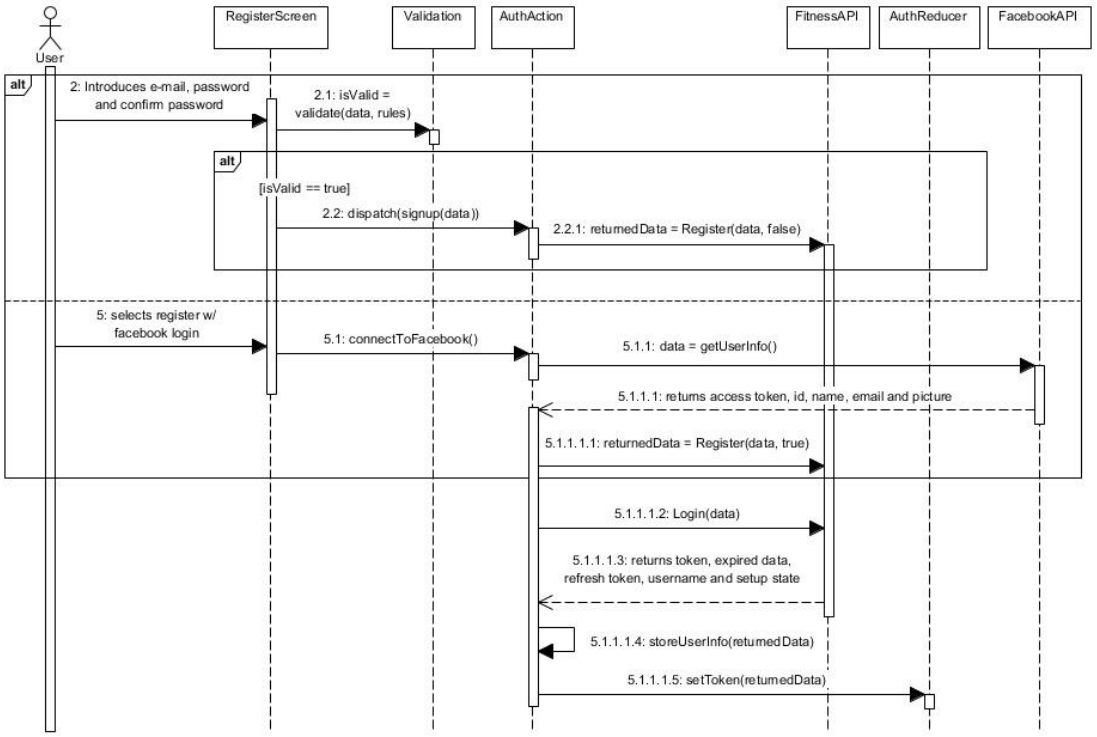


Figura 81 - Diagrama de sequência entre os três componentes principais no registo

Depois de introduzidos os dados no método de registo tradicional (e-mail e *password*), o sistema valida-os invocando uma função que recebe como parâmetros as regras às quais tem de passar e os dados inseridos. Neste caso, as regras são: e-mail válido; password com mais de 6 caracteres; e *password e* confirme *password* iguais (esta última não presente no *login*).

Sendo os dados válidos, é invocada a ação do *Redux* responsável pela autenticação, que tem como tarefa comunicar com a API para o registo do utilizador. É enviado um parâmetro a *false*, visto não se tratar do registo com o Facebook.

Por outro lado, caso o método de registo escolhido tenha sido pelo Facebook, o sistema invoca, em primeiro lugar, a API do Expo referente ao Facebook [8] de forma ao utilizador conseguir realizar o *login* e, só de seguida, é invocada a API do Facebook para validar os dados introduzidos e retornar informação do utilizador, como o email, id, nome e fotografia. Na Figura 82 encontra-se um excerto de código da comunicação da aplicação móvel com a API e o ecrã de introdução de dados.

[8] https://docs.expo.io/versions/latest/sdk/facebook/

```
export const connectToFacebook = () => {
    console.log("## ConnectToFacebook action ##");
    return async dispatch => {
        const {
            type,
            token,
        } = await Expo.Facebook.
        logInWithReadPermissionsAsync('2229830060397077', {
            permissions: ['public_profile', 'email', 'user_friends'],
            behavior: 'native'
        });

        if (type === 'success') {
            const response = dispatch(callGraph(token)).
            then((result) => {
                result = {
                    ...result,
                    facebookToken: token
                };
                return result;
            });;

            return Promise.resolve(response);

        } else {
            //Operation cancelled by User
            return Promise.resolve({ data: null, success:
                false, facebookToken: null });
        }
    };
};
```

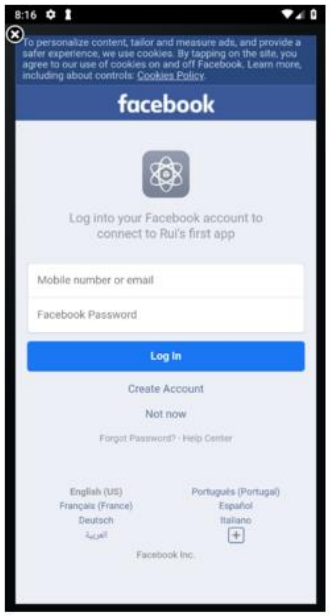


Figura 82 - Comunicação com a Facebook API

De seguida, tal como no método de registo tradicional, é invocada a FitnessAPI para registar o novo utilizador, mas desta vez com o envio do parâmetro "*IsFromFacebook*" a *true*.

Como referido na secção 5.3.1 da Aplicação Servidora o uso de *tokens* é imprescindível e essencial. Desta forma, depois do utilizador ser guardado na base de dados através da *FitnessAPI*, a aplicação móvel invoca novamente a API para a obtenção de um *access token* (login).

A API usando o conjunto de dados e-mail e *password* invoca o método *getUser(email, password)* da classe *UserRepository* para verificar se o utilizador existe no sistema. No caso deste ter usado o Facebook para realizar o registo, a *password* é uma junção do identificador com o e-mail. Neste último caso, utilização do Facebook, a autenticação e a geração do *token* só é garantida após invocar, novamente, a API da rede social Facebook com a chave recebida, comparando, no fim, se os dados retornados desse pedido são iguais aos presentes na base de dados do utilizador específico.

Depois de verificar que o utilizador existe no sistema, o *token* é gerado através do *ClaimsIdentity* do *C#* com um tempo limitado (este tema encontra-se explorado no fim da presente secção). Por fim, a API retorna o *token* gerado, a data de expiração, o *refresh token* e o estado do *setup* do utilizador. Na Figura 83 encontra-se ilustrado um diagrama de sequência descrevendo a lógica de *login* implementada na API.

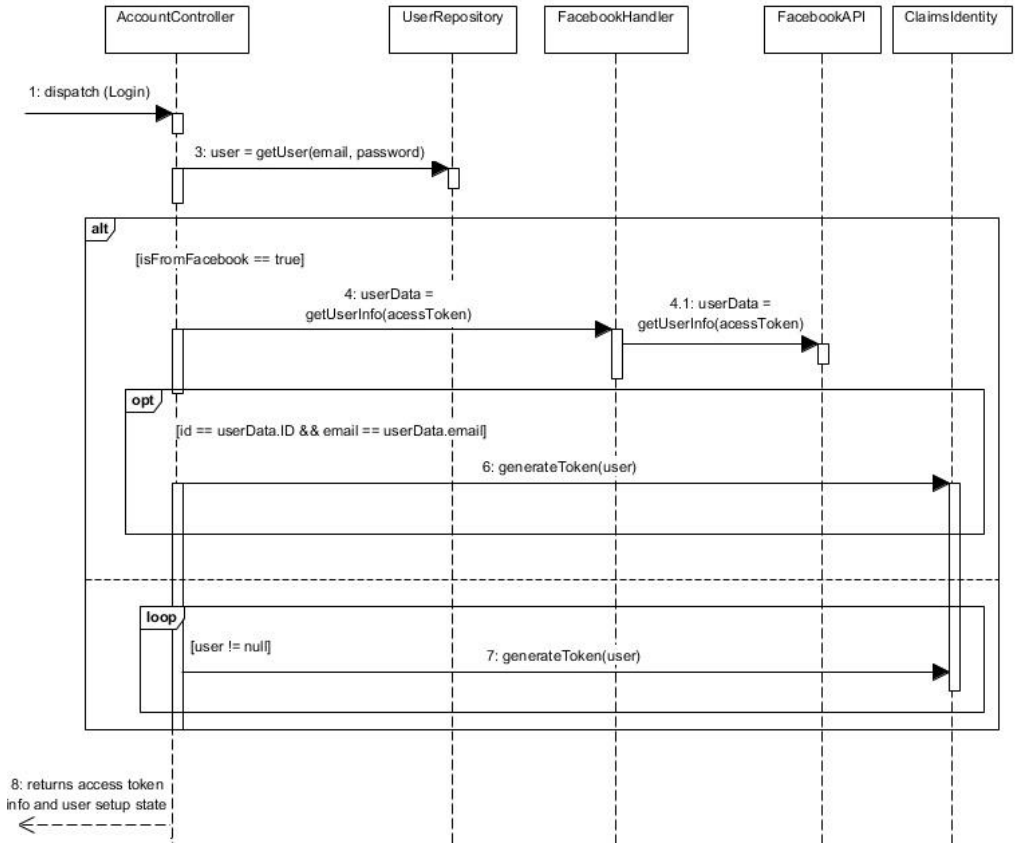


Figura 83 - Diagrama de Sequência do método login na API

Depois de obtidos os dados da sessão, a aplicação móvel guarda-os em memória. Utiliza o *AsyncStorage* para que os dados não sejam perdidos aquando do fecho da aplicação e usa o *Redux* para uma rápida acessibilidade dos dados em diferentes componentes da aplicação (ver Figura 84).

```
/** Save user info in Async Storage. */
export const authStoreToken = (token, expiresIn, refreshToken, email, isSetupDone) => {
    return async dispatch => {
        const now = new Date();
        const expiryDate = now.getTime() + expiresIn * 1000;
        dispatch(authSetToken(token, expiryDate, email));

        await setAsyncProperty(AUTH_TOKEN, token);
        await setAsyncPropertyJSON(AUTH_EXPIRYDATE, expiryDate);
        await setAsyncProperty(AUTH_REFRESHTOKEN, refreshToken);
        await setAsyncProperty(AUTH_EMAIL, email);
        await setAsyncProperty(AUTH_SETUP_DONE, isSetupDone);
    };
};
```

```
/** Save user info in Redux */
export const authSetToken = (token, expiryDate, email) => {
    return {
        type: AUTH_SET_TOKEN,
        token: token,
        expiryDate: expiryDate,
        email: email
    };
};
```

Figura 84 - Excerto de Código: Guardar dados da sessão em memória

Como referido em cima, um *access token* possui um tempo limite x (*default* é de 16 dias), que quando expirado, é necessária, novamente, a introdução do e-mail e *password* para a requisição de um novo. Para evitar que de x em x dias fosse pedido ao utilizador a realização do login, um processo de *refresh token* foi implementado. Assim, quando um *access token* é gerado, é também gerado um outro com tempo ilimitado ou alargado que tem como objetivo substituir o e-mail e *password* no pedido de *tokens* de acesso. "A combinação destes dois *tokens* garante uma máxima segurança e flexibilidade" [160] .

Sempre que a aplicação móvel é iniciada, o fluxo da Figura 85 é executado.

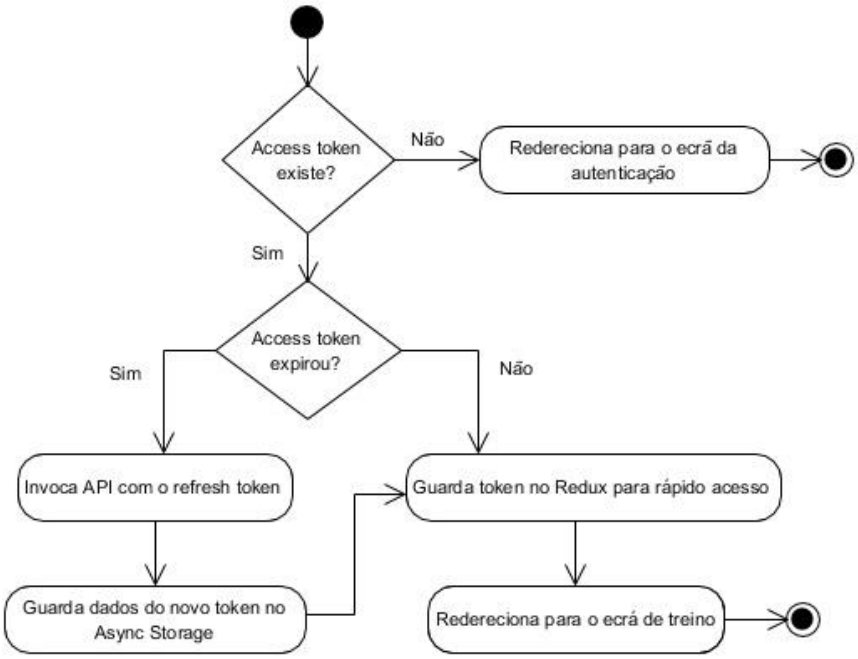


Figura 85 - Diagrama de atividades na utilização do access token

5.5.2 UNA03: Recuperar password

Na Figura 86 encontra-se ilustrado o ecrã de recuperação da *password* realizado na aplicação móvel.

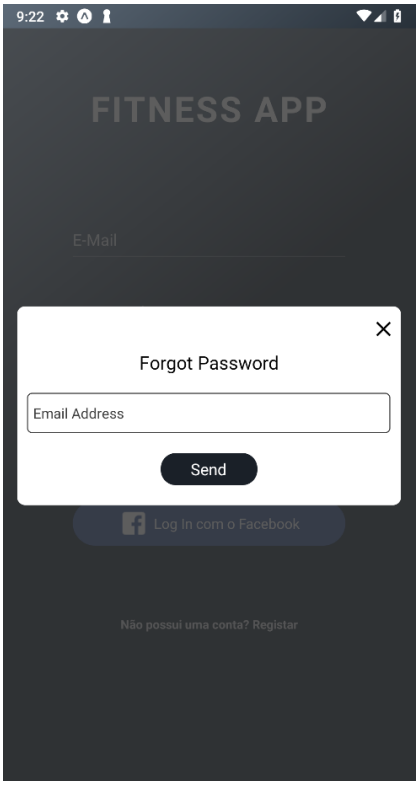


Figura 86 -Ecrã de recuperação de password na aplicação móvel

O fluxo de implementação para a recuperação da *password* é o seguinte:

1. O utilizador seleciona opção de recuperar *password*;
2. Introduz o endereço de e-mail e carrega no botão (ver Figura 86);
3. A aplicação verifica se o endereço introduzido é válido;
    a. **SE** for invoca a API para fazer tratamento do pedido;
    b. **SE** não for notifica o utilizador que o e-mail é inválido.
4. A API verifica se o endereço de e-mail é referente à conta de um utilizador existente;
    a. **SE** for gera um *token* para esse pedido com tempo de expiração (24 horas), enviando um email ao destinatário com instruções (ver Figura 87);
    b. **SE** não for, nenhum email é enviado.
5. O utilizador recebe o email e clica no link.
    a. **SE** o tempo não expirou, o sistema gera uma nova palavra passe;
    b. **SE** o tempo expirou, o utilizador terá de requisitar nova *password*.

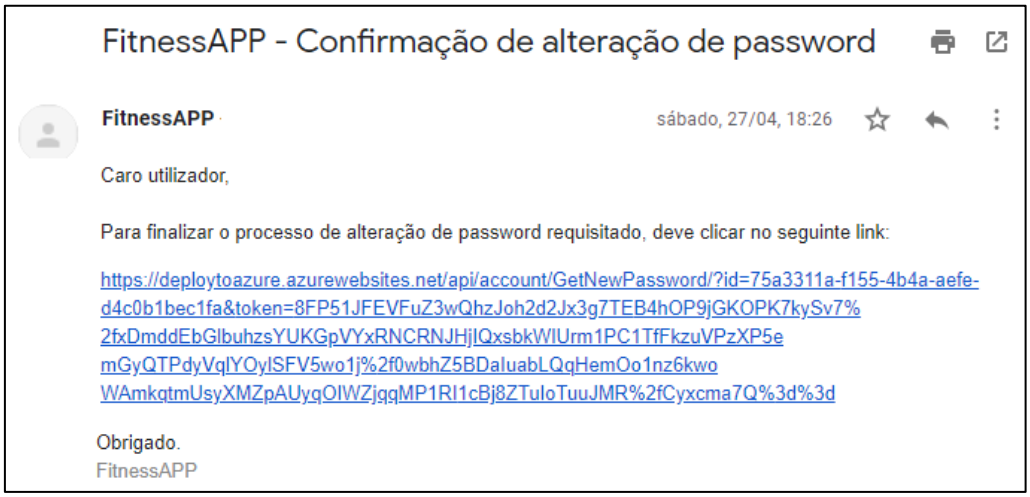


Figura 87 - Exemplo de e-mail para a recuperação da password

Para a geração de um *token* foi utilizado o método *GeneratePasswordResetTokenAsync* [9] da classe *UserManager* (gestão dos utilizadores) que gera um código para a alteração da palavra passe para o utilizador específico.

5.5.3 US01: Realizar setup

Depois do utilizador realizar o seu registo no sistema, uma séria de perguntas é efetuada de maneira a construir um perfil e recomendar itens baseados nas suas preferências. As perguntas relacionam-se com os seus dados pessoais (género, idade, peso, altura e preferências métricas), número de dias de treino por semana (1-2, 2-3, 3-4, 4-5 e 5-6), principais objetivos (ganhar músculo, perder peso, ganhar força, preparação física, ganhar peso e manter um estilo de vida saudável), nível de experiência (iniciante, intermédio e avançado) e condição física (má, normal ou boa forma). Na Figura 88 encontram-se ilustrados os vários ecrãs e o fluxo desta funcionalidade.

[9] https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.identity.usermanager-1.generatepasswordresettokenasync?view=aspnetcore-3.0

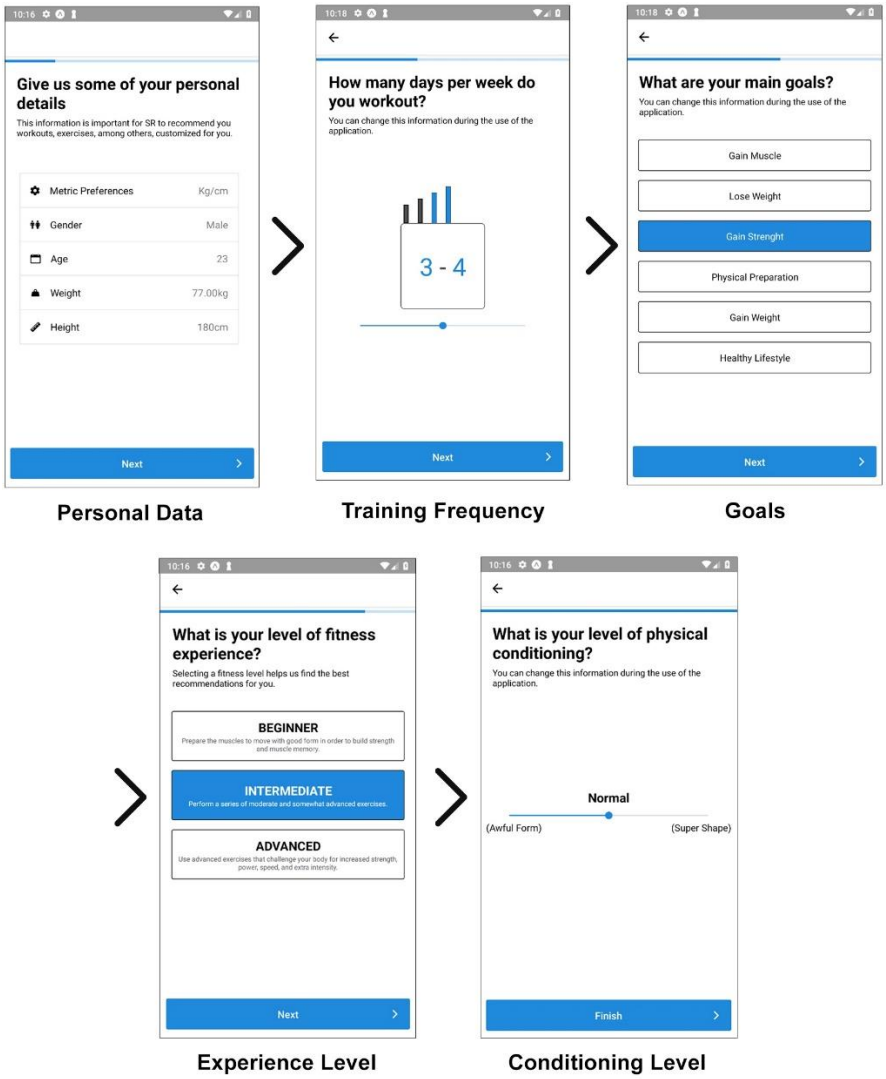


Figura 88 – Vários ecrãs no processo de setup

Esta funcionalidade foi implementada com o auxílio da aplicação móvel e da API, como ilustrado na Figura 89.

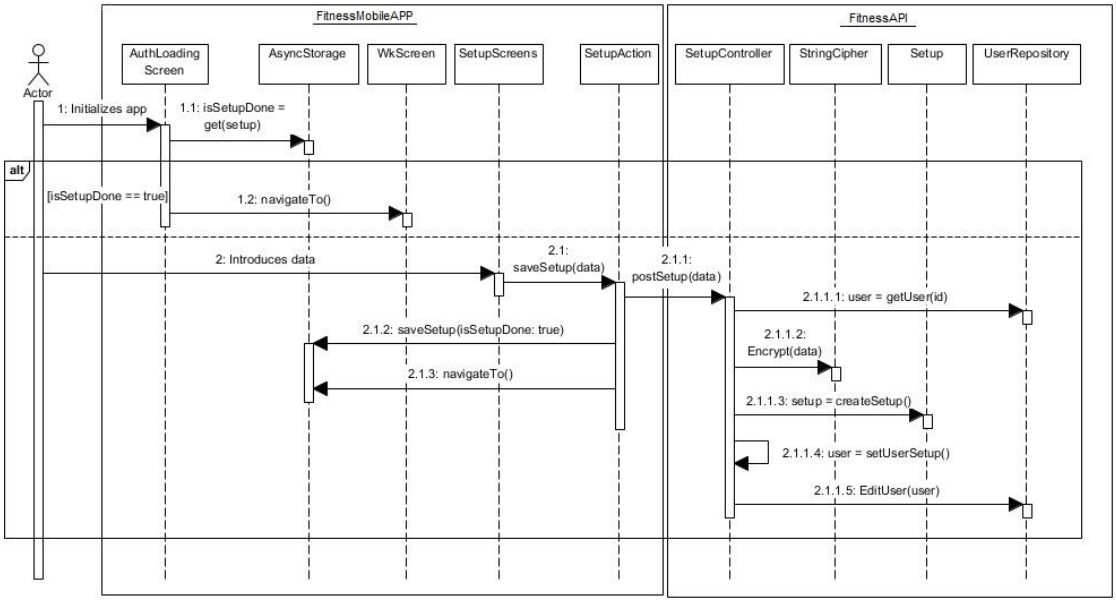


Figura 89 - Diagrama de sequência: setup

Quando a aplicação móvel é iniciada, é verificado através da variável guardada no *AsyncStorage* se o utilizador já finalizou o processo de *setup*. Caso já o tenha feito, é redirecionado para o ecrã do treino. Caso contrário, introduz todos os dados pedidos.

Depois de introduzidos os dados, a app invoca a API para guardar os dados do *setup* referente ao determinado utilizador (ver Figura 90).

```
export const saveSetup = (data) => {
    console.log("## Setup action ##");
    return async (dispatch, getState) => {
        let returnResult;
        await fetch(FITNESS_API + "api/Setups", {
            method: "POST",
            body: JSON.stringify(
                data
            ),
            headers: {
                "Authorization": getState().auth.token,
                "Content-Type": "application/json"
            }
        })
            .then(async (res) => {
                if (!res.ok) {
                    throw new Error("Server Error, try again later.");
                } else {
                    await setAsyncProperty(AUTH_SETUP_DONE, "True");
                    returnResult = { data: "", success: true };
                }
            })
            .catch(err => {
                console.log(err.message);
                returnResult = { data: err.message, success: false };
            });

        return Promise.resolve(returnResult);
    };
};
```

Figura 90 – Excerto de Código: Comunicação com a aplicação servidora

Em primeiro lugar, o utilizador correspondente ao *setup* criado é obtido através do seu identificador usando o *UserRepository*.

De forma a cumprir a confidencialidade dos dados do utilizador, como expresso nos Requisitos não funcionais, é feita a encriptação dos dados sensíveis deste utilizando o algoritmo descrito na secção Encriptação da Aplicação Servidora.

Tendo os dados encriptados, é criada uma instância da classe *Setup* e atribuída ao objeto do utilizador obtida inicialmente. De seguida, é invocado o *UserRepository* para guardar na base de dados o utilizador atualizado.

Por fim, havendo sucesso no pedido à API, a aplicação móvel guarda na *AsyncStorage* o estado do *setup* (concluído) e redireciona o utilizador para o ecrã de treino.

5.5.4 US03: Gerir exercícios e US08: Visualizar treinos

Gerir exercícios contempla a sua criação, edição, visualização individual e visualização em lista. Nesta fase, apenas a visualização em lista, considerada a mais importante, foi implementada.

Visto que, a visualização de exercícios e de treinos têm comportamentos semelhantes, os dois casos de uso foram juntos de maneira a simplificar as suas explicações. Desta forma, será unicamente usada a lógica no que toca à visualização de exercícios.

Na Figura 66 encontram-se ilustrados os ecrãs com as listas de exercícios e treinos.

Quando o ecrã de exercícios é selecionado, no seu método *componentDidMount()*, responsável por inicializar os dados necessários, a API é invocada (ver Figura 91).

```
componentDidMount() {
    if (isEmptyList(this.props.movements)) {
        if (isEmptyList(this.props.equipments) || isEmptyList(this.props.muscleGroups)) {
            this.props.onGetMovementsAndFilters().then((result) => {
                this.apiRequestBodyHandler(result);
            });
        } else {
            this.props.onGetMovements().then((result) => {
                this.apiRequestBodyHandler(result);
            });
        }
    } else {
        this.setState({
            movements: JSON.parse(JSON.stringify(this.props.movements))
        });
    }

    […]
}
```

Figura 91 - Excerto de Código: Tratamento de dados relacionados com os exercícios

De maneira a responder ao requisito não funcional Desempenho, foi adotada uma boa prática de modo a não invocar a API desnecessariamente, melhorando, assim, o tempo de resposta. Portanto, antes da aplicação comunicar com a API, verifica se os exercícios já existem no estado do *Redux*. Apenas se não existirem é que a comunicação é feita. Esta prática é utilizada sempre que é necessária a comunicação com o servidor.

Para realizar a comunicação com a aplicação servidora, existem dois métodos com a responsabilidade de obter os exercícios: *getMovementsAndFilters()* e *getMovements()*. A diferença entre os dois é que o primeiro, além de retornar os exercícios, retorna, também, uma lista de filtros (explorado no caso de uso seguinte). Estes dois métodos foram implementados, visto que os filtros podem ser obtidos em ecrãs diferentes, acedidos em qualquer componente através do *Redux*. Por isso, para eliminar informação repetida e de uma certa forma, melhorar o desempenho da aplicação, esta só invoca o método *getMovementsAndFilters()* caso a lista de filtros ainda não exista no sistema.

```
public async Task<List<MovementDTO>> GetMovements()
{
    string userID = User.Identity.GetUserId();
    UserSettingsDTO userSettings = await userRepository.GetUserSettings(userID);

    return await movementRepository.GetMovements(userID, userSettings.Language);
}

public async Task<MovementFiltersDTO> GetMovementsAndFilters()
{
    string userID = User.Identity.GetUserId();
    UserSettingsDTO userSettings = await userRepository.GetUserSettings(userID);

    var movements = await movementRepository.GetMovements
        (userID, userSettings.Language);
    var equipmentCategories = await equipmentCategoryRepository.
        GetEquipmentsByCategory(userID, userSettings.Language);
    var muscleGroups = await muscleGroupRepository.
        GetCategorizedMGsWithMuscles(userSettings.Language);

    return new MovementFiltersDTO { Movements = movements,
        EquipmentCategories = equipmentCategories, MuscleGroups = muscleGroups };
}
```

Figura 92 - Excerto de Código: Obter exercícios na API

Os dois métodos referidos, encontram-se implementados na API como ilustrado na Figura 92.

5.5.5 US04: Filtrar exercícios

De maneira a facilitar a procura pelos exercícios desejados, foi implementado um mecanismo de filtragem. Na Figura 93 encontram-se ilustrados os ecrãs para a realização do filtro de exercícios. De realçar que o código foi implementado tendo em conta a reutilização, sendo possível utilizá-lo em qualquer outro domínio (p.e. treinos, planos, programas, entre outros).

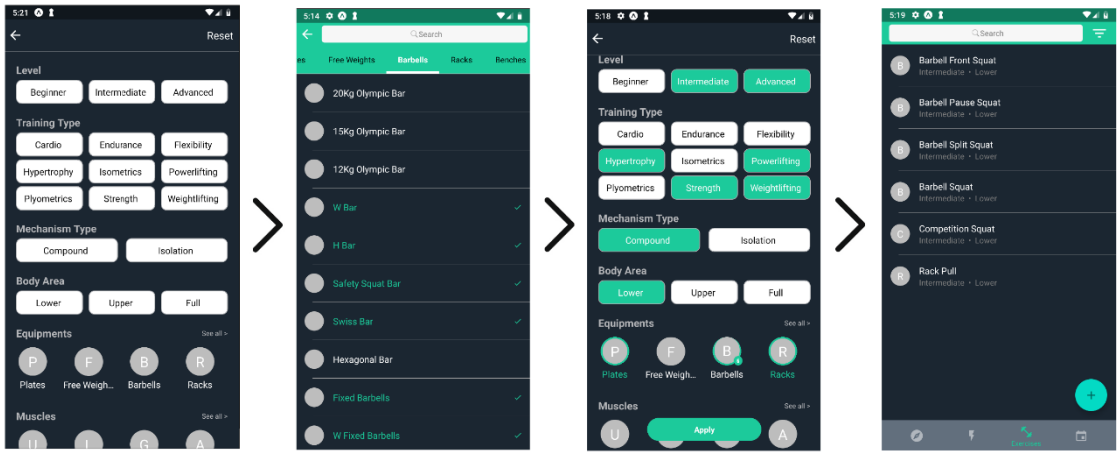


Figura 93 – Filtrar exercícios

Como ilustrado, são possíveis realizar diversos filtros a aplicar simultaneamente: nível, tipo de treino, tipo de mecanismo, área do corpo, categorias de equipamentos, equipamentos específicos, grupos de músculos e músculos específicos.

```
applyFilters = () => {
    let list = [...this.state.movements];

    //- LEVELS
    if (this.state.filters.selectedLevels.length > 0)
        list = list.filter(x => containsItemByID
            (this.state.filters.selectedLevels, x.Level));
    //- TRAINING TYPES
    if (this.state.filters.selectedTrayningTypes.length > 0)
        list = list.filter(x => findOne(
            this.state.filters.selectedTrayningTypes, x.TrainingTypes, "TrainingType"));
    //- MECHANISM
    if (this.state.filters.selectedMechanismType.length > 0) {
        const isCompound = this.state.filters.selectedMechanismType[0].ID === 1;
        list = list.filter(x => x.IsCompound === isCompound);
    }
    //- BODY AREA
    if (this.state.filters.selectedBodyAreaType.length > 0)
        list = list.filter(x => x.BodyArea === this.state.filters.selectedBodyAreaType[0].ID);
    //- EQUIPMENT CATEGORIES || EQUIPMENTS
    if (this.state.filters.selectedEquipmentsCat.length > 0
        && this.state.filters.selectedEquipments.length > 0) {
        list = list.filter(x => (findOne(this.state.filters.
            selectedEquipmentsCat, x.Equipments, "CategoryID")
            || findOne(this.state.filters.selectedEquipments, x.Equipments, "ID")));
    }

    //MUSCLES GROUPS //MUSCLES
    [...]

    this.setState({ filteredMovements: list });
}
```

Figura 94 - Excerto de Código: Aplicação dos filtros

Depois do utilizador selecionar os filtros desejados, o método da Figura 94 é executado de forma a filtrar os exercícios.

5.5.6 US09: Criar treino

A presente funcionalidade juntamente com a próxima (secção 5.5.7), são consideradas as mais complexas dentro da aplicação móvel, visto que envolvem múltiplas funções, diversos mecanismos na interface gráfica e validações. As principais funções do presente caso de uso são:

i. Selecionar, alterar, remover e ordenar exercícios;
ii. Adicionar e remover exercícios de um *superset*, bem como a sua total remoção;
iii. Diferentes interfaces para cada tipo de *set*;
iv. Adicionar e remover *sets* de um exercício;
v. Guardar treino na base de dados.

Para criar um treino é fundamental a adição de exercícios. Na Figura 95 estão presentes os ecrãs criados que permitem adicionar exercícios da lista existente do sistema, e remover ou alterar um exercício pretendido.

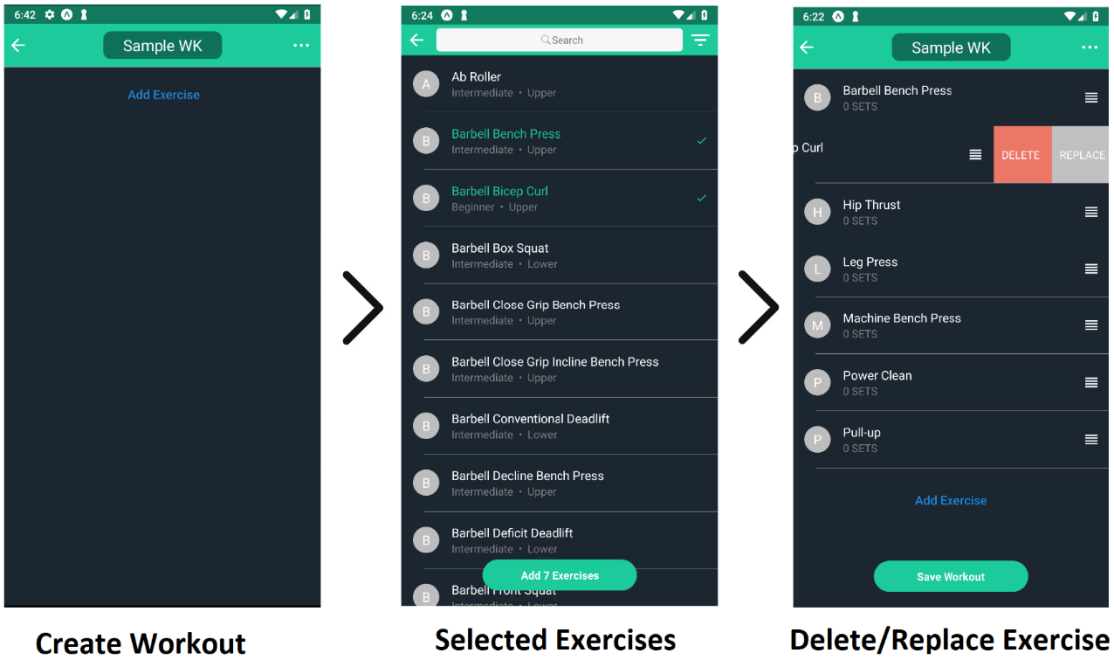


Figura 95 – Criar um treino

Um *superset* é a realização de um conjunto de exercícios de forma alternada sem descanso entre eles. Na aplicação móvel foi implementado um mecanismo de criação de *supersets*. Na verdade, o utilizador só necessita de escolher o conjunto de exercícios pretendido dentro daqueles que escolheu na fase inicial (exercícios resultantes da função i), como ilustrado na

Figura 96. Caso pretenda, também pode remover o *superset* criado, não removendo os exercícios.

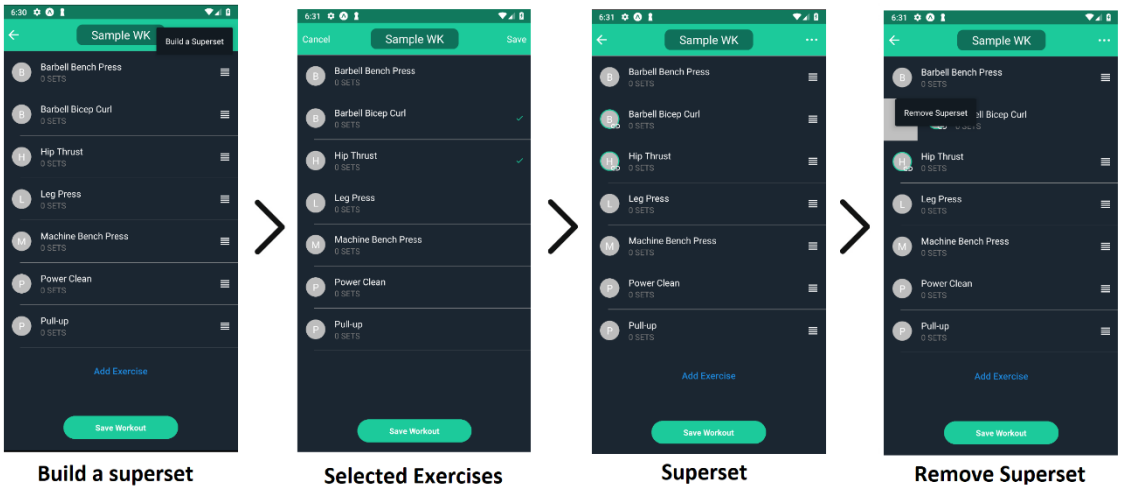


Figura 96 – Gerir um *superset*

A existência de um *superset* exige a presença de várias validações. Na Figura 97 são esquematizadas as validações referidas, onde N é o tamanho do *superset*.

$N < 2$

- Tamanho inferior a 2 ⟶ *superset* não é criado

$N = 2$

- Exercício removido dentro do *superset* ⟶ *superset* é eliminado
- Exercício ordenado para fora do *superset* ⟶ *superset* é eliminado

$N > 2$

- Exercício ordenado para fora do *superset* ⟶ o exercício é removido do *superset*

$N \geq 2$

- Exercício ordenado para dentro do *superset* ⟶ exercício adicionado ao *superset*

Figura 97 - Validações no *superset*

Como referido ao longo do documento, um exercício é composto por vários *sets*, sendo estes de quatro tipos distintos. Portanto, é fundamental que haja uma interface diferenciada consoante o tipo de *set*, pois estes possuem atributos diferentes. Foram, então, implementados quatro componentes, cada um referente a um tipo de *set*.

Na Figura 98 é verificado qual o tipo de registo do exercício, que determina qual o tipo de *set* a ser apresentado ao utilizador.

```
<View style={{ flexDirection: "column", }}>
    {this.state.exercise.Movement.DefaultRecordType === 1 ?
        <WeightAndRepsInput [...] />
        : this.state.exercise.Movement.DefaultRecordType === 2 ?
            <RepsOnlyInput [...] />
            : this.state.exercise.Movement.DefaultRecordType === 3 ?
                <CardioInput [...] />
                :
                <TimeInput [...] />
    }

    {this.renderSeparator()}
</View>
```

Figura 98 - Excerto de Código: Implementação dos diferentes tipos de sets

Os ecrãs implementados de cada tipo de *set* encontram-se na Figura 99.

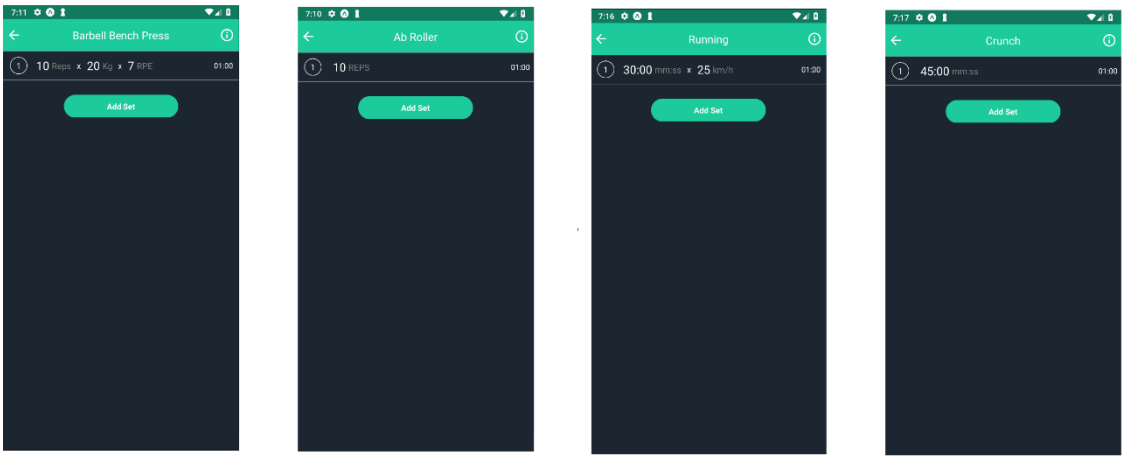


Figura 99 – Diferentes tipos de *sets*

Por fim, dentro de cada ecrã de qualquer tipo de *set*, as funcionalidades implementadas são as mesmas: adicionar, editar e remover *sets*, como ilustrado na Figura 100.

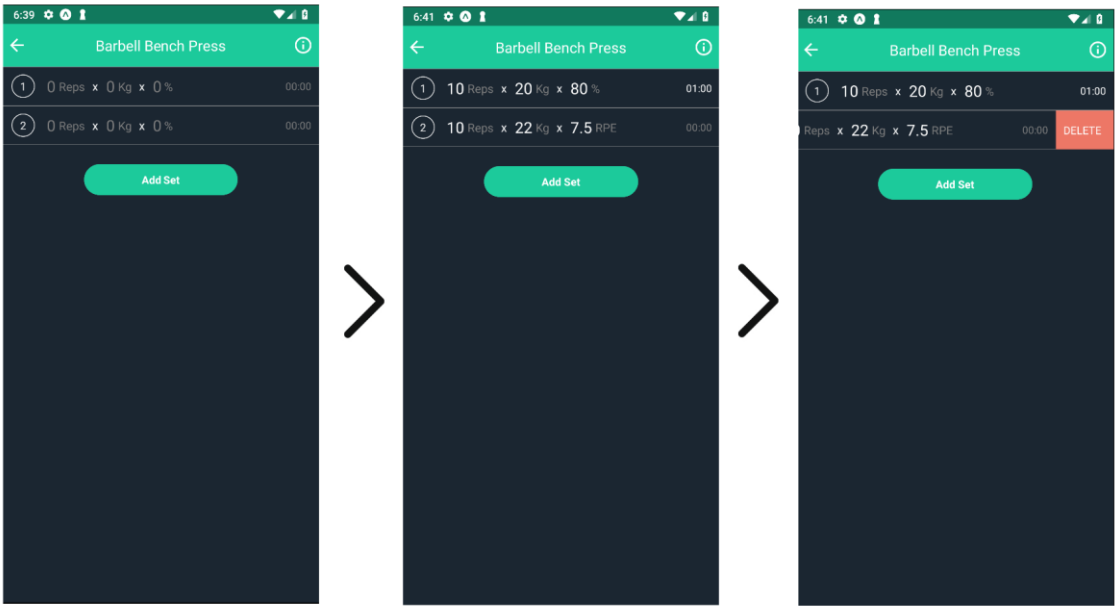


Figura 100 – Gerir *sets*

Depois de criado o treino, a aplicação móvel invoca a API para guardar os novos dados. A API, em primeiro lugar, itera todos os exercícios do treino criado de forma a obter uma lista de inteiros correspondente aos identificadores dos *supersets* criados na app (ver Figura 101).

```
private static IEnumerable<int?> GetExercisesSupersets(List<ExerciseDTO> exercises)
{
    return exercises.Where(y => y.SuperSetId != -1).Select(x => x.SuperSetId).Distinct();
}
```

Figura 101 - Excerto de Código: Obter *supersets* criados na aplicação móvel

Caso a lista de inteiros retornada seja diferente de zero, são criados na base dados n *supersets*, onde n é o tamanho da lista. De seguida, é invocado o método *SetSupersetsInExercises(…)* com a responsabilidade de substituir o atributo *supersetID* de cada exercício que esteja agrupado a um *superset*, para o atributo que foi retornado da base de dados (ver Figura 102).

```
private static SingleWorkoutDTO SetSupersetsInExercises(SingleWorkoutDTO singleWorkout,
    IEnumerable<int?> supersets, List<SuperSet> createdSupersets)
{
    int length = supersets.Count();
    for (int i = 0; i < length; i++)
    {
        int ssID = (int)supersets.ElementAt(i);
        int createdSsID = createdSupersets.ElementAt(i).ID;

        singleWorkout.Exercises.Where(x => x.SuperSetId == ssID).ToList().
            ForEach(s => s.SuperSetId = createdSsID);
    }

    return singleWorkout;
}
```

Figura 102 - Excerto de Código: Alteração do id do *superset* nos exercícios

Finalmente, é invocado o método *CreateSingleWorkout()* do *SingleWorkoutRepository* que guarda o objeto na base de dados.

5.5.7 US10: Registar dados do treino efetuado e US11: Realizar treino em tempo real

Depois de criar um treino, o utilizador pode proceder à sua realização. A realização de um treino consta em registar quais os exercícios e seus *sets* efetuados, indicando para cada um destes as repetições, o peso ou o tempo de descanso, dependentemente de cada tipo de *set*. Existem duas formas diferentes de o fazer: forma manual e em tempo real. A diferença entre as duas reside de que na segunda, além de possuir todas as funcionalidades da primeira, possui, adicionalmente, um contador e um temporizador.

Na Figura 103 encontra-se ilustrado o ecrã principal da realização de um treino, onde constam os seus exercícios e um contador, que monitoriza o tempo desde o momento em que o utilizador selecionou a opção *start* até ao momento em que ele seleciona a opção *finish*, ou seja, a duração total de tempo do treino em hh:mm:ss. Como referido em cima, o contador só está presente no treino em tempo real, tendo o treino manual a interface gráfica exatamente igual, com essa exceção.

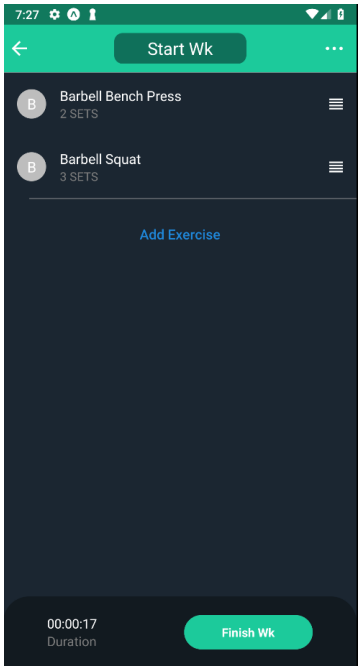


Figura 103 – Treino em tempo real

Outro fator importante é a existência das mesmas funcionalidades presentes no caso de uso criar treino. Na verdade, aquando do registo da realização de um treino o utilizador pode pretender alterá-lo, adicionando, removendo ou alterando exercícios, bem como os seus *sets*.

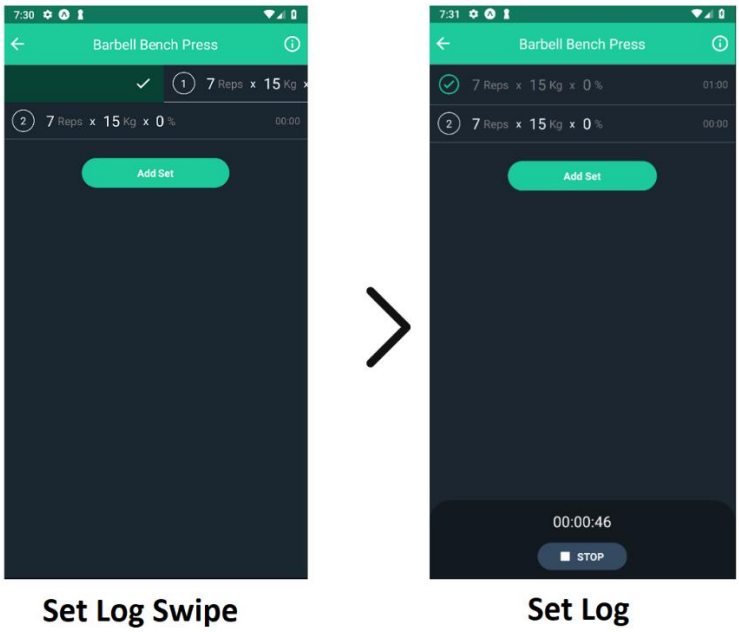


Figura 104 – Registar conclusão de um *set*

De forma ao utilizador indicar quais os *sets* realizados, foi implementado um mecanismo de *swipe* (ver Figura 104). Assim, sempre que o utilizador realizar um *set,* só necessita de dar *swipe* no *set* em questão e o sistema altera o estado deste para concluído.

Caso o determinado *set* possua um tempo de descanso associado e o modo usado seja registo em tempo real, um temporizador é iniciado. Na Figura 104 o primeiro *set* possui um tempo de descanso igual a um minuto, mal o utilizador indique a sua conclusão, o temporizador é imediatamente inicializado (na Figura 104 já possui um tempo igual a 46 segundos).

O ecrã principal de registo do treino está constantemente a ser atualizado. De facto, sempre que *sets* são realizados ou o tempo do contador é alterado, o ecrã é atualizado. Anteriormente, foi indicado que o primeiro *set* do exercício *Barbel Bench Press* foi realizado. Assim, o ecrã principal indica que apenas um de dois *sets* desse exercício foi feito (ver Figura 105).

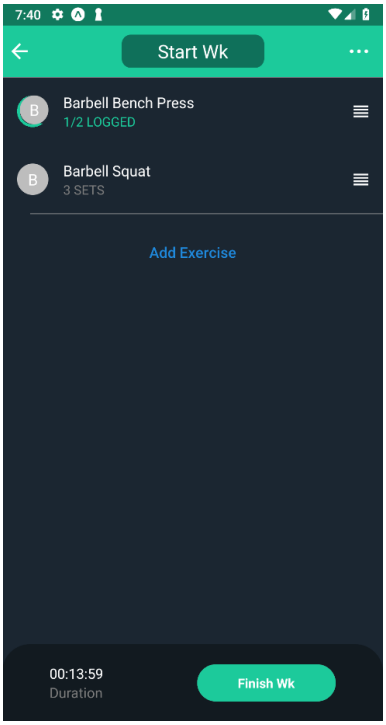


Figura 105 – Treino em tempo real atualizado

Foi implementado, também, um mecanismo para que o utilizador consiga fazer *log* de todos os treinos num simples passo.

Tendo realizado pelo menos um *set*, é possível finalizar o treino começado e invocar a API para guardar os dados. Antes da comunicação ser estabelecida, o sistema mostra ao utilizador um resumo do treino feito, indicando a duração, os exercícios completos e o volume (soma total de peso de todos os *sets* realizados), como ilustrado na Figura 106.

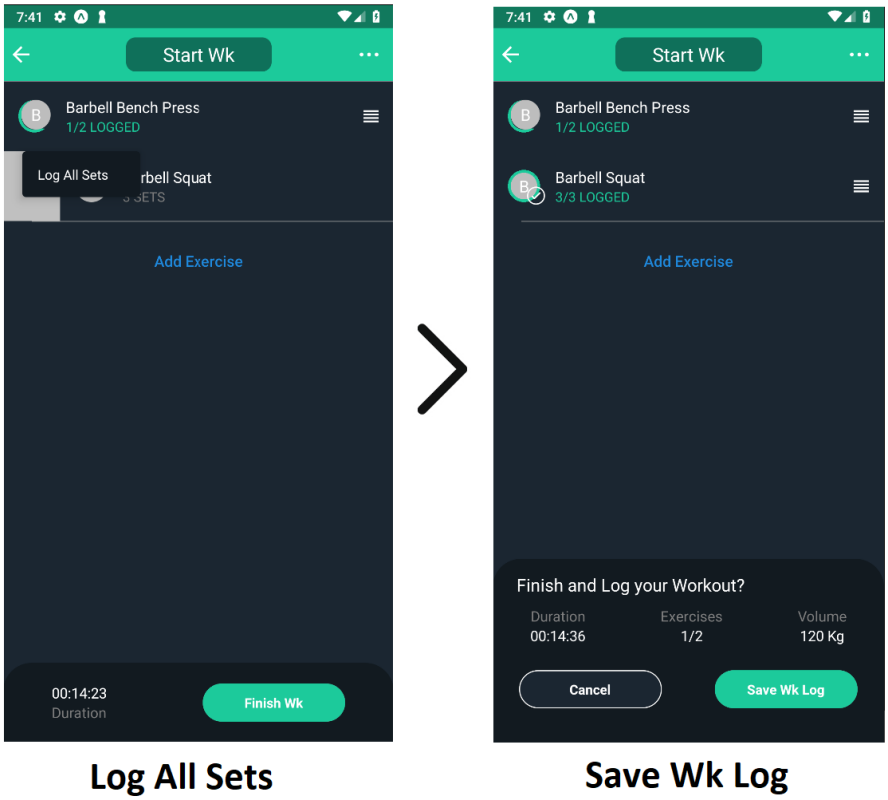


Figura 106 – Guardar treino realizado

Na API, em primeiro lugar, é obtido o treino da base de dados. De seguida, é verificado se existem novos *supersets* ou alterações aos já existentes. Caso algo se verifique, são adicionados novos *supersets* ou alterados os já existentes, respetivamente. Seguidamente, caso existam alterações nos exercícios, estes são alterados. Por fim, todos os *logs* criados na aplicação móvel são registados na base de dados.

5.6 Testes

Os testes foram planeados e projetados em quatro níveis segundo a definição do guia SWEBOK (*Software Engineering Body of Knowledge*) [161], são estes:

- **Testes unitários** a partir da implementação;
- **Testes de integração** a partir da arquitetura;
- **Testes de sistema** a partir da análise;
- **Testes de aceitação** a partir do documento de requisitos.

Nesta secção, o leitor será apresentado com os vários tipos de testes realizados, bem como a forma como foram feitos.

5.6.1 Testes unitários

Para a elaboração dos testes unitários foi utilizada a *framework Unit Testing* do *Visual Studio*. Portanto, foram definidas classes de testes para cada unidade testada (classe, controlador ou outro componente) com mais do que um teste para cada operação. Como exemplo, mostra-se na Figura 107 a implementação de um teste unitário presente neste projeto.

```
[TestMethod]
0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
public void CompareVectorsWithHighSimilarity()
{
    // Arrange
    IComparer compare = new CoRatedCosineUserComparer();
    double expectedResult = 0.94868329805051377;

    double[] userFeaturesOne = { 4, 0, 4 };
    double[] userFeaturesTwo = { 4, 2, 2 };

    // Act
    double result = compare.CompareVectors(userFeaturesOne, userFeaturesTwo);

    // Assert
    Assert.AreEqual(expectedResult, result);
}
```

Figura 107 - Teste unitário ao método *CompareVectors()*

Como ilustrado na figura anterior, o código encontra-se organizado segundo o padrão para escrita de testes *Arrange Act Assert* (AAA) [162]. O padrão divide o código dos testes unitários em três fases:

- ***Arrange:*** configurar a unidade sob teste através de todas as pré-condições e inputs;
- ***Act:*** invocar o método (comportamento) sob teste, guardando e capturando o estado resultante;
- ***Assert:*** verificar os resultados através de asserções.

Esta divisão separa claramente o que se está a testar das etapas de configuração e verificação, e organizam o código para que seja facilmente compreensível o que se pretende testar, tornando, assim, os testes mais intuitivos.

No Anexo 9 encontram-se os resultados dos testes unitários efetuados.

5.6.2 Testes de integração

Os testes de Integração foram realizados em função dos requisitos funcionais e agrupados pelos módulos principais do sistema. Devido à existência de diversos componentes, este tipo de teste é fundamental para averiguar com sucesso a integração entre todos estes.

```
[TestInitialize]
0 references | Rui Barbosa, 110 days ago | 1 author, 2 changes | 0 exceptions
public void Initialize()
{
    mock = new Mock<IMovementRepository>();

    movements = new List<Movement>
    {
        //Barbell Bench Press
        new Movement {
            Code = "1", CreatedType = CreatedType.Template, BodyAreaId = BodyAreaType.Upper,
            Type = MovementType.Simple, DefaultRecordType = RecordType.WeightAndReps,
            IsCompound = true, Level = ExperienceLevelType.Intermediate },

        //Barbell Incline Bench Press
        new Movement {
            Code = "2", CreatedType = CreatedType.Template, BodyAreaId = BodyAreaType.Upper,
            Type = MovementType.Simple,  DefaultRecordType = RecordType.WeightAndReps, IsCompound = true,
            Level = ExperienceLevelType.Intermediate },
    };

    mock.Setup(m => m.GetMovements()).ReturnsAsync(movements);
}
```

Figura 108 – Inicialização dos dados através do Mock da base dados

```
[TestMethod]
0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
public async Task GetMovementsTestAsync()
{
    var expectedResult = movements;
    MovementsController controller = new MovementsController(mock.Object);

    // Act
    List<MovementDTO> result = await controller.GetMovements();

    // Assert
    CollectionAssert.AreEquivalent(result, expectedResult);
}
```

Figura 109 -Teste ao retorno de exercícios da base de dados

Na Figura 108 encontra-se ilustrado o processo de inicialização de dados na base de dados e na Figura 109 é testado o retorno desses dados.

No Anexo 10 encontram-se os resultados dos testes de integração efetuados.

5.6.3 Testes de Sistema

Os testes de sistema avaliam o software na procura de falhas da utilização do mesmo como um todo, como se tratasse de um utilizador final. Os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados que um utilizador usaria na utilização do *software*. O objetivo é a verificação se o produto satisfaz os requisitos funcionais e os não funcionais.

Na Tabela 9 pode-se verificar o teste de sistema relacionado com a criação de um treino e a sua realização em tempo real.

Tabela 9 - Teste de sistema: criar um treino e fazer o seu registo em tempo real

Descrição:	**Criar um treino e fazer o seu registo em tempo real**
Resultado obtido:	PASSOU

No Anexo 11 encontram-se os resultados dos testes de sistema efetuados.

5.6.4 Testes de aceitação

Os testes de aceitação são testes formais para determinar se um sistema satisfaz os critérios de aceitação. Estes permitem ao cliente determinar a aceitação ou não do sistema. Assim, os testes de aceitação foram desenvolvidos baseados em cenários de utilização para testar aspetos da utilização do sistema. Cada teste de aceitação testa uma funcionalidade – caso de uso –, podendo haver mais do que um teste para um caso de uso.

Na Tabela 10 encontra-se um teste de aceitação realizado ao caso de uso - Criar treino.

Tabela 10 - Teste de aceitação: US09: Criar treino

Caso de Uso	**US09: Criar treino**
Descrição:	Criar um treino com todos os atributos existentes (p.e. exercícios, *supersets* e *sets*)
Resultado obtido:	PASSOU

No Anexo 12 encontram-se os resultados dos testes de aceitação efetuados.

6 Avaliação

Neste capítulo será apresentado todo o processo de avaliação da solução desenvolvida. Em primeiro lugar, serão apresentados os aspetos que foram avaliados – as métricas. De seguida, serão abordadas as hipóteses a testar e a metodologia de avaliação. Por fim, os resultados obtidos serão apresentados e analisados.

6.1 Métricas

A solução encontra-se dividida em três grandes módulos: o *back-end* – lógica de acesso à base de dados (*FitnessAPI*), sugestões de treinos baseadas numa recomendação condicional (*FitnessRecommenderSystem*), e geração de sugestões baseada em técnicas de Inteligência Artificial (*FitnessRecommenderSystem*); o *front-end* – aplicação móvel que disponibiliza ao utilizador uma interface simples, personalizada e intuitiva para realizar as diversas funcionalidades; e o *personal virtual assistant* – responsável pelo lógica subjacente ao esclarecimento de dúvidas. Visto que, cada módulo apresenta objetivos e responsabilidades distintas, definiram-se métricas para cada um destes.

Relativamente ao *front-end*, a aplicação móvel foi alvo de avaliação. Para isso, foi definida a seguinte métrica:

- Satisfação do utilizador relativamente à utilização geral da aplicação.

O sucesso de uma aplicação está dependente da aceitação do utilizador, sendo imprescindível avaliar a satisfação do mesmo, de forma a melhorar a interação com o sistema.

Relativamente ao *back-end*, mais especificamente, ao sistema de recomendação condicional, foram definidas as seguintes métricas:

- Tempo de execução;
- Satisfação do especialista tendo em conta a precisão do conteúdo recomendado.

Aqui foram definidas duas métricas. A primeira, tempo de execução, encontra-se associada a um indicador de *performance* com o intuito de avaliar o tempo de execução dos algoritmos para a geração de recomendações. Devido a aspetos evidentes, as recomendações devem ser apresentadas de forma rápida. A segunda métrica apresentada é utilizada para verificar a satisfação de um especialista perante o conteúdo recomendado pelo sistema.

Por outro lado, mas também relativamente ao *back-end*, foram definidas as seguintes métricas para o sistema de recomendação inteligente:

- Erro de previsão de cada uma das técnicas de recomendação;
- Impacto da variação do valor de k no erro de previsão.

As duas métricas descritas têm como objetivo principal medir a precisão das diferentes técnicas de recomendação. A primeira avalia o erro de previsão de cada uma delas, enquanto que a segunda avalia qual o impacto da variação do valor de k em função do erro de previsão. O valor de k para as técnicas colaborativas será relacionado com o número de vizinhos/utilizadores-itens semelhantes e para a técnica de M*atrix Factorization* relacionar-se-á com o número de *latent features*.

Por fim, foi definida uma métrica para o PVA:

- Satisfação do utilizador tendo em conta o esclarecimento de dúvidas.

É fundamental entender se o módulo de *personal virtual assistant* permite esclarecer as dúvidas do utilizador.

6.2 Hipóteses

Tendo em conta as métricas identificadas anteriormente, as hipóteses a testar foram formuladas. Visto que, foram enumerados três tipos de métricas relacionadas com a satisfação – utilização geral da aplicação, precisão do conteúdo recomendado e esclarecimento de dúvidas, foi decidido generalizar este tipo de variável numa única só hipótese que englobasse esses tipos distintos. Como tal, a hipótese a testar seria:

H0: Satisfação do utilizador

H1: Inquéritos de satisfação possuem um grau de satisfação igual ou superior a 4 (0-5)

H2: Inquéritos de satisfação não possuem um grau de satisfação igual ou superior a 4 (0-5)

Foi escolhido o valor igual a 4, pois entende-se que é o valor aceitável para reconhecer que se trata de um sistema que responde eficazmente a este tipo de grandeza.

Outra hipótese a testar, relaciona-se com o tempo de execução na geração de recomendações, ou seja, é avaliado o tempo que o sistema demora a realizar recomendações tendo em conta diversos aspetos do utilizador. Como tal, a hipótese a testar seria:

H0: Tempo de execução

H1: O sistema demora menos de 2 segundos a gerar recomendações

H2: O sistema demora mais de 2 segundos a gerar recomendações

Foi escolhido o valor de 2 segundos, pois entende-se, mais uma vez, que é o valor aceitável para reconhecer que se trata de um sistema que responde eficazmente a este tipo de grandeza.

As restantes hipóteses são relativas ao erro de previsão do sistema de recomendação inteligente. Na primeira é avaliado qual o erro de previsão associado a cada técnica de recomendação utilizando diferentes métricas de análise. A segunda, usando as mesmas métricas, é avaliado o impacto da variação da variável *k* (descrita na secção acima) no erro de previsão.

H0: Erro de previsão

H1: Cada técnica de recomendação possui um erro de previsão inferior a 1,5

H2: Cada técnica de recomendação possui um erro de previsão superior a 1,5

De novo, foi escolhido um erro com valor igual a 1,5, pois entende-se que é o valor aceitável para reconhecer que se trata de um sistema que responde eficazmente a este tipo de grandeza.

H0: Impacto da variação do valor de k

H1: Um aumento do valor de *k* resulta num erro de previsão inferior

H2: Um aumento do valor de *k* resulta num erro de previsão superior

Na secção 6.3, será explicado como estas hipóteses deverão ser avaliadas no sentido de obter resultados possíveis de análise.

6.3 Metodologia

Para testar e avaliar as grandezas e as hipóteses anteriormente mencionadas, serão utilizadas diferentes metodologias, dado que as grandezas a testar não podem ser testadas da mesma maneira devido às suas diferenças. Na Tabela 11 encontram-se ilustradas as metodologias a utilizar para cada uma das grandezas.

Tabela 11 – Metodologias a aplicar

Grandeza	**Metodologia**
Satisfação do utilizador/especialista	Grupo de teste e Inquérito de satisfação
Tempo de execução	Testes unitários
Erro de previsão	Divisão do *dataset* e aplicação de métricas

Para avaliar a satisfação dos utilizadores relativamente à utilização geral da aplicação e esclarecimento de dúvidas, testando a primeira hipótese anteriormente referida, foi disponibilizado um inquérito de satisfação a um grupo de teste. Adicionalmente, devido à importância de validar o conteúdo sugerido e a lógica subjacente ao sistema de recomendação condicional, foi realizado um inquérito com questões mais técnicas para ser preenchido por um especialista na área de *fitness*.

De forma a facilitar a realização dos inquéritos, uma demonstração e uma breve explicação de todas as funcionalidades da aplicação foram feitas, permitindo ao grupo de teste obter conhecimento acerca do comportamento geral da app. A classificação de cada questão segue a escala presente na Tabela 12.

Tabela 12 – Escala de avaliação das questões do inquérito

Escala	**Descrição**
1	Discordo completamente
2	Discordo
3	Sem opinião
4	Concordo
5	Concordo completamente

O inquérito disponibilizado aos indivíduos que utilizaram a aplicação, Anexo 2, apresenta afirmações relativamente ao funcionamento geral dos componentes principais do sistema (aplicação móvel e assistente pessoal).

O inquérito para o especialista, Anexo 3, apresenta afirmações mais técnicas, direcionadas para a lógica subjacente à recomendação condicional e ao seu conteúdo recomendado.

Quanto à segunda grandeza, tempo de execução, foram utilizados testes unitários para verificar o tempo de execução do algoritmo responsável pelo retorno de sugestões (treinos).

O sistema regista o tempo despendido na execução do algoritmo para possibilitar o cálculo da média de tempos nas condições definidas para os testes.

Para a última grandeza, erro de previsão, o *dataset* usado para o sistema de recomendação inteligente foi dividido em dois grupos: um grupo composto por 80% dos dados usado para o treino e outro, com os restantes dados (20%), utilizado para a fase de teste.

Tendo os dois grupos constituídos, foram aplicadas as métricas de avaliação. Existem diferentes tipos de métricas que podem ser usadas de forma a avaliar a qualidade de um sistema de recomendação. "Usar apenas uma métrica pode originar uma visão limitada de como os sistemas funcionam" [163], portanto, foram utilizadas duas – *Mean Absolute Error* (MAE) e *Root Mean Square Error* (RMSE). As métricas referidas são as mais populares e as mais utilizadas na avaliação e análise de um SR [164][165]. Ambas calculam o erro de previsão entre o valor real e o previsto, sendo que a RMSE coloca mais ênfase em erros absolutos maiores [165]. As equações de ambas as métricas são as seguintes:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \tag{21}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \tag{22}$$

, onde Y_i é o valor previsto, $\hat{Y}_i$ o valor real e n é o número de classificações.

6.4 Análise de Resultados

Nesta secção será feita a análise dos resultados obtidos de cada uma das grandezas já enumeradas – usabilidade, tempo de execução e erro de previsão, verificando se a solução desenvolvida cumpre os objetivos propostos recorrendo às hipóteses descritas na secção 6.2.

6.4.1 Satisfação dos utilizadores/especialista

De modo a testar as funcionalidades da aplicação, obter *feedback* e recolher informação sobre o sistema desenvolvido, vários indivíduos usaram a app móvel e no fim preencheram um inquérito sobre a mesma. Foi escolhida uma amostra por conveniência de 50 indivíduos dos quais 20 forneceram *feedback*.

Os inquéritos de satisfação englobam dois dos principais componentes do presente sistema (aplicação móvel e assistente pessoal). Dessa forma, esta secção será divida nesses dois elementos juntamente com a recomendação condicional.

Relativamente à avaliação do especialista, foi-lhe feita uma contextualização do projeto, explicada a lógica do algoritmo e proposto o preenchimento de um inquérito.

6.4.1.1 Aplicação móvel

A parte do inquérito relativa à aplicação móvel é constituída por 12 afirmações. Na Tabela 13, para cada uma delas, é indicada a frequência relativa em função do valor da escala já descrito na secção 6.3.

Tabela 13 – Frequência relativa de cada resposta do inquérito (aplicação móvel)

ID	Afirmações	1	2	3	4	5
1	A aplicação possui uma boa apresentação/design.	0%	0%	0%	25%	75%
2	A utilização da aplicação é intuitiva e simples.	0%	0%	0%	40%	60%
3	A aplicação é rápida.	0%	0%	10%	55%	35%
4	A aplicação está adaptada ao meu dispositivo.	0%	0%	15%	65%	20%
5	O processo de registo e login é realizado de forma rápida.	0%	0%	0%	15%	85%
6	O processo de criação de treinos e a sua realização é feita de forma simples e eficaz.	0%	0%	5%	55%	40%
7	É possível criar planos e programas de treino com todos os pormenores necessários à sua realização.	0%	0%	10%	75%	15%
8	Os filtros apresentados para os exercícios, treinos, entre outros, são eficientes na procura do desejado.	0%	0%	0%	40%	60%
9	A aplicação é de fácil configuração.	0%	0%	10%	50%	40%
10	A aplicação poderia ser útil enquanto pratico exercício físico (ginásio).	0%	0%	0%	45%	55%
11	Recomendaria a aplicação a qualquer praticante de ginásio.	0%	0%	5%	65%	30%
12	Classificaria a aplicação como muito boa.	0%	0%	0%	45%	55%

As quatro primeiras afirmações são relativas ao funcionamento geral da aplicação. Através da tabela é possível constatar que o resultado foi bastante positivo para estas afirmações. Para as duas primeiras, design e intuição da aplicação, existem mais respostas na escala 5 (concordo completamente) do que na 4 (concordo). Por outro lado, nas afirmações 3 e 4, rapidez e adaptabilidade da aplicação ao dispositivo, respetivamente, a escala 4 é a que possui mais respostas, com uma pequena percentagem de respostas na escala 3 (sem opinião). Daqui pode-se deduzir que a aplicação apresenta uma interface simples, intuitiva e *user friendly*. Por outro lado, podem ser realizadas pequenas melhorias na rapidez da aplicação (comunicação com o servidor) e melhorar alguns componentes da interface gráfica.

As afirmações 5 a 9 são referentes às principais funcionalidades da aplicação. Dentro dessas, as afirmações 5 e 8, relacionadas com a facilidade da realização do processo de registo/login e aplicação de filtros nos exercícios, treinos, entre outros, respetivamente, obtiveram 100% das respostas divididas entre as escalas 4 e 5, tendo mais de 50% na escala 5. Aliás, a afirmação 5 conta com 85% das respostas na escala 5, sendo a afirmação com maior percentagem nessa escala, demonstrando a facilidade e boa execução do processo registo/login.

Relativamente às afirmações 6, 7 e 9, as suas respostas encontram-se dividias nas escalas 3, 4 e 5, sendo que apresentam 90% ou mais entre as escalas 4 e 5, o que evidencia uma boa experiência na criação e realização de treinos (afirmação 6), criação de planos e programas (afirmação 7) e realização de configurações (afirmação 9).

Por último, as afirmações 10, 11 e 12 relacionam-se com a utilidade da aplicação. 55% dos utilizadores concordou completamente que a aplicação poderia ser útil no exercício físico (afirmação 10) e classificaria a app como muito boa (afirmação 12). Por outro lado, 65% e 30% dos utilizadores concordou (escala 4) e concordou completamente (escala 5), respetivamente, que recomendaria a aplicação a qualquer praticante de fitness (afirmação 11).

Comparativamente à hipótese formulada, média das respostas do inquérito superior a 4, pode-se concluir que a média das respostas da parte do inquérito relativa à aplicação móvel foi igual a 4,4 demonstrando uma ótima satisfação do utilizador relativamente à app.

6.4.1.2 Assistente pessoal

A parte do inquérito relativa ao assistente pessoal é constituída por 4 afirmações. Na Tabela 14, para cada uma delas, é indicada a frequência relativa em função do valor da escala.

Tabela 14 – Frequência relativa de cada resposta do inquérito (assistente pessoal)

ID	Afirmações	1	2	3	4	5
1	A conversa com o assistente pessoal é fluída.	0%	0%	0%	20%	80%
2	O assistente pessoal consegue interpretar e responder com sentido ao que foi dito.	0%	0%	5%	55%	35%
3	O assistente pessoal esclarece com qualidade as dúvidas sobre os exercícios.	0%	0%	0%	80%	20%
4	O assistente pessoal é uma mais valia para a aplicação.	0%	0%	0%	5%	95%

Após a análise da tabela verifica-se que as afirmações 1, 3 e 4 apenas possuem respostas na escala do 4 (concordo) e 5 (concordo completamente), demonstrando que a experiência realizada neste componente mostrou ser fluída e com um bom esclarecimento de dúvidas.

Analisando especificamente a afirmação 4, 95% dos utilizadores classificaram como “concordo completamente” a afirmação “O assistente pessoal é uma mais valia para a aplicação”, o que evidencia uma excelente aceitação deste componente por parte dos utilizadores.

A afirmação 2, embora com resultados bastante positivos, foi a única que obteve respostas na escala 3 (sem opinião) e com mais respostas na escala 4 do que na 5. Este resultado demonstra que ainda existem certas melhorias a serem empregues na interpretação de mensagens por parte do assistente pessoal.

Comparativamente à hipótese formulada, média das respostas do inquérito superior a 4, pode-se concluir que a média das respostas da parte do inquérito relativa ao assistente pessoal foi igual a 4,6, demonstrando uma ótima satisfação do utilizador relativamente a este componente.

6.4.1.3 Recomendação condicional

A avaliação do sistema de recomendação condicional foi feita através de um especialista.

De maneira a facilitar o processo de contextualização, lógica subjacente à sugestão de treinos, análise e avaliação, um documento formal foi enviado ao perito. Esse documento encontra-se no Anexo 4 e retrata todas as etapas detalhadas realizadas pelo algoritmo conforme os dados do perfil do especialista.

Com o referido documento, o especialista avaliou, não só o conteúdo recomendado, mas também todo o processo.

Tabela 15 – Frequência relativa de cada resposta do inquérito (recomendação condicional)

ID	Afirmações	1	2	3	4	5
1	As recomendações sugeridas são boas.	0%	0%	0%	100%	0%
2	Os treinos recomendados fazem sentido tendo em conta os meus gostos/perfil.	0%	0%	0%	100%	0%
3	A avaliação do volume e intensidade é eficiente.	0%	100%	0%	0%	0%
4	A forma de previsão do volume e intensidade é eficiente.	0%	0%	0%	100%	0%
5	As variáveis usadas para filtrar são suficientes.	0%	0%	0%	0%	100%
6	A aplicação das variáveis é eficiente (filtros às não existentes).	0%	0%	100%	0%	0%

Através da análise da Tabela 15 é possível observar que para as afirmações relacionadas com a qualidade do conteúdo recomendado (afirmações 1 e 2) o perito concordou que as sugestões eram boas e faziam sentido tendo em conta o seu perfil.

Relativamente a aspetos mais técnicos do algoritmo, o perito discordou que a forma de avaliação do volume e intensidade eram eficientes (afirmação 3) e não teve opinião acerca da eficiência da aplicação de filtros (afirmação 6). Contudo, concordou com a forma de previsão do volume e intensidade – aplicação da técnica de regressão linear *Line Of Best Fit* e concordou completamente com as variáveis usadas na filtragem.

Além da realização do inquérito por parte do especialista, este fez alguns comentários/sugestões. Estes comentários foram considerados fundamentais, não só para entender o que um perito na área de fitness acha sobre todo o processo de recomendação feito, mas também, aproveitar essas opiniões para realizar ajustes no algoritmo e serem uma parte importante no trabalho futuro. Resumidamente, as sugestões foram as seguintes:

- **Avaliação do volume e intensidade:** esta etapa foi a que teceu mais análise e críticas por parte do perito. O volume e a intensidade de cada treino são calculados através da obtenção do número de repetições e do seu valor de intensidade (escala numérica representada por 0, 1, 2 ou 3), respetivamente. Segundo o especialista, o presente cálculo possui um conceito impreciso e vago de avaliar o volume e, ainda mais, a intensidade.

    - **Volume:** comparando dois treinos, um treino com maior número de repetições não significa que possui um maior volume, pois existem outras variáveis que influenciam o volume, como a duração ou exercícios repetidos;
    - **Intensidade:** a avaliação da intensidade, além de considerar as métricas percentagem e RPE, o que segundo o perito são variáveis que coincidem corretamente com o tema, poderia incluir o peso levantado e a frequência cardíaca.
- **Generalização do volume e intensidade:** na fase de avaliação do volume e intensidade, estes estão a ser generalizados para todo o treino. Esta generalização pode resultar numa má representação destes parâmetros. Uma sugestão seria avaliar cada exercício individualmente e fazer suposições para o peso que ele deveria ter sobre o valor geral do treino, com base na sua dificuldade, duração, número de *sets*, repetições ou intensidade. Por exemplo, um exercício que tenha uma maior dificuldade e um número de *sets* maior, deve ter um peso superior na avaliação do treino.
- **Filtragem de treinos:** as filtragens de treino poderiam ser detalhadamente analisadas antes de aplicadas. Isto é, o histórico de treinos do utilizador podia ser minuciosamente avaliado com o objetivo de calcular em forma de percentagem o peso que cada variável teve na realização dos treinos. Por exemplo, se em 90% dos exercícios de um treino realizado pelo utilizador, o músculo peito estiver presente, é expectável que treinos com a vertente peito sejam mais do interesse deste.

Tendo em conta o feedback disponibilizado pelo especialista, conclui-se, que, apesar da média de respostas do questionário não ser superior a 4 (3.7), o resultado final para a primeira versão deste algoritmo foi positivo.

6.4.2 Tempo de execução

Foi executado um teste com o intuito de analisar o tempo de execução do algoritmo responsável pela sugestão de treinos (recomendação condicional). Este teste foi executado trinta vezes num computador com o processador Intel™ Core™ i5-4690K CPU @ 3.50GHZ utilizando uma vasta quantidade de dados: 250 mil treinos possíveis de sugerir e 200 treinos realizados pelo utilizador (histórico).

Os resultados de todas as instâncias do teste encontram-se presentes no Anexo 5, estando a média, o menor e o maior valor de todos esses resultados presentes na Tabela 16.

Tabela 16 – Resultados em milissegundos do teste de tempo de execução

Menor tempo (ms)	Maior tempo (ms)	Média (ms)
848	903	872

Perante uma grande quantidade de dados, o resultado do teste foi bastante positivo, não havendo uma variação significativa entre os tempos obtidos, e todos estes respeitam o tempo limite de 2 segundos, definido na hipótese relacionada.

6.4.3 Erro de previsão

Para esta grandeza existem dois tipos de teste a serem executados que vão ao encontro das hipóteses formuladas. Para cada um dos testes foram aplicadas as métricas já descritas (secção 6.3) e os seus resultados encontram-se apresentados e analisados nas duas secções seguintes. Para complementar, foram realizadas outras experiências que também serão descritas ao longo desta secção.

6.4.3.1 Erro de previsão de cada uma das técnicas

Todas as técnicas do sistema de recomendação foram avaliadas utilizando as métricas MAE e RMSE. As técnicas sujeitas ao teste foram as seguintes:

- **Filtragem colaborativa:** baseada em utilizadores (FCBU) e em itens (FCBI);
- ***Matrix Factorization* (MF):** baseada em modelos;
- **Híbrida:** todas as combinações possíveis, ou seja – MF + FCBU; MF + FCBI; FCBU + FCBI; MF + FCBU + FCBI.

Para o presente teste não foram realizadas variações nas variáveis dos algoritmos das técnicas do SR. Assim sendo, na *Matrix Factorization* o número de iterações é igual a 100 e o número de *latent features* é igual a 20. Já na filtragem colaborativa o número de utilizadores/itens semelhantes ou vizinhos é igual a 20.

Relativamente à métrica de cálculo de semelhança usada na técnica colaborativa, foi utilizada a *Co-Rated Cosine Similarity*, visto ser a medida que obteve o menor erro de previsão. Este assunto encontra-se aprofundado na secção 6.4.3.3 – Outras experiências.

Como referido ao longo do documento, apenas a técnica *Matrix Factorization* apresenta um processo iterativo de aprendizagem inteligente para a criação de um modelo, sendo o valor do erro de previsão significativamente diferente em cada iteração de teste. Por outro lado, as técnicas colaborativas não possuem inteligência na criação do modelo, sendo o valor do erro de previsão igual em cada iteração. Portanto, apenas uma iteração foi executada com o objetivo de determinar o erro de previsão das técnicas colaborativas e híbrida (FCBU + FCBI). Por outro lado, trinta iterações foram executadas nas técnicas *Matrix Factorization* e híbrida (MF + FCBU, MF + FCBI e MF + FCBU + FCBI), e no fim calculada a média. Os resultados das trinta iterações das técnicas mencionadas encontram-se no Anexo 6.

Tabela 17 – Erros de previsão de cada técnica de recomendação

ID	Técnica do SR	MAE	RMSE	$RMSE - MAE$	$\frac{MAE + RMSE}{2}$
1	MF	0,77	1,01	0,24	0,89
2	FCBU	1,80	2,15	0,35	1,98
3	FCBI	0,91	1,11	0,20	1,01
4	MF + FCBU	1,55	1,78	0,23	1,67
5	MF + FCBI	0,78	0,98	0,20	0,88
6	FCBU + FCBI	1,79	1,95	0,16	1,87
7	MF + FCBU + FCBI	1,42	1,59	0,17	1,51

Na Tabela 17 encontram-se ilustrados os valores resultantes da aplicação das métricas a cada técnica. Como se pode observar, a técnica com menor erro de previsão é a 5, seguida da 1. Ambas possuem uma técnica em comum – *Matrix Factorization*. Por outro lado, a técnica que possui um erro maior é a filtragem colaborativa baseada em utilizadores.

As razões que justificam a diferença dos valores das técnicas não são evidentes e de fácil dedução. Podem estar relacionadas com pormenores dos algoritmos, dimensão do *dataset*, relações entre os utilizadores e itens, entre outras. Contudo, pode-se concluir que nas condições descritas, excluindo as combinações entre as técnicas, *Matrix Factorization* foi a técnica que obteve um menor erro de previsão.

Comparando os valores das métricas MAE e RMSE, é possível constatar que os valores relacionados com a segunda métrica são sempre superiores em qualquer técnica. Por norma, os valores do RMSE são sempre maiores quando comparados com valores do MAE [166], e nesta avaliação esse padrão mantém-se. Essa superioridade no valor do erro justifica-se pelo facto que a métrica RMSE tende a penalizar valores discrepantes ou previsões fracas. Contudo, a diferença entre os valores das duas métricas não apresenta uma variação significativa (média igual a 0,22).

Analisando as combinações entre as várias técnicas (híbrido), a que obteve um menor erro de previsão foi a combinação entre as técnicas *Matrix Factorization* e filtragem colaborativa baseada em itens. Este resultado é totalmente compreensível, pois, como analisado em cima, foram as técnicas, excluindo as combinações, que obtiveram um menor erro.

Relativamente à hipótese especificada para esta grandeza, erro de previsão inferior a 1,5, neste caso, o erro de previsão é o resultado da média entre os valores das métricas MAE e RMSE, pode-se concluir que o resultado para este teste foi positivo.

De uma forma geral, três das sete técnicas analisadas possuem um erro de previsão inferior a 1,5, sendo que duas dessas quatro que têm um erro superior a 1,5, apresentam um valor muito aproximado de 1,5.

Como documentado ao longo do relatório, a técnica mais valorizada e priorizada para o presente projeto é a *Matrix Factorization*. Os resultados para esta técnica foram bastante

positivos. De facto, possui valores dos erros baixos e os erros das métricas MAE e RMSE são abaixo de 1,5.

6.4.3.2 Impacto da variação do valor de k no erro de previsão

Este teste divide-se em duas partes: (i) avaliar o impacto da variação do número de *latent features* na técnica *Matrix Factorization* e (ii) avaliar o impacto da variação do número de vizinhos na filtragem colaborativa baseada em utilizadores.

Para a parte (i) foram realizadas sete variações através do aumento do número de *latent features* (1, 5, 10, 15, 20, 30 e 40). Para cada variação foram executadas vinte iterações de teste, calculando os erros de teste das métricas MAE e RMSE, e o erro de treino através da métrica RMSE. Os resultados de todas as iterações encontram-se no Anexo 7.

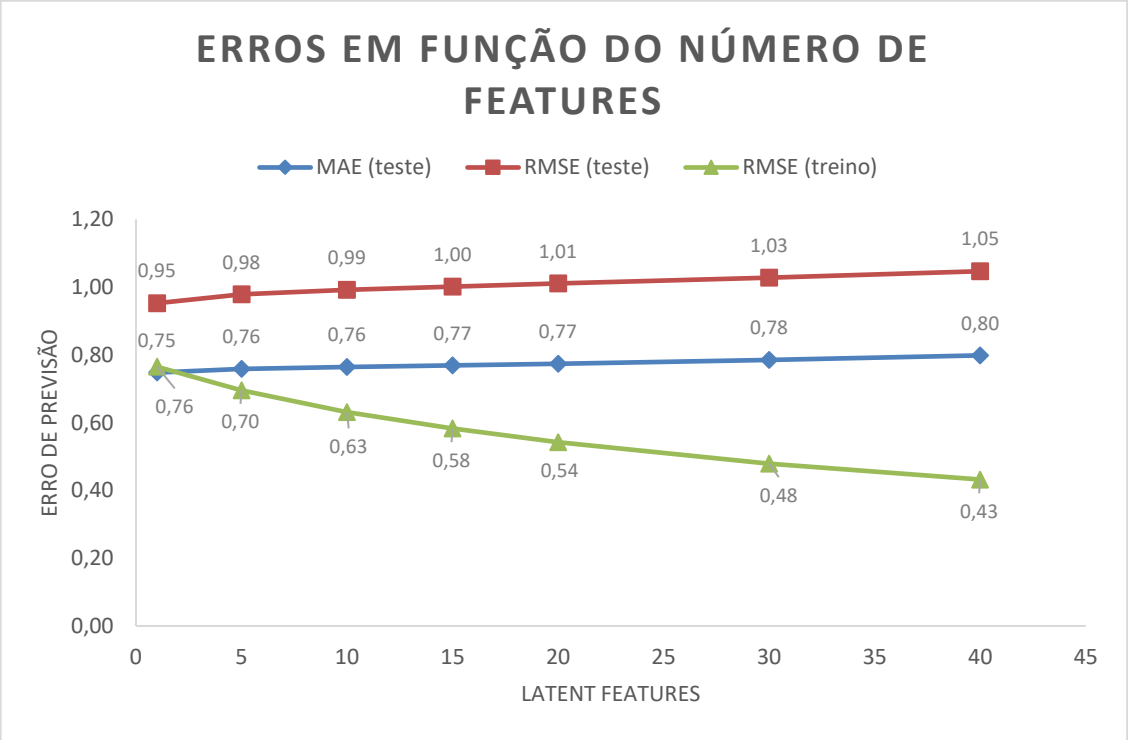


Figura 110 – Erros de previsão em função do número de *latent features*

No gráfico da Figura 110 é possível observar os valores dos diversos erros de previsão (médias dos resultados presentes no Anexo 7) em função da variação do número de *latent features*.

Analisando apenas os valores das métricas de teste, conclui-se que o aumento do número de *latent features* resulta num erro de previsão maior, embora o seu aumento entre cada variação não seja significativamente alto. Mais uma vez, e como já analisado no Erro de previsão de cada uma das técnicas, os valores da métrica de teste RMSE são sempre superiores aos valores da métrica MAE.

Relativamente aos valores do erro de treino, é possível observar que o aumento do número de *latent features* resulta num erro de previsão menor. Neste caso, a descida do valor do erro

é acentuada. De facto, para uma *feature* o erro é igual a 0,76 e para 40 features o erro desce para 0,43, o que demonstra ser um bom indicador.

Comparando os valores do erro de teste e do erro de treino da métrica RMSE, conclui-se que estes têm comportamentos distintos. Na verdade, o de teste tem um contínuo crescimento, enquanto o de treino um acentuado decréscimo. Além disso, a diferença dos seus valores entre cada variação é significativamente alta, tendo o seu auge na *feature* 40 (0,61).

Quando o erro de treino é consideravelmente mais baixo que o erro de teste, está-se perante um modelo *overfitted* relativamente aos dados do *dataset* [167]. *Overfitting* é originado quando um modelo se adapta ou aprende facilmente com os dados do treino, mas tende a falhar na generalização dos dados que nunca viu. As razões por detrás desta adversidade podem estar relacionadas com a existência de diversos parâmetros no modelo (alta capacidade) ou no elevado número de iterações. Para solucionar este problema, a técnica de *data science cross-validation* pode ser aplicada, isto é, criar modelos com diferentes graus e avaliar cada um deles usando a validação cruzada. O modelo com a pontuação mais baixa terá o melhor desempenho e alcançará um equilíbrio relativamente ao *overfitting* [167]. Adicionalmente, diminuir a complexidade do modelo também pode ser uma solução.

Analisando os erros obtidos com a hipótese formulada, conclui-se, como referido anteriormente, que o aumento do valor de k (*latent features*), aumenta o erro de previsão relacionado com o teste e diminui o erro de previsão relativo ao treino.

Para a parte (ii) foram realizadas sete variações através do aumento do número de vizinhos (1, 5, 10, 15, 20, 30 e 40). Para cada variação foram calculados os erros de teste das métricas MAE e RMSE. Desta vez, foi utilizada a *Pearson Correlation Similarity* como métrica de cálculo de semelhança. Foi escolhida esta como base de demonstração para o presente teste, mas poderia ser escolhida uma outra das restantes três.

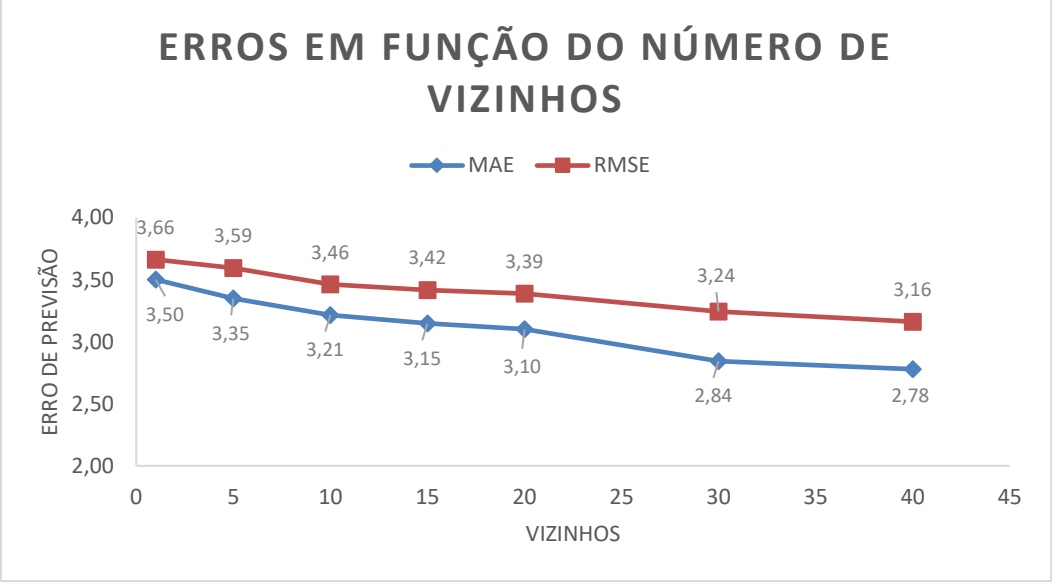


Figura 111 – Erros de previsão em função do número de vizinhos

No gráfico da Figura 111 é possível observar os valores dos erros de previsão em função da variação do número de vizinhos/utilizadores-itens semelhantes.

Analisando os valores dos erros, conclui-se que o aumento do número de vizinhos resulta num erro de previsão menor. Este resultado é de fácil interpretação. De facto, aumentando o número de vizinhos, o algoritmo responsável por retornar as sugestões analisará mais classificações de itens dos utilizadores considerados mais semelhantes, o que resulta, como demonstrado no gráfico, num menor erro e numa melhor *performance*.

Relativamente à hipótese formulada, conclui-se, como referido anteriormente, que o aumento do valor de k (número de vizinhos), diminui o erro de previsão.

6.4.3.3 Outras experiências

Foram realizadas mais duas experiências: (i) variação do número de iterações em função do erro de treino na técnica Matrix Factorization e (ii) cálculo do erro de previsão da técnica filtragem colaborativa baseada em utilizadores utilizando todas as medidas de cálculo de semelhança.

Para a experiência (i) foram executados três testes, um com 100 iterações – grupo A, outro com 200 – grupo B, e o último com 500 – grupo C. Os resultados detalhados de cada um dos testes encontram-se no Anexo 8.

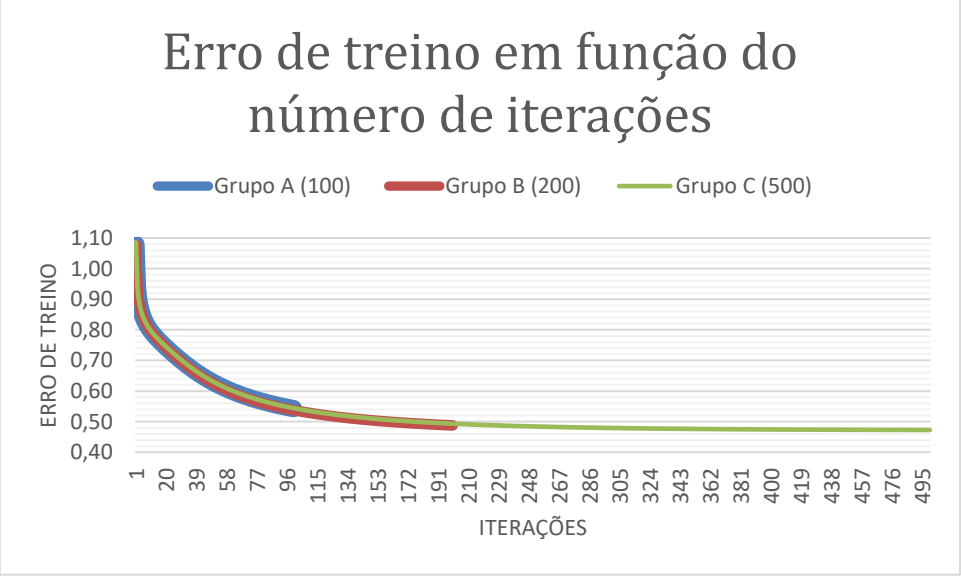


Figura 112 – Erro de treino em função do número de iterações

No gráfico da Figura 112 é possível observar os erros de treino para cada iteração. Por exemplo, para a iteração número 1, o erro de treino dos grupos A e B é igual a 1,08 e do grupo C é igual a 1,09.

Através da análise do gráfico é possível concluir que:

- O erro de treino tende a decrescer em função do número de iterações;
- O maior decréscimo do erro ocorre entre as iterações 1 a 150, sendo que a partir daí o decréscimo não é tão acentuado;
- A diferença do erro de treino entre cada grupo (100, 200 e 500) para cada iteração é mínima, sendo os valores praticamente iguais. Por exemplo, para a iteração número 50, a maior diferença é entre os grupos B e C com o valor da diferença igual a 0,0057. Outro aspeto que evidencia essa diferença mínima é a sobreposição das linhas de cada grupo.

Para a experiência (ii) foram executados quatros testes para calcular o erro de previsão usando as métricas MAE e RMSE na técnica de filtragem colaborativa baseada em utilizadores utilizando as quatro medidas de cálculo de semelhança.

Tabela 18 – Erro de previsão para as medidas de cálculo de semelhança

ID	Medida de cálculo de semelhança	MAE	RMSE
1	*Pearson Correlation Similarity*	3,10	3,39
2	*Cosine Similarity*	2,04	2,33
3	*Co-Rated Cosine Similarity*	1,80	2,15
4	*Root Mean Square Similarity*	3,16	3,37

Observando os resultados dos erros de previsão para cada medida de cálculo presentes na Tabela 18, é possível concluir que as medidas *Cosine* e a sua variação *Co-Rated Cosine*, são as que apresentam menor erro de previsão, tanto na métrica MAE como na RMSE. Por outro lado, *Root Mean Square Similarity* apresenta um maior erro na métrica MAE e a *Pearson Correlation Similarity* um maior erro na métrica RMSE.

7 Conclusão

A elaboração desta dissertação teve como principal objetivo o desenvolvimento de um sistema de *fitness* aliado ao domínio de exercícios de treino, contemplando diferentes módulos: aplicação móvel, sistema de recomendação e assistente pessoal. Este sistema é direcionado para utilizadores que praticam ginásio, sendo-lhes disponibilizado uma variedade de funcionalidades que os vai auxiliar na prática de exercício físico.

Em primeiro lugar, o tema deste projeto foi contextualizado, juntamente com seu domínio. De seguida, foi efetuada uma análise que permitisse elaborar uma proposta de valor e determinar o contributo das aplicações *fitness* na atividade dos indivíduos. Foi possível concluir que este tipo de aplicações tem bastante procura por parte dos praticantes de ginásio, tendo uma oportunidade de negócio extraordinária.

Depois de realizada uma análise de valor para o presente projeto, foi feita uma investigação a múltiplas aplicações fitness. Através dessa análise foi possível descobrir quais as funcionalidades imprescindíveis na prática de ginásio. Além disso, as principais técnicas de sistema de recomendação foram estudadas, com o intuito de identificar as que melhor se adequavam ao sistema a desenvolver.

Através dos resultados da referida investigação, em conjunto com os resultados de um inquérito desenvolvido sobre a análise de utilização de aplicações de ginásio, foi possível determinar quais os requisitos funcionais e não funcionais para o presente projeto, bem como a prioridade de cada caso de uso.

No processo de decisão de qual a melhor arquitetura a implementar, foi escolhida a arquitetura que possui uma API centralizada com lógica de negócio e acesso à base de dados, podendo, numa fase posterior, evoluir lentamente para micro serviços.

Relativamente à fase mais extensa do projeto, implementação, todos os módulos pretendidos foram abordados e implementados.

Numa primeira fase de implementação, as funcionalidades presentes nos requisitos funcionais foram implementadas. Nesta etapa, a aplicação móvel e o servidor foram os componentes desenvolvidos, resultando numa aplicação que permite ao utilizador realizar o registo e o login, definir o seu perfil de atleta de ginásio, e executar as funcionalidades relativas ao domínio desta dissertação, como gerir e filtrar exercícios, visualizar treinos existentes e criar e registar dados do treino efetuado, de forma manual ou em tempo real.

Numa fase posterior, um sistema de recomendação inteligente foi desenvolvido. Foi neste componente que se deparou com uma considerável limitação – falta de dados. Considera-se que esta limitação foi superada com sucesso. De facto, foi utilizado um *dataset* existente (*MovieLens*) que não se adequa ao tema do projeto, mas que permitiu desenvolver e testar um sistema que possui diversas técnicas de recomendação, tais como: *Matrix Factorization*, filtragem colaborativa baseada em utilizadores e itens, e híbrida. A estrutura de dados do SR foi implementada de forma a que quando se adquirisse um aceitável conjunto de dados relativos a classificações de treinos por parte dos utilizadores, fosse extremamente fácil a sua importação para o sistema.

Visto que, com a utilização de um *dataset* não correspondente ao tema do presente projeto não permitia a sugestão de treinos, foi desenvolvido um sistema de recomendação condicional. Este sistema com base no histórico de treinos e preferências do utilizador, utiliza filtros e técnicas matemáticas para sugerir os melhores treinos a este.

Na última fase de implementação, foi desenvolvido um componente responsável por interagir com o utilizador de forma a esclarecer dúvidas acerca de exercícios. Foi utilizado o *LUIS*, serviço da Microsoft baseado em *machine learning* responsável por criar linguagem natural para utilização com o *bot*. Desta forma, todas as mensagens introduzidas pelo utilizador são interpretas por este serviço que retorna a intenção desejada (p.e. lista de exercícios, dicas, instruções ou erros comuns de um determinado exercício). Obtendo a intenção apropriada, resta comunicar com o servidor para retornar os dados da base de dados necessários.

Tendo os módulos implementados, foi necessária a realização de uma avaliação para demonstrar a utilidade da ferramenta desenvolvida. Desta forma, foram utilizadas diferentes metodologias para avaliar cada componente, como inquéritos de satisfação, avaliação por parte de um perito, aplicação de métricas e testes unitários.

Resumidamente, os inquéritos revelam um elevado grau de satisfação no funcionamento geral da aplicação, nas suas funcionalidades e na sua utilidade. Além disso, também foi possível obter um resultado muito satisfatório na *performance* dos algoritmos sujeitos a teste.

Conclui-se, assim, que, o resultado final deste projeto é bastante positivo, existindo um enorme sentimento de satisfação.

De seguida, os objetivos para este projeto serão detalhados conforme o seu alcance, bem como as limitações e o trabalho futuro.

7.1 Objetivos Alcançados

Nesta secção apresenta-se a relação dos objetivos gerais da aplicação mencionados no capítulo da Introdução, bem como alguns objetivos específicos de elevada importância para o presente sistema, com o seu grau de tangibilidade.

Tabela 19 – Objetivos do projeto e seu grau de tangibilidade

Objetivo	**Grau de tangibilidade**
Investigar e analisar diversas aplicações/sistemas fitness	Atingido
Investigar e modelar o domínio "exercícios e treinos de fitness"	Atingido
Efetuar o levantamento e especificação dos requisitos	Atingido
Analisar propostas de arquitetura e implementar uma que satisfaça os requisitos identificados	Atingido
Desenvolver um módulo de recomendação através do uso de técnicas de Inteligência Artificial	Atingido
Permitir o registo de informação de treinos	Atingido
Consultar informação dos exercícios e treinos existentes, bem como os treinos já realizados	Atingido
Integração com o *personal trainer*	Não atingido
Recomendar treinos adaptados às necessidades, perfil e histórico do utilizador	Atingido
Desenvolver um módulo de esclarecimento de dúvidas de exercícios através de um assistente pessoal	Atingido
Monitorizar e controlar os dados do utilizador, tendo como objetivo capturar a sua evolução	Não atingido
Realizar um estudo que demonstre a utilidade da ferramenta desenvolvida, seguida da sua análise	Atingido

Como ilustrado na Tabela 19, os objetivos – integração com o *personal trainer* (treinador), isto é, um componente onde os treinadores podem orientar e acompanhar a evolução dos seus orientandos; e uma monitorização detalhada da evolução dos utilizadores, foram os únicos a não serem atingidos. Apesar de serem consideras peças importantes para o presente sistema, foi dada uma maior prioridade a outros componentes, como o sistema de recomendação, o assistente pessoal, entre outros.

Dos componentes descritos na secção Arquitetura escolhida – *FitnessMobileApp, FitnessBackOffice, FitnessWebApp, FitnessAPI, ChatBot, LuisAPI, FitnessDatabase e FitnessRecommenderSystem, o FitnessBackOffice e o FitnessWebApp* não foram implementados. Mais uma vez, foi dada uma maior prioridade aos restantes componentes, visto que o principal objetivo era possuir uma aplicação móvel que comunicasse com um servidor com base de dados, contendo módulos de recomendação e esclarecimento de dúvidas. Não havendo uma necessidade, nesta primeira fase, da existência de uma aplicação *web* para os utilizadores ou administradores.

Tabela 20 – Casos de uso em função da prioridade e da sua ou não realização

Nome	Prioridade	Realizado
UA01: Registar Utilizador	5	Sim
UA02: Login	5	Sim
UA03: Recuperar password	5	Sim
US01: Realizar *setup*	5	Sim
US03: Gerir exercícios	5	Sim
US08: Visualizar treinos	5	Sim
US09: Criar treino	5	Sim
US10: Registar dados do treino efetuado	5	Sim
US11: Realizar treino em tempo real	5	Sim
US12: Recomendação de treinos	4	Não
US04: Filtrar exercícios	4	Sim
US05: Pedir recomendação de exercícios	3	Não
US02: Gerir objetivos	2	Não
US06: Gerir exercícios favoritos	2	Não
US07: Visualizar estatísticas de exercícios	2	Não
US13: Importar e Exportar informação	2	Não
US14: Usar calculadoras	2	Não
AD01: Validar e gerir exercícios	2	Não
US15: Visualizar planos de nutrição	1	Não
PT01: Propor desafios	1	Não
AD02: Visualizar KPI	1	Não
NU01: Gerir planos de nutrição	1	Não

Relativamente aos requisitos funcionais (casos de uso), foram realizados dez dos vinte e dois enumerados. Inicialmente, foi atribuída a prioridade de cada funcionalidade numa escala numérica entre 1 a 5, sendo o valor 1 correspondente à menor prioridade e o valor 5 à maior (ver Tabela 20). Tendo em conta isto, dos onze casos de uso com prioridade mais alta (4 e 5), foram realizados dez, correspondente a 91% de casos de uso com prioridade alta implementados.

De facto, numa primeira abordagem, levantamento e especificação de requisitos, todas as funcionalidades desejadas para o sistema foram documentadas. Na fase de implementação foi dado prioridade aos casos de uso com prioridade igual a 4 ou 5, daí o resultado mencionado anteriormente. De realçar, que outros componentes como o SR condicional, o SR inteligente e o assistente pessoal, embora não estejam descritos nos casos de uso, fizeram parte do processo de implementação e ficaram totalmente implementados, embora com possíveis melhorais.

7.2 Limitações e Trabalho Futuro

Apesar do esforço de desenvolvimento para atingir objetivos e mitigar as suas limitações, geralmente, não é possível suprir todos os requisitos e limitações. Na verdade, um projeto dificilmente pode ser considerado como acabado e completo, dado que existem sempre melhorias e modificações que podem ser feitas.

A grande limitação do presente projeto relaciona-se com a falta de dados relativa a classificações de treinos por parte de diferentes utilizadores, ou dados relacionados, como mencionado na secção Dataset do SR. De facto, esta falta de dados impossibilitou a geração de treinos usando técnicas inteligentes. Contudo, conclui-se, que, esta limitação foi superada com sucesso através de: utilização de um *dataset* existente, que permitiu reconhecer o bom desempenho do sistema desenvolvido; e da criação de um sistema de recomendação condicional, que permitiu sugerir treinos, embora feita de uma forma não inteligente.

Por outro lado, existem certas melhorias a serem realizadas. Visto que, o sistema possui diversos módulos, de seguida, para cada um deles, será indicado o trabalho futuro a ser empregue:

- **Sistema de Recomendação inteligente:**
    - Recolher dados de classificações de treinos atribuídas pelos utilizadores e posterior aplicação no sistema;
    - Implementar novas técnicas de recomendação, por exemplo: baseada em conteúdo – incluir o perfil e os atributos dos utilizadores; baseada em conhecimento – incluir regras ou requisitos impostos pelos utilizadores; e demográfica – mapeamento de utilizadores através dos seus atributos pessoas;
    - Implementar novas medidas de cálculo de semelhança para as técnicas de filtragem colaborativa;
    - Aplicar a técnica *cross-validation* ou diminuir a complexidade do modelo criado na técnica *Matrix Factorization* com o objetivo de diminuir o valor do erro de previsão de teste, removendo assim o *overfitting;*
    - Realizar testes de maneira a averiguar, analisar e resolver o elevado valor de erro de previsão da técnica filtragem colaborativa baseada em utilizadores;
- **Sistema de Recomendação condicional:**
    - Aplicar as sugestões aconselhadas pelo perito (descritas na secção 6.4.1.3);
    - Utilizar novos parâmetros nos filtros dos treinos (p.e. equipamentos disponíveis ou restrições/requisitos do utilizador);
- **Assistente pessoal (*Personal Virtual Assistant*):**
    - Adicionar novas intenções/funcionalidades;

- Integração de um FAQ[10] (*Frequently Asked Questions*) dinâmico com as questões mais pertinentes perguntadas pelos utilizadores;
- Treinar o modelo de previsão através da adição de novas expressões para cada intenção, de forma a melhorar o resultado da interpretação da intenção (*LUIS*).

- **Aplicação móvel e servidor:**
  - Implementação dos casos de uso em falta;
  - Ajustes na interface móvel de forma a responder ao resultado do inquérito de satisfação, isto é, ao resultado da adaptabilidade da interface gráfica no dispositivo móvel.

[10] https://www.qnamaker.ai/

Referências

[1] IHRSA, "IHRSA 2018 Global Report: Health Club Industry Revenue Totaled $87.2…," *IHRSA*, 29-May-2018. [Online]. Available: https://www.ihrsa.org/about/media-center/press-releases/ihrsa-2018-global-report-club-industry-revenue-totaled-87-2-billion-in-2017/. [Accessed: 27-Jan-2019].

[2] Deloitte, "European Health & Fitness Market - Report 2018." 2018.

[3] C. NUNES, "Expresso | Negócio dos ginásios está a 'encher,'" *Jornal Expresso*, 11-Feb-2018. [Online]. Available: //expresso.pt/economia/2018-02-11-Negocio--dos-ginasios-esta-a-encher. [Accessed: 27-Jan-2019].

[4] D. E. R. Warburton, C. W. Nicol, and S. S. D. Bredin, "Health benefits of physical activity: the evidence," *CMAJ Can. Med. Assoc. J.*, vol. 174, no. 6, pp. 801–809, Mar. 2006.

[5] J. P. Higgins, "Smartphone Applications for Patients' Health and Fitness," *Am. J. Med.*, vol. 129, no. 1, pp. 11–19, Jan. 2016.

[6] L. Kesiraju and T. Vogels, "Health & Fitness App Users Are Going the Distance with Record-High Engagement," *Flurry Blog*, 07-Sep-2017. [Online]. Available: https://flurrymobile.tumblr.com/post/165079311062/health-fitness-app-users-are-going-the-distance. [Accessed: 30-Jan-2019].

[7] G. Mary, S. Gwin, M. Cheney, and T. Wann, "Health and Fitness App Use in College Students: A Qualitative Study," *ResearchGate*, Jul-2015. [Online]. Available: https://www.researchgate.net/publication/282317897_Health_and_Fitness_App_Use_in_College_Students_A_Qualitative_Study. [Accessed: 31-Jan-2019].

[8] M. D. Molina and S. S. Sundar, "Can Mobile Apps Motivate Fitness Tracking? A Study of Technological Affordances and Workout Behaviors," *Health Commun.*, pp. 1–10, Oct. 2018.

[9] C. Cardoso, "Want better results at the gym? A structured program could be the answer," *Copeman Healthcare Centre*. .

[10] World Health Organization, "Physical activity," 23-Feb-2018. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/physical-activity. [Accessed: 27-Jan-2019].

[11] B. Midgley, "The Six Reasons The Fitness Industry Is Booming," *Forbes*, 26-Sep-2018. [Online]. Available: https://www.forbes.com/sites/benmidgley/2018/09/26/the-six-reasons-the-fitness-industry-is-booming/. [Accessed: 27-Jan-2019].

[12] A. S. Rojas, "I'm super-setting my life! An ethnographic comparative analysis of the growth of the gym market," *Sport Sci. Rev.*, vol. 25, no. 5–6, pp. 276–299, Dec. 2016.

[13] Les Mills, "Fitness is the world's biggest sport," 2013.

[14] AGAP, "Barómetro do Fitness em Portugal," 2016.

[15] E. Lord, "How health and fitness became a trillion dollar industry," *Lightspeed POS*, 14-Jan-2017. .

[16] D. P. SONI, "The rise and rise of fitness industry: A case of Trio-Fit," *Medium*, 18-Jul-2017. .

[17] V. Jain, "#8 Reasons why Popularity of Gyms is Increasing Exponentially," *Entrepreneur*, 05-Jul-2017. [Online]. Available: https://www.entrepreneur.com/article/296797. [Accessed: 29-Jan-2019].

[18] R. Bailey, "Fitness Franchise Industry Report 2018 | FranchiseDirect.com," *Franchise Direct*, 10-Sep-2018. [Online]. Available: https://www.franchisedirect.com/information/fitness-franchise-industry-report-2018. [Accessed: 29-Jan-2019].

[19] L. Bell, “The Future Of Health And Fitness: Personalization And Genetic Insight Through DNA Testing,” *Forbes*, 09-Feb-2018. [Online]. Available: https://www.forbes.com/sites/leebelltech/2018/02/09/genetic-insight-is-the-future-of-health-and-fitness-and-heres-why/. [Accessed: 29-Jan-2019].
[20] 9yards, “The Future of the Fitness Industry is not just Fitness,” *9 Yards*, 29-Nov-2017. [Online]. Available: https://www.9yards.com/news/the-future-of-fitness. [Accessed: 29-Jan-2019].
[21] K. Jang Yul, “Determinants of Users Intention to Adopt Mobile Fitness Applications: an Extended Technology Acceptance Model Approach,” *Health Exerc. Sports Sci. ETDs*, p. 129, Dec. 2014.
[22] A. Gabbiadini and T. Greitemeyer, “Fitness mobile apps positively affect attitudes, perceived behavioral control and physical activities,” *J. Sports Med. Phys. Fitness*, Apr. 2018.
[23] R. R. Pai and S. Alathur, “Assessing mobile health applications with twitter analytics,” *Int. J. Med. Inf.*, vol. 113, pp. 72–84, May 2018.
[24] P. R. Sama, Z. J. Eapen, K. P. Weinfurt, B. R. Shah, and K. A. Schulman, “An Evaluation of Mobile Health Application Tools,” *JMIR MHealth UHealth*, vol. 2, no. 2, p. e19, May 2014.
[25] H. E. Payne, C. Lister, J. H. West, and J. M. Bernhardt, “Behavioral Functionality of Mobile Apps in Health Interventions: A Systematic Review of the Literature,” *JMIR MHealth UHealth*, vol. 3, no. 1, Feb. 2015.
[26] M. Bracko, “Institute for Hockey Research - Dr. Mike.” [Online]. Available: http://www.hockeyinstitute.org/profile.htm. [Accessed: 16-Feb-2019].
[27] MedlinePlus, “Exercise and Physical Fitness,” 30-Aug-2017. [Online]. Available: https://medlineplus.gov/exerciseandphysicalfitness.html. [Accessed: 02-Feb-2019].
[28] Y. Ali, “Is Exercise the Magic Pill?,” *Verywell Health*, 17-Apr-2017. [Online]. Available: https://www.verywellhealth.com/exercise-the-magic-pill-2509617. [Accessed: 02-Feb-2019].
[29] Harvard Health, “The 4 most important types of exercise,” *Harvard Health*, Jan-2017. [Online]. Available: https://www.health.harvard.edu/exercise-and-fitness/the-4-most-important-types-of-exercise. [Accessed: 02-Feb-2019].
[30] Powerliftingtowin, “What is Powerlifting? Who Powerlifts? Why?,” *PowerliftingToWin*, 29-Jan-2014. .
[31] O. Walker, “Olympic Weightlifting,” *Science for Sport*, 08-Apr-2016. .
[32] American Council on Exercise, “Exercise Library: Chest Press,” *Exercise Library: Chest Press*, 2018. [Online]. Available: /education-and-resources/lifestyle/exercise-library/5/chest-press. [Accessed: 04-Feb-2019].
[33] Jefit, “Exercise Database | Jefit - Best Android and iPhone Workout, Fitness, Exercise and Bodybuilding App | Best Workout Tracking Software,” *Exercise By BodyPart*, 2018. [Online]. Available: https://www.jefit.com/exercises/. [Accessed: 04-Feb-2019].
[34] A. Lane, “Gym Equipment Guide,” *GymVentures*, 23-Jan-2016. [Online]. Available: https://www.gymventures.com/gym-equipment-names-and-pictures/. [Accessed: 04-Feb-2019].
[35] N. Gammell, S. Morrell, and D. Rasmussen, “Method And System For Creating Personalized Workout Programs,” US20110281249A1, 17-Nov-2011.
[36] Jefit, “Jefit: Barbell Bench Press,” *Jefit Workout App*, 2018. [Online]. Available: https://www.jefit.com/exercises/2/Barbell-Bench-Press. [Accessed: 04-Feb-2019].
[37] American Council on Exercise, “Exercise Library: Plank-ups,” 2018. [Online]. Available: /education-and-resources/lifestyle/exercise-library/320/plank-ups. [Accessed: 04-Feb-2019].

[38] Jefit, “Jefit: Plank,” *Jefit Workout App*, 2018. [Online]. Available: https://www.jefit.com/exercises/631/Plank. [Accessed: 04-Feb-2019].
[39] American Council on Exercise, “Exercise Library: Deadlift,” 2018. [Online]. Available: /education-and-resources/lifestyle/exercise-library/6/deadlift. [Accessed: 04-Feb-2019].
[40] Jefit, “Jefit: Barbell Sumo Deadlift,” *Jefit Workout App*, 2018. [Online]. Available: https://www.jefit.com/exercises/496/Barbell-Sumo-Deadlift. [Accessed: 04-Feb-2019].
[41] P. A. Koen *et al.*, “Fuzzy Front End : Effective Methods , Tools , and Techniques,” 2002.
[42] P. Koen *et al.*, “Providing Clarity and A Common Language to the ‘Fuzzy Front End,’” *Res.-Technol. Manag.*, vol. 44, no. 2, pp. 46–55, Mar. 2001.
[43] C. Grönroos and P. Voima, “Making Sense of Value and Value Co-Creation in Service Logic,” Hanken School of Economics, Working Paper, Jan. 2012.
[44] D. Morar, “An overview of the consumer value literature – perceived value, desired value,” *ResearchGate*, Nov. 2013.
[45] T. Woodall, “Conceptualising ‘value for the customer’: an attributional, structural and dispositional analysis,” *Acad. Mark. Sci. Rev.*, vol. 2003, 2003.
[46] M. Skok, “4 Steps To Building A Compelling Value Proposition,” *Forbes*, 14-May-2014. [Online]. Available: https://www.forbes.com/sites/michaelskok/2013/06/14/4-steps-to-building-a-compelling-value-proposition/. [Accessed: 06-Feb-2019].
[47] A. Osterwalder and Y. Pigneur, *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. OSF, 2009.
[48] S. Raza and C. Ding, “Progress in context-aware recommender systems — An overview,” *Comput. Sci. Rev.*, vol. 31, pp. 84–97, Feb. 2019.
[49] R. Burke, A. Felfernig, and M. H. Göker, “Recommender Systems: An Overview,” *AI Mag.*, vol. 32, no. 3, pp. 13–18, Jun. 2011.
[50] E. Vozalis and K. G. Margaritis, “Analysis of Recommender Systems’ Algorithms,” p. 14, 2003.
[51] C. C. Aggarwal, “An Introduction to Recommender Systems,” in *Recommender Systems: The Textbook*, C. C. Aggarwal, Ed. Cham: Springer International Publishing, 2016, pp. 1–28.
[52] S. Khusro, Z. Ali, and I. Ullah, “Recommender Systems: Issues, Challenges, and Research Opportunities,” in *Information Science and Applications (ICISA) 2016*, 2016, pp. 1179–1189.
[53] Y. Koren, R. Bell, and C. Volinsky, “Matrix Factorization Techniques for Recommender Systems,” *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[54] S. Ghosh, “Simple Matrix Factorization example on the Movielens dataset using Pyspark,” *Medium*, 31-Mar-2018. [Online]. Available: https://medium.com/@connectwithghosh/simple-matrix-factorization-example-on-the-movielens-dataset-using-pyspark-9b7e3f567536. [Accessed: 11-Sep-2019].
[55] R. Burke, “Hybrid Recommender Systems: Survey and Experiments,” *User Model. User-Adapt. Interact.*, vol. 12, no. 4, pp. 331–370, Nov. 2002.
[56] E. Ezin, E. Kim, and I. Palomares, “‘Fitness that Fits’: A prototype model for Workout Video Recommendation,” p. 6, 2018.
[57] S. Dharia, V. Jain, J. Patel, J. Vora, S. Chawla, and M. Eirinaki, “PRO-Fit: A personalized fitness assistant framework,” presented at the The 28th International Conference on Software Engineering and Knowledge Engineering, 2016, pp. 386–389.
[58] M. Donciu, M. Ionita, M. Dascalu, and S. Trausan-Matu, “The Runner – Recommender System of Workout and Nutrition for Runners,” in *2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2011, pp. 230–238.
[59] JavaServer Faces, “JavaServer Faces.org.” [Online]. Available: http://www.javaserverfaces.org/. [Accessed: 12-Oct-2019].

[60] J. Skillings, “Apple reveals details of 50 billionth App Store download,” *CNET*, 13-May-2013. [Online]. Available: https://www.cnet.com/news/apple-reveals-details-of-50-billionth-app-store-download/. [Accessed: 23-Jan-2019].
[61] Pocketgamer.biz, “App Store Metrics,” 23-Jan-2019. [Online]. Available: https://www.pocketgamer.biz/metrics/app-store//. [Accessed: 23-Jan-2019].
[62] Pocketgamer.biz, “App Store Metrics Categories,” 23-Jan-2019. [Online]. Available: https://www.pocketgamer.biz/metrics/app-store/categories/. [Accessed: 23-Jan-2019].
[63] Appbrain, “Number of Android applications on the Google Play store,” *AppBrain*, 23-Jan-2019. [Online]. Available: https://www.appbrain.com/stats/number-of-android-apps. [Accessed: 23-Jan-2019].
[64] AppBrain, “Top categories on Google Play,” *AppBrain*, 23-Jan-2019. [Online]. Available: https://www.appbrain.com/stats/android-market-app-categories. [Accessed: 23-Jan-2019].
[65] J. CORPUZ and B. EASTWOOD, “Best Workout Apps to Get Healthy and Stay Fit in 2019,” *Tom’s Guide*, 04-Jan-2019. [Online]. Available: https://www.tomsguide.com/us/pictures-story/702-best-workout-apps.html. [Accessed: 31-Jan-2019].
[66] B. J. Duffy, “The Best Fitness Apps for 2019,” *PCMAG*, 28-Dec-2018. [Online]. Available: https://www.pcmag.com/article2/0,2817,2485287,00.asp. [Accessed: 24-Jan-2019].
[67] J. HINDY, “15 best Android fitness apps - Android Authority,” 17-Oct-2018. [Online]. Available: https://www.androidauthority.com/best-fitness-apps-android-567999/. [Accessed: 24-Jan-2019].
[68] C. Haslam, “Best workout apps we’ve used: improve your fitness in just 20 minutes a day,” *TechRadar*, 29-Dec-2018. [Online]. Available: https://www.techradar.com/news/best-fitness-apps-best-hiit-apps-best-strength-apps. [Accessed: 24-Jan-2019].
[69] A. Tubek and M. Duplaga, “The assessment of functionalities of mobile applications supporting physical activity,” *Rozpr. Nauk. Akad. Wych. Fiz. We Wrocławiu*, vol. 61, pp. 77–87, 2018.
[70] Perigee, “Seven – 7 Minute Workout App by Perigee,” *Perigee*, 2018. .
[71] App Store, “Seven - 7 Minute Workout,” *App Store*, 2017. [Online]. Available: https://itunes.apple.com/us/app/seven-7-minute-workout/id650276551?mt=8. [Accessed: 17-Dec-2018].
[72] Apple, “Siri,” *Apple*. [Online]. Available: https://www.apple.com/siri/. [Accessed: 17-Dec-2018].
[73] Apple, “iOS - Health,” *Apple*. [Online]. Available: https://www.apple.com/ios/health/. [Accessed: 17-Dec-2018].
[74] Google Play, “Desafio Fitness de 30 Dias – Aplicações no Google Play,” 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.popularapp.thirtydayfitnesschallenge&hl=pt_PT. [Accessed: 18-Dec-2018].
[75] Intensity, “Intensity - Progression Based Workout Tracking App For Android & iOS,” *Intensity*. [Online]. Available: https://www.intensityapp.com/. [Accessed: 18-Dec-2018].
[76] Google Play, “Freeletics Bodyweight – Aplicações no Google Play,” 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.freeletics.lite&hl=pt_PT. [Accessed: 11-Jan-2019].
[77] App Store, “Freeletics Bodyweight,” *App Store*, 2018. [Online]. Available: https://itunes.apple.com/pt/app/freeletics-bodyweight/id654810212?mt=8. [Accessed: 11-Jan-2019].
[78] Freeletics, “Coach de Treino | FREELETICS,” 2018. [Online]. Available: https://www.freeletics.com/pt/bodyweight/coach/get. [Accessed: 11-Jan-2019].

[79] R. Moraes, “Freeletics Bodyweight – Google Play (Comentário),” 24-Dec-2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.freeletics.lite&hl=pt_PT&reviewId=gp%3AAOqpTOEaBUH4KxkIO9GpnVvpMRDzHBOiYQ5KXbNz2kzUHNhxTYooR7yKqXdSJGhZIxQMy36nV0QlGA9rZZvQ2YI. [Accessed: 11-Jan-2019].
[80] AmazinGym, “AmazinGym.” [Online]. Available: https://amazingym.pt. [Accessed: 09-Jan-2019].
[81] Virtuagym, “Fitness Software for Gyms & Personal Trainers,” *Virtuagym*, 2018. [Online]. Available: https://virtuagym.com/software/en/. [Accessed: 09-Jan-2019].
[82] 7minuteworkout, “Official 7 Minute Workout | Johnson & Johnson,” *Official 7 Minute Workout | Johnson & Johnson*, 11-Dec-2013. [Online]. Available: https://7minuteworkout.jnj.com/. [Accessed: 12-Jan-2019].
[83] Keelo, “Keelo - Strength High Intensity HIIT Workouts,” 2018. [Online]. Available: https://keelo.com/. [Accessed: 14-Jan-2019].
[84] App Store, “Keelo - Strength HIIT Workouts,” *App Store*. [Online]. Available: https://itunes.apple.com/us/app/keelo-strength-hiit-workouts/id1004824537?mt=8. [Accessed: 15-Jan-2019].
[85] Google Play, “Keelo - Strength HIIT Workouts WOD at Home & Gym – Aplicações no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=us.throwdown.keelo&hl=pt_PT. [Accessed: 15-Jan-2019].
[86] H. Kim, “How Keelo Recommends Your Next Workout,” *Keelo*, 03-Jun-2016. .
[87] Strong, “Strong Workout Tracker Gym Log,” 2018. [Online]. Available: https://strong.app/. [Accessed: 15-Jan-2019].
[88] Jefit, “Jefit,” *Jefit - #1 Gym workout app*, 2018. [Online]. Available: https://www.jefit.com/. [Accessed: 16-Jan-2019].
[89] B. J. Duffy, 2018 4:45PM EST September 14, and 2018 September 14, “The Best Heart Rate Monitors for 2019,” *PCMAG*, 04-Dec-2018. [Online]. Available: https://www.pcmag.com/roundup/352257/the-best-heart-rate-monitors. [Accessed: 21-Jan-2019].
[90] N. Rozanski and E. Woods, *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2011.
[91] OMG, *Unified Modeling Language Specification*. 2001.
[92] A. Cockburn, *Writing Effective Use Cases*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.
[93] M. Fowler, *Patterns of Enterprise Application Architecture*, 1 edition. Boston: Addison-Wesley Professional, 2002.
[94] P. Kruchten, *The Rational Unified Process: An Introduction*. Addison-Wesley Professional, 2004.
[95] P. B. Kruchten, “The 4+1 View Model of architecture,” *IEEE Softw.*, vol. 12, no. 6, pp. 42–50, Nov. 1995.
[96] C. Richardson, “Microservices Pattern: Database per service,” *microservices.io*. [Online]. Available: http://microservices.io/patterns/data/database-per-service.html. [Accessed: 12-Oct-2019].
[97] C. Richardson, “Microservices Pattern: API gateway pattern,” *microservices.io*. [Online]. Available: http://microservices.io/patterns/apigateway.html. [Accessed: 12-Oct-2019].
[98] C. Richardson, “Microservices Pattern: Monolithic Architecture pattern,” *microservices.io*. [Online]. Available: http://microservices.io/patterns/monolithic.html. [Accessed: 14-Feb-2019].

[99] C. Richardson, “Microservices Pattern: Microservice Architecture pattern,” *microservices.io*. [Online]. Available: http://microservices.io/patterns/microservices.html. [Accessed: 14-Feb-2019].
[100] Microsoft, “Plataforma de Computação na Cloud e Serviços do Microsoft Azure,” 2019. [Online]. Available: https://azure.microsoft.com/pt-pt/. [Accessed: 11-Oct-2019].
[101] O. Sobajic, M. Moussavi, and B. Far, “Parameterized strategy pattern,” in *Proceedings of the 17th Conference on Pattern Languages of Programs - PLOP ’10*, Reno, Nevada, 2010, pp. 1–11.
[102] J. Chen, “Line Of Best Fit,” *Investopedia*, 20-Apr-2019. [Online]. Available: https://www.investopedia.com/terms/l/line-of-best-fit.asp. [Accessed: 16-Sep-2019].
[103] MyCurveFit, “Online curve-fitting at mycurvefit.com,” *MyCurveFit*, 16-Sep-2019. [Online]. Available: https://www.mycurvefit.com. [Accessed: 16-Sep-2019].
[104] N. Katakam, “How Can We ‘Design’ An Intelligent Recommendation Engine?,” *Medium*, 07-Jan-2019. [Online]. Available: https://uxplanet.org/how-can-we-design-an-intelligent-recommendation-engine-b9bb1db4d050. [Accessed: 23-Sep-2019].
[105] S. Clayton, “Building a Recommendation Engine in C#,” 22-Aug-2018. [Online]. Available: https://www.codeproject.com/Articles/1232150/Building-a-Recommendation-Engine-in-Csharp. [Accessed: 23-Sep-2019].
[106] CodeProject, “CodeProject - For those who code.” [Online]. Available: https://www.codeproject.com/. [Accessed: 11-Oct-2019].
[107] E. Team, “The Importance of Machine Learning and of Building Data Sets,” *insideBIGDATA*, 11-Oct-2017. .
[108] BPU Holdings, “The importance of having a good dataset,” *BPU Holdings*, 24-Dec-2018. .
[109] Grouplens, “MovieLens,” *GroupLens*, 06-Sep-2013. [Online]. Available: https://grouplens.org/datasets/movielens/. [Accessed: 18-Sep-2019].
[110] F. M. Harper and J. A. Konstan, “The MovieLens Datasets: History and Context,” *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Dec. 2015.
[111] W. Kirwin, “Implicit Recommender Systems - Biased Matrix Factorization,” 11-Jan-2016. [Online]. Available: http://activisiongamescience.github.io/2016/01/11/Implicit-Recommender-Systems-Biased-Matrix-Factorization/. [Accessed: 22-Sep-2019].
[112] M. Binieli, “Machine learning: an introduction to mean squared error and regression lines,” *freeCodeCamp.org*, 16-Oct-2018. [Online]. Available: https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/. [Accessed: 23-Sep-2019].
[113] P. Pandey, “Understanding the Mathematics behind Gradient Descent.,” *Medium*, 28-Mar-2019. [Online]. Available: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e. [Accessed: 23-Sep-2019].
[114] A. Agarwal, M. Chauhan, and Ghaziabad, “Similarity Measures used in Recommender Systems : A Study,” 2017.
[115] A. Gunawardana and G. Shani, “A Survey of Accuracy Evaluation Metrics of Recommendation Tasks,” p. 28, 2009.
[116] E. A. Coutsias, C. Seok, and K. A. Dill, “Using quaternions to calculate RMSD,” *J. Comput. Chem.*, vol. 25, no. 15, pp. 1849–1857, Nov. 2004.
[117] R. Native, “React Native · A framework for building native apps using React,” 2019. [Online]. Available: https://facebook.github.io/react-native/. [Accessed: 24-Sep-2019].
[118] Expo, “Expo,” *Expo*, 2019. [Online]. Available: https://expo.io/. [Accessed: 24-Sep-2019].
[119] Microsoft, “Visual Studio Code - Code Editing. Redefined.” [Online]. Available: https://code.visualstudio.com/. [Accessed: 24-Sep-2019].

[120] D. Abramov, “Getting Started with Redux · Redux,” 2015. [Online]. Available: https://redux.js.org/. [Accessed: 24-Sep-2019].
[121] D. Abramov, “Motivation · Redux,” 2015. [Online]. Available: https://redux.js.org/. [Accessed: 24-Sep-2019].
[122] M. Schwarzmüller, “Redux vs React’s Context API,” 19-Feb-2019. [Online]. Available: https://www.academind.com/learn/react/redux-vs-context-api/. [Accessed: 24-Sep-2019].
[123] React Native, “AsyncStorage · React Native.” [Online]. Available: https://facebook.github.io/react-native/. [Accessed: 24-Sep-2019].
[124] React Navigation, “Getting started · React Navigation.” [Online]. Available: https://reactnavigation.org/index.html. [Accessed: 24-Sep-2019].
[125] R. Bulat, “Introduction to React Navigation and Navigators in React Native,” *Medium*, 13-Aug-2018. [Online]. Available: https://medium.com/@rossbulat/introduction-to-react-navigation-and-navigators-in-react-native-3efcf7239a43. [Accessed: 24-Sep-2019].
[126] Expo, “Localization - Expo Documentation.” [Online]. Available: https://docs.expo.io/versions/v32.0.0/sdk/localization/. [Accessed: 24-Sep-2019].
[127] J. Saring, “11 React Native Component Libraries you Should Know in 2019,” *Medium*, 16-Sep-2019. [Online]. Available: https://blog.bitsrc.io/11-react-native-component-libraries-you-should-know-in-2018-71d2a8e33312. [Accessed: 24-Sep-2019].
[128] J. Arvidsson, *oblador/react-native-animatable*. 2019.
[129] J. Lauritzen, *Jacse/react-native-app-intro-slider*. 2019.
[130] E. Sharabi, *wix/react-native-calendars*. Wix.com, 2019.
[131] J. Arvidsson, *oblador/react-native-collapsible*. 2019.
[132] M. Mazzarolo, *mmazzarolo/react-native-dialog*. 2019.
[133] J. Katsumata and K. Roach, *react-native-training/react-native-elements*. React Native Training, 2019.
[134] F. Safi, *FaridSafi/react-native-gifted-chat*. 2019.
[135] N. Baugh, *joinspontaneous/react-native-loading-spinner-overlay*. Spontaneous, 2019.
[136] M. Milyutin, *mxck/react-native-material-menu*. 2019.
[137] A. Nazarov, *n4kz/react-native-material-textfield*. 2019.
[138] F. Bluemle and M. Mazzarolo, *react-native-community/react-native-modal*. React Native Community, 2019.
[139] React Native Paper, *callstack/react-native-paper*. Callstack, 2019.
[140] A. Vitanov, *i6mi6/react-native-parallax-scroll-view*. 2019.
[141] zooble, *beefe/react-native-picker*. beefe, 2019.
[142] LawnStarter, *lawnstarter/react-native-picker-select*. LawnStarter, 2019.
[143] J. Arvidsson, *oblador/react-native-progress*. 2019.
[144] H. lee, *oksktank/react-native-pure-chart*. 2019.
[145] R. Caferati, *rcaferati/react-native-really-awesome-button*. 2019.
[146] Agiletech, *react-native-vietnam/react-native-search-box*. React Native Vietnam, 2019.
[147] Ren, *renrizzolo/react-native-sectioned-multi-select*. 2019.
[148] B. Delmaire and M. Bertonnier, *archriss/react-native-snap-carousel*. Archriss, 2019.
[149] D. McPherson, *deanmcpherson/react-native-sortable-listview*. 2019.
[150] M. Stevens, *michaeljstevens/react-native-stopwatch-timer*. 2019.
[151] J. Hanson, *jshanson7/react-native-swipeable*. 2019.
[152] Akveo team, *akveo/react-native-ui-kitten*. akveo, 2019.
[153] jongalloway, “Introduction to ASP.NET Identity - ASP.NET 4.x,” 22-Jan-2019. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity. [Accessed: 28-Sep-2019].

[154] Microsoft, “RijndaelManaged Class (System.Security.Cryptography).” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.rijndaelmanaged. [Accessed: 28-Sep-2019].
[155] R. Rosa and A. Faleiros, “ANÁLISE DO ALGORITMO VENCEDOR DO AES: O RIJNDAEL,” 2003.
[156] L. Mathias, “Algoritmo de Criptografia AES,” 2005. [Online]. Available: https://www.gta.ufrj.br/grad/05_2/aes/. [Accessed: 28-Sep-2019].
[157] Microsoft, “Rfc2898DeriveBytes Class (System.Security.Cryptography).” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.rfc2898derivebytes. [Accessed: 28-Sep-2019].
[158] LastPass, “Iterações de Senhas (PBKDF2).” 2017.
[159] ivorb, “About Direct Line channel - Bot Service,” 02-Feb-2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-channel-directline. [Accessed: 29-Sep-2019].
[160] okta, “Access Token Lifetime,” *OAuth 2.0 Servers*. .
[161] P. Bourque and R. E. Fairley, *Software Engineering Body of Knowledge (SWEBOK)*. 2014.
[162] P. Gomes, “Unit Testing and the Arrange, Act and Assert (AAA) Pattern,” *Medium*, 13-Jan-2018. [Online]. Available: https://medium.com/@pjbgf/title-testing-code-ocd-and-the-aaa-pattern-df453975ab80. [Accessed: 10-Oct-2019].
[163] C. Pinela, “How to evaluate Recommender Systems,” *Medium*, 29-Nov-2017. [Online]. Available: https://medium.com/@cfpinela/how-to-evaluate-recommender-systems-874a313d7724. [Accessed: 04-Oct-2019].
[164] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Recommender Systems for Large-scale E-Commerce scalable neighborhood formation using clustering,” vol. 50, Jan. 2002.
[165] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” *Egypt. Inform. J.*, vol. 16, no. 3, pp. 261–273, Nov. 2015.
[166] T. Rackaitis, “Evaluating Recommender Systems: Root Means Squared Error or Mean Absolute Error?,” *Medium*, 31-May-2019. [Online]. Available: https://towardsdatascience.com/evaluating-recommender-systems-root-means-squared-error-or-mean-absolute-error-1744abc2beac. [Accessed: 06-Oct-2019].
[167] W. Koehrsen, “Overfitting vs. Underfitting: A Complete Example,” *Medium*, 28-Jan-2018. [Online]. Available: https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765. [Accessed: 07-Oct-2019].

Anexos

Anexo 1

Análise de Utilização de Aplicações de Ginásio/Fitness

Este questionário é realizado no âmbito da disciplina Tese de Mestrado (TMDEI) no curso Mestrado de Engenharia de Software no Instituto Superior de Engenharia do Porto.
O objetivo é recolher dados acerca da utilização de aplicações fitness, bem como da sua não utilização.

Agradecemos, desde já, a sua disponibilidade ao preencher o questionário, realçando, que, todos os dados inseridos são confidenciais e que serão apenas usados para análise deste projeto.
Prevemos que o seu preenchimento seja breve, não demorando mais do que 3 minutos.

Qualquer dúvida ou sugestão por favor contactar: 1140400@isep.ipp.pt ou 1140402@isep.ipp.pt

*Obrigatório

Objetivos

Tendo em conta que o tema da Tese de Mestrado é o desenvolvimento de uma aplicação fitness, este questionário tem como objetivo analisar vários aspetos da utilização de aplicações relacionados com o presente tema, tais como:

- objetivos da sua utilização;
- quais as funcionalidades mais importantes;
- principais razões de como as aplicações fitness podem ajudar a atingir os objetivos específicos;
- funcionalidades que poderiam ser inseridas;
- analisar o porquê da sua não utilização;

Dados pessoais

1. Sexo *

Marcar apenas uma oval.

- ◯ Masculino
- ◯ Feminino

2. Idade *

_____

3. Frequência de utilização do ginásio *

Marcar apenas uma oval.

- ◯ 1-2x por semana
- ◯ 2-3x por semana
- ◯ 3-4x por semana
- ◯ 4-5x por semana
- ◯ 5x por semana ou mais

4. Há quanto tempo anda no ginásio? *

Marcar apenas uma oval.

- ◯ Menos de 6 meses
- ◯ 6 meses a 2 anos
- ◯ 2 a 3 anos
- ◯ 3 a 5 anos
- ◯ 5 a 10 anos
- ◯ Mais de 10 anos

5. Objetivos no ginásio (selecione 1 ou mais) *

Marcar tudo o que for aplicável.

- ☐ Ganhar músculo
- ☐ Perder peso
- ☐ Ganhar força
- ☐ Preparação física (orientada a algum desporto/atividade física)
- ☐ Ganhar peso
- ☐ Manter estilo de vida saudável

Plano de treino

6. Segue um plano de treino? *

Marcar apenas uma oval.

- ◯ Não
- ◯ Sim, feito por mim
- ◯ Sim, feito pelo Personal Trainer
- ◯ Sim, feito por uma aplicação fitness

7. Regista plano de treino? *

Marcar apenas uma oval.

- ◯ Não
- ◯ Sim, em papel
- ◯ Sim, no Excel ou semelhante
- ◯ Sim, numa aplicação fitness

Uso de aplicações fitness

8. Utiliza ou já utilizou aplicações fitness (para registo de dados, planeamento, etc)? *

Marcar apenas uma oval.

- ◯ Utilizo neste momento *Passe para a pergunta 10.*
- ◯ Já utilizei, mas não utilizo neste momento *Passe para a pergunta 14.*
- ◯ Não utilizo e nunca utilizei *Passe para a pergunta 9.*

Não utiliza aplicações de fitness

9. Porque nunca utilizou aplicações fitness? *

Marcar apenas uma oval.

- ◯ Não tenho interesse *Pare de preencher este formulário.*
- ◯ Não uso dispositivos eletrónicos durante o treino *Pare de preencher este formulário.*
- ◯ Ainda não encontrei uma que tivesse as funcionalidades desejadas *Passe para a pergunta 16.*
- ◯ Tenho um treino demasiado específico *Passe para a pergunta 16.*
- ◯ Outra: _____ *Pare de preencher este formulário.*

Pare de preencher este formulário.

Utilização

10. Qual o nome da aplicação fitness que usa?

_____

11. Há quanto tempo usa aplicações fitness?

Marcar apenas uma oval.

- ◯ 2 semanas ou menos
- ◯ Entre 2 a 4 semanas
- ◯ Entre 1 a 3 meses
- ◯ Entre 3 a 12 meses
- ◯ Mais de 1 ano

12. Quais as razões pelas quais usa a aplicação fitness? (selecione 1 ou mais) *

Marcar tudo o que for aplicável.

- ☐ Aplicação usada pelo ginásio que frequento
- ☐ Marcação de aulas
- ☐ Visualizar informações de exercícios e planos
- ☐ Consultar progresso (peso, % gordura, etc)
- ☐ Registar treinos realizados
- ☐ Planear treinos
- ☐ Acompanhar o treino em tempo real
- ☐ Comunicar com Personal Trainer
- ☐ Ver planos de nutrição
- ☐ Publicar nas redes sociais
- ☐ Sugestão automática de exercícios/treinos/planos por parte da aplicação
- ☐ Outra: _____

13. Pontos negativos da aplicação fitness que usa (selecione 1 ou mais) *

Marcar tudo o que for aplicável.

- [ ] Publicidade exagerada
- [ ] Registo de dados exaustivo
- [ ] Preço elevado
- [ ] Experiência de utilização medíocore
- [ ] Falta de funcionalidades
- [ ] Notificações incómodas
- [ ] Não existem pontos negativos
- [ ] Outra: _____

Passe para a pergunta 15.

Não Utilização

14. Porque deixou de usar aplicações fitness ? (selecione 1 ou mais) *

Marcar tudo o que for aplicável.

- [ ] Publicidade exagerada
- [ ] Registo de dados exaustivo
- [ ] Preço elevado
- [ ] Experiência de utilização pouco intuítiva
- [ ] Falta de funcionalidades
- [ ] Notificações incómodas
- [ ] Outra: _____

Passe para a pergunta 15.

Dispositivo de utilização

15. Em que dispositivo utiliza maioritariamente as aplicações de fitness? *

Marcar apenas uma oval.

- ◯ Android
- ◯ Apple
- ◯ Web Browser
- ◯ Outra: _____

Nova aplicação fitness

16. Gostaria que existisse uma aplicação de fitness que facilitasse o alcance dos seus objetivos? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não *Pare de preencher este formulário.*

Novas funcionalidades

Selecionar, em baixo, as funcionalidades que acharia mais atrativas numa nova aplicação fitness. Caso tenha alguma ideia não listada, adicionar em "Outra".

17. Que novas funcionalidades gostaria de ver numa nova aplicação fitness? (selecione 1 ou mais)

Marcar tudo o que for aplicável.

- [ ] Recomendação automática de exercícios/treinos/planos por parte da aplicação
- [ ] Acompanhamento por um Coach/PT (Monitorização da evolução, sugestões, ajuda, etc.)
- [ ] Assistente Pessoal Virtual (Esclarecimento de dúvidas de fitness com um "robô")
- [ ] Sincronização com SmartWatches
- [ ] Visualizar artigos científicos sobre Fitness
- [ ] Outra: _____

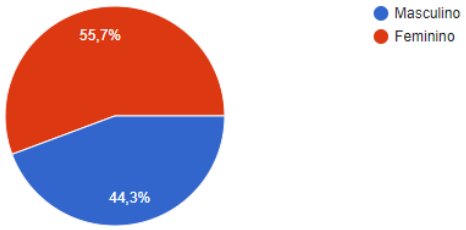
Resultados do Inquérito

88 respostas

Dados pessoais

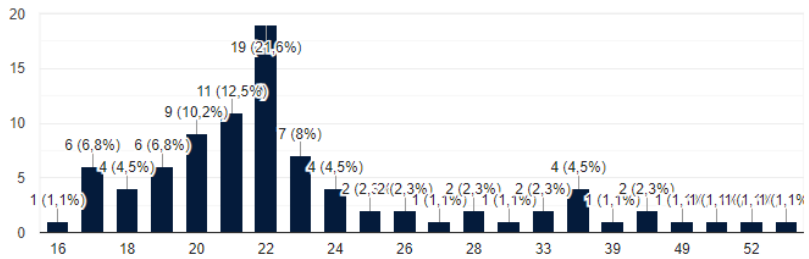
Sexo

88 respostas



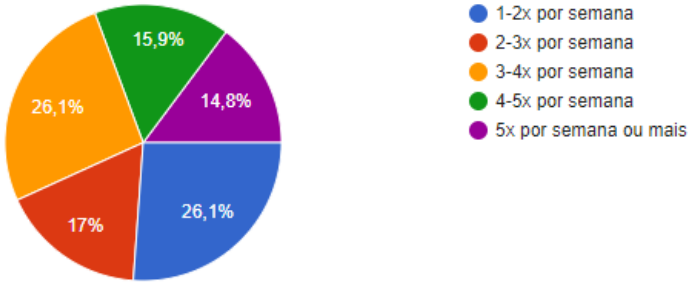
Idade

88 respostas



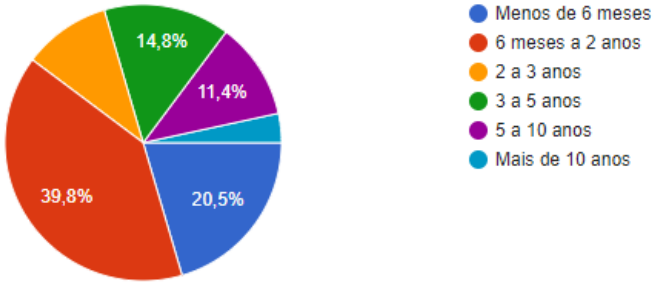
Frequência de utilização do ginásio

88 respostas



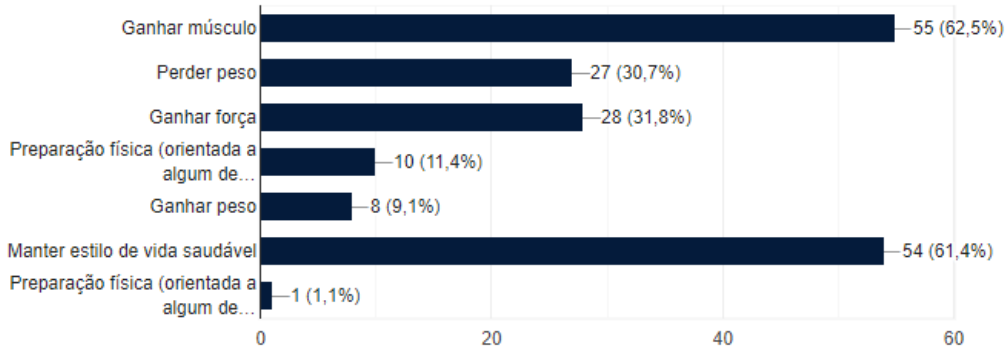
Há quanto tempo anda no ginásio?

88 respostas



Objetivos no ginásio (selecione 1 ou mais)

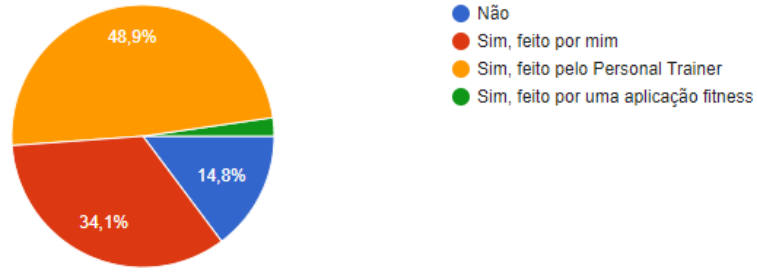
88 respostas



Plano de treino

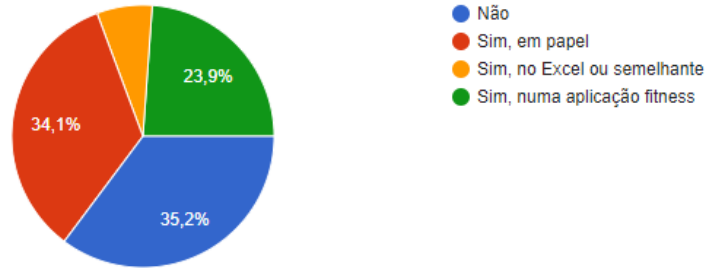
Segue um plano de treino?

88 respostas



Regista plano de treino?

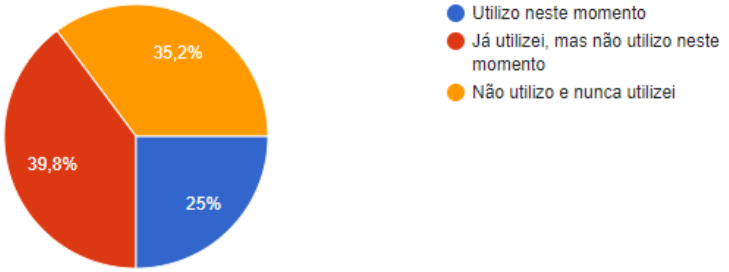
88 respostas



Uso de aplicações fitness

Utiliza ou já utilizou aplicações fitness (para registo de dados, planeamento, etc)?

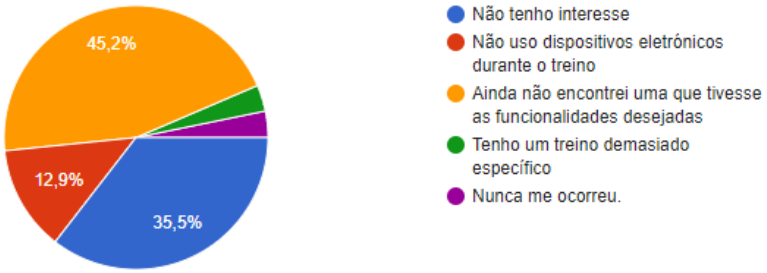
88 respostas



Não utiliza aplicações de fitness

Porque nunca utilizou aplicações fitness?

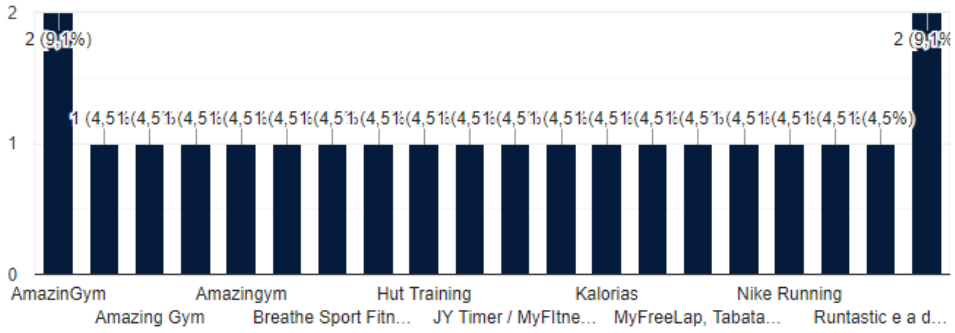
31 respostas



Utilização

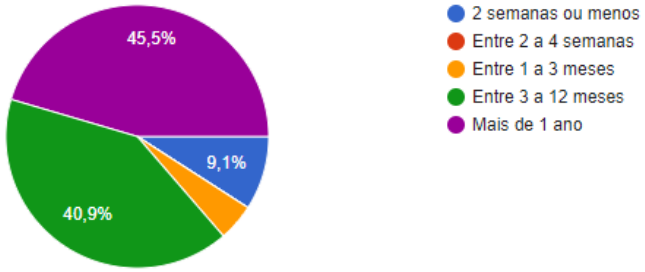
Qual o nome da aplicação fitness que usa?

22 respostas



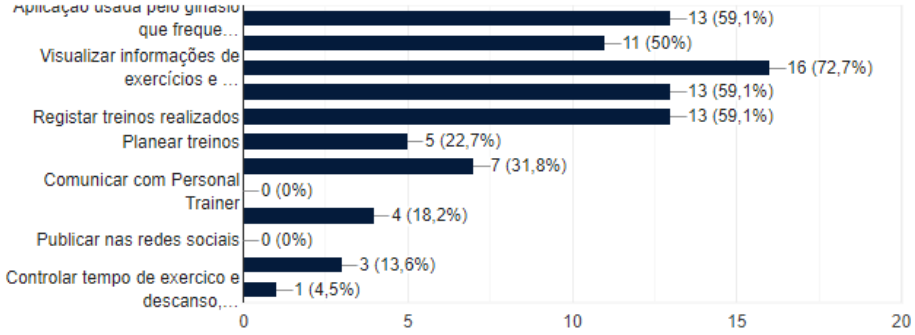
Há quanto tempo usa aplicações fitness?

22 respostas



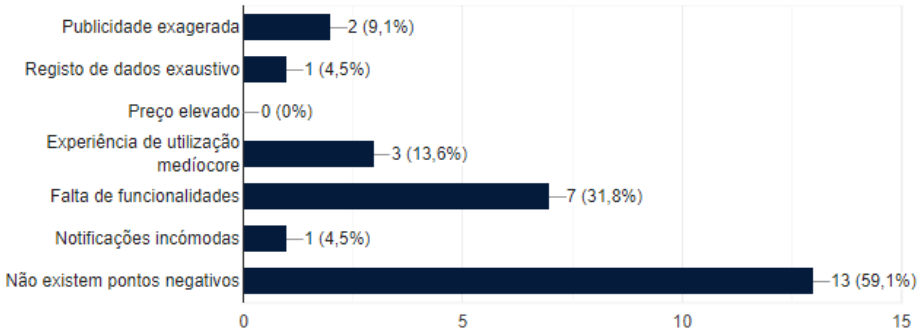
Quais as razões pelas quais usa a aplicação fitness? (selecione 1 ou mais)

22 respostas



Pontos negativos da aplicação fitness que usa (selecione 1 ou mais)

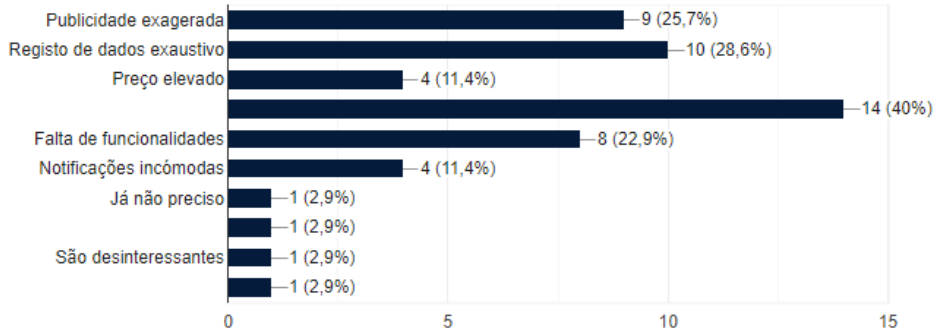
22 respostas



Não Utilização

Porque deixou de usar aplicações fitness ? (selecione 1 ou mais)

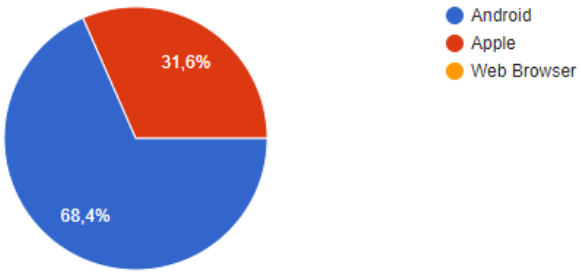
35 respostas



Dispositivo de utilização

Em que dispositivo utiliza maioritariamente as aplicações de fitness?

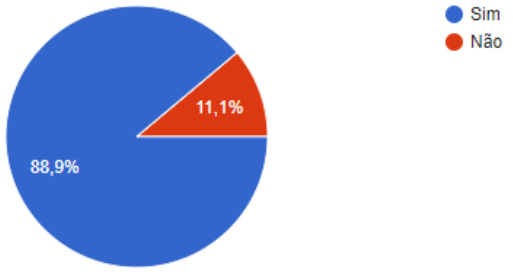
57 respostas



Nova aplicação fitness

Gostaria que existisse uma aplicação de fitness que facilitasse o alcance dos seus objetivos?

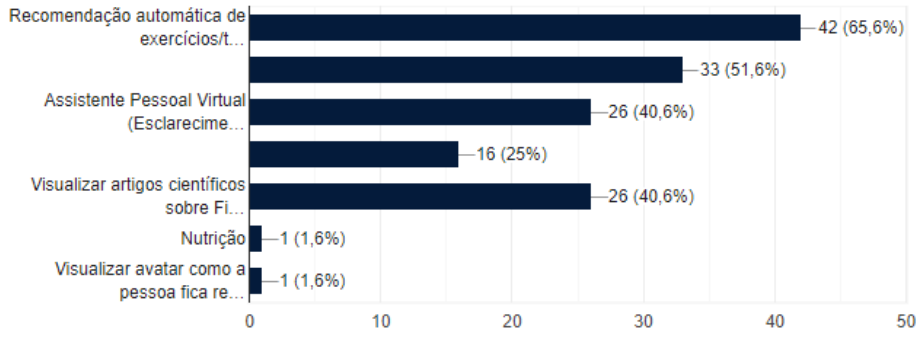
72 respostas



Novas funcionalidades

Que novas funcionalidades gostaria de ver numa nova aplicação fitness? (selecione 1 ou mais)

64 respostas



Anexo 2

Inquérito de uma aplicação móvel de fitness

Um sistema de auxílio à prática de exercício físico (ginásio) foi desenvolvido. O sistema disponibiliza diversas funcionalidades com componentes distintos (aplicação móvel, assistente pessoal e recomendações de treinos). Como forma de avaliar cada um destes componentes são apresentadas várias afirmações com o objetivo de as avaliar numa escala de 1 a 5: 1- Discordo completamente; 2- Discordo; 3- Sem opinião; 4- Concordo; e 5- Concordo completamente.

Aplicação Móvel

	1	2	3	4	5
A aplicação possui uma boa apresentação/design.					
A utilização da aplicação é intuitiva e simples.					
A aplicação é rápida.					
A aplicação está adaptada ao meu dispositivo.					
O processo de registo e login é realizado de forma rápida.					
O processo de criação de treinos e a sua realização é feita de forma simples e eficaz.					
É possível criar planos e programas de treino com todos os pormenores necessários à sua realização.					
Os filtros apresentados para os exercícios, treinos, entre outros, são eficientes na procura do desejado.					
A aplicação é de fácil configuração.					
A aplicação poderia ser útil enquanto pratico exercício físico (ginásio).					
Recomendaria a aplicação a qualquer praticante de ginásio.					
Classificaria a aplicação como muito boa.					

Assistente Pessoal

	1	2	3	4	5
A conversa com o assistente pessoal é fluída.					
O assistente pessoal consegue interpretar e responder com sentido ao que foi dito.					
O assistente pessoal esclarece com qualidade as dúvidas sobre os exercícios.					
O assistente pessoal é uma mais valia para a aplicação.					

Anexo 3

Inquérito ao sistema de recomendação condicional

Um sistema de recomendação de treinos foi desenvolvido. Como forma de avaliar a sua lógica e o conteúdo retornado são apresentadas várias afirmações com o objetivo de as avaliar numa escala de 1 a 5: 1- Discordo completamente; 2- Discordo; 3- Sem opinião; 4- Concordo; e 5- Concordo completamente.

	1	2	3	4	5
As recomendações sugeridas são boas.					
Os treinos recomendados fazem sentido tendo em conta os meus gostos/perfil.					
A avaliação do volume e intensidade é eficiente.					
A forma de previsão do volume e intensidade é eficiente.					
As variáveis usadas para filtrar são suficientes.					
A aplicação das variáveis é eficiente (filtros às não existentes).					

Anexo 4

Demonstração da Recomendação Condicional com Dados Reais

Em primeiro lugar, os dados dos treinos passados do especialista, juntamente com os seus dados preferenciais (p.e. lista de objetivos, tipos de treino, etc.) e a lista de treinos para recomendação, existentes na base de dados foram fornecidos ao algoritmo.

Devido a restrições de tempo, a base de dados de treinos para recomendação é relativamente reduzida, mas com diversidade suficiente para permitir a demonstração da aplicação de todos os filtros. Na tabela seguinte pode-se visualizar a informação destes mesmos, juntamente com as características que os definem.

Treinos na base de dados

ID	Nome	Nível	Objetivos	Tipos de Treino	Músculos	Área do Corpo	Intensidade [0-3]	Volume (Reps)
1	Chest	Intermédio	Ganhar Músculo, Ganhar Força	Hipertrofia, Força	Braços, Peito	Upper	1.6	150
2	Legs	Intermédio	Ganhar Músculo, Ganhar Força	Hipertrofia, Força	Glúteos, Upper Leg, Lower Leg	Lower	2	164
3	MAX Lower Body	Intermédio	Ganhar Músculo, Ganhar Força	Hipertrofia, Força	Glúteos, Upper Leg, Lower Leg	Lower	2.25	128
4	High Level Weightlifting	Avançado	Ganhar Músculo, Ganhar Força	Força, Weightlifting	Glúteos, Upper Leg, Upper Back, Core	Full	2.5	60
5	Low Level Weightlifting – Competition Peaking	Iniciante	Competição	Força, Weightlifting	Glúteos, Upper Leg, Upper Back, Core	Full	2.1	51
6	Low Level	Iniciante	Ganhar	Weightlifting	Glúteos,	Full	1.9	55

	Weightlifting		Músculo, Ganhar Força		Upper Leg, Upper Back, Core			
7	Upper Body Hypertrophy	Iniciante	Ganhar Músculo, Ganhar Força	Hipertrofia	Braços, Peito	Upper	1.5	166
8	Hypertrophy w/ Flexibility	Intermédio	Ganhar Músculo, Ganhar Força	Flexibilidade, Hipertrofia	Upper Leg, Lower Leg	Lower	1.8	137

Os dados do especialista – preferências e histórico de treinos – podem ser sumarizados nas seguintes tabelas, respetivamente.

Preferências do especialista

Nível	**Objetivos**	**Tipos de Treino**	**Áreas do Corpo**	**Músculos**
Experiente	Ganhar músculo	Hipertrofia, Powerlifting, Força, Flexibilidade	Full	Todos

Histórico e treinos do especialista

ID	**Nome**	**Nível**	**Objetivos**	**Tipos de Treino**	**Músculos**	**Área do Corpo**	**Intensidade**	**Volume**
1	Dia 1	Intermédio	Ganhar Músculo	Hipertrofia, Powerlifting, Força	Upper Back, Peito, Core, Braços, Ombros	Upper	2.7	133
2	Dia 2	Intermédio	Ganhar Músculo	Hipertrofia, Powerlifting, Força	Upper Back, Core, Braços, Ombros, Upper Leg	Lower	2.3	132
3	Dia 3	Intermédio	Ganhar Músculo	Hipertrofia, Powerlifting, Força	Upper Back, Core, Ombros	Upper	1.6	125
4	Dia 4	Intermédio	Ganhar Músculo	Hipertrofia, Powerlifting, Flexibilidade	Upper Back, Core, Ombros	Upper	1.6	171
5	Dia 5	Intermédio	Ganhar Músculo	Hipertrofia, Powerlifting, Força	Upper Back, Peito, Braços, Ombros	Upper	2.6	72
6	Dia 6	Intermédio	Ganhar Músculo	Hipertrofia, Powerlifting, Força	Upper Leg, Braços	Lower	2.5	165

Com o historial do utilizador em conta, foram calculadas as preferências em falta. Isto é, foram calculadas quais as preferências definidas que não foram encontradas no historial. Por exemplo, se os tipos de treino definidos sejam Hipertrofia, Força e Flexibilidade, e no historial existam treinos que apresentem Hipertrofia e Flexibilidade, mas nenhum apresente Força, então Força será o tipo de treino desejado para a recomendação do novo treino. Do mesmo modo, caso todos os tipos de treino se encontrem no historial, então as preferências não são ajustadas. Com isso em conta, as novas preferências são as seguintes.

Preferências ajustadas

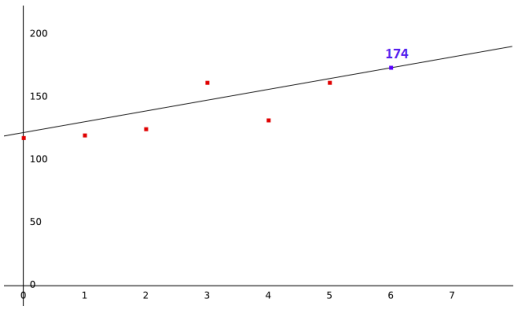
Nível	Objetivos	Tipos de Treino	Áreas do Corpo	Músculos
Experiente	Ganhar músculo	Hipertrofia, Powerlifting, Força, Flexibilidade	Full	Lower Leg, Glutes, Lower Back, Forearms

De seguida, os filtros das preferências ajustadas do especialista foram aplicados, resultando na remoção de alguns treinos da lista inicial, conforme a seguinte tabela.

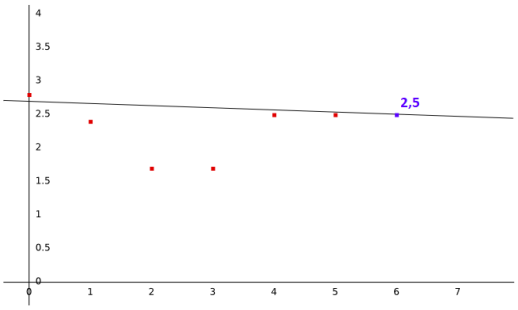
Explicação da filtragem

ID	Explicação
1	Removido por não conter músculos usados que se alinhem com os do especialista.
2	Mantido.
3	Mantido.
4	Removido por não conter tipos de treino que se alinhem com os do especialista.
5	Removido por não conter objetivos que se alinhem com os do especialista.
6	Removido por não conter tipos de treino que se alinhem com os do especialista.
7	Removido por não conter músculos usados que se alinhem com os do especialista.
8	Mantido.

Tendo a lista reduzida apenas aos treinos que se identificam melhor com as preferências do especialista, é possível fazer desde já uma recomendação com os 3 treinos. De qualquer das formas, falta decidir qual o treino que melhor se adapta, comparando o volume e intensidade. Para tal, foi previsto um valor ideal para volume e intensidade com base nos treinos passados do especialista, como evidenciado nas figuras seguintes.



Previsão do Volume



Previsão da Intensidade

Previsões para Volume e Intensidade

Cada ponto vermelho indica um valor de volume/intensidade nos treinos passados do especialista. Traçando uma *Line of Best Fit*, é possível prever que o próximo treino teria 174 de volume e 2.5 de intensidade. É possível também observar que há um evidente aumento de volume e uma ligeira descida de intensidade.

Com estes dados, é possível identificar qual dos 3 treinos se identifica melhor com os valores previstos. Para tal, o valor de volume e intensidade foi multiplicado, obtendo uma previsão de $174 \times 2.5 = 435$. Na seguinte tabela apresentam-se os valores desta multiplicação para os 3 treinos finalistas.

Valores de volume/intensidade das recomendações

ID	$\boldsymbol{volume \times intensidade}$	Diferença
2	$164 \times 2 = 328$	107
3	$128 \times 2.25 = 288$	147
8	$137 \times 1.8 = 246$	189

Com os dados da tabela anterior é possível verificar que o treino que melhor se adequa ao valor previsto da multiplicação do volume pela intensidade é o de ID 2, pois possui a menor diferença. Como tal, essa é a recomendação final.

Anexo 5

Tempos de execução do algoritmo do sistema de recomendação condicional

Instância	Tempo (milissegundos)
1	900
2	887
3	849
4	849
5	852
6	869
7	896
8	887
9	873
10	866
11	872
12	853
13	857
14	858
15	878
16	893
17	887
18	865
19	865
20	858
21	872
22	903
23	859
24	882
25	890
26	880
27	861
28	848
29	877
30	859
Média	872

Anexo 6

Resultados dos erros de previsão usando as métricas MAE e RMSE para todas as técnicas (parte 1)

Iteração	MF (MAE)	MF (RMSE)	MF + FCBU (MAE)	MF + FCBU (RMSE)
1	0,780741337	1,020981976	1,544661649	1,778041811
2	0,773678045	1,009705026	1,540897323	1,791600701
3	0,771248465	1,009431649	1,556980875	1,772536403
4	0,773458628	1,010285927	1,540348599	1,788668079
5	0,77253258	1,008379117	1,540547783	1,779759583
6	0,777445994	1,013401576	1,544101545	1,784917521
7	0,780634448	1,01848434	1,540551252	1,786001884
8	0,77409969	1,009143472	1,554931369	1,790697273
9	0,768428916	1,006784921	1,552707956	1,771527105
10	0,773762957	1,008885634	1,540518593	1,78018635
11	0,774318279	1,013125248	1,544556291	1,787400763
12	0,775188441	1,012791834	1,551600196	1,792311101
13	0,773376016	1,010853236	1,557600646	1,789161361
14	0,775024576	1,013839445	1,541110083	1,78662915
15	0,776068194	1,015591295	1,551961751	1,778176009
16	0,772797527	1,01091993	1,557188849	1,778479992
17	0,772310806	1,010548633	1,553033659	1,77333607
18	0,776740226	1,01661032	1,559333951	1,789102188
19	0,77662687	1,016914014	1,548349213	1,775831831
20	0,769370415	1,006815734	1,561049981	1,790941984
21	0,775822298	1,012851253	1,548143423	1,779388579
22	0,772858831	1,012412051	1,562113436	1,77146569
23	0,775619304	1,015798338	1,546018884	1,77402286
24	0,770441523	1,010192947	1,54351397	1,78283881
25	0,777423958	1,016942331	1,564757137	1,783454795
26	0,767840614	1,004311801	1,5541732	1,781932085
27	0,775743616	1,010901972	1,557993333	1,792319217
28	0,771313941	1,008865803	1,558095772	1,776831276
29	0,776263467	1,011504582	1,562568573	1,788955659
30	0,777653532	1,014749656	1,549606908	1,775385618

Resultados dos erros de previsão usando as métricas MAE e RMSE para todas as técnicas (parte 2)

Iteração	MF + FCBI (MAE)	MF + FCBI (RMSE)	MF + FCBU + FCBI (MAE)	MF + FCBU + FCBI (RMSE)
1	0,776793576	0,978792897	1,3016349	1,543623
2	0,77670431	0,978295258	1,457158	1,466774
3	0,777882338	0,980018474	1,3854201	1,647687
4	0,778357426	0,981806943	1,5183734	1,513453
5	0,7775417	0,978533108	1,3679033	1,34467
6	0,774071039	0,973035406	1,3325993	1,679842
7	0,778093017	0,980253022	1,3015361	1,6453677
8	0,777994909	0,979216783	1,3864883	1,5675343
9	0,776057609	0,976731799	1,3280456	1,2356778
10	0,781457111	0,982319407	1,4517297	1,752372
11	0,777762771	0,979624833	1,469889	1,467832
12	0,780915133	0,983225495	1,4501857	1,584485
13	0,78100988	0,983115516	1,5723531	1,6988452
14	0,776534005	0,976245014	1,3434897	1,785162
15	0,778993117	0,980042137	1,3053905	1,782166
16	0,775079187	0,977167858	1,3559547	1,3614564
17	0,778720309	0,977399832	1,4307969	1,7548132
18	0,776492227	0,977337846	1,5593589	1,6515621
19	0,778684343	0,980221778	1,4450246	1,2325154
20	0,776288623	0,979081319	1,329739	1,5618456
21	0,77482572	0,975281767	1,5380591	1,4516515
22	0,780044065	0,98116758	1,5124173	1,7981512
23	0,777365829	0,980496679	1,3833593	1,6211454
24	0,777430801	0,980406109	1,4234124	1,611551
25	0,774533068	0,975698974	1,5219934	1,621854
26	0,776491581	0,977195824	1,4311004	1,65156
27	0,776858354	0,97713926	1,4225993	1,74164
28	0,777952702	0,980675164	1,4128309	1,75445
29	0,778291803	0,979612319	1,4481453	1,56145
30	0,780927807	0,982530945	1,4176105	1,62153

Anexo 7

Resultados dos erros de previsão usando as métricas MAE e RMSE para a técnica *Matrix Factorization* variando o número de *latent features*

	1 *latent feature*			**5 *latent features***		
Iteração	**MAE (teste)**	**RMSE (teste)**	**RMSE (treino)**	**MAE (teste)**	**RMSE (teste)**	**RMSE (treino)**
1	0,750165	0,981586	0,699145	0,760787	0,699145	0,981586
2	0,749693	0,975489	0,700899	0,754733	0,700899	0,975489
3	0,74695	0,971281	0,696326	0,751727	0,696326	0,971281
4	0,744701	0,977798	0,691052	0,757874	0,691052	0,977798
5	0,747244	0,973677	0,698056	0,753749	0,698056	0,973677
6	0,747807	0,9805	0,696481	0,759216	0,696481	0,9805
7	0,746408	0,978129	0,692259	0,760722	0,692259	0,978129
8	0,749038	0,977268	0,695434	0,756195	0,695434	0,977268
9	0,747735	0,97658	0,693847	0,755345	0,693847	0,97658
10	0,747003	0,980739	0,701405	0,761216	0,701405	0,980739
11	0,745344	0,979603	0,692553	0,760363	0,692553	0,979603
12	0,746462	0,981048	0,69473	0,761355	0,69473	0,981048
13	0,746792	0,982148	0,694173	0,761783	0,694173	0,982148
14	0,748624	0,976946	0,695297	0,756818	0,695297	0,976946
15	0,74542	0,976022	0,699518	0,758724	0,699518	0,976022
16	0,749471	0,980032	0,694889	0,760208	0,694889	0,980032
17	0,747897	0,975945	0,688447	0,755763	0,688447	0,975945
18	0,748589	0,984389	0,696926	0,76415	0,696926	0,984389
19	0,748234	0,979344	0,694856	0,761369	0,694856	0,979344
20	0,751002	0,980422	0,696201	0,759554	0,696201	0,980422

	10 *latent features*			**15 *latent features***		
Iteração	**MAE (teste)**	**RMSE (teste)**	**RMSE (treino)**	**MAE (teste)**	**RMSE (teste)**	**RMSE (treino)**
1	0,768285	0,993332	0,637305	0,774883	1,007299	0,58207
2	0,765672	0,994742	0,631058	0,773233	1,005963	0,580093
3	0,765506	0,994583	0,629607	0,766795	0,999519	0,580145
4	0,764452	0,991155	0,632016	0,765689	0,999886	0,585326
5	0,760206	0,985825	0,631624	0,76821	1,000685	0,582404
6	0,762496	0,994445	0,631384	0,768964	1,001158	0,580334
7	0,757605	0,9854	0,630501	0,766298	1,000308	0,587289
8	0,768512	0,99691	0,630216	0,76744	1,002727	0,579578
9	0,768105	0,998166	0,633496	0,76821	0,999892	0,590149
10	0,764364	0,991753	0,631046	0,768741	1,002309	0,583478
11	0,763924	0,990019	0,630454	0,771296	1,004606	0,57848

12	0,763103	0,989694	0,629435	0,766752	1,001673	0,587245
13	0,768598	0,994124	0,630378	0,768328	1,003727	0,582545
14	0,762803	0,992151	0,631273	0,769611	1,002958	0,584138
15	0,758983	0,988772	0,629416	0,770415	1,003628	0,579681
16	0,765277	0,992946	0,630691	0,766085	0,996414	0,580494
17	0,760337	0,989145	0,634106	0,766667	1,000187	0,583683
18	0,761114	0,98959	0,629584	0,761795	0,991062	0,584068
19	0,761894	0,98607	0,632845	0,773115	1,001166	0,579495
20	0,76578	0,995626	0,63124	0,773318	1,002968	0,583502

	20 *latent features*			**30 *latent features***		
Iteração	**MAE (teste)**	**RMSE (teste)**	**RMSE (treino)**	**MAE (teste)**	**RMSE (teste)**	**RMSE (treino)**
1	0,773186	1,009322	0,542283	0,779641	1,024967	0,479504
2	0,77976	1,019631	0,539893	0,783938	1,028694	0,476513
3	0,768136	1,007132	0,543612	0,782461	1,021809	0,480237
4	0,773256	1,009535	0,540888	0,782068	1,02595	0,478548
5	0,777131	1,01424	0,541284	0,785714	1,025679	0,479072
6	0,775926	1,012374	0,542833	0,786488	1,032419	0,479085
7	0,773792	1,008159	0,542977	0,786198	1,028774	0,479701
8	0,772637	1,010517	0,541531	0,782111	1,026802	0,480738
9	0,774803	1,011176	0,54318	0,783676	1,027707	0,480907
10	0,772957	1,010097	0,54257	0,784809	1,024515	0,480391
11	0,775719	1,014204	0,543486	0,782041	1,027213	0,479686
12	0,766959	1,006986	0,543691	0,788356	1,03011	0,479021
13	0,771727	1,008237	0,544043	0,785927	1,029198	0,480065
14	0,777807	1,017974	0,538045	0,78405	1,028321	0,478385
15	0,777692	1,014299	0,54458	0,787317	1,0284	0,476733
16	0,7695	1,006432	0,54367	0,792986	1,035104	0,480253
17	0,77725	1,016695	0,539979	0,7849	1,027106	0,480578
18	0,773338	1,010057	0,546629	0,790202	1,032637	0,477237
19	0,771416	1,006319	0,540555	0,780893	1,022016	0,478854
20	0,772455	1,006674	0,542092	0,78582	1,029956	0,480872

	40 *latent features*		
Iteração	**MAE (teste)**	**RMSE (teste)**	**RMSE (treino)**
1	0,801925	1,048007	0,433387
2	0,797409	1,044467	0,43493
3	0,800989	1,047077	0,432283
4	0,796813	1,047168	0,43193
5	0,800235	1,049477	0,432659
6	0,794691	1,041953	0,434886
7	0,801822	1,051323	0,431908
8	0,796764	1,043416	0,432247
9	0,792445	1,041078	0,434021
10	0,797181	1,050698	0,429494
11	0,795311	1,042357	0,431482
12	0,807389	1,056243	0,431199
13	0,801676	1,052606	0,432347
14	0,80114	1,049831	0,431221
15	0,796123	1,042331	0,432633
16	0,798955	1,047792	0,430993
17	0,798166	1,043114	0,433949
18	0,801044	1,050208	0,429781
19	0,792775	1,04165	0,431544
20	0,797007	1,044078	0,429671

Anexo 8

Resultados do erro de treino usando a métrica RMSE para a técnica *Matrix Factorization* variando o número de iterações

	100 Iterações	200 Iterações	500 Iterações
Iteração	**RMSE (treino)**	**RMSE (treino)**	**RMSE (treino)**
1	1,079046718	1,07729285	1,086042326
2	0,936851393	0,936470002	0,93899721
3	0,892226808	0,891877228	0,89294286
4	0,866809545	0,866436369	0,867278025
5	0,848940735	0,848637324	0,849487461
6	0,83517056	0,834939291	0,835851189
7	0,823933679	0,8237516	0,824751972
8	0,814384839	0,814226322	0,815336417
9	0,806016126	0,805858093	0,807095066
10	0,798500693	0,798323029	0,799700409
11	0,791618098	0,791403189	0,792931768
12	0,7852143	0,784946666	0,786635276
13	0,779178582	0,778844703	0,780700603
14	0,773429568	0,773017831	0,77504676
15	0,767906471	0,767407136	0,76961319
16	0,762563422	0,761968594	0,764354071
17	0,757365686	0,756669253	0,75923456
18	0,752287038	0,751484582	0,754228242
19	0,747307868	0,746396556	0,749315309
20	0,742413772	0,741392223	0,744481212
21	0,737594458	0,736462589	0,739715609
22	0,732842883	0,731601747	0,735011527
23	0,728154555	0,726806161	0,730364669
24	0,723526961	0,722074089	0,725772836
25	0,718959093	0,717405111	0,721235434
26	0,714451057	0,712799736	0,716753062
27	0,71000375	0,708259098	0,712327158
28	0,705618595	0,703784699	0,707959704
29	0,701297323	0,699378221	0,703652989
30	0,697041802	0,695041372	0,699409414
31	0,692853907	0,690775786	0,695231338
32	0,688735413	0,68658294	0,691120973
33	0,684687928	0,682464108	0,687080296
34	0,680712844	0,67842033	0,683111005
35	0,676811301	0,6744524	0,679214485
36	0,672984177	0,670560863	0,675391801
37	0,669232087	0,666746023	0,671643697
38	0,665555382	0,663007954	0,667970609
39	0,661954167	0,659346518	0,664372691

40	0,658428317	0,65576138	0,660849831
41	0,654977493	0,652252033	0,65740168
42	0,651601168	0,648817812	0,654027684
43	0,648298643	0,645457914	0,650727103
44	0,645069073	0,642171419	0,647499043
45	0,641911482	0,638957302	0,644342476
46	0,638824787	0,635814454	0,641256265
47	0,635807812	0,63274169	0,638239181
48	0,632859305	0,629737768	0,635289927
49	0,629977956	0,626801396	0,632407146
50	0,627162404	0,623931245	0,62958944
51	0,624411255	0,621125955	0,626835383
52	0,621723088	0,618384148	0,624143529
53	0,619096466	0,615704432	0,621512421
54	0,616529946	0,613085408	0,618940601
55	0,61402208	0,610525676	0,616426615
56	0,611571428	0,608023837	0,613969019
57	0,609176559	0,605578504	0,611566385
58	0,606836056	0,603188297	0,609217301
59	0,604548519	0,600851853	0,60692038
60	0,602312571	0,598567826	0,604674256
61	0,600126856	0,596334889	0,602477591
62	0,597990044	0,594151737	0,600329075
63	0,595900834	0,592017086	0,598227425
64	0,59385795	0,589929678	0,596171392
65	0,591860148	0,587888281	0,594159753
66	0,589906212	0,585891689	0,59219132
67	0,587994957	0,583938721	0,590264932
68	0,586125228	0,582028225	0,588379462
69	0,584295901	0,580159076	0,586533814
70	0,582505881	0,578330177	0,584726921
71	0,580754107	0,57654046	0,582957747
72	0,579039542	0,574788883	0,581225286
73	0,577361185	0,573074432	0,579528562
74	0,575718059	0,571396122	0,577866627
75	0,574109218	0,569752992	0,57623856
76	0,572533743	0,56814411	0,574643471
77	0,570990743	0,566568571	0,573080493
78	0,569479354	0,565025493	0,571548787
79	0,567998737	0,563514023	0,570047541
80	0,566548078	0,562033329	0,568575965
81	0,565126591	0,560582607	0,567133295
82	0,56373351	0,559161073	0,565718789
83	0,562368095	0,557767969	0,564331729
84	0,561029628	0,556402559	0,562971419
85	0,559717412	0,555064128	0,561637183
86	0,558430773	0,553751984	0,560328367

87	0,557169056	0,552465453	0,559044337
88	0,555931629	0,551203884	0,557784477
89	0,554717875	0,549966644	0,556548192
90	0,553527201	0,548753121	0,555334902
91	0,552359027	0,547562718	0,554144048
92	0,551212795	0,54639486	0,552975085
93	0,550087961	0,545248987	0,551827486
94	0,548983999	0,544124555	0,55070074
95	0,547900399	0,543021038	0,54959435
96	0,546836666	0,541937925	0,548507835
97	0,54579232	0,540874722	0,547440728
98	0,544766895	0,539830946	0,546392576
99	0,543759939	0,538806132	0,545362937
100	0,542771015	0,537799827	0,544351386
101		0,536811592	0,543357507
102		0,535841001	0,542380898
103		0,534887639	0,541421168
104		0,533951106	0,540477936
105		0,53303101	0,539550834
106		0,532126974	0,538639503
107		0,53123863	0,537743594
108		0,530365621	0,536862769
109		0,529507599	0,535996699
110		0,528664229	0,535145064
111		0,527835182	0,534307552
112		0,52702014	0,53348386
113		0,526218794	0,532673695
114		0,525430844	0,531876769
115		0,524655998	0,531092804
116		0,523893971	0,530321529
117		0,523144487	0,52956268
118		0,522407278	0,528815999
119		0,521682082	0,528081237
120		0,520968645	0,527358149
121		0,520266719	0,526646498
122		0,519576065	0,525946054
123		0,518896446	0,52525659
124		0,518227637	0,524577888
125		0,517569413	0,523909732
126		0,51692156	0,523251916
127		0,516283866	0,522604235
128		0,515656127	0,521966491
129		0,515038142	0,521338491
130		0,514429718	0,520720047
131		0,513830663	0,520110974
132		0,513240793	0,519511093
133		0,512659929	0,518920229
134		0,512087894	0,518338211

135		0,511524516	0,517764873
136		0,510969629	0,517200051
137		0,51042307	0,516643586
138		0,50988468	0,516095324
139		0,509354303	0,515555113
140		0,508831789	0,515022805
141		0,508316989	0,514498254
142		0,507809759	0,513981321
143		0,507309959	0,513471866
144		0,506817451	0,512969755
145		0,506332101	0,512474856
146		0,505853778	0,511987041
147		0,505382354	0,511506182
148		0,504917703	0,511032157
149		0,504459705	0,510564846
150		0,504008238	0,510104129
151		0,503563187	0,509649894
152		0,503124437	0,509202025
153		0,502691878	0,508760414
154		0,502265399	0,508324952
155		0,501844895	0,507895534
156		0,501430261	0,507472056
157		0,501021396	0,507054417
158		0,5006182	0,506642518
159		0,500220574	0,506236262
160		0,499828424	0,505835554
161		0,499441657	0,505440301
162		0,499060181	0,505050413
163		0,498683906	0,504665799
164		0,498312746	0,504286372
165		0,497946613	0,503912048
166		0,497585426	0,503542742
167		0,4972291	0,503178371
168		0,496877557	0,502818856
169		0,496530716	0,502464118
170		0,496188502	0,502114078
171		0,495850838	0,501768662
172		0,49551765	0,501427796
173		0,495188866	0,501091405
174		0,494864415	0,500759419
175		0,494544226	0,500431769
176		0,494228232	0,500108384
177		0,493916366	0,499789198
178		0,493608562	0,499474145
179		0,493304755	0,49916316
180		0,493004884	0,498856179
181		0,492708885	0,49855314

182		0,492416698	0,498253982
183		0,492128264	0,497958645
184		0,491843524	0,49766707
185		0,491562422	0,497379199
186		0,4912849	0,497094976
187		0,491010904	0,496814344
188		0,490740381	0,49653725
189		0,490473276	0,496263639
190		0,490209538	0,495993458
191		0,489949116	0,495726657
192		0,489691959	0,495463185
193		0,489438019	0,495202991
194		0,489187247	0,494946026
195		0,488939597	0,494692244
196		0,48869502	0,494441595
197		0,488453472	0,494194035
198		0,488214908	0,493949518
199		0,487979284	0,493707998
200		0,487746556	0,493469433
201			0,493233778
202			0,493000992
203			0,492771033
204			0,492543859
205			0,492319432
206			0,492097711
207			0,491878658
208			0,491662234
209			0,491448402
210			0,491237126
211			0,491028369
212			0,490822095
213			0,49061827
214			0,490416859
215			0,490217828
216			0,490021145
217			0,489826777
218			0,489634691
219			0,489444856
220			0,489257241
221			0,489071816
222			0,48888855
223			0,488707415
224			0,48852838
225			0,488351419
226			0,488176502
227			0,488003602
228			0,487832691
229			0,487663744

230			0,487496734
231			0,487331636
232			0,487168423
233			0,487007071
234			0,486847555
235			0,486689851
236			0,486533935
237			0,486379785
238			0,486227376
239			0,486076686
240			0,485927694
241			0,485780376
242			0,485634712
243			0,485490679
244			0,485348258
245			0,485207427
246			0,485068167
247			0,484930457
248			0,484794278
249			0,484659609
250			0,484526434
251			0,484394731
252			0,484264484
253			0,484135674
254			0,484008282
255			0,483882292
256			0,483757686
257			0,483634447
258			0,483512558
259			0,483392002
260			0,483272763
261			0,483154826
262			0,483038173
263			0,48292279
264			0,48280866
265			0,48269577
266			0,482584104
267			0,482473647
268			0,482364384
269			0,482256302
270			0,482149386
271			0,482043623
272			0,481938998
273			0,481835499
274			0,481733111
275			0,481631822
276			0,481531619

277			0,481432489
278			0,481334419
279			0,481237397
280			0,481141411
281			0,481046449
282			0,480952499
283			0,480859549
284			0,480767588
285			0,480676603
286			0,480586585
287			0,480497521
288			0,480409402
289			0,480322215
290			0,480235951
291			0,480150598
292			0,480066147
293			0,479982588
294			0,479899909
295			0,479818102
296			0,479737156
297			0,479657062
298			0,479577809
299			0,47949939
300			0,479421793
301			0,479345011
302			0,479269034
303			0,479193852
304			0,479119458
305			0,479045842
306			0,478972996
307			0,478900911
308			0,478829579
309			0,478758991
310			0,47868914
311			0,478620016
312			0,478551613
313			0,478483922
314			0,478416936
315			0,478350646
316			0,478285045
317			0,478220126
318			0,478155881
319			0,478092302
320			0,478029383
321			0,477967117
322			0,477905496
323			0,477844513
324			0,477784162

325			0,477724435
326			0,477665326
327			0,477606828
328			0,477548935
329			0,477491639
330			0,477434936
331			0,477378818
332			0,477323278
333			0,477268312
334			0,477213912
335			0,477160073
336			0,477106788
337			0,477054052
338			0,477001859
339			0,476950204
340			0,476899079
341			0,47684848
342			0,476798402
343			0,476748837
344			0,476699783
345			0,476651231
346			0,476603179
347			0,476555619
348			0,476508547
349			0,476461959
350			0,476415847
351			0,476370209
352			0,476325038
353			0,47628033
354			0,47623608
355			0,476192283
356			0,476148934
357			0,476106028
358			0,476063562
359			0,47602153
360			0,475979928
361			0,475938751
362			0,475897995
363			0,475857655
364			0,475817728
365			0,475778208
366			0,475739091
367			0,475700374
368			0,475662052
369			0,475624121
370			0,475586577
371			0,475549416

372			0,475512634
373			0,475476227
374			0,475440191
375			0,475404522
376			0,475369216
377			0,47533427
378			0,47529968
379			0,475265442
380			0,475231553
381			0,475198009
382			0,475164806
383			0,47513194
384			0,475099409
385			0,475067209
386			0,475035337
387			0,475003788
388			0,47497256
389			0,47494165
390			0,474911054
391			0,474880768
392			0,47485079
393			0,474821117
394			0,474791745
395			0,474762671
396			0,474733892
397			0,474705406
398			0,474677208
399			0,474649297
400			0,474621669
401			0,474594321
402			0,474567251
403			0,474540455
404			0,474513931
405			0,474487675
406			0,474461686
407			0,47443596
408			0,474410495
409			0,474385288
410			0,474360337
411			0,474335638
412			0,474311189
413			0,474286988
414			0,474263032
415			0,474239319
416			0,474215845
417			0,47419261
418			0,474169609
419			0,474146841

420			0,474124304
421			0,474101995
422			0,474079911
423			0,474058051
424			0,474036411
425			0,474014991
426			0,473993787
427			0,473972798
428			0,473952021
429			0,473931454
430			0,473911094
431			0,473890941
432			0,473870991
433			0,473851243
434			0,473831694
435			0,473812343
436			0,473793187
437			0,473774225
438			0,473755454
439			0,473736873
440			0,47371848
441			0,473700272
442			0,473682248
443			0,473664406
444			0,473646744
445			0,47362926
446			0,473611953
447			0,47359482
448			0,47357786
449			0,473561072
450			0,473544452
451			0,473528001
452			0,473511715
453			0,473495594
454			0,473479635
455			0,473463837
456			0,473448198
457			0,473432717
458			0,473417392
459			0,473402222
460			0,473387205
461			0,473372339
462			0,473357622
463			0,473343054
464			0,473328633
465			0,473314357
466			0,473300225

467			0,473286236
468			0,473272387
469			0,473258678
470			0,473245107
471			0,473231673
472			0,473218373
473			0,473205208
474			0,473192176
475			0,473179274
476			0,473166503
477			0,47315386
478			0,473141344
479			0,473128955
480			0,47311669
481			0,473104548
482			0,473092529
483			0,473080631
484			0,473068852
485			0,473057192
486			0,47304565
487			0,473034223
488			0,473022911
489			0,473011714
490			0,473000629
491			0,472989655
492			0,472978792
493			0,472968038
494			0,472957392
495			0,472946853
496			0,47293642
497			0,472926092
498			0,472915868
499			0,472905747
500			0,472895728

Anexo 9

De maneira a organizar todos os testes unitários realizados, foram feitas tabelas para cada método mostrando o nome deste, a descrição do teste, os cenários de teste e o resultado obtido.

FitnessAPI

Função:	***AreSupersetsEqual(exercises1, exercises2)***
Descrição:	Verificar se as listas de supersets de duas listas de exercícios são iguais
Testes:	• Testar com *supersets* iguais • Testar com *supersets* diferentes
Resultado:	PASSOU

Função:	***GetExercisesSupersets(exercises)***
Descrição:	Testar o retorno de uma lista de *supersets*
Testes:	• Testar com zero *supersets* • Testar com 1 ou mais *supersets*
Resultado:	PASSOU

Função:	***SetSupersetsInExercises(workout, supersets, createdSupersets)***
Descrição:	Testar se os novos *supersets* ficaram atribuídos aos exercícios
Testes:	• Testar se os exercícios ficaram com os novos *supersets* • Testar se os exercícios não se alteraram
Resultado:	PASSOU

Função:	***UpdateExerciseWithSetsAndLogs(model, exercise, index, logInstance, duration, volume)***
Descrição:	Testar a atualização de *sets* e *logs* num exercício
Testes:	• Testar com novos *sets* e verificar se o exercício alterou • Testar com novos *logs* e verificar se o exercício alterou
Resultado:	PASSOU

FtinessRecommenderSystem

Função:	***CompareVectors(userFeaturesOne, userFeaturesTwo)*** – *CoRatedCosineUserComparer, CorrelationUserComparer, CosineUserComparer, RootMeanSquareUserComparer*
Descrição:	Testar o grau de semelhança
Testes:	• Testar com grau de semelhança alta • Testar com grau de semelhança baixa
Resultado:	PASSOU

Função:	***GetDotProduct (matrixOne, matrixTwo)***
Descrição:	Testar o produto de duas matrizes
Testes:	• Testar com duas matrizes
Resultado:	PASSOU

Função:	***FactorizeMatrix(ratings)***
Descrição:	Testar o cálculo da técnica SVD para uma matriz
Testes:	• Testar com uma matriz válida • Testar com uma matriz inválida
Resultado:	PASSOU

Função:	**Train(DatasetModel db) – para todas as técnicas**
Descrição:	Testar a criação do modelo
Testes:	• Testar se o modelo é valido para um certo *dataset*
Resultado:	PASSOU

Função:	***GetRating(userId, itemId) – para todas as técnicas***
Descrição:	Testar a sugestão de item para um user
Testes:	• Testar com user e item válidos • Testar com user e item inválidos
Resultado:	PASSOU

Função:	***GetSuggestions(userId, numSuggestions)***
Descrição:	Testar o retorno de sugestões
Testes:	• Testar com um user válido • Testar o número de sugestões retornada
Resultado:	PASSOU

Função:	***GetNearestNeighbors(articleId, numArticles)***
Descrição:	Testar
Testes:	• Testar o número de vizinhos retornados • Testar se o os vizinhos apresentam grau de semelhança alto
Resultado:	PASSOU

Função:	**GetUserUnusedData(userHistory, userData)**
Descrição:	Testar o retorno de dados não usados pelo utilizador
Testes:	• Testar com uma lista userData vazia • Testar o retorno de dados não usados para uma lista userData não vazia
Resultado:	PASSOU

Anexo 10

Os testes de integração foram realizados em função dos requisitos funcionais e não funcionais. As tabelas de testes apresentam os componentes envolvidos, as funções testadas, a descrição do que se pretende do teste, o resultado esperado e o resultado obtido no teste.

Componentes:	**FitnessMobileAPP, FitnessAPI e Database**
Função:	*Get()*
Descrição:	A API deve obter todos os dados (exercícios, treinos, *logs* e *setups*) da base de dados e retornar à aplicação móvel
Resultado esperado:	Os dados pretendidos são obtidos da base de dados e retornados
Resultado obtido:	PASSOU

Componentes:	**FitnessMobileAPP, FitnessAPI e Database**
Função:	*Post()*
Descrição:	A API deve registar os novos dados (exercícios, treinos, *logs* e *setups*) na base de dados e comunicar o sucesso à aplicação móvel
Resultado esperado:	Os dados são criados na base dados e é retornado sucesso
Resultado obtido:	PASSOU

Componentes:	**FitnessMobileAPP, FitnessAPI e Database**
Função:	*Put()*
Descrição:	A API deve atualizar as entidades (exercícios, treinos, *logs* e *setups*) da base de dados e comunicar o sucesso à aplicação móvel
Resultado esperado:	Os dados são criados na base dados e é retornado sucesso
Resultado obtido:	PASSOU

Componentes:	**ChatBotAPI e LUIS**
Função:	*GetIntentScore()*
Descrição:	A ChatBotAPI comunica com o serviço LUIS para obter a intenção que obteve melhor classificação
Resultado esperado:	A interpretação com melhor classificação é retornada
Resultado obtido:	PASSOU

Componentes:	**ChatBotAPI, FitnessAPI e Database**
Função:	*Get()*
Descrição:	A FitnessAPI deve obter informações dos exercícios da base de dados e retornar à ChatBotAPI
Resultado esperado:	Os dados pretendidos são obtidos da base de dados e retornados
Resultado obtido:	PASSOU

Componentes:	**FitnessMobileApp, ChatBotAPI, LUIS, FitnessAPI e Database**
Função:	*GetResponse()*
Descrição:	O utilizador envia uma mensagem na FitnessMobileApp, esta comunica com a ChatBotAPI para a interpretação da intenção (LUIS), que depois requisita os dados necessários à FitnessAPI.
Resultado esperado:	Uma resposta é recebida
Resultado obtido:	PASSOU

Anexo 11

Os testes de sistema foram realizados às funcionalidades da aplicação. As tabelas de teste têm a descrição da funcionalidade, o resultado esperado e o resultado obtido.

Descrição:	Realizar o processo de registo, login e, de seguida, o de *setup*
Resultado obtido:	PASSOU

Descrição:	Realizar o processo de registo, login e, de seguida, o *logout*
Resultado obtido:	PASSOU

Descrição:	Realizar o processo de *login* e, de seguida, o de *logout*
Resultado obtido:	PASSOU

Descrição:	Realizar o processo de *login* e, de seguida, o de setup
Resultado obtido:	PASSOU

Descrição:	Realizar o processo de esquecimento de *password* e, de seguida, o de *login*
Resultado obtido:	PASSOU

Descrição:	Filtrar e visualizar os exercícios pretendidos
Resultado obtido:	PASSOU

Descrição:	Criar um treino e fazer o seu registo de forma manual
Resultado obtido:	PASSOU

Descrição:	Criar um treino e fazer o seu registo em tempo real
Resultado obtido:	PASSOU

Anexo 12

Os testes de aceitação foram realizados em função dos requisitos funcionais e agrupados pelos casos de uso. As tabelas de teste têm a identificação do UC, a descrição do módulo, o resultado esperado e o resultado obtido.

Caso de Uso	UA01: Registar Utilizador
Descrição:	Registar um utilizador usando a combinação email, password e confirme *password*
Resultado obtido:	PASSOU

Caso de Uso	UA01: Registar Utilizador
Descrição:	Registar um utilizador usando o *Facebook*
Resultado obtido:	PASSOU

Caso de Uso	UA02: Login
Descrição:	Login de um utilizador usando a combinação email e *password*
Resultado obtido:	PASSOU

Caso de Uso	UA02: Login
Descrição:	Login de um utilizador utilizando a conta de *Facebook* usada no processo de registo
Resultado obtido:	PASSOU

Caso de Uso	UA03: Recuperar password
Descrição:	Introduzir email e obter uma *password* nova
Resultado obtido:	PASSOU

Caso de Uso	US01: Realizar setup
Descrição:	Introduzir os dados requisitados pela aplicação
Resultado obtido:	PASSOU

Caso de Uso	US03: Gerir exercícios
Descrição:	Visualizar lista de exercícios existentes
Resultado obtido:	PASSOU

Caso de Uso	US04: Filtrar exercícios
Descrição:	Utilizar diferentes tipos de filtros (p.e. músculos, nível, equipamentos, tipo de mecanismo, entre outros)
Resultado obtido:	PASSOU

Caso de Uso	US08: Visualizar treinos
Descrição:	Visualizar lista de treinos existentes
Resultado obtido:	PASSOU

Caso de Uso	US09: Criar treino
Descrição:	Criar um treino com todos os atributos existentes (p.e. exercícios, *supersets* e *sets*)
Resultado obtido:	PASSOU

Caso de Uso	US10: Registar dados do treino efetuado e US11: Realizar treino em tempo real
Descrição:	Registar dados do treino efetuado, como os sets, número de repetições e peso
Resultado obtido:	PASSOU